Distributed software defined information centric networking

Rihab Jmal* and Lamia Chaari Fourati

Laboratory of Technologies for Smart Systems (LT2S),
Digital Research Center of Sfax,
B.P. 275, Sfax, Tunisia
and
Sfax University,
3021 Sfax, Tunisia
Email: rihab.jmal1@gmail.com
Email: lamia.chaari@enis.rnu.tn
*Corresponding author

Abstract: Recently, a new trend has emerged based on combining software defined networking (SDN) and information centric networking (ICN) as a promising approach for the future internet. More serious control plane problems related to scalability, fault-tolerance and consistency may affect software defined information centric networking (SD-ICN) compared to traditional SDN environment regarding new augmented features such as content name-based communication and in-network caching. In this paper, we propose a distributed software defined information centric networking (DSD-ICN) that provides ICN features over SDN network with multiple controllers. We addressed in our design the fault-tolerant and strong consistency of the control plane which allows the transparent distribution of the content over different network domains.

Keywords: software defined networking; SDN; information-centric networking; ICN; multiple controllers; inter-domain; distributed.

Reference to this paper should be made as follows: Jmal, R. and Fourati, L.C. (2020) 'Distributed software defined information centric networking', *Int. J. High Performance Computing and Networking*, Vol. 16, No. 1, pp.14–25.

Biographical notes: Rihab Jmal is a Researcher at the Laboratory of Technologies for Smart Systems (LT2S) belonging to the Digital Research Center of Sfax (CRNS), Tunisia. She received her PhD degree in Computer Systems Engineering from Sfax National Engineering School (ENIS) in 2018. She received her BS and MS degrees in Computing and Multimedia from the Higher Institute of Computer and Multimedia of Sfax University in 2011 and 2013, respectively. Her research interests include multimedia, communications and networking which are specially related to video streaming, software defined networking, content centric networks and IoT.

Lamia Chaari Fourati is an Associate Professor at the Higher Institute of Computer and Multimedia of Sfax University. She received her Engineering, PhD and HDR degrees in Telecommunications from Sfax National Engineering School (ENIS) in TUNISIA. She is also a researcher at the Laboratory of Technologies for Smart Systems (LT2S) belonging to the Digital Research Center of Sfax (CRNS). Her research interests include communications, networking and signal processing which are specially related to wireless, e-health and new generation networks.

1 Introduction

The current internet is used in significantly different contexts, forms and sizes from the basic design precepts and original vision. This development of internet usage dominated by content distribution and retrieval introduced new internet architecture named information centric networking (ICN) (Vasilakos et al., 2015).

ICN proposes a paradigm shift from the traditional communication between end hosts to a more sophisticated paradigm centred on data, content and users instead of hosts.

Thus, this novel paradigm poses the focus on the delivery of the desired content to the intended users by decoupling information from its sources. Objects are invoked directly by their name instead of being addressed through their location.

The ICN presents a receiver-driven networking model. It consists of building the features directly into the networking design. When end-users request contents based on name, these interests are managed in the network with consideration of ICN natively including features such as routing by name, location-independent naming, in-network caching, self-secured content, multicast, etc.

Although many works (Vasilakos et al., 2015; Amadeo et al., 2016; Jmal and Fourati, 2017c) have studied ICN in the past few years to propose a concrete deployable solution in reality, it is still in its early stage.

Firstly, the migration from IP to named content is so costly and produces incompatibility issues, especially when using a clean slate architecture design.

Secondly, it is hard to perform routing by name in a large-scale network like the internet. Following ICN principle, the routers broadcast interest packets to its neighbour routers which makes the network traffic uncontrollable on front of the increase of the network size or the requests number. When the size of the network increases, the time to receive requested data will be delayed. Indeed, the effectiveness of network resources is decreased.

Thus, the realisation of an ICN is a big challenge.

The focus of recent research (Jmal and Fourati, 2017a; Gao et al., 2016; Jmal and Fourati, 2019) has been on integrating software defined networking (SDN) with ICN (Jmal and Fourati, 2017b).

In the real world implementation, SDN architecture is a potential paradigm to prove the feasibility of ICN regarding its effective impact in other fields such as energy consumption (Wu et al., 2018; Huang et al., 2016; Fu et al., 2016; Maaloul et al., 2018). SDN provides high programmability of network components which assure the development of new routing and forwarding approaches with the current hardware components facilitating the migration to ICN. In addition, thanks to the virtualisation of the whole network enabled by SDN, the material cost and complexity are reduced while network functionality is augmented. Besides, SDN accelerates the network configurations provisioning by administrators through decoupling the physical network infrastructure and logical elements.

Even though the efficiency of the combination SDN-ICN has been investigated in recent years (Jmal and Fourati, 2017a; Gao et al., 2016; Jmal and Fourati, 2017b), most studies have been proved the feasibility of ICN through SDN architecture and argued the improvements realised regarding to routing, caching, etc. Nonetheless, such hybrid architecture integrating SDN and ICN is not explored in a wide area and in case of multiple domains while evolved control plane issues such as scalability, fault-tolerant and consistency have raised.

With this goal, we propose in this paper a distributed software defined information centric networking (DSD-ICN). In our earlier work (Jmal and Fourati, 2017a), we focused on the intra-domain communication between the controller and ICN nodes.

In this article, we are seeking inter-domain communication between the different controllers managing each network domain carrying ICN packets. Our proposal presents a novel distributed SDN architecture that supports ICN features while guaranteeing fault-tolerant and strong consistency of the control plan.

The remainder of this paper is organised as follows. Section 2 presents a background on ICN/CCN and SDN. We detail the literature in Section 3. In Section 4 we introduce our proposed DSD-ICN while its performance evaluation is in Section 5. Finally, conclusions are drawn in Section 6.

2 Background

2.1 Background on ICN and CCN

ICN shifts the networking paradigm from the current host centric paradigm to a content-centric paradigm. Consequently, a content is named and decoupled from its location. It can be stored in different locations over the network, and each content can be addressed and requested by its name.

Several ICN architectures have been proposed such as named-data networking/content-centric networking (NDN/CCN), publish-subscribe internet routing paradigm (PSIRP), data oriented network architecture (DONA), and network of information (NetInf) (Vasilakos et al., 2015).

Although each architecture is characterised by its specific details, they share numerous fundamental properties such as name-based routing, the unicity of content name, in-network caching, and content integrity.

PSIRP routing scheme involves four main components: rendezvous nodes, topology nodes, branching nodes and forwarding nodes.

In DONA, the resolution handles are leveraged to allow requested content discovery, and NetInf routing is achieved by a multi-level DHT mechanism.

Content router is implemented by CCN to realise their longest prefix matching routing scheme. The in-network caching feature is integrated into CCN as a content router function while the other projects are generally depended on dedicated modules like rendezvous nodes in PSIRP and storage engine in NetInf.

CCN is built around three data structures: PIT, FIB and CS.

The client request is forwarded in an interest packet through CCN nodes. Each traversed CCN node checks its CS in order to send a data packet as a response in case the content is stored. Otherwise, the PIT is checked. In case there is a PIT entry for a previously requested content chunk, the interest is aggregated in this entry. Alternatively, the CCN node checks its FIB and while an outgoing interface for this interest exists in the FIB, a new PIT entry is made and it forwards the interest. In the worst cases, the interest is simply rejected.

If the intermediate nodes do not have a copy of the requested content, the original provider forwards the appropriate data packet and the traversed nodes store a copy for future requests (Jmal and Fourati, 2017c).

2.2 Background on SDN

The SDN's main feature consists of separating control and data planes which provides a network-wide abstraction.

The decoupled control plane instructs the forwarding devices, which are commonly OpenFlow switches, via the OpenFlow protocol.

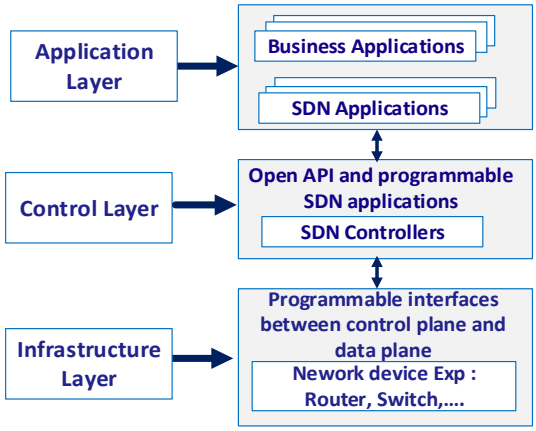
An OpenFlow switch is composed of:

- flow table indicating to the switch how processing each flow entry by associating different actions
- secure channel ensuring safe passage of transmission rules between OpenFlow controller and OpenFlow switch.

The architecture design of SDN is composed of three functional layers as illustrated in Figure 1.

- *the infrastructure layer (the data plane):* corresponds physically to the network devices
- *the control layer:* incorporates an SDN controller that serves as the key of the whole architecture
- *the applications layer:* communicates with the SDN controllers via open and programmable API.

Figure 1 SDN layered architecture (see online version for colours)



Source: Jmal and Fourati (2017b)

2.3 Definition of an SDN domain

An SDN domain presents a part of a network determined by the network operator. A domain is controlled by an SDN controller. It can cover multiple network operation systems (NOS) and communicate directly with certain SDN-enabled devices. Each NOS typically consists of many interconnected devices that are compatible with SDN. The SDN controller aggregates the network topology views from multiple NOSs and maintains a global view of the networks covered by the domain. The controller is responsible for forwarding the application requests to the corresponding NOS. Two SDN domains are adjacent if there is a physical link between the two underlying networks.

Within each SDN domain, the appropriate controller can set domain-specific policies on importing information from devices, aggregating, and exporting to external entities. These policies may not be created publicly; as a result, controllers in other domains may not be aware of such policies for a given SDN domain.

The network operator that creates the SDN domains aims to provide a flexible network administration. The operator decides to divide the entire network into SDN domains depending on the scale of the underlying network. For some small-scale data centres, only one SDN domain can be sufficient.

For a service provider with a large transport network, it is best to divide the network into SDN domains because the centralised control with a single controller will create a bottleneck. For example, the operator can divide his network into different SDN domains based on physical locations. He can rent such a part of his network to the local content provider, etc. Such a deployment scenario requires an SDN controller that provides powerful network service capability to applications. In addition, to define domains and interconnections between them involves more than just simple connections between SDN boxes; there are various aspects to consider, such as how their network topologies connect, what enclosures the neighbouring controllers face and how to get their addresses, what rights and policies control the conversation, and so on. Other aspects to consider as those allowed to deploy 'programs' on the SDN infrastructure, what actions can a 'program' perform depending on who has deployed them: the SDN network manager is likely to control the deployment of 'programs' on the SDN infrastructure.

The focus is then on how deployed programs can affect other domains and what mechanism we want to use to communicate this effect to other domains.

3 Related work

In order to provide a deep description about the state of the art, we have been looking into distributed SDN architectures. We founded that this is already a recent research field in progress and there is no standardisation until now, to the best of our knowledge.

The distributed SDN architectures are managed through two main approaches as shown in Figure 2:

1 logically centralised

2 logically distributed.

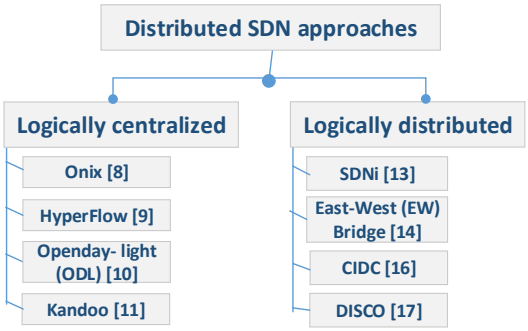
3.1 Logically centralised approach

This approach is characterised by a set of controllers with the same view of the network and that share the same database. Among the most known, we cite Onix (Koponen et al., 2010), HyperFlow (Tootoonchian and Ganjali, 2010), OpenDay-light (ODL) (Medved et al., 2014) and Kandoo (Yeganeh and Ganjali, 2012).

Onix (Koponen et al., 2010) uses the distributed hash table (DHT) to store information about the distributed network. HyperFlow (Tootoonchian and Ganjali, 2010)

builds a global network view through the distributed file system WheelFS (Stribling et al., 2009) and each controller is in charge of its network. Link status changes and other events affecting the network view are synchronised between the controllers.

Figure 2 Classification of distributed SDN architectures (see online version for colours)



Although HyperFlow and Onix allow the distribution of the SDN control plane, they produce a lack of flexibility and scalability limitation.

ODL (Medved et al., 2014) builds the data structure trees which consist on the configuration tree and the operational tree. The desired state of the system is stored in the configuration tree while the current runtime status is provided by the operational tree. To support multiple controllers, the ODL forms the cluster after building the trees. However, the cluster with its operations require more resources, bandwidth and time which could decrease the performance of the network.

Kandoo (Yeganeh and Ganjali, 2012) proposes a hierarchical framework based on two layers of controllers:

- *The top layer:* is a logically centralised controller that preserves the state of the whole network. It is a root controller that runs non-local applications.
- *The bottom layer:* presents a set of controllers which are not interconnected and without knowledge of the network-wide state. These local controllers execute local applications as near as possible to switches.

Generally, the logically centralised approach offers fault-tolerance, elasticity and decentralisation which it is not feasible for logically distributed architecture where each domain is managed by its controller having its own database. For this purpose, our DSD-ICN architecture fellow the logically centralised approach.

3.2 Logically distributed approach

This approach is more suitable for networks that are vastly distributed over multiple domains. Each domain is managed by its controller. Different controllers communicate each other to share only some useful information enabling some services like the topology view. In other words, the main idea is to build an 'east-west (EW)' communication between SDN controllers, as an analogy to OpenFlow being a 'north-south' protocol between NOS and network devices.

Yin et al. (2012) proposed a protocol named 'SDNi' enabling the coordination of behaviours between different SDN controllers and allowing to exchange control information related to multiple SDN domains.

EW bridge (Lin et al., 2015) is proposed as a design that support different controllers with various local network view storage systems. The publish/subscribe model (Tarkoma, 2012) is used to synchronise data between different controllers. Eventually, the publish/subscribe model attempts multicast or group messaging problem. This model is used for SDN regarding to its scalability compared to the client-server model. Nevertheless, the scalability is still a research challenge especially under high load. Recently, another EW interface is proposed in Benamrane et al. (2017) called communication interface for distributed control (CIDC) plane. This interface provides communication modes (notifications, messages, services, etc.) to be exchanged between different controllers. Distributed services (firewall, load balancer and SSL) are supported thanks to a mechanism based on policy sharing.

Distributed SDN controllers (DISCO) (Phemius et al., 2014) proposal supports a logically centralised controller for the intra-domain operation, as well as a logically distributed controller to achieve inter-domain visibility.

The main drawback of the logically distributed approach is represented by its weak consistency on semantics; eventually, all controller nodes are informed by the data updates on different nodes. Thus, there is a delay taken by distinct nodes to read new or old values for the same property. Otherwise, strong consistency ensures that all controller nodes have the access to the most updated property value in the same time (Kreutz et al., 2015).

Works on ICN in distributed areas are so limited. Gao et al. (2016) proposed an intra-domain communication in software defined information centric networking (SD-ICN) and argued that their solution is compatible with CoLoR (Luo et al., 2014) which presents a general architecture design for ICN inter-domain routing.

Our proposed DSD-ICN follows the logically centralised approach in term of centralising the treatment of control plane operating. We focalise our work on consistency and fault-tolerance of the control plan while managing ICN contents. It provides strong consistency through its architecture based on inter-domain information base (IIB) system which allows multiple controllers to coordinate their actions.

Consequently, we studied recently proposed approaches focusing on SDN consistency and fault-tolerance.

Schiff et al. (2016) proposed a synchronisation framework for control planes built around atomic transactions realised in-band on the switches of the data plane. The data-plane switch configuration space is used in this approach as a transactional shared memory. An additional information about conflicts between the controllers can be stored in this memory. Thus, a transaction is designated to include standard control, update operations and synchronisation primitives functioning on this shared memory.

Botelho et al. (2016) provide a modular architecture supporting a fault-tolerant data store that allows a transparent distribution of the control plane through the strong consistency properties. Each controller is responsible for managing a network domain and the coordination between different controllers is achieved thanks to the replicated, fault tolerant data store.

The main drawback of this solution is presented by the performance overhead and the limited scalability.

Beehive (Yeganeh and Ganjali, 2016) allows the calculation of offloading that depends on the state of any application connected to any controller belonging to the set of distributed controllers. The state of the application in beehive is stored as key-value pairs in a shared distributed data store. The calculations in the application are mapped to the corresponding replication controller based on an application-specific mapping function. This function queries a globally synchronised dictionary and maps the calculation tasks to the appropriate location (where the data resides). Indeed, the computation process is executed at different controllers dynamically by moving around the state stored in a distributed data store. However, dynamic state placement results overheads such as running a consensus algorithm between controllers to determine the location of the state, which causes expensive synchronisation and affects performance.

In addition, Hydra (Chang et al., 2016) presents a framework for distributed SDN controllers based on functional slicing that presents a complementary approach to scaling. Thus, several SDN applications belonging to the same topological partition can be placed in physically separate servers. In Hydra, the choice of partitions is based on the convergence time as the main metric. Application instances are assigned to partitions in order to minimise response times while taking into consideration the communication between applications of a partition and instances of an application over partitions. Nevertheless, Hydra increases latency when critical paths cover multiple servers.

Therefore, we can deduce the importance of a shared memory between the different distributed SDN controllers. In addition, the communication between the different partitions, to execute the SDN applications, can generate overloads that affect the system performances.

We summarise the related works in Table 1 where we highlight the difference between our proposal DSD-ICN and other cited approaches. DSD-ICN is distinguished from the others by the fact that is the only approach that consider a logically centralised information management, provides consistency and fault-tolerance in addition to the support of ICN.

4 DSD-ICN proposal

DSD-ICN allows distributing ICN content in a large-scale supervised network which is benefic to many relevant applications such as mobile edge computing (Jararweh et al., 2017) and cloud computing (Li et al., 2018; Gupta et al., 2017) where the access to the requested information is facilitated even in large networks where the requested information is located in a different side. Besides, with DSD-ICN the network security becomes scalable. Security can scale as software scales and as new clouds, workloads and network segments are provisioned. Thus, our proposal provides a promising environment to implement security applications (Jiang et al., 2018; Deka, 2015; Gupta, 2018) and improve the user experience (Gupta et al., 2017; Gupta and Gupta, 2015).

Table 1 A summary of related works

Approach and date of publication	*Logically centralised*	*Logically distributed*	*ICN supported*	*Consistency and fault-tolerance*
Koponen et al. (2010)	✓	×	×	×
Tootoonchian and Ganjali (2010)	✓	×	×	×
Medved et al. (2014)	✓	×	×	×
Yeganeh and Ganjali (2012)	✓	×	×	×
Yin et al. (2012)	×	✓	×	×
Lin et al. (2015)	×	✓	×	×
Benamrane et al. (2017)	×	✓	×	×
Phemius et al. (2014)	×	✓	×	×
Gao et al. (2016)	×	✓	✓	×
Schiff et al. (2016)	×	✓	×	✓
Botelho et al. (2016)	✓	×	×	✓
Yeganeh and Ganjali (2016)	✓	×	×	✓
Chang et al. (2016)	×	✓	×	✓
DSD-ICN (our proposal)	✓	×	✓	✓

DSD-ICN is performing two main functionalities. The first one concerns providing ICN features over SDN architecture while the second functionality is based on inter-domain communication. We follow an overlay architecture based on content centric networking (CCN) (Jacobson et al., 2009) implementation with the existing OpenFlow switch.

When a CCN client requests a content by name, the request is received by a CCN node which checks its content store (CS), if the requested content is existent a data packet is forwarded. In the other case, the content name is mapped to an IP address in order to be processed by the OpenFlow switch. The configuration of CCN implementation on top of OpenFlow switch is applied through an adaptation layer which perform hashing and mapping of the content name into IP addresses. Then, the request is forwarded to the controller, the responsible of the management of the whole software defined information centric network. The controller is content-aware, it generates rules on ICN nodes to optimise on content routing regarding to its broad view of the network. Finally, the data packet is forwarded from the content provider through the optimised path following the controller rules.

The optimised path **P** is considered as the shortest path with the higher available bandwidth. For this purpose two equations are verified (Jmal and Fourati, 2019):

$$P = Min\left(\sum_{i=0}^{N} l_k\right)$$

$$P_{LC} = Max\left(\sum_{i=1}^{k-1} abw_{li}\right)$$

The parameters used in these equations are described in Table 2.

As above, we described the intra-domain operation of our proposed design combining SDN and CCN as shown in Figure 3.

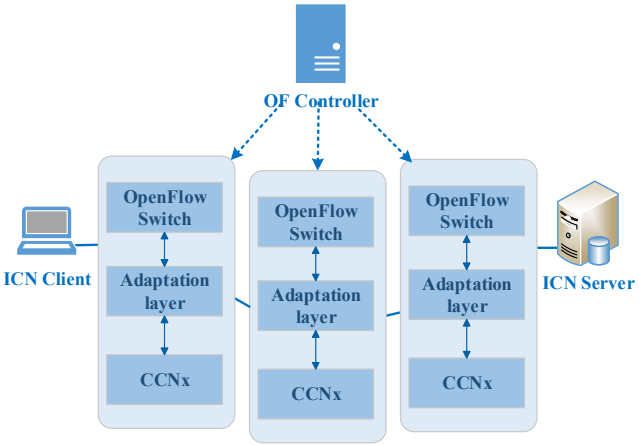
However, what would happen in case of controller failure?

For this purpose, the second process addressed in our design is related to inter-domain communication and multiple SDN controllers. We aim to overcome one point failure issue as well as provide strong consistency of the control plane.

Table 2 Optimised path parameters

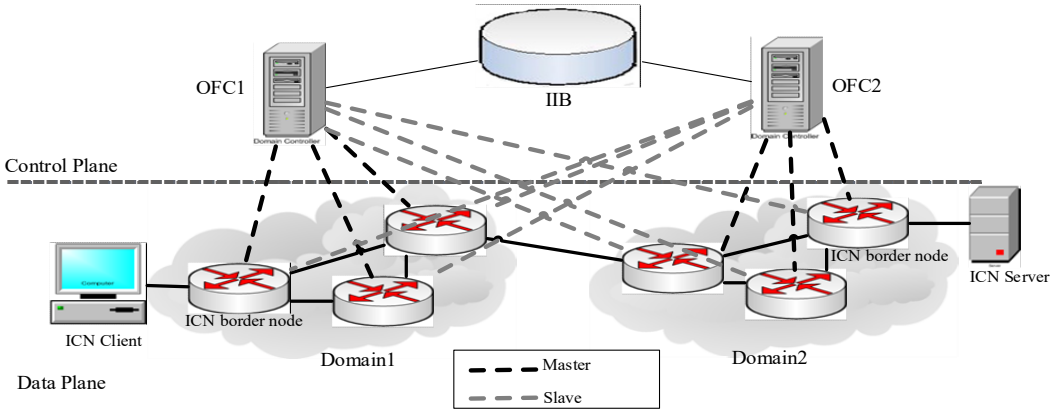
Parameter	*Designation*
P	The optimised path
N	The network nodes
L	Links between different nodes
k	The length of the path
$l = (n, m)$	Link between an origin node n and a destination m
abw_l	Available bandwidth for a link l
P_{lc}	The links capacity of the path P

Figure 3 Intra-domain design (see online version for colours)



To achieve our goal, we exploited the master-slave configuration defined in OpenFlow 1.3 (Pfaff et al., 2012), which allows switches to tolerate controller crashes thanks to its ability of connection to more than one controller. In our design, the primary controller of one domain presents the backup controller of another domain to provide controller fault tolerance as depicted in Figure 4.

Figure 4 Inter-domain communication architecture (see online version for colours)



The synchronisation between different controllers is performed thanks to the shared IIB. The controller decisions are taken by the control plane on the basis of data plane events while the consistent network state is performed through the IIB due to the communicated information such as read and write operations.

The controller is aware of content locations. When an ICN server publish a new content, the controller is informed about its name '$Name_i$' and the IIB is updated to share that the content with $Name_i$ is existent in domain D_j for future requests on this content from other domains. We noted the set of domains D and controllers C as follow:

$$D = \{D_j,\ j = 1, 2, 3, ..., n\}$$

$$C = \{C_j,\ j = 1, 2, 3, ..., n\}$$

Algorithm 1 Request dissemination between several controllers

1. **if** (local lookup process (D_j, $Name_i$)= true)
   // the requested content exists in the local domain
2. Install rules () // the data packet is sent locally
3. **else** // the requested content does not exist in the local domain
4. C_j insert Interest entry in IIB <C_j, D_j, $Name_i$, ID_h>
5. **While** (query= false)
6. C_k = Find the best controller to ask ()
7. Send query
8. **if** (query= true)
9. local lookup process (D_k, $Name_i$)
10. Install rules ()
11. **if** (content ($Name_i$)= true)
    // the content is sent to the client
12. Update IIB () // to indicate that the request is satisfied

ID_h is the user device identifier used by the controller to handle the packet forwarding. The flow chart describing the operational functioning of requesting a content is illustrated by Figure 5.

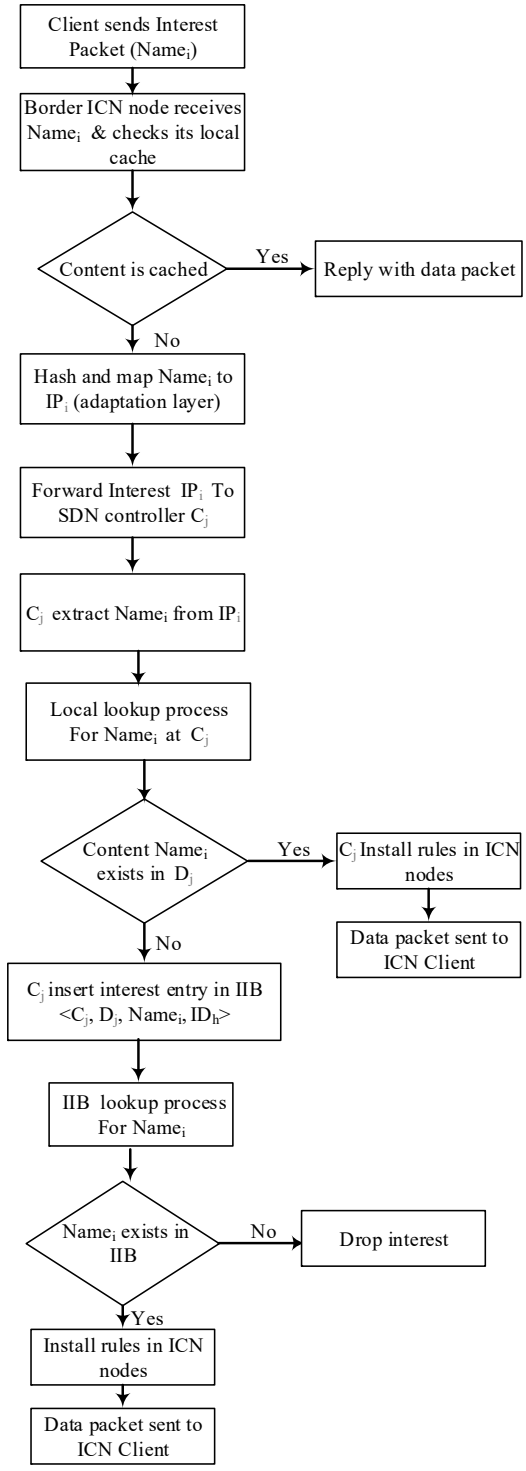
The IIB stays tuned. When a new controller is connected, the IIB is updated by the new controller and its domain. When a controller is disconnecting, a backup controller takes its role. The information exchanged between the different controllers and the IIB is sent through JSON objects in order to optimise the recovery time.

In fact, the signalling messages include the controllers reachability update, the network operation update such as QoS and available software capabilities in the domain, as well as content publication and request.

We propose Algorithm 1 for the request dissemination between several controllers in order to find the requested content.

In an environment that contains multiple controllers, when the primary controller fails what is the backup controller that will take its place?

Figure 5 Operational functioning of requesting a content



In this context, we execute a leader (or master) election process that is executed when a communication is established between the different controllers. This process allows early discovery of a controller fail, as well as the network scheduling in a distributed system with multiple controllers based on the roles assigned to the controllers. Consequently, it allows the creation of a consistent fault-tolerant system and is able to recover the state of a failed controller and restore its state.

Indeed, a group of distributed controllers, which is responsible for the same domain, executes an election algorithm to determine a new leader controller.

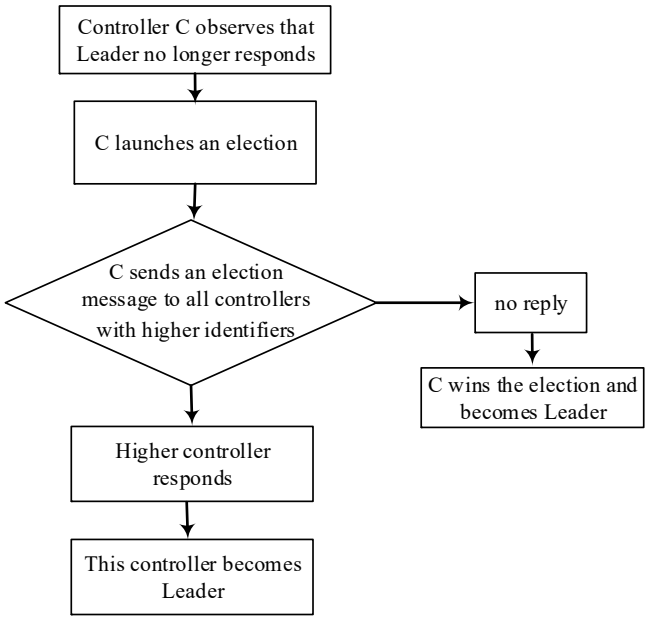
This algorithm assumes that each active controller has a unique priority identifier. At any time, a controller may receive an election message from one of his lower colleagues.

The receiver sends an 'OK' message to the sender and proceeds to his election process.

Finally, only the worthy controller remains. This controller announces his victory to all distributed controllers in the group.

The election principle executed, described in Figure 6, is inspired by the Bully's algorithm (Stoller, 1997).

Figure 6 Flowchart of the election algorithm



5 Performance evaluation

We evaluated the DSD-ICN architecture in Mininet (http://mininet.org/) environment, the well-known SDN emulator. Mininet facilitates the realisation of tests similar to real experiments. We created three domains, each one is composed of three connected switches handled by one master controller and configured as slave to the other controllers. In case of controller crash, the switches are reconfigured automatically to be handled by another master controller determined thanks to the election algorithm executed at the active distributed controllers. We used the Openvswitch 2.7 that supports OpenFlow 1.3 and above. We implemented a new module on the floodlight (FL) (http://www.projectfloodlight.org/floodlight/) controller to handle ICN networking and content names as well as another module for the election process.

Moreover, we used the 'sync service' offered by FL to achieve the synchronisation between different controllers and to apply the IIB functioning. We ran an ICN client on a host attached to domain 1 and an ICN server attached to domain 3. We configured on top of OpenFlow switches CCNx (Smetters et al., 2010), the official implementation of CCN. The combination is enabled thanks to the proposed adaptation layer (Jmal and Fourati, 2017a). In our scenario, the content is published at the server node by *ccnputfile* command while the client node downloads the content through *ccngetfile* testing tool. We repeated tests for different files with different sizes as depicted in the following figures. Figure 7 illustrates the upload time of each file from the server statistics while Figure 8 shows the download time from the client statistics.

Figure 7 Upload time at the server (see online version for colours)

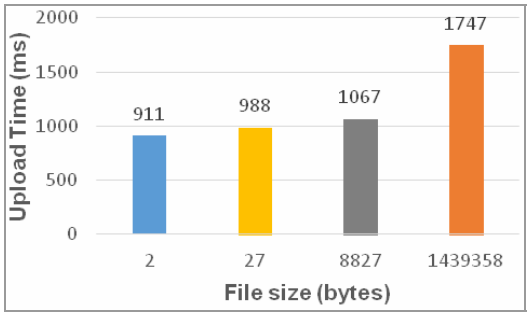
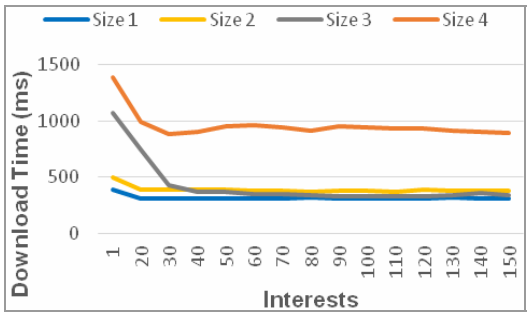


Figure 8 Download time at the client (see online version for colours)



For the first interests sent, the download time was higher than the next requests on the content. Hence, thanks to in-network caching feature supported by DSD-ICN, the content is served from the closest node to the client which reduces the download time.

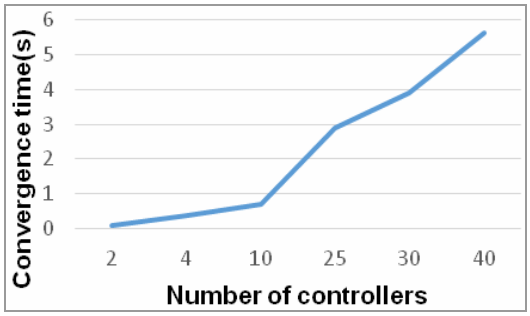
Consequently, there is no variation in the download time after requesting the first interests since it is served from the border nodes. Even without caching the download time is considered interesting since our FL controllers with the IIB guarantee the lookup latency and accelerate the content location resolution. Our architecture allows to a content, published from a domain, to be requested from other domains and served with an optimised download time.

In order to analyse the performance of the control plane in a distributed architecture, we consider synchronisation metrics such as latency and inter-controller communication overhead.

The convergence time is considered as the time required for the controllers to establish a communication between themselves to be aware of the connected controllers and to

determine the leader (master). It is illustrated in terms of increased number of controllers in Figure 9.

Figure 9 The convergence time according to the number of controllers (see online version for colours)



Subsequently, we evaluate our architecture considering different types of SDN applications. For this, we conduct tests on a topology based on five controllers and nine switches as described in Table 3.

Table 3 Details on the topology used

Controllers	*Domains*	*Roles*
C1	SW1, SW2	Master
	SW3, SW4, SW5, SW6, SW7, SW8, SW9	Slave
C2	SW3, SW4	Master
	SW1, SW2, SW5, SW6, SW7, SW8, SW9	Slave
C3	SW5, SW6, SW7	Master
	SW1, SW2, SW3, SW4, SW8, SW9	Slave
C4	SW8, SW9	Master
	SW1, SW2, SW3, SW4, SW5, SW6, SW7	Slave
C5	SW1, SW2, SW3, SW4, SW5, SW6, SW7, SW8, SW9	Slave

Figure 10 Throughput for firewall (see online version for colours)

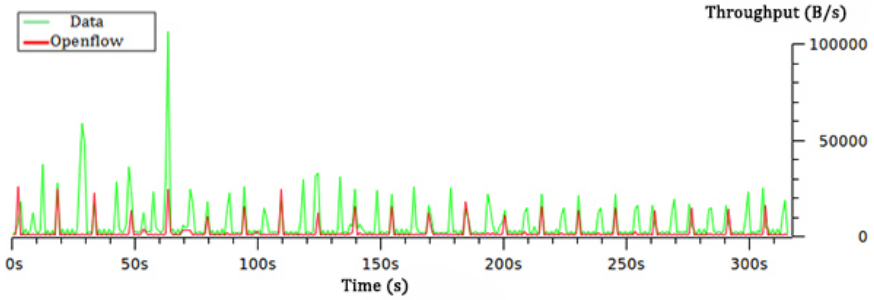


Figure 11 Throughput for learning switch (see online version for colours)

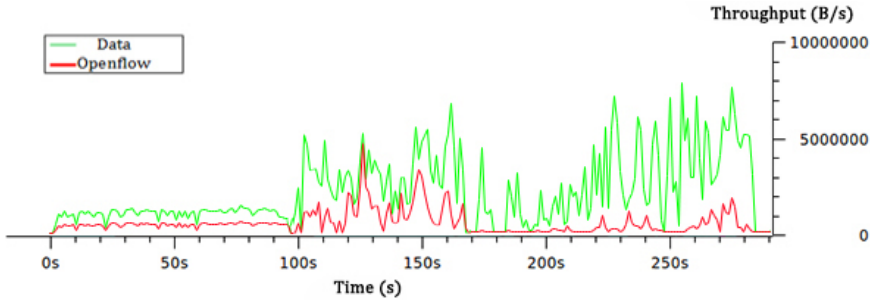
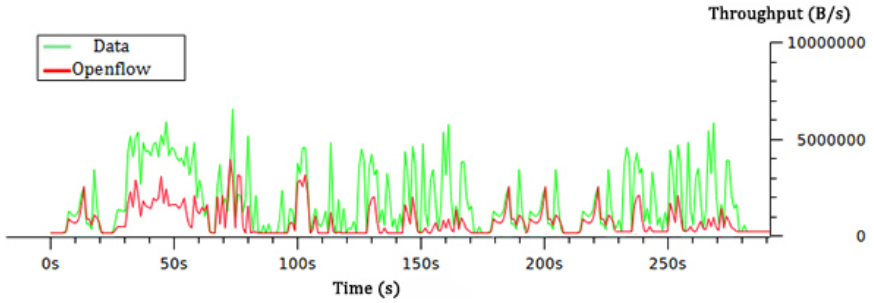


Figure 12 Throughput for load balancer (see online version for colours)



The highlighted SDN applications and functions are:

- *Firewall:* a latency-sensitive application that filters messages sent to the controller (packet-in) based on a set of rules.
- *Learning switch:* this application emulates the process of a layer 2 switch forwarding based on a switch table that associates MAC addresses with switch ports. The switch is able to generate this table by listening to each incoming packet which, in turn, is transmitted according to the existing information in the table.
- *Load balancer:* this application uses a round-robin algorithm to distribute requests addressed to a virtual IP (VIP) address on a set of servers.

The different analyses of these applications and functions are described as follow. Figure 10, Figure 11 and Figure 12 show the throughput over the time at the control plane when different OpenFlow and CCN flows are generated in the data plane. We captured this traffic for firewall, learning switch, and load balancing using Wireshark (Wireshark Analyzer, http://www.wireshark.org).

When the firewall is executed, the OpenFlow packets are 16.26% and the data are 50% of the packets. For the learning switch, OpenFlow packets are 22.3% and the data exceed 60% of the packets. For the load balancing, the OpenFlow packets are 15% and the data is 51%.

Figure 13 Comparison of the average delay between DSD-ICN and ODL (see online version for colours)

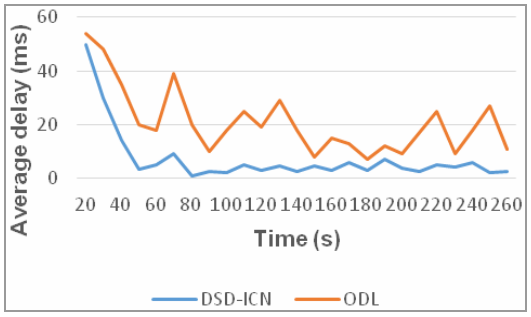
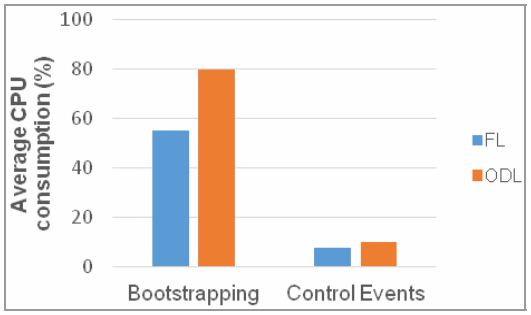


Figure 14 Comparison of the CPU consumption (see online version for colours)



We can deduce that the overhead of OpenFlow packets is acceptable over DSD-ICN architecture regarding the amount of the data forwarded.

The importance and the effectiveness of DSD-ICN are represented by reducing the convergence time while maintaining a significant throughput as well as ensuring responsiveness to latency-sensitive applications.

Afterwards, we conducted tests under the same conditions in order to compare our FL controllers to the OpenDay-light (ODL) (Medved et al., 2014) in term of delay and CPU consumption. The average delay between DSD-ICN and ODL is depicted in Figure 13 while the average results in percent of the CPU consumption are described in Figure 14 for the 'bootstrapping' phase as well as for the 'control events' phase. The 'bootstrapping' presents the phase where the controllers start and discover their neighbours. In this step, ODL controllers form a cluster after the start and share their database, while FL controllers use the 'sync service'. In case of the 'control events' phase the controller is on production (i.e., insert rules).

Results show that our extended FL outperforms ODL.

6 Conclusions

In this paper, we have proposed and evaluated DSD-ICN, an ICN network over SDN with multiple controllers. We tried to keep relevant network with a consistent network view that handles content on the basis of its name even over different domains. Our proposal provides strong consistency and fault-tolerance of the control plane. The main components of the proposed architecture are:

1 ICN nodes composed of CCNx configured on top of OpenFlow switches with the adaptation layer providing the conformity between both paradigms. This method is characterised by simplicity and flexibility as it does not imposes extensions on either CCNx or OpenFlow.

2 SDN controller handles ICN packets and provides optimised routing thanks to centralised intelligence.

3 IIB enable synchronisation between different controllers.

The shared aspect offers high speed data transmission and intelligent data processing. For future work, we plan to test DSD-ICN in larger networks as well as analysing load balancing and multipath selection features.

References

Amadeo, M., Campolo, C. and Molinaro, A. (2016) 'Information-centric networking for connected vehicles: a survey and future perspectives', *IEEE Communications Magazine*, Vol. 54, No. 2, pp.98–104.

Benamrane, F., Ben Mamoun, M. and Benaini, R. (2017) 'An east-west interface for distributed SDN control plane: implementation and evaluation', *Computers & Electrical Engineering*, January, Vol. 57, pp.162–175.

Botelho, F., Ribeiro, T.A., Ferreira, P., Ramos, F.M. and Bessani, A. (2016) 'Design and implementation of a consistent data store for a distributed SDN control plane', in *2016 12th European Dependable Computing Conference (EDCC)*, pp.169–180.

Chang, Y., Rezaei, A., Vamanan, B., Hasan, J., Rao, S. and Vijaykumar, T.N. (2016) *Hydra: Leveraging Functional Slicing for Efficient Distributed SDN Controllers*, arXiv preprint arXiv: 1609.07192.

Deka, G.C. (2015) 'BDS: browser dependent XSS sanitizer', *Handbook of Research on Securing Cloud-Based Databases with Biometric Applications*, pp.174–191, IGI Global.

Fu, S.H., Wen, H., Wu, J. and Wu, B. (2016) 'Cross-networks energy efficiency tradeoff: from wired networks to wireless networks', *IEEE Access*, June, Vol. 5, pp.15–26.

Gao, S., Zeng, Y., Luo, H. and Zhang, H. (2016) 'Scalable control plane for intra-domain communication in software defined information centric networking', *Future Generation Computer Systems*, March, Vol. 56, pp.110–120.

Gupta, B.B. (2018) *Computer and Cyber Security: Principles, Algorithm, Applications, and Perspectives*, p.666, CRC Press, Taylor & Francis.

Gupta, B.B., Gupta, S. and Chaudhary, P. (2017) 'Enhancing the browser-side context-aware sanitization of suspicious HTML5 code for halting the DOM-based XSS vulnerabilities in cloud', *International Journal of Cloud Applications and Computing (IJCAC)*, Vol. 7, No. 1, pp.1–31.

Gupta, S. and Gupta, B.B. (2015) 'PHP-sensor: a prototype method to discover workflow violation and XSS vulnerabilities in PHP web applications', in *Proceedings of the 12th ACM International Conference on Computing Frontiers*, ACM, May, p.59.

Huang, H., Guo, S., Wu, J. and Li, J. (2016) 'Green DataPath for TCAM-based software-defined networks', *IEEE Communications Magazine*, November, Vol. 54, No. 11, pp.194–201.

Jacobson, V., Smetters, D.K., Thornton, J.D., Plass, M.F., Briggs, N.H. and Braynard, R.L. (2009) 'Networking named content', *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies*, pp.1–12.

Jararweh, Y. et al. (2017) 'Software-defined system support for enabling ubiquitous mobile edge computing', *The Computer Journal*, Vol. 60, No. 10, pp.1443–1457.

Jiang, F., Fu, Y., Gupta, B.B., Lou, F., Rho, S., Meng, F. and Tian, Z. (2018) 'Deep learning based multi-channel intelligent attack detection for data security', *IEEE Transactions on Sustainable Computing*, 1 April–June, Vol. 5, No. 2, pp.204–212, doi: 10.1109/TSUSC.2018.2793284.

Jmal, R. and Fourati, L.C. (2017a) 'An OpenFlow architecture for managing content-centric-network (OFAM-CCN) based on popularity caching strategy', *Computer Standards & Interfaces*, Vol. 51, pp.22–29.

Jmal, R. and Fourati, L.C. (2017b) 'Content-centric networking management based-on software defined networks: survey', *IEEE Transactions on Network and Service Management*, December, Vol. 14, No. 4, pp.1128–1142, doi: 10.1109/TNSM.2017.2758681.

Jmal, R. and Fourati, L.C. (2017c) 'Emerging applications for future internet approach based-on SDN and ICN', in *2017 IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA)*, Hammamet, pp.208–213.

Jmal, R. and Fourati, L.C. (2019) 'Assisted DASH-aware networking over SDN-CCN architecture', *Photonic Network Communications*, Vol. 38, No. 1, pp.37–50.

Koponen, T., Casado, M., Gude, N., Stribling, J., Poutievski, L., Zhu, M. et al. (2010) 'Onix: a distributed control platform for large-scale production networks', in *Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation OSDI10*, pp.1–6.

Kreutz, D., Ramos, F.M.V., Verissimo, P.E., Rothenberg, C.E., Azodolmolky, S. and Uhlig, S. (2015) 'Software-defined networking: a comprehensive survey', *Proceedings of the IEEE*, January, Vol. 103, No. 1, pp.14–76.

Li, T., Gupta, B.B. and Metere, R. (2018) 'Socially-conforming cooperative computation in cloud networks', *Journal of Parallel and Distributed Computing*, Vol. 117, pp.274–280.

Lin, P. et al. (2015) 'A west-east bridge based SDN inter-domain testbed', *IEEE Communications Magazine*, Vol. 53, No. 2, pp.190–197.

Luo, H., Chen, Z., Cui, J., Zhang, H., Zukerman, M. and Qiao, C. (2014) 'CoLoR: an information-centric internet architecture for innovations', *IEEE Network*, Vol. 28, No. 3, pp.4–10.

Maaloul, R., Taktak, R., Chaari, L. and Cousin, B. (2018) 'Energy-aware routing in carrier-grade ethernet using SDN approach', *IEEE Transactions on Green Communications and Networking*, Vol. 2, No. 3, pp.844–858.

Medved, J., Varga, R., Tkacik, A. and Gray, K. (2014) 'OpenDaylight: towards a model-driven SDN controller architecture', in *IEEE 15th International Symposium on World of Wireless, Mobile and Multimedia Networks WoWMoM*, pp.1–6.

Pfaff, B., Lantz, B., and Heller, B. (2012) *Openflow Switch Specification, Version 1.3. 0*, pp.39–46, Open Networking Foundation.

Phemius, K., Bouet, M. and Leguay, J. (2014) 'Disco: distributed multi-domain SDN controllers', in *2014 IEEE Network Operations and Management Symposium (NOMS)*, pp.1–4.

Schiff, L., Schmid, S. and Kuznetsov, P. (2016) 'In-band synchronization for distributed SDN control planes', *ACM SIGCOMM Computer Communication Review*, Vol. 46, No. 1, pp.37–43.

Smetters, D., Golle, P. and Thornton, J. (2010) *CCNx Access Control Specifications*, Technical Report, PARC, Tech. Rep.

Stoller, S.D. (1997) *Leader Election in Distributed Systems with Crash Failures*, Technical Report, Indiana University, April, Vol. 169.

Stribling, J., Sovran, Y., Zhang, I., Pretzer, X., Li, J., Kaashoek, M.F. et al. (2009) 'Flexible, wide-area storage for distributed systems with WheelFS', in *Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation NSDI09*, pp.43–58.

Tarkoma, S. (2012) *Publish/Subscribe Systems: Design and Principles*, John Wiley & Sons.

Tootoonchian, A. and Ganjali, Y. (2010) 'HyperFlow: a distributed control plane for OpenFlow', in *Proceedings of the Internet Network Management Conference on Research on Enterprise Networking*, p.3.

Vasilakos, A.V., Li, Z., Simon, G. and You, W. (2015) 'Information centric network: research challenges and opportunities', *Journal of Network and Computer Applications*, June, Vol. 52, pp.1–10.

Wireshark Analyzer [online] http://www.wireshark.org.

Wu, J. et al. (2018) 'Information and communications technologies for sustainable development goals: state-of-the-art, needs and perspectives', *IEEE Communications Surveys & Tutorials*, March, Vol. 20, No. 3, pp.2389–2406.

Yeganeh, S.H. and Ganjali, Y. (2012) 'Kandoo: a framework for efficient and scalable offloading of control applications', in *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, pp.19–24.

Yeganeh, S.H. and Ganjali, Y. (2016) 'Beehive: simple distributed programming in software-defined networks', in *Proceedings of the Symposium on SDN Research*, ACM, pp.1–12.

Yin, H., Xie, H., Tsou, T., Lopez, D., Aranda, P. and Sidi, R. (2012) *SDNi: a Message Exchange Protocol for Software Defined Networks (SDNs) across Multiple Domains*, IETF draft, work in progress.

Websites

http://mininet.org/ (accessed 2017).

http://www.projectfloodlight.org/floodlight/ (accessed 2017).