2012 European Workshop on Software Defined Networking

Pursuing a Software Defined Information-Centric Network

Dimitris Syrivelis *, George Parisis†, Dirk Trossen†, Paris Flegkas*,
Vasilis Sourlas*, Thanasis Korakis* and Leandros Tassiulas‡
*CERTH-ITI, Greece
Email: {jsyr, pflegkas,vsourlas, korakis} @iti.gr
†Cambridge University
Computer Lab, Cambridge, UK
Email: {georgios.parisis, dirk.trossen}@cl.cam.ac.uk
‡Dept of Computing Engineering and Telecommunications
University of Thessaly, Volos, Greece Email: leandros@inf.uth.gr

***Abstract*—The areas of Software-Defined Networking (SDN) and Information-Centric Networking (ICN) have gained increasing attention in the wider research community, while gaining credibility through corporate interest and investment. With the promise of SDN to simplify the deployment of alternative network architectures, the question arises how SDN and ICN could concretely be combined, deployed and tested. In this paper, we address this very question within a particular architectural context for ICN. We outline a possible realization in a novel design for ICN solutions and point to possible testbed deployments for future testing.**

I. Introduction

Software-Defined Networking has been touted as an approach to bring high programmability to network components by disintegrating the forwarding function of a network into an efficient fast path and a programmable slow path. It is the extensibility introduced through the latter, which is seen as enabling the development of new routing and forwarding approaches without the need to replace hardware components in the core network. This allows for virtualizing entire networks on a per-flow basis, a significant improvement over existing solutions. Beyond the research community, SDN has received significant endorsement through companies such as Google, building out dedicated SDN-based infrastructure for supporting their user-facing Internet services, while several manufacturers are working on making future components SDN-compliant.

Information-Centric Networking has been gaining attention among those who would like to replace the current IP-based internetworking layer with a new solution. Based on the observation that the WHAT in a communication is often the primary objective rather than connecting to a specific WHO, this leads to declaring information as the main principle in all of the major approaches to ICN, such as [1]-[3]. Another crucial aspect of ICN approaches is the separation of functions that control the matching of information demand and supply from the functions that deal with the forwarding resources within a network.

This separation of functions seems to be aligned with the separation of network configuration control and forwarding in SDN. Hence, it begs the question as to the opportunities that combining SDN and ICN could potentially bring about. Beyond the potential technical aspect of such combination, there is another possible advantage of combining SDN and ICN technologies. The authors in [4] point out that ICN deployments are more likely to be driven by powerful players and industries such as the content industry, the financial services industry or the healthcare industry, rather than relying on an adoption of ICN through key core Internet players. We see this edge-driven deployment somewhat aligned with the recent drive to adopt SDN solutions for building out, e.g., Google's infrastructure. Hence, we assert that combining ICN solutions with SDN deployments could drive the possible adoption of ICN approaches, raising the combination of SDN and ICN from a purely technical question to an interesting business strategic one.

In this paper, we take such interest in combining ICN and SDN for granted and focus on a possible integration of both solutions at a technical level. For this, we structure the remainder of the paper as follows. We first outline in Section II the architectural context that we assume for ICN. We then present a node design that realizes our architectural context in Section III, forming the basis for integration our SDN solution. Section IV addresses the core question of what a (SDN) flow in our architectural context would be, leading to realizing our ICN architectural functions based on that flow notion. Finally, in Section V, we outline a possible testbed deployment for our combined solution before concluding the paper.

II. Architectural Context

The architectural context with which we tie a Software-Defined Network approach is the one presented in [1].

Information labeling in contrast to end-point addressing. The major shift in the way communications take place is the fact that end-points are not explicitly addressed. Instead, information is identified using *statistically unique fixed sized labels*. These labels carry no semantics and are meaningless to most network components and applications.

Information aggregation through scoping. Information must always reside within a context, called *scope*. Hence, a scope represents a set of information and is therefore an information item itself, being identified as such with an individual label. Being information items, scopes can be nested under other scope(s), allowing for building complex directed acyclic graphs of information. Scoping is crucial when scaling information management to real information structures. Moreover, as we discuss in our fifth principle, scoping provides a way to *functionally differentiate* the way information is disseminated to the network by assigning different dissemination strategies to sub-graphs of information.

Decoupling of communicating parties. The primitive service model that is exported to all applications is a publish/subscribe one. Hence, communicating entities do not know the location (based on identifiers or end-point addresses) of each other. Instead, each node is (self) assigned with a statistically unique node label that is only used by the core network functions to resolve requests for information.

Separation of network functions. Core network functions are cleanly separated, as discussed in [5]. Each node supports three network functions that realize the dissemination of information within a given scope of the information structure. The first one, rendezvous, matches demand for and supply of information. This process results in some form of (location) information that is being used for binding the provisioning of information to a network location. This information is used by the second function, topology management and formation, to determine a suitable delivery relationship for the transfer of the information, this transfer being executed by the third function, forwarding.

Flexible information dissemination based on information scoping and well-defined strategies. The fifth principle addresses the methods used for implementing the aforementioned functions as well as issues regarding information space governance and management. For this, dissemination strategies, which are associated to (parts of) the information structure, capture the implementation details. Together with scoping, they establish a functional scoping through which the core functions can be optimized based on requirements of communicating entities that access specific parts of the information graph. For simplicity, we assume a domain-local dissemination strategy in this paper, i.e., information is disseminated across a single domain. Domain-wide rendezvous can be realized in dedicated network nodes that may share or replicate the information structure and the interested nodes (publishers and subscribers). Topology management is realized in dedicated nodes that know the network topology and are notified when nodes attach to or detach from a forwarding node. Depending on the domain size, one or more cooperating topology managers may be required. Finally, forwarding is realized using LIPSIN [6] identifiers, which natively support source-based multicasting in small to medium scale networks.

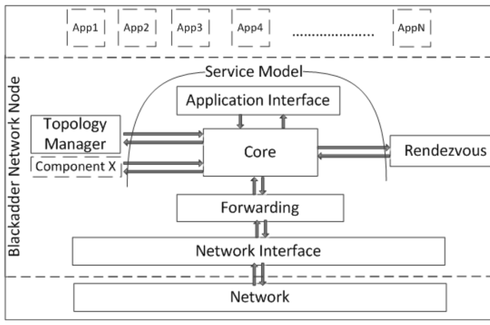

Fig. 1. Node Design.

III. Node Design

In this section we briefly describe the design of a network node in our ICN architecture, which is currently implemented as a Click [7] router. Our prototype is publicly available in [8]. Figure 1 presents the components that comprise the ICN network node (called *Blackadder* in the remainder of the paper).

The *Rendezvous* component implements the respective network function. All publish/subscribe requests finally reach this element (in a network node), which manages the information graph, matches publishers with subscribers and triggers the formation of a forwarding path by publishing a request to a Topology Manager.

The *Topology Manager* component manages the network topology and upon request by a Rendezvous component, it creates forwarding paths, expressed as LIPSIN identifiers, from one or more publishers to one or more subscribers, for a specific information identifier. These paths are, then, published as information to the respective publishers that utilize them when publishing information for a specific information identifier. During the bootstrapping process, a network attachment protocol is used to assign a new node with Link identifiers which are published to a TM component which updates the network graph and configures the forwarding information on the new node.

The *Forwarding* component implements one of the three network functions described in the previous section. In our current prototype, it utilizes LIPSIN identifiers [6] to forward publications to other nodes or its network stack.

The *Core* component implements the publish/subscribe service model described in the previous section. It receives all publish/subscribe requests sent by applications and other node components and, according to the dissemination strategy, forwards them to the local rendezvous component (e.g. in a node-local strategy) or publishes them to the network (e.g. in a domain-wide strategy). Moreover, it receives publications from the network and dispatches them to subscribed applications. Finally, it receives publish/subscribe notifications from the network or from the local rendezvous component and notifies all interested applications. For instance, when subscribers for an information item, which was previously advertised by a local application, express their interest in a rendezvous node, it will receive and forward to the application a request to start publishing data for that item.

Note that, as shown in Figure 1, the Topology Manager and Rendezvous components utilize the same publish/subscribe service model that is exported to all applications. Therefore, their implementation can be realized within the network node or as separate applications. Currently, we implement the Topology Manager component as a separate C++ application while the Rendezvous component is integrated in the Click configuration. For a given domain, the Topology Manager and Rendezvous components subscribe to a globally agreed information scope where they receive requests in the form of publications from all the domain nodes. In addition, the Topology Manager is pre-configured with the forwarding identifier to the Rendezvous node and vice versa, while the former pre-calculates the LIPSIN forwarding identifier from any node to the Rendezvous node and delivers it during network attachment. As a result, when attachment is complete a node has the necessary information to contact Rendezvous for subscription and publication requests. When the Rendezvous needs to send a notification to a publisher or a subscriber it sends it via a proper publication to the Topology Manager which in turn sends it to the recipient node. This is because the Topology manager has all the necessary information to calculate the forwarding identifiers from publishers to subscribers (in case it is a notification to start publish) as well as the required forwarding identifier to deliver the notification to the recipient. More details about the interaction of the internals can be found in [17].

IV. ICN in a Software-Defined Network

While software-defined networking (SDN) has been reported [10],[11] as a powerful tool for supporting content distribution, there has been some preliminary research work on how an OpenFlow deployment can fully support an information-centric approach. In the following, we will address two key issues when attempting to realize an ICN architecture that uses a software-defined network. The first is that of mapping the notion of a flow into an ICN, while the second is concerned with realizing the core ICN functions of our architectural context with SDN concepts.

A. *What is a Flow in the proposed ICN design?*

SDN is in large part about understanding and managing a network as a unified abstraction via a centralized software-driven control plane. Scalability issues have already been addressed using a hierarchical design of SDN controllers [12], [13]. A controller manages the data plane of one or more network switches by changing the flow table entries on each device. This is achieved via the OpenFlow protocol [14], which defines the types of flow modification operations as well as the communication messages between switches and the controller.

From an Information-centric perspective publishers get associated with subscribers and push information. The volume of data under an information identifier is segmented to the underlying media maximum transmission size units (MTUs) and, based on the currently active subscriptions, these information packets are delivered to one or more subscribers. From a switch perspective, each switch datapath needs to match a forwarding identifier that is prepended on each information packet with a list of forwarding actions to the required egress ports where subscribers reside [1]. Note that the forwarding identifiers provide only forwarding information to reach subscribers from publishers and are not associated with information identifiers in any way.

In case a different piece of information from the same publisher requires the same delivery graph to subscribers, the same forwarding identifier is used, without requiring any additional OpenFlow entries. This is the main reason for using a forwarding-specific identifier, which represents actual routes to destinations, rather than the content names themselves, as it happens in the design proposed in [15]. For example, one physical machine will be usually subscribed to and/or publish a few thousands of information items simultaneously, many of which might be exchanged with the same machines or caches over the network. If we programmed the SDN datapaths using content names, we would have many entries with the same forwarding action list that would route content between involved machines that would quickly exceed the datapath forwarding rule table capacity. Since the forwarding identifiers in our case are built to be unique for each forwarding action list, we just have one unique entry per forwarding action list that can be reused for delivery of different content that gets exchanged between the same machines. Finally in our design, when information packets need to cross several switch datapaths to reach all subscribers, the proper uplinks on each switch are also considered as subscribers and the same forwarding identifier is associated with a different list of forwarding actions in the context of each switch datapath that is involved in the transfer.

In the described context of Section II, a flow is identified by a forwarding identifier that requires a different list of forwarding actions inside each switch datapath. Therefore, the scope of a flow is the switch datapath. The ICN topology manager, as we have described in Section II, has all the necessary information to build a statistically unique forwarding identifier that encodes the information delivery graph in each case. Each openflow controller that governs one or a set of switch datapaths, will get the necessary information during bootstrap so it can decode the forwarding identifier and assign the required local egress ports to the list of forwarding actions for each datapath.

B. *Realizing the Core ICN Functions*

In the following, we explore how the core network functions defined in our fourth design principle, can be realized within an OpenFlow deployment in an efficient and scalable way. One of the important aspects that facilitate such integration is the clean separation of functions within the described ICN

[1]As outlined in Section II, the current domain-local dissemination strategy implements a forwarding identifier as a Bloom filter encoded graph of all links from the publisher to the subscriber.

architecture. Such separation, along with the realization of each function in specific dissemination strategies, provides a large design space that allows for deployment-specific optimizations. Based on the SDN basic concepts and the OpenFlow protocol, information dissemination can be done using an SDN-specific strategy.

As we have previously discussed, the *topology and formation* network function has all the domain node deployment overview in a form of a graph. Having this information centrally available allows for graph adjustment operations to avoid loops in multi datapath interconnections. In the SDN deployment the graph will be primarily defined by the available switch datapaths and the way they are interconnected. For each switch datapath, just a map of ports, attached nodes and local link identifiers is needed. Each network node along with the respective port and uplink(s) on each switch datapath will be assigned a fixed length, statistically unique link identifier. Using the LIPSIN [6] bloom filter-based encoding scheme, the topology manager can build forwarding identifiers for an information delivery graph by encoding the respective link identifiers of the subscriber egress ports. Switch uplink identifiers will be included when more than one datapath is involved in the transfer. The constructed forwarding identifier will be sent via a publication back to the publisher. For that publication the topology manager will also construct a forwarding identifier. The network nodes are notified during bootstrap of the forwarding identifiers to the topology manager and the rendezvous node. Note that topology management utilizes the same publish/subscribe service just like any other application in the network. It also needs to be a virtually centralized entity within a domain and it can run on any node on the network, even collocated with one or more openflow controllers. Special care must be taken when calculating LIPSIN identifiers so that false positives are avoided. In [6] there is a discussion about this issue and how rare it can be with a proper identifier size.

The ICN core function that is primarily supported by SDN is the Forwarding which is assisted by the Openflow controllers that govern the switch datapaths. For each switch datapath the topology manager publishes to the controller, during network bootstrap, all the link identifiers that are assigned to the governed switch datapath(s). A possible implementation of the bootstrap mechanism will be presented later. After initialization takes place, when a packet bounces in a datapath, the respective controller will receive it and check if local link identifiers are encoded in the packet forwarding identifier. For all the link identifiers it finds, it adds a subsequent port forwarding action to the action list and finally installs the respective flow entry that matches the forwarding identifier. The process will be repeated for each datapath. The controllers that control the switch datapaths are independent and can be distributed to available physical machines to improve Openflow configuration response performance. After proper flow installation on all involved datapaths takes place, the forwarding is carried by the network.

The *Rendezvous* network function is independent from the specific changes we made to the other two core functions. We can therefore re-use the rendezvous function that is currently already available for domain-local dissemination strategies, i.e., a centralized rendezvous server that serves the local domain. Any optimization for this domain-local function, such as through a highly replicated distributed solution, is independent from the specific SDN-based delivery mechanism we proposed here. Therefore, the rendezvous function can be independently optimized. The same holds for an extension of the domain-local rendezvous towards a global rendezvous solution, such as proposed in [16].

In Figure 2, we present an ICN architectural blueprint for nodes attached on two different SDN datapaths. A selected node on datapath 1 runs the domain rendezvous function and a selected node on datapath 2 run the topology manager function. Each datapath has a local Openflow controller which uses the LIPSIN forwarding logic to decode forwarding identifiers and install flows.

C. Anatomy of An SDN-based ICN Node

A prototype implementation of the SDN-based Blackadder can be realized on current Openflow-enabled switches that support the specification version 1.0. The Rendezvous information structures and subsequent operations can communicate via the Blackadder service model with the topology manager using the forwarding implemented by an Openflow controller. The topology manager role in this design is to have a deployment overview of the controlled switch datapaths and the interconnection of their uplinks and downlinks and produce LIPSIN [6] identifiers. During bootstrap the topology manager gets from the Openflow controller the port layout of the local switch and assigns local link identifiers. These identifiers are communicated to the local controller via bouncing packets. The Topology manager firstly configures the controller on the switch that hosts it, the immediate neighbour switches using local uplink and so on. Some hardcoded information about the deployment of the switches will have to be provided at this stage. The Topology manager messages may now be transferred to neighbor datapaths because the local controller can install flows and redirect traffic to the uplink of the local switch. This way all switches in a domain are discovered and a network graph is constructed by the topology manager. When a new node gets attached at a port, the local controller notifies the topology manager to associate the node identifier with the respective port.

When a pub/sub rendezvous takes place, the topology manager will be notified and it will produce two forwarding identifiers for two different delivery graphs, namely one for the information delivery from publisher to subscribers and one for the notification delivery from the topology manager to the publisher. Since the current Openflow specification matches fields of certain types of protocols instead of an arbitrary number of bytes, the 14-byte ethernet header can be used for hosting the forwarding identifier. The flows will be therefore identified if we regard Ethernet header fields as parts of the forwarding identifier and match the respective offsets accordingly. For this

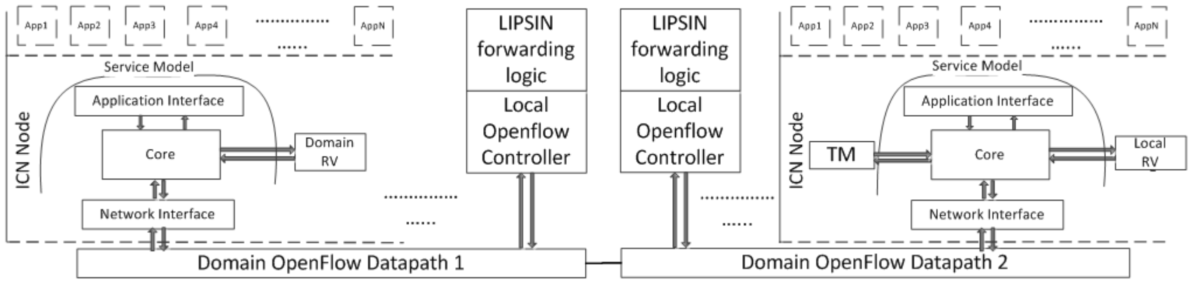

Fig. 2. An ICN network architectural blueprint that uses SDN in forwarding.

approach to work, all the Ethernet interfaces on all the physical network nodes have to be configured in promiscuous mode and capture all arriving traffic. After all, when a packet arrives on a specific port it is sure that is destined to the attached node because of the described system operation. Hence, Ethernet source and destination addresses are no longer relevant.

D. Benefits of the SDN-based node design

The forwarding component that used LIPSIN [6] in the original Blackadder prototype is removed from the node architecture and placed in the Openflow controllers, which control a datapath. The network graph within the topology manager represents datapath interconnections and uses an inverted list to associate terminals with each datapath node in the graph. Moreover LID and port information is distributed among openflow controllers that govern the switch datapaths. Without SDN support, each node in the graph represents an actual terminal and centrally holds all necessary interface information for the forwarding (i.e ethernet addresses). In figure 4 the described differences are depicted. It is evident that scalability is significantly improved. Since the switch datapath in Openflow sends packets that don't match the current flow configuration directly to the controller, some processes are further simplified.

More specifically, during bootstrap when the link identifiers have not yet been disseminated, the topology manager communicates with local Openflow controller on each switch via the bouncing packet mechanism. As a result the network attachment protocol requirements are very simple compared to the ones of the original Blackadder. The topology manager just needs to know the switch where network nodes are attached and it does not have to ask the nodes for that info. The local openflow controller is the only one that needs to know the actual ports, which are traced via a bouncing discovery packet during node attachment. Subsequently the topology manager gets notified of the event via a proper publication as it happens with standard Blackadder. On the other side of the attachment process, the nodes just need to receive from the local Openflow controller the topology forwarding identifiers to the topology manager and the rendezvous nodes without any other link specific information. These are all significant improvements of the system and operations that are described in [17].

Finally, since flows are characterized solely by a forwarding

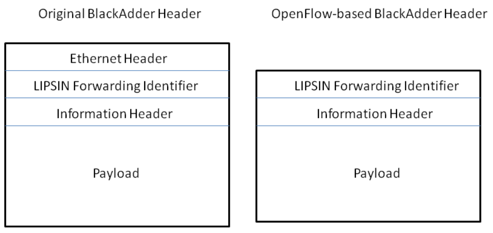
Original BlackAdder Header
OpenFlow-based BlackAdder Header
Ethernet Header
LIPSIN Forwarding Identifier
Information Header
Payload
LIPSIN Forwarding Identifier
Information Header
Payload

Fig. 3. Differences between packet headers.

identifier, no additional headers are required for link specific protocols and therefore the original Blackadder packet headers get reduced. The header difference between original Blackadder, which used Ethernet for forwarding, and the SDN-based Blackadder version is depicted in figure 3.

E. Drawbacks of the SDN-based node design

Nevertheless, there are some drawbacks in the proposed SDN design, which are directly related to implementation limits. Overall scalability of the design may be hindered by the Openflow channel capacity and the Openflow controller ability to respond to incoming forwarding requests. If the controller cannot deal with the requests fast enough, the response of the network will be slow and its performance will decrease as it scales. However, these limitations are addressable to a certain extend and already dealt with in other SDN systems [12]-[13]. We intend to further investigate the Openflow controller performance as soon as we implement the proposed system.

V. Possible Testbed Deployment

Currently, the NITOS [18] Testbed is being used for the development and preliminary testing of the described SDN-based ICN architecture. This testbed has 40 physical high-end nodes wired via ethernet on Pronto 3290 switches that run the latest Openflow Indigo distribution. NITOS is primarily a wireless testbed and the available nodes feature several wireless interfaces, which prevents them from being virtualized. Still 2 openflow switches and 40 nodes are not enough to assess the scalability of the proposed system. For that reason, we plan to use the Ofelia [9] openflow testbed which has enough

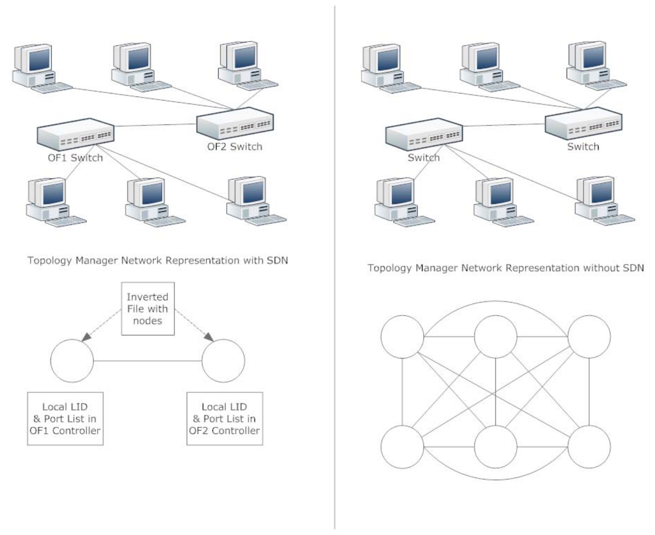
OF1 Switch
OF2 Switch
Topology Manager Network Representation with SDN
Inverted File with nodes
Local LID & Port List in OF1 Controller
Local LID & Port List in OF2 Controller
Switch
Switch
Topology Manager Network Representation without SDN

Fig. 4. Topology Manager Graph Representation with and without SDN support.

switching resources and virtualized terminals to represent a domain.

VI. Conclusions

In this work, we proposed an ICN architectural blueprint that uses SDN support to implement the crucial forwarding function within an architectural context that is currently entirely implemented in software. Crucial here is the notion of a flow, which we aligned with the architectural ICN context by anchoring the notion of forwarding identifiers to this central concept. Based on this, we proposed the replacement of the current forwarding function within our ICN prototype with one that directly utilizes the SDN capabilities to implement forwarding functions based on incoming unknown flow labels. While our work is an early step in the direction of combining SDN and ICN in an efficient way, we are confident that early realizations of our ideas with soon be tested and demonstrated in real-life testbed settings. This availability will drive the debate how SDN can drive the further adoption of ICN deployments in the near future.

Acknowledgment

The research presented in this paper is supported by the EU FP7 project PURSUIT under grant FP7-INFSO-ICT 257217.

References

[1] D. Trossen, M. Sarela, K. Sollins. "Arguments for an Information-Centric Internetworking Architecture" ACM Computer Communication Review, April 2010.
[2] V. Jacobson, D. K. Smetters, J. D. Thornton, M. Plass, N. Briggs, R. Braynard, "Networking named content" Communications of the ACM, Vol. 55, No. 1, 2012.
[3] T. Koponen, M. Chawla, B. Chun, A. Ermolinskiy, K. Kim, S. Shenker, Ion Stoica. "A data-oriented (and beyond) network architecture". SIGCOMM 2007.
[4] D. Trossen (ed) et al., "Final Evaluation Report on Deployment Incentives and Business Models", PSIRP technical deliverable D4.6, available at http://www.psirp.org/publications
[5] D. Trossen, G. Parisis. "Designing and Realizing an Information-Centric Internet" in IEEE Communications Magazine - Special Issue on Information-centric Networks, 2012.
[6] P. Jokela, A. Zahemszky, C. E. Rothenberg, S. Arianfar, P. Nikander. "LIPSIN: line speed publish/ subscribe inter-networking." ACM SIGCOMM 2009.
[7] E. Kohler, R. Morris, B. Chen, J. Jannotti, F. Kaashoek. "The click modular router" ACM Trans. Comput. Syst. 18, 3 (Aug. 2000), 263-297.
[8] The PURSUIT project, "Blackadder Prototype" https://github.com/fp7-pursuit/blackadder.
[9] Ofelia Testbed: http://www.fp7-ofelia.eu
[10] L. Veltri, G. Morabito,S. Salsano,N. Blefari-Melazzi, A Detti., "Supporting Information-Centric Functionality in Software Defined Networks", Workshop on Software Defined Networks 2012.
[11] I. Carvalho, F. Faria, E. Cerqueira, A. Abelem., "ContentFlow: An Introductory Routing Proposal for Content Centric Networks using Openflow" API, 7th Think-Tank Meeting 2012.
[12] OpenVswitch architecture: www.openvswitch.org
[13] Nicira Inc. Network virtualization platform. http://nicira.com/en/network-virtualization-platform
[14] The OpenFlow Protocol: http://www.openflow.org/documents/openflow-wp-latest.pdf
[15] N. Blefari-Melazzi, A. Detti, G. Mazza, G. Morabito, S. Salsano, L. "An OpenFlow-based Testbed for Information Centric Networking", Future Network & Mobile Summit 2012, 4 - 6 July 2012, Berlin, Germany
[16] J. Rajahalme, M. Sarela, K. Visala, J. Riihijarvi, "On name-based inter-domain routing" Computer Networks 55:975-986, 2011.
[17] J. Kjallman (ed.), "First Lifecycle Prototype Implementation", PURSUIT Deliverable D3.2, September 2011 (at http://www.fp7-pursuit.eu).
[18] The NITOS Testbed: http://nitlab.inf.uth.gr