Cost-effective Design of Real-time Home Healthcare Telemonitoring based on Mobile Cloud Computing

by

Po-Chou Liang

A thesis submitted in the fulfilment of the requirements for the degree of Doctor of Philosophy

Department of Computer Science
Faculty of Engineering and Physical Sciences
University of Surrey
Guildford, Surrey, GU2 7XH
January 2016
Abstract

The rise in both ageing and chronic disease populations has highlighted a pressing demand for better access to quality healthcare at home. Meanwhile, studies have shown that home-based treatments for older patients as a substitute for hospital care can produce better clinical outcomes and reduce healthcare expenditure. However, there remains a considerable question relating to the low adoption rate of home telehealthcare technologies due to a lack of robust evidence for their cost-effectiveness. In light of both the epoch-making advancements in smartphone-centric technologies and the pervasive uptake of smartphones, we set up as our core objective the cost-effective design of a real-time home healthcare telemonitoring system based on mobile cloud computing. Our second objective was to develop a simulation environment to produce robust evidence for cost-effectiveness of a telemonitoring system so as to explore technology choices in different settings prior to moving to full-scale trials on a more scientific basis.

A proof-of-concept system consisting of three main monitoring functions, namely vital sign, safety (for fall detection) and movement pattern monitoring (for real-time indoor location tracking), was developed based on a smartphone. With the exception of vital sign monitoring design which was not regarded as a search problem, the results of the other two were promising with sensitivity and specificity for successfully detected falls and recognised non-fall activities being both 95.5% and an average estimation error of 0.47 metres for real-time indoor location tracking. A large number of patients and their activities of daily living, as well as real-world like telehealthcare scenarios involving a number of different stakeholders and telemonitoring interventions, were modelled and created through simulations. The cloud-based components of our proposed telemonitoring system were also modelled and simulated together with our proposed forward-looking unused capacity-based auto scaling (FLUCAS) algorithm to enhance system performance and scalability and reduce the costs. Economic evaluations of our proposed system were conducted based on a comparative cost-effectiveness analysis approach and the results of our simulation experiments. Although exploratory, this study not only offers some insight into the great potential of smartphone-centric technologies in support of a cost-effective design of real-time home healthcare telemonitoring, but also provides justifiable evidence for cost-effectiveness of telemonitoring.

Keywords: telehealthcare, telemonitoring, vital sign, fall detection, step detection, indoor location tracking, cost-effectiveness analysis, mobile cloud computing, auto scaling, simulation
Acknowledgements

I would like to thank all the individuals and organisations who have helped me throughout my PhD study. In particular, I would like to express my sincere gratitude to my supervisor Professor Paul Krause for his invaluable help, informed guidance and continuous encouragement throughout this research.

I would also like to thank my co-supervisor Dr. Lee Gillam, Head of Department Dr. Helen Treharne, my English writing instructor Dr Alison Yeung, and Professor Anthony T.S. Ho, and the entire administrative staff of the Department of Computer Science, University of Surrey for their constant help throughout my study here. I would like to thank two PhD students, Aaron Mason and Georgios Michalakidis, in our research group for helping me with continuous access to Apple Developer Program.

In addition, I would like to thank my Chairperson Dr. Howard S.H Shyr, Vice Chairperson Dr. Hsiao-Cheng Yu, ex-Chairperson Dr. Herng Su, ex-Chairperson Dr. Bonnie Peng, and several other colleagues at the National Communications Commission, Taiwan, for giving me the opportunity to conduct this study and rendering their continuous support.

Many thanks to my parents and mother-in-law for their love and support. Last but not least, special thanks and love to my wife, Hsin-Yu, who gave up her career in Taiwan, for providing me with every support and taking excellent care of our kids, Claire and George, throughout our stay in the UK.

This work is supported in part by the Taiwan Government under a professional cultivation programme.
Publications


Glossary

7zip an open-source file archiver
ADL Activities of Daily Living
AES Advanced Encryption Standard
A&E Accident and Emergency department
Apache an open-source software foundation
Apache 2.4.7 Apache HTTP Server version 2.4.7
API Application Programming Interface
App Mobile Application
App&DB the web application and database server module, a term used in this thesis
AMI Amazon Machine Image
AOA Angle of Arrival
Arduino an open-source microcontroller platform for physical computing
Arduino UNO R3 a version of an Arduino-based microcontroller board
appServer a web application server in a CloudSim-based simulation
asCloudlet a workload on the web application server in a CloudSim-based simulation
AWS Amazon Web Services
BLE Bluetooth Low Energy, a wireless standard
Bluetooth Smart a wireless standard, also called Bluetooth Low Energy
BSN Body Sensor Network
CPASIL Common Awareness and Knowledge Platform for Studying and Enabling Independent Living, funded by the 7th European Framework Programme
CER Comparative Effectiveness Research
CER Cost-effectiveness Ratio
CERP-IoT The Cluster of European Research Projects on the Internet of Things
CHD Coronary Heart Disease
CloudAnalyst a CloudSim-based simulation toolkit with graphical user interface
Cloudlet the representation of a workload in the CloudSim simulation toolkit
CloudSim a Java-based cloud simulation toolkit
CloudSimEx an extension of the CloudSim simulation toolkit
COPD Chronic Obstructive Pulmonary Disease
CPU Central Processing Unit
DALLAS Delivery of Assisted Living Lifestyles at Scale Programme
DB a database server, an abbreviated term used in this thesis
DCSim a Java-based cloud simulation toolkit
dbCloudlet a workload on the web database server in a CloudSimEx-based simulation
dbServer  a web database server in a CloudSimEx-based simulation
DES     Discrete-event Simulation
DLNA    Digital Living Network Alliance
DNS     Domain Name System
DSSS    Direct Sequence Spread Spectrum
EBM     Evidence-based Medicine
EBS     Elastic Block Store, a persistent block level storage volume in Amazon EC2
EC2     Amazon Elastic Compute Cloud
ECG     Electrocardiogram, a heart test to check its rhythm and electrical activity
Eclipse Java EE IDE  a cross-platform integrated development environment released by the Eclipse Foundation for developing Java applications
ED      Emergency Department
EQ-5D   a standardised instrument for use as a measure of health outcome
EU      European Union
FAST    Facial drooping, Arm weakness, Speech difficulties and Time; a term used to help detect and increase responsiveness to acute stroke patients
FCFS    First Come First Served
FHSS    Frequency Hopping Spread Spectrum
FIFO    First In First Out
fio     an I/O performance test software tool
FLUCAS  Forward-looking Unused Capacity-based Auto Scaling algorithm, developed by this study
FMDB    an open-source software framework for SQLite database operations
GAE     Google App Engine
GP      General Practitioner
GPS     Global Positioning System
GSM     Global System for Mobile communications
HddVm   a virtual machine with a hard disk in a CloudSimEx-based simulation
HealthVault  a cloud-based platform launched by Microsoft for storing and sharing health information
HTA     Health Technology Assessment
HealthInf The International Conference on Health Informatics
HomeKit  a home automation framework developed by Apple
HTTP    The Hypertext Transfer Protocol
HVM     Hardware Virtual Machine
IaaS    Infrastructure as a Service, one of the cloud service models
iBeacon a Bluetooth Smart enabled proximity sensor/device developed by Apple
IBM Blade Cluster HS21  A computer model developed by IBM
ICER    Incremental Cost-effectiveness Ratio
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICT</td>
<td>Information and Communications Technology</td>
</tr>
<tr>
<td>ICT PSP</td>
<td>European Commission ICT Policy Support Programme</td>
</tr>
<tr>
<td>ICP</td>
<td>Integrated Care Platform for the COPD patient care pathway in NW Sussex, England</td>
</tr>
<tr>
<td>ID</td>
<td>identification number, an abbreviation used in this thesis</td>
</tr>
<tr>
<td>IDC</td>
<td>The Interactive Data Corporation</td>
</tr>
<tr>
<td>IDE</td>
<td>Integrated Development Environment</td>
</tr>
<tr>
<td>IEEE</td>
<td>The Institute of Electrical and Electronics Engineers</td>
</tr>
<tr>
<td>i-Focus</td>
<td>an initiative of the DALLAS programme</td>
</tr>
<tr>
<td>iperf</td>
<td>a network bandwidth test software tool</td>
</tr>
<tr>
<td>I/O</td>
<td>Input/Output</td>
</tr>
<tr>
<td>IOPS</td>
<td>Input/Output Operations Per Second</td>
</tr>
<tr>
<td>iOS</td>
<td>a mobile operating system developed by Apple</td>
</tr>
<tr>
<td>iostat</td>
<td>an input/output device monitoring tool (command) for Unix/Linux-based systems</td>
</tr>
<tr>
<td>IoT</td>
<td>Internet of Thing</td>
</tr>
<tr>
<td>ITU</td>
<td>The International Telecommunication Union</td>
</tr>
<tr>
<td>Java EE</td>
<td>Java Platform, Enterprise Edition</td>
</tr>
<tr>
<td>JDK</td>
<td>Java Standard Edition Development Kit</td>
</tr>
<tr>
<td>JRE</td>
<td>Java Runtime Environment</td>
</tr>
<tr>
<td>Kalman filter</td>
<td>a computational algorithm to produce estimates of unknown variables</td>
</tr>
<tr>
<td>KNX</td>
<td>an open standard introduced by the Konnex Association for home and building control</td>
</tr>
<tr>
<td>KXTJ9</td>
<td>a tri-axis accelerometer developed by Kionix</td>
</tr>
<tr>
<td>Linux</td>
<td>an open-source Unix-like operating system</td>
</tr>
<tr>
<td>Living it Up</td>
<td>an initiative of the DALLAS programme</td>
</tr>
<tr>
<td>LTS</td>
<td>Long Term Support, used to specify that a version of operating system has a long term support on Ubuntu</td>
</tr>
<tr>
<td>LYG</td>
<td>Life-years Gained</td>
</tr>
<tr>
<td>MaxWeight scheduling</td>
<td>an algorithm for throughput-optimal in wireless networks</td>
</tr>
<tr>
<td>MIPS</td>
<td>Million Instructions Per Second</td>
</tr>
<tr>
<td>MIOPS</td>
<td>Million Input/Output Operations Per Second</td>
</tr>
<tr>
<td>MLE</td>
<td>Maximum-Likelihood Estimation</td>
</tr>
<tr>
<td>MobileMed</td>
<td>International Conference on Mobile and Information Technologies in Medicine and Health</td>
</tr>
<tr>
<td>MoCAsh</td>
<td>a project on mobile cloud for assistive healthcare</td>
</tr>
<tr>
<td>MongoDB</td>
<td>a document-oriented (or NoSQL) database</td>
</tr>
<tr>
<td>Monte Carlo method</td>
<td>a broad class of computational algorithms</td>
</tr>
<tr>
<td>More Independent</td>
<td>an initiative of the DALLAS programme</td>
</tr>
<tr>
<td>Term</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MS Windows</td>
<td>graphical operating systems developed by Microsoft</td>
</tr>
<tr>
<td>MySQL</td>
<td>an open-source relational database management system</td>
</tr>
<tr>
<td>NHS</td>
<td>The National Health Service</td>
</tr>
<tr>
<td>NHS 111</td>
<td>the NHS non-emergency number</td>
</tr>
<tr>
<td>NICE</td>
<td>The National Institute for Health and Clinical Excellence</td>
</tr>
<tr>
<td>NIST</td>
<td>The National Institute of Standards and Technology</td>
</tr>
<tr>
<td>NoSQL</td>
<td>a non-relational database system</td>
</tr>
<tr>
<td>OFCOM</td>
<td>an independent regulator and competition authority for the UK communications industries</td>
</tr>
<tr>
<td>OSGi</td>
<td>Open Service Gateway Initiative</td>
</tr>
<tr>
<td>OS X</td>
<td>a Unix-based graphical operating system developed by Apple</td>
</tr>
<tr>
<td>PaaS</td>
<td>Platform as a Service, one of the cloud service models</td>
</tr>
<tr>
<td>PHP</td>
<td>a programming language</td>
</tr>
<tr>
<td>PHR</td>
<td>Personal Health Record</td>
</tr>
<tr>
<td>ps</td>
<td>a UNIX/Linux command to display statistics on running processes</td>
</tr>
<tr>
<td>PS2</td>
<td>The PlayStation 2 game console developed by Sony</td>
</tr>
<tr>
<td>QALY</td>
<td>Quality Adjusted Life Year</td>
</tr>
<tr>
<td>RAID</td>
<td>Redundant Array of Independent Disks</td>
</tr>
<tr>
<td>Redpark</td>
<td>a company focusing on the production of computer accessories</td>
</tr>
<tr>
<td>REST</td>
<td>Representational State Transfer, an architecture style for designing web applications</td>
</tr>
<tr>
<td>ROI</td>
<td>Return on Investment</td>
</tr>
<tr>
<td>RSSI</td>
<td>Received Signal Strength Indication</td>
</tr>
<tr>
<td>RUBiS</td>
<td>Rice University Bidding System, a toolkit for web application benchmarking</td>
</tr>
<tr>
<td>SaaS</td>
<td>Software as a Service, one of the cloud service models</td>
</tr>
<tr>
<td>sar</td>
<td>a performance monitoring tool (command) for Unix/Linux-based systems</td>
</tr>
<tr>
<td>SD</td>
<td>standard deviation</td>
</tr>
<tr>
<td>SDK</td>
<td>Software Development Kit</td>
</tr>
<tr>
<td>SEMG</td>
<td>surface electromyography</td>
</tr>
<tr>
<td>SIG</td>
<td>Special Interest Group</td>
</tr>
<tr>
<td>SKPSMTPMessage</td>
<td>an open-source software framework for email operations</td>
</tr>
<tr>
<td>SLA</td>
<td>Service Level Agreement</td>
</tr>
<tr>
<td>SOAP</td>
<td>Simple Object Access Protocol</td>
</tr>
<tr>
<td>SPECweb2005</td>
<td>a performance benchmarking framework (a successor to SPECweb99) for web servers (retired in Jan. 2012)</td>
</tr>
<tr>
<td>SPECweb99</td>
<td>a performance benchmarking framework for web servers (retired in Oct. 2005)</td>
</tr>
<tr>
<td>SpO2</td>
<td>estimate of blood oxygen saturation</td>
</tr>
<tr>
<td>SQLite</td>
<td>a relational database system</td>
</tr>
</tbody>
</table>
SSD  Solid-State Drive
SV  the sum vector of accelerations in X-Y-Z axes
sysstat  a collection of performance monitoring tools for Unix/Linux-based systems
TDOA  Time Difference of Arrival
TI  Texas Instruments Incorporated
TI SensorTag  an BLE-based sensing device developed by TI
TOA  Time of Arrival
top  a UNIX/Linux command to display real-time status of a running system
TPC-C  an on-line transaction processing benchmark framework
TPC-W  a transactional web e-commerce benchmark framework (Obsolete as of Apr. 2005)
TTL  Transistor-Transistor Logic
TV  Television
Ubuntu  an open-source Linux operating system
UML  Unified Modelling Language
UPnP  Universal Plug & Play
USB  Universal Serial Bus
VA  Veterans Affairs
vCPU  virtual central processing unit
VM  Virtual Machine
VMM  Virtual Machine Manager
VSP  Vital Sign Parameter
Wi-Fi  a wireless local area network based on IEEE 802.11 standards
WSD  Whole System Demonstrator cluster randomised trial
Xanboo  a company providing remote home monitoring and control services
XBOX 360  a home video game console developed by Microsoft
Xen  an open-source hypervisor
XML  Extensible Markup Language
Year Zero  an initiative of the DALLAS programme
ZigBee  a wireless standard based on the IEEE 802.15.4
# Contents

1. Introduction .................................................................................................................. 1  
   1.1 Background and Motivation ....................................................................................... 1  
   1.2 Methodology and Scope of Research ......................................................................... 2  
   1.3 Contributions and Structure of this Thesis ............................................................... 5  

2. Telehealthcare and Recent Developments of Supportive Commodity Technologies .... 8  
   2.1 Telehealthcare ........................................................................................................... 8  
      2.1.1 Cost-Effectiveness Analysis .................................................................................... 8  
      2.1.2 Care Pathways and Telehealthcare for Patients with Chronic Diseases ............ 12  
      2.1.3 Case Study: Whole System Demonstrator (WSD) Cluster Randomised Trial .... 13  
      2.1.4 Some Related Work at Scale ................................................................................ 15  
         2.1.4.1 Delivery of Assisted Living Lifestyles at Scale (DALLAS) Programme ............ 15  
         2.1.4.2 enhanced Complete Ambient Assisted Living Experiment (eCAALYX) ......... 16  
         2.1.4.3 Integrated Network for Completely Assisted Senior Citizen’s Autonomy (inCASA) ........................................................................................................................................... 16  
         2.1.4.4 Home Telehealth Services of the US Department of Veterans Affairs (VA) ...... 17  
      2.2 Smart Home and Internet of Things .......................................................................... 17  
         2.2.1 Smart Home ........................................................................................................ 17  
         2.2.2 Internet of Things (IoT) ...................................................................................... 19  
      2.3 Sensor Technologies/Platforms ................................................................................ 20  
         2.3.1 ZigBee .................................................................................................................. 21  
         2.3.2 Bluetooth Smart (Low Energy) ........................................................................... 22  
         2.3.3 Arduino ............................................................................................................... 23  
      2.4 Cloud Computing and Mobile Cloud Computing ..................................................... 23  
         2.4.1 Some Essential Building Blocks ........................................................................... 24  
            2.4.1.1 Virtualisation ................................................................................................. 24  
            2.4.1.2 Load Balancing .............................................................................................. 25  
            2.4.1.3 Auto Scaling ................................................................................................. 25  
         2.4.2 Public Cloud Service Providers ........................................................................... 26  
            2.4.2.1 Google App Engine ....................................................................................... 26  
            2.4.2.2 Amazon Elastic Compute Cloud ..................................................................... 27  
         2.4.3 Economic Implications ....................................................................................... 27  
         2.4.4 Mobile Cloud Computing ................................................................................... 28  
         2.4.5 Related Work on Cloud-based/Mobile Cloud-based Healthcare ....................... 29  

3. Research Design ......................................................................................................... 31  
   3.1 System Requirements .................................................................................................. 31  
      3.1.1 Functional Requirements ...................................................................................... 31  
      3.1.2 Non-functional Requirements .............................................................................. 32  
   3.2 System Design .......................................................................................................... 34
3.2.1 High-level System Architecture................................................................. 34
3.2.2 Design of the User Agent........................................................................... 37
3.2.3 Design and Simulation of Home Telemonitoring Interventions.................. 39
3.2.4 Design and Simulation of the Service Gateway and App&DB Modules ......... 40
3.3 Comparative Cost-effectiveness Analyses..................................................... 41
  3.3.1 Comparisons of healthcare resource utilisation between simulated patients in different groups ........................................................................................................... 42
  3.3.2 Comparisons of costs and functional effectiveness with other telemonitoring systems .......................................................... 42
  3.3.3 Comparisons of costs and functionality with usual care ......................... 43
  3.3.4 Comparisons of costs and health effects with existing randomised controlled trials ........... 43

4. Real-time Human Fall Detection .................................................................... 46
  4.1 Introduction.................................................................................................... 46
  4.2 Literature Review and Related Work............................................................ 47
    4.2.1 Basic Settings for the Sensing Mechanisms ............................................ 47
    4.2.2 Classification Mechanisms and Achieved Accuracy ............................. 48
  4.3 Design and Implementation ....................................................................... 49
    4.3.1 Sensing Accelerations from Falls or Other Activities ............................. 49
    4.3.2 Power Consumption Experiments ......................................................... 52
    4.3.3 Integration with an External Sensing Unit .............................................. 53
    4.3.4 Fall Detection Algorithm ....................................................................... 55
    4.3.5 Implementation of Fall Detection Alerts ................................................. 55
  4.4 Cost-effectiveness Analysis ......................................................................... 57
    4.4.1 Estimated Costs and Effectiveness ......................................................... 57
    4.4.2 More on the Device Shaking Events and Accuracy ............................... 58
  4.5 Evaluation and Discussion ......................................................................... 58

5. Real-time Indoor Patient Movement Tracking ................................................. 60
  5.1 Introduction.................................................................................................... 60
  5.2 Literature Review and Related Work............................................................ 61
    5.2.1 Step Detection......................................................................................... 61
    5.2.2 Indoor Radio-based Location Tracking .................................................. 61
    5.2.3 Received Signal Strength Indication (RSSI)............................................ 63
    5.2.4 Kalman Filter ....................................................................................... 63
  5.3 Design and Implementation ........................................................................ 64
    5.3.1 Step Detector ....................................................................................... 65
    5.3.2 Kalman Filter for Distance Estimates .................................................... 68
    5.3.3 Mechanisms for Estimate Optimization ............................................... 69
  5.4 Cost-effectiveness Analysis ........................................................................ 72
    5.4.1 Estimated Costs and Effectiveness ......................................................... 73
    5.4.2 More on the Different Settings and Overall Accuracy ............................. 74
  5.5 Evaluation and Discussion ........................................................................ 76
6. Simulation of Patients’ Activities of Daily Living and Telehealthcare Interventions ..... 78
   6.1 Introduction ................................................................................................................. 78
   6.2 Literature Review and Related Work ............................................................................ 79
      6.2.1 Healthcare Simulation ............................................................................................ 79
      6.2.2 Statistics of Emergency Care Service Delivery ...................................................... 80
      6.2.3 Statistics of Falls among Older Adults and Related Issues ..................................... 82
      6.2.4 Statistics of Carers and Internet Usage among Older Adults .................................. 82
   6.3 Simulation Design ......................................................................................................... 83
      6.3.1 Modelling of Patients ............................................................................................. 86
         6.3.1.1 Modelling of Patients’ Activities of Daily Living ............................................... 88
         6.3.1.2 Modelling of Falls among Older Adults and related Telemonitoring and Emergency Services ................................................................. 90
         6.3.1.3 Modelling of Vital-Sign-Exceeding-Threshold Events and related Telemonitoring and Emergency Services ......................................... 100
         6.3.1.4 Modelling of Patients’ Web Surfing Behaviours ............................................... 107
      6.3.2 Modelling of User Agents ....................................................................................... 109
      6.3.3 Modelling of Other Stakeholders .......................................................................... 111
         6.3.3.1 Modelling of Healthcare Professionals ................................................................. 111
         6.3.3.2 Modelling of Carers ............................................................................................ 113
         6.3.3.3 Modelling of Emergency Tele-Consultant Team ................................................. 115
   6.4 Cost-effectiveness Analysis .......................................................................................... 117
      6.4.1 Results of a 90-Day Simulation based on Default Set-ups ....................................... 119
         6.4.1.1 Results of Fall-Event Simulation ................................................................................. 119
         6.4.1.2 Results of Vital-Sign-Exceeding-Threshold Event Simulation .................................. 121
         6.4.1.3 Allocation and Utilisation of Urgent Tele-consultation Resources .......................... 122
         6.4.1.4 Estimation of Costs and Effectiveness ................................................................... 125
      6.4.2 More on the Allocation and Utilisation of Emergency Tele-Consultants .................. 126
      6.4.3 Experiments on the Timing for Taking Vital Signs in the Afternoon ......................... 128
      6.4.4 Sensitivity Analysis ................................................................................................. 130
   6.5 Evaluation and Discussion .......................................................................................... 132

7. Simulation of (Mobile) Cloud Computing for Supporting Large-scale Real-time
   Telehealthcare Application ................................................................................................. 134
   7.1 Introduction ................................................................................................................ 134
   7.2 Literature Review and Related Work ........................................................................... 135
      7.2.1 Cloud Simulation Toolkits ....................................................................................... 135
         7.2.1.1 CloudSim .............................................................................................................. 136
         7.2.1.2 CloudSimEx ....................................................................................................... 136
         7.2.1.3 DynamicCloudSim ............................................................................................ 137
         7.2.1.4 DCSim ............................................................................................................... 137
      7.2.2 Web Application Performance Benchmarking Tools ................................................. 138
      7.2.3 Related Work .......................................................................................................... 138
List of Figures

Figure 2.1 Components of a Cost-effectiveness Analysis, after [19]........................................9
Figure 2.2 The Cost-Effectiveness Plane .............................................................................10
Figure 2.3 The Concept Diagram of a Typical Smart Home System.................................19
Figure 3.1 High-level System Architecture........................................................................35
Figure 3.2 Component Diagram of the User Agent Module...............................................37
Figure 3.3 Photo of a Vital Sign Monitoring Test.................................................................38
Figure 3.4 Comparative Effectiveness Analysis Based on Simulations..............................44
Figure 4.1 Signatures of SV and Accelerations in X, Y, and Z axis in an Intentional Fall Test (User Agent Mounted at Ear Side) ..................................................................................50
Figure 4.2 Signatures of SV and Accelerations in X, Y, and Z axis in an Intentional Fall Test (User Agent Placed in a Jacket Pocket) ...........................................................................51
Figure 4.3 Signatures of SV and Accelerations in X, Y, and Z axis during an Intentional Fall onto the Knees (User Agent held in Hand) ....................................................................51
Figure 4.4 Signatures of SV and Accelerations in X, Y, and Z axis during a Phone Shaking Test (User Agent held in Hand) .....................................................................................52
Figure 4.5 Signatures of SV and Accelerations in X, Y, and Z axis in a Device Throwing Test .........54
Figure 4.6 Signatures of SV and Accelerations in X, Y, and Z axis in a Device Dropping Test ........55
Figure 5.1 Signatures of SV and Y-axis Acceleration, and Heading in a Movement Test ..........66
Figure 5.2 Scenarios of Step Detection before Receiving New RSSI Measurements ..........67
Figure 5.3 Scenarios of Radius Manipulation Process ..........................................................70
Figure 5.4 Screenshot of the User Agent during Real-time Location Tracking ....................72
Figure 5.5 Tracked Paths based on Different Mechanisms ...............................................72
Figure 5.6 Average Estimation Errors based on Different Mechanisms .........................73
Figure 5.7 Maximum Estimation Errors based on Different Mechanisms .......................73
Figure 6.1 Concept Diagram of Our Healthcare Telemonitoring Simulation ....................84
Figure 6.2 High-level Structure (Class) Diagram of Our Telemonitoring Simulation ...........85
Figure 6.3 Patients’ Activities of Daily Living and their Transitions ..................................88
Figure 6.4 Post-fall Event Execution Algorithm for Intervention Group ...........................90
Figure 6.5 Post-fall Event Execution Algorithm for Control Group ..................................92
Figure 6.6 Post-vital-sign-exceeding-threshold Event Execution Algorithm for Intervention Group .................................................................101
Figure 6.7 Post-vital-sign-exceeding-threshold Event Execution Algorithm for Control Group .....102
Figure 6.8 Process and Data Flow Diagram of User Agent in our Simulation .............................................109
Figure 6.9 Modelling of Emergency Tele-Consultant Team Based on Queuing Theory ....................116
Figure 6.10 Total Number of Falls, Alerts and Cases Requiring Medical Care by Day ......................119
Figure 6.11 Average Number of 90-Day Falls, Alerts and Cases Requiring Medical Care by Time of Day..............................................................................................................................120
Figure 6.12 Average Number of 90-day Vital Sign Measurements Taken (by Both Groups) by Time of Day .....................................................................................................................................121
Figure 6.13 Average Number of 90-Day Vital Sign Alerts and Cases Requiring Medical Care (Intervention Group) by Time of Day ......................................................................................122
Figure 6.14 Number of Total Arrived Alerts, Waiting Alerts, and Idle Tele-Consultants Recorded by Time of the First Simulation Day ........................................................................................................123
Figure 6.15 Avg. Number of Total Assigned Alerts, Waiting and Idle Counts Per Tele-Consultant Per Day ......................................................................................................................................124
Figure 6.16 Average Alert Waiting Time and Tele-Consultant Idle Time per Day ................................124
Figure 6.17 Avg. Number of Total Assigned Alerts, Waiting and Idle Counts Per Tele-Consultant Per Day ......................................................................................................................................127
Figure 6.18 Average Alert Waiting Time and Tele-Consultant Idle Time per Day ................................128
Figure 6.19 Average Number of 90-day Vital Sign Measurements Taken (by Both Groups) by Time of Day (Based on a revised simulation) ........................................................................129
Figure 6.20 Average Number of 90-day Vital Sign Alerts and Cases Requiring Medical Care (Intervention Group) by Time of Day ......................................................................................129
Figure 6.21 Average Alert Waiting Time and Tele-Consultant Idle Time per Day ................................130
Figure 6.22 Changes of Number of Unnecessary A&E Attendances in Sensitivity Analysis ............131
Figure 6.23 Changes of the Number of Patients Requiring A&E Care in Sensitivity Analysis ............132
Figure 7.1 Cloud-based Multi-Tier Application Architecture .................................................................142
Figure 7.2 Application/Database Server Overloading in CloudSimEx Simulation ................................147
Figure 7.3 Load Dispatcher and Waiting Queue for Over-capacity Requests in Our Simulation ..........148
Figure 7.4 State Diagram of Workload Processing in the App/DB Server in Our Simulation ..........151
Figure 7.5 “7zip” CPU Benchmark Results of t2.micro Instances in MIPS ........................................155
Figure 7.6 “fio” I/O Benchmark Results of t2.mirco Instances in IOPS ................................................155
Figure 7.7 “iperf” Network Benchmark Results of t2.mirco Instances in Mbps ......................................156
Figure 7.8 Web Workload Test Configuration Diagram .................................................................159
Figure 7.9 Number of Cloudlets and Running VMs in the Comparison Simulations Using Different Auto Scaling Algorithms ..................................................................................................................178
Figure 7.10 Number of Cloudlets and Running VMs in the Comparison Simulations Using Different Auto Scaling Algorithms .................................................................179
Figure 7.11 Avg. Number of Web Session Requests Made by Patients (in the Intervention Group) by Time of Day .................................................................183
Figure 7.12 Avg. Number of Web SessionRequests Made by Carers by Time of Day ..............184
Figure 7.13 Avg. Number of Web Session Requests Made by Professionals by Time of Day ......185
Figure 7.14 Avg. Number of Web Sessions for Uploading Location Tracking Data by Time .......186
Figure 7.15 Avg. Number of Web Sessions for Uploading Vital Sign Data and Fall Detection Alerts by Time .........................................................................................187
Figure 7.16 CPU Loads on Web Application and Database Servers during Two Separate Time Periods ........................................................................................................188
Figure 7.17 Memory Loads on Web Application and Database Servers during Two Separate Time Periods ........................................................................................................189
Figure 7.18 I/O Loads on Web Application and Database Servers during Two Separate Time Periods ........................................................................................................189
Figure 7.19 I/O Loads on Database Servers based on Revised dbCloudlet Representation ..........192
Figure 7.20 The Number of Session Requests Generated for Uploading Patients’ Location Tracking Data based on a Fixed Time Interval .................................................................194
Figure 7.21 I/O Loads and the Number of Launched DB Servers based on Different Auto Scaling Algorithms ..................................................................................................198
List of Tables

Table 2.1. Costs (2009-10) Per Participant and QALY Comparison between Usual Care and Telehealth over 12 Months (Extracted and adjusted from [30]) .......................................................... 14
Table 4.1 iPhone 5’s Battery Life Time Measurements in Power Consumption Experiments ............ 53
Table 4.2 Fall Detection Algorithm ............................................................................................... 56
Table 4.3 Results of our Trials ......................................................................................................... 57
Table 5.1. Two Sets of Kalman Filter Equations for State Estimation, after [126] ......................... 64
Table 5.2 RSSIs Recorded at different Distances from Sensor S1 .................................................. 64
Table 5.3 Step Detection Algorithm .............................................................................................. 66
Table 5.4 Tight-Coupling Sensor Fusion Mechanism ...................................................................... 71
Table 5.5 Estimation Errors based on Different mechanisms .......................................................... 74
Table 5.6 Estimated Equipment Costs based on Our Settings ...................................................... 74
Table 5.7 Changes of Estimation Results due to Different Parameter Settings ............................... 75
Table 6.1 Patients’ Profiles in our Healthcare Simulations .............................................................. 117
Table 6.2 Grouping of Patients based on their Internet Habits for Modelling Web Session Workloads ............................................................................................................................. 118
Table 6.3 Professionals and Carers for Modelling Web Session Workloads ................................... 118
Table 6.4 Default Set-ups of the Emergency Tele-consultants ....................................................... 118
Table 6.5 Default Set-ups of the Cost Elements .............................................................................. 119
Table 6.6 Estimation of Costs and Effectiveness of Healthcare Telemonitoring ............................... 125
Table 6.7 Estimated Healthcare Savings through Telemonitoring ............................................... 126
Table 6.8 A Modified Set-up with One Whole-day and 16 Daytime Emergency Tele-consultants .... 127
Table 6.9 A Modified Set-up with One Whole-day and 12 Daytime Emergency Tele-consultants .... 130
Table 7.1 Pseudo code snippets of CloudSim main() method ....................................................... 144
Table 7.2 Pseudo code snippets of CloudSim.startSimulation() method ..................................... 145
Table 7.3 Pseudo code snippets of our runSimulationEngine() method ...................................... 146
Table 7.4 Pseudo code snippets of our main() method ................................................................. 146
Table 7.5 Configurations for App/DB Servers and Workloads in Comparison Simulations ............ 148
Table 7.6 Results of Workload Execution Simulation Using CloudSimEx ...................................... 149
Table 7.7 Results of Workload Execution Simulation Using Our Proposed Mechanism ................ 150
Table 7.8 Key Documented Performance Attributes of Amazon EC2 t2.micro Instances ............ 154
Table 7.9 Performance Figures for VM Representation in Cloud Simulations ............................... 157
Table 7.10 Pricing of EC2 t2.micro Linux Instances ..................................................................... 158
Table 7.11 Savings of On-Demand over Reserved t2.micro Linux Instances based on EU (Ireland) 158
Table 7.12 Recorded CPU, Memory, and I/O Loads on Web and RUBiS Application Server .......... 161
Table 7.13 Recorded Loads outside the Time Period of the Experiments ................................ 161
Table 7.14 Calculated Average Loads per Request ..................................................................... 162
Table 7.15 Estimated Loads per Request on Web and RUBiS Application Server ..................... 163
Table 7.16 Recorded CPU, Memory, and I/O Loads on MySQL Database Server ..................... 164
Table 7.17 Recorded Loads outside the Time Period of the Experiments ............................... 164
Table 7.18 Calculated Average Loads per Database Operation/Connection ............................. 165
Table 7.19 Estimated Loads per Operation/Connection on MySQL Database Server ............... 166
Table 7.20 Average Execution Time per Database Insertion based on Different Data Sizes ...... 166
Table 7.21 Representations of asCloudlets and dbCloudlets in Cloud Simulation ..................... 168
Table 7.22 Pseudo code snippets of our Regional Load Balancer Algorithm ........................... 172
Table 7.23 Pseudo code snippets of our Auto Scaling (FLUCAS) Algorithm ........................... 176
Table 7.24 Configurations for App Servers and Workloads in the Comparison Simulations ....... 177
Table 7.25 Usages of VM Charging Periods, Costs and Response Time in the Comparison Simulations Using Different Auto Scaling Algorithms .................................................. 179
Table 7.26 Default Set-ups of the VMs in the 10-day Simulation .......................................... 180
Table 7.27 Grouping of Patients (in the intervention group) and Carers based on their Internet Habits ............................................................................................................................................ 181
Table 7.28 Default Set-ups for a Session Generated by Patients and Carers with Different Internet Habits ............................................................................................................................................. 181
Table 7.29 Default Set-ups for a Session Generated by Professionals and Tele-consultants ......... 181
Table 7.30 Default Set-ups for a Session Generated by the User Agent .................................... 182
Table 7.31 Number of Patients Requiring Medical Care and Average Number of Reviews per Professional ................................................................................................................................................. 185
Table 7.32 Results of Workload Execution during Two Separate Time Periods ....................... 190
Table 7.33 Estimated Costs of Required Servers on the App&DB Module ................................. 191
Table 7.34 Estimated Equipment Costs of the Proposed Telemonitoring System ................... 191
Table 7.35 Revised Representations of dbCloudlet I/O Loads for Type II and III Requests .......... 192
Table 7.36 Results of Workload Execution based on the Revised I/O Representations .............. 193
Table 7.37 Maximum Number of Sessions Generated By User Agents based on Different Time Interval for Data Uploading ...................................................................................................................... 194
Table 7.38 Results of Workload Execution based on an I/O Capacity of 2480 IOPS of the Database Server .......................................................................................................................................................... 196
Table 7.39 Estimated Costs of Required Servers on the App&DB Module ............................... 196
Table 7.40 Results of Workload Execution based on Different Auto Scaling Algorithms .......... 198
1. Introduction

1.1 Background and Motivation

According to [1], in 2012 one in nine persons in the world are aged 60 years or over and the number is projected to increase to one in five by 2050. In the European Union alone, people aged 65 and over are expected to rise from 17% of the population in 2010 to 30% by 2060 [2]. In addition, the prevalence of chronic diseases continues to increase [3]. For example, more than 6% of people aged 10-79 years in the European Union, or 30 million people, had diabetes in 2011 [3]. The rise in both ageing and chronic disease populations has become a global issue which calls for a top policy priority to provide proper access to quality healthcare. Though information and communications technologies (ICTs) have been used in almost all aspects of our life, there remains a considerable question relating to the low adoption rate of home telehealthcare technologies. One of the main reasons, as indicated by a number of studies [4-6], is a lack of robust evidence for cost-effectiveness.

To address this issue, we set up as our core objective the cost-effective design of a real-time home healthcare telemonitoring system based on mobile cloud computing. Our hypothesis is that the increasing availability of smartphone-centric technologies can dramatically reduce the infrastructure costs and intrusion of telemonitoring. The rationale for focusing on home healthcare and smartphone-centric technologies was two-fold. Firstly, studies [7,8] have shown that home-based treatments for older patients as a substitute for hospital care can produce better clinical outcomes and reduce healthcare expenditure. Secondly, the use of smartphones is becoming pervasive, i.e. subscriptions reached 2.6 billion in 2014 and are forecasted to reach 6.1 billion (or 70% of the world population) in 2020 [9], representing an excellent opportunity to bring telehealthcare into the home, if cost-effective solutions based on smartphone-centric technologies are in place. Here smartphone-centric technologies refer to not only the technologies that are embedded in the smartphone, but also those which can seamlessly integrate with a smartphone and substantially enhance its functionality through add-on capabilities, such as data storage provided by cloud computing and sensory functions by external sensing devices.

Our second objective was to develop a simulation environment to produce robust evidence for cost-effectiveness of a telemonitoring system so as to explore technology choices in different settings prior to moving to full-scale trials. To this end, a number of simulation models were carefully designed to computationally provide a macro view of possible healthcare scenarios. By using these models, hundreds of thousands of patients and their activities of daily living (ADL), as well as other stakeholders, such as healthcare professionals, emergency tele-consultants and carers, and their interactions with the patients using our proposed system were simulated.
Currently, as there is neither a universal definition of home healthcare telemonitoring nor a unified terminology for it, this research creates its own definition with no intention to make a distinction between telehealth and telecare. In this thesis, “home healthcare telemonitoring” is defined as “a type of healthcare application which remotely measures and monitors the vital signs, safety, and activities of the patients and elderly at home”, and is generally abbreviated as “telehealthcare”, “telehealth”, or “telemonitoring” interchangeably.

Two overviews [10,11] of this work were reported at MobileMed 2013 and HealthInf 2014, respectively. This thesis, which expands significantly on those two presentations in all aspects, such as the overall scope, technical details and extent of the evaluations, represents the details of the author’s work.

1.2 Methodology and Scope of Research

A broad literature review in related fields was conducted during the early stage of this work. This was done to better understand the development of telehealthcare and healthcare economic evaluations, as well as the implications of recent ICT advances, such as sensor technologies, smart home, Internet of Things (IoT), cloud computing, and mobile cloud computing, in support of achieving a cost-effective design of real-time home healthcare telemonitoring. Through this review, certain smartphone-centric technologies essential for our cost-effective design were identified.

A proof-of-concept system consisting of three main monitoring functions, namely vital sign monitoring, safety monitoring (for fall detection) and movement pattern monitoring (for real-time indoor location tracking), was developed using smartphone-centric technologies (including an iPhone 5 and a number of external sensors) to demonstrate the achievable functionality and to provide supportive data for our simulation modelling. The design of vital sign monitoring was more a technological than a research problem, as it involved primarily the implementation of a workable iPhone application (or abbreviated as App) to interwork with eight external Arduino-compatible sensors*, including blood pressure, body temperature, breathing air flow, electrocardiogram (ECG), glucometer, sweating, heartbeat rate and blood oxygen saturation (SpO2), and body posture sensors. In contrast, the design of safety and movement pattern monitoring, using an iPhone 5 and Bluetooth Low Energy (BLE) sensors**, involved the modelling of the problem domains and the development of novel solutions to address our identified research problems, such as, reducing costs and intrusion, and enhancing usability and accuracy.

* More discussions about Arduino can be found in section 2.3.3.
** An introduction about BLE (or called Bluetooth Smart) sensors is given in section 2.3.2.
A novel simulation environment with two main themes, constituting an essential part of this research, was developed based on discrete-event simulation (DES) using Java. The first was the simulation of both the activities of daily living of 100,000 patients (divided into a control group and an intervention group) and home healthcare telemonitoring interventions for patients in the intervention group. For example, on each simulation day, each patient would perform a number of different activities (such as wake up, go to bed, walk, take vital signs a few times based on his/her unique care plan, receive a home visit if prescheduled, go online to interact with our monitoring system according to his/her personal Internet habit, visit a general practitioner (GP) if pre-booked, have a fall based on a Poisson distribution, or make an accident and emergency (A&E) department attendance after a urgent event). A number of different stakeholders, including carers, healthcare professionals, and emergency tele-consultants, were also created through simulation to interact with patients through the proposed telemonitoring interventions/system.

In addition, specific healthcare scenarios were defined to deal with falls and vital-sign-exceeding-threshold events among patients in both groups to model the utilisation of healthcare services with assumptions based on both published data and our knowledge and justification. For example, when no telemonitoring is in place, some patients immediately after a fall might not be physically able to seek the required medical care by themselves, whereas some patients might tend to make unnecessary A&E attendances. However, with telemonitoring, these patients would get assistance to attend A&E when necessary, and avoidable A&E attendances could be reduced. An evaluation on the allocation and utilisation of the emergency tele-consultants was performed to reduce the alert waiting time and tele-consultant idle time. Then cost-effectiveness analysis was conducted to assess whether the change of healthcare service utilisation due to the introduction of the proposed telemonitoring intervention could in fact reduce the overall costs of healthcare service provision under certain circumstances. These included the assessment on whether the savings from reducing unnecessary A&E attendances recoup the expenses of providing telemonitoring. Finally, sensitivity analysis was performed by testing two key parameters, including the percentage of unnecessary A&E attendances and the number of annual A&E attendances per patient, to examine their impact on our models.

The second theme of our novel simulation work was to emulate a (mobile) cloud computing scenario in which cloud-based web application and database servers are able to automatically scale themselves in a cost-effective manner in response to the amount of incoming dynamic web session workloads. Both the representations of these cloud-based servers (based on a certain type of virtual machine instances available in the real public cloud market) and workloads (based on CPU, memory and I/O requirements) were carefully modelled through empirical benchmarking experiments. Three types of workloads were defined to represent the requests for uploading load-intensive monitor data.
(such as 50,000 emulated patients’ real-time location tracking data) and for reviewing/updating/commenting on personal health data and/or diagnosis, performed by the emulated patients in the intervention group and other stakeholders, including carers, healthcare professionals, and tele-consultants.

Moreover, an existing cloud computing simulation toolkit, i.e. CloudSimEx (an extension of CloudSim)\(^3\), was chosen based on a broad survey on cloud simulation frameworks. Then we enhanced it with a number of new features, such as the submission and execution of runtime dynamic workloads, and a mechanism to queue up and resume over-capacity requests in a real-world manner. A cost-aware load balancing algorithm based on least pending requests and a novel forward-looking unused capacity-based auto scaling (FLUCAS) algorithm for the cloud-based servers were proposed to enhance system performance and scalability, and reduce costs. Finally, cost-effectiveness analysis together with sensitivity analysis was conducted to investigate and justify the costs and effectiveness of our proposed solutions, as well as to test the parameters in the modelling of workloads.

In sum, one of the main focuses of our work was to design, implement, and evaluate fall detection and real-time location tracking prototypes, using smartphone-centric technologies, aimed at enhancing the usability, in terms of ease of use, ease of deployment and minimal intrusion, and accuracy at low costs compared with other studies. This study also involved the novel modelling work of specific home healthcare telemonitoring and (mobile) cloud computing scenarios, and the design and implementation of simulations to computationally create those scenarios. This would enable us to explore technology choices/settings and cost factors, develop robust algorithms, and compile evidence for cost-effectiveness of our solutions.

Although both [5] and [12] stressed the importance of incorporating business modelling into the development process of healthcare technologies so as to ensure a successful value-driven implementation and service provisioning, as well as to identify the underlying costs, the design of business models were not included in this research due to its complexity. For cost-effectiveness analysis, this study did not cover the design and implementation of randomised controlled telemonitoring trials, patient care pathways and business models to measure costs and effects, as well as the estimation of cost-effectiveness thresholds and acceptability curve, as normally done in the health sector. Instead, we proposed a comparative approach by examining the costs and effectiveness of our technology solutions and benchmarking with other work.

Meanwhile, this study was confined to assess the effect of telemonitoring from technological viewpoints, such as usability, functionality, and accuracy, as well as the achievable improvements on

\(^3\) Introductions of CloudSim and CloudSimEx are given in sections 7.2.1.1 and 7.2.1.2, respectively.
the resource utilisation as evidenced in the healthcare and cloud computing simulations. This restriction was due to the lack of available published quantitative data and lack of access to clinical data sets already collected by other studies on the clinical and health outcomes, such as quality of life and life expectancy, of urgent medical care for patients having a fall or vital sign abnormalities. With regard to cloud computing, this study emphasised on maximising the overall performance and scalability and reducing the costs of cloud usage, but did not cover other important issues, such as network dynamics, security, data privacy, database, disaster recovery, service level agreement (SLA), and legal and regulatory framework.

1.3 Contributions and Structure of this Thesis

The main contributions of this work are as follows:

− The realisation of a proof-of-concept system that demonstrates a low-cost, easy-to-use, less intrusive, but feasible and robust smartphone-based solution for fall detection. Specifically, the solution directly addresses the two commonly found issues, i.e. the need for fixed placement of the phone or sensors, and over draining of battery power, in related work. Moreover, the sensitivity and specificity of our system for successfully detected falls and recognised non-fall activities were both 95.5%, whilst the specificity for effectively identifying device drops or throws was 100%.

− A solid proof that cost-effective design of indoor real-time location tracking is possible based on commodity smartphone-centric technologies with an average estimation error of 0.47 metres in our trials. Higher usability (including deployability and portability) and reliability of our solution at low costs were achieved through the proposed sensor-fusion approach with the integration of a distance estimation algorithm based on received wireless signal strengths, a step detection mechanism based on acceleration and a Kalman filter to improve distance estimation.

− The development of the novel simulation models to emulate both a large number of patients and their activities of daily living, and a number of different stakeholders and their interactions with patients through the proposed telemonitoring intervention/system. Moreover, the modelling of specific telehealthcare scenarios to enable us to predict patient behaviours towards the use of healthcare services, such as GP and A&E, after a fall or vital sign abnormalities with and without telemonitoring interventions through simulations, and to formulate evidence for cost-effectiveness of our telemonitoring interventions.
The new features developed, such as the submission and execution of runtime dynamic workloads, and a mechanism to queue up and resume over-capacity requests in a real-world manner, for a popular cloud simulation toolkit, and the modelling work on the representations of cloud-based web application and database servers, and their workloads through empirical benchmarking experiments.

The novel simulation work to integrate a large amount of web session workloads from a telemonitoring domain with dynamic provisions of cloud resources and produce evidence for cost-effectiveness. Besides, the proposed FLUCAS algorithm demonstrated its novelty to enhance cloud-based system performance and scalability and reduce costs in our specific healthcare scenarios.

A methodology for using simulations to enable ourselves to make a case about the cost-effectiveness of the telemonitoring solution prior to moving to full scale trials on a more scientific basis.

A systematic approach to integrating all our work together to help justify the great potential of achieving cost-effective home healthcare telemonitoring based on smartphone-centric technologies.

The remainder of this thesis is organised as follows. The next chapter introduces the development trends and related work in telehealthcare, sensor technologies, digital home, IoT, cloud computing, and mobile cloud computing based on literature review. In chapter 3, we elaborate our research design concerning system requirements, system architecture, and our proposed approach for conducting comparative cost-effectiveness analysis, which serves as a framework for the subsequent work.

In chapter 4 and 5, the design of smartphone-based fall detection and real-time indoor location tracking systems together with our solutions to effectively address the identified issues and cost-effectiveness analysis based on the results of experiments are described. Chapter 6 presents the design of healthcare simulation work, which emulated a large number of patients and their activities of daily living, and a number of stakeholders, with especial emphases on the modelling of telehealthcare interventions for falls and vital-sign-exceeding-threshold events. It also presents the results of cost-effectiveness analysis and concludes with our evaluation and comments on future work. Chapter 7 details the design of cloud simulation, which integrated web session workloads from a telemonitoring domain with dynamic provisions of cloud resources, modelled through empirical benchmarking experiments. It also provides our assessments of the evidence for cost-effectiveness and suggestions
for future work. Chapter 8 concludes this thesis with both a critical review of what have been achieved and recommendations for further research.
2. Telehealthcare and Recent Developments of Supportive Commodity Technologies

2.1 Telehealthcare

The concept of telehealthcare or telemedicine has been explored for more than thirty years, as evidenced by the emergence of nurse call centres in the 1970s in the UK. As mentioned in section 1.1, in recent years, the problems of ageing and increasing numbers of people with chronic diseases have further underpinned the importance of telehealthcare. Therefore, a great number of studies on remote home care have emerged. According to three systematic reviews [13-15] of the existing scientific literature on telehealthcare, together covering the period from 1990 to 2011, the trend in telehealthcare research focuses was identified as follows:

- In the early 1990s: vital sign parameter (VSP) measurement in a wired environment was the main stream;
- In the mid-1990s: video- and tele-consultation and nursing became the largest share of research work;
- In the late 1990s and early 2000s: home telehealthcare based on smart home*4 technologies and mobile telecommunications platform, as well as wearable sensors, started to gain momentum; the design of information and support systems was one of main focuses; and
- In the late 2000s and early 2010s: vital sign parameter measurement with video- or tele-consultation continued to be the main stream.

Although the number of mobile healthcare applications are increasing quickly, in a systematic review of 83 identified smartphone-based applications (from 2,894 articles), [16] concluded that most of them are standalone applications and the full potential of smartphones has not been realised yet. This report also revealed that among the 83 applications, only five focused on telehealthcare or telemedicine.

2.1.1 Cost-Effectiveness Analysis

The increasing demand for better healthcare, in terms of improved quality, effectiveness, and long-term sustainability of healthcare systems, has manifested in the need to provide better evidence for informed decision making through economic evaluation. In this context, evidence-based medicine (EBM), health technology assessment (HTA) and comparative effectiveness research (CER) have

*4 The concept and development of smart home are presented in section 2.2.
been used respectively in many countries and related authority organisations to evaluate the benefits and harms of alternative treatments, technologies or healthcare deliveries. Among all techniques of economic evaluations in healthcare, **cost-effectiveness analysis** is widely adopted.

The National Institute for Health and Clinical Excellence (NICE) defined **cost-effectiveness analysis** as [17]: “an economic study design in which consequences of different interventions * are measured using a single outcome, usually in ‘natural’ unit (for example, life-years gained, deaths avoided, heart attacks avoided or cases detected). Alternative interventions are then compared in terms of cost per unit of effectiveness.”

To conduct cost-effectiveness analysis, [18,19] suggested that three types of costs need to be considered:

- **Direct costs**: such as drugs, staff time, equipment, transport of patients
- **Indirect (or Productivity) costs**: production losses, other uses of time
- **Intangibles**: pain, suffering, adverse effects

It is crucial to note that analyses adopting different perspectives need to include or exclude diverse cost elements [19]. For example, an analysis from an insurance company’s perspective does not need to include patients’ transportation, while one from a societal perspective would need to include such costs. Therefore, an analysis should assume the same perspective for different interventions when comparing their costs.

![Figure 2.1 Components of a Cost-effectiveness Analysis, after [19]](image)

The effects of an intervention generally refer to the changes in patients’ health status. Since there is no direct way to measure health status, a cost-effectiveness analysis instead examines patients’ quantity and quality of life with a given health status [19]. Figure 2.1 represents the concept that there

---

*5 According to [19], “a health intervention can be a treatment, screening test, or primary prevention technique (for example, vaccinating children to prevent measles).”
are changes in the health status, associated costs and resulting quality of life and life expectancy of an observed group of patients having received an intervention for a period of time. When the effects of different interventions are estimated, **cost-effectiveness ratios (CERs)** and **incremental cost-effectiveness ratios (ICERs)** can then be calculated for independent interventions and for mutually exclusive interventions respectively to assess the extent to which a new intervention can be regarded as cost effective. Equations (2-1) and (2-2) provide the formulas for the calculation [18].

(i) **For independent interventions**

\[
CER = \frac{C}{E} \tag{2-1}
\]

Where \( CER \): cost-effectiveness ratio

\( C \): costs of an intervention

\( E \): health effects produced, e.g. life-years gained (LYG) or quality adjusted life years (QALYs)

Then, all interventions are ranked based on their CERs and the intervention with the lowest CER should be given priority over the others.

(ii) **For mutually exclusive interventions**

\[
ICER = \frac{\Delta C}{\Delta E} \tag{2-2}
\]

Where \( ICER \): incremental cost-effectiveness ratio

\( \Delta C \): difference in costs between two interventions

\( \Delta E \): difference in health effects between two interventions
The cost-effectiveness plane (see Figure 2.2) can be used for evaluating the differences in costs and effects between two interventions of interest. It is clear that a new intervention, which costs less and has increased effects with both $\Delta E$ and $\Delta C$ fallen into quadrant IV, should be adopted. NICE [17] recommends that for cost-effectiveness analysis, health effects should be expressed in QALYs. A QALY combines life expectancy and quality of life of patients into a single index number. Each extra life-year is given a value between 1 (best possible health) and 0 (worst possible health or death). As such, cost-effectiveness is expressed as “£ per QALY”. However, since there are some uncertainties, such as different methodologies adopted and potential variation in the estimates of costs and effects, associated with an intervention, it is important that sensitivity analysis is adopted to test all the assumptions used in the cost-effectiveness analysis. (More information about how to calculate costs and QALY can be found in existing literature on the topic, such as [17] and [19].)

With regard to the cost-effectiveness of telehealthcare solutions, a 2007 systematic review [14] of 98 randomised trials and observational studies concluded that cost-effectiveness of these interventions was less certain, and that there was insufficient evidence of the effects of home safety and security alert systems. Although in recent years the situation has gradually been improved, a 2010 systematic review [20] of economic evaluations found only 33 articles that measured both costs and non-resource consequences of using telemedicine in direct patient care. It concluded that most studies lacked information about perspective and costing method. Another 2014 systematic review [21] performed a wide-ranging search of four electronic literature databases and two telemedicine journals and found only 17 studies, all published after 2007, that performed economic evaluations of telehealthcare interventions based on QALYs. It also stated that the estimation of QALY gain reported by most evaluations, however, was less convincing.

As for the effect of telemonitoring ADL of the elderly on their quality of life, a 2013 review [22] of 25 selected studies stated that there is a gap in research on this subject and concluded that long-term studies based on larger sample size with clinically relevant outcomes and comparable outcome measures are needed. It also identified a number of preferred sensor characteristics, such as accuracy, reliability, ease of use, comfort of use, lightweight, small size, and non-intrusiveness, which are the key factors for acceptance. Specifically, it mentioned that invasive tools, such as video cameras and microphones, were resisted by participants. All the aforementioned problems and suggestions represent both an opportunity and a challenge for this research to make a real contribution to the knowledge in this field.

---

*6 For example, methodologies based on probability distributions for estimating the likelihood of subjects becoming ill and going to see a doctor, and Markov models for incorporating and tracking changes in the quality of life and the cost of a disease over time.
2.1.2 Care Pathways and Telehealthcare for Patients with Chronic Diseases

In general, different countries or regions might well have their unique patient care pathways which define the course of actions to be taken for the care of patients with different diseases. For example, according to [23], after a patient is diagnosed with a chronic obstructive pulmonary disease (COPD) in North West Sussex, England, structured care in primary care plays the central role in providing a personalised care plan, conducting periodical reviews, and managing specific problems/symptoms by offering, for example, assessments, advices and therapies. When necessary, the primary care will refer a patient to specialised care services.

Meanwhile, telehealth monitoring is incorporated into this care pathway when a patient is identified by a COPD nurse specialist as suitable. Based on the telehealth, each day the monitor equipment reminds the patient of taking his/her vital sign measurements and automatically sends the data to the integrated care platform (ICP). When the measurements fall outside pre-defined parameters, an alert is displayed on the triage screen of the ICP for review performed by specialist nursing teams on a daily basis and further discussions with the GP are made if required. Patient reviews for suitability of continued telehealth are performed every 12 weeks. All these arrangements not only help achieve better integration of acute care, specialist care, primary care, community care and home care, but also help support patient self-management at home.

According to [24], hospital readmissions for patients with chronic heart failure are common, e.g. about one in four patients are readmitted within three months, whilst effective multidisciplinary specialist services can help reduce recurrent hospital stays by 30–50% and have a positive effect on patients’ quality of life and life expectancy. Although it is widely believed that telehealthcare can play an important role in supporting home care and self-management for patients after hospital discharge, currently it is still not common to have telehealthcare included in a chronic disease care pathway. For example, the guidance regarding the pathways for both COPD and type 1 diabetes developed by the NICE [25] does not include telehealthcare. In our opinion, this can be attributed to both the low adoption rate of telehealthcare technologies and the lack of sustainable business models.

[26] suggested that in the UK a healthcare telemonitoring system be integrated into primary care with a team-based approach for information sharing among different healthcare sectors for continuity of care. It also suggested that a 24-hour telemonitoring response centre be vital with non-urgent and urgent data being forwarded to the clinical team (i.e. the GP that the patient was registered with) and the most appropriate service (after being triaged for immediate action), respectively. To integrate telehealth into the care pathways for patients with long-term conditions in North Yorkshire and York, England, [27] reported that an intense process of clinical engagement had been conducted and a number of changes, such as to offer specialist opinion within 24 hours in acute trusts and to provide
early supported discharge, had been made to the pathways. These were done to ensure that care is provided closer to home when appropriate and self-management is emphasised. The outcomes from the phase one implementation included a 40% reduction in non-elective hospital admissions and a 28% drop in A&E attendances. Meanwhile, it helped community services staff reduce their need to travel.

Based on these examples, we found that in recent years, home healthcare telemonitoring, though still not pervasively adopted, has been mainly used in some countries/regions for supporting patient self-management at home, serving as a replacement of long-term institutional care for some specific patients or a practical measure for early supported discharge and post-discharge support, such as home visits made by community service nurses in England. As [26] commented that 13 identified telemonitoring projects in mid-2000s in the UK had either minimal or no integration into normal healthcare services, these also indicated that in recent years, there have been much improvement in the integration of telemonitoring and patient care pathways in the UK. According to [155], the weighted average (unit) costs of inpatient stays (including elective and non-elective) and all outpatient procedures were £697 and £135, respectively. Therefore, we believed that significant savings for public healthcare could be achieved, if effective home telehealthcare is in place to help reduce patients’ length of stay in hospitals.

2.1.3 Case Study: Whole System Demonstrator (WSD) Cluster Randomised Trial

In order to better understand what telecare and telehealth are capable of and how the technology supports people to live independently and take control of their own health and care, the Department of Health in England launched the Whole System Demonstrator (WSD) programme in May 2008. The total project fund was about 31 million pounds sterling. Three sites, Kent, Cornwall and Newham, were selected to be part of a cluster randomised controlled trial. With 238 GPs and 6,191 patients with diabetes, COPD, and coronary heart disease (CHD), it was believed that this trial is the world’s largest randomised controlled trial of telecare and telehealth [28].

Under this trial, each intervention participant was given a home unit together with a pendant alarm and up to 27 peripheral devices for functional monitoring (such as the home unit and bed and chair occupancy sensors), security monitoring (such as infrared movement sensors and property exit sensors) and standalone devices (not connected to a monitoring centre, such as big button phones).

Key Findings:

With regard to the key findings of this trial, the Department of Health in Dec. 2011 [28] announced that if used correctly telehealth can deliver a 15% reduction in A&E visits, a 20%
reduction in emergency admissions, a 14% reduction in elective admissions, a 14% reduction in bed
days, and a 45% reduction in mortality rates. However, several in-depth studies on the effect and cost-
effectiveness of this trial reached different or unfavorable conclusions, as in the following:

- Regarding the effect of telehealth interventions on use of secondary care and mortality, [29] found that though both hospital admissions and mortality for intervention patients were lower, there were no significant differences between the intervention group (containing 1,605 patients) and the control group (containing 1,625 patients) both in the number of elective admissions, outpatient attendance, and emergency visits and in notional hospital costs to commissioners of care.

- Regarding the cost-effectiveness of telehealth for patients with long term conditions, [30] found that the QALY gained by patients using telehealth in addition to usual care was similar to that by patients receiving usual care only, and that total costs in relation to telehealth were also higher. As such, this study concluded that telehealth does not seem to be a cost effective addition to standard support and treatment. Table 2.1 shows some key outcomes of this economic evaluation study.

<table>
<thead>
<tr>
<th></th>
<th>Usual care</th>
<th>Telehealth</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Equipment costs</strong> ^7 (base unit and peripherals)</td>
<td>£15.2</td>
<td>£674</td>
</tr>
<tr>
<td><strong>Total Raw Mean Costs</strong> (including delivery and equipment)</td>
<td>£5,559</td>
<td>£6,384</td>
</tr>
<tr>
<td><strong>QALY (raw mean difference)</strong></td>
<td>0.549</td>
<td>0.564</td>
</tr>
<tr>
<td><strong>ICER (£ per QALY)</strong></td>
<td>-</td>
<td>£92,000</td>
</tr>
</tbody>
</table>

Table 2.1. Costs (2009-10) Per Participant and QALY Comparison between Usual Care and Telehealth over 12 Months (Extracted and adjusted from [30])

- In respect of the effect of telecare on use of health and social care services, [31] concluded that telecare did not significantly reduce the use of those services.

- As for the adoption of telehealth and telecare interventions, [32] identified that concerns about both competency to operate equipment and threats to identity, independence and self-

^7 According to [30], costs of base units were annuitised over five years and costs of peripherals were annuitised either over the lifetime of the item (if information was available from sites or manufacturers’ specifications) or over five years.
care (which might be undermined, among others, by not getting outside, but doing monitoring indoors even on holidays) were two of the main barriers to adoption within this trial.

Although this large-scale trial did produce some positive evidence for the effects of telehealth interventions, there was some adverse evidence for both effectiveness and cost-effectiveness. Accordingly, this research considers that the WSD trial could serve as an important reference for conducting cost comparison, selecting inexpensive technologies, devising proper service models and designing the system architecture of the proposed home healthcare telemonitoring system.

2.1.4 Some Related Work at Scale

The pressing demand for better telehealthcare solutions has resulted in many other government-funded pilot projects. In the following subsections, we briefly introduce three large-scale programmes, each with different focuses and objectives from this research. However, by reviewing these programmes, our understanding of telehealth related issues was improved.

2.1.4.1 Delivery of Assisted Living Lifestyles at Scale (DALLAS) Programme

The Delivery of Assisted Living Lifestyles at Scale (DALLAS) programme was launched in May 2012 with £37 million by the Innovate UK (formerly the Technology Strategy Board) and joint funded by the National Institute for Health Research and the Scottish Government. The objective of the DALLAS programme was to explore ways of using innovative products, systems and services to create more independent lifestyles and help nearly 170,000 older people across the UK by summer 2015 with improved health, wellbeing and quality of life.

Four consortia were tasked to run this programme and test it with communities throughout the UK. Four major initiatives, including i-Focus, Year Zero, More Independent and Living it Up, were launched. A preliminary evaluation framework [33] was developed, aiming to measure future health and wellness outcomes, and to identify how new technologies and services are used, accepted and embedded into people’s live. It also planned to look at best practices for person-centred design, as well as conduct assessment of care technologies and services delivered at scale.

According to a recently published study [34], among the five identified challenges for designing and delivering new services under this programme, the only one concerning technology development was interoperability and information governance due to the dominance of proprietary models. With regard to telehealth, 1,600 patients under the More Independent initiative [35] were recruited in Liverpool to use telehealth based on Philips Motiva system, comprising a TV set top box or a tablet computer together with sensory equipment, to take vital sign parameters and receive education videos.
or reminders from the central monitor centre, called telehealth hub. The monitoring was carried out by, for example, the GP practice or health trainers, and could include direct referral. For patients who only needed self-monitoring of blood pressure levels, health messages were sent and received from their own mobile phones.

At the time of this writing, neither detailed information about delivered products and services, nor evaluation results have been found by this research. However, we considered that the abovementioned telehealth service could serve as a reference for the formulation of our healthcare intervention scenarios.

### 2.1.4.2 enhanced Complete Ambient Assisted Living Experiment (eCAALYX)

The three-year (Jun. 2009 to May 2012) enhanced Complete Ambient Assisted Living Experiment (eCAALYX) [36] was funded at about 4.118 million euros through the EU Ambient Assisted Living (AAL) joint programme. The project consortium consisted of 11 organisations across five European countries. Its main objective was to provide a practical, reliable, scalable, commercially viable, long-term solution for health monitoring of the elderly with multiple chronic diseases to improve the quality of life of the elderly and prevent deterioration of their health conditions.

According to [37], under this project, a remote server, a smart garment (with temperature, fall, and activity detection sensors) able to connect to a smartphone (with a customised App) and an intelligent sensor system (linked with ECG, SpO2, and blood pressure sensors through cables) able to connect to a home system (i.e. a set-to-box and a TV) both via Bluetooth were developed. During the project period, two phases of field trials with 10 participants were conducted. Though we had not found any published studies revealing a comprehensive evaluation of this project, barriers identified by [38], such as high costs, low usability and battery power inefficiency, towards the adoption of healthcare mobile App did serve as a checklist for our cost-effective design.

### 2.1.4.3 Integrated Network for Completely Assisted Senior Citizen’s Autonomy (inCASA)

The 39-month (ended in Jun. 2013) Integrated Network for Completely Assisted Senior Citizen’s Autonomy (inCASA) project [39] was funded by the European Commission under the ICT Policy Support Programme (or ICT PSP) at 2.14 million euros. The main objective was to provide elderly people with means to profile their habits and monitor their health conditions at home via an integrated automation system which enables remote control of devices, early decision making by health professionals for personalised care, and interactions between the elderly and carers.

A number of pilots were carried out across five European countries with 207 enrolled patients, aimed at developing integrated service models to achieve the project objectives. According to [40,41],
one of the main achievements of these pilots was a proof that the adoption of standards can facilitate better interoperability among system components, smooth integration of telehealth and telecare at data level, and rapid development of additional system functionality, and reduce costs.

2.1.4.4 Home Telehealth Services of the US Department of Veterans Affairs (VA)

According to [42], the home telehealth provided by VA is aimed at providing care and case management, including non-institutional care and chronic/acute care management, to veteran patients with chronic conditions, such as diabetes, chronic heart failure, COPD and post-traumatic stress disorder, in their home. With home telehealth, patients can take vital sign measurements and check on symptoms using special devices at home and be connected to a VA hospital through regular telephone lines, text messages and mobile phones. A care coordinator, usually a nurse or social worker, serves as the point of contact for patients and helps make clinic appointments and arrange hospital admissions.

A VA report [43] revealed that in 2013, there were 144,520 veterans enrolled in home telehealth services, of which 41,430 patients who would otherwise have needed long-term institutional care could live independently in their home based on the support of these services. It also concluded that in 2013 the use of home telehealth resulted in a 59% reduction in bed days and a 35% reduction in hospital admissions, as well as an annual saving of US$1,999 per patient. As almost half of the patients lived in rural areas, home telehealth played an important role in improving patients’ access to VA healthcare.

To further support the development of VA telehealth in the US, a Senate bill titled as the “Veterans E-Health and Telemedicine Support Act of 2015” [44] was proposed in Oct. 2015 to allow physicians to treat veterans across different states using telehealth without the need for them to be licensed in each state. In our opinion, to some extent, this indicates the difference of how telehealth could integrate into existing patient care pathways in the UK and US. In the former, community health services and primary care in local areas currently make up the main part of healthcare outside hospitals. [45] suggested that in England, community services, often led by community nurses, are central to meet a relatively high level of care needs and to enable early discharge from hospitals and prevent avoidable admissions, as well as to support the implementation of telehealthcare and self-care.

2.2 Smart Home and Internet of Things

2.2.1 Smart Home

The concept of the so-called “smart home” or “digital home” (used mainly in the Asia-Pacific
region) has been proposed for more than a decade, aiming to transform our home environment into an intelligence-embedded living space. Because there is no universal definition of these two terms at the moment, this thesis uses both terms alternately. According to the Department of Trade and Industry’s “Smart Homes Project”, smart home is defined as [46]:

“A dwelling incorporating a communications network that connects the key electrical appliances and services, and allows them to be remotely controlled, monitored or accessed.”

There were several industrial initiatives for smart home driven mainly by manufacturers and network providers. For instance, the Open Service Gateway Initiative (OSGi) Alliance founded in 1999 focuses on open specifications for remote management and the delivery of services into the home. At almost the same time, the Konnex Association was formed in 1999 to promote an open standard, called KNX, for home and building control. Other similar effort included the establishment of Universal Plug & Play (UPnP) Forum in 1999 for achieving seamless interconnection between two devices and the Digital Living Network Alliance (DLNA, originally named “Digital Home Working Group”) in 2003 for promoting the interoperability of multimedia devices at home.

Since their inception, these industrial initiatives have gradually expanded and gained wider support across different industrial sectors and players. For example, today PS2, XBOX 360 and personal computers with MS Windows 7 (and above) installed all support DLNA standards, and both OSGi and DLNA specifications support UPnP standards. However, there has been very limited support for integrating smartphone devices into these standards. Meanwhile, on the service side, the market has only developed to a very limited extent. For example, AT&T launched its “Remote Home Monitoring Video Service” in cooperation with Xanboo based on OSGi framework in Oct. 2006 [47], which combined live and recorded video (non-audio) capabilities with a range of sensor options for remote video monitoring, lighting control, and motion and window movement detection. It was then reported that just like other similar offerings that had failed largely at retail, this home services programme was shut down with a short life span [48]. One of the main reasons for the low adoption of smart home, as stated by [49], was that when more sensors are added to a smart home system, the system becomes complicated to handle and the maintenance becomes a challenge. In our opinion, this is also true for conventional home healthcare telemonitoring solutions.

Figure 2.3 shows a concept diagram of a typical smart home system. Generally, a home gateway or a control hub interconnects one or more home networks and the Internet/access network (sometimes a cable modem or router is also needed), and controls other in-home devices and sensors [50,51]. Today, there are many available portable sensor devices based on different networking or interfacing standards. Among others, ZigBee, BLE, and Arduino based are commonly used as less
expensive options for research purposes. More discussions about sensor technologies/platforms are presented in section 2.3.

Figure 2.3 The Concept Diagram of a Typical Smart Home System

For a commercialised smart home service package, the central server, i.e. one or a group of computers, is usually located at the service provider’s premise. However, in other cases it is common to see that the proposed system architectures require one or multiple servers (or called controllers) to be set up within the smart home environment, one for each platform that is being used by a controlled device [52]. From our viewpoint, this kind of design would increase the complexity of system installation and maintenance for general users. In 2014, Apple introduced its HomeKit Accessory Protocol [53], aiming to promote the integration of different home automation accessories and iOS devices and allow the user to control the accessories through mobile Apps using voice commands. Though currently the HomeKit framework focuses only on home automation, we think that this new development is line with our philosophy toward smartphone-centric technologies.

2.2.2 Internet of Things (IoT)

One important evolution of recent ICTs, which has great implications for the development of smart home, as well as home healthcare telemonitoring, is the emergence of the Internet of Things. The International Telecommunication Union (ITU) [54] described the IOT as a new form of communication between people and things, and between things themselves, which “connects the world’s objects in both a sensory and an intelligent manner.”

The basic architecture of the IOT consists of three layers: application layer, network layer and sensor layer [55,56], which can be naturally fitted into the concept framework of a digital home system as depicted previously in Figure 2.3. The sensor layer is mainly composed of sensors and actuators which collect context information around them and/or execute assigned tasks based on the
command from the application layer. The network layer provides a platform for transmitting data, information and commands among different modules or system components and serves as an intermediary between the sensing layer and the application layer. The application layer acts as a conductor for the entire system with a number of different functional modules, such as data retrieval, context reasoning, command dispatching, user interfaces, and system management.

According to the Cluster of European Research Projects on the Internet of Things (CERP-IoT), a large number of application domains in the field of IoT have been identified [57]. We believe that among others, intelligent buildings (automatic energy metering/ home automation/ wireless monitoring), healthcare, (personal area networks, monitoring of parameters, positioning, real time location systems), independent living (wellness, mobility, monitoring of an aging population), environment monitoring, and entertainment, are all applicable to supporting our envisioned smart home, as well as home healthcare telemonitoring. LinkSmart [58] is an open source middleware platform, originally developed within the EU Hydra project [59], functioning as a unified interface for heterogeneous devices to seamlessly interoperate with the other layers of an IoT application based on web services. A 2011 study [60] investigating the potential of IoT to monitor health and wellness concluded that the underlying technology is available but needs to be turned into a solution which can become pervasive in society. In light of this longstanding issue, this research intended to use low-cost, off-the-shelf smartphone-centric technologies to build up evidence for cost-effectiveness of a practical solution.

2.3 Sensor Technologies/Platforms

As mentioned previously in Section 2.2, sensors form an indispensable component of both a smart home system and an IoT system, as well as a healthcare telemonitoring system. In general, a sensor is capable of simultaneously detecting three but innately related categories of events [61]:

- Direct or proximal phenomena: events that directly trigger the sensor device;
- Indirect or distal phenomena: remote causes of the local events actually triggering the sensor; and
- Context and subtext: the situation surrounding an event.

However, contextual information inferred from both direct and indirect phenomena might still involve some degree of uncertainty. This highlights the importance of a well-designed event reasoning algorithm that can increase the accuracy of context inference based on a limited set of monitored data.
Today there are a great variety of electronic sensors available in the marketplace, such as accelerometers for acceleration measurements, gyroscopes for rotation, GPS (Global Positioning System) sensors for speed, distance and location, glucometers for blood glucose, pulse sensors for heartbeat rate, and sphygmomanometers for blood pressure. In the field of telehealthcare, there are also increasing focuses on the development of the so-called Body Sensor Networks (BSNs). Based on CPASIL’s roadmap, BSNs are described as a specific category of wireless sensor networks intended to operate in a pervasive manner for on-body applications. As such, special considerations are given to issues such as BSNs’ usability, durability and robustness, as well as how well the sensor fits in with the application and reliability and security of the data [62].

Currently most available off-the-shelf sensors ready to interoperate with smartphones do not fall into the domain of BSNs. For the purpose of our cost-effective design of home healthcare telemonitoring, this research pays special attention to existing inexpensive, portable and easy-to-use sensor technologies/platforms.

2.3.1 ZigBee

ZigBee is a standards-based low-power wireless technology mainly operating in the 2.4 GHz and sub-1 GHz (including 915 MHz for Americas, 868 MHz for Europe and 920 MHz for Japan) radio frequency bands. It is based on the IEEE 802.15.4 standard with add-on network and security layers and an application framework and has data rates of 250kbps, 40kbps, and 20kbps. Generally, the transmission distance is around 70 metres indoors (or up to 400 metres outdoors), and in general can cover a home-like environment. The ZigBee Alliance was established in 2002 to develop relevant specifications and to promote ZigBee standards adoption. Today, the ZigBee Alliance has about 450 members and over one-thousand certified products, ranging from home appliances, energy efficiency apparatuses, networking devices, to health and fitness sensors. ZigBee Health Care [63] was introduced with full support for Continua’s interoperability Design Guidelines [64] and IEEE 11073 [65] device specialization profiles to provide an industry-wide standard for exchanging data between a variety of medical and non-medical devices.

Based on different topologies, such as pair, star, mesh and cluster tree, a ZigBee sensor network consists of one coordinator node and at least one router or one end-device node [61]. In a ZigBee network, each node can communicate with all the others by way of its nearest neighbour so that only small amount of power is needed for radio transmission. In addition, with the embedded capability to

*8 “CPASIL” stands for International Support of a Common Awareness and Knowledge Platform for Studying and Enabling Independent Living. It was funded within the specific programme "Cooperation" and the research theme "ICT" of the 7th European Framework Programme.
perform self-healing, a ZigBee mesh network can reconfigure itself and route around problem area when one or more network nodes are failed or removed. In addition, a ZigBee network can be integrated with a local area network and hence is suitable for remote control applications. Other important features of the latest ZigBee 2012 specifications [66] include group broadcasts to simplify device management, data security based on Advanced Encryption Standard (AES) with a 128-bit key, low-power consumption for better battery life, and low cost in comparison with other wireless technologies.

In our opinion, one of the important merits of the ZigBee standard is the self-organising mesh networking capacity, which makes adding new ZigBee devices into an existing network or expanding the network coverage less complicated. Although ZigBee standards possess many advantages as mentioned above, which are deemed to be relevant to this research, at the time of this writing there are only a few kinds of sensors available in the health and fitness sub-category and ready to interoperate with smartphones.

2.3.2 Bluetooth Smart (Low Energy)

Bluetooth is a wireless technology standard created in 1994. It operates in the 2.4 GHz radio frequency band with a transmission distance of up to about 100 metres. Bluetooth is based on Frequency Hopping Spread Spectrum (FHSS), which is more susceptible to narrowband noise and interference, but can work better in severe multipath environments (such as indoors) than Direct Sequence Spread Spectrum (DSSS) [67]. (ZigBee is based on DSSS.) According to [68], in 2014, 90% of mobile phones were equipped with Bluetooth wireless technology. Bluetooth Smart (or Bluetooth Low Energy, BLE) technology (based on Bluetooth v4.0 or above) was introduced in 2010 with new features, such as extra-low power consumption, years of coin-cell battery life, and low cost. It supports very short data packets of up to 27 bytes based on AES-128 encryption and a transmission rate of 1 Mbps. In general, the transmission range is about 10 metres based on one-to-one or one-to-many (using star topology) connections. Though the Bluetooth Special Interest Group (SIG) formally announced [69] that the key technology updates in 2016 will include longer range, higher speeds and mesh networking capacity, currently Bluetooth Smart has a relatively smaller geographical coverage than ZigBee.

BLE devices are generally embedded with sensors, able to advertise their presence, and act as clients waiting to be connected by a Bluetooth Smart Ready device (i.e. a master). More technical details can be found at Bluetooth Developer Portal [70]. In our opinion, BLE and ZigBee share many similar features, whilst two important distinctions between them are the geographical coverage and their ability to from different network topologies. Since its inception, BLE has been gradually gaining attention from researchers and the industry. For example, in Nov. 2012 Texas Instruments (TI)
released the world’s first BLE development kit with on-board inertial sensors, named TI CC2541 SensorTag (hereafter abbreviated as TI SensorTag) [71], for smartphone applications. Then in Sept. 2013, Apple introduced the iBeacon [72], a Bluetooth Smart enabled proximity sensor/device, for iPhone-based geographical region monitoring. Meanwhile, the first BLE shield (served as a communications module) able to interconnect with the Arduino platform was also introduced in 2013. BLE-enabled medical devices in compliance with Continua interoperability Design Guidelines and the IEEE 11073 Personal Health Data family of standards also started to emerge.

For our prototyping, we considered that the ready-to-use development kit and the built-in capacity to interoperate with smartphones can facilitate our work. More detailed discussions on how we incorporated BLE devices to devise our proposed monitoring functions are elaborated in the following three chapters.

2.3.3 Arduino

Arduino is an open-source microcontroller platform for physical computing. It was originally designed in 2005 to provide students with an inexpensive microcontroller to drive their robotic projects. To date, it has evolved into a popular tool kit for prototyping and do-it-yourself work. By attaching different combinations of various sensors and actuators to a programmable microcontroller board, many different tasks, such as environmental (e.g. light level, temperature and humidity) monitoring and home automation (e.g. door/window opening and garden irrigation), can be performed in a way that is based on the user-uploaded software program. There are also a number of different communications modules, such as ZigBee, (classic) Bluetooth, BLE, serial port (e.g. USB), Ethernet, Wi-Fi, Global System for Mobile communications (GSM), and web server, available for use to receive inputs and transmit programmed outputs, such as the status of the board and the monitored data, to/from other devices, for example a laptop, a mobile phone or a web client.

According to [73], the main advantages of Arduino include: low-cost as compared with other microcontroller platforms, cross-platform, simple programming environment, and open source with extensible software and hardware modules. From our point of view, the capabilities both to conduct on-board data processing by the microcontroller to provide more reliable and meaningful monitored data, and to interconnect and interoperate with other wireless standards, such as ZigBee and BLE, are two other important features that enable Arduino and ZigBee/BLE devices to complement each other well to provide more flexible and seamless sensory solutions.

2.4 Cloud Computing and Mobile Cloud Computing

Though cloud computing has been often regarded as the Fifth Generation of Computing (after
Mainframe, Personal Computer, Client-Server Computing, and the Web), currently there is no unique definition of it. According to the NIST of the US Government, cloud computing is defined [74] as:

“A model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or cloud provider interaction.”

In general, there are three types of cloud service models, i.e. Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS), which all possess the essential characteristics of on-demand self-service, resource pooling, rapid elasticity, and pay-per-use. As such, both the private and public sectors have regarded cloud computing as an excellent enabler to create/provide better services and business models at lower costs and greater flexibility. [74] also classifies cloud deployment models into four categories: (1) public cloud is a cloud infrastructure operated by one or several organisations and is provided for the use of the general public; (2) private cloud is a cloud infrastructure generally operated and used by a single organisation; (3) community cloud is a cloud infrastructure operated by one or several organisations and provided for a certain community with shared concerns; and (4) hybrid cloud is a mixture of the above two or three types of cloud infrastructures.

2.4.1 Some Essential Building Blocks

The emergence of the cloud computing paradigm could be attributed to a number of different factors. For the purpose of this thesis, we consider that the technological advances in virtualisation and the implementation of load balancing and auto scaling to make the best use of cloud computing resources, among others, are essential for supporting our cost-effective design of cloud-based healthcare telemonitoring.

2.4.1.1 Virtualisation

Virtualisation is the technology that enables one physical machine or computing resource, such CPU, memory, and storage, to serve as multiple machines (called virtual machines, VM) or resources, which can then be shared by a number of users. There are several types of approaches to achieving virtualisation, such as full virtualisation and partial virtualisation (or called paravirtualisation) [75]. The hypervisor, or the so-called virtual machine manager (VMM), is a software tool that has been widely adopted to create the virtualisation environment. In general, hypervisors are classified into the following two types [75]:

- Type 1 or bare metal hypervisor: Type 1 hypervisors run directly on top of the hardware of a host machine with access and control of the physical resources. One or several guest
operating systems are running as processes on the hypervisor, but are completely decoupled from the underlying infrastructure by the hypervisor. In comparison with the other type of hypervisors, the main merit of this approach is better performance. Examples of Type 1 hypervisors include Xen, Microsoft Hyper-V, and Oracle VM Server.

- Type 2 or hosted hypervisor: Type 2 hypervisors run like an application on a host operating system, only through which they can access the physical resources. One or several guest operating systems are running within the hypervisor and hence degrade the performance. VMWare Workstation and Oracle VM Virtualbox are examples of Type 2 hypervisors.

It is the virtualisation technology that enables the creation of VMs and virtualised computing resources, as well as the sharing of them, makes cloud computing (built upon features, such as resources pooling and multi-tenancy) possible. For instance, both Amazon Elastic Compute Cloud (EC2) and Rackspace Cloud were first launched based on Xen hypervisors.

2.4.1.2 Load Balancing

Load balancing is the function to distribute workloads to numerous nodes of a kind in a system so as to make the best use of the resources and achieve the best system throughput. Here a node could be, for example, a physical machine, a VM, a hard disk, or an application instance. For cloud computing, [76] divided load balancing into two different levels: cloud provider level and cloud client level. In the case of the former, the nodes and workloads are represented by the physical machines and cloud client requests for virtual resources, respectively. In the latter, they refer to the virtual resources running the client’s applications and the user requests to be executed by the applications, respectively. For the purpose of this thesis, we only concern load balancing at the cloud client level.

In an IaaS platform, such as Amazon EC2, a load balancing service based on different criterion, for example geographical proximity and Round-robin, is provided on a subscription basis. For a PaaS platform, such as Google App Engine (GAE) and Microsoft Azure, a load balancing service is generally performed by the platform and is transparent to the cloud clients.

2.4.1.3 Auto Scaling

Scaling is the function to dynamically adapt a system to the changes of workloads so as to maintain required throughput or service quality. In general, there are two ways to scale a system, i.e. vertical and horizontal scaling. The former is performed by adding (scaling up) or removing (scaling down) compute resources, such as CPU, memory, and storage, to/from the existing nodes in the system, while the latter is done by adding more nodes (scaling out) or removing some nodes (scaling in) to/from the system. Here a node could be, for example, a VM or an application instance. For
cloud-based applications, auto scaling generally refers to automatically execute horizontal scaling when certain conditions predefined as a set of metrics, for example the average CPU utilisation across all VMs running the same applications, are met. This can help achieve better user experience and maximise the benefit of rapid elasticity and pay-per-use.

For IaaS cloud service provision, such as Amazon EC2, auto scaling is basically offered as a service based on subscription, allowing the cloud users to define their own criteria, i.e. a set of metrics, and the corresponding actions for scaling out and in. As for PaaS, some platforms, such as GAE, auto scaling is provided and controlled in a transparent manner by the service platform, in which cloud clients have very limited control of it, whilst some, such as Microsoft Azure, leaves the clients to define and manage their own auto scaling. Under these different service provision models, we believe that to select which cloud service platform/infrastructure would highly depend on the unique needs of each cloud client.

2.4.2 Public Cloud Service Providers

The public cloud service market is growing rapidly and the number of public cloud service providers, such as Amazon, Google, Microsoft Azure, Redhat OpenShift, OpenStack, Eucalyptus, and CloudStack, is increasing quickly. In the following subsections, we briefly illustrate the main services offered by two popular public cloud providers, including Amazon and Google.

2.4.2.1 Google App Engine

GAE, first introduced in 2008, is generally considered as a PaaS framework for implementing and deploying web services. This framework \[77\] incorporates sets of ready-to-use runtime environments, their related Application Programming Interfaces (APIs), Software Development Kits (SDKs), and a range of options for data storage. It supports several programming languages, such as Java, PHP, and Python. A web-based administration console is used for developers to create and manage their applications. The SDKs allow developers to build and test their applications on local computers and then upload them to GAE to start providing web-based services with a custom domain or using a free name on the “appspot.com” domain.

With GAE, an application does not need to deal with scaling and load balancing issues, as they are automatically dealt with by the framework when needed. Generally speaking, an application is running in a sandbox environment which the developers and application have no control of, but can use most of the abovementioned programming languages’ standard function calls and library features. Hence, the application is still given a certain level of flexibility, with exceptions to accessing the underlying operating system (such as writing to a file) or to opening a socket.
In Mar. 2014, Google announced its Managed Virtual Machine (VM) service, an IaaS-like service, allowing an application to run on a configurable hosting environment with more CPU and memory options. For load balancing, HTTP(s) traffic is distributed to different VM instances based on geographical proximity. As for auto scaling, the average CPU utilisation across all running instances is the only one metric that the cloud client can set to manage the scaling. However, at the time of this writing, the Managed VM service only supports Windows Server 2008 R2, Red Hat Enterprise Linux, and SUSE Linux Enterprise Server, and is still based on a Beta release without any support for SLA.

2.4.2.2 Amazon Elastic Compute Cloud

Amazon launched its Amazon Web Services (AWS) in 2006, a platform that provides a variety of web services, e.g. compute capacity, storage, networking, application hosting, and content delivery, with a set of instilled functionalities and tools, such as SDKs, APIs, and Management Console for application management, data management, security, and billing. Under the heading of AWS, EC2 [78] is an IaaS-based service which allows users to launch an instance from one of the 22 AWS-provided Amazon Machine Images (AMI) or the tens of thousands user-community-provided AMIs, each with a number of instance types [79] based on various combinations of compute resources, such as CPU, memory and storage, and operating systems (for example, Amazon Linux AMI 2015.09.1 (HVM), Red Hat Enterprise Linux version 7.1, Ubuntu Server 14.04 LTS (HVM), and Microsoft Windows Server 2012 R2). It also allows users to choose their pre-configured-and-saved machine images to start as many as needed server instances via AWS Management Console or remotely via web service APIs.

With EC2, on the one hand, developers can enjoy a high degree of flexibility to create and configure custom application environment, to install software and libraries as needed, and to develop, deploy, scale up/down or out/in, and interact with their applications, just like using physical machines. On the other hand, it needs a lot of expertise and time to fine-tune the running applications and environments, such as scale up and down, as well as to manage the running resources. Though Amazon never discloses detailed figures on its AWS revenues, it is universally believed that currently Amazon is the biggest public cloud service provider in the marketplace around the world with a very big gap to the second one.

2.4.3 Economic Implications

According to a small-scale study done by IDC (and sponsored by Amazon) in early 2012 [80] on the long-term economic implications of moving workload onto Amazon cloud infrastructure services, some of the main findings include:
Five-year return on investment (ROI): 626%

Payback period: 7.1 months

Downtime reduction: 72%

IT staff productivity increase: 52%

Reduction in capital expenditure and operational expenditure per application per year: US$276,000

These results are consistent with the general perception of the merits of cloud computing. In fact, the benefits of the underlying economics of cloud services, as stated by [81], are realised through the following three areas:

- Supply-side savings: economies of scale in datacentres can enable the public cloud service providers to lower their costs of energy usage, labours, infrastructures, and operations.

- Demand-side aggregation: Through virtualisation which enables the sharing of physical compute resources among a number of cloud clients, better utilisation of the resources can be achieved and therefore drives down the costs.

- Multi-tenancy efficiency: An application can be reengineered to enable itself to serve multiple users in a multi-tenancy environment, rather than one user in an on-premises environment. Hence, the costs of the application, as well as its maintenance, can be shared by a number of users.

However, we believe that migrating an application from an on-premises environment to the cloud does not necessarily guarantee the achievement of these benefits. More details about how we leveraged cloud computing to achieve cost-effective design of home healthcare telemonitoring are given in chapter 7.

2.4.4 Mobile Cloud Computing

According to [82], “as of February 2010, there were 5,805 health, medical, and fitness Apps in the Apple AppStore. Of these, 73% were intended for use by consumer or patient end-users, while 27% were targeted to health care professionals.” The number of health-related mobile Apps then quickly increased to more than 100 thousands in the first quarter of 2014 [83]. There are also Apps using available sensors, including accelerometers, infrared photo-detectors and glucometers, for home monitoring. However, by analysing 656 Apps targeting patients with diabetes, [84] found that most of them required manual input of the recorded data, and that only 2.5% of iOS-based Apps and 5.3% of Android-based Apps were able to connect with an external sensor/device. In our view, all these
figures and developments represent both a gap and an opportunity, in terms of insufficient functionality of the Apps and very large demands for healthcare applications, for researchers.

Meanwhile, mobile cloud computing have also gained increasing attention from ICT researchers and developers. In broad terms, there are two different viewpoints [85] regarding mobile cloud computing. One refers mobile cloud computing as making use of cloud resources, such as computing power and storage, to help perform tasks or store data from mobile devices, which generally have limited computing capacity and data storage. The other recommends that with mobile cloud computing, both data processing and storing be done by the mobile device. For the purpose of this thesis, we take both views to give a broader definition of mobile cloud computing. With this, it is apparent that by adopting mobile cloud computing, an application can be further empowered by mobility together with the main advantages, such as on-demand self-service, resource pooling, rapid elasticity, and pay-per-use, derived from cloud computing.

According to [86], the architecture of mobile cloud computing can be divided into two types: Agent-client scheme and Collaborated scheme. In the former scheme, the overall resource management for mobile devices is performed by the cloud side, which generates one agent for each device to communicate with other entities for the coordination of cloud resources. The latter regards each mobile device as a part of the cloud. Hence several devices can work together to perform a particular task or each device can provide its remaining resources for other devices.

However, as currently almost all public cloud services are web-based, it is not surprising that an increasing numbers of research work adopted web-based technologies to design and implement mobile cloud computing. Among others, the REST (REpresentational State Transfer) framework has been considering as one promising technology in this field. A study [87] on the comparison between SOAP-based (SOAP stands for Simple Object Access Protocol) and REST-based frameworks on performance, scalability, reliability and resource consumption within the mobile hosts concluded that a REST-based mobile web services framework is more suitable for mobile environments. Besides, we considered that by using REST-based services framework, we could decouple mobile devices from the cloud platform and achieve higher flexibility for the system architecture. Consequently, we decided to adopt the REST-based web services for the design of our mobile cloud-based telemonitoring solution.

2.4.5 Related Work on Cloud-based/Mobile Cloud-based Healthcare

To present a general overview of the trend toward (mobile) cloud-based healthcare, this subsection presents some related work as follows. [88] developed a proof-of-concept cloud-based system to collect patients’ real-time monitored data and to store and distribute the data. In this design,
a local exchange server (an application running in a local computer) was needed to actually collect the data and send/retrieve the data to/from the cloud storage. [89] is a cloud-based prototype emergency medical system. It integrated the emergency system with personal health record (PHR) systems, aimed at facilitating instant access to a person’s medical information via a web portal in an emergency situation. From our point of view, these two prototype systems simply used cloud-based storage for storing and retrieving data.

MoCAsh [90] was a proposed mobile cloud for assistive healthcare. The proposed system architecture included: (1) sensors and mobile agents, (2) a context-aware middleware, (3) a collaborative cloud, and (4) a cloud portal. In our view, this project could serve as a good reference for our prototyping and design. However, at the time of this writing, no information about the implement of non-built-in sensors and patient monitoring is available.

HealthVault [91] is a cloud-based platform launched in 2007 by Microsoft, allowing individuals to create and store their health records and control who (such as their family members and healthcare providers) can have access to their data via a web portal. HealthVault also provides an XML-over-HTTP interface and SDKs for web-based applications to interoperate with it. Although Microsoft is committed to making long-term investment and continuous improvement on this platform, currently, we can generally consider HealthVault as a cloud-based storage built upon web service technologies for storing and managing PHR data.
3. Research Design

3.1 System Requirements

3.1.1 Functional Requirements

The primary set of functional requirements used to model a real-time home healthcare telemonitoring system is enumerated below. Based on our research scope and limitations, some requirements that were not incorporated in this thesis are indicated.

– **Vital sign monitoring:** This refers to the monitoring of patients’ vital sign parameters, such as body temperature, heartbeat rate, blood oxygen saturation, blood pressure, blood glucose, cardiogram, and sweat level. This functionality basically works on an on-demand basis, because some available sensors for this purpose cannot be worn permanently or cannot function automatically, e.g. the glucometer. When the measured vital signs exceed predefined thresholds, an alert is automatically sent to designated carers, healthcare professionals, and the telemonitor centre (handled by emergency tele-consultants) via text message, video phone, web applications and web-based communication, such as email. Subject to time availability and technical constraints, only the last two methods were implemented in this research.

– **Safety monitoring:** Its main function is real-time human fall detection with alerts being automatically sent to designated carers, healthcare professionals, and the telemonitor centre via the same methods as used for vital sign monitoring. Once switched on, this function is performed automatically and round the clock. A proof-of-concept fall detection system together with the experiment results and our evaluations is presented in chapter 4.

– **Movement Pattern monitoring:** This concerns mainly the tracking of the real-time movement of the patient at home with data on his/her real-time location being automatically sent to the telemonitor centre. A proof-of-concept indoor location tracking system together with the experiment results and our evaluations is detailed in chapter 5.

– **Emergency call-for-help toolkit:** This refers to the provision of a portable alarm; once pressed by a patient, it would send out an alert to designated cares, healthcare professionals and the telemonitor centre. As the design and implementation of this toolkit involve mainly technical, rather than research, issues, this research did not fulfill this function.

– **Service portal, management console and healthcare dashboard:** The service portal and management console allows authenticated patients, carers, healthcare professionals, and telemonitor centre personnel to set their preferences and care plans for healthcare
monitoring and to review/update/comment on personal health data and/or diagnosis via a healthcare dashboard. A proof-of-concept web-based system, which was implemented at the early stage of this research, is not included in this thesis, because it is outside the scope of this research.

- **Authenticated database management and access:** The large volume of monitored data requires a well-designed database system that can provide authenticated users with management and access functions and can seamlessly scale out in response to increased data. A proof-of-concept MySQL database, which was implemented at the early stage of this research, is not included in this thesis, because it is outside the scope of this research. However, a database server was modelled in chapter 7 for our cloud simulations to generate performance-related data for conducting cost-effectiveness analysis.

- **Home telemonitoring interventions and a referral mechanism:** In addition to vital sign, safety and movement pattern monitoring, the proposed interventions also included prescheduled home visits, prescheduled and emergency tele-consultation phone calls, and direct referral to GPs or A&E when appropriate. A proof-of-concept simulation environment was modelled and implemented to examine the allocation and utilisation of relevant healthcare resources. Nevertheless, for the purpose of this thesis, only emergency tele-consultation calls in response to alerts for falls and vital sign abnormalities and, when necessary, a follow-up referral to GPs or A&E are presented and discussed in chapters 6 and 7.

### 3.1.2 Non-functional Requirements

For the design of a real-time home healthcare telemonitoring system, we considered that the following non-functional requirements are important and hence need to be addressed throughout the whole system development life cycle. These include requirements that are outside the scope of this research.

- **Low costs** (from both a patient and public healthcare provider point of view): Cost is one of the major factors that have direct impact on the cost-effective design of our proposed telemonitoring. Therefore, there should be no significant amount of capital and operational expenditure on system setup and daily operations. Regarding the equipment used at home, in addition to a smartphone and a wireless broadband router, the end-user (i.e. the monitored patient) should only need some off-the-shelf, low-cost sensor devices. In terms of the web application and database servers, one way to drive down the costs is to make use of a free-of-charge service offer, e.g. HealthVault. However, for the purpose of this research, we chose to model the use of public cloud services so as to explore ways to leverage the
features of elasticity and pay-per-use in cloud computing to reduce the overall infrastructure costs.

- **High usability** (from a patient perspective): The client-side system should be easy to use and deploy. In general, the patients, especially those living independently, should be able to set up the hardware connections (by either wired or wireless means) for sensor devices and to switch on/off the monitoring services. Hence, the system should be user-centric. In addition, as we focus on smartphone-centric technologies, the design should consider the limitation on the smartphone battery life and the flexibility of placing the phone when performing telemonitoring. However, the usability of the mobile App user interface, service portal, management console and healthcare dashboard is outside the scope of this research.

- **Less intrusion** (from a patient perspective): Generally, neither the operation of telemonitoring nor the adopted sensor devices should hinder the patient’s normal daily routine. Mechanisms that are deemed to be privacy intrusive, such as those using round-the-clock video cameras throughout the home to perform fall detection or movement monitoring, should be avoided.

- **High performance and elasticity** (from both a patient and public healthcare provider point of view): System performance needs to be good enough to provide a streamlined end-user experience and to meet the requirements for timely alert dissemination. In addition, the system should be able to scale out and in on the cloud side based on the workloads at a minimal cost and to interwork with a dynamic number of sensor devices based on different home settings or care plans. Only the mechanisms for auto scaling and load balancing are within the scope of this research and are presented in section 7.3.5.2.

- **High Reliability and Accuracy** (from both a patient and public healthcare provider perspective): To ensure that false alerts for the fall detection and vital sign abnormalities, and adverse impacts on performing correct diagnoses are kept to minimal levels, both high reliability and accuracy of the monitor data should be achieved. For vital sign monitoring, due to the unavailability of clinically certified, low-cost vital sign sensors able to interconnect with a smartphone, we excluded the accuracy of our vital sign monitoring system from the research scope.

In contrast, this research aimed to focus on comparable accuracy only, in comparison with related work, for both fall detection and real-time location tracking. More specifically, we considered that the sensitivity of our fall detection mechanism should be higher than 90% and the maximum estimation error of location tracking should be less than two metres at a 95% confidence level to ensure their usefulness.
- **High Robustness** (from both a patient and public healthcare provider point of view): The system should embrace fault-tolerant and resilient design to maximise service availability. On the client side, the mobile App, as well as the monitoring task, should be able to continuously function properly when the Internet is not available or the cloud side is unreachable. Once the problem has been resolved, previously monitored data should be automatically sent to the cloud side database.

With regard to the transmission of event alerts, multiple channels, such as telephone networks and Internet, should be used to ensure alerts are received in a timely manner by the designated people. On the cloud side, the system can make use of auto scaling to increase availability. However, high availability and disaster recovery though VM redundancy and database replica across multiple geographical regions, as well as the support for a distributed telemonitor centre and a robust alert dissemination system, are outside the scope of this research.

- **High security and data privacy** (from both a patient and public healthcare provider perspective): The system should employ secure access control, user authentication, and data encryption and transmission to enhance data privacy and security, as well as maintain patient trust and confidence. These mechanisms not only help meet regulatory requirements, but also encourage people to opt for the adoption of our solution. In light of the challenge of integrating security and data privacy into a healthcare system, which is a research question in its own right and is outside the scope of this research, we generally followed best practice during the implementation of our prototype system.

- **Less vendor lock-in** (from both a patient and public healthcare provider point of view): The system design should minimise the impact of the vendor lock-in issue by taking into account portability at the system and data levels. As mentioned in section 2.4.4, we considered that the REST-based services framework, which can decouple mobile devices from the cloud platform and enable greater flexibility for the design of system architecture and the implementation of the server systems, is able to help enhance system portability. Meanwhile, we believed that a database system based on an open standard can help ensure data portability. However, this requirement is outside the scope of this research.

### 3.2 System Design

#### 3.2.1 High-level System Architecture

The proposed system architecture for the home healthcare telemonitoring system consists of four
main modules: User Agent(s), Service Gateway (Cloud Broker), App&DB, Sensor Nodes. Figure 3.1 shows the high-level architecture diagram. The main functionality of each module is illustrated below:

![High-level System Architecture Diagram]

**Figure 3.1 High-level System Architecture**

(i) **The User Agent(s) module**: As we focused on smartphone-centric technologies, we developed a native App based on an iPhone 5 with iOS version 6.1.3 to fulfill this module. The main functions of this module include:

- **A portable personal healthcare assistant** that allows users to control and manage a variety of sensors, including external sensing devices and built-in sensors, set their preferences and care plans for healthcare monitoring and manage personal health data. It can work with and without an Internet connection. A local database is used to store recent relevant data to provide offline access and improve system performance. When the Internet is available, data synchronisation between the local and cloud databases will be performed automatically.

- **An intelligent monitoring agent** that interoperates with the sensors, collects real-time monitor data, uploads them to the cloud database on the App&DB module using web services APIs, and performs context/health data reasoning based on predefined parameters (such as vital sign thresholds) and algorithms (such as fall detection and real-time location tracking). This automatically triggers an alert when certain conditions are met, to designated
carers, healthcare professionals and telemonitor centre, via multiple communications channels.

(ii) The Service Gateway module: This module acts as an intermediary between the User Agent and the App&DB modules based on web service APIs to increase the flexibility and portability of the latter across different cloud platforms. Its main functions include:

- **A cloud manager/broker** that serves as an Internet-facing cloud entry point, maintains lists of available cloud resources, performs protocol translations for requests and responses between the User Agent(s) and the App&DB, and re-directs traffic to different cloud platforms based on user preferences or its own criteria. For cost-effectiveness analysis, we modelled this module to perform simulation trials, in which only one public cloud platform was emulated. Accordingly, we neither implement nor model these functions.

- **A load balancer and a scaler** that distribute requests or workloads to different cloud platforms, web applications or VMs, and dynamically perform on-demand scale-out/in either across different service regions of one or several cloud service providers or within one service region of one cloud service provider based on predefined criteria. The modelling and simulation of an intra-region load balancer and an auto scaler are presented in section 7.3.

(iii) The App&DB module: This module generally consists of one logical web application server and one logical database server. With the help of the Service Gateway module, both the logical web application and database servers can be composed of multiple heterogeneous/homogeneous servers across different cloud platforms. For the purpose of this thesis, we simplified this module to have one web application server and one database server, as illustrated below, running on a number of VMs provided by one public service provider.

- **A web application server** that provides both a service portal and a management console, as well as a dashboard, for all the authenticated end-users, including patients, carers, healthcare professionals, and telemonitor centre personnel, such as tele-consultants, to access, review, and update relevant data stored in the database and to manage the monitoring. It also serves as a networking platform for all end-users to interact among themselves. The modelling and simulation of a web application server based on our telemonitoring scenarios are presented in section 7.3.

- **A database server** that stores all relevant data for the telemonitoring system, such as the authentication data of all end-users, the care plans and personal health records of all telemonitored patients, the diagnosis and comments made by healthcare professionals on patients’ monitor data, management data on sensor deployment and status, and monitor data,
such as vital signs, fall records, movement pattern records. The modelling and simulation of a web database server based on our telemonitoring scenarios are presented in section 7.3.

(iv) **Sensor Nodes:** This module is composed of a number of off-the-shelf portable sensor devices and the smartphone built-in sensors to collect/produce sensory data for telemonitoring. As discussed in section 2.3, at the time of this writing, this research deems that Arduino-compatible sensors and BLE sensors, among others, can better fit its requirements. With an on-board microprocessor, the former, which are able to produce structured and encrypted data for the User Agent module, were adopted for our vital sign monitoring, whilst the latter, which generally produce raw data, such as acceleration and received wireless signal strengths, were adopted for safety (chapter 4) and movement pattern monitoring (chapter 5).

### 3.2.2 Design of the User Agent

![Component Diagram of the User Agent Module](image)

The design of the User Agent mainly involved the development of a mobile App to realise the three monitoring functions, including vital sign, safety, and movement pattern monitoring (as mentioned in 3.1.1) of our proposed telehealthcare system on the client side. Figure 3.2 shows the component diagram of the User Agent module. Each component was implemented via one or a group of classes using Objective-C to fulfill its particular functions. Each arrowed line represents the interface between two related components. Two third-party open-source software frameworks, including ‘FMDB’ for SQLite database operations and ‘SKPSMTPMessage’ for sending alert emails, were adopted to speed up the implementation of the ‘Database Management’ and ‘Web
Communications’ components, respectively. Nevertheless, for the purpose of this thesis, we do not intend to go into the details of implementing these components. Instead, the emphases of this thesis are on what research problems we identified regarding safety and movement pattern monitoring and how we developed the desired solutions.

At the time of this writing, the smart home initiatives mentioned in section 2.2.1 have very limited support for smartphones. In particular, we have not found any off-the-shelf sensor devices ready to interoperate with smartphones based on the related smart home standards. Besides, our earlier survey suggested that there had not been available clinically certified, integrated, low-cost vital sign sensors, able to interconnect with iPhone 5 and measure all the major vital signs. Consequently, we adopted clinically uncertified Arduino-based eHealth Sensor Platform v1.0\(^9\), consisting of eight sensors, to develop a proof-of-concept vital sign monitoring subsystem. Figure 3.3 shows the photo of our vital sign monitoring test for measuring heart beat rate and blood oxygen saturation, using an iPhone 5, a Redpark TTL cable\(^{10}\), an Arduino UNO R3 board and the eHealth Sensor Platform. On the right of the picture is a screenshot of the App, allowing a user to choose a sensor to perform a

\[^9\] The eHealth Sensor Platform was manufactured by cooking hacks. (http://www.cooking-hacks.com) at a retail price of €447.7 (or £335.78 based on an exchange rate of one euro to £0.75 pounds sterling).

\[^{10}\] The TTL cable for iOS device was produced by Redpark (http://redpark.com) at a price of €66.88 or £50.16.
desired measurement. Since the design of vital sign monitoring was mainly a technical problem, further details of the design are not presented in this thesis.

With regard to the safety (for fall detection) and movement pattern (for real-time indoor location tracking) monitoring, our literature review suggested that in both topics a number of research problems, such as accuracy and usability, have remained unsolved. Consequently, we conducted substantial research work to investigate and identify major issues in related work and to develop solutions to resolve them. The details of this research on developing these two monitoring functions together with cost-effectiveness analysis based on the results of our experiments are elaborated in chapters 4 and 5, respectively.

3.2.3 Design and Simulation of Home Telemonitoring Interventions

To be able to conduct cost-effectiveness analysis of the proposed telemonitoring solutions, a number of healthcare interventions, such as prescheduled home visits, prescheduled and emergency tele-consultation phone calls (as mentioned in section 3.1.1) were proposed. Among all, we considered that both the vital sign and safety monitoring exercises followed by an emergency tele-consultation call in response to an alert sent to the telemonitor centre for either vital sign abnormalities or a detected fall are most relevant to this research. Through the tele-consultation call, the tele-consultant would evaluate the physical conditions of the patient associated with the alert and then refer him/her to a GP or A&E when necessary. Another important intervention is real-time movement pattern monitoring together with the evaluation of patient movement data conducted offline by healthcare professionals.

However, in light of the limited resources, time and funding in particular, it was impractical for this research to design and implement a randomised controlled trial by recruiting a large number of patients and related stakeholders to join/use our proposed home healthcare telemonitoring interventions/system. It was also outside the scope of this research to re-engineer existing patient care pathways, as well as business models, to incorporate telehealthcare. Consequently, we believed that carefully designed simulations could create real-world like healthcare scenarios based on our proposed telemonitoring interventions and modelling and hence produce required data for further evaluation. Our simulation strategy was devised as follows:

(i) Generate a large number of patients with chronic diseases and randomly divide them into a control group (without receiving telemonitoring) and an intervention group (with telemonitoring). This provides a macro view of the proposed system, in terms of economic and functional effects.
(ii) Model each patient's activity of daily life, such as taking vital sign measurements and having a fall based on certain user scenarios, care plans and statistical distributions.

(iii) Simulate how emergency tele-consultants respond to alerts and make tele-consultation calls so as to examine the relationship between the allocation and utilization of healthcare resources.

(iv) Model the utilisation of healthcare services, including GP, A&E, and hospital treatments, by both groups in relation to falls and vital sign abnormalities based on certain statistical distributions and assumptions. Although (as mentioned in section 2.1.2) telehealthcare might well have a great potential for reducing patients' length of stay in hospitals and hence help achieve significant savings for public healthcare, we did not model the impact of telehealthcare on patients’ length of stay due to a lack of related clinical data.

(v) Simulate the real-time transmission of monitor data and alerts generated by the User Agents owned by patients in the intervention group.

(vi) Model how relevant stakeholders, including carers, healthcare professionals and telemonitor centre personnel, interact with patients in the intervention group through the telemonitoring system/interventions.

Detailed descriptions of the modelling work, simulations and cost-effectiveness analysis based on the results of our experiments are presented in chapter 6.

3.2.4 Design and Simulation of the Service Gateway and App&DB Modules

As mentioned in section 3.1.1, at the early stage of this research, we developed a proof-of-concept web application, based on the ‘Ruby on Rails’ framework using REST-based web services and MySQL database, to explore possible issues on the design of related functions. These issues included transmitting real-time monitor data to a remote web application server, displaying them on the healthcare dashboard, and conducting user authentication for database access. By doing this, we identified that for the purpose of this research, the main challenges in designing these two modules are how to enhance system performance (especially response time) and scalability at limited costs when a large amount of workloads are simultaneously, continuously and dynamically generated by the monitored patients and related stakeholders.

However, due to lack of resources for conducting a randomised controlled trial, we once again opted for using simulations to emulate the Service Gateway and App&DB modules and to conduct
experiments on workload executions so as to produce relevant cost and effect data to examine our research hypothesis. Our simulation strategy was as follows:

(i) Conduct a survey on existing cloud simulation toolkits and choose one from them based on its functionality and our simulation requirements.

(ii) Customise the chosen cloud simulation toolkit to fit our specific simulation needs.

(iii) Choose a cloud platform which can better meet our system requirements.

(iv) Model the web application and database servers and workloads based on benchmarking experiments on the chosen cloud platform.

(v) Develop auto scaling and load balancing mechanisms to leverage the benefits of scalability and pay-per-use in cloud computing.

(vi) Examine how cloud components (i.e. the App&DB module) effectively scaled in response to the dynamical changes of workloads via simulations.

All these modelling and simulation designs together with the cost-effectiveness analysis based on the results of our experiments on large-scale workload executions are given in chapter 7.

3.3 Comparative Cost-effectiveness Analyses

In light of the abovementioned limitations on resources, this research did not calculate CER or ICER directly, but conducted cost-effectiveness analysis based on a revised comparative effectiveness analysis approach. Based on this approach, this research conducted simulated trials to predict the costs and effectiveness of the proposed system and compared costs and functional effectiveness, usability, or performance levels of the proposed monitoring subsystems and related work. (Since the design of business models was outside the scope of this research, the costs of system components were estimated mainly based on their retail prices, rather than a pre-defined case-specific business model that drives the implementation, operation and delivery of the proposed system.) This was supported by [92], which states that due to the complexity of trying to forecast the main outcome of an intervention and the need for large sample sizes to detect statistically significant differences, economists have been focusing on estimating differences in costs and effects and assessing whether an intervention is cost-effective. For the purpose of this thesis, four types of comparisons for building up evidence of cost-effectiveness are illustrated in the following.
3.3.1 Comparisons of healthcare resource utilisation between simulated patients in different groups

As mentioned in section 3.2.3, this research used simulations to generate the needed patients with chronic diseases and some baseline characteristics, and to randomise them to form a control group and an intervention group to enable a simulated randomised controlled trial. Additionally, these simulations enabled us to predict patients’ activities of daily living and their behaviours towards the use of healthcare services, such as GP and A&E, after a fall or vital sign abnormalities with and without telemonitoring interventions.

We then conducted cost-effectiveness analysis by examining in particular how telemonitoring helped patients in the intervention group avoid either missing the required medical treatments or undertaking unnecessary healthcare attendances in comparison with patients in the control group to achieve cost savings in public healthcare. We finally claimed our proposed system is cost-effective, as the conclusion was supported by the comparison results.

3.3.2 Comparisons of costs and functional effectiveness with other telemonitoring systems

This was conducted by benchmarking the costs and functional effectiveness (or functionality), usability, or performance levels against other existing telemonitoring systems, such as the WSD trial and some work on fall detection and indoor location tracking. The results were used not only for conducting cost-effectiveness comparisons, but also for improving the design and implementation of the proposed system. The key steps were:

(i) Use literature review to obtain data about the costs and functional effectiveness, usability, or performance of other existing telemonitoring systems.

(ii) Compare the costs and functional effectiveness between the WSD and our proposed system. Since there were no significant cost differences in most work on fall detection and indoor location tracking using smartphone-based solutions, we did not compare the costs, but the functional effectiveness, usability and accuracy, of our proposed fall detection and indoor location tracking solutions and other work.

(iii) Improve or refine our design and implementation before repeating step (ii) when necessary, or move to step (iv).

(iv) Claim our proposed system is cost-effective when functional effectiveness (or functionality), usability, or performance levels are similar but less costly or when functional effectiveness
(or functionality), usability, or performance levels are better at similar costs.

3.3.3 Comparisons of costs and functionality with usual care

The main purpose of this type of comparison is to understand the extent to which the proposed system can substitute usual care for out-patients or elderly at home. By examining what benefits the proposed system can offer, such as savings in human labour time and costs, as well as added functionality, we can also conduct cost-effectiveness analysis. Due to time limitation, we planned to perform these comparisons in the future work. The main process of the proposed comparisons includes the following steps:

(i) Conduct literature review to obtain data about the costs and care plans of two to three usual care services formerly given to out-patients suffering from various chronic diseases or elderly at home in a particular region/country.

(ii) Examine what work or extra functionality under those care plans can be provided by the proposed system at similar or better quality/performance levels.

(iii) Estimate the costs of using the proposed system to conduct the work identified in step (ii) based on simulations.

(iv) Compare the costs and benefits between usual cares and the proposed system;

(v) Adjust or add more functions to our proposed system and repeat step (ii) to (iv) when necessary, or move to step (vi).

(vi) Claim the proposed system is cost-effective, if there are cost savings and/or added benefits associated with it.

3.3.4 Comparisons of costs and health effects with existing randomised controlled trials

This type of comparison is aimed to fulfill a more formal cost-effectiveness analysis based on ICER, but using simulations to estimate the health effects on patients receiving our proposed telemonitoring interventions. However, to do this, more work is needed in the future to model the health effects of our telemonitoring system using data, in relation to the effects on the quality and quantity of life of patients produced by telemonitoring interventions, from medical literature and/or randomised controlled trials. The concept of this comparative analysis is depicted in Figure 3.4 and the procedure is illustrated below:
Figure 3.4 Comparative Effectiveness Analysis Based on Simulations

(i) Use literature review to obtain data about the introduction of a known Intervention Y, its Cost Y, and the resulting changes in a group of patients’ health status from Health Status X (with Quality of Life X and Life Expectancy X) to Health Status Y (with Quality of Life Y and Life Expectancy Y).

(ii) Conduct functional/performance comparison (functional/performance items and criteria for comparison need to be further designed) between the proposed Intervention Z (based on experiments and simulations) and Intervention Y or other related work.

(iii) Model the health effects of using the proposed Intervention Z based on data from Intervention Y or other related work. For example, we can claim that the proposed Intervention Z is able to provide the same QALY effects (i.e. Health Status Y with Quality of Life Y and Life Expectancy Y) or better QALY effects (i.e. Health Status Y+ with Quality of Life Y+ and Life Expectancy Y+) than the compared, if Intervention Z has the same or better functionality/performance.

(iv) Estimate the costs and health effects of applying the proposed Intervention Z through simulations.

(v) Calculate ICER.
(vi) Compare the achieved ICER with international/country-specific cost-effectiveness thresholds, such as the lower boundary of approximately £20,000/QALY to the upper boundary of approximately £30,000/QALY used by NICE [93].

(vii) Claim that the proposed system is cost-effective, for example if the achieved ICER is lower than the upper boundary used by NICE.
4. Real-time Human Fall Detection

4.1 Introduction

Falls among older adults have been widely recognised as one of the leading causes for physical injuries and A&E attendances, resulting in significant loss of confidence and independence of living. Although there have been an increasing number of studies on smartphone-based fall detection in recent years, a number of issues, such as the need for fixed placement of the phone or sensors, and over draining of battery power, generally remain unsolved. The main purpose of this chapter is therefore to present our proposed fall detection solution for safety monitoring which does not only address these two issues, but also possesses the merits of less intrusiveness, easy deployment, and high accuracy. Our goal in developing this solution was to successfully detect a serious fall event and automatically alert the telemonitor centre and carers, but not to further identify what kind of the fall it is, for instance, a forward fall or backward fall.

A preliminary version [11] of this work was reported at HealthInf 2014. This chapter expands significantly on that presentation in both technical details and extent of the evaluation. One thing to note is that in both [11] and this chapter the technical design, execution of experiments, and evaluations were the work of the author of this thesis.

To build our proposed system, we adopted acceleration parameters with carefully defined thresholds for the construction of a fall detection algorithm, and we chose an external low-cost TI SensorTag, which has on-board inertial sensors, to provide the acceleration measurements with the intention to overcome the two aforementioned issues. Based on the results from 22 trials, the sensitivity and specificity of our system for successfully detected falls and recognised non-fall activities were both 95.5%, and specificity for effectively identified SensorTag drops or throws was 100%. The main contribution of this work is the realisation of a proof-of-concept system that demonstrates a low-cost, easy-to-use, less intrusive, but feasible and robust solution for fall detection.

The remainder of this chapter is structured as follows. Section 4.2 presents related studies on smartphone-based fall detection by highlighting the main approaches and key issues in their design. Section 4.3 illustrates the design and implementation of our proposed system in detail, including the systematic approach we took to resolve the identified issues. Section 4.4 provides an assessment of costs and effectiveness based on the experimental settings and results, and summarises our main findings. Finally, section 4.5 offers a review of the results and discusses the limitations of our work together with suggestions for future work.
4.2 Literature Review and Related Work

[94] divided fall detection systems based on their adopted methods into three main categories: wearable device based, ambience sensor (such as acoustic and infrared sensors) based, and camera based, whilst [95] divided them into context-aware systems and wearable devices. For the purpose of this thesis, we limited our discussion on the wearable device based systems with special emphases on smartphone-based fall detection. According to [95], the number of studies on smartphone-based has been increasing recently, but the low usability of their solutions due to the limitation on the placement of the device has remained one of biggest challenges. They also concluded that the performance of smartphone-based fall detection systems has been significantly hindered by several identified problems, such as the depletion of battery power for performing continuous sensing and the instability of the accelerometer’s sampling frequency, and that the variations in smartphone hardware models and operating systems might cause the same fall detection mechanism to function differently.

[96] conducted a systematic review of 51 articles on smartphone-based fall detection and reached similar conclusions. They found that most of the studies demanded a fixed placement of the sensing device (which is either the smartphone itself or a number of external sensing units) and thus imposed a usability constraint on the system, and suggested that using external sensing and computing units can minimize the computational load on the smartphone and reduce its battery consumption. According to this review, the accelerometer was used by all smartphone-based fall detection solutions, whilst the GPS receiver was used by 42% of them and the gyroscope was the third commonly used sensor. With regard to the classification method, most of the studies adopted threshold-based algorithms with a few exceptions using machine learning algorithms. Nevertheless, less than half of these studies reported on the performance or accuracy of their solutions.

The above studies provided us with a broader overview of the development trend, as well as the constraints, of the existing work. In the following subsections, the designs of sensing and classification mechanisms done by a number of related studies are presented.

4.2.1 Basic Settings for the Sensing Mechanisms

The work of [97-100] were all based on the built-in sensors on smartphones. [97] proposed to place the phone in the user’s trouser pocket and to use both the accelerometer and compass to detect the user’s posture of motion activities. The sampling rate for the accelerometer was 150 Hz. [98] used Lenovo Le-phone with Android operating system, which was worn on the user’s waist for fall detection with a sampling rate of 40 Hz for the accelerometer. They concluded that the waist-mounted solution has a problem of feasibility. [99] used Google Nexus S with Android operating system, which was placed in the user’s trouser pocket, for fall detection experiments. The sample rate for the
accelerometer was 50 Hz. [100] used Google Nexus One with Android operating system, which could be held by hand or be put in the chest or trouser pocket. They did not report on the sampling rate for the accelerometer.

[101] used both surface electromyography (SEMG) sensors and external accelerometers to detect gait activities and falls. Four SEMG electrodes were placed upon muscles of the right lower limb and two tri-axial accelerometers were placed on the chest and the right thigh of the user. The sampling rates for both the SEMG and accelerometers were 1,000 Hz. [102] used the accelerometer in a Google G1 phone with Android operating system and an independent accelerometer attached to the waist of the user. The latter was used mainly for comparison purpose. The sampling rate for all the used accelerometers was 50 Hz.

Both [103] and [104] proposed to attached an accelerometer at the head level in order to get more information about the body motion, by which they believed the accuracy of detection could be improved. The sampling rate for the accelerometer was 200 Hz in the former study and 50 Hz in the latter. [105] used a Samsung Galaxy SIII with Android operating system and a TI sensorTag with six on-board sensors, including an accelerometer. The former was used to send an SMS alert, while the latter was used for detecting the fall. It is interesting to note that our design was also based on a smartphone and a TI sensorTag and therefore we were keen to learn more about their work. However, although they mentioned that they planned to integrate the sensorTag into a wearable platform, they did not report on the system structure, design or results.

According to the above studies, we found that the accelerometer was commonly used as the main sensing unit with a sampling rate of from 40 Hz to 1,000 Hz, depending on what detailed acceleration signals were needed for their respective fall detection algorithms. In addition, we learned that the majority of them proposed to mount the phone or the accelerometer to the waist or put it in a trouser pocket, whereas a few studies allowed the phone or the accelerometer to be place at several pre-defined places, such as in the user’s hand or in a trouser or chest pocket. These findings were consistent with the conclusions made by [95,96], which however considered these system requirements as unsolved problems. Another interesting finding was that all these studies used smartphones with Android operating system. We thought that the least-privilege security model employed by Apple on the iOS operating system, which imposes more restrictions on developers, might well be the reason.

4.2.2 Classification Mechanisms and Achieved Accuracy

To classify falls and non-fall activities, threshold-based mechanisms were found to be of the mainstream, followed by machine learning-based ones. For example, [98,100-104] all employed a
classifier based on pre-defined thresholds on the amplitude of acceleration, especially the sum vector (SV) of accelerations in X-Y-Z axes, whilst [97,99] used machine learning mechanisms. Equation (4-1) shows how to calculate SV from the accelerations in the three axes. In addition to the function of detecting a fall, [97,100-102] also tried to identify a number of different pre-defined activities, such as walking and sitting, whereas [100,102] attempted to detect different kinds of falls, such as forward falls and backward falls, based on different threshold values.

\[ SV = \sqrt{x^2 + y^2 + z^2} \]  

(4-1)

With regard to the accuracy of fall detection, most studies [97,99,101,104] were able to achieve sensitivity and specificity of both higher than 90%, whereas some [100,102] had one index of higher than 80% and the other one higher than 90%. Here sensitivity is defined as the percentage of successfully identified falls and specificity is the percentage of successfully identified non-fall activities.

In our opinion, to some extent, the overall accuracy of these studies, no matter whether they adopted threshold-based or machine learning-based mechanisms, was similar to one another. Consequently, we believed that the usability and feasibility, among others, of the proposed solutions would be of equal importance with sensitivity and specificity. With this in mind, in our work we attempted to address issues on both the power consumption and fixed placement of the phone/sensor.

4.3 Design and Implementation

4.3.1 Sensing Accelerations from Falls or Other Activities

Our early work started by conducting a series of experiments to examine the accelerations arising from either an fall event or other activities together with the gravity based on the iPhone 5’s (i.e. the User Agent) built-in accelerometer with the acceleration range set to ±2g and the sampling rate at 10 Hz. Figure 4.1 shows the signatures of SV and accelerations in X, Y, and Z axis, respectively, during an intentional forward-fall test with the User Agent mounted at the ear side. The test was performed by a subject originally standing in front of a cushion (approximately 1m × 2m × 0.12m). He then moved one step forward and intentionally fell on the cushion.

As we did not concern about which way he fell, we observed that the signatures of SV alone had enough information about the fall. When the fall started, the SV decreased as a result of near free-fall motions. However we noticed that the recorded SV during this stage of falling down in all tests had never reached zero or near zero gravity as stated by some studies, such as [102], which we believed was caused by both the low sampling rate we deployed and the forward acceleration produced by the
intentional fall itself. Then when the body of the subject or the User Agent (iPhone) hit the cushion, the SV reached a maximal value. As during the fall the subject turned his head and used his hands to protect himself, we believed that to some extent this also added some noises to the measured accelerations.

Figure 4.1 Signatures of SV and Accelerations in X, Y, and Z axis in an Intentional Fall Test
(User Agent Mounted at Ear Side)

With regard to the time length between the start of a fall and a collision of the body of the subject (or the User Agent) and the ground (or the cushion), it can be calculated by using the second equation of motion [106], as represented by Equation (4-2). In (4-2), \( t \) is the time length in second; \( u \) is the initial velocity in the downward direction; \( s \) refers to the distance (i.e. the height of the accelerometer when the fall starts) in metre; and \( g \) is the acceleration due to gravity. For example, when the phone is mounted at ear side at a height of 1.7 metres, it would take about 0.6 seconds for the phone to drop onto the ground, no matter whether it is a free fall or it starts from a horizontal motion caused by a sideway force. We considered that this time length could be used as one of the criteria for classifying a fall from other activities.

\[
s = ut + \frac{1}{2} gt^2, \quad \text{where} \quad u = 0 \quad \text{and} \quad g = 9.8 m/s^2
\]

(4-2)

Figure 4.2 gives the signatures of SV and accelerations in three axes recorded during another intentional forward-fall test with the User Agent placed in the subject’s jacket pocket. When the subject was moving toward the cushion, the User Agent was swaying with his jacket, causing the SV to reach its peak before the fall. When the fall occurred, the SV first decreased and then increased. One important thing to note was that the signature of accelerations in the Y direction was outstandingly different from that in the previous test, mainly due to the different placement of the User Agent. This suggested that we should not rely on the signature of accelerations in a particular
direction, but the signature of the SV, to detect the fall, if we did not demand a fixed placement of the phone or accelerometer.

![Image of Acceleration + Gravity (g) graph with X, Y, Z, and SV axes.](image)

**Figure 4.2 Signatures of SV and Accelerations in X, Y, and Z axis in an Intentional Fall Test (User Agent Placed in a Jacket Pocket)**

Figure 4.3 gives the signatures of SV and accelerations in three axes recorded during a test with the User Agent held in the subject’s hand. In this test, the subject walked toward the cushion and then made an intentional fall on his knees. We found that it was difficult to classify the fall onto the knees from walking solely by comparing the signature of SV caused by these two activities, unless we also put the less vibrated signature of SV after the fall into consideration. Consequently, to build our fall detection algorithm, we assumed that only when a fall is followed by lying motionless, it is considered as an emergency that needs to trigger an alert.

![Image of Acceleration + Gravity (g) graph with X, Y, Z, and SV axes.](image)

**Figure 4.3 Signatures of SV and Accelerations in X, Y, and Z axis during an Intentional Fall onto the Knees (User Agent held in Hand)**
Figure 4.4 provides the signatures of SV and accelerations in the three axes recorded during a phone shaking test. In this test, the subject held the User Agent in his hand and shook the phone horizontally for a few times. By repeating this kind of tests, we found that it is highly possible to emulate the signature of SV aroused from a fall by shaking the User Agent carefully. However, we also notice that when the accelerations in one particular direction (for example the X axis in the chart in this case) bounced between the maximum and minimum values of the set range, for example ±2g, for several times, we could be certain that this must not be a fall event.

Figure 4.4 Signatures of SV and Accelerations in X, Y, and Z axis during a Phone Shaking Test
(User Agent held in Hand)

4.3.2 Power Consumption Experiments

In order to better understand what difference of power consumption would entail on the smartphone when the built-in accelerometer operates at different sampling rates or when the phone repeatedly receives acceleration data from an external BLE sensor, we conducted the following three power consumption experiments. Each was repeated for three times based on the same environmental settings:

- Power consumption experiment 1: we run iSeismometer App [107] (with network and alarm functions set to off) on an iPhone 5 (iOS 6.1.4 with the ‘Auto-lock’ function set to ‘never’) to display accelerations in X, Y, and Z axis, respectively, measured by the built-in accelerometer at a 10-Hz sampling rate. We also recorded how much time it took to consume the battery power from 100% to 90%, and to 80%, respectively.

- Power consumption experiment 2: we run iSeismometer App with the same set-ups as experiment 1, except the sampling rate for the accelerometer was set to 100 Hz, and recorded how much time it took to consume the battery power from 100% to 90%, and to 80%, respectively.
- Power consumption experiment 3: we run TI-BLE SensorTag App [108] on the same iPhone 5 to receive (via Bluetooth 4.0 connection) and display accelerations in X, Y, and Z axis, respectively, measured by the on-board KXTJ9 accelerometer of an external TI SensorTag (firmware version 1.4 with the other on-board sensors set to off) at a 10-Hz sampling rate. The time lengths it took to consume the battery power from 100% to 90%, and to 80%, respectively, were recorded.

Table 4.1 summarises the results of these three experiments. By inspecting these results, we found that the battery life of the iPhone 5 increased about 1.6 times (for example from 3,127.7 seconds in experiment 2 to 5,011.7 seconds in experiment 1) when the sampling rate for the iPhone’s built-in accelerometer decreased from 100 Hz to 10 Hz. Besides, the battery life of the iPhone 5 for using an external accelerometer at a 10-Hz sampling rate was about three times longer than that for using the iPhone’s built-in accelerometer at a 100-Hz sampling rate. These findings echoed the conclusions and suggestions made by [95].

<table>
<thead>
<tr>
<th></th>
<th>Power Consumption Experiment 1 (iPhone 5 built-in accelerometer at 10Hz)</th>
<th>Power Consumption Experiment 2 (iPhone 5 built-in accelerometer at 100Hz)</th>
<th>Power Consumption Experiment 3 (TI SensorTag on-board accelerometer at 10Hz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg. Time (sec.) for battery life from 100% to 90%</td>
<td>3,015</td>
<td>1,879.3</td>
<td>5,395.3</td>
</tr>
<tr>
<td>Avg. Time (sec.) for battery life from 100% to 80%</td>
<td>5,011.7</td>
<td>3,127.7</td>
<td>9,315.3</td>
</tr>
</tbody>
</table>

Table 4.1 iPhone 5's Battery Life Time Measurements in Power Consumption Experiments

4.3.3 Integration with an External Sensing Unit

Based on the results of the aforementioned experiments, we believed that by integrating an external sensing unit, such as the TI SensorTag, into the smartphone-based fall detection system, we could improve the issues on both power consumption and usability. The TI SensorTag was first released in Nov. 2012 as the first BLE development kit for smartphone applications. It is a relatively lightweight and ultra-low-power device (0.01 kg at a size of 71.2×36×15.5 mm) powered by a single coin cell battery (CR2032) with years of battery life. It is equipped with six on-board sensors, including TMP006 contactless IR temperature sensor, SHT21 humidity sensor, IMU-3000 gyroscope, KXTJ9 accelerometer, MAG3110 magnetometer, and T5400 (C953H) barometric pressure sensor. The cost of each TI SensorTag is US$25.

However, we also found that the long battery life comes with a trade-off with both a maximum sampling rate of 10 Hz and a limited resolution of accelerometer at ±2g, which could undermine the
details of acceleration measurements. For example, in our earlier emulated fall experiments described in section 4.3.1, we had never observed the situation in which the value of SV reached zero or near zero gravity due to a fall when the accelerometer was operating at a 10-Hz sampling rate and ±2g range. Under the circumstances, we needed to identify particular threshold values and developed our own algorithms, different from those adopted by the studies mentioned in section 4.2, for the construction of the fall detection subsystem.

By using TI SensorTag as an external sensing unit which repeatedly transmitted the measured accelerations in X-Y-Z axes, respectively, to the User Agent every 0.1 seconds, we conducted a number of experiments, some with settings similar to those mentioned in section 4.3.1 and some with new settings (such as placing the TI SensorTag in a shirt pocket or trouser pocket). Nevertheless, we do not intend to repeat the discussions on the findings and issues here, as most of them were analogous to those in our earlier experiments. Two exceptions that were important for our design were device throwing and dropping tests. Figure 4.5 provides the signatures of SV and accelerations in three axes recorded during a device (i.e. the TI SensorTag) throwing test. In this test, the subject first carried the TI SensorTag in his hand at a height of about 1.4 metres. He then gently threw the device forward and it dropped onto the ground at a distance of about 0.8 metres from the subject. By inspecting the signature of SV, we found that the value of SV reached 0.05g due to the (near) free-fall motions.

![Figure 4.5 Signatures of SV and Accelerations in X, Y, and Z axis in a Device Throwing Test](image)

Figure 4.5 gives the signatures of SV and accelerations in three axes recorded during a device dropping test. In the test, the TI SensorTag was first held by the subject at a height of about 0.8 metres and then dropped onto to the ground. Similar to the device throwing test, the value of SV reached 0.04 g due to the free-fall motions. We hence considered this as a particularly important feature to
distinguish a device throw or drop from a human fall when the fall detection system is based on the new settings with an external sensing unit.

![Image](image.png)

**Figure 4.6 Signatures of SV and Accelerations in X, Y, and Z axis in a Device Dropping Test**

### 4.3.4 Fall Detection Algorithm

Based on the aforementioned empirical experiments and signal analysis, five thresholds were defined in our fall detection algorithm, as shown in Table 4.2. This algorithm was invoked every time when the User Agent received new updates of accelerations in the three axes from the TI SensorTag. The algorithm works as follows: Whenever the derived SV is less than 0.15g (SV_FREE_FALL_THRESHOLD), it is considered that a device throw or drop event has occurred and therefore all flags are reset (lines 2 and 3). Otherwise, if the SV first drops below 0.79g (SV_THRESHOLD1) followed by bouncing over 1.48g (SV_THRESHOLD2) within 0.7 seconds, both the first and second flags are set (lines 6 to 15). After that, if in 0.7 seconds (with a few oscillations), the SV drops to and remains in the range of between 1.12g (SV_STILL_THRESHOLD1) and 0.88g (SV_STILL_THRESHOLD2) for more than 2 seconds, the third and fourth flags are set (lines 20 to 38) and a fall detection alert is triggered (line 47).

The 0.7 seconds was adopted to classify other activities from a fall based on the discussion (in section 4.3.1) that it took about 0.6 seconds for the device to fall from a height of about 1.7 metres to the ground. Therefore, if the time between the SV reaching SV_THRESHOLD1 and SV_THRESHOLD2 is longer than 0.7 seconds, it is considered that an activity other than a fall has occurred, and hence all flags are reset accordingly.

### 4.3.5 Implementation of Fall Detection Alerts

Having triggered a fall detection alert, the User Agent will first produce a beep and wait 10
seconds for the telemonitored patient to confirm his/her condition. If the patient disables the beep by clicking a button on the User Agent’s screen, no alert will be sent out. Otherwise, a fall detection alert containing minimum information about the patient (including his/her location based on Google Maps Geolocation API and the built-in GPS of the iPhone 5) will be automatically sent out.

```plaintext
1 SV ← getLatestSV( );
2 if SV < SV_FREE_FALL_THRESHOLD then
3   resetAllFlags( ); freeFallTimeStamp ← getTime( );
4   else if isFreeFallTimeStampSet( ) == false or timeFromFreeFallTimeStampSet( ) > 1600ms then
5     if firstFlagSV == 0 then
6       if SV < SV_THRESHOLD1 then
7         firstFlagSV ← SV; firstFlagTimeStamp ← getTime( );
8     end
9     else if sndFlagSV == 0 then
10    if timeFromFirstFlagSV( ) <= 700ms then
11      if SV < firstFlagSV then
12        firstFlagSV ← SV; firstFlagTimeStamp ← getTime( );
13      else if SV > SV_THRESHOLD2 then
14        sndFlagSV ← SV; sndFlagTimeStamp ← getTime( );
15     end
16     else
17       resetAllFlags( );
18     end
19   else if thirdFlagSV == 0 then
20     if timeFromSndFlagSV( ) <= 700ms then
21       if SV > sndFlagSV then
22         sndFlagSV ← SV; sndFlagTimeStamp ← getTime( );
23       else if SV < SV_STILL_THRESHOLD1 and SV > SV_STILL_THRESHOLD2 then
24         thirdFlagSV ← SV; thirdFlagTimeStamp ← getTime( );
25     end
26     else
27       resetAllFlags( );
28     end
29   else if fourthFlagSV == 0 then
30     if timeFromSndFlagSV( ) <= 2700ms then
31       if SV > SV_STILL_THRESHOLD1 or SV < SV_STILL_THRESHOLD2 then
32         thirdFlagSV ← 0; resetThirdFlagTime_stamp( ); motionlessCount ← 0;
33     else
34       motionlessCount ++;
35    if motionlessCount > 20 then
36       fourthFlagSV = SV;
37     end
38     if thirdFlagSV == 0 and timeFromSndFlagSV( ) > 700ms then
39       resetAllFlags( );
40     end
41   else
42     resetAllFlags( );
43   end
44 end
45 if fourthFlagSV != 0 then
46   triggerAnAlert( ); resetAllFlags( );
47 end
```

Table 4.2 Fall Detection Algorithm
In our early implementation, in addition to automatically sending a fall detection alert to the telemotor centre through the Internet, we also designed mechanisms for automatically sending the alert to designated recipients, such as carers of the patient, via both a text message and an email. However, the implementation of automatically sending a text message-based alert was no longer functional after iOS version five, as it was considered as a security vulnerability by Apple to allow an iPhone App to automatically make a phone call or send a text message without the user’s explicit permission (by clicking a confirm button). However, for the purpose of this thesis, we regarded this as a pure technical issue, rather than a research problem.

4.4 Cost-effectiveness Analysis

In total, 30 trials each with a sequence of simulated activities of daily living, such as walking, sitting, and standing up, followed by an intentional forward fall onto a cushion, were performed. The TI SensorTag (used in 27 trials) or the User Agent with a built-in accelerometer (used in three trials) were put at different places of a volunteer’s body, such as ear side, jacket pocket, shirt chest pocket, trouser pocket, or handheld. To address the usability issue, we did not strictly confine the TI SensorTag or User Agent to a certain tilting angle or orientation. Such a research design is apparently different from most of studies mentioned in section 4.2. Eight out of the 30 trials were excluded from our assessment, as they were either interrupted or interfered with by other people in the test environment.

4.4.1 Estimated Costs and Effectiveness

The results from the 22 trials were promising. Only one fall was not successfully detected and one non-fall activity was mistakenly classified as a fall. Table 4.3 summarises the results of these trials based on our proposed solution with sensitivity and specificity of both 95%. Meanwhile, we also performed dozens of trials on device (i.e. the TI SensorTag) throws or drops with a resultant specificity of 100%. We found that the accuracy of our proposed fall detection solution is among the highest in comparison with that of other studies mentioned in section 4.2.2.

<table>
<thead>
<tr>
<th></th>
<th>Sensitivity</th>
<th>Specificity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fall Detection</td>
<td>95.5%</td>
<td>95.5%</td>
</tr>
<tr>
<td>Device Throws/Drops</td>
<td>N/A</td>
<td>100%</td>
</tr>
</tbody>
</table>

Table 4.3 Results of our Trials

With an external sensing unit based on the TI SensorTag at a cost of US$25 (or an annualised
cost of £3.23\textsuperscript{*11} pounds sterling over a five-year period), we were able to enhance usability of our proposed system and prolong the battery life of the User Agent (i.e. an iPhone 5). It is our belief that these achievements have demonstrated that a cost-effective design of a smartphone-based fall detection solution is highly feasible.

### 4.4.2 More on the Device Shaking Events and Accuracy

As mentioned in section 4.3.1, it is possible to distinguish some device shaking events (or some similar actions) from falls by looking at the accelerations in each individual axis, especially when the accelerations bounce between the maximum and minimum values of the set range. However, some minor shaking actions that do not cause the accelerations to bounce between the upper and lower boundaries could produce almost identical SV signature to human falls and hence might well be classified as falls. To address this issue, we also developed another algorithm able to better identify this kind of device shaking events. Unfortunately, this new feature was introduced at the expense of less sensitivity of fall detection.

From a patient safety perspective, we considered that a fall detection system with higher sensitivity and lower specificity (in terms of lower percentage of successfully recognised device shaking or similar events) is better than another one with lower sensitivity and higher specificity. In addition, from a practical standpoint, we believed that by adding the aforementioned function to ask for user confirmation before an alert is sent out to the telemonitor centre and remote carers (section 4.3.5), the number of faulty alerts can be reduced to a minimal level.

### 4.5 Evaluation and Discussion

In this chapter, we have presented our proposed smartphone-based fall detection subsystem able to achieve highly comparable accuracy with sensitivity and specificity being both 95\%, but better usability and longer battery life in comparison with other related work. Through systematic assessments and carefully designed experiments, we demonstrated our solutions to resolve the outstanding problems facing current smartphone-based fall detection systems, for example the requirement for a fixed placement of the sensing device and the rapid exhaustion of smartphone battery power due to continuously performing sensing functions.

The results strongly suggest that by incorporating an external sensing unit, which is lightweight, easy-to-use, and less-intrusive, together with a dedicated fall detection algorithm, a cost-effective design of smartphone-based fall detection systems is highly feasible. We also believe that the

\textsuperscript{*11} The used exchange rate for US dollars to pounds sterling is 0.645.
proposed mechanism based on an external lightweight sensing unit is a good solution for the safety monitoring of patients with impaired ability, such as memory loss, as the external sensing unit can be simply pinned on their clothes for round-the-clock monitoring.

However, as our fall detection algorithm was developed based on experiments of simulated falls performed by a normal subject, we considered this as a limitation on applying our system to real life monitoring. Accordingly, we plan to perform further tests and evaluations using a wider range of subjects with both simulated and real falls. With a view to achieving a robust and reliable fall detection system, we also plan to further enhance the sensitivity and specificity of our solution either by improving our algorithm or by adopting emerging technologies in smartphone and sensing devices.
5. Real-time Indoor Patient Movement Tracking

5.1 Introduction

In healthcare, it has been widely acknowledged that in-home monitoring of the elderly or chronic disease outpatients’ daily movement patterns is useful for detection of early signs of new or deteriorating health issues. However, to achieve satisfactory accuracy remains a challenge for indoor location tracking [109]. The purpose of this chapter is to present our proposed real-time indoor movement tracking mechanism on the client side, aimed to be able to identify a patient’s real-time location within a home environment with maximum estimation error of two metres at a 95% confidence level. Key considerations about the development of our system included lower cost, reduced intrusiveness, and higher mobility, usability, deployability, and portability.

The material of this chapter is mainly based on our paper [110], which, after two minor revisions on 2 July 2015 and 9 Oct. 2015, was accepted on 11 Nov. 2015 by the IEEE Journal of Biomedical and Health Informatics. A preliminary version [111] of this work was also reported at MobiHealth 2014. The paper [110] (so does this chapter) expands significantly on that presentation in both technical details and extent of the evaluation. One thing to note is that in [110,111] the technical design, execution of experiments, and evaluations were the work of the author of this thesis.

A radio-based localisation technique using trilateration, trigonometry and received signal strength indication (RSSI) from three triangular deployed Bluetooth Low Energy (BLE) sensors was chosen to build the required functionality. Meanwhile, to tackle the problem of arbitrary variations of RSSI readings in an indoor environment and to enhance overall reliability of our system, we developed the three following elements: a step detection mechanism to produce extra patient location information; a discrete-time Kalman filter to improve distance estimation; and a tight coupling sensor fusion approach to integrating all these features as a whole. The results of our 24 experiments, each performed by a user carrying a User Agent and walking around a small office (54 square metres) for up to 65 seconds, were promising with an average estimation error of 0.47 metres. The main contribution of our work is a proof that a cost-effective design of indoor real-time location tracking is possible based on commodity smartphone-centric technologies.

To illustrate both the issues encountered and our solutions to those issues, the remainder of this chapter is structured as follows. In section 5.2, we introduce related work on step detection, indoor location tracking and the concepts of RSSI and Kalman filter and comment on the lessons learned. In Section 5.3, we elaborate our design and implementation of our proposed mechanisms in order to resolve the issues we encountered and to formulate an integral solution. Then in section 5.4, we provide the details of our cost-effectiveness analysis based on the experimental settings and results,
and summarise the main findings. Finally, in section 5.5 we discuss the limitations of our work and provide our evaluations and suggestions for future work.

5.2 Literature Review and Related Work

5.2.1 Step Detection

As stated by [112], most studies on step detection have emphasized estimation of step counts, for example [113,114], rather than indoor location tracking. However, for indoor localisation based on step detection, [112] used triads of accelerometers, angular rate sensors, and magnetometers (sampling rate ≈ 70 Hz) attached to the instep of a user to estimate the velocity and distance of movement and detect gait events. The average distance estimation error for their indoor 16-step straight-line walking experiments was 5.5% with a maximum error of 2.05 metres.

[115,116] used smartphones’ built-in inertial sensors (based on < 25 Hz and 50 Hz sampling rate, respectively) to detect steps and heading direction in order to estimate user location. Together with a floor map, [115] in their hand-held experiments achieved a mean error of 1.5 metres and a 95th percentile error of 4.3 metres. [116] required using two sets of sensors (i.e. two smartphones) at one time, both placed in the user’s trousers pockets, to achieve an average estimation accuracy of about 1.6 metres in their 10 2-minute walking experiments.

In our opinion, the high sampling rate employed by these studies, though providing more precise measurements, would significantly drain a smartphone’s battery. Besides, both mounting sensors to the instep and using two smartphones at one time were either intrusive or impractical.

5.2.2 Indoor Radio-based Location Tracking

For indoor radio-based location tracking, there are a number of techniques, including RSSI, angle of arrival (AOA), time of arrival (TOA), and time difference of arrival (TDOA). The first one estimates the distance directly from the strength of the received signal, whereas AOA is based on both the signal strength and angle of each beam from a usually costly antenna array. The remaining two basically convert the travel time of signals into distances. We chose RSSI to build our system because AOA is more costly, and both TOA and TDOA require a very high resolution of timer, which is generally not available on a commodity device, such as a smartphone. Though all these techniques have gained popularity in recent years, some major problems in this field, as indicated by [109], remain unresolved, such as computationally intensive algorithms, excessive access point installations and unstable wireless signal transmission.
According to [117], both the human body, which attenuates radio signals, and the use of the Wi-Fi in a smartphone can cause errors in signal strength measurements based on BLE. To improve distance estimates, both the Monte Carlo [118,119] and Kalman filter [120-122] methods were commonly adopted for constructing RSSI-based localisation algorithms. This is because both can produce statistically more precise estimation of system states than those solely based on one or a few noisy measurements. However, from our perspective, the need for an extra computer to perform location estimates using the Monte Carlo method, as proposed by some studies [118,119], raises cost and reliability issues.

Another common technique was to produce and store a detailed wireless signal strength map at each specific survey location before performing localisation [109,119,121]. Location tracking was then conducted by comparing real-time RSSI measurements with the stored signal strength map. However, we consider that the need to produce such a map for each survey location would cause issues on system usability, deployability and portability, as well as on longer processing time and higher system requirements, such as storage and database. In some cases, excessive sensor nodes were needed to conduct target tracking. For instance, [120] used both eight static sensor nodes and one or more mobile nodes in its simulations.

With regard to accuracy and performance, some studies could achieve an overall estimation error of about two metres. For example, the distance estimation errors of [109] implemented in an iPhone were within 2.3 metres with 90% precision, whereas the average error for walk-through tests of [119] was 2.1 metres. By applying a modified extended Kalman filter on an existing RSSI data set, [122] achieved an average estimation error of 2.11 metres. Though the overall distance estimate error of [118] was 1.2 metres, its maximum error was about 2.5 metres and the use of a networked computer for performing heavy computational tasks caused latency of up to eight seconds. [123] claimed that they had estimation errors of below micrometres in their software simulations by using TOA based trilateration with IEEE 802.11v. We argue that this can neither be applied to current smartphones, nor to commodity desktop systems, because these systems’ timer resolutions would have needed to be up to 3.33 femtoseconds (i.e. $10^{-15}$) to achieve a distance precision in micrometres. (According to Microsoft [124], the maximum timer resolution in MS Windows is 1 millisecond.)

The identified issues mentioned above indicate a remaining gap in achieving wider uptake of indoor location tracking in society. Moreover, lessons learned from these studies suggest that we need to develop a lightweight, but accurate localisation algorithm suitable for execution in a smartphone. We should also avoid adopting detailed wireless signal maps, as well as excessive hardware installations. These would help achieve higher mobility, reliability, usability, deployability and portability of our real-time telemonitoring system at a lower cost.
5.2.3 Received Signal Strength Indication (RSSI)

Theoretically, RSSI is based on the inverse-square law that the wireless signal strength is proportional to the inverse of the square of the distance from the signal source. Equation (6-1) denotes the relationship between received signal strength and corresponding distance [125]. However, in reality, due to several issues, such as multi-path fading, indoor shadowing, and interference, the relation between signal strength and distance in an indoor environment usually involves a much wider range of factors. This significantly increases the complexity of RSSI-based distance estimates. To improve the accuracy of RSSI-based location tracking, numerous approaches and algorithms, such as those mentioned in the previous sub-section, have been proposed.

\[
RSSI_d = -10n \log(d) + RSSI_0
\] (5-1)

In Equation (5-1), \(RSSI_d\) (in Decibel-milliwatts) is the RSSI measured at distance \(d\) (in metres) from the source; \(n\) is the path loss exponent and \(RSSI_0\) (in Decibel-milliwatts) is the RSSI measured at one-metre distance. Further details about how (5-1) is derived is outside the scope of this thesis, but can be found in a number of foundational texts, such as [125]. We can also rewrite Equation (5-1) as Equation (5-2) or (5-3) to estimate the value of the path loss exponent \(n\) or distance \(d\) when the values of the other variables are known.

\[
n = -((RSSI_d - RSSI_0) / 10\log(d))
\] (5-2)

\[
d = 10^{(RSSI_d - RSSI_0) / (10n)}
\] (5-3)

5.2.4 Kalman Filter

Because a Kalman filter is relatively lightweight and has a much better convergence rate than a Monte Carlo filter, we chose the former to build our localisation algorithm. Equations (5-4) and (5-5) explain how to model a system using the discrete-time Kalman filter [126]:

\[
x_k = Ax_{k-1} + Bu_k + w_k
\] (5-4)

\[
z_k = Hx_k + v_k
\] (5-5)

In Equation (5-4), \(x_k\) is the estimate of the system state variable at time \(k\); \(u_k\) is the control signal; and \(w_k\) is the process noise. In Equation (5-5), \(z_k\) is the measurement value at time \(k\); and \(v_k\) is the measurement noise. \(A\), \(B\) and \(H\) are general form matrices, introduced to model the system, and in many cases can be simplified as numeric values. To perform the estimates based on a discrete-time Kalman filter, the two sets of equations in Table 5.1 can be used repeatedly.
Table 5.1. Two Sets of Kalman Filter Equations for State Estimation, after [126]

In Table 5.1, $\hat{x}_k$ is the “a priori estimate” of the system state at time $k$ before measurement update correction; $\hat{x}_{k-1}$ is the updated estimate (or a posteriori estimate) at time $k-1$ after measurement; $P_k^-$ is the “a priori error covariance”; $R$ is the measurement error covariance; $K_k$ is the Kalman Gain; $x_k$, which is the updated estimate (or a posteriori estimate) at time $k$ after measurement, is the very value we try to find; and $P_k$ is the updated error covariance after measurement. More details about how these parameters are applied can be found in existing literature on the topic, such as [126]

5.3 Design and Implementation

<table>
<thead>
<tr>
<th>Distance (metre)</th>
<th>Min. RSSI (dBm)</th>
<th>Max. RSSI (dBm)</th>
<th>Mean RSSI (dBm)</th>
<th>Standard deviation (dBm)</th>
<th>n (path loss exponent)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-79</td>
<td>-68</td>
<td>-72.9</td>
<td>3.68</td>
<td>NA</td>
</tr>
<tr>
<td></td>
<td>-70</td>
<td>-63</td>
<td>-66.9</td>
<td>1.87</td>
<td>NA</td>
</tr>
<tr>
<td></td>
<td>-87</td>
<td>-66</td>
<td>-73</td>
<td>5.83</td>
<td>NA</td>
</tr>
<tr>
<td>2</td>
<td>-90</td>
<td>-72</td>
<td>-78.7</td>
<td>4.63</td>
<td>1.93</td>
</tr>
<tr>
<td></td>
<td>-94</td>
<td>-67</td>
<td>-73.7</td>
<td>5.57</td>
<td>2.24</td>
</tr>
<tr>
<td></td>
<td>-93</td>
<td>-74</td>
<td>-78.4</td>
<td>5.06</td>
<td>1.78</td>
</tr>
<tr>
<td>4</td>
<td>-107</td>
<td>-77</td>
<td>-85.9</td>
<td>6.98</td>
<td>2.16</td>
</tr>
<tr>
<td></td>
<td>-74</td>
<td>-69</td>
<td>-71.1</td>
<td>1.41</td>
<td>0.69</td>
</tr>
<tr>
<td></td>
<td>-103</td>
<td>-78</td>
<td>-87.4</td>
<td>5.2</td>
<td>2.38</td>
</tr>
<tr>
<td>6</td>
<td>-103</td>
<td>-87</td>
<td>-94.6</td>
<td>4.27</td>
<td>2.79</td>
</tr>
<tr>
<td></td>
<td>-80</td>
<td>-73</td>
<td>-76.1</td>
<td>1.9</td>
<td>1.17</td>
</tr>
<tr>
<td></td>
<td>-100</td>
<td>-83</td>
<td>-89.8</td>
<td>5.13</td>
<td>2.16</td>
</tr>
<tr>
<td>8</td>
<td>-99</td>
<td>-87</td>
<td>-93.9</td>
<td>3.2</td>
<td>2.33</td>
</tr>
<tr>
<td></td>
<td>-88</td>
<td>-71</td>
<td>-78.3</td>
<td>3.91</td>
<td>1.26</td>
</tr>
<tr>
<td></td>
<td>-94</td>
<td>-86</td>
<td>-89</td>
<td>2.03</td>
<td>1.77</td>
</tr>
</tbody>
</table>

Table 5.2 RSSIs Recorded at different Distances from Sensor $S_i$

Our designs and experiments started from the use of raw RSSI data received from three triangular deployed BLE sensors (i.e. TI SensorTags) to estimate the location and movement of a tracked target who held a User Agent (i.e. an iPhone 5) in their hand and walked around a small
concrete-walled office (9 metres × 6 metres). The three BLE sensors (denoted as $S_0$, $S_1$, and $S_2$) were placed against three different walls of the office at 1.1-metre height from the floor. However, because of the very diverse and unstable RSSI signals for the reasons discussed above, it was impossible to use the raw RSSI data alone to reliably estimate the location and movement of the target.

Table 5.2 shows the results of three 10-minute raw RSSI measurements recorded at a distance of one to eight metres from sensor $S_1$. For example, during the first recording, 600 RSSI updates at one-metre distance from sensor $S_1$ were received and the derived mean RSSI was -66.9 dBm with a standard deviation of 1.87 dBm, whereas the mean decreased to -73 dBm with a standard deviation of 5.83 dBm during the second recording. The values of the path loss exponent at different distances were estimated by using Equation (5.2).

We then implemented a Kalman filter to improve RSSI estimates based on noisy RSSI measurements. However, some identified issues in relation to the one-second RSSI update rate implemented by iOS caused the resultant RSSI and distance estimates to become very unreliable. Consequently, we decided to take a sensor fusion approach to performing location estimates based on the following three considerations:

- The target (or patient) movement needs to be kept track of all the time. By integrating information about the target’s each movement into the Kalman filter, we can avert the problem of missing out movement updates due to no available RSSI measurements when performing estimation.

- To simplify the deployment of the three reference sensors, a mechanism needs to be in place to record where the patient places these sensors prior to performing real-time monitoring.

- It is possible to more accurately estimate a reasonable location based on the information about ongoing patient movements, previous estimated locations, and the results of trilateration than to estimate distances and locations from raw RSSI readings without any other reference information.

Based on this approach, the target movement information, including both distance and heading data produced through step detection, was used as the control signal $u_k$ of Equation (5.4) for real-time estimation of distances. Moreover, the state variable $x_k$ in Equation (5.4) represented the estimate of the distance (rather than RSSI) between the target (holding a User Agent) and a reference sensor. The detailed design and implementation of a step detector, a discrete-time Kalman filter, and several estimate optimization mechanisms are described in the following.

5.3.1 Step Detector

A step detector was designed and implemented on the User Agent using acceleration parameters
(generated by the iPhone’s built-in accelerometer) and heading information (by compass) at a 10 Hz sampling rate. Figure 5.1 shows the recorded signatures of the SV of accelerations in X-Y-Z axes, forward accelerations in the Y axis, and heading angles in a movement test performed by a user holding a User Agent in his hand with the screen facing upward at an elevation angle of approximately 25 degrees.

![Signature of SV and Y-axis Acceleration, and Heading in a Movement Test](image)

**Figure 5.1** Signatures of SV and Y-axis Acceleration, and Heading in a Movement Test

```plaintext
1  SV ← getLatestSV( );
2  if firstFlagSV == 0 then
3      if SV > SV_THRESHOLD1 then
4          firstFlagSV ← SV; fwdAcc ← getForwardACC( );
5      end
6  else if sndFlagSV == 0 and timeFromFirstFlagSV( ) <= 400ms then
7      if SV > firstFlagSV then
8          firstFlagSV ← SV; fwdAcc ← getForwardACC( );
9      else if (firstFlagSV - SV) > SV_THRESHOLD2 then
10         sndFlagSV ← SV;
11             if (getHeadingChangeIn100ms( ) >= HEADING_THRESHOLD
12                and fwdAcc > FWD_ACC_THRESHOLD) or timeFromPreviousStep( ) <= 200ms then
13                resetAllFlags( );
14            else
15                signifyOneStepIsDetected( );
16            end
17        end
18 else
19   resetAllFlags( );
20 End
```

**Table 5.3 Step Detection Algorithm**

Table 5.3 summarises the step detection algorithm. Based on empirical experiments and signal analysis, two thresholds, i.e. 1.07 gravities (SV_THRESHOLD1) followed by a reduction of 0.209 gravities (SV_THRESHOLD2) of the SV within 400 milliseconds, were defined. If these thresholds
were met during real-time monitoring, the User Agent would signify a detected step when, at the same time, the forward acceleration was less than or equal to -0.3 gravities (FWD_ACC_THRESHOLD) or the change of heading angle within 100 milliseconds was less than seven degrees (HEADING_THRESHOLD). If two consecutive steps were signified within 200 milliseconds, the second one was regarded as faulty and hence was discarded.

Whenever a step or several steps were detected before the User Agent received new RSSI updates from all three reference sensors, a new location of the target, as well as the change of distance between the target and each of the sensors during this period of time, was calculated based on both the pre-defined length of a step (e.g. 0.65 metres) and the heading data.

Figure 5.2 gives two scenarios in which the target T is moving from one known location, denoted as \( T_{\text{old}}(x_0, y_0) \), to another unknown location, denoted as \( T_{\text{new}}(x_i, y_i + \Delta y) \) for one step detected or \( T_{\text{new}}(x_i + \Delta x_1 + \Delta x_2, y_i + \Delta y_1 - \Delta y_2) \) for two steps detected before the User Agent receives new RSSI updates. Here \((x, y)\) is the coordinate of \( T_{\text{old}} \) and \( S_i \) refers to the \( i \)-th sensor placed at location \((x_i, y_i)\). The distances between \( S_i \) and \( T_{\text{old}} \) and between \( S_i \) and \( T_{\text{new}} \) (or \( T_{\text{new}} \)) are indicated as \( d_{i,\text{old}} \) and \( d_{i,\text{new}} \) (or \( d_{i,\text{new}} \)) respectively and the length of each step of the target (or patient) is \( \lambda \). The \( \text{North}_{\text{mag}} \) represents the direction of magnetic north with its heading angle at 0°, while \( X_{\text{env}} \) and \( Y_{\text{env}} \) refer to the X- and Y-axis, respectively, in the monitoring environment. \( Y_{\text{env}} \) is pointing toward \( \theta \), \( X_{\text{env}} \) is pointing toward \( \theta + 90° \), and the target is heading toward \( h_1 \) in Figure 5.2(a) and \( h_2 \) in Figure 5.2(b) respectively. For the scenario in Figure 5.2(a), we can calculate the displacement of the target in both X- and Y-axis and the change of distance (denoted as \( d_{i,\text{chg}} \)) between the target and sensor \( S_i \) using Equations (5-6), (5-7), and (5-8), as follows:
\[ \phi_i = ((\theta + 90') \mod(360')) - h \]
\[ \Delta x_i = \lambda \times \cos(\phi_i) \quad \text{and} \quad \Delta y_i = \lambda \times \sin(\phi_i) \]
\[ d_{i,\text{chg}1} = d_{i,\text{new}1} - d_{i,\text{old}} = \sqrt{(x_i + \Delta x_i - x_j)^2 + (y_i + \Delta y_i - y_j)^2 - \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}} \]

For the scenario in Figure 5.1(b), we can calculate \( \phi_2, \Delta x_2, \Delta y_2, \) and \( d_{i,\text{new}2} \) in a similar way and then work out the displacement of the target and the change of distance (denoted as \( d_{i,\text{chg}2} \)) between the target and sensor \( S_i \) using Equation (5-9).

\[ d_{i,\text{chg}2} = d_{i,\text{new}2} - d_{i,\text{old}} \]  

5.3.2 Kalman Filter for Distance Estimates

A discrete-time Kalman filter was developed for estimating the distance (denoted as \( d_{i,k} \); hereafter an added subscript \( i \) to each variable shown in Table 5.1 refers to sensor \( S_i \)) between the target and sensor \( S_i \) (subscript \( i \) could be 0, 1 or 2) at time \( k \). Upon receiving new RSSI updates from a sensor, the distance information calculated by the step detector would be used as the control signal \( u_{i,k} \) of the Kalman filter. The reason for waiting for new RSSI updates before starting a new run of state estimation via the Kalman filter was to synchronize the target movement with RSSI readings. This was because the iOS updates RSSI measurements for each sensor at a maximum rate of once per second and updates for different sensors usually occur at different times, whereas a walking step generally takes less than one second. This synchronization process would incur a latency of less than a second in most cases, but up to five seconds in the worst case.

When a step was correctly detected, the process noise, which occurred mainly due to the inaccuracy of inferring the heading angle and step length, would be relatively small. So we used a value between zero and 0.13 metres (i.e. the standard deviation of step detection, if we assumed that on average one in every twenty-five steps was incorrectly detected or missed) as the process noise \( w_{i,k} \), depending on how much correlation between the step detector and the Kalman filter was needed. The process noise covariance \( Q_i \) was equal to \( E[w_{i,k} w_{i,k}^T] \), where \( E[w_{i,k}] \) is the expected value of \( w_{i,k} \) and \( w_{i,k}^T \) is the transpose of \( w_{i,k} \).

The value for measurement noise generally became bigger with the increase of distance between the User Agent and sensor \( S_i \). Based on our empirical experiments in fine tuning different parameters to get the best estimation results, we assumed that the measurement noise \( v_{i,k} \) was equal to the converted distance multiplied by a constant factor of 0.6. Meanwhile, matrices \( A, B \) and \( H \) in Equations (5-4) and (5-5) were all simplified as a numerical constant of one and the path loss exponent \( n \) in (5-1) was set to 2.4 for \( S_0 \), as well as 2.2 and 2.1 for \( S_1 \) and \( S_2 \) respectively. The initial
value of \(d_{i,0}\) was set to a pre-defined distance, as we assumed the real-time monitoring always started at a known location and the error covariance \(P_{i,0}\) was simply set to a non-zero value, e.g. 0.5. With all these considerations and assumptions, the two sets of the Kalman filter equations as shown in Table 5.1 were ready for estimating the distance between the User Agent and each of the three sensors. Both a time update equation and a measurement update equation are rewritten using the aforementioned variables in the following:

\[
d_{i,k}^{\text{chg}} = d_{i,k-1} + d_{i,chg} + w_{i,k}
\]

\[
d_{i,k} = d_{i,k}^{\text{chg}} + k_{i,k} (d_{i,\text{mea}} - d_{i,k}^{\text{chg}})
\]

In Equation (5-10), \(d_{i,chg}\), which refers to the change of distance between target and sensor \(S_i\) during the time period between RSSI updates received by the User Agent, is the control signal of our Kalman filter. In Equation (5-11), \(d_{i,\text{mea}}\) is the distance converted from RSSI measurements received from sensor \(S_i\).

### 5.3.3 Mechanisms for Estimate Optimization

To calculate the real-time location of the tracked target based on the outputs from the Kalman filter, traditional techniques, including trilateration, trigonometric functions and Maximum-Likelihood Estimation (MLE) were first adopted. To apply trilateration, the distances from three sensors estimated by the Kalman filter at a certain time were used as radii to create three circles each centered at \(S_0\), \(S_1\), and \(S_2\) respectively. (The details of implementing trilateration are outside the scope of this thesis.) Ideally, there should have been a joint intersection point among the three circles, representing the optimally estimated real-time location of the target at that precise moment. However, the reality was much more complicated with both underestimates and overestimates of the three distances, probably happening at the same time.

Therefore, when there was an area of intersections, rather than a single point, found among the three circles, both trigonometric functions and an iterative procedure based on Equation (5-12), the equation for MLE, were adopted to find an estimated target location \((x, y)\) within the intersections with a least value of \(\sigma_{x,y}\). In Equation (5-12), \((x_i, y_i)\) refers to the coordinate of sensor \(S_i\) and \(d_i\) is the estimated distance from \(S_i\) via Kalman filter. Alternatively, if intersections were found only between two circles or there was not any intersection among the three circles, we progressively enlarged some or all of the circles by 0.05 metres at a time until intersections were found, and then we applied the techniques mentioned above. However, the resultant estimates were still not satisfactory.

\[
\sigma_{x,y} = \sum_{i=0}^{2} \left| \sqrt{(x-x_i)^2 + (y-y_i)^2} - d_i \right|
\]
Our assessment revealed that there might be two main problems that caused such unsatisfactory results. Firstly, the inaccuracy and insufficient number of RSSI measurements had inborn negative impacts on the accuracy of distance estimation. Secondly, the process of manipulating estimated distances to create joint intersections among the three circles had introduced further noise into the system.

Figure 5.3 provides an example of how the radius manipulation process affects the final location estimation when the estimated distances between the target and each of three sensors are not accurate. In Figure 5.3(a), three estimated distances via Kalman filter at time $k$ are denoted as $d_{0,k}$, $d_{1,k}$, and $d_{2,k}$, and the actual location of the tracked target is marked with a red “x” and labeled as $T$. Meanwhile, $T_{est}(x_{est}, y_{est})$ denotes the estimated location and its coordinate. Since there are no common intersections among the three circles, centered at $S_0$, $S_1$, and $S_2$ respectively, there is a possibility of underestimate of either $d_{1,k}$ or $d_{2,k}$, or of both. In order to apply location estimation mechanisms, we can enlarge either circle $S_1$ or $S_2$ by increasing $d_{1,k}$ or $d_{2,k}$ respectively, as depicted in Figures 5.3(b) and 5.3(c).

Figure 5.3 Scenarios of Radius Manipulation Process

In Figure 5.3(b), the created intersection area (the arrow-pointing area) is further away from the actual target location, which would compromise the accuracy of $T_{est}(x_{est}, y_{est})$. In comparison, the created intersection area in Figure 5.3(c) is nearer to the actual location of the target. If we further enlarge circle $S_2$ a bit, the result would be even more accurate. However, as we do not possess the knowledge of the target’s actual location, it is not feasible to set a clear rule about whether we should enlarge circle $S_1$ or $S_2$ (or both) and to what extent we should adjust their radii (i.e. the estimated distances $d_{i,k}$) in such a situation.
It is apparent that one way to tackle the first identified problem is to take a large number of RSSI measurements; enough to produce both a detailed wireless signal strength map for each patient’s home and find out more appropriate values for relevant parameters prior to performing real-time monitoring. However, by doing so we would complicate the process of system deployment and set-up.

Consequently, a follow-step-detection mechanism was developed to address this problem by choosing a location \((x, y)\), nearest to the estimated target location \((x_s, y_s)\) based on step detection rather than MLE, within the intersection area. This means finding a location \((x, y)\) where \(\sigma_{x,y}\) in Equation (5-13) has the least value.

\[
\sigma_{x,y} = \sqrt{(x-x_s)^2 + (y-y_s)^2}
\]  

(5-13)

Furthermore, to address these two problems simultaneously, another tight coupling mechanism was also developed by using the average of distances both estimated by step detection and converted from RSSI readings as the measurement value, e.g. \(d_{i,\text{mea}}\) in Equation (5-11). When the difference between two consecutive RSSI readings was less than or equal to two (i.e. \(\text{RSSI\_CHG\_THRESHOLD}\)), meaning these RSSI measurements might well be more accurate than usual, we halved the measurement noise factor for the next run of the Kalman filter. This mechanism then use Equation (5-13) to choose one existing intersection point nearest to the target location estimated by step detection as the final estimated location of the target at that precise moment. Table 5.4 summarises this mechanism.

```plaintext
1 if hasReceivedNewRSSI== TRUE then
2    distanceByStep ← getNewDistanceFromStepDetection();
3    distanceByRSSI ← getNewDistanceFromRSSI();
4    rssiChange ← abs(currentRSSI – previousRSSI);
5    if rssiChange <= RSSI_CHG_THRESHOLD then
6       measurementNnoiseFactor ← measurementNnoiseFactor/2;
7    end
8    measurementValue ← (distanceByStep + distanceByRSSI)/2;
9    newDistanceByKalman ← performKalmanMeasurementUpdate();
10   newLocation ← performFollowStepDetectionForNewLocation();
11 end
```

Table 5.4 Tight-Coupling Sensor Fusion Mechanism

Figure 5.4 shows the screenshot of the User Agent during a live test of real-time location tracking. (To perform the monitoring, it is not necessary for the user to navigate to this view, which was mainly used for testing purpose.) The upper part of the view shows both the acceleration measurements and heading data for step detection, whilst the lower part gives the RSSI readings and estimated distances from three BLE sensors, as well as the estimated coordinates, i.e. \((1.5, 1.9)\) and
(1.6, 1.9), of the target based on step detection and tight-coupling sensor fusion mechanism, respectively.

![Figure 5.4 Screenshot of the User Agent during Real-time Location Tracking](image)

5.4 Cost-effectiveness Analysis

![Figure 5.5 Tracked Paths based on Different Mechanisms](image)

24 trials were performed by a user holding a User Agent in their hand with the screen facing upward at an elevation angle of approximately 25 degrees and walking around a concrete-walled office (9 metres x 6 metres) with the same set-ups of three BLE sensors (as mentioned in section 5.3) for up to 65 seconds. The four mechanisms mentioned in the previous section i.e. step detection, MLE, follow-step-detection, and tight-coupling sensor fusion, were implemented to track the user’s real-
time movement. Figure 5.5 shows the one-minute tracked paths based on the step detection (represented as orange dashed lines) and tight-coupling sensor fusion mechanism (grey dashed lines) in the 22nd trial together with the real movement (red lines) with a walking sequence of ① ② ③ ④ ③ ② ③ ④ ③ of the user.

5.4.1 Estimated Costs and Effectiveness

Figure 5.6 shows the average estimation errors based on the aforementioned four mechanisms, including step detection (red line), MLE (purple dashed line), follow-step-detection (green dotted line), and tight-coupling sensor fusion (blue line) in each of these 24 trials. As shown in the chart, the performance of the step detection and tight-coupling sensor fusion mechanisms were somewhat similar in terms of accuracy, and in some circumstances one could outperform the other.

![Figure 5.6 Average Estimation Errors based on Different Mechanisms](image)

As shown in Figure 5.7, the maximum estimation error recorded within the 24 trials was 1.25 metres for step detection, 3.05 metres for MLE, 2.98 meters for follow-step-detection, and 1.81 metres for tight-coupling sensor fusion mechanism.

![Figure 5.7 Maximum Estimation Errors based on Different Mechanisms](image)
Table 5.5 summarises the results of target movement estimation based on these four mechanisms. For example, the overall average estimation error based on step detection was 0.47 metres (with a standard deviation of 0.154 metres), whilst it was 0.56 metres (with a standard deviation of 0.165 metres) when based on the tight-coupling sensor fusion mechanism. Accordingly, we concluded that both the step detection and tight-coupling sensor fusion mechanism could realise real-time indoor location tracking with one metre precision based on the settings mentioned above.

<table>
<thead>
<tr>
<th>Mechanisms for Estimate Optimization</th>
<th>Average Max. Estimate Error (metre)</th>
<th>Avg. Estimate Error (metre)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step Detection</td>
<td>0.86</td>
<td>0.47</td>
</tr>
<tr>
<td>Maximum-Likelihood Estimate</td>
<td>1.78</td>
<td>1.04</td>
</tr>
<tr>
<td>Follow-step-detection Mechanism</td>
<td>1.30</td>
<td>0.71</td>
</tr>
<tr>
<td>Tight Coupling Mechanism</td>
<td>1.01</td>
<td>0.56</td>
</tr>
</tbody>
</table>

Table 5.5 Estimation Errors based on Different mechanisms

To estimate the costs of the required equipment for performing our proposed location tracking, we assumed that having a smartphone, i.e. the User Agent (or an iPhone in this case), is a prerequisite for every patient to enroll in the intervention group for telemonitoring. Accordingly, the three BLE sensors (i.e. TI SensorTags), each at a cost of US$25, were the only items to be included for the calculation. Table 5.6 provides the results of the estimated costs. The total annualised cost over a five-year period was US$15 or £9.68 pounds sterling.

<table>
<thead>
<tr>
<th>Item</th>
<th>Cost per Item (US$)</th>
<th>Number of Items</th>
<th>Total Costs (US$)</th>
<th>Annualised Costs over 5 Years (US$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLE Sensor</td>
<td>25</td>
<td>3</td>
<td>75 (or £48.39)</td>
<td>15 (or £9.68)</td>
</tr>
</tbody>
</table>

Table 5.6 Estimated Equipment Costs based on Our Settings

5.4.2 More on the Different Settings and Overall Accuracy

To evaluate how different parameter settings affect the results of our algorithms, we also tuned a number of parameters for different variables, such as path loss exponent, process noise, and measurement noise, and simulated the location tracking together with RSSI measurements derived from the real trials. Our key finding was that the overall accuracy of our localisation algorithm is not significantly affected by path loss exponent and measurement noise, but mainly dependent on the accuracy of the step detector, the value of process noise, and our choice of a particular coordinate as
our optimal estimate of the real-time target location. Table 5.7 shows the impact of different parameter settings on estimate accuracy based on the tight coupling sensor fusion mechanism with RSSI measurements from the 20th trial.

<table>
<thead>
<tr>
<th>One-metre RSSI values for ( S_0, S_1, S_2 )</th>
<th>Process noise (( \sigma ))</th>
<th>Measurement noise factor (( \nu ))</th>
<th>Max. estimate error (metre)</th>
<th>Avg. estimate error (metre)</th>
</tr>
</thead>
<tbody>
<tr>
<td>75, 72, 71</td>
<td>0.023</td>
<td>0.6</td>
<td>0.58</td>
<td>0.33</td>
</tr>
<tr>
<td>72, 72, 72</td>
<td>0.10</td>
<td>0.6</td>
<td>0.91</td>
<td>0.68</td>
</tr>
<tr>
<td>72, 72, 72</td>
<td>0.05</td>
<td>0.6</td>
<td>0.83</td>
<td>0.54</td>
</tr>
<tr>
<td>72, 72, 72</td>
<td>0.023</td>
<td>0.5</td>
<td>0.47</td>
<td>0.37</td>
</tr>
</tbody>
</table>

**Table 5.7 Changes of Estimation Results due to Different Parameter Settings**

With regard to the accuracy of the step detector, the average number of misestimated steps was 1.04 per trial. Seven out of 24 trials had no misestimated steps. It is our observation that the misestimate of a step usually happened when the user (carrying a User Agent) stomped in one place without geographically moving or when the user’s leading foot and lagged foot in the last step of walking caused separate, noteworthy vibrations of the SV and forward acceleration. The 10 Hz sampling rate was adopted to reduce power consumption at the expense of losing much detailed acceleration information. However, we believed the results were still promising.

Another issue that also caused notable misestimates of the movement was the compass heading deviation, which could gradually accumulate up to about ±20°, or even worse when the compass was interfered with by a local magnetic field. It is also important to note that the higher accuracy of measuring the orientation of the test environment (i.e. knowing \( \theta \) in Figure 5.2) before telemonitoring, the better the overall accuracy of the step detector. (In our earlier experiments, the step detectors had a maximum error of 1.17 metres and an average error of 0.58 metres.) Although the iPhone would automatically calibrate the compass from time to time, such an occasional drift could cause a deviation of around 0.22 metres in one-step estimation (with step length of about 0.65 metres), or even accumulating to several metres when the user walked multiple steps in a straight line. To calibrate the iPhone’s compass heading estimation, we had tried using iPhone’s gyroscope measurements, including yaw rotation and quaternion, as references for calibration. However, this effort proved to be ineffective, as the gyroscope outputs were even more inaccurate.

We also conducted some experiments using three BLE sensors in a real home environment (with a size of about 52 square metres), divided into five different-purpose rooms, such as bedroom, bathroom, and kitchen. We found that at some particular locations within this environment, one of the
three BLE sensors would lose the wireless connection with the User Agent and then automatically turned itself off after some 180 seconds, causing the localisation algorithm to malfunction. Consequently, in our future work, we plan to deploy more BLE sensors in a home environment to improve the signal coverage and to integrate accurate location information from low-cost force sensitive resistor (FSR) sensors deployed at certain known locations inside that environment to further enhance accuracy of our localisation algorithm.

Regarding the orientation of the User Agent (or the iPhone) within the trials, our intention was to use a small, lightweight sensor tag (e.g. TI SensorTag with on-board inertial sensors) attached to the user’s clothing to minimize the intrusiveness. Therefore, during each of the trials, the phone was carefully carried by hand to simulate a fixed attached sensor tag. Nevertheless, by applying a transformation matrix to account for the change, if any, in the phone orientation, both the direction of travel and acceleration in a predefined reference frame can still be well retrieved.

### 5.5 Evaluation and Discussion

In this chapter, we have presented a proof-of-concept localisation system for real-time indoor patient movement pattern monitoring. With both the maximum estimation error well under two metres and an average estimation error of 0.47 metres based on step detection (and 0.56 metres based on the tight-coupling sensor fusion mechanism) in our trials, we have achieved the essential part of our research objective.

The results strongly suggest that a highly effective design of real-time indoor location tracking with the desired merits of low cost, less intrusiveness, but high mobility, usability, deployability, and portability is possible based on commodity smartphone-centric technologies and devices. Meanwhile, we believe that this work has also demonstrated a feasible solution to address many of the issues we found in related work.

For future work, one of our priorities is to further adapt our current design to a home-like environment where wireless signal transmission might well be significantly affected by various floor plans, partition walls, and furniture. To this end, we plan to use more BLE sensors to get better signal coverage across different rooms and lower measurement noises, as well as to deploy FSR sensors to provide accurate location information as mentioned in Section 5.4.2. As ZigBee is able to provide a better geographical coverage than BLE, it is also worthwhile to perform tests using ZigBee wireless sensors in a home-like environment.

Moreover, we considered the estimation results might be very unreliable if the patient makes movements with excessive speed or unusual formations (e.g. walk in strides or lamely) which could
cause serious misestimates on step detection. As our experiments were based on a normal subject/user, we plan to perform further tests and evaluations using a wider range of subjects, including for example Parkinson’s disease patients with a tremor issue, as well as using a variety of working patterns, in the future so as to improve the robustness of the step detection algorithm. We also plan to develop a mechanism based on the step detection to automatically record where the patient deploys all the BLE and FSR sensors in the home prior to performing real-time location tracking. By doing so, we could enhance not only the reliability and deployability of our system, but also the usability and portability.
6. Simulation of Patients’ Activities of Daily Living and Telehealthcare Interventions

6.1 Introduction

The designs of the three key monitoring functions, i.e. vital sign, safety, and movement pattern monitoring (as discussed in chapters 3, 4 and 5) in our proposed telemonitoring system emphasised on how to effectively gather/detect important data/events relating to a single patient on the client side using low-cost, less-intrusive, off-the-shelf smartphone-centric technologies. This chapter takes a different perspective, aiming to provide a macro view of possible client-side scenarios based on our proposed system/interventions by which hundreds or dozens of thousands of patients were being simultaneously monitored and were continuing generating requests/workloads to the Service Gateway (Cloud Broker) and App&DB modules. In consideration of the practical constraints on the scale and clinical settings, the activities of daily living (such as walking, sleeping, taking vital sign parameters, falling, visiting GPs/A&E, and receiving tele-consultations) of dozens of thousands of patients, as well as some possible telehealthcare interventions for the patients, were therefore modelled and emulated through discrete-event simulation.

The main contribution of this work is two-fold. The first is the development of the novel simulation models to emulate both a large number of patients (divided into a control group and an intervention group) and their activities of daily living, as well as a number of different stakeholders, including carers, professionals, and tele-consultants, and their interactions with patients through the proposed telemonitoring interventions/system. The second is the modelling of specific telehealthcare scenarios to predict patient behaviours towards the use of healthcare services, such as GP and A&E, after a fall or vital sign abnormalities with and without telemonitoring interventions through simulations. This enables us to formulate evidence for cost-effectiveness of our telemonitoring interventions.

The remainder of this chapter is structured as follows. In section 6.2, we highlight a wide range of literature on a variety of subjects, such as healthcare simulation, and statistics of emergency healthcare service delivery and fall related issues concerning older adults to support our simulation design and modelling. Then in section 6.3, we talk about our simulation design, from conceptual, high-level structure to detailed modelling of different stakeholders (for example patients with long-term conditions) and post-event activities, as well as healthcare utilisation, with and without telemonitoring settings. In section 6.4, we conduct cost-effectiveness analysis by examining the simulation results to explore in particular how telemonitoring helps patients avoid missing the required medical treatments, as well as undertaking unnecessary healthcare attendances, to achieve.
cost savings in public healthcare. Finally, we further evaluate our methods and make arguments about how different factors might well have great impacts on the simulation results, as well as what the proposed future work is to enhance our design.

6.2 Literature Review and Related Work

6.2.1 Healthcare Simulation

The complex nature of healthcare management and system development has made simulation a valuable means to predict possible outcomes for further evaluation. [127] conducted a systematic review of 2,226 papers published in the late 20th century in healthcare and identified 182 papers as relevant to computer simulation modelling in population health and healthcare delivery. Of the 182 papers, 52 per cent (or 94 papers) worked on hospital scheduling and organisation concerning mainly resource availability, utilisation, and waiting time, and only 17 papers focused on costs and economic evaluation.

After an initial search result of around 10 thousand publications during 1952-2007 in healthcare, modelling and/or simulation, [128] made full-text reviews of 342 articles, of which 41 used simulation (including 31 with discrete-event simulation, six with system dynamics, and four with Monte Carlo simulation) as the primary method for research and 26 used simulation as a subsidiary method. The study also found that simulation was dominant in planning and system/resource utilisation. Nevertheless, [129] had a different finding that 60 out of 201 reviewed papers published during 1970-2007 in healthcare simulation focused on health risk assessment using Monte Carlo simulation, whereas only 13 worked on healthcare service planning and 10 on health economic modelling both using discrete-event simulation. With regard to the software/programming language for developing discrete-event simulations, [129] found that Arena [130], Borland Delphi, and Simul8 [131] chosen by six, five, and three studies respectively were the top three most used.

[132-134] all worked on healthcare scheduling issues using discrete-event simulation. The objective of [132] was to compare the performance of two appointment scheduling systems, i.e. open access and overbooking, for outpatient clinics. Discrete-event simulations were performed repeatedly to estimate a set of performance metrics, including overtime work, proportion of unmet demand, in-clinic waiting time, and appointment slot utilisation, in different settings (such as probabilities of no-shows and daily demands) for both systems. A comparison of these two systems based on the derived performance measures concluded that open access could achieve better capacity utilisation, but was inferior to overbooking in terms of undesired results in the other three metrics.
The aim of [133] was to reduce patient waiting time and to improve the overall service delivery and system throughput in emergency departments. Simulations were repeatedly performed to find the optimal number of staff and examination rooms within budgetary constraints by changing the number of physicians, nurses, registration staff, and examination rooms at different time slots in a weekday with statistical distributions of patient arrivals, physician assessments, and activity durations from real observation data. [134] aimed to accurately model the process of an A&E so as to help develop alternative strategies to reduce patients’ length of stay. Data based on the actual time spent by patients in the A&E in three different weeks were collected and used as inputs for the model. The study concluded that although it is difficult to incorporate informal processes (such as the arbitrary time, rather than fixed scheduled time, for staff members to take breaks) into the model, there was excellent agreement between actual data and the predictions made by the model.

The abovementioned literature review provided the evidence that simulations have been widely used in research in healthcare. Nevertheless, to the best of our knowledge, none has attempted to model in-home patients’ activities of daily living and combine such a model with proposed healthcare telemonitoring to assess related resource utilisation and potential cost savings through discrete-event simulation.

6.2.2 Statistics of Emergency Care Service Delivery

[135] conducted a structured interview survey to 103 patients admitted to a stroke unit and found that only 41% of the patients correctly assessed their symptoms to be stroke, and that 59% of the patients would have found an NHS (National Health Service) helpline useful for seeking medical assistance. Studies, such as [136], suggest that to receive treatment within three hours from the onset of a stroke be vital for the patient as it increases the chance of recovery. Nevertheless, [137] estimated that an annual 20-30% excess Emergency Department (ED) mortality rate in Australia can be directly attributed to overcrowding and access block. All these indicate that telemonitoring (such as fall detection and a follow-up tele-consultation call) can help quickly identify the risk/problem of each incident through remote clinical assessment. Based on each patient’s unique conditions, the tele-consultant can refer him/her to the most appropriate healthcare specialist service in time or help avoid an unnecessary healthcare attendance.

To build a simulation model for conducting cost-effectiveness analysis of our proposed healthcare telemonitoring, we first attempted to review studies on the costs and effects (especially, changes in patients’ quality of life and health states) of medical treatments and telehealthcare for patients having a detected/unattended fall or vital sign abnormalities. Although a number of studies reported that a large proportion of patients (80% according to [138], for example) participating in telehealthcare think their quality of life have improved, our such effort proved to be ineffective. At the
time of this writing, no studies revealing data with clear figures on the level of change in patients’ quality of life and health states have been found. To resolve this problem, we therefore proposed a comparative cost-effectiveness analysis approach (section 3.3.3) to estimating possible cost savings and impacts of our proposed telemonitoring interventions on current public healthcare system, rather than to make direct comparisons of costs and effects between two interventions.

According to [139] and [140] published by the Health and Social Care Information Centre, as well as [141][142] and [24] by NHS and NICE, a number of statistical figures that were used in (or relevant to) the modelling of our healthcare simulations are listed as follows:

- 36% of incidents receiving ambulance service managed without need for transport to A&E department [139];
- 5.9% of emergency calls closed by telephone advice [139]; and
- 96.1% of incidents with ambulance vehicle arriving within 19 minutes [139];
- 95.1% of calls receiving ambulance service originated via 999 and 4.9% from the 111 route [139];
- 40% of patients who attend an A&E department are discharged requiring no treatment [141];
- 64.1% of total A&E attendances were self-referral and 10.1% were by Emergency Services [140];
- 2% of all NHS inpatient bed-days and 5% of all emergency medical admissions to hospital were made by heart failure patients [24]; and
- On average, patients presenting with evidence of acute stroke (the so-called FAST positive patients) arrived at hospital 70 minutes after a 999 call was made [142].

With regard to the costs of healthcare services, figures released by [143] and [144] include:

- The mean average episode length of a 111 call was 15 minutes 57 seconds in December 2014 (very similar to the statistics of the remaining months in 2014) [143];
- The mean cost of an emergency ambulance call was £32.90 [144];
- The mean cost of an emergency ambulance incident was £201.10 (£32.90 cost of the ambulance call has been subtracted) [144];
- The total Emergency Department (ED) activity cost was £115.51 (A weighted average of admitted (£147) and not admitted (£106) ED attendances) [144]; and

- The system cost per NHS 111 call for the NHS was £12.26 (based on use of NHS 111 in all pilot sites; procurement ceiling per call specified by the Department of Health was £8) [144].

### 6.2.3 Statistics of Falls among Older Adults and Related Issues

As mentioned in the previous subsection, we proposed a comparative cost-effectiveness analysis approach for estimating the possible cost savings and impacts of our telemonitoring interventions on the current healthcare system. Consequently, falls among older adults were chosen as the main target for our modelling effort in the simulation, as they are imposing increasingly heavy burdens on public healthcare.

Statistics worldwide, such as [145,146], show that each year around one-third of old adults aged 65 or older have at least one fall a year. [146] also indicated that of those who fell, less than 50% talked to their healthcare providers and 20% to 30% suffered moderate to severe injuries, and that 772 thousands (i.e. more than 32%) out of 2.4 million nonfatal falls treated in emergency department had to be hospitalised. These incurred direct medical costs of about 30 billion US dollars in the US in 2012. As for the death rate of unintentional falls, it was about 47.84 per 100 thousand older adults in the US during 2004 and 2010 [147]. Another notable figure regarding falls is that according to a one-year study [148] on ED presentation following falls in a defined geographical area, covering Cardiff city and the neighbouring regions, around 23.4% of all ED presentations actually followed a fall.

### 6.2.4 Statistics of Carers and Internet Usage among Older Adults

According to a research report [149] published by OFCOM in 2014, around 42% of people aged 65 or above in the UK had ever gone online. Another report [150] done by Oxford Internet Institute found that 78% of the UK’s population had access to the Internet in 2013. This report also categorised those who went online but were moderate in both hopes and fears about the Internet as ‘cyber-moderates’, and commented that the ‘cyber-moderates’ were mostly older and retired people and spent on average 11.3 hours per week in 2013 using the Internet at home.

According the [151], there are around seven million carers in the UK and about two thirds of people with dementia live at home and most are supported by unpaid carers.
6.3 Simulation Design

To analyse whether our design of telemonitoring is cost-effective, we intended to produce three sets of data through simulation. The first emphasised on the differences of healthcare service utilisation between patients having and not having telemonitored. As mentioned in section 6.2.2, in our best effort we have not found any literature revealing the effect of each A&E or GP treatment for a detected fall or vital sign abnormalities on patients’ health status, such as quality of life and life expectancy in relation to QALYs and EQ-5D\(^*\). This was why we did not try to model the effect of each urgent care on patients’ health status, but instead their utilisation of healthcare services. For example, some patients immediately after a fall might not be physically able to seek the required medical treatment by themselves, but with telemonitoring these patients would get assistance to attend A&E when necessary. The second set of data provided further cost information regarding whether the change of healthcare service utilisation through telemonitoring intervention could in fact reduce the costs of healthcare service provision under certain circumstances. These included data that helped identify if the savings from reducing unnecessary A&E attendance could recoup the expenses of using telemonitoring. The third were the amount of traffic for both monitoring data and web user requests, which were repeatedly or instantaneously sent from the User Agent or web browsers to the App&DB module in the cloud. The web user requests were the web session workloads generated by the healthcare professionals, emergency tele-consultant team, cares, and patients to review/comment/check on patients’ personal information and telemonitored records stored in the App&DB module.

Figure 6.1 provides the concept diagram concerning the design of our telemonitoring simulation. Through simulation, dozens of thousands of independent patients, each initialised with a number of entity attributes (such as age, gender, the type and number of chronic diseases, and Internet habit) based on probability distributions, are created and then assigned randomly to either the intervention group (having telemonitoring) or control group (not have telemonitoring). Every day each patient in both groups performs his/her unique activities of daily living (such as going to bed, sleeping, walking at home, visiting a healthcare organisation (for example a GP/hospital), receiving a home visit from healthcare professionals, and taking vital sign parameters at different times). These are general events that happen to all patients, whereas a detected fall or having vital sign parameter(s) exceeding certain thresholds, which would occur at a particular time to a certain patient based on a probability distribution, is regarded as an urgent event, requiring further attention.

---

*12 EQ-5D is a standardised instrument for use as a measure of health outcome
As patients in the intervention group are telemonitored at all time (for vital signs, safety, and real-time movements) with monitoring data/alerts being sent to the telemonitor centre via a User Agent, an emergency tele-consultant team at the centre would respond as soon as possible to every urgent event by making a phone call to the associated patient. When necessary, this team would refer the patient in the urgent condition to suitable healthcare emergency services (such as ambulance and A&E). In contrast, urgent events that happen to patients in the control group would be left unattended, except that some of these patients might still be able to seek required healthcare services by themselves, such as to book a GP visit on a coming day or make an immediate A&E visit.

As mentioned in section 2.1.4.4, in the UK healthcare system, community nurses play a central role in supporting telehealthcare and patient self-care. Therefore, in our opinion, further coordination and integration between community care service teams and emergency tele-consultant teams through information sharing or even work sharing is very important. We believed that such coordination and integration could be facilitated by the mobile cloud framework adopted in our design, which can offer the flexibility of enabling either a centralised telemonitoring centre or a distributed one (consisting of sub-centres in several geographical locations each with a community care service team, for example). The former might be more suitable for a telehealthcare system covering a large geographical rural area, such as the VA telehealth, while the latter might be more appropriate for community-based care systems. However, how to form the telemonitor centre and tele-consultant team is outside the scope of this research.
Figure 6.2 gives the high-level structure (class) diagram of our telemonitoring simulation based on Unified Modelling Language (UML). Each rectangle with a name in it represents a classifier (such as a class or a data type) that we defined and created in our simulation. Relationships among the classifiers are represented through inheritance or association. For example, in Figure 6.2, both the ‘Patient’ and ‘Carer’ classes inherit from the ‘Person’ class, and there is a composition association between the ‘Patient’ class (i.e. the container class in this case) and each of the ‘Carer’, ‘CarePlan’, ‘ComplianceFactor’, ‘DiseaseType’, ‘UserAgent’, ‘VitalSignType’, ‘DailyActivityType’, ‘DailySchedule’, and ‘InternetHabit’ classes/data types (i.e. the contained). The lifecycle of the instances/variables of the contained classes/data types is totally dependent on the lifecycle of the instance of the container class. So when an instance of ‘Patient’ is created, it will contain an instance of ‘CarePlan’, for example; and when an instance of ‘Patient’ is removed, its contained instance of ‘CarePlan’ is removed too.

![Figure 6.2 High-level Structure (Class) Diagram of Our Telemonitoring Simulation](image)

There is also an aggregation association between the ‘Patient’ class (i.e. the container) and each of ‘HomeVisitTeam’, ‘TeleConsultantTeam’, ‘HcareProfessional’, ‘GeneralPractitioner’, ‘AandEDepartment’, ‘Hospital’, and ‘TeleMonitorServer’ classes (i.e. the contained). Unlike composition, aggregation does not impose strong lifecycle dependency between an instance of the
container and an instance/variable of the contained. For example, each patient in the intervention group will have an emergency tele-consultant team which at the same time also offers services to other patients in the group. Therefore, the creation and removal of an instance of ‘Patient’ has no linkage with those of an instance of ‘TeleConsultantTeam’.

To represent the abovementioned classes in our simulation, we first tried to adopt agent-based modelling technique, by which each entity in the system was emulated by an autonomous agent in ways that attempt to model the behaviours of each entity and the dynamics of interactions among them under certain circumstances. However, our experiments revealed that when tens of thousands of agents were created, not only did the performance of the system\textsuperscript{*13} drop significantly, but synchronisation among the agents (each running on a thread) also became a serious issue. In comparison, discrete-event modelling attempts to simulate a system mainly based on a sequence of events that happen to certain entities at any specific times, causing the change of system state. Our other experiments suggested that discrete-event modelling technique fit the purpose of our specific simulation scenarios better together with improved performance. Consequently, we opted for employing this technique to model our telemonitoring simulation and to estimate costs and healthcare resource utilisation.

### 6.3.1 Modelling of Patients

To model the ‘Patient’ class in Figure 6.2, we first constructed its superclass, the ‘Person’ class, with five main attributes (or instance fields), including name, age, gender, average wakeup time, and average go-to-sleep time. The values of these attributes were produced by random number generators based on a normal distribution, except that those of name and gender were based on a uniform distribution. However, in order to model older adults, random numbers were generated based on a truncated normal distribution with a mean of 68, a standard deviation (SD) of 3.0, and a minimum age of 65 to represent their ages. With regard to the ‘Patient’ class, the main attributes of its contained classes/data types were defined as follows:

- id: a unique integer number, i.e. the patient id, which represents each individual patient.
- groupId: an integer number of 1 or 2, i.e. the group id, which represents either the control group or intervention group that each patient belongs to.
- complianceFactor: a double number between 81.6 and 99 produced by one-sided Gaussian distribution with a mean of 99 and a SD of 5.0 to represent the patient’s compliance with his

\textsuperscript{*13} \textsuperscript*\textsuperscript{\textsuperscript*}The experiments were implemented in Java Standard Edition Development Kit (JDK) 7u71 release based on MS Windows 7 running on a Dell Optiplex 790 with an Intel i5-2400 CPU at 3.1GHz and 4.0 Gbytes of RAM.
daily routine and care plan. The higher the complianceFactor is, the less the deviations from the patient’ average wakeup/go-to-sleep time and the time for taking vital signs are.

- carer: an instance field that is based on our defined ‘Carer’ class. It was assumed that some patients have one carer who looks after them, whereas the others do not have any carer. In our simulation, a carer would generate web session workloads (as discussed in section 7.3.4.4) to the App&DB module. The modelling of carers is given in section 6.3.3.2.

- internetHabit: an integer number in the range of 0 to 3 to represent the frequency that the patient go online to check his/her monitored data or healthcare professionals’ comments. More discussions about how we modelled patients’ Internet behaviours are given in section 6.3.1.4, as well as the web session workloads generated by the patient in section 7.3.4.4.

- diseaseType: an attribute of our defined ‘DiseaseType’ data type, an ‘enum’ type in Java. The value of diseaseType is used for representing what disease(s) that the patient has. It was assumed that each patient has at least one or a random combination of up to four chronic diseases, including heart failure, Type 1 diabetes, Type 2 diabetes, and COPD, based on a uniform distribution, and that about one in ten patients whose condition was regarded as severe (and the others were less severe).

- carePlan: an instance field that is based on our defined ‘CarePlan’ class to represent the patient’s care plan. A default care plan includes taking body temperature twice a day, having a periodic home visit every 90 days, having a pre-scheduled tele-consultation every 30 days, and visiting a GP every 30 days. Depending on which disease(s) a patient has, he/she needs to take various measurements of other vital signs, i.e. heart rate, oxygen uptake and blood pressure for heart failure; blood glucose and pressure for diabetes; and air volume, heart rate and oxygen uptake for COPD. For a patient in a severe condition, he/she would need to take vital signs three times a day and both the time periods for having a home visit and a tele-consultation would be halved.

- userAgent: an instance field that is based on our defined ‘UserAgent’ class to represent the User Agent module. Every patient in the intervention group has one User Agent, which would generate and send traffic to the App&DB module when performing vital sign monitoring, safety monitoring, and/or movement pattern monitoring. We will talk about the modelling of the User Agent module for the simulation in section 6.3.2.

- dailyActivityType: an attribute that is based on our defined ‘DailyActivityType’ data type, an ‘enum’ type in Java. The value of dailyActivityType is used for representing the activity
that the patient is performing (or is happening to the patient) at any given moment. More details about the defined activities are to be discussed in section 6.3.1.1.

- dailySchedule: an instance field that is based on our defined ‘DailySchedule’ class to store timed events that would happen to the patient later on the current day of the simulation.

The abovementioned configurations for the attributes of the ‘Patient’ class used in our simulation were all based on our assumptions in an attempt to create more real-world-like scenarios. Such a limitation on modelling using assumptions was mainly due to the lack of published research work on these subjects. For the purpose of generating the desired three sets of data as set forth in the very beginning of section 3.1, more details about how we modelled patients’ activities of daily living, deliveries of healthcare services for patients experiencing a fall and a vital sign abnormalities situation are illustrated in the following sub-sections.

6.3.1.1 Modelling of Patients’ Activities of Daily Living

Figure 6.3 depicts the defined activities of daily living that can be performed by (or can happen to) a patient in our simulation. These activities can take place either at home or outside, except that ‘FALL’ and ‘DEATH’ could occur anywhere. All activities are represented by a rectangle with a name in it and are mutually exclusive from one another, meaning that only one activity can happen at any given moment.

![Figure 6.3 Patients’ Activities of Daily Living and their Transitions](image-url)
When the simulation starts, all patients are initialised as sleeping (i.e. performing the activity of ‘SLEEP’). Then each patient would wake up at different time and perform another activity, such as walking at home (i.e. the activity of ‘MOVE’), doing a GP visit, surfing on the Internet, or having a fall, based on pre-scheduled/accidental events, probability distributions, or pre-configured attributes, such as personal care plan and Internet habit. Each arrowed line represents the possibility of a transition from one activity to another. The arrow itself shows the possible direction of the transition.

To simplify the diagram, activities are divided into five groups. A transition from one group to another, for example group one to group two, means that an activity in group one (i.e. in this case the activity of ‘SLEEP’) can be followed by any one activity in group two (such as ‘TEL_CONSULT’ or ‘WEB_SURF’). A transition to and from the same group means that an activity can be followed by another activity in that group. For instance, an A&E visit in group three can be followed by a hospital visit in the same group. ‘HOME_OTHERS’ and ‘OUT_OTHERS’ are defined to represent activities other than those specifically listed in Figure 6.3 in the home and outside environment respectively.

When a patient was created, he/she was assigned an average wakeup time using a normal distribution with a mean of 7:00am and a SD of 36 minutes, as well as an average go-to-bed time with a mean of 9:30pm and a SD of 60 minutes. Based on these two values, on each simulation day the wakeup time and go-to-bed time of each patient were generated by adding or subtracting a normal distributed random number ranging from 100 to 1,900 (in seconds), depending on the value of the compliance factor of each patient.

As mentioned in the previous section, activities, such as tele-consultation, home visits and GP visits can be pre-scheduled based on a patient’s care plan; but an urgent/emergency tele-consultation would always happen to a patient in the intervention group right after he/she experienced a fall or vital sign abnormalities. Such an urgent tele-consultation might refer the associate patient to have an A&E visit when necessary. We also set default time lengths for some activities. For example, we assumed that it would take 20 minutes for taking vital sign parameters, 16 minutes for a pre-scheduled tele-consultation (based on statistics of ‘111’ servers mentioned in section 6.2.2), 30 minutes for a home visit, 90 minutes for a GP visit, and 180 minutes for both an A&E visit and a hospital visit (if not hospitalised).

Although it is common to use prediction techniques, such as Markov chains, hidden Markov models or Bayesian network-based classifiers, to predict a state/event sequence, in which the process moves from one state/event to another, we did not apply any of these techniques to model patients’ ADL due to the complicated relationships among the activities. For instance, some activities, like home visits, were pre-booked based on the availability of the home visit team and the conditions set forth in each patient’s care plan, rather than a conditional probability of another activity. In
comparison, some activities, such as falls, were predicted based on both a Poisson distribution and a uniform distribution, which were independent from all other activities. Accordingly, we believed that the modelling of patients’ ADL could be a complex research subject in its own right.

For our proposed telemonitoring system, we also made the following assumptions to create linkages between the system and the patient’s activities. Firstly, the vital sign monitoring would be switched on every time when the associate patient starts to take vital sign parameters (i.e. ‘TAKE_VITAL_SIGNS’ in Figure 6.3) and then be switched off 20 minutes later when the patient finishes the measurements. Secondly, the safety monitoring for fall detection would be always on after the simulation starts, no matter whether the patient is at home or not. Finally, the movement pattern monitoring would only be on when the patient is at home.

6.3.1.2 Modelling of Falls among Older Adults and related Telemonitoring and Emergency Services

![Figure 6.4 Post-fall Event Execution Algorithm for Intervention Group](image)

Figure 6.4 depicts the post-fall event execution algorithm in our simulation for patients in the intervention group. When a fall event of a patient in the intervention group is detected (i.e. the starting
point of our algorithm in Figure 6.4), an alert is sent to the telemonitor centre immediately, requiring the emergency tele-consultant team to make an urgent tele-consultation phone call to the patient as soon as any tele-consultant in the team is available. To estimate the cost-effectiveness of the allocation and utilisation of a number of dedicated tele-consultants, both the delay time in making this urgent tele-consultation call and the idle time of the calling tele-consultant were recorded. The number of falls was also increased by one, as represented as ‘fall++’ in Figure 6.4. During the call, the tele-consultant would evaluate whether the patient needs urgent medical care. If yes, the tele-consultant would refer the patient to A&E via ambulance service. As a result, an A&E visit event at 70-minute later is added to the patient’s daily schedule. This intervention could avoid the patient missing the required medical care when he/she is unwilling or unable to seek medical care after the fall. If there is no need for urgent medical care, the algorithm is finished.

The abovementioned 70-minute time length was adopted in our simulation based on the average time for an acute stroke patient to arrive at a hospital after an emergency call [142]. As for the urgent tele-consultation call itself, we assumed that it would take eight minutes, i.e. a half of the default time length for a non-urgent one mentioned in section 6.3.1.1.

Then 70 minutes (in the simulation time) after the urgent tele-consultation call, an A&E event is fired to represent the arrival of the patient at an A&E department. Here we assumed that the A&E department is always ready to admit the patient upon his/her arrival and the patient would spend 180 minutes at the A&E before he/she is discharged or hospitalised. For the latter case, a hospital event is added and then executed 180 minutes later after the A&E arrival. The utilisation and costs of emergency healthcare services, such as the number of tele-consultation calls and A&E attendances, were recorded along this post-fall event execution process. For simplicity, we did not take fatal falls into consideration and we assumed that a hospitalised post-fall patient would be discharged 180 minutes later due to no available detailed data on post-fall hospitalisation and related treatments.

As [144] commented that with NHS 111 “patients who request urgent medical care should be assessed and directed to the right service first time”, we therefore suggested that the telemonitor centre for our proposed healthcare telemonitoring be integrated with NHS 111 call centres, and assumed that the urgent tele-consultation call can be modelled based on statistical figures of NHS 111 calls. Consequently, we regarded an urgent tele-consultation call as a NHS 111 call in our simulation and cost-effectiveness evaluation.

Figure 6.5 provides the post-fall event execution algorithm in our simulation for patients in the control group. The main differences between this algorithm and the one for the intervention group are twofold. Firstly, there is no emergency tele-consultant team in the telemonitor centre to telemonitor the fall of the patients in the control group. Secondly, after a fall occurs, the immediate action taken
by the patient is completely dependent on his/her physical conditions and willingness to seek outside assistance. Therefore, a fall may be left unattended and a required A&E attendance could be missed when the patient is unable or unwilling to seek A&E treatment, whilst an unnecessary A&E attendance or unnecessary emergency phone call may be made when the patient is willing to do so. When the patient needs urgent medical care and makes an emergency call by himself/herself, the follow-up events are similar to those following an urgent tele-consultation call to the patient in the interventional group. For simplicity, we assumed that an emergency call can be regarded as a NHS 111 call.

![Figure 6.5 Post-fall Event Execution Algorithm for Control Group](image)

The utilisation and costs of emergency services, such as the number of unnecessary emergency calls and missed A&E attendances, which are different from those previously mentioned for the intervention group, were recorded along the post-fall event execution process for patients in the control group. These figures were then adopted for our comparative cost-effectiveness analysis stated in section 6.4.

To simulate possible scenarios by which dozens of falls might well occur on each day to some of the hundreds of thousands of patients (i.e. older adults aged 65 and over) in both the intervention and control group, a number of mathematical models were established. We used them in our simulation to
predict the variables and states necessary for both the decision and process blocks (represented as diamonds and rectangles respectively) in Figures 6.4 and 6.5. More details about these mathematical models are discussed in the following.

**Predict those who will have a fall on each day**

It is common to see that the Poisson distribution has been used by research, such as [152], to model the occurrences of discrete events, for example diseases and falls, over a period of time (under the conditions that all the events occur independently and randomly and that the probability of the event occurrence remains a constant over time). Consequently, we adopted the Poisson distribution in our simulation to predict the values of several defined events/variables, such as the number of patients having a fall and the number of patients having a fall need urgent medical care, on each simulation day.

According to the statistics mentioned in section 6.2.3, one in three adults aged 65 and over has a fall each year [145,146]. We hence assumed that the probability of fall occurrences among older adults annually is equal to 0.33, i.e. \( p_{\text{fall}} = 0.33 \) in Equation (6-1). Using (6-1) and (6-2), we could estimate the mean number of falls per day, denoted as \( \lambda_{\text{fall}} \), and predict the number of falls on each simulation day, denoted as \( n_{\text{falls}} \). In Equation (6-1), \( n_{\text{patients}} \) is the number of total patients in both groups. In Equation (6-2), ‘poissonDistRandomGen(\( \lambda_{\text{fall}} \)).nextInt( )’ is a function which generates a random integer number based on a Poisson distribution with a mean of \( \lambda_{\text{fall}} \). It was executed once at the beginning of each day in our simulation to predict the number of patients who are going to have a fall on that specific day. The implementation of the ‘poissonDistRandomGen(\( \lambda_{\text{fall}} \)).nextInt( )’ function was based on the ‘PoissonDist’ class of the ‘umontreal.iro.lecuyer.probdist’ package within the SSJ [153] framework, a Java library for stochastic simulation developed by the University of Montreal.

\[
\lambda_{\text{fall}} = \frac{(n_{\text{patients}} \times p_{\text{fall}})}{365} \tag{6-1}
\]

\[
n_{\text{falls}} = \text{poissonDistRandomGen}(\lambda_{\text{fall}}).\text{nextInt}( ) \tag{6-2}
\]

\[
\rightarrow \text{patient}_{\text{fall}} = \text{patient}_{\text{fall}.in.CG} \cup \text{patient}_{\text{fall}.in.IG} \tag{6-3}
\]

\[
\rightarrow n_{\text{falls}} = n_{\text{falls}.in.CG} + n_{\text{falls}.in.IG} \tag{6-4}
\]

Then a number of \( n_{\text{falls}} \) patients, denoted as \( \text{patient}_{\text{fall}} \), were randomly selected from both groups based on a uniform distribution, each of which was assigned a random time flag, generated based on a uniform distribution of time interval between [0, 86399], to signify the event time of ‘FALL’. Through this selection process, these \( n_{\text{falls}} \) patients, i.e. \( \text{patient}_{\text{fall}} \), were identified to represent those who were going to have a fall and then by checking each patient’s group id they were
divided into two groups, i.e. $\text{patients}_{\text{fall in CG}}$ for patients in the control group and $\text{patients}_{\text{fall in IG}}$ for patients in the intervention, as represented by Equation (6-3). In order to predict those who are going to have a fall on each simulation day, we assumed that the maximum number of falls that a patient could have on each day is one. The number of patients who are going to have a fall in both groups on each simulation day are denoted as $n_{\text{falls in CG}}$ and $n_{\text{falls in IG}}$ respectively, as represented by Equation (6-4).

**Predict those who are going to have a fall and need further medical care on each day**

It is based on [146] that about 20% to 30% of falls among older adults need further medical care. We therefore assumed that the probability of nonfatal falls among older adults that need further medical care, denoted as $p_{\text{fall need A&E}}$ in Equation (6-5), is equal to 0.25, which was used for estimating the mean number of patients who have a fall and need further medical care per day, denoted as $\lambda_{\text{fall need A&E}}$.

Then we used a Poisson distribution to predict the number of patients who are going to have a nonfatal fall and need further medical care on each simulation day, denoted as $n_{\text{falls need A&E}}$. A random selection process was also performed to choose a number of $n_{\text{falls need A&E}}$ patients, denoted as $\text{patients}_{\text{fall need A&E}}$, from $\text{patients}_{\text{fall}}$ across two groups based on a uniform distribution. Thus we could predict and identify those who in the control group are going to have a fall and need further medical care, denoted as $\text{patients}_{\text{fall need A&E in CG}}$ and those in the same medical conditions in the intervention group, denoted as $\text{patients}_{\text{fall need A&E in IG}}$. This is represented by Equation (6-7). Meanwhile, through this selection process, the number of these patients in both groups, denoted as $n_{\text{falls need A&E in CG}}$ and $n_{\text{falls need A&E in IG}}$ respectively, was derived to form Equation (6-8).

\[
\begin{align*}
\lambda_{\text{fall need A&E}} &= n_{\text{falls}} \times p_{\text{fall need A&E}} \quad (6-5) \\
n_{\text{falls need A&E}} &= \text{poissonDistRandomGen}(\lambda_{\text{fall need A&E}}).nextInt( ) \quad (6-6) \\
\rightarrow \text{patients}_{\text{fall need A&E}} &= \text{patients}_{\text{fall need A&E in CG}} \cup \text{patients}_{\text{fall need A&E in IG}} \quad (6-7) \\
\rightarrow n_{\text{falls need A&E}} &= n_{\text{falls need A&E in CG}} + n_{\text{falls need A&E in IG}} \quad (6-8) \\
\rightarrow \text{patients}_{\text{fall in CG}} &= \text{patients}_{\text{fall need A&E in CG}} \cup \text{patients}_{\text{fall no need A&E in CG}} \quad (6-9) \\
\rightarrow \text{patients}_{\text{fall in IG}} &= \text{patients}_{\text{fall need A&E in IG}} \cup \text{patients}_{\text{fall no need A&E in IG}} \quad (6-10) \\
\rightarrow \text{patients}_{\text{fall no need A&E in CG}} &= \text{patients}_{\text{fall in CG}} \setminus \text{patients}_{\text{fall no need A&E in CG}} \\
\rightarrow \text{patients}_{\text{fall no need A&E in IG}} &= \text{patients}_{\text{fall in IG}} \setminus \text{patients}_{\text{fall no need A&E in IG}}
\end{align*}
\]

By now, we could also identified those who are going to have a fall but will not need medical care, denoted as $\text{patients}_{\text{fall no need A&E in CG}}$ and $\text{patients}_{\text{fall no need A&E in IG}}$ respectively, in
both the control group and intervention group on each simulation day. They could be represented by Equations (6-9) and (6-10) respectively. In these two equations, ‘\’ is a set symbol used to indicate the elements that belong to the first set, i.e. patients_{fall\_in\_IG} \cdot but not to the second set, i.e. patients_{fall\_in\_IG}.

**Predict those who are going to have a fall and miss/make a necessary A&E attendance on each day**

According to [146], more than 50% of older adults having a fall do not talk to their healthcare providers about it. By taking this figure into consideration, we assumed that when no outside intervention is provided, the probability of patients who have a nonfatal fall need further medical care but do not or cannot seek any medical care, denoted as \( p_{fall\_tend\_to\_ignore\_A&E} \), is 0.2. Then using Equation (6-11) and (6-12), we could derive the mean number of patients who have a fall but tend to miss the needed medical care per day, denoted as \( \lambda_{fall\_tend\_to\_ignore\_A&E} \) when no outside intervention is provided, as well as the predicted number of patients who are going to experience the same situation on each simulation day, denoted as \( n_{falls\_tend\_to\_ignore\_A&E} \).

\[
\lambda_{fall\_tend\_to\_ignore\_A&E} = n_{falls\_need\_A&E} \times p_{fall\_tend\_to\_ignore\_A&E} \\
(6-11)
\]

\[
n_{falls\_tend\_to\_ignore\_A&E} = \text{poissonDistRandomGen}(\lambda_{fall\_tend\_to\_ignore\_A&E})\text{.nextInt(} ) \\
(6-12)
\]

\[
\rightarrow patients_{fall\_tend\_to\_ignore\_A&E} = \\
patients_{fall\_tend\_to\_ignore\_A&E\_in\_CG} \cup patients_{fall\_tend\_to\_ignore\_A&E\_in\_IG} \\
(6-13)
\]

\[
\rightarrow n_{falls\_need\_&\_go\_A&E} = n_{falls\_need\_A&E} - n_{falls\_tend\_to\_ignore\_A&E\_in\_CG} \\
(6-14)
\]

\[
\rightarrow patients_{fall\_go\_A&E\_in\_CG} = \\
patients_{fall\_need\_A&E\_in\_CG} \setminus patients_{fall\_tend\_to\_ignore\_A&E\_in\_CG} \\
(6-15)
\]

\[
\rightarrow patients_{fall\_go\_A&E\_in\_IG} = patients_{fall\_need\_A&E\_in\_IG} \\
(6-16)
\]

Then a number of \( n_{falls\_tend\_to\_ignore\_A&E} \) patients, i.e. patients_{fall\_tend\_to\_ignore\_A&E}, were randomly selected from patients_{fall\_need\_A&E} based on a uniform distribution and thus we could predict and identify patients_{fall\_tend\_to\_ignore\_A&E\_in\_CG} for those in the control group and patients_{fall\_tend\_to\_ignore\_A&E\_in\_IG} for those in the intervention group, both of whom are going to have a fall but tend to miss the needed medical care when no outside intervention is provided. This is represented by Equation (6-13).

Nevertheless, as we assumed that for those who belong to patients_{fall\_tend\_to\_ignore\_A&E\_in\_IG} would eventually receive the needed medical care due to telemonitoring intervention, the number of patients who are going to have a fall and will eventually make a needed A&E attendance, i.e. \( n_{falls\_need\_&\_go\_A&E} \), would equal to \( n_{falls\_need\_A&E} \) minus \( n_{falls\_tend\_to\_ignore\_A&E\_in\_CG} \), as
represented by Equation (6-14). Meanwhile, those who have a fall and eventually have a necessary A&E attendance, denoted as $\text{patients}_{\text{fall,go,A\&E\_in\_CG}}$ and $\text{patients}_{\text{fall,go,A\&E\_in\_IC}}$ for both groups respectively, could be represented by Equation (6-15) and (6-16).

**Predict those who are going to have a fall and make an unnecessary A&E attendance and the aroused costs on each day**

As mentioned in section 6.2.2, 40% of patients who attend an A&E are discharged without treatment [141], if we assumed that this figure also applies to patients having a fall, we could formulate Equations (6-17) and (6-18) to predict the number of patients, denoted as $n_{\text{falls,need\_but\_go,A\&E}}$, who are going to have a fall and make an unnecessary A&E attendance on each simulation day. We also assumed that patients in the intervention group would not make unnecessary A&E attendance because of telemonitoring intervention. Therefore, the number of patients in the control group who are going to have a fall and make unnecessary A&E attendance, denoted as $n_{\text{falls,need\_but\_go,A\&E\_in\_CG}}$, is equal to $n_{\text{falls,need\_but\_go,A\&E}}$, as presented by Equation (6-19). In our simulation, a number of $n_{\text{falls,need\_but\_go,A\&E\_in\_CG}}$ patients, denoted as $\text{patients}_{\text{fall,need\_but\_go,A\&E\_in\_CG}}$, were selected from $\text{patients}_{\text{fall,need\_but\_go,A\&E\_in\_CG}}$ to represent those in the control group who are going to have a fall and make an unnecessary A&E attendance, while the remainder were those who are going to have a fall and do not make an unnecessary A&E attendance, denoted as $\text{patients}_{\text{fall,need\_not\_go,A\&E\_in\_CG}}$. This is represented by Equation (6-20). Then by using Equation (6-21), we could predict and identify those who are going to have a fall and make an unnecessary A&E attendance on each simulation day.

\[
\frac{n_{\text{falls,need\_but\_go,A\&E}}}{n_{\text{falls,need\_but\_go,A\&E}} + n_{\text{falls,need\_but\_go,A\&E\_in\_CG}}} = 0.4 \quad (6-17)
\]

\[
\Rightarrow n_{\text{falls,need\_but\_go,A\&E}} = 0.667 \times n_{\text{falls,need\_but\_go,A\&E}} \quad (6-18)
\]

\[
\Rightarrow n_{\text{falls,need\_but\_go,A\&E\_in\_CG}} = n_{\text{falls,need\_but\_go,A\&E}} \quad (6-19)
\]

\[
\Rightarrow \text{patients}_{\text{fall,need\_but\_go,A\&E\_in\_CG}} = \text{patients}_{\text{fall,need\_but\_go,A\&E\_in\_CG}} \cup \text{patients}_{\text{fall,need\_not\_go,A\&E\_in\_CG}} \quad (6-20)
\]

\[
\Rightarrow \text{patients}_{\text{fall,need\_but\_go,A\&E\_in\_CG}} = \text{patients}_{\text{fall,need\_but\_go,A\&E\_in\_CG}} \setminus \text{patients}_{\text{fall,need\_not\_go,A\&E\_in\_CG}} \quad (6-21)
\]

As mentioned in section 6.2.2, the cost of a weighted average of not admitted A&E attendances, denoted as $c_{\text{per\_unadmitted\_A\&E}}$, was £106 [144]. Accordingly, the total costs of unnecessary A&E attendances, denoted as $c_{\text{total\_unadmitted\_A\&E\_in\_CG}}$, aroused by the patients having a fall in the control group on each simulation day could be estimated by Equation (6-22).

\[
c_{\text{total\_unadmitted\_A\&E\_in\_CG}} = n_{\text{falls,need\_but\_go,A\&E\_in\_CG}} \times c_{\text{per\_unadmitted\_A\&E}}
\]
Predict those who in the control group are going to have a fall and attend A&E by ambulance service on each day

Given that in England 10.1% of total A&E attendances were referred by emergency services [140], we assumed that this figure could also apply to post-fall A&E attendances of patients arriving by ambulance (through emergency services) in the control group and thus we could predict the number of patients, denoted as \( n_{\text{falls.go.A&E.by.amb.in.CG}} \), who in the control group are going to have a fall and attend A&E by ambulance service on each simulation day. This was done using Equations (6-23), (6-24) and (6-25).

\[
\frac{n_{\text{falls.go.A&E.by.amb.in.CG}}}{n_{\text{falls.need.&.go.A&E.in.CG}} + n_{\text{falls.no.need.but.go.A&E.in.CG}}} = 0.101 \tag{6-23}
\]

\[
n_{\text{falls.need.&.go.A&E.in.CG}} = n_{\text{falls.need.A&E.in.CG}} - n_{\text{falls.tend.to.ignore.A&E.in.CG}} \tag{6-24}
\]

\[
\rightarrow n_{\text{falls.go.A&E.by.amb.in.CG}} = 0.101 \times \left(n_{\text{falls.need.&.go.A&E.in.CG}} + n_{\text{falls.no.need.but.go.A&E.in.CG}}\right) \tag{6-25}
\]

Then in our simulation, a number of \( n_{\text{falls.go.A&E.by.amb.in.CG}} \) patients, denoted as \( \text{patients}_{\text{fall.go.A&E.by.amb.in.CG}} \), were randomly selected from \( \text{patients}_{\text{fall.go.A&E.in.CG}} \) based on a uniform distribution to represent those who in the control group are going to have a fall and attend A&E by ambulance on each simulation day.

Predict those who in the control group are going to have a fall and make an unnecessary emergency call and the aroused costs on each day

We assumed that those in the control group who have a fall and attend A&E by ambulance through emergency services must have made an emergency call. Therefore, we could predict the number of emergency calls, denoted as \( n_{\text{falls.call.amb.&.need.A&E.in.CG}} \), to be made by patients in the control group who will receive an ambulance and need to attend A&E by using Equation (6-26).

\[
n_{\text{falls.call.amb.&.need.A&E.in.CG}} = n_{\text{falls.go.A&E.by.amb.in.CG}} \tag{6-26}
\]

According to [139], in England 36% of incidents receiving ambulance service did not need to attend A&E, and 5.9% of emergency calls were closed by telephone advice. Again, we assumed that these could apply to post-fall emergency calls made by patients in the control group, and that an emergency call closed by telephone advice is an unnecessary call. Therefore, we could predict the number of patients in the control group denoted as \( n_{\text{falls.call.amb.but.no.need.A&E.in.CG}} \), who are going to have a fall and receive an ambulance after an emergency call but do not need to attend A&E
on each simulation day by using Equations (6-27), (6-28) and (6-29), as well as the number of patients in the control group denoted as \( n_{\text{falls,unneeded,call, in, CG}} \), who are going to have a fall and make an unnecessary emergency call on each simulation day by using Equations (6-30) to (6-34).

\[
n_{\text{falls,call,amb, but, no, need, A&E, in, CG}} / (n_{\text{falls,call,amb, but, no, need, A&E, in, CG}} + n_{\text{falls,call,amb, & need, A&E, in, CG}}) = 0.36
\]

\[
\rightarrow n_{\text{falls,call,amb, but, no, need, A&E, in, CG}} = 0.5625 \times n_{\text{falls,call,amb, & need, A&E, in, CG}}
\]

\[
\rightarrow n_{\text{falls,call,amb, but, no, need, A&E, in, CG}} = 0.5625 \times n_{\text{falls,go,A&E,by,amb,in,CG}}
\]

\[
n_{\text{falls,unneeded,call,in,CG}} / n_{\text{falls,total,call,in,CG}} = 0.059
\]

\[
\rightarrow n_{\text{falls,unneeded,call,in,CG}} / (n_{\text{falls,unneeded,call,in,CG}} + n_{\text{falls,call,amb, but, no, need, A&E, in, CG}} + n_{\text{falls,call,amb, & need, A&E, in, CG}}) = 0.059
\]

\[
\rightarrow n_{\text{falls,unneeded,call,in,CG}} = 0.0627 \times (n_{\text{falls,call,amb, but, no, need, A&E, in, CG}} + n_{\text{falls,call,amb, & need, A&E, in, CG}})
\]

\[
\rightarrow n_{\text{falls,unneeded,call,in,CG}} = 0.0627 \times ((0.5625 \times n_{\text{falls,go,A&E,by,amb,in,CG}}) + n_{\text{falls,go,A&E,by,amb,in,CG}})
\]

\[
\rightarrow n_{\text{falls,unneeded,call,in,CG}} = 0.098 \times n_{\text{falls,go,A&E,by,amb,in,CG}}
\]

Then in our simulation, a number of patients, denoted as \( n_{\text{patients,fall,unneeded,call,in,CG}} \), were randomly selected from \( n_{\text{patients,fall,non,need, A&E, in, CG}} \) based on a uniform distribution to represent those who in the control group are going to have a fall and make an unnecessary emergency call on each simulation day.

As mentioned in section 6.2.2, the cost of an emergency ambulance call was £32.90 and the cost of a NHS 111 call was £12.26 [144]. We assumed that the cost of an emergency call for our telemonitoring, denoted as \( c_{\text{per,call}} \), is equal to that of a NHS 111 call, i.e. £12.26. Accordingly, the total costs of unnecessary emergency calls, denoted as \( c_{\text{total,unneeded, calls, in, CG}} \), aroused by the patients having a fall in the control group on each simulation day could be estimated by Equation (6-35).

\[
c_{\text{total,unneeded, calls, in, CG}} = n_{\text{falls,unneeded,call,in,CG}} \times c_{\text{per,call}}
\]

\[
c_{\text{total,unneeded, calls, in, CG}} = n_{\text{falls,unneeded,call,in,CG}} \times £12.26
\]

Predict those who in the control group are going to have a fall and miss/have a necessary hospital admission on each day

According to [146], more than 30% of older adults having a nonfatal fall treated in emergency
department had to be hospitalised. Therefore, we assumed that the probability of patients who having a nonfatal fall treated in A&E need to be hospitalised, denoted as $p_{\text{fall, from A&E, to hospital}}$, is 0.3. Then using Equation (6-36) and (6-37), we can derived the mean number of patients who having a fall treated in A&E need to be hospitalised per day, denoted as $\lambda_{\text{fall, need, hospital}}$, as well as the predicted number of patients who are going to experience the same situation on each simulation day, denoted as $n_{\text{falls, need, hospital}}$.

In our simulation, a number of $n_{\text{falls, need, hospital}}$ patients, denoted as $\text{patients}_{\text{fall, need, hospital}}$, were randomly selected from both $\text{patients}_{\text{fall, need, A&E, in CG}}$ and $\text{patients}_{\text{fall, need, A&E, in IG}}$ based on a uniform distribution to represent those who are going to have a fall and need both A&E and hospital care. Thus we could predict and identify these patients in both groups, denoted as $\text{patients}_{\text{fall, need, hospital, in CG}}$ and $\text{patients}_{\text{fall, need, hospital, in IG}}$ respectively, as well as their quantities, i.e. $n_{\text{falls, need, hospital, in CG}}$ and $n_{\text{falls, need, hospital, in IG}}$. These could be represented by Equations (6-38) and (6-39).

$$\lambda_{\text{fall, need, hospital}} = n_{\text{falls, need, A&E}} \times p_{\text{fall, from A&E, to hospital}} \quad (6-36)$$
$$n_{\text{falls, need, hospital}} = \text{poissonDistRandomGen}(\lambda_{\text{fall, need, hospital}}).nextInt(\quad) \quad (6-37)$$
$$\text{patients}_{\text{fall, need, hospital}} = \text{patients}_{\text{fall, need, hospital, in CG}} \cup \text{patients}_{\text{fall, need, hospital, in IG}} \quad (6-38)$$
$$\rightarrow n_{\text{falls, need, hospital}} = n_{\text{falls, need, hospital, in CG}} + n_{\text{falls, need, hospital, in IG}} \quad (6-39)$$
$$\rightarrow n_{\text{falls, need, hospital, in CG}} = n_{\text{falls, miss, hospital, in CG}} + n_{\text{falls, go, hospital, in CG}} \quad (6-40)$$
$$\rightarrow \text{patients}_{\text{fall, go, hospital, CG}} =$$
$$\quad \text{(patients}_{\text{fall, need, A&E, in CG}} \setminus \text{patients}_{\text{fall, tend, to, ignore, A&E, in CG}}) \cap$$
$$\quad \text{patients}_{\text{fall, need, hospital, in CG}} \quad (6-41)$$
$$\rightarrow \text{patients}_{\text{fall, miss, hospital, CG}} =$$
$$\quad \text{patients}_{\text{fall, need, hospital, in CG}} \setminus \text{patients}_{\text{fall, go, hospital, in CG}} \quad (6-42)$$

Since we assumed that patients in the intervention group who have a fall and need hospital care after treated in A&E would always be hospitalised due telemonitoring intervention, we put our main focus on patients in the control group to predict and identify those who are going to have a fall and receive necessary hospital care, denoted as $\text{patients}_{\text{fall, go, hospital, CG}}$, as well as those who miss necessary hospital care, denoted as $\text{patients}_{\text{fall, miss, hospital, CG}}$, by using Equations (6-40), (6-41) and (6-42). In our simulation, these were done by randomly select a number of $n_{\text{falls, go, hospital, in CG}}$ patients, i.e. $\text{patients}_{\text{fall, go, hospital, in CG}}$, from the union of $\text{patients}_{\text{fall, go, A&E, in CG}}$ and $\text{patients}_{\text{fall, need, hospital, in CG}}$ based on a uniform distribution and the remainders were those who are going to have a fall but miss necessary hospital care, i.e. $\text{patients}_{\text{fall, miss, hospital, CG}}$. 
6.3.1.3 Modelling of Vital-Sign-Exceeding-Threshold Events and related Telemonitoring and Emergency Services

How to well define vital sign thresholds for a chronic disease of each individual patient is a complicated issue, involving the medical history and several other factors relating to the patient. For instance, [154] developed a dynamic threshold algorithm to estimate the long-term trend of daily self-measured SpO2 data and found that the accuracy of detecting an exacerbation of COPD was improved in comparison with one using a fixed threshold. However, this kind of work is outside the scope of this research. To facilitate our modelling, we assumed that when a patient was enrolled into the intervention group, appropriate vital sign thresholds were defined in his/her care plan by a team of healthcare professionals who had looked after the patient.

Figure 6.6 depicts the post-vital-sign-exceeding-threshold event execution algorithm in our simulation for patients in the intervention group. It starts when any vital sign parameters of a patient exceed pre-defined thresholds by sending an alert to the emergency tele-consultant team at the telemonitor centre. As soon as a tele-consultant in the team is available after receiving the alert, he/she will make an urgent tele-consultation phone call to evaluate the patient’s health conditions and the need of healthcare services. The remainder of this algorithm is similar to the post-fall event execution algorithm for patients in the same group, except for the additions of two decision blocks and three process blocks. These two decision blocks are used for determining whether a GP visit is necessary and whether the tele-consultation call avoids missing a necessary GP visit, while the three process blocks are for recording the utilisation and costs of the GP visit and for adding and executing the GP visit event. These additions were made based on our assumption that a patient who experiencing a vital-sign-abnormalities event does not need urgent medical care might still need to see a GP.

Other assumptions included that a GP visit is always available to a patient the third working day after the day when the appointment is made, and that a patient might decide not to make a necessary GP visit or might be willing to make a GP appointment after experiencing one time or up to six consecutive times of vital sign exceeding thresholds. The predictions of the number of patients who are going to experience vital signs exceeding thresholds and need/do not need urgent/GP medical care, as well as tend to miss necessary medical care or simply ignore the vital sign issues, were all made based on Poisson distributions. Then to predict and identify who are those patients that are/are not going to experience certain events, such as an A&E visit and a GP visit, random numbers were generated based on a uniform distribution. For simplicity, we did not further consider what might happen to the patient after being treated in a hospital or a GP.
Figure 6.6 Post-vital-sign-exceeding-threshold Event Execution Algorithm for Intervention Group

Figure 6.7 shows the post-vital-sign-exceeding-threshold event execution algorithm in our simulation for patients in the control group. Like the post-fall event execution algorithm for patients in the same group, no emergency tele-consultant team will be responding to the vital-sign-exceeding-threshold event. Whether a patient would make an A&E visit or a GP visit after experiencing vital-sign abnormalities is dependent on the physical conditions and willingness of the patient to seek external assistance in medical care. The main differences between these two algorithms are the additions of five decision blocks and four process blocks in Figure 6.7 to determine whether either a GP or an A&E visit is needed or missed and to estimate the utilisation and costs of the GP or A&E visit. A notable assumption we made here was that a patient who is about to miss a necessary A&E attendance (and a hospital admission) might be willing to book and make a GP visit. In such a case, in addition to the record of a missed A&E visit (and a missed hospital admission), a record of an unnecessary GP visit is also added. Meanwhile, to avoid double counting, even though a patient might experience up to three times of vital sign exceeding thresholds within a day, the number of unnecessary A&E attendance or unnecessary emergency call would only be counted at most one time on that day.
To simulate possible scenarios by which dozens or hundreds of patients would experience vital sign abnormalities on each simulation day in both the intervention and control group, a number of mathematical models were established. We used them in our simulation to predict the variables and states necessary for both the decision and process blocks in Figures 6.6 and 6.7. However, the models established for the below listed prediction tasks are very similar to those discussed in section 6.3.1.2 and hence are not stated again in this section.

- Predict those who will experience vital signs exceeding thresholds and need a GP visit or an A&E attendance on each day;
- Predict those who are going to experience vital signs exceeding thresholds and made an unnecessary GP visit or A&E attendance and the total costs of unnecessary attendances on each day; According to [155], the unit cost per patient contact lasting 11.7 minutes for a GP visit excluding direct care staff costs was £41. As mentioned previously that in England 40% of patients who attend an A&E are discharged without treatment [141], we assumed that this can also apply to patients experiencing vital sign exceeding thresholds (the same as the assumption we made for the fall events.)
– Predict those who in the control group are going to experience vital signs exceeding thresholds and attend A&E by ambulance service, as well as the number and costs of unnecessary emergency calls, on each day; (Again, we adopted the statistics that 36% of incidents receiving ambulance service did not need to attend A&E, 5.9% of emergency calls were closed by telephone advice, and each NHS 111 call costs on average £12.26, to formulate our prediction models.)

– Predict those who in the control group are going to experience vital signs exceeding thresholds and miss/have a necessary hospital admission on each day.

**Generate the times for each patient to take vital sign parameters on each day**

In our prototype system, the vital sign monitoring worked on an on-demand basis. Therefore, in our simulation, we assumed that the event of vital sign exceeding thresholds would only happen when patients were taking their vital sign parameters. To generate the time when each patient takes his/her vital sign parameters on each simulation day, we formulated Equations (6-43), (6-44), (6-45), and (6-46) based on our own assumptions. TIME\_DEVIATION is a constant, but can range from 100 to 1,860 (in seconds) for different patients, depending on each individual’s compliance factor (ranging from 99 to 81.4). $t_{vs,morning}$ is the time for a patient to take the measurements in the morning, whilst $t_{vs,evening}$ is for measurements in the evening. If a patient is in severe long-term conditions, meaning his/her care plan would require him/her to take three times of measurements per day, then he/she would need to take another measurement at around mid-day, represented as $t_{vs,noon}$. The 5,400 seconds in Equation (6-44) equal to one and a half hour and 9,000 seconds in Equation (6-46) equal to two and a half hours, whilst the 46,800 seconds in Equation (6-45) were adopted to represent the time of 13:00 after lunch in the afternoon. $t_{wakeup}$ and $t_{go, to, bed}$ represent the time that this patient wake up and go to bed respectively on the current simulation day. The function ‘normalDistRandomGen( )’ was implemented to generate a random double number based on a normal distribution with a mean of zero and a SD of one.

\[
\text{TIME\_DEVIATION} = (100 - \text{complianceFactor}) \times 100 \quad (6-43)
\]

\[
t_{vs,morning} = t_{wakeup} + 5400 \text{ sec.} + \text{normalDistRandomGen( )} \times \text{TIME\_DEVIATION} \quad (6-44)
\]

\[
t_{vs,noon} = 46800 \text{ sec.} + \text{normalDistRandomGen( )} \times \text{TIME\_DEVIATION} \quad (6-45)
\]

\[
t_{vs,evening} = t_{go, to, bed} - 9000 \text{ sec.} + \text{normalDistRandomGen( )} \times \text{TIME\_DEVIATION} \quad (6-46)
\]
Predict those who will experience vital signs exceeding thresholds and need a GP visit or an A&E attendance on each day

According to [30], the total self-reported number of GP (surgery) visits and emergency department attendances were around 1,643 and 288 respectively among 969 patients aged 70.3, on average, with long-term conditions of heart failure, COPD, or diabetes during a 12-month follow-up period. These suggest that the mean number of GP (surgery) and emergency department service uses per patient annually were 1.7 and 0.3 respectively. We assumed that the abovementioned two annual figures can be applied to our simulation models. Accordingly, the mean number of patients who experience vital signs exceeding thresholds and need a GP visit, denoted as $\lambda_{\text{vs,need,GP}}$, or an A&E attendance, denoted as $\lambda_{\text{vs,need,A&E}}$, could be estimated by using Equations (6-47) and (6-48) respectively, where $n_{\text{patients}}$ refers to the total number of patients in our simulation in both the intervention and control groups. Then on each simulation day, we used Poisson distributions to predict the number of patients who are going to experience vital signs exceeding thresholds and need a GP visit, denoted as $n_{\text{vs,need,GP}}$, or an A&E attendance, denoted as $n_{\text{vs,need,A&E}}$. There could be represented by Equations (6-49) and (6-50) respectively.

As we also assumed that only half of patients who experience vital signs exceeding thresholds need medical care, we could predict the number of patients who are going to experience vital signs exceeding thresholds on each simulation day by using Equation (6-51). Then a number of $n_{\text{vs}}$ patients, denoted as $\text{patients}_{\text{ps}}$, were randomly selected from both groups based on a uniform distribution, each of whom being assigned a random number, generated based on a uniform distribution between $[1, 6]$, to represent how many consecutive times the vital-sign-exceeding-threshold event would happen to him/her. Through this selection process, $\text{patients}_{\text{ps}}$, were identified to represent those who are going to experience vital signs exceeding thresholds on each simulation day and then by checking each patient’s group id, they were divided into two groups, i.e. $\text{patients}_{\text{vs, in,CG}}$ for patients in the control group and $\text{patients}_{\text{vs, in,IG}}$ for patients in the intervention group, as represented by Equation (6-52). The number of patients who are going to experience vital signs exceeding thresholds in both groups on each simulation day are denoted as $n_{\text{vs, in,CG}}$ and $n_{\text{vs, in,IG}}$ respectively to form Equation (6-53).

\[
\lambda_{\text{vs,need,GP}} = (n_{\text{patients}} \times 1.7)/365 \tag{6-47}
\]

\[
\lambda_{\text{vs,need,A&E}} = (n_{\text{patients}} \times 0.3)/365 \tag{6-48}
\]

\[
n_{\text{vs,need,GP}} = \text{poissonDistRandomGen}(\lambda_{\text{vs,need,GP}}).nextInt( ) \tag{6-49}
\]

\[
n_{\text{vs,need,A&E}} = \text{poissonDistRandomGen}(\lambda_{\text{vs,need,A&E}}).nextInt( ) \tag{6-50}
\]

\[
n_{\text{vs}} = (n_{\text{vs,need,GP}} + n_{\text{vs,need,A&E}}) \times 2 \tag{6-51}
\]
Moreover, we assumed that the probability of patients in severe long-term conditions who experience vital signs exceeding thresholds and need further medical care is twice of that of patients in less-severe conditions. Based on this assumption, we could predict the number of patients in both severe and less-severe long-term conditions need a GP visit, denoted as \( n_{\text{vs.severe.need.GP}} \) and \( n_{\text{vs.less.severe.need.GP}} \), due to vital signs exceeding thresholds by using Equations (6-54) and (6-55). The ‘roundup(number)’ function in Equation (6-54) was used to raise the input number to the nearest whole number. Then we could predict and identify those who in both severe and less-severe conditions are going to experience vital signs exceeding thresholds and need a GP visit, denoted as \( \text{patients}_{\text{vs.severe.need.GP}} \) and \( \text{patients}_{\text{vs.less.severe.need.GP}} \) respectively, on each simulation day using uniform distributed random numbers. This could be represented as Equation (6-56). Similarly, the number of patients in both severe and less-severe long-term conditions need an A&E attendance, denoted as \( n_{\text{vs.severe.need.A&E}} \) and \( n_{\text{vs.less.severe.need.A&E}} \), could be estimated by using Equations (6-57) and (6-58). Those who in both severe and less-severe conditions are going to experience vital signs exceeding thresholds and need an A&E attendance, denoted as \( \text{patients}_{\text{vs.severe.need.A&E}} \) and \( \text{patients}_{\text{vs.less.severe.need.A&E}} \) respectively, on each simulation day were predicted and identified by using uniform distributed random numbers, as represented by Equation (6-59).

\[
\begin{align*}
\text{patients}_{\text{vs.severe.need.GP}} &= \text{patients}_{\text{vs.in.CG}} \cup \text{patients}_{\text{vs.in.JG}} \\
\text{patients}_{\text{vs.less.severe.need.GP}} &= \text{patients}_{\text{vs.in.CG}} \cup \text{patients}_{\text{vs.in.JG}} \\
\rightarrow n_{\text{vs.severe.need.GP}} &= n_{\text{vs.in.CG}} + n_{\text{vs.in.JG}} \\
\rightarrow n_{\text{vs.less.severe.need.GP}} &= n_{\text{vs.in.CG}} + n_{\text{vs.in.JG}} \\
\rightarrow n_{\text{vs.severe.need.GP}} &= \text{roundup}(n_{\text{vs.need.GP}} \times 2/3) \\
n_{\text{vs.less.severe.need.GP}} &= n_{\text{vs.need.GP}} - n_{\text{vs.severe.need.GP}} \\
\rightarrow \text{patients}_{\text{vs.severe.need.GP}} &= \text{patients}_{\text{vs.severe.need.GP}} \cup \text{patients}_{\text{vs.less.severe.need.GP}} \\
n_{\text{vs.severe.need.A&E}} &= \text{roundup}(n_{\text{vs.need.A&E}} \times 2/3) \\
n_{\text{vs.less.severe.need.A&E}} &= n_{\text{vs.need.A&E}} - n_{\text{vs.severe.need.A&E}} \\
\rightarrow \text{patients}_{\text{vs.severe.need.A&E}} &= \text{patients}_{\text{vs.severe.need.A&E}} \cup \text{patients}_{\text{vs.less.severe.need.A&E}} \\
\rightarrow \text{patients}_{\text{vs.severe.need.GP}} &= \text{patients}_{\text{vs.severe.need.GP in.CG}} \cup \text{patients}_{\text{vs.severe.need.GP in.JG}} \\
\rightarrow \text{patients}_{\text{vs.less.severe.need.GP}} &= \text{patients}_{\text{vs.less.severe.need.GP in.CG}} \cup \text{patients}_{\text{vs.less.severe.need.GP in.JG}}
\end{align*}
\]
→ \( \text{patients}_{\text{vs_severe_need_A&E}} = \text{patients}_{\text{vs_severe_need_A&E\ in\ CG}} \cup \text{patients}_{\text{vs_severe_need_A&E\ in\ IG}} \)  
\( (6-62) \)

→ \( \text{patients}_{\text{vs_nonsevere_need_A&E}} = \text{patients}_{\text{vs_less_severe_need_A&E\ in\ CG}} \cup \text{patients}_{\text{vs_less_severe_need_A&E\ in\ IG}} \)  
\( (6-63) \)

Predict those who are going to experience vital signs exceeding thresholds and miss/make a necessary GP visit on each day

By referring to our assumption about \( p_{\text{fall\_tend\_to\_ignore\_A&E}} \), we assumed that when no outside intervention is provided, the probability of patients who experience vital signs exceeding thresholds and need further medical care, such as GP or A&E, but do not or cannot seek any medical care, denoted as \( p_{\text{vs\_tend\_to\_ignore\_GP\_AE}} \), is 0.2. By using Equation (6-64) and (6-65), we could derive the mean number of patients who experience vital signs exceeding thresholds but tend to miss the needed medical care per day, denoted as \( \lambda_{\text{vs\_tend\_to\_ignore\_GP\_AE}} \) when no outside intervention is provided, as well as the predicted number of patients who are going to experience the same situation on each simulation day, denoted as \( n_{\text{vs\_tend\_to\_ignore\_GP\_AE}} \).

Then a number of \( n_{\text{vs\_tend\_to\_ignore\_GP\_AE}} \) patients, i.e. \( \text{patients}_{\text{vs\_need\_GP}} \) and \( \text{patients}_{\text{vs\_need\_A&E}} \) based on a uniform distribution. Thus we could predict and identify \( \text{patients}_{\text{vs\_tend\_to\_ignore\_GP\_AE\ in\ CG}} \) for those in the control group and \( \text{patients}_{\text{vs\_tend\_to\_ignore\_GP\_AE\ in\ IG}} \) for those in the intervention group, both of whom are going to experience vital signs exceeding thresholds but tend to miss the needed medical care when no outside intervention is provided. This is represented by Equation (6-66). Meanwhile, based on the grouping information from Equations (6-56) and (6-59), \( \text{patients}_{\text{vs\_tend\_to\_ignore\_GP\_in\_CG}} \) could be divided into \( \text{patients}_{\text{vs\_tend\_to\_ignore\_GP\_in\_CG}} \) and \( \text{patients}_{\text{vs\_tend\_to\_ignore\_A&E\_in\_CG}} \), and \( \text{patients}_{\text{vs\_tend\_to\_ignore\_GP\_in\_IG}} \) into \( \text{patients}_{\text{vs\_tend\_to\_ignore\_GP\_in\_IG}} \) and \( \text{patients}_{\text{vs\_tend\_to\_ignore\_A&E\_in\_IG}} \).

\[ \lambda_{\text{vs\_tend\_to\_ignore\_GP\_AE}} = (n_{\text{vs\_need\_GP}} + n_{\text{vs\_need\_A&E}}) \times p_{\text{vs\_tend\_to\_ignore\_GP\_AE}} \]  
\( (6-64) \)

\[ n_{\text{vs\_tend\_to\_ignore\_GP\_AE}} = \text{poissonDistRandomGen}(\lambda_{\text{vs\_tend\_to\_ignore\_GP\_AE}}).nextInt() \]  
\( (6-65) \)

→ \( \text{patients}_{\text{vs\_tend\_to\_ignore\_GP\_AE}} = \text{patients}_{\text{vs\_tend\_to\_ignore\_GP\_AE\ in\ CG}} \cup \text{patients}_{\text{vs\_tend\_to\_ignore\_GP\_AE\ in\ IG}} \)  
\( (6-66) \)

where \( \text{patients}_{\text{vs\_tend\_to\_ignore\_GP\_AE\ in\ CG}} = \text{patients}_{\text{vs\_tend\_to\_ignore\_GP\_in\_CG}} \cup \text{patients}_{\text{vs\_tend\_to\_ignore\_A&E\_in\_CG}} \)

and \( \text{patients}_{\text{vs\_tend\_to\_ignore\_GP\_AE\ in\ IG}} = \text{patients}_{\text{vs\_tend\_to\_ignore\_GP\_in\_IG}} \cup \text{patients}_{\text{vs\_tend\_to\_ignore\_A&E\_in\_IG}} \)
As we assumed that those who belong to $\text{patients}_{\text{vs.tend.to.ignore,GP,E_in,IG}}$ would eventually receive the needed medical care due to telemonitoring intervention, the number of patients who are going to experience vital signs exceeding thresholds and will eventually make a needed GP visit, i.e. $n_{\text{vs.need & go,GP}}$, would equal to $n_{\text{vs.need & go,GP}}$ minus $n_{\text{vs.tend.to.ignore,GP,E_in,CG}}$, as represented by Equation (6-67). The same applied to $n_{\text{vs.need & go,A&E}}$, represented by Equation (6-68). Those who are going to experience vital signs exceeding thresholds and will eventually have a necessary GP visit or A&E attendance, denoted as $\text{patients}_{\text{vs.go,GP_in,CG}}$ and $\text{patients}_{\text{vs.go,A&E_in,CG}}$ respectively for the control group, as well as $\text{patients}_{\text{vs.go,GP_in,IG}}$ and $\text{patients}_{\text{vs.go,A&E_in,IG}}$ respectively for the intervention groups, could be represented by Equations (6-69), (6-70), (6-71) and (6-72).

$$n_{\text{vs.need & go,GP}} = n_{\text{vs.need,GP}} - n_{\text{vs.tend.to.ignore,GP,in,CG}}$$ (6-67)
$$n_{\text{vs.need & go,A&E}} = n_{\text{vs.need,A&E}} - n_{\text{vs.tend.to.ignore,A&E,in,CG}}$$ (6-68)
$$\text{patients}_{\text{vs.go,GP_in,CG}} = \text{patients}_{\text{vs.need,GP_in,CG}} \backslash \text{patients}_{\text{vs.tend.to.ignore,GP,in,CG}}$$ (6-69)

where $\text{patients}_{\text{vs.need,GP_in,CG}} = \text{patients}_{\text{vs.severe.need,GP_in,CG}} \cup \text{patients}_{\text{vs.less.severe.need,GP,in,CG}}$

$$\text{patients}_{\text{vs.go,GP_in,IG}} = \text{patients}_{\text{vs.need,GP_in,IG}}$$
$$= \text{patients}_{\text{vs.severe.need,GP_in,IG}} \cup \text{patients}_{\text{vs.less.severe.need,GP_in,IG}}$$ (6-70)

where $\text{patients}_{\text{vs.need,A&E_in,CG}} = \text{patients}_{\text{vs.severe.need,A&E_in,CG}} \cup \text{patients}_{\text{vs.less.severe.need,A&E_in,CG}}$

$$\text{patients}_{\text{vs.go,A&E_in,IG}} = \text{patients}_{\text{vs.need,A&E_in,IG}}$$
$$= \text{patients}_{\text{vs.severe.need,A&E_in,IG}} \cup \text{patients}_{\text{vs.less.severe.need,A&E_in,IG}}$$ (6-72)

### 6.3.1.4 Modelling of Patients’ Web Surfing Behaviours

The purpose of modelling patients’ web surfing behaviours is to provide estimated web session workloads generated by patients in the intervention group when they are browsing their personal historical health data or professionals’ comments on diagnoses or questions, or are updating their personal information or leaving messages on the App&DB module of our telemonitoring system. As stated earlier, we used an attribute, i.e. internetHabit of the Patient class, to represent the frequency that each patient goes online to perform the abovementioned activities. We defined internetHabit as zero for the patient who never goes online, one for light use, two for median use, and three for heavy use.

As described in section 6.2.4, around 42% of people aged 65 or above in the UK have ever gone
Accordingly, we assumed that only 42% of our patients, denoted as $n_{\text{patients\_online}}$, in the intervention group would use the Internet and generate workloads to the App&DB module. Meanwhile, based on the concept of normal distribution, we assumed that among those who go online, 68% would be the median users making about $\lambda_{\text{online\_browsing}}$ times of browsing requests (without database access) and $\lambda_{\text{online\_db}}$ times of database related requests per day; 16% would be the light users creating about half of median users’ workloads; and the other 16% would be the heavy users creating about twice of median users’ workloads. These would form Equations (6-73), (6-74), (6-75), and (6-76). Then, we randomly selected $n_{\text{patients\_online\_in\_IG}}$ patients, denoted as $\text{patients\_online\_in\_IG}$, from the intervention group based on a uniform distribution to predict and identify those who are going to go online on each simulation day. By using the internetHabit attribute, we could further predict and identify who were light users, median users, and heavy users, as represented by Equation (7-77) and estimated the mean number of the browsing and database related requests, denoted as $m_{\text{patient\_light\_online\_browsing}}$, $m_{\text{patient\_light\_online\_db}}$, $m_{\text{patient\_median\_online\_browsing}}$, $m_{\text{patient\_median\_online\_db}}$, $m_{\text{patient\_heavy\_online\_browsing}}$, and $m_{\text{patient\_heavy\_online\_db}}$, based on Poisson distributions on each simulation day, as represented by Equations (6-78) to (6-83) respectively.

\[ n_{\text{patients\_online\_in\_IG}} = n_{\text{patients\_in\_IG}} \times 0.42 \]  
(6-73)

\[ n_{\text{patients\_median\_online\_in\_IG}} = n_{\text{patients\_online\_in\_IG}} \times 0.68 \]  
(6-74)

\[ n_{\text{patients\_light\_online\_in\_IG}} = n_{\text{patients\_online\_in\_IG}} \times 0.16 \]  
(6-75)

\[ n_{\text{patients\_heavy\_online\_in\_IG}} = n_{\text{patients\_online\_in\_IG}} - n_{\text{patients\_median\_online\_in\_IG}} - n_{\text{patients\_light\_online\_in\_IG}} \]  
(6-76)

\[ \rightarrow n_{\text{patients\_online\_in\_IG}} = n_{\text{patients\_light\_online\_in\_IG}} \cup n_{\text{patients\_median\_online\_in\_IG}} \cup n_{\text{patients\_heavy\_online\_in\_IG}} \]  
(6-77)

\[ m_{\text{patient\_median\_online\_browsing}} = \text{poissonDistRandomGen}(\lambda_{\text{online\_browsing}})\text{.nextInt(}) \]  
(6-78)

\[ m_{\text{patient\_median\_online\_db}} = \text{poissonDistRandomGen}(\lambda_{\text{online\_db}})\text{.nextInt(}) \]  
(6-79)

\[ m_{\text{patient\_light\_online\_browsing}} = \text{poissonDistRandomGen}(\lambda_{\text{online\_browsing}} \times 0.35)\text{.nextInt(}) \]  
(6-80)

\[ m_{\text{patient\_light\_online\_db}} = \text{poissonDistRandomGen}(\lambda_{\text{online\_db}} \times 0.35)\text{.nextInt(}) \]  
(6-81)

\[ m_{\text{patient\_heavy\_online\_browsing}} = \text{poissonDistRandomGen}(\lambda_{\text{online\_browsing}} \times 2.15)\text{.nextInt(}) \]  
(6-82)

\[ m_{\text{patient\_heavy\_online\_db}} = \text{poissonDistRandomGen}(\lambda_{\text{online\_db}} \times 2.15)\text{.nextInt(}) \]  
(6-83)

Finally, for each patient, a random number was generated within the period of his/her awake time based on a uniform distribution to represent the time when he/she would go online on each simulation day. If the time for a patient’s Internet surfing overlapped with that for another activity,
except for a fall event, A&E visit or hospital visit, a mechanism was developed to delay the time to resolve the overlap. Regarding how we adopted these predicted figures to create user session workloads on the App&DB module, more details are given in section 7.3.4.4.

6.3.2 Modelling of User Agents

As mentioned in section 6.3.1, every patient in the intervention group has one User Agent, which would repeatedly generate and send data to the App&DB module when performing vital sign, safety, and/or movement pattern monitoring. In a real-world-like scenario, a healthcare telemonitoring system can have, for example, dozens of thousands of User Agents, which altogether would place tremendous network traffic and workloads on the App&DB module (or server systems). Therefore, the main purpose of modelling the User Agent in the simulation is to generate synthetic traffic and web user session workloads, and to see if different traffic and workload patterns affect the cost-effective design (such as auto balancing and scaling) of the App&DB module. However, in this section, we focus entirely on the modelling of the User Agent, while leaving the simulation and evaluation of the App&DB module to chapter 7.

Figure 6.8 Process and Data Flow Diagram of User Agent in our Simulation
In our simulation, the User Agent was implemented as an instance field of the Patient class based on our defined ‘UserAgent’ class. Figure 6.8 depicts the process and data flows of the User Agent in the simulation. After the simulation starts, four different process paths can be executed concurrently depending on the occurrences of certain events or activities of daily living, such as ‘TAKE_VITAL_SIGNS’, ‘FALL’, and those only happening at home, as illustrated by Figure 6.3. When ‘TAKE_VITAL_SIGNS’ event occurs, synthetic vital sign monitoring records, including patient id, simulation date and time, and a number of vital sign measurements, are generated based on the value of vitalSignType, an attribute of the Patient class, which indicates each time what vital sign parameters the associate patient should measure. If this happens to be joined by a vital-sign-exceeding-threshold event, an alert for vital sign abnormalities is also created. Similarly, when a ‘FALL’ event occurs, a fall alert, including patient id, simulation date and time, and the longitude and latitude of the patient’s location, is created.

Meanwhile, when the User Agent identifies that the patient is currently staying at home based on the current event or the patient’s activity, synthetic location tracking records, including patient id, simulation date and time, and 150 to 600 coordinates of the patient’s in-home locations during the past 150 to 600 seconds (or 2.5 to 10 minutes), depending on a uniform distributed random number, were packed as one data package and sent to the App&DB module. (When the patient has no any movements for more than a minute, a size-reduced location tracking record containing the time length of this no movement period without any coordinate data is generated.)

In a real telemonitoring system, all the created alerts and monitoring data are packaged and encrypted as HTTP requests and then sent separately to the App&DB module to update the database. The exceptions include that location tracking records generated within a varying time window of 2.5 to 10 minutes are merged together before being sent out, and every alert is also sent to the emergency tele-consultant team and carers using emails and text messages. However, in our simulation, there was no real transmission of alerts and monitoring data. Instead, in order to simulate dynamic web session workloads for the App&DB module, the User Agent output only the data type and simulation time when the alert or monitoring data was about to be uploaded. It was based on our experiments that when the size of data for a MySQL database insertion was less than 8 kbytes (i.e. half of the default page size of a MySQL server), the variations of the average execution time of a MySQL server were less than 10.4%. This was the reason that we did not take the sizes of the alert and monitoring data, ranging from several dozen bytes for an alert for vital sign abnormalities to around 2.6 kbytes for 10-minute location tracking records, into consideration when simulated web session workloads on the App&DB module (especially the database server). More discussions about the effect of the data size, as well as the database processing time, on the performance of the database server in our cloud simulation are given in both sections 7.3.4.2 and 7.4.2.
In reality, when access to the Internet is unavailable, all the alerts and monitoring data are stored on the smartphone and wait to be re-sent to the App&DB module at later time. In our simulation, the network status could be estimated based on a random number with a Poisson distribution; but for the simplicity of this modelling work, we assumed that the network is always working properly. Moreover, to improve data transmission speed and to reduce the size of database tables, we can apply the techniques of both HTTP compression and MySQL table compression. Nevertheless, we do not intend to further address this issue as it is outside the scope of this thesis.

\[ m_{user\_agent\_vs} = \text{the number of times that each patient take his/her vital sign parameters on each day} \]  
\[ m_{user\_agent\_vs\_alert} = \text{the number of times that each patient experiences vital signs parameters exceeding threshold on each day} \]  
\[ m_{user\_agent\_fall\_alert} = \text{the number of fall alerts that each patient has on each day} \]  
\[ m_{user\_agent\_location} = \text{the number of times that each patient’s location tracking records are uploaded on each day} \]

Based on the above discussions and assumptions, we further defined four variables, as represented by Equations (6-84) to (6-87), which were adopted to estimate web session workloads created by each User Agent, as stated in section 7.3.4.4.

6.3.3 Modelling of Other Stakeholders

According to Figure 6.2, there were a number of different stakeholders defined in our simulation, such as home visit team, tele-consultant team, healthcare professionals, and general practitioners. However, in the following sub-sections, we only selectively discuss the modelling of healthcare professionals, carers, and emergency tele-consultant team to demonstrate the designs and assumptions about web session workload generation, as well as resource allocation and utilisation, in our simulation.

6.3.3.1 Modelling of Healthcare Professionals

As depicted by Figure 6.2, the healthcare professional was implemented as the ‘HcareProfessional’ class. The main purpose of modelling healthcare professionals, who are responsible for reviewing the outcomes of the telemonitoring, was twofold. Firstly, we sought to estimate patterns of resource allocation and utilisation in relation to the number of healthcare professionals and how much time they can spend on reviewing the recorded data of each selected patient. Secondly, we aimed to simulate possible web session workloads generated by healthcare professionals when they are conducting the review.
The first defined scenario in our simulation is explained as follows. There were \( n_{\text{prof}} \) healthcare professionals, each responsible for tracing the health conditions of a subgroup of the patients in the intervention group. We assumed that through this grouping assignment, each professional could know his/her patients better. The number of patients in the intervention group and each subgroup were denoted as \( n_{\text{patients.in.JG}} \) and \( n_{\text{patients.per.prof}} \) respectively (when not taking into account that some professionals might have one less patient than the others). The latter could be calculated by using Equation (6-88).

Whenever a fall or a vital-sign-exceeding-threshold event happened to a patient belonging to a subgroup that a certain professional was responsible for, this professional would need to review that patient’s telemonitoring data on the next day. In this context, the \( i \)-th professional would need to review a different number of patients, denoted as \( n_{\text{reviews.prof.i}} \), from other professionals on each simulation day, depending on what happened to his/her responsible subgroup the day before. If on each day, every professional would spend a fixed amount of time, denoted as \( t_{\text{prof}} \), on reviewing, the time available for the \( i \)-th professional to review a patients’ monitoring data, denoted as \( t_{\text{per.review.prof.i}} \), could be estimated by Equation (6-89). Finally, we could estimate the maximum, minimum and average time length for reviewing a patient’s monitoring data, denoted as \( t_{\text{max.review}} \), \( t_{\text{min.review}} \), and \( t_{\text{avg.review}} \) respectively, by using Equations (6-90), (6-91) and (6-92).

\[
\begin{align*}
n_{\text{patients.per.prof}} &= n_{\text{patients.in.JG}}/n_{\text{prof}} \\
t_{\text{per.review.prof.i}} &= n_{\text{reviews.prof.i}}/t_{\text{prof}} \\
t_{\text{max.review}} &= \max(t_{\text{per.review.prof.i}}), \ i = 1..n_{\text{prof}} \\
t_{\text{min.review}} &= \min(t_{\text{per.review.prof.i}}), \ i = 1..n_{\text{prof}} \\
t_{\text{avg.review}} &= \frac{\text{avg}(t_{\text{per.review.prof.i}}), \ i = 1..n_{\text{prof}}}{n_{\text{prof}}} 
\end{align*}
\]

The second scenario was that on each simulation day the total number of reviews on the monitoring data of a small number of selected patients (such as those who required medical care the day before), denoted as \( n_{\text{reviews}} \), would be evenly distributed to all the healthcare professionals. By sharing the reviewing work together, each professional would have a same number of patients, denoted as \( n_{\text{reviews.per.prof}} \) (when not taking into account that some professionals might have one less patient than the others). This was represented by Equation (6-93). We also assumed that each professional would spend at most 15 minutes, denoted as \( t_{\text{max.per.review}} \), on reviewing a patient’s data. For web session workload simulation, we assumed that in general a review, containing \( m_{\text{prof.online.browsing}} \) (defined as Type I requests in section 7.3.4.3) and \( m_{\text{prof.online.db}} \) requests (as Type II requests), which represent the number of browsing and database related requests respectively.
The value of $m_{\text{prof\textunderscore online\textunderscore browsing}}$ was generated by a random number between five and eight, as represented by Equation (6-94), while the value of $m_{\text{prof\textunderscore online\textunderscore db}}$ was estimated using (6-95) to ensure that each review could be finished within $t_{\text{max\textunderscore per\textunderscore review}}$. Here $t_{\text{think\textunderscore browsing}}$ and $t_{\text{think\textunderscore db}}$ refer to the thinking times for browsing and database related requests, respectively.

\[
\text{n}_{\text{reviews\textunderscore per\textunderscore prof}} = \text{roundup}(n_{\text{reviews}}/n_{\text{prof}}) \tag{6-93}
\]

\[
m_{\text{prof\textunderscore online\textunderscore browsing}} = 5.8 \tag{6-94}
\]

\[
m_{\text{prof\textunderscore online\textunderscore db}} = (t_{\text{max\textunderscore per\textunderscore review}} - m_{\text{prof\textunderscore online\textunderscore browsing}} \times t_{\text{think\textunderscore browsing}})/t_{\text{think\textunderscore db}} \tag{6-95}
\]

\[
t_{\text{all\textunderscore reviews\textunderscore per\textunderscore prof}} = \sum_i (m_{\text{prof\textunderscore online\textunderscore browsing}} \times t_{\text{think\textunderscore browsing}})_i + (m_{\text{prof\textunderscore online\textunderscore db}} \times t_{\text{think\textunderscore db}})_i \quad i = 1..n_{\text{reviews\textunderscore per\textunderscore prof}} \tag{6-96}
\]

In the simulation, the values of these two think times were dynamically generated by normal distributed random numbers with a mean of 50 seconds and 70 seconds (both with a SD of 20 seconds), respectively. The total review time (denoted as $t_{\text{all\textunderscore reviews\textunderscore per\textunderscore prof}}$) that each professional spent on each day could be estimated by (6-96). To emulate the dynamic reviewing behaviour of each professional, we assumed that between two consecutive reviews of different patients, a professional would spend some time, ranging from zero to ten minutes, on some other things. We also assumed that the maximum time length available for each professional to conduct the reviews was 360 minutes per day. Accordingly, in our simulation, when $t_{\text{prof}}$ reached the limit of 360 minutes or $t_{\text{min\textunderscore review}}$ was less than 15 minutes in the first scenario, or when $t_{\text{all\textunderscore reviews\textunderscore per\textunderscore prof}}$ was greater than 360 minutes in the second scenario, we should increase the number of professionals.

Given that the number of vital-sign-exceeding-threshold events on each day could reach dozens of thousands when one hundred thousand emulated patients were created in our simulation, we found that the first scenario would require an unrealistic large number of healthcare professionals to conduct the reviews. Moreover, as the discussions on the allocation and utilisation of emergency tele-consultants are given in section 6.4.1.3, in this thesis we would only describe how we used the second scenario to estimate web session workloads generated by a default number of 15 healthcare professionals (i.e. $n_{\text{prof}} = 15$) when reviewing several hundred patients’ data on each day. More discussions about how we performed web session representation based on the modelling of healthcare professionals are stated in section 7.3.4.4.

**6.3.3.2 Modelling of Carers**

Similar to the purpose of modelling healthcare professionals, the main objective of modelling carers is to provide estimated web session workloads generated by carers, each looking after a patient
in the intervention group. As mentioned in section 6.2.4, about two-thirds of people with dementia live at home and most of them have a carer [151], and about 78% of the UK population had access to the Internet [150]. By referring to these facts, we assumed that two-thirds of the patients in both groups has one carer, whereas the remaining one-third of patients do not have any carers, and that 78% of the carers would go online and generate workloads on the App&DB module. Accordingly, we derived the number of patients who have a carer, denoted as \( n_{\text{patients, has a carer}} \) by using Equation (6-97). Then random numbers were generated based on a uniform distribution to predicted and identified those in both groups who has a carer, denoted as \( n_{\text{patients, has a carer, in CG}} \) and \( n_{\text{patients, has a carer, in IG}} \), respectively, as represented by Equation (6-98). By examining these identified patients’ group id, we also derived the number of patients in each group, denoted as \( n_{\text{patients, has a carer, in CG}} \) and \( n_{\text{patients, has a carer, in IG}} \) respectively, who has a carer, as represented by Equation (6-99). Meanwhile, we could estimate the number of carers in the intervention group who would create traffic to the App&DB module, denoted as \( n_{\text{caers, online, in IG}} \), using Equation (6-100).

\[
\begin{align*}
  n_{\text{patients, has a carer}} &= n_{\text{patients}} \times 0.667 \\ 
  \rightarrow n_{\text{patients, has a carer}} &= n_{\text{patients, has a carer, in CG}} \cup n_{\text{patients, has a carer, in IG}} \\ 
  \rightarrow n_{\text{patients, has a carer}} &= n_{\text{patients, has a carer, in CG}} + n_{\text{patients, has a carer, in IG}} \\ 
  \rightarrow n_{\text{caers, online, in IG}} &= n_{\text{patients, has a carer, in IG}} \times 0.78 
\end{align*}
\] (6-97) (6-98) (6-99) (6-100)

Regarding the Internet surfing behaviours of the carers who go online, we assumed that they have the same distribution as patients do, meaning that 68% would be the median users making about \( \lambda_{\text{online browsing}} \) (with a default value of one) times of browsing requests and \( \lambda_{\text{online db}} \) (with a default value of one) times of database related requests per day; 16% would be the light users creating about half of median users’ workloads; and the other 16% would be the heavy users creating about twice of median users’ workloads. This would lead to the formulation of Equations (6-101), (6-102), and (6-103). Then by using random numbers based on a uniform distribution, we could further predict and identify who are light users, median users, and heavy users, as represented by Equation (6-104) and estimated the mean number of the browsing (defined as Type I requests in section 7.3.4.3) and database related requests (as Type II requests), denoted as \( m_{\text{carers, light, online browsing}} \), \( m_{\text{carers, medium, online browsing}} \), \( m_{\text{carers, heavy, online browsing}} \), and \( m_{\text{carers, heavy, online db}} \), based on a Poisson distribution on each simulation day, as represented by Equations (6-105) to (6-110) respectively.

\[
\begin{align*}
  n_{\text{carers, median, online in IG}} &= n_{\text{caers, online in IG}} \times 0.68 \\ 
  n_{\text{carers, light, online in IG}} &= n_{\text{caers, online in IG}} \times 0.16
\end{align*}
\] (6-101) (6-102)
Finally, for each carer, a random number was generated within the time period between 08:00 and 22:00 based on a uniform distribution to represent the time when he/she would go online on each simulation day.

6.3.3 Modelling of Emergency Tele-Consultant Team

The tele-consultant team was implemented as the ‘TeleConsultantTeam’ class, as shown in Figure 6.2, with two main roles. The first role was to make pre-booked tele-consultation phone calls to patients in both groups based on each individual patient’s care plan. The tele-consultants responsible for this kind of pre-booked tele-consultation calls were assumed to work eight hours a day, five days a week. The second was to make urgent/emergency tele-consultation phone calls to evaluate the physical conditions and needs of medical care of the patients in the intervention group, who have just experienced vital signs abnormalities or a fall. The group responsible for these urgent tele-consultations was called emergency tele-consultant team and we assumed that some of them work round the clock. In order to properly respond to the high surge of vital-sign-exceeding-threshold events during the daytime on each simulation day, more emergency tele-consultants were needed during the daytime.

Since the service time and inter-arrival time for the pre-booked tele-consultation calls were predictable or sometimes assumed to be fixed, the relation between the allocation and utilisation of the tele-consultants could be well predicted using a mathematic model. Nevertheless, the dynamic and unpredictable nature of the inter-arrival times of two different types of telemonitoring alerts makes the prediction of the relation between the allocation and utilisation of the emergency tele-consultants become more complicated. (The inter-arrival times of vital-sign-exceeding-threshold events were not exponentially distributed, but those of fall events were.) Consequently, we believed that simulation
was a better solution to address the issue. For the purpose of this thesis, we only discuss the modelling of the emergency tele-consultant team in this section.

As we assumed that the emergency tele-consultant team is dedicated to make urgent tele-consultation calls in response to telemonitoring alerts, we therefore attempted to explore whether under certain circumstances there is an optimal number of emergency tele-consultants in the team, in terms of having both the shortest mean waiting time and the shortest mean idle time. The waiting time was defined as the time length between the arrival of an alert and the start of a corresponding urgent tele-consultation call, while the idle time was defined as the time length between two consecutive tele-consultation calls made by the same tele-consultant. Another assumption we made was that an urgent tele-consultation call would last eight minutes, i.e. about half of the average length of a NHS 111 call as mentioned in section 6.3.1.2, not only because the call itself is urgent, but also because the telemonitoring data and pre-stored personal medical data might well help the tele-consultant better profile the patient and his/her conditions.

To model the emergency tele-consultant team, we adopted a multiple-queue, multiple-server system. As illustrated by Figure 6.9, this queuing system contains \( n \) servers (i.e. the tele-consultants) for providing the required services (i.e. making tele-consultation calls), each of which equips with one First-In-First-Out (FIFO) queue for storing the incoming service requests (i.e. the alerts). All the requests are evenly distributed to these queues based on the Round-robin scheduling policy. We assumed that each queue has literally infinite capacity, and that when its associate server becomes available, the request with the earliest arrival time in the queue will immediately be served.

![Figure 6.9 Modelling of Emergency Tele-Consultant Team Based on Queuing Theory](image)

In the diagram, \( t_{a,i,j} \) denotes the arrival time of the \( i \)-th service request stored in the \( j \)-th queue. Meanwhile, \( t_{s,i,j} \) is the time when the \( j \)-th server starts to serve the \( i \)-th request, and \( t_{d,i,j} \) is the time when the \( j \)-th server finishes the service for the \( i \)-th request. Accordingly, the waiting time for the \( i \)-th
request, denoted as $t_{\text{request\_waiting},i}$, could be derived by Equation (6-11). If we assumed that the $i$-th and $k$-th requests are served consecutively by the $j$-th server, we could estimate the idle time of the $j$-th server between serving these two requests, denoted as $t_{\text{server\_idle\_between},i\_and\_k,j}$, by Equation (6-12).

$$t_{\text{request\_waiting},i} = t_{s,i,j} - t_{a,i,j} \quad (6-11)$$
$$t_{\text{server\_idle\_between},i\_and\_k,j} = t_{s,k,j} - t_{d,i,j} \quad (6-12)$$

When a server (for example the $j$-th server) only works during the daytime, its first idle time, denoted as $t_{\text{server\_first\_idle},j}$, before serving the first request (for example the $l$-th request) on a simulation day can be calculated by Equation (6-13), where $t_{s,l,j}$ is the time that the $j$-th server starts providing service to the $l$-th request, and $t_{\text{server\_start\_time},j}$ is the time that the $j$-th server starts working on that day. Similarly, the last idle time of the $j$-th server, denoted as $t_{\text{server\_last\_idle},j}$, can be derived by Equation (6-14), where $t_{\text{server\_finish\_time},j}$ is the time that the $j$-th server finishes working on that day and $t_{d,m,j}$ is the time that it completes the service for the last request, for example the $m$-th request. The function ‘$\max(\cdot)$’ was used to ensure that no negative value would be generated if the last service is finished overtime.

$$t_{\text{server\_first\_idle},j} = t_{s,l,j} - t_{\text{server\_start\_time},j} \quad (6-13)$$
$$t_{\text{server\_last\_idle},j} = \max(0, t_{\text{server\_finish\_time},j} - t_{d,m,j}) \quad (6-14)$$

### 6.4 Cost-effectiveness Analysis

<table>
<thead>
<tr>
<th>Total Number of Patients</th>
<th>Avg. Age</th>
<th>Control Group</th>
<th>Intervention Group</th>
<th>Female</th>
<th>Male</th>
<th>Severe Conditions</th>
<th>Less-severe Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>100,000</td>
<td>68.9</td>
<td>50,076</td>
<td>49,924</td>
<td>50,339</td>
<td>49,661</td>
<td>10,014</td>
<td>89,986</td>
</tr>
</tbody>
</table>

**Table 6.1 Patients’ Profiles in our Healthcare Simulations**

Based on our design and default set-ups as discussed in section 6.3, the patients’ profiles generated by our healthcare simulations are summarised in Table 6.1. The total number of emulated patients was 100 thousand, of which 50,339 were female and 49,661 were male, and the average age of them was 68.9. These patients were randomly selected into either the control group (having 50,076 patients) or the intervention group (49,924 patients) based on a uniform distribution. Among all patients, 10,014 were grouped as having severe long-term conditions, whilst the remaining 89,986 were grouped as having less-severe long-term conditions. The size of the emulated patient population was chosen as a balance between our concern about the performance of the desktop system for
running the simulations and our consideration in generating a large sample population enough to apply the statistical data from literature review.

Table 6.2 provides the grouping of patients based on their Internet habits. For instance, among all 100 thousand patients, 58,096 did not use Internet, whilst 6,512 were heavy Internet users. These figures were used for modelling web session workloads to the App&DB module.

<table>
<thead>
<tr>
<th>Patients</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of Non-Internet Users</td>
</tr>
<tr>
<td>58,096</td>
</tr>
<tr>
<td>No. of Light Internet Users</td>
</tr>
<tr>
<td>6,640</td>
</tr>
<tr>
<td>No. of Median Internet Users</td>
</tr>
<tr>
<td>28,752</td>
</tr>
<tr>
<td>No. of Heavy Internet Users</td>
</tr>
<tr>
<td>6,512</td>
</tr>
</tbody>
</table>

Table 6.2 Grouping of Patients based on their Internet Habits for Modelling Web Session Workloads

Table 6.3 gives the number of healthcare professionals and the grouping of carers based on their Internet habits. More discussions about the web session workloads generated by the three kinds of Internet users (i.e. patients, carers, and healthcare professionals) are given in section 7.3.4.4.

<table>
<thead>
<tr>
<th>No. of Professionals</th>
<th>Carers for Patients in Intervention Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of Non-Internet Users</td>
<td>No. of Light Internet Users</td>
</tr>
<tr>
<td>15</td>
<td>7,326</td>
</tr>
</tbody>
</table>

Table 6.3 Professionals and Carers for Modelling Web Session Workloads

As the evaluation of the relation between the allocation and utilisation of healthcare resources was set up as one of the main objectives for our healthcare simulation, Table 6.4 provides the default set-ups on the allocation of the emergency tele-consultants. For example, there were three whole-day tele-consultants and 12 daytime tele-consultants (working from 06:00 to 20:00), representing a collective working time of 240 hours per day.

<table>
<thead>
<tr>
<th>Total Number of Emergency Tele-consultants</th>
<th>No. of whole-day (24 hours) Tele-consultants</th>
<th>No. of Daytime (06:00-20:00) Tele-consultants</th>
<th>Collective Working Hours per day</th>
<th>Avg. Time (min.) for Each Tele-consultation Call-</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>3</td>
<td>12</td>
<td>240</td>
<td>8</td>
</tr>
</tbody>
</table>

Table 6.4 Default Set-ups of the Emergency Tele-consultants
Another important aspect in our default set-ups were cost elements. The costs of several healthcare services as tabulated in Table 6.5 were assumed based on our literature review and simulation design as mentioned in both section 6.2 and 6.3. These cost elements were then used for our cost-effectiveness analysis.

<p>| Cost (£) Per | Cost (£) Per Unnecessary | Cost (£) Per Unnecessary | Cost (£) Per- |</p>
<table>
<thead>
<tr>
<th>Telex-consultation Call</th>
<th>Emergency Call</th>
<th>A&amp;E Attendance</th>
<th>GP Visit</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.26</td>
<td>12.26</td>
<td>106</td>
<td>41</td>
</tr>
</tbody>
</table>

Table 6.5 Default Set-ups of the Cost Elements

6.4.1 Results of a 90-Day Simulation based on Default Set-ups

According to our simulation design as stated in detail in section 6.3, there was generally no need for a warm-up period in our simulation, which modelled all patients’ activities of daily living, as well as related events on a daily basis. However, some activities, such as A&E and hospital attendances, might well happened on the next day of a fall or a vital-sign-exceeding-threshold event occurred in the evening. Moreover, we adopted the Poisson distribution to predict the number of different events in concern. Therefore, in order to get a better picture of our estimation results through simulation, we set the default length of our simulation to 90 days. The results and related cost-effectiveness analysis are given in the following.

6.4.1.1 Results of Fall-Event Simulation

![Figure 6.10 Total Number of Falls, Alerts and Cases Requiring Medical Care by Day](image)
In the 90-day simulation, a total of 8,231 falls were recorded. Figure 6.10 provides the total number of falls (i.e. areas in blue), alerts (in red) and cases requiring medical care (in green), respectively, by day. This represented an average of 91.46 falls, as shown by the dotted line, per day, of which 45.67 were happened to patients in the intervention group with fall alerts being sent to the monitor centre (as well as the emergency tele-consultant team) and 11.59 cases required further medical care. Here both the alerts and cases requiring medical care only concern patients in the intervention group.

To further examine at what time of day falls occurred, the distributions of the average number of falls in both groups, fall alerts sent to the monitor centre and cases that required further medical care recorded from the 90-day simulation with a five-minute recording window were depicted, as Figure 6.11. We found that given the 90-day sampling time, the occurrences of falls (i.e. the areas in blue) were almost evenly distributed throughout the time of day with an average of 0.32 falls, as shown by the upper dotted line, per five minutes (or one fall per 15.75 minute). The same distribution could be observed in the occurrences of fall alerts (i.e. areas in red with an average of 0.16 alerts per five minutes, or one alert per 31.53 minutes) and cases requiring medical care (areas in green with an average of 0.04 cases requiring medical care per five minutes, or one case per 124.26 minutes). This would suggest that three whole-day (round the clock) emergency tele-consultants based on our default set-ups be definitely redundant and that one would suffice when only fall events are considered.

![Figure 6.11 Average Number of 90-Day Falls, Alerts and Cases Requiring Medical Care by Time of Day](image-url)
6.4.1.2 Results of Vital-Sign-Exceeding-Threshold Event Simulation

The total number of vital sign measurements taken by both control and intervention groups during the 90-day simulation was 18,890,990, representing an average of 209,900 measurements taken per day. Figure 6.12 shows the distribution of the average number of 90-day measurements taken by time of day with a five-minute recording window.

![Figure 6.12 Average Number of 90-day Vital Sign Measurements Taken (by Both Groups) by Time of Day](image)

The main two bell shaped areas that appear in both the morning and evening referred to the minimum requirement in all patients’ care plans to take vital signs twice a day, whilst the third one that appears around 13:00 was made of one additional measurement taken by every patient in severe long-term conditions. In the simulation, the time for each patient to take vital sign parameters in the morning was randomly generated mainly based on his/her wakeup time on each simulation day using Equations (6-43) and (6-44), and the time for the evening measurements was mainly based on go-to-sleep time, which in comparison had a more stretched period of time. As the total number of measurements taken by all the patients in the morning was equal to that in the evening, a more squeezed shape in the morning resulted in a higher mean number of 5,223 measurements within the period of 08:30 and 08:34 and a lower mean of 3,241 at around 19:00 to 19:04. Meanwhile, the time for patients in severe conditions to take one additional vital sign measurement was randomly generated based on a normal distribution with a mean time of 13:00. This produced a relatively squeezed shape with a mean number of 2,783 measurements at about 13:00 to 13:04.

To estimate the relation between the allocation and utilisation of telemonitoring and healthcare resources for the intervention group, both the number of alerts for vital sign abnormalities and cases
requiring further medical care were also recorded in the simulation (shown in Figure 6.13). The highest mean number of alerts (i.e. the areas in red) sent to the monitor centre during a day was 74.46, which occurred at around 12:55 to 12:59, followed by the second largest mean number of 14.58 alerts at around 08:15 to 08:19, and the lowest mean of 8.82 alerts at around 19:10 to 19:14. These would suggest that more emergency tele-consultants be needed during the time periods when the three surges of vital sign alerts would occur every day to reduce the response time.

Moreover, we also observed that the highest mean number of cases requiring further medical care (i.e. areas in green) during a day was 29.07, which occurred at around 13:00 and 13:04, followed by the second largest mean number of 7.5 cases at around 08:15 to 08:19, and the lowest mean of 2.47 cases at around 19:10 to 19:14. The reason that the highest mean number of both alerts and cases requiring medical care occurred at about 13:00 was because we assumed that only patients in severe conditions are required to take measurements around the mid-day, and that the probability of patients in severe conditions, who experience vital signs exceeding thresholds and need further medical care, is twice of that of patients in less-severe conditions.

6.4.1.3 Allocation and Utilisation of Urgent Tele-consultation Resources

Figure 6.14 provides a different prospect of the allocation and utilisation of emergency tele-consultant team by looking at the number of total arrived alerts, alerts waiting in the queue, and idle tele-consultants when assigned an incoming alerts, respectively, based on the data of the first simulation day. Under the default allocation of three whole-day and 12 daytime tele-consultants as
summarised in Table 6.4, we observed that on each simulation day when each of the three surges of alerts started to emerge (areas in green), all the arrived alerts could be assigned to an originally idle tele-consultant for immediate service. However, after some time, for example 07:50 for the first surge, all the subsequent incoming alerts were placed in the queue waiting to be served (i.e. areas in red). These red areas indicated a shortage of the emergency tele-consultants to serve these surges of alerts in a timely manner. Since most of arrived alerts (i.e. areas in blue) were either placed in the queue or served by an originally idle tele-consultant, most blue areas were covered by either green or red areas. A few exceptions that blue areas could be spotted were alerts arriving at a tele-consultant at exactly the same time when the tele-consultant just finished serving another alert.

Figure 6.14 Number of Total Arrived Alerts, Waiting Alerts, and Idle Tele-Consultants Recorded by Time of the First Simulation Day

The largest number of incoming alerts (all of which were ever placed in the queue) within the five-minute recording window during the day was 76, which arrived during 13:00 and 13:04, while it was 59 during 12:55 and 12:59, or even down to 30 during 12:50 and 12:54. In the morning, the largest number of 22 incoming alerts appeared during 08:15 and 08:19 and all of them were ever placed in the queue, whilst in the evening, the largest number of 13 incoming alerts arrived during 19:15 and 19:19, but only nine were ever put in the queue.

Figure 6.15 shows the average number of total assigned alerts and waiting alerts per tele-consultant, as well as how many times a tele-consultant was originally idle when assigned an incoming alert. For example, on average, each day a whole-day tele-consultant was assigned 73.06 alerts, of which 51.19 alerts were ever placed in the queue before being served. Meanwhile, there were 22.14 times that the tele-consultant was originally idle when assigned an alert, or had been idle
for a while when it was time to get off work.

![Graph showing average number of total assigned alerts, waiting, and idle counts per tele-consultant per day.](image1)

**Figure 6.15** Avg. Number of Total Assigned Alerts, Waiting and Idle Counts Per Tele-Consultant Per Day

Figure 6.15 depicts the average alert waiting time and tele-consultant idle time. No matter to whom (either whole-day or daytime tele-consultants) alerts were assigned, the average maximum waiting time occurred at around 13:30 was slightly less than 115 minutes and the average waiting time (without taking into account those immediately served) was around 31 minutes.

![Graph showing average alert waiting and idle time per day.](image2)

**Figure 6.16** Average Alert Waiting Time and Tele-Consultant Idle Time per Day

As mentioned in section 6.2.2, for patients in some emergency situations, such as having a stroke (which might well cause a fall), it is vital for them to receive treatment within three hours from the onset of the illness. Hence, the average maximum waiting time of 115 minutes would be undesirable.
when we also considered the eight-minute tele-consultation call and 70-minute additional time to arrive at an A&E department, as part of our assumptions. Meanwhile, the average maximum idle time and the average idle time for the three whole-day tele-consultants were 324.42 and 37.23 minutes respectively, whilst for the 12 daytime tele-consultants these two figures were 173.98 and 24.76 minutes respectively. This further confirmed our previous judgment (stated in section 6.4.1.1) that there was no need for three whole-day tele-consultants, even when both fall and vital sign alerts were taken into account.

6.4.1.4 Estimation of Costs and Effectiveness

<table>
<thead>
<tr>
<th>Activity/Event</th>
<th>Total Alert Occurrences</th>
<th>Total Cases Requiring A&amp;E</th>
<th>Cases Requiring A&amp;E in CG</th>
<th>Cases Requiring A&amp;E in IG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fall</td>
<td>8,231</td>
<td>2,050</td>
<td>1,007</td>
<td>1,043</td>
</tr>
<tr>
<td>Vital Sign</td>
<td>189,257</td>
<td>45,786</td>
<td>24,546</td>
<td>21,240</td>
</tr>
<tr>
<td>Total</td>
<td>197,488</td>
<td>47,836</td>
<td>25,553</td>
<td>22,283</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Fall</td>
<td>185</td>
<td>203</td>
<td>1,217</td>
<td>393</td>
<td>32</td>
<td>3,067</td>
</tr>
<tr>
<td>Vital Sign</td>
<td>7,495</td>
<td>4,271</td>
<td>25,563</td>
<td>6,509</td>
<td>230</td>
<td>44,533</td>
</tr>
<tr>
<td>Total</td>
<td>7,680</td>
<td>4,474</td>
<td>26,780</td>
<td>6,902</td>
<td>262</td>
<td>47,600</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Activity/Event</th>
<th>Cases Requiring Hospital care</th>
<th>Missed Hospital care (CG)</th>
<th>Avoided Missing Hospital care (IG)</th>
<th>Cases Requiring GP care</th>
<th>Missed GP care (CG)</th>
<th>Avoided Missing GP care (IG)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fall</td>
<td>630</td>
<td>60</td>
<td>63</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>Vital Sign</td>
<td>13,719</td>
<td>2,197</td>
<td>1,246</td>
<td>13,054</td>
<td>1,358</td>
<td>2,396</td>
</tr>
<tr>
<td>Total</td>
<td>14,349</td>
<td>2,257</td>
<td>1,309</td>
<td>13,054</td>
<td>1,358</td>
<td>2,396</td>
</tr>
</tbody>
</table>

Table 6.6 Estimation of Costs and Effectiveness of Healthcare Telemonitoring

As stated in section 6.3, we were not able to conduct conventional cost-effectiveness analysis due to the lack of available published quantitative data on the clinical and health outcomes (such as quality of life and life expectancy in relation to QALYs and EQ-5D) of urgent medical care for patients having a fall or vital sign abnormalities. Therefore, we took a different approach by adopting a comparative cost-effectiveness analysis. Instead of looking at patients’ health status and aroused costs, we put our emphases on the effects of how telemonitoring can help reduce the number of
patients ignoring/missing required treatments and avoid unnecessary healthcare attendances, as well as lessen the costs of public emergency healthcare.

Table 6.6 provides the estimation of certain costs and effectiveness of healthcare telemonitoring based on the 90-day simulation. Rest on our assumption that 20% of patients who needing further medical care for a nonfatal fall or vital sign exceeding thresholds situation tend not to or are physically not able to seek required treatments, the number of missed A&E attendances by patients in the control group in the 90-day simulation was 7,680 for both falls and vital sign issues, whilst the number of the avoided missing A&E attendances by patients in the intervention group was 4,474. By adopting a statistical probability that 40% of A&E attendances are discharged without treatment, the simulation predicted that the total number of unnecessary A&E attendances from patients in the control group were 26,780 for both issues. Meanwhile, among the 6,902 emergency calls made by patients in the control group, 262 were regarded as unnecessary, while the number of unnecessary urgent tele-consultation calls for patients in the intervention group was 47,600. The predictions also include the utilisation of hospital care for patients with both issues, but those of GP care only for patients with vital sign issues.

<table>
<thead>
<tr>
<th>Activity/Event</th>
<th>Costs (£) of Unnecessary Emergency Call (CG)</th>
<th>Costs (£) of Unnecessary A&amp;E Attendance (CG)</th>
<th>Costs (£) of Unnecessary Tele-consult Calls (IG)</th>
<th>Healthcare Savings (£) Through Telemonitoring</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fall</td>
<td>392.3</td>
<td>129,002</td>
<td>37,601.4</td>
<td>91,792.9</td>
</tr>
<tr>
<td>Vital Sign</td>
<td>2,819.8</td>
<td>2,709,678</td>
<td>545,974.6</td>
<td>2,166,523.2</td>
</tr>
<tr>
<td>Total Costs/Savings</td>
<td>3,212.1</td>
<td>2,838,680</td>
<td>583,576</td>
<td>2,258,316.1</td>
</tr>
</tbody>
</table>

Table 6.7 Estimated Healthcare Savings through Telemonitoring

Table 6.7 provides the estimated costs of unnecessary emergency calls and unnecessary A&E attendances made by patients in the control group, as well as those of unnecessary tele-consultation calls for patients in the intervention groups and the possible savings through telemonitoring. Only by looking at these three cost items, the resultant cost savings (or avoidable expenditures) through telemonitoring were 2,258,316.1 pounds sterling in 90 days. In our opinion, this figure could serve as a strong advocacy for the cost-effectiveness of healthcare telemonitoring.

6.4.2 More on the Allocation and Utilisation of Emergency Tele-Consultants

To address the undesirable lengthy average maximum waiting time of incoming alerts assigned to either the whole-day or daytime tele-consultants, as well as the lengthy average maximum idle time
of the three whole-day tele-consultants, as discussed in section 6.4.1.2, we increased the total number of tele-consultants by two to 17, but decreased the number of whole-day tele-consultant to one. Meanwhile, we shift the working time of the daytime tele-consultants by one hour to the period between 07:00 and 21:00. The collective working time was 248 hours per day, i.e. eight hours more than the default set-ups. These new set-ups for the allocation of emergency tele-consultants are summarised in Table 6.8.

<table>
<thead>
<tr>
<th>Total Number of Emergency Tele-consultants</th>
<th>No. of whole-day (24 hours) Tele-consultants</th>
<th>No. of Daytime (07:00-21:00) Tele-consultants</th>
<th>Collective Working Hours per day</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>1</td>
<td>16</td>
<td>248</td>
</tr>
</tbody>
</table>

Table 6.8 A Modified Set-up with One Whole-day and 16 Daytime Emergency Tele-consultants

Figure 6.17 Avg. Number of Total Assigned Alerts, Waiting and Idle Counts Per Tele-Consultant Per Day

Based on the new set-up, although both the average number of incoming alerts and waiting alerts assigned to the whole-day tele-consultant per day were higher than those based on the default set-ups (see Figure 6.17), the average maximum waiting time for alerts assigned to the whole-day tele-consultant decreased from around 115 to 98.6 minutes (see Figure 6.18). Meanwhile, as the number of daytime tele-consultants increased, the maximum waiting time for alerts assigned to them also decreased from around 115 to 97.18 minutes. However, with this maximum waiting time, the total time needed for a patient in such an extreme case to received required treatment from the onset of the
illness or a fall was still longer than three hours. Regarding the average maximum idle time of the whole-day tele-consultant, it was reduced from 324.42 to 228.13 minutes. For the daytime consultants, the time was down from 173.98 to 117.77 minutes. Based on the new set-ups, the average idle times for these two types of tele-consultants were 24.08 and 21.65 minutes (originally 37.23 and 24.76 minutes) respectively.

![Figure 6.18 Average Alert Waiting Time and Tele-Consultant Idle Time per Day](image)

**6.4.3 Experiments on the Timing for Taking Vital Signs in the Afternoon**

In our default set-ups, the time for patients in severe long-term conditions to take an additional vital sign measurement in the afternoon were randomly generated based on a normal distribution with a mean time of 13:00 in an attempt to represent the time after lunch. This led to highly concentrated measurement data taken around 13:00. As we had not found any published work on this timing issue, we hence tried to further explore the impact of changing this timing set-up on the allocation and utilisation of the emergency tele-consultants. To this aim, we modified Equation (6-45) by replacing the fixed mean value of 46,800 seconds (i.e. 13:00) with a dynamically generated number using each patient’s wake-up and go-to-sleep time on each day as references. Equation (6-115) represents the new formula.

\[ t_{vs,noon} = \left( \frac{t_{wakeup} + t_{go_to_bed}}{2} \right) - 3600 \text{ sec.} + \text{normalDistRandomGen()} \times \text{TIME_DEVIATION} \]  

(6-115)

Figure 6.19 shows the average number of vital sign measurements taken by time of day based on the results of the revised simulation using Equation (6-115). The average maximum number of the vital sign measurements taken at the mid-day fell from 2,783 (as depicted in Figure 6.12 based on the default set-ups) to about 531.3, which occurred during 13:15 and 13:19.
As shown in Figure 6.20, this new set-up also produced a much smaller mean number of 14.2 vital sign alerts by the intervention group around the mid-day, which originally was 74.46 (see Figure 6.13) based on the default set-ups. The mean number of cases requiring medical care was also down to 5.7 (originally 29.07).

We then allocated one whole-day and 12 daytime tele-consultants (see Table 6.9), which altogether made up of 204 working hours, to examine how their utilisation changes accordingly. The results given in Figure 6.21 shows that the average maximum waiting time for alerts assigned to the
whole-day tele-consultant was 65.32 minutes, whilst that was 66.96 minutes for alerts assigned to the daytime tele-consultants. Through these experiments, we observed that the variation of timing for patients to take vital signs on each day has a profound impact on the allocation and utilisation of healthcare resources. Hence, to have better informed data on this timing issue would be vital for further enhancing our modelling work.

<table>
<thead>
<tr>
<th>Total Number of Emergency Tele-consultants</th>
<th>No. of whole-day (24 hours) Tele-consultants</th>
<th>No. of Daytime (06:30-21:30) Tele-consultants</th>
<th>Collective Working Hours per day</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>1</td>
<td>12</td>
<td>204</td>
</tr>
</tbody>
</table>

Table 6.9 A Modified Set-up with One Whole-day and 12 Daytime Emergency Tele-consultants

![Figure 6.21 Average Alert Waiting Time and Tele-Consultant Idle Time per Day](image)

6.4.4 Sensitivity Analysis

In section 6.3, we described in detail the design of our healthcare simulation, in which a number of assumptions were made based on either published data on analogous healthcare issues or our own judgments. Given that our cost-effectiveness analysis depended highly on the number of A&E attendances, missed A&E attendances, and avoided missing A&E attendances, in the following we state the sensitivity analysis we performed by testing two key parameters to see their impact on our models.
Learning from [141], we originally assumed that unnecessary A&E attendances made by patients in the control group would constitute 40% of all A&E attendances made by patients in both groups after having a fall or experiencing vital sign issues. This resulted in the prediction of a notable large number of 26,780 unnecessary A&E attendances in 90 days (based on an emulated sample population of 50,076 patients in the control group). To assess how this assumption affects the prediction of unnecessary A&E attendances, as well as the associated costs, we tested the outcomes with two lower percentages, including 30% (i.e. a reduction by 25%) and 20% (i.e. a reduction by 50%). Accordingly, the predicted results were 17,222 (based on 30%) and 10,051 (based on 20%) unnecessary A&E attendances respectively and their associated costs were 1,825,523 and 1,065,406 pounds sterling respectively. Figure 6.22 shows that there is a near-linear relationship among all output values when we varied the percentage.

Figure 6.22 Changes of Number of Unnecessary A&E Attendances in Sensitivity Analysis

Another key parameter that has a significant effect on the prediction of A&E attendances made by patients in both groups after experiencing vital sign issues is the number of annual A&E attendances per patient. It was based on the findings of [30] that 1.7 was adopted as our default set-up, as represented by Equation (6-47). We conducted two new runs of 90-day simulations by varying this parameter by ±30% (i.e. 1.19 and 2.21) and the results were summarised in Figure 6.23. The predicted number of total patients who need A&E care after experiencing vital sign issues in 90 days were 32,200 (based on 1.19 annual rate) and 59,511 (based on 2.21) respectively, whilst the predicted number of patients in the control group who miss required A&E attendances were 5,398 (based on 1.19) and 9,848 (based on 2.21) respectively.
6.5 Evaluation and Discussion

The main objective of this healthcare simulation was to provide a macro view of possible client-side scenarios so as to enable ourselves to further assess the allocation and utilisation of urgent healthcare services and the potential of cost savings when embracing telemonitoring. Based on our modelling design, not only were one hundred thousand emulated patients with long-term conditions created in the simulations, but also their activities of daily living and dozens of thousands of critical events requiring urgent healthcare services were predicted on each simulation day. The findings clearly suggest that when looking at urgent healthcare services alone for the incidents of falls and vital signs exceeding thresholds, telemonitoring can be highly cost-effective, in terms of preventing a great number of patients from making unnecessary healthcare attendances or missing required medical treatments. The estimated cost savings greatly outmatched the costs of providing additional urgent healthcare services, such as the emergency tele-consultation calls for the intervention groups in particular. For instance, even when we reduced the percentage of unnecessary A&E attendances to total A&E attendances to half of statistical figure published by NHS, the costs resulting from such unnecessary healthcare attendance were still above one million pounds sterling in 90 days, as evidenced in Figure 6.22. We believe that these findings strongly support our research hypothesis.

To the best of our knowledge, this simulation work is the first attempt to model a great number of patients and their critical incidents, such as falls and vital sign exceeding thresholds, together with the inclusion of emulated telemonitoring alerts and post-event healthcare service provision. With all its modelling work, this study has taken a step forward to put the evaluation of healthcare
telemonitoring solutions prior to moving to full scale trials on a more scientific basis and has opened up great opportunities for researchers and medical education institutes to investigate further what telemonitoring can offer through simulation.

However, a number of parameters in our modelling work would need to be well optimised in the future work due to the unavailability of relevant published data and lack of access to clinical data sets already collected by other studies. This might limit the accuracy of our models, as well as the resultant estimations and predictions. Besides, the scenarios we created about how and when patients take vital signs were based on the design of our prototype system, by which the vital sign monitoring worked on an on-demand basis. As such, in our simulation, we assumed that the event of vital sign exceeding thresholds would only happen when patients were taking their vital sign parameters twice or three times a day based on each individual patient’s care plan. With the invention and increasing sophistication of less-intrusive body wearable sensors able to continuously collect desired data from the user, this assumption might not stand very long in some contexts in the near future.

Another limitation in our model was the assumption that the emergency tele-consultants are able to accurately evaluate each patient’s post-event conditions so that they can refer the patient to the most suitable healthcare service/provider. One way to address this issue is to put the accuracy of evaluating patients’ conditions as a parameter and adopt sensitivity analysis to enhance the degree of confidence in the estimation results. It is also worthwhile to explore the possibility of applying prediction techniques, such as Markov chains or Bayesian network-based classifiers, to the prediction the transition of patients’ ADL in a more systematic way.

Lastly, although we did examine the relation between the allocation and utilisation of urgent healthcare services, such as emergency tele-consultation calls, the efficiency and optimal allocation of these services were not our main concern in the simulation. As a result, we adopted the multiple-queue-multiple-server mechanism to model the emergency tele-consults in order to gain the flexibility of applying different allocation policies to different queues/servers at the expense of possible longer maximum waiting time for the incoming alerts. To rectify this problem when efficiency and shorter maximum waiting time of healthcare services are of the main concerns, we believe that a single-queue-multiple-server queuing system should be adopted.
7. Simulation of (Mobile) Cloud Computing for Supporting Large-scale Real-time Telehealthcare Application

7.1 Introduction

To get in-depth understanding of how cloud computing can well support our cost-effective design of real-time home healthcare telemonitoring, this chapter introduces our approach for building and conducting simulations of cloud computing, aiming at enhancing system performance and scalability, while reducing its costs. As mentioned previously in Chapter 3, both the Service Gateway (Cloud Broker) and App&DB modules were built upon a cloud infrastructure. Hence, it is essential to ensure cloud resources are configured and managed properly when there are a large number of User Agent modules and web browser users interconnecting with and generating workloads to the App&DB module simultaneously. To this end, simulations offer the advantage of repeatable and quickly adaptable ways of conducting experiments and evaluations in a less-costly, but better controlled environment prior to live implementation.

The main contribution of our work is three-fold. The first includes both the improvements we made on a popular cloud simulation toolkit and the modelling work on the representations of cloud-based web application and database servers, as well as their workloads based on empirical benchmarking experiments. The second is the novel simulation work to integrate a large amount of web session workloads from a telemonitoring domain with dynamic provisions of cloud resources. The third is the proposed FLUCAS algorithm, which has demonstrated its novelty to enhance cloud-based system performance and scalability and reduce costs in our specific healthcare scenarios.

In the following sections, we first examine the existing work on cloud simulation, as well as studies on performance of web applications and cloud services. Both key factors that have significant implications for deploying scalable applications on cloud computing platform were identified through literature review in section 7.2, and a Java-based cloud simulation toolkit that better suited our requirements was chosen. Then in section 7.3 we discuss in detail how we modelled our work to make use of the chosen cloud simulation toolkit and customised it to fit our very specific telehealthcare scenario. Improvements made on the simulation toolkit itself are also explored (such as making runtime workload assignments possible, rather than purely based on predefined timing and patterns for workload generation, and enabling dynamic HTTP session re-assignments, rather than fixed assignments, for failover and better utilisation of virtual machines in cloud datacentres). In addition, we describe the mechanism we developed to deal with over-capacity requests on both the emulated web application and database servers to effectively model their request-execution behaviours in real-world scenarios. Moreover, we elaborate the details about how the essential elements, such as virtual
machines and dynamic workloads, in support of web-based cloud applications were carefully modelled through a series of benchmarking experiments on a real cloud infrastructure.

Grounded on the results of our simulations together with the knowledge of current public cloud service provisions, in section 7.4 we further go through cost-effectiveness analysis accompanied by sensitivity analysis and summarise our key findings in detail. We also make arguments about how different factors might well have great impacts on the results. Finally, in section 7.5 we talk about how commodity cloud resources and mobile applications when integrated properly could be a desirable solution for cost-effective real-time home telehealthcare and provide our suggestions for future work.

7.2 Literature Review and Related Work

7.2.1 Cloud Simulation Toolkits

As research on cloud computing has gained great popularity in recent years, researchers have also developed a number of simulation tools to facilitate analysis of highly dynamic cloud systems in a better controlled environment. [156] categorised 18 simulation tools for cloud computing into three groups:

- CloudSim [157] and its variants, such as CloudAnalyst [158], CDOSim, EMUSIM, NetworkCloudSim, and TeachCloud;
- Simulation of virtual datacentre: such as DCSim [159], GDCSim, and GreenCloud, iCanCloud, MDCSim [160], and SPECl; and
- Other simulation tools: such as GSSIM, Open Cirrus, Open Cloud Testbed, PerfCould, GroupSim, and SimGrid.

In addition to the abovementioned simulation tools, there are also some similar work and projects [157], e.g. CloudReports, RealCloudSim, CloudAnalyst, CloudSimEx [161], DynamicCloudSim [162], and SimWare [163]. Except that the last one, i.e. SimWare, focuses its design on simulation of datacentres mainly to measure performance and power consumption/costs, the remainder are basically extensions of CloudSim, aimed at providing additional features, such as graphic user interface, report generation, modelling of heterogeneity and uncertainty of computational resource performance, or disk I/O workloads. For the purpose of this thesis, we summarise the main objective and functionality of CloudSim, DynamicCloudSim, CloudSimEx and DCSim in the following.
7.2.1.1 CloudSim

CloudSim [157,164,165] is a Java-based event-driven software framework, in which key cloud components (such as datacentres, hosts, and VMs), workloads and event handling engine, as well as resource (such as CPU and memory) provisioning, allocation and scheduling policies, are all implemented as Java classes in a layered architecture. It was designed to support modelling and simulation of scalable cloud computing environments. Instead of starting from scratch, researchers can model and implement their own use cases and scenarios in a more productive way by re-using and extending the provided classes of CloudSim. As such, there have been an increasing number of studies choosing CloudSim to perform their cloud simulation work, as evidenced by the above descriptions.

Before starting a simulation with CloudSim, researchers can specify CPU capacity, memory space, storage size, and network bandwidth for a list of hosts (representing the physical machines) in one or several datacentres and create a number of specified VMs, each equipped with required resources, in each host. Then workloads are created and submitted through a datacentre broker to dedicated VMs, depending on the load sharing policy, for execution. Here CPU capacity is defined by Million Instructions Per Second (MIPS) and a workload is represented as a cloudlet, which consumes CPU capacity, memory space, storage, and network bandwidth of the running VM. The execution time of a cloudlet in a VM is dependent on the total MIPS it needs and the allotted available MIPS from the VM. A VM can concurrently execute multiple cloudlets based on a time-shared or space-shared scheduling policy.

Although CloudSim provides a relatively comprehensive and extensible simulation framework, the main problem, as we see it, is everything, such as a host, a VM or a workload, needs to be defined and created prior to starting a simulation. Hence we argue that simulations based on CloudSim are not truly runtime dynamic by nature. In section 7.3, we will discuss in more detail about our enhancement to the CloudSim event handling engine to make it truly runtime dynamic.

7.2.1.2 CloudSimEx

CloudSimEx [157,161] is a set of extensions to CloudSim, but currently is not officially supported by the CloudSim team. Its goal was to support three-tier web application modelling by introducing new features, such as web user session workload generation, better logging utilities, disk I/O operation workloads (represented as a new type of cloudlets especially with required Million Input/Output Operations Per Second, or MIOPS), creation/submission of VMs and workloads with a delay, and network latency modelling utilities. A user session is represented by a sequence of combined CPU and disk I/O workloads with fixed idle time (or called “think time”) in between. These
CPU and disk I/O workloads, called asCloudlets and dbCloudlets, are respectively executed in a web application server (i.e. a VM) and a database server (i.e. another VM). As we chose CloudSimEx and CloudSim to model our cloud simulation, more discussions on CloudSimEx and related performance modelling and simulation of three-tier cloud-based applications [166] are given in section 7.3.

7.2.1.3 DynamicCloudSim

DynamicCloudSim [162] aimed to address the instability, such as inhomogeneity of resource provisioning, variations of runtime performance, and failures of task execution, in a shared cloud computing environment. Based on CloudSim, DynamicCloudSim introduced additional functionality, such as the assignment of VMs to different type of hosts, randomisation of the performance of each VM in terms of CPU capacity, I/O and network bandwidth, and probabilisation of a VM being a straggler or failed during execution time. From our prospective, the underlying considerations for these enhancements are important for creating a more real-world simulation scenario and hence are incorporated into our simulation design, as discussed in section 7.3.

7.2.1.4 DCSim

DCSim [159] is a Java-based event-driven datacentre simulator, aimed at providing a simulation tool for research on virtualised datacentre management. More specifically, DCSim emphasised on how different VM allocation/relocation policies can increase host (physical) machine utilisation in one or several datacentres, while at the same time reduce the overall power consumption and SLA violations. In our opinion, the fundamental architecture of DCSim is very similar to that of CloudSim. For example, their implementation of event handling engine and the process of creating and allocating resources to VMs and hosts were very identical. Nevertheless, to better simulate a real-world datacentre, DCSim contained more detailed implementation of physical components in a datacentre, such as racks (each representing a collection of homogeneous hosts) and clusters (each representing a collection of homogeneous racks).

Like CloudSim, DCSim also provides a comprehensive set of Java classes and helpers, and hence can be quickly extended by researchers to model their own simulation scenarios. However, according to our study, the notable distinctions between CloudSim and DCSim are that in the latter a workload for a particular multi-tier application consists of a set of tasks, each running within a single VM, and that each VM can only execute a single task at a time. It was these distinctions that suggested CloudSim be more suitable than DCSim for modelling our cloud simulation, in which a single VM might well need to serve dozens or hundreds of thousands of concurrent requests (i.e. workloads) created by a large number of User Agents and web users.
### 7.2.2 Web Application Performance Benchmarking Tools

There are a number of widely used benchmarking tools, such as RUBiS [167], TPC-W [168] (obsolete in April 2005), TPC-C [169], and SPECweb99 [170], for testing the performance and scalability of web applications. We focused on the first, not only because it is open-source software but also because it simulates workloads based on sessions – the same as our simulation work. RUBiS is an auction website prototype, which consists of three main components, namely the client-browser emulator, web application, and database. The client-browser emulator simulates web user behaviours through three kinds of workload sessions, including browsing, bidding, and selling, and generates the estimated performance, such as end-user response time and throughputs. The web application is implemented using a number of different web application platforms, such as PHP and Java servlets, and the database is based on MySQL.

Although RUBiS has been extensively used in web application benchmarking, [171] argued that the RUBiS workload generator itself can introduce significant errors in measuring end-user response time due to its heavy usage of CPU and memory. For our work, we did not use RUBiS to estimate the end-user response time of our system, but to estimate the utilisation of CPU, memory and disk I/O on both the web application server and database server in order to serve a client request or a database operation. We then used the derived information to model the client workloads in our simulations, as discussed in section 7.3.4.

For our simulation design, one important thing we learned from the TPC-C is that it specifies at least 90% of all on-line transactions of each type, such as making a new order and browsing order status, must have a transaction response time of less than 5 seconds [169]. The TPC-C defines a response time as the time period between the last character of input data being entered by the emulated user and the last character of output being received by the emulated terminal. Though estimation and modelling for network transmission time was outside our simulation domain, we factored this requirement into the design of our auto scaling mechanism, as discussed in section 7.3.5.

### 7.2.3 Related Work

Due to the nature of dynamic and complicated resource provisioning in cloud computing, how to achieve high performance and scalability of a cloud-based application in a cost-effective way constitutes a great challenge for system design and implementation. According to our survey, not many studies have been done in this field to thoroughly address all related issues, such as client request patterns, workload balancing, scaling, failover, response time, costs, and resource allocation, scheduling and utilisation.

138
7.2.3.1 Load Balancing and Auto Scaling

As discussed in section 2.4.1.2, load balancing plays a key role in designing a scalable cloud-based web application. [158] developed CloudAnalyst based on CloudSim to simulate large-scale cloud application. They designed seven simulation scenarios, ranging from a single cloud datacentre with 50 VMs to host a simulated social network application, to three load sharing datacentres each with 25, 50, and 75 VMs respectively. The results suggested that bringing the service closer to users shortens the response time, and that when employing load balancing across different datacentres which possess sufficient capacity to meet the peak demand, service quality can be further improved. [172] proposed a service broker strategy for datacentre selection based on CloudAnalyst. They concluded that in addition to service proximity based routing (i.e. selecting the closest datacentre for each client), choosing a datacentre with less VM costs is more cost-effective, if multiple datacentres in the same region are available. This conclusion seems very straightforward, but we argue that it lacked supporting information on resource allocation in VMs and the resultant response time and success rates in different settings.

[173] stated in their study that due to security reasons, Amazon EC2 disabled many layer 2 capacities, such as IP spoofing, causing traditional techniques, like TCP handoff and direct web server return for employing software load balancers, to be impracticable in EC2 instances. Besides, they commented that HTTP traffic redirection at layer 7 is less scalable because of SSL termination and renegotiation. Another issue for employing a load balancer in an EC2 instance was that they found the effective client throughput is only half of the network interface throughput, for example 400 Mbps when using EC2 m1.small instance with 800 Mbps bandwidth, because the balancer kept receiving and forwarding packets between the client and web server. As for performance, they found that using an m1.small instance as a web server, it can pass a SPECweb2005 [174] bank testing at 1,350 simultaneous sessions. When they implemented a load balancer in GAE to redirect session workloads to the EC2 web server, this figure dropped to 100 because of GAE’s quota limitation.

[175] also found that though some cloud platforms, such as GAE, implicitly offering a hardware-based load balancer, it was difficult to get around their limitations to customise application-specific features. For instance, GAE is only able to handle 10 Mbps in/out or less traffic, as they stated. Though the above studies can serve as good references for our simulation design, none of them addressed the issue of VM scaling out/in for dynamic workloads, which in our opinion is essential for achieving a real scalable design of cloud-based application with cost consideration in mind.

[166] proposed dynamic resource provisioning and load distribution algorithms for deploying three-tier applications across multiple clouds based on their design of CloudSimEx. A comprehensive set of Java classes were implemented for conducting simulations. For example, they implemented a
latency-based routing concept of Amazon Route 53 to route user traffic to the closest datacentre, AWS Elastic Load Balancer with Round-robin algorithm, and AWS Auto Scaling (by which a new VM will be started if all current VMs within a datacentre reach 80% threshold of CPU utilisation and a VM will be stopped if the average CPU utilisation is below 10%) for baseline experiments.

In order to reduce the costs of VM utilisation, [166] suggested that a better approach for load balancing be to distribute user sessions (i.e. workloads) to as few servers as possible (i.e. to maximise the number of stoppable VMs) as long as this does not violate the QoS requirements. They also implemented both a datacentre controller to perform a scaling policy, called Compressed auto scaling policy, with a number of over-provisioned VMs for unexpected workload spikes, and a sticky load balancing policy for stateful applications by assigning all the successive workloads of the same session to the same VM. They concluded that their design made a significant improvement over the baseline in availability and accumulative session delay while maintaining acceptably low costs and network latency.

However, from our point of view, the above load balancing approach to some extent contradicts their scaling policy. Our reason is that such an approach would in most cases leave those over-provisioned VMs idle, even when the other VMs are very busy and response times for client requests are prolonged. Moreover, we argue that it is not necessarily true that stateful applications always require a sticky load balancing policy, because either a browser cookie can be used to store session states [173] or HTTP session state replication can be done to replicate the state to other VMs in the same cluster. In particular, as mentioned in section 2.4.4, we can also use REST-based web framework, which provides a uniform set of stateless operations. By doing so, we can also better handle the failover of VMs and increase the utilisation of a newly launched VM as a result of scaling out. More detailed design of our load balancing mechanism will be presented in section 7.3.5.

7.2.3.2 Virtual Machine Provisioning and Workload Execution

For capacity planning and performance evaluation, both theoretical and empirical studies have been widely undertaken. [176] in their experiments observed that Tomcat running in 8 VMs deploying on a physical node (IBM Blade Cluster HS21) worked much better, in terms of shorter response time and higher overall CPU utilisation, than a single Tomcat server running on the same physical node directly. However, they found that when the number of VMs in the same physical node further increased, for example 16 VMs, both the number of error connections and response time were also higher because the management of a larger number of VMs led to higher pressure on the VM manager.

[177] proposed a dynamic VM allocation algorithm using K-Means clustering algorithm which partitions a dataset into K groups called clusters, each associating with a centre point. In their
implementation, K referred to the number of datacentres, and VMs (each with a predefined geographical location) were created in the nearest datacentre. It concluded that the proposed algorithm could improve the system performance and provide faster access and release of resources than allocating VMs in a host with minimum processing elements. However, it was not stated in [177] what the location of each VM referred to, as in a real-world scenario VMs are located at the site where their hosting datacentre is. Accordingly, we were not able to further evaluate their work.

Among all techniques, queuing theory has been widely used for capacity planning. [178] considered the creation/allocation of a VM among a set of host machines as a job and examined the performance of the Join-the-Shortest-Queue routing and Power-of-two-choices routing together with a proposed myopic MaxWeight scheduling [179]. However, we believed that these resource allocation/scheduling algorithms do not fit our study domain, as firstly we do not have control over the hosts and secondly one of our main focuses is to which VMs we can assign our client workloads, rather than in which host under the consideration of its resource contention and utilisation we should allocate a particular VM.

[180] used a closed system with a network of queues to model a web server serving a fixed number of clients to evaluate the maximum achievable system throughput. Upon receiving the response from the server, the client issued another transaction request after a certain think time (exponentially distributed). The processing time of each transaction in the front-end server or database server was estimated based on exponential distribution. [181] proposed an open queuing model for virtualised multi-tier application. Requests at each tier were processed in a first-come-first-served (FCFS) manner and a request could be dispatched to the next tier only when it had been served by the current tier. Processing time at each tier was assumed to be based on a known distribution. We considered that many of their implementations, such as fixed number of clients and random processing time based on certain distributions, cannot be applied to our simulation. Moreover, these studies that were based on queuing theory all assumed a server can only process one request at a time, whereas in our case a VM can execute hundreds of thousands of cloudlets at a time in a time-sharing manner.

7.2.3.3 Cost and Performance Analysis

[182] conducted a cost and performance analysis of three types of Amazon EC2 instances, including micro, small, and medium, for hosting multi-tier web applications. In their experiments, RUBiS was used as the sample web application and httperf was used to generate workloads. Two scale-out strategies, i.e. scaling out when CPU utilisation exceeds a threshold (the so-called CPU Reactive) or when response time exceeds a threshold (Response Reactive), were employed respectively for comparison. They concluded that micro instances are more cost-effective than the
other two types when using either CPU Reactive or Response Reactive scaling-out strategy. Another notable finding was that they never observed any significant increase in the response time even when micro instances’ CPU utilisation reached 100%, while this did not stand when using the other two types of instances. They attributed this phenomenon to Amazon’s special CPU allocation policy for micro instances, which allows occasional CPU bursts above baseline capacity. By referring to these findings, we hence conducted our simulation design and cost-effective analysis based on EC2 micro instances, as presented in the following sections.

### 7.3 Simulation Design

As depicted by Figure 3.1, our proposed home healthcare telemonitoring system consisted of four modules, i.e. Sensor Node(s), User Agent(s), Service Gateway (or Cloud Broker), and App&DB. The third and fourth modules were implemented on the IaaS cloud platform so that we could better leverage the scalability of cloud computing, while at the same time take more control of cloud resources. This consideration was also in line with the comment made by [175], regarding the difficulty in getting around the limitations imposed by PaaS cloud platform to customise application-specific features.

![Figure 7.1 Cloud-based Multi-Tier Application Architecture](image)

Figure 7.1 shows the architecture diagram of a cloud-based multi-tier application. The “Web
Clients” shown on the upper part of the diagram include numerous User Agents and end-users, such as patients, carers, and healthcare professionals. The remaining part of the diagram depicts the IaaS cloud platform in which load balancers and scalers, as well as web application and database servers, are hosted. The ‘Cloud Broker/Global Load Balancer & Scaler’ serves as an Internet-facing cloud entry point that maintains lists of available cloud resources, distributes client workloads to them, and dynamically performs on-demand scale-out/in across different service regions of one or several cloud service providers. A simple example of this mechanism is Amazon Route 53, which distributes user requests to the applications running in AWS based on Latency Based Routing, Geo-location DNS, or Weighted Round Robin, though it does not have the capability of either routing traffic to other cloud service providers or initiating auto scaling. Depending on each cloud service provider’s infrastructure, a service region could be a datacentre or a geographic area containing multiple datacentres or availability zones, like Amazon does.

The “Regional Load Balancer & Scaler” is responsible for distributing incoming workloads to available application-specific servers (each running on a VM, within an IaaS cloud provider’s service region) and for performing scaling-out/in by adding new VMs/database replication or closing down existing VMs/database replication in the same auto scaling group (circled by dashed lines) when certain pre-defined conditions are met. For security reasons, workloads for databases need to go through web application servers. Here high availability, scalability, performance, and failover of web application servers are achieved by having several identical server instances running on different VMs within a service region or across multiple regions for better disaster tolerance. With regard to the database availability, scalability, performance, and failover, there are many approaches in engineering design. Examples include database replication (for master-slave or active-standby configuration), clustering, and sharding, as well as redundant array of independent disks (RAID) at storage level. Accordingly, to effectively manage heavy workloads, in our design web applications servers and database servers were split into different auto scaling groups.

As mentioned in section 2.4.1.2, Amazon provides separate load balancing and auto scaling services in each service region. Amazon also provides optional multiple-availability-zone deployment (for high availability and failover) and read replicas (for capacity scaling-out) for its Relational Database Service (such as MySQL), as well as auto replicas across multiple availability zones and data sharding (for horizontal scale-out) for its MongoDB (i.e. a NoSQL database) service. To simplify the problem domain, we chose to employ master-slave replication concept for database scaling out, with each MySQL server instance running on a VM in our simulations. Nevertheless, we assumed the overhead of data replication among MySQL servers was very limited and hence was not taken into account during simulations. Besides, as we did not distinguish disk read and write operations from each other when modelling workloads for cloud simulations, we did not intend to further explore the
limitation of read-only transactions by MySQL slaves on database availability, scalability, and performance.

To achieve higher scalability and availability of our cloud-based solution in a cost-effective way, in the following subsections we focused our simulation designs on how we customised the simulation tool to fit our purpose and then attained better solutions for load balancing and auto scaling within a service region. We believed that findings of such simulations can be well extended to cover similar problem domains across multiple service regions of one cloud provider or of several providers.

7.3.1 Runtime Dynamic Simulation

To better simulate the dynamic nature of resource provisioning and unexpected events in a cloud computing environment, it is essential to perform the simulation in a runtime dynamic way. As mentioned in section 7.2.1.1, we conducted our cloud simulation based on CloudSim. However, we found that when using CloudSim, one needs to predefine and complete all the configurations, such as the amount and sizes of workloads, and the pattern and timing of generating and submitting a sequence of workloads, before starting a simulation. Although it was possible to set certain rules (embedded in a Java object) before the start of a simulation, which would then act to change some of these configurations, such as the sizes of workloads, along the execution of the simulation, the whole simulation process was still not truly runtime dynamic.

Table 7.1 Pseudo code snippets of CloudSim main( ) method

The abovementioned issue was mainly caused by the design of CloudSim’s event handling engine, the CloudSim.startSimulation( ) method. As shown in both Table 7.1 and 7.2, once the main( ) method passes control to CloudSim.startSimulation( ), the latter will not stop executing until there is no more future event in the queue. Accordingly, it is possible to predict what are going to be done by CloudSim.startSimulation( ) based on the configurations performed beforehand. Besides, based on the fundamental concept of discrete-event simulation models, the simulation timer will always jump from the current simulation time to the next pre-stored event time when there is no more current event for the simulation entities, such as datacentres and brokers. For example, the timer can jump from the 20th
second to the 55th second when all entities have completed processing their respective events at current simulation time, i.e. the 20th second, and the next soonest event time is 55 seconds. After the timer jumps to 55 seconds, any new occurring events, even created at the 20th second, with event time less than 55 seconds are discarded. This makes runtime dynamic event handling infeasible.

```java
1:   CloudSim.startSimulation( )
2:   startEntities( );
3:   do
4:      entities check and process received events in referredEventQueue
5:     for event in futureEventQueue do
6:        futureEventQueueNotEmpty ← true;
7:        simulationTimer ← event.getTime( );
8:        if event.getType( ) == CREATE_ENTITY then
9:           createNewEntity(event.getData( ));
10:       else if event.getType( ) == SEND_EVENT then
11:          add event to referredEventQueue;
12:       end if
13:   end for
14:   while futureEventQueueNotEmpty == true
15: end CloudSim.startSimulation
```

Table 7.2 Pseudo code snippets of CloudSim.startSimulation( ) method

To make our simulation truly runtime dynamic, we re-wrote CloudSim simulation engine as the RuntimeCloudSim.runSimulation( ) method, as shown in Table 7.3, which relinquishes the control back to the caller, e.g. the main( ) method, every time when all entities have completed processing their respective events at current simulation time or when the next event time is greater than the current simulation time. Moreover, we could have two options about how to update the simulation timer depending on our simulation requirements. The first option was to let the new simulation timer count up a definable interval, for example 0.001 seconds depending on the value of the minimumTimeBetweenEvents variable, at a time when the variable isDiscreteTimeFeatureSet is set to true (line 5 in Table 7.3). Therefore, it is possible to dynamically create new events and submit new workloads at any specified time in the future, or based on certain conditions or outputs of other concurrent running programmes, after the simulation has started. The second one was to allow the simulation timer to jump to the next event time (line 15 in Table 7.3), the same as the original design of CloudSim, but using the while loop of the main( ) method as a timing synchronisation mechanism to enable dynamic creations and submissions of new events, as shown in Table 7.4. By doing so, we were able to create and execute new events, such as the generation and submission of the workloads, just in time.
1: RuntimeCloudSim.runSimulation()
2: if referredEventQueueNotEmpty or futureEventQueueNotEmpty then
3:     event \leftarrow\text{getFirstEvent( )};
4:     nextEventTime \leftarrow\text{event.getTime( )};
5:     if nextEventTime > simulationTimer and isDiscreteTimeFeatureSet == true then
6:         simulationTimer \leftarrow\text{simulationTimer + minimumTimeBetweenEvents};
7:         return true; \quad // back to the caller, e.g. main( ), and continue the simulation
8: end if
9: end if
10: entities check and process received events in referredEventQueue;
11: if futureEventQueueIsEmpty then
12:     return false; \quad // back to the caller and stop the simulation
13: end if
14: for event in futureEventQueue do
15:     if isDiscreteTimeFeatureSet == false then
16:         simulationTimer \leftarrow\text{event.getTime( )};
17:     end if
18:     if event.getType( ) == CREATE_ENTITY then
19:         createNewEntity(event.getData( ));
20: else if event.getType( ) == SEND_EVENT then
21:     add event to referredEventQueue;
22: end if
23: end for
24: return true; \quad // back to main( ) and continue the simulation
25: end RuntimeCloudSim.runSimulation

Table 7.3 Pseudo code snippets of our runSimulation() method

1: main()
2: initialise simulation library and simulationTimer;
3: createSimulationEntities(brokers, datacentres, …);
4: createAndSubmitResourcesAndWorkloads(hosts, VMs, cloudlets/sessions, …, withDelay);
5: RuntimeCloudSim.startEntities( );
6: while RuntimeCloudSim.runSimulation( ) do
7:     createSimulationEntities(brokers, datacentres, …); //according to time or certain conditions
8:     createAndSubmitResourcesAndWorkloads(hosts, VMs, cloudlets/sessions, …, withDealy); //according to time or certain conditions;
9: end while
10: RuntimeCloudSim.stopSimulation( );
11: end main

Table 7.4 Pseudo code snippets of our main() method

7.3.2 Server Behaviour Simulation and Workload Execution Modelling

Based on our assessment, to effectively simulate request-execution behaviours in both the web application and database servers, there are two important features needed to be addressed in the simulation design. Firstly, the server needs to be able to handle overload situations when concurrent
requests exceed its capacity limit, such as the number of concurrent requests and the amount of required memory space. For example, either a process-based Apache web server or a thread-based MySQL database server possesses a limit on the maximum number of concurrent requests/connections, but is able to queue up the over-capacity requests/connections for later processing once required resources or processes/threads become available. The same situation applies when the required memory exceeds an upper limit set by the operating system of the server. Secondly, paired workloads (consisting of one asCloudlet and one related dbCloudlet mentioned previously in the CloudSimEx framework in section 7.2.1.2) which are incurred by the same user request but are executed separately in the web application and database servers need to be synchronously processed. For instance, in response to a browsing request for displaying sales items, an ecommerce web application server needs to wait for the results of a database query operation performed by a backend database server before sending the desired web content back to the client.

Figure 7.2 Application/Database Server Overloading in CloudSimEx Simulation

Figure 7.2 shows how CloudSimEx handles the overload situations. As illustrated by Figure 7.2 (a), when concurrent requests do not exceed the maximum capacity of a web application/database server, all requests can be successfully executed and responded. However, there has not been implementation of a waiting queue mechanism, nor of a policy for dealing with the maximum number of concurrent requests, in CloudSimEx (as well as CloudSim). Therefore, when the concurrent requests require more than available resources, such as memory, from the server, the server would simply become unavailable and all requests fail, as shown in Figure 7.2 (b).

To overcome the abovementioned overload problem, in our simulation we implemented both a load dispatcher and a waiting queue mechanism in the server to queue up over-capacity requests when the number of concurrent requests exceeds a predefined threshold of the server (for instance 256
processes for the web server and 150 threads for the database server, or when the required memory exceeds the total memory capacity or the upper limit of memory commit). As shown in Figure 7.3, the load dispatcher submits the arrived requests to available execution processes/threads and put the over-capacity requests to the waiting queue. Once an execution process/thread or a chunk of memory becomes available again, the load dispatcher will select and submit a request, based on for example its arrival time or memory requirement, for execution. By doing so, we could perform our simulation in a more realistic way and all requests can be successfully executed at the expense of a longer response time for those ever queued. In section 7.3.5, we illustrate a proposed mechanism for scaling out the web application and database servers on demand to ensure that reasonable service quality is maintained in response to dynamically surging workloads.

![Figure 7.3 Load Dispatcher and Waiting Queue for Over-capacity Requests in Our Simulation](image)

**Table 7.5 Configurations for App/DB Servers and Workloads in Comparison Simulations**

<table>
<thead>
<tr>
<th>Simulation Objects</th>
<th>CPU Capacity/Load (MIPS)</th>
<th>Memory Capacity/Load (Mbytes)</th>
<th>I/O Capacity/Load (IOPS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>VM (appServer/dbServer)</td>
<td>1,000</td>
<td>1,000</td>
<td>500</td>
</tr>
<tr>
<td>Workloads (asCloudlet/dbCloudlet)</td>
<td>100</td>
<td>10</td>
<td>100</td>
</tr>
</tbody>
</table>

To demonstrate how our proposed mechanism enhances the simulation to act more like a real-world scenario, comparison simulations of workload execution were performed based on both the original CloudSimEx and ours. Table 7.5 shows the configurations for the web application/database
servers (denoted as appServer/dbServer) and workloads (asCloudlet/dbCloudlet) used in either CloudSimEx-based or our enhanced simulations. In these simulations, each server had a CPU capacity of 1,000 MIPS, a memory capacity of 1,000 Mbytes, and an I/O capacity of 500 IOPS (Input/Output Operations per second), whereas each workload required 100 million CPU instructions, 10 Mbytes memory, and 100 I/O operations.

<table>
<thead>
<tr>
<th>Cloudlet ID</th>
<th>Session ID</th>
<th>Vm</th>
<th>Delay (sec.)</th>
<th>Arrival Time (sec.)</th>
<th>Start Exec Time (sec.)</th>
<th>Execution Time (sec.)</th>
<th>Completion Time (sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>4</td>
<td>appServer2</td>
<td>0</td>
<td>10.5</td>
<td>10.5</td>
<td>0.596</td>
<td>11.096</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>appServer2</td>
<td>0</td>
<td>10.5</td>
<td>10.5</td>
<td>0.596</td>
<td>11.096</td>
</tr>
<tr>
<td>11</td>
<td>8</td>
<td>appServer2</td>
<td>0</td>
<td>10.5</td>
<td>10.5</td>
<td>0.596</td>
<td>11.096</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>appServer1</td>
<td>0</td>
<td>10.5</td>
<td>10.5</td>
<td>0.796</td>
<td>11.296</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>appServer1</td>
<td>0</td>
<td>10.5</td>
<td>10.5</td>
<td>0.796</td>
<td>11.296</td>
</tr>
<tr>
<td>9</td>
<td>7</td>
<td>appServer1</td>
<td>0</td>
<td>10.5</td>
<td>10.5</td>
<td>0.796</td>
<td>11.296</td>
</tr>
<tr>
<td>13</td>
<td>9</td>
<td>appServer1</td>
<td>0</td>
<td>10.5</td>
<td>10.5</td>
<td>0.796</td>
<td>11.296</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>dbServer1</td>
<td>0</td>
<td>10.5</td>
<td>10.5</td>
<td>1.388</td>
<td>11.888</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>dbServer1</td>
<td>0</td>
<td>10.5</td>
<td>10.5</td>
<td>1.388</td>
<td>11.888</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>dbServer1</td>
<td>0</td>
<td>10.5</td>
<td>10.5</td>
<td>1.388</td>
<td>11.888</td>
</tr>
<tr>
<td>8</td>
<td>6</td>
<td>dbServer1</td>
<td>0</td>
<td>10.5</td>
<td>10.5</td>
<td>1.388</td>
<td>11.888</td>
</tr>
<tr>
<td>10</td>
<td>7</td>
<td>dbServer1</td>
<td>0</td>
<td>10.5</td>
<td>10.5</td>
<td>1.388</td>
<td>11.888</td>
</tr>
<tr>
<td>12</td>
<td>8</td>
<td>dbServer1</td>
<td>0</td>
<td>10.5</td>
<td>10.5</td>
<td>1.388</td>
<td>11.888</td>
</tr>
<tr>
<td>14</td>
<td>9</td>
<td>dbServer1</td>
<td>0</td>
<td>10.5</td>
<td>10.5</td>
<td>1.388</td>
<td>11.888</td>
</tr>
</tbody>
</table>

Table 7.6 Results of Workload Execution Simulation Using CloudSimEx

Table 7.6 summarises the results of the simulation using CloudSimEx with two web application servers (i.e. appServer1 and appServer2) and one database server (i.e. dbServer1) to process the workloads of seven simultaneous user sessions – each consisting of one asCloudlet, i.e. a cloudlet with an odd ID number, and one dbCloudlet, i.e. a cloudlet with an even ID number - all arrived at 10.5 seconds (simulation time). For example, session 3 was composed of paired workloads of cloudlet 1 (an asCloudlet) and cloudlet 2 (a dbCloudlet). However, there had not been any coordination/synchronisation mechanisms for the execution of asCloudlets and dbCloudlets between the web application and database servers. Upon their arrival, all the workloads were simply submitted to the available servers and started being independently executed. Load balancing between the two web application servers was based on a Round-robin algorithm for even load distribution. The execution time of each cloudlet can be calculated by dividing the total workloads in a server by the server capacities. For instance, the 0.6-second execution time of cloudlet 3 in appServer2 can be calculated by solving: max($\frac{300}{1000(MIPS)}$, $\frac{300}{500(IOPS)}$). (The listed 0.596-second execution time for cloudlet 3 was derived due to the precision issue with double number arithmetic operations in Java.)
Table 7.7 tabulates the results of the comparison simulation based on our proposed mechanism. In addition to the same configurations and settings as mentioned above, we also set limits with a maximum number of two concurrent processes (each dealing with one asCloudlet) for each of the web application servers and a maximum number of three concurrent threads (each dealing with one dbCloudlet) for the database server. Upon the arrival of the seven asCloudlets at 10.5 seconds (simulation time) for example, four were immediately processed by appServer1 and appServer2, and the other three were queued. The same queuing mechanism applied to dbCloudlets. It is important to note that the proposed synchronisation process was performed to ensure the following two conditions were met: Firstly, the asCloudlet and related dbCloudlet of the paired workloads (belonging to the same user session), if both were queued respectively in the web application and database server, were resumed at the same time. Secondly, only when an asCloudlet and its related dbCloudlet were both completed, the responsible web application and database server were able to accept other paired workloads. For instance, appServer1 can release resources consumed by cloudlet 1, which was completed at 10.9 seconds, and resume the execution of other workloads, such as cloudlet 9, only when the related cloudlet 2 was completed by dbServer1 at 11.098 seconds.

<table>
<thead>
<tr>
<th>Cloudlet Id</th>
<th>Session Id</th>
<th>Vm</th>
<th>Delay (sec.)</th>
<th>Arrival Time (sec.)</th>
<th>Start Exec Time (sec.)</th>
<th>Execution Time (sec.)</th>
<th>Completion Time (sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>appServer1</td>
<td>0</td>
<td>10.5</td>
<td>10.5</td>
<td>0.4</td>
<td>10.9</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>appServer1</td>
<td>0</td>
<td>10.5</td>
<td>10.5</td>
<td>0.4</td>
<td>10.9</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>appServer2</td>
<td>0</td>
<td>10.5</td>
<td>10.5</td>
<td>0.4</td>
<td>10.9</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>appServer2</td>
<td>0</td>
<td>10.5</td>
<td>10.5</td>
<td>0.4</td>
<td>10.9</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>dbServer1</td>
<td>0</td>
<td>10.5</td>
<td>10.5</td>
<td>0.598</td>
<td>11.098</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>dbServer1</td>
<td>0</td>
<td>10.5</td>
<td>10.5</td>
<td>0.598</td>
<td>11.098</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>dbServer1</td>
<td>0</td>
<td>10.5</td>
<td>10.5</td>
<td>0.598</td>
<td>11.098</td>
</tr>
<tr>
<td>9</td>
<td>7</td>
<td>appServer1</td>
<td>0.598</td>
<td>10.5</td>
<td>11.098</td>
<td>0.2</td>
<td>11.298</td>
</tr>
<tr>
<td>11</td>
<td>8</td>
<td>appServer2</td>
<td>0.598</td>
<td>10.5</td>
<td>11.098</td>
<td>0.2</td>
<td>11.298</td>
</tr>
<tr>
<td>8</td>
<td>6</td>
<td>dbServer1</td>
<td>0.598</td>
<td>10.5</td>
<td>11.098</td>
<td>0.596</td>
<td>11.694</td>
</tr>
<tr>
<td>12</td>
<td>8</td>
<td>dbServer1</td>
<td>0.598</td>
<td>10.5</td>
<td>11.098</td>
<td>0.596</td>
<td>11.694</td>
</tr>
<tr>
<td>10</td>
<td>7</td>
<td>dbServer1</td>
<td>0.598</td>
<td>10.5</td>
<td>11.098</td>
<td>0.596</td>
<td>11.694</td>
</tr>
<tr>
<td>13</td>
<td>9</td>
<td>appServer1</td>
<td>1.194</td>
<td>10.5</td>
<td>11.694</td>
<td>0.1982</td>
<td>11.8922</td>
</tr>
<tr>
<td>14</td>
<td>9</td>
<td>dbServer1</td>
<td>1.194</td>
<td>10.5</td>
<td>11.694</td>
<td>0.1982</td>
<td>11.8922</td>
</tr>
</tbody>
</table>

Table 7.7 Results of Workload Execution Simulation Using Our Proposed Mechanism

To model the processing of paired workloads (consisting of two related cloudlets, i.e. one appCloudlet and one dbCloudlet) incurred by the same user request but executed in the web application and database server respectively, a state diagram of workload processing in the web application/database server in our simulation is given in Figure 7.4. In the diagram, the rounded
rectangles denote different states; the arrowed connectors refer to transitions of states, and the diamond represents the control of flows. Each connector is labeled with actions, variable assignments, and/or guard conditions (enclosed in square-brackets) which trigger/enable a specific transition.

Figure 7.4 State Diagram of Workload Processing in the App/DB Server in Our Simulation

In Figure 7.4, starting from the initial state (denoted by a filled black circle) with the transition of creating and submitting an asCloudlet/dbCloudlet to the web application/database server, the asCloudlet/dbCloudlet enters the ‘Submitted’ state with three time variables being initialised. Here $t_a$ denotes arrival time and is set to the present simulation time (retrieved by the $\text{time()}$ function call); $t_s$ and $t_c$ denote actual start time and completion time respectively. When required resources, such as an execution process/thread and memory, in the web application/database server are available, $t_s$ is reset to the simulation time at that precise moment and the asCloudlet/dbCloudlet goes to the ‘In Execution’ state. Otherwise the asCloudlet/dbCloudlet enters the ‘Queued’ state, from which it can return back to the ‘In Execution’ state if upon the completion of the synchronisation process in both servers, the required resources become available and its sibling cloudlet in the other server is ready to be resumed or is completed. (The asCloudlet and related dbCloudlet of the paired workloads are a sibling cloudlet of each other. Each server, including the web application server and database server, has its own queue.) Finally, the asCloudlet/dbCloudlet enters the ‘Succeeded’ or ‘Failed’ state and $t_c$ is reset to the present simulation time, if the execution is completed, or the server is failed or does not have the required data. This indicates that the asCloudlet/dbCloudlet has reaches its final state (denoted by a circle with a dot inside).

Based on the abovementioned three time variables, we can calculate the actual execution time
and delay time of each cloudlet (i.e. either an asCloudlet or a dbCloudlet), as well as the response time of each user request (i.e. paired workloads with one asCloudlet and one related dbCloudlet) using Equations (7-1), (7-2), and (7-3) in the simulation. The actual execution time of a cloudlet (denoted as cloudlet_execution_time) refers to the time period that the cloudlet is processed by an execution process/thread with dedicated resources in the responsible server, whereas the delay time of each cloudlet (denoted as cloudlet_delay_time) is the time period that the cloudlet is in the waiting queue. The response time of a user request (denoted as request_response_time) is the time period that starts from the arrival of the request with the creation and submission of the related asCloudlet and dbCloudlet and ends when the related asCloudlet and dbCloudlet are both completed. This definition of response time does not take into account the network transmission time for sending and receiving the request and responses between the client and servers, as network transmission is outside our simulation domain. In Equation (7-3), max(asCloudlet.t_c, dbCloudlet.t_c) refers to the later time between the completion time of the asCloudlet and dbCloudlet, and min(asCloudlet.t_a, dbCloudlet.t_a) refers to the earlier time between the arrival time of the asCloudlet and dbCloudlet.

\[
\text{cloudlet_execution_time} = t_c - t_s \quad (7-1)
\]

\[
\text{cloudlet_delay_time} = t_s - t_a \quad (7-2)
\]

\[
\text{request_response_time} = \max(asCloudlet.t_c, dbCloudlet.t_c) - \min(asCloudlet.t_a, dbCloudlet.t_a) \quad (7-3)
\]

The calculations of Equations (7-1), (7-2), and (7-3) are quite straightforward in our simulation with related time information, such as \( t_c \) and \( t_s \), stored in the member variables of an asCloudlet/dbCloudlet instance. However, along with the introduction of a limit on the maximum number of concurrent requests/connections in the web application/database server, the waiting queue mechanism, and the synchronisation process, the predictions of the delay time and response time of a request become very complicated. Equations (7-4), (7-5), (7-6), (7-7), and (7-8) altogether provide a discrete-time mathematical solution to estimate the response time of each request completed at the \( k \)-th processing update after being submitted to a web application server (denoted as appServer). (For simplification, we do not list Equations for dbCloudlets, as the same concepts apply.) When a request arrives, arrival_time is set to the present simulation time. Meanwhile, number_of_asCloudlets_in_execution_0 and number_of_queued_asCloudlets_0 are the number of asCloudlets already in execution and in the waiting queue respectively in the appServer; number_of_new_arriving_asCloudlets_k is the number of total new asCloudlets arrived before the \( k \)-th processing update; max_concurrent_processes is the maximum number of available concurrent processes of the appServer; time_at_processing_update_k is the simulation time when the \( k \)-th
processing update occurs; \( \sum CPU_{load\_of\_asCloudlets\_in\_execution_k} \) and \( \sum I/O_{load\_of\_asCloudlets\_in\_execution_k} \) are the total CPU and I/O loads respectively then.

\[
\text{number of asCloudlets in execution}_k = \min(\text{max concurrent processes}_as, \\
\text{number of new arriving asCloudlets}_{k-1} \text{ } + \text{number of queued asCloudlets}_{k-1}) \quad (7-4)
\]

\[
\text{asCloudlets execution time}_k = \max(\frac{\sum CPU_{load\_of\_asCloudlets\_in\_execution_k}}{\text{MIPS of the responsible appServer}}, \frac{\sum I/O_{load\_of\_asCloudlets\_in\_execution_k}}{10FS \text{ of the responsible appServer}}) \quad (7-5)
\]

\[
\text{number of queued asCloudlets}_k = \text{number of new arriving asCloudlets}_{k-1} \text{ } + \text{number of queued asCloudlets}_{k-1} \text{ } - \text{number of asCloudlets in execution}_k \quad (7-6)
\]

\[
\text{response time}_k = \max(\text{asCloudlets execution time}_k, \text{dbCloudlets execution time}_k) + \text{delay time}_{k-1} \quad (7-7)
\]

\[
\text{delay time}_k = \text{time at processing update}_k \text{ } - \text{arrival time} \quad (7-8)
\]

There were also two other improvements we made to the CloudSimEx (including CloudSim) framework in workload execution. Firstly, we have resolved the problem of faulty disappearance of certain cloudlets when the needed time to execute them is equal to or less than the minimum interval of simulation steps. Secondly, we have fixed the problem of failing to add the newly arriving requests created and submitted at current simulation time to the unfinished workloads of each responsible server, which caused load balancing and auto scaling to work on miscalculated figures.

7.3.3 Virtual Machine (VM) Modelling

As illustrated by Figure 7.1, a VM which represents an integrated set of virtualised computing resources, such as CPU, memory, storage, and network bandwidth, with the ability of running a dedicated operating system and various applications, is one of the key building blocks for architecting our proposed multi-tier application on an IaaS cloud platform.

In general, the computing capabilities of a VM are affected not only by the shared computing resources, but also by the type of employed virtualisation technology and the workloads of the underlying physical host machine. Hence, different kinds of VMs would exhibit distinct performance aspects, and even the same VM might also show notable performance variations at different times. For example, [182] found that the performance of Amazon’s EC2 ‘small’ instances decrease when their CPU utilisation is over 65%, whereas the “micro” instances can continue performing well with 100% CPU utilisation. [183] found that over a 17-day test period, a same Amazon EC2 instance showed an average CPU capability of 922 MIPS with a SD of 83 MIPS, and an average network throughput of 166 Mbps with a relatively high SD of 162 Mbps. Consequently, how to correctly model a group of VMs for cloud simulations is truly a challenging issue.
7.3.3.1 Virtual Machine Representation in Cloud Simulation

For simulations based on CloudSim, the main properties of a VM include the number of million instructions per second (MIPS) per CPU core, number of CPU cores, size of the VM image (Mbytes), size of memory (Mbytes), network bandwidth (Mbps), and a cloudlet scheduler. The time-shared or space-shared scheduling policy implemented by the scheduler defines how concurrent execution of numerous cloudlets is performed in the VM. In CloudSimEx, an additional property regarding the number of million disk operations per second (MIOPS) is introduced to represent a new type of VM, denoted as HddVm, which has a disk. (We believed that it is more suitable to use IOPS, rather than MIOPS, to represent disk performance. As such, thereafter we use IOPS when describe this property of HddVm. In addition, for simplicity, we use VM to represent all kinds of virtual machines, including HddVm.)

As there are a wide selection of VMs available in the IaaS cloud service market, a fundamental question in modelling a VM for our cloud simulations is how to justifiably define the abovementioned properties of a VM. To answer this question, the approach we took was to benchmark against a targeted off-the-shelf VM selected from Amazon EC2 instances, as they are globally available and widely used. Given that cost-effectiveness was one of our major considerations in system design and that [182] in their cost-performance analysis concluded Amazon EC2 “micro” instances were 51.83% and 13.33% more cost-effective than “small” and “medium” instances respectively, we therefore chose “t2.micro” instances as our target for benchmarking.

<table>
<thead>
<tr>
<th>vCPU</th>
<th>Baseline CPU Performance</th>
<th>Memory (Gbytes)</th>
<th>EBS Performance (IOPS)</th>
<th>Networking Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10%</td>
<td>1</td>
<td>24 ~ 3,000</td>
<td>Low to Moderate</td>
</tr>
</tbody>
</table>

Table 7.8 Key Documented Performance Attributes of Amazon EC2 t2.micro Instances

According to Amazon website [79], t2 instances are general-purpose, low-cost, burstable (up to full core) performance instances which provide a balance of compute, memory, and network resources. It also states that the networking performance of t2.micro is low to moderate and the baseline performance of EBS general purpose solid-state drive (SSD) volumes is three IOPS per Gbytes with bursts up to 3,000 IOPS. Table 7.8 summarises the key documented performance attributes of EC2 t2.micro instance. Here a vCPU refers to a virtual processor and the baseline CPU performance is a percentage of a full core performance.

To gain better knowledge of EC2 t2.micro instances, we decided to use “7zip”, “fio”, and “iperf”
to benchmark their CPU, storage, and network performance, as these three tools have been used by a number of cloud performance studies, for example [183] and [184]. Figure 7.5 shows the average results of CPU rating in MIPS by running “7zip” benchmarking command for 80 iterations at two different times of a day, each compressing and decompressing a file with a size of between 4 and 32 Mbytes, on each of three VMs (based on t2.micro instances with Ubuntu 14.04.1 installed). The results were quite consistent among three VMs with a mean of 3,373.833 MIPS and a SD of 28.344. Among all recorded figures, the lowest one was 3,101 MIPS when VM2 was decompressing a file.

Figure 7.5 “7zip” CPU Benchmark Results of t2.micro Instances in MIPS

Figure 7.6 illustrates the average results of I/O performance in IOPS by running “fio” benchmarking command twice, one for five minutes and the other for 10 minutes, on each of the three

Figure 7.6 “fio” I/O Benchmark Results of t2.micro Instances in IOPS

Figure 7.6 illustrates the average results of I/O performance in IOPS by running “fio” benchmarking command twice, one for five minutes and the other for 10 minutes, on each of the three
VMs. The five-minute benchmarking involved extensive, random read/write operations with accumulated amount of I/O data reaching up to about 12 Gbytes, whereas the 10-minute one involved random read/write operations with accumulated data up to about 24 Gbytes. The resultant overall mean of the I/O performance of the three VMs was 4,965.667 IOPS and the SD was 223.02. Among all figures, the lowest one was 4,711 IOPS for the writing performance of VM3.

![Figure 7.7 “iperf” Network Benchmark Results of t2.micro Instances in Mbps](image)

Figure 7.7 “iperf” Network Benchmark Results of t2.micro Instances in Mbps

Figure 7.7 shows the average results of bandwidth performance in Mbps by running “iperf” benchmarking command for 40 iterations at two different times of a day under the configuration of one server and one client in each experiment. VM1 was launched at a different Amazon Availability Zone (i.e. us-east-1e) from that of VM2 and VM3 (both launched in us-east-1c). The mean of the overall bandwidth performance for communication between VM1 and VM2 located at two different Availability Zones (abbreviated as “Inter AZ”) was around 136.4 Mbps (with a SD of 8.87), while the mean for communication between VM2 and VM3 (denoted as “Intra AZ”) reached around 900 Mbps (with a SD of 155.61). Both the lowest performance figure of 66.1 Mbps and the highest performance figure of 4,044.8 Mbps we got were for communication between VM2 and VM3 (suggesting t2.micro instances’ networking performance is burstable up to more than 4,000 Mbps). When we excluded figures that were higher than 500 Mbps, the mean was about 84.65 Mbps (with a SD of 14.04) for the “Inter AZ” case and 78.7 Mbps (with a SD of 5.815) for “Intra AZ”, respectively.

Since we were not concerned about VM migration (which means a transfer of VM from one host machine to another) and we assumed that there are always enough hosts available for hosting as many VMs as required in a datacentre/service region, and that we can get knowledge of EC2 instances’ startup time from other research, the size of a VM became irrelevant in our simulations. Therefore, we
do not discuss the size of a VM in this chapter. However, instance startup time is another important attribute of VM performance. [185] defined VM startup time as the duration from the time of issuing VM acquisition requests to the time that the acquired instances can be logged in remotely. According to [185], the startup time of EC2 Linux instances is around 96.9 seconds, independent of time of the day, and is consistent among different instance types and datacentre locations. For our cloud simulation, we further assumed that when a new VM is launched for scaling-out, this new VM has equipped with required applications and/or databases with necessary settings. In Amazon EC2, this can be done, for example, by booting a VM from a user created instance image with an EBS/database snapshot.

Based on the outputs of the abovementioned performance benchmarking and findings of [185], we could use the figures, listed in Table 7.9, based on a normal distributions when a related SD was available, to represent a VM in our cloud simulations.

<table>
<thead>
<tr>
<th></th>
<th>Mean</th>
<th>Standard deviation</th>
<th>Lowest Bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU (MIPS)</td>
<td>3,350</td>
<td>25</td>
<td>3,100</td>
</tr>
<tr>
<td>Memory (Gbytes)</td>
<td>1</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>Storage I/O (IOPS)</td>
<td>4,960</td>
<td>220</td>
<td>4,700</td>
</tr>
<tr>
<td>Network Bandwidth (Mbps)</td>
<td>135</td>
<td>10</td>
<td>60</td>
</tr>
<tr>
<td>Startup Time (seconds)</td>
<td>100</td>
<td>10</td>
<td>80</td>
</tr>
</tbody>
</table>

**Table 7.9 Performance Figures for VM Representation in Cloud Simulations**

With regard to the scheduling policy mentioned above, we believed that the time-shared policy can better characterise how numerous cloudlets are simultaneously executed on a VM, as the space-shared one means that a VM can execute only a cloudlet at a time.

7.3.3.2 Cost Modelling of VM Usage in Cloud Simulation

According to Amazon [186], the prices of EC2 t2.micro Linux instances for both reserved and on-demand usage are listed in Table 7.10. Two important aspects of Amazon’s hourly charging scheme for on-demand instance usage are: (1) one-hour usage charge is applied for running an EC2 instance for less than an hour; and (2) starting and stopping an EC2 instance within a single hour for two times, for example, two-hour usage charge is applied. In consideration of these two aspects, we proposed that in order to make the best use of each running VM in a cost-effective way, stopping an
on-demand VM for the purpose of scaling in should happen only when its running time approaches a multiple of an hour. Moreover, instead of leaving some VMs with no any workloads to maximise the number of stoppable VMs as proposed by [166], the load balancing mechanism should distribute workloads to all running VMs to maximise throughput and reduce end-user response time.

<table>
<thead>
<tr>
<th>Amazon Service Region</th>
<th>Reserved for One-Year Term with upfront payment (USD)</th>
<th>On-Demand Hourly (USD)</th>
</tr>
</thead>
<tbody>
<tr>
<td>EU (Ireland)</td>
<td>$81/year</td>
<td>$0.014/hour</td>
</tr>
<tr>
<td>EU (Frankfurt)</td>
<td>$88/year</td>
<td>$0.015/hour</td>
</tr>
<tr>
<td>US East (N. Virginia)</td>
<td>$75/year</td>
<td>$0.013/hour</td>
</tr>
</tbody>
</table>

Table 7.10 Pricing of EC2 t2.micro Linux Instances

Table 7.11 lists the possible savings (or losses) of using on-demand EC2 t2.micro instances over reserved instances for one-year term with upfront payment based on the tariff of EU (Ireland) region [186]. The utilisation rate is the accumulated usage time of t2.micro instances (with each constituting time fragment being rounded up to hours) over total hours in a year. As shown by Table 7.11, one can make savings by using an on-demand instance, rather than a reserved one when the utilisation rate is not over 66% in the Amazon EU (Ireland) region.

<table>
<thead>
<tr>
<th>Utilisation Rate</th>
<th>Charges for On-Demand Instances (USD)</th>
<th>Savings over Reserved Instances (USD)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10%</td>
<td>$12.264</td>
<td>$68.376</td>
</tr>
<tr>
<td>66%</td>
<td>$80.942</td>
<td>$0.058</td>
</tr>
<tr>
<td>100%</td>
<td>$122.64</td>
<td>$41.64</td>
</tr>
</tbody>
</table>

Table 7.11 Savings of On-Demand over Reserved t2.micro Linux Instances based on EU (Ireland)

To estimate the costs of VM usage in our cloud simulation, we adopted the tariff of Amazon EU (Ireland) region. The basic settings were that the two VMs for both the first web application server and the database master server (as shown in Figure 7.1) were based on reserved t2.micro instances, and that additional running VMs for the purpose of scaling out were based on on-demand t2.micro instances. Equation (7-9) expresses how to calculate the total costs of VM usage.
\[C_{total} = 2 \times C_{r,vm} \times \text{roundup}(H/8760) + \sum_{t=1}^{H}(N_{on,vm,t} \times C_{on,vm})\] (7-9)

In equation (7-9), \(C_{total}\) is the total costs of VM usage; \(C_{r,vm}\) is the cost of one reserved VM per year, i.e. US$81; \(H\) is total hours of our simulation time; 8,760 is the number of hours in a year; \(N_{on,vm,t}\) is the number of on-demand VMs running at the \(t\)-th hour in our simulation; and \(C_{on,vm}\) is the cost of one on-demand VM per hour, i.e. US$0.014.

### 7.3.4 Dynamic Workload Modelling

We basically adopted the concept implemented in CloudSimEx [161] that web user sessions can better represent the real-world workloads for a multi-tier web-based application than a set of unrelated individual requests as used in [172,181]. In CloudSimEx, a session generally consists of a sequence of requests, from the same web user, with a think time between two consecutive requests. Moreover, a request is made of two separate cloudlets, i.e. an asCloudlet to be executed by the web application server and a dbCloudlet by the database server. In addition to the use of the Poisson distribution to represent the number of arrival sessions, as supported by a number of studies [181,187], experiments done by [166] based on CloudSimEx also used a frequency function to represent the 12-hour difference in sessions arrival time between the US datacentres and European ones.

![Figure 7.8 Web Workload Test Configuration Diagram](image)

However, how to map real CPU load (in terms of the number of executed instructions per second), memory load (i.e. the size of required memory), and disk I/O load (i.e. the number of disk
I/O operations) incurred by each user request and database read/write operation into a Cloudlet and dbCloudlet is very challenging due to the complexity of the problem domain. Not only do the actual figures differ on different hardware and software platforms, but also they are dependent on the design and implementation of user agents, such as our User Agent module and web browsers, and HTTP server and server applications. Since the purpose of our cloud simulation is to model the scalability and cost features of a cloud-based solution, we decided to perform some workload tests on the Amazon Web Service platform to reasonably estimate these figures. Three virtual machines (abbreviated as VM1, VM2, and VM3 respectively) based on Amazon EC2 t2.micro instances with Ubuntu 14.04.1 installed were launched in the same region to perform these tests. (As summarised in Table 7.9, this type of VMs has an estimated CPU speed of 3,350 MIPS and storage performance of 4,960 IOPS on average.) Figure 7.8 shows the configuration diagram for the workload tests.

On VM1, RUBiS Client was installed to generate the workloads with the number of clients increasing from 100 to 5,000, each with several runs being performed. These workloads were composed of three kinds of user sessions, including visitor, buyer, and seller. A visitor session consisted of a sequence of all-browsing requests from the same client, whereas a session of the other two was a mix of browsing and bidding requests. The think time of each session was produced from the default negative exponential distribution with a mean of 7 seconds (compatible with TPC-W [168]). On VM2, Apache 2.4.7 and PHP5 5.5.9 were installed together with RUBiS bidding application based on PHP implementation. On VM3, MySQL 5.5.43 was installed as the database server and populated with a large amount of data, such as 5,000 users and 32,667 items for sale.

7.3.4.1 Web and RUBiS Application Server

For the web application server running on VM2, we recorded CPU, memory, and I/O loads, by using ‘sar’ (part of the ‘sysstat’ package), ‘iostat’, ‘ps’, and ‘top’ commands. Table 7.12 provides some of the recorded figures. (Results of experiments with 4,000 and 5,000 clients are excluded from Table 7.12, as VM1 crashed when running RUBiS Client to generate the workloads, due to “out of memory” errors.) The ‘Total CPU Usage (%)’ shows the sum of recorded CPU load, including user and system usages, in percentage throughout each run of an entire experiment. The ‘Max CPU Usage per sec. (%)’ represents the maximum percentage of CPU load recorded at a particular second in time in an experiment. The ‘Max. Memory Usage per sec. (%)’ and ‘Max. Memory Commit per sec. (%)’ give the maximum percentage of real memory usage and committed memory respectively, recorded at a particular second in time, whereas ‘Total Memory Commit (%)’ represents the sum of committed memory in percentage recorded throughout an experiment. The ‘Total Reads and Writes (I/O)’ gives the total number of read/write operations throughout an experiment, whilst the ‘Max. Reads and
Writes (I/O) per sec.’ represents the maximum sum of read/write operations, recorded at a particular second in time.

<table>
<thead>
<tr>
<th>No. of Clients</th>
<th>100</th>
<th>100</th>
<th>500</th>
<th>500</th>
<th>1,000</th>
<th>1,000</th>
<th>2,000</th>
<th>2,000</th>
<th>3,000</th>
<th>3,000</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of Requests</td>
<td>9,077</td>
<td>9,781</td>
<td>99,118</td>
<td>96,677</td>
<td>155,631</td>
<td>154,450</td>
<td>249,352</td>
<td>248,383</td>
<td>336,011</td>
<td>348,547</td>
</tr>
<tr>
<td>Total CPU Usage (%)</td>
<td>298.40</td>
<td>356.42</td>
<td>2,600.45</td>
<td>2,535.87</td>
<td>3,680.09</td>
<td>3,817.42</td>
<td>5,983.4</td>
<td>5,826.73</td>
<td>7,655.40</td>
<td>7,965.10</td>
</tr>
<tr>
<td>Max. CPU Usage per sec.(%)</td>
<td>20.39</td>
<td>23.23</td>
<td>79.00</td>
<td>90.00</td>
<td>97.03</td>
<td>94.94</td>
<td>93.00</td>
<td>97.00</td>
<td>96.04</td>
<td>94.05</td>
</tr>
<tr>
<td>Max. Memory Usage per sec.(%)</td>
<td>43.77</td>
<td>44.87</td>
<td>62.1</td>
<td>66.67</td>
<td>72.02</td>
<td>79.13</td>
<td>86.71</td>
<td>94.06</td>
<td>94.02</td>
<td>93.98</td>
</tr>
<tr>
<td>Total Memory Commit (%)</td>
<td>15,450.02</td>
<td>15,460.52</td>
<td>38,357.54</td>
<td>37,789.05</td>
<td>39,396.81</td>
<td>39,450.90</td>
<td>41,974.29</td>
<td>40,870.64</td>
<td>41,837.69</td>
<td>42,959.42</td>
</tr>
<tr>
<td>Max. Memory Commit per sec.(%)</td>
<td>31.87</td>
<td>30.69</td>
<td>249.93</td>
<td>249.94</td>
<td>250.01</td>
<td>250.19</td>
<td>249.98</td>
<td>250.25</td>
<td>249.98</td>
<td>250.19</td>
</tr>
<tr>
<td>Total Reads and Writes (I/O)</td>
<td>509.69</td>
<td>472.26</td>
<td>908.49</td>
<td>850.15</td>
<td>1,088.07</td>
<td>1,083.84</td>
<td>1,406.45</td>
<td>1,537.71</td>
<td>1,816.74</td>
<td>1,946.88</td>
</tr>
<tr>
<td>Max. Reads and Writes (I/O) per sec.</td>
<td>16.00</td>
<td>13.86</td>
<td>195.96</td>
<td>190.82</td>
<td>147.47</td>
<td>222.12</td>
<td>160.61</td>
<td>151.52</td>
<td>246.00</td>
<td>189.90</td>
</tr>
</tbody>
</table>

Table 7.12 Recorded CPU, Memory, and I/O Loads on Web and RUBiS Application Server

<table>
<thead>
<tr>
<th>Avg. CPU Usage per sec. (%)</th>
<th>Avg. Memory Usage per sec. (%)</th>
<th>Avg. Memory Commit per sec.(%)</th>
<th>Avg. I/O Operations per sec.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.293</td>
<td>56.809</td>
<td>27.116</td>
<td>0.969</td>
</tr>
</tbody>
</table>

Table 7.13 Recorded Loads outside the Time Period of the Experiments

Table 7.13 provides the average CPU, memory, and I/O loads, recorded outside the time period of the abovementioned experiments over a time period of 4,611 seconds. By evaluating all these figures in both Table 7.12 and 7.13, we found that both the CPU and I/O operations recorded outside the time period of the experiments were incurred mainly by the monitoring commands, such as ‘sysstat’, ‘iostat’, and ‘top’, and that the operation system would overcommit memory up to around 250.25% of the size of the total memory (i.e. one Gbytes dedicated to each VM) for heavy workloads without causing any runtime memory issues. Meanwhile, the recorded real memory usage was not proportional to the amount of workloads.

To estimate the CPU, memory and I/O loads for serving user requests, averaged figures obtained outside the time period of the experiments were multiplied by the time length of each experiment.
respectively. Then each resultant product was first subtracted from the recorded figure of the same type of loads in each experiment listed in Table 7.12 and divided by the number of user requests. The results, provided in Table 7.14, showed that in general the CPU loads per user request were consistent among all experiments, whereas no consistent relationship between memory loads and the number of requests was found. Our assessment was that the capped percentage of committed memory at around 250% by the operation system was the main cause for such inconsistency. To get better picture of memory usage, we then examined the figures recorded by ‘ps’ command, which showed that the Apache root process consumed about 1.489% of memory (or 15,240 Kbytes) and each child process consumed about 0.788% of memory (or 8,048 Kbytes) throughout their lifetime. Although I/O load for rendering web pages and serving user requests should be application specific, the I/O load in these experiments were relatively small in comparison with the amount of requests. More specifically, the estimated I/O operations per request in all these experiments were all well under 0.0039 and some were even negative. We believed that apart from the difficulty of accurately identifying the cause of each I/O, one of the main reasons for these very small I/O figures could be attributed to the file caching done by the operation system for frequently accessed static files.

<table>
<thead>
<tr>
<th>No. of Requests</th>
<th>9,077</th>
<th>9,781</th>
<th>99,118</th>
<th>96,677</th>
<th>155,631</th>
<th>154,450</th>
<th>249,352</th>
<th>248,383</th>
<th>336,011</th>
<th>348,547</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg. CPU Usage per Request (%)</td>
<td>0.0154</td>
<td>0.0205</td>
<td>0.0245</td>
<td>0.0244</td>
<td>0.0225</td>
<td>0.0236</td>
<td>0.0233</td>
<td>0.0227</td>
<td>0.0222</td>
<td>0.0223</td>
</tr>
<tr>
<td>Avg. Committed Memory per Request (%)</td>
<td>0.0830</td>
<td>0.1030</td>
<td>0.2242</td>
<td>0.2254</td>
<td>0.1470</td>
<td>0.1490</td>
<td>0.0999</td>
<td>0.0968</td>
<td>0.0737</td>
<td>0.0738</td>
</tr>
<tr>
<td>Avg. I/O per Request</td>
<td>-0.00172</td>
<td>-0.00453</td>
<td>0.00335</td>
<td>0.00288</td>
<td>0.00320</td>
<td>0.00321</td>
<td>0.00377</td>
<td>0.00377</td>
<td>0.00359</td>
<td>0.00382</td>
</tr>
</tbody>
</table>

Table 7.14 Calculated Average Loads per Request

Given that the default Multi-Processing Module (MPM) of Apache running on Ubuntu is process-based (or say non-threaded), meaning each HTTP connection is handled by one Apache child process, we believed that the average CPU load per user request (with a mean of 0.0221% and a standard deviation, or SD, of 0.0026%) derived from the above calculation is reasonable to be used for our cloud simulation. As the virtual machine has an estimated CPU speed of 3,350 MIPS, we converted these two figures into 0.74 and 0.087 million instructions respectively.

Regarding memory usage, we conceived that it is rational to use the abovementioned figures, i.e. 1.489% (or 15.25 Mbytes) for Apache root process and 0.788% (or 8 Mbytes) for each child process, in our cloud simulations, as long as the total required memory does not exceed 250% and the number of concurrent Apache child processes, each dealing with one user request, is not bigger than the
predefined upper limit, e.g. 256 by default. Nevertheless, we also noted that according to Table 7.13, even when there is no workload, the basic memory utilisation (commit) is around 27.12%.

As for I/O loads per request, we believed that both a derived mean of 0.0021 and a SD of 0.0028 were not valid to be used in our cloud simulations. Even when we excluded the two negative figures regarding average I/O load per request, the derived mean of 0.0034 (with a SD of 0.0002) was still too small. Consequently, we decided to build up our cloud simulations based on our own assumptions for I/O loads along with reasonable justifications. More details are given when we discuss how to make up different asCloudlets and dbCloudlets later in section 7.3.4.3.

Table 7.15 summarises the estimated figures for modeling the CPU, memory, and I/O loads per request on a web application server built upon Amazon EC2 t2.micro instance with Ubuntu 14.04.1 LTS and Apache 2.4.7 installed.

<table>
<thead>
<tr>
<th>Avg. CPU Load per Request (Million Instructions)</th>
<th>Mean</th>
<th>Standard deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory Usage by Apache Root Process (Mbytes)</td>
<td>15.25</td>
<td>Na</td>
</tr>
<tr>
<td>Avg. Memory Usage per Request (Mbytes)</td>
<td>8</td>
<td>Na</td>
</tr>
<tr>
<td>Avg. I/O Operations per Request</td>
<td>0.0021 (not to be used in simulations)</td>
<td>0.0028 (not to be used in simulations)</td>
</tr>
</tbody>
</table>

Table 7.15 Estimated Loads per Request on Web and RUBiS Application Server

7.3.4.2 MySQL Database Server

For the MySQL database server running on VM3, Table 7.16 shows the figures with respect to CPU, memory, and I/O loads recorded using ‘sar’ (‘sysstat’) and ‘iostat’ from five runs of experiments (different from those listed in Table 7.12). To record database operations, we also enabled the ‘general log’ setting on the MySQL database server. Except for the first run with 100 clients, the others had the maximum concurrent database connections capped by 150, the default number of maximum concurrent connections in MySQL.

Table 7.17 provides the average CPU, memory, and I/O loads, recorded outside the experiments over a time period of 2,066 seconds. When there was no assigned workloads, the 5.1% (of the one Gbytes) memory usage by MySQL represented a memory space of 52.224 Mbytes.
<table>
<thead>
<tr>
<th>No. of Clients</th>
<th>100</th>
<th>500</th>
<th>1,000</th>
<th>2,000</th>
<th>3,000</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of Requests</td>
<td>8,826</td>
<td>48,375</td>
<td>94,936</td>
<td>164,001</td>
<td>234,809</td>
</tr>
<tr>
<td>Total CPU Usage (%)</td>
<td>797.45</td>
<td>1,268.87</td>
<td>1,745.82</td>
<td>2,538.89</td>
<td>3,366.33</td>
</tr>
<tr>
<td>Max. CPU Usage per Sec. (%)</td>
<td>11.22</td>
<td>22.43</td>
<td>21.21</td>
<td>22.33</td>
<td>20.4</td>
</tr>
<tr>
<td>Max. Memory Commit per sec. (%)</td>
<td>39.09</td>
<td>41.93</td>
<td>41.58</td>
<td>41.93</td>
<td>41.93</td>
</tr>
<tr>
<td>Min. Memory Commit per sec. (%)</td>
<td>38.7</td>
<td>38.36</td>
<td>38.81</td>
<td>38.36</td>
<td>38.36</td>
</tr>
<tr>
<td>Total Reads and Writes (I/O)</td>
<td>1,318.49</td>
<td>4,875.19</td>
<td>6,755.28</td>
<td>11,456.82</td>
<td>15,189.1</td>
</tr>
<tr>
<td>Max. Reads and Writes (I/O) per Sec.</td>
<td>45.92</td>
<td>116.31</td>
<td>133.33</td>
<td>136</td>
<td>176.04</td>
</tr>
<tr>
<td>Total DB Operations</td>
<td>15,583</td>
<td>86,275</td>
<td>168,766</td>
<td>290,510</td>
<td>416,345</td>
</tr>
<tr>
<td>Total DB Connections</td>
<td>17</td>
<td>151</td>
<td>150</td>
<td>156</td>
<td>155</td>
</tr>
<tr>
<td>Max. Concurrent DB Connections</td>
<td>17</td>
<td>150</td>
<td>150</td>
<td>150</td>
<td>150</td>
</tr>
</tbody>
</table>

Table 7.16 Recorded CPU, Memory, and I/O Loads on MySQL Database Server

<table>
<thead>
<tr>
<th>Avg. CPU Usage per sec. (%)</th>
<th>Avg. Memory Usage by MySQL Process (%)</th>
<th>Avg. I/O Operations per sec.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.676</td>
<td>5.1%</td>
<td>1.003</td>
</tr>
</tbody>
</table>

Table 7.17 Recorded Loads outside the Time Period of the Experiments

Given that some web user requests do not result in any database operations and some might well require more than one operation, we believed that there is no direct relationship between the number of user requests and workloads on the database server. Instead, our assessment was that the main workloads on the database server are related to the number of required database operations and the number of concurrent database connections. Similar to the calculation process adopted for VM2, to estimate the CPU and I/O loads for serving database operations, averaged figures obtained outside experiments were multiplied by the time length of each experiment respectively. Then each resultant product was first subtracted from the recorded figure of the same type of workloads in each experiment listed in Table 7.16 and divided by the number of database operations. To estimate memory usage, the difference between the maximum and minimum committed memory recorded in each experiment was divided by the number of maximum concurrent database connections.
Table 7.18 Calculated Average Loads per Database Operation/Connection

Table 7.18 gives the estimated CPU and I/O loads per database operation, as well as the estimated memory usage per database connection. The figures for the average CPU usage per database operation in Table 7.18 were quite consistent among most experiments, except for the first one based on the lowest number of user requests and database operations. When we double checked with other runs of experiments with a similar number of user requests and database operations, the total CPU usage of 797.45% shown in Table 7.16 is relatively much higher than others. Consequently, we conceived that it should be acceptable to exclude this run from calculating both the mean value (i.e. 0.0081% or 0.271 MIPS) and SD (i.e. 0.0014% or 0.047 MIPS) of the average CPU usage per database operation for our cloud simulations.

With regard to the estimated memory load per database connection, the estimated figures shown in Table 7.18 were consistent with a mean of 0.0226% (or 0.231 Mbytes) and a SD of 0.0023% (or 0.0235 Mbytes), but they were much smaller than our expectation. According to the documentation of MySQL [188], each connection executed by an individual thread would require additional memory space for buffers and thread stack. Based on the default settings of MySQL, such additional memory usage per connection could be up to about 2.7 Mbytes (representing 0.264% of the one Gbyte memory on VM3). When double-checking with recorded figures based on ‘ps’ command, we found that the related memory usage footprints were even smaller than those in Table 7.18. Therefore, we were convinced that the real memory usage by MySQL server on Ubuntu system is much lower than what it needs and allocates, and that it should be reasonable to use these figures in Table 7.18 in our cloud simulations.

As stated previously, the number of required I/O operations is application specific. Hence, we decided that the estimated I/O loads listed in Table 7.18 should not be used directly in our cloud simulations. Instead, we modelled our simulations based on our own assumptions for I/O loads along with reasonable justifications. More details are given when we discuss how to make up different asCloudlets and dbCloudlets later in section 7.3.4.3. Table 7.19 summarises the estimated figures for

<table>
<thead>
<tr>
<th>Total DB Operations</th>
<th>15,583</th>
<th>86,275</th>
<th>168,766</th>
<th>290,510</th>
<th>416,345</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max Concurrent DB Connections</td>
<td>17</td>
<td>150</td>
<td>150</td>
<td>150</td>
<td>150</td>
</tr>
<tr>
<td>Avg. CPU Usage per DB Operation (%)</td>
<td>0.0280</td>
<td>0.0102</td>
<td>0.0080</td>
<td>0.0073</td>
<td>0.0071</td>
</tr>
<tr>
<td>Avg. Memory Commit per DB Connection (%)</td>
<td>0.0229</td>
<td>0.0238</td>
<td>0.0185</td>
<td>0.0238</td>
<td>0.0238</td>
</tr>
<tr>
<td>Avg. I/O Operations per DB Operation</td>
<td>0.0502</td>
<td>0.0498</td>
<td>0.0365</td>
<td>0.0373</td>
<td>0.0350</td>
</tr>
</tbody>
</table>
modelling the CPU, memory, and I/O loads per database operation/connection on a database server based on Amazon EC2 t2.micro instance with Ubuntu 14.04.1 LTS and MySQL 5.5.43 installed.

<table>
<thead>
<tr>
<th></th>
<th>Mean</th>
<th>Standard deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg. CPU Load per DB Operation (Million Instructions)</td>
<td>0.271</td>
<td>0.047</td>
</tr>
<tr>
<td>Memory Usage by MySQL without Workloads (Mbytes)</td>
<td>52.224</td>
<td>n/a</td>
</tr>
<tr>
<td>Avg. Memory Usage per DB Connection (Mbytes)</td>
<td>0.231</td>
<td>0.0235</td>
</tr>
<tr>
<td>Avg. I/O Operations per DB Operation</td>
<td>0.0417</td>
<td>0.0076</td>
</tr>
</tbody>
</table>

Table 7.19 Estimated Loads per Operation/Connection on MySQL Database Server

Finally, to examine the effect of data size on the performance of database operations, we conducted another trial on the abovementioned MySQL server. As shown in Table 7.20, in the trial, the size of data for each database insertion varied from 250 bytes (for inserting one row of data into a compressed MySQL table) to 10,000 bytes (for inserting 40 rows at a time). For each data size, an insertion command was repeatedly executed for more than 50 times and then results from the last 20 times were selected to calculate the average execution time. For example, the average execution time of inserting 10 rows of data into the database using one insertion command was 0.0379 seconds, representing 10.39% increase of time when compared with that of inserting one row of data. Our main finding was that in general the bigger data size, the longer execution time was needed.

<table>
<thead>
<tr>
<th>Number of Rows</th>
<th>1</th>
<th>5</th>
<th>10</th>
<th>20</th>
<th>30</th>
<th>40</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Size (bytes)</td>
<td>250</td>
<td>1,250</td>
<td>2,500</td>
<td>5,000</td>
<td>7,500</td>
<td>10,000</td>
</tr>
<tr>
<td>Avg. Exec. Time (sec.)</td>
<td>0.00343</td>
<td>0.00336</td>
<td>0.00379</td>
<td>0.00400</td>
<td>0.00434</td>
<td>0.00444</td>
</tr>
<tr>
<td>Time Increase (%)</td>
<td>-</td>
<td>-2.02%</td>
<td>10.39%</td>
<td>16.63%</td>
<td>26.49%</td>
<td>29.33%</td>
</tr>
</tbody>
</table>

Table 7.20 Average Execution Time per Database Insertion based on Different Data Sizes

However, both the CloudSim and CloudSimEx platforms do not have any implementations on database processing events. Consequently, in consideration of both the facts that the maximum number of concurrent database connections in the MySQL server is 150, and that the maximum
number of insertions that the database can perform in a second would be literally up to around 250, we did not take this database processing time into consideration when modelling the workloads for the default set-ups of our cloud simulations. To examine the possible impact of this processing time on the performance of the database server, we also performed some experiments based on a revised presentation of dbCloudlet I/O loads. More discussions about this issue are given in section 7.4.2.

7.3.4.3 User Request Representation in Cloud Simulations

Grounded on the results of the above workload experiments, the following assumptions were made based on some rational justifications for three types of user requests, each with a different combination of asCloudlet (representing the workloads on the web application server) and dbCloudlet (representing the workloads on the database server), as summarised in Table 7.21:

- Type I requests: a web page browsing without any database read/write. The asCloudlet contained all the workloads, while dbCloudlet did not contain any workload. Here we further assumed that most of data needed for executing the asCloudlet can be found in the memory in consideration of the very small number of I/O operations we observed from the workload experiments on VM2. Hence, we proposed that the asCloudlet would require only 0.05 physical I/O operations on average with a SD of 0.01. (During simulation runtime, the randomly generated minimum value of I/O operations was bound to zero. Although in reality the number of physical I/O operations should not be a fraction, we use 0.05 for simulation because we learned that the actually required I/O operations were relatively small on a web server and that the operating system can combine several logical I/O operations into one physical operation.) For simplicity, we adapted the mean value of CPU loads per request listed in Table 7.21 to 0.74 million instructions (with a SD of 0.087) and the memory loads to 8.0 Mbytes (with a SD of 0.4) to constitute the asCloudlet.

- Type II requests: a web user request requiring several logical database read/write operations. As this type of requests, such as user authentication and historical health data retrieval, might well require similar resources as Type I requests from the web application server, we assume that the asCloudlet would therefore be the same as that of the previous type. As learned from our workload experiments on VM3 with MySQL, either one or several database query commands could invoke zero to several physical read operations, depending on the size of the data, whether the operating system combined several logical I/O operations into one physical operation, and whether the data were already stored in the database cache. However, when we looked at the database system performance as a whole, the average number of physical read/write per database operation was very small, as evidenced in Table 7.19. We took into consideration the figures in Table 7.19 and then
assumed that the dbCloudlet (representing workloads for receiving and transmitting data from and to the web application server, one database connection, and seven database operations) would require 0.27 million instructions (with a SD of 0.047), 0.25 Mbyte memory (with a SD of 0.025) and 0.3 physical I/O operations (with a SD of 0.05). More discussions about the effect of changing the number of database operations represented by the dbCloudlet are provided in section 7.4.

The 0.3 physical I/O operations were roughly estimated by using Equation (7-10) with \( \bar{n}_{IO\_per\_DB\_operation} \) representing the average number of I/O operations for one database operation (i.e. 0.0417 in Table 7.19) and \( n_{DB\_operation} \) referring to the required number of database operations (i.e. seven as suggested above).

\[
IO\_load = \bar{n}_{IO\_per\_DB\_operation} \times n_{DB\_operation} \tag{7-10}
\]

- Type III requests: a database insertion request without a need for the web application server to render a webpage. An example of this type of request is to insert real-time indoor patient location tracking records into the database. Since the workload would consume less CPU instructions than the previous two and no I/O operations would be needed on the web application server, we assumed that the asCloudlet would require 0.37 million instructions (with a SD of 0.043), 8.0 Mbyte memory (with a SD of 0.4) and zero physical I/O operations. Meanwhile, we assumed the dbCloudlet (representing one database connection and five database operations) would require 0.13 million instructions (with a SD of 0.023), 0.25 Mbyte memory (with a SD of 0.025) and 0.2 physical I/O operations (with a SD of 0.03).

<table>
<thead>
<tr>
<th>Type of Requests</th>
<th>Cloudlets</th>
<th>CPU Load (million instructions) Mean</th>
<th>SD</th>
<th>Memory Usage (Mbytes) Mean</th>
<th>SD</th>
<th>No. of I/O Operations Mean</th>
<th>SD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type I</td>
<td>asCloudlet</td>
<td>0.74</td>
<td>0.087</td>
<td>8.0</td>
<td>0.4</td>
<td>0.05</td>
<td>0.01</td>
</tr>
<tr>
<td></td>
<td>dbCloudlet</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Type II</td>
<td>asCloudlet</td>
<td>0.74</td>
<td>0.087</td>
<td>8.0</td>
<td>0.4</td>
<td>0.05</td>
<td>0.01</td>
</tr>
<tr>
<td></td>
<td>dbCloudlet</td>
<td>0.27</td>
<td>0.047</td>
<td>0.25</td>
<td>0.025</td>
<td>0.3</td>
<td>0.05</td>
</tr>
<tr>
<td>Type III</td>
<td>asCloudlet</td>
<td>0.37</td>
<td>0.043</td>
<td>8.0</td>
<td>0.4</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>dbCloudlet</td>
<td>0.13</td>
<td>0.023</td>
<td>0.25</td>
<td>0.025</td>
<td>0.2</td>
<td>0.03</td>
</tr>
</tbody>
</table>

Table 7.21 Representations of asCloudlets and dbCloudlets in Cloud Simulation
7.3.4.4 User Session Representation in Cloud Simulations

As stated previously, we used sessions to represent workloads of our web-based system. A session generally consists of a sequence of requests from the same web user with a response time and a think time in between every two consecutive requests. Equation (7-11) shows the concept of how a session is composed of \( m \) requests, \( m \) response times and \( m-1 \) think times, where each request can be represented by a combination of an asCloudlet and a dbCloudlet. The response time was defined as the time elapsed between the submission and completion of a request, and the think time was the time between receiving a reply from the App&DB module and making a subsequent request. As network transmission was outside our simulation domain, here we assumed that the time needed for data transmission between the user/User Agent and the web application server is negligible.

\[
A \text{ session} = \sum_{i=1}^{m} (\text{request}_i + \text{responseTime}_i) + \sum_{j=1}^{m-1} \text{thinkTime}_j \\
= \sum_{i=1}^{m} (\text{asCloudlet}_i + \text{dbCloudlet}_i + \text{responseTime}_i) + \sum_{j=1}^{m-1} \text{thinkTime}_j 
\]  

(7-11)

The response time of each request is subject to both the amount of workloads contained in that request and the availability of computing resources on the App&DB module for executing these workloads. The think time on the other hand varies depending mainly on the user himself/herself and contents of each webpage in browsing. It is our perception that perhaps due to the lack of runtime dynamic workload generation/submission capacity in CloudSim’s core design, as mentioned in the previous section, existing research work, such as [158,161,166], all used a fixed time interval to represent the think time for all users/sessions. (For example, [161,166] stored the think time in a member variable of the WebBroker class for all sessions throughout the entire run of a simulation.) As we re-wrote the simulation event handling engine to make our simulation truly runtime dynamic, we were able to set a unique value for each think time between two consequent requests within each user session. Accordingly, we also added a new member variable, an array of double numbers, to the WebSession class so as to store a set of various think times for each user session.

For our proposed home healthcare telemonitoring, there were four main groups of users, as discussed in section 6.3, each generating a distinct kind of session workloads to the App&DB module.

- Patients and carers: Workloads generated by this group of users were mainly to check personal historical health data and professionals’ comments on diagnoses or questions, and to update personal information or raise questions, for instance. Each day, a patient who went online would create \( m_{\text{patient\_online\_browsing}} \) Type I and \( m_{\text{patient\_online\_db}} \) Type II requests, while a carer who went online would generate \( m_{\text{carer\_online\_browsing}} \) Type I and \( m_{\text{carer\_online\_db}} \) Type II requests. As mentioned in section 6.3.1.4, depending on a patient’s
Internet habit, the value of $m_{patient\_online\_browsing}$ and $m_{patient\_online\_db}$ could be predicted based on heavy, median, and light Internet usage, respectively. The same concept could apply to the prediction of both $m_{patient\_online\_browsing}$ and $m_{carer\_online\_db}$. To simulate a web user session from each individual in this subgroup of users, workloads were first generated from a sequence of Type I requests and then followed by Type II requests.

- Healthcare professionals: It was our assumption that a number of professionals were given the responsibility to review the outcomes of telemonitoring, as mentioned in section 6.3.3.1. A professional needed to spend some time each day reviewing the monitoring data of a number of patients with a fall or vital sign parameters exceeding thresholds and a number of selected patients’ location tracking records. This would constitute a user session, involving $m_{prof\_online\_browsing}$ Type I and $m_{prof\_online\_db}$ Type II requests with a think time of $t_{think\_browsing}$ and $t_{think\_db}$, respectively. For simplicity, workloads to be submitted to the App&DB module were first generated from a sequence of $m_{prof\_online\_browsing}$ Type I requests and then from $m_{prof\_online\_db}$ Type II requests.

- Emergency tele-consultants: It was our assumption that a number of tele-consultants were given the responsibility to response to emergency alerts for detected falls and critical vital sign conditions of patients, and then to make emergency teleconsultation phone calls to those patients. Upon or during each teleconsultation phone call with a particular patient, a tele-consultant would make $m_{tele\_online\_browsing}$ Type I and $m_{tele\_online\_db}$ Type II requests checking both the patient’s personal information and telemonitored records that triggered the alert. These numbers were predicted by using Equations (7-12) and (7-13). The think times $t_{tele\_think\_browsing}$ and $t_{tele\_think\_db}$ were both normal distributed with a mean of 40 and 50 seconds, as well as a SD of six and eight seconds, respectively. The maximum time length of a urgent tele-consultation call, denoted as $m_{tele\_online\_db}$, was set as eight minutes according to section 6.3.3.3.

$$m_{tele\_online\_browsing} = 2 .. 4$$  \hspace{1cm} (7-12)

$$m_{tele\_online\_db} = (t_{max\_per\_call} - m_{tele\_online\_browsing} \times t_{think\_browsing}) / t_{think\_db}$$  \hspace{1cm} (7-13)

- User Agents: As stated in section 3.2, the User Agent module acts as an intelligent monitoring agent which collects telemonitoring results, such as vital sign parameters, fall detection alerts, and real-time location tracking records, and makes use of the App&DB module to store these results. The workloads generated by a User Agent mainly consisted of a one-off Type III request without a think time. In general, timing for generating this type of workloads was event-driven, except that workloads to upload real-time location tracking records to the MySQL database server were created at varying time intervals. To
avoid database overload with a large number of simultaneous requests made by User Agents to upload real-time location tracking records, a random, variable waiting time (uniform distributed between 150 and 600 seconds) for sending the next request was generated by each User Agent right after each request made. When the waiting time for the next upload of location tracking records becomes longer, the amount of data increase proportionally.

\[ m_{\text{user_agent.db.upload}} = m_{\text{user_agent.vs}} + m_{\text{user_agent.fall.alert}} + m_{\text{user_agent.location}} \] (7-14)

Equation (7-14) was formulated to represent that the number of Type III requests, denoted as \( m_{\text{user_agent.db.upload}} \), was actually the sum of vital sign measurements, fall detection alerts and location tracking data sets (denoted as \( m_{\text{user_agent.vs}} \), \( m_{\text{user_agent.fall_alert}} \), and \( m_{\text{user_agent.location}} \), respectively). For reliability reason, alerts for the vital-sign-exceeding-threshold events were not separately generated and sent to the App&DB module, but embedded in the vital sign monitoring data.

### 7.3.5 Load Balancing and Auto Scaling

For our cost-effective design, the main purpose of load balancing is to optimise efficient use of all available VMs/servers in a cost-effective way without unduly sacrificing service quality and performance, such as throughput and response time. Meanwhile the main purpose of auto scaling is to optimise dynamic provisioning of VMs/servers based on the amount of workloads in a cost-effective way to maximise service quality. Since the concepts of load balancing and auto scaling can be effectively applied to both web application servers and database servers (not including load balancing and scaling at the database level), in this section we therefore limit our discussions to the design of load balancing and auto scaling for web application servers.

#### 7.3.5.1 Load Balancing based on “Cost-aware” Least Pending Requests

For load balancing, a geographic DNS server can generally serve as a global load balancer, such as Amazon Route 53 service, to distribute traffic (i.e. user workloads) across regions to the nearest cloud datacentre/service region. Hence in this section our discussions focus mainly on the design of the regional load balancer. Besides, we did not further consider the limitation imposed by EC2 on deploying software load balancing. In practice, load balancers based on Round-robin algorithm or least pending/outstanding/connection requests have been widely implemented to distribute the traffic evenly among all VMs or to the VM with least pending workloads or connections. To embrace the consideration of cost-effectiveness, we introduced a “cost-aware” least pending requests (CALPR) algorithm that basically routes incoming traffic to the least loaded VM when non-sticky session policy is adopted (i.e. re-assignments of session requests to different VMs are allowed). The only exception
is that from the time when the least loaded VM(s) is (are) selected as a potential candidate(s) for being stopped at the next scaling-in action for cost saving, the load balancer would stop routing traffic to it (them) but to the least loaded one of the remaining VMs. This is to ensure that each candidate VM for scaling in can finish the execution of all existing workloads before it is stopped.

Table 7.22 gives the algorithm for constructing our regional load balancer, which is activated to select a VM for submitting each incoming user request (consisting of an asCloudlet for the web application server and a dbCloudlet for the database server). If ‘sticky’ policy is adopted, the load balancer chooses the same VM for all the requests of each user session. The runningVMsList stores all running VMs which pass the latest health check and vmSortedSet provides an ascending-order set of VMs based on firstly the total number of assigned (but not yet executed) cloudlets and processes (or threads for the database server) in execution. When two VMs have equal assignments, vmSortedSet would sort them based on their resource utilisation. This selection criterion was set based on our findings from earlier experiments that the level of resource utilisation, such as CPU, I/O, or memory utilisation, only represented the workloads currently in execution on a VM, but did not include those assigned but not yet executed cloudlets. This was particularly important when a large number of requests usually arrived at the load balancer at the same time. Besides, since the load balancer did not possess the information about the workloads contained in an arriving cloudlet, it was therefore not feasible to calculate the amount of workloads, but the number of arriving cloudlets, assigned to a VM.

```
1: RuntimeCloudSim.regionalLoadBalancer(WebSession session)
2:   if adoptNonstickyPolicy == FALSE and session.getVmId() != null then
3:       return webSession.getVmId(); // return the Id of the VM, if this session has been
4:         // assigned to it and sticky policy is adopted
5:   end if
6:   for vm in runningVMsList do
7:     vmSortedSet.add(vm); // get an ascending-order set of VMs based on the assigned number of
8:       // cloudlets;
9:   end for
10:  for vm in vmSortedSet do
11:     if vm.isCandidateForNextScalingIn() == FALSE then
12:       return vm.getVmId(); // return the Id of the VM with the least number of assigned cloudlets and
13:         // processes (if two are equal then based on resource utilisation;
14:     end if
15: end for
16: end RuntimeCloudSim.regionalLoadBalancer
```

Table 7.22 Pseudo code snippets of our Regional Load Balancer Algorithm

Based on our observation, the implementation of the SimpleWebLoadBalancer in the CloudSimEx framework did not take the abovementioned issue into account, causing the Balancer to
assign all the concurrently arriving cloudlets to the same VM, but to leave the other VMs with no any new assignments.

7.3.5.2 Auto Scaling based on “Forward-looking” Unused CPU, I/O, or Memory Capacity

For auto scaling, it is common to see that rule-based thresholds on average CPU and/or memory utilisation are set and checked regularly to dynamically activate auto scaling out/in by starting/stopping a predefined number of VMs, such as Amazon’s “Auto Scaling” service. Potential problems with this approach are over-provisioning and under-provisioning of VMs/servers after scaling out and in, respectively. Our proposed solutions for scaling-out/in mechanisms to solve these problems are illustrated below.

Scaling Out

Oftentimes, a scaling-out policy is set based on, for example, 80% of average CPU, I/O utilisation, or memory utilisation (the so-called CPU-, I/O-, or memory-Reactive as mentioned in section 7.2.3.4), among all running VMs to add one additional VM into the auto scaling group. If originally there are five running VMs in the auto scaling group and presumably every current request has at least a subsequent request from the same user session, the total unused CPU, I/O, or memory capacity would approximately be up to two VMs or 200% of one VM’s MIPS, IPOS, or memory capacity (i.e. 20%×5 + 100% and minus the capacity used by new arriving requests) after the new VM joins the group. A possible remedy to this over-provisioning issue is to set a scaling-out policy based on the total unused CPU, I/O, or memory capacity, for example 30%, in the whole auto scaling group. Accordingly, after a new VM is added, the total unused CPU, I/O, or memory capacity would approximately be up to 130% of one VM’s MIPS, IOPS, or memory capacity (i.e. 30% + 100% and minus the capacity used by new arriving requests).

However, with the introduction of an upper limit on the number of concurrent processes/threads in the web application/database server, a scaling-out policy based on actual resource utilisation/allocation cannot work effectively. For example, the policy might never be triggered because of the capped number of concurrent execution requests, or might be undesirably triggered if all the queued workloads are added up to compute the allocation quantity. Our proposed solution was similar to the abovementioned remedy by setting the scaling-out policy based on a predefined margin of the total unused CPU, I/O, and memory capacity (in percentage) in the whole group in conjunction with a consideration about possible response time. When the lowest value of the total unused CPU, I/O, or memory capacity is less than the margin (denoted as SCALE_OUT_CAPACITY_THRESHOLD) for a period of time equal to or greater than a predefined time length (denoted as SCALE_OUT_TIME_LENGTH), such as two seconds, the policy is triggered
to avoid the possible maximum response time of currently unfinished and potential incoming requests exceeding an acceptable upper threshold. We called this solution as the forward-looking unused capacity-based auto scaling (FLUCAS) algorithm.

Equations (7-15), (7-16) and (7-17) define how to estimate the forward-looking unused CPU, I/O, and memory capacities of an auto scaling group, in which the value of \textsc{upper\_response\_time\_threshold} can be set based on service quality requirements. As mentioned in section 7.2.2, TPC-C specifies at least 90\% of all on-line transactions must have a transaction response time of less than 5 seconds. To compensate for not taking the network transmission time into account in our simulation, we therefore set \textsc{upper\_response\_time\_threshold} to 3.5 seconds. This means that when there is not any workload in the group which has only one server, the unused CPU, I/O, or memory capacity would be equal to 350\%. Moreover, \textit{predicted\_incoming\_cpu\_load\_per\_second}, \textit{predicted\_incoming\_io\_load\_per\_second}, and \textit{predicted\_incoming\_ram\_load\_per\_second} are set based on the average arriving workloads per second in a predefined time length of the past, for example six seconds. Unlike Equation (7-7), here it is not necessary to concern about the different execution time of sibling cloudlets in a web application server and a database server belonging to different auto scaling groups, because each group scales out or in independently when it becomes a bottleneck or has redundant resource capacities.

\begin{align*}
\text{unused\_cpu\_capacity} &= \frac{((\textsc{upper\_response\_time\_threshold} \times \text{total\_mips\_of\_the\_group}) - (\textsc{upper\_response\_time\_threshold} - 1) \times \text{predicted\_incoming\_cpu\_load\_per\_second}) \times \text{cpu\_load\_of\_total\_unfinished\_cloudlets}}{\text{mips\_per\_server}} \times 100\% \quad (7-15) \\
\text{unused\_io\_capacity} &= \frac{((\textsc{upper\_response\_time\_threshold} \times \text{total\_iops\_of\_the\_group}) - (\textsc{upper\_response\_time\_threshold} - 1) \times \text{predicted\_incoming\_io\_load\_per\_second}) \times \text{io\_load\_of\_total\_unfinished\_cloudlets}}{\text{iops\_per\_server}} \times 100\% \quad (7-16) \\
\text{unused\_ram\_capacity} &= \frac{((\textsc{upper\_response\_time\_threshold} \times \text{total\_ram\_of\_the\_group}) - (\textsc{upper\_response\_time\_threshold} - 1) \times \text{predicted\_incoming\_ram\_load\_per\_second}) \times \text{ram\_load\_of\_total\_unfinished\_cloudlets}}{\text{ram\_per\_server}} \times 100\% \quad (7-17)
\end{align*}

Finally, Equation (7-18) calculates the lowest value, denoted as \textit{minimum\_unused\_capacity}, among the unused CPU, I/O, and memory capacities. When \textit{minimum\_unused\_capacity} is less than \textsc{scale\_out\_capacity\_threshold}, a scale-out process will be invoked by creating a new VM. Then 350\% of the new added unused capacity will be added to \textit{minimum\_unused\_capacity}. If
minimum_unused_capacity is still smaller than SCALE_OUT_CAPACITY_THRESHOLD, one additional VM will be created and 350% will be added to minimum_unused_capacity again. This process will continue until minimum_unused_capacity is larger than SCALE_OUT_CAPACITY_THRESHOLD.

\[
\text{minimum\_unused\_capacity} = \min(\min(\text{unused\_CPU\_capacity}, \text{unused\_IO\_capacity}), \text{unused\_RAM\_capacity})
\]  
(7-18)

Scaling In

A common scenario for scaling-in is that a scaling-in policy is set based on, for example, 20% of average CPU, I/O, or memory utilisation to stop one running VM from the auto scaling group. If originally there are two running VMs in the auto scaling group and presumably every current request has at least a subsequent request from the same user session, the total CPU, I/O, or memory utilisation in the remaining VM might well be up to 40% (i.e. 20% × 2 minus the capacity used by requests of newly arriving sessions) after the policy is triggered. Then it might be very soon that the scaling-out policy would be triggered if the workloads increase a bit. A possible remedy is to pre-estimate the total unused CPU, I/O, or memory utilisation of the remaining VMs, if a running VM is stopped for scaling-in. When the lowest value between the total unused CPU, I/O or memory capacity in the group is greater than a predefined margin (denoted as SCALE_IN_CAPACITY_THRESHOLD), for example 300%, the policy is triggered to stop a VM upon it finishes all its current workloads.

Equations (7-15) to (7-18) can be reused to predict the total unused CPU, IO or memory capacity of the group if the policy is triggered and a VM is stopped, except that we should subtract a VM’s MIPS, IOPS and memory capacities from the total MIPS, IOPS, and memory of the group for the value of total_MIPS_of_the_group, total_IOPS_of_the_group, and total_RAM_of_the_group respectively.

Another important consideration for our cost-effective design is the lifetime of each VM to account for Amazon’s hourly charging scheme for on-demand instances, for example (as discussed in section 7.3.3.2), which has a great impact on the resultant costs. Hence, a running VM based on an on-demand instance can serve more requests without incurring more costs, if for the purpose of scaling-in it is stopped right before its next charging hour, rather than in the middle of the current charging hour, and it continues to serve at full capacity before it is selected as the candidate VM for scaling-in. This also suggests that for cost-saving we should choose a running VM which has the shortest time left before entering its next charging hour as the candidate for scaling in.
Table 7.23 Pseudo code snippets of our Auto Scaling (FLUCAS) Algorithm

Table 7.23 gives the algorithm for constructing our auto scaling (FLUCAS) algorithm. The scaleOutTimer is used to record the time length during which the total unused CPU, I/O, or memory capacity is less than the predefined SCALE_OUT_CAPACITY_THRESHOLD, and is checked against the SCALE_OUT_TIME_LENGTH, for example two seconds, to trigger the scaling-out policy. The cooldown period is used to avoid repeatedly triggering the scaling-out policy to launch unrequired VMs during the time period when a newly added VM is still in its start-up process after being launched for scaling-out. Based on the performance figures for VM representation, as listed in Table 7.9, we used the average 100 seconds of VM start-up time and 10 seconds of standard deviation to generate a random start-up time for launching a new VM and the related cooldown period.

As for scaling-in, when the predicted total unused CPU, I/O, or memory capacity after stopping a VM is greater than the predefined SCALE_IN_CAPACITY_THRESHOLD and the shortest time to a VM’s next charging hour is equal to or less than the double of UPPER_RESPONSE_TIME_THRESHOLD, the policy is triggered with sufficient time for the candidate VM to finish its current workloads and be stopped before it enters the next charging hour.

```java
1: RuntimeCloudSim.autoScaling( )
2: if isStillInCooldownPeriod( ) == TRUE then
3:    return;
4: end if
5: if getTotalUnusedCapacityForScaleOut( ) < SCALE_OUT_CAPACITY_THRESHOLD then
6:   if getScaleOutTimer( ) > 0 then
7:     if getSimulationTimer( ) - getScaleOutTimer( ) >= SCALE_OUT_TIME_LENGTH then
8:       setCooldownTimer(generateVmStartUptime( )) and setScaleOutTimer(-1);
9:       startUpOneNewVM(getCooldownTimer( ));
10:  end if
11: else
12:    setScaleOutTimer(getSimulationTimer( ));
13: end if
14: else
15:    setScaleOutTimer(-1);
16: if getTotalUnusedCapacityForScaleIn( ) > SCALE_IN_CAPACITY_THRESHOLD then
17:   for vm in vmSortedSet do
18:     if vm.hasShortestTimeToNextChargingHour( ) and
19:        vm.getTimeToNextChargingHour( ) <= (2 × UPPER_RESPONSE_TIME_THRESHOLD) then
20:       vm.setAsCandidateForScaleIn( );
21:       stopVm(vm, vm.getTimeToFinishCurrentWorkloads( ));
22:     end if
23:   end for
24: end if
25: end RuntimeCloudSim.autoScaling
```
To ensure auto scaling timing is working correctly, this auto scaling algorithm must be repeatedly executed at a period not greater than the value of UPPER_RESPONSE_TIME_THRESHOLD.

To demonstrate how our algorithm worked, we conducted three runs of 25-second simulations based on three different auto scaling algorithms, including our proposed FLUCAS, the CPU-reactive, and the Compressed algorithm introduced by [166]. Table 7.24 shows the configurations for the VM and workloads, by which the only one default running VM would reach 100% CPU utilisation when 10 asCloudlets were received at the same time. The threshold of CPU utilisation for both the CPU-reactive and Compressed ones to trigger their scale-out policy was set to 80% (meaning the unused capacity is one-fifth of the total capacities), whilst the value of SCALE_OUT_CAPACITY_THRESHOLD for the FLUCAS was set to 70% (which also stands for one-fifth of the total unused CPU capacity of one VM, i.e. 350%). Both the VM startup time and cool-down time for auto scaling were set as six seconds. The VM charging period was assumed to be 20 seconds, meaning when a VM’s lifetime is short than 20 second, a charge of 0.1 pounds sterling for the usage of one charging period will incur. Likewise, if the lifetime of a VM is between 20 (inclusive) and 40 (exclusive) seconds, a charge of 0.2 pounds sterling for the usage of two charging periods will incur.

<table>
<thead>
<tr>
<th>Simulation Objects</th>
<th>CPU Capacity/Load (MIPS)</th>
<th>Memory Capacity/Load (Mbytes)</th>
<th>I/O Capacity/Load (IOPS)</th>
<th>VM Startup Time (sec.)</th>
<th>VM Charging Period (sec.)</th>
<th>Cost per Charging Period (£)</th>
</tr>
</thead>
<tbody>
<tr>
<td>VM (appServer)</td>
<td>1,000</td>
<td>1,000</td>
<td>1000</td>
<td>6</td>
<td>20</td>
<td>0.1</td>
</tr>
<tr>
<td>Workloads (asCloudlet)</td>
<td>100</td>
<td>10</td>
<td>100</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
</tbody>
</table>

Table 7.24 Configurations for App Servers and Workloads in the Comparison Simulations

Figure 7.9 shows both the number of cloudlets (i.e. asCloudlets) submitted and the resultant number of running VMs based on different auto scaling algorithms. During the first six seconds of the warm-up period in each of these three simulations, seven cloudlets were submitted at each second. Then the number of cloudlets increased to eight, ten, and eight, at the 7th, 8th, and 9th second, respective, to represent a small burst of workloads. At the 10th second, the number of cloudlets dropped to seven again and at the 20th second there was another burst with the number of cloudlets going up to nine.

The number of running VM based on our proposed FLUCAS algorithm was one (represented by the red line) throughout the simulation, whilst the number went up to two at the 13th second when based on CPU-reactive (the purple line), and the number kept increasing until it reached seven when based on Compressed (the light blue line). According to [166], the Compressed scaling algorithm
always reserves a number of spare VMs, for example two as pre-defined in this simulation, in order to cope with a potential sudden burst of incoming workloads and maintain a proper response time, whilst the CPU-reactive one triggers the action of scaling out/in whenever the amount of workloads reaches a pre-defined upper/lower threshold. As our proposed FLUCAS takes both the past and future predicted workloads into consideration, a small burst of workloads, such as those in these comparison simulations, would not trigger the scaling out action.

Figure 7.9 Number of Cloudlets and Running VMs in the Comparison Simulations Using Different Auto Scaling Algorithms

To further demonstrate the differences between the FLUCAS and the other two algorithms when taking into account the charging period of each VM, we conducted another three runs of 39-second simulations with the given number of cloudlets and results shown in Figure 7.10. Except that the Compressed (represented by the light blue line) launched two new VMs at the very beginning of the simulation, both CPU-reactive (the purple line) and FLUCAS (the red line) launched a new VM after encountered the first burst of workloads. There was a delay between launching a new VM and having it ready to work because of the assumed VM startup time of six seconds. When the workloads dropped to below the threshold for triggering a scaling-in action, the CPU-reactive stopped the newly started VM right away, whereas the FLUCAS kept the newly started VM running until both the workloads were low and the VM reached the end of its current charging period.
Figure 7.10 Number of Cloudlets and Running VMs in the Comparison Simulations Using Different Auto Scaling Algorithms

Table 7.2 provides the usage of VM charging periods, the costs and the response time from the simulations based these three algorithms. Though the Compressed could better handle the burst of the workloads in terms of having the shortest average response time, the FLUCAS used less than half of the VM charging periods consumed by the Compressed while still maintaining a better average response time than the CPU-reactive.

<table>
<thead>
<tr>
<th>Auto Scaling Algorithm</th>
<th>Number of Launched VMs</th>
<th>Usage of VM Charging Periods (20 sec. per period)</th>
<th>Costs (£0.01 per period)</th>
<th>Avg. Response Time (sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLUCAS</td>
<td>2</td>
<td>4</td>
<td>0.04</td>
<td>0.545</td>
</tr>
<tr>
<td>CPU-reactive</td>
<td>3</td>
<td>4</td>
<td>0.04</td>
<td>0.677</td>
</tr>
<tr>
<td>Compressed</td>
<td>7</td>
<td>9</td>
<td>0.09</td>
<td>0.339</td>
</tr>
</tbody>
</table>

Table 7.25 Usages of VM Charging Periods, Costs and Response Time in the Comparison Simulations Using Different Auto Scaling Algorithms

7.4 Cost-effectiveness Analysis

It was based on the conclusions of both section 7.3.3 on virtual machine modelling and section 7.3.4 on dynamic workload modelling that the configurations shown in Table 7.26 were adopted for
our cloud simulations. Each of these cloud simulations started with one web application server and one database server both based on a reserved VM (US$81 per year) emulating the EC2 t2.micro instance running Ubuntu 14.04.1. These two VMs would never be shut down even when there was not any workload. For simplicity, we assumed all VMs have the same fixed CPU, memory, and I/O capacities. The memory capacity was set as 2,560 Mbytes, rather than the default 1,024 Mbytes in a t2.micro instance, because we found that the operation system could overcommit memory for up to 250% of the physical memory size, as mentioned in section 7.3.4.1.

<table>
<thead>
<tr>
<th>Simulation Objects</th>
<th>Number and Costs (US$) of Reserved VM</th>
<th>CPU Capacity (MIPS)</th>
<th>Memory Capacity (Mbytes)</th>
<th>I/O Capacity (IOPS)</th>
<th>Max. No. of Processes/Threads</th>
<th>New VM Startup Time (sec.)</th>
<th>Costs of On-demand VM (US$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>App Server</td>
<td>1 ($81/year)</td>
<td>3,350</td>
<td>2,560</td>
<td>4,960</td>
<td>256</td>
<td>100</td>
<td>$0.014/hour</td>
</tr>
<tr>
<td>DB Server</td>
<td>1 ($81/year)</td>
<td>3,350</td>
<td>2,560</td>
<td>4,960</td>
<td>150</td>
<td>100</td>
<td>$0.014/hour</td>
</tr>
</tbody>
</table>

Table 7.26 Default Set-ups of the VMs in the 10-day Simulation

The maximum number of concurrent processes for the application server was set as 256 to emulate an Apache process-based server, whilst the maximum number of concurrent threads for the database server was set as 150 to emulate a MySQL server. Whenever a new server was needed due to an auto scaling out action, a VM would be created based on an on-demand EC2 t2.micro instance (US$0.014 per hour) with a startup time of 100 seconds.

For the web session workload emulation, four groups of emulated users, including patients in the intervention group and their carers, healthcare professionals, emergency tele-consultants, and User Agents, as stated in section 7.3.4.4 and modelled in section 6.3, were created to generate web session requests (i.e. workloads) to the application and database servers (or the App&DB module). Except that the requests made by the User Agents were to upload alerts and monitoring data (or the so-called Type III requests as described in section 7.3.4.3) to the database server, those made by the other three groups of users were general web user operations, such as system login, information browsing, and data creations and updates (or the Type I and II requests). Meanwhile, we adopted the conclusions set forth in Table 7.21 as our default set-ups to represent the asCloudlet and dbCloudlet for the three types of requests in the simulations. Under these set-ups, the application server could execute about 4,500 asCloudlets of Type I requests or about 9,000 asCloudlets of Type III requests (when not taking into account the possible deviations of cloudlet CPU, memory and I/O loads) per second without causing any delay, whilst the database server could execute about 24,800 dbCloudlets of Type III requests per second without causing any delay.
Regarding the workloads generated by the patients (in the intervention group) and their carers who went online, we used the models stated in sections 6.3.1.4 and 6.3.3.2. Table 7.27 gives the grouping of patients (in the intervention group) and their carers, based on each individual’s Internet habit. A notable assumption was that the online activities of a patient were independent from those of his/her carer (if he/she had one carer). When the predicted number of steps for a patient or carer’s Type I request was zero (using Equations in section 6.3.14), we simply assumed that he/she did not go online on that day. However, a user session sent to the server could only contain Type I (browsing) requests without any Type II (database related) request. Table 7.28 shows the default set-ups of a session generated by patients (in the intervention group) and their carers.

<table>
<thead>
<tr>
<th>Web Users (Intervention group)</th>
<th>Total</th>
<th>Non-Internet Users</th>
<th>Low-Internet Users</th>
<th>Median-Internet Users</th>
<th>Heavy-Internet Users</th>
</tr>
</thead>
<tbody>
<tr>
<td>Patients</td>
<td>49,924</td>
<td>29,017</td>
<td>3,311</td>
<td>14,320</td>
<td>3,276</td>
</tr>
<tr>
<td>Carers</td>
<td>33,300</td>
<td>7,326</td>
<td>4,156</td>
<td>17,662</td>
<td>4,156</td>
</tr>
</tbody>
</table>

**Table 7.27 Grouping of Patients (in the intervention group) and Carers based on their Internet Habits**

<table>
<thead>
<tr>
<th>Web Users (Intervention group)</th>
<th>Average Think Time for Type I Requests (sec.)</th>
<th>Number of Steps for Type I Requests</th>
<th>Average Think Time for Type II Requests (sec.)</th>
<th>Number of Steps for Type II Requests</th>
</tr>
</thead>
<tbody>
<tr>
<td>Patients</td>
<td>50 (SD=20)</td>
<td>1 ~ 14</td>
<td>70 (SD=20)</td>
<td>0 ~ 14</td>
</tr>
<tr>
<td>Carers</td>
<td>50 (SD=20)</td>
<td>1 ~ 14</td>
<td>70 (SD=20)</td>
<td>0 ~ 14</td>
</tr>
</tbody>
</table>

**Table 7.28 Default Set-ups for a Session Generated by Patients and Carers with Different Internet Habits**

<table>
<thead>
<tr>
<th>Web Users</th>
<th>Number of Users</th>
<th>Average Think Time for Type I Requests (sec.)</th>
<th>Number of Steps for Type I Requests</th>
<th>Average Think Time for Type II Requests (sec.)</th>
<th>Number of Steps for Type II Requests</th>
</tr>
</thead>
<tbody>
<tr>
<td>Professionals</td>
<td>15</td>
<td>50 (SD=20)</td>
<td>5 ~ 8</td>
<td>70 (SD=20)</td>
<td>1 ~ 14</td>
</tr>
<tr>
<td>Tele-consultants</td>
<td>13</td>
<td>40 (SD=6)</td>
<td>2 ~ 4</td>
<td>50 (SD=8)</td>
<td>1 ~ 14</td>
</tr>
</tbody>
</table>

**Table 7.29 Default Set-ups for a Session Generated by Professionals and Tele-consultants**
Table 7.29 provides the default set-ups for a session generated by the professionals and tele-consultants. These figures were defined to emulate real-world scenarios, while ensuring that the maximum length of each session was less than 900 seconds for the professional, and 480 seconds for the tele-consultant.

<table>
<thead>
<tr>
<th>Type III Request</th>
<th>Timing of Workload Generation</th>
<th>Number of Step</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vital Sign Data and Alert</td>
<td>Upon the completion of a patient’s vital sign measurements</td>
<td>1</td>
</tr>
<tr>
<td>Fall Alert</td>
<td>Upon the detection of a fall</td>
<td>1</td>
</tr>
<tr>
<td>Location Tracking Data</td>
<td>Every 150 to 600 seconds (depending on a repeatedly generated random number on each User Agent)</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 7.30 Default Set-ups for a Session Generated by the User Agent

There were two kinds of monitoring data, including vital signs and location tracking (or activity monitoring) and two kinds of alerts, including vital sign and fall detection alerts, generated by the User Agent, each constituting one user session with a step number of one and a pair of asCloudlet and dbCloudlet. It is important to note that for reliability reason, there was no need to send a separate vital sign alert apart from the monitoring vital sign data to the server, because the latter already contained enough information to signify the critical conditions of the monitored patient. (However, in such a situation, it might well be necessary to send a separate vital sign alert to the patient’s carer and/or associated emergency tele-consultants, depending on the desired settings.) Table 7.30 shows the default set-ups of a session generated by the User Agent.

7.4.1 Results of Cloud Simulations based on Default Set-ups

Due to an ‘out-of-memory’ issue that always occurred in our cloud simulations after the CloudSimEx (Web) platform had received and executed a large amount of workloads, we hence split the desired cloud simulations into two phases. Firstly, we performed a number of 10-day simulations and stored all the generated workload data in disk files without submitting them to the CloudSimEx platform so that we could successfully record the quantity and distribution of workloads generated from different emulated entities and activities. Then we redid the simulations but allowed the workloads to be submitted to the CloudSimEx platform starting from a specific simulation time and recorded the results before a crash occurred. In the following subsections, we describe the results of those two-phase simulations and explain our findings in detail.
7.4.1.1 Workloads from Emulated Patient Online Activities

As shown in Table 7.27, in the intervention group there were 29,017 patients who did not use the Internet at all, whilst the other 20,907 patients were divided into subgroups of light-, median-, or heavy-Internet users. Figure 7.11 shows the average number of web session requests which were made by patients in the intervention group and recorded by time of the 10-day simulation with a five-minute recording window. On average, there were 16,392 session requests made per day, representing that less than one-third of the patients in the intervention group went online each day. These sessions, which consisted of 42,677.2 Type I and 30,728.2 Type II requests, were almost evenly distributed between around 07:00 and 21:00, as we assumed patients might go online anytime when they were awake. Each session consisted of on average 2.6 Type I requests and 1.87 Type II requests.

There were also on average 934 undone sessions per day as affected by patients’ other activities, such as taking vital signs and attending A&E. It was these undone sessions that caused both a V-shape notch to the area represented by the average number of session requests at around 08:30 and a slope between 17:00 and 21:00. When we looked at the records of each simulation day, the maximum number of sessions created within a five-minute recording window was 132, which in our opinion was relatively small in terms of workloads for the web application and database servers in comparison with their processing capacities.
7.4.1.2 Workloads from Emulated Carer Online Activities

![Figure 7.12 Avg. Number of Web Session Requests Made by Carers by Time of Day](image)

Since we adopted the same formulas to emulate the workloads generated by both patients and carers, the resultant distributions of session requests from these two subgroups of users during a day were very similar. Figure 7.12 depicts the average number of web session requests made by carers and recorded by time of the 10-day simulation. It was observed that between around 09:00 and 20:30, the average number of session requests made within one five-minute window was around 120. When we inspected the workloads on each simulation day, the maximum number of sessions within a five-minute window was 163, which was relatively small in terms of workloads for the web application and database servers.

7.4.1.3 Workloads from Emulated Professional Online Activities

As mentioned previously, on each day 15 healthcare professionals were responsible for reviewing the monitoring data of those (patients in the intervention group) who were in need of medical care the day before due to a fall or a vital-sign-exceeding-threshold event. Table 7.31 summarises the relevant figures. For example, on the first simulation day there were 330 patients requiring medical care due to vital sign events, and another 10 patients due to fall events. Therefore on the second day, the total number of reviews needed to be undertaken by the professionals was 340 and the average number of review for each professional was 23.
Table 7.31 Number of Patients Requiring Medical Care and Average Number of Reviews per Healthcare Professional

To emulate the workloads aroused from the reviews, we assumed that all professionals worked between 6:00 and 20:00 and each professional would perform the required reviews some time in between by retrieving patients’ personal health records and uploading their comments or diagnosis results from/onto the servers. Figure 7.13 shows the average number of web session requests made by all professionals and recorded by time within day two to day 10. (The first day was excluded from the calculation, as there was no review done on that day.) When we looked at the records of each simulation day, the maximum number of sessions created within a five-minute recording window was 9. In comparison with the processing capacities of the web application and database servers, these workloads were extremely small.

Figure 7.13 Avg. Number of Web Session Requests Made by Professionals by Time of Day
7.4.1.4 Workloads from Emulated User Agent Monitoring Data Uploading

As we assumed that each patient in the intervention group had a User Agent, and that all User Agents were working properly during the 10-day simulation, there were in total 49,924 User Agents trying to repeatedly upload location tracking data, vital sign measurements (including alerts), and fall detection alerts. Figure 7.14 shows the average number of web session requests made by User Agents for uploading location tracking data. On average, there were 11,515,825 web sessions generated per day in this regard. Except for a large variation of workloads at the initial stage during the first 10 minutes of the first simulation day, within all the other five-minute recording windows the average number of web session requests was very stable with a mean of 39,985.5. When we looked at each simulation day’s records, the maximum number of sessions created within a five-minute recording window was 44,836, which occurred at the second recording window on the first day. In comparison with workloads from other sources, these session requests generated from the User Agents apparently constituted the main workloads for the web application and database servers.

![Avg. Number of Session Requests for Uploading Location Tracking Data by Time](image)

**Figure 7.14 Avg. Number of Web Sessions for Uploading Location Tracking Data by Time**

Figure 7.15 depicts the average number of web session requests made by User Agents for uploading vital sign measurement data (including vital-sign-exceeding-threshold alerts) and fall detection alerts. As in these 10-day simulations we adopted the revised model for the timing for patients to take vital signs in the afternoon as stated in section 6.4.3, the distribution of the session workloads shown in Figure 7.15 was almost identical to Figure 6.19 for the average number of vital sign measurements, except that the session workloads were related to patients in the intervention group only and hence their quantity was about half of the vital sign measurements. When we examined the figures on each simulation day, the maximum number of session requests made within a
five-minute recording window for uploading vital sign data and fall alerts was 2,711 during the time period between 08:25 and 08:30 on the second simulation day. (On some other days, the maximum number of session requests occurred during 08:30 and 08:35.) Throughout an entire simulation day, we could also observe that there were a few sessions for uploading fall alerts in some five-minute recording windows. However, they were not recognisable on the chart because their quantity was very small. Hence we could conclude that the surge of workloads for uploading vital sign data represented the second largest source of workloads for the server systems.

![Figure 7.15 Avg. Number of Web Sessions for Uploading Vital Sign Data and Fall Detection Alerts by Time](image)

**Figure 7.15 Avg. Number of Web Sessions for Uploading Vital Sign Data and Fall Detection Alerts by Time**

### 7.4.1.5 Workloads of asCloudlets and dbCloudlets from All Emulated Entities and Activities

When each of the abovementioned session requests was submitted to our revised CloudSimEx (Web) simulation platform, it was first stored in the event queue of the CloudSim. Then when the session’s event time was due, a sequence of or a pair of cloudlets (including both an asCloudlet and a dbCloudlet) were timely created with dynamically assigned CPU, memory, and I/O loads based on the request type, step number and think time of the session. These asCloudlets and dbCloudlets were then sent to the web application and database servers respectively for execution. Unfortunately, the CloudSimEx platform could not bear a large amount of workloads as many as in our cases and thus every time it caused an ‘out-of-memory’ error when the number of submitted/executed cloudlets had reached 1,041,138. After some efforts in revising the ways of creating/cleaning-up objects and linked lists in CloudSimEx (Web) platform and increasing the size of the heap space for the Java Runtime Environment (JRE) (based on Eclipse Java EE IDE 4.4.1) from 400 Mbytes to 1,536 Mbytes, we just could double the number of successfully submitted/executed cloudlets to around 2 million.
To better examine how our design and modelling of the App&DB module responded to the largest amount of the emulate workloads under the circumstances of the abovementioned ‘out-of-memory’ issue, we performed the following two runs of cloud simulations separately. The first run of simulation started from the 0-th second and crashed at around the 7,200-th second with records of 1,876,182 executed cloudlets stored on disk files. The second run was started from the 0-th second too, but only submitted the sessions created after the 27,000-th second (or 07:30, on the first day in the simulation time) to the CloudSimEx (Web) platform and it crashed at around 33,000-th second with records of 1,461,325 cloudlets stored. (In order to speed up the simulation process and reduce the usage of compute resources on the desktop system running the simulations, results were written to the disk every 1,200 or 1,800 seconds in the simulation time.)

Figure 7.16 shows the total CPU loads on the application and database servers respectively during the aforementioned two separate time periods, i.e. 00:00 to 02:00 and 07:30 to 09:00. The second period covered the time interval between 08:25 and 08:35 during which the number of session requests reached the largest on each simulation day. The maximum CPU load recorded within a five-minute recording window was 16,368.82 million instructions (or approximately 54.56 MIPS in a second) on the web application server and it was 5,609.37 million instructions (or around 18.7 MIPS in a second) on the database server.

![CPU Loads on App & Database Servers during Two Time Periods](image_url)

**Figure 7.16 CPU Loads on Web Application and Database Servers during Two Separate Time Periods**

Figure 7.17 provides the total memory loads on the application and database servers respectively during the two separate time periods. The maximum memory load on the web application server
within a five-minute recording window was 358,627 Mbytes (or approximately 1,195.42 Mbytes in a second), whilst it was 11,210.6 Mbytes (or around 37.37 Mbytes in a second) on the database server.

Figure 7.17 Memory Loads on Web Application and Database Servers during Two Separate Time Periods

Figure 7.18 I/O Loads on Web Application and Database Servers during Two Separate Time Periods

Figure 7.18 shows the total I/O loads on the application and database servers respectively during the two separate time periods. The maximum I/O load on the web application server within a five-minute recording window was 50.61 operations, which was not recognisable on the chart, whilst the maximum I/O load was up to 8,734.54 operations (or approximately 29.12 IOPS in a second) on the database server.
7.4.1.6 Estimation of Costs and Effectiveness

<table>
<thead>
<tr>
<th>Time Period</th>
<th>Servers (VM id)</th>
<th>Total Finished Cloudlets</th>
<th>Successful Rate</th>
<th>Avg. Delay Time (sec.)</th>
<th>Max. Delay Time (sec.)</th>
<th>Avg. Response Time (sec.)</th>
<th>Max. Response Time (sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>00:00-02:00</td>
<td>App (1)</td>
<td>938,091</td>
<td>100%</td>
<td>0</td>
<td>0.001</td>
<td>0.001</td>
<td>0.002</td>
</tr>
<tr>
<td>00:00-02:00</td>
<td>DB (2)</td>
<td>938,091</td>
<td>100%</td>
<td>0</td>
<td>0</td>
<td>0.001</td>
<td>0.002</td>
</tr>
<tr>
<td>07:35-09:00</td>
<td>App (1)</td>
<td>730,662</td>
<td>100%</td>
<td>0</td>
<td>0</td>
<td>0.001</td>
<td>0.003</td>
</tr>
<tr>
<td>07:35-09:00</td>
<td>DB (2)</td>
<td>730,662</td>
<td>100%</td>
<td>0</td>
<td>0</td>
<td>0.001</td>
<td>0.003</td>
</tr>
</tbody>
</table>

Table 7.32 Results of Workload Execution during Two Separate Time Periods

Table 7.32 summarises the results of workload execution on the web application and database servers respectively during the two separate time periods. The average delay time in both runs of simulations was zero, meaning all cloudlets were immediately served after being submitted to the servers. Meanwhile, the average response time was 0.001 seconds in both runs and the maximum response time was 0.003 seconds in the second run. Since the auto scaling policy based on our FLUCAS mechanism was not triggered during the time period of peak workloads, we believed that such a default set-up based on one web application server and one database server can well serve all the emulated workloads across other periods of time on each day. Here the response time was calculated by subtracting the submission time of a request (including both an asCloudlet and a dbCloudlet) from the latest completion time of either the asCloudlet or dbCloudlet without taking the network transmission time into account. These outcomes also suggested that with just one web application server and one database server, the App&DB module was able to provide good services with very limited response time when the default set-ups were based on the modelling work stated in both sections 6.3 and 7.3.

Table 7.33 shows the estimated costs of US$162 (or about £105 pounds sterling) per year for deploying the server systems of the App&DB module based on our defined healthcare telemonitoring scenario. In this scenario, 49,924 patients’ activities of daily living, including vital signs, falls, and real-time movements, were telemonitored with data and alerts being sent to the servers, as well as the online activities of a number of stakeholders, including 15 healthcare professions, 13 emergency teleconsultants, and 25,974 carers to browse, update, and store relevant data on the telemonitoring system were supported. We believed that this is a relatively low cost for running both a web application server and a database server for a year.
Table 7.33 Estimated Costs of Required Servers on the App&DB Module

<table>
<thead>
<tr>
<th>Servers</th>
<th>Number and Costs (US$) of Reserved VM(s)</th>
<th>Number and Costs (US$) of On-demand VM(s)</th>
<th>Subtotal Costs (US$)</th>
<th>Total Costs (US$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>App Server</td>
<td>1 ($81/year)</td>
<td>0 ($0.014/hour)</td>
<td>$81/year</td>
<td>$162/year (or £105/year)</td>
</tr>
<tr>
<td>DB Server</td>
<td>1 ($81/year)</td>
<td>0 ($0.014/hour)</td>
<td>$81/year</td>
<td></td>
</tr>
</tbody>
</table>

Table 7.34 Estimated Equipment Costs of the Proposed Telemonitoring System

<table>
<thead>
<tr>
<th></th>
<th>Costs of Servers (£ per year)</th>
<th>Costs of Commodity Desktop Computers (£ per year)</th>
<th>Costs of Peripherals (£ per year)</th>
<th>Total Costs (£ per year)</th>
</tr>
</thead>
<tbody>
<tr>
<td>System-wide</td>
<td>105</td>
<td>2,100</td>
<td>4,521,117</td>
<td>4,523,322</td>
</tr>
<tr>
<td>Per Patient</td>
<td>0.0021</td>
<td>0.042</td>
<td>90.56</td>
<td>90.60</td>
</tr>
</tbody>
</table>

If we assumed that the monitor centre needs 30 commodity desktop computers, each at the cost of £350, to support the work of emergency tele-consultants and healthcare professionals, the annualised computer costs over a five-year period would be around £2,100. Table 7.34 gives the total costs of computer hardware and equipment per year based on this assumption, where the estimated peripheral costs (i.e. three BLE sensors, one TTL cable, one Arduino UNO R3, and one eHealth Sensor Platform)\(^{14}\) and total costs per patient per year were about £90.56 and £90.60 pounds sterling, respectively. In comparison with the equipment costs (i.e. costs of home based units and monitor peripherals) of about £334-£825 per patient per year in the WSD trial [30], we could further claim that our design is much more cost-effective.

7.4.2 Experiments with Considerations of Database Insertion Time

As discussed in section 7.3.4.2, the average execution time for a MySQL database insertion of 5,000-byte data on an EC2 t2.micro instance was about 0.004 second. In our aforementioned default set-ups for workload representations, we did not take this database processing time into account as there had not been any implementation of database processing events in both the CloudSim and CloudSimEx frameworks. However, we believed that it would be worthwhile to further examine the possible impact of this issue on the performance of the database server. Since the I/O loads for the

\(^{14}\) The total costs of used peripherals in this research were around £452.8 pounds sterling (or an annualised cost of £90.56 over a five-year period), exclusive of the cost of an iPhone 5 at £551.79.
database server incurred from Type II requests were estimated based on assumed database operations, we therefore revised Equation (7-10) as (7-19) to account for the database processing time in the modelling of I/O loads. With regard to the I/O loads from Type III requests, Equation (7-20) was used to make the transformation. The reason that we did not directly use (7-20) to estimate the I/O loads from Type II requests was to account for the uncertainty about what kind of database operations were involved, whilst we were certain that a Type III request would definitely invoke a database insertion.

\[
\begin{align*}
I_{\text{load.from.Type.II}} &= \left( \bar{n}_{\text{operation.per.DB.operation}} \times n_{\text{DB.operation}} \right) \times 0.004 \text{ sec} \times IOPS_{\text{of.the.server}} \quad (7-19) \\
I_{\text{load.from.Type.III}} &= 0.004 \text{ sec} \times IOPS_{\text{of.the.server}} \quad (7-20)
\end{align*}
\]

<table>
<thead>
<tr>
<th>Type of Requests</th>
<th>Cloundlets</th>
<th>No. of I/O Operations</th>
<th>Mean</th>
<th>SD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type II</td>
<td>dbCloudlet</td>
<td>6</td>
<td>0.45</td>
<td></td>
</tr>
<tr>
<td>Type III</td>
<td>dbCloudlet</td>
<td>20</td>
<td>1.5</td>
<td></td>
</tr>
</tbody>
</table>

Table 7.35 Revised Representations of dbCloudlet I/O Loads for Type II and III Requests

Table 7.35 provides the revised representations of dbCloudlet I/O loads for Type II and III requests, respectively. Given that the new values were six and 20 I/O operations, respectively, i.e. about 20 and 100 times of what we set forth in Table 7.21, we perceived that our original modelling without taking into account the database processing time might underestimate its impact.

7.4.2.1 I/O Loads on the Database Server based on the Revised dbCloudlet Representations

![Figure 7.19 I/O Loads on Database Servers based on Revised dbCloudlet Representations](image)
Figure 7.19 shows the I/O loads on the database server during the time period between 08:05 and 09:30 based on the revised dbCloudlet representations. The maximum I/O load recorded within the five-minute recording window between 08:25 and 08:30 was 857,832.8 operations (or approximately 2,859.44 IOPS in a second).

### 7.4.2.2 Estimation of Costs and Effectiveness

Table 7.36 summarises the results of workload execution on the web application and database servers respectively during the time period between 08:05 and 09:30 based on the revised I/O representations. With the large increase of I/O loads on the database server, the average response time was 0.018 seconds, whilst the maximum response time increased to 0.104 seconds. However, we considered the App&DB module, consisting of only one web application server and one database server, was still very cost-effective, as either the average or maximum response times was far under our defined upper response time threshold, i.e. 3.5 seconds.

<table>
<thead>
<tr>
<th>Time Period</th>
<th>Servers (VM id)</th>
<th>Total Finished Cloudlets</th>
<th>Successful Rate</th>
<th>Avg. Delay Time (sec.)</th>
<th>Max. Delay Time (sec.)</th>
<th>Avg. Response Time (sec.)</th>
<th>Max. Response Time (sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>08:05-09:30</td>
<td>App (1)</td>
<td>731,565</td>
<td>100%</td>
<td>0</td>
<td>0.002</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DB (2)</td>
<td>731,564</td>
<td>100%</td>
<td>0</td>
<td>0.002</td>
<td>0.012</td>
<td>0.105</td>
</tr>
</tbody>
</table>

Table 7.36 Results of Workload Execution based on the Revised I/O Representations

Since the auto scaling policy was not triggered during the period of high peak workloads, we believed that the default setting of one web application server and one database server could well serve all requests across other periods of time on each day. As a result, the estimated costs of required servers on the App&DB module were equivalent to about £105 pounds sterling per year, the same as the figure in Table 7.33.

### 7.4.3 Experiments with a Fixed Time Interval for Uploading Location Tracking Data

As stated in section 7.4.1.4, session requests generated from the User Agents for uploading patients’ location data represented the prime workloads for the application and database servers. According to our design as described in section 6.3.2, we adopted a varying time interval of 150 to 600 seconds between every two consecutive upload sessions performed by a User Agent in order to avoid a large number of session requests from different User Agents to occur at the same time. To examine how this strategy worked, we performed some experiments on the generation of session requests based on a number of different fixed time intervals.
Figure 7.20 The Number of Session Requests Generated for Uploading Patients’ Location Tracking Data based on a Fixed Time Interval

Figure 7.20 shows the results of two experiments, each with a fixed time interval of 300 and 600 seconds, respectively, which were recorded on the second simulation day. On the first day of these two experiments, each User Agent started sending its first session request at a random time, and then it repeatedly sent out another request to the App&DB module every 300 or 600 seconds. For comparison, Figure 7.20 also gives the number of sessions generated on the second day of our previous simulations based on a varying time interval of 150 to 600 seconds (represented by the grey line).

<table>
<thead>
<tr>
<th>Time Interval between Two Consecutive Uploads</th>
<th>Max. Number of Sessions Within a Five-Minute Window</th>
<th>Max. Number of Sessions within a One-Second Window</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fixed 300 sec.</td>
<td>49,924</td>
<td>201</td>
</tr>
<tr>
<td>Fixed 600 sec.</td>
<td>25,060</td>
<td>111</td>
</tr>
<tr>
<td>Varying between 150 and 600 sec.</td>
<td>40,328</td>
<td>185</td>
</tr>
</tbody>
</table>

Table 7.37 Maximum Number of Sessions Generated By User Agents based on Different Time Interval for Data Uploading

By inspecting the results, we found that the shorter the fixed time interval we adopted, the smoother figures we obtained, in terms of less variations in the number of sessions submitted within every five-minute recording window throughout the day. Besides, whether a fixed or varying time interval was adopted did not cause significant difference in the maximum number of sessions within a
one-second recording window; but the length of the time interval for uploading patients’ location data did. Table 7.37 provides the maximum number of sessions generated/submitted within both a five-minute and a one-second recording window based on different time intervals. Built upon these experiments, we concluded that due to the large amount of sessions being created, the varying time interval we adopted as part of our default set-ups for cloud simulations did not produced our desired effect by spreading the session requests in time over the day.

7.4.4 Sensitivity Analysis and Comparison with other Auto Scaling Policies

In section 7.3, we described in detail the modelling of both virtual machines and user workloads mostly based on the results of our benchmarking experiments. We considered that most of the figures we adopted for the cloud simulation were quite reasonable with two main exceptions. The first exception was the average storage I/O capacity of 4,960 IOPS we used as part of our default set-ups. Though it was the actual measured value from our experiments, it was much higher than the specifications of EC2 t2.micro instance. According to Amazon, the I/O capacity of the EBS (or Elastic Block Store) storage volume used by t2.micro instances is three IOPS per Gbytes with bursts up to 3,000 IOPS. To assess how the value of the database server I/O capacity affects the results of cost-effective analysis, we tested the outcomes with a lower value of 2,480 IOPS (i.e. a reduction by 50%).

The second exception was the assumption about I/O loads, especially from Type II and III requests, on the database server due to the difficulty of correctly identifying the relationship between I/O operations and database operations. In section 7.4.2, we explained how we revised the model for estimating the I/O loads on the database server to account for the database processing time. However, as the estimation of I/O loads using both Equations (7-19) and (7-20) were proportional to the I/O capacity of the database server when we decreased the I/O capacity value for sensitivity analysis, the I/O loads would decrease accordingly. This would cause the decrease of I/O capacity to make no effect on the execution time of dbCloudlets on the database server. Consequently, for the sensitivity analysis, we decided to use the same figures, i.e. six and 20, respectively, as tabulated in Table 7.35, to represent the I/O loads from Type II and Type III requests on the database server. When 2,480 IOPS was adopted as the storage capacity, such a decision to use the same figures for the representation of I/O loads actually had the effects of both increasing the workloads by two times and limiting the maximum number of database insertions per second to 124 (about a half of the figure of 250 as observed in section 7.3.4.2 regarding the effect of data size on the performance of database operations).
7.4.4.1 Estimation of Costs and Effectiveness based on the FLUSCAS Algorithm

Two runs of simulations were performed to test both how our auto scaling policy responded to the new settings and how requests were served during the peak load period. Table 7.38 shows the results of workload execution based on the revised I/O capacity of 2,480 IOPS of the database server. As the I/O capacity of the database server decreased, both the delay time and response time increased. At the 319th second (or 00:05:19 in the simulation time), the auto scaling policy based on our FLUCAS algorithm was triggered so that a new database server, i.e. DB (3), joined the scaling group 100 seconds later (at the 419th second) to share the workloads. Nevertheless, during the period between the 400th and 425th second, the response time of some requests exceeded 3.5 seconds, our defined upper response time threshold, with the maximum response time up to 6.347 seconds as a result of a prolonged delay time and execution time of dbCloudlets on the default database server, i.e. DB (2). From the 426th second (or 00:07:06) onward, the maximum delay time of dbCloudlets was 0.001 second and the maximum response time of requests was down to 0.134 seconds.

<table>
<thead>
<tr>
<th>Time Period</th>
<th>Servers (VM id)</th>
<th>Total Finished Cloudlets</th>
<th>Successful Rate</th>
<th>Avg. Delay Time (sec.)</th>
<th>Max. Delay Time (sec.)</th>
<th>Avg. Response Time (sec.)</th>
<th>Max. Response Time (sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>00:00-02:10</td>
<td>App (1)</td>
<td>1,001,416</td>
<td>100%</td>
<td>0</td>
<td>0.001</td>
<td>0.0689</td>
<td>6.347</td>
</tr>
<tr>
<td></td>
<td>DB (2)</td>
<td>507,944</td>
<td>100%</td>
<td>0.0308</td>
<td>5.102</td>
<td>0.0689</td>
<td>6.347</td>
</tr>
<tr>
<td>00:07:06 - 02:10</td>
<td>DB (3)</td>
<td>493,471</td>
<td>100%</td>
<td>0</td>
<td>0.001</td>
<td>0.0689</td>
<td>6.347</td>
</tr>
<tr>
<td>07:30-08:47</td>
<td>App (1)</td>
<td>601,309</td>
<td>100%</td>
<td>0</td>
<td>0</td>
<td>0.0143</td>
<td>0.111</td>
</tr>
<tr>
<td>07:30-08:47</td>
<td>DB (2)</td>
<td>300,716</td>
<td>100%</td>
<td>0</td>
<td>0</td>
<td>0.0143</td>
<td>0.111</td>
</tr>
<tr>
<td>07:30-08:47</td>
<td>DB (3)</td>
<td>300,593</td>
<td>100%</td>
<td>0</td>
<td>0</td>
<td>0.0143</td>
<td>0.111</td>
</tr>
</tbody>
</table>

Table 7.38 Results of Workload Execution based on an I/O Capacity of 2480 IOPS of the Database Server

<table>
<thead>
<tr>
<th>Servers</th>
<th>Number and Costs (US$) of Reserved VM(s)</th>
<th>Number and Costs (US$) of On-demand VM(s)</th>
<th>Subtotal Costs (US$)</th>
<th>Total Costs (US$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>App Server</td>
<td>1 ($81/year)</td>
<td>0 ($0.014/hour)</td>
<td>$81/year</td>
<td>$284.64/year</td>
</tr>
<tr>
<td>DB Server</td>
<td>1 ($81/year)</td>
<td>1 ($0.014/hour)</td>
<td>$203.64/year</td>
<td>(or £183.64)</td>
</tr>
</tbody>
</table>

Table 7.39 Estimated Costs of Required Servers on the App&DB Module
For the second run of the simulation between 07:30 and 08:47, all the requests were well served by one web application server and two database servers with a maximum delay time of zero and a maximum response time of 0.111 seconds. Therefore, we concluded that except for the initial stage of the first run of simulation, the App&DB module was able to server all the requests with a predefined requirement for response time based on one web application server and two database servers. Table 7.39 provides the estimated costs of £183.64 pounds sterling for the required servers on the App&DB module for one year. Since the load patterns based on our defined healthcare scenarios would be similar on each day, we could further reduce the system costs by US$41.64 (or £26.86) and avoid the aforementioned longer response time by deploying two default database servers both using reserved VMs from the outset based on capacity planning prior to live implementation.

7.4.4.2 Comparison among Different Auto Scaling Algorithms

To analyse the cost-effectiveness of our design, we also compared the results of workload execution on the database server(s) based on three different auto scaling algorithms, including the FLUCAS, I/O-reactive, and Compressed [166] algorithms. The I/O capacity of each database server was assumed to be 2,480 IOPS and the mean number of I/O operations required on the database server for Type II and III requests was set as six and 20, respectively. The threshold of I/O utilisation for both the I/O-reactive and Compressed ones to trigger their scale-out policy was set to 80% (meaning the unused capacity is one-fifth of the total capacities), whilst the value of SCALE_OUT_CAPACITY_THRESHOLD for the FLUCAS was set to 70% (standing for one-fifth of the total unused I/O capacity of one VM, i.e. 350%). Meanwhile, the threshold for scaling-in for both the I/O-reactive and Compressed was set to 10% (meaning the used capacity is equal to 10% multiplied by the number of VMs before scaling in), whereas the value of SCALE_IN_CAPACITY_THRESHOLD for the FLUCAS was set to 280% (meaning the used capacity is equal to 20% of the total I/O capacity of one VM after scaling-in). For the Expressed algorithm, the number of spare VMs was set to two.

Figure 7.21 depicts both the number of VMs launched to serve as the database servers based on these three algorithms, and the workloads, in terms of the required number of I/O operations, submitted to the database servers recorded during two separate time periods of simulations, i.e. 00:00 to 02:00 and 07:30 to 08:42. As stated in section 7.3.5.2, a possible drawback of a general threshold-reactive algorithm is that the scaling-out policy might never be triggered due to a cap on the maximum number of concurrent processes/threads running on the server system. Unfortunately, this problem did happen to the I/O-reactive algorithm during the simulations, causing no any new VMs to be added to share the heavy I/O loads.
Table 7.40 Results of Workload Execution based on Different Auto Scaling Algorithms

<table>
<thead>
<tr>
<th>Time Period</th>
<th>Auto Scaling Algorithms</th>
<th>Number of Launched Server(s)</th>
<th>Usage of VM Charging Periods (per hour)</th>
<th>Avg. Delay Time (sec.)</th>
<th>Max. Delay Time (sec.)</th>
<th>Avg. Response Time (sec.)</th>
<th>Max. Response Time (sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>00:00-02:00</td>
<td>FLUCAS</td>
<td>2</td>
<td>2</td>
<td>0.0154</td>
<td>5.102</td>
<td>0.0689</td>
<td>6.347</td>
</tr>
<tr>
<td>00:07-02:00:00</td>
<td>FLUCAS</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0.001</td>
<td>0.0153</td>
<td>0.134</td>
</tr>
<tr>
<td>00:00-02:00</td>
<td>I/O-reactive</td>
<td>1</td>
<td>1</td>
<td>146.9238</td>
<td>414.264</td>
<td>148.1525</td>
<td>414.645</td>
</tr>
<tr>
<td>00:06-02:00</td>
<td>Compressed</td>
<td>3</td>
<td>5</td>
<td>0</td>
<td>0.001</td>
<td>0.0132</td>
<td>0.138</td>
</tr>
<tr>
<td>07:30-08:47</td>
<td>FLUCAS</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0.0143</td>
<td>0.111</td>
</tr>
<tr>
<td>07:30-08:47</td>
<td>I/O-reactive</td>
<td>1</td>
<td>1</td>
<td>231.5015</td>
<td>900.166</td>
<td>232.7277</td>
<td>900.573</td>
</tr>
<tr>
<td>07:30-08:47</td>
<td>Compressed</td>
<td>6</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0.0095</td>
<td>0.112</td>
</tr>
</tbody>
</table>

Table 7.40 provides another perspective of comparison. Since the I/O-reactive algorithm had no chance to trigger its auto scaling-out policy throughout the simulations, it caused a maximum response time of about 900 seconds and an average response time of about 232 seconds during the second time period. Both were generally not acceptable. If we assumed that the pricing of all VMs were based on an hourly-charging scheme, the usage of VM charging periods for the FLUCAS, I/O-reactive, and Compressed was two, one, and six, respectively, during the first time period, while it was eight for the Compressed during the second time period. This meant that the cost of running all the launched servers based on the Compressed algorithm was 3.25 times of that based on the
FLUCAS. Although the Compressed achieved the best performance, in terms of the lowest response time, in both time periods, we are confident that the FLUCAS that provided comparable performance most of the time, except some prolonged response time occurred within an interval of 25 seconds between the 400th (or 00:06:40) and 425th (or 00:07:05) second, is reasonably acceptable and cost-effective. Actually, when we looked at the time period between 00:07:06 and 02:00:00, the FLUCAS did have similar performance to the Compressed. Besides, through careful capacity planning, we could deploy enough servers and capacity at the beginning of the live implementation to avoid the undesirable overloading, such as the situation mentioned above during the said 25-second period.

7.5 Evaluation and Discussion

The purpose of our work stated in this chapter was to investigate how we could leverage mobile cloud computing to well support our cost-effective design of real-time home healthcare telemonitoring in terms of enhanced system performance and scalability and reduced costs. By improving and customising an existing cloud simulation toolkit, the CloudSimEx framework, together with our proposed modelling work and novel auto scaling algorithm, we were able to perform real-world-like simulations that fitted our very specific telehealthcare scenarios and produced critical evidence for cost-effectiveness. The findings clearly suggest that it is highly feasible and desirable to use commodity cloud services to well support load-intensive real-time healthcare telemonitoring in a cost-effective way. For instance, both the conclusions drawn in sections 7.4.1.6 and 7.4.2.2 demonstrate that the proposed App&DB module consisting of one web application and one database server built upon a cloud infrastructure at a yearly cost of about £105 pounds sterling can serve tens of millions of web user sessions a day with reasonable performance. In addition, the results of experiments on the comparison of three different auto scaling algorithms, as discussed in section 7.4.4.2, manifest the novelty of our proposed FLUCAS.

To the best of our knowledge, at the time of this writing, our work is the only one that integrates a large amount of emulated web session workloads from a telemonitoring domain with dynamic provisions of cloud resources which were carefully modelled through empirical benchmarking experiments. It is our belief that not only do the modelling work and the enhancement of the CloudSimEx framework (and CloudSim) provide a good foundation for other researchers to better devise their cloud simulations, but the approach and results of our simulations also help justify the great potential of achieving cost-effective home telehealthcare.

Nevertheless, due to both the limitation of the simulation framework and the complexity of the involved problem domains, we were not able to factor all determinants into our modelling work. For example, although the reliability of cloud services, as well as the overall performance of cloud
computing, is highly dependent on the underlying network infrastructure, we did not incorporate network related variables into our design. Besides, we were not able to thoroughly examine the relationship between disk I/O and database operations when modelling workloads for cloud simulations. Another uncovered but important issue was database scalability, which is also a challenging research subject in the field of cloud computing. Therefore, we believe that for future work it is worthwhile to further explore the implications of network dynamics, disk I/O and database performance, and database scalability when constructing and evaluating cloud simulations.

Finally, with both the increasing uptake of mobile cloud services and enhancements in supporting technologies, we are confident that mobile cloud computing is the right direction to go for the cost-effective design of healthcare telemonitoring. A number of emerging cloud-based services that were not covered in our work might well pave the way for the development of future telemonitoring systems. As such, more work will be required to systematically evaluate and identify those potential building blocks for the early realisation of a widely adopted, cost-effective telehealthcare solution in a global context.
8. Conclusions and Future Work

8.1 Conclusions

The rise in both ageing and chronic disease populations has highlighted a pressing demand for better access to quality healthcare at home. Meanwhile, studies have shown that home-based treatments for older patients as a substitute for hospital care can produce better clinical outcomes and reduce healthcare expenditure. However, there remains a considerable question relating to low adoption rate of home telehealthcare technologies due to a lack of robust evidence for cost-effectiveness. In light of both the epoch-making advancements in smartphone-centric technologies and the pervasive uptake of smartphones, we believed that there is an excellent opportunity to bring telehealthcare into the home, if cost-effective smartphone-based solutions are in place. Consequently, we set up as our core objective the cost-effective design of a real-time home healthcare telemonitoring system based on mobile cloud computing.

A broad literature review, covering telehealthcare, smart home, IoT, sensor technologies, cloud computing, and mobile cloud computing, was conducted to identify both the enabling technologies essential for our cost-effective design and the unsolved problems in related work. A proof-of-concept system consisting of three main monitoring functions, namely vital sign, safety (for fall detection) and movement pattern monitoring (for real-time indoor location tracking), was developed based on smartphone-centric technologies. The main contributions of this work include:

- **Fall detection** (chapter 4): The first contribution is the realisation of a low-cost, easy-to-use, less intrusive, but feasible and robust solution. This solution directly addresses, in particular, the two commonly found issues, i.e. the need for fixed placement of the phone or sensors, and over draining of battery power, in related work. The second is the achievement of attaining sensitivity and specificity of both 95% for successfully detected falls and recognised non-fall activities, respectively, in our trials, plus specificity of 100% for effectively identified device drops or throws.

- **Indoor location tracking** (chapter 5): The first contribution is a solid proof that cost-effective design of indoor real-time location tracking is possible based on commodity smartphone-centric technologies, as evidenced by an average estimation error of 0.47 metres in our trials. The second is the achievement of higher usability (including deployability and portability of the system) and reliability at low costs in comparison with other work.

Notwithstanding its limitations on resources to conduct a randomised controlled telemonitoring trial, this study does develop a simulation environment, as our second research objective, in order for
us to produce robust evidence for cost-effectiveness of a telemonitoring system to explore technology choices in different settings prior to moving to full-scale trials. Work was done to develop simulation models for constructing our specific telehealthcare scenarios. This enabled the assessment of supportive technologies and services delivered at scale through simulation trials. The main contributions of this work are listed as follows:

- **Simulation of patients’ activities of daily living and telehealthcare interventions** (chapter 6): The first contribution is the development of the novel simulation models to emulate a large number of patients and their activities of daily living, as well as a number of different stakeholders and their interactions with patients through the proposed telemonitoring interventions/system. The second is the modelling of specific telehealthcare scenarios to enable us to predict patient behaviours towards the use of healthcare services, such as GP and A&E, after a fall or vital sign abnormalities with and without telemonitoring interventions through simulations, and to formulate evidence for cost-effectiveness of our telemonitoring interventions.

- **Simulation of (mobile) cloud computing for supporting large-scale real-time telehealthcare application** (chapter 7): The first contribution includes both the improvements we made on a popular cloud simulation toolkit and the modelling work on the representations of cloud-based web application and database servers, as well as their workloads, based on empirical benchmarking experiments. These can serve as a good foundation for other researchers to better construct their cloud simulations. The second is the novel simulation work to integrate a large amount of web session workloads from a telemonitoring domain with dynamic provisions of cloud resources. The third is the proposed FLUCAS algorithm, which has demonstrated its novelty to enhance cloud-based system performance and scalability and reduce costs in our specific healthcare scenarios.

- **A comparative cost-effectiveness analysis approach using simulations** (section 3.3): The first contribution is the methodology for using simulations to enable ourselves to make a case about the cost-effectiveness of the telemonitoring solution prior to moving to full scale trials on a more scientific basis. The second is the systematic approach we proposed to integrate all the work together to help justify the great potential of achieving cost-effective home telehealthcare.

Although exploratory, this study not only offers some insight into the potential of smartphone-centric technologies in support of a cost-effective design of real-time home healthcare telemonitoring, but also provides evidence for cost-effectiveness of telemonitoring. Given this, it can be concluded that we have achieved our research objectives.
8.2 Future Work

As noted previously in other chapters, there were a number of limitations in this research. To bring our proposed telemonitoring solutions into daily routine use, more work needs to be done in the future to further enhance overall reliability, usability and robustness. Several identified areas of suggested future work are given in the following.

- There is a strong need to integrate standardised (e.g. IEEE 11073 compliant), clinically certified vital sign sensors, including but not limited to less-intrusive, low-cost body wearable sensors (though highly depending on their availability) into our monitoring system to ensure that diagnoses and treatment decisions are made based on accurate and trustable physiological measurements. This can greatly enhance not only system interoperability, but also extensibility.

- For fall detection, more tests and evaluations using a wider range of subjects with both simulated and real falls are needed so that sensitivity and specificity of the proposed algorithm can be further enhanced.

- Two directions are recommended to improve our indoor location tracking solution: The first is to further adapt our current design to a home-like environment by using ZigBee sensors or more BLE sensors together with the deployment of FSR sensors to ensure best estimates of patients’ real-time movement. The second is the enhancement of our step detection algorithm by performing further tests and evaluations using a wider range of subjects to ensure its generalisability, and by introducing a mechanism to automatically record sensor locations to facilitate system deployment.

- Regarding our modelling work for the simulations of healthcare scenarios, a number of parameters need to be optimised through interdisciplinary cooperation and surveys on relevant published data. Meanwhile, more work on modelling patients’ ADL and estimating the probability of cost-effectiveness, as well as sensitivity analysis, is suggested to better assess the uncertainty. Meanwhile, there is a need to address how telehealthcare can be seamlessly integrated with patient care pathways based on a case-specific business model to improve the design, implementation and evaluation of the provisioning and costs of telehealthcare services.

The unavailability of published data and lack of access to data sets already collected by other studies on the clinical and health outcomes have imposed undesired limitations on the scope and depth of this research, particularly with regard to the modelling of healthcare simulation and implementation of comparative effectiveness analysis. In light of this experience, we recommend that policy and regulation on the access and publication of clinical data aim to facilitate access by scientific research, whilst maintain proper protection of patient privacy.
For the modelling and simulations of (mobile) cloud computing, further research to explore and incorporate network dynamics, disk I/O and database performance, and database scalability into the design is needed so as to improve the quality of simulation results. Moreover, more tests using different service offerings, enhanced security mechanisms and multiregional deployment on a real cloud platform is needed to strengthen the modelling work and enhance robustness and security of our proposed system.

Last but not least, interdisciplinary cooperation and surveys on relevant published data, concerning especially health effects of telemonitoring and urgent care for post-falls and vital sign abnormalities on patients’ quality and quantity of life, are needed. These will enable us to undertake comparisons of costs and functionality with usual care and existing randomised controlled trials to produce robust evidence for cost-effectiveness of telemonitoring solutions.
9. Bibliography


“S. 2170: To amend title 38, United States Code, to improve the ability of health care professionals to treat veterans through the use of telemedicine, and for other purposes”, the US Congress website, [online] available: https://www.congress.gov/114/bills/s2170/BILLS-114s2170is.pdf [accessed: 25 Dec. 2015]


de_pr, [accessed: 4 Jan. 2013]


209


patients age 50 years or older”, Journal of Medical Internet Research, 2014 Apr 9;16(4):e104. doi: 10.2196/jmir.2968.


