Performance Measurement of the CCNx Synchronization Protocol

Hila Ben Abraham
Computer Science and Engineering
Washington University in St. Louis
hila@wustl.edu

Patrick Crowley
Computer Science and Engineering
Washington University in St. Louis
pcrowley@wustl.edu

ABSTRACT
The CCNx Synchronization protocol is one of the protocols published under the CCNx distribution, and is used to synchronized shared collections of 2 CCNx neighbors. In this paper, we evaluated the performance of the synchronization protocol over different topologies and different network scales.

Categories and Subject Descriptors
C.2.2 [**Network Protocols**]: Applications

General Terms
Measurement, Performance.

Keywords
CCNx synchronization protocol; Named Data Networking; Content-Centric-Networking; CCNx; ONL;

1. INTRODUCTION
Content-centric-networking project (CCNx) [1] is a software implementation of the content-centric networking approach. The motivation behind the CCNx synchronization protocol is to keep a collection of information synchronized between 2 CCNx nodes. An example of a possible synchronized collection is a private music directory that should be kept up-to-date in a user's smartphone and a personal computer. Another example of possible synchronized information is the network topology graph that should be similar in all the network routers to enable the correct operation of common routing protocols such as OSPF and ISIS [4]. Unlike the common approach, CCNx synchronization protocol keeps a synchronized collection up-to-date by sending the differences of the collection rather than the entire content. The benefit of sending the differences over sending the entire collection might clear when it comes to the network traffic of large collections [2]. However, the effect of synchronizing the differences on the synchronization time is unclear. In this work, we started to study the synchronization protocol and to explore the time it takes to synchronize a collection over different topologies and different network scales.

2. BACKGROUND
2.1 Named-Data Networking
Named Data Networking (NDN) is a recently suggested Internet architecture that efficiently supports content distribution. Unlike the current Internet architecture, NDN takes the content-centric approach by delivering a packet according to its content rather than to a pre-defined destination address. To receive data, a consumer expresses an interest packet that contains the requested data name. The NDN router forwards the interest packet to the next hop by looking into the Forwarding Information Base (FIB) table, finding the longest name prefix match, and sending the interest to the attached face. On the forwarding of an interest packet, the router saves a copy of the interest and its incoming face in its Pending-Interest table (PIT). The producer responds to an interest packet by sending a corresponded data packet. The router uses the PIT to forward the data packet back to the consumer on the same incoming path. In addition to the forwarding of interests and data packets, the NDN router stores the incoming data packets in its local Content Store (CS). On the reception of an interest packet, the NDN router first checks for the content in its local CS. If the content exists, the NDN router generates a data response that contains the stored information. Otherwise, the router forwards the packet according to the interest name. [3]

2.2 CCNx project
CCNx is an open source distribution developed by PARC. Due to the fact that CCNx implementation includes the main attributes of the NDN architecture, it is used as one of the NDN evaluation platforms. The CCNx consists of 2 main software components: ccnd and ccnr. The ccnd implements the forwarding and the routing parts of the CCNx while the ccnr implements the CCNx repository. A repository can be used by an application or by the network to preserve required data, such as file content or an application state.

2.3 CCNx Synchronization protocol
The Sync Agent is the CCNx software module that implements the Synchronization protocol, and is part of the CCNx Repository process. The synchronization protocol synchronizes all the content items that share the same name prefix, and belong to a pre-defined collection. The Sync Agent stores the collection content prefixes using a tree structure called the sync tree. Each node in the sync tree holds a combined hash representing the arithmetic sum of the names hashes in that node, and the combined hashes of its child nodes. To stay up-to-date, each Sync Agent sends a periodic Root Advise interest to all of its neighbors, including its root hash. On the reception of a Root Advise interest, the Sync Agent compares its local root hash to the remote root hash. If the root hashes are equal, there is no reply to the incoming Root Advise interest, otherwise, the remote Sync Agent express a Node Fetch to request the content of each different hash element.

3. EXPERIEMENT SETUP
We used the Open Network Laboratory (ONL) [5] to evaluate the performance of the sync protocol, and to measure the time consumed by the protocol to synchronize a repository collection

(also known as a slice) over different topologies and different network scales. Since the repo saves its content in 4096 kbyte chunks, we explored the effect of the content size on the synchronization time. Table 1. summarize our experiment factors.

Table 1. Experiment factors

Factors	Levels
Slice Status	Empty/ Not Empty
Device Count	6/12
Network Topology	Fully connected mesh / chain
Content Size	1 chunk of data / 19 chunks of data

We developed a script that used the ONL interface, and defined the CCNx overlay topology according to the configured factors. We used the ccnputfile application to write a local content to a selected CCNx host. We used the ccnsyncwatch application to report on the arrival timestamp of each file chunk to all the other participating nodes. We calculated the timestamps differences and reported on the longer synchronization period. It is important to mention that our goal was to evaluate the synchronization time without the time it takes to write content into the repo, and therefore the writing timestamp reported by our setup was recorded once the ccnputfile finished the writing operation, and not before that. At the time we performed our first experiments, we discovered that the ccnsyncwatch delays the notification of the first item added to an empty repo, and therefore we used tcpdump captures to verify the accurate arrival timestamps. We performed each experiment 3 times and reported on the average.

4. RESULTS

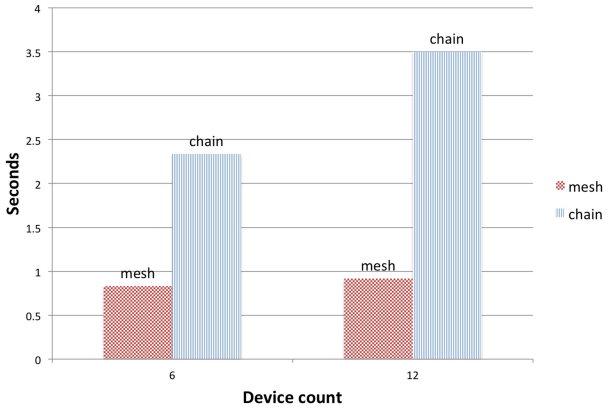

Figure 1. mesh VS. chain over different network scales

Our results show that the slice status and the examined content size don't have a significant effect on the synchronization times. Figure 1. shows the average synchronization time for the examined topologies and network scales. We can see that for each network scale, the time it takes to synchronize content over the chain topology is longer than the time it takes to synchronize the same content over the mesh topology. In addition, our results show that the network scale affects the synchronization time of the chain topology, while it has almost no effect on the fully connected mesh topology. Our results also show that the examined sizes of the Content Size factor have no effect on the synchronization time.

5. CONCLUSION

In this paper, we explored and measured the synchronization time of the CCNx synchronization protocol. For the examined scales, we found that it takes longer to synchronize a chain topology than it takes to synchronize a fully connected mesh topology. We also found that due to the broadcast notifications of the Root Advise interest, the network scale has almost no effect on the synchronization time of the fully connected mesh topology.

6. FUTURE WORK

To better understand the performance of the synchronization protocol, we continued to explore large-scale networks and additional topologies. We observed a large variation in our results, and as part of our preliminary exploration, we identified that the insertion time affects the synchronization time. We suspect that the insertion time has an effect on the results due to a different memory state, hence, PIT entries and CS content. This additional "insertion time" factor may prove to have a substantive impact on the performance of large-scale networks, and it is the subject of future work. In addition, we plan to continue and explore the effect of the Content Size factor for larger content.

7. REFERENCES

[1] Project ccnx: http://www.ccnx.org/.

[2] D. Eppstein, M. T. Goodrich, F. Uyeda, G.VargheseIn. "What's the Difference? Efficient Set Reconciliation without Prior Context", In Proceedings of SIGCOMM, 2011.

[3] 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.

[4] V. Jacobson, R. L. Braynard, T. Diebert, P. Mahadevan, M. Mosko, N. H. Briggs, S. Barber, M. F. Plass, I. Solis, E. Uzun, B. J. Lee, M. Jang, D. Byun, D. K. Smetters and J. D. Thornton, "Custodian-Based Information Sharing", in IEEE Communications Magazine 07/12

[5] C. Wiseman at al., The open network laboratory. In ACM SIGCOMM. ACM, 2009