Implementation of NDN Function Chaining Using Caching for IoT Environments

Yohei Kumamoto
Waseda University
Tokyo, Japan
yoheimmtw@akane.waseda.jp

Hidenori Nakazato
Waseda University
Tokyo, Japan
nakazato@waseda.jp

ABSTRACT

In this paper, we discuss how to implement a mechanism that combines function chaining and cache in Named Data Networking (NDN), an incarnation of information centric networking technology, for real-world IoT environments. We explain our new architecture, called NDN-FC+, for combining function chaining with cache over NDN, and how to extend existing NDN software to support function chaining and caching. The key features discussed in this paper are Interest and Data packet structure, forwarding methods, and naming schemes for a cached content. In particular, it is important to implement the cache, which is one of the major features of NDN. By using the cache, the network will be able to keep contents closer to the users and send them with low latency. Also, by combining function chaining and caching, and caching the content that has been processed by several functions in advance, it will be possible to communicate the processed content without processing. The feasibility of our proposed protocol for caching and forwarding methods is displayed through a prototype implementation. The performance evaluation was performed in a topology that executes the functions chained to the image data from the sensor, assuming use in the real world IoT environment.

CCS CONCEPTS

• **Networks** → **Naming and addressing**; **In-network processing**; **Routing protocols**.

KEYWORDS

NDN, function chaining, IoT, caching

ACM Reference Format:
Yohei Kumamoto and Hidenori Nakazato. 2020. Implementation of NDN Function Chaining Using Caching for IoT Environments. In *Cloud Continuum Services for Smart IoT Systems (CCIoT '20), November 16–19, 2020, Virtual Event, Japan.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3417310.3431401

CCIoT '20, November 16–19, 2020, Virtual Event, Japan

ACM ISBN 978-1-4503-8131-4/20/11…$15.00
https://doi.org/10.1145/3417310.3431401

1 INTRODUCTION

In recent years, the number of IoT devices have been increasing rapidly. Consequently, there has also been a growth in IoT applications and services, many of which require low latency such as factory automation and autonomous driving.

In response to this, edge computing has been proposed. Edge computing processes data by computing infrastructures placed closer to the source of the data, and provides results in a short time. It is more efficient in terms of time and resource usage compared to sending data out to a cloud [6]. However, the problem with the current edge computing is its heavy reliance on the edge computing resources adjacent to the sources of the data.

As a solution to this problem, we apply the idea of Service Function Chaining (SFC). With SFC, users can control traffic through software to route packets to the desired network services, which creates a virtual chain of network services [5]. We use this idea and place computing resources throughout the network, and run functions on them to process IoT data. By chaining these functions, data can be processed in a sequential manner to obtain the desired output. We can also strategically and dynamically place these functions to prevent and relieve network congestion and load. In order to achieve this, we adopted a communication protocol called Named Data Networking (NDN) [8]. NDN routes data by Content Name as opposed to the traditional IP, which routes according to location. This will allow a more intuitive and simple management of chained functions. In addition, NDN routers can cache content. By combining function chaining and the features of caching, content can be cached after application of functions. In this way, we can obtain the desired output with less function processing and with even shorter response time. This architecture that combines NDN and SFC provides us the flexibility to choose an appropriate function instance out of many available instances with the same functionality and name. By combining SFC and NDN we can make a more efficient IoT network.

In this paper we will discuss how to extend NDN to support function chaining and in-network processing. *Content* in this paper means the equivalent of Data.

2 RELATED WORKS

L. Liu et al. [2] proposed a solution called ICN Function Chaining (ICN-FC), which is an Information-Centric Networking (ICN) based framework for function chaining. It uses the ← symbol to connect functions within the Interest name. A simple example is shown in Fig. 1.

In this example, *A* and *B* represent functions, and *Data* represents the producer where the required data is located. In order to chain functions, the Interest name will be set to */A←/B←/Data*. Each

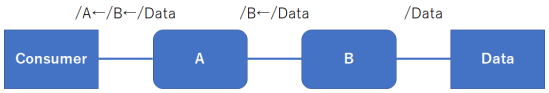

Figure 1: ICN-FC Example

time the Interest packet passes through a function, that function will be removed from the namespace. By removing the function from the namespace, it is possible to dynamically change the route of the Interest to a different function.

ICN-FC is evaluated assuming video processing. Although the segmentation / reassembly between the source of a video and the first function to process the video is discussed, how to transfer a large video file between functions is also missing.

H. Yoshii extends NDN to support function chaining. This NDN extension is called *NDN-FC* [1, 7]. NDN-FC takes content segmentation in consideration as a part of its mechanism. Typically, a content is split into segments to be transmitted. Splitting is necessary because the content may not fit into a packet if the size of the content exceeds the maximum size of a packet. NDN-FC reassembles the segments to reconstruct the content before applying a function to the content. To reassemble the requested content, the *consumer* which is the node requesting the content first obtains the number of segments of the content. The number is called *Final Block ID*. The consumer sends Interest packets up to the number expressed by the Final Block ID to the *producer* all at once. Producer is a node supplying content. Data packets requested by the number of Interest packets arrive at the function all at once. The function can reassemble segments and reconstruct the content in this way.

In NDN-FC, consumers, producers, and functions are considered as applications, and NDN Forwarding Daemon (NFD) is used as the router [4]. In addition, the cache is disabled in this implementation to make sure the execution of functions in the chain. An example communication in NDN-FC is shown in Fig. 2. Segmented content

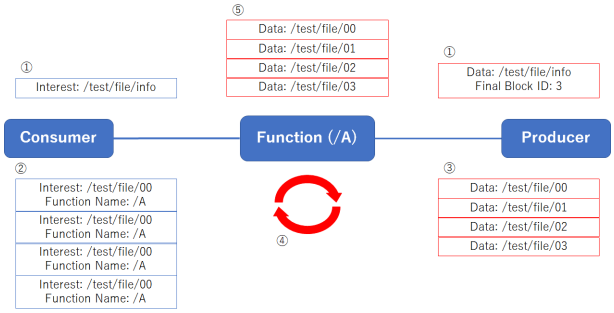

Figure 2: NDN-FC Example

in the figure is handled as follows.

(1) The consumer sends out an Interest packet with content name */test/file/info* to obtain *Final Block ID* of content with its name */test/file*. Final Block ID is the number of segments of the content.
(2) Using the Final Block ID, the consumer transmits all Interest packets required to retrieve the entire content.
(3) The producer holding the request content sends out Data packets with segmented content.
(4) When all Data packets arrive at the function */A*, the function reconstructs the content from the received data segments, and executes processing on the obtained content.
(5) The content after executing the function is segmented again and sent to the consumer. If the size of the content becomes large than the original size, the function requests the consumer for additional Interest packets.

In NDN-FC, Function Name field was added to the Interest packet format from the original packet format [3]. In this field, the chain of functions is listed. For example `/A/B/C` in the Function Name filed means, the consumer sending this Interest packet want to apply functions `/C`, `/B`, and `/A` in this order on the content specified in the Content Name filed. When the Interest packet passes through the function `/A`, the function name is removed from the Function Name field and sent out to the next router. That is, the Function Name field of the Interest packet becomes `/B/C`. The modified Interest packet format is shown in Fig. 3. The Data packet is kept the same as the original packet format.

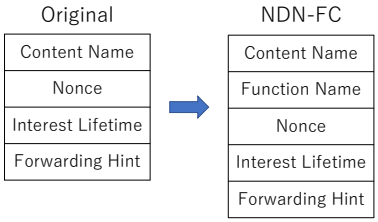

Figure 3: NDN-FC Interest Packet Format

When an Interest and Data packet goes through a function node, it enters the NFD, the router at the node, twice. The path of an Interest and Data packet is shown in Fig. 4.

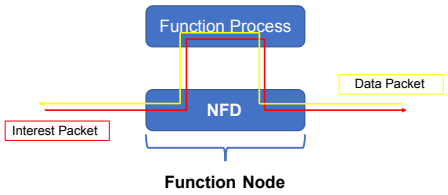

Figure 4: Interest and Data Packet Path in NDN-FC

When a function receives an Interest packet, it will forward the Interest packet back to NFD in order for the Interest packet to be forwarded to the next function or to the producer of the requested content. This step is required for creating the return path for the corresponding Data packet. Faces that come in the first and second times of the same Interest packet are distinguished by assigning a sequence number to each. 1 is given to the first face and 2 is given to the second arrival. When the NFD receives the Data packet, it sends the Data packet to the face with the largest sequence number.

This will ensure that the Data packet will go in the reverse path of the Interest packet in NDN-FC.

When the function receives a Data packet, it will check the Final Block ID in the Data packet to see how many Data packets to expect before further forward the Data packet. When all Data packets for a content are received, the function reassemble the content. After the reassembly is complete, the function apply its processing on the content. When the function execution is complete, the outcome is segmented again, and sent towards the consumer.

This implementation does not expect to make use of the cache. In the next chapter, we will show the problems caused by enabling the cache and propose a method to solve problems in enabling cache.

3 NDN-FC+

We are proposing a system that makes use of caching capability while supporting function chaining in NDN environment. We call the system *NDN-FC+*. NDN-FC+ is an extension of NDN-FC to make the system exploits caching capability of NDN.

Consumers can retrieve contents from a node closer to them by using cached Data packets in NDN. Since a Data packet after application of functions can also be stored in a cache in NDN-FC+, some function processing can be omitted in processing of function chains by making use of cached Data packets and low-delay function chain processing can be realized. A simple example is shown in Fig. 5.

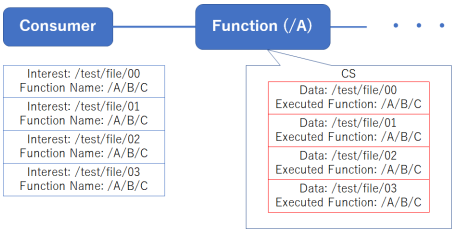

Figure 5: Example of Using Cache

Content Store (CS) is a router component that caches Data packets in NDN. In the example in Fig. 5, the CS in the router hosting the function named `/A` stores the Data packet with content `/text/file` after application of three functions: `/A`, `/B` and `/C` on the content. Here, function `/C` is applied first and function `/A` is applied last. In this state, if the consumer sends Interest packets requesting content `/text/file` and applying functions `/C`, and `/B` and `/A` on it in that order, i.e., the Function Name field set as `/A/B/C`,the Interest packets hit this cache and the Data packets are sent back immediately without forwarding farther. Since NDN-FC+ is an extension of NDN-FC, NDN-FC+ uses the same Interest packet format as NDN-FC.

If we simply enable cache in NDN-FC, a problem will be encountered. In NDN-FC, one Data packet enters the same router twice: once for receiving the packet from another router and once for receiving from a function as shown in Fig. 4. However, the content name is the same for the Data packet before and after the function execution. As a result, the Data packet after application of the function is not cached. NDN-FC+ solves this problem.

3.1 Packet Format Extension

Interest packet format has been extended from the original NDN format in NDN-FC. NDN-FC+ requires further extension in packet format, this time, the format for Data packet. It is necessary because Data packets need to show what functions have been applied to the content carried by the Data packet in order to cache the packet. The extension is addition of Function Name field to the Data packet format in NDN-FC which is the same as the original NDN packet format. The names of the executed functions will be listed in this field. The Data packet format in NDN-FC+ is shown in Fig. 6.

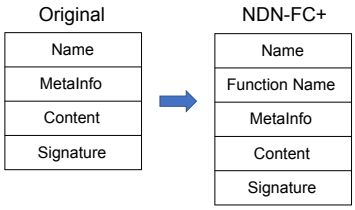

Figure 6: NDN-FC+ Data Packet Format

3.2 Packet Forwarding Procedure

In NDN, the Pending Interest Table (PIT) is a router component that is responsible for keeping track of pending Interest packets by recording the incoming faces of each Interest packet and sending Data packets back to the consumer. Also, the Forwarding Information Base (FIB) is a router component that is responsible for forwarding Interest packets to their corresponding data sources as a routing table.

In NDN-FC+, a PIT entry has a Function Name field in addition to a Content Name field and Incoming Face field of original NDN. In NDN-FC+, Interest packets before and after passing a function are not distinguished by the sequence number as in NDN-FC. Instead, they are distinguished by the two names: Content Name and Function Name in the PIT entry. Interest packets are processed as follows.

(1) When an Interest packet is received by a router, NFD, the router searches its PIT for the content name and the function name in the Interest packet. If a match is found, the incoming face of the Interest packet is added to the found entry, the Interest packet is discarded, and processing ends. If a matching entry is not found, a new entry is inserted with the content name, function name, and incoming face information.

(2) The router searches the CS for the Data packet with the same Content Name field and Function Name field as the received Interest packet. If a matched CS entry is found, the cached Data packet is returned, the Interest packet is discarded, and processing ends.

(3) If functions are listed in the Function Name field in the Interest packet, the router searches the first function name in the list as a name in the FIB. If a match is found, the Interest packet is forwarded to the face specified in the matched entry. If Function Name field of the Interest packet is empty, the

router searches its FIB for the name in the Content Name field.

(4) Every time the Interest packet passes through the function specified in the Function Name field, the function name corresponding to the function is deleted from the Function Name field.

(5) When the Interest packet is received by a producer, a corresponding Data packet is returned, and processing ends.

In NDN-FC+, Data packets, which trace the route that the Interest packets have taken in reverse order, are processed as follows.

(1) When a Data packet is received by a router, the router searches the PIT for the content name and the function name in the Data packet. If a matching entry is not found, the Data packet is discarded, and processing ends. If a matching entry is found, the Data packet is sent out to the face recorded in the entry and the entry is removed. At this time, the Data packet is cached in the CS.

(2) When a Data packet is received by a function process, the function wait until all Data packets necessary to reconstruct the content of the Data packets. The number of Data packets to wait can be know by the Final Block ID included in the first Data packet of the content. The function is applied on the reassembled content. After applying the function, the content is segmented again to Data packets and sent to the router hosting the function. The function name is prepended at the front of the Function Name field of each Data packet.

(3) When a Data packet is received by a consumer, the consumer waits all Data packets and reassembles the content of the Data packets.

3.3 Content transfer among consumer, producer, and functions

A content does not fit in a Data packet, typically. To retrieve a content, many Interest and Data packets are exchanged between a consumer and a producer in the original NDN. There the mechanism to inform Final Block ID is used to transfer a content involving multiple Data packets. Final Block ID is a value to specify the number of Data packets to be used to transfer a content. When the first Interest packet requesting a large content is received by a producer, the producer returns a Data packet containing the Final Block ID of the content as well as the first segment of the content. In the original NDN, this exchange of the Final Block ID involves only a consumer and a producer. Since application of a function requires entire content to be received before application of the function, the function need to participate in the exchange of the Final Block ID.

Functions perform the following procedure to collect necessary Data packets, apply the functions, and transfer back the result to the consumer.

(1) Receive the first Data packet of a content and find its Final Block ID.
(2) Transmit as many Interest packets as $< \textit{Final Block ID} > -1$.
(3) Receive all Data packets
(4) Reconstruct the content
(5) Apply the function on the content
(6) Segment the result
(7) Transmit the first Data packet of the result
(8) Wait for the following Interest packets and transmit the corresponding Data packets

Since Interest packets are propagated through functions requested by the Interest packets, the first Data packet sent out from the producer is received by the first function from the producer in the chain. The function can find the Final Block ID of the content in the first Data packet. The function autonomously creates Interest packets requesting the rest of the content using the Final Block ID. By doing so, the function can receive all Data packets necessary to reassemble the content. Then, the function applies its function on the content, segments the result, and prepare Data packets of the result. Since the first Interest is already received by the function, the first Data packet is returned and wait for the additional Interest packets. When they are received, the rest of the Data packets are returned.

Fig. 7 shows an example of communication by this method. The

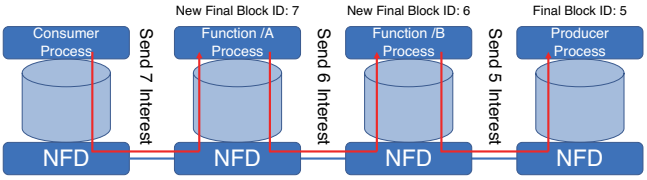

Figure 7: Interest Packet Flow Example

function /B receives the Final Block ID 5 in the first Data packet from the producer. The function transmits the remaining four Interest packets to the producer and receive all Data packets necessary to reconstruct the content. After applying the function, the content size increases and now the Final Block ID for the content is 6. Function/B returns the first Data packet with its Final Block ID 6 to the next function /A. Function /A creates and transmits additional five Interest packets to function /B. As is shown in this example, necessary Interest packets are sent between adjacent functions as needed. As a result, all Interest packets are not sent from the consumer to the producer all at once unlike.

4 EVALUATION

4.1 Scenario

To confirm the correctness of the proposed method, the configuration shown in Fig. 8 is used. We have created 5 virtual machines

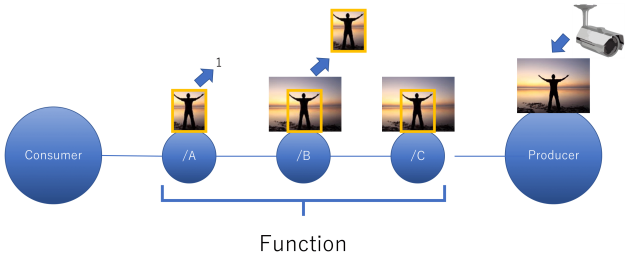

Figure 8: Implementation Testing Scenario

in VirtualBox where each machine resembles a consumer, function

/A, function /B, function /C and producer. NFD will be run on each machine. If we enable caching of the NFD on each machine, the NFD will cache all Data packets that pass through it in the CS and remove the elements (when the CS is full) according to the LRU policy. The producer supplies the image data from the camera with a content name "*/test/producer/test.jpg*." The consumer transmits Interest packets with Function Name field "/A/B/C" to the producer. Function /C detects all humans from the image data received from the producer, and outputs its position information in the image data and the encoded information of the image in text format. We have created Function /C based on YOLO, one of the real-time object detection software. Function /B decodes the image data in text format received from function /C, and cuts out only the part of the image data showing the person based on the position information of the person. Furthermore, function /B encode the cut image data into again text format, and pass to function /A. Function /A calculates the number of cut images (humans) based on the received information in text format, and sends that number of humans and image encoded into text format to the consumer.

In this scenario of Fig. 8, we tested that a Data packet sent back by a cache hit is properly processed by function chaining and delivered to the consumer. In addition, we show the performance improvement from NDN-FC by the proposed method by measuring the communication time when the caches of the nodes hosting the functions /A, /B, and /C are used.

4.2 Behavior of each component

We enabled the cache only in the router hosting function /B. Then, in order to fill the cache, we let the consumer obtains the content with function execution in the order of /C → /B → /A in advance. After this preparation, we tested the operation of cache by sending the Interest packet with the content name of the camera image and Function Name field /A/B/C from the consumer. The result is shown in Figs. 9 ~ 11. It should be noted that due to the length of the output, only the important parts are shown.

Fig. 9 is a router log of the router hosting function /B. In the log, we can see the CS is being searched for the content name and the function name "/B/C." In this implementation, the search string is "/test/producer/content/test.png /*<segment number>*/B/C" which is the content name with the function name attached at the end of the content name. Then, when an Interest packet comes in, it simply searches the concatenated name in the CS.

Fig. 10 is the output of function /A in the scenario. From this output, it can be confirmed that the Data packet of segment number 00 is received by function /A, the function obtains the Final Block ID from the Data packet, and the additional Interest packets are transmitted (Fig. 10 ①). After the transmission of Interest packets, the function receives all the Data packets and executes the function using the contents obtained by reassembling the Data packets (Fig. 10 ②). After finishing the execution of the function, this function again segments the content and transmits the content to the consumer. The content size which is 13.842kB before applying function /A becomes 14.514kB after applying function /A.

Fig. 11 shows the output of the consumer. From this output, we can see the consumer is sending the first Interest packet for a content with the segment number 00 first (Fig. 11 ①). Return to the first Interest packet, the Data packet of segment number 00 is received from function /A (Fig. 11 ②). Then, the final block ID is obtained from the Data packet, the rest of the Interest packets are transmitted (Fig. 11 ③), and that all the Data packets are received (Fig. 11 ④). Finally, the consumer reassembles them to obtain the content where the functions /C, /B, and /A are applied.

From these results, it can be seen that NDN-FC+ is implemented properly.

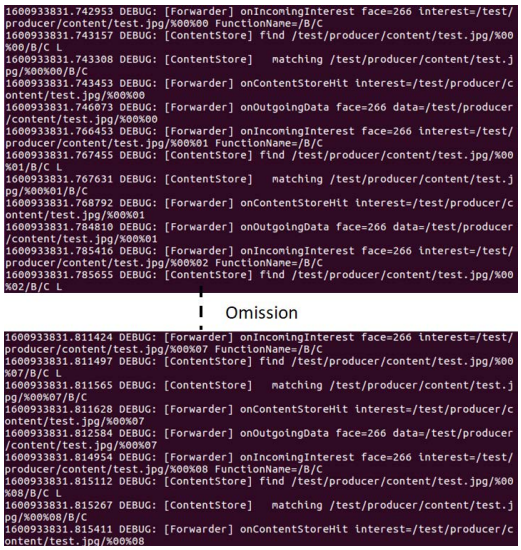

Figure 9: Output of Router Hosting Function /B

Figure 10: Function /*A* Output

Figure 11: Consumer Output

4.3 Content acquisition time comparison

We measured the time required for a consumer to retrieve contents, or *content acquisition time,* with and without using the cache. The time the consumer sends the first Interest packet with segment number 00 is set to time 0. Time is counted until the consumer receives the last Data packet and the content is reassembles.The following cases are compared.

(1) All the caches are disabled.
(2) Only the cache of the router hosting function `/C` is enabled.
(3) Only the cache of the router hosting function `/B` is enabled.
(4) Only the cache of the router hosting function `/A` is enabled.

In all cases, the content to be transferred is unified with the image data of 18.140 kB, which is the image of only one person, before the execution of the function. Also, before starting the measurement, one request on the content with execution of functions `/C`, `/B`, and `/C` is performed to populate the caches. The one request involves many Interest and Data packets because the content is large. In each of the above four cases, 100 measurement was performed and the average was taken. The result is shown in Table 1.

According to Table 1, content acquisition time is shorter when using cache. In particular, the time is greatly shortened between the case where the cache is not used and the case where the cache of the router for function `/C` is enabled. This is because the execution time of function `/C` is longer than other functions, and the execution time of function `/C` can be saved by the cache. Likewise, when it takes time to execute a function, by caching the Data packet after application of the function, the time is greatly improved. Of course, the content acquisition time for the case where Data packets after application of all the functions are cached is the shortest, but even when the intermediate results are cached the content acquisition time can be greatly shortened.

Another reason the content acquisition time is shortened is that the content can be obtained from the node closer to the consumer and some network delay can be eliminated. As a consequence, network traffic can be reduced.

Table 1: Content Acquisition Time

case	time(ms)
without cache	112242.74
/C cache	970.09
/B cache	241.87
/A cache	81.14

5 CONCLUSION AND FUTURE WORK

In this paper, we have discussed the necessity of combining function chaining with caching in IoT networks. We have also proposed an architecture that support function chaining and caching in NDN. The performance evaluation in this paper showed us the effectiveness of our proposed mechanism NDN-FC+.

Our future work includes testing with more complex scenarios. Our results show that caching properly works with a basic network structure, but further work examining scalability may be required.

In addition, as a future extension of this work, we are going to consider an architecture that caches the latest content from IoT devices on nodes near consumers. In this architecture, nodes that send periodic interest packets to IoT devices are deployed close to consumers. As a result, it is considered that the latest content from an IoT device such as a sensor whose content is updated as time elapses is always cached in a node near consumers and consumers can obtain that cached content with low latency. This architecture will allow us to make more efficient IoT networks.

ACKNOWLEDGMENTS

The research leading to these results has been supported by the EU-JAPAN initiative by the EC Horizon 2020 Work Programme (2018-2020) Grant Agreement No. 814918 and Ministry of Internal Affairs and Communications "Strategic Information and Communications R& D Promotion Programme (SCOPE)" Grant no. JPJ000595, "Federating IoT and cloud infrastructures to provide scalable and interoperable Smart Cities applications, by introducing novel IoT virtualization technologies (Fed4IoT)."

REFERENCES

[1] Yohei Kumamoto, Hiroki Yoshii, and Hidenori Nakazato. 2020. Real-World Implementation of Function Chaining in Named Data Networking for IoT Environment. In *2020 IEEE ComSoc International Communications Quality and Reliability Workshop (CQR)*. 1 – 6.
[2] L. Liu, Y. Peng, M. Bahrami, L. Xie, A. Ito, S. Mnatsakanyan, G. Qu, Z. Ye, and H. Guo. 2017. ICN-FC: An Information-Centric Networking based framework for efficient functional chaining. In *2017 IEEE International Conference on Communications (ICC)*. 1–7.
[3] NAMED DATA NETWORKING. [n.d.]. *Interest Packet.* Retrieved September 11, 2020 from https://named-data.net/doc/NDN-packet-spec/current/interest.html
[4] NAMED DATA NETWORKING. [n.d.]. *NFD Overview.* Retrieved September 11, 2020 from https://named-data.net/doc/NFD/current/overview.html
[5] Paul Quinn and Jim Guichard. 2014. Service Function Chaining: Creating a Service Plane via Network Service Headers. *Computer* 47, no. 11 (November 2014), 38–44. https://doi.org/10.1109/MC.2014.328
[6] Weisong Shi and Schahram Dustdar. 2016. The Promise of Edge Computing. *Computer* 49, no. 5 (May 2016), 78–81. https://doi.org/10.1109/MC.2016.145
[7] Hiroki Yoshii. 2019. *Real World Implementation of Function Chaining in Named Data Networking.* Master's thesis. School of Fundamental Science and Engineering, Waseda University.
[8] Lixia Zhang, Alexander Afanasyev, Jeffrey Burke, Van Jacobson, kc claffy, Patrick Crowley, Christos Papadopoulos, Lan Wang, and Beichuan Zhang. 2014. Named Data Networking. *SIGCOMM Comput. Commun. Rev.* 44, no. 3 (July 2014), 66–73.