When AIMD meets ICN: a bandwidth sharing perspective

Damien Saucez, Ilaria Cianci, Luigi Alfredo Grieco, Chadi Barakat

When AIMD meets ICN: a bandwidth sharing perspective

Damien Saucez
Inria
Sophia Antipolis, France
Email: damien.saucez@inria.fr

Ilaria Cianci
DEI - Politecnico di Bari
Bari, Italy
Email: i.cianci@poliba.it

Luigi Alfredo Grieco
DEI - Politecnico di Bari
Bari, Italy
Email: a.grieco@poliba.it

Chadi Barakat
Inria
Sophia Antipolis, France
Email: chadi.barakat@inria.fr

***Abstract*—Information-centric networking (ICN) leverages content demand redundancy and proposes in-network caching to reduce network and servers load and to improve quality of experience. In this paper, we study the interaction between in-network caching of ICN and Additive Increase Multiplicative Decrease (AIMD) end-to-end congestion control with a focus on how bandwidth is shared, as a function of content popularity and cache provisioning. As caching shortens AIMD feedback loop, the download rate of AIMD is impacted. Supported by an analytical model based on Discriminatory Processor Sharing and real experiments, we observe that popular contents benefit from caching and realize a shorter download time at the expense of unpopular contents, which see their download time inflated by a factor bounded by $\frac{1}{1-\rho}$, where ρ is the network load. This bias can be removed by redefining congestion control to be delay independent or by over-provisioning link capacity at the edge so that to compensate for the greediness of popular contents.**

I. Introduction

Internet usage has undergone an important shift during the last years. We have moved from an Internet designed to connect us to well defined hosts into an Internet mainly used to publish and retrieve contents, without caring about the location contents come from [17]. Nowadays, content dissemination such as file sharing and media streaming prevails in the Internet [18]. This new *information-centric* context poses new challenges on the current architecture, which is not able to handle them in an effective way. These challenges are currently tackled by application-layer solutions such as Content Delivery Networks (CDN), but the Internet architecture eventually has to evolve to provide inherent support of content distribution if we want efficient and seamless interactions between services and applications. The so-called *Information-Centric Networking* (ICN) approach has consequently become the leading rationale of many relevant proposals under the Future Internet umbrella [25].

In the proposed ICN architectures [8], contents are independent entities that users can retrieve without being aware of the locations of service providers. As a consequence, contents can be replicated across different ICN nodes thanks to distributed in-network caching, hence shifting the network traffic to the edge. Popular contents are cached close to the network edge with a high probability and statistically enjoy a short network delay, whereas requests for non-popular contents often result in cache misses, thus requiring to fetch them at their original servers with a longer network delay than popular contents. This mismatch of the network delay caused by in-network caching interplays with the congestion control deployed at end users to regulate the rate at which they download contents and to avoid network congestion. In this paper, we focus on this interplay and try to provide an answer to the following question: *Who wins and who looses with ICN in terms of download time?* For sake of clarity, we consider NDN (also called CCN [17]) as basis for this work even though the proposed investigation can be applied to any architecture presenting similar in-network caching features, even CDN.

Note that in the current Internet architecture based on TCP/IP, the difference in network delay is also a reality because on one hand of the widespread of servers and on another hand of CDNs that bring contents close to clients. We believe though that the actual difference in network delays is not necessarily correlated with popularity, hence reducing the bias against non-popular contents. We expect the correlation between delay and popularity, and hence the bias against non-popular contents, to be more apparent with ICN deployments.

Background and motivation. In NDN [17], contents (e.g., files or videos) are divided into chunks that users retrieve by issuing Interest packets. When a node that disposes of the requested chunk receives an Interest packet, it replies with a Data packet containing the chunk that is routed along the reverse path traced by the Interest packet. On their way back to the user, chunks can be cached by intermediate nodes reducing so network congestion and servers load [21].

To control the rate at which Interest packets are sent, several congestion control protocols have been proposed for NDN, such as in [5], [6], [23], [7], [22]. All these protocols rely (in different ways) on a window-based *Additive Increase Multiplicative Decrease* (AIMD) algorithm as in TCP [10]. The AIMD algorithm adjusts its window according to network delay and packet losses (i.e., in AIMD the window is increased by a constant value every round-trip delay) and is then sensitive to in-network caching, which shortens round-trip delays and makes faster the congestion window increase.

*The authors would like to thank Florin Coras, Luigi Iannone, and Xavier Misseri for their fruitful comments. This work is supported by the PON project RES NOVAE funded by the Italian MIUR and by the European Union (European Social Fund) and the EIT KIC ICT-Labs Networks for Future Media Distribution activity

This interaction has been investigated in prior work from the viewpoint of bandwidth sharing within NDN. Carofiglio et al., in [5], show for example that AIMD can guarantee max-min fair bandwidth sharing within NDN. Our work studies this interaction from the viewpoint of an end-to-end AIMD based architecture not caching contents in the core, and tries to assess if the introduction of in-network caching, combined with AIMD, introduces any bias against some classes of content. The steady-state model that considers a set of long-lived downloads bottlenecked at the same link studied in [24] as shown an instantaneous bias against non-popular contents. Unfortunately, such model does not provide time-average performance indicators for each class and does not explore the effects of the peculiar dynamics of Internet traffic due to the realistic finite-volume downloads and the stochastic nature of the content request process.

Summary of contributions. To assess the impact of AIMD and NDN on network resource sharing, we use *Discriminatory Processor Sharing* (DPS) arguments as in [12] by tailoring them to the peculiar facets of the NDN architecture when jointly deployed with the AIMD congestion control. We devise a general mathematical model for AIMD performance in NDN and derive insightful results for a chain topology with *equal capacity* links, while changing the configuration of in-network caches along this chain. The model equally applies to links of different capacities and to more general network topologies as long as all downloads share the same bottleneck (i.e., between the client and the first cache). We finally discuss the implications on the performance of AIMD when the edge is over-provisioned with respect to the other links over the path. Our theoretical results are validated against real experiments using *CCN-Java Opensource Kit EmulatoR* (CCN-Joker) [11] that we extended to support AIMD congestion control scheme. Our contributions can be summarized as follows:

- We provide implicit expressions for the mean download time with NDN for different popularity classes and normalize them to their counterparts in absence of caching. These expressions show that the introduction of NDN has a negligible impact on the download time at low network load. However, when network load increases, the download time of popular contents improves, at the expense of non-popular contents. The bias strongly depends on the volume of cached content and how close it is to the client. To give a flavor of the results, in the extreme case of all popular contents cached close to the requesters, the bias against the download time of non-popular contents scale as $1/(1-\rho)$, where ρ is the network load.
- We provide an implementation of window-based AIMD congestion control within CCN-Joker [11], which mimics the finest details of the algorithms of TCP. The only difference is that congestion is only detected by timeouts given the impossibility of applying the fast retransmit algorithm of TCP (i.e., congestion detection by 3 duplicate ACKs) as packets reordering is an inherent property of NDN.
- We identify the roots for the bias: it comes from the AIMD algorithm itself which is sensitive to the network delay, thus making popular contents greedier in their tussle for bandwidth. One possible solution is to redesign AIMD to make its mean performance independent of network delay (i.e., similar to what has been proposed in the literature for TCP over long-delay paths). Another solution is to over-provision the bottleneck so as to absorb the greediness of popular downloads. We present rules of thumb for this over-provisioning, which finally turns out to be few percents of the initial bottleneck capacity.

Outline. In Sec. II, we present our DPS-based general model and provide implicit expressions for the mean download time. In Sec. III, we refine these expressions for the particular case of two classes of popularity. In Sec. IV, we follow up with numerical results for the general case of M classes of contents, while Sec. V describes our AIMD implementation in CCN-Joker and presents experimental results that support our theoretical findings. We discuss in Sec. VI the roots of the observed bias and propose possible solutions, then we draw conclusions and perspectives in Sec. VII.

II. Problem formulation

We consider a set of co-located users downloading contents with NDN [17] and focus on a scenario where all downloads follow the same path from a set of co-located servers holding original copies of the contents. A set of cache-enabled NDN routers are deployed over this path (see Fig. 1 for the particular case of one NDN router). Contents have different popularities and are grouped into M classes, where contents of class k are requested with rate λ_k. We assume contents have the same mean size S even though our model can be applied to the case of variable mean sizes. Finally, we consider links to be equally provisioned and of capacity B. Denote $1/\mu = S/B$ the mean download time of a content in case of full link speed. In Sec. VI, we discuss the implications of having links of different capacities.

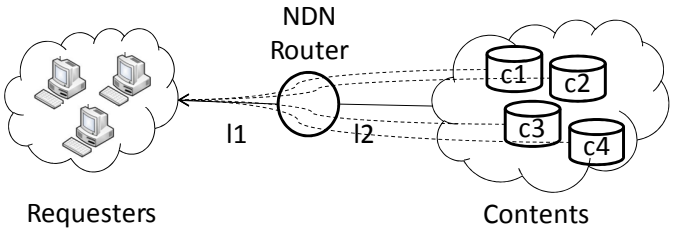


Fig. 1. Scenario with a chain topology composed of NDN routers.

We suppose that requesters control the congestion of the network by adapting the rate of their Interest packets according to the well-known window-based *Additive Increase Multiplicative Decrease* (AIMD) congestion control as in TCP. Congestion is inferred by Data packet losses and the window is increased linearly between congestion events and is decreased multiplicatively upon congestion. The loss of Data packets can be inferred either by explicit loss notification or by a timeout. We further assume that congestion only occurs in the download direction which is used by Data packets as the upload direction is used only by Interest packets, which, given their small size,

do not congest the network and therefore do not experience losses.

A. General throughput expression

We leverage well-known results in the literature for the download rate of an AIMD rate control [2]. If we refer to p_c as the probability that a Data packet of a chunk of content c is lost due to congestion[1] and to RTT_c as the mean round-trip time to retrieve chunks of content c, the mean download rate T_c for content c can be written according to the square root formula for AIMD throughput [2]:

$$T_c = \frac{K}{RTT_c\sqrt{p_c}}, \tag{1}$$

where K is a constant. The main difference in our case is that chunk copies can be found along the path to the server, and so the round-trip time can be smaller than in the case of TCP where the download is end-to-end. The probability to find chunks cached along the path depends on the hit ratio of NDN caches, which depends on the popularity of the content being downloaded and the caches capacity.

Eq. (1) states that all content downloads active at a given time and bottlenecked at the same link (and thus with the same congestion probability $p_c = p,\ \forall c$) will get a share of the bottleneck capacity proportional to the inverse of their respective RTT_c. The congestion probability p_c, as long as it is strictly positive, got canceled in the case of a single bottleneck link (which holds under our assumption of a chain of links of equal capacity where the bottleneck is at the access). We leverage this property later in the design of our DPS-based model for dynamic bandwidth sharing among NDN downloads but first have to precisely define RTT_c.

B. Evaluating the mean RTT

Popular contents are likely to have cached copies of their chunks at intermediate NDN routers so that a small RTT_c will be obtained. On the contrary, chunks of low popularity contents will be mostly retrieved from the original servers, thus achieving a large RTT_c.

If we denote by len_c the number of links located between the requesters and the original servers by $\omega_i(c)$ the hit ratio for content c of the cache of the NDN router located at i NDN hops from the requester, and if all chunks of a content have statistically the same hit ratio (without necessarily meaning they are cached or not cached together), we can write RTT_c as follows:

$$RTT_c = \sum_{i=1}^{len_c} d_i\omega_i(c)\prod_{j<i}[1-\omega_j(c)], \tag{2}$$

where d_i is the average round-trip delay (composed of both propagation and queuing delays) between the requester and the NDN router at the ith hop. It is easy to verify that RTT_c is close to d_1 for popular contents and to d_{len_c} for non popular ones.

The hit ratio $\omega_i(c)$ is a function of content popularity [19], [15] and is therefore an input of the problem that depends on how frequently the different content chunks are requested and the size and location of the NDN caches.

C. DPS sharing model

In this section, we apply the Discriminatory Processor Sharing model developed in [12] to derive expressions for the mean download time of AIMD in NDN. In queuing theory, a DPS models the case of a processor (bottleneck link in our case) distributing its capacity among the different available clients (downloads in our case) with constant weights that sum to one and that depend on the class to which the clients belong. Assuming that there are N_k active clients in the DPS belonging to class k, then each of them receives a quota of the capacity of the processor equal to $\frac{g_k}{\sum_{j=1} N_j \cdot g_j}$. By comparison to our case, g_k, the weight of a class of popularity k, is equivalent to the inverse of the mean RTT and the mean service time $(1/\mu)$ is equivalent to the mean download time of the content at full link speed (S/B). The contribution of [12] is to establish a linear system of equations having as solution the mean download time of the different classes of popularity, that we denote as W_k for class k. This is done under the particular assumptions of Poisson arrivals for requests and exponentially distributed content sizes, with clearly the difficulty to relax these assumptions. Fortunately, it has been demonstrated later in [3], [14] by means of simulations that these assumptions have minor impact on average performances when it comes to realistic arrival process and content size distributions.

Following the DPS approach, we clearly make the assumption that AIMD is able to grab instantaneously any available bandwidth and fairly share it among the different downloads according to the square root formula in Eq. (1). Our experiments and the ones in [3], [14] show that this modeling leads to reasonable approximation of bandwidth sharing and the authors in [14] advocate that approximating the instantaneous download rate of AIMD with its long-term download rate is an acceptable assumption within the DPS framework.

By adapting the result of [12] to our case, the average download time for contents belonging to the different popularity classes can be obtained by solving the following system of equations for all $k \in [1, M]$:

$$W_k \cdot \left[\mu - \sum_{j=1}^{M} \frac{\lambda_j \cdot g_j}{g_j + g_k}\right] - \sum_{j=1}^{M} \frac{\lambda_j \cdot g_j \cdot W_j}{g_j + g_k} = 1, \tag{3}$$

which is a linear system that we solve exactly next for the case of two classes of popularity and numerically for the general case of M classes. In this system, λ_k is the rate of requests for contents of class k, $\mu = B/S$ is the bottleneck capacity normalized to the mean content size, and g_k is the weight for class k. According to what we discovered in [24] and said earlier, g_k can be taken equal to $\frac{1}{RTT_k}$ where

[1] If NDN routers implement explicit loss notification, p_c can model the probability that a Data packet sent back to the requester carries such a signal.

RTT_k is provided by Eq. (2). Finally, this system is clearly established under the assumption of unsaturated bottleneck, i.e., $\rho = \sum_{j=1}^{M} \frac{\lambda_j}{\mu} < 1$.

III. The case of 2 classes of contents

To assess the impact of in-network caching on resource sharing, we define the *expansion factor* $\eta_k = \frac{W_k}{\overline{W}_k} \in [0, \infty)$ that compares download time with and without caching, where $\overline{W}_k$ is the download time obtained without caching. $\eta_k < 1$ indicates that average download time is shorter with caching (i.e., *contraction*). On the contrary, a value over one indicates a longer download time (i.e., *expansion*). In this section, we consider the extreme case of two classes of contents in Eq. (3) and one single cache and discuss general trends on how in-network caching influences resource sharing. The confirmation of the trends for a more general case is given in Sec. IV.

In a chain topology, all classes experience the same average download time in absence of caching,

$$\overline{W}_1 = \overline{W}_2 = \frac{1}{\mu - \lambda_1 - \lambda_2}, \tag{4}$$

which, combined with Eq. (3), gives

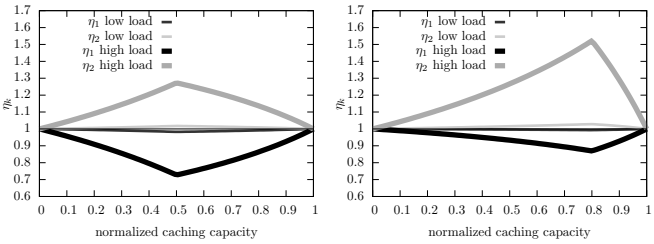
$$\eta_1 = \frac{W_1}{\overline{W}_1} = \frac{1 - a_{21} \cdot \frac{\lambda_1}{\mu} - a_{21} \cdot \frac{\lambda_2}{\mu}}{1 - a_{21} \cdot \frac{\lambda_1}{\mu} - a_{12} \cdot \frac{\lambda_2}{\mu}}, \tag{5}$$

$$\eta_2 = \frac{W_2}{\overline{W}_2} = \frac{1 - a_{12} \cdot \frac{\lambda_1}{\mu} - a_{12} \cdot \frac{\lambda_2}{\mu}}{1 - a_{21} \cdot \frac{\lambda_1}{\mu} - a_{12} \cdot \frac{\lambda_2}{\mu}}, \tag{6}$$

with $a_{kj} = \frac{g_j}{g_j + g_k} = \frac{RTT_k}{RTT_j + RTT_k}$.

The extreme values of η_1 and η_2 are obtained when the caching strategy is ideal and popularity is perfectly known, i.e., start caching high popularity class (class 1) then if space left cache contents of low popularity class (class 2). Fig. 2 shows the expansion factor as a function of caching capacity in this situation and when the cache is equidistant from the requester and the original server, at both low (i.e., $\rho = 0.1$) and high (i.e., $\rho = 0.9$) network load. The normalized caching capacity on the x-axis indicates the average proportion of downloads satisfied by the cache.

At low load, the impact of caching is negligible. On the contrary, at high load the download time for contents of class 1 is contracted but this comes at the expense of contents of class 2 that see their average download time expanded. Furthermore, the relative download time expansion that affects contents of class 2 is higher than the relative download time contraction that benefit contents of class 1. This phenomenon is more evident when class 1 is much more popular than class 2 as shown in Fig. 2(b). This behavior can be explained as the proportion of downloads related to contents of class 1 increases with the popularity of the class, thus resulting in a larger number of downloads that compete with the same RTT. In these conditions, an increase in the popularity of contents belonging to class 1 translates into less pronounced contraction of the download time (i.e., as it was in Fig. 2(a)). At the same time, available bandwidth for contents of class 2 decreases as well, due to the increased load at the link fed by the cache, thus yielding a further expansion of the download time.



(a) $\lambda_1 = \lambda_2 + \epsilon$ (b) $\lambda_1 = 4 \cdot \lambda_2$

Fig. 2. Evolution of the expansion factor with the cache size and the network load for two classes of contents.

Fig. 2 also shows that expansion (for contents of class 2) and contraction (for contents of class 1) saturate and converge to 1 when caching capacity increases. To explain this behavior, we should note that, as the cache size increases, a larger proportion of chunks can be retrieved from the cache which reduces the average RTT and increases the aggressiveness of downloads competing at the bottleneck, that after a given saturation point, translates into a reduced contraction. This results in a smaller bandwidth for each cached cached and finally in a smaller contraction of download time. Contrariwise, the download time of contents belonging to class 2 decreases (i.e., smaller expansion factor) after the saturation has been reached, because it is more likely that these contents can be retrieved from the cache with a smaller RTT than before the saturation.

The case where there is one dominant class of contents (namely class 1) with a request rate λ_1 much higher than other classes (i.e., $\lambda_1 >> \sum_k \lambda_k$, $k > 1$) can be analyzed by clustering all classes with $k \geq 2$ in a single class with a negligible request rate $\sum_{k=2}^{M} \lambda_k$. That way, we can use the results derived for the case of two classes. From Eq. (5), we get $\eta_1 \approx 1$ which means that the average download time of the popular class of contents is not influenced by the cache as class 1 accounts for the vast majority of flows. Also, since $\eta_1 \approx 1$, we can write $W_1 \approx \overline{W}_1 = \frac{\frac{1}{\mu}}{1-\rho}$, and for the second class:

$$\eta_2 \approx \frac{1 - a_{12} \cdot \rho}{1 - a_{21} \cdot \rho}. \tag{7}$$

Now, considering that Eq. (4) yields $\overline{W}_2 = \overline{W}_1$ and that $\overline{W}_1 \approx W_1$, it holds $\eta_2 \approx \frac{W_2}{W_1}$ that leads to:

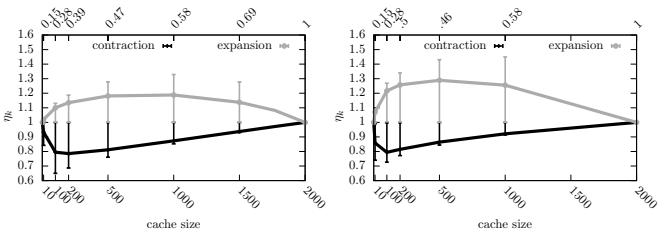
$$W_1 \approx \frac{\frac{1}{\mu}}{1 - \rho}, \tag{8}$$

$$W_2 \approx \left(\frac{1 - a_{12} \cdot \rho}{1 - a_{21} \cdot \rho} \right) \cdot W_1. \tag{9}$$

Proposition 1: Under the assumption that one class of contents is dominant compared to other classes, the average download time of non-popular classes of contents is larger than the download time of the dominant class by a factor that

len	length of the chain
B	bottleneck capacity
C_i	cache size at node i (in number of contents)
ρ	network load
M	number of content classes
λ_k	class k request rate
$\omega_i(k)$	class k hit rate at node i
W_k	class k download time
η_k	class k expansion factor
κ	over-provisioning factor

Fig. 3. Terminology summary.



(a) $\lambda_k \sim Zipf(0.8)$ (b) $\lambda_k \sim Zipf(1.0)$

Fig. 4. Evolution of the expansion factor with cache size for the most and least popular classes of contents [M = 2000, ρ = 0.9, len = 2].

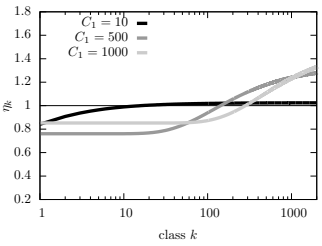


Fig. 5. Evolution of the expansion factor with cache size for the all classes of contents [M = 2000, ρ = 0.9, len = 2, $\lambda_k \sim Zipf(0.8)$].

is super-linear with network load and dependent upon cache size and position.

As contents of class 1 are more popular than those of class 2 and caching is in favor of popular contents, then $RTT_1 \leq RTT_2$ and $a_{21} \in [.5, 1]$[2]. In this case, considering that $a_{12} = 1 - a_{21}$, $\frac{1-a_{12}\cdot\rho}{1-a_{21}\cdot\rho} \geq 1$. From Eq. (9), $W_2 \geq W_1$ with a factor that is super linear with the load and a_{21}. a_{21} is function of the RTT that is influenced by cache size and position as shows in Eq. (2). $W_2 = W_1$ only when $RTT_1 = RTT_2$ or when $\rho = 0$.■

Proposition 2: When popular contents are cached much closer to the requesters than contents of lower popularity, the bias against non popular contents scales as $\frac{1}{1-\rho}$.

If popular contents are cached much closer to requesters than non popular contents (e.g., non popular contents are not cached), then $a_{21} \to 1$ and, from Eq. (9), $W_2 \approx \frac{1}{1-\rho} \cdot W_1$. ■

Proposition 3: The average download time of the non-popular class of contents is bounded by $\left[W_1, W_1 \cdot \frac{RTT_2}{RTT_1}\right]$.

As $a_{21} \in [.5, 1]$, Eq. (9) shows that for a given value of a_{21}, W_2 is a strictly increasing function that reaches its maximum for $\rho = 1$. In this case, $W_2 \approx \left(\frac{1-(1-a_{21})}{1-a_{21}}\right) \cdot W_1$ and, by substitution, we obtain $W_2 \approx \frac{RTT_2}{RTT_1} \cdot W_1$. ■

To sum up, Eq. (8) and Eq. (9), together with Proposition 1 and Proposition 2, show that download times W_1 and W_2 increase with network load but the penalty incurred by non popular contents grows much faster than for popular contents. Similarly, Proposition 3 demonstrates that the asymmetry between the caching location of popular and non popular contents increases the penalty as the closer to the requester the cache is, the larger the bias against non popular contents. These observations highlight the fact that at high load, the interplay between AIMD and in-network caching may impair the download time of less popular contents while not providing significant advantages to contents of higher popularity.

IV. The case of M classes of contents

Sec. III shows trends on the interactions between in-network caching and AIMD for two classes of contents. In this section, we extend the study to the general case of M classes of contents. However, expressions derived from Eq. (3) would become unintelligible, we thus use numerical resolutions to confirm and extend our previous findings. The download time W_k of any class $k \in [1, M]$ depends on the download time of the other classes of contents, which is tightly coupled with their RTT and hence with the cache hit function (see Eq. (2)).

As implementing perfect caching is hardly doable in practice, we consider *Least Recently Used* (LRU) cache eviction policy instead, which is commonly used in practice but that is not efficient if the number of classes of contents is small. By extending [9], [15] to the case of chain of LRU caches, the probability that a content c taken from a catalog of N contents is available at the node with cache size C_i, at i hops distance from the requester is $\omega_i(c) \approx 1 - e^{-q_i(c)\cdot\tau_{C_i}}$, with τ_{C_i} the root of $\sum_{c=1}^{N} 1 - e^{-q_i(c)\cdot t} = C_i$, solved with respect to t, and $q_i(c)$ is the probability that the node at i hops distance receives an Interest packet for content c. As we are in the case of a chain of caches, only the first node of the chain (i.e., the requester) generates Interest packets and $q_1(c)$ follows the same distribution as the requester demand, which is known to be a Zipf(α) with the α decay parameters ranging from 0.7 to 1 [15]. For any node $i > 1$, the demand for content equals the demand at the previous cache multiplied by its miss probability, so that $q_i(k) = q_{i-1}(k)\,[1 - \omega_{i-1}(k)]$.

Without loss of generality, we assume that each class of contents is composed of only one content. All chunks of the same content have the same popularity, hence the same average hit ratio, without necessarily being simultaneously cached or not at the same node. Fig. 4 shows the expansion factor for the case of a single LRU cache topology that is equidistant from the requester and the original server, and for M = 2000 classes of contents. Fig. 4(a) and Fig. 4(b) are obtained for a Zipf decay parameter 0.8 and 1.0, respectively. Fig. 4 is similar to Fig. 2 but shows the average values (with their extreme values) for the two categories of contents, those with a download time *contraction* and those with a download time *expansion*. The secondary x-axis shows the proportion of downloads with a contracted download time for the considered cache size.

The trends discussed in Sec. III are confirmed by Fig. 4. We verified that the impact of caching is negligible at low network load (i.e., $\rho = 0.1$) so we only present results for high network load (i.e., $\rho = 0.9$). Fig. 4 confirms that download time contraction comes at the expense of other contents. It

[2] a_{21} = 0.5 when $RTT_1 = RTT_2$ and a_{21} = 1 if contents of class 2 are not cached and contents of class 1 are cached on the client (RTT_1 = 0).

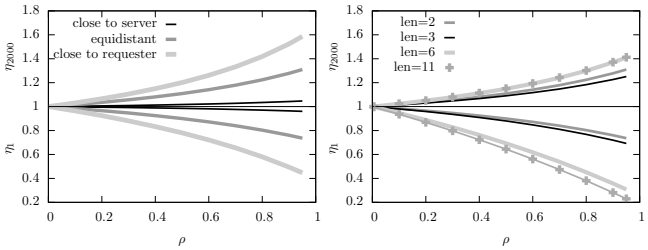
also confirms that the contraction of download time is lower than the expansion and that increasing too much the cache size does not improve individual download times. The extremes show that, within a category, resources are not allocated equally, with some contents much impacted than others. The comparison between Fig. 4(a) and Fig. 4(b) shows that the relative difference of popularity between classes of contents does not affect much the download time for usual Zipf decay parameters. However, as popular contents are more popular when the decay parameter increases, a smaller cache size is necessary to serve the same number of downloads from the cache. Nevertheless, performance gain rapidly decreases as more downloads are in competition with the same low delay, meaning also that other contents suffer more from caching. Fig. 5 helps to understand how resources are spread between classes of contents. As we can see, whatever the cache size is, most popular contents benefit the more from caching, at the expense of the least popular ones. Fig. 5 also confirms that when the cache size increases, more downloads can be served from the cache so more classes of contents contract their average download time. Finally, we can note that cache size growth for which contraction and expansion stop increasing are different, while they are the same in Fig. 2. This difference comes from the caching eviction policy. With the perfect caching used in Sec. III, least popular contents cannot benefit from caching if most populars don't. With LRU, instead, any content can benefit from caching.

A. Caching storage capacity distribution and download time

As Propositions 1 and 3 prove it, the location of the caches along the path to the original server, as well as their capacity, influence download time. However, these propositions are derived assuming two classes of contents and one cache only. In this section, we relax these constraints and evaluate also the case of M classes of contents and several caches spread along the path.

With two classes of contents and one cache the RTT of popular contents is reduced, i.e., a_{21} gets closer to 1, if the cache is placed closer to the requester, contracting so even more the download time of popular contents than in the equidistant scenario, at the expense of least popular ones. On the contrary, if the cache is placed closer to the original server, average RTTs are increased, i.e., a_{21} gets closer to 0.5, limiting so the download time contraction for popular contents but also the negative impact on least popular contents.

Fig. 6(a) shows the expansion factor for the most (i.e., η_1) and the least (i.e., η_{2000}) popular classes of contents in the more general case of M = 2000 classes and network different loads (and one cache). It confirms that placing the cache closer to the requester (e.g., 10% of the the RTT with the server) is in favor of the popular content and at the detriment of the non-popular one, but also that placing the cache closer to the original server (e.g., 90% of the RTT with the server) has limited impact on download times. Furthermore, it confirms Proposition 1 by showing a download time super-linearly influenced by the network load and the cache position.



(a) one cache (b) chain of caches, $C_i = \frac{500}{len-1}$

Fig. 6. Evolution of the expansion factor with the network load for the most and least popular classes of contents for different topologies [M = 2000, total caching capacity = 500, $\lambda_k \sim Zipf(0.8)$].

To finish the generalization, we consider a chain of caches and M classes of contents. Fig. 6(b) shows the expansion factor for different chain length of equal memory capacity summing up to 500 entries, and where each link has the same delay. The case of 1 cache (i.e., len = 2) is the same as in Fig. 6(a) and serves as a baseline. We can observe that increasing the number of caches has a positive impact on the download time even though the total caching capacity is unchanged. In our particular scenario with several caches, the most popular contents get similar benefits than in the situation where one single cache is used and put close to the requester. However, while this gain comes at a high cost for non popular content in the later case, the impact remains acceptable when several caches are used. This happens because less downloads are served by each cache as they are smaller meaning that there are less downloads in competition with the same RTT (and then throughput). Unfortunately, the benefit of chaining smaller caches comes at the cost of a lower overall hit ratio if caches do not collaborate as the same content might be replicated on several caches along the chain that eventually compensate the benefits. The study of the best tradeoff is out of the scope of this paper.

V. Experimental validation

To validate our model, we conducted real experiments with CCN-Joker [11], an open source emulation platform for NDN that allows the emulation of the essential aspects of NDN, such as Interest/Data handshakes and cache management policies [11]. For this paper we added an AIMD congestion control mechanism to CCN-Joker.[3] CCN-Joker can handle multiple downloads in parallel each with its own AIMD instance (see Fig. 7). All the architectural details of CCN-Joker are given in [11], so we focus on the main aspects of our AIMD implementation in the following.

Chunking and sequencing. In NDN, every content is decomposed into chunks that can be retrieved independently by sending Interest packets and potentially from different locations in the network. In CCN-Joker, every chunk that belongs to the same content is unambiguously identified with a progressive sequence number. The concatenation of this

[3]Our code is available at http://telematics.poliba.it/ccn-joker

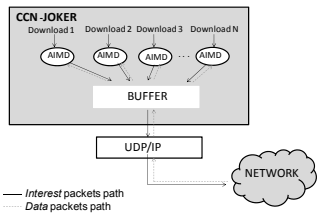
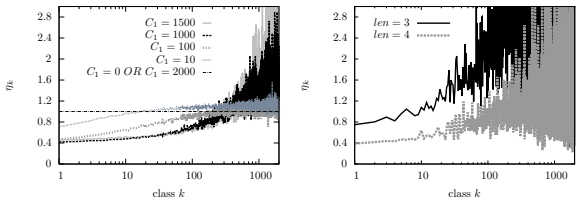


Fig. 7. Integration of AIMD within the CCN-Joker architecture.



(a) one cache (b) chain of caches, $C_i = \frac{1000}{len-1}$

Fig. 8. Evolution of the expansion factor with the cache size for all classes of contents [M = 2000, ρ = 0.9, $\lambda_k \sim Zipf(0.8)$].

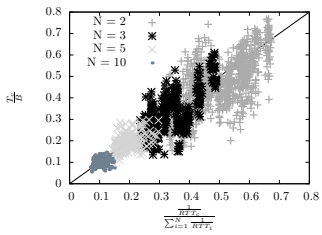


Fig. 9. Equivalence between throughput and $\frac{1}{RTT}$.

number with the name of the content the chunk belongs to is used to name the chunk of data than can hence be retrieved by its name without ambiguity.

Slow start and Congestion Avoidance We use the slow start and congestion avoidance algorithms to adapt the rate at which CCN-Joker injects Interest packets in the network. Accordingly to RFC5681 [1], the *Additive Increase* phase is stopped whenever a congestion is detected.

Congestion detection. In TCP a receiver should send an immediate duplicate ACK when an out-of-order segment arrives and the arrival of three duplicate ACKs is an indication that a segment has been lost and the fast retransmit mechanisms can be launched [1] to react to the event. Since there is no acknowledgment mechanism in NDN, we can only rely on the expiration of a *retransmission timeout* (RTO) as an implicit congestion notification, comparably to [5].

In particular, when an Interest packet is sent, the timer is set only if it was not turned on yet. Every time an in-order Data packet is received (i.e., there is no hole in the sequence numbers), the timer is reset (if there are still in-flight packets). When an out-of-order Data packet is received, it is temporally stored into a dedicated buffer, waiting for prior Data packets before being delivered to the client application. The RTO value is updated according to the RTT as in RFC6298 [20].

Congestion reaction. When the retransmission timer expires, the host supposes that a congestion event occurred and the *Multiplicative Decrease* mechanism is triggered according to RFC5681 [1]. CCN-Joker implements the so called *recovery phase*: all in-flight Interest packets are considered lost and thus re-transmitted. In this phase, RTT and RTO are no longer updated whereas the congestion window is updated for every Data packet retrieved. The recovery phase ends when all Interest packets that were in flight at the beginning of the congestion episode are successfully satisfied.

A. Experimental results

For the purpose of the model validation, we consider the chain topology and the generic case of M = 2000 classes of contents. Every content has the same content size S of 1MBytes, split in chunks of 10KBytes. Every node in the chain runs CCN-Joker and we use Dummynet [4] to emulate the capacity B of the bottleneck to 20Mbps. Requests for contents are generated according to a Poisson process of parameter λ computed from the network load ρ. In particular, $\lambda = \rho \frac{B}{S}$ and we impose content popularities to follow a Zipf distribution with parameter α = 0.8. For every cache size, we run one experiment with about 100,000 downloads.

Fig. 8(a) shows the average expansion factor η_k of every class of contents for different cache sizes C_1. Our experimentation confirms that the expansion for the most popular contents occurs at the expense of the less popular ones. However, a higher contraction (resp. expansion) is observed for the most (resp. least) popular classes of contents than in the numerical evaluation. This can be explained as in practice the most popular contents can adapt their rate faster to bandwidth changes than the least popular ones as their feedback loop is shorter (i.e., RTT) which is not taken into account by our model. Fig. 8(a) also validates that after some cache size increase, the contraction of popular contents is less important (e.g., download time contraction of popular contents is higher with a cache of 1,000 entries than a larger cache of 1,500 entries). Moreover, Fig. 8(b) verifies the expansion factor for a chain with several caches. As predicted by the model, increasing the number of caches improves the average download time of popular contents and reduces the negative impact on the less popular ones. The high variations observed for less popular contents comes from the limited number of downloads for the experiments and the contraction of download time observed for some non popular contents is because, on retransmission, a non-popular content might be served by the cache instead of the original server, which is not taken into account by our hit function model. We however verified that the average empirical hit ratio obtained while ignoring retransmitted Interest packets is similar to the one predicted by the model.

Fig. 9 validates that when N long lasting parallel downloads share the same bandwidth B, the throughput is proportional to the inverse of the RTT of the flow and that it holds true that $\frac{T_c}{B} = \frac{RTT_c^{-1}}{\sum_{i=1}^{N} RTT_i^{-1}}$, according to the AIMD throughput approximation at stationary regime. Furthermore, the similarity between numerical and empirical results confirms that the usage of the stationary regime formula for short term bandwidth sharing in DPS is acceptable. Indeed, Fig. 9 validates the assumption that AIMD driven flows rapidly obtain their fair-share throughput.

VI. Discussion

In this paper, we demonstrate the existence of a bias against non popular contents when NDN and its intrinsic in-network caching is used. In the following, we discuss the roots of this bias together with ideas on possible solutions.

Client access link to be over-provisioned. So far we have considered a chain of caches connected through links of equal capacity B. This choice entails that the bottleneck is of capacity B, independently of the presence of caches. Without caches, the bottleneck is likely to be towards the server, whereas with NDN, and because of traffic reduction toward the server thanks to caching, it is at the access link. Herein, we discuss the effects of relaxing the assumption of having links of equal capacity B and assess to what extent increasing access link capacity could improve bandwidth sharing. The idea is to over-provision the access link such that, when caching is activated, the average download time for the least popular class of contents is not higher than without in-network caches. In this case, popular contents still keep obtaining shorter download times leveraging in-network caching but without any perceivable negative effect on the least popular contents.

All the findings of the paper also apply to chains of different link capacities, as long as the bottleneck is shared by every download (i.e., located at the access link when NDN is activated). If the bottleneck is not shared by every download (i.e., it is located after a cache) then a similar DPS-based approach as in Eq. (4.12) of [12] can be applied, with the weights g_k defined by taking into account the difference in congestion probability p_c and normalized link capacity ($\mu = \frac{B}{S}$) between downloads. We leave this investigation to a future research and limit ourselves here to the case of one bottleneck shared by all downloads (i.e., the access link), and identify how its capacity should be increased in presence of caches so that every download gets at least the same performance as in absence of cache.

Let $\overline{W}_k$ be as before the mean download time for class k when caching is disabled and with a bottleneck of capacity B yielding to $\mu = \frac{B}{S}$. If bottleneck capacity is multiplied by a factor $\kappa \geq 1$, the bottleneck yields to $\mu' = \mu \cdot \kappa$ instead of μ. The new mean download time for class k is then W'_k, and is obtained by solving Eq. (3) with μ' instead of μ so that:

$$W'_k \cdot \left[\mu \cdot \kappa - \sum_{j=1}^{M} \frac{\lambda_j \cdot g_j}{g_j + g_k} \right] - \sum_{j=1}^{M} \frac{\lambda_j \cdot g_j \cdot W'_j}{g_j + g_k} = 1, \qquad (10)$$

for $k = 1 \dots M$. Our goal is to evaluate the minimum increase of capacity at the access link, κ, so that $W'_M \leq \overline{W}_M$, M being the index of the most unpopular class that, according to our previous analysis, undergoes the maximum degradation in its download time.

Fig. 10 plots the ratio $W'_k/\overline{W}_k$ for all classes of contents, by solving the above system numerically for different values of κ. We consider a scenario of one cache of size equal to 500 classes, located equidistantly between the requester and the original server. For every value of κ, different bars are plotted for different loads, each bar showing the span of the ratio $W_k/\overline{W}_k$ over all classes.

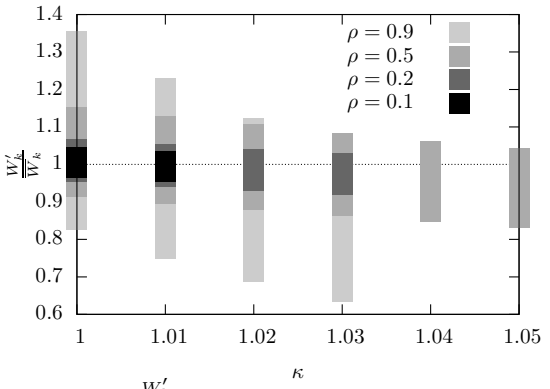


Fig. 10. Evolution of $\frac{W'_k}{\overline{W}_k}$ for different loads [C_1 = 500, $\lambda_k \sim Zipf(0.8)$, M=2000].

Surprisingly and for all considered loads, a capacity increase of few percent is enough to make $W'_k/\overline{W}_k$ less than 1 (i.e., download time with caching is no larger than without), with the required value of κ evolving non-linearly with the load. Therefore, increasing the capacity of the access link by a few percent while activating NDN and in-network caching allows non popular contents to preserve the same mean download time than before the migration and popular contents still improve their download time. Obviously, at high network load, the necessary over-provisioning is smaller as all downloads were already subject to congestion. We observe the same over-provisioning factor order in other scenarios. This encouraging results shows that only a small effort is needed to remove any negative impact introduced by the adoption of NDN.

We can provide the analytic expressions for the factor κ in the particular case of two classes of contents, with one dominant (namely class 1) with respect to the other (namely class 2). Leveraging the results of Sec. III, where we provide explicit expressions for the mean download time in this case (Eq. (8) and (9)), we have:

$$\frac{1-(1-a_{21})\cdot\frac{\lambda_1+\lambda_2}{\mu\cdot\kappa}}{1-a_{21}\cdot\frac{\lambda_1+\lambda_2}{\mu\cdot\kappa}} \cdot \frac{\frac{1}{\mu\cdot\kappa}}{1-\frac{\lambda_1+\lambda_2}{\mu\cdot\kappa}} = \frac{\frac{1}{\mu}}{1-\frac{\lambda_1+\lambda_2}{\mu}}. \qquad (11)$$

The term on the left-hand side is no other than W'_2 whereas the term on the right hand-side is $\overline{W}_2$. By replacing $\frac{\lambda_1+\lambda_2}{\mu}$ by ρ, which is the load on the network without over-provisioning, we obtain a polynomial of degree 2, whose solution (the positive one larger than 1) is no other than κ:

$$\kappa^2 - \kappa\left(1 + a_{21}\cdot\rho\right) + \left[\left(2a_{21}-1\right)\rho^2 + \left(1-a_{21}\right)\rho\right] = 0. \qquad (12)$$

Remarkably, the polynomial defining κ solely depends on the load ρ and the factor a_{21} that models the relative difference of delay between contents of class 1 and contents of class 2. Similar to Proposition 2, when popular contents have a much smaller RTT than non popular contents (i.e., $a_{21} \to 1$), the polynomial becomes:

$$\kappa^2 - \kappa\cdot\left(1+\rho\right) + \rho^2 = 0. \qquad (13)$$

Given that this latter case can be seen as a worst case for unpopular contents, resolving this polynomial gives a good

approximation of how to over-provisioning access links upon the introduction of NDN.

Dependence of AIMD on RTT. A source of the bias against non popular contents is the dependence of the throughput of AIMD on the round-trip time. Indeed, caching is reflected as network delay and hence directly influence the throughput. If one breaks this dependency and ensures the same throughput for AIMD for any RTT, and as long as the congestion probability p_c is the same (Eq. (1)), the mean download time for all contents will be the same (the weights of the DPS, g_k, will be equal yielding the same W_k for all classes). Compared to the over-provisioning proposition which guarantees that the mean download time is not negatively impacted for non popular contents and lets popular ones improve their download time, this solution leads to equal download times for all classes of popularity. Similar discussions took place within the TCP community regarding bandwidth sharing in case of TCP connections of different round-trip times [13], [16]. Such throughput equality can be achieved either by changing clients so that the window increase for unpopular contents (those with large RTT) can be made faster (for example proportionally to RTT) or by introducing mechanisms inside NDN routers to differentiate between downloads upon congestion (for example by dropping with higher probability Data packets served locally compared to those in transit from other routers). We keep the exploration of these issues for a future research.

VII. Conclusion and Future Research

we have studied the problem of bandwidth sharing in Information-Centric Networking and identified whether contents of low popularity are penalized. Assuming that content requesters implement the well-known AIMD schema to control their request rate and by the help of a new model based on the theory of Discriminatory Processor Sharing and real experiments with CCN-Joker, we have observed that content classes of low popularity can loose in download time because of the greediness of cached popular contents. We quantified this bias and proposed solutions to recover from it, either in the form of bandwidth over-provisioning or rethinking the congestion control to make it delay insensitive. Our observations, derived in the particular case of a chain of equal capacity links, sheds light on more complex scenarios, as long as popular and non popular contents are bottlenecked by the same link. We plan to examine whether more general scenarios (e.g., complex network topologies, contents of different characteristics according to popularity) have other implications on the way bandwidth is sharing. Our model is general and can be used to model the impact of CDNs on fairness, however in this context the network delay is not as strongly correlated with popularity as in the case of Information-Centric Networking which would diminish the bias against non-popular contents. ICN is a promising approach to improve network performance but the case of non popular contents must be treated carefully if we want better quality of experience for everyone. This paper focused on individual performances, but we plan to study performance at the global level as well, particularly since we observed that the sum of download times is the same whether or not caching is used.

References

[1] M. Allman, V. Paxson, and E. Blanton. RFC 5681 (rfc5681) - TCP Congestion Control. Technical Report 5681, 2009.

[2] E. Altman, K. Avrachenkov, and C. Barakat. A stochastic model of tcp/ip with stationary random losses. *IEEE/ACM Trans. Netw.*, 13(2), Apr. 2005.

[3] T. Bonald and L. Massoulié. Impact of fairness on internet performance. In *Proc. of the ACM SIGMETRICS*, June 2001.

[4] M. Carbone and L. Rizzo. Dummynet Revisited. *ACM SIGCOMM Computer Communication Review*, 40(2):12–20, Apr. 2010.

[5] G. Carofiglio, M. Gallo, and L. Muscariello. ICP: Design and Evaluation of an Interest Control Protocol for Content-Centric Networking. In *Proc. of the IEEE INFOCOM, NOMEN Workshop*, Mar. 2012.

[6] G. Carofiglio, M. Gallo, and L. Muscariello. Joint hop-by-hop and receiver-driven interest control protocol for content-centric networks. In *Proc. of the ACM SIGCOMM, ICN workshop*, Aug. 2012.

[7] G. Carofiglio, M. Gallo, L. Muscariello, and M. Papalini. Multipath congestion control in content-centric networks. In *Proc. of the IEEE INFOCOM, NOMEN Workshops*, Apr. 2013.

[8] G. Carofiglio, G. Morabito, L. Muscariello, I. Solis, and M. Varvello. From Content Delivery Today to Information Centric Networking. *Elsevier Computer Networks*, July 2013.

[9] H. Che, Y. Tung, and Z. Wang. Hierarchical web caching systems: Modeling, design and experimental results. *IEEE J.Sel. A. Commun.*, 20(7):1305–1314, Sept. 2006.

[10] D. M. Chiu and R. Jain. Analysis of the increase and decrease algorithms for congestion avoidance in computer networks. *Computer Networks and ISDN Systems*, 17(1), Jan. 1989.

[11] I. Cianci, L. A. Grieco, and G. Boggia. CCN - Java Opensource Kit EmulatoR for wireless ad hoc networks. In *7th ACM Int. Conf. on Future Internet Technologies*, Seoul, Korea, Sept. 2012.

[12] G. Fayolle, I. Mitrani, and R. Iasnogorodski. Sharing a processor among many job classes. *Journal of the ACM (JACM)*, 27(3):519–532, June 1980.

[13] S. Floyd. Connections with multiple congested gateways in packet-switched networks part 1: one-way traffic. *SIGCOMM Comput. Commun. Rev.*, 21(5):30–47, Oct. 1991.

[14] S. B. Fred, T. Bonald, A. Proutiere, G. Régnié, and J. W. Roberts. Statistical bandwidth sharing: a study of congestion at flow level. In *Proc. of the ACM SIGCOMM*, Aug. 2001.

[15] C. Fricker, P. Robert, and J. Roberts. A versatile and accurate approximation for LRU cache performance. In *Proc. of the International Teletraffic Congress (ITC)*, Krakow, Poland, Sept. 2012.

[16] T. R. Henderson, E. Sahouria, S. McCanne, and R. H. Katz. On improving the fairness of tcp congestion avoidance. In *Proc. of IEEE GLOBECOM*, Nov. 1998.

[17] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard. Networking named content. In *Proc. of the ACM CoNEXT*, Dec. 2009.

[18] C. Labovitz, S. Iekel-Johnson, D. McPherson, J. Oberheide, F. Jahanian, and M. Karir. ATLAS Internet Observatory 2009 Annual Report. Technical report, 2009.

[19] L. Muscariello, G. Carofiglio, and M. Gallo. Bandwidth and storage sharing performance in information centric networking. In *Proc. of the ACM SIGCOMM ICN workshop*, Aug. 2011.

[20] V. Paxson, M. Allman, J. Chu, and M. Sargent. RFC 6298 (rfc6298) - Computing TCP's Retransmission Timer. Technical Report 6298, 2011.

[21] D. Rossi and G. Rossini. On sizing CCN content stores by exploiting topological information. In *Proc. of the IEEE INFOCOM NOMEN Workshops*, Mar. 2012.

[22] L. Saino, C. Cocora, and G. Pavlou. CCTCP: A Scalable Receiver-driven Congestion Control Protocol for Content Centric Networking. In *in Proc. of the IEEE ICC*, Jan. 2013.

[23] S. Salsano, A. Detti, M. Cancellieri, M. Pomposini, and N. Blefari-Melazzi. Transport-layer issues in information centric networks. In *Proc. of the ACM workshop on Information-centric networking (ICN)*, Aug. 2012.

[24] D. Saucez, L. A. Grieco, and C. Barakat. AIMD and CCN: past and novel acronyms working together in the Future Internet. In *ACM Capacity Sharing Workshop (CSWS)*, Dec. 2012.

[25] G. Xylomenos, C. N. Ververidis, V. A. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, and G. C. Katsaros, K. V.and Polyzos. A Survey of Information-Centric Networking Research. *IEEE Communications Surveys and Tutorials*, PP(99):1–26, July 2013.