Providing CCN functionalities over OpenFlow switches

Xuan Nam Nguyen, Damien Saucez, Thierry Turletti

Providing CCN functionalities over OpenFlow switches

Xuan-Nam Nguyen, Damien Saucez, Thierry Turletti

INRIA Sophia Antipolis Mediterranée, France

{FirstName.LastName}@inria.fr

Abstract

Content-Centric Networking (CCN) and Software-Defined Networking (SDN) are gathering special attention from academia and industry and are perceived as a potential future of networking. Providing CCN functionalities over SDN devices is an important requirement to enable the innovation and optimization of network resources. However, current SDN devices like OpenFlow switches do not support CCN functionalities such as name forwarding and caching. In this paper, we propose an approach to provide CCN functionalities over OpenFlow switches without having to change OpenFlow, by adding an abstraction layer, called the Wrapper, between OpenFlow and CCN. The preliminary results performed with CCNx, the reference CCN implementation, show the feasibility and the low overhead of this approach.

1 Introduction

Internet usage today becomes more and more mobile and is dominated by content delivery services. Nowadays end-users do not care of where contents come from or how. To fit today's trend, *Content-Centric Networking* (CCN) [5] has been proposed. The objective of CCN is to provide efficient content delivery framework and better cope with mobility and security issues than traditional networks.

Beside CCN, there is an important trend in networking: Software-Defined Networking (SDN) [4]. SDN decouples forwarding hardware from control decisions and promises to dramatically simplify network management, enable new services through its programmability. Enabling CCN funtionalities on SDN devices might open a new chances for optimizing network resources usage. However, current OpenFlow switches, the most notable implementation of SDN, do not support CCN functionalities such as name forwarding and content caching. So our research question is "*How to provide OpenFlow switches with CCN functionalities?*". In this paper, we propose an architecture which enables name-

based forwarding on OpenFlow switches. The main idea is to hash the name inside the fields which OpenFlow switch can process.

The rest of this paper is organized as follows. We begin with the state-of-the-art in Section 2. Then we describe our architecture in Section 3 and conduct the evaluation of our proposal architecture in Section 4. Finally, we conclude this work in Section 5 and open new perspectives for future work.

2 Related work

In CCN, the content is addressable and routers use content name as the primitive for processing. Content is published with an unique name to the network by content providers.

CCN communication are receiver-driven and composed of two types of packets: *Interest* sent by nodes who are interested in a content; and *Data* that contain chunks of the requested content. Each node in CCN maintains three data structures. First, The *Pending Interest Table* (PIT) maintains the list of interfaces (called faces) waiting for contents. Second, the *Forwarding Information Base* (FIB) that lists the next hops toward potential content sources. And third, the *Content Store* (CS) that caches chunks of contents that can be served by the node. A user who is interested in a content issues an Interest to one or several CCN nodes. Intermediate CCN nodes use their FIBs to forward the Interest packet towards potential sources, while keep state and aggregate Interests in PITs for content delivery after. Content which satisfies the Interests is encapsulated in Data packets and back-tracks the reverse-path to the user. Interestingly, CCN nodes may hold a copy of content in CS to serve future Interests. In other words, CCN enables in-network caching. CCNx [1] is the official implementation of CCN, which is currently under development. To be compatible with the existing architecture, CCNx builds an IP overlay to transport Interest and Data packets. The current version of CCNx is 0.8 and it supports Linux and Android platform.

In OpenFlow [3], the most common SDN implementation, the OpenFlow switch is responsible for packet forwarding and the network control runs on a logical centralized controller, which maintains a global view of networks. An OpenFlow switch contains a *flow table*, which is a set of *flow entries* that specify the forwarding rules for the flows passing the switch and use the OpenFlow protocol to communicate with the controller via a secured connection, over network itself (in-band control) or a dedicate network (out-of-band control). A flow entry has three components: *matching rule* – set of the packet header fields (current OpenFlow switches cover the headers at layer 2, 3 and 4 of the IP stack), *actions* – list of actions to execute (e.g., forwarding, drop, modifying) when it receives a packet matching with this rule, and *statistics* – list of statistics about the flows covered by the entry (e.g., number of packets, number of bytes, duration). An OpenFlow switch looks up its flow table. Upon matching between the packet header fields and a flow table entry, the switch performs the action specified in the matched entry. Otherwise, the switch enqueues the packet and

queries the controller to learn the action to perform on the packet to finally execute the action. The switch inserts the corresponding flow entry in its flow table in order to reduce interactions with the controller for similar subsequent packets. In this paper, we consider OpenFlow 1.0, as it is the most popular implementation on current switches.

The idea of combining alternative architectures of CCN and SDN appeared in [9] where Syrivelis et al. describe how to pursue a Software-Defined Information-Centric Networks and how SDN implementation should be modified to support their content-centric architecture called Blackadder. Melazzi et al. [6] propose to add a new IP option field and to extend the OpenFlow protocol to support their CONET architecture. However this proposal requires modifications in IP protocol and does not support CCN architecture.

3 Architecture

3.1 Overview

OpenFlow switches cannot process CCN packet since to determine the content name, they would have to perform deep packet inspection.

Instead of modifying OpenFlow to support CCN or CCN to be supported by OpenFlow, we follow an approach that requires modifications in neither OpenFlow, IP, or CCN. Therefore, we can directly use CCNx, the CCN reference implementation. However, CCNx daemon cannot work directly with OpenFlow switches as there is no mechanism for OpenFlow switch to understand the name inside CCN packets. So the main idea is that we build an intermediate layer between CCNx and OpenFlow, which we called the *Wrapper* to support them. The Wrapper takes every packet from the CCNx daemon, hashes its content name, and encapsulates it in an IP packet where the header fields that OpenFlow can match (e.g., IP source address, destination address) are forged to reflect the content name' hash value. In this manner, the OpenFlow switch is able to forwarding and monitoring packets based on their content name. For that, the controller has to decide actions based on the content names summarized in the hash valued spread over the IP packet fields.

There are several advantages of this approach. First, we exploit the OpenFlow hardware capacity for forwarding and monitoring CCN packets (e.g., maintaining Forwarding Interest Base); and CCNx daemon for content caching and managing state of Interests (e.g., maintaining CS and PIT). Second, we are also able to use OpenFlow centralized controller to populate name forwarding decision via OpenFlow protocol. Third, this approach requires modifications in neither OpenFlow nor CCNx allowing them to evolve independently. In case of protocol change, only the wrapper must be modified, minimizing so the time to market.

A first drawback of this approach is that the fields used to store the name information lose their original meaning. This limitation is however very limited as, on the one hand, within the OpenFlow network, the forwarding decisions are

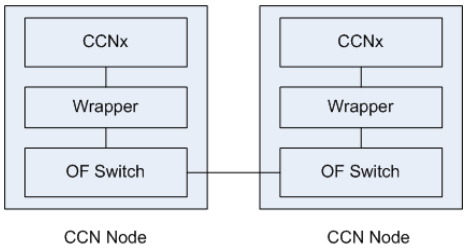

Figure 1: Integration of the Wrapper in nodes

consistent thanks to the centralized controller and, on the other hand, the fields modified by the Wrapper can be restored to their initial values if the Wrapper communicates this information to the controller. A second drawback is related to the forwarding performance as additional operations are necessary on every packet.

3.2 Design

Fig. 3.2 shows the three main components in our design: The OpenFlow switch, the Wrapper, and CCNx daemon. The Wrapper and CCNx are executed on an external machine which is connected to a dedicated port of the OpenFlow switch. CCNx carries CCN packets over UDP and the CCNx demon listens on port 9695 as default configuration. The CCNx instance and the Wrapper are connected via UDP (e.g., localhost:10001 <-> localhost:9695), each such link corresponding to a face in the CCNx demon.

The OpenFlow switch forwards every packets it receives from other ports to the Wrapper, and the Wrapper forwards it to the CCNx daemon. Furthermore, the OpenFlow switch needs to help the Wrapper knowing which port of the switch the packet comes from, so we borrow the ToS field – which can be modified by OpenFlow switches, to let the Wrapper know from which port the packet comes from. The OpenFlow switch is configured to set ToS value of all packets it receives to the corresponding incoming port value and then forwarding all of them to the Wrapper's port. So the rule on OpenFlow switch is the followings: *`in_port` = any except Wrapper's port ⇒ set ToS value to `in_port value`, forward to Wrapper port.* Upon reception of a packet from the Wrapper, the OpenFlow switch forwards it to its corresponding output port (e.g., *`in_port` = Wrapper's port, ToS = 1 ⇒ forward to port 1*).

Wrapper' behavior is shown in Fig. 2 and Fig. 3. The Wrapper needs to map a face of CCNx to a interface (i.e., port) of OpenFlow switches using ToS value. In our design, face `W` is a special face between Wrapper and CCNx daemon. `W` receives every Data packet from the wrapper and is used to send every Interest packets from CCNx to the Wrapper.

Upon the arrival of an Interest packet from the OpenFlow switch, the Wrapper extracts its ToS value and forwards it to corresponding face of CCNx. The

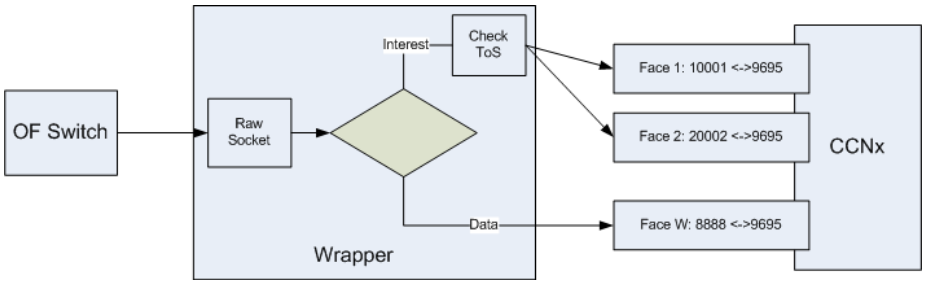

Figure 2: Packet flow from OpenFlow Switch to CCNx

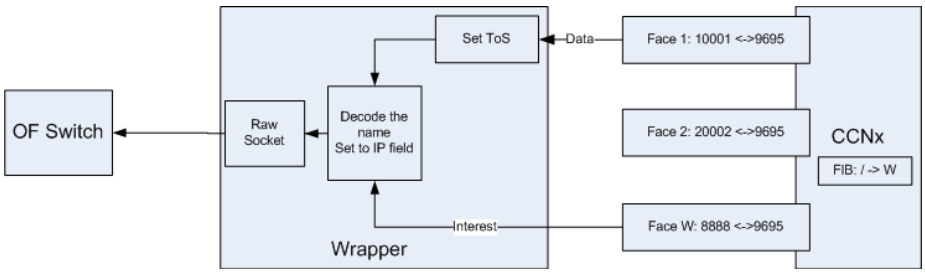

Figure 3: Packet flow from CCNx to OpenFlow Switch

Interest is then processed by the CCNx daemon. If CCNx hold a copy of content, it returns a Data packet back to the incoming face. Otherwise, it forwards this Interest to face `W` and update its PIT accordingly. Upon Data packet arrival from the OpenFlow switch, the Wrapper forwards it directly to face `W` (see Fig. 2).

When the Wrapper receives a Data packet from the CCNx demon, it sets the ToS field accordingly. Then, for any packet, it decodes the packet to extract the content name related to the packet. The name is hashed and the source IP address of the packet is set to correspond to the hashed value. Finally, the wrapper forwards the packets to OpenFlow switches (see Fig. 3). Data packets are returned to their corresponding incoming face. Interest packets have ToS value set to zero so they are forwarded to next hop by OpenFlow switch. At that moment, the OpenFlow switch might have to invoke the controller to make the decision (e.g., *`ip_src`=hash(/name) ⇒ forward to port `x`*).

We use the 32-bit of IP source address to carry the hashed value of content names; which means that 2^{32} contents can be tracked at the same time, which represents an temporary addressing space large enough. If this address space is too short, it can be extended by using other fields in the IP header such as the destination IP address or the IP identification field.

The wrapper is implemented in C and uses raw sockets. We use LibCCN for encoding and decoding CCN messages and the hash function djb2 [2] to map a string to an Integer 32-bit.

Element	Description	Note
OpenFlow Switch	Pronto 3290, 48 Ethernet (1Gbps)	Indigo Firmware, OpenFlow 1.0 Compatible
Controller	Dell Latitude E6400 Core 2 Duo P8400 2.4Ghz, Ram 3.4 GB, Fedora 13	Beacon Controller 1.0
PC1, PC2, PC3	Dell Latitude E6500 Core 2 P8400 2.4GHz x 2 RAM 4GB Ubuntu 12.04	

Table 1: Wrapper testbed description

4 Evaluation

To demonstrate the idea, we setup a testbed which is described in Table 4. We have three PCs connected connected to a Pronto 3290 Switch. PC1 runs an Interest generator that issues Interest for randomly chosen names. PC2 is a trace collector, and PC3 executes the Wrapper and CCNx daemon. The controller is used to install forwarding rules on the Pronto 3290.

In the first experiment, we check the compliance of the Wrapper with the CCN protocol by inspecting packets on PC2. As we can see on Fig. 4, the content name of CCN packets from the Wrapper is hashed into source IP field of IP packets (in the rectangle), and these packet is compatible with CCN protocol. During the experiment, we have noticed an important packet loss (20%) while handling Interest packets with short length name at high sending speed. The short names cause high packet overhead that leads to high packet loss while capturing. In our experiment, the size of content name is set to big value (1000 characters + numbering).

In the second experiment, we evaluate the impact of Wrapper to performance of the system in term of packets per second (pps) in three scenarios when forwarding Interest packets: (*i*) using the OpenFlow switch; (*ii*) using the OpenFlow switch and the CCNx daemon; (*iii*) using the OpenFlow switch, the wrapper and the CCNx daemon. We change the generating speed of Interest generator and measure the corresponding output.

Fig. 4 shows the outgoing rate (PPS out) for different incoming rates in three scenarios. At the low speed, there is no difference in PPS out for all cases. At high incoming rate, using only the OpenFlow switch gives the best PPS out because CCN packets are forwarded at switch level. In contrast, using the Wrapper slightly degrades forwarding performance but no more than 5%. This can be justified by the fact that the wrapper processes CCN packets coming from both the OpenFlow switch and the CCNx daemon. This results shows that the wrapper does not significantly degrade forwarding performance. These results have been reported in [7][8]. The source code of the Wrapper is available

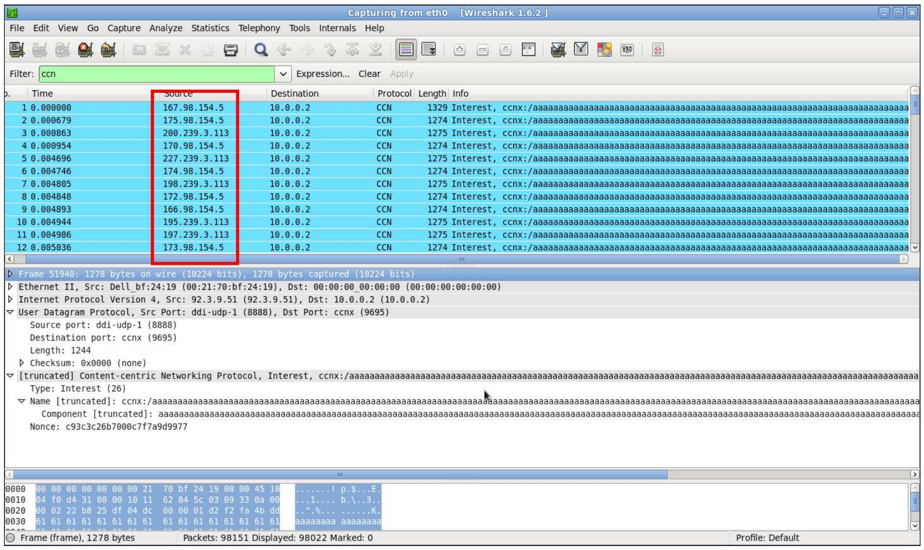

Figure 4: Captured packets from the Wrapper

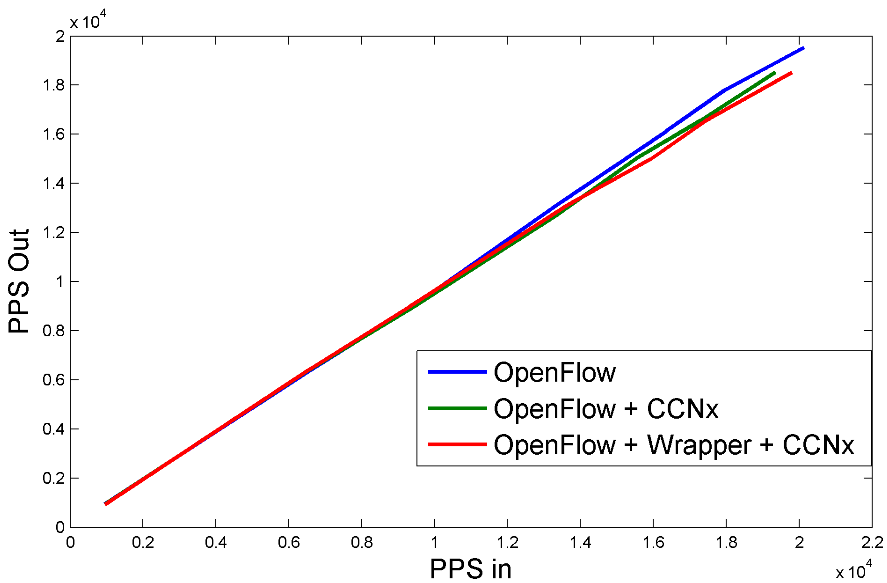

Figure 5: Performance Evaluation 2

at https://github.com/namnx87/Wrapper.

5 Conclusion

Providing CCN functionalities over SDN devices is an important requirements to enable the innovation and optimization of network resources. In this paper, we propose an approach to provide CCN functionalities over OpenFlow switches. The main idea is using the Wrapper to hash the content name of CCN packets into fields which OpenFlow switch can process. This preliminary results show that the Wrapper is feasible and does not significantly degrade forwarding performance. This architecture exploits the capacity of SDN hardware for name forwarding and monitoring and might open new changes for interesting optimization for CCN, like access control, traffic engineering, or cache optimization.

With the current architecture, we can use an OpenFlow controller to centralize the control of FIB. However a centralized control for CS and PIT might also be useful for further optimization, such as caching optimization.

References

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

[2] Djb2. http://www.cse.yorku.ca/ oz/hash.html.

[3] OpenFlow Switch Specification. https://www.opennetworking.org/.

[4] Bruno Nunes Astuto, Marc Mendonça, Xuan Nam Nguyen, Katia Obraczka, and Thierry Turletti. A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks. Research Report hal-00825087, INRIA - UCSC, October 2013.

[5] Van Jacobson, Diana K. Smetters, James D. Thornton, Michael F. Plass, Nicholas H. Briggs, and Rebecca L. Braynard. Networking named content. In *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, CoNEXT '09, pages 1–12, New York, NY, USA, 2009. ACM.

[6] N.B. 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*, pages 1 –9, july 2012.

[7] X.N. Nguyen. Software defined networking in wireless mesh network. Msc. thesis, INRIA, UNSA, August 2012.

[8] Xuan-Nam Nguyen, Damien Saucez, and Thierry Turletti. Efficient caching in Content-Centric Networks using OpenFlow. In *2013 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 67–68. IEEE, April 2013.

[9] D. Syrivelis, G. Parisis, D. Trossen, P. Flegkas, V. Sourlas, T. Korakis, and L. Tassiulas. Pursuing a software defined information-centric network. In *Software Defined Networking (EWSDN), 2012 European Workshop on*, pages 103–108, 2012.