Analyzing the Performance of ICN Forwarders on the Wire

Adam Drescher
adrescher@wustl.edu
Washington University in St. Louis

John DeHart
jdd@wustl.edu
Washington University in St. Louis

Jyoti Parwatikar
jp@wustl.edu
Washington University in St. Louis

Patrick Crowley*
pcrowley@wustl.edu
Washington University in St. Louis

ABSTRACT

Information Centric Networking (ICN) is growing in both popularity and maturity. Two highly-related architectures under the ICN umbrella, Content Centric Networking (CCNx) and Named Data Networking (NDN), received significant effort to improve their forwarding performance. Despite this focus, little work has been done to evaluate ICN forwarders in a comprehensive and rigorous manner. Furthermore, the preexisting literature in IP can only apply broadly due to the substantial differences between the architectures.

In this paper, we provide a methodology to analyze the performance of ICN forwarders. Our testing methodology has two key focuses: (i) packet processing performance is the primary metric of exploration, as bytes are usually cheap; and (ii) the PIT, FIB, and Content Store are the primary structures to probe when considering performance impact. With these focuses in mind, we present a series of behavioral microbenchmarks that can probe the performance of CCNx/NDN forwarders in a rigorous way. To show the efficacy of these experiments, we apply them to the reference forwarders of the CCNx and NDN architectures, Metis and NFD, giving us a careful understanding of their performance characteristics. Additionally, these microbenchmarks should readily apply to high performance forwarders in the space.

CCS CONCEPTS

• **Networks** → **Network architectures**; **Network measurement**; *Routers*.

KEYWORDS

Named Data Networking, Content Centric Networking, Forwarder Performance Analysis, ICN Packet Processing

ACM Reference Format:
Adam Drescher, John DeHart, Jyoti Parwatikar, and Patrick Crowley. 2020. Analyzing the Performance of ICN Forwarders on the Wire. In *7th ACM Conference on Information-Centric Networking (ICN '20), September 29-October 1, 2020, Virtual Event, Canada.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3405656.3418722

*Also with Cisco Systems, Inc..

ICN '20, September 29-October 1, 2020, Virtual Event, Canada

ACM ISBN 978-1-4503-8040-9/20/09…$15.00
https://doi.org/10.1145/3405656.3418722

1 INTRODUCTION

Information Centric Networking remedies much of the inflexibility of the current architecture of the Internet, and provides a rich abstraction that has prompted many new areas of exploration. However, this paradigm is not without its own unique challenges. Scalability is a key challenge when in-network forwarding decisions are based on content names. Indeed, significant research effort has targeted how to forward named content in a high performance manner [1, 9, 13, 15–17, 21]. However, there is little prior work on the proper way to evaluate the performance of ICN forwarders. Because of the differences in forwarding architecture, the space of exploration is quite different from IP-based experimental methodologies. ICN forwarders need an ICN-focused experimental methodology.

In this work, we aim to provide first steps towards a rigorous experimental methodology for ICN forwarding performance. First, we discuss practical concerns of testing in ICN. Then, as our primary contribution, we present a set of behavioral microbenchmarks that interrogate the packet processing capabilities of the core data structures in an ICN forwarder: the PIT, FIB, and Content Store. These microbenchmarks are designed for running on the wire, as they gather their performance measurements without the need to instrument the target forwarder's code. Last, we apply this methodology to NFD and Metis, the reference forwarders for NDN and CCNx, in order to show the microbenchmark's ability to reveal complex behavior in the forwarder's packet processing. These are not bleeding edge performance numbers, as even for the ICN landscape the performance of these forwarders are quite low—however, they do reveal the performance of forwarders that many people use. More importantly, these microbenchmarks should be readily applicable to high performance ICN forwarders.

2 BACKGROUND

We designed these behavioral microbenchmarks to reveal the packet processing performance of ICN forwarders, focusing on the core data structures as potential bottlenecks. As a vehicle for experimentation, we apply these microbenchmarks to the reference forwarders for NDN and CCNx, NFD and Metis respectively. To understand the importance of these data structures, we need a high-level understanding of the forwarding architectures of NFD and Metis.

Both NFD and Metis follow the same key steps. Every forwarding operation takes place on the name of data/requested data. When an Interest packet is received, they insert a record of it into the

Pending Interest Table (PIT). The PIT is a catalog of what requests are in flight, and are the breadcrumbs that data packets follow on the return path. Next, they check the Content Store to see if the requested data has already been cached in the forwarder. If the data is in the Content Store, it can be sent immediately without forwarding the request further. If it is not, they check the Forwarding Information Base (FIB) to determine what interface to forward the request out. If there are multiple interfaces available for that prefix, forwarding strategy is the component that selects which interfaces to actually use. When a Data packet is received, a PIT lookup is performed to ensure the data is asked for. If it is, the forwarder inserts the data into the Content Store, and then forwards the data along the interface(s) marked in the PIT entry.

However, NFD and Metis do have differences in their implementations and in their parent architectures, which must be specifically addressed to ensure a fair comparison. First, NFD uses a tree-based data structure to provide a unified PIT and FIB [2], whereas Metis uses separate data structures for the PIT and FIB [11]. While this is a difference in implementation, we have nothing to control for since there is no difference in forwarding semantics. Second, NFD and Metis have different default behavior for selecting an output face (ICN interface) in the presence of multiple faces: NFD selects a single 'best' face, whereas Metis splits the load evenly across the candidate faces. In experiments involving multiple faces, we ensured that NFD achieved the same load-splitting behavior that is the default in Metis. Third, NFD's Content Store naturally has more overhead because it supports more rich functionality: matching interest prefixes with data names. Our microbenchmarks do not leverage this extra functionality, but we cannot eliminate overhead incurred from its support.

3 DETAILS OF EXPERIMENTATION

This section discusses some practical details of our experimental method. In particular, we present the way we generated our synthetic workloads and details of the machines used for our experiments.

3.1 Workload Generation

We must carefully consider the workload for these microbenchmarks, as ICN names are hierarchical and variable length. This adds additional complexity to the packet processing tasks of an ICN forwarder, and the structure of content names can affect forwarding performance. To our knowledge, there are no satisfactory ICN workloads on the Internet; `icn-names.net` seems to have been a useful resource, but at the time of writing is no longer hosted. As a result, we generate our own synthetic workload for testing ICN forwarders, inspired by published techniques in [5, 14].

Many previous works in ICN leverage the Alexa and DMOZ top URL lists to generate their synthetic workloads, but these are no longer available. Fortunately, Cisco provides the top 1 million domains on the Internet in their Umbrella Popularity List [3]. We use this as a starting point, but must adapt it to ICN: we convert the domains into ICN names, preserving the intended hierarchical structure. As an example, `www.google.com` becomes `/com/www/google`. By retaining the hierarchical structure in the URL, the number of name components (and their respective lengths) are derived directly from the domains themselves.

However, because the Umbrella dataset gives us domain names, we need to do more work to have our synthetic workload properly reflect how content names are distributed. While we are unaware of any academic work on average URL length, [6] reports that the average URL length is 76 characters. The average name in our workload is 20 characters, so we add 56 characters of padding to allow our average name length to match the average URL name length. For simplicity, we use this extra padding as the content name being requested. Finally, to incorporate content popularity we use RFC 7945 as our model for achieving this in the context of ICN [5]. Following the 'Web' portion of the RFC, we generate our synthetic workload by sampling our padded, translated names via a Zipf distribution with $\alpha = 0.64$. For higher fidelity workload generation, separate distributions could be used for the "domain names" and the content names. Additionally, the amount of name overlap could be explicitly parameterized.

3.2 Experimental Setup

Our experiments were conducted on machines from the Open Networking Laboratory [19], which allows us to run these ICN forwarders on bare metal with guaranteed isolation. For nearly all of the experiments, our logical topology is a simple dumbbell topology, with one producer machine and one consumer machine directly connected to the device under test (DUT). In specific cases, we have two producers and/or two consumers directly connected to the DUT on separate interfaces. The DUT has an Intel Xeon E5520 clocked at 2.27GHz, which has 8 cores (16 hyperthreads) spread across two NUMA nodes. It also has 12 GB of DDR3 memory clocked at 1066 MHz, and an Intel I350 NIC with 4 ports at 1 Gbps. For software, the DUT is running Ubuntu 19.04 with Linux kernel 5.0.0, NFD version 0.6.6, and Metis at commit e59f9c7.

ICN experiments are often bottlenecked by the traffic output of end hosts as opposed to the pure forwarding nodes, as NFD / Metis must also be run on the end hosts serving up the test traffic. To avoid this bottleneck, we wrote simple yet powerful NDN/CCNx packet crafting tools to generate and send our workloads at much higher speeds than Metis or NFD can support. In doing this, we ensured that producer and consumer machines are not the performance bottleneck. On the DUT, we verified that any bottlenecks lie in the ICN forwarders processing capabilities and not in any lower layers (e.g. socket layer, kernel, or hardware).

4 MEASURING THE IMPACT OF THE PIT

Focusing on the PIT is essential for interrogating the performance of an ICN forwarder, as it must handle per-packet writes to provide the ICN breadcrumbs. To find the bottleneck processing rate of the PIT, we need to minimize the impact of the other key data structures in the forwarding pipeline: the FIB and the CS. The FIB effects can be minimized by sending all traffic through a single default prefix. Unfortunately, the default prefixes in Metis are bugged, so we must rely on a workaround by adding 40 single component prefixes that nearly span our entire dataset. The Content Store has the smallest impact when it is disabled via configuration. With this setup, we maximize our odds of seeing bottleneck behavior from the PIT.

4.1 Interest lifetime and PIT scaling

First, we explore how the interest forwarding rate scales with larger and larger PIT sizes. Because the PIT is a dynamic data structure which must continually clean out old entries, we cannot simply configure the number of items resident in the PIT. However, we can configure the Interest lifetime in both NDN and CCNx, which is directly proportional to the PIT occupancy. PIT occupancy is a function of the interest lifetime and the round trip time. As a result, we design our first experiment to show how the forwarding performance scales over a wide variety of interest lifetimes, ranging from the minimum value 1 millisecond to a rather large value of 5 seconds. For the PIT occupancy to grow and reach its steady-state, we must ensure that the interests in the PIT will not get satisfied and cleaned out. To accomplish this, our experiment uses interest forwarding only—no data is sent. Specifically, we measure the mean interest forwarding rate of each forwarder over the following Interest lifetimes (in milliseconds): 1, 5, 10, 50, 100, 500, 1000, 5000.

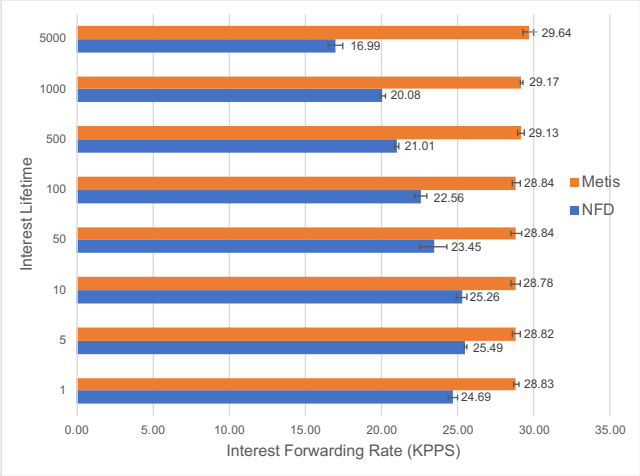

Figure 1: Interest forwarding rate over varying interest lifetimes.

In Fig. 1 we see the interest forwarding rate over the interest lifetimes for both NFD and Metis. Metis forwards approximately 29,000 interests per second, or 29 kilo-packets per second (KPPS), irrespective of the configured Interest lifetime. NFD's behavior is more interesting. The lifetime of 5 ms is a performance knee. Below the knee (with a lifetime of 1ms), NFD takes a performance hit because it is forced by the small lifetime to clean the PIT too aggressively. However, after the knee, there is a substantial drop-off in the interest forwarding rate. In the worst case scenario, a lifetime of 5 seconds can only forward 17,000 interests per second — a 33% drop from the peak forwarding rate at the knee. Furthermore, there is a clear downward slope starting with a lifetime of 50 ms.

We learned from this experiment that the best interest lifetime for NFD's PIT efficiency is 5 milliseconds. 5 milliseconds seems like a good candidate to create PIT churn, so achieving the fastest forwarding rate is unexpected. However, it makes more sense once we recall that larger Interest lifetimes increase the PIT occupancy. It is important to keep in mind that 5 milliseconds might be the best for a pure interest forwarding workload, but it would be too small for some deployments of NFD because the data would not make it back on time. This elucidates a tension with interest lifetime timing: we want it small enough to reduce the PIT occupancy (and by extension the PIT processing burden), but large enough to ensure our data will still make it back. The default interest lifetime in NDN is 4 seconds. As a result of this experiment, we know in certain scenarios this default lifetime could harm forwarding performance. Fortunately, this is not a strong concern for standard operation of NFD, as interest forwarding with no data coming back is not realistic. However, a denial of service attack could use larger lifetimes to induce more processing burden on the forwarder.

4.2 Interest/Data exchange

In the previous experiment, we explored the influence of the PIT and Interest lifetimes on the interest forwarding pipeline in an ICN forwarder. Now we explore the influence of the PIT on interest/data exchange, which incorporates part of the data processing pipeline. Some ICN forwarders, e.g. NFD, also access the PIT on data reception to ensure that the data packet was asked for and not unsolicited. As a result, the PIT has the potential to be a noticeable bottleneck in the data processing pipeline. Additionally, it is important to run an interest/data exchange experiment simply so we have a performance baseline for each forwarder. The Interest lifetime experiment showed us that 5 milliseconds and 5 seconds were the two most extreme performance outcomes. As a result, we do not need to run experiments across the entire grid of Interest lifetime values. We simply probe the interest/data exchange rate, measured in thousands of transactions per second (KTPS), over these two lifetimes.

Table 1: Interest/Data exchange rate over low/high lifetimes.

Forwarder	Lifetime (ms)	Mean Exchange Rate (KTPS)
NFD	5	12.05
NFD	5000	12.04
Metis	5	15.37
Metis	5000	15.21

Table 1 shows the result of the experiment. We can see the interest lifetime has little effect after incorporating the data pipeline. Data packets satisfy the interests, keeping the PIT occupancy low enough that the bottlenecks lie elsewhere. We do get a baseline measure of performance for interest/data exchange on each forwarder: NFD can support 12 KTPS and Metis can support just over 15 KTPS. With these experiments, we have explored the PIT's influence on both major pathways in an ICN forwarder. One pathway left to explore is the scenario where we do not forward a request because we have the data cached locally.

5 MEASURING THE IMPACT OF THE CONTENT STORE

The Content Store also has a difficult job: it must do per-packet lookups in the interest processing pipeline, and per-packet insertions on the data processing pipeline. On top of that, it must actually

hold the bytes for each data packet it stores. In order to minimize the impact of the PIT, we will use an Interest lifetime of 5 milliseconds (the lifetime that had the peak interest forwarding rate). In order to minimize the impact of the FIB, we again use our default prefix scheme. With this configuration, we maximize our odds of seeing bottleneck behavior from the Content Store. Metis and NFD have different default sizes for their content stores. In order to make a fair comparison, we configured Metis to have the same size Content Store as NFD: 65,536 items. Also, while we do not enumerate over different Content Store sizes in these experiments, that would be a natural next-step for more in-depth inquiry.

5.1 Interest forwarding: enabled but empty

As a first step to revealing the overhead of the Content Store, we first want to probe the overhead of an empty lookup. We can do exactly this with an Interest forwarding experiment. On reception of an Interest, the forwarder will perform the logic of the Content Store lookup. Because there will be no data packets flowing on the return path, the Content Store will remain empty. As a result, this experiment is simply the Interest forwarding with the Content Store disabled versus the Content Store enabled but empty.

Table 2: Mean Interest forwarding rate with an enabled & empty or disabled Content Store.

Forwarder	Content Store)	Mean Forwarding Rate (KPPS)
NFD	Off	25.49
NFD	On	22.33
Metis	Off	28.82
Metis	On	27.97

The results of the experiment are shown in Table 2. Both forwarders experience performance degradation due to the empty Content Store lookup: NFD has a 12% drop in forwarding rate, compared to a 3% drop for Metis. It is true that we are performing an extra lookup, and there will be some cost associated with that. However, the extent of the performance hit is surprising—particularly for NFD—given that the extra lookup is little more logically than an empty hash table lookup (which are quite cheap). NFD's Content Store is a significant bottleneck, even without the cost associated with storing the data packets.

5.2 Interest/Data exchange

The meat of the Content Store experimentation is when we have interest/data exchange, as opposed to interest forwarding. Now that we allow data to flow, we consider three distinct scenarios for an enabled Content Store: (i) the Content Store has the entirety of the data, so we have a 100% hit rate; (ii) the Content Store is writable, but serving that cached data is disabled; (iii) the Content Store is not writable, but it is full of data that is not ours. The last two are both Content Store misses, but they target different paths of the processing pipeline. These scenarios are labeled 'hit', 'miss', and 'store' respectively. We measure each forwarder's interest/data exchange performance under these three scenarios.

We see the results of the experiment in Fig. 2. The difference in Content Store performance between Metis and NFD is even more pronounced than the previous experiment. In the 'hit' scenario, Metis sustains nearly 30 KTPS whereas NFD can only sustain 16.5 KTPS—nearly 80% higher performance from Metis. For the other scenarios, Metis is roughly 50% higher performance than NFD. We also see that writing to the content store is slightly more volatile/costly than the lookup itself.

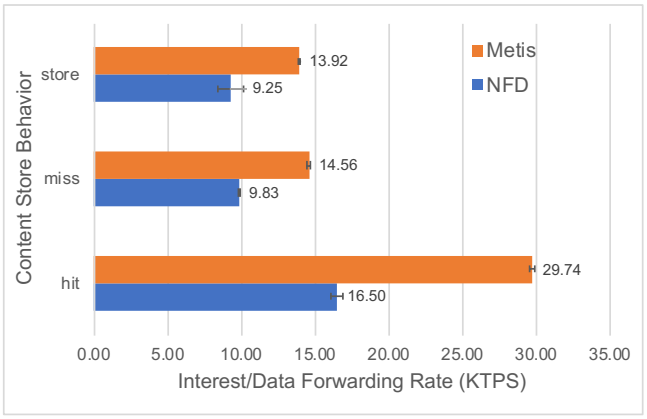

Figure 2: Interest/data exchange rate in different Content Store scenarios.

It is clear from these Content Store experiments that NFD's content store is a significant bottleneck to its packet processing. Depending on the workload, it is worth considering disabling the Content Store in NFD. It also seems to be a fruitful direction to pursue optimization, since we saw Metis's Content Store perform substantially better.

6 MEASURING THE IMPACT OF THE FIB

The FIB has an easier processing task than the other core data structures in an ICN forwarder—it only requires per-packet lookups, but no per-packet modications to the FIB. At a high level, we expect the FIB to naturally be more efficient for the usual scope of these forwarders (the edge, not the network core). Because we want to probe the efficiency of the FIB, we must minimize the influence of the PIT and Content Store. In order to minimize the impact of the PIT, we use the 5 millisecond Interest lifetime. In order to minimize the impact of the Content Store, we disable it.

6.1 Scaling the FIB

As a first experiment, it is natural to explore how FIB performance scales with the number of entries. In order to quickly cover the space, exponential increase from a default prefix to a large number of prefixes is desirable. One million prefixes is a viable upper end of the range, as it could completely encompass an entire BGP full view—since no ICN core routers exist at the time of writing, this is a reasonable heuristic. Under this design, our number of prefixes would scale as follows: 1, 10, 100, 1000, 10,000, 100,000, 1,000,000. Unfortunately, we cannot apply this design to NFD and Metis verbatim. Metis has the issues with default prefixes that we mentioned earlier, so the smallest number of prefixes we can assign is 40. Additionally, both forwarders have issues adding large number of prefixes (Metis crashes, NFD is prohibitively slow). As

a result, our best approximation of this design is setting the low end of the range to 40 prefixes and the upper end to 1000 prefixes. We acknowledge that this upper limit is not reasonable in normal circumstances, but we are constrained by NFD and Metis.

Table 3: FIB scaling with number of prefixes for interest forwarding and interest/data exchange

Forwarder	Num Prefixes)	Mean KPPS	Mean KTPS
NFD	40	24.88	12.36
NFD	1000	25	12.31
Metis	40	28.82	15.36
Metis	1000	26.98	15.09

Table 3 shows the results of the experiment. For the interest/data exchange, there is not a substantial change when jumping to 1000 prefixes. We need a higher number of prefixes in order to accomplish a meaningful interest/data exchange experiment. However, for the Interest forwarding experiment, Metis shows a degradation in performance when jumping to 1,000 routes (roughly 7% performance drop).

6.2 Multiplexing interfaces

Number of prefixes is not the only way the FIB can scale in an ICN forwarder. In ICN, the FIB supports multiple egress interfaces, and the subset of interfaces selected for forwarding is determined by the forwarding strategy. This speaks to a larger experiment which not only probes multiple egress interfaces, but also multiple ingress interfaces. As a result, we must expand our logical topology so that it allows for multiple producers and consumers on each end of the dumbbell. To probe the cost of multiplexing across multiple interfaces, we perform a simple interest/data exchange for topologies of the form 1x1, 1x2, 2x1, and 2x2—where the number of consumers is on the left and the number of producers is on the right. For example, a 2x1 topology has 2 consumers sending requests to 1 producer.

We need to be careful using multiple interfaces, as NFD and Metis have vastly different default forwarding strategies (the component responsible for choosing the egress interface). It would be unfair to compare them without remedying this. As a result, we ported Metis's 'random' strategy to NFD, which randomly selects the egress interface out of the list of possible interfaces. This strategy is quite inexpensive since it does not maintain any state, so we can also be confident we are not introducing a new bottleneck.

The results are shown in Fig. 3. There is a performance hit for any form of additional multiplexing over the base 1x1 configuration. Interestingly, both forwarders have the worst performance drop for the 2x1 configuration, though for Metis it could be in the noise due to higher variance. The multiplexing over multiple interfaces in the FIB, at least for selecting between two interfaces, does not seem to be an additional bottleneck. The bottleneck is likely in other multiplexing-related code paths.

7 ADDITIONAL EXPERIMENTATION

There are other directions that fall out of our primary focus that should absolutely be considered when testing the performance of an ICN forwarder. We identify the following directions:

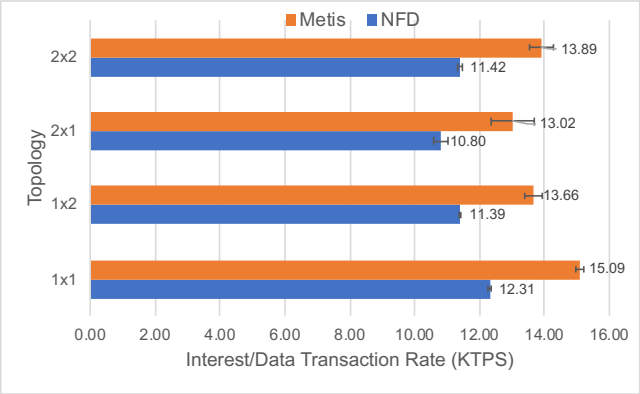

Figure 3: FIB multiplexing over different interfaces.

(1) Throughput / Goodput testing
(2) Latency measurements
(3) Sensitivity to Name Distribution

Large packet goodput experiments can help reveal sources of bufferbloat in the stack, which could be lurking in any number of layers of buffers. Plus, in order to fully profile a forwarder, having an idea of its goodput is essential. Latency measurements, at the same time as the packet processing experiments and the goodput experiments, are also crucial. In this paper we captured latency measurements for every interest/data exchange experiment—they did not show much variation, but it is important to investigate. Name distribution is an incredibly complex topic, as there are so many directions to go. However, one area name distribution will certainly be important is probing multicore forwarders, as the structure of the names could help reveal sources of contention.

We design one experiment to address these topics in a substantive way. Specifically, we perform a large packet goodput test where we vary the popularity of the names in the generation of our synthetic workload—during this large packet test, we measure the latency of the interest/data exchange. By using large packets and measuring the latency, this experiment will help reveal if there are bufferbloat issues in the forwarders. By varying the popularity of the names, we create stronger possibilities for request overlap—we accomplish this by comparing a workload generated with a Zipf $\alpha = 0.84$ against the standard $\alpha = 0.64$ we used for the rest of the experiments. Large data packets with varying amounts of request overlap are the ideal factors to perform a Content Store experiment, as the Content Store is the only data structure that stores a significant amount of bytes beyond the metadata (by storing the data packets). As a result, we probe the effect of these factors while the Content Store is enabled and disabled. This experiment also serves as a sensitivity analysis for the Content Store, as our main experimentation focused on all-or-nothing configurations and this experiment leverages the workloads' actual request overlap. In order to prevent fragmentation from being an issue, we set the payload length to 1250 bytes to accommodate the NDN/CCNx overhead.

Fig. 4 shows the goodput for Metis and NFD under the different configurations. Metis achieves substantially higher goodput than NFD with the same payload size, as a direct result of it being able to

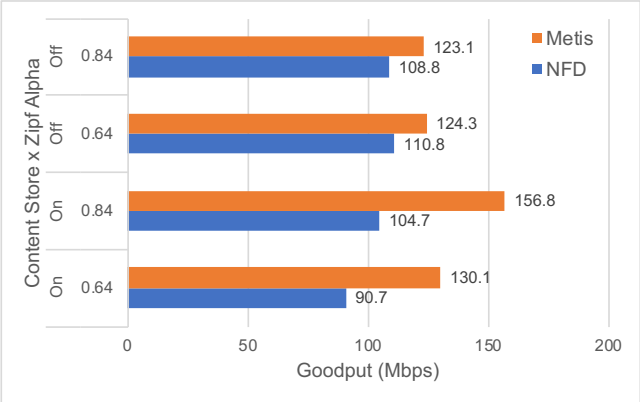

Figure 4: Goodput for different Content Store configurations and name distributions.

process more packets. In addition, there is a difference in behavior when the Content Store is 'on' vs 'off': turning the Content Store on results in increased goodput for Metis but decreased goodput for NFD. When the Zipf $\alpha = 0.64$, the Content Store hit rate is just over 10%, and the hit rate is nearly 50% when $\alpha = 0.84$. NFD's Content Store is so inefficient that even a 50% hit rate does not benefit the forwarding performance.

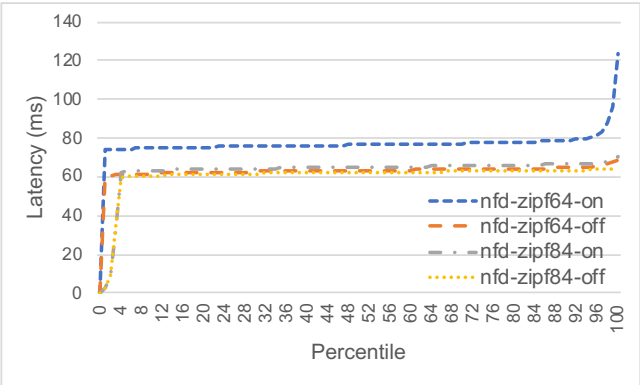

Figure 5: RTT distribution for NFD during the goodput test.

Fig. 5 and Fig. 6 show the measured latencies during the goodput test for NFD and Metis, respectively. The latency distributions are in the same ballpark between the two forwarders, except for two outliers. Taking advantage of the 50% hit rate, Metis has reduced latency (about 10 ms) in the case of $\alpha = 0.84$ and Content Store 'on'. NFD has increased latency in the case of $\alpha = 0.64$ and Content Store 'on'. In other words, NFD is not breaking even on performance for the Content Store until it surpasses a 50% hit rate—the 10% hit rate of $\alpha = 0.64$ is not even close, so it is a performance bottleneck.

8 RELATED WORK

There is a wide body of ICN literature focusing on performance and scalability, but we limit our scope to research containing thorough analyses. For work that focuses on specific components in

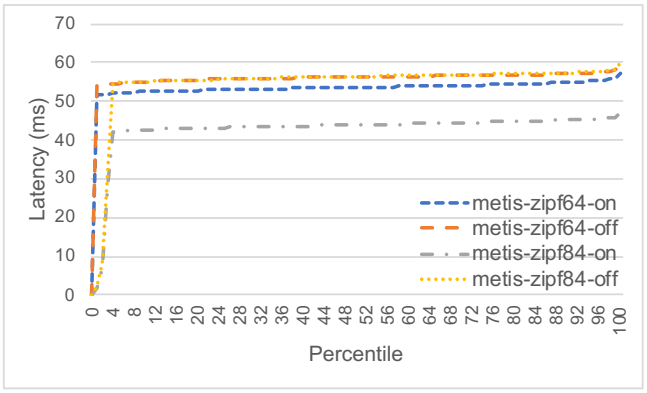

Figure 6: RTT distribution for Metis during the goodput test.

ICN forwarders, Yuan and Crowley design and analyze a higher performance PIT in [20], and they provide a FIB design with better worst-case complexity guarantees than longest prefix matching in [21]. Virgilio et al. [18] focus on analyzing the performance and memory consumption of the PIT under various scenarios. Rossini and Rossi [4] and Kim and Yeom [8] both analyze the Content Store under a variety of workloads.

For works that analyze ICN forwarders, Ohsugi et al. [12] measure and model the performance of a multicore NDN router, particularly with regards to its bandwidth and power consumption. So et al. have multiple works exploring the use of hash tables for an NDN forwarding engine [15, 16]. Kirchner et al. [9] provide a high speed, DPDK-based implementation of a CCN router that can forward over 10 MPPS. Rossini et al. [13] model and measure the performance of a CCN router to sustain gigabytes of video streaming.

Finally, there are works that focus on testing the performance of existing ICN implementations. Marchal, Colez, and Festor [10] measure the server-side performance of NDN. They find that the heavy asymmetric encryption used by end hosts substantially slows down the NDN server's data generation performance. Khatouni et al. [7] compare the performance of multiple ICN prototypes including CCNx. We should note that, in their test topologies, they must 40-50 hosts connected to a single forwarder instance in order to fully test its performance.

9 CONCLUSION

Robust performance analysis of ICN forwarders has been an underserviced area by the community. In this paper, we provide a thorough and practical method for analyzing the performance of an ICN forwarder. We detail our synthetic workload generation, which incorporates a number of key dimensions for ICN workloads. We then present a series of microbenchmarks that focus on the packet processing capabilities of ICN forwarders while probing their core data structures (PIT, FIB, and Content Store). We then apply these experiments to two ICN forwarders—NFD and Metis—to show the viability of these microbenchmarks, and how they reveal subtle behaviors in the ICN forwarding pipelines tested.

ACKNOWLEDGMENTS

The authors would like to thank Dave Oran, their shephard, for repeatedly providing thoughtful and constructive feedback. The authors would also like to thank the anonymous reviewers for their helpful suggestions and comments. This work is supported by a gift from Intel Corporation and by the National Science Foundation (NSF) under the following grants: CNS-1719366 and CNS-1629807.

REFERENCES

[1] 2017. CICN. Available: https://wiki.fd.io/view/Cicn. [Online; accessed 7 May 2017].

[2] A. Afanasyev et al. 2018. *NFD Developer's Guide*. Retrieved June 2020 from http://named-data.net/publications/techreports/ndn-0021-10-nfd-developer-guide/

[3] Cisco. 2020. *Umbrella Popularity List*. Retrieved June 2020 from http://s3-us-west-1.amazonaws.com/umbrella-static/index.html

[4] D. Rossi G. Rossini. 2012. *A dive into the caching performance of Content Centric Networking*. Retrieved June 2020 from https://nonsns.github.io/paper/rossi12camad.pdf

[5] K. Pentikousis, Ed., B. Ohlman, E. Davies, S. Spirou, and G. Boggia. 2016. Information-Centric Networking: Evaluation and Security Considerations. RFC 7945. https://doi.org/10.17487/RFC7945

[6] Kelvin. 2010. *Average length of a URL (Part 2)*. Retrieved June 2020 from https://www.supermind.org/blog/740/average-length-of-a-url-part-2

[7] A. S. Khatouni, M. Mellia, L. Venturini, D. Perino, and M. Gallo. 2016. Performance Comparison and Optimization of ICN Prototypes. In *2016 IEEE Globecom Workshops (GC Wkshps)*. 1–6. https://doi.org/10.1109/GLOCOMW.2016.7848997

[8] Yusung Kim and Ikjun Yeom. 2013. Performance analysis of in-network caching for content-centric networking. *Computer Networks* 57, 13 (2013), 2465 – 2482. https://doi.org/10.1016/j.comnet.2012.11.026

[9] Davide Kirchner, Raihana Ferdous, Renato Lo Cigno, Leonardo Maccari, Massimo Gallo, Diego Perino, and Lorenzo Saino. 2016. Augustus: A CCN Router for Programmable Networks. In *Proceedings of the 3rd ACM Conference on Information-Centric Networking* (Kyoto, Japan) *(ACM-ICN '16)*. ACM, New York, NY, USA, 31–39. https://doi.org/10.1145/2984356.2984363

[10] Xavier Marchal, Thibault Cholez, and Olivier Festor. 2016. Server-side Performance Evaluation of NDN. In *Proceedings of the 3rd ACM Conference on Information-Centric Networking* (Kyoto, Japan) *(ACM-ICN '16)*. ACM, New York, NY, USA, 148–153. https://doi.org/10.1145/2984356.2984364

[11] M. Mosko. 2017. Metis CCNx 1.0 Forwarder. arXiv:1707.04832 [cs.NI]

[12] K. Ohsugi, J. Takemasa, Y. Koizumi, T. Hasegawa, and I. Psaras. 2016. Power Consumption Model of NDN-Based Multicore Software Router Based on Detailed Protocol Analysis. *IEEE Journal on Selected Areas in Communications* 34, 5 (May 2016), 1631–1644. https://doi.org/10.1109/JSAC.2016.2520278

[13] G. Rossini, D. Rossi, M. Garetto, and E. Leonardi. 2014. Multi-Terabyte and Multi-Gbps Information Centric Routers. In *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*. 181–189. https://doi.org/10.1109/INFOCOM.2014.6847938

[14] U. Schnurrenberger. 2017. Comparing apples to apples in ICN. In *14th IEEE Annual Consumer Communications & Networking Conference, CCNC 2017, Las Vegas, NV, USA, January 8-11, 2017*. 89–94.

[15] Won So, Ashok Narayanan, and David Oran. 2013. Named Data Networking on a Router: Fast and Dos-resistant Forwarding with Hash Tables. In *Proceedings of the Ninth ACM/IEEE Symposium on Architectures for Networking and Communications Systems* (San Jose, California, USA) *(ANCS '13)*. IEEE Press, Piscataway, NJ, USA, 215–226. http://dl.acm.org/citation.cfm?id=2537857.2537892

[16] Won So, Ashok Narayanan, Dave Oran, and Yaogong Wang. 2012. Toward Fast NDN Software Forwarding Lookup Engine Based on Hash Tables. In *Proceedings of the Eighth ACM/IEEE Symposium on Architectures for Networking and Communications Systems* (Austin, Texas, USA) *(ANCS '12)*. ACM, New York, NY, USA, 85–86. https://doi.org/10.1145/2396556.2396575

[17] Tian Song, Haowei Yuan, Patrick Crowley, and Beichuan Zhang. 2015. Scalable Name-Based Packet Forwarding: From Millions to Billions. In *Proceedings of the 2Nd ACM Conference on Information-Centric Networking* (San Francisco, California, USA) *(ACM-ICN '15)*. ACM, New York, NY, USA, 19–28. https://doi.org/10.1145/2810156.2810166

[18] Matteo Virgilio, Guido Marchetto, and Riccardo Sisto. 2013. PIT Overload Analysis in Content Centric Networks. In *Proceedings of the 3rd ACM SIGCOMM Workshop on Information-Centric Networking* (Hong Kong, China) *(ICN '13)*. Association for Computing Machinery, New York, NY, USA, 67–72. https://doi.org/10.1145/2491224.2491225

[19] Charlie Wiseman, Jonathan Turner, Michela Becchi, Patrick Crowley, John DeHart, Mart Haitjema, Shakir James, Fred Kuhns, Jing Lu, Jyoti Parwatikar, Ritun Patney, Michael Wilson, Ken Wong, and David Zar. 2008. A Remotely Accessible Network Processor-based Router for Network Experimentation. In *Proceedings of the 4th ACM/IEEE Symposium on Architectures for Networking and Communications Systems* (San Jose, California) *(ANCS '08)*. ACM, New York, NY, USA, 20–29. https://doi.org/10.1145/1477942.1477946

[20] H. Yuan and P. Crowley. 2014. Scalable Pending Interest Table design: From principles to practice. In *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*. 2049–2057.

[21] Haowei Yuan and Patrick Crowley. 2015. Reliably Scalable Name Prefix Lookup. In *Proceedings of the Eleventh ACM/IEEE Symposium on Architectures for Networking and Communications Systems* (Oakland, California, USA) *(ANCS '15)*. IEEE Computer Society, Washington, DC, USA, 111–121. http://dl.acm.org/citation.cfm?id=2772722.2772739