Performance Measurement of Name-Centric Content Distribution Methods

Haowei Yuan and Patrick Crowley
Computer Science and Engineering
Washington University in St. Louis
{hyuan, pcrowley}@wustl.edu

ABSTRACT

The recently proposed Named Data Networking (NDN) architecture and the widely deployed HTTP infrastructure both support content distribution in a name-centric fashion. In this paper, we evaluated the content distribution performance of NDN-based and HTTP-based content distribution solutions.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Store and forward networks

General Terms

Performance, Measurement

1. INTRODUCTION

As content distribution traffic keeps increasing, efficient content distribution methods are needed. On the one hand, several recently proposed content-centric network architectures support content distribution intrinsically. *Named Data Networking* (NDN) [4] is one of these architectures. NDN names packets rather than end-hosts, which enables the caching of packets in the network for future distribution. On the other hand, HTTP is recognized as a name-centric protocol. A significant portion of today's content distribution services leverage widely deployed HTTP infrastructures, such as web servers and caching proxies [5]. HTTP is the practical solution for efficient content distribution. However, there is a lack of knowledge on the performance difference between NDN-based and HTTP-based solutions. In this paper, 1) we evaluate NDN-based and HTTP-based file distribution performance in a real world testbed with more than 40 hosts; 2) we make a modest modification on the current NDN prototype implementation which nearly doubles system throughput.

2. BACKGROUND

2.1 Named-Data Networking

In NDN, each packet has a unique name, and it is forwarded at each NDN node based on a lookup on its name. An NDN node consists of three logical components, the *Forwarding-Information Base* (FIB), the *Content Store* (CS) and the *Pending-Interest Table* (PIT). NDN packets are either *interest* packets or *data* packets. Content consumers send interest packets to fetch data. For each incoming interest packet, an NDN node first checks whether the requested content can be satisfied from its local CS, and it then either sends back a data packet or forwards this interest packet. For an incoming data packet, an NDN node will cache it and then check the PIT. It will then forward the data packet if there are pending interests that can be satisfied by this content.

CCNx [2] is an NDN prototype software implementation developed by PARC. The key component of CCNx is the *ccnd* daemon, which supports packet forwarding and caching. The current ccnd program runs as an overlay network on top of IP. The CCNx implementation we evaluated in this paper was released on March 8, 2011.

2.2 HTTP-Caching System

The HTTP protocol is recognized as a name-centric protocol, where the names are the URLs. As a result, HTTP packets can be cached and redistributed in the network by utilizing HTTP caching proxies. With the widely deployed HTTP infrastructure, such as web servers and caching proxies, HTTP is a practical solution for content distribution. It should also be noted that there has been a resurgence of HTTP traffic in today's internet [5]. In this paper, we configured an HTTP-caching system using the popular http web server *lighttpd* [1] and HTTP caching proxy *Squid* [3].

3. PERFORMANCE

Although there are a lot of similarities between NDN-based and HTTP-based content distribution methods, there is a lack of performance comparisons between these two approaches. We evaluated the file distribution performance using these two methods in the Open Network Laboratory (ONL) [6].

3.1 Experiment Setup

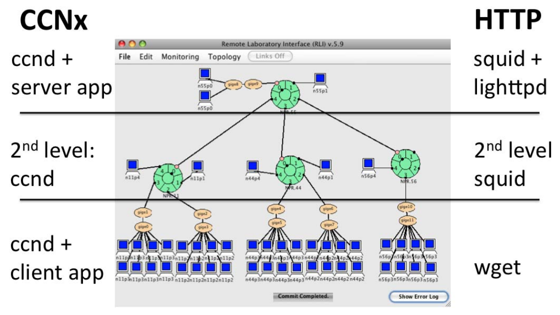

Figure 1: Testbed Topology

The testbed includes 40 client hosts, 1 server and two levels of intermediate nodes, which can be either CCNx routers or Squid proxies. Every 8 clients form a *cluster* and share a common second level intermediate node. All the hosts in the testbed have 1 Gbps interfaces. Figure 1 shows the experiment setup. In the CCNx case, the client host runs the ccnd daemon and the *ccncatchunks2* application to download the file. The second level intermediate nodes, which are CCNx routers in this case, run the ccnd daemon. The server runs a ccnd daemon and a file server program we developed. For the HTTP case, the clients run the *wget* program to fetch data. The second level nodes run Squid proxies and the server runs a lighttpd web server. To perform file distribution, a file is stored in the server, and the clients will try to fetch the file simultaneously. The file *downloading time* (DT) is the time from when the client application sends the request until the file is downloaded completely, and DT is recorded on each client host. Based on our experience, CCNx is much slower than HTTP caching system. As a result, in our experiment, the CCNx clients download a 10MB file and HTTP clients download a 100MB file.

3.2 CCNx vs. Squid

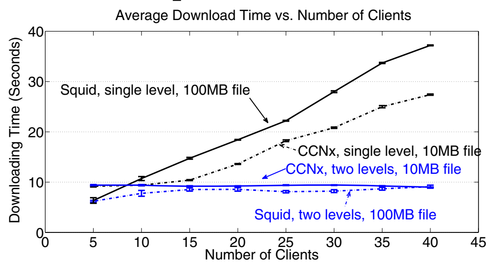

Figure 2: CCNx vs. Squid (90% CI)

The factors we studied in this performance evaluation are the levels of cache and the number of clients in the system. In the single level case, all the clients connect through the top level CCNx router or Squid proxy to the server, and the second level intermediate nodes are unused. In the two level case, clients are connected via a second level cache. Every cluster has the same number of active clients, i.e., the number of clients that will request the file. We start with 5 clients requesting the file, with 1 client in each cluster. The number of active clients is increased by one until all of the clients are active. For each configuration, we run the experiment 5 times and the average DT across all the active clients are reported. Figure 2 shows the results. For the single level cache case, the DT of using Squid increases linearly with the number of clients. This is due to the fact that the Squid implementation is very efficient thus the proxy host's physical outgoing link capacity (1Gbps) becomes the bottleneck. In the CCNx case, the downloading time stays relatively constant until there are more than 10 hosts. What happens is when there are fewer than 10 hosts, the ccnd daemon is not saturated, leaving our own naive file server program as the bottleneck. When there are more than 10 hosts, the ccnd daemon is saturated, and thus the DT increase linearly as the number of clients increases. For the two-level cache case, both Squid and CCNx perform much better, as the physical link bottleneck and the ccnd daemon processing bottleneck disappear. Overall, the current CCNx implementation is about 10 times slower than Squid. This is partially due to the fact that CCNx is still in an early development stage, while Squid is a mature and widely used system.

3.3 Improving CCNx Performance

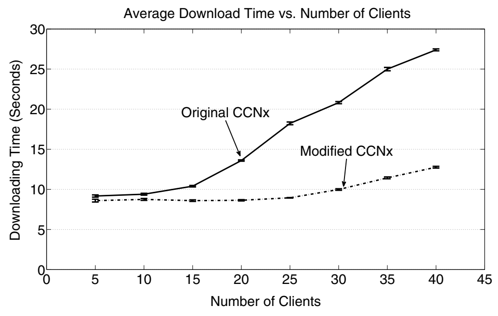

Figure 3: Performance Improvement (90% CI)

We profiled the ccnd daemon when it was saturated using *Gprof*. To our surprise, more than 50% of the time was spent on functions related to packet name decoding. In particular, the *ccn_skeleton_decode* function, which is the lowest level packet decoding function, takes 47.34% of the time. Upon further investigation, it turns out that the CCNx implementation chooses to store cached names in XML-encoded wire format, and as a result, every content store lookup requires $log(n)$ of the name decoding on average, where n is the number of content stored in the CS. Our modified CCNx stores cached names in decoded format. This simple change almost doubled the system throughput as shown in Figure 3.

4. CONCLUSION

In this paper, we evaluated the file distribution performance of NDN-based and HTTP-based solutions and found that the throughput of the current NDN prototype implementation is much slower than Squid. This is a first step in evaluating their performances as there is only a single file being distributed. We plan to develop traffic generators for these two systems and further evaluate their performance. We will also explore methods to optimize the NDN prototype implementation.

References

[1] lighttpd: http://www.lighttpd.net/.
[2] Project ccnx: http://www.ccnx.org/.
[3] Squid: http://www.squid-cache.org/.
[4] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard. Networking named content. In *CoNEXT '09*, pages 1–12, New York, NY, USA, 2009. ACM.
[5] L. Popa, A. Ghodsi, and I. Stoica. Http as the narrow waist of the future internet. In *Hotnets '10*, pages 6:1–6:6, New York, NY, USA, 2010. ACM.
[6] C. Wiseman, J. Turner, and P. Crowley. The open network laboratory. In *ACM SIGCOMM*. ACM, 2009.