Future Network and MobileSummit 2013 Conference Proceedings
Paul Cunningham and Miriam Cunningham (Eds)
IIMC International Information Management Corporation, 2013
ISBN: 978-1-905824-36-6

OpenFlow-based Content-Centric Networking Architecture and Router Implementation

Atsushi Ooka[1], Shingo Ata[2], Toshio Koide[3], HIDEyuki Shimonishi[3], and Masayuki Murata[1]
[1]*Graduate School of Information Science and Technology, Osaka University*
1-5 Yamadaoka, Suita, Osaka, 565-0871 Japan
Tel.: +81-6-6879-4542, Fax: +81-6-6879-4544
Email: {a-ooka, murata}@ist.osaka-u.ac.jp
[2]*Graduate School of Engineering, Osaka City University*
3-3-138 Sugimoto, Sumiyoshi-ku, Osaka-shi, Osaka 558-8585, Japan
Tel.: +81-6-6605-2191, Fax: +81-6-6690-5382, Email: ata@info.eng.osaka-cu.ac.jp
[3]*Cloud System Research Laboratories, NEC Corporation*
1753 Shimonumabe, Nakahara, Kawasaki, Kanagawa 211-8666, Japan
Tel.: +81-44-431-7702, Fax: +81-44-431-7644
Email: {t-koide@bk, h-shimonishi@cd}.jp.nec.com

Abstract:
Content-Centric Networking (CCN) is a novel architecture that has been proposed as a solution for dealing with various problems facing the Internet, such as the excessive bandwidth costs that result from peer-to-peer (P2P) traffic and content delivery networks (CDN). In order to deal with the difference between the principles of the Internet and the purposes for which it is actually used, CCN entails a change from a host-centric to content-centric architecture. However, there are deployment issues that will require a gradual approach in order to realize migration from the Internet to CCN. Additionally, the shortage of testbed environments needs to be addressed as quickly as possible. We focus on OpenFlow, which is a promising candidate to provide a programmable environment without disrupting existing networks. Specific implementations of CCN using OpenFlow have not been examined in sufficient detail, although there have been investigations into the conceptual design. This paper presents the detailed design and implementation of an OpenFlow-based CCN with the primary aim to achieve forwarding and end-to-end communication. Our approach can solve the issues in existing designs by using a map from content names to hierarchically structured hash values and the longest prefix match. We also discuss the advantage of retaining significant attributes of CCN and OpenFlow such as high-speed forwarding and network slicing which promotes deployment of CCN and research for routing, caching, and security strategies.

Keywords:
Future Networks, Content-Centric Networking, Architecture, OpenFlow

1. Introduction

The Internet, which is now a global network of networks, is used in a form and scale considerably different from the original design principles and assumptions, and this gives rise to many problems. Information-Centric Networking (ICN) and Content-Centric Networking (CCN) [1], which provide a location-independent network architecture in which the content is named, has been proposed as a measure for coping with these problems facing the Internet; the goal is for CCN to become the Internet of the future.

A number of research projects have studied ICN/CCN approaches such as Named Data Networking (NDN) [2], PURSUIT [3], and SAIL [4]. These all share the common concept of addressing the content that is exchanged in communication by "name", which is mnemonic and unique to each chunk of data. In this paper, we focus on CCN/NDN, which is characterized by a hierarchical structure and variable-length names.

The features of CCN create key challenges for implementation. There are feasibility problems for achieving routing, caching, and security and deployment problems about a gradual approach to ICN/CCN deployment, costs, and business models, and they have been analyzed and evaluated [5, 6, 7]. It is necessary to demonstrate that CCN could become the successor to the Internet and to encourage general network users to migrate to the CCN.

Taking this situation into account, OpenFlow is expected to aid with the development of CCN. OpenFlow implements the software-defined networking (SDN) concept [8]. This allows researchers to test novel networking protocols like CCN in an actual network environment [9]. The possibility of deployment supported and promoted by OpenFlow has also been investigated [10, 11].

While the benefits of designing a network using OpenFlow based on content-centric principles have been actively discussed, there has been a lack of discussion on the major characteristics of CCN/NDN and a lack of concrete implementations. Most existing research differs from the CCN/NDN concept in the details of the implementation, with some features such as aggregation scaling routing states and availability of uniquely named content sacrificed. There has not been either much discussion on how to implement the flow tables and OpenFlow controller.

In order to deal with these issues, we present the detailed design and implementation of an OpenFlow network in which packet forwarding is realized based on hierarchically structured hash values of names. We employ Trema as the OpenFlow framework for implementation and testing.

2. Background

2.1 Content-Centric Networking

CCN is a novel network architecture that was designed with a focus not on the location of data but on the content of data. This approach has the following advantages:

- Content-centric rather than host-centric communication
- *Names* that provide each chunk of data with unique, human-readable, and hierarchical structured addresses
- Mechanisms for native multicasting, in-network caching, and security that is built into the data

The content-centric design was inspired by recent developments in the utilization of the Internet. A main use of current networks is the distribution and retrieval of vast amounts of data such as HTML documents, images, and high-definition video. However, specifying the locations of providers and consumers is not necessary for this purpose. CCN is a solution for dealing with the incompatibilities and security concerns by shifting the routing behavior based on from "where" to "what". In CCN, it is not necessary to know where the device we want to communicate with is located, and instead we directly identify the name of data that we want. Names enable us to do this by providing each chunk of data with a unique, human-readable, and hierarchical address. In addition, longest prefix match (LPM) look-up on names is implemented, which enables to aggregate routing state just like Internet.

Communication via CCN is realized through the exchange of ***Interest*** packets and ***Data*** packets. Interest packets are sent by data consumers and contain the request for and the name of the content. Data packets are sent back by the data producer in response to Interest packets received by the producer and contain the actual data.

CCN supports inherent security and privacy protection. Every packet is verified using encryption and signatures. There is demand for security that is embedded in the content data itself, and research is ongoing into how to implement these inclusive structures. Because of this and our intention to implement forwarding independently from other functions, we do not take account of the security.

2.2 OpenFlow

OpenFlow was implemented as the first standard interface for an SDN architecture. SDN provides network operators with programmable management which is decoupled from the underlying network infrastructure for current L2/L3 switches. This programmability removes the barriers to experimenting with new network protocols in a real network infrastructure. The separation of data paths from network intelligence results in two basic components called the switch and controller, respectively. The controller gives the switches the operational instructions, which are written to a flow table contained in the switch as individual flow entries. When the switch receives a packet, it searches the flow table for corresponding entries by using the matching rules, and then processes the packets according to the actions in the entries.

Note that the first version and the latest version of the OpenFlow switch specifications differ in terms of the matching rules and actions. The OpenFlow 1.0 specifications do not support matching and modification to IPv6 addresses. However, these are supported in the OpenFlow 1.3 specifications.

3. Architectural Description

3.1 Design Principles

In this section, we describe an architecture of OpenFlow-based CCN. This guarantees a high-speed forwarding by using hierarchically structured hashes of names. In order to decide the flow for sending back a Data packet immediately after receiving a corresponding Interest packet, it is necessary to give these names hierarchically structured hash values for matching them by using LPM. Because a name of an Interest packet may be different from one of a Data packet satisfying it, this quick decision is impossible in existing solutions which give them different hash values that are not hierarchically structured.

We explain a detailed design including configuration of the flow entries, which is missing in the existing researches. In accordance with the design principles of CCN, a forwarding strategy is separate from the routing strategy. This promotes the deployment of CCN concurrently with research on routing systems. Owing to our concern about the need of early-deployment, the design is based on UDP and OpenFlow 1.0 specifications, both of which are currently available. In addition, our solution can be easily expanded by new technologies such as OpenFlow 1.3 specifications.

3.2 Specification of Packets

UDP packets are defined specifically for CCN and are treated as Interest packets and Data packets. End-users are able to send and receive these packets natively.

In order to simplify application developments, we propose that a predefined set of fields in the packet headers are examined, and have adopted the following rules:

UDP Port Number Used for distinction between Interest packets and Data packets. It is necessary to assign port numbers that is uniquely used in the network. We denote their port numbers as 50001 and 50002 respectively for purposes of illustration in this paper.

IPv4 Address Associated with the name of the content through an appropriate hash function (detailed in Subsection 3.3).

MAC Address Determines whether the I/O port where the packet has arrived is inbound or outbound. Specially defined MAC address must be assigned to the packets. 11:11:11:11:11:11 indicates unprocessed packets, and 22:22:22:22:22:22 indicates processed packets by OpenFlow switches for purposes of illustration in this paper.

The CCN packets (i.e., the UDP payloads) that we use are simplified as much as possible and do not support security features such as signatures since our aim is only to demonstrate the feasibility of packet exchange using CCN. The packets therefore contain the minimal information for communicating on CCN, as shown in Figure 1.

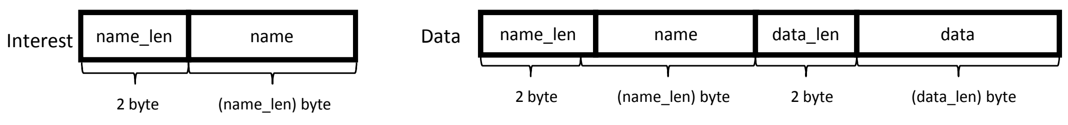


Figure 1: UDP payloads representing Interest packets (left) and Data packets (right)

Since each of the length fields are 2 bytes long, the sizes of the name and data are limited to less than 64 Kbytes. This maximum length is reasonable because it is obvious that the length of the names is less than 65,536, and Data packets where the entire length is more than 1,500 bytes are first split up in order to avoid fragmentation in the lower layers. These UDP packets customized for CCN allow end-users to retrieve content without needing equipment based on OpenFlow.

3.3 Matching and Forwarding

In content-centric solutions, only the names of content are examined for forwarding and routing decisions except in the case of security strategies. CCN routers require the ability to perform LPM on names. However, OpenFlow does not offer any matching rules for inspection of the UDP payload. Instead, the IP address field can be subnet masked in the OpenFlow 1.0 specifications, and we therefore propose a solution that writes hash values that are converted from the name into the IP address field.

Each component of the name is assigned 4 bits, allowing names consisting of up to 8 components to be handled. Each component is converted to a value from 1 to 15, with the value 0 indicating that the corresponding component does not exist. For example, consider the case of content named "`/pict/a.jpg/v1/s2`." The name contains four components: "`pict`", "`a.jpg`", "`v1`", and "`s2`". If these components are converted by a hash function to 1, 3, 4, and 8, respectively, the IP address becomes "19.72.0.0/16" in CIDR notation or "0x13480000" in hexadecimal. When a Data packet with the name "`/pict/a.jpg/v1/s2`" is cached, an Interest packet named "`/pict/a.jpg`", which converts to "19.0.0.0/8", would match the Data packet by using LPM on the IP address. OpenFlow switches performs forwarding based on this virtually implemented LPM by name, so the LPM rule for the IP address is common to all flow entries.

Actions targeting Interest and Data packets must include forwarding of the packet to the appropriate destination I/O port and controller. The destination I/O ports of Interest packets are looked up in the routing table and of Data packets are determined when Interest packets that have the same name pass through the switch. In OpenFlow networks, which is usually composed of multiple switches, the packets are simply transfered from the source host to the destination host depending on the flows.

On sending a packet out to an external host, it is necessary to make the following two modifications to the packet before forwarding: (1) adjust the IP and MAC destination addresses to those of the destination host; and (2) change the IP and MAC source addresses to those of a virtual gateway. Note that (2)'s modifications make it possible to prevent end-hosts from becoming confused in the ARP table by providing end-hosts with non-duplicate IP addresses and the uniquely defined MAC address 11:11:11:11:11:11 . On receiving a packet from an external host, which are assigned MAC destination address of 11:11:11:11:11:11 , this MAC destination address is modified to 22:22:22:22:22:22 for multihop transmission in OpenFlow networks.

In addition, Network slicing can be easily implemented. We can isolate the CCN traffic from the IP traffic and handle them by examining the field of UDP port number, i.e, UDP packets with port number 50001 or 50002 are processed as the CCN traffic, and others are processed as the IP traffic.

3.4 Caching

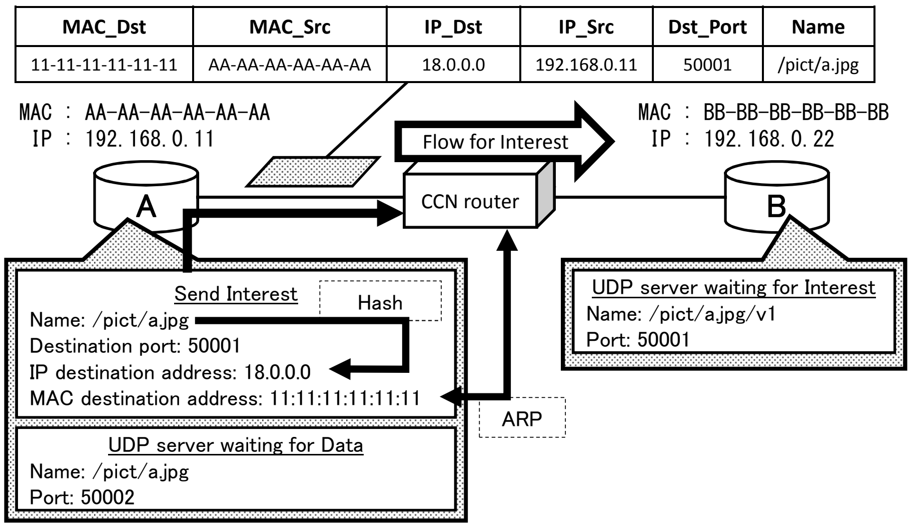
CCN routers possess a native cache for Data packets called the CS. However, OpenFlow switches do not have this kind of caching mechanism. We therefore introduce a software cache in the OpenFlow controller. In particular, whenever the OpenFlow controller receives Data packets, the controller stores the packets for later requests.

Although the controller is useful for caching, the software-based implementation lacks the speed of the CS which is implemented in hardware. A dedicated storage device for caching would certainly have the potential for achieving performance comparable with that of the CS. However, we have not adopted this device as it lacks the flexibility required for testing during development.

3.5 Description of Communication

Figures 2, 3, and 4 give a detailed description of the communication in our architecture. There are a single CCN router comprising an OpenFlow switch and controller, and two hosts that are exchanging a piece of content named "`/pict/a.jpg/v1`". The host denoted "A" is a consumer who sends Interest packets and receives Data packets, and the host denoted "B" is a content producer who does the opposite.

Figure 2 illustrates the beginning of a scenario in which the requester sends an Interest packet named "`/pict/a.jpg`". The requester computes the hash value of



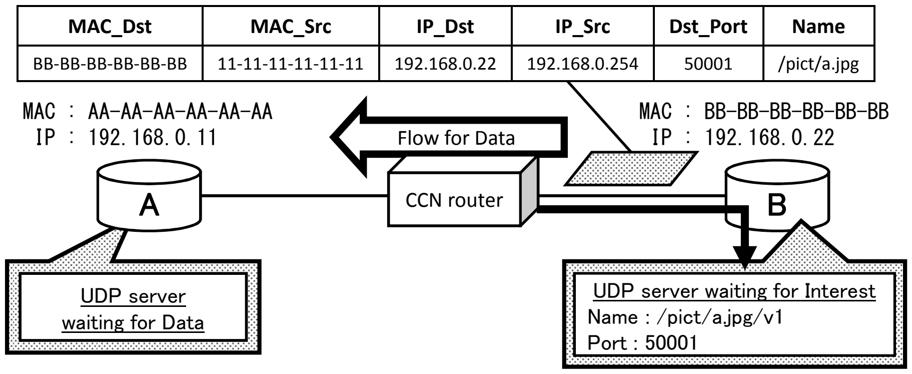
MAC_Dst	MAC_Src	IP_Dst	IP_Src	Dst_Port	Name
11-11-11-11-11-11	AA-AA-AA-AA-AA-AA	18.0.0.0	192.168.0.11	50001	/pict/a.jpg

Send Interest
Name: /pict/a.jpg
Destination port: 50001
IP destination address: 18.0.0.0
MAC destination address: 11:11:11:11:11:11

UDP server waiting for Data
Name: /pict/a.jpg
Port: 50002

UDP server waiting for Interest
Name: /pict/a.jpg/v1
Port: 50001

Figure 2: Detailed description of communication: (1) Sending an Interest packet

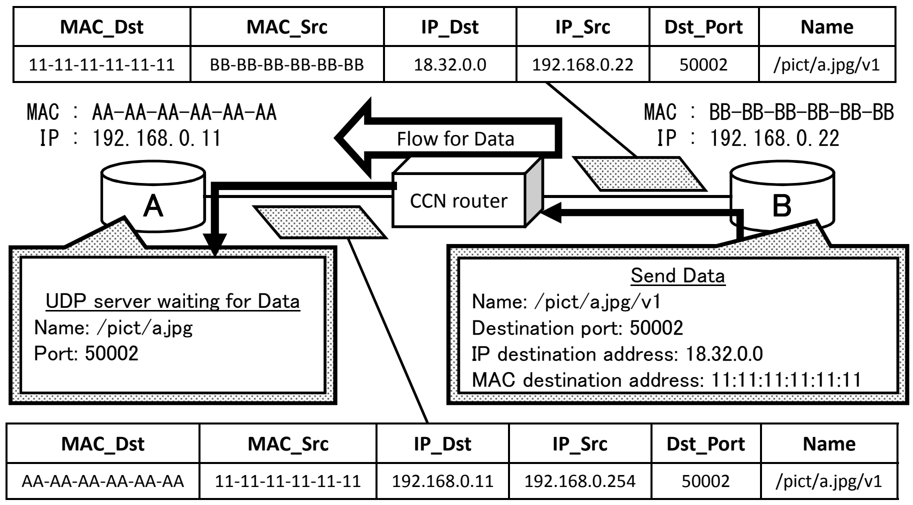


MAC_Dst	MAC_Src	IP_Dst	IP_Src	Dst_Port	Name
BB-BB-BB-BB-BB-BB	11-11-11-11-11-11	192.168.0.22	192.168.0.254	50001	/pict/a.jpg

UDP server waiting for Data

UDP server waiting for Interest
Name : /pict/a.jpg/v1
Port : 50001

Figure 3: Detailed description of communication: (2) Receiving an Interest packet



MAC_Dst	MAC_Src	IP_Dst	IP_Src	Dst_Port	Name
11-11-11-11-11-11	BB-BB-BB-BB-BB-BB	18.32.0.0	192.168.0.22	50002	/pict/a.jpg/v1

UDP server waiting for Data
Name: /pict/a.jpg
Port: 50002

Send Data
Name: /pict/a.jpg/v1
Destination port: 50002
IP destination address: 18.32.0.0
MAC destination address: 11:11:11:11:11:11

MAC_Dst	MAC_Src	IP_Dst	IP_Src	Dst_Port	Name
AA-AA-AA-AA-AA-AA	11-11-11-11-11-11	192.168.0.11	192.168.0.254	50002	/pict/a.jpg/v1

Figure 4: Detailed description of communication: (3) Sending and receiving a Data packet

the name to determine the IP destination address before sending the packet. The MAC destination address is resolved by the CCN router. The flow for forwarding an Interest packet named "`/pict/a.jpg`" is preliminarily written in a flow table in the switch. In addition to sending the packet, the requester needs to prepare the UDP server for receiving the Data. Figure 3 shows the forwarded Interest packet arriving at the producer. Both the IP address and MAC address of the packet are modified properly. What needs to be emphasized is that the flow is also rewritten for Data packets. Finally, Figure 4 shows the flow of the Data packet. In response to the Interest packet, the producer sends a Data packet named "`/pict/a.jpg/v1`". The Data packet is modified and forwarded by the CCN router similarly to the Interest packet. The consumer subsequently receives the packet and the request is satisfied. What is important is that all packets contain names with different hashes, but these do not disrupt the processing owing to the LPM and hierarchically structured hashes.

4. Implementations of Each Component

4.1 Configuration of Flow Entries in the OpenFlow Switch

The flow entries in the OpenFlow switch are managed according to the states computed by the OpenFlow controller. The state assigned to each name can be defined as the data structure (i.e., FIB, PIT, or CS) where the packet with that name should be processed. An index table is used for looking up the state of the name.

Suppose an Interest packet that requests content named "`/pict/a.jpg`" reaches the switch for the first time. In this case, the FIB should be looked up to process this packet since the CS has not yet cached the Data packet for this name and the PIT entries have not been registered yet either. The state for "`/pict/a.jpg`" therefore needs to be pre-defined as FIB, with the controller having already registered the corresponding flow entries in the switch. The flow entries then enable the switch to forward the packet appropriately, and the new state becomes PIT since the unsatisfied Interest packet has been added to the PIT.

The state transition rules and the operations performed on received packets are listed in Table 1. Operations to add and remove entries are intended to not only deal with the flow entries in the switch but also to change the state of the controller.

Table 1: State transition rules and operations on flow entries

State	On Receiving Interest	New State	On Receiving Data	New State
FIB	Add the PIT entry	PIT	Cache Data and add the FIB entry	CS
PIT	Add PIT entry	PIT	Remove PIT entry, and add FIB entry	CS
CS	Send back cached Data	CS	Do nothing	CS

4.2 OpenFlow Controller

In our solution, the OpenFlow controller is programmed to behave as a CCN router in combination with the OpenFlow switch. It also needs to ensure consistency of communication with the end-users. To that end, the controller needs to manage the state of each content name, configure flow entries (as detailed in Subsection 4.1), handle ARP packets properly, and associate the I/O port to which the host is directly connected.

As noted in Subsection 3.3, the virtual CCN router implemented by the OpenFlow controller and switch handles ARP table confusion. Without ARP, it is impossible for hosts to communicate over Ethernet, and UDP is no exception. However, it is quite likely that the large number of dynamic IP addresses derived from the names would trigger an explosion of entries in the ARP table. As a consequence, the CCN router must provide fixed IP addresses and MAC addresses and act as a virtual gateway router.

5. Discussion

5.1 *Comparison with Existing Solutions*

Existing solutions implement a dynamic mapping between content names and fixed-length hash values [10, 9]. However, those solutions lack some of the important characteristics of CCN. A hierarchically structured name is required for performing the LPM in CCN in order to allow for different names between the Interest and Data packets.

Consider two packets, an Interest packet with the name "`/pict/a.jpg`" and a Data packet with the name "`/pict/a.jpg/v1/s2`". Obviously, the two names give different hash values, which are not hierarchically structured and have nothing in common in the existing solutions. This makes it impossible to use the LPM between two names from the hash values. It is therefore unreasonable for a CCN router that have received an Interest packet both to register all flow entries for forwarding conceivable Data packets, and to determine the single name of the corresponding Data packet. Because of this, the router cannot create the flow entries before receiving the Data packet.

In our proposed solution, by contrast, the content names are mapped to IPv4 addresses while maintaining the hierarchical structure, which makes it possible to look up content names using the LPM on IPv4 addresses. Since a CCN router can create the flow entries immediately on receiving an Interest packet, a Data packet is sent back quickly along the path of switches. It should also be noted that this also facilitates new routing system research which employs aggregation based on hierarchically structured names.

5.2 *Contribution to Deployment Issues*

The network implemented according to our specifications meets the minimal requirements for CCN with regard to realizing end-to-end communication. Forwarding based on a hierarchically structured name is performed by using the IP address field to store hierarchically hashed names. Routing strategies are easy to implement and expand, including strategies that use the LPM on the content name, owing to their independence from the forwarding strategies.

In addition to this feasibility, CCN implemented using OpenFlow can be deployed in a way that allows for gradual migration from the Internet. While the deployment of OpenFlow networks represents a cost to network operators, isolating experimental networks from the existing network by network slicing potentially reduces the overall burden including the migration.

The transition from OpenFlow-based CCN to pure CCN remains a topic for further research. However, we are not yet concerned with this issue because the specifications of CCN are still being explored.

5.3 *Consideration of OpenFlow 1.3 Specifications*

The OpenFlow 1.3 specifications have the potential to improve the probabilities of hash collisions by supporting matching on IPv6 addresses. This makes it possible to use arbitrary bitmasks for matching IPv6 address fields and MAC addresses. We focus on applying this to the probabilities of hash collisions, i.e., to use the LPM on IPv6 and MAC addresses to reduce the rate of collisions.

Assuming each component of a name is assigned B bits, the probabilities P_B^{protocol} for the three protocols (IPv4, MAC, and IPv6) are given by the following equation:

$$P_B^{\text{protocol}} = \left(1 - \prod_{i=1}^{N} \frac{(2^B - 1) - i}{2^B - 1}\right)^{\frac{len}{B}} \tag{1}$$

where N is the number of names and len is the bit-length of the addresses in the protocol. Figures 5, 6 show the probabilities P_B^{IPv4}, P_B^{MAC} and P_B^{IPv6} for $B = 4$ and 16 respectively.

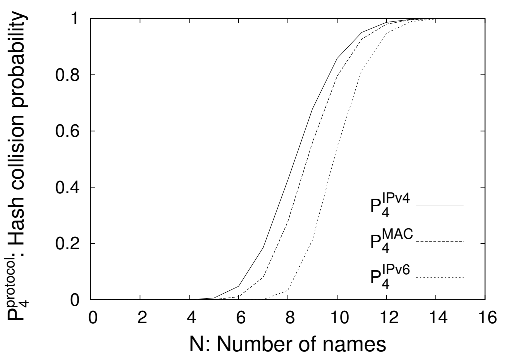


Figure 5: Probability of a hash collision ($B = 4$)

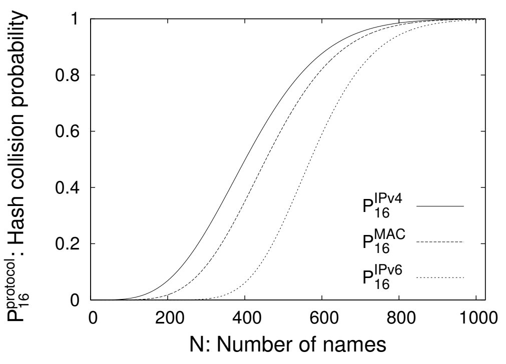


Figure 6: Probability of a hash collision ($B = 16$)

The graphs show that the parameter B is much more influential on the probability of a hash collision than the difference between protocols. This means that it is better to adopt as large a value of B as possible. However, 99.9% of URLs consist of less than 30 components according to [5]. It is therefore preferable to adopt matching on IPv6 addresses with $B = 4$. The trade-off between the hash collision probability and the number of components is left for investigation in future work.

6. Conclusions

With the growing recognition that the Internet is reaching its limits, CCN has been drawing attention around the world. This promising architecture offers tremendous advances including native support for multicasting, in-network caching, and security that depends on the content. OpenFlow, which is also gaining an increasing presence, is expected to become the intermediary for migration from the Internet to CCN.

This paper presented the architecture and implementation of an OpenFlow-based CCN. Our examination demonstrated the feasibility and possibility for deployment, and the ability to simultaneously study challenges such as routing and caching strategies by using hierarchically structured hashes of content names. The trade-off between hash collision probabilities and the number of components that can be handled in names will be investigated once products and a framework are released.

7. Acknowledgment

This work was supported by the Strategic Information and Communications R&D Promotion Programme (SCOPE) of the Ministry of Internal Affairs and Communications, Japan.

References

[1] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *5th international conference on Emerging networking experiments and technologies*, pp. 1–12, December 2009.

[2] L. Zhang, D. Estrin, J. Burke, V. Jacobson, J. Thornton, D. Smetters, B. Zhang, G. Tsudik, D. Massey, C. Papadopoulos, *et al.*, "Named data networking (NDN) project," Tech. Rep. NDN-0001, PARC, October 2010.

[3] N. Fotiou, P. Nikander, D. Trossen, and G. C. Polyzos, "Developing information networking further: From PSIRP to PURSUIT," in *7th International ICST Conference on Broadband Communications,Networks, and Systems*, pp. 1–13, October 2010.

[4] T. Levä, J. Gonçalves, R. J. Ferreira, *et al.*, "Description of project wide scenarios and use cases," Tech. Rep. D2.1, SAIL, February 2011.

[5] D. Perino and M. Varvello, "A reality check for Content Centric Networking," in *the ACM SIGCOMM workshop on Information-centric networking*, pp. 44–49, August 2011.

[6] S. Arianfar, P. Nikander, and J. Ott, "On content-centric router design and implications," in *the Re-Architecting the Internet Workshop*, pp. 1–6, November 2010.

[7] V. Jacobson, D. K. Smetters, N. H. Briggs, M. F. Plass, P. Stewart, J. D. Thornton, and R. L. Braynard, "VoCCN: Voice-over Content-Centric Networks," in *the 2009 workshop on Re-architecting the internet*, pp. 1–6, December 2009.

[8] ONF Market Education Committee, "Software-Defined Networking: The new norm for networks," Tech. Rep. ONF White Paper, ONF, April 2012.

[9] N. Melazzi, A. Detti, G. Mazza, G. Morabito, S. Salsano, and L. Veltri, "An OpenFlow-based testbed for information centric networking," in *Future Network Mobile Summit 2012*, pp. 1 –9, July 2012.

[10] D. Chang, J. Suh, H. Jung, T. T. Kwon, and Y. Choi, "How to realize CDN Interconnection (CDNI) over OpenFlow?," in *Proceedings of the 7th International Conference on Future Internet Technologies*, pp. 29–30, September 2012.

[11] I. Carvalho, F. Faria, E. Cerqueira, and A. Abelem, "ContentFlow: An introductory routing proposal for Content Centric Networks using Openflow," *API, 7th Think-Tank Meeting*, pp. 1–2, June 2012.