Performance Evaluation of an Open-Source Implementation of Content-Centric Networking

Ryo Nakamura and Hiroyuki Ohsaki
Department of Informatics
Graduate School of Science and Technology
Kwansei Gakuin University
Hyogo 669–1337, Japan
E-mail: {r-nakamura,ohsaki}@kwansei.ac.jp

***Abstract*—In this paper, we investigate the scalability of CCNx, an open-source CCN (Content-Centric Networking) implementation, in terms of the number of nodes. As performance metrics, we measure the total throughput of content delivery, the packet loss rate in the network, and the average content delivery time. We also examine the performance bottleneck of CCNx through system-wide profiling, which quantitatively shows that per-packet digest-based authentication is the performance bottleneck in CCNx. We therefore investigate how the scalability of CCNx in terms of the number of nodes can be improved by hardware offloading of Data-chunk digest computation.**

***Keywords*-CCN (Content-Centric Networking); CCNx; scalability, hardware offloading**

I. Introduction

In the recent years, Content-Centric Networking (CCN) [1] has been under the spotlight as one of the networks mainly focusing on the contents that are transmitted and received (data-centric networks) instead on the hosts that transmit and receive the contents (host-centric networks). CCN adopts a request-and-response communication model. In CCN, a unique identifier is assigned to every content. The content request packet (*Interest packet*) from a user is routed among CCN routers using longest-prefix matching of the content identifier to search for the location of the content. The content discovered is delivered to the user as a response packet (*Data packet*) by retracing the path of the request packet.

A CCN router has a buffer memory called CS (Content Store), and it caches the forwarded Data packet in the buffer memory. CCN routers on a network cache the forwarded contents, and reuses data. When a CCN router receives another Interest packet for the same content, it returns the Data packet cached so that the amount of traffic on network can be reduced and the content delivery time can be shortened.

In the literature, the effectiveness of CCN has been investigated mainly through simulation experiments. On the contrary, a software implementation called CCNx [2] has been developed as an open-source software, and a few performance studies of CCN through experiments have been performed.

However, scalability of CCN in terms of the number of nodes has not been fully understood. For large-scale deployment of CCN in real networks, as a communication infrastructure for various types of social activities, it is crucial to clarify how the CCN architecture itself and its components such as CCN routers and repositories are scalable in terms of the number of nodes.

In this paper, we therefore investigate the scalability of CCNx, one of major open-source CCN implementations, in terms of the number of nodes (i.e., CCN routers and repositories) through experiments. Using virtualization technology, a large-scale CCN network with dozens of nodes are constructed in a single physical computer. In our experiments, contents stored in the repositories are repeatedly requested from different entities for performance benchmarking. As performance metrics, we measure the total throughput of content delivery, the packet loss rate in the network, and the average content delivery time. We also examine the performance bottleneck of CCNx through system-wide profiling, which quantitatively shows that per-packet digest-based authentication is the performance bottleneck in CCNx. We therefore investigate how the scalability of CCNx in terms of the number of nodes can be improved by hardware offloading of Data-chunk digest computation.

This paper is organized as follows. Section II introduces previous works related to scalability of CCN and performance studies using CCNx. Section III explains the methodology of our experiments such as hardware and software used for experiments, workload generation, performance metrics, and factors to be studied. Section IV presents experiment results for addressing the scalability of CCNx in terms of the number of nodes, including detailed examination of the CCNx performance bottleneck through system-wide profiling. Section V investigates how the scalability of CCN can be improved with hardware offloading of Data-chunk digest computation at CCN routers. Finally, Section VI concludes this paper and discusses future works.

II. Related work

In [3], Perino *et al.* quantitatively discuss the scalability of CCN routers to address whether an Internet-scale CCN network can be realized with the latest technologies. Consequently, the authors conclude that using the state-of-the-art technologies for CCN routers, the CCN architecture could scale to the size of the current ISP networks and CDNs (Content Distribution Networks), but not to the size of the current Internet.

In the literature, several simulation studies for investigating the CCN performance have been performed. For instance, in [4], Chiocchetti *et al.* address the scalability of CCN in terms of the number of nodes by comparing simulation times and memory usages when running simulations of different network sizes. The authors show that the scalability of CCN is mostly determined not by the number of nodes in the network but by the number of contents in the network.

Several experimental performance evaluations of CCNx, an open-source software implementation of CCN, have been performed for measuring the end-to-end performance and examining the performance bottleneck in CCN routers [5], [6]. In [5], Yuan *et al.* measure the throughput and the resource usage from experiments using CCNx in a small-scale network. The authors show, to realize a CCN network with 1 [Gbit/s] effective throughput, performance issues of CCNx such as exact string matching in PIT (Pending Interest Table) and CS (Content Store) lookups and longest-prefix string matching in FIB (Forwarding Information Base) lookup should be solved. On the contrary, in [6], Tang measures the end-to-end performance of CCNx running in a virtualization infrastructure called SAVI (Smart Applications on Virtual Infrastructure). The author shows that the throughput of CCNx is increased by more than 12% by optimizing the function for Data packet decoding called `ccn_skeleton_decode`. However, in those studies, the scalability of CCN in terms of the number of nodes has not been addressed. Hence, it has not been understood how the end-to-end performance of CCN (e.g., throughput and content delivery time) is degraded as the number of nodes in the network increases. In this paper, we therefore experimentally investigate the scalability of CCNx in terms of the network of nodes.

III. Experiment Methodology

In this paper, we investigate the scalability of CCNx in terms of the number of nodes through experiments. In what follows, we explain the methodology for our experiments.

We used the CCNx software version 0.8.2 running on Gentoo GNU/Linux with 32-bit Linux Kernel version 3.15.10 and an Intel-based desktop computer (Core i7 2640M 2.80 [GHz] with 8 [Gbyte] memory). The clock speed of the CPU was fixed at 2.80 [GHz] using cpufreq, the CPU frequency scaling module in the standard Linux Kernel.

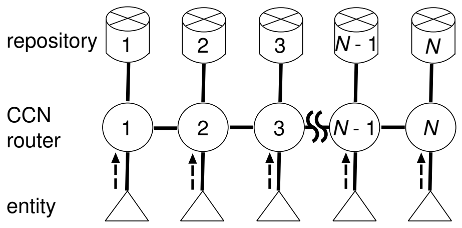


Figure 1. Linear network topology used in experiments (constructed on a single physical computer using network virtualization)

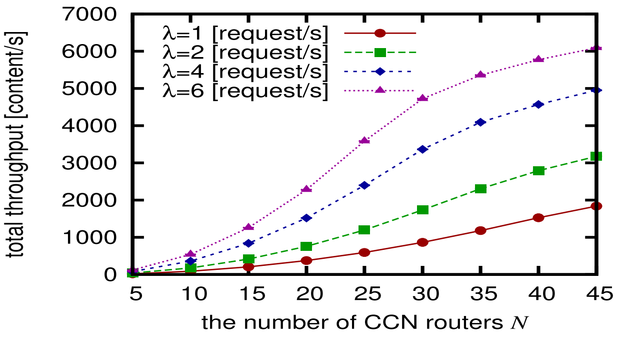
We assume that multiple CCN router slices are constructed on a single physical router using network virtualization. So, $2N$ `ccnd` daemons, which correspond to N CCN routers and N repositories, were executed on the single physical computer.

CS (Content Store) size of all CCN routers were equally set to 10 [content]. Every `ccnd` daemons (i.e., CCN routers and repositories) were assigned an unique IP address, which was configured as the alias for the Ethernet interface of the desktop computer. We used a simple linear network topology shown in Fig. 1. In the linear network topology, N CCN routers are connected in serial, and N repositories are connected to respective CCN routers. In other words, entries of FIBs in all CCN routers and repositories are configured to construct the linear network topology. Every virtual link among CCN routers and repositories was fixed at 1 [Gbit/s] with 0% packet loss using netem, a kernel module for providing network emulation functionality.

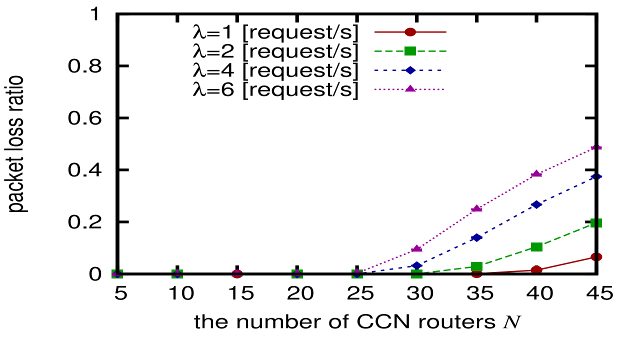
Every repository was provided with a single unique content of 8 [Kbyte]; i.e., there were N contents with unique content identifiers in the network. In our experiments, every content was repeatedly requested from entities, each of which was connected to a different CCN router. Specifically, ith entity $(1 \leq i \leq N)$ repeatedly issued Interest packets to ith CCN router for all the contents in the network. For workload (i.e., series of Interest packets) generation, we developed a CCN request generator, which randomly sent Interest packets encoded in the CCNB (CCN Binary) format to specified CCN router's face. The interval between successive Interest packet generation was given by the exponential distribution with the mean $1/\lambda$, where λ [content/s] is the request rate.

As performance metrics, we used followings.

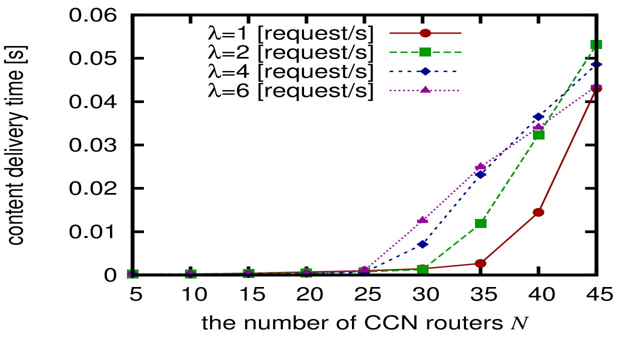
- Throughput
  The number of successful content deliveries in the network per a unit time. Note that our throughput definition is for the entire network (i.e., network-level throughput), rather than for every entity (i.e., end-to-end throughput).
- Packet loss rate
  The rate of Interest packet losses in the network, which is defined as the ratio of the number of unsuccessful



(a) Total throughput

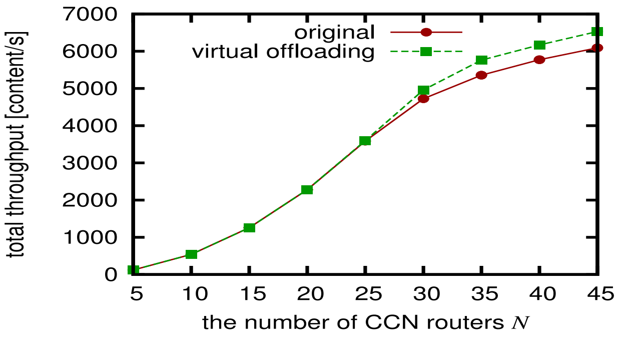


(b) Packet loss rate

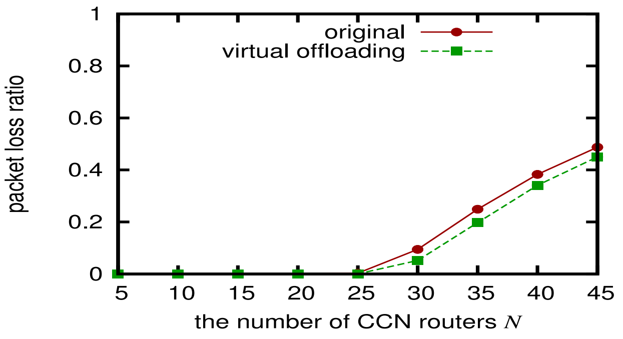


(c) Average content delivery time

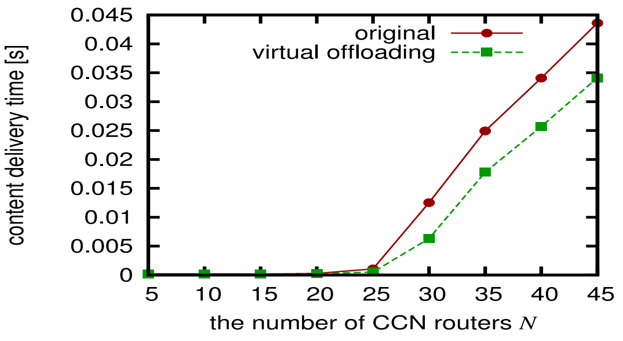
Figure 2. Experiment results for different numbers of CCN routers



(a) Total throughput



(b) Packet loss rate



(c) Average content delivery time

Figure 3. Experiment results with/without virtual offloading for different numbers of CCN routers

content deliveries to the number of total Interest packets injected in the network.

- Average content delivery time
  The average time elapsed since an entity sends an Interest packet to the network by the time when the entity receives the corresponding Data packet from the network. Note that the average content delivery time is the average of all *successful* content deliveries, which do not include unsuccessful (e.g., lost or pending) content deliveries.

We calculated those performance metrics from log files of `ccnd` daemons. A single experiment was lasted for 300 [s], and log files of `ccnd` daemons in the last 120 [s] were used for measurement to ignore variability in transient state. For every Interest packet generated from an entity, both timestamps of the Interest packet reception and the Data packet transmission at the connected CCN router were extracted from the log file of the corresponding `ccnd` daemon. From all timestamps of both successful and unsuccessful content deliveries, the throughput, the packet loss rate, and the average content delivery time were calculated. Experiments were repeated for 10 times for a given condition, and the average and its 95% confidence interval for every measurement were calculated.

Table I
RESULT OF A SYSTEM-WIDE PROFILING WITH OPROFILE FOR $N = 30$ AND $\lambda = 6$ [CONTENT/S]

%	image name	app name	symbol name
29.5	libcrypto.so.1.0.0	ccnd	sha256_block_data_order()
3.35	ccnd	ccnd	ccn_skeleton_decode()
3.07	vmlinux	ccnd	sock_poll()
2.50	ccnd	ccnd	siphash_2_4()
2.32	vmlinux	ccnd	tcp_poll()

IV. EXPERIMENT RESULTS

Total throughput, packet loss rate, and the average content delivery time when changing the number N of CCN routers are shown in Fig. 2. In this figure, the request rate λ is varied from 1 to 6 [content/s].

These results show that the throughput increases gently rather than linearly when N exceeds around 30 for $\lambda = 6$ [content/s], that the packet loss rate and the average content delivery time rapidly increase when N exceeds around 25 regardless of the request rate λ.

To investigate the performance bottleneck of CCNx when, in particular, the network is large and the request rate is high (i.e., $N = 30$ and $\lambda = 6$ [content/s]), we performed a system-wide profiling using OProfile, a system-wide statistical profiling tool for Linux. Table I is a profiling result, which shows the five most time-consuming functions in our experiment. This table indicates that `sha256_block_data_order` function, which is a part of OpenSSL library, utilizes approximately 30% of the CPU time, which is apparently the performance bottleneck in CCNx. In CCNx, `sha256_block_data_order` function is invoked for every Data packet reception to check the validity of the Data chunk using digest-based authentication.

V. ESTIMATING THE IMPACT OF HARDWARE OFFLOADING

One possible solution for improving the scalability of CCNx, when a number of CCN router slices are constructed on a single physical computer, is hardware offloading of such CPU-intensive processings.

In what follows, we therefore investigate how the scalability of CCNx in terms of the number of nodes can be improved with hardware offloading of the Data-chunk digest computation at CCN routers.

To estimate the performance improvement with hardware offloading, processing of the Data-chunk digest computation at all CCN routers are disabled by bypassing a function invocation for `ccn_digest_ContentObject` from the `process_incoming_content` function in the `ccnd` dameon.

Total throughput, packet loss rate, and the average content delivery time when changing the number N of CCN routers are shown in Fig. 3. This figure shows the results of experiments for $\lambda = 6$ [content/s]. Other conditions are the same with those of Fig. 2 except that the Data-chunk digest computation at all CCN routers are bypassed. In this figure, lines labeled as *virtual offloading* show the results with bypassed Data-chunk digest computation, and ones labeled as *original* the results of the original CCNx. These results show that, with hardware offloading of the Data-chunk digest computation, improvements in the throughput and the packet loss rate are modest (i.e., approximately 7% in both cases). On the contrary, one can also find from these results that the average content delivery time decreases by approximately 25%, which seems to be a significant reduction.

VI. CONCLUSION

In this paper, we have investigated the scalability of CCNx, an open-source CCN implementation, in terms of the number of nodes. As performance metrics, we have measured the total throughput of content delivery, the packet loss rate in the network, and the average content delivery time. Consequently, we have shown that, in our experiments, the CCNx performance degrades when the number of CCN routers on a single physical computer exceeds around 25, and that the performance bottleneck was the Data-chunk digest computation at CCN routers.

As future work, we are planning to investigate the scalability of CCNx in terms of the link speed and the number of contents stored in the network.

ACKNOWLEDGEMENTS

This work was partly supported by JSPS KAKENHI Grant Number 25280030.

REFERENCES

[1] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proceedings of the Fifth International Conference on emerging Networking EXperiments and Technologies (CoNEXT 2009)*, Dec. 2009, pp. 1–12.

[2] "Project CCNx," http://www.ccnx.org.

[3] D. Perino and M. Varvello, "A reality check for content centric networking," in *Proceedings of 2011 ACM SIGCOMM Workshop on Information-Centric Networking (ICN)*, Aug. 2011, pp. 44–49.

[4] R. Chiocchetti, D. Rossi, and G. Rossini, "ccnSim: an highly scalable CCN simulator," in *Proceedings of 2013 International Conference on Communications (ICC)*, Jun. 2013, pp. 2309–2314.

[5] H. Yuan, T. Song, and P. Crowley, "Scalable NDN forwarding: Concepts, issues and principles," in *Proceedings of the 2012 21st International Conference on Computer Communications and Networks (ICCCN)*, Jul. 2012, pp. 1–9.

[6] T. Tang, "High performance content centric networking on virtual infrastructure," Master's thesis, University of Toronto, 2013.