Supporting Information-Centric Functionality in Software Defined Networks

Luca Veltri[1], Giacomo Morabito[2], Stefano Salsano[3], Nicola Blefari-Melazzi[3], Andrea Detti[3]

[1]University of Parma / CNIT, [2]University of Catania / CNIT, [1]University of Rome "Tor Vergata" / CNIT

Abstract—**The Information-Centric Networking (ICN) paradigm is expected to be one of the major innovation of the Future Internet An ICN can be characterized by some key components like: (i) the content-centric request/reply paradigm for data distribution, (ii) route-by-name operations, and (iii) in-network caching. In this paper we focus on a framework for ICN called CONET (COntent NETwork) and in particular on a solution devised under this framework called coCONET. coCONET characteristics make it suitable for deployment in accordance to the Software Defined Networks (SDN) philosophy. In this paper, we will describe how coCONET can be implemented over an OpenFlow (the most popular SDN instantiation, to date) network and how OpenFlow should be modified to better suit the operations of coCONET and, more in general, of ICN solutions.**

Information Centric Networking; ICN; Software Defined Networks; SDN; OpenFlow

I. INTRODUCTION

Recently there has been an increasing interest in the so called *Information-Centric Networking* (ICN) paradigm [1][2] which is considered one of the major characteristics of the Future Internet (FI) [3]. ICN proposes a paradigm shift from the traditional *host-to-host communication* – which has been at the very basis of the design of the architectures and protocols of the current Internet – to the *content-to-user communication* paradigm which poses the focus on the delivery of the desired content to the intended users. Motivations of such paradigm shift find their origin in a clear evidence [1][2]: users value the Internet for the content they can obtain from it rather than for the possibility to interact with a specific host. The increasing interest in ICN is demonstrated by:

- the large number of research project which are being carried out on the topic such as CONVERGENCE [4], Named-Data Networking [5], PURSUIT [6], and SAIL [7], for example;
- the large number of research papers appearing which focus on the topic (see the papers presented at the ACM SIGCOMM workshop on Information-Centric Networking 2011 or at IEEE Infocom Workshop on Emerging Design Choices in Name-Oriented Networking 2012);
- the proposition of the Information-Centric Networking Research Group (ICNRG) within the Internet Research Task Force (IRTF) [9] (still waiting for approval).

As a result of such increasing interest several solutions have been proposed for ICN. In this paper we focus on an ICN framework named *COntent NETwork* (CONET), and on a specific implementation based on this framework, called coCONET[1] [21] [10], which we will take as an exemplary ICN implementation. In the CONET framework each piece of content (called *chunk*) is characterized by a unique name. When a user is interested in a given piece of content its terminal generates an *interest message*, which is forwarded by network nodes towards the origin node that can provide the content. Forwarding process is based on the name of the content that is included in the interest message. When the content is sent back (in chunks) toward the interested terminals, intermediate nodes can cache it. Therefore interest messages can also be served by intermediate nodes storing a copy of such piece of content. At this level of abstraction, this approach is the same proposed in [1].

The CONET framework is modular and can support different solutions for fundamental issues like naming, name based routing, name based interest forwarding, data forwarding, transport protocols. In the specific coCONET solution that we will consider, the forwarding mechanism of interest messages (i.e. content requests) is called "lookup-and-cache". It foresees that network nodes, which do not know the next-hop node to forward an interest message, send a query to an appropriate layer called NRS (Name Routing System). The NRS answers providing the necessary routing information. It follows that network nodes are required to run forwarding operations only, while the creation and maintainement of forwarding rules is demandated to a "control plane" implemented by the NRS nodes.

The decoupling of switching/forwarding function from the routing and control functions is typical of the so called Software Defined Network (SDN) paradigm. In this paper we will discuss how SDN can be used to support the need of an ICN, starting from the concrete requirements coming from the specific ICN solution coCONET.

Probably one of the most popular implementation of the SDN paradigm is OpenFlow [8][12]. In OpenFlow, network nodes are called OpenFlow Switches and are responsible of forwarding while routing decisions and control function can be delegated in a centralized element called "Controller". OpenFlow Switches communicate with the Controller through the OpenFlow protocol. The success of such architecture is

This work was supported in part by the EU under the projects FP7–257123 "CONVERGENCE" and ICT-258365 "OFELIA"

1 coCONET: convergence CONET, named after the EU project CONVERGENCE

demonstrated by the fact that the OpenFlow protocol is already implemented by a number of commercial products, and available in several research and prototyping projects. Therefore we focused on OpenFlow as a reference implementation of SDN concepts.

When it comes to supporting ICN in OpenFlow, we see two basic options: a "short term" one, i.e. use the existing OpenFlow switches and API in order to realize the ICN functionality; a "long term" one, i.e. consider future, ICN capable switches and accordingly design an extended OpenFlow interface. We are exploring both options, but this paper deals with the latter one, focusing on the required enhancements to the OpenFlow architecture, interfaces and protocol that allow the supporting of ICN.

We are doing this work in the context of the OFELIA project [14], which provides a pan-European experimental platform open to researchers based on OpenFlow[2]. We plan to integrate our implementations (both for the short term and long term approaches) in the OFELIA testbeds, in particular we will build two additional OFELIA "islands" (i.e. in Rome and Catania) which will be able to run our implementations.

In section II we shortly recall some general ICN concepts, then provide few details about the CONET framework, the coCONET solution and its implementation. Section III discusses the two above mentioned "short term" and "long term" options. It also provides few details about the "short term" approach and its ongoing implementation. Section IV includes the main contribution of the paper, i.e. the analysis of OpenFlow extensions to support ICN. In Section V we discuss the status and the plan of our implementation activities.

II. ICN, CONET AND COCONET

While the debate on the specific procedures which should be executed by ICNs is still ongoing, there is almost universal consensus on the fact that an ICN solutions should support:

Content-centric request/reply paradigm for data distribution. Users interested in a given content issue an *interest* which is propagated in the network towards the host storing the desired content (called the *serving node*). Upon reception of an *interest* the serving replies by sending the desired data to the requesting user.

Route-by-name operations. Each data item is given a unique name and routing of all related information is performed based on such name. This allows the interaction between the users interested in a given content and the closest copy of the desired data item.

In-network caching. Routers in the end-to-end path between the serving node and the user can store in a local cache content. Accordingly, upon reception of an interest a router checks whether the requested piece of content is stored in the local cache. If this is the case then the desired content is sent to the requesting user while the interest message is no further propagated. If this is not the case (i.e., the piece of content is not in the local cache) then the interest message is propagated towards the serving node. Note that in principle, in-network caching can significantly improve efficiency of network resource utilization.

- ***Security embedded in the content.*** To date most of the effort has been focused on securing the communication channel between two end-hosts or two parties running a given application. In ICN solutions, instead, it is the content which matters accordingly mechanisms have been proposed which embed security information in the content itself and therefore avoid fake version of a content to be even disseminated in the network.

Content-Centric Networking (CCN) is one of the most relevant existing ICN solutions [1]. CCN can be implemented as a clean slate network in which a novel layer 3 protocol is meant to replace IP, or as an overlay layer, e.g., on top of UDP packets over an IP network. CCNx [23] is an implementation of CCN following the overlay approach.

CONET [11] [10] is an ICN framework based on the same general principles than CCN. Compared to CCN, its main contribution is the proposal of a third approach, in addition to the clean slate and the overlay ones, namely the "integration approach". The idea is to extend the IP protocol towards ICN support in a backward compatible way. This is achieved by adding an IP option (both for IPv4 and IPv6 headers) than can take care of ICN related information. CONET can be seen as a generic ICN framework that is capable to accommodate different choices with respect to specific aspects like naming, routing, forwarding, and caching.

Within the CONET framework we have designed a specific solution called coCONET by choosing the naming, the routing mechanism, etc. We have implemented the proposed coCONET solutions and realized a testbed. Further details and the open source implementation are available, see [21].

III. SHORT TERM VS. LONG TERM APPROACH

In this section we discuss how OpenFlow can be used to support CONET and, in general, ICN solutions. The OpenFlow standard has been created to process IP packets; forwarding rules can be created in terms of the header fields defined by IP and by standard transport protocols. Therefore, problems arise when dealing with non-IP packets, such as for "clean-slate" ICN solution. Likewise, current OpenFlow specifications allows to process TCP and UDP ports, but it is not possible to deeply analyze the payload, as it could be needed for efficient solution in case of overlay transport of ICN packets. Note that also an IP-based ICN solution, as proposed for CONET [11], that reuses the IP header for carrying ICN relevant information (e.g. content-name and chunk number) by exploiting the IP Options field, cannot be supported on the current version of OpenFlow, due to the lack of support of full IP header processing rules (the IP Options field in our case).

Two possible SDN approaches can be envisaged for building a ICN based on the OpenFlow protocol: i) a short term approach based on the version of OpenFlow protocol implemented in the available OpenFlow-enabled switches; ii) a long term (clean slate) approach in which the OpenFlow architecture is properly extended in order to accommodate non-

2 Instructions for accessing the experimental platform can be found at http://www.fp7-ofelia.eu.

IP packets, to support new (and more flexible) matching/forwarding rules and new switch processing functions as well. In this paper we concentrate on the analysis of OpenFlow extensions for fulfilling the second, "long term", approach and only provide some information on our ongoing work on the "short term" approach.

A. *Few notes on the short term approach*

We have designed and we are currently implementing a testbed in which OpenFlow 1.0 equipment is used to support coCONET ICN functionality (see [20][22]). As previously mentioned, CONET framework provides that content-name is carried by an IP option in the IPv4 header, but OpenFlow 1.0 equipment cannot parse IP options. To overcome this impasse, the basic idea is to map the content-name into a tag, added in the same header position that is used to transport TCP and UDP port number. In general, border nodes need to add/remove such tag when entering/exiting in/out a domain composed by a set of cooperating nodes that use OpenFlow technology. The tag will uniquely identify the content in such domain, more details can be found in [22].

IV. EXTENDING OPENFLOW

Following the SDN approach, we consider an OpenFlow-based ICN architecture in which the intelligence of the ICN is de-coupled from the forwarding (of interest messages and content data) and caching functions. These leads to an architecture that is summarized in Figure 1 and that is composed by two different planes, that are: i) a data plane with the ICN nodes, the serving nodes (i.e. the content producers), and the terminals (i.e. the content requesters/consumers); ii) a control plane that includes both the Name Routing System (composed by NRS nodes) and PKI for security. The two planes communicate through an *extended OpenFlow* interface described in this section. Such an extended OpenFlow interface is used by the NRS nodes (acting as OpenFlow Controllers) to control one or more ICN nodes (acting as OpenFlow Switches). We believe that such architecture is very well suited to support ICN and can help in overcoming some critical issues of ICNs like scalability. For example, as pointed out in [17], the size of the name-based forwarding table is a scalability issue for ICN routers. In [10] (see also [21] for further details) we argue that by keeping the full routing tables in the NRS nodes and using the forwarding tables as route "caches", scalability can be achieved. The architecture depicted in Figure 1 based on the extended OpenFlow approach, is perfectly suited to such scenario, where the routing intelligence runs in the NRS, implemented as a set of OpenFlow controllers.

Starting from the experience in the design and implementation of coCONET on one side and of the "short term" approach for integrating ICN operations with OpenFlow on the other side, we considered the more general problem of the evolution of OpenFlow in order to naturally support ICN. We observe that there are two directions of work for making OpenFlow compliant with the ICN paradigm, that are: i) revision/ extension of the OpenFlow packet matching rules; ii) extension of the OpenFlow protocol/API. Let us consider these two aspects in the following two subsections.

A. *Packet matching extension*

The first direction, which is already followed by the OpenFlow community, is to have a more flexible match in terms of new fields and protocol types that can be successfully parsed and matched with proper flow table entries. This is needed in all the three possible approaches for an ICN: clean slate ICN, overlay ICN, and IPv4 compatible layer 3 ICN, as in CONET. In the clean slate ICN approach, a complete new layer 3 protocol and new packet format is used. Currently no standard has been defined yet; however, whatever packet format will be defined for the new ICN layer 3, OpenFlow will need to match the new format and the corresponding header fields. In the overlay ICN approach, ICN packet are encapsulated into standard IP packet, for example as UDP payload; in this case OpenFlow would need to further process the incoming packet in order to elaborate the UDP payload to match the ICN packet fields. Finally, in the CONET approach, although standard IP packets are used, OpenFlow still needs to match ICN fields inserted as IPv4 or IPv6 options header fields. We will further examine this aspects later on in this section.

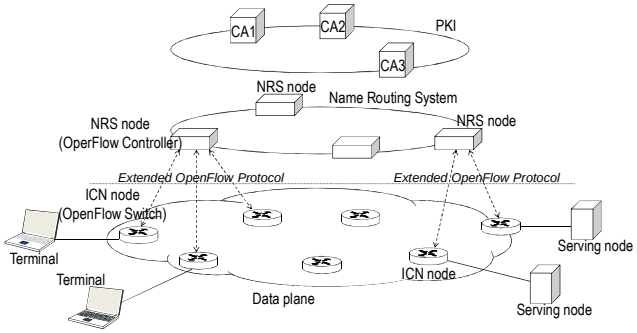

Figure 1 Information Centric Network based on extended OpenFlow

The definition of a new and more flexible match mechanism, in terms of new fields and protocol types that can be successfully parsed and matched by an OpenFlow compliant switch, has been already envisaged as an important requirement by the OpenFlow community, and it is already under discussion. New flexible match mechanisms may in fact help in both introducing new extensions to the current supported protocols (of the TCP/IP stack) and probably supporting future layer 3 and 4 protocols. Fortunately the OpenFlow Protocol and the corresponding API already consider the possibility of extending the current standard match mechanism by defining new OpenFlow *match types* (called OFPMTs) and the corresponding *mach structure* (i.e. the set of fields that compose the match) [15]. Hence, one simple extension could be to define a new match type that allows the matching of the entire layer two payload, seen as an opaque bit string, together with a variable-length matching mask. Whether this simple matching method works with an ICN packet format or not, depends on the particular ICN protocol considered. In case of CONET, this mechanism is sufficient to properly parse the CONET packet header and to make the flow tables able to forward the packet. Note that in case of CONET, due to the fact that rather than defining a new packet format, it exploits IP (v4 or v6) options field, it would simply require some matching rules against such IPv4 and IPv6 header fields. Unfortunately,

at the moment we are writing this contribution, the current OpenFlow protocol (version 1.1) [15] does not support packet match based on IP options fields. However we believe that this will available in standard mode in the next OpenFlow release.

B. OpenFlow protocol extension

However, flexible match, allowing the parsing of new packet formats, is just one step in the direction of supporting ICN, but it is not enough. In fact, having variable match allows the redirection of ICN packets somewhere, but does not fulfill all the functional requirements of an ICN node, like reactive or proactive caching, intelligent forwarding-by-names of interest messages, security, and other functionalities. This leads to the second direction of work that has a deeper impact on OpenFlow, as it concerns the support by OpenFlow of new ICN specific methods. In fact, in order to fully implement an ICN paradigm as described in the previous sections, the OpenFlow API should be changed or extended in order to better handle the concept of "content" and to support new content-related methods, such as key management, caching and routing-by-name, and so on. The new set of ICN-specific operations that can be supported by an ICN node and by the corresponding control protocol, can be classified as: "content-related", "routing-related", and "security-related", hereafter summarized.

1) Content-related operations

Interest messages handling – An ICN node is requested to handle incoming content interest messages that have to be properly routed based on a forwarding-by-name strategy (see later). At this phase the node could also need to keep track of requested contents (for example if "Pending interest" state is used to forward back the content data) or it may want to keep track of the requests to optimize the processing of further interest requests for the same content received from other interesting end-nodes. The ICN node could keep track locally or delegate such operation to an external controller. Moreover, also in case of stateless interest processing (no information is maintained for the already processed interest requests), in order to better operate caching decisions, the node may still want to inform the controller about the received request. This may be performed by notifying the controller on a per-request basis (every time a new interest message arrives) or on a batch basis, sending to the controller some periodic summary information reports about the requested contents. In this way the controller can build ordered lists of most popular content, useful also for the following caching decision.

Caching decisions – An ICN node may provide caching functionality for achieving a native, in-network caching function. This may drastically improve multicast content distribution (when the same content has to be distributed from one source to multiple destinations), and, more in general, it allows a more efficient content delivery in both fixed and mobile environments [16] (when the same content is successively requested by to other destinations). Due to the large amount of packets (and contents) that a node is requested to forward, it is expected that a node decides to cache only a portion of the overall forwarded contents. Such decision of which content is to be cached and which not, could be made locally, inside the node or, more likely, by relying on a logically centralized controller. A further decision related on caching, is on which content has to be removed from cache when the cache is full and new content has to be added. Also this decision could be delegated to an external controller.

Caching notification – It could be expected that the controller is notified when some content has been cached by a node. This in useful order to have an updated map within the controller of the availability of contents.

Proactive caching – The controller can proactively push some content in an ICN node, therefore anticipating the "automatic" caching procedures that are the only solution if a purely distributed approach (i.e. without OpenFlow) is used. These "proactive push" approaches could prove very useful for example for distribution of live content in ICN (e.g. audio/video real-time streaming).

2) Routing-related operations

An ICN adopts an addressing scheme based on names, which do not include references to their location; as consequence, interest messages have to be routed toward the closest copy of the content, based on a "destination" content-name. The main routing-related operations are:

Forwarding-by-name – When receiving a new interest message, the ICN node is expected to forward it to a next-hop node on the basis of the targeted content name. This forwarding operation is performed through a proper name-based Forwarding table present within the ICN node that maps possible content names to the corresponding next-hops. The insertion of new entries in such routing table is in charge of a logic entity that in our architecture is mapped onto the controller. When a new content is requested with a name that is not in the local forwarding table, the node requests the name-to-next-hop lookup to the controller. If a route is found and returned, this is stored into the forwarding table. Particular care should be taken when the forwarding table is full and a previous entry has to be deleted and replaced by the new one. Such decision may be assisted by the controller (for example through some priority information), or can be autonomously taken by the node on a basis on fixed and simple rules (e.g. "do remove the oldest entry").

Forwarding table management – Such forwarding table is dynamically updated each time a name-to-next-hop is requested to and returned from the controller. However the controller is expected to have the possibility to populate and modify the forwarding table according to some upper level strategy (for example by distributing the forwarding-by-name information for the most popular contents, or for contents that require some form of prioritization). Such operations should be controller-driven and they can be executed asynchronously with respect to the incoming packet events.

Forwarding table exportation – In case the controller does not keep a copy of the routing table of each node attached to it (e.g. for scalability reason), it is required that the ICN nodes could send to controller their current routing-by-name information stored in the table.

3) Security-related operations

An ICN node is expected to exploit security information embedded in the content to avoid the diffusion of fake versions

of contents and to protect the content, as opposed to exploit connection-based or application-based security [18].

Security enforcement – Contents (or content chunks, like in CONET) are cryptographically protected in order to assure content (and content generator) authentication and data integrity. This security service is provided through digital signature and can be verified through the public key associated to the private key of the content (or of thecontent generator). Every ICN node should verify such signature before forwarding the content toward the interested end-nodes, in order to protect the network against DoS or other attacks. Such function in turn requires that the ICN node obtain the public key associated to the content. One solution can involve the NRS node (that acts as controller) that may provide the public key together with routing information (see next point). Other possible solutions could be to use identity-based cryptography, or self-certifying names. The investigation on what solution better fits the ICN paradigm and requirements is out the scope of this paper. The CONET framework can support all these solutions.

Key management and distribution – In case some human readable names are used, and an association between names and public keys is required, this should be executed by the controller NRS node, according to a proper key management mechanism (e.g. through the use of a public key infrastructure - PKI, the use of a Web of Trust, or other key management mechanism). The result of such mechanism should let the NRS node to be aware of the correct name-to-key associations and be able to pass this information to the ICN nodes.

Key revocation – In parallel to a proper key management and distribution mechanism, it should be implemented also a key revocation mechanism that allow the revocation of compromised or withdrawn keys. A part of the properly selected mechanism, it will still in charge of the NRS node to communicate such revocation information to the ICN node.

4) Analysis of operations

According to the previous description, all ICN-related operations that involve both the ICN and NRS nodes can be driven: i) by the NRS node, on the basis of a control logic; this happens asynchronously respect to the ICN node, or ii) by the ICN node, when a new networking event happens, e.g. when interest or content packet arrives. In the former case the operation may start in correspondence of an internal timeout or in accord to a control logic executed in the control plane between different NRS nodes. In the latter case instead, the operation can be driven by six different type of packet events at the ICN node:

- An interest message arrives to the switch - no route to content (& no content in cache) is present. Hence these following operations are required: *Handling of content interest request, Name lookup, Key distribution*.
- An interest request arrives to the switch – a route is available but no content is cached. Only this following operation is now executed: *Handling of content interest request.*
- An interest request arrives to the switch: content available. The request can be fulfilled and only the following operation is executed: *Handling of content interest request.*
- A data packet arrives to the switch. The packet is forwarded to all the destination (that requested such content), and the following operation is executed: *Caching decision.*
- A full chunk of content arrives to the switch. In addition to the operations executed for data packets, the following operation is required: *Security enforcement, Caching notification*.

According to the above operations, a NRS node (OpenFlow Controller) should be able to command the ICN node (OpenFlow Switch) to operate the following atomic ICN-related tasks:

- (C1) Add/remove a route entry in the interest routing table;
- (C2) Add/remove a chunk of content in the switch cache;
- (C3) Add/remove a key for security checks in the switch key repository;
- (C4) Add/remove an entry in the Pending Interest Table.
- (C5) Configure/change caching policy.
- (C6) Query for caching capability.

Moreover, if the ICN capable node is used to distribute live content:

- (C7) Add/remove an entry in a static Pending Interest Table

Likewise the ICN node should be able to query for name-lookup, notify events, and update state information as follows:

- (S1) query for a name-lookup and routing information;
- (S2) query for a content-name public key;
- (S3) notify of different content related events (e.g. arrival of an interest message that does not require name lookup, content chunks completed, failure in authentication, etc).

Such new methods should be properly encaspulated in the following three standard OpenFlow types of messages:

controller-to-switch – initiated by the controller and used to manage the state of the switch; they may or may not require a response from the switch; the following methods are supported: Feature, Configuration, Modify-State, Read-State, Packet-out, Barrier. New Modify-State messages should be defined for C1-4, and C7. New Configuration message should be defined for C5. New Feature message should be defined for C6.

asynchronous – initiated by the switch and used to update the controller of state changes or network events; the following main methods are supported: Packet-in, Flow-Removed, Port-status, Error. New Packet-in messages should be defined for S1-2. A new *asynchronous* method should be introduced in order to handle S3 non-error event notification.

symmetric – initiated asynchronously by either the switch or the controller and sent without a solicitation by the other party; the following methods are defined: Hello, Echo (and Experimental). No specific new *symmetric* message is required.

V. Implementation Activities and Plans

We are currently implementing the proposed OpenFlow-based ICN architecture in the OFELIA project testbed [14], pursuing both the “short term” approach that was only mentioned in section III and the “long term” approach which

was extensively considered in previous section. Following the "short term" approach (i.e. using existing OpenFlow switches), the goal of the implementation is to show how Information Centric functionality can be realized on top of OpenFlow. Following the "long term" approach (i.e. assuming OpenFlow switches that are natively ICN enabled) we have the more ambitious goal of realizing an ICN capable SDN network that can be exposed to experiments.

The implemented architecture is the one described in the previous sections, using the content protocol and packet format described in [11]. For the ICN nodes (both border and internal nodes) that include forwarding and caching functionality we use Linux boxes with Open vSwitch. We are modifying the current Open vSwitch implementation in order to adapt it to our architecture. In particular the main changes/extensions that we are introducing in Open vSwitch are: i) allowing the forwarding based on a forwarding-by-name strategy (proper new forwarding tables and table management mechanisms); ii) implementing the OpenFlow extensions described in section IV, in order to support content-related, routing-related, and security-related operations. Caching functionality are implemented through a new caching unit, realized in the same Linux node, including both content storage and retrieval functions and caching control logic. The caching unit communicates with the controller (NRS node) via the ICN-modified version of the OpenFlow protocol. For the controllers (NRS nodes) we use Linux boxes with the nox [13] OpenFlow Controller. OpenFlow 1.0 has been used as starting version for the implementation/modification of the control protocol between NRS node (acting as OpenFlow controller), the ICN nodes (both border and internal node), and cache unit. According to the timeline of the OFELIA project our plans are to release the "short term" implementation by late spring 2012 and have a first "long term" release by late summer 2012.

VI. CONCLUSIONS

In this paper we have discussed some issues related to the application of SDN concepts to Information Centric Networks. We are considering how OpenFlow architecture and protocols can be modified and enhanced to support ICN. Our work is at early stage and we have reported here our first analysis of the ICN functionality that can be offered on a controller/switch interface based on the OpenFlow architecture. Based on these first results, we believe that the application of SDN concepts to ICN is feasible and can bring important benefits. We think that the SDN research community should start discussing the support of ICN, this work is a contribution to the discussion.

REFERENCES

[1] V. Jacobson, D. K. Smetters, J. D. Thornton, M.F. Plass, N.H. Briggs, and R. L. Braynard, "Networking Named Content", Fifth ACM International Conference on emerging Networking EXperiments and Technologies (CoNEXT), 2009.

[2] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica. "A Data-Oriented (and Beyond) Network Architecture". Proc. of ACM Sigcomm. August 2007.

[3] European Future Internet Portal, http://www.future-internet.eu/

[4] The CONVERGENCE EU FP7 project, http://www.ict-convergence.eu

[5] The Named-Data Networking (NDN) project, under NSF Future Internet Architecture (FIA) program, http://named-data.org/

[6] The PURSUIT EU FP7 project, http://www.fp7-pursuit.eu/

[7] The SAIL (Scalable & Adaptive Internet Solutions) EU FP7 project, http://www.sail-project.eu/

[8] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling Innovation in Campus Networks". White paper. March 2008 (available at: http://www.openflow.org).

[9] http://wiki.tools.ietf.org/group/irtf/trac/wiki/icnrg

[10] A. Detti, N. Blefari-Melazzi, S. Salsano, and M. Pomposini. "CONET: A Content-Centric Inter-Networking Architecture", Proc. of ACM Sigcomm – ICN 2011. August 2011

[11] A. Detti, S. Salsano, N. Blefari-Melazzi. IPv4 and IPv6 Options to Support Information Centric Networking. Internet draft (draf-detti-conet-ip-option-02). November 2011.

[12] http://www.openflow.org/

[13] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker, "NOX: Towards an Operating System for Networks", ACM Computer Communication Review. pp. 105-110. July 2008

[14] A. Köpsel and H. Woesner, "OFELIA – Pan-European Test Facility for OpenFlow Experimentation", Lecture Notes in Computer Science. Vol. 6994/2011. 2011

[15] B. Pfaff, et al., "OpenFlow Specification", Version 1.1, February 28, 2011, http://www.openflow.org/documents/openflow-spec-v1.1.0.pdf

[16] K Katsaros, G. Xylomenos, G. C. Polyzos: "MultiCache: An overlay architecture for information-centric networking", Computer Networks, Elsevier, Volume 55, Issue 4, 10 March 2011, Pages 936-947

[17] Diego Perino, Matteo Varvello, "A reality check for content centric networking", ACM SIGCOMM Workshop on Information Centric Networking (ICN), Toronto, Canada, August 2011

[18] D. Smetters, V. Jacobson: "Securing Network Content", PARC technical report, October 2009

[19] S. Oueslati, J. Roberts, N. Sbihi: "Ideas on Traffic Management in CCN", Information-Centric Networking, Dagstuhl Seminar

[20] N. Blefari-Melazzi, A. Detti, G. Mazza, G. Morabito, S. Salsano, L. Veltri: "An OpenFlow-based Testbed for Information Centric Networking", Future Network & Mobile Summit 2012, 4 - 6 July 2012, Berlin, Germany

[21] N. Blefari Melazzi, M. Cancellieri, A. Detti, M. Pomposini, S. Salsano, "The CONET solution for Information Centric Networking", Technical Report, December 2011, http://netgroup.uniroma2.it/CONET/

[22] G. Mazza, G. Morabito, S. Salsano, "Supporting COntent NETworking in OpenFlow", Technical Report, November 2011, http://netgroup.uniroma2.it/CONET/

[23] http://www.ccnx.org/