Enabling ICN in IP Networks Using SDN

Markus Vahlenkamp, Fabian Schneider, Dirk Kutscher, Jan Seedorf
NEC Laboratories Europe, Heidelberg, Germany
<first.last>@neclab.eu

***Abstract*—In this work, we outline how to enable Information-Centric Networking (ICN) on existing IP networks, such as ISP or data center networks, using Software-Defined Networking (SDN) functions and control. We describe a mechanism that requires neither new or extended network/L3 and transport/L4 protocols nor changes of ICN host network stacks, and supports aggregation of routes inside the SDN controlled network. The proposed solution is agnostic of the specific ICN protocol in use, and does not require all network elements to be SDN-enabled. It supports advanced ICN routing features like request aggregation and forking, as well as load-balancing, traffic engineering, and explicit path steering (e. g., through ICN caches). We present the design as well as our first implementation of the proposed scheme—based on the Trema OpenFlow controller-framework and CCNx.**

I. Introduction

The Software-Defined Networking (SDN) paradigm proposes an open interface to network element (e. g., switches, routers) that enables programming the behaviour of entire networks. In this work, we apply SDN to a new type of networking: Information-Centric-Networking (ICN) [1].

In the ICN paradigm consumers send content requests to the network, asking for Named Data Objects (NDOs) that have been published before and that are available in one or many copies in the network. The network is thereupon responsible for the content acquisition and delivery to the consumer.

Existing Approaches to Realize ICN: An ideal or native deployment of ICN requires all user devices, content sources as well as all intermediary network elements to be ICN aware. We do not believe that in the near future such an ICN deployment is viable given the need to exchange or at least update all networking equipment installed today. In this work we present a migration path towards ICN, through SDN-enabling more and more parts of the network.

A slightly different solution to our approach has been presented by Blefari-Melazzi et al. [2]. Their approach is based on OpenFlow switches and dedicated border nodes which perform name-to-location resolution – with the help of an external system – for the requested NDO. The proposal defines a new IP option which is supposed to include the requested NDOs name and an ICN specific transport protocol.

Benefits of Our Solution: Our SDN backed ICN deployment seeks to advance the approach of Blefari-Melazzi et al., providing the following benefits: *(i)* Facilitate ICN deployment over existing networks. We particularly focus on initial and partial ICN deployments that can emerge with time. We achieve this by allowing for the use of off-the-shelf network protocols and network stacks in host operating systems, i. e., unmodified IP and UDP. *(ii)* Routing or content location learning, is simplified through separating routing and forwarding through an SDN approach. The centralized view of the controller enables easier path selection/finding and more sophisticated forwarding decisions. Moreover, state maintenance and complexity requirements on network elements can be kept low and actually moved between SDN switches and controllers. *(iii)* Routing can be further improved through knowledge about both interests/requests and publications/content locations. For instance, multiple requests can be aggregated into one or split across multiple paths.

978-1-4799-1270-4/13/$31.00 ©2013 IEEE

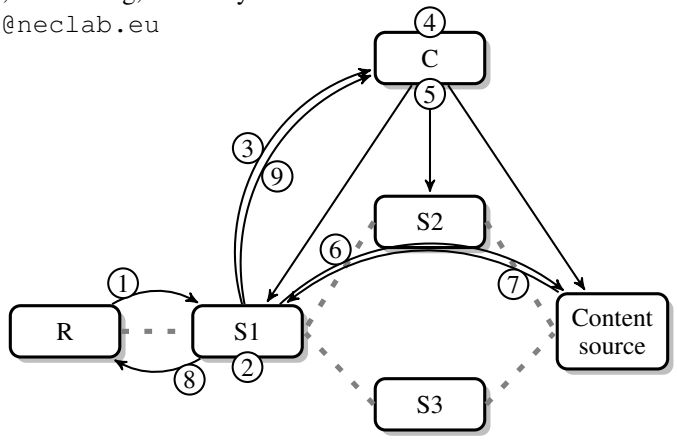

Fig. 1. Overall Operation

II. Proposed Approach

We assume an ICN protocol on top of the Internet Protocol, specifically we assume that ICN messages are transferred with UDP or TCP. The generic method is to enable an SDN controller to install appropriate forwarding state for an ICN request in a way so that network elements only have to support IP forwarding and do not require ICN protocol knowledge. This method is leveraging ICN protocol Message IDs and features of SDN instantiations such as OpenFlow [3] to rewrite packet header information. For our solution we require:

1) An SDN network element matchable *ICN Protocol Identifier* that is carried by all ICN packets.
2) A *publicly routable network address* (I.C.N.P.) per domain that is to be announced on every SDN enabled network element and made public via e. g., DNS.
3) The *object's name*, which is used to determine the path of the packets (routing).
4) A *Message ID*, which will be used for forwarding decisions in the network elements on the determined path. The Message IDs will replace destination IP addresses in requests and source IP addresses in responses.

Request Forwarding: Figure 1 shows the processing of ICN requests. When a Requester R wants to query an object from the ICN service an IP packet is sent (1) with destination IP address of I.C.N.P. The destination port number will be set to the ICN Protocol Identifier. As application payload R will follow the specification of the ICN protocol which includes at least the name of the requested object.

S1 will try to match (2) the packet to its rules and find that only the "default ICN" rule matches. This rule matches for the combination of the specific I.C.N.P. and ICN port. Subsequently the switch sends the packets to the Controller C. Upon reception of the packet by C (3), C will parse the ICN protocol payload and extract the requested object name. Next the location of the NDO, that is the address of a cache that can serve the requested object, is determined (4). Then the IPs and Port numbers of the packet are rewritten: The new destination IP is the IP of the cache from which the object should be served. The new source IP is the MsgID. The destination port number is kept to identify the packet as ICN. In addition the source port number is changed, in order to identify the "ingress" network element. Next (5) C installs forwarding rules on all the SDN network elements on the path to the content source and sends the re-written packet back to S1. Eventually (6), the packet is forwarded to the content source, the cache serving the object.

Response Forwarding: Steps 7-9 of Figure 1 show the processing of ICN responses. When requests arrive at the content source they will be processed and a corresponding response will be generated. The first SDN enabled element will forward the first response packet to the controller. This can be achieved by matching for ICN in the source port. However the controller might already have installed a specific (based on the MsgID) or aggregated (based on the IRI) rule to forward responses; in that case the packet(s) are not sent to the controller. Then (7) the network forwards the packet(s) to the origin, performing the address and port re-write on the egress SDN node. When the response has been completely delivered to R (8) the egress SDN node will notify C that all state for MsgID can be removed (9).

III. Implementation

The Controller implementation consists of different parts, as depicted in Figure 2. For each controlled switch, the Trema *Switch Manager* (not depicted) forks a *Switch Daemon*, that is responsible for the communication with its associated *OpenFlow Switch*. Each packet yielded by the *OpenFlow Switches* is, via the *Switch Daemon*, first delivered to the *Packet_In Filter*, whose task is to filter out Link Layer Discovery Protocol (LLDP) packets and deliver them to the *Topology* component, who is likewise generating and receiving those LLDP packets for the purpose of detecting the network topology. According to our *Packet_In Filter* configuration, all other packets are handed over to the *CCNx-SDN-Controller* process.

The *CCNx-SDN-Controller* maintains a *FIB* to manage the object name to ICN node address mappings. It learns this information from observing name prefix announcement traffic of the controlled *CCNx nodes*. Moreover, it processes the incoming packets and further sends information back to the *OpenFlow Switch*, via Packet_Out messages. The core *CCNx-SDN-Controller* further utilizes the *Topology* component to calculate paths through the network. The actual path provisioning is partially delegated to the *Path Manager*, that consistently handles the path creation.

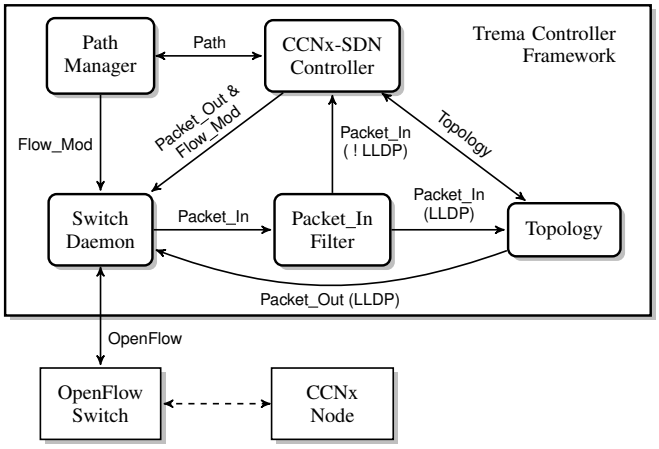

Fig. 2. ICN-SDN Implementation Architecture

IV. Conclusion

In this work we introduce a mechanism to deploy ICN protocols in IP networks via the assistance of the SDN paradigm. Our approach utilizes smart packet header rewriting in combination with a single IP prefix to initially contact the ICN network. This approach is backed by the centralized SDN controller, that is used to generate paths through the SDN controlled network domain. The benefits include ease of deployment in existing IP networks, not requiring additional transport protocols or extensions to the IP protocol and separation of the ICN data and control plane. The latter enables simpler ICN routing mechanism through centralized knowledge as well as improved traffic engineering capabilities. Furthermore our approach allows trading the need for keeping state in network elements for reduced interactions between the controller and network elements.

Acknowledgment

This work has been supported by the GreenICN project (GreenICN: Architecture and Applications of Green Information Centric Networking), a research project supported jointly by the European Commission under its 7th Framework Program (contract no. 608518) and the National Institute of Information and Communications Technology (NICT) in Japan (contract no. 167). The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the GreenICN project, the European Commission, or NICT.

References

[1] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman, "A survey of information-centric networking," *Communications Magazine, IEEE*, vol. 50, no. 7, pp. 26–36, 2012.

[2] N. Blefari-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 (FutureNetw), 2012*, 2012, pp. 1–9.

[3] OpenFlow project at Stanford, "OpenFlow Switch Specification, Version 1.0.0," 2009, http://www.openflow.org/documents/openflow-spec-v1.0.0.pdf.