ndn||mem : an Architecture to Alleviate the Memory Bottleneck for Named Data Networking

Mostafa Rezazad
National University of Singapore
mostafa@comp.nus.edu.sg

Y.C. Tay
National University of Singapore
dcstayyc@nus.edu.sg

ABSTRACT

Named Data Networking (NDN), as one of the candidates for a future Internet, has the potential to overcome many of the current Internet's difficulties (e.g., security, mobility, multicasting, etc.). However, there are also many open problems to be taken care of before it becomes operational.

One of the major concerns is that NDN may be memory-intensive. Using the router's buffer as cache adds a search time to all requests. If the cache hit rate is not high enough (considering current buffer sizes and traffic rates), cache search time may be a burden on the system instead of solving any problems.

Huge fast memories are not affordable yet. To balance the memory speed with incoming rate to a router, we propose a memory architecture, called **ndn||mem**, along with a content caching strategy, called **CCndn**. Using these two ideas, only a small fraction of packets need to engage in a time-intensive cache search process.

Simulation results demonstrate the efficacy of our architecture.

Categories and Subject Descriptors

C.2 [**Computer Communication Networks**]: Network management

Keywords

Content Networking; Cache Policy; Memory Architecture

1. INTRODUCTION

Content Centric Networking (CCN) [2] proposes to rectify the architectural problems of the current Internet by naming data rather than hosts. Also called Named Data Networking (NDN), content is broken into **chunks** that are individually addressable by name. NDN routers act like caches for these chunks. Since the Internet has a huge amount of content, router memory may have to be large for caching to be effective. Routers that require a large amount of fast memory would be expensive and power-hungry [4].

Feasibility of the architecture using the current memory technology is questioned in [4]. Somaya et al., proposed an architecture in [1] to match the forwarding pace with line speed. In [7] the effectiveness of forwarding algorithm is studied. Memory look up algorithm is considered as a serious problem for NDN router to match with the line speed.

None of the above studies considered pipelining among different functional units. In this study we compare our new architecture (ndn||mem) with a pipelined version of the NDN design (i.e.,*serial* architecture). The number of accesses to memory to look up data depends on the content name length [4]. We show that the ndn||mem architecture outperforms serial architecture even if the number of accesses to memory could be reduced to only one access. Of course, our architecture will benefit more if the number of accesses to memory increases.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profi or commercial advantage and that copies bear this notice and the full citation on the firs page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specifi permission and/or a fee. Request permissions from permissions@acm.org.
CoNEXT Student Workshop'13, December 9, 2013, Santa Barbara, CA, USA.
Copyright 2013 ACM 978-1-4503-2575-2/13/12 ...$15.00.
http://dx.doi.org/10.1145/2537148.2537154.

2. ARCHITECTURE

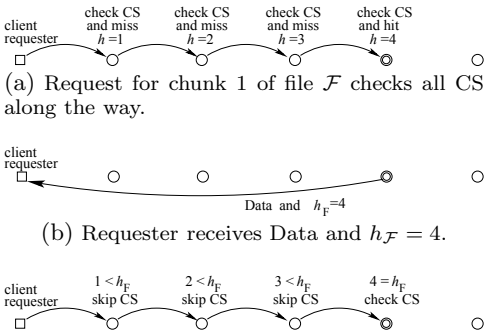
We observed that queuing delay for memory is a major problem. Parallelism is our solution to rectify the problem.

The forwarding plane of an NDN router contains three main tables, *Forwarding Information Base (FIB)*, *Pending Interest Table (PIT)* and *Content Store (CS)*. The functionality of the FIB is similar to the IP routing table. The PIT provides the reverse path information for the data to route back to the requester(s). The CS keeps a copy of the data chunks to satisfy future requests for the same data chunk.

For CS, we tried to reduce the size of the queue for the CS index table by checking it only for chunks that are likely to be there. One assumption is that a chunk is likely to be in CS only if previous chunks of the same file were found there. Interest (i.e. request) for the first chunk of a file searches all CS along its path. The returned data packet contains the hop distance to the router where the chunk was found. The rest of the Interests for the file from the same requester will carry this hop-distance information. A router will decide to search the CS index table only if its hop distance from the requester matches that information (Fig. 1).

In this way, Interests that are unlikely to find their data from a router do not waste time searching the index table. We emphasize that multipathing is not a concern here. If the router cannot find the correct path, even searching the entire path will not help.

The next idea is a concurrent search on PIT and FIB. Since the NDN paradigm names content and there is a gigantic amount of content on the net, FIB needs to be large



(a) Request for chunk 1 of file $\mathcal{F}$ checks all CS along the way.

(b) Requester receives Data and $h_{\mathcal{F}} = 4$.

(c) Next Interests for file $\mathcal{F}$ only checks CS when $h = h_{\mathcal{F}}$.

Figure 1: Data for chunk k of $\mathcal{F}$ passes hop count $h_{\mathcal{F}}$ to request for chunk $k+1$. The latter checks a CS only when its hop count matches $h_{\mathcal{F}}$.

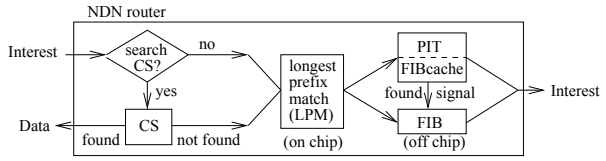


Figure 2: The ndn||mem architecture. It allows Interests to bypass CS, and possibly abort a visit to FIB.

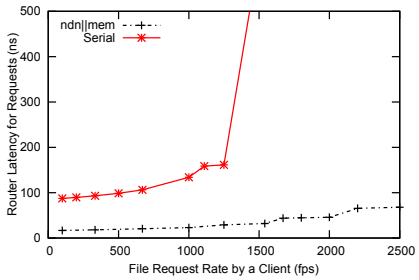


Figure 3: ndn||mem has much lower router memory latency than serial, and postpones router saturation.

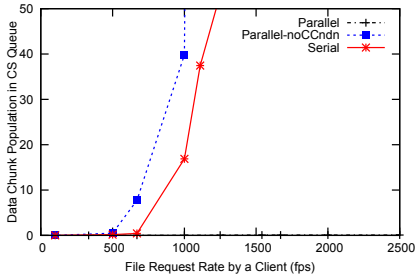


Figure 4: Without CCndn, the CS queue at the router can be saturated by Data chunks.

to make effective decisions. Therefore, we expect the FIB memory technology to be much slower than PIT. The scenario can be modeled as a system with one queue and two servers where one of the servers is faster than the other. Thus, in a parallel search of the two tables, the result of PIT search will be available earlier than the result of FIB search. Finding an entry in PIT will send a signal to FIB to stop its search. Otherwise the search on FIB determines the interface to send out the Interest. Since FIB is slower than PIT, the performance of the system can be further improved by using a small *FIB-cache* [3] at the PIT table.

Using the same assumption that routing information of the different data chunks of a file are the same, the obtained routing information for the first data chunk can be cached in PIT table and be used for the rest of the Interests for the same file. In this way FIB is going to be examined only for the first Interest for a file. Fig. 2 illustrates the architecture.

Our experiments show that Interests are not the only packets that populate the CS index queue. To cache data chunks, the CS index table needs to be updated. Thus, to handle the incoming traffic the router must process only a fraction of its traffic (both Interests and chunks). Any kind of caching policies could be used, but to be consistent with ndn||mem, our CCndn caching policy uses the hop distance to cache different parts of a file in different routers along the path from a requester to the source of the file.

Previous traffic measurements have shown that users have a tendency to abort downloads [5, 6]. This means chunks at the head of a file are more popular than those at the tail. CCndn therefore divides a file into **segments**, and distributes these segments along the path between requester and source. The chunks for a segment are cached at the same router, the popular segments (with chunks from the file head) are cached near the requester, while the tail segments are cached near the source.

The segment size is fixed, so every segment (except the last) has the same number of chunks. Depending on the hop distance of an Interest to the source of the file, routers might cache more than one segment. Knowing the segment size and hop distance, a router can decide to cache the received data chunk or just simply send it out (here caching the data chunks refers to updating the index table).

The ideas are simulated on an Abilene[1] like network and Fig. 3 shows the effectiveness of ndn||mem and CCndn in reducing router memory latency to forward Interests. Fig. 4 depicts the effectiveness of CCndn policy compared with serial and parallel architectures, both without using CCndn. A larger queue length translates into larger packet processing delay and earlier router saturation. Our simulation results also show that the hop distance to data chunk for the two architecture are identical and bypassing CS for some of the Interests does not reduce the effectiveness of using cache inside the router (not presented here due to lack of space).

3. CONCLUSION

Memory intensiveness has been considered as one of the major concerns for NDN design. In this work-in-progress study we introduced a new router design to alleviate the problem. Insertion of FIB-cache into PIT table and conducting parallel search on FIB and PIT both save considerable time of router. Moreover, using hop distance to bypass CS search for Interests with less hit probability reduces the CS index table queue length. The CCndn caching policy helps to further reduce the queue length by caching only a subset of data chunks in each router.

[1] *http : //nic.onenet.net/technical/category5/core_topology.htm*

4. REFERENCES

[1] S. Arianfar, P. Nikander, and J. Ott. On content-centric router design and implications. In *Proc. ReArch*, pages 5:1–5:6, 2010.

[2] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard. Networking named content. In *Proc. CoNEXT*, pages 1–12, 2009.

[3] Y. Liu, S. O. Amin, and L. Wang. Efficient FIB caching using minimal non-overlapping prefixes. *SIGCOMM Comput. Commun. Rev.*, 43(1):14–21, Jan. 2012.

[4] D. Perino and M. Varvello. A reality check for content centric networking. In *Proc. ICN*, pages 44–49, 2011.

[5] D. N. Tran, W. T. Ooi, and Y. C. Tay. SAX: A tool for studying congestion-induced surfer behavior. In *Proc. PAM Conf.*, 2006.

[6] H. Yu, D. Zheng, B. Y. Zhao, and W. Zheng. Understanding user behavior in large-scale video-on-demand systems. In *Proc. EuroSys*, pages 333–344, 2006.

[7] H. Yuan, T. Song, and P. Crowley. Scalable NDN forwarding: Concepts, issues and principles. In *Proc. ICCCN*, pages 1–9, 2012.