In-network Caching of Internet-of-Things Data

Serdar Vural*, Pirabakaran Navaratnam*, Ning Wang*, Chonggang Wang†, Lijun Dong†, and Rahim Tafazolli*

*Centre for Communication Systems Research, University of Surrey, Guildford, Surrey, GU2 7XH, UK
e-mail: {s.vural, p.navaratnam, n.wang, r.tafazolli}@surrey.ac.uk
†InterDigital Communications, Inc., 781 Third Avenue, King of Prussia, PA, 19406, USA
e-mail: {Chonggang.Wang, Lijun.Dong}@InterDigital.com

***Abstract*—The recent forecast of billions of devices, all connected to the Internet and generating low-rate monitoring, measurement, or automation data that many end-users/applications frequently request, signifies the need for applying in-network caching techniques to Internet-of-Things (IoT) traffic. Although time delay is not critically important for small-sized IoT content, the expected total traffic load on the Internet from a large number of devices is significant. However, the main challenge as opposed to the typically cached content at content routers, e.g. multimedia files, is that IoT data are transient and therefore require different caching policies. This paper studies in-network caching of IoT data at content routers in the Internet. An IoT *data item* is uniquely defined not only by its time and location tags, but also a time-range value set by end-users/applications. We provide a model for the trade-off between multihop communication costs and the *freshness* of a transient data item. Results show that the model can successfully capture the effect of data transiency and can accurately represent the expected gains of a caching system: considerable savings in terms of reduction of network load, especially for highly requested data items.**

I. Introduction

Billions of Internet-of-Things (IoT) [1] devices connected to the Internet and generating low-rate traffic of measurement, monitoring, and automation data is now a significant challenge for network providers and the Internet as a whole. This is because, although each IoT traffic has a low rate, the aggregate load on core networks is expected to be large, which is likely to even disrupt regular data traffic. With a large number of end-users around the world running applications that request similar IoT data, such as weather statistics, monitoring results, etc, it is a necessity to reduce the redundancy caused by delivering similar content over the Internet.

One property of IoT data streams that can be exploited to remedy the IoT traffic load problem is that many IoT users/applications are mostly interested in a single value summarizing the measurement values taken over a relatively short period of time, such as an hour, 10 mins, 1 min, or several seconds, depending on data properties and application needs; some examples are the average temperature in location X in the last 10 minutes, or the maximum wind speed in location Y in the last hour. Depending on which time ranges are more popular among users/applications for specific IoT data collected at specific locations, content routers [2] in the Internet can cache these results [3] without relaying user requests all the way to data sources, i.e. data servers, or Machine-to-Machine (M2M) gateways that store collected information from a region of interest. The idea is similar to multimedia content caching [4] [5] at Internet content routers. However, unlike multimedia files, IoT data are transient and small in size. This means that IoT data items "expire" in contrast to large data files that have the same content forever. Hence, caching decisions must be performed based on dynamic variables, related with not only the user requests but also the IoT data items, such as data size, time-range, and *lifetime*.

As the first study on caching IoT data in Internet content routers, this paper provides a method to determine whether a given data item should be cached at network routers according to the item's lifetime, the rate and time range of incoming requests, and router hop distances to the data source and the end-users. In a distributed way, routers capture data popularity [6] via dynamically updating their caching probability values.

The analysis considers a key property of IoT data, which does not apply to traditionally cached multimedia data: *transiency*. Based on this, the paper addresses a trade-off: First, IoT data items become *less fresh* when the caching location is further away from the source location; i.e. retrieving data items from the cache of a router that is located closer to a requesting user rather than the data source provides items that have likely been generated at an earlier time in past. In short, the closer the caching location is to the source, the fresher the retrieved data item is. Secondly, retrieving data items from locations closer to the source rather than the requesting user incurs higher multihop data traffic load on the network. In order to capture this trade-off between multihop traffic load and data freshness, we derive a cost function based on the properties of the data item and request, as well as the most recent router variables. In an attempt to reduce their expected cost, routers dynamically modify their caching probabilities. Numerical results demonstrate that, as compared to the case in which the data item is not cached at routers but is always fetched from the source, considerable percent cost gains are obtained by in-network caching.

In the rest of the paper, first, the related work on in-network caching is mentioned briefly in Sec. II. Then in Sec. III, the concept of "freshness" of IoT data is explained, and then the theory of data item existence probability at content routers is presented. Sec. IV presents the analysis on the freshness vs. multihop retrieval tradeoff and derives a cost function. Analytical derivations on freshness are provided in Sec. V. Router actions are explained in Sec. VI, followed by numerical results in Sec. VII. Finally, Sec. VIII concludes the paper.

II. Related Work

The caching algorithms in previous studies are designed for intransient and often time-invariant multimedia files, large data files, and similar popular content, frequently requested by

a large number of users over the Internet. Caching IoT-related data has not been studied so far, except for [1] that discusses caching IoT variables in the MobilityFirst architecture.

Mostly coupled with the newly emerging content-based future Internet architectures [2] (as opposed to the existing address-based Internet architecture), caching algorithms are considered as a feature of content delivery and request forwarding systems. The common approach is to design content router functionalities to support incoming requests for different data files, dynamically cache those files, and efficiently manage router caching space through suitable cache replacement policies [5] [7]. For instance, the Breadcrumbs system in [8] presents a best-effort caching and query routing policy that uses the caching history of passing contents to modify the forwarding rates of request packets. The Cache-and-Forward (CNF) protocol architecture presented in [9] is based on content routers with sufficient storage spaces that can cache large data files. Based on this architecture, later studies propose different caching algorithms to be deployed in CNF routers. In [6], en-route autonomous caching with the idea of "content popularity" is introduced, where the least accessed content is removed first when the residual caching space is low. Other studies on CNF formulate optimization problems [10], which target at minimizing the total expected content delivery delay.

Analytical models have been proposed for in-network caching systems, which are designed for caching large and intransient data files. In [11], a model for general cache networks is provided, which solves a system of equations to find the incoming and outgoing request rates of all routers, using the global information of network topology, request and data traffic, and router variables. The caching probability is taken to be directly proportional to the rate of incoming requests, scaled by the sum of the rates of all existing data files in the network. Poisson streams of requests with exponential inter-arrival times are considered. Another central algorithm, the Traffic Engineering Collaborative Caching (TECC), proposed in [12], includes traffic engineering constraints, such as link cost and utilization, besides routers' limited caching spaces, in its general framework of cache coordination. The optimization target is to minimize the maximum link utilization in the network, considering limited caching spaces and popularity of different data pieces. The work in [3] provides hybrid cache management algorithms, in which distributed caching decisions are supported by a parent node that connects a cluster of caching nodes. Content placement among these nodes is modelled as a linear program towards minimizing a global cost function. Then, a set of local decisions are determined that nodes should make to achieve near optimal caching performance. A more distributed model is provided in [13], in which the probability to cache is considered over a path of routers towards the source, based on router caching spaces and the number of hops that packets traverse.

III. Preliminaries

In this section, the network model as well as two key concepts, namely "data item freshness" and "probability of data item existence", are introduced, which are used in Sec. IV to derive the cost function for caching a data item at routers.

A. Network model

This study is on the in-network caching of a particular IoT data item that is generated by a source node, and requested by a number of *requester* nodes. Each requester is located in a random location and assigned to a router that acts as its access point. Routers are connected to the source over multihop paths. $40\ Gbps$ Internet core links that are randomly formed between routers are considered.

B. Data item freshness

When caching IoT data items, it is essential to consider the *time-range* that defines a request, denoted by Δt. For instance, a request for the average temperature in London in the past 10 minutes has a value of $\Delta t = 10\ min$. For a data item to be "fresh" for its requester, the Δt time period preceding the item's arrival at the requester must have some overlap with the Δt time period preceding the item's generation at the source. This is illustrated in Fig. 1.

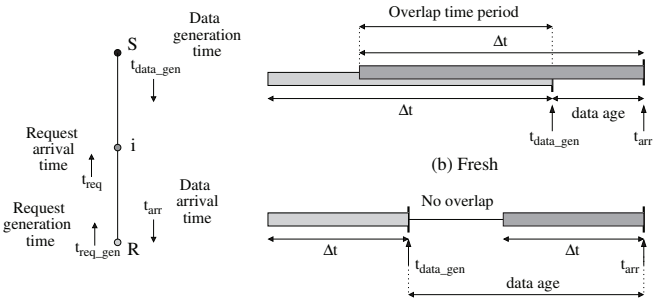


Fig. 1: Data item freshness.

In Fig. 1(a), the time instances of events for a request-data exchange between a source S and a requester R is depicted. Data freshness is evaluated at an intermediate router i. In the case shown in Fig. 1(b), a time overlap exists and the item is regarded as fresh, whereas in Fig. 1(c), there is no overlap between the two Δt periods, hence the data item is not fresh. Basically, the larger the overlap is, the fresher the item is. Accordingly, the *freshness* of a data item can be defined as:

$$Freshness = \frac{\Delta t - \text{data age}}{\Delta t}, \tag{1}$$

where "data age" denotes the age of the item, which is the time period between the arrival at the router and the generation at the source. Items with a negative freshness value are not cached at content routers.

1) Freshness loss: If the source S and the requester R were co-located such that there would be no delay in transferring the item, then the data age would be 0, in which case the freshness of the data item by the time of delivery to the requester would be 1. However, due to in-network delays, data items have finite non-zero data ages when received by requesters. Hence, freshness at the requester side is simply $\frac{\Delta t - d(S,R)}{\Delta t}$, where $d(S,R)$ is the time delay between S and R, which includes the propagation, queueing, and processing delays. When the data item is cached at an intermediate router i, there is also an additional *caching delay*, which is the time period that the item resides in the cache until being requested by a requester. The items retrieved from a router's cache are hence "less fresh", compared to the items retrieved from the source S, and have a freshness of $\frac{\Delta t - d(S,R) - Caching\ delay}{\Delta t}$. Hence, we define the

freshness loss of a data item as the reduction of its freshness caused by caching at routers, given by:

$$FL = \frac{Caching\ delay}{\Delta t}. \quad (2)$$

C. Cache-hit ratio: Probability of data item existence

Routers can respond a request for an IoT data item only if the item exists in their caches. Therefore, it is essential to first define the probability that a given data item exists in a router, called the *probability of existence*, P_e. This probability is equal to 1 at the source node and 0 at requesters, by definition.

Consider a router to perform in-network caching of IoT data, and consider a specific IoT data item with residual lifetime T_{res}. The incoming request rate for the data item is r_{in}, which is the total rate from all requesting neighbours of the router. Since routers simply forward request packets when data items do not exist in their caches, the outgoing rate of data requests for the same item at the router is $r_{out} = (1 - P_e) r_{in}$. Furthermore, data items are returned to only those routers that request them; i.e. the data arrival rate: $r_d \approx r_{out}$.

Let P_c denote the probability to cache the item when it arrives at the router. Hence, the rate of caching the item is $r_c = r_d P_c \approx (1 - P_e)\, r_{in} P_c$. Considering exponential inter-arrival times, the time period between successive caching events is approximately $\frac{1}{r_c}$. When $\frac{1}{r_c} > T_{res}$, [1] the fraction of time that the item exists in the router's cache is simply its probability of existence: $P_e = \frac{T_{res}}{1/r_c} = T_{res} r_c$. Therefore, we have $\frac{P_e}{T_{res}} = (1 - P_e)\, r_{in} P_c$, which gives:

$$P_e = \frac{r_{in} P_c}{\frac{1}{T_{res}} + r_{in} P_c}. \quad (3)$$

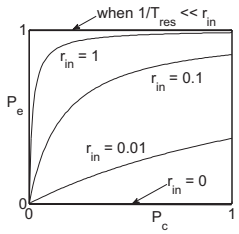


Fig. 2: Probability of existence P_e as a function of probability of caching P_c and incoming request rate r_{in}.

Fig. 2 illustrates how P_e changes with respect to P_c and r_{in}: (1) It is more likely to find the data item in the cache if the incoming request rate is higher, (2) If the router chooses to cache the items more often by picking a higher P_c, this increases the likelihood of finding the item in its cache.

IV. Cost Function

Getting a data item from the network involves a trade-off between two options: (1) fetching a newly generated "fresh" item from the source that is usually several hops away from the requester, and (2) fetching a not-so-fresh item from an intermediate router's cache but incurring less multihop retrieval cost. This is shown in Fig. 3, where R is the requester and S is the source. In this section, a cost function for retrieving a data item over the multihop path between a requester and the source node is derived by combining these two cost items, namely *freshness cost* and *multihop cost*.

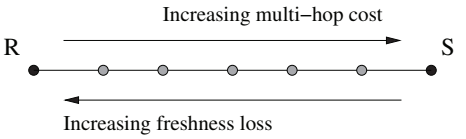


Fig. 3: Freshness loss vs multihop retrieval cost trade-off.

When a requester sends a request to the data source, the request travels hop by hop towards the source until it encounters a node where the requested item exists. This could be an intermediate router or the source node if none of the routers on the path have cached the item. Denoting the probability that the item exists in the router at the j^{th} hop from the requester towards the source as $P_e(j)$, the probability that the router at the i^{th} hop returns the item is:

$$Prob\{\text{Router at hop i replies}\} = P_e(i) \prod_{j=1}^{i-1} (1 - P_e(j)). \quad (4)$$

Based on Eqn. 4, we can represent the average freshness cost of the data item retrieved by the router at hop i as:

$$FC(i) = P_e(i) s_d \widetilde{FL}(i) + \sum_{j=i+1}^{N} \left(\prod_{k=i}^{j-1} (1 - Pe(k)) \right) P_e(j) s_d \widetilde{FL}(j), \quad (5)$$

where s_d is data item size, and $\widetilde{FL}(i)$ denotes the per-bit expected freshness loss of a returned item at a random time from the router that is i hops away from the requester on the path towards the source. The derivation of $\widetilde{FL}(i)$ is explained in detail in Sec. V. The second term in Eqn. 5 denotes the expected freshness loss when router i retrieves the item from its ancestors, i.e. hops $i+1, i+2, \ldots N-1$.

Similarly, the multihop cost can be derived using Eqn. 4. Modelling communication cost as the number of injected bits to a link[2], the cost of retrieving the data item over a single hop is $(s_d + s_r)$, where s_r is the request size. Then, the multihop retrieval cost is:

$$MC(i) = \sum_{j=i+1}^{N} \left(\prod_{k=i}^{j-1} (1 - Pe(k)) \right) P_e(j) \alpha (s_d + s_r)(j - i), \quad (6)$$

where α is called the *communication coefficient*, which is a scaling constant that is introduced to capture the relative importance that a user application gives to multihop retrieval cost, i.e. a high α means higher cost of retrieval and indicates that the application does not prefer frequent multihop retrieval. $\alpha = 1$ is chosen in numerical evaluations in Sec. VII.

[1] When $\frac{1}{r_c} < T_{res}$, the item in the cache never expires, as new caching events occur before item expiry; i.e. $P_e = 1$, by definition.

[2] Link cost can be generically defined as a function $f(s_d, s_r, \ldots)$

A. Total expected cost

Using Eqns. 5 and 6, the total cost of the data item at a router that is i hops away from the requester is:

$$\underset{0\le i\le N}{C(i)} = \frac{FC(i)}{T_{res}/\Delta t} + \frac{MC(i)}{i} = P_e(i)s_d\frac{\widetilde{FL}(i)}{T_{res}/\Delta t} + \sum_{j=i+1}^{N}\left(\prod_{k=i}^{j-1}(1-P_e(k))\right)P_e(j)\left[s_d\frac{\widetilde{FL}(j)}{T_{res}/\Delta t} + \alpha(s_d+s_r)\frac{(j-i)}{i}\right]. \quad (7)$$

where $\frac{T_{res}}{\Delta t}$ and i are normalization divisors applied to $FC(i)$ and $MC(i)$, respectively, when the two different costs are combined. In this equation, the first term is essentially the cost of an existing item for router i, whereas the second term (summation) is the expected cost of a retrieved item. Since the data item always exists at the source node, we have $P_e(N) = 1$, and since the requester always requires it, we have $P_e(0) = 0$. With $\widetilde{FL}(N) = 0$ at the source node, the cost at the last hop $N-1$ before the source becomes:

$$C(N-1) = P_e(N-1)s_d\frac{\widetilde{FL}(N-1)}{T_{res}/\Delta t} + (1-P_e(N-1))\alpha\frac{(s_d+s_r)}{(N-1)}. \quad (8)$$

B. Recursive cost terms

After some mathematical derivations, Eqns. 5, 6, and 7 can be rewritten in recursive form as follows:

$$\begin{aligned} FC(i) &= P_e(i)s_d\widetilde{FL}(i) + (1-P_e(i))FC(i+1), \\ MC(i) &= (1-P_e(i))\left\{MC(i+1) + \alpha(s_d+s_r)\right\}, \\ C(i) &= (1-P_e(i))\left\{C(i+1) + \alpha(s_d+s_r)/i\right\} \\ &\quad + P_e(i)s_d\frac{\widetilde{FL}(i)}{T_{res}/\Delta t}. \end{aligned} \quad (9)$$

V. Freshness Loss

In this section, the expected per-bit freshness loss $\widetilde{FL}$ used by the freshness cost function FC in Eqn. 5 is derived for a router located at hop i from a requester.

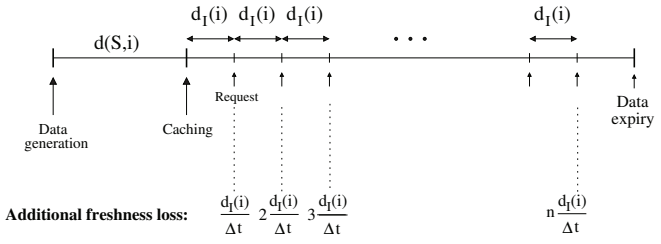


Fig. 4: Additional freshness loss.

A. Average additional freshness loss at a router

Caching delay at a router is related with the rate of incoming requests. Considering exponential inter-arrival times for requests [11], the waiting time until the first request arrives at the router at hop i at a random time instance is the reciprocal of the incoming request rate, i.e. $\frac{1}{r_{in}(i)}$. The average time period between consecutive request arrivals is therefore the inter-arrival time, $d_I = \frac{1}{r_{in}(i)}$. This is shown in Fig. 4. With this, the average number of times that the router's cache can respond incoming requests after the data item is cached is approximately $n \approx \frac{T_{res}-d(S,i)}{d_I} = (T_{res} - d(S,i)).r_{in}(i) \cong T_{res}.r_{in}(i)$, where $d(S,i)$ is the time delay of delivering a data item to the router from source S, which is negligible for small IoT data items over Internet core links of 40 Gbps bandwidth.

As shown in Fig. 4, the caching delay of a returned data item from a router's cache depends on when the request arrives at the router after the item is cached. In other words, an additional freshness loss is introduced to the data item for the time it spends in the cache, which is a multiple of $\frac{d_I}{\Delta t}$. Considering that a request arrives at a random time instance at a router, the average additional freshness loss that the data item suffers due to having been cached at the router is:

$$\overline{FL} \approx \frac{nd_I(i)}{2\Delta t} \approx \frac{T_{res}}{2\Delta t}. \quad (10)$$

B. Expected freshness loss

A data item that is cached at the i^{th} hop router from the requester might have been previously cached by other routers on its way from the source to the requester. In other words, the average freshness loss given by Eqn. 10 designates only the amount of freshness loss caused by caching at a single router, say the one at the i^{th} hop. When a request arrives at the router at a random time, the total expected freshness loss that the retrieved item has is denoted by $\widetilde{FL}(i) \ge \overline{FL}(i)$.

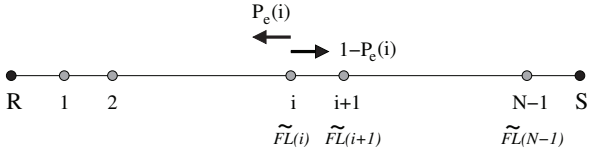


Fig. 5: Expected freshness loss.

When the item is not available at the router at hop i, the router forwards the request to its next hop router at hop $i+1$ towards the source, which happens with a probability $1-P_e(i)$. If the item exists in its next hop, then the additional freshness loss from its next hop is simply $(1-P_e(i))P_e(i+1)\widetilde{FL}(i+1)$. Considering all its ancestors $i+1, i+2, \ldots, N-1$ towards the source and a hop distance of N between the source and the requester as depicted in Fig. 5, the expected freshness loss at the router at hop i is then:

$$\begin{aligned} &\widetilde{FL}(i) = \overline{FL}(i) + (1-P_e(i)). \\ &\left\{\begin{array}{l} P_e(i+1)\widetilde{FL}(i+1) + \\ (1-P_e(i+1))P_e(i+2)\widetilde{FL}(i+2) + \\ (1-P_e(i+1))(1-P_e(i+2))P_e(i+3)\widetilde{FL}(i+3) + \\ \ldots + \left(\prod_{j=i+1}^{N-2}(1-P_e(j))\right)P_e(N-1)\widetilde{FL}(N-1) \end{array}\right\} \\ &\widetilde{FL}(i) = \overline{FL}(i) + (1-P_e(i))G(i+1), \end{aligned} \quad (11)$$

where $G(.)$ is a recursive function that represents the extra freshness loss coming from hop i's ancestors. Routers provide their computed $G(.)$ value to their neighbours as feedback. The $G(i)$ value that hop i provides to hop $i-1$ is:

$$G(i) = P_e(i)\overline{FL}(i) + (1-P_e(i)^2)G(i+1). \quad (12)$$

1) Source node and the last hop router: The average freshness loss at the source node as shown in Fig. 5 is by definition $\overline{FL}(N) = 0$. Furthermore, $\widetilde{FL}(N) = 0$ and $G(N) = 0$, as there are no previous hops from where any additional freshness loss can come. As a result, using Eqns. 11 and 12, we have $\widetilde{FL}(N-1) = \overline{FL}(N-1)$ and $G(N-1) = P_e(N-1)\overline{FL}(N-1)$ for the last hop router before the source node.

VI. Router Action towards Reducing the Expected Total Cost

Previous sections provide the core equations that represent the set of content routers between a requester and the source as a caching system. Eqn. 3 is a generic definition of the existence probability of a data item at a router as a function of its caching probability $P_c(i)$ and the incoming rate of requests $r_{in}(i)$ for the data item. Eqn. 9 computes the total expected cost of the data item for a particular $P_e(i)$ (which is determined by the incoming rate $r_{in}(i)$ and a chosen caching probability $P_c(i)$). The $G(i)$ and $C(i)$ values computed by Eqns. 9 and 12 are feedbacks to neighbour routers[3] so that next hop routers can compute their own costs.

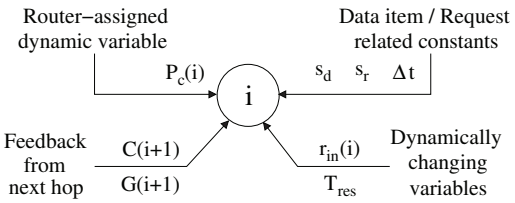


Fig. 6: Router variables and constants.

Router variables and constants are summarized in Fig. 6. Every router starts with $P_c = 0.5$ for each data item that it receives a request for. Then, routers dynamically update their cost functions for the data item and decide whether to increment or decrement P_c according to the instantaneous slope of the cost function, which is $\frac{A-B}{B}$ as shown in Eqn. 13. Note that this slope at hop i depends on the cost function feedback from hop $i+1$ as well as the expected freshness loss at hop i.

$$\begin{aligned} C(i) &= \underbrace{C(i+1) + \alpha(s_d + s_r)/i}_{B} + \\ &\quad P_e.[\underbrace{s_d \frac{\widetilde{FL}(i)}{T_{res}/\Delta t}}_{A} - \underbrace{(C(i+1) + \alpha(s_d + s_r)/i)}_{B}] \\ &= (A-B)P_e(i) + B. \end{aligned} \tag{13}$$

VII. Numerical Results

In this section, the numerical simulation results obtained with Matlab [14] are presented for networks with 400 content routers. All results are averages of 10 random networks. The source node is located at the centre of the topology. Shortest paths between the source and the requester are considered. Table I summarizes the simulation parameters.

[3]These two values are embedded in the forwarded data packets; hence there is no need for separate feedback packets.

TABLE I: Simulation parameters.

Parameter Name	Value
Data size s_d	512 Bytes
Request size s_r	60 Bytes
Average link bandwidth	40 Gbps
Communication coefficent α	1
Number of routers, N	400
Data item lifetime T_{res}	1.5 min
Increment/decrement of Pc	0.0001

In Fig. 7, histograms of the probability of existence P_e at routers are shown for base request rates of $1, 0.1, 0.01$ $requests/sec$ and different numbers of requesters, $40, 80, 160, 320$ out of 400 routers, i.e. requesters are attached to a sub-set of routers. The histograms are normalised, i.e. all bar values are divided by the total number of caching routers, so that the distribution of P_e in the network can be observed; and bars are shown for 10 ranges of P_e: $[0\ 0.1], [0.1\ 0.2], \ldots [0.9\ 1]$. The figure illustrates that when the base rate is 1, routers tend to cache the data item more deterministically, with an observable peak at range $[0.9\ 1]$, whereas for the low base rate setting, a wider distribution of P_e among all ranges is obtained. Furthermore, having a larger number of requesters shifts this distribution towards higher P_e ranges. Hence, in a network with a large number of requesters with a high request rate, a number of routers cache the data item almost deterministically while others choose not to. The caching routers are observed to be those where incoming request rates are higher: the routers where request flows merge.

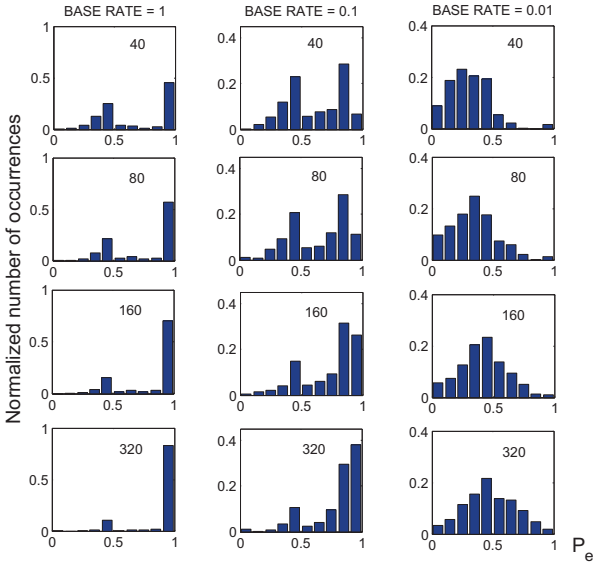
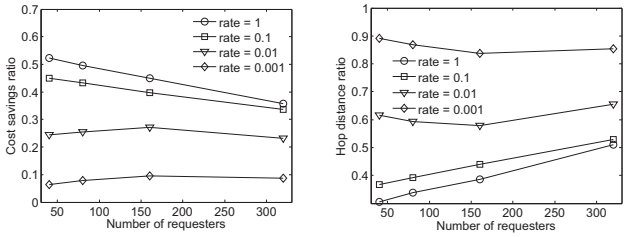


Fig. 7: Distribution of the probability of existence P_e among routers. $T_{res} = 1.5\ min$, $\Delta t = 1\ min$.

In Fig. 8, the comparison of caching IoT data items with pure source retrieval (no caching) is provided. In Fig. 8(a), this comparison is shown in terms of the mean of the expected total cost at the requesters. The cost of caching the data item (denoted CC) is compared with the cost value when no caching takes place at content routers (denoted NCC), i.e. only the

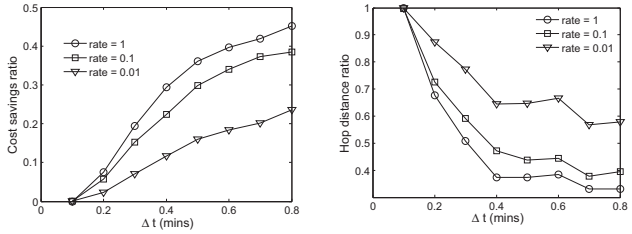
multihop delivery cost from the source node. For this, a metric called *cost savings ratio (CSR)* given by $CSR = \frac{NCC-CC}{NCC}$ is used. It can be observed that higher savings are obtained for high base rates, i.e. it is more beneficial to cache when the item is more popular and requesters are more demanding. However, cost savings decrease in high base rate settings when a larger number of requesters exist. This can be attributed to the findings shown in Fig. 7: with higher rates and more requesters, only a few routers tend to cache the item, making it costly to retrieve the item for some requesters. Still, these routers are closer to the requesters (than the source) and can provide cost savings. The *hop-distance ratio (HR)* results shown in Fig. 8(b) support this observation. HR is the average (among all requesters) of the ratio between the requester's hop distance to its serving caching router and the requester's hop distance to the source. For increasing number of requesters, the caching routers tend to be located at those locations closer to the source where multihop paths merge. When the base rate is higher, HR is lower, which means that it is possible to retrieve the data item at a closer caching location.



(a) Cost Savings Ratio (b) Hop-distance Ratio

Fig. 8: Cost savings and hop distance gain of caching wrt no caching. $T_{res} = 1.5\ min$, $\Delta t = 1\ min$.

Finally, Fig. 9 demonstrates that the time-range Δt of the request must be sufficiently large so that the routers will choose to cache the data item, which is reflected by the lower HR and higher CSR for increasing Δt values. Similar to the results in Fig. 8, caching is found to be more favourable for a higher request rate.



(a) Cost Savings Ratio (b) Hop-distance Ratio

Fig. 9: Effect of Δt on CSR and HR. $T_{res} = 1.5\ min$.

VIII. Conclusion

In this paper, the cost benefits of in-network caching of Internet-of-Things (IoT) data are studied analytically. With the expectation of billions of devices connected to the Internet, IoT data streams are likely to add large extra load; hence source retrievals must be avoided as much as possible, which makes caching at content routers an attractive option. However, different from caching large data files or multimedia data, caching IoT data items has an additional complexity: IoT data are transient, i.e. have certain lifetimes determined by content "requesters", e.g. end-user applications. Towards this, the paper provides a cost function that considers a key trade-off between (1) the multihop delivery from a source location to a requester, and (2) the expected loss in "data freshness" for the delivered data item.

Results demonstrate that the derived model accurately represents the expected behaviour of a distributed caching system: (a) Higher cost reductions are observed when (i) the data item is requested at a higher rate, i.e. a more popular content, and (ii) requesters demand the item with higher request rates; (b) IoT data caching provides less load on the network when caching routers are closer to requesters than data sources.

References

[1] J. Li, Y. Shvartzshnaider, J.-A. Francisco, R. P. Martin, K. Nagaraja, and D. Raychaudhuri, "Delivering Internet-of-Things services in MobilityFirst future Internet architecture," in *The* 3^{rd} *Int. Conference on the Internet of Things (IOT)*. IEEE, 2012, pp. 31–38.

[2] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Int. Conference on Emerging Networking Experiments and Technologies CoNEXT*. ACM, 2009.

[3] S. Borst, V. Gupta, and A. Walid, "Distributed caching algorithms for content distribution networks," in *The* 29^{th} *Annual Int. Conference on Computer Communications (INFOCOM)*. IEEE, 2010.

[4] J. Ni and D. Tsang, "Large-scale cooperative caching and application-level multicast in multimedia content delivery networks," *IEEE Communications Magazine*, vol. 43, no. 5, pp. 98–104, May 2005.

[5] M. Diallo, S. Fdida, V. Sourlas, P. Flegkas, and L. Tassiulas, "Leveraging caching for Internet-scale content-based publish/subscribe networks," in *Int. Conference on Communications (ICC)*. IEEE, 2011.

[6] L. Dong, H. Liu, Y. Zhang, S. Paul, and D. Raychaudhuri, "On the cache-and-forward network architecture," in *Int. Conference on Communications (ICC)*. IEEE, 2009.

[7] Z. Li, G. Simon, and A. Gravey, "Caching policies for in-network caching," in *Int. Conference on Computer Communication Networks (ICCCN)*. IEEE, 2012.

[8] E. J. Rosenweig and J. Kurose, "Breadcrumbs: Efficient, best-effort content location in cache networks," in *The* 28^{th} *Annual Int. Conference on Computer Communications (INFOCOM): Mini-conference*. IEEE, 2009.

[9] S. Paul, R. Yates, D. Raychaudhuri, and J. Kurose, "The cache-and-forward network architecture for efficient mobile content delivery services in the future Internet," in *Innovations in NGN: Future Network and Services*. ITU, 2008.

[10] L. Dong, D. Zhang, Y. Zhang, and D. Raychaudhuri, "Optimal caching with content broadcast in cache-and-forward networks," in *Int. Conference on Communication (ICC)*. IEEE, 2011.

[11] E. J. Rosensweig, J. Kurose, and D. Towsley, "Approximate models for general cache networks," in *The* 29^{th} *Conference on Computer Communications (INFOCOM)*. IEEE, 2010.

[12] H. Xie, G. Shi, and P. Wang, "TECC: Towards collaborative in-network caching guided by traffic engineering," in *The* 31^{st} *Annual Int. Conference on Computer Communications (INFOCOM) Mini-conference*. IEEE, 2012.

[13] I. Psaras, W. K. Chai, and G. Pavlou, "Probabilistic in-network caching for information-centric networks," in *Proceedings of the* 2^{nd} *edition of the ICN workshop on information-centric networking*. ACM, 2012.

[14] "MATLAB," www.mathworks.co.uk/products/matlab.