Toward efficient protocol design through protocol profiling and performance assessment: using formal verification in a different context

Stylianos Georgoulas*,† and Klaus Moessner

Centre for Communication Systems Research (CCSR), Faculty of Engineering and Physical Sciences, University of Surrey, Guildford, Surrey, GU2 7XH, UK

SUMMARY

The most common use of formal verification methods so far has been in identifying whether livelock and/or deadlock situations can occur during protocol execution, process, or system operation. In this work, we aim to show that an additional equally important and useful application of formal verification methods can be in protocol design in terms of performance-related metrics. This can be achieved by using the methods in a rather different context compared with their traditional use, that is, not only as model checking tools to assess the correctness of a protocol in terms of lack of livelock and deadlock situations but rather as tools capable of building profiles of protocol operations, assessing their performance, and identifying operational patterns and possible bottleneck operations. This process can provide protocol designers with an insight about the protocols' behavior and guide them toward further optimizations. It can also assist network operators and service providers to assess the protocols' relative performance and select the most suitable protocol for specific deployment scenarios. We illustrate these principles by showing how formal verification tools can be applied in this protocol profiling and performance assessment context using some existing protocol implementations in mobile and wireless environments as case studies.

KEY WORDS: formal verification; probabilistic model checking; protocol profiling; energy efficiency

1. INTRODUCTION

Formal verification methods, and the corresponding front-end tools, provide a systematic way to assess the correctness of protocols, processes, and systems. Their main difference compared with simulation methods and tools is that instead of only examining a limited area of the operational space of the system under consideration, they can be used to examine the whole state space of possible operations and conditions under which the system may operate. This means that all possible combinations of inputs and actions can be taken into account, and therefore, all possible outputs can be derived and evaluated.

One could regard the outcome of formal verification methods as the outcome of an infinitely large number of simulation runs. This means that contrary to simulations, formal verification methods are capable of capturing conditions and operations that may otherwise remain unnoticed, even after a very large number of simulation runs. Traditionally, correctness means guaranteeing two properties: *liveness*, that is, some desired properties will be satisfied eventually, and *safety*, that is, some undesired properties will never occur [1]. Two notable counterexamples of such properties are the

existence of deadlock and livelock states. A deadlock state is a state where a deterministic loop occurs, which does not allow the system to leave that state; in other words, the system stalls. A livelock state is a state that allows the system to exit from; however, all possible exit actions will eventually lead the system back to this very same state; in other words, the system will not 'progress' any further.

Numerous examples of applying formal verification tools in that context can be found in the literature (e.g., [2] and [3]). Works such as [4] and [5] suggest that formal verification methods can be used not only to assess the correctness—as defined above—of protocols, processes, and systems but to additionally derive performance-related bounds, such as time to converge or time to reach a desired state. Contrary to simulations, which can derive performance bounds with a limited degree of confidence, by using formal verification tools for performance evaluation, it is possible to derive in many cases the absolute worst, best, and average performance bounds.

In this work, which builds on our previous work in [6] and [7], we argue that formal verification methods can be an even more powerful tool. That is, they can be used not only to derive worst, best, and average case bounds on performance but, additionally, they can be used to build profiles of protocols in terms of operations. This way, patterns in the behavior of protocols can be derived, the contribution or impairment incurred by each operation in terms of the defined metrics can be assessed, and possible bottleneck operations can be identified. Therefore, one can understand and reason about the behavior of a protocol and deduce with a high degree of confidence the absolute and relative—against other protocols—performance for specific deployment scenarios. Figure 1 illustrates at a high level the main concepts of applying formal verification in this context.

Based on such use of these tools, protocol designers can fine tune and optimize protocols so that these protocols operate in a manner that is not only deadlock and livelock free, but also optimized toward the metrics of interest. In addition, when many options exist regarding the selection of a protocol for a particular network and service scenario, the tools can assist network operators and service providers in selecting the most suitable one based on the considered metrics. The aim ofthis work is

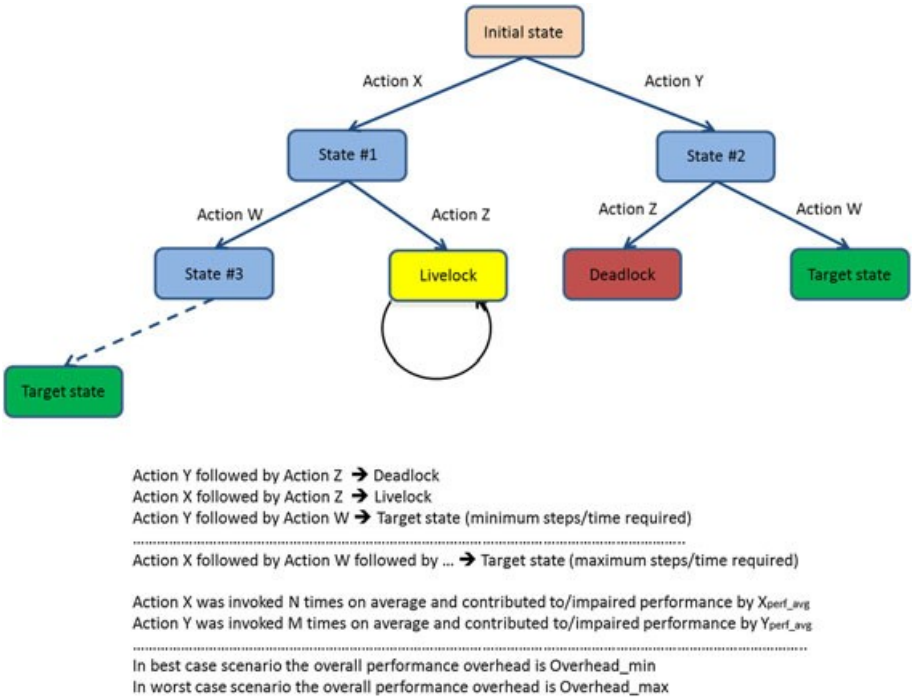


Figure 1. Formal verification in the context of protocol profiling and performance assessment.

to showcase the capabilities of formal verification tools in this protocol profiling and performance assessment context by considering some existing protocol implementations in wireless and mobile environments as case studies, covering protocol mechanisms operating at layers ranging from the physical layer to the application layer.

The rest of this paper is organized as follows. In Section 2, we briefly describe the underlying theory and model checking concepts of formal verification as well as the formal verification tool PRISM [8] used in this work. In Section 3, we briefly describe the considered metric of interest, and in Section 4, we describe the protocols we considered in our case studies and how the formal verification tool can be used for profiling and assessing the performance against this specific metric. In Section 5, we present our findings with respect to the performance and the behavior of the protocols, and we discuss some issues that are raised when using formal verification tools in this context. Finally, in Section 6, we present our conclusions and some directions for future work.

2. MODEL CHECKING CONCEPTS AND TOOLS

Probabilistic model checking is suitable for modeling and analyzing systems (the term system can refer to protocols as well as to processes) that exhibit probabilistic behavior [9]. It involves the construction of a probabilistic model describing the system to be analyzed, typically in the form of a state-transition system where states of this model represent the ways in which the system can evolve, associated with likelihood probabilities for their occurrence. In probabilistic model checking, four types of probabilistic models are commonly used, depending on the characteristics of the system to be modeled and analyzed; these are discrete-time Markov chains (DTMCs), Markov decision processes (MDPs), continuous-time Markov chains (CTMCs), and continuous-time Markov decision processes (CTMDPs) [10, 11]. According to the probabilistic model deployed, appropriate temporal logics to reason about the validity of properties—as expressed through formulas during system evolution—are used.

In DTMCs, all transitions can take place in discrete (time) steps, and the associated probabilities describe the likelihood of moving from that given state to any other possible state in the subsequent step. Because the behavior of a DTMC is fully probabilistic, the likelihood of a particular event occurring can be quantified over all the possible system evolutions (the term path is commonly used to refer to distinct system evolutions in time).

Markov decision processes extend DTMCs to model non-deterministic behavior, that is, behavior where the transition probabilities cannot be clearly defined, for example, probabilities for transitions triggered by external factors at random instances or incurred because of poor/unknown behavior specification, being therefore difficult to model using a unique probability distribution. To overcome this, in MDPs, we associated each state with a set of probability distributions, and the transition between states occurs in two steps [5]: first, there is a non-deterministic choice between available distributions in the current state, and then, the next state is selected at random according to the chosen distribution. Contrary to DTMCs, when MDPs are used, one can reason about the minimum and maximum (*absolute* and *expected/weighted*) likelihood of an event occurring over all the resolutions of non-determinism but not for the 'exact' average probability over all the possible paths.

In CTMCs, time is modeled not in discrete steps, but rather in a continuous manner. Therefore, CTMCs offer a much 'denser' notion of time compared with DTMCs and MDPs. In CTMCs, transitions are associated with rates rather than probabilities [5]. These rates represent parameters of negative exponential distributions and give the delay until the transition is enabled. CTMDPs, which constitute an area of active research interest themselves at the moment, extend CTMCs to take into account non-deterministic behavior, the way MDPs extend DTMCs for the same reason.

PRISM, which is being developed and maintained by the University of Oxford, provides direct support for DTMCs, MDPs, and CTMCs at the moment, and the PRISM property specification language incorporates several well-known temporal logics, such as linear time logic [10], probabilistic computation tree logic [12], which itself is an extension to computation tree logic [10], and continuous stochastic logic [13] for CTMCs.

Additionally, PRISM incorporates a discrete-event simulation engine allowing for deriving approximate solutions by evaluating over a finite number of paths. PRISM has been used for the analysis of probabilistic communication protocols and distributed algorithms and applications. More details can be found at the PRISM website where the relevant use cases are listed together with the corresponding code. We also used the code from the PRISM website as the base code for the protocols considered in our case studies for our work; we extended it though appropriately to enable the calculation of properties and allow for performance assessment and profiling against our metric of interest.

3. METRIC AND RELATION TO PROTOCOLS

The metric we considered in our work as metric of interest is energy consumption, which can be directly linked to energy efficiency. Energy efficiency as a research area has been gaining significant momentum during that past few years because of the savings it can provide to many technology areas and industries, as well as for environmental reasons.

In the context of telecommunications, entire projects are dedicated to energy efficiency related research (e.g., the MVCE Core 5 Green Radio Research Programme [14]), and network design and creation technologies have started being used in that context, even though they were not originally conceived to be used for that purpose. For example, Wang *et al.* [15] proposed to use network virtualization to save energy by 'migrating' on-demand underutilized virtual routers so that the physical equipment they reside on can be completely switched off (as shown in [16], routers in 'idle' state consume as much as 90% of the energy they consume when under full load).

Communication protocols could also be an area of great interest with respect to energy efficiency because all communication devices run protocols on top of them. Therefore, whereas the individual energy savings per device that could be contributed through protocol optimizations may look rather small compared with other possible savings (e.g., the energy savings by shutting down a router), the large number of devices running such protocols may make such small savings per device worthwhile. Additionally, contrary to energy savings through hardware optimizations (e.g., CPU and router design) that would require replacement of existing hardware, changes in protocols toward higher energy efficiency would only require software updates on the devices that run these protocols, without need for changes to the existing hardware.

Energy efficiency is even more important in mobile and wireless environments compared with fixed networks because of the limited battery life of the devices in such environments. In the following section, we briefly describe the six protocols we considered in this work, and we show how the latest version of PRISM, with its ability to associate costs ('negative' rewards) to states and transitions and query about these costs along all the possible evolutions of the protocols, can be used to build profiles of the protocols' operations with respect to energy consumption and assess their absolute and relative performance.

The process of associating costs in PRISM is relatively straightforward; this is carried out by associating labels to states and transitions that relate to changes in the metrics of interest. For example, a label 'time' can be associated with every transition that corresponds to time passing, and then a 'time' reward can be assigned to these transitions. In a similar manner, a 'send' label can be associated with every transition that corresponds to a packet being sent, and a 'send' reward can be associated with these transitions. Querying along all the possible paths to find, for example, the minimum time spent until an 'end' condition is met—that is, until a certain state is reached—would involve invoking the PRISM property specification language ***R*** operator as follows: ***R***{"time"}*min* =? [***F*** "end" {"init"}]. This expression asks the PRISM engine to return the minimum accumulated reward (time in this case) until (this is expressed through the ***F*** operator) the condition 'end' is met, considering all the possible initial states from where the protocol can start.

4. PROTOCOLS AND OPERATIONS

The protocol case studies we considered in this work were the following: (i) the Bluetooth device discovery protocol [17]; (ii) the IEEE 802.11 wireless local area network protocol [18] and the

IEEE 802.3 Ethernet protocol [19] (although IEEE 802.3 is designed for wired environments, we included it in order to show how PRISM can be used to assess the relative performance of IEEE 802.11 against IEEE 802.3, as this is induced by the different collision handling mechanism at the MAC layer); (iii) the Gossip protocol with peer sampling [20]; (iv) the ZeroConf dynamic configuration protocol for IPv4 link-local addresses [21]; and (v) a simple peer-to-peer protocol based on the BitTorrent protocol [22].

These specific protocol case studies were selected for two reasons. First, they allowed us to show the applicability of formal verification to protocol mechanisms operating at various layers, from the physical layer to the application layer of the Open Systems Interconnection (OSI) model. The layers of the OSI model where the considered protocol mechanisms belong to are illustrated in Figure 2.

Second, as it will be shown in the following sections, with the specific selection of protocol case studies, we were able to include all three probabilistic models supported by PRISM in our analysis in order to show the full extent and capabilities of this formal verification tool.

4.1. Bluetooth device discovery protocol

The first protocol case study we considered was the Bluetooth protocol and, in particular, its device discovery process, which was originally implemented in PRISM in [4], and available base code exists at the PRISM website [8]. As shown in [23] and [24], the operations during the device discovery have higher energy consumption than the operations during normal Bluetooth communications, making the device discovery process the most interesting from an energy efficiency point of view.

In order to communicate, Bluetooth devices form small networks called piconets, comprising one master and up to seven slave devices. The master device in a piconet alternates between two states; inquiry mode and inquiry-scan mode [23]. At inquiry mode, which has an average power consumption of 200 mW, the master device sends out inquiry messages at different frequencies to probe for slave devices that want to be discovered. At inquiry-scan mode, which has an average power consumption of 100 mW, the master device listens to replies from the slave devices.

During device discovery, the slave devices also operate in these two modes with the same power requirements, and additionally, they operate into a standby mode with an average power consumption of 50 mW [23, 24]. Contrary to the inquiry and inquiry-scan modes, which have fixed time durations, the standby mode can have varying duration, which leads to a probabilistic behavior of the overall model. Because the durations of all modes for all devices can be described as a multiple of a basic time unit (312.5 j-ts), the clocks of the master and slave devices can be assumed synchronized, and there is no non-determinism in the behavior of all involved devices [4], the whole discovery process can be modeled as a DTMC in PRISM.

By associating costs to the states and transitions of the model in PRISM, taking into account the duration of each mode (inquiry, inquiry scan, and standby) with respect to the basic time unit, and querying about the cost along all possible execution paths, we can reason about the average and worst case energy consumption of each mode as well as for the average and worst case energy consumption of the whole discovery process. Contrary to simulations, these values depict the ‘absolute’ average and worst case values, allowing us to deduce whether a possible evolution of the Bluetooth discovery process can lead to excessive energy consumption.

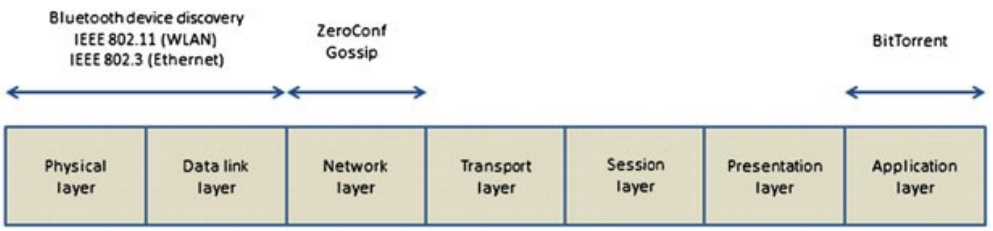


Figure 2. Considered protocols and relationship with the layers of the Open Systems Interconnection model.

4.2. IEEE 802.11 and IEEE 802.3

As a second protocol case study, we considered a comparison between the IEEE 802.11 and IEEE 802.3 protocols, originally implemented in PRISM in [25] and [26].

The main difference between these two protocols is the mechanism at the MAC layer, used to handle collisions because of simultaneous transmissions. In IEEE 802.3, the stations can listen to their own transmissions; therefore, carrier sense multiple access with collision detection (CSMA/CD) is used. The basic structure of the protocol is as follows: when a station has data to send, it listens to the medium, and if the medium is free, the station starts sending its data. On the other hand, if the medium is sensed as busy, the station waits a random amount of time and repeats this process. If, however, there is a collision while a station transmits, the station enters an exponential backoff process before re-attempting to transmit, where the duration of it (the exponent) is determined by the number of detected collisions, bounded by an upper limit.

In contrast to wired devices, stations of a wireless network cannot listen to their own transmission; therefore, they cannot employ medium access control schemes such as CSMA/CD to prevent simultaneous transmissions. To overcome this, IEEE 802.11 employs a collision avoidance scheme (CSMA/CA), which also uses an exponential backoff process as a function of the number of detected collisions. In CSMA/CA, when a station finishes its transmission, it immediately listens to the channel to detect whether another station is transmitting, and if so, it decides that a collision has occurred. Because of the parameters involved in the standards, the models for both protocols (2 Mbps IEEE 802.11 and 10 Mbps IEEE 802.3) are MDPs. However, by using the approach of Kwiatkowska *et al.* [5], that is, replacing non-determinism with uniform probabilistic choices and converting the model to a DTMC, we can deduce average values as well. This is not an accurate representation of these protocols; it can be seen though as the outcome of averaging over a very large number of simulation runs [5].

With respect to energy efficiency, similar to [27], we considered the number of sent packets, expressed in this case as the number of collisions because of retransmissions, as the main indicator of energy consumption; we also considered the time needed for the protocol to complete its operations (i.e., send an amount of data) as a secondary indicator.

By using PRISM and assigning costs to states and transitions corresponding to collisions and time passing, we are able to profile these two protocols' performance as a function of the exponential backoff and also assess their relative performance in terms of number of collisions and time needed to send the same amount of data.

4.3. Gossip protocol with peer sampling

The third protocol case study we considered was the Gossip protocol with peer sampling [20], originally implemented in PRISM in [5] with available base code at the PRISM website [8].

Gossip protocols are a class of communication protocols, which, inspired by the way that gossiping propagates messages in social networks, disseminate content through a network based on periodic exchanges of data with random members of the network. These techniques are designed to function robustly and efficiently in networks that are large, heterogeneous, and dynamic in nature. They are becoming, therefore, important because of the increasing use of mobile *ad hoc* networks, wireless sensor networks, and peer-to-peer technologies [5].

In the peer sampling version of the Gossip protocol, each node in a network maintains a small local membership table providing a partial view of the network, which is periodically updated using the gossiping procedure. At each round of periodic execution of the Gossip protocol, each node sends its data exactly once. These data are the view that a node has of the network, and the receiving node uses this information to update its own view and then further propagate it. Because of its distributed nature, the order in which the nodes participate in each round is unknown and can vary from round to round. Because of this non-deterministic behavior, the Gossip protocol is modeled as an MDP in PRISM.

With respect to energy efficiency, similar to [27], we make the assumption that the biggest amount of energy consumption occurs when a node sends a message to another node. By associating a cost

equal to one to each state of the model where one node enters a 'sending' mode, we can reason about the number of messages sent by each node as well for the total number of messages sent.

Because the model is an MDP, we can deduce the maximum and minimum number of messages sent over all resolutions of non-determinism but not the average number of messages. However, as in the previous case, by converting the model to a DTMC, we can deduce average values as well.

4.4. ZeroConf protocol

The fourth protocol case study we considered was the ZeroConf protocol, originally implemented in PRISM in [28]. ZeroConf offers a distributed 'plug-and-play' solution, in which IP address configuration is managed by individual devices, therefore making it highly applicable in case of mobile *ad hoc* networks.

In brief, the sequence of operations in the ZeroConf protocol is as follows. When a new device is connected to a network, it randomly selects an IP address from a pool of 65,024 addresses. The new device waits for a random time between 0 and 2 s and then starts transmitting four Address Resolution Protocol (ARP) packets (probes) to all other already connected devices, with a 2-s interval between them. These packets contain the IP address the new device has chosen. A device that is already using the IP address selected by the new device will respond with an ARP reply packet, and the new device has to restart the probing process selecting a new address. Each time the new device receives an ARP reply packet, a counter is incremented, and when this counter reaches 10, the new device has to backoff and remain idle for 1 min.

If the new device sends four ARP probe packets without receiving an ARP reply packet, it starts using the chosen IP address and sends two gratuitous ARP packets at 2-s intervals as confirmation of this. Because there may be the case that two devices can be using the same address because of packet loss (e.g., when ARP probe messages are lost and, therefore, no ARP replies are generated even if there is an IP address conflict, or when ARP replies are generated but are lost) as a last resort—but also prone to packet loss—the protocol instructs the following: if for some reason a device receives any ARP packet with a conflicting IP address, other than an ARP probe, it can either defend its IP address or defer. It can defend it if it has not received a previous conflicting ARP packet within the last 10 s; otherwise, it is forced to defer.

The standard assumes that it takes between 0 and 1 s to send a packet between devices; because the exact time delay is non-deterministic, this leads to an MDP model for which we can reason about minimum and maximum values for properties. However, by using the same approach as before, that is, replacing non-determinism with uniform probabilistic choices and converting the model to a DTMC, we can deduce average values as well.

With respect to energy efficiency, similar to [27], we considered the number of sent packets as the main indicator of energy consumption; we also considered the time needed for the protocol to complete its operations (i.e., assign an IP address to the new device) as a secondary indicator. By associating costs to the states and transitions that correspond to packets being sent and time passing and querying about the cost along all possible execution paths, we can reason about the minimum expected, maximum expected, and average number of packets of all types (probes, replies, and gratuitous) sent and received by the devices and about the time needed for protocol termination. By using as variables the packet loss rate of the medium and the number of existing devices, we can profile the protocol's operations against these two 'environmental' variables and assess its overall performance.

4.5. Peer-to-peer BitTorrent-like protocol

As the last protocol case study, we considered a simple peer-to-peer protocol based on the increasingly popular BitTorrent protocol with available base code at the PRISM website [8] as a CTMC model.

In this simple peer-to-peer protocol, as in [8], we assume that some new clients connect to an existing client in order to download a file split into a number of blocks. These clients can also download pieces of blocks from other clients as soon as these blocks become available and can also upload pieces of blocks to other clients as soon as they themselves have completed downloading

these blocks. There can be a limit to the download speed per block from each connected client as well as a limit to the maximum number of concurrent downloads for each block and to the file size (expressed in number of blocks).

Even though this is a rather oversimplified model of a peer-to-peer protocol, lacking the level of details in other works evaluating through simulation the behavior of BitTorrent [29] and BitTorrent-like [30] protocols in terms, for example, of bandwidth utilization, it still incorporates the main features and concepts of a peer-to-peer protocol. It is also worth mentioning that the main purpose of this work is not to derive detailed models for the examined protocols or question their validity and performance but rather to showcase how formal verification tools can be used as complementary tools to simulation tools for protocol profiling and performance assessment.

As shown in [31], for a single mobile device running BitTorrent, after the data connections are set up, the power consumption is rather stable, both when the device is connected to a 3G mobile network and also when it is connected to a wireless local area network. Therefore, we assumed that the time to finish the download is the main indicator for energy efficiency.

Using PRISM, we can reason about the average time needed for each client to complete the download as well as for the time needed collectively for all clients to complete their downloads. We can, therefore, reason about the fairness of the protocol with respect to energy consumption among the connected clients and also about the relationship between individual download times, total download time, and their associated energy values. Using the number of allowed concurrent downloads per block and the file size as variables, we can assess its individual and collective performance for a set of clients and examine how fine tuning this parameter can affect performance.

4.6. Summary of protocols

Table I summarizes what was presented previously regarding the protocol case studies with respect to the type of probabilistic model used, the energy efficiency indicators used, and the parameters used as variables, as well as the type of results that were derived and will be discussed in the following section.

5. FINDINGS AND DISCUSSION

In this section, we present our findings regarding the behavior of the considered protocols from an energy efficiency point of view.

5.1. Bluetooth device discovery protocol

For the evaluation of the Bluetooth device discovery protocol, we used the same scenario as in [4] with one master device and one slave device. We also assumed that initially, the master device is in inquiry mode sending inquiry messages and the slave device is in inquiry-scan mode listening for inquiry messages. The possible system evolutions were modeled until the slave device replies to the master device.

Tables II and III summarize the results regarding the energy consumption of the operations during the device discovery process. The numbers in the columns when multiplied by $10^{-3} \times 312.5$ j-ts give energy values in joules; we keep them though in their current format (integer numbers and without any units) for ease of presentation.

As one can see, the master device consumes more energy than the slave device (on average and in the worst case scenario) because it is not allowed to enter the standby mode. It also consumes much more energy on average in inquiry mode than in inquiry-scan mode, and this can be attributed not only to the fact that the inquiry mode has twice as much energy consumption than the inquiry-scan mode but also to the fact that the master device is assumed to always start in inquiry mode.

Regarding the slave device, we observe the opposite when it comes to maximum energy consumption; it consumes much more energy in inquiry-scan mode than in inquiry mode. The energy consumption in inquiry mode is fixed, because it remains always the same at all possible execution paths and is equal to the energy needed to send a reply message to the master device. The energy

Table I. Summary of considered protocol case studies.

Protocol	Model	Energy efficiency indicators	Variable parameters	Type of results
Bluetooth device discovery	DTMC	Energy consumption in watt x time in master/slave devices	None	Maximum and average energy consumption in total and per device mode
IEEE 802.11 and IEEE 802.3	MDP (natively)	Primary indicator: number of collisions because of retransmissions; secondary indicator: time needed to send an amount of data	Exponential backoff	Maximum/minimum/average values for the primary and secondary indicator for each protocol; relative performance of the two protocols
Gossip protocol	MDP (natively)	Number of messages sent in total and per individual node	None	Maximum and average number of messages sent in total and per individual node
ZeroConf protocol	MDP (natively)	Primary indicator: number of packets sent; secondary indicator: time needed for protocol termination	Packet loss rate of the medium and number of already existing devices	Maximum/minimum/average values for the primary and secondary indicator
BitTorrent-like protocol	CTMC	Time needed to complete a file download	Concurrent allowed downloads per block and file size	Average time needed for each client to complete a file download; average time needed for all clients to complete a file download

DTMC, discrete-time Markov chain; MDP, Markov decision process; CTMC, continuous-time Markov chains.

Table II. Energy consumption for the master device.

	Maximum	Average
Total	154	18
Inquiry mode	116	16
Inquiry-scan mode	58	2

Table III. Energy consumption for the slave device.

	Maximum	Average
Total	93	15
Inquiry mode	12	12
Inquiry-scan mode	74	2
Standby mode	9	1

consumption in inquiry-scan mode can deviate largely from its average value depending on how long it takes for the frequencies used by the master and slave device to coincide.

Based on the average values for the slave device, one could argue that the inquiry mode is a bottleneck operation in terms of energy efficiency. However, as already stated, this is not because this operation is repeated many times so it could be reduced by redesigning the protocol, but it is a 'fixed' operation with stable energy consumption per 'run'.

A general observation is that for both master and slave devices, the maximum (worst case) values of energy consumption deviate significantly from the average values. For the Bluetooth device discovery protocol, this should not raise major concerns because the energy consumption is rather small. However, for other more energy-demanding protocols, such deviations could mean that there may be execution paths that will put a considerable strain on the battery life of the devices.

Table IV summarizes the main findings for this protocol case study.

5.2. IEEE 802.11 and IEEE 802.3

For this case study, we considered the scenario where two stations are connected through IEEE 802.11 and IEEE 802.3, respectively. We calculated the number of collisions and time needed until both stations managed to successfully transmit a packet, as a function of the exponent *exp* used in the backoff function of both protocols.

Our results showed that for the IEEE 802.11 protocol, varying the exponent *exp* between 2 and 6 does not significantly affect the number of collisions or the time needed for both stations to successfully transmit a packet. As the exponent increases, there is a decrease in the number of collisions and an increase in the time needed; however, the variations are almost negligible, and for all values of *exp*, the average number of collisions is ""0.8, and the average time needed is ""2243 j-ts. For the IEEE 802.3 protocol, the same trends are observed as the exponent increases; however, in this case, there exists a noticeable variation, especially in the time needed. For *exp* D 2, the average

Table IV. Summary of results for the Bluetooth device discovery protocol case study.

Protocol	Energy efficiency indicators	Range of variable parameters	Observations
Bluetooth device discovery	Energy consumption in watt x time in master/slave device until synchronization frequencies between one master and one slave device	None	Master device consumes more energy than the slave device. In master device, more energy is consumed in inquiry of mode. In slave device, more energy is consumed in inquiry-scan mode. In both devices, maximum energy consumption deviates significantly from average energy consumption

number of collisions and time needed is ""1.67 and ""2065 j-ts, respectively, whereas for *exp* D 6 the corresponding values are ""1.64 and ""2608 j-ts.

The results also showed that whereas for the IEEE 802.3 protocol the minimum and maximum expected values for these two indicators are close to their average, for the IEEE 802.11 protocol they vary significantly (e.g., from ""1325 j-ts minimum expected time to ""52,682 j-ts maximum expected time). This means that the IEEE 802.3 protocol is more predictable than IEEE 802.11 in terms of energy efficiency. It is also worth noting that the IEEE 802.3 protocol, despite the higher number of collisions compared with IEEE 802.11, requires in some cases less time—on average—to send the same amount of data because of its highest transmission speed and the setting of its parameters

Table V summarizes the main findings for this protocol case study.

5.3. Gossip protocol with peer sampling

For the evaluation of the Gossip protocol with peer sampling, we used the same scenario as in [5] with a small network topology comprising of four nodes. As in [5], we considered the case that the size of view (that is, the number of other nodes each node keeps track of) is equal to 2, that all four nodes are initially in a 'non-sending' state, and that node 2 can be initially seen by the other three nodes but cannot see any of them. We modeled the possible system evolutions until the network of the four nodes becomes connected through the propagation of gossiping messages and the updating of the local views of the nodes, that is, until there are paths from every node to every other node (either direct or through other nodes).

Table VI summarizes the results regarding the maximum and average number of messages sent by each node and by the network as a whole, until the network becomes connected.

As one can observe that the maximum number of messages sent by all four nodes is the same; the average numbers also closely match. This means that the Gossip protocol, at least in terms of number of packets sent and—consequently—in terms of the corresponding energy consumption, is fair among the four nodes and distributes the energy overhead evenly.

Table V. Summary of results for the IEEE 802.11 and IEEE 802.3 protocols case study.

Protocol	Energy efficiency indicators	Range of variable parameters	Observations
IEEE 802.11 and IEEE 802.3	Primary indicator: number of collisions due to retransmissions until two stations successfully send one packet; secondary indicator: time needed until two stations successfully send one packet	Exponential backoff ranging from 2 to 6	For IEEE 802.11, the exponent does not significantly affect the number of collisions and the time needed. For IEEE 802.3, as the exponent increases, the number of collisions decreases, and the time needed increases. For IEEE 802.11, minimum and maximum values deviate significantly from average values. For IEEE 802.3, minimum and maximum values almost match the average values. IEEE 802.3 is more predictable than IEEE 802.11 with respect to energy efficiency

Table VI. Number of messages sent.

	Maximum	Average
Total	53	35
Node 1	14	9
Node 2	14	8
Node 3	14	9
Node 4	14	9

Table VII. Summary of results for the Gossip protocol case study.

Protocol	Energy efficiency indicators	Range of variable parameters	Observations
Gossip	Number of messages sent in total and per individual node until four nodes become connected	None	Maximum number of messages sent by each node is the same. Average number of messages sent by each node is almost the same. Gossip protocol is fair among the nodes with respect to energy related overhead

Compared with, for example, the Bluetooth device discovery protocol (for the specific configurations used for these two case studies), the deviation of the maximum values from the average values is smaller, meaning that the Gossip protocol is more predictable and 'smooth' in terms of energy consumption.

Table VII summarizes the main findings for this protocol case study.

5.4. ZeroConf protocol

For the evaluation of the ZeroConf protocol, we used a scenario where one new device tries to connect to a network of already connected devices and acquire a new unused IP address. As in [28], we considered as variables the number N of the already connected devices, to evaluate the performance of the protocol in very small to very big networks, and the packet loss rate *PLR* of the channel, to evaluate the performance in a variety of channel reliability conditions. For N, we considered the values 20, 1000 (as in [28]), and 10,000 (the last one is quite unlikely to occur in any practical scenario; we used it, however, to stress the protocol and observe its behavior in this extreme case). For *PLR*, we considered the values of 0.0, 0.01, and 0.1.

As also shown in [28], despite the inherent redundancy of the protocol in terms of messages sent before the new device acquires a new address, when the medium is lossy, there always exists the chance that eventually the new device will pick an already used IP address. However, this is a very remote event with an average probability to happen ranging from ""10^{-8} to ""10^{-4} depending on the number of existing devices and the packet the loss rate of the medium.

In our evaluation with respect to number of packets sent and received and time for the protocol to consider its operations completed, we observed that for each (N , *PLR*) pair, the minimum expected and maximum expected values derived from the MDP models do not differ significantly from each other and actually closely match the average values obtained when converting the model to a DTMC. This in principle means that the performance of the protocol is smooth along all possible resolutions of non-determinism, with no particular resolution of it (e.g., resulting from selecting a particular distribution for the packet transmission time) leading to some spike in the performance. As also the average values revealed, there does not exist any big deviation in the average performance for different (N , *PLR*) pairs; the average total number of packets sent during the whole process ranges from 6.00065 for .N , *PLR*/ D .20, 0.0/ to 6.4282 for .N , *PLR*/ D .10,000, 0.1/, and the average time for the protocol to complete its operations ranges from 13.0006 s for .N , *PLR*/ D .20, 0.0/ to 13.4478 s for .N , *PLR*/ D .10,000, 0.1/.

Table VIII summarizes the main findings for this protocol case study.

5.5. Peer-to-peer BitTorrent-like protocol

For the evaluation of the peer-to-peer protocol, we used a scenario where five new clients connect to a client that originally hosts a file split in N blocks, in order to download it. We set the download speed per block from each connected client to two blocks per time unit, and we experimented with the number K of allowed concurrent downloads per block and the file size. Table IX shows the average time (in time units) for the whole download process to be completed for a file of N D 4, 5, 6, 7, and 8 blocks as a function of K. That is, the average time it took within the same 'run' for all five

Table VIII. Summary of results for the ZeroConf protocol case study.

Protocol	Energy efficiency indicators	Range of variable parameters	Observations
ZeroConf	Primary indicator: number of packets sent until a new device acquires an IP address; secondary indicator: time needed for protocol termination	Packet loss rate (*PLR*) of the medium: 0.0, 0.01, and 0.1; number of already existing devices: (N) 20, 1000, and 10,000	Minimum and maximum values for all (N , *PLR*) pairs do not vary significantly from average values; no significant difference in values between individual (N , *PLR*) pairs; ZeroConf protocol is smooth and predictable in terms of energy efficiency

Table IX. Average time needed for the whole download process to be completed.

N nK	2	3	4
4	0.9626	0.7152	0.6132
5	1.0185	0.7548	0.6461
6	1.0642	0.7872	0.6728
7	1.1029	0.8144	0.6952
8	1.1364	0.8379	0.7164

Table X. Average time needed for the whole download process as a function of block size.

N nK	2	3	4
4 big blocks	1.9251	1.4304	1.2265
8 normal blocks	1.1364	0.8379	0.7164

clients to download the file, representing therefore the average of the slowest download times among the five clients within each 'run'.

By examining each row, one can deduce that as the number of allowed concurrent downloads per block increases, the average time needed for the whole process to be completed decreases. More interestingly, by examining the columns, one can observe that as the file size increases, the average time needed also increases, not, though, proportionally with the file size. For example, for N D 8 and K D 4, it takes 0.7164 time units on average for all clients to download the file, whereas for N D 4 (which corresponds to a file half the size of N D 8) and K D 4, it takes 0.6132 time units on average. These results suggest that this simple peer-to-peer protocol performs better for larger files.

In order to see how the protocol would perform when trying to download a file of a certain size when the block size is changed, we rebuilt the model setting N D 4 and the download speed per block from each connected client equal to one block per time unit. This would correspond to downloading a file with size equal to the size of N D 8 in our original scenario, but split into blocks twice the size. Table X shows the average time needed for download in this case.

These results suggest that splitting the same file in more blocks of smaller size improves the download performance. To examine how fair the protocol is with respect to energy efficiency among the participating clients, we also calculated for our original scenario the average download time per client this time. As the results showed, individual average download times per client very closely match one another, which means that the protocol is fair with respect to energy efficiency among the participating clients. However, they are rather different from the results of Table IX, as Table XI shows.

These results suggest that at each distinct 'run' of the scenario, individual clients do not finish simultaneously their downloads. Therefore, the average time needed for the whole download process to be completed is higher than individual average download times. This was also confirmed by having PRISM to calculate the average probability with which all five clients finish their individual

Table XI. Average time needed for each client to complete the file download.

N nK	2	3	4
4	0.5878	0.4675	0.4252
5	0.6390	0.5045	0.4568
6	0.6815	0.5349	0.4826
7	0.7178	0.5607	0.5043
8	0.7494	0.5905	0.5318

downloads in time less than their individual average (as in Table XI) within the same 'run'. This probability was less than 15% in all cases, meaning that indeed individual download times within the same 'run' can greatly vary among the participating clients.

Table XII summarizes the main findings for this protocol case study.

5.6. Discussion

The reader may have already noticed that so far, we have applied formal verification tools to rather small configurations. This is a consequence of the detailed nature of model checking that unavoidably leads to an issue known as state space explosion.

However, even though the state space explosion problem unavoidably limits the size of the systems that can be model checked, small systems can highlight interesting behaviors that may also occur in larger more realistic configurations [5] and any errors found in smaller systems will most likely persist in larger systems (because of the larger number of possible interactions at larger systems, these errors are likely to be more frequent and severe).

The scalability of this approach can be improved, in general, by omitting operations and features that are not of interest for the scenarios from the model and by 'abstracting' appropriately the operations that need to be included in the model. It is also worth noting that there exist some proposals in the literature on how to scale the applicability of model checking by breaking down large systems into smaller ones, model checking the smaller systems, and combining the results to derive the correctness (or faultiness) and properties of the larger system. This approach is known as compositional verification [32, 33]. Also, for larger systems, simulation runs—either using the simulation engine of PRISM or other stand-alone simulators, such as ns-2 [34] and the OPNET modeler [35]—can be used to verify and confirm whether the findings of model checking applied to smaller systems match the findings at larger, more realistic systems, with the degree of confidence that simulation tools allow for.

Table XII. Summary of results for the BitTorrent-like protocol case study.

Protocol	Energy efficiency indicators	Range of variable parameters	Observations
BitTorrent-like protocol	Time needed for five clients to complete a file download	Concurrent allowed downloads per block (K) 2, 3, and 4; file size in blocks (N) 4, 5, 6, 7, and 8	As file size increases, download time increases but not linearly. As block size increases, download performance decreases. Individual download times per client closely match. Total download time is considerably higher than individual download times; this means that within each 'run', individual download times can vary significantly

6. CONCLUSIONS AND FUTURE WORK

In this work, we demonstrated that formal verification methods and tools can be used not only to assess the correctness of protocols in terms of absence of deadlock and livelock situations but also to build profiles of protocols and assess their performance in terms of metrics of interest. Formal verification as a methodology is generic enough and is therefore able to model protocol mechanisms operating at all layers, from the physical layer to the application layer.

As we showed, through their use as 'exhaustive' simulation tools, they can allow for results to be derived with much higher degree of confidence than simulation tools and can also allow for identifying operations that, even though from a 'correctness' point of view are acceptable, may constitute bottlenecks in terms of the considered metrics.

The metric we considered was energy efficiency, which, especially in mobile and wireless environments, is of significant importance. It is worth noting though that a similar approach can be followed for other metrics and not only for protocols but for processes in general, using formal verification tools as performance evaluation tools, as long as protocols and processes can be described formally and as long as states and transitions can be associated with metric-related costs and rewards. Examples of such metrics can be packet loss, collisions/retransmissions, delay, and various overheads associated with protocols' and processes' operations.

In the future, we will attempt to fully exploit the capabilities of formal verification tools and derive distributions for energy consumption for each operation of the considered protocols as well as use conditional probabilities and counterexamples (also referred to as witnesses) in order to identify and reason about the sequence of operations and possible patterns that lead to certain behaviors. We will also consider compositional verification in order to improve the scalability of this approach and perform similar analysis to more protocols.

ACKNOWLEDGEMENTS

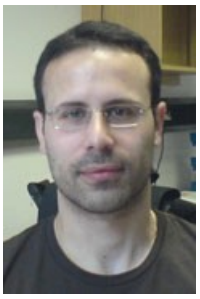
The research leading to these results has been performed within the Virtual Centre of Excellence in Mobile and Personal Communications (Mobile VCE) consortium (http://www.mobilevce.com/) as a part of Core 5 Programme (EP/G064091/1), which was partly funded by the Engineering and Physical Sciences Research Council (EPSRC) of the United Kingdom and the UniverSelf (http://www.univerself-project.eu/) project receiving funding from the European Community's Seventh Framework Programme (FP7/2007–2013) under grant agreement n° 257513.

REFERENCES

1. Lamport L. Proving the correctness of multiprocess programs. *IEEE Transactions on Software Engineering* 1977; **3**:125–143.
2. Islam S, Squalli M, Khan S. Modeling and formal verification of DHCP using SPIN. *International Journal of Computer Science and Applications* 2006; **3**:145–159.
3. Duflot M, Kwiatkowska M, Norman G, Parker D, Peyeronnet C, Picaronny C, Sproston J. Practical applications of probabilistic model checking to communication protocols. In *FMICS Handbook on Industrial Critical Systems*. IEEE Computer Society Press: Washington, D.C., 2010.
4. Duflot M, Kwiatkowska M, Norman G, Parker D. A formal analysis of Bluetooth device discovery. *International Journal on Software Tools for Technology Transfer* 2006; **8**:621–632.
5. Kwiatkowska M, Norman G, Parker D. Analysis of a gossip protocol in PRISM. *ACM SIGMETRICS Performance Evaluation Review* 2008; **36**:17–22.
6. Georgoulas S, Moessner K, Mcaleer B, Tafazolli R. Towards efficient protocol design through protocol profiling and verification of performance and operational metrics. In *Proceedings of the 6th International Wireless Communications and Mobile Computing Conference*, 2010; 848–852.
7. Georgoulas S, Moessner K, Mcaleer B, Tafazolli R. Using formal verification methods and tools for protocol profiling and performance assessment in mobile and wireless environments. In *Proceedings of the 21st International Symposium on Personal Indoor and Mobile Radio Communications*, 2010; 2471–2476.
8. PRISM website. Available from: http://www.prismmodelchecker.org/.
9. The PRISM manual. Available from: http://www.prismmodelchecker.org/.
10. Huth M, Ryan M. *Logic in Computer Science: Modeling and Reasoning about Systems*. Cambridge University Press: Cambridge, 2004.

11. Katoen J, Zapreev I, Hahn E, Hermanns H, Jansen D. The ins and outs of the probabilistic model checker MRMC. In *Proceedings of the 6th International Conference on the Quantitative Evaluation of Systems*, 2009; 167–176.
12. Hansson H, Jonsson B. A logic for reasoning about time and reliability. *Formal Aspects of Computing* 1994; **6**:512–535.
13. Baier C, Haverkort B, Hermanns H, Katoen J. Model checking algorithms for continuous-time Markov chains. *IEEE Transactions on Software Engineering* 2003; **29**:524–541.
14. MVCE Core 5 Green Radio Research Programme. Available from: http://www.mobilevce.com/frames.htm?core5research.htm.
15. Wang Y, Keller E, Biskeborn B, Merwe J, Rexford J. Virtual routers on the move: live router migration as a network-management primitive. In *Proceedings of ACM SIGCOMM*, 2008; 231–242.
16. Chabarek J, Sommers J, Barford P, Estan C, Tsiang D, Wright S. Power awareness in network design and routing. In *Proceedings of the 27th Conference on Computer Communications*, 2008; 457–465.
17. Bluetooth specification, version 1.2, Bluetooth SIG, 2003. Available from: http://www.bluetooth.com.
18. IEEE 802.11 Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Standard, 1997.
19. IEEE 802.3-2002. Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Standard, 2002.
20. Jelasity M, Voulgaris S, Guerraoui R, Kermarrec A, Steel M. Gossip-based peer sampling. *ACM Transactions on Computer Systems* 2007; **25**:8/1–8/36.
21. Cheshire S, Adoba B, Guttmann E. Dynamic configuration of IPv4 link-local addresses. Internet RFC 3927, 2005.
22. BitTorrent website. Available from: http://www.bittorrent.com.
23. Kasten O, Langheinrich M. First experiences with Bluetooth in the smart-its distributed sensor network. In *Proceedings of the International Conference on Parallel Architectures and Compilation Techniques*, 2001.
24. Ericsson Microelectronics. ROK 101 007 Bluetooth Module Datasheet Rev. PA5, April 2000.
25. Kwiatkowska M, Norman G, Sproston J. Probabilistic model checking of the IEEE 802.11 wireless local area network protocol. In *Proceedings of the 2nd Joint international Workshop on Process Algebra and Probabilistic Methods, Performance Modeling and Verification*, 2002; 169–187.
26. Kwiatkowska M, Norman G, Sproston J, Wang F. Symbolic model checking for probabilistic timed automata. *ACM Information and Computation Journal* 2004; **205**:1027–1077.
27. Cano J, Manzoni P. A performance comparison of energy consumption for mobile ad hoc network routing protocols. In *Proceedings of the 8th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, 2000; 57–64.
28. Kwiatkowska M, Norman G, Parker D, Sproston J. Performance analysis of probabilistic timed automata using digital clocks. *Formal Methods in System Design* 2006; **29**:33–78.
29. Bharambe A, Herley C. Analyzing and improving BitTorrent performance. *Microsoft Research Technical Report MSR-TR-2005-03*, February 2005.
30. Qiu D, Srikant R. Modeling and performance analysis of BitTorrent-like peer-to-peer networks. In *Proceedings of ACM SIGCOMM*, 2004; 367–378.
31. Nurminen J, Noyranen J. Energy consumption in mobile peer-to-peer—quantitative results from file sharing. In *Proceedings of the 5th IEEE Consumer Communications and Networking Conference*, 2008; 729–733.
32. Cobleigh J, Giannakopoulou D, Pasareanu C. Learning assumptions for compositional verification. In *Proceedings of the 9th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, 2003; 331–346.
33. Berezin S, Campos S, Clarke E. Compositional reasoning in model checking. In *Proceedings of the International Symposium on Compositionality*, 1997; 81–102.
34. ns-2 Simulator website. Available from: http://www.isi.edu/nsnam/ns/.
35. OPNET Modeler website. Available from: http://www.opnet.com/solutions/network_rd/modeler.html.

AUTHORS' BIOGRAPHIES



Stylianos Georgoulas is a Research Fellow in the Centre for Communication Systems Research at the University of Surrey, UK. He holds a Diploma in Electrical and Computer Engineering from University of Patras in Greece and a PhD in Electronic Engineering from University of Surrey. His research interest are in the area of formal verification, energy efficient protocol design, dynamic service management, and traffic engineering.

Prof. Klaus Moessner i.s a Professor for Cognitive Networks and Services, in the Centre for ConunWlication Systems Research at the University of Surrey, UK. Klaus earned his Dipl-Ing (FH) at the University of Applied Sciences in Offenburg, Germany, an MSc from Brunei University and PhD from the University of Surrey (UK). His research interests include dynamic spectrum allocation, cognitive networks, reconfiguration management, service platforms and adaptability ofmultimodal user interfaces.