Caching Transient Data in Internet Content Routers

Serdar Vural, Ning Wang, *Member, IEEE*, Pirabakaran Navaratnam, and Rahim Tafazolli, *Senior Member, IEEE*

***Abstract*— The Internet-of-Things (IoT) paradigm envisions billions of devices all connected to the Internet, generating low-rate monitoring and measurement data to be delivered to application servers or end-users. Recently, the possibility of applying in-network data caching techniques to IoT traffic flows has been discussed in research forums. The main challenge as opposed to the typically cached content at routers, e.g., multimedia files, is that IoT data are transient and therefore require different caching policies. In fact, the emerging location-based services can also benefit from new caching techniques that are specifically designed for small transient data. This paper studies in-network caching of transient data at content routers, considering a key temporal data property: *data item lifetime*. An analytical model that captures the trade-off between multi-hop communication costs and *data item freshness* is proposed. Simulation results demonstrate that caching transient data are a promising information-centric networking technique that can reduce the distance between content requesters and the location in the network where the content is fetched from. To the best of our knowledge, this is a pioneering research work aiming to systematically analyze the feasibility and benefit of using Internet routers to cache transient data generated by IoT applications.**

***Index Terms*— Internet-of-Things, caching, data-transiency, data freshness, analysis, simulations.**

I. Introduction

THE realization of Internet-of-Things (IoT) [1], which envisions ubiquitous connectivity among billions of Machine-to-Machine (M2M) devices over the Future Internet [2], will necessitate effective measures that reduce the traffic load from IoT applications. Even though such devices each generate low-rate traffic of measurement, monitoring, and automation data, the presence of M2M traffic flows may become disruptive to network operations [3] and the Internet as a whole [4], [5].

A. In-Network Caching

One traditional technique that avoids unnecessary end-to-end (E2E) communications is *in-network caching* [6]–[8], which has been proven to be an effective way of delivering popular content to multiple users without explicitly contacting data sources for each and every communication attempt from requesting end-points. By temporarily storing multimedia files in content routers [9], data caching improves end-user experience by providing requested contents more quickly, and can significantly reduce in-network traffic [10].

Both push-based [11] and pull-based [12] caching have been shown to reduce in-network traffic in the past, with their own merits and deficiencies. In push-based schemes, data servers push data items to a number of fixed repositories in the network, whereas in pull-based ones, caching decisions[1] are made on an on-demand basis, i.e. based on incoming requests from data clients. Effective planning can be achieved by a push-based mechanism for static networks, where demand and supply trends are well-known or can be statistically learned, and an origin server can coordinate when and how to update cache repositories [13]–[15]. However, for dynamic environments where there exists a mixture of both near-periodic and sporadic data request traffics, the network itself should quickly adapt to the changes in locations, numbers, and rates of request traffics. In such a case, a push-based caching strategy becomes rigid. Therefore, pull-based caching, in which network nodes decide what to cache in a distributed manner, is a more suitable choice for more dynamic networks.

On the other hand, typical pull-based [16] caching techniques, which estimate data popularity [8] solely based on the rate of received requests [12] at content routers, cannot be directly applied to the case of IoT data. This is because, unlike multimedia files, IoT data are *transient* [17], i.e. in contrast to those data files that have the same content forever, IoT items "expire" in a certain time period, called *data item lifetime*, after being generated at source locations; hence, expired IoT items are no longer useful and must be discarded.

In-network caching can potentially be applied to monitoring applications which can greatly benefit from IoT data collected at specific locations, such as ambient monitoring in urban areas [18]–[21] and tracking current traffic conditions [22], [23]. For instance, an update on the recent ambient conditions, such as pollution level indicators in a location X, is popular among many location-based services which deliver local information to end-users. Depending on the popularity of specific IoT data streams collected at specific locations, content routers [9] in the Internet can cache query results [10] without relaying the requests all the way to IoT data sources.

The feasibility of using in-network caching techniques for IoT is being discussed in the Information-Centric Networking Research Group (ICNRG) [24] under the Internet Research Task Force (IRTF). Information Centric Networking (ICN) has been proposed as a network architecture which enables native content/data awareness by the underlying network,

Manuscript received December 5, 2014; revised July 15, 2015 and April 24, 2016; accepted September 23, 2016; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor S. Sen. Date of publication October 21, 2016; date of current version April 14, 2017.

The authors are with the 5G Innovation Centre, Institute for Communication Systems, Electronic Engineering Department, University of Surrey, Guildford GU2 7XH, U.K. (e-mail: s.vural@surrey.ac.uk; n.wang@surrey.ac.uk; p.navaratnam@surrey.ac.uk; r.tafazolli@surrey.ac.uk).

Digital Object Identifier 10.1109/TNET.2016.2616359

[1]The decision on whether to cache a received data item or not.

including content searching/resolution and caching. Some latest works have introduced ICN support for machine type communications, including publish/subscribe (PubSub) based data delivery in IoT applications [25], [26].

Caching decisions are likely to be dependent on the delay tolerance of different IoT applications [27] and their requirements on *data freshness* [28], [29]. Some information pieces that are periodically delivered to a remote monitoring centre for performance tracking purposes are delay tolerant [30], whereas others relating to critical events must be relayed urgently to a remote control centre [31]. For instance, caching the collected data for a time-critical IoT application, such as e-Health [32], would require high data freshness, yet at the same time a punctual response, e.g. in the e-Health context, quick medical response to sudden changes in a patient's blood pressure and heart rate [33]. Similarly, in smart-grid monitoring use-cases, locally caching recovery instructions/data instead of server retrievals could allow faster response to device failures [25]. Hence, by means of local data processing and data caching in an information-aware network, efficiency and Quality-of-Service (QoS) support can be facilitated [25].

Besides the emerging market of IoT, which this paper mainly focuses on, some other applications may also benefit from new caching technologies designed for transient data items. For instance, many mobile user applications, such as location-based services [34] that are provided to a mobile user equipment (UE) require regular and timely location updates from the UE to be delivered to application servers. Another example is news items published on websites, often advertised with tag lines like "latest news" or "now trending", and requested by many users either regionally or world-wide. Such items also have certain lifetimes [35], e.g. a few hours, until being discarded by the publishers. Finally, stock quotes and trade transactions [29] also deal with real-time data service requests. By caching these transient items in routers, systems can react to application requests for real-time data more quickly.

B. Problem Statement

In short, for transient data [15], there is a need to consider data freshness when making caching decisions, which must be performed based on dynamic variables related with not only the rate of data requests coming from applications but also data item lifetime. This leads to the following question: Based on their temporal properties, how should content routers adjust the probability to cache received transient data items? This must take into account item lifetimes, i.e. how quickly items expire.

C. Contribution

Considering this problem, this paper provides methods to determine the following:

- At a high level: considering their lifetimes, the quantifiable benefits of caching transient data items,
- At the content router level: how to efficiently cache received items to achieve high caching gains.

Towards this, the paper proposes a model, which considers "data transiency" and "in-network caching" simultaneously. The model is designed to evaluate the possibility of pull-based caching of transient data items at Internet content routers, and addresses the interplay among data freshness [36] and multihop communications [37]. The model uses a cost function which is based on data item lifetime as well as dynamically adapted system variables, such as the rate of received requests. The cost function introduces a *communications coefficient*, which is to cater for data freshness constraints posed by an IoT application provider, while also efficiently using the communication resources provided by the network operator, and hence it can be determined based on a service level agreement (SLA) between IoT application providers and network operators.

Following ICN principles, routers keep a Content Interest Table (CIT). In ICN, a CIT is used to track the neighbors interested in specific contents, hence effectively establishing a multihop data path between each requester and the data source of interest. As such, the presented analysis considers a generic multihop path between a content source and a content requester. Along this path, in an attempt to reduce their expected costs, routers dynamically modify the *probability to cache* of each content based on a received data item for that content. Hence, this proposed approach is a type of *selective caching* in which routers decide how often to cache data items based on how often contents are requested, i.e. content popularity. The concept of selective caching is concerned with whether or not to cache data items, and it can follow various strategies, not necessarily content popularity. For instance, in [38], topological characteristics are considered when making caching decisions.

Taking our previous work [39] as a starting point, this paper provides (i) a thorough description of the model, (ii) algorithmic descriptions of the simple router actions to be taken at packet reception events, and (iii) extensive packet-level simulation results performed using the Network Simulator 3 (ns3) [40] platform for performance demonstration in various parameter settings. We believe that this work will impact the long term content router architecture evolution, with intelligence and flexibility of caching transient data generated from future emerging IoT applications.

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 transient data is explained, and the theory of data item existence probability at content routers is presented. Sec. IV presents the analysis that captures the trade-off between item freshness and multihop retrieval, and then derives a cost function. Packet-level simulation results are presented in Sec. VI. Finally, Sec. VII concludes the paper.

II. Related Work

The caching algorithms in previous studies are designed for popular data files, such as multimedia, that are frequently requested by a large number of users over the Internet. However, the contents of such files are in-transient; i.e. these contents never expire. Recently the feasibility of caching transient IoT data has been discussed and supported by ICNRG [24] under the Internet Research Task Force (IRTF), and some experiments [41] have been conducted.

A. In-Transient Data

Mostly coupled with the newly emerging content-based future Internet architectures [9], [42], [43] (as opposed to the existing host-based Internet architecture), caching algorithms are considered as a feature of content delivery networks and information centric networking. The common approach is the design of 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 [7], [44]. For instance, the Breadcrumbs system in [45] 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 in [46] is based on content routers with sufficient storage that can cache large data files. Based on this architecture, later studies propose different caching algorithms to be deployed in CNF routers. In [8], 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 [47], targeting at minimal total expected content delivery delay.

Besides architectural approaches, analytical models have also been proposed for in-network caching systems, which are designed for caching in-transient data files. In [12], 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 probability to cache a file is considered to be directly proportional to the rate of incoming requests for that file, 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 [48], includes traffic engineering constraints in its general framework of cache coordination, such as link cost and utilization, besides routers' limited caching spaces. 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 [10] 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, which the network nodes should make to achieve a near optimal caching performance. A more distributed model is provided in [49], 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.

B. Transient Data

Caching transient data has not been extensively studied so far, however the possibility of caching IoT data has been argued for, and supported by some discussions [1], [24], [25], preliminary observations [41], and proactive caching models [15]. The possibility of caching either IoT system variables [1] or application data/instructions [25] has also been discussed.

The recent study in [41] supports the idea of in-network IoT data caching, as part of its general discussions on the feasibility of ICN techniques in the IoT domain. Although the study is mainly on ICN in general, but not on content caching, the benefits of IoT caching are highlighted from an energy-efficiency point of view. By means of experiments, it has been observed that up to 50% reduction in radio transmissions can be achieved by opportunistically caching IoT data packets at resource-constrained devices. Data freshness has been mentioned as a challenge that conflicts with caching, yet the paper does not study the temporal properties of IoT data items and does not provide a specific method to address the trade-off between data freshness and communication costs.

In [15], caching dynamic content in content distribution networks is analytically studied. Proactive content distribution is promoted, as opposed to pull-based caching, when contents are dynamic. By means of a central decision at an origin data server, which is based on statistical estimates on user requests for contents, data items are distributed to fixed replica servers that provide the cached contents to requesting users. Such push-based caching systems are suitable for those use cases where the locations of content distribution servers (replicas) can be strategically decided based on the locations of data servers, such as dynamic web content distribution. However, considering the large number of IoT traffic sources generating data traffic at a high variety of rates, collecting information on these dynamic traffic flows at central servers to facilitate finer push-based caching decisions would not be scalable. IoT sources may turn on and off instantaneously, make transitions between active and dormant device states, may connect/disconnect to/from the network sporadically, and even move. Hence a pull-based mechanism in which routers adapt to device/network dynamics in a distributed way based on incoming request trends is a more feasible caching solution for the IoT paradigm.

These recent works and the latest discussions on research groups signify the potential of opportunistic in-network caching of dynamic/transient data. This paper is the first study that systematically analyzes both the feasibility and the specific strategy of caching IoT data in Internet content routers. The paper shows that a simple data freshness measure that is based on data item lifetime makes it possible and beneficial to cache transient IoT items on an on-demand basis.

III. Preliminaries

A. Contents and Data Items

In this paper, all contents are uniquely named with a static content identifier, referred as CID. Different data items associated with a common content (e.g. specific reading values for a given content at different time instances) share the same CID. For each content, the source node generates *data items* in a pull-based manner, i.e. only upon receiving a request packet based on that CID. In addition to the CID field, each data item that is generated for a specific content also contains a

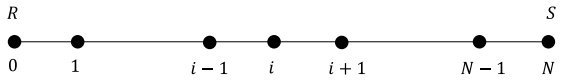

Fig. 1. Multihop path between a requester R and a data item source S.

timestamp field indicating when the data item is generated at the source, as well as a *lifetime* field indicating the duration for which the value carried in the item is valid after its generation time (indicated by the timestamp field). Based on the timestamp and lifetime values, the actual time instance when this data value expires can be obtained. Lifetime T of a content is determined and expressed by the data source, and learned by a router or requester upon receiving the data message that carries the CID of that content. Regarding request messages, each request carries a CID field, indicating what content is requested. Routers index cached data items according to their CIDs only; i.e. there can be at most one data item in a router's cache regarding a specific content at a time.

B. Data Paths

Routers keep track of their neighbor routers that have requested a particular content by recording the router IDs in their CIT entry for that content, and forward received data items to these neighbors in the reverse direction, effectively forming a multihop path between each requester of a content and its serving point.

In this paper, the path that a data item traverses is modelled as shown in Fig. 1, where the source and requester routers are depicted as S and R, respectively, and S is N hops away from R. Node R is a generic IoT data requesting client, which connects to the Internet, where the first-hop Internet router is represented by node 1 in Fig. 1. Similarly, node S is where a data item is generated. Without loss of generality, it can represent either directly an IoT device or an IoT gateway, connected to an Internet router depicted by node $N-1$ in the figure.

Request packets travel from R to S, whereas the data item provided by S is forwarded to R in the reverse direction. For instance, the router that is i hops away from R receives the item from hop $i+1$,[2] which was fetched either from the data source S itself or from the cache of another router located on the multihop path towards the source, i.e. $i+2, \ldots, N-1$.

In the remainder of this section, two key concepts, namely "data item freshness" and "probability of data item existence" are introduced.

C. Data Item Freshness

When caching a transient data item, it is essential to consider its lifetime T, which is the time period during which the item is considered to be valid, following the instance when the item is generated at its source. When a data item is received by a router, the router evaluates how 'fresh' the item is, so as to decide on its "cacheability". The router takes into account: the time of arrival t_{arr} of the item at the router, the time of item generation/availability t_{gen} at its source (the edge Internet router), and its lifetime T that it can spend in the Internet. The router considers two time windows of length T, following the time instances t_{gen} and t_{arr}, and determines how much overlap exists between these two time windows. This is shown in Fig. 2(a) and (b). In Fig. 2(a), there is some overlap, indicating a certain amount of data freshness, whereas in Fig. 2(b), no overlap exists, i.e. a non-fresh item. In these figures, *data age* denotes the age of the item, which is the time period between the arrival at the router and the availability at the source. In short, when the item's *data age* at the time of reception by the router is smaller than its lifetime T, the item is considered to have a certain level of data freshness.

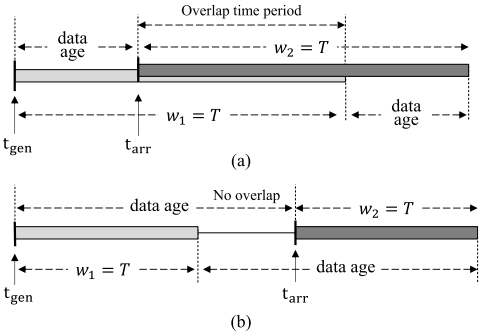

Fig. 2. Data item freshness. (a) Fresh. (b) Non-fresh.

Basically, the larger the overlap, the fresher the item (i.e. if the source and the router were co-located with no latency in packet delivery, then 100% freshness would be achieved). Accordingly, the *freshness* of a transient data item can be defined as:

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

Items with a negative freshness value (when data age is larger than lifetime) are not cached at content routers.

1) Freshness Loss: Due to in-network time delays, data items have finite non-zero data ages when received by routers. When a data item is retrieved from its source S by a router x, a certain amount of cumulative networking time delay[3] occurs, denoted by $d(S,x)$. This reduces the item's *residual* lifetime during which the item can be considered to have some level of freshness, i.e. $T_{res} = T - d(S,x)$, which yields a freshness of $\frac{T-d(S,i)}{T}$.

If the data item additionally gets cached at network routers and then this cached item is retrieved by requesters, then there is an extra amount of *caching age*,[4] which is the cumulative time period that the item resides in router caches until requested. An item retrieved from a router's cache is hence "less fresh", compared to an item retrieved from its source S, and have a freshness of $\frac{T-d(S,i)-Caching\ age}{T}$.

[2]The terms "router i", "router at hop i", and hop "i" are used interchangeably.

[3]The networking time delay refers to the sum of propagation, packet transmission, queueing, and processing delays on a multihop path.

[4]The time period spent at a router's cache contributes to a retrieved item's total age when received, hence the term "age".

Therefore, we define the *freshness loss* (FL) of a data item as the reduction in freshness caused by caching at routers, given by:

$$FL = \frac{Caching\ age}{T}. \tag{2}$$

D. Probability of Fresh Data Item Existence

Routers can respond to a request for a transient data item only if a non-expired copy of the item exists in their caches at the time the request is received, i.e. a *cache hit*. Therefore, it is essential to first define the probability that a given non-expired data item exists in a router at a random time, called the *probability of existence*, P_e.[5] 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 transient data, and consider a specific data item received by the router with a *residual lifetime* $T_{res} = T - data\ age$. The incoming request rate for the content is denoted by r_{in}, which is the total rate of requests for that content received by the router from its neighbors.[6] 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 $r_{out} = (1 - P_e)r_{in}$. In other words, r_{out} is the rate at which a router sends a request to its next hop towards the content's source. Furthermore, following content-centric networking principles, data items are returned to only those routers that request them; i.e. the rate of data messages arriving at the router is $r_d \approx r_{out}$. This is achieved by keeping a content interest table (CIT) at routers, which keeps a record of which data item has been requested by which neighboring router.

Let P_c denote the probability to cache the item when it arrives at the router. Then, the rate r_c of caching the item is:

$$r_c = r_d P_c \approx (1 - P_e)\, r_{in} P_c. \tag{3}$$

For a caching rate of r_c, the average time period between successive caching events is approximately $\frac{1}{r_c}$. When $\frac{1}{r_c} > T_{res}$,[7] the fraction of time that the item exists in the router's cache is simply its probability of existence: $P_e = \frac{T_{res}}{1/r_c} = T_{res} r_c$ (See Fig. 3). Therefore, $\frac{P_e}{T_{res}} \approx (1 - P_e)\, r_{in} P_c$, which gives:

$$P_e \approx \frac{r_{in} P_c}{\frac{1}{T_{res}} + r_{in} P_c}. \tag{4}$$

Based on this definition, two conclusions can be drawn on how P_e changes with respect to P_c and r_{in}, as illustrated by Fig. 4: (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 an item more often by picking a higher P_c for that item, this increases the likelihood of finding the item in its cache.

[5]Note that cache hit occurs provided that a fresh item exists in the router cache, and only when a request is received.

[6]The rate of request arrival r_{in} differs at different routers, and depends on the average rate of request generation at requesters as well as the position of routers with respect to requesters and data source.

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

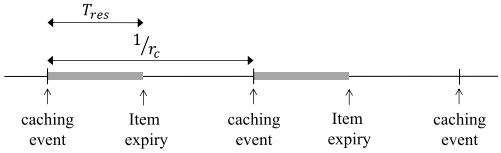

Fig. 3. Caching events and item expiry at a content router: Probability of item existence in cache.

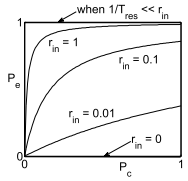

Fig. 4. Probability of existence P_e as a function of probability of caching P_c and incoming request rate r_{in}.

IV. Analysis

In this section, the trade-off between communication costs of data item retrieval and freshness loss cost of cached items is studied analytically. First, caching age, i.e. the time period during which items stay in router caches, is modelled, which is used to model freshness loss. Then, communication cost of an item at a router is analyzed. Finally, a cost function is introduced, which combines freshness loss cost and communication cost.

The analysis in this section considers a multihop path between a requester R and a data item source S, as illustrated in Fig. 1. Mathematical expressions are derived for the router located at hop i from R.

A. Caching Age (CA)

When an item is cached at a router, a certain amount of caching age occurs for the time the item stays in the router's cache. In fact, the item fetched from hop i's cache might have been cached several times on the multihop path from the source. Due to this probabilistic nature of the item's retrieval location, in this section, the expected total caching age, denoted by $E[CA(i)]$, is studied.

The expected total caching age $E[CA(i)]$ of the item at hop i is the sum of: (1) the expected caching age of the item at the instance when it was received by hop i, and (2) the average period of time the item has stayed in hop i's cache by the time a request for the item arrives at hop i, called the *expected additional caching age*. In the following, these two components of caching age are studied.

1) Expected Caching Age of an Item Received by Hop i: When a router i receives a request for a data item, there are two cases that may happen: (1) the router can respond to the request immediately if it has a cached item, which has a probability $P_e(i)$, and (2) the router does not have a cached item and thus forwards the request to its next hop towards the data source, which has a probability $1 - P_e(i)$. In the second case, the item could be fetched from an intermediate router on the path to the source node, or from the source itself if none of the routers on the path have cached the item. Hence, a cached item at hop i may have been previously cached at one

or more routers, i.e. $i+1, \ldots, N-1$. The expected caching age that the received item at hop i has is denoted by $G(i+1)$, as the item comes from hop $i+1$.

The value of $G(i+1)$ depends on where the item that is received by hop i from hop $i+1$ is actually fetched from. Denoting the probability that the item exists at hop k as $P_e(k)$, when hop i forwards the request to hop $i+1$ towards the data source, the probability that hop j, where $i < j \leq N$, returns the item is:

$$Prob\{\text{Router at hop } j \text{ replies}\} = P_e(j) \prod_{k=i+1}^{j-1} (1 - P_e(k)), \tag{5}$$

where the product term denotes the probability that all hops $i+1 \leq k < j$ do not have a cached item. Using Eqn. 5, the expected caching age that hop i receives from hop j due to potential caching at a hop j, where $i < j \leq N-1$, is:

$$\left(\prod_{k=i+1}^{j-1} (1 - Pe(k)) \right) P_e(j) E[CA(j)]. \tag{6}$$

Considering all possibilities, the expected caching age of an item received by hop i can then be estimated by:

$$\begin{aligned} &G(i+1) \\ &\quad = P_e(i+1)E[CA(i+1)] \\ &\qquad + (1 - P_e(i+1))P_e(i+2)E[CA(i+2)] + \ldots \\ &\qquad + \left(\prod_{k=i+1}^{N-2} (1 - Pe(k)) \right) P_e(N-1)E[CA(N-1)]. \end{aligned} \tag{7}$$

For instance, if the item was fetched from hop $i+1$'s cache, i.e. $P_e(i+1) = 1$, then $G(i+1) = E[CA(i+1)]$, where $E[CA(i+1)]$ is the expected total caching age of the item at hop $i+1$. If the item was fetched from hop $i+2$'s cache, i.e. $P_e(i+1) = 0$ and $P_e(i+2) = 1$, then $G(i+1) = E[CA(i+2)]$. The special case $G(i+1) = 0$ occurs when the item was fetched from the source, as items at their sources have no caching age, i.e. $E[CA(N)] = 0$.

The function $G(i+1)$ in Eqn. 7 can be written in recursive form as $G(i+1) = P_e(i+1)E[CA(i+1)] + (1 - P_e(i+1))G(i+2)$. This gives the following expression for $G(i)$:

$$G(i) = P_e(i)E[CA(i)] + (1 - P_e(i))G(i+1). \tag{8}$$

At the source node $G(N) = 0$, as $P_e(N) = 1$ and $E[CA(N)] = 0$. Furthermore, $G(N+1) = 0$, as hop $N+1$ does not exist.

2) Expected Additional Caching Age, $\overline{CA}(i)$: Upon reception by a hop i, an additional caching age is introduced to the data item if hop i decides to cache the item as well. This accounts for the time period the item spends in the router's cache until when it is used to respond to a request. By definition, the expected value of the additional caching age at the source node is $\overline{CA}(N) = 0$.

As shown in Fig. 5, the expected additional caching age $\overline{CA}(i)$ depends on the time instance when the request arrives at the router after the item is cached. In other words, before the item expires (has a remaining lifetime $T_{res} = 0$) different requests are answered by items with different caching ages, depending on when these requests arrive at the router.

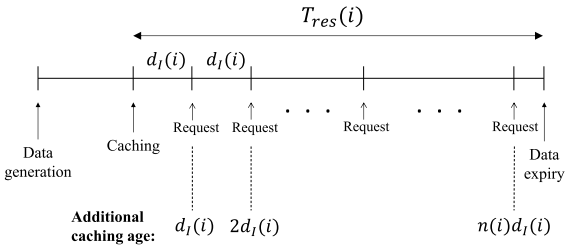

Fig. 5. Additional caching age.

The average inter-arrival time between consecutive request arrivals for an item at hop i is approximately equal to the reciprocal of the incoming request rate, i.e. $d_I(i) = \frac{1}{r_{in}(i)}$. Here, the request rate $r_{in}(i)$ is an average value computed and updated by router i based on a recent history of request arrivals, and reflects an instantaneous snapshot of the recent request arrival rate (Algorithm 1 in Section V-A). Using $d_I(i)$, the average number of times $n(i)$ that the cache of the router at hop i can respond to incoming requests can be approximated by:

$$n(i) \cong \frac{T_{res}(i)}{d_I(i)} = T_{res}(i).r_{in}(i), \tag{9}$$

where $T_{res}(i)$ is the residual lifetime of the item when it arrives at hop i, as defined in Section III-D. For the k^{th} request arriving at the router, the expected additional caching age is approximately k average inter-arrival times, i.e. $k.d_I(i)$.

Considering that a request arrives at a random time instance, the expected value of the caching age that the item experiences at hop i depends on a comparison of $d_I(i)$ and $T_{res}(i)$, as follows:

$$\begin{aligned} &\text{If } \frac{T_{res}(i)}{2} \leq \frac{1}{r_{in}} < T_{res}(i), \quad \text{then: } \overline{CA} = \frac{1}{r_{in}}, \\ &\text{else if } T_{res}(i) \leq 1/r_{in}, \quad \text{then: } \overline{CA} = 0, \\ &\text{else if} \frac{1}{r_{in}} < \frac{T_{res}(i)}{2}, \quad \text{then:} \\ &\overline{CA} = \frac{\left\lfloor \frac{T_{res}(i)}{1/r_{in}} \right\rfloor}{2} . \frac{1}{r_{in}} \leq \frac{T_{res}}{2} < \frac{T}{2}. \end{aligned} \tag{10}$$

In Eqn. 10, the different cases arise from the expected number of times that the item can be used to respond to a request after being cached. In the first case, the item can be used only once, as there is enough time just to fit in one interarrival time $d_I(i)$ in $T_{res}(i)$. In the second case, $d_I(i) > T_{res}(i)$, i.e. it is expected that the first request packet is to arrive after the item expires; hence routers do not cache such items. In the last case, more than one cache returns are expected to happen, hence $\overline{CA}(i)$ depends on how many $d_I(i)$ can fit in T_{res}.

3) The Expected Total Caching Age of an Item at Hop i: The expected total caching age $E[CA(i)]$ of a cached item at hop i is the sum of the expected additional caching age $\overline{CA}(i)$ caused by caching at hop i, which is given by Eqn. 10, and

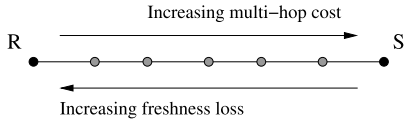

Fig. 6. The trade-off between freshness loss cost and communication cost.

the expected caching age $G(i+1)$ of the retrieved item from hop $i+1$, given by Eqn. 7. Hence:

$$E[CA(i)] = \overline{CA}(i) + G(i+1), \tag{11}$$

Note that $E[CA(N-1)] = \overline{CA}(N-1)$ for the last hop $N-1$ before the source node, as $G(N) = 0$. Plugging this in Eqn. 8, and as $G(N) = 0$, we have $G(N-1) = P_e(N-1)\overline{CA}(N-1)$.

B. Communication Cost and Freshness Loss Cost

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 communication cost. This is shown in Fig. 6, where R is the requester and S is the source. In this section, these two cost terms are derived for the router at hop i.

1) Expected Freshness Loss Cost of an Item Returned by Hop i: When hop i receives a request, it either returns the item itself if it has a non-expired cached copy, or otherwise it forwards the request towards the source. Using the definition given by Eqn. 2, the expected freshness loss cost per bit of an item returned by hop i from its cache is:

$$\widetilde{FL}(i) = \frac{E[CA(i)]}{T}, \tag{12}$$

where the expected caching age $E[CA(i)]$ is given by Eqn. 11 in Sec. IV-A. This happens with probability $P_e(i)$. On the other hand, if the item does not exist in hop i's cache, which has a probability $1 - P_e(i)$, it is retrieved from another hop $i < j \leq N$ on the path towards the source S. In that case, the expected freshness loss cost of the fetched item by hop i is:

$$P_e(i+1)\widetilde{FL}(i+1) + \sum_{j=i+2}^{N} \left(\prod_{k=i+1}^{j-1} (1 - Pe(k)) \right) P_e(j)\widetilde{FL}(j). \tag{13}$$

Hence, the expected freshness loss cost of an item returned by hop i, either from its own cache or retrieved, is:

$$FC(i) = P_e(i)\widetilde{FL}(i) + \sum_{j=i+1}^{N} \left(\prod_{k=i}^{j-1} (1 - Pe(k)) \right) P_e(j)\widetilde{FL}(j). \tag{14}$$

Using Eqns. 13 and 14, $FC(i)$ can be written in the following recursive form:

$$FC(i) = P_e(i)\widetilde{FL}(i) + (1 - P_e(i))FC(i+1). \tag{15}$$

where $FC(i+1)$ is simply Eqn. 13. Since there is no freshness loss cost at the source, i.e. $FC(N) = 0$, this gives $FC(N-1) = P_e(N-1)\widetilde{FL}(N-1)$ for the last hop router before the source.

The freshness loss cost function $FC(i)$ is in fact a multiple of $G(i)$. Multiplying both sides of Eqn. 8 by $\frac{1}{T}$, and using Eqn. 12 we obtain:

$$FC(i) = \frac{G(i)}{T}. \tag{16}$$

Then, using Eqns. 11 and 12 for $\widetilde{FL}(i)$ and Eqn. 16 for $FC(i+1)$ in Eqn. 15:

$$\begin{aligned} FC(i) &= \frac{P_e(i)}{T}\left(\overline{CA}(i) + G(i+1)\right) + \frac{1}{T}(1 - P_e(i))G(i+1) \\ &= \frac{1}{T}\left(P_e(i)\overline{CA}(i) + G(i+1)\right) \end{aligned} \tag{17}$$

Furthermore, using Eqn. 11 in Eqn. 8:

$$G(i) = P_e(i).\overline{CA}(i) + G(i+1). \tag{18}$$

Eqns. 16 and 18 are the final expressions to estimate the freshness loss cost $FC(i)$ of an item retrieved from hop i. The lemma below states that $FC(i)$ is a normalized value.

Lemma 1: The freshness loss cost $FC(i)$ of an item retrieved from hop i is normalized, i.e. $FC(i) \leq 1$.

Proof: A non-expired (fresh) data item received by hop i must have a non-zero residual lifetime, i.e. $T_{res}(i) > 0$. This suggests that the sum of all caching age contributions from hops $i+1, \ldots N-1$ (routers towards the source) cannot exceed item lifetime, hence $\sum_{j=i}^{N-1} \overline{CA}(j) \leq T$. Then, using Eqn. 18 and since $\overline{CA}(N) = 0$, we have:

$$FC(i) = \frac{G(i)}{T} = \frac{1}{T}\sum_{j=i}^{N-1} P_e(j)\overline{CA}(j) \leq \frac{1}{T}\sum_{j=i}^{N-1} \overline{CA}(j) \leq 1, \tag{19}$$

■

This concludes the analysis on freshness loss cost. Next, communication cost is analyzed.

2) Expected Communication Cost of an Item Returned by Hop i: The expected communication cost of an item returned by hop i is defined by:

$$CC(i) = \frac{\overline{h}(i)}{N}, \tag{20}$$

where $\overline{h}(i)$ is the expected number of hops between router i and the location where the item is fetched from by hop i (when it does not possess it) on the multihop path towards the source node. Communication cost incurs when hop i does not have a cached item, which happens with probability $1 - P_e(i)$. In Eqn. 20, $\overline{h}(i)$ is divided by N, i.e. the hop distance between the requester and the data source, so that the cost is normalized. The *expected hop distance* $\overline{h}(i)$ can be computed by:

$$\overline{h}(i) = \sum_{j=i+1}^{N} \left(\prod_{k=i}^{j-1} (1 - Pe(k)) \right) P_e(j)(j - i), \tag{21}$$

where $(j - i)$ is the number of hops from i to j, with $i < j \leq N$, which is multiplied by the probability that the item is fetched from hop j when hop i does not possess it.

The probability of fetching an item from hop j, with $i < j \leq N$, is given by Eqn. 5, and is multiplied by $1 - P_e(i)$ to reflect the cases when the item does not exist at hop i. Note that if $P_e(i) = 1$, then $\overline{h}(i) = 0$.

Since the source node, which is at hop N, always has a fresh data item, $P_e(N) = 1$ by definition. With this, Eqn. 21 can be reduced to:

$$\overline{h}(i) = \sum_{j=i+1}^{N} \left(\prod_{k=i}^{j-1} (1 - Pe(k)) \right) < N. \tag{22}$$

This equation can then be written in recursive form as:

$$\overline{h}(i) = (1 - P_e(i)) \left[\overline{h}(i+1) + 1 \right]. \tag{23}$$

Since there is no communication cost at the source, i.e. $\overline{h}(N) = 0$, for the last hop router at hop $N-1$, this gives $\overline{h}(N-1) = (1 - P_e(N-1))$.

Using Eqns. 20 and 22, the expected communication cost can be written in the following final form as provided in Eqn. 24, where the division by N is a normalization so that the communication cost of a full path-traversal between a requester and the data source is 1.

$$CC(i) = \frac{1}{N}(1 - P_e(i)) \left[\overline{h}(i+1) + 1 \right]. \tag{24}$$

C. Cost Function

The two cost terms, i.e. freshness loss cost $FC(i)$ and communication cost $C(i)$ are contradictory. The minimal freshness cost, which happens when the item is fetched from its source, comes at the cost of maximal communication cost. On the contrary, when communication cost is minimized (items are deterministically cached ($P_c = 1$) at routers) the expected freshness loss of the retrieved item is not minimal. This suggests that if the network routers are configured to cache items with higher probability, this reduces their communication costs with a sacrifice in item freshness. Similarly, if a certain level of item freshness is desired, then routers must retrieve the item from its source occasionally. In fact, retrieval from source is inevitable as items are transient, i.e. they expire in time.

In order to strike the balance between these two contradicting objectives, the following cost function is defined:

$$C(i) = \alpha CC(i) + (1 - \alpha) FC(i), \tag{25}$$

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 communication retrieval cost, i.e. a high α means higher cost of retrieval and indicates that the application does not prefer frequent source retrieval.[8] The cost function $C(i)$ can be written as a function of $P_e(i)$, as follows:

$$C(i) = \alpha \frac{\overline{h}(i)}{N} + (1-\alpha)\frac{G(i)}{T} = \frac{\alpha}{N}(1-P_e(i)) \left[\overline{h}(i+1)+1 \right] + \frac{(1-\alpha)}{T} \left[P_e(i)\overline{CA}(i) + G(i+1) \right]. \tag{26}$$

[8]The specific value of α must be determined based on a service level agreement (SLA) between IoT application providers and network operators, so as to cater for specific IoT application constraints (in particular in terms of data freshness) while efficiently using the network's communication resources.

TABLE I
SYSTEM VARIABLES

Notation	Name	Description
r	Request rate	Rate of requests sent by the requester
r_{in}	Incoming request rate at a router	Monitored over a window of most recent request packet reception time instances
T_{res}	Residual data item lifetime	The remaining time period after an item's reception time at a router until its expiry
$\overline{CA}$	Expected added caching age	Average time period that an item resides at a router's cache, when cached
P_c	Probability of caching	Probability that a router caches a received item

In Eqn. 26, $G(i+1)$ and $h(i+1)$ are feedback values[9] from hop $i+1$ to hop i.

The cost function in Eqn. 26 can be written in the following form, as a function of P_e:

$$C(i) = \left[\frac{1}{T}(1-\alpha)\overline{CA} - \frac{\alpha}{N}(\overline{h}(i+1)+1) \right] \times P_e(i) + \frac{\alpha}{N}(\overline{h}(i+1)+1) + \frac{1}{T}(1-\alpha)G(i+1). \tag{27}$$

V. Router Action

In this section, the router actions are described. First, the main system variables are introduced. Then, the actions at data item and request packet reception events are explained. The key system variables are listed in Table I.

A. Incoming Request Rate, r_{in}

The rate of incoming request packets is updated each time a request packet is received. To achieve this, a sliding time window of request reception times is kept and updated. Algorithm 1 outlines this procedure. The algorithm records the time period of the most recent W request packet reception instances to estimate r_{in}.

B. Update of the Probability of Caching, P_c

Initially, $P_c = 0$, as no data item has been received so far. Upon reception of a fresh data item, routers first compute the residual lifetime of the item and determine whether it can be used to serve incoming requests. The criterion of this initial filtration is $d_I < T_{res}$, i.e. item lifetime should be more than the expected inter-arrival time of the first request packet for it, so that at least one request can be serviced while the item is in the cache. If $d_I \geq T_{res}$, then the router sets $P_c = 0$. Otherwise, P_c is increased/reduced according to expected gains in the cost function (see Eqn. 27), as follows:

$$\begin{aligned} &\text{If } \frac{1}{T}(1-\alpha)\overline{CA} \geq \frac{\alpha}{N}(\overline{h}(i+1)+1) \text{ then:} \\ &\qquad \textit{Decrement Pc} \\ &\text{Else if } \frac{1}{T}(1-\alpha)\overline{CA} < \frac{\alpha}{N}(\overline{h}(i+1)+1) \text{ then:} \\ &\qquad \textit{Increment Pc} \end{aligned} \tag{28}$$

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

Algorithm 1 Update of Incoming Request Rate r_{in}

N_{req}: number of requests received so far for this item
$t_{req}[1 \dots N_{req}]$: array of request reception times (most recent last)
W: Time window length
$time$: Time of arrival of the current request packet

procedure UPDATEREQUESTRATE($time$)
  if $N_{req} < W$ **then**
    $t_{req}[N_{req}+1] = time$;
    $N_{req} = N_{req} + 1$;
  else
    Shift the window W one position
    Write new $time$ value to the last position in $t_{req}[\dots]$
  end if
  if $N_{req} == 1$ **then**
    $r_{in} = 1/W$; ▷ Initialization value
  else
    if $N_{req} < W$ **then**
      $r_{in} = N_{req} / (t_{req}[N_{req}-1] - t_{req}[0])$;
    else
      $r_{in} = W / (t_{req}[N_{req}-1] - t_{req}[N_{req}-W])$;
    end if
  end if
end procedure

Equation 28 suggests that if in-network caching for a particular content is not desired at all, then $\alpha = 0$ is to be set for that content, as this sets the right hand side of the equation to 0, effectively setting caching probability to 0.

A monotonic increase/decrease in P_c causes the same type of change in P_e due to the relation between the two, as provided by Eqn. 4.

C. Data Item Arrival

Algorithm 2 outlines the operations performed upon reception of a data item. Router i makes decisions based on the residual lifetime $T_{res}(i)$, current rate of incoming requests $r_{in}(i)$, and received feedbacks $G(i+1)$ and $\overline{h}(i+1)$.

The router first estimates the probability of existence $P_e(i)$ value, based on the current $P_c(i)$, $r_{in}(i)$, and $T_{res}(i)$. Then, if the item does not have sufficient lifetime, it is discarded, and $P_c(i)$ is reset to 0. Otherwise, the added caching age $\overline{CA}(i)$ is estimated by Eqn. 10, which is used to update $G(i)$ in the next step.

Following the update of $P_e(i)$, $P_c(i)$ is modified based on the criterion in Eqn. 28, and finally the item is cached probabilistically based on the updated value of $P_c(i)$. In the proposed system, if the available caching space is full, then any expired item is evicted from the cache to make space for the new item (Section V-F).

D. Request Packet Arrival

Upon receiving a request packet, router i first updates the request rate $r_{in}(i)$. The operations are outlined in Algorithm 3. If the router has a matching cached item and it has expired,

Algorithm 2 Data Item Arrival

t_{rec}: Item reception time at router
t_{gen}: Item generation time at source
h_f: $\overline{h}(i+1)$ feedback from the neighbor sending the item
G_f: $G(i+1)$ feedback from the neighbor sending the item
Δp: Increment/decrement amount in P_c

$T_{res} = t_{rec} - t_{gen}$;
$P_e = \frac{P_c \times r_{in}}{P_c \times r_{in} + 1/T_{res}}$;
if $\frac{1}{r_{in}} \geq T_{res}$ OR $T_{res} \leq 0$ **then**
  Discard item;
  $\overline{h} = h_f + 1$; $G = G_f$; $P_c = 0$;
else
  Compute $\overline{CA}$; ▷ See Eqn. 10
  $G = P_e \times \overline{CA} + G_f$; ▷ See Eqn. 18
  $\overline{h} = (1 - P_e) \times (h_f + 1)$; ▷ See Eqn. 23
  if $\frac{1}{T}(1-\alpha)\overline{CA} \geq \frac{\alpha}{N}(h_f + 1)$ **then**
    $P_c = P_c + \Delta p$;
  else
    $P_c = P_c - \Delta p$;
  end if
  if $rand(0,1) < P_c$ **then**
    Cache item;
  end if
  Send item to the Earliest Requesting Node in CIT;
  Remove the Earliest Requesting Node from CIT;
end if

then the cached item is evicted,[10] and the request packet is forwarded to the next hop towards the source. Router i adds the requesting node $prev$'s ID to its CIT. If a fresh cached item is found, this is a cache-hit situation, in which case the router updates the $\overline{h}$ and G values in the item header (i.e. enters its own feedback information) and returns the item to $prev$. The feedback values have been computed upon item reception right before caching (Alg. 2). If there is no matching item in cache, $prev$ is added to the CIT and the request is forwarded towards the source.

E. Algorithm Complexity

The proposed caching system has two core decision points: (1) Decision on item cacheability, and setting $\overline{CA}$ by Eqn. 10, (2) Increment/decrement of P_c based on the comparison operation given by Eqn. 28. These are simple ALU operations, along with the rest of the operations, i.e. computation of P_e, G, $\overline{h}$, and T_{res}. These operations are performed upon data item reception (Algorithm 2). Request arrival operations are typical in any caching system; and the feedback values of $\overline{h}$ and G have already been computed at data item arrival time (as these computations are performed only when a cache hit occurs). Hence, the complexity of the proposed system is the number of these ALU operations performed in each data item reception instance.

[10]Timers are not used for cache entries; items are evicted when found expired at the time of request reception or a new item arrival when cache space is limited.

Algorithm 3 Request Arrival

```
Update r_in                                   ▷ See Algorithm 1
if Cache entry exists then
    if Entry has expired then                 ▷ T_res ≤ 1/r_in
        Evict cached item;
        Forward Request to next hop towards Source;
        Add prev in Content Interest Table;
    else                                      ▷ Cache hit
        Update h̄ and G in item header;
        Return the item to the requesting neighbor prev in
          CIT;
    end if
else                                          ▷ No matching entry in cache
    Forward Request to next hop towards Source;
    Add prev in Content Interest Table;
end if
```

F. Cache Space Use and Cache Evictions

The distinguishing property of transient data items is that such items expire in time.[11] Therefore, as opposed to conventional in-transient items, a separate timer would be needed to keep track of which item expires in the cache, or else a periodic scan of the cache space would be required to monitor residual item lifetimes. To avoid this, cache evictions must be applied only when needed, i.e. when a new fresh item arrives.

Another important aspect of the proposed caching mechanism is that there can be at most one data item about a given content at a time in each router's cache. As a result, if a fresher data item arrives (as compared to the cached data item), only then the router makes a caching decision (based on the current value of the caching probability regarding that content), and replaces the existing item if a caching decision has been made.

The cache eviction policy for transient items can hence be summarized as follows. When an unexpired data item X arrives at a router and a caching decision has been made for the item, then another data item Y is evicted from the cache if one of the following conditions holds:

1) The cache space is full and the item Y (whether or not X and Y are of the same content) has expired,
2) The cache space is full and the item Y (whether or not X and Y are of the same content) has not expired, but Y is the least fresh item in the cache (called the Least Fresh First (LFF) eviction policy),
3) Cache is not full, and items X and Y are of the same content. This is the case when X is an updated value, and hence it is fresher than Y.

VI. Performance Evaluation

The performance of the proposed caching mechanism for transient contents is evaluated via simulations using Network Simulator 3 (ns-3) [40]. A multihop path between a requester R and a data source S (see Fig. 1) is evaluated.

[11]The system envisions a separate cache space for transient data items so as to avoid any potential dependency between transient and in-transient items.

TABLE II
KEY PERFORMANCE METRICS

Name	Description
Cache hit ratio	Fraction of times a request finds a fresh cached item
Retrieved freshness	Average freshness of items received by the requester
Normalised average hop distance to retrieval location	Ratio $\lvert hops\rvert = \frac{\overline{h}(0)}{N}$ between (1) the average number of hops $\overline{h}(0)$ from the requester to the location where the item is fetched from, and (2) the path length N

A. Simulation Settings

The paper specifically focuses on caching decisions based on item lifetimes, as the focus of the paper is data transiency. Hence, the main system parameter is item lifetime T, and in each setting 8 different contents are present at S, each of which has a different lifetime, i.e. $T = 1, 5, 10, 30, 60, 120, 240, 300$ seconds. Data item size is 512 kB, and link time delay is 10 ms. The increment/decrement of Pc in Alg. 2 is set to $\Delta p = 0.001$.

Various simulations settings have been evaluated, to observe the behavior of the proposed caching system with respect to varying system parameters, which are: *(i) request rate* r (requests/second), *(ii) path length* $N = 5, 10, 20$ hops between R and S, and *(iii) communication coefficient* $\alpha = 0, 0.1, \ldots, 0.9, 1$ (see Section IV-C).

Request packets are sent from R to S for every content, where the request packet generation follows a stationary Poisson process with exponential inter-arrival times,[12] corresponding to an average rate denoted by r requests/second, where request rates are $r = 0.01, 0.05, 0.1, 0.2, 0.5, 1$.

The first set of results are produced with sufficient cache space at routers to accommodate all of the 8 contents at a time. This is because IoT data items are negligibly small as compared to today's router cache spaces, and the proposed caching scheme allows at most one data item for each content. However, for analysis purposes, the results for caching limited cases in which router cache sizes have $1, 2, 4$ spaces are also presented.

In all figures presented, the data points represent average values calculated over 40 simulation runs for each setting.

B. Key Performance Metrics

The key performance metrics presented in this section are summarized in Table II.

C. Results

In this section, results for the three performance metrics (Table II) are illustrated. In each figure, results are with respect to T or α in the x-axis, and graphs for different values of one or two other parameters are presented in order to observe the changing behavior with respect to multiple parameters at a time. When a parameter's variation has not been shown in a figure, a default value is used for that parameter. Unless otherwise stated, the default values of the system parameters are as follows: $T = 60$ seconds, $N = 10$ hops, $\alpha = 0.5$, and

[12]This is a typical packet generation distribution frequently used to model telecommunication systems.

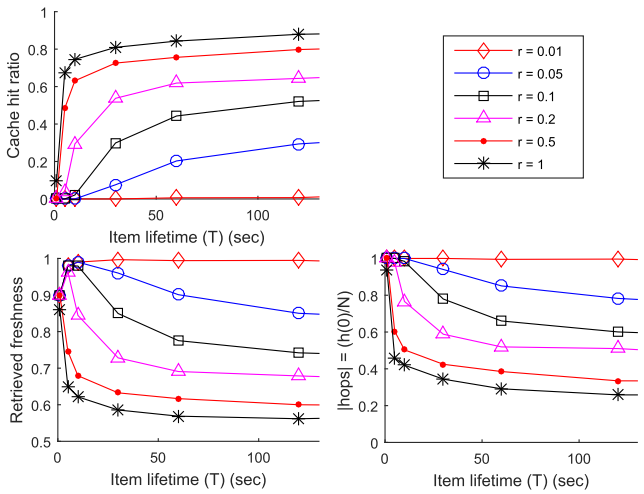

Fig. 7. Effect of item lifetime T and request rate r; $\alpha = 0.5$, $N = 10$ hops.

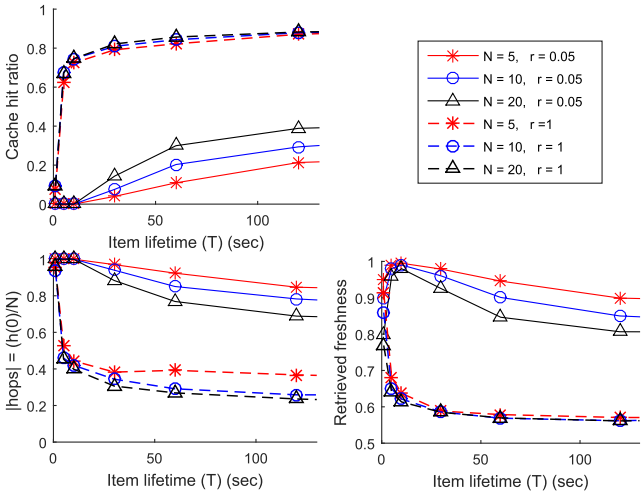

Fig. 8. Effect of item lifetime T and path length N; $\alpha = 0.5$, rates: $r = 0.05$ and $r = 1$ reqs/sec.

$r = 0.2$ requests/second. In Figs. 7, and 8, 9, for increasing lifetime values after the $T = 120$ datapoints, insignificant changes in performance metrics have been observed; hence to better demonstrate the variations for lower T, these figures show $T = 1, 5, 10, 30, 60, 120$.

1) Request Rate: Fig. 7 considers different request rate (r) scenarios, and illustrates how contents with different lifetimes (T) are affected by the request rate parameter. As expected, higher cache hit rates are achieved for higher r, except for the content with the lowest lifetime $T = 1$, which the routers prefer not to cache due to the expectation that items of that content would expire quickly and hence received requests would not be served from cache (see Eqn.10, the precondition for caching, i.e. $T \geq T_{res}(i) > 1/r_{in}$ must hold). If the request rate is sufficiently low, all tested content lifetime cases do not show cache hits ($r = 0.01$ reqs/sec).

The figure also demonstrates the tradeoff between retrieved freshness and hop distance to retrieval location. For the low request rate case, due to no cache hits, items are almost always retrieved from the source, and have maximal freshness.

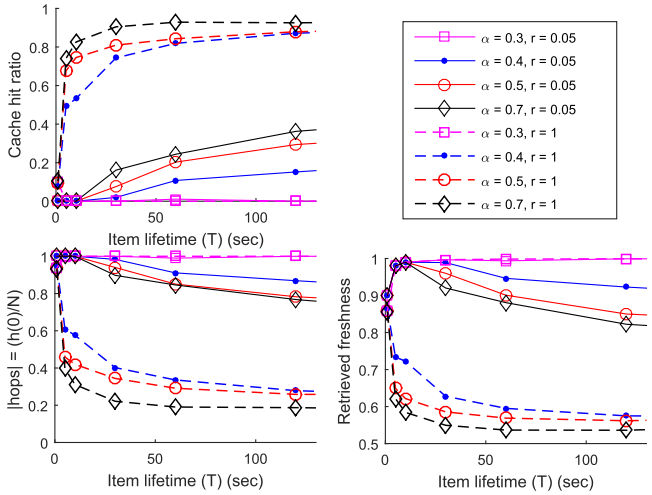

Fig. 9. Effect of item lifetime T and communication coefficient α; $N = 10$ hops, rates: $r = 0.05$ and $r = 1$ reqs/sec.

The exception is when lifetime is comparable to the propagation delay between the source and the requester, i.e. $T = 1$ sec (a path length of $N = 10$ means 100ms propagation delay). This causes a maximal achievable freshness of ≈ 0.9 for the $T = 1$ case. A less severe case is observed to be $T = 5$ sec, which shows cache retrievals when request rate is high ($r = 0.2, 1$), as high request rate triggers caching at routers.

2) Path Length: In Fig. 8, different path lengths (N) are considered, for higher ($r = 1$ reqs/sec) and a lower ($r = 0.05$ reqs/sec) request rate scenarios. When the request rate is lower, cache hit ratio shows more variations with respect to path length N. In this case, cache retrievals from the source are more often observed (hence higher $|hops|/N$) when path length is shorter (e.g. $N = 5$), as the communication cost is lower. Increase in request rate promotes caching at routers less for a higher path length (e.g. $N = 20$), as the communication cost is still relatively higher when N is larger. The changes with respect to lifetime T are similar to those shown in Fig. 7 and explained in Section VI-C1.

3) Communication Coefficient: A higher communication coefficient α promotes caching more, hence a higher cache hit ratio, lower average freshness of retrieved items, and lower average hop distance to retrieval location, as illustrated in Fig. 9. The figure also shows higher caching effects when request rate is higher. The changes with respect to lifetime T are similar to those shown in Fig. 7 and explained in Section VI-C1.

Simulation results shown in Fig. 9 demonstrate that caching gains are not clearly observable for $\alpha = 0.3$, but evidently seen for $\alpha = 0.4, 0.5, 0.7$. This is related with Eqn. 28, which describes routers' strategy of updating their caching probability variables. Basically, if the first condition in this equation (i.e. $\frac{1}{T}(1-\alpha)\overline{CA} \geq \frac{\alpha}{N}(\overline{h}(i+1)+1)$) consistently holds, then P_c is to be consistently decremented; if initially $P_c > 0$, then it eventually reaches $P_c = 0$; in simulations $P_c = 0$ initially, hence no caching gains are observed for low $\alpha < 0.4$. The choice of α values in Fig. 9 reasonably demonstrates this effect.

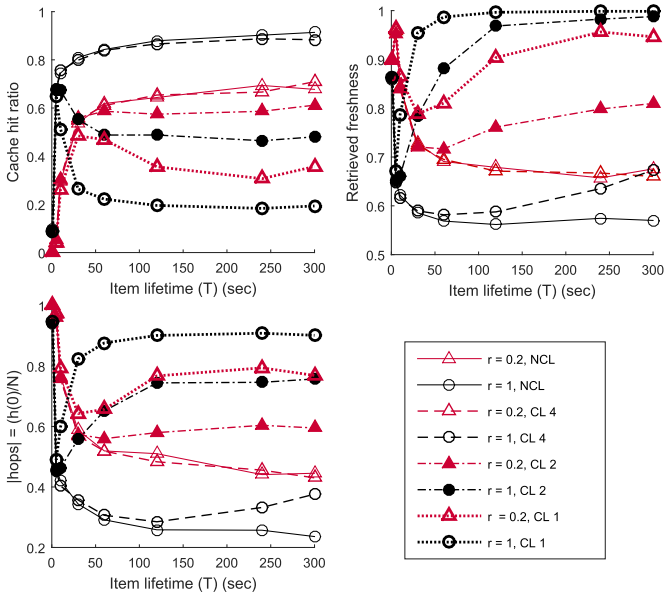

Fig. 10. Cache Limits: Effect of item lifetime T and request rate r; $\alpha = 0.5$, $N = 10$ hops.

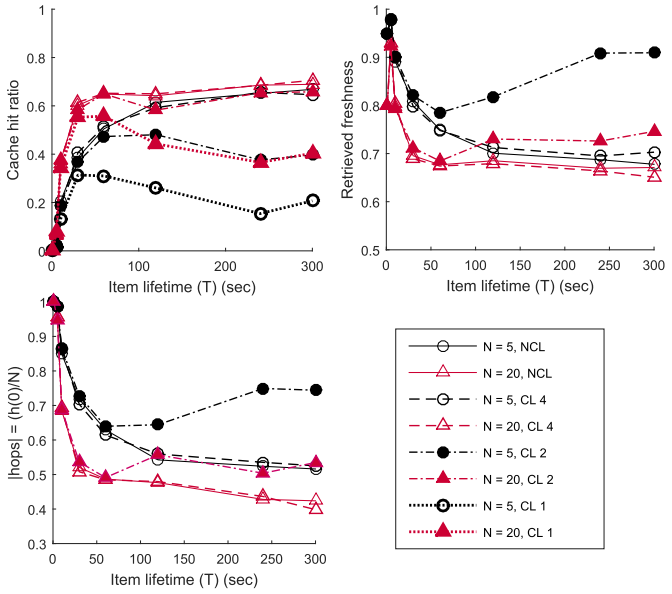

Fig. 11. Cache Limits: Effect of item lifetime T and path length N; $\alpha = 0.5$, $r = 0.2$ reqs/sec.

D. Scenario With Limited Cache Space

In this section, the effect of cache space on caching performance is evaluated. Three scenarios are considered, named CL 4 (Cache Limit of 4 items), CL 2, and CL 1; hence the number of data items that a router's cache can hold at a time is 4, 2, and 1, respectively. Recall that there can be at most one item for each content at a time, and there are 8 contents with different lifetimes at the source. The case in which router cache space is unlimited is named as NCL (No Cache Limit). In the CL (Cache Limit) scenarios, besides router cache spaces, other simulation settings are unchanged (as compared to NCL), and are as outlined in Section VI-A. Results are demonstrated in Figs. 10, 11, 12.

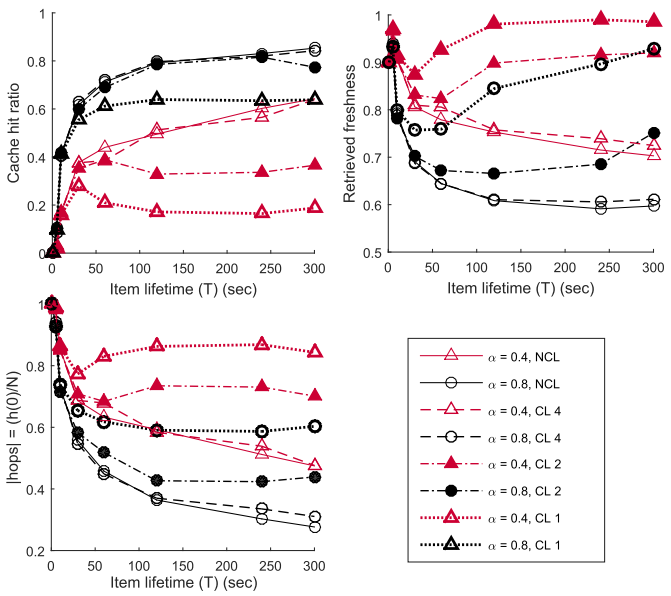

Fig. 12. Cache Limits: Effect of item lifetime T and communication coefficient α; $N = 10$ hops, $r = 0.2$ reqs/sec.

A common observation in these results is that with the decrease in cache space, items are no longer cached in the CL cases as much as they are in the NCL case. This causes them to be retrieved from the source node, effectively increasing the average normalized hop distance to the retrieval location ($|hops|$) and also increasing the average freshness of retrieved items (since those items that are retrieved from the source are much fresher than those taken from caches). The effect on cache hit ratio is the reverse: with smaller cache space, cache hits are less frequent.

Another observation is that the effect of cache space is not the same in all cases of the item lifetime T. The main reason is that the cache hit ratio of items with larger T are normally higher, as such items stay longer in cache after being cached. Inevitably, this has a negative impact on the average freshness: items with large T have less freshness; this can be observed for the NCL case in the figures. However, with smaller cache space, this is no longer the case, as now the items with larger lifetime T are removed from cache more often (as they tend to stay in cache longer), hence their cache hit ratio gets a hit from a limited cache space.

In all figures, the initial (starting from the smallest T and as T increases) increase in cache hit ratio (and the decreases in freshness and hop distance) is due to the fact that small lifetime items have generally a low "caching performance", i.e. low cache hit ratio, high freshness, and high $|hops|$.

What is most striking in these figures is that, when cache space is limited, for a given request rate and a cache limit condition, there is a "best" item lifetime T that strikes the balance between item freshness and the pair of "cache hit ratio and hop distance" (please see the dipping point for $|hops|$ and retrieved freshness, which is co-located (same T) with the peak observed for cache hit ratio. This trend is more magnified when "caching conditions" get worse, i.e. a lower request rate, or a lower alpha (recall that a high communication coefficient

causes more emphasis on communication costs), or a higher path length N. When cache spaces are limited, such factors play a bigger role.

Note that when the cache is unlimited and (i) request rate is high ($r = 1$) (Fig. 10), or (ii) path length N is low (Fig. 11) or (iii) α is high (Fig. 12), then the caching gains are highest across a wide range of item lifetimes.

VII. Conclusion

The benefit of caching in-transient multimedia files is clear; yet, to date, there has been no study or development effort that evaluates the potential of in-network caching techniques to be applied to transient data items. The difference to in-transient multimedia files is that transient items have a certain lifetime, and become less *fresh* over time before completely expiring. This paper is the first attempt to understand if caching transient data items in Internet content routers (or any generic multihop network) is beneficial, and provides a quantifiable way to measure caching gains. By defining the freshness of a transient data item and considering the trade-off between item freshness and multihop communication costs, a model is proposed for content routers to adapt their caching strategy.

Caching benefits are verified through packet-level simulations using the ns-3 simulator on a multihop path between a requester and a data source. Results demonstrate that the proposed caching strategy adapts to the two key system variables: a data item's lifetime and received rate of requests for the data item. In doing so, it is shown that caching transient items is indeed possible, making it possible to fetch items from routers directly with tolerable reduction in item freshness, depending on item lifetime.

Acknowledgment

The authors would like to acknowledge the support of the University of Surrey 5GIC (http://www.surrey.ac.uk/5gic) members. Special thanks to Dr. V. Vassilakis for his contribution to the initial stages of the simulator development.

References

[1] J. Li *et al.*, "Delivering Internet-of-Things services in MobilityFirst future Internet architecture," in *Proc. 3rd Int. Conf. Internet Things (IOT)*, Oct. 2012, pp. 31–38.

[2] G. Wu, S. Talwar, K. Johnsson, N. Himayat, and K. D. Johnson, "M2M: From mobile to embedded Internet," *IEEE Commun. Mag.*, vol. 49, no. 4, pp. 36–43, Apr. 2011.

[3] M. Z. Shafiq, L. Ji, A. X. Liu, J. Pang, and J. Wang, "Large-scale measurement and characterization of cellular machine-to-machine traffic," *IEEE/ACM Trans. Netw.*, vol. 21, no. 6, pp. 1960–1973, Dec. 2013.

[4] Z. Shelby, "Embedded Web services," *IEEE Wireless Commun.*, vol. 17, no. 6, pp. 52–57, Dec. 2010.

[5] J. Song, A. Kunz, M. Schmidt, and P. Szczytowski, "Connecting and managing M2M devices in the future Internet," *Mobile Netw. Appl.*, vol. 19, no. 1, pp. 4–17, Feb. 2014.

[6] J. Ni and D. H. K. Tsang, "Large-scale cooperative caching and application-level multicast in multimedia content delivery networks," *IEEE Commun. Mag.*, vol. 43, no. 5, pp. 98–105, May 2005.

[7] M. Diallo, S. Fdida, V. Sourlas, P. Flegkas, and L. Tassiulas, "Leveraging caching for Internet-scale content-based publish/subscribe networks," in *Proc. Int. Conf. Commun. (ICC)*, Jun. 2011, pp. 1–5.

[8] L. Dong, H. Liu, Y. Zhang, S. Paul, and D. Raychaudhuri, "On the cache-and-forward network architecture," in *Proc. Int. Conf. Commun. (ICC)*, Jun. 2009, pp. 1–5.

[9] V. Jacobson *et al.*, "Networking named content," in *Proc. CoNEXT*, 2009, pp. 1–12.

[10] S. Borst, V. Gupta, and A. Walid, "Distributed caching algorithms for content distribution networks," in *Proc. 29th IEEE Int. Conf. Comput. Commun. (INFOCOM)*, Mar. 2010, pp. 1478–1486.

[11] K. Fawaz and H. Artail, "DCIM: Distributed cache invalidation method for maintaining cache consistency in wireless mobile networks," *IEEE Trans. Mobile Comput.*, vol. 12, no. 4, pp. 680–693, Apr. 2013.

[12] E. J. Rosensweig, J. Kurose, and D. Towsley, "Approximate models for general cache networks," in *Proc. 29th IEEE Conf. Comput. Commun. (INFOCOM)*, 2010, pp. 1100–1108.

[13] M. Bhide *et al.*, "Adaptive push-pull: Disseminating dynamic Web data," *IEEE Trans. Comput.*, vol. 51, no. 6, pp. 652–668, Jun. 2002.

[14] J. Yin, L. Alvisi, M. Dahlin, and A. Iyengar, "Engineering Web cache consistency," *ACM Trans. Internet Technol.*, vol. 2, no. 3, pp. 224–259, Aug. 2002.

[15] J. Tadrous, A. A. Eryilmaz, and H. El Gamal, "Proactive content distribution for dynamic content," in *Proc. Int. Symp. Inf. Theory (ISIT)*, Jul. 2013, pp. 1232–1236.

[16] S. Zhu and C. V. Ravishankar, "Stochastic consistency, and scalable pull-based caching for erratic data stream sources," in *Proc. 30th Int. Conf. Very Large Data Bases (VLDB)*, vol. 30. Jun. 2004, pp. 192–203.

[17] W. Wang, S. De, R. Toenjes, E. Reetz, and K. Moessner, "A comprehensive ontology for knowledge representation in the Internet of Things," in *Proc. 11th Int. Conf. Trust, Secur. Privacy Comput. Commun. (TrustCom)*, Jun. 2012, pp. 1793–1798.

[18] Y. Liu *et al.*, "CitySee: Not only a wireless sensor network," *IEEE Netw.*, vol. 27, no. 5, pp. 42–47, Oct. 2013.

[19] H. Yang, Y. Qin, G. Feng, and H. Ci, "Online monitoring of geological CO_2 storage and leakage based on wireless sensor networks," *IEEE Sensors J.*, vol. 13, no. 2, pp. 556–562, Feb. 2013.

[20] D. Bartlett, W. Harthoorn, J. Hogan, M. Kehoe, and R. J. Schloss, "Enabling integrated city operations," *IBM J. Res. Develop.*, vol. 55, nos. 1–2, pp. 15:1–15:10, Mar. 2011.

[21] P. Bellavista, G. Cardone, A. Corradi, and L. Foschini, "Convergence of MANET and WSN in IoT urban scenarios," *IEEE Sensors J.*, vol. 13, no. 10, pp. 3558–3567, Oct. 2013.

[22] V. Tyagi, S. S. Kalyanaraman, and R. Krishnapuram, "Vehicular traffic density state estimation based on cumulative road acoustics," *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 3, pp. 1156–1166, Sep. 2012.

[23] V. Kostakos, T. Ojala, and T. Juntunen, "Traffic in the smart city: Exploring city-wide sensing for traffic control center augmentation," *IEEE Internet Comput.*, vol. 17, no. 6, pp. 22–29, Nov. 2013.

[24] *Information-Centric Networking Research Group (ICNRG)*, accessed on Jul. 15, 2015. [Online]. Available: https://irtf.org/icnrg/

[25] K. Katsaros *et al.*, "Information-centric networking for machine-to-machine data delivery: A case study in smart grid applications," *IEEE Netw.*, vol. 28, no. 3, pp. 58–64, May/Jun. 2014.

[26] W. K. Chai *et al.*, "An information-centric communication infrastructure for real-time state estimation of active distribution networks," *IEEE Trans. Smart Grid*, vol. 6, no. 4, pp. 2134–2146, Jul. 2015.

[27] S. Salinas, M. Li, P. Li, and Y. Fu, "Dynamic energy management for the smart grid with distributed energy resources," *IEEE Trans. Smart Grid*, vol. 4, no. 4, pp. 2139–2151, Dec. 2013.

[28] W. Kang, S. H. Son, and J. A. Stankovic, "Design, implementation, and evaluation of a QoS-aware real-time embedded database," *IEEE Trans. Comput.*, vol. 61, no. 1, pp. 45–59, Jan. 2012.

[29] K.-D. Kang, Y. Zhou, and J. Oh, "Estimating and enhancing real-time data service delays: Control-theoretic approaches," *IEEE Trans. Knowl. Data Eng.*, vol. 23, no. 4, pp. 554–567, Apr. 2011.

[30] S. Chen, N. B. Shroff, and P. Sinha, "Heterogeneous delay tolerant task scheduling and energy management in the smart grid with renewable energy," *IEEE J. Sel. Areas Commun.*, vol. 31, no. 7, pp. 1258–1267, Jul. 2013.

[31] Y. Xu and W. Wang, "Wireless mesh network in smart grid: Modeling and analysis for time critical communications," *IEEE Trans. Wireless Commun.*, vol. 12, no. 7, pp. 3360–3371, Jul. 2013.

[32] P. Castillejo, J. F. Martinez, J. Rodriguez-Molina, and A. Cuerva, "Integration of wearable devices in a wireless sensor network for an E-health application," *IEEE Wireless Commun.*, vol. 20, no. 4, pp. 38–49, Aug. 2013.

[33] D. B. Santana, Y. A. Zócalo, and R. L. Armentano, "Integrated e-health approach based on vascular ultrasound and pulse wave analysis for asymptomatic atherosclerosis detection and cardiovascular risk stratification in the community," *IEEE Trans. Inf. Technol. Biomed.*, vol. 16, no. 2, pp. 287–294, Mar. 2012.

[34] S. Dhar and U. Varshney, "Challenges and business models for mobile location-based services and advertising," *Commun. ACM*, vol. 54, no. 5, pp. 52–57, May 2011.
[35] W.-S. Li, O. Po, W.-P. Hsiung, K. S. Candan, and D. Agrawal, "Freshness-driven adaptive caching for dynamic content," in *Proc. Int. Conf. Database Syst. Adv. Appl.*, Mar. 2003, pp. 203–212.
[36] A. Labrinidis and N. Roussopoulos, "Exploring the tradeoff between performance and data freshness in database-driven Web servers," *VLDB J.*, vol. 13, no. 3, pp. 240–255, Sep. 2004.
[37] K. D. Kang, S. H. Son, and J. A. Stankovic, "Managing deadline miss ratio and sensor data freshness in real-time databases," *IEEE Trans. Knowl. Data Eng.*, vol. 16, no. 10, pp. 1200–1216, Oct. 2004.
[38] W. K. Chai, D. He, I. Psaras, and G. Pavlou, "Cache 'less for more' in information-centric networks (extended version)," *Comput. Commun.*, vol. 36, no. 7, pp. 758–770, Apr. 2013.
[39] S. Vural *et al.*, "In-network caching of Internet-of-Things data," in *Proc. Int. Conf. Commun.*, Jun. 2014, pp. 3185–3190.
[40] *Network Simulator 3 (ns3)*. accessed on Sep. 5, 2014. [Online]. Available: http://www.nsnam.org/
[41] E. Baccelli, C. Mehlis, O. Hahm, T. C. Schmidt, and M. Wählisch, "Information centric networking in the IoT: Experiments with NDN in the wild," in *Proc. 1st ACM Conf. Inf.-Centric Netw. (ICN)*, Sep. 2014, pp. 77–86.
[42] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman, "A survey of information-centric networking," *IEEE Commun. Mag.*, vol. 50, no. 7, pp. 26–36, Jul. 2012.
[43] G. Xylomenos *et al.*, "A survey of information-centric networking research," *IEEE Commun. Surveys Tut.*, vol. 16, no. 2, pp. 1024–1049, 2nd Quart. 2014.
[44] Z. Li, G. Simon, and A. Gravey, "Caching policies for in-network caching," in *Proc. Int. Conf. Comput. Commun. Netw. (ICCCN)*, Jul./Aug. 2012, pp. 1–7.
[45] E. J. Rosensweig and J. Kurose, "Breadcrumbs: Efficient, best-effort content location in cache networks," in *Proc. 28th IEEE Int. Conf. Comput. Commun. (INFOCOM)*, Apr. 2009, pp. 2631–2635.
[46] 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 *Proc. Innov. NGN, Future Netw. Services*, May 2008, pp. 367–374.
[47] L. Dong, D. Zhang, Y. Zhang, and D. Raychaudhuri, "Optimal caching with content broadcast in cache-and-forward networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2011, pp. 1–5.
[48] H. Xie, G. Shi, and P. Wang, "TECC: Towards collaborative in-network caching guided by traffic engineering," in *Proc. 31st IEEE Int. Conf. Comput. Commun. (INFOCOM)*, May 2012, pp. 2546–2550.
[49] I. Psaras, W. K. Chai, and G. Pavlou, "Probabilistic in-network caching for information-centric networks," in *Proc. 2nd Ed. ICN Workshop Inf.-Centric Netw.*, 2012, pp. 55–60.

Serdar Vural received the B.S. degree in electrical and electronics engineering from Boğaziçi University, Istanbul, Turkey, in 2003, and the M.S. and Ph.D. degrees in electrical and computer engineering from The Ohio State University, Columbus, OH, USA, in 2005 and 2007, respectively. He is currently a Research Fellow with the 5G Innovation Centre, Institute for Communication Systems, University of Surrey, Guildford, U.K. His current research interests include wireless sensor, *ad hoc*, mesh networks, Internet-of-Things, machine-to-machine communications, and future Internet.

Ning Wang (A'11–M'12) received the B.Eng. degree (Hons.) from the Changchun University of Science and Technology, Changchun, China, in 1996, the M.Eng. degree from Nanyang University, Singapore, in 2000, and the Ph.D. degree from the University of Surrey, Guildford, U.K., in 2004. He is currently a Reader (Associate Professor) with the 5G Innovation Centre, Institute for Communication Systems, University of Surrey. His current research interests include information centric networking, content and data caching management, network optimization techniques, and quality of service.

Pirabakaran Navaratnam received the B.Sc.Eng. degree in electronics and telecommunication engineering from the University of Moratuwa, Moratuwa, Sri Lanka, and the Ph.D. degree in mobile communications from the University of Surrey, Guildford, U.K. He is currently a Research Fellow with the 5G Innovation Centre, Institute for Communication Systems, University of Surrey, Guildford, U.K. He was the Technical Manager of the EU FP7 EXALTED Project on machine-to-machine (M2M) communications over LTE. He has been involved in several EU funded projects, such as eSENSE, SENSEI, EXALTED, and City-Pulse. His current research interests include resource management, M2M networking protocols, and future Internet architecture design.

Rahim Tafazolli (M'93–SM'08) has been a Professor of Mobile and Satellite Communications since 2000, the Director of the Institute for Communication Systems since 2010, and the Director of the 5G Innovation Centre, University of Surrey, Guildford, U.K., since 2012. He has authored or coauthored over 500 research publications, and holds 30 granted patents in the field of digital communications. He has been a member of the U.K. Smart Cities Forum and the IET Communications Policy Panel since 2013, a member of the U.K. Business, Innovation and Skills Advisory Working Group to the National Measurement Office for NPL Programs since 2012, and a member of the Innovate U.K. ICT Industry Advisory Board since 2014. He is a Fellow of the IET and WWRF.