Characterizing Interest Aggregation in Content-Centric Networks

Ali Dabirmoghaddam* Mostafa Dehghan† J. J. Garcia-Luna-Aceves*‡
*University of California Santa Cruz, †University of Massachusetts Amherst, ‡PARC
{alid, jj}@soe.ucsc.edu, mdehghan@cs.umass.edu

***Abstract*—The Named Data Networking (NDN) and Content-Centric Networking (CCN) architectures advocate Interest aggregation as a means to reduce end-to-end latency and bandwidth consumption. To enable these benefits, Interest aggregation must be realized through Pending Interest Tables (PIT) that grow in size at the rate of incoming Interests to an extent that may eventually defeat their original purpose. A thorough analysis is provided of the Interest aggregation mechanism using mathematical arguments backed by extensive discrete-event simulation results. We present a simple yet accurate analytical framework for characterizing Interest aggregation in a CCN router, and use our model to develop an iterative algorithm to analyze the benefits of Interest aggregation in a network of interconnected routers. Our findings reveal that, under realistic assumptions, an insignificant fraction of Interests in the system benefit from aggregation, compromising the effectiveness of using PITs as an integral component of Content-Centric Networks.**

I. Introduction

The fact that the content and not its location is what matters to end-users has given rise to many recent proposals classified under the generic name of Information-Centric Networking (ICN) [1]. Many ICN blueprints can be seen as *Interest-driven* communication models, where users ask for content by name through *Interest* packets. The most prominent examples of Interest-driven ICN (NDN [2] and CCNx [3]) are based on three main components. Routers maintain *Forwarding Information Bases* (FIB) listing routes to name prefixes, which enable routing to *names* rather than addresses. Given that all copies of the same content are equivalent, routers cache content opportunistically in their local *Content Store* (CS). In addition, each router maintains a *Pending Interest Table* (PIT) to suppress unnecessary Interests.

When a router receives an Interest that cannot be satisfied through its local CS, it creates a PIT entry and forwards the Interest if there is no entry for the same content name in its PIT. If a PIT entry already exists for the content name, the Interest is *aggregated* at the router and is not forwarded. The idea of Interest aggregation is hardly new. It has been implemented in Web caching architectures in the past, *e.g.*, Squid [4]—where it was referred to as *collapsed forwarding*—and commercially used on production content delivery networks since the early days of the Web.

The expectation of Interest aggregation in ICN architectures has been that network and server loads can be drastically reduced by suppressing similar Interests, and that end-to-end latencies can be reduced by integrating caching with Interest aggregation. These benefits, however, come at a non-trivial cost. Creating and maintaining the PIT is expensive, especially when performed at Internet scale. The routers used in the Internet backbone handle hundreds of thousands of packets every second. The proposed data structure must be fast enough when operating at its peak capacity to not act as a source of latency and overhead itself. Thus, considerable work has focused on optimization and scalability of the PIT (*e.g.*, see [5]–[8]).

We are not aware of any comprehensive analytical work characterizing the expected benefits of Interest aggregation. Some experimental efforts [9], [10] have been made in understanding the dynamics of the PIT size; however, important questions such as *what fraction of Interests are subject to aggregation under realistic conditions* and *whether or not that justifies the use of PITs* have remained unanswered to this date.

To answer these questions, Section II presents a simple yet powerful analytical toolbox for characterizing a CCN router with a CS and a PIT. We compute the cache hit probability at the CS, the Interest aggregation probability at the PIT, as well as the router response time at an object-level granularity. Section III uses these constructions for the analysis of a network of interconnected content routers. Through extensive event-driven simulations, Section IV demonstrates how accurately our proposed framework can predict the steady-state behavior of such a complex system. Furthermore, it shows how the model can be used to study the performance of a large network under realistic conditions, such as that of today's Internet, for which event-driven simulations are prohibitive.

Numerical evaluations of our model for large-scale systems reveal that only a small fraction of Interests may actually benefit from Interest aggregation in realistic settings. These benefits are highly dependent on the amount of caching budget available to the network. For example, a 5% cumulative aggregation percentage can be achieved using a per-node caching capacity of equal to 0.05% of the total number of objects in the system, while increasing the budget to just 0.5% reduces the aggregation percentage to below 1%. Even worse, our findings show that most Interest aggregation takes place closer to where the content is permanently stored—*i.e.*, near the producers deep in the core of the network—where aggregating Interests hardly makes sense anymore.

The insights from our modeling results lead to the necessary conclusion that Interest aggregation should *not* be an integral

ISBN 978-3-901882-83-8 © 2016 IFIP

component of future ICN architectures and Content-Centric Networks. On-path caching or edge caching provide all the benefits of reducing the number of Interests that request similar content, without the costs of maintaining PITs. In turn, if per-Interest forwarding state is not needed for other reasons, this realization makes Content-Centric Networking at Internet scale more feasible than the current NDN design, given that forwarding data structures (*e.g.*, CCN-DART [11], [12] and CCN-GRAM [13]) smaller and more efficient than PIT could be used for routing data back to the consumers.

II. CCN Router with Non-zero Download Delays

We develop a mathematical model to characterize a CCN router with a Content Store (CS) to enable caching functionality and a Pending Interest Table (PIT) that allows Interest aggregation. Unlike previous work [14]–[17], we assume that the content download delays are non-zero. Our derivations, hence, can be regarded as an extension to a highly accurate approximation of LRU cache introduced by Che *et al.* [15].

Consider a content router with a CS of capacity C implementing LRU replacement policy receiving Interests indexed (without loss of generality) in their decreasing order of popularity from 1 to N. In the rest of this paper, the terms CS and cache refer to the same concept and we shall use them interchangeably. Assume that Interests conform to the Independent Reference Model (IRM), *i.e.*, for every object, Interests inter-arrival times to the router are independent, identically distributed (i.i.d.) random variables. Fig. 1 illustrates Interests (shown as combs) as arriving at a content router over time. We focus on the Interests for an object i, which are highlighted in color. In Fig. 1, the red and green zones respectively specify time intervals when object i is absent or present in the CS. Upon receiving an Interest, if the CS contains a copy of object i, a *cache hit* occurs (see green combs); the Interest is immediately satisfied and a copy of that object is sent back to the requester. Otherwise, we say a *cache miss* occurs (see red combs).

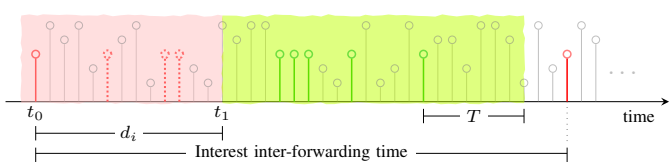

Fig. 1: Interests arriving at a content router over time

Let t_0 be the instant when the first cache miss for object i occurs at the CS. At that point, the content router creates a PIT entry for that object and forwards the Interest to another router/source (forwarded Interests shown as solid red combs). Let d_i be the random variable indicating the duration till a copy of the requested object is downloaded and stored in the CS, and mark that instant as $t_1 = t_0 + d_i$. We shall refer to d_i as the *download delay* of object i. Any subsequent Interest for object i during the interval (t_0, t_1) is aggregated at the content router due to the existence of a PIT entry under object i's name (aggregated Interests shown as dotted red combs).

A set of events take place at time t_1, namely the content router: (1) stores a copy of object i in the local CS; (2) removes the PIT entry for the corresponding Interest, and (3) forwards a copy of object i on all interfaces from which a request for it had been received. A copy of object i remains in the CS so long as the inter-arrival time of consecutive Interests for object i is smaller than T denoting the *characteristic time* of the cache introduced by Che *et al.* [15] (see Fig. 1). In essence, T is a random variable signifying the duration it takes until C distinct objects other than i are downloaded into the cache and object i is dismissed. For relatively large C and N and when the content download delays into the cache are zero, Fricker *et al.* [18] proved that T becomes deterministic. Later Dehghan *et al.* [19] proved this right even when download delays are non-zero.

The characteristic time T depends on the cache capacity C, the Interests arrival rate and the object popularity distribution and is computed according to

$$\mathbb{E}\left[\sum_{i=1}^{N} X_i\right] = C\,, \tag{1}$$

where X_i is the Bernoulli random variable indicating whether object i is present in the cache or not. Eq. (1) comes from the fact that the cache has the capacity for C objects. Note that $\mathbb{E}[X_i]$ equals the probability of object i being present in the cache, *i.e.*, the cache occupancy probability. With Poisson arrivals and thanks to the PASTA property, the cache occupancy probability equals the cache hit probability for any object i. Hence, we arrive at

$$\sum_{i=1}^{N} h_i = C\,. \tag{2}$$

where h_i denotes the cache hit probability of object i. We shall later use (2) as a constraint in computing the cache hit probability of individual objects.

A. Computing the Cache Hit Probability

Under the assumption that request inter-arrival times are independent exponential random variables, for a particular object i, the p.m.f. of exactly $n_i = k$ cache hits is the probability of the event that the first k Interest inter-arrival times are smaller than T, while the following Interest inter-arrival time is greater than T. This probability can be formalized by the geometric distribution $\mathbb{P}\left(n_i = k\right) = \left(1 - e^{-\lambda_i T}\right)^k e^{-\lambda_i T}$, and the expected number of cache hits is derived as

$$\mathbb{E}[n_i] = \sum_{k=0}^{\infty} k\,\mathbb{P}\left(n_i = k\right) = e^{\lambda_i T} - 1\,.$$

After forwarding a missed request for object i, if $\mathbb{E}[d_i]$ denotes the expected time to download a copy of i into the CS, the expected number of missed requests during this interval would be $\mathbb{E}[\bar{n}_i] = 1 + \lambda_i \mathbb{E}[d_i]$, of which one Interest is forwarded and the remaining $\lambda_i \mathbb{E}[d_i]$ are aggregated at the PIT. The expected total number of Interests received for object i during one such inter-forwarding cycle is then $\mathbb{E}[N_i] = \mathbb{E}[n_i] + \mathbb{E}[\bar{n}_i]$.

Consequently, the probability of a cache hit for object i is derived as

$$h_i = \frac{\mathbb{E}[n_i]}{\mathbb{E}[N_i]} = \frac{e^{\lambda_i T} - 1}{\lambda_i \mathbb{E}[d_i] + e^{\lambda_i T}} \,. \tag{3}$$

Eq. (3) can be regarded as an extension to the LRU approximation of Che *et al.* [15] where download delays can be non-zero. In fact, setting download delays to zero simplifies Eq. (3) to their renown form of $h_i = 1 - \exp(-\lambda_i T)$.

B. Computing the Interest Aggregation Probability

We now turn to computing the probability of Interest aggregation at the PIT. From our previous discussion, the expected number of aggregated requests during the download interval $\mathbb{E}[d_i]$ are $\mathbb{E}[\bar{n}_i] - 1 = \lambda_i \mathbb{E}[d_i]$. The probability of an Interest for object i being aggregated at the PIT is subsequently derived as

$$a_i = \frac{\mathbb{E}[\bar{n}_i] - 1}{\mathbb{E}[N_i]} = \frac{\lambda_i \mathbb{E}[d_i]}{\lambda_i \mathbb{E}[d_i] + e^{\lambda_i T}} \,. \tag{4}$$

Put differently, Eq. (4) states what fraction of Interests for object i arriving at the content router are aggregated in the long run.

C. Computing the Router Response Time

Another important measure in the analysis of interconnected routers when the download delays are non-zero is the router response time. Due to the potential existence of a PIT entry in the content router when an Interest for object i is received, the time it takes the router to satisfy that Interest can be any value from interval $(0, d_i]$.

We define the *pending time* of an Interest in the PIT as the time difference between the arrival of the Interest to the router and the subsequent moment when the Interest is served. With Poisson arrivals, Interest arrival times are uniformly distributed over $(0, d_i]$; hence, the sum W_i of pending time of Interests during a download interval d_i can be formulated as

$$W_i = d_i + \lambda_i \int_0^{d_i} (d_i - t)\, \mathrm{d}t = d_i(1 + 0.5\lambda_i d_i) \,.$$

We define the response time r_i of the content router for a particular object i as the expected pending time of Interests for that object which is readily derived as

$$r_i = \frac{\mathbb{E}[W_i]}{\mathbb{E}[N_i]} = \frac{\mathbb{E}[d_i(1 + 0.5\lambda_i d_i)]}{\lambda_i \mathbb{E}[d_i] + e^{\lambda_i T}} \,. \tag{5}$$

The router response time depends on the distribution of download delays, though knowledge of only the first two moments is sufficient.

III. An Algorithm for the Analysis of Hierarchical CCN Networks

We investigate how an interconnected network of CCN routers can be analyzed using the results from the previous section. Consider a hierarchy of routers as depicted in Fig. 2. Consumers are located at the bottom level where their requests for objects of interest are directed to the first level CCN routers (*i.e.*, ℓ_1 routers). An ℓ_1 router searches its local CS for a copy of the requested object and if failed, it forwards the request to the next level router (*i.e.*, the parent ℓ_2 router). This process is repeated on every cache miss and in the worst case, the requested object is downloaded directly from the producer at the top of the hierarchy storing permanent copies of all the objects in the system. On the reverse path back to the original requester, a copy of the object is stored in the CS of every CCN router it passes through.

For simplicity, we consider only a single producer in the network. The producer in our model can alternatively be conceived as a collection of several producers at the core of the network collapsed into one single entity. This single-source spanning-tree simplification of the network topology is standard in many studies of content delivery [20] and publish-subscribe networks [21].

There are two important challenges in the analysis of the foregoing structure. First, the Interest stream into a higher-level router (*i.e.*, all except ℓ_1 routers) is no longer a simple Poisson process, but an aggregate of miss streams from a number of lower-level routers. It is known, however, that the superposition of multiple streams tends toward Poisson as the load increases [22], [23]. We shall use this insight when extending the results of the previous section to the analysis of CCN networks by primarily focusing on trees of higher arity.

Secondly, chaining routers may cause circular dependencies in the computation of some router performance metrics. For instance, the cache hit probability in an ℓ_1 router depends on the download delay of the objects as mandated by Eq. (3). That delay is determined by the response time of the parent ℓ_2 router which in turn is a function of its input rate. The input into an ℓ_2 router itself partially relies on the miss stream of its descendant ℓ_1 router, and that is how a dependency cycle is formed. To overcome this hurdle, we present an iterative approach as outlined in Algorithm 1.

Procedure Analyze-CCN-Tree is proposed to compute the important router performance metrics we discussed in Section II for a hierarchical network structure such as the one portrayed in Fig. 2. The procedure analyzes a complete k-ary tree of height $L + 2$ in which consumers are at level 0; L layers of CCN routers are employed in the middle, and the producer is located at level $L + 1$ as the root of the tree. The available caching budget is provided by vector

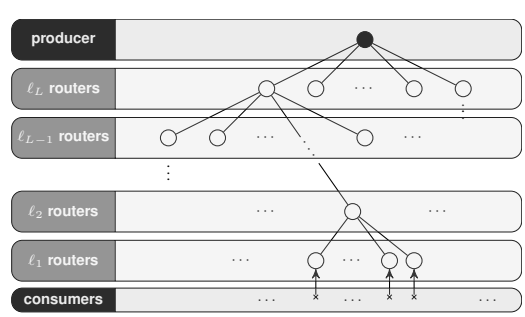

Fig. 2: A partial view of a hierarchy of interconnected routers

Algorithm 1 Method to characterize a hierarchical CCN represented by a complete tree as depicted in Fig. 2

Input: k: arity of the tree; L: number of tree levels; λ: consumer input rate to each first level router; δ: round-trip delay across each link; $\boldsymbol{C}$: vector of caching budget per node per layer; $\boldsymbol{q}$: probability vector reflecting the object popularity profile.

Output: T: characteristic time of caches at each level; $\boldsymbol{h}$: vector of cache hit probabilities at each level; $\boldsymbol{a}$: vector of aggregation probabilities at each level; $\boldsymbol{r}$: vector of router response times at each level; $\boldsymbol{m}$: vector of incoming Interest rates to each level.

1 **procedure** ANALYZE-CCN-TREE($k, L, \lambda, \delta, \boldsymbol{C}, \boldsymbol{q}$)
2 $\quad i \leftarrow 0$
3 $\quad$ **for** ℓ from 1 to L **do**
4 $\quad\quad \boldsymbol{r}_{\ell+1}^{(i)} \leftarrow \delta \times (L - \ell)$
5 $\quad$ **end for**
6 $\quad$ **while** not converged **do**
7 $\quad\quad i \leftarrow i + 1$
8 $\quad\quad$ **for** ℓ from 1 to L **do**
9 $\quad\quad\quad \boldsymbol{d}_{\ell}^{(i)} \leftarrow \delta + \boldsymbol{r}_{\ell+1}^{(i-1)}$
10 $\quad\quad$ **end for**
11 $\quad\quad \boldsymbol{m}_{1}^{(i)} \leftarrow \lambda \times \boldsymbol{q}$
12 $\quad\quad$ **for** ℓ from 1 to L **do**
13 $\quad\quad\quad T_{\ell}^{(i)} \leftarrow$ CHAR-TIME($\boldsymbol{m}_{\ell}^{(i)}$, $\boldsymbol{d}_{\ell}^{(i)}$, C_{ℓ})
14 $\quad\quad\quad \boldsymbol{h}_{\ell}^{(i)} \leftarrow$ HIT-PROB($\boldsymbol{m}_{\ell}^{(i)}$, $\boldsymbol{d}_{\ell}^{(i)}$, $T_{\ell}^{(i)}$)
15 $\quad\quad\quad \boldsymbol{a}_{\ell}^{(i)} \leftarrow$ AGG-PROB($\boldsymbol{m}_{\ell}^{(i)}$, $\boldsymbol{d}_{\ell}^{(i)}$, $T_{\ell}^{(i)}$)
16 $\quad\quad\quad \boldsymbol{r}_{\ell}^{(i)} \leftarrow$ RESP-TIME($\boldsymbol{m}_{\ell}^{(i)}$, $\boldsymbol{d}_{\ell}^{(i)}$, $T_{\ell}^{(i)}$)
17 $\quad\quad\quad \boldsymbol{m}_{\ell+1}^{(i)} \leftarrow$ MISS-RATE(k, $\boldsymbol{m}_{\ell}^{(i)}$, $\boldsymbol{h}_{\ell}^{(i)}$, $\boldsymbol{a}_{\ell}^{(i)}$)
18 $\quad\quad$ **end for**
19 $\quad$ **end while**
20 **end procedure**

$\boldsymbol{C}$ in which element C_ℓ indicates the allocated CS capacity for each of the routers at level ℓ. The initial rate at which Interests are produced by the consumers and fed into each ℓ_1 router is λ. The object popularity profile follows a Zipfian distribution as determined by the probability vector $\boldsymbol{q}$. Without loss of generality, in this paper we always rank objects in their decreasing order of popularity. As such, the normalized popularity of the nth ranked object is determined by the power-law $q(n) = n^{-\alpha}/\sum_{u=1}^{N} u^{-\alpha}$, where exponent $\alpha > 0$ is the parameter to the Zipf distribution. Finally, each link induces a round-trip delay of δ for transporting an individual content object. These parameters are inputs to Algorithm 1.

As pointed out, Algorithm 1 works in iterations to tackle circular dependencies. The superscript (i) used throughout the algorithm denotes the latest count of iterations. At the 0th iteration, *i.e.*, the initial phase, since all caches are empty and all requests are fulfilled directly by the producer, the router response times (denoted by $\boldsymbol{r}$) are simply set based on the hop-distance of routers from the root (lines 3–5). The notation $\boldsymbol{r}_{\ell+1}^{(i)}$ describes the response time of a $(\ell+1)^{\text{th}}$ level router computed at the ith iteration. Note that variables denoted in bold face are in fact vectors with values corresponding to individual objects in the system as ordered in popularity profile $\boldsymbol{q}$. Next, at any subsequent iteration:

1) Download delays for all levels are updated (lines 8–10) according to the heuristic that the delay for downloading files into the CS of an arbitrary router is equal to the response time of its parent router plus the round-trip delay of the link connecting them together. Assuming all objects are unit-sized, we can deduce that the average link delays are the same. In a hierarchical structure, thus, we can compute the download delays into a particular router by knowing the average link delays and the response time of the next level (*i.e.*, parent) router.
2) All performance measures discussed in Section II are computed/updated across all tree levels (lines 12–18).

Starting from the bottom working towards the top, at each tree level the measures are computed in the following order:

a) Procedure CHAR-TIME*:* is called at line 13 to compute the cache characteristic times by solving the following fixed-point equation for variable T:

$$\sum_{j=1}^{N} \frac{e^{m[j]T} - 1}{m[j]d[j] + e^{m[j]T}} = C\,, \tag{6}$$

where $m[j]$ and $d[j]$ are the jth elements in vectors $\boldsymbol{m}$ and $\boldsymbol{d}$, respectively denoting the input rate and download delay for object j at the corresponding router. Note that Eq. (6) is indeed the expanded form of (2) using (3).

b) Procedures HIT-PROB*,* AGG-PROB *and* RESP-TIME*:* are called at lines 14–16 to use the above computed characteristic time for computing the cache hit probability, PIT aggregation probability and response time of routers for individual objects according to Eqs. (3), (4) and (5), respectively.

c) Procedure MISS-RATE*:* is called at line 17 to compute the aggregate miss rate *into* the next level (*i.e.*, parent) router using the above computed hit- and aggregation probabilities according to the following relation:

$$\boldsymbol{m}_{\ell+1} = k \cdot \boldsymbol{m}_{\ell} \odot (\mathbf{1} - \boldsymbol{h}_{\ell}) \odot (\mathbf{1} - \boldsymbol{a}_{\ell})\,, \tag{7}$$

where $\odot$ signifies component-wise multiplication of corresponding vectors. In essence, Eq. (7) suggests that the input stream of a router at level-$(\ell+1)$ is the superposition of k miss streams from its descendant level-ℓ routers. The only exception are ℓ_1 routers whose input is directly provided by consumers according to line 11.

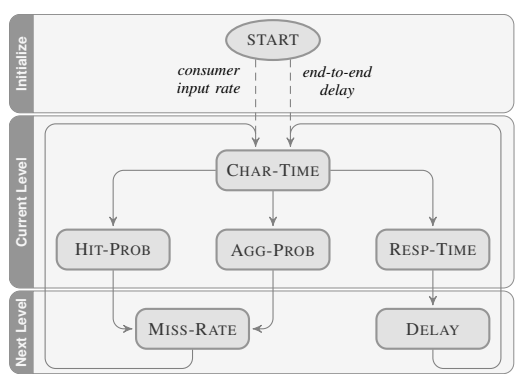

Fig. 3: Dependency among procedure calls in Algorithm 1.

To better understand the dependency between these procedure calls, the diagram in Fig. 3 provides a pictorial view

of their relationship. At the beginning, consumers' input rate and the end-to-end delay (between the consumers and the producer) are used to compute the cache characteristic time for all ℓ_1 routers. Cache hit- and PIT aggregation probabilities as well as router response times are then computed for ℓ_1 routers. Next these results are used to calculate the input rate and download delays for ℓ_2 routers. Then level 2 becomes the current level and a similar procedure is repeated for all remaining levels from the bottom to the top of the tree.

The computations in the middle and bottom boxes in Fig. 3 may be repeated in consecutive iterations as needed; the results from one iteration will be used in computing the next as the computed measures gradually converge to their steady-state values. In our numerical simulations—to be discussed next—we noticed that the first few iterations usually suffice to get an accuracy of better than 0.1% while no more than 10 iterations were needed in all cases studied (irrespective of the input size).

Implementing Algorithm 1 is straightforward in many off-the-shelf numerical computing environments. In our simulations, for solving Eq. (6), we leveraged `fsolve` function from MATLAB's Optimization Toolbox which uses trust-region methods [24] for solving systems of nonlinear equations. It is known [25] that trust-region methods take $\mathcal{O}(\epsilon^{-2})$ iterations to drive the norm of the gradient of the objective function below desired threshold ϵ. The time-complexity of Algorithm 1 is hence $\mathcal{O}(NL\,\epsilon^{-2})$.

IV. Performance Evaluation

We present simulation results to show how the proposed method can be used to accurately predict the complex behavior of a network of content routers. First, a detailed comparison of the numerical results of the presented model versus the results from extensive event-driven simulations in ndnSIM [26] is presented. Next, the results from our model are used to analyze more complex scenarios, such as networks with larger content base, which are far more cumbersome and time-consuming to study using conventional event-driven simulations.

We focus on two major strategies of cache allocation, namely *uniform caching* and *edge caching*. In the former, a fixed caching budget is evenly distributed across all content routers, whereas with the latter, the budget is entirely allocated to the routers at the edge of the network, *i.e.*, the ones directly serving the consumers. With edge caching, the upper level routers simply act as routers with no caching capability (that is, their CS size is set to zero). Yet they still perform Interest aggregation upon receiving Interests for which they have pending entries in their PITs.

A. Comparison of Model with Event-driven Simulations

We consider a tree of degree $k = 10$ and height $H = 5$ as the underlying topology, where $L = 3$ levels of content routers are used in the middle. The reason for using such a configuration is to keep the overall aggregate traffic pattern in the middle layers as close to being Poisson as possible, as discussed in Section III. Although the model was able to capture the overall trends in our experiments with trees of lower arity, we noticed that more accurate results are generally obtained when nodes have higher fan-in (*e.g.*, 10 or more). This assumption, however, is not unrealistic as some studies [27] of the actual Internet router-level topology have reported an average degree of more than 22 per router.

For the first set of experiments, we begin with a fairly small content catalog comprising only 100 objects. The reason for this choice is as follows. When performing event-driven simulations with a large content base, the system takes much longer to come to a steady-state; while growing worse with an increasing caching budget. In such case, a large number of requests must be used just to "warm-up" the system—hence, not to be used for collecting statistics—from the initial state where all caches are empty. Besides, because of the Zipfian nature of object popularity, a larger number of requests must be generated in total to ensure that objects at the long tail of the distribution also get a reasonable chance to appear in the generated request stream. Even with a content catalog of size 100 objects with a Zipf parameter of 1, we had to generate roughly 4 million requests—while disregarding the first half—to make sure all caches in all levels have their capacity almost fully utilized before collecting statistics.

For the foregoing set-up, in Fig. 4 we compare the aggregation probability for individual objects as predicted by model versus the results from extensive event-driven simulations. Curves in each plot represent the PIT aggregation probability as attained by *each* of the content routers at the corresponding level. Thanks to the symmetry of the topology, all routers at the same level share similar statistics. Graphs in the top row contrast uniform caching against edge caching employed for graphs at the bottom. In each row, the total caching budget (CB) increases from left to right. The model accurately predicts aggregation across various caching levels even at a fine object-scale resolution. Edge caching results in higher aggregation probability at higher levels. This behavior is expected because with edge caching naturally no cache hit may occur at higher levels in the tree. Therefore, many requests that would have hit those caches if a non-zero cache size were used will now end up being aggregated at PITs.

To obtain a more insightful view of Interest aggregation, graphs in Fig. 5 show the odds of a generic Interest (irrespective of the object popularity rank) getting aggregated at each level of the tree when the link delay gradually increases. It is clear that at a fixed Interest rate, an increased link delay generally improves the aggregation probability. However, larger cache sizes tend to offset some of these improvements, especially with uniform caching strategy.

Interest aggregation occurs at a higher probability at upper levels of the tree. This can be attributed to the higher input rate into those levels considering the fact that the aggregate miss stream from many lower level routers constitutes the input of their parent router. Results from Fig. 6 suggest that significant benefits are likely to accrue from Interest aggregation; however, this promising gain should be taken with a grain of salt due to the reasons discussed in the following.

First, the small object catalog consisting of only 100 objects

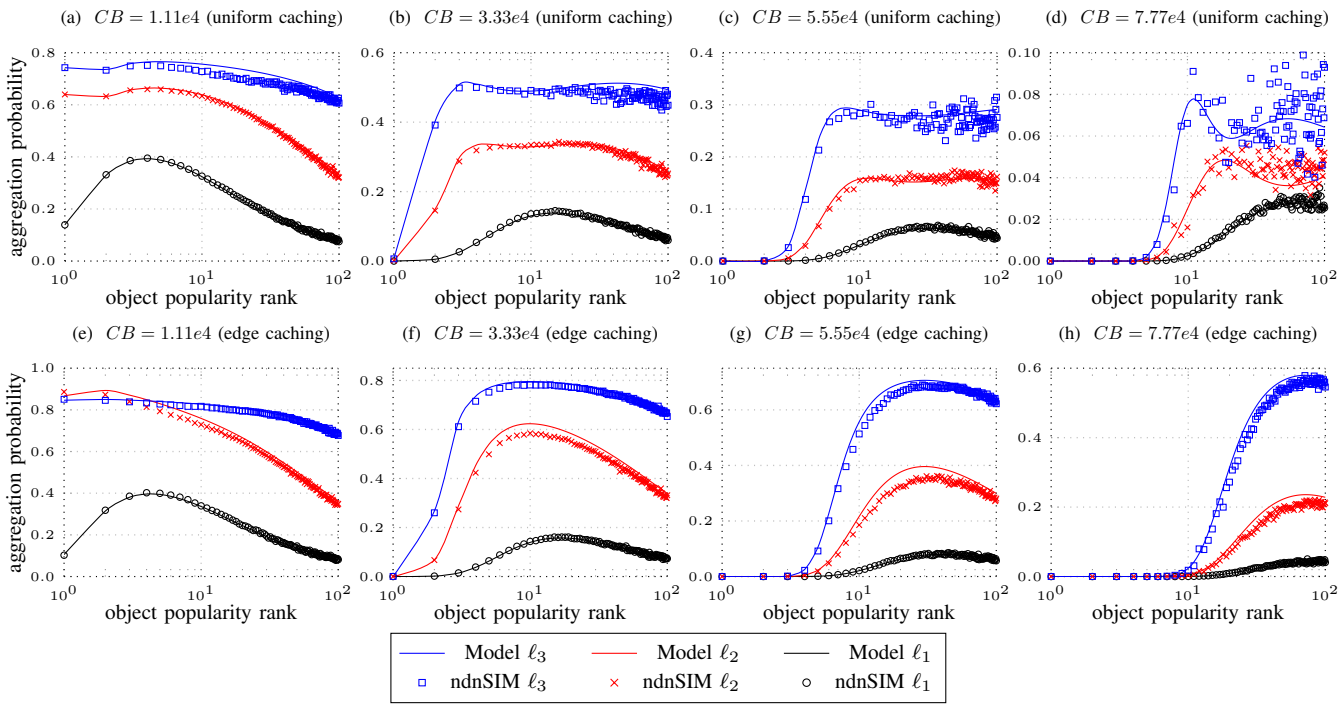

Fig. 4: A comparison of model versus event-driven simulations. Input rate into each edge router is 100 Interests/sec. Model predicts aggregation probability for individual objects fairly accurately across all levels of the tree topology.

naturally gives rise to a higher frequency of similar Interests arriving at the router, thereby an increased aggregation probability. Despite the event-driven simulation which turns out to be extremely tedious especially for a large number of objects, numerical simulations using the proposed model are practicable even on commodity hardware. Our numerical results in the next subsection confirm that the actual benefits of Interest aggregation are indeed much less in reality.

Secondly, the notion of aggregation probability itself may give a magnified image of the real benefits. In fact, aggregation probability at a certain level in the hierarchy indicates what fraction of Interests making it up to that level end up getting aggregated. Since the request stream observed by the higher level routers is a "filtered" version of the input stream to their descendants, it is clear that fewer Interests are received in total towards the top of the hierarchy. For this, we define a new measure called *aggregation percentage* that determines the percentage ratio of the count of aggregated Interests at a certain level (or at a particular router) over the total count of produced Interests in the whole system. Since every generated Interest can be aggregated at most once on its path towards the producer, aggregation percentage provides a more reasonable and unbiased measure, and we shall use that in our later assessments of aggregation benefits.

Given the foregoing remarks, we emphasize that the results demonstrated in Figs. 4 and 5 are particularly meant to verify the accuracy of the proposed analytical framework, and to provide a side-by-side comparison of how varying different parameters affects the relative odds of Interest aggregation. The true benefits of Interest aggregation are discussed in the following subsection, where more realistic input parameters are used.

B. Numerical Evaluations

Fig. 6 sheds light on the combined impact of download delay and input rate on the Interest aggregation probability. The symmetry of plots in Fig. 6 suggests that it is in fact the combination of the link delay and input rate which regulates the overall trend of aggregation probability. In fact, doubling the input rate for a fixed link delay has the same effect on the aggregation probability as keeping the input rate fixed and doubling the link delay. Therefore, we define *system load* as the product of these two quantities to build our next set of experiments on it. As a combined metric, system load does not identify a specific delay or input rate, rather defines an infinite range for these parameters. For example, a system load of 10 may imply an input rate of 100 Interests/sec with link delay of 0.1 seconds, or equivalently, an input rate of 500 Interests/sec with link delay of 0.02 seconds.

TABLE I: Table of default parameter values

Parameter	Symbol	Value
Tree height	H	5
Number of cache layers	L	3
Node degree	k	10
Total number of objects	N	140 million
Cache capacity per cache node	C	100,000 objects
Zipf exponent	α	0.8
Input rate into each edge cache	λ	100,000/sec
Link delay each way	d	15 milliseconds

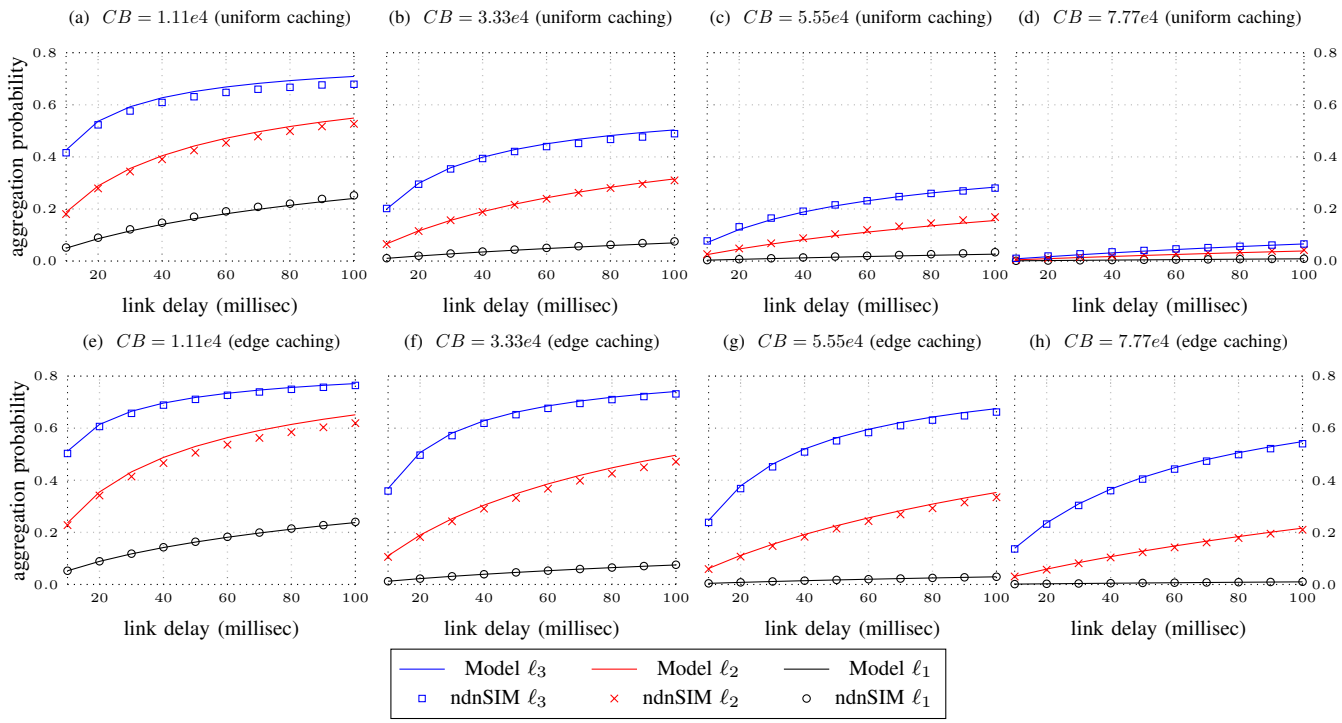

Fig. 5: Interest aggregation probability at various router levels as a function of link delay for increasing cache sizes (left to right) and different cache allocation strategies (top vs bottom rows). Input rate into each edge router is 100 Interests/sec.

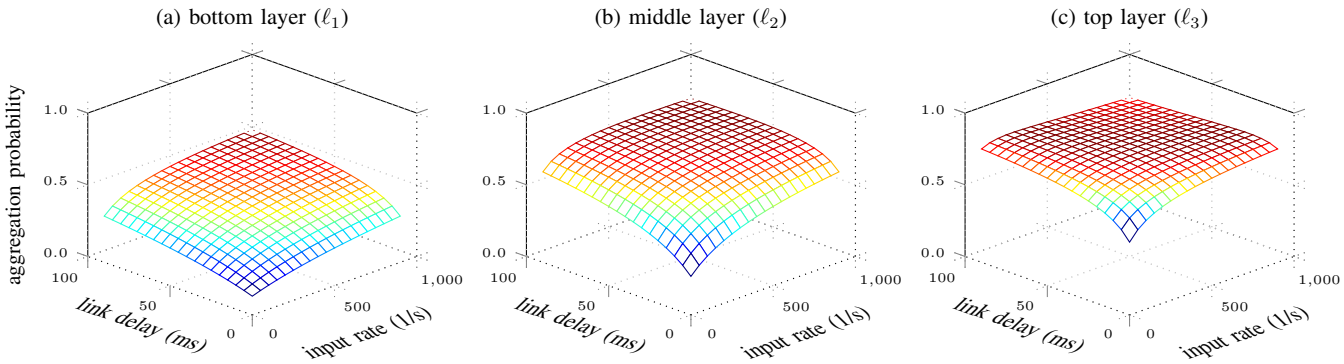

Fig. 6: The combined impact of link delay and input rate on aggregation probability. Increasing one factor has exactly the same effect on Interest aggregation as increasing the other.

In the experiments to be discussed next, we consider the same tree topology as in the previous subsection, with the general configurations listed in Table I (unless otherwise stated). The total number of objects considered, *i.e.*, 140 million, is an estimation of the total number of videos on YouTube in 2008 [28] and the Zipf parameter of 0.8 is taken from empirical studies [29], [30] of real content networks. The input rate of 100,000 Interests/sec and link delay of 15 milliseconds are also chosen such that the average generated traffic in the network is comparable with the load experienced by the Internet's backbone routers [31], [32].

Fig. 7 shows the probability of Interest aggregation at each tree level as a function of system load. Contrary to the results in Fig. 5, a side-by-side comparison of uniform- vs. edge-caching reveals that when the object catalog is large, there is no remarkable difference between these two cache allocation strategies. It is interesting that even with the highest system load of 3000, the maximum aggregation probability observed at the top most level is around 0.06. This almost 12-fold degradation compared to the previous results highlights the importance of the size of the object catalog in the overall odds of Interest aggregation. This rather surprising finding can be explained as follows. With the Zipf popularity distribution of objects, a highly popular object is requested frequently. Therefore, once such an object is downloaded into the CS, due to the frequent references to it, it stays there for a long time. Hence, Interests for that object mostly result in cache hits and are rarely aggregated. On the other hand, Interests

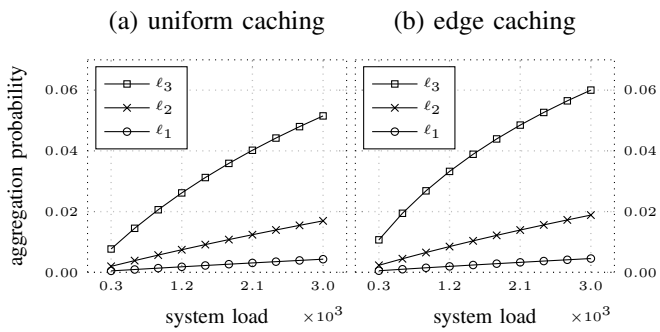

Fig. 7: Impact of system load on the aggregation probability.

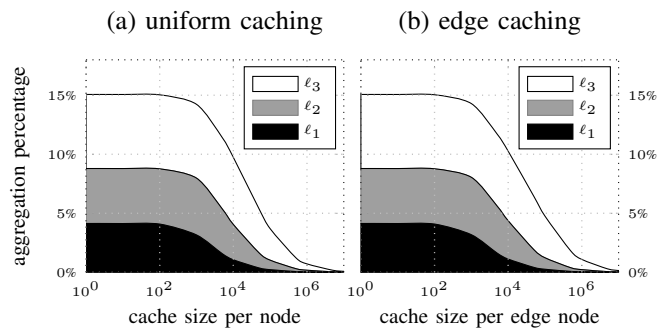

Fig. 9: Impact of cache size on overall aggregation percentage.

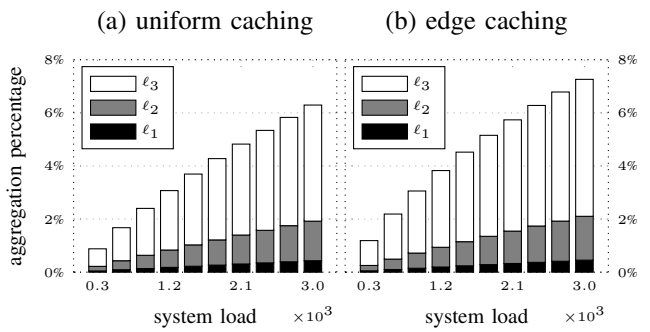

Fig. 8: Impact of system load on cumulative aggregation percentage.

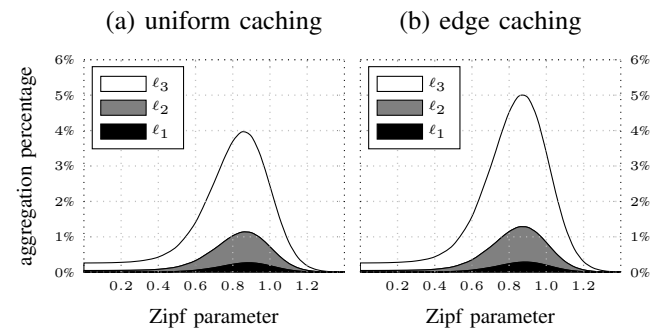

Fig. 10: Impact of popularity distribution on aggregation percentage.

for an unpopular object (in the long tail of the distribution) are received so sporadically over time that the odds of them co-occurring in the short time span when the router is awaiting the content are almost nil. As a result, in practice, Interest aggregation occurs only for a small fraction of Interests.

To make this argument even stronger, Fig. 8 shows the cumulative percentage of aggregated Interests in the system against an increasing system load. Evidently, the overall percentage of Interests being aggregated is less than 5% under a low to moderate load, and around 7% under heavy load. Note that for these results each cache node has capacity to store only 0.07% of the entire object catalog.

Increasing cache size further shrinks the benefit margin by improving the overall cache hit rates. As Fig. 9 suggests, with a small cache capacity, sizable gain (around 15% total) can be attained through Interest aggregation. However, this gain drastically decays as cache size per node increases. Eventually, with a cache size of 500,000 per node (*i.e.*, $< 0.4\%$ of the size of the content base), there is virtually no benefit in Interest aggregation. A secondary observation from the results in Fig. 9 is that with smaller cache size, all layers in the hierarchy contribute about the same in aggregation percentage; however, as more cache is added to the nodes, most Interest aggregations occur at the upper layers, while aggregation percentage at the edge approaches zero more rapidly.

Finally, Fig. 10 captures the impact of the object popularity distribution on the cumulative percentage of aggregated Interests. The non-monotonic trend of curves in Fig. 10 exhibits a diminishing returns type of effect. To explain this behavior, we note that with a Zipf popularity distribution, objects can heuristically be categorized into two groups, namely, an unpopular majority and a popular minority. The Zipf parameter (α) controls the relative size of each group as well as the skewness of the distribution. In fact, the larger the Zipf parameter, the smaller the proportion of the minority group, and the greater their popularity intensity. The latter signifies a higher access frequency to the objects in the popular group as α increases, hence a higher aggregation rate for them. However, as α increases and the proportion of the popular objects shrinks, the higher access frequency also results in higher hit rates, because a lot of those objects eventually find their way into the caches; therefore, subsequent requests for them no longer get aggregated. On the other hand, thanks to their diminishing popularity, the Interests for the majority group are also becoming so sparsely dispersed that the odds of finding a relevant entry for them at the PIT becomes negligible. For this, only a small fraction of Interests representing those fairly popular objects which may not have found a free spot in (limited-size) caches remain subject to aggregation. Further increasing the Zipf parameter shrinks down the size of the popular group gradually such that at some point, every one of them finds a permanent place in all caches. Thenceforth, the probability of aggregation becomes effectively zero.

From another viewpoint, Fig. 10 also provides suggestive evidence that even under a non-stationary content popularity distribution, no remarkable benefit can be anticipated from Interest aggregation. For example, when object references are temporally localized, a data object becomes highly popular over a certain duration of time, while its popularity gradually vanishes over time as some other data object becomes popular.

In that case, if the objects popularity is measured within smaller discrete time windows, each piece can independently be approximated with a Zipf distribution with a possibly different parameter. As Fig. 10 suggests, irrespective of how different the popularity profile looks like, only an insignificant number of Interests may be aggregated at various intervals; hence, the benefits of Interest aggregation would still remain minimal, with the maximum aggregation of less than 5% taking place around a Zipf parameter of 0.9.

V. Final Remarks and Conclusions

We presented the first analytical treatment of Interest aggregation in Content-Centric Networks using a simple yet accurate model where content download delays into the routers are non-zero. Based on our model, we introduced an iterative algorithm for analyzing a hierarchical network of content routers in terms of CS hit- and PIT aggregation probabilities and router response times. This method enables the evaluation of large-scale hierarchical caching structures, such as that of an ICN at Internet scale, with high accuracy and low computational cost for which discrete-event simulations are entirely impractical due to high processing and time demands.

Our numerical evaluations of a network of caches under realistic assumptions revealed that: (1) even with very small caching budgets, less than 5% of total Interests on average are subject to aggregation; (2) increasing caching budgets rapidly diminishes the benefits of Interest aggregation; (3) most aggregations take place closer to the producers, negating the expected benefits of reducing latency and bandwidth utilization desired from aggregation; and (4) aggregation gains are almost invariant to the choice of cache allocation strategy (*i.e.*, edge- vs. uniform-caching). Together, these observations imply that Interest aggregation should only be an optional mechanism in Content-Centric Networking. Furthermore, if per-Interest forwarding state is not needed for other purposes, the state-full forwarding plane of NDN (realized through PITs) can effectively be replaced with more efficient mechanisms, such as CCN-DART [11], [12] and CCN-GRAM [13], in which forwarding state is stored only per route or per destination while providing similar end-to-end content delivery latencies.

Our model relies on the assumption that input streams conform to the independent reference model, which need not be true in reality. However, the simulation results in [12], [13] indicate that in-network caching makes Interest aggregation unnecessary even with spatio-temporal locality of Interests.

Acknowledgments

This work was supported in part by the Jack Baskin Chair of Computer Engineering at UCSC, NSF grant CNS-1413998, and MURI ARO grant W911NF-12-10385.

References

[1] B. Ahlgren *et al.* , "A Survey of Information-Centric Networking," *IEEE Commun. Mag.*, vol. 50, no. 7, pp. 26–36, 2012.
[2] L. Zhang *et al.* , "Named Data Networking," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, pp. 66–73, 2014.
[3] [Online]. Available: http://www.ccnx.org/
[4] [Online]. Available: http://www.squid-cache.org
[5] H. Dai *et al.* , "On Pending Interest Table in Named Data Networking," in *Proc. ACM/IEEE ANCS*, Austin, USA, 2012, pp. 211–222.
[6] Y. Wang *et al.* , "Scalable Name Lookup in NDN Using Effective Name Component Encoding," in *Proc. IEEE ICDCS*, Macau, China, 2012, pp. 688–697.
[7] M. Varvello *et al.* , "On the Design and Implementation of a Wire-Speed Pending Interest Table," in *Proc. IEEE INFOCOM Workshops*, Turin, Italy, 2013, pp. 369–374.
[8] H. Yuan and P. Crowley, "Scalable Pending Interest Table Design: From Principles to Practice," in *Proc. IEEE INFOCOM*, Toronto, Canada, 2014, pp. 2049–2057.
[9] M. Virgilio *et al.* , "PIT Overload Analysis in Content Centric Networks," in *Proc. ACM SIGCOMM Workshop on ICN*, Hong Kong, China, 2013, pp. 67–72.
[10] A. Abu *et al.* , "Interest Packets Retransmission in Lossy CCN Networks and Its Impact on Network Performance," in *Proc. ACM ICN*, Paris, France, 2014, pp. 167–176.
[11] J. Garcia-Luna-Aceves, "A More Scalable Approach to Content Centric Networking," in *Proc. ICCCN*, Las Vegas, USA, 2015, pp. 1–8.
[12] J. Garcia-Luna-Aceves and M. Mirzazad-Barijough, "A Light-Weight Forwarding Plane for Content Centric Networks," in *Proc. IEEE ICNC*, Kauai, USA, 2016.
[13] ——, "Content-Centric Networking Using Anonymous Datagrams," in *Proc. IFIP Networking*, Vienna, Austria, 2016.
[14] A. Dan and D. Towsley, "An Approximate Analysis of the LRU and FIFO Buffer Replacement Schemes," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 18, no. 1, pp. 143–152, 1990.
[15] H. Che *et al.* , "Hierarchical Web Caching Systems: Modeling, Design and Experimental Results," *IEEE J. Sel. Areas Commun.*, vol. 20, no. 7, pp. 1305–1314, 2002.
[16] S. Ioannidis and P. Marbach, "On the Design of Hybrid Peer-to-Peer Systems," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 36, no. 1, pp. 157–168, 2008.
[17] E. Rosensweig *et al.* , "On the Steady-State of Cache Networks," in *Proc. IEEE INFOCOM*, Turin, Italy, 2013, pp. 863–871.
[18] C. Fricker *et al.* , "A Versatile and Accurate Approximation for LRU Cache Performance," in *Proc. ITC*, Krakow, Poland, 2012, pp. 1–8.
[19] M. Dehghan *et al.* , "On the Analysis of Caches with Pending Interest Tables," in *Proc. ACM ICN*, San Francisco, USA, 2015, pp. 69–78.
[20] J. Ni and D. Tsang, "Large-Scale Cooperative Caching and Application-Level Multicast in Multimedia Content Delivery Networks," *IEEE Commun. Mag.*, vol. 43, no. 5, pp. 98–105, 2005.
[21] R. Chand and P. Felber, "XNET: A Reliable Content-Based Publish/Subscribe System," in *Proc. IEEE SRDS*, Florianpolis, Brazil, 2004, pp. 264–273.
[22] J. Cao *et al.* , "Internet Traffic Tends Toward Poisson and Independent as the Load Increases," in *Lect. Notes. Stat.* Springer New York, 2003, vol. 171, pp. 83–109.
[23] E. Cinlar and R. Agnew, "On the Superposition of Point Processes," *J. Roy. Stat. Soc. B Met.*, vol. 30, no. 3, pp. 576–581, 1968.
[24] A. Conn *et al.* , *Trust Region Methods*, ser. MPS/SIAM S. Optimizat. Philadelphia, USA: SIAM, 2000.
[25] S. Gratton *et al.* , "Recursive Trust-Region Methods for Multiscale Nonlinear Optimization," *SIAM J. Optimiz.*, vol. 19, no. 1, pp. 414–444, 2008.
[26] A. Afanasyev *et al.* , "ndnSIM: NDN Simulator for NS-3," NDN, Technical Report NDN-0005, 2012. [Online]. Available: http://named-data.net/techreports.html
[27] "Internet Topology at Router- and AS-levels," accessed: 2015-07-17. [Online]. Available: http://www.caida.org/research/topology/generator
[28] A. Russakovskii, "How to Find Out the Number of Videos on Youtube," 2008, accessed: 2015-07-22. [Online]. Available: http://beerpla.net/2008/08/14/how-to-find-out-the-number-of-videos-on-youtube/
[29] A. Mahanti *et al.* , "Traffic Analysis of a Web Proxy Caching Hierarchy," *IEEE Netw.*, vol. 14, no. 3, pp. 16–23, 2000.
[30] C. Fricker *et al.* , "Impact of Traffic Mix on Caching Performance in a Content-Centric Network," in *Proc. IEEE INFOCOM Workshops*, Orlando, USA, 2012, pp. 310–315.
[31] J. Cowie *et al.* , "Modeling the Global Internet," *Comput. Sci. Eng.*, vol. 1, no. 1, pp. 42–50, 1999.
[32] C. Fraleigh, "Provisioning Internet Backbone Networks to Support Latency Sensitive Applications," Ph.D. dissertation, Stanford University, Stanford, USA, 2002.