IEICE TRANS. INF. & SYST., VOL.E105–D, NO.4 APRIL 2022

LETTER

NFD.P4: NDN In-Networking Cache Implementation Scheme with P4

Saifeng HOU[†a)], **Yuxiang HU**[†], **Le TIAN**[†], ***and*** **Zhiguang DANG**[††], ***Nonmembers***

SUMMARY This work proposes NFD.P4, a cache implementation scheme in Named Data Networking (NDN), to solve the problem of insufficient cache space of prgrammable switch and realize the practical application of NDN. We transplant the cache function of NDN.P4 to the NDN Forwarding Daemon (NFD) cache server, which replace the memory space of programmable switch.

key words: *NDN, in-networking cache, NFD, programming protocol-independent packet processors (P4)*

1. Introduction

Named Data Networking is a new network architecture. Different from today's IP network, NDN replaces the IP address with the name data content, and use it for routing and addressing. This change causes the processing of Data packets in NDN to be significantly different from that of IP. In order to meet this challenge, scholars have proposed many software-based NDN router solutions. And even highly optimized designs tailored to specific hardware platforms have limited performance, which hinders applications.

Rencently, the emergence of programmable switch chips and languages to program them, such as P4, have brought the possibility for the actual deployment of NDN. As a programming language, the idea of P4 is to use software ideas to logicalize message forwarding, the compiled configuration files are loaded into the forwarding layer equipment (switches, network cards, firewalls, etc.) through the runtime interface to guide the underlying hardware to complete customized Data plane function [1]. Rui Signorello et al. [2] proposed NDN.P4 — the first attempt to use P4 language to implement NDN routers. They implemented part of the NDN logic functions with the actual P4 language specifications. On this basis, Miguel et al. [3] proposed NDN.P4_16, which improved Signorello's attempt from two aspects, one is the scalability of FIB design, and the other is the expansion of NDN functions, including content storage (software implementation) and multicast function. However, neither of the two documents actually deployed NDN's content caching technology.

The communication of NDN is driven by data consumers, NDN supports Interest and Data packets and provides name-based routing and forwarding function that allows direct communication using application data names over both packets. NDN node includes 3 basic data structures: Content Store (CS), Pending Interest table (PIT), and Forwarding Information Base (FIB). Data forwarding is realized by matching the 3 data structures.

Caching is the key technologies of the NDN [4], an important manifestation of NDN improving network efficiency, as it can reduce the content acquistion delay and reduce the network load. However, the P4 programmable switch is composed of programmable ASIC, FPGA, NPU or CPU. There is no too much memory space to cache data. At the same time, the P4 language pays attention to the protocol-independent characteristics in the design, and lacks support for memory space. Using the register to realize the cache has the best performance, but it is easy to cause competition for memory with PIT, which leads to a decrease in forwarding efficiency and cannot solve the problem of insufficient memory space.

In order to solve the problem of insufficient memory space for NDN cache when implementing NDN data plane functions in the P4 switch, this letter utilizes NFD cache server to implement NDN cache. NFD [5] abstracts the underlying network transmission mechanism into NDN Faces, and maintains basic data structures such as CS, PIT, and FIB to implement Data packet forwarding logic. This letter proposes NFD.P4, which aims to solve the problem of insufficient cache space in P4 switch, and achieves NDN by transplanting the cache function to the NFD cache server. Then we redesign the process of Interest and Data Packets with P4. The main contributions are summarized as follows:

- We achieve NDN's content caching in NDN.P4 by utilizing modified NFD architechture as a cache server, which is served as memory space.
- We modify and optimize P4 language, which defines the process of Data and Interest packets, to implement NDN in-networking cache.
- We deploy actual network environment to verify NDN function, especially caching.

2. Architecture and Implementation

We use P4 programmable language to apply for a register array space on the data plane to record the local cache. The actual cache function is provided by the NFD cache server by setting the cache port, which is defined in P4 to forward

Manuscript received October 30, 2021.
Manuscript revised November 30, 2021.
Manuscript publicized December 27, 2021.
[†]The authors are with Information Engineering University, Zhengzhou, Henan, China.
[††]The author is with Zhejiang Lab, Hangzhou, Zhejiang, China.
a) E-mail: 18326991660@163.com
DOI: 10.1587/transinf.2021EDL8100

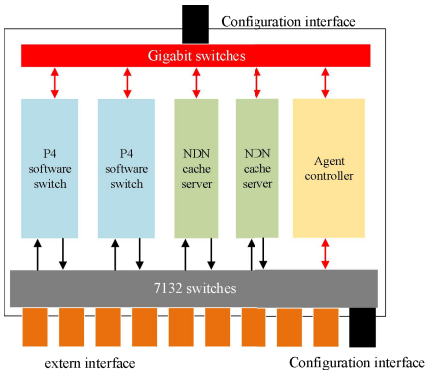

Fig. 1 The basic function platform of NFD.P4.

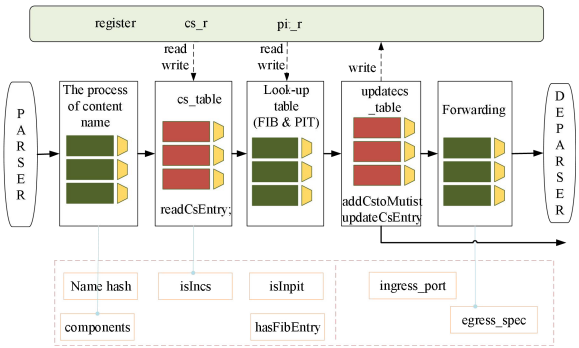

Fig. 2 The updated ingress pipeline of NFD.P4.

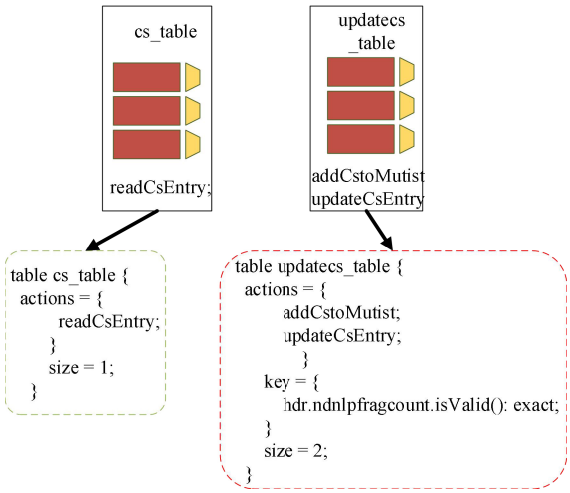

Fig. 3 The definition of ca_table and updatecs_table.

packets that need to cache. And P4 switch will forward these Data Packets to the NFD server from the cache port. On the contrary, the NFD server will response the requested data cached in it from the cache port too. From the point view of the data plane, P4 switch only records the cache status, and has no actual data storage. The stored data is forwarded to the NFD cache server to realize NDN logic function.

As shown in Fig. 1, the basic function platform of NFD.P4 consists of P4 software switches, NDN cache server, agent controller, 7132 switch, gigabit switch, configuration and extern interfaces. It is the NDN processing platform, to implement all the NDN function. NFD.P4 is the processor that consists of P4 software switches, NDN cache server, agent controller and interfaces. It focuses on NDN cache function, to implement cache function with P4.

The agent controller is used to deliver the flow table. The P4 software switch mainly deals with Interest packets and Data packets. There are three types of final processing results of the Interest packet: 1) Discarding, including the discarding caused by the same Data packet request and FIB table miss; 2) Forwarding to the NFD cache server, that is, the CS table contains the Data packet item requested by the Interest packet; 3) Forwarding to an external port, that is, the platform does not have the Data packet cache requested by the Interest packet, and currently no other Interest packet has made the same request and is waiting for the Data packet to return. The processing of Data packets is divided into two situations. 1) If the Data packet comes from the internal port (i.e., the Data packet comes from the NFD cache server), the PIT table is directly queried, and the Data packet is forwarded according to the port matched by the PIT; 2) If the Data packet comes from an external port, then the port corresponding to the NFD cache server is added to the port to be forwarded, and then the PIT table is queried, and forwarding is performed according to the port matched by the PIT and the port corresponding to the NFD cache server.

The updated ingress pipeline of NFD.P4 shows in Fig. 2. The open-source project 'NDN.P4' did not take CS storage function into account, so we need to add ***cs_table*** to store the contents of recieved Data packet. As Fig. 3 shows, ***readCsEntry*** reads out wether the register has the content prefix, and puts it into flow_metadata ***isIncs*** for further processing of packet. ***updatecs_table*** will update cs table according to the presence or absence of the current content prefix. If the Data packet is recorded in cs table, it will perform ***addCstoMutist*** to update output port. If not, it will add a CS entry to record the content prefix of the Data packet by performing ***updateCsEntry***.

The normal NFD's processing logic for the arrival of a Data packet is to check the PIT table to see whether there are PIT entries that can be satisfied by this Data packet. If this Data packet cannot satisfy any PIT entry, it is unsolicited and will be discarded. Otherwise, the Data packet will be added to the cs_table, and the Data packet will be forwarded according to the PIT entry. As it is showed in Fig. 4, for the NFD cache server, even though there are no PIT entries that can be satisfied by the coming Data packet, the Data packet will also be cached, instead of being discarded. In the NFD cache server, only the function of caching and reading the cached Data packet is used. Therefore, when the Data packet arrives, the processing method is to directly insert it into the cs_table, even if it is unsolicited Data packet. After receiving the Interest packet from the switch, the normal process will be followed without any modification.

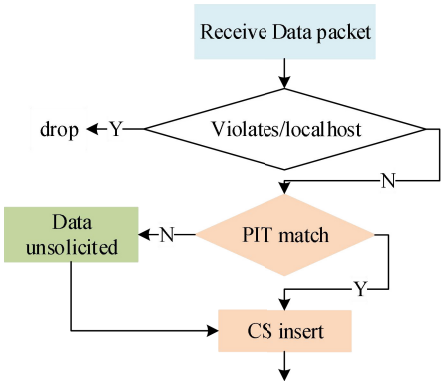

Fig. 4 The processing flow of data packet in cache server.

3. Experimental Evaluation

As Fig. 5 shows, the experimental environment includes 3 hosts as NDN terminals, 2 P4 switches that support the cache function, 2 access switches, 2 networking switches and 3 controllers. The NDN terminal installs the Ubuntu 18.04 system, the P4 switch uses the Tofino chip, the access switch uses the Zhaoxin switch, and the inter-domain and intra-domain controller uses the ONOS distributed controller. The bandwidth of the access network is 10Gb/s, and that of the core network is 100Gb/s. Cache capacity of the NFD cache servers is 65536, which is defined with P4 switches and limited by P4 switches. Two NDN autonomous domains are connected through the core Network, NDN autonomous domain 1 serves as the request terminal, and NDN autonomous domain 2 serves as the responding terminal. Data packets pass through the core network, and by monitoring the core network traffic, the advantages of in-network caching can be reflected.

The NDN-host requests files including text, pictures, and videos sucessfully, it verifies that the experimental network can implement forwarding of various file types. Each type of file is requested with 20 different prefix names, and the file size increases gradually. Then, a second request for a file with the same prefix name is sent to verify the function of cache by comparing the response time of packet request and the traffic load of core network.

The response time of the packet request refers to the response time from the start of sending the Interest packet request to the end of the Data packet response, and the same Interest packet request is sent twice. By comparing the response time of the first and second packet requests, that is, the response time with or without cache, it reflects the advantages of in-network caching. Figure 6 shows the comparison result of the response time of the first packet request and the response time of the second packet request when transferring files with different sizes. It can be seen from the figure that the response time with caching is about 60% faster than that without caching. The improvement of response speed by caching verifies the effectiveness of NDN caching.

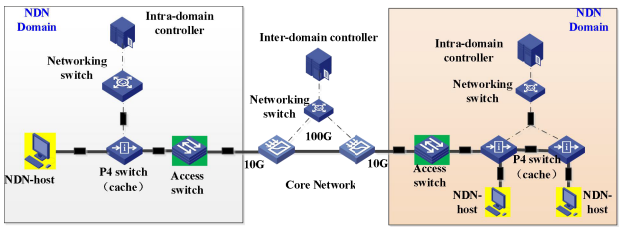

Fig. 5 The topology of experiment.

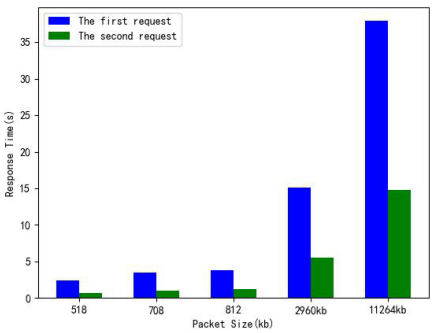

Fig. 6 The comparison of the response time of packet request.

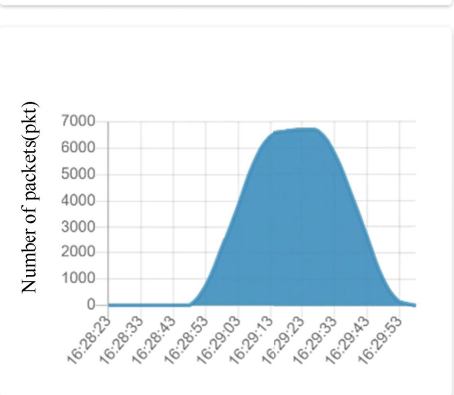

Fig. 7 The traffic rates statistics of core network with cache.

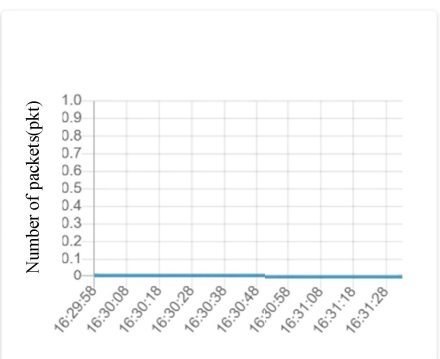

Fig. 8 The traffic rates statistics of core network without cache.

By monitoring the traffic load of the core network, it can also reflect the advantages of in-network caching. Figure 7 is the flow of the core network that sends the Interest packet for the first time, and Fig. 8 is the core network flow that sends the same Interest packet for the second time. It can be found that the core network traffic is 0 at the second request, because the Data packet is directly obtained from the cache of the P4 switch during the second request, without going through the core network at all, which can reduce the load on the core network.

4. Conclusion

In order to realize the actual deployment of NDN logic functions in programmable networks and solve the problem of insufficient memory space in programmable switches, this letter proposes an NFD cache server to implement NDN's in-network caching function. Experimental results show that the response time of packet requests can be reduced by about 60% through caching, which proves the effectiveness of caching in the NDN network, it can improve response speed and reduce network load. In the future, a caching strategy can be designed to optimize the performance of the network cache and improve the efficiency of the cache.

Acknowledgments

We thank the anonymous reviewers for their insightful feedback. This work was surpported by National Key R&D Project of China under grant 2019YFB1802505.

References

[1] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, and D. Walker, "P4:Programming protocol-independent packet processors," ACM SIGCOMM Computer Communication Review, vol.44, no.3, pp.87–95, 2014. DOI: 10.1145/2656877.2656890

[2] S. Signnorello, R. State, J. François, and O. Festor, "NDN.p4: Programming information-centric data-planes," IEEE NetSoft Conference and Workshops, Seoul, South Korea, pp.384–389, June 2016. DOI: 10.1109/NETSOFT.2016.7502472

[3] R. Miguel, S. Signnorello, and F.M.V. Ramos, "Named data networking with programmable switches," IEEE 26th International Conference on Network Protocols, Cambridge, UK, pp.400–405, Sept. 2018. DOI: 10.1109/ICNP.2018.00055

[4] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, K. claffy, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang, "Named data networking," ACM SIGCOMM Computer Communication Review, vol.44, no.3, pp.66–73, 2014. DOI: 10.1145/2656877.2656887

[5] A. Alexander, "NFD developer's guide," Technical Report NDN-0021 revision 6, https://named-data.net/pulication/techreports/, March 2016.