In-network packet-level caching for error recovery in ICN

Yannis Thomas, George Xylomenos, George C. Polyzos
Mobile Multimedia Laboratory
Department of Informatics, School of Information Sciences and Technology
Athens University of Economics and Business, Greece
{thomasi,xgeorge,polyzos}@aueb.gr

***Abstract*—In-network packet-level caching is one of the salient features of Information-Centric Networking (ICN) architectures, offering reduced communication latency, sender load, and network traffic. The literature on ICN caching is quite extensive, exploring in depth different aspects of caching, such as the positioning of storage resources in the network, the caching policies of individual caches, as well as the interaction of caching and routing. However, many researchers have questioned the value of in-network caching, arguing that the limited storage resources of ICN routers cannot eliminate a significant amount of redundant transmissions, hence the performance advantages of in-network caching may be similar to those of traditional edge-caches. Nevertheless, there is another potential use of in-network caching that poses less challenging storage requirements: exploiting it to enhance the efficiency of error recovery by speeding-up retransmissions of lost packets. In this paper, we investigate the impact of in-network packet-level caching on transport-layer error recovery. We first sketch a retransmission-oriented packet-level caching scheme, highlighting the differences in the performance requirements compared to general-purpose caches. Then we model and assess its expected gains in terms of latency and network traffic reduction, and finally we provide a reality check of the approach based on current network requirements and hardware specifications.**

***Index Terms*—Information-Centric Networking, latency, delay, retransmission, receiver-driven, transport protocol, router, cache, content**

I. Introduction

The *Information-centric Networking* (ICN) paradigm proposes a clean-slate network architecture, where all network operations refer to the information itself, in contrast to IP-based networking, where communication is endpoint-oriented [1]. ICN builds communication upon *names*: each information item has a unique (at least statistically) name (or identifier) that is used for discovering content, as well as making routing and forwarding decisions. By exploiting named content items, or even packets, ICN inherently supports efficient content delivery mechanisms such as multipath, multisource, seamless consumer mobility, and in-network packet-level caching. With ICN, routers are designed to exploit their buffers as caching modules, where forwarded packets can be inserted, evicted, and looked up based on their names. Thereafter, (on-path) routers are able to respond to packet requests by sending back the packet that is stored in their local buffer/cache, thus reducing the communication latency, the network traffic footprint and the load of the original packet sender.

By regarding buffers as caching modules, ICN introduces a radical change in network operation that allows rethinking the status quo of network caching. Numerous studies have been published discussing caching capacity allocation in the network, local and cooperative caching policies, as well as integrated caching and routing designs [2]. Despite the expectations of the ICN proposals, in-network caching does not appear to greatly enhance network performance, especially considering the hardware limitations of routers and the drastic changes that the network must undergo. In [3] and [4], the authors argue that the networks of small router-caches does not offer considerable gains when compared to the large caches that are typically located at the edges and at the central nodes of the network, respectively, thus fueling the skepticism about in-network caching.

The popularity of in-network caching has been further decreased by studies that argue against its feasibility based on the current technological status. ICN's practicality was first questioned in [5], thus triggering a wave of resource-aware studies on ICN routers [6]–[9]. Modern hardware routers are highly optimized devices driven by IP-specific requirements, hence ICN requirements, such as forwarding packets based on names and operating packet-level caches, are not easily met. Currently, there is no established design of a scalable ICN router that takes into consideration a general-purpose caching module, thus hindering the research on this topic.

However, there is another well-known potential use of in-network caching that can be exploited to enhance network performance: speeding-up retransmissions of lost packets [10]. According to this design, the in-network routers are equipped with a relatively small cache module that handles retransmissions of lost packets by storing only recently transmitted content. In this work, we sketch a retransmission-oriented packet-level cache, we analyze the expected gains in terms of latency and network traffic reduction, and we provide a reality check of the solution based on current network requirements and hardware specifications. We observe that the main advantage of this design is found in the brief inter-arrival time of retransmissions, which reduces the memory requirements of the caches. The second important feature of the design is that retransmissions are expected to occur over the same router port, thus simplifying the implementation of caches in a distributed per-port fashion which is compatible with the architecture of terabit core routers.

The structure of the remainder of this paper is as follows. In Section II, we review the literature on ICN router designs, while in Section III, we present a design of a retransmission-oriented ICN cache router. In Section IV, we model and preliminarily assess the foreseen gains, while in Section V we explore the feasibility of deploying retransmission-oriented packet-level caching in core routers. Finally, we present our conclusions and directions for future work in Section VI.

II. Related work

In order to support name-based network operation, ICN research reshaped the operation of network routers introducing sophisticated features, such as native multisource, multicast and caching, but also inducing significant overhead. The feasibility concerns of ICN networks are mostly raised due to the processing and memory overhead which can become a performance bottleneck for ICN routers. Typically, a large-scale router is equipped with two types of memory: a fast but small memory ($M1$), which is local at each line card and a slow but large memory ($M2$), which is common for all line cards [11]. The operations that take place for each packet, such as buffering packets and making forwarding decisions, exploit the $M1$ memory in order to support wire-speed performance. Based on current memory access latency and packet forwarding rate, in order to offer terabit operation, it is not recommended to use the $M2$ for packet-level insertions, evictions or lookups in a cache. At this point, we are not aware of a study that describes how $M1$ memory can be shared by caching and forwarding functions, hence we consider forwarding out of the scope of this paper and focus on the usage of memory only for caching.

In the literature, there are several candidate memory technologies for router memory. From the fastest to the slowest, *Static Random-Access Memory* (SRAM), *Dynamic Random-Access Memory* (DRAM), *Reduced Latency DRAM* (RLDRAM) and *Solid State Disk* (SSD) are most frequently considered [5]. These technologies exhibit a trade-off between capacity and latency, hence the fastest ones offer the least capacity. In large-scale routers, $M1$ and $M2$ memories are typically based on SRAM and DRAM, respectively.

In an effort to increase the buffering space while maintaining line-speed operation, hybrid SRAM/DRAM designs are also proposed [12], [13]. According to these studies, a buffer that follows the *First In First Out* (FIFO) policy can be implemented via a hierarchy with two SRAM memories, one acting as the tail and the other as the head of the FIFO, and one DRAM memory, where blocks (or batches) of packets are written periodically from and to the tail and head SRAM memories, respectively. The hybrid design matches the performance of SRAM with the density characteristics of a DRAM, however it cannot be considered compatible with packet-level caching, since DRAM is accessed periodically instead of on per-packet basis.

Acknowledging the size disharmony between the two memory areas, most studies on ICN caching routers propose using a *hierarchical* design, where the large $M2$ memory is used for storing the content and the smaller $M1$ memory is used for storing an index to the cached content, thus introducing the L2 and L1 cache layer, respectively [8], [9], [14]. In [14], the authors propose that the L1 storage and the L2 index is stored in the DRAM and the L2 storage is kept in an SSD. This design can sustain cache operations of up to 10 Gbps running on commodity hardware. In [8], the authors introduce a cache design for storing video items, where the L1 index is stored in the SRAM while the L1 storage remains in the DRAM, hence at L1 a SRAM chip indexes a DRAM cache, while at L2 a DRAM chip indexes an SSD cache. Acknowledging that SSD access time is rather high, it is suggested that batches of packets should be moved from SSD to DRAM occasionally, similarly to the hybrid SRAM/DRAM designs. In [9] the authors propose a cache architecture for ICN where the index of the cache is placed in the SRAM, indexing the cached packets that are stored in the DRAM. For each packet one (at least) SRAM lookup is made, but DRAM is accessed only for a subset of packets, thus potentially avoiding the L2 performance bottleneck.

When it comes to supporting terabit operation, a distributed routing design is exploited by core routers [15]. The first distributed ICN router, where each router core is responsible for routing a subset of the name space, was introduced in [7] for *Named-Data Networking* (NDN) [1], one of the most popular ICN architectures. The design was further developed in [6], where certain issues that are specific to NDN are revealed, e.g., the request and response of a packet needs to be processed by the same router core, thus imposing stateful operation. This issue could question the feasibility of a distributed routing design for NDN since it could hinder router scalability. Overall, the impelementation of content-based routing in core routers is rather challenging, thus not leaving much space for other operations; the operation of caching in distributed ICN routers has not been technically examined yet.

III. System description

The *Information-centric Networking* (ICN) paradigm proposes a clean-slate network architecture that bases network operations on content itself. Different variants of ICN architectures have been introduced [1], each presenting different forwarding and routing methods, however the majority of these architectures inherently support packet-level in-network caching. In what follows, we describe the essential features of an ICN architecture in the context of retransmission-oriented caching:

- **Named-packets**: Each content item is fragmented into *chunks* that are transmitted autonomously over the network. Given that ICN routers perform routing and forwarding based on names instead of addresses, a chunk is encapsulated in a *named*-packet, that is, a network packet that carries a (statistically) unique name that identifies the carried chunk. For the rest of this paper, the term packet refers to a named-packet.
- **Receiver-driven model**: A *receiver*, the terminal of the user who requests the content item, fetches the content item by sending a *Request* packet for each chunk of that

content item to the *sender*, the provider of the content. The sender responds with a *Data* packet. The two packets carry the same name, the unique name that identifies the chunk, hence an in-network router can identify pairs of Request-Data packets. The sender is stateless, responding to incoming Requests, hence the receiver is responsible for managing congestion control and retransmissions. There are some congestion control designs that assume in-network rate control and error recovery, but past experience suggests that end-to-end algorithms are more likely to scale up [16]. In our system, we assume that the receiver implements a TCP-like congestion control algorithm [17].

- **In-network cache routers**: All network routers exploit the buffers where packets are temporarily stored before being forwarded, as caches, where Data packets are inserted, removed and indexed. Upon the receipt of a Data packet, the router inserts it in the cache and, upon the receipt of a Request, the router searches for the corresponding Data in its cache. If it is found, it sends the Data back to the receiver, otherwise, it simply forwards the Request to its destination.
- **Cache replacement policy**: Content allocation in in-network caches is not static, instead routers dynamically store the incoming streams of Data in order to adapt to the changing popularity of content items. When a cache is full, then some cached items need to be evicted before inserting fresh content, thus realizing a *cache policy*. Typically, general-purpose ICN routers, that target Web traffic, consider the *Least Recently Used* (LRU) policy in order to keep in the cache the most popular content, exploiting the zipf-like distribution of web content popularity. In retransmission-oriented caching, we propose instead the exploitation of the FIFO policy, in order to keep in the cache the most recently transmitted content, thus targeting retransmissions.
- **Symmetric routing**: Although asymmetric routing is supported by certain ICN variants [18], the most common approach is to route the Request and Data packets over the same dissemination path.
- **Multicast transport**: Multicast content delivery is inherently supported by ICN architectures. Typically, multicast transport is suggested for fixed-rate sender-driven transmissions that incorporate *Forward Error Correction* (FEC) techniques to overcome packet losses [19]. However, the NDN architecture does support receiver-driven multicast delivery by, first, aggregating the Requests for the same content at routers and, then, "cloning" the received Data to multiple receivers. In this case, multicast transport is transparent to the end-users and, in turn, compatible with our analysis.
- **Multipath transport**: ICN architectures offer multipath deliveries by routing Requests over different paths to one or multiple content sources (native multisource). The majority of multipath protocols follow the receiver-driven paradigm, also assuming end-to-end error control through retransmissions [18], hence our analysis is directly applicable to them. However, for reasons of simplicity, we do not investigate multipath transport in this paper.

In Fig. 1, we demonstrate a use case scenario of retransmission-oriented packet-level caching, assuming a dissemination path of three caching routers, namely, $R1$, $R2$ and $R3$. Data retrieval is split into four phases. In the first phase, the receiver emits the Request to the sender and, in the second phase, the sender sends back the requested Data. If the Data is lost at some point of the delivery, for example, before reaching $R1$, it can be found cached at the previous on-path routers (in $R2$ and $R3$ in our example). In the third phase, the receiver emits a retransmission of its Request, which will find the Data cached at an on-path router ($R2$ in our example). Finally, in the fourth phase, the caching router replies to the retransmitted Request with the cached Data. By allowing $R2$ to send the Data, the network performance is improved in three directions: first, the receiver recovers from loss faster, second, the dissemination path is shortened, which also reduces the network load, and finally the sender load is reduced.

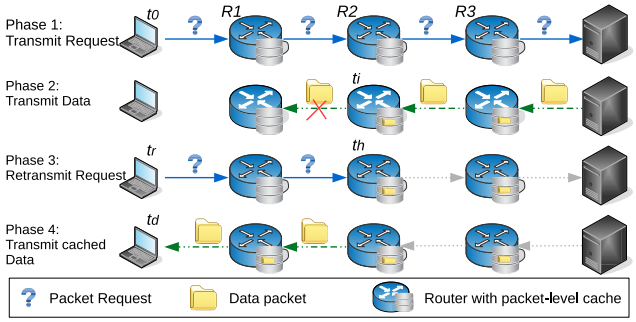

Fig. 1. Example of cache response to a retransmit packet request.

IV. Gains Assessment

In this section, we model and assess the foreseen gains of retransmission-oriented in-network caching. We begin by exploring the most obvious gain, the latency reduction of recovering from loss through faster retransmissions. The overall error recovery latency consists of the *loss detection latency*, e, and the *retransmission latency*, d.

A. Loss detection latency

The loss detection latency, e, is the time between the emission of the Request and the detection of the loss. Typically, e depends on the error detection mechanism that the transport protocol of the network adopts; we discuss below the three most common methods: time-based detection, sequence-based detection and notification-based detection.

The most popular method exploits a time-based loss detection mechanism where the packet must be delivered in a specific time period, namely, the *Time-Out* (TO) period, otherwise it is considered lost. The TO must be slightly higher that the Round-Trip Time (RTT) of the connection, r, to avoid retransmitting packets that are marginally late [17]. In the first TCP implementations a low threshold of 1 s was considered for TO, in order to avoid spurious retransmissions [20]. In modern implementations this threshold may be lowered, to

meet the performance needs of modern technologies, hence it is excluded from this study. RTT-based loss detection can become inexact in the face of RTT fluctuations, presenting significant latency as well as spurious retransmissions, hence a sequence-based method is also considered. The second method, known as *Fast Retransmit* (FR), specifies a sequence-based loss detection mechanism where the loss of a packet is signaled when three out of order packets are delivered [17]. Assuming that the N^{th} packet is lost, then the $(N+3)^{rd}$ packet indicates the loss. Therefore, the latency of FR is roughly $(w+3)b$, where b is the inter-arrival time of packet deliveries and w is the congestion window size in packets. The inter-arrival time can be approximated as r/w, hence the FR detection latency is approximately $r + 3r/w$, which can be simplified to r in case of large windows. The third method introduces a notification-based detection approach that typically exploits the *Random Early Detection* (RED) algorithm, a proactive flow control algorithm that can prevent congestion collapses [21]. Assuming that the network participates in congestion control, the routers can use RED in order to inspect the number of queued packets and proactively drop packets before the queue overflows and congestion leads to a burst of errors. In contrast to previous approaches, this method "selects" a loss, thus presenting no detection latency. However, a loss-indicating packet needs to be sent to the end-user upon a packet drop. The expected value of the detection latency with RED enabled in network routers is roughly $r/2$, thus significantly speeding up the process of error recovery. Note that this method only supports losses due to congestion, and cannot handle error-prone links, such as wireless links, hence end-hosts need to also deploy one of the previous methods. In this work, we assume a sequence-based loss detection mechanism due to its broader range of application and the relatively faster and more accurate loss detection that it presents.

B. Retransmission latency

After the loss is detected, the Request is retransmitted in order to fetch the Data. The time period between the emission of the retransmitted Request and the reception of the Data is the retransmission latency, d. Without caching, d is equal to r, but it can be reduced if the Data is found cached at an on-path router, since the Request and the Data do not traverse the entire dissemination path. The reduction of latency depends on two factors: the type of packet that is lost (Request or Data) and the distance from the source that the loss occurs. First, if the Request is lost, then the on-path routers will not have the chance to cache the corresponding Data in order to respond to the forthcoming retransmission, hence the latency cannot be reduced. Second, for a Data, the later the loss happens (closer to the destination), the higher the reduction of latency is. For instance, the latency reduction is maximized when the loss occurs at the last hop before the receiver (R1 in Fig. 1), since the Data can be found cached at the first on-path router. On the other hand, the latency is reduced the least when the loss happens in the first hop after the sender (R3 in Fig. 1), since the retransmitted Request will be served by the last on-path router.

In order to model latency reduction, we initially assume that the path links exhibit equal propagation latency. We study d in relation to the number of hops that compose the dissemination path, H, which is equal to the number of packet forwardings needed to send a packet from the sender to the receiver. If the dissemination path is one-hop long ($H = 1$), then the Data is retrieved directly from the sender, hence $d = r$. If the dissemination path is k-hops long ($H = k$), the value of d depends on the order of the hop (towards to the sender) where the loss occurred, i, i.e.:

$$d_i = ri/H \tag{1}$$

For instance, if a Data was lost at the 2^{nd} hop of a 5-hop path ($i = 2$ and $d = 5$), then the latency of the retransmission will be $d = 2/5r$, thus reducing it by 60%.

In order to estimate the expected latency of fetching a retransmitted packet, $\overline{d}$, we take into account the distribution of losses in the hops of the path, which we call *distribution of loss occurrences*. We introduce π_i^n, the fraction of losses that occur at hop i, where n is the type of packet, e.g., $\pi_2^{Request}$ is the fraction of losses of Requests, that occur 2 hops away from the receiver, and π_2^{Data} is the fraction of losses of Data packets, that occur 2 hops away from the receiver. Note that the distribution of loss occurrences is related, but different, to the error-rate of the links. For example, if all links have the same loss rate, we will have a decreasing distribution of losses in the hops, since as each links drops a fraction of the packets, later links see fewer packets overall. When considering only the packet requests that lead to losses, we can conclude that:

$$\sum_{i=1}^{H} \pi_i^{Request} + \sum_{i=1}^{H} \pi_i^{Data} = 1 \tag{2}$$

Thereupon, we combine (2) with (1) in order to estimate the average amount of retransmission latency throughout the transfer, based on the the fraction of losses on each link and the latency reduction that each link offers.

$$\overline{d} = r \sum_{i=1}^{H} \pi_i^{Request} + \sum_{i=1}^{H} \pi_i^{Data} d_i$$

$$\overline{d} = r \sum_{i=1}^{H} \pi_i^{Request} + \frac{r}{H} \sum_{i=1}^{H} \pi_i^{Data} i \tag{3}$$

Equation (3) shows that, when only Requests are lost, then the expected retransmission latency is r; caching does not help. On the other hand, when only Data packets are lost, then the expected retransmission latency depends on the distribution of losses. For example, in a path of H hops where losses are uniformly distributed among hops and only Data packets are lost ($\pi_i^{Request} = 0$ and $\pi_i^{Data} = 1/H$), $\overline{d}$ is equal to the retransmission latency measured when losses take place only in the middle hop, that is, $r(H + 1)/(2H)$.

Furthermore, we expect that, given a hop i, the fraction of losses of Requests will be typically smaller than the fraction of losses of the Data packets, due to packet size asymmetry; when traveling over links with the same error rate, the larger packets are more likely to be corrupted. Therefore, assuming

that Request and Data packets present fixed size, we suggest the following:

$$\pi_i^{Data} = a\pi_i^{Request}$$

where a is a positive real number that describes the path loss rate asymmetry between Data and Requests. For example, in a path of H hops with a uniform loss occurrence distribution all hops have equal fraction of losses, $\pi_i^{Request} = \pi_i^{Data} = 1/(2H)$. Assuming a loss rate asymmetry $a = 9$, then $\pi_i^{Request} = (1/H)(1/(9+1))$ and $\pi_i^{Data} = (1/H)(9/(9+1))$. Accordingly, $\overline{d}$ will be equal to $(1/10)r + (9/10)(r(H+1)/(2H))$.

In Fig. 2, we plot the expected latency with in-network caching as a function of path length normalized to the case without in-network caching. In this experiment, we assume that loss occurrences are uniformly distributed in the path hops and we examine two different asymmetries of unidirectional path loss rate, namely, when a is equal to 1 and 20, respectively. The results show that the expected retransmission latency can be significantly reduced by exploiting the on-path caches for paths with two or more hops. The gains are enhanced further when the loss probability of Data requests is increased compared to the Requests. We finally observe that the gains are not significantly affected when the path length increases beyond 6 hops.

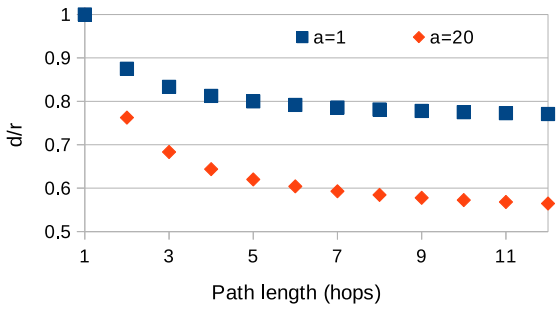

Fig. 2. The expected retransmission latency normalized to the case without caching (d/r) as a function of path length (in hops). The distribution of loss occurrences is uniform and a is equal to 1 or 20.

We also examine the case where the access links present higher loss rate that the in-network links, a situation that is likely to occur in error-prone wireless access networks [22] or lossy inter-domain links [23]. In Fig. 3 we plot the expected latency when most losses occur in the receiver's access link, namely, when π_1 is 2, 5 and 10 times higher than π_k, where $H \geq k > 1$. Notice that the sender's access link behavior is not changed, thus simulating the case where the content provider is in the Internet, e.g., in a *Content Delivery Network* (CDN) [24]. The relation between latency reduction and path length persists, but the lossier access links offer higher latency reduction. The result is not surprising since caching at the edges does maximize the reduction of retransmission latency.

Our model remains valid when links present different propagation latencies, by adapting the interpretation of i and H. Specifically, (1) is valid if i reflects the cumulative link latency from the receiver to the sender up to a node, and H refers to

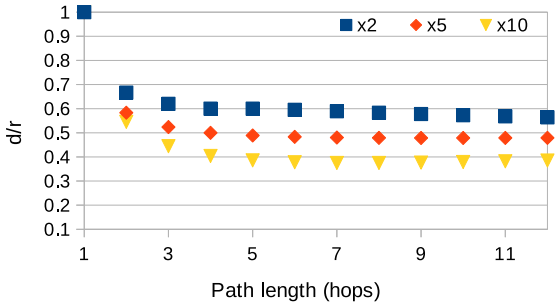

Fig. 3. The normalized expected latency reduction (d/r) offered by in-network caching as a function of path length (in hops) when the loss occurrences of the receiver's access link are 2, 5 and 10 times the loss occurrences of the other links.

the total latency from the receiver to the sender. For instance, in the setup in Fig. 4, if a Data is lost at the hop from R2 to R1, hence can be found cached at R2, then i and H are equal to 60 and 80 ms, respectively, thus offering retransmission latency of 60 ms, as suggested by (1).

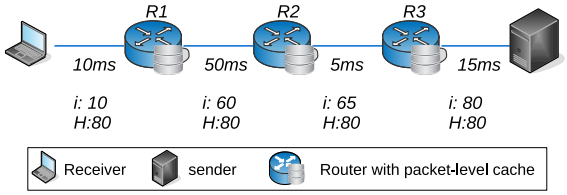

Fig. 4. Example of a path where links present different propagation latency. i is the cumulative link latency from the receiver to the sender and H is the total propagation latency from the receiver to the sender.

C. Traffic load reduction

The effect of in-network caching on traffic load reduction is twofold: first, it reduces the load of the senders and, second, it reduces the traffic footprint of the network. The estimation of sender load reduction is direct, being equal to the number of Data packets that are lost at least two hops away from the sender, since any lost packet will be returned by an on-path cache. In this work, we assume that the number of Data packets that are lost one hop away from the sender, which cannot be cached, is negligible, since senders are typically located at the core network, e.g., a CDN network, and losses are more likely near the access links due to the inter-domain connections [23].

The foreseen reduction of the traffic footprint, i.e., the amount of bytes that are forwarded in the network, follows the reduction of retransmission latency (Sect IV-B). Specifically, the retransmission latency analysis is based on the hops that the retransmitted Request and Data packets need to travel. This method is also appropriate for estimating the reduction of the number of packets that are forwarded in the network, since routing is symmetric and the packet size is fixed throughout the path. For instance, if a Data was lost at the 2^{nd} hop of a 5-hops path ($i = 2$ and $d = 5$), then the reduced traffic footprint would

be $2/5$ of the traffic footprint without in-network caching, thus reducing by 60% the retransmission cost in bandwidth.

Considering that the error rate of intracontinental Internet paths is found to be 0.3-3.5% in [25] and the results of Fig. 3, which yield up to 66% reduction, we estimate that traffic footprint reduction can reach roughly 2.5% when caches are used for retransmissions. The result is not impressive, however the gains come with no apparent cost; we just need to rethink the use of existing router buffers.

V. Feasibility

The requirements of in-network routers can exceed the capabilities of hardware memory modules when aiming at generic content-delivery, however they can become feasible when targeting only retransmissions. The retransmission mechanisms are deterministic modules that always emit a retransmission exactly when the loss is detected, as discussed in Section IV-A. Therefore, retransmitted Requests are bound to reach an in-network router a specific time after the related Data was delivered to that router, rendering useless the cached Data packets after this time. A retransmission-oriented cache does not need to store the most popular content items, it only needs to store the most recent Data packets for this specific time, hence we propose the exploitation of a FIFO cache eviction policy. However, we do consider the investigation of different cache policies as an important topic for future work.

We define as *caching life*, c, the time that a Data must be stored in the cache in order to hit a retransmission. This variable, that is equal to the period between the receipt of the Data and the receipt of the retransmitted Request at the caching router (Fig. 1), also defines the storage requirement of cache modules. In order to find the Data cached at the on-path router, the cache size has to be bigger than the product of c, and the throughput of the content router, T. The caching life is less than r, therefore it is a relatively small period. In the example of Fig. 1, $c = t_h - t_i$ which is equal to $(t_r + r/4) - (t_0 + 3r/4)$. We know that $t_r - t_0$ is the loss detection latency, $e = r$, hence $c < r$.

Modern commercial core routers offer up to 922 Tbps of aggregate switching capacity (by exploiting up to 1152 slots) [11], which account for roughly 13 Gpps traffic of Jumbo Ethernet sized frames. Ignoring any other processing latency, the cache access time can not be higher than 70 ps in order to support one cache lookup per packet in a serialized manner, which is highly challenging with current technology. The *pipelined* SRAM architectures, that divide the forwarding process in individual sub-processes and perform multiple steps in parallel, allow routers to forward one packet per SRAM access time [26], however the SRAM access latency is still blocking terabit operation. Such forwarding rates are available through distributed architectures, where the router functions are split among discrete physical line cards, thus allowing the parallel processing of one packet per line card [15]. In these setups, each line card has its own forwarding engine along with a copy of the routing table and two memory systems for packet buffering. Assuming 1152 line cards, the available time for forwarding each packet theoretically grows to 80 ns and 13 ns for Jumbo and standard Ethernet sized frames, respectively, thus being supported by SRAM and possibly by RLDRAM (0.4 and 15 ns access time, respectively) [5].

In [7], the authors propose an ICN-driven router design where name-based routing information is distributed among multiple line cards, however the distribution of cached storage is not discussed. Operating a packet-level cache in a distributed fashion is more challenging than accessing in parallel the relatively static routing tables. The stored content of an in-network cache is highly dynamic, thus inducing great overhead in order to keep the cache indexes of the line cards synchronized. Overall, we cannot rule out the feasibility of such a solution, but the problem is rather challenging. On the other hand, the retransmission-oriented caches do not need to keep synchronized cache indexes; each line card index can be autonomous, since retransmissions are bound to happen from the same receiver over the same line card. Being compatible with the distributed router architecture is considered a significant feature of retransmission-oriented caching, since the enables building high-speed core routers.

The estimation of the storage requirements of a retransmission-oriented cache is an equally complex problem. In particular, the capacity of the cache is defined by the aggregate switching capacity of the router, T, the required caching-time c (bounded by the connection's r) and the entry size of the cache index and cache entries. In particular the cache capacity, S, must be large enough to store all forwarded packets, T/p, within c, where p is the packet size. Assuming that the index to the cached packet, with size p_{index}, and the cached Data, with size p_{data}, are placed in the SRAM, the required memory allocation for each stored packet is $p_{index} + p_{data}$, hence:

$$S \geq (p_{index} + p_{data}) c T / p$$

We preliminarily investigate the cache capacity requirements in Fig. 5, where we plot the size of the retransmission-oriented cache in the SRAM as a function of c which is slightly less than the path latency. As suggested by several studies, we assume that the index in SRAM is a 40 bit hash of the content name [5], [8]. In order to implement a FIFO policy, thus being able to keep in cache the most recent packets, a 32 bit pointer (to the previous packet entry) must be introduced per packet, thus requiring in total 72 bits per index entry. Finally, assuming that the cache is implemented as a HashTable, the cache storage per entry is equal to the packet size, hence $p_{data} = p$. We focus our study on the router specification presented in [11], where a slot equipped with a line card with individual processing and memory resources can deliver from 140 Gbps up to 400 Gbps line-rate throughput. We consider standard and Jumbo Ethernet sized frames, but we demonstrate only the first since packet size is not found to significantly affect the result. Finally, we present the y axis in logarithmic scale in order to more clearly depict the memory requirements for lower RTTs.

Our analysis shows that the relation of path RTT and cache capacity is linear. In case of 400 Gbps, it is revealed that an SRAM of 210 Mb capacity cannot support retransmissions for any given c, thus rendering the design impractical. An

RLDRAM of 2 Gb capacity can support retransmission-oriented caching when $c < 5ms$, hence is also considered barely practical; only edge-routers could be considered. If Jumbo Ethernet sized frames are assumed (in order to loosen the access time constraint), then the use of a DRAM memory of 10 GB, or 80 Gb, capacity and 55 ms access time [5] could be realistic. Then, a core router with a DRAM cache could store packets for hundreds of milliseconds. The feasibility is better in case of 140 Gbps line-rate throughput, but the DRAM can keep cached content for roughly 200 ms even in case of 400 Gbps, thus being able to support a wide range of network setups.

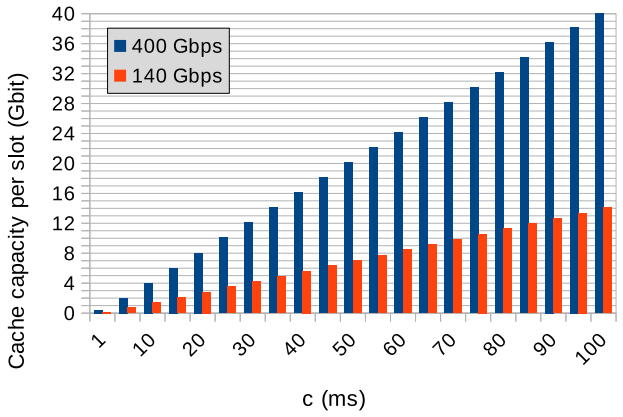

Fig. 5. The per slot cache capacity requirement of a retransmission-oriented in-network cache router as a function of caching-life, c, for 140 Gbps and 400 Gbp line-rate throughput.

A. Discussion

Our evaluation reveals that the feasibility of retransmission-oriented ICN cache-routers is challenged by the memory and processing overhead that the in-network core routers present. At the same time, our analysis validates that the gains are increased as the caches are pushed closer to the receivers, thus re-questioning the effectiveness of in-network caching compared to edge-caching. Regarding retransmission-oriented caches, edge-caching seams more promising than ever, since the cache hits are based on subsequent Request transmissions from the same receiver that typically follow the same route, instead of being based on subsequent Requests from different users that are more likely to meet at a central in-network core router. Thereafter, the probability to find the cached content at an edge-router is based on the ability of that router to store Data packets for long enough, being irrelevant to the content popularity and centrality of the router. On top of that, edge-routers present significantly lower forwarding rates than core routers, thus being able to store content longer with the same storage capacity.

The distribution of loss occurrences in the network plays a critical role in the cache placement policy. For edge-caches to thrive, the access links must be very error-prone while the core network must present minimum error-rate, hence the Data packets can reach safely to the edge-router and, then, be lost. The opposite situation can be in favor of the in-network caches which can support losses that occur in the core network too, thus enhancing the overall reduction of retransmission latency collectively. In general, we expect the type of the network (and related pattern of losses) to have substantial impact on the performance of retransmission-oriented caching, thus constituting a important topic for future work.

VI. Conclusions and Future Work

In this paper we investigated the operation of ICN in-network packet-level caching that targets only retransmissions. First, we explored the envisioned gains, revealing that the expected retransmission latency can be reduced by up to 40%, thus offering considerably faster reaction to network loss. Then we conducted a feasibility assessment, based on modern router specifications, which showed that the operation of a retransmission-oriented cache module on a core router can be feasible, assuming a distributed router design and Jumbo Ethernet size frames. Our results suggest that, under these conditions, a terabit-per-second core router can keep cached packets for hundreds of milliseconds, thus being able to respond to retransmissions in a wide range of network setups. Therefore, we conclude that with ICN, in-network packet-level caching for retransmissions is a valuable error-control component.

There are many different directions that this work could be extended towards. First, an experimental evaluation of the design needs to be conducted in order to validate the accuracy of our model. Second, novel caching strategies that are specifically designed for addressing retransmissions can be developed in order to enhance the envisioned gains. Then, the mutual exploitation of router fast memory by the forwarding and the caching functions needs to be addressed; this approach could significantly improve the performance and impact the dimensioning of ICN routers. Finally, the conclusions of this work could encourage studies on the relation of user experience and loss recovery latency, thus further motivating research on retransmission-oriented ICN caching.

Acknowledgment

This research was supported by the Research Center of the Athens University of Economics and Business.

References

[1] G. Xylomenos, C. N. Ververidis, V. A. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. V. Katsaros, and G. C. Polyzos, "A survey of information-centric networking research," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 2, pp. 1024–1049, 2014.

[2] A. Ioannou and S. Weber, "A survey of caching policies and forwarding mechanisms in information-centric networking," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 4, pp. 2847–2886, 2016.

[3] S. K. Fayazbakhsh, Y. Lin, A. Tootoonchian, A. Ghodsi, T. Koponen, B. Maggs, K. Ng, V. Sekar, and S. Shenker, "Less pain, most of the gain: Incrementally deployable ICN," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4, pp. 147–158, 2013.

[4] W. K. Chai, D. He, I. Psaras, and G. Pavlou, "Cache less for more in information-centric networks (extended version)," *Computer Communications*, vol. 36, no. 7, pp. 758–770, 2013.

[5] D. Perino and M. Varvello, "A reality check for content centric networking," in *Proceedings of the ACM SIGCOMM workshop on Information-centric networking*, 2011, pp. 44–49.

[6] A. Detti, L. Bracciale, P. Loreti, G. Rossi, and N. B. Melazzi, "A cluster-based scalable router for information centric networks," *Computer Networks*, vol. 142, pp. 24–32, 2018.

[7] M. Varvello, D. Perino, and J. Esteban, "Caesar: A content router for high speed forwarding," in *Proceedings of the ICN Workshop on Information-Centric Networking*, 2012, pp. 73–78.

[8] G. Rossini, D. Rossi, M. Garetto, and E. Leonardi, "Multi-terabyte and multi-gbps information centric routers," in *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*. IEEE, 2014, pp. 181–189.

[9] Y. Thomas, G. Xylomenos, C. Tsilopoulos, and G. C. Polyzos, "Object-oriented packet caching for ICN," in *Proceedings of the 2nd ACM Conference on Information-Centric Networking*, 2015, pp. 89–98.

[10] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, K. Claffy, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang, "Named data networking," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 66–73, 2014.

[11] "Cisco Carrier Routing System–Compare Models," Cisco Systems, 2020. [Online]. Available: https://www.cisco.com/c/en/us/products/routers/carrier-routing-system/models-comparison.html

[12] S. Iyer, R. R. Kompella, and N. McKeown, "Designing packet buffers for router linecards," *IEEE/ACM Transactions On Networking*, vol. 16, no. 3, pp. 705–717, 2008.

[13] J. Garcia-Vidal, M. March, L. Cerdà, J. Corbal, and M. Valero, "A DRAM/SRAM memory scheme for fast packet buffers," *IEEE Transactions on Computers*, vol. 55, no. 5, pp. 588–602, 2006.

[14] R. B. Mansilha, L. Saino, M. P. Barcellos, M. Gallo, E. Leonardi, D. Perino, and D. Rossi, "Hierarchical content stores in high-speed ICN routers: Emulation and prototype implementation," in *Proceedings of the 2nd ACM Conference on Information-Centric Networking*, 2015, pp. 59–68.

[15] "The Evolution of High-end Router Architectures," White Paper, Cisco Systems, 2001.

[16] Y. Thomas, "Multipath internet transport," Ph.D. dissertation, Athens University of Economics and Business, 2018.

[17] M. Allman, V. Paxson, W. Stevens *et al.*, "TCP congestion control," Internet Requests for Comments, RFC Editor, RFC 2581, April 1999.

[18] Y. Thomas, C. Tsilopoulos, G. Xylomenos, and G. C. Polyzos, "Accelerating file downloads in publish subscribe internetworking with multisource and multipath transfers," in *WTC 2014; World Telecommunications Congress 2014*. VDE, 2014, pp. 1–6.

[19] M. Luby, L. Vicisano, J. Gemmell, L. Rizzo, M. Handley, and J. Crowcroft, "The use of forward error correction (FEC) in reliable multicast," RFC 3453, December, Tech. Rep., 2002.

[20] V. Paxson, M. Allman, J. Chu, and M. Sargent, "Computing TCP's retransmission timer," Internet Requests for Comments, RFC Editor, RFC 2988, November 2000.

[21] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions on networking*, vol. 1, no. 4, pp. 397–413, 1993.

[22] J.-M. Vella and S. Zammit, "A survey of multicasting over wireless access networks," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 2, pp. 718–753, 2012.

[23] K. Mochalski, J. Micheel, and S. Donnelly, "Packet delay and loss at the Auckland Internet access path," in *Passive and Active Measurement Workshop, Fort Collins, Colorado USA*. Citeseer, 2002.

[24] A. Passarella, "A survey on content-centric technologies for the current internet: CDN and P2P solutions," *Computer Communications*, vol. 35, no. 1, pp. 1–32, 2012.

[25] M. S. Borella, D. Swider, S. Uludag, and G. B. Brewster, "Internet packet loss: Measurement and implications for end-to-end qos," in *Proceedings of the 1998 ICPP Workshop on Architectural and OS Support for Multimedia Applications Flexible Communication Systems*. IEEE, 1998, pp. 3–12.

[26] A. Basu and G. Narlikar, "Fast incremental updates for pipelined forwarding engines," *IEEE/ACM Transactions on Networking*, vol. 13, no. 3, pp. 690–703, 2005.