The impact of chunk size on Named Data Networking performance

Christos Natsis[†], Christos-Alexandros Sarros[†], Vassilis Tsaoussidis[*†]
[*] *Athena Research and Innovation Center*
[†] *Electrical and Computer Engineering Dept.*
Democritus University of Thrace
{*chrinats, csarros, vtsaousi*}*(at)ee.duth.gr*

***Abstract*—Internet usage has evolved during the years towards accessing content, regardless of its location. The Named Data Networking architecture was designed to satisfy that need and replace today's host-centric TCP/IP stack by placing named content at the core of the architecture. In this paper, we provide the first evaluation specifically targeting the impact of packet size in NDN performance. Our results show the impact of chunk sizes is far from negligible; it clearly affects NDN performance as measured by metrics such as throughput, delay, cache hits and cache entry lifetime. Attention should, therefore, be given to the selection of an appropriate chunk size in NDN deployments.**

***Index Terms*—Information-Centric Networking, Named Data Networking, ICN, NDN, network performance**

I. Introduction

The current TCP/IP Internet architecture was designed to meet the needs of a different era, which focused on the interconnection of fixed hosts. However, technological advancements led to the huge expansion of the Internet and a shift of its main use towards content dissemination and retrieval.

This led to new network architectures, such as Named Data Networking (NDN) [12], the origins of which can be traced back to the Information-Centric Networking (ICN) paradigm [9]. Although previous studies have, so far, evaluated various aspects of the NDN architecture, little attention has been given to the impact of packet size on NDN performance. In this paper, we aim to close this gap by conducting experiments specifically focused in this aspect.

Current packet size distributions on the Internet are concentrated around 2 main sizes (40KB and 1500 KB), according to previous research [8]. NDN traffic in large scale deployments may follow similar characteristics, as clusters of Interests and Data packets (which vary largely in size) are expected around their respective typical sizes. Data packets are typically larger than Interests, as they introduce a bigger overhead (even more than TCP/IP [1]) and carry more payload information; they are therefore responsible for most of the NDN traffic volume. Interest packets also yield less network traffic due to the NDN Interest suppression mechanism, which results in similar Interests arriving from different sources being aggregated at routers. Therefore, we focus this study on Data packets and their respective chunk size.

The maximum packet size in NDN is currently, by default, 8800KBs[1]. This may be sufficient for supporting typical traffic, however larger chunk sizes might be useful in certain circumstances. For instance, NDN deployments over Delay/Disruption-Tolerant Networks (DTNs) [4] may benefit from an increased packet size in order to decrease the total number of chunks needed to retrieve some content.

A decreased number of chunks means that fewer Interests need to be sent by the consumer, something especially desirable in such network conditions. First, more chunks will cause more PIT entries and significantly larger memory consumption, as those entries are expected to stay in the PIT for long periods of time (as DTNs involve delays in the orders of minutes or hours). Secondly, a consumer in a delay-/disruption- tolerant environment benefits greatly from requesting multiple chunks at a parallel fashion, as additional round trips of sequential requests and responses are very costly in these circumstances, incurring large additional delays. Therefore, the ability to encapsulate a object content into a single Data packet which can be retrieved by sending a single Interest, can make a large overall difference. Lastly, splitting a content object into several chunks which are encapsulated into different DTN bundles, increases the chance that a single chunk may be lost (as delivery is not guaranteed in DTNs). In some cases, this would mean that the whole content may be rendered useless. An example is service-centric networking over DTN [5], in which case the service cannot be executed unless the entire docker container has been transferred. All of the aforementioned problems may be alleviated by increasing NDN's default chunk size.

As larger sizes might also be desirable in other deployments, we focus this study on the impact of an increased chunk size and target more typical, fixed network deployments. We shed light on its effect on several network metrics such as Throughput, Delay, Cache Hit Ratio and Cache Entry Lifetime. In the rest of the sections, we present previous work on the subject, our methodology and the results of our experiments.

This research was partly funded by the Greek State Scholarships Foundation (IKY) action "Supporting human research resources through doctoral research", Operational Programme "Developing Human Resources, Education and Lifelong Learning", 2014–2020

[1] https://github.com/named-data/ndn-cxx/blob/master/ndn-cxx/encoding/tlv.hpp

II. Related Work

No previous peer-reviewed work has, to the best of our knowledge, specifically studied the influence of packet size on NDN performance. Most relevant results are scattered across several, unrelated, papers that focus on other primary targets.

The authors of [1] model the impact of different payload sizes to find the optimal chunk size. They develop a model linking chunk size, loss rate and Goodput-to-Throughput ratio (G2T) and subsequently develop a mechanism aiming to dynamically adjust the chunk size depending on the link loss ratio to achieve the optimum G2T. Their model and simulations indicate that, when NDN operates over lossless links or reliable lower-level protocols, greater chunk sizes result in better performance (with respect to G2T). When NDN is deployed over lossy links or unreliable lower-level protocols, there exists an optimum chunk size which depends on the loss rate - either too small or too large chunk sizes can cause G2T to degrade. However, the authors do not include other metrics in their evaluation (such as delay or cache hit ratio, on which chunk size can also have an effect). Another report [10] which analyzed the contribution of different NDN parameters (Content Store Capacity, Content Name Length, Packet Size, Content Diversity) to the overall network throughput, concluded that packet size had the largest impact on system performance.

In paper [11], experiments were conducted with variable NDN packet size for two different scenarios (caching/non-caching). 32 and 16 client-server pairs were used for traffic generation, respectively, and the throughput was measured for different Data payload sizes. Their results showed that, for larger packet sizes, throughput increased. However, in their experiments packet rates were the same for all payload sizes, contrary to our setup in which they were adjusted. Therefore, the fact that data rates were increasing while payload size increased was of no surprise. Of relevance to our work is the fact that the peak throughput achieved in the 'all packet shared' scenario (caching-enabled), was not only much bigger than the no-caching scenario, but that the difference between the peak throughput achieved by smaller packet sizes and the one achieved by larger ones, is more pronounced. This reinforces the results in the present paper, where caching was included, and correlates well with our findings.

Approaching the issue from a different angle, paper [2] evaluates server-side NDN performance, focusing on the signature generation overhead on Data packets. The authors set up four relevant experiments: In the first two, the receiver cannot retrieve packets from cache; packet sizes were set to 1024, 2048, 4096 and 8192 octets. In the first experiment, Data packets were generated using a simple SHA-256 hash and packet throughput decreased from 9000 packets per second (1024 payload size) to less than 8000 packets per second (for the 8192 payload case), a change around the -10% mark. In the second experiment an RSA digital signature was used; packet throughput also decreased as the packet size increased, but the decrease was much less noticeable. However, in both cases the packet throughput decrease was more than counterbalanced by the fact that the payload size was increasing, therefore the total 'byte throughput' significantly increased when using bigger payload sizes.

In the next two experiments, the authors evaluated the NFD cache performance throughput - the Data packets were received from client-side NFD cache in one experiment and server-side cache, on the other. In both cases, the general trend was that packet throughput decreased as payload size increased (by around 5%), results which are consistent with the previous experiments. Based on their results, the authors concluded that the PIT and FIB lookup process on the client-side NFD has a high price performance-wise.

The authors of [6] take a different approach. They do not assume that Data packet sizes are known or predictable and evaluate their proposed congestion control scheme by varying the packet size during the simulations. Their results show that, in their setup, when the Data chunk size suddenly increases, the router queues fill up and consumers will experience a temporary higher delay - proportional to the change in Data size - before the congestion control algorithm reduces the cwnd and the queue starts draining.

The authors of [3] measure the computation overhead of authentication mechanisms for two different packet sizes: 1500 bytes and 9000 bytes (corresponding to the standard Ethernet and the jumbo Ethernet MTU, respectively). Their results show that bigger Data chunk sizes reduced the time during which the data authentication credentials are prepared at the content producer ('generation time') for their manifestation mechanism. This is explained by the fact that a single manifest could hold more hashes and fewer manifests were needed, overall. Larger Data chunk sizes also reduced the communication overhead in some mechanisms, as the digital signature and hash values introduce a smaller overhead on each Data packet.

Lastly, researchers in [7] focus on an IoT environment, in which they experiment with two different Data sizes during their evaluation of NDN-RIOT (100/196 bytes). When a packet is fetched from the producer, both packet sizes result in a similar Round Trip Time (280ms and 286ms, respectively). The extra 6ms delay is attributed to the fact that the 196-byte packet is fragmented into two pieces at the producer, which is not the case for 100-byte packets. However, when the consumer fetches data from its local cache, no fragmentation is required for either packet size.

III. NDN Overview

The Named Data Networking architecture (NDN) is the most prominent Information-Centric Networking (ICN) implementation. It is a receiver-driven network layer protocol, whose aim is to network named data. Human-readable content names replace IP host addresses and form the centerpiece of the architecture. Some of its main features include in-network caching and name-based forwarding and routing.

Communication in NDN is performed using two types of packets: Interest and Data packets. NDN Consumers initiate

communications by sending an Interest packet, which corresponds to a unique content object. The Interest packet is forwarded to a Producer which can satisfy the Interest and the Producer responds to the Interest with a Data packet containing the requested content object. Data packets can be received either from NDN Producers, or from intermediate caches (e.g. router caches), if the Interest packet arrives at a node where the requested content object has been previously cached.

To achieve the above, three main structures are used by NDN: the Pending Interest Table (PIT), the Forwarding Interest Base (FIB) and the Content Store (CS). The PIT contains information about all the Interests that have been forwarded but not been satisfied and their corresponding interfaces ('faces'). The FIB contains forwarding information and is used to forward Interests for certain prefixes towards faces that are most likely to return the corresponding Data packets. The CS serves as each node's cache and stores the node's received Data packets according to the selected cache replacement policy.

IV. Methodology

In our experiments a dumbbell topology was deployed consisting of 6 nodes: 2 Consumers, 2 Producers and 2 Routers. The Consumers were located on one side of the network and connected to one Router, while the Producers are located on the other side and connected to a different router; the two routers are connected via a backbone link.

The Named Data Networking Simulator (ndnSIM) was used for the experiments. The following simulator parameters were used: Point-to-point interconnection (P2P) links were set up between nodes and two scenarios were formed, depending on the link bandwidth: 'Congestion-Free' and 'Mild Congestion'. In the 'Congestion-Free' scenario, the Producer-Router and Consumer-Router links were set at 40 Mbps, while the backbone link (Rtr1-Rtr2) was set to 160Mbps. In the 'Mild Congestion' scenario, the backbone bandwidth was reduced to 60 Mbps while the other ones remained at 40Mbps.

The propagation delay on all P2P links was set at 10ms and kept constant throughout the experiments, while the cache size was set at 1000 content objects on each node, regardless of the size of each object. The ConsumerZipf application was used for Interest generation; the number different content names that could be selected by the application was set to 2000. Finally, the simulation time was set to 30 seconds.

During the simulations, we gradually increased the Data chunk size. To account for the payload increase, each time the chunk size was doubled, the application's Interest generation rate was halved. Thus, the total volume of data sent in all experiments was the same; this way we factored out the impact of payload size on network throughput. For each one of the different chunk size values, we repeated the experiment using a different cache replacement policy and compared their results against each other. Overall, the following cache replacement policies were used: First In First Out (FIFO), Least Frequently Used (LFU), Least Recently Used (LRU), and Random Replacement (RANDOM).

Packet Size [KB]	Interests [per second]
5	900
10	450
20	225
40	112
80	56
120	42
160	28

TABLE I
Chunk size-Interest rate correlation

To assess performance, we use the following metrics: Throughput, Delay, Cache Hit Ratio, Cache Entries Lifetime (more specifically, their average values). Measurements regarding the first and last metrics were taken in the consumer-side Router, while the rest were taken in the consumer nodes.

V. Results

In this section, we present the results of our experiments. In each figure, we use a different metric to assess different aspects of NDN performance. Our overall goal is to assess the impact of various Data chunk sizes on different aspects of NDN performance. In all experiments, content requests by the Consumers follow a Zipf-Mandelbrot distribution to approximate real-world user behaviour.

A. Cache Hit Ratio

In Figure 1, each consumer's average Cache Hit Ratio can be observed with respect to different Data packet sizes. Based on the figure, we can infer that larger Data packet sizes incur a lower Cache Hit Ratio. This holds true in all cases, regardless of the cache replacement policy being chosen.The aforementioned results can be explained by the fact that our experiments run until a fixed amount of data was retrieved (i.e. 4.5 MB of data). Therefore, increasing the Data chunk size meant that less Interests had to be sent to retrieve the specified amount of data. Less Interests meant that less Data packets were returned which, in turn, meant less overall cache entries during the experiment. For instance, 900 Interests were sent overall in the case of 5 KB packet size, while only 28 Interests were sent in the case of 160 KB packet size. This meant that, e.g. for 5KB packets, when the last Interest of each sub-experiment was received by the cache, there were up to 899 entries stored in the Content Store. Those 899 entries could potentially satisfy the Interest, while in the case of 160KB packet size there were only 27 entries. Therefore, self-evidently, the chance that a subsequent Interest could be answered by previously-received content that was stored in the cache, was less for bigger chunk sizes, which eventually resulted in a lower cache hit ratio.

What can also be inferred by Figure 1, is the fact that setting the right cache replacement policies seems to make more of a difference (with respect to the Cache Hit Ratio) when using smaller Data packet sizes, up to 40 KB. In this range, RANDOM replacement yields the worst results, while LFU replacement yields the best ones. However, when increasing the packet size to 80-160 KB, the difference seems to be

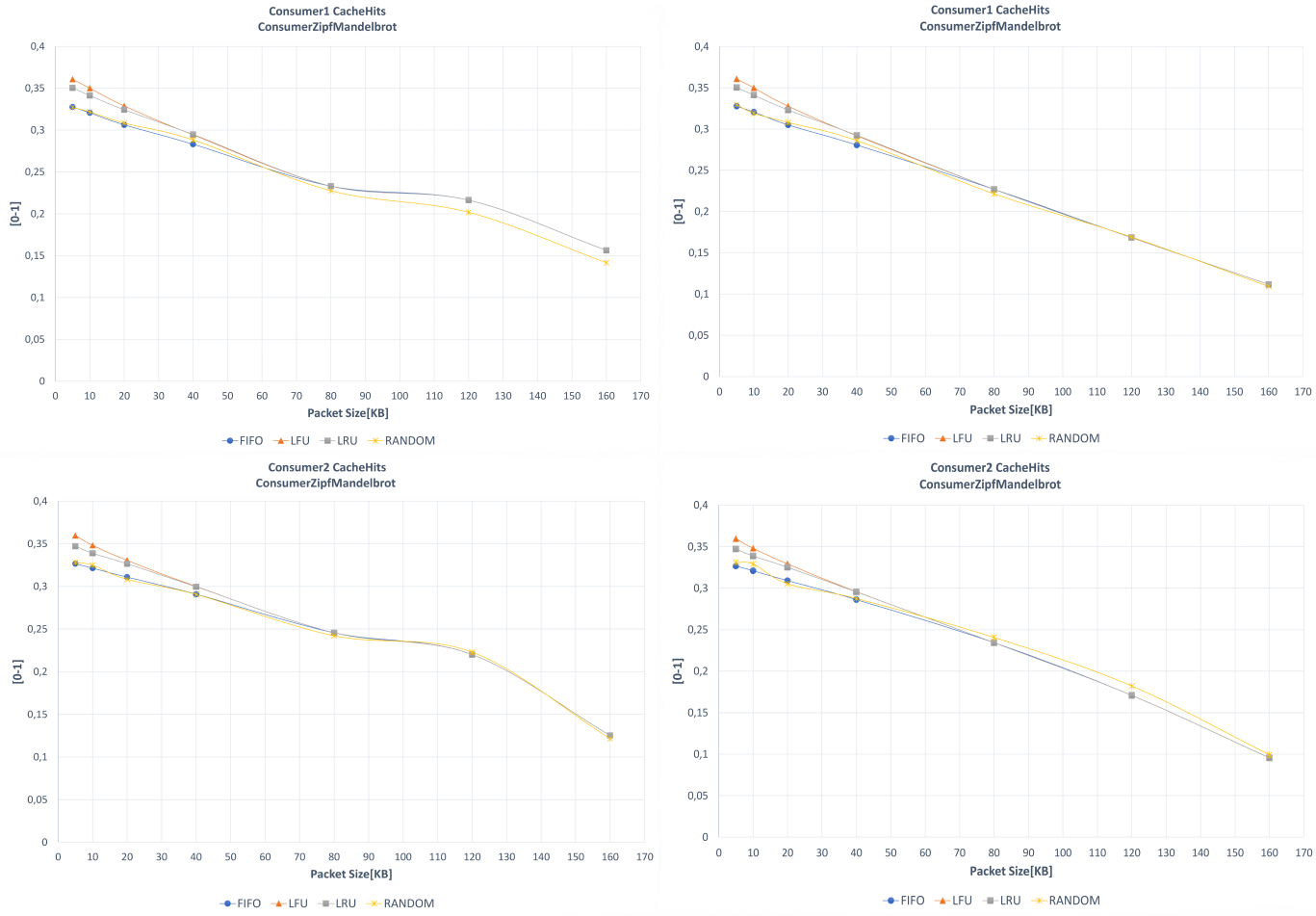

Fig. 1. Average Cache Hit Ratio (left: no congestion, right: mild congestion)

minimized and choosing a more sophisticated policy over random replacement seems to make less of a difference.

B. Average Cache Entries Lifetime

In Figure 2, the Cache Entries Lifetime can be observed, with respect to different Data packet sizes. On the left side, the results regarding Scenario 1 (no-congestion) are depicted while on the right side, the results regarding Scenario 2 (mild congestion) are depicted. In both cases, there is a notable difference between using a random cache replacement policy and using more sophisticated ones, as cache entries stay significantly longer in the Content Store when using either LRU, LFU or FIFO (an 100% increase can be observed when compared to the time measured for the RANDOM policy).

Furthermore, it also holds true in this case that the difference between LRU, LFU and FIFO policies is more pronounced when using smaller packet sizes, with the LFU policy outperforming the rest. Again, as packet sizes increase, the performance difference between the caching policies decreases, as was the case concerning the Cache Hit Ratio metric.

Of note are also two things: a) the fact that the entry lifetimes do not differ significantly between Scenario 1 and Scenario 2 when using the same packet size and b) the fact that the entry lifetimes increase as packet sizes increase. This can be explained by the same cause as the one described in the previous sub-section: in our setup, larger Data sizes mean that fewer packets are being transmitted in an experiment. Therefore, there is less need for replacing the cache entries during the simulation, resulting in larger lifetimes for each entry.

C. Average Throughput

In Figure 3, the average throughput can be observed with respect to the two scenarios. While there is no significant difference for packet sizes up to 80 KB, a deviation from the previous trend can be seen for 120 KB and 160 KB packet sizes. When using these sizes, the throughput is markedly larger in the non-congested case. It can therefore be inferred that deployments that use larger packet sizes might be more susceptible to the effects of congestion that the ones using smaller Data packets, resulting in degraded performance.

Of note is also the fact that, once again, the performance difference between the various caching policies is more pronounced for smaller packet sizes. For packet sizes larger than

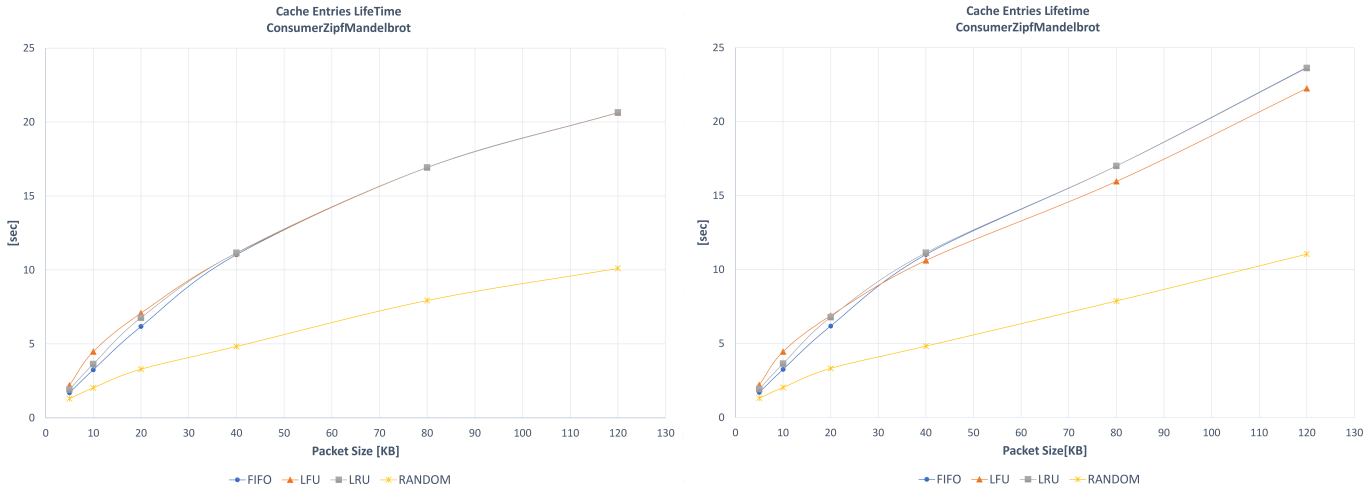

Fig. 2. Cache Entries Lifetime (left: no congestion, right: mild congestion)

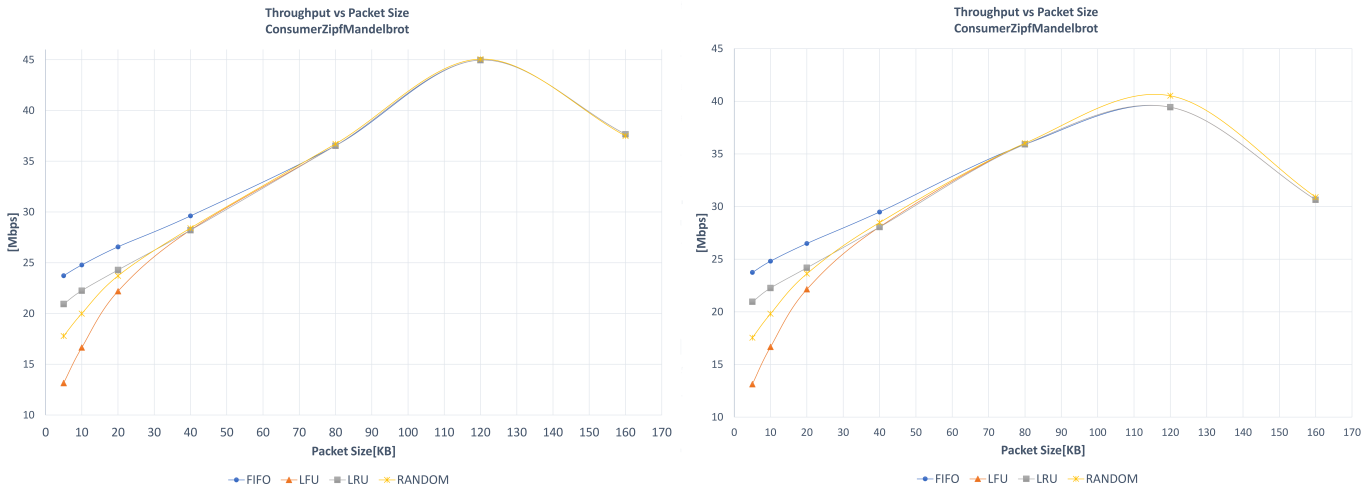

Fig. 3. Throughput (left: no congestion, right: mild congestion)

80 KB, the performance is the same regardless of the cache replacement policy being chosen. In the smallest range of packet sizes (5-40 KB), FIFO outperforms the rest of the policies with respect to the average throughput being achieved. The worst performance is, counter-intuitively, achieved not by random replacement but by LFU. Lastly, we can infer that larger packet sizes result in increased average throughput. More specifically, increasing the packet size from 5 to 120 KB (24 times), results in a throughput increase ranging from 1.6 (FIFO, congestion) to 3.5 times (LFU, no congestion).

D. Delay

In Figure 4, the average packet delay for each packet size is being shown. The average delay per packet increases when experimenting with larger packet sizes. This is to be expected, as larger packet sizes incur larger processing delays in intermediate routers and consumers. Furthermore, we can once more notice the fact that the performance of the different cache replacement policies converges for larger packet sizes. For smaller packet sizes, LFU performs better (smaller delays) while FIFO performs the worst (larger delays).

Furthermore, there is a notable difference regarding the average delays between the non-congested and congested scenario, which is more pronounced for larger packet sizes. For the 160KB case, we can see the average delay doubles when congestion is incurred (Scenario 2), when compared against the corresponding average delay in Scenario 1.

VI. Conclusions

In this paper, we performed experiments to quantify the impact of packet size on NDN performance. Our results show that using larger chunk sizes is feasible, but offers some trade-offs. For the same amount of data being retrieved, bigger data chunks mean that cache replacement policies assert less impact on network performance. Furthermore, network throughput can increase, as bigger chunks entail less overhead per chunk. However, the average delay also increases, due to an increased processing overhead. Therefore, using bigger

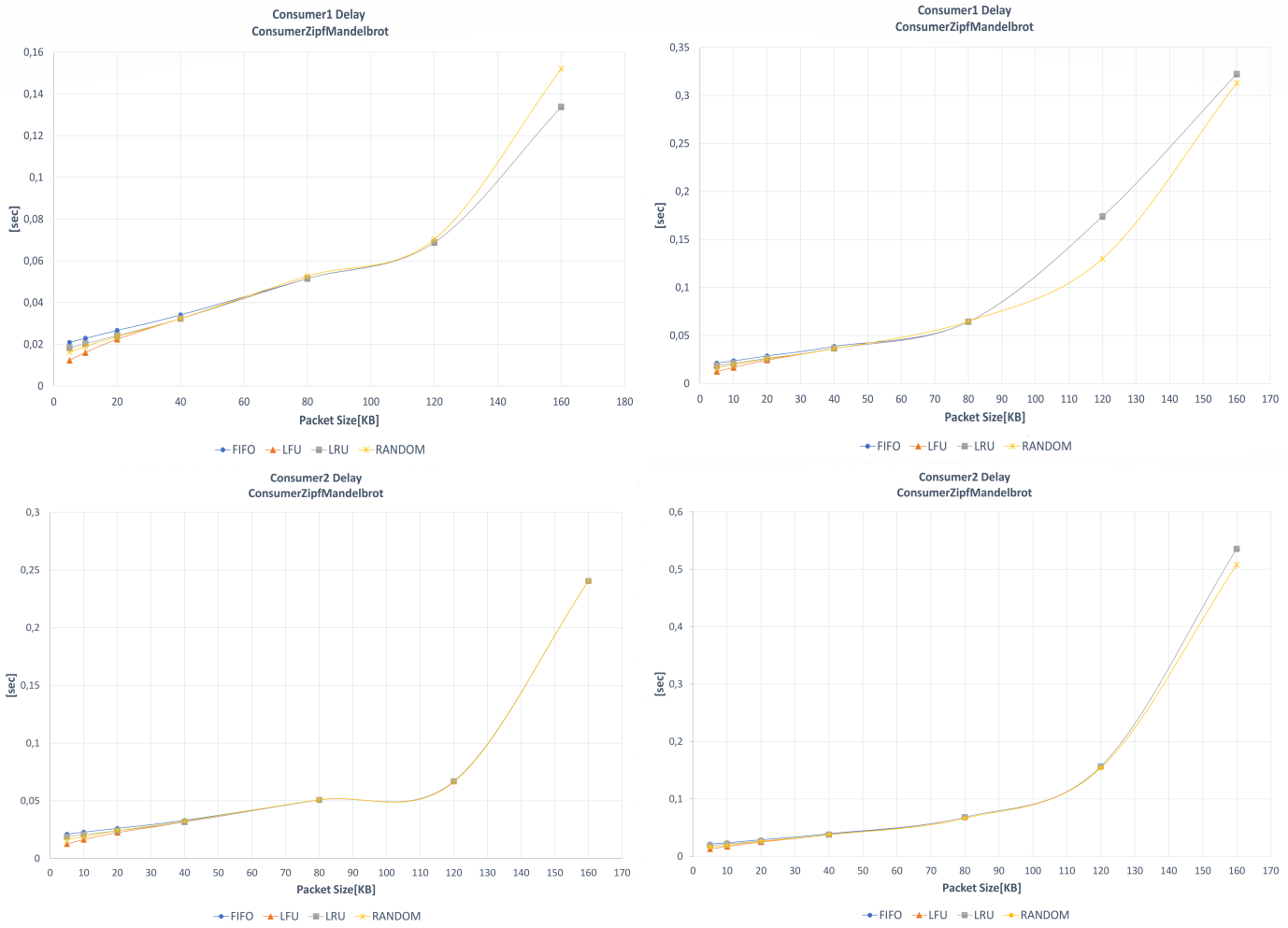

Fig. 4. Average Delay (left: no congestion, right: mild congestion)

chunk sizes seems to be better suited for certain use cases, such as bulk data transfers, while smaller chunk sizes seem to be more suitable for delay-sensitive applications.

References

[1] Xiaoke Jiang, Jun Bi, and Lifeng Lin. Modelling the optimum chunk size for efficient communication over named data networking. Technical report, Tsinghua University. http://netarchlab.tsinghua.edu.cn/~shock/optimum-chunksize.pdf (accessed July 2020).

[2] Xavier Marchal, Thibault Cholez, and Olivier Festor. Server-side performance evaluation of ndn. In *Proceedings of the 3rd ACM Conference on Information-Centric Networking*, ACM-ICN '16, page 148–153, New York, NY, USA, 2016. Association for Computing Machinery.

[3] T. Refaei, M. Horvath, M. Schumaker, and C. Hager. Data authentication for ndn using hash chains. In *2015 IEEE Symposium on Computers and Communication (ISCC)*, pages 982–987, 2015.

[4] C. Sarros, S. Diamantopoulos, S. Rene, I. Psaras, A. Lertsinsrubtavee, C. Molina-Jimenez, P. Mendes, R. Sofia, A. Sathiaseelan, G. Pavlou, J. Crowcroft, and V. Tsaoussidis. Connecting the edges: A universal, mobile-centric, and opportunistic communications architecture. *IEEE Communications Magazine*, 56(2):136–143, 2018.

[5] C. Sarros, A. Lertsinsrubtavee, C. Molina-Jimenez, K. Prasopoulos, S. Diamantopoulos, D. Vardalis, and A. Sathiaseelan. Icn-based edge service deployment in challenged networks. In *Proceedings of the 4th ACM Conference on Information-Centric Networking*, ICN '17, page 210–211, New York, NY, USA, 2017. Association for Computing Machinery.

[6] Klaus Schneider, Cheng Yi, Beichuan Zhang, and Lixia Zhang. A practical congestion control scheme for named data networking. In *Proceedings of the 3rd ACM Conference on Information-Centric Networking*, ACM-ICN '16, page 21–30, New York, NY, USA, 2016. Association for Computing Machinery.

[7] Wentao Shang, Alex Afanasyev, and Lixia Zhang. The design and implementation of the ndn protocol stack for riot-os. In *2016 IEEE Globecom Workshops (GC Wkshps)*, pages 1–6. IEEE, 2016.

[8] Rishi Sinha, Christos Papadopoulos, and John Heidemann. Internet packet size distributions: Some observations. *USC/Information Sciences Institute, Tech. Rep. ISI-TR-2007-643*, pages 1536–1276, 2007.

[9] George Xylomenos, Christopher N Ververidis, Vasilios A Siris, Nikos Fotiou, Christos Tsilopoulos, Xenofon Vasilakos, Konstantinos V Katsaros, and George C Polyzos. A survey of information-centric networking research. *IEEE communications surveys & tutorials*, 16(2):1024–1049, 2013.

[10] Haowei Yuan and Raj Jain. Performance measurement of content distribution in named data networking. Technical report, Washington University in St. Louis. https://www.cse.wustl.edu/~jain/cse567-11/ftp/ndn.pdf (accessed July 2020).

[11] Haowei Yuan, Tian Song, and Patrick Crowley. Scalable ndn forwarding: Concepts, issues and principles. In *2012 21st International Conference on computer communications and networks (ICCCN)*, pages 1–9. IEEE, 2012.

[12] Lixia Zhang, Alexander Afanasyev, Jeffrey Burke, Van Jacobson, KC Claffy, Patrick Crowley, Christos Papadopoulos, Lan Wang, and Beichuan Zhang. Named data networking. *ACM SIGCOMM Computer Communication Review*, 44(3):66–73, 2014.