SD-ICN: Toward Wide Area Deployable Software Defined Information Centric Networking

Changyou Xing, Ke Ding, Chao Hu, Ming Chen, and Bo Xu
Collage of Command Information System
PLA University of Science and Technology
Nanjing, China
[e-mail: changyouxing@126.com]
*Corresponding author: Changyou Xing

Received August 24, 2015; revised February 14, 2016; accepted March 26, 2016; published May 31, 2016

Abstract

Information Centric Networking that uses content name instead of IP address as routing identifier can handle challenges such as traffic explosion and user mobility, but it also suffers from scalability and incompatibility problems. In this paper by combining the concept of software defined networking and Internet end to end arguments, we propose a wide area deployable software defined information centric networking service model named SD-ICN. SD-ICN employs a dual space structure that separates edge service network and core transmission network. The enhanced SDN techniques are used in edge service network in order to implement intelligent data routing and caching, while traditional IP technique is reserved in core transmission network so as to provide wide area high speed data transmission. Besides, a distributed name resolution system based on the cooperation of different controllers is also presented. The prototype experiments in our campus network show that SD-ICN can be deployed in a scalable and incremental way with no modification of the core network, and can support typical communication modes such as multicast, mobility, multihoming, load balancing, and multipath data transmission effectively.

Keywords: information centric networking, software defined networking, name resolution, routing, dual space

http://dx.doi.org/10.3837/tiis.2016.05.017 **ISSN : 1976-7277**

1. Introduction

With the evolution of the Internet, mobile access traffic has become a main component of Internet traffic, thereby presenting significant pressure and challenge to the current network infrastructure. Moreover, the emergence of new applications such as multimedia content distribution results in a traffic explosion problem. In the face of these challenges, the TCP/IP architecture that aims to achieve host to host data transmission cannot satisfy network application requirements. Information Centric Networking (ICN) is one of the future Internet architectures aiming to solve such problems [1]. The typical features of ICN, such as content name based routing, intelligent data caching, and transparent content location, are appropriate for the evolution of the Internet from a host to host communication infrastructure to an information-sharing service platform. Many ICN implementations have been proposed in recent years, i.e. NetInf [2], NDN [3], PURSUIT [4], SOFIA [5], etc.

Although many efforts with regard to ICN have been made, a few open problems still exist that need to be solved. Firstly, ICN proposals such as NDN use the clean slate design philosophy, whose architectures are incompatible with the TCP/IP architecture. Thus deploying these architectures widely in a short time is impractical. Secondly, the Internet scale name based routing mechanism is difficult to perform routing aggregation, and thus the routing scalability becomes another barrier. Thirdly, according to the design philosophy, ICN can provide a better host mobility support. However, due to problems such as routing update convergence and routing table aggregation, most of current ICN service models still have drawbacks in supporting the mobility of content publishers [6].

In this paper, we attempt to find solutions by investigating the fundamental requirements of network communication. There are two key technical mainlines in the Internet evolution history: one is high speed data transmission and the other is intelligent data processing. The former provides fast routing and high speed connection among wide area distributed hosts, and thus, this component should have a simple packet processing capability. The latter provides on demand data processing capabilities such as reliable data transmission and intelligent data caching; such features require a complicated processing logic. In the TCP/IP architecture the former is implemented by routers, whereas the latter is implemented on the end systems, thereby forming the well-known end-to-end arguments [7]. However, with the emergence of new network applications, these arguments have shortcomings in solving problems such as traffic explosion and user mobility [8].

ICN implements the data caching and name based routing mechanism in the network core. Such a mechanism can help optimize the performance of applications, but it increases the complexity and cost of routers. Furthermore, letting all nodes participate in the intelligent data processing is not the only solution. For example, some studies show that the pervasive caching in ICN provides small performance improvement compared with the edge caching [9]. Thus, in the development of ICN, a reasonable definition on the boundary of high speed data transmission and intelligent data processing should be given by extending the end to end arguments. One the one hand, the core network should be kept simple and efficient, so as to reserve the wide area high speed data transmission capability of the Internet. On the other hand, the data processing capability in the edge network should be enhanced by using the flexible software and hardware structure, so as to improve the intelligent on demand data processing capability.

Software defined networking (SDN) separates the control plane and data plane, and uses the logically centralized controller to define the packet processing policy. Such a mechanism opens the network control plane, and provides a technical support to improve the network data processing capability [10]. Thus, by combining the SDN concept and the end to end arguments, we provide a wide area deployable software defined information centric networking service model SD-ICN. We also give the dynamic name resolution and routing mechanism under the dual routing space. Furthermore, we discuss the functions of SD-ICN, and implement a prototype in our wide area campus network to evaluate the feasibility and performance of SD-ICN. The main contributions of this paper are as follows:

Firstly, based on the analysis of the data communication requirements, we provide an innovated dual space ICN service model named SD-ICN. Such a model separates the high speed data transmission capability in the core network and intelligent data processing capability in the edge network, thereby effectively improving the ICN scalability. SD-ICN also provides an incrementally deployable architecture; hence, the qualitative benefits of ICN can be achieved without any changes to the network core.

Secondly, we propose a two-layer content search mechanism in SD-ICN and implement a prototype of SD-ICN. By doing so, we demonstrate that SD-ICN can support many features such as unicast, multicast, load balancing, multi-homing, and multi-path transmission, etc.

The remainder of this paper is organized as follows. Section 2 gives the related works. Section 3 presents the architecture as well as the content name routing mechanism of SD-ICN. Section 4 discusses the typical communication types supported by SD-ICN. Section 5 develops a prototype of SD-ICN to evaluate its feasibility. Finally, section 6 concludes the paper.

2. Related Work

Existing ICN proposals [2, 3, 4] build ICN architectures upon content abstraction. For example, NDN adopts a distributed name based routing mechanism, and maintains forwarding information base, pending interest table, as well as content store in each network nodes. NetInf uses a multilevel DHT-based name resolution service called MDHT that provides name-based routing. Intra-area routing and forwarding is done according to the rules of the local DHT algorithms, while inter-area routing is done by the node in both the local DHT and the next higher level DHT [11]. PURSUIT shifts the current send-receive based Internet toward the new publish-subscribe paradigm, and uses the rendezvous network to perform name resolution and data forwarding [4]. These proposals share some commonalities in the perspectives of basic primitives, name based routing, universe in-networking caching, etc [3].

However, there are also some deficiencies in wide area content location as well as data forwarding of these proposals. Though quite a number of improvement mechanisms are proposed, such as stateless forwarding [12], semi-stateless forwarding scheme [13], it still has problems such as routing inefficiency, poor scalability, etc. Besides, due to the routing aggregation constraint, the mobility of content publisher cannot be supported effectively by most of the current proposals [6], and a better node mobility support mechanism for ICN is still needed.

On the other hand, the separation of control plane and data plane, as well as the software defined packet processing mechanism in SDN provides a way of flexible packet processing. Due to the flexibility of SDN, how to use it to cope with the problems in ICN is also discussed comprehensively. Syrivelis et al. propose a SDN and ICN combined architecture [14]. Salsano et al. implement a SDN based ICN architecture and gives an experiment on the OFELIA

testbed [15]. However, most of current proposals focus on extending the OpenFlow protocol so as to support ICN packet forwarding, but the problems such as wide area deployment and scalability are not discussed in detail. How to design an evolvable, scalable, and incrementally deployable ICN service model is still an open problem.

Incremental deployment is another focus of attention recently. Fayazbakhsh et al. analyze the necessity of pervasive caching and nearest-replica routing in ICN, and presents an incrementally deployable ICN architecture [9]. Mukerjee et al. propose a general incremental deployment method of new network architecture based on intelligent control plane [16], which can be seen as an upper layer abstraction of our method. XIA [17] is an expressive Internet architecture with native support for multiple proposals, and its main aim is how to support the long term evolution of network architecture, which is quite different from that of SD-ICN. The relationship between SD-ICN and XIA can be regarded as that the former could be implemented within the latter's architecture.

3. Architecture Design of SD-ICN Service Model

3.1 Overview

In accordance with the functional requirement, SD-ICN includes two parts: Core Transmission Network (CTN) and Edge Service Network (ESN). The two parts use different technologies to solve high speed data transmission and intelligent data processing tasks. Thus, a dual space service architecture that allows network edge and network core to evolve separately is formed. In such architecture, CTN that provides wide area connection service is located in the network core and is unique globally. ESN is located in the network edge and includes a group of edge service subnets. As shown in **Fig. 1**, four edge service subnets are connected through the CTN to provide intelligent data processing service.

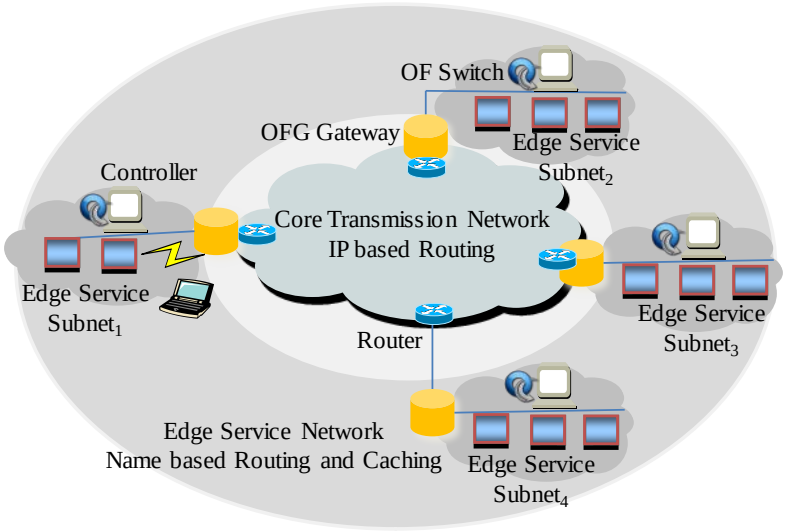

Fig. 1. architecture model of SD-ICN

The CTN constitutes the high speed data transmission space of SD-ICN, and it has an independent unified address space. Given that the wide area routing function of the Internet is one of its most valuable parts, the CTN of SD-ICN also uses IP to provide packet routing and

forwarding service for ESN. Hence, SD-ICN is compatible with the widely deployed Internet architecture.

The ESN constitutes the intelligent data processing space of SD-ICN, and it is divided into different edge service subnets connected by the CTN, $ESN = \{ESN_i, i = 1,2,...n\}$. Considering that SDN can implement complicated data processing and control functions based on software defined method in the controller, the enhanced SDN mechanism is introduced in the edge service subnet to perform intelligent data processing and content name based routing and caching. Each edge service subnet includes a logically unique controller and some ICN enabled OpenFlow switches. The intelligent content routing is performed based on the controller decision.

Fig. 2 shows the logical structure of the edge service subnet, which adopts the control plane and data plane separation concept of SDN. The upper control plane maintains the topology view and the resource view of the edge service subnet. The control plane is responsible for controlling decision related operations such as content name registration and resolution, global content location, content processing rule generation, and content transmission optimization, etc. Meanwhile, the bottom layer data plane is composed of name routing enabled OpenFlow switches, content caching devices, and end systems. The data plane is responsible for content name based data transmission.

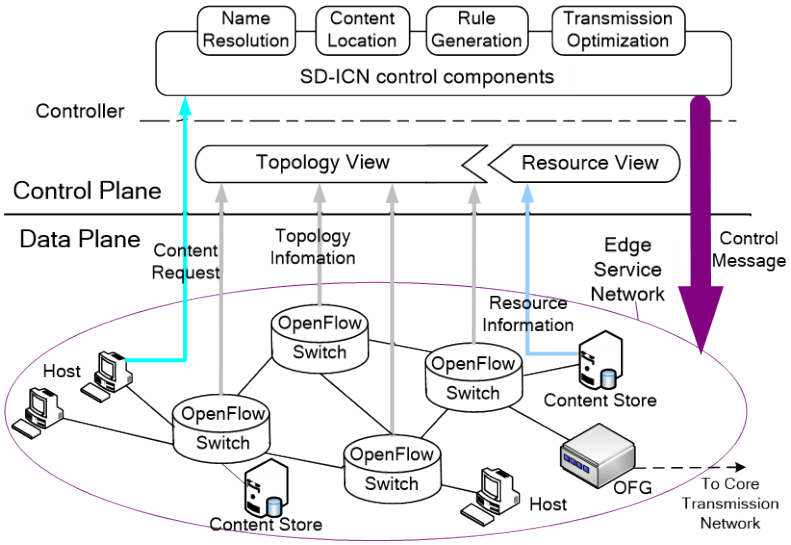

Fig. 2. logical structure of edge service subnet

In each edge service subnet, the content name based routing mechanism is implemented by enhancing the functions of the controller and OpenFlow switches. The dynamic registration and query mechanism of content name in an edge service subnet are also designed. Furthermore, to interconnect the ESN and CTN, the OpenFlow gateway (OFG) is designed by extending the basic functions of OpenFlow switch. Each edge service subnet has a unique OFG logically, but it can have many implementation instances so as to satisfy the scalability requirement. The OFG plays an interconnection role between the edge service subnet and the core transmission network. For one thing, packets can be routed to the OFG through name based routing in the edge service subnet. For the other, each OFG also has a global IP address of the core transmission network, and thus packets can be routed to different OFGs through IP based routing in the core transmission network.

Fig. 3 demonstrates the content transmission process in SD-ICN through a simple network structure. Suppose host H_0 and H_1 are located in ESN_1, whose controller and OFG is $Ctrl_1$ and OFG_1 respectively. Host H_2 is located in ESN_2, with $Ctrl_2$ and OFG_2 as its controller and OFG. When host H_0 requires content $Name_i$ that stored in host H_2, its request will reach $Ctrl_1$ through the forwarding of OpenFlow switches. By using the name resolution system, $Ctrl_1$ finds that $Name_i$ is located in ESN_2 that has IP_2 as the address of OFG_2, and then $Ctrl_1$ will add rules for the request packet and the response packet in the OpenFlow $switch_1$ and OFG_1. The format of the rule is <*Name*, *Type*, *Next-hop*>, in which *Name* is the identifier of the content, *Type* shows whether the packet is a request packet or a response one, and *Next-hop* denotes the next hop that the packet should be forwarded. The request packet will then be routed to OFG_1, which encapsulates the ICN packet into an IP packet with source address as IP_1 and destination address as IP_2. Next, the encapsulated IP packet will be routed to OFG_2 through the CTN. $Ctrl_2$ will also add rules for the request packet and the response packet in the OpenFlow $switch_2$ and OFG_2. OFG_2 extracts the ICN packet and routes this packet to host H_2 through name based routing. When H_2 sends the response packet, given that all the related switches and OFGs already have the rules in their flow table, the response packet will be routed back to H_0 in reverse.

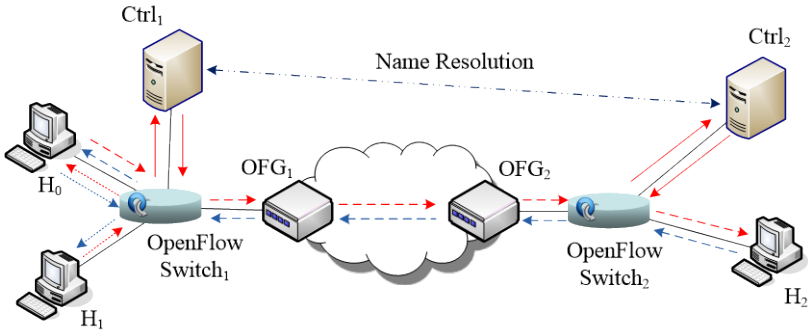

Fig. 3. demonstration of content transmission process in SD-ICN

Moreover, the received content can be cached in the end system such as host H_0 or other specific caching systems. Other end systems, such as host H_1, can fetch the content in the local edge service subnet directly. Thus the network transmission load is decreased, whereas the content fetching efficiency is increased.

3.2 Controller Cooperation based Content Routing Mechanism

In SD-ICN, the CTN of SD-ICN uses the IP routing mechanism, whereas the ESN uses software defined content name routing mechanism. Given that the edge service subnets are distributed in a wide area and that different edge service subnets are independent of each other, the way to implement global name based content location and routing is an important problem in SD-ICN. In this paper, by using the logically unique controller that maintains information such as network topology and routing policy in each edge service subnet, we construct a scalable wide area content routing mechanism based on controller cooperation.

As in SDN, the routing in a single edge service subnet can be solved through the controller scheduling. By enhancing the functions of controllers in different edge service subnets, we construct a ***two-layer abstraction content registration structure*** to solve the global content name routing problem. In such a structure, the controllers in different edge service subnets form a distributed global name resolution system, and cooperate with each other to respond the content name registration and lookup. Considering the high efficiency of Distributed Hash

Table (DHT), SD-ICN adopts a DHT based distributed name resolution system, which is shown in **Fig. 4**. The controllers of each edge service subnets build a logical DHT structure using their global IP addresses. As a result, the content name resolution problem becomes a distributed content lookup problem similar to that in P2P network. This problem can be solved effectively through the DHT structure.

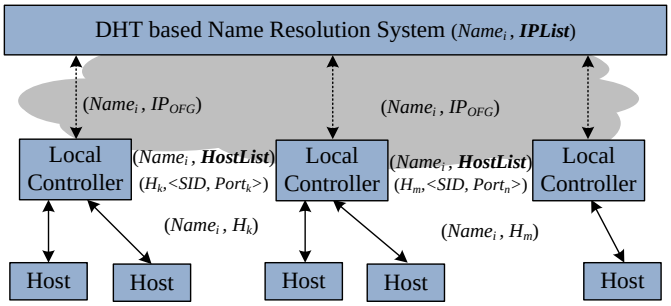

Fig. 4. DHT based content registration and lookup

Suppose a node H_k in the edge service subnet ESN_m adds content $Name_i$. Hence, H_k will notify the controller C_m in ESN_m to add the mapping relationship between $Name_i$ and H_k by registering $Name_i$ to the controller in ESN_m. The registration format is $(Name_i, Host_k)$, which means content $Name_i$ is located in host H_k. Moreover, each controller must maintain a mapping between host and its current location. The format is (H_k, <*SID*, $port_i$>), which means host H_k is connected to $port_i$ of switch *SID*, if host H_k moves to another location such as <*SID*, por_k>, an then the mapping will be updated to be (H_k, <*SID*, $port_k$>).

During the content registration process, if other hosts in ESN_m already have the replica of content $Name_i$, then C_m will process no operation in the upper DHT based global name resolution system; otherwise, C_m will publish the registration information in the DHT structure with the format $(Name_i, IP_m)$, in which IP_m is the IP address of the OFG in ESN_m. Such registration information will be stored in the corresponding controller of the DHT structure determined by the hash value of $Name_i$. When searching for the content $Name_i$, other nodes will discover that one or more replica of $Name_i$ is located in ESN_m, whose OFG address is IP_m. Given that many edge service subnets may have the replica of content $Name_i$, the registration information is stored with the format $(Name_i, IP_1, ..., IP_n)$, which means content $Name_i$ can be found in *n* edge service subnets with OFG addresses IP_1, …, IP_n.

Algorithm 1 presents the basic description of the content registration process.

Algorithm 1: two layer abstraction content registration
Input: (ESN_m, $Name_i$, $Host_i$)

// $Name_i$ already has replica in local ESN
1: if *Value*($Name_i$, ESN_m) != NULL
    // *SID* and $port_i$ determine the host
2: *localUpdate*($Name_i$, $Host_i$);
3: else
4: BEGIN
    // local record process
5: *localInsert*($Name_i$, $Host_i$);
6: IP_m = *getOFGIP*();

```
      // Name_i already recorded in DHT by other ESN
7:    if Value(Name_i, DHT) != NULL
        // DHT process, announce ESN_m also has Name_i
8:      globalUpdate(Name_i, IP_m);
9:    else
        // DHT process, add a new record of Name_i
10:     globalInsert(Name_i, IP_m);
11: END
```

Lines 1-5 demonstrate the local record operation process, in which *localUpdate* and *localInsert* are the information update and insert process in the local controller respectively. Lines 6-10 represent the global record operation process in the DHT structure, in which *globalUpdate* and globalInsert are the information update and insert process of content $Name_i$, respectively. As observed from Algorithm 1, only the first addition of a content replica in an ESN will cause the insert or update operation in the DHT structure. Furthermore, the variation of content replica number is handled locally with no affect on the global DHT structure. Thus, the overall load on the DHT structure is decreased effectively.

The content lookup also includes a two-layer mechanism: ***global fuzzy search*** and ***local accurate search***. The former provides a means to quickly locate the ESNs that have the content replica, whereas the latter determines which hosts have the content replica in a special ESN. When searching the content $Name_i$, H_i firstly queries the controller C_m in its ESN_m locally. If the matching result is found $D = \{< SID, port_i >\}$ (*SID* is the special OpenFlow switch identifier and $port_i$ is a port of switch *SID*, and the two values can locate a host uniquely), then the controller C_m will notify the related switches to construct a route from H_i to a certain host in set ***D***. Otherwise, if no matching is found locally, then the controller C_m will use $Name_i$ as the key to search in the DHT structure. As a result, the DHT structure will send back a list of OFG addresses whose ESNs have the replica of content $Name_i$. After receiving the OFG list, the controller C_m will select one or more ESNs to request content $Name_i$ based on some specific criteria. Suppose the ESN_k with OFG_k is selected, and then C_m will notify the related switches to construct a route from H_i to OFG_m and will notify OFG_m to encapsulate the request packet so as to transmit this packet globally through IP routing. Finally, the controller in ESN_k will perform the local accurate search in its local ESN.

The deletion of content is also performed hierarchically. When host H_i in ESN_m deletes content $Name_n$, H_i will notify the controller C_m in ESN_m. C_m will then delete the mapping relationship between $Name_n$ and H_i. If other hosts in ESN_m still have the replica of content $Name_n$, then C_m will process no operation in the upper DHT based global name resolution system; otherwise, C_m will request to delete the mapping relationship between $Name_n$ and its OFG IP address. After the deletion in the upper DHT, if no other ESN has the replica of content $Name_n$, then the recording entry of $Name_n$ will be deleted from the DHT structure.

Algorithm 2 describes the content lookup and deletion process. For content lookup, if a replica is found in local ESN, then the controller can select one or more hosts as the content publishers, and can notify the related switches to construct a route to these hosts. During the communication process, the route and the hosts can be adjusted dynamically.

Algorithm 2: Content Lookup and Deleting

Lookup:

1: <*SID*, *port*> = *findHostSet*($Name_i$); // local process
2: if <*SID*, *port*> == NULL
// no replica in local ESN, starting global lookup
3: BEGIN
4: *IPList* = *findESNSet*($Name_i$); // local process
//Locating the best replica
5: *IPSet* = *ContentLocating*(*IPList*);
6: END

Deleting:

7: *localDel*($Name_i$);
8: if *localValue*($Name_i$) == NULL
9: BEGIN
10: *globalDel*($Name_i$, IP_m); // DHT process
11: if *globalValue*($Name_i$) == NULL
//No copy of content $Name_i$ exists
12: BEGIN
13: *remove*($Name_i$);
14: END
15: END

Suppose there are n ESNs in the SD-ICN architecture, and each ESN has m hosts. The time complexity of local content lookup can be reduced to $O(1)$ by using Hash method [18]. According to the definition of DHT, in a DHT structure with n nodes, the time complexity of global content lookup is $O(logn)$ [19]. If the content requested by a node p_i locates evenly among different ESNs, then the probability of the requested content in the same ESN with p_i is $\frac{1}{n}$, whereas the probability of the requested content in different ESN with p_i is $1-\frac{1}{n}$. Accordingly the time complexity of finding such a content will be $\frac{1}{n}O(1)+(1-\frac{1}{n})O(\log n)$. Notably, in each ESN, only the first request of a content needs the global content lookup process. Subsequent requests can be served locally because of the local caching feature of SD-ICN. Thus, the average content lookup complexity for a random host will be $\frac{1}{m}(\frac{1}{n}O(1)+(1-\frac{1}{n})O(\log n))$, which means it has a time complexity of $O(\frac{\log n}{m})$. As n increases, the multi-level DHT [11] technology could be used to guarantee the lookup latency.

If multiple ESNs that have the required content replica are found, then one or more optimized targets need to be selected from them. To utilize the advantage of ICN's content name based routing, we design a multiple controller cooperation based content locating mechanism in SD-ICN. This mechanism can perform the best mapping between content name and content location based on criteria such as network performance, network policy, etc. Algorithm 3 describes the basic concept of such a mechanism.

Algorithm 3: Content Locating
Input: Candidate ***IPList***
Output: one or more preferred content locations

1: $PathSet$ = FindPath(uIP,$IPList$) ;
// find the available paths from each IP in *IPList* to *uIP*
3: **for** $path_i$ **in** $PathSet$ **do**
// calculate the cost of each path
4: BEGIN
5: $< X_{i1}, X_{i2}, ..., X_{in} >$= GetAttrs($path_i$) ;
// get the sub-cost of path attributes numbered from 1 t
6: $cost_i = \sum_{j=1}^{n} \lambda_j X_{ij}$; // calculate the total cost
7: END
8: k = MinCost($< cost_1, cost_2, ..., cost_m >$) ;
// get the preferred content locations set from all candidate
9: **return** $\{IP_k\}$;

In algorithm 3, ***IPList*** represents the set of ESNs that have content $Name_i$, and *uIP* represents the OFG address of the content requester's ESN. Firstly the path information as well as the path properties to all candidate ESNs is acquired. Then an integrated path cost is calculated. Finally one or more ESNs are selected as the communication targets based on the calculated costs, whose OFG addresses are returned back to the content requester's controller.

After receiving the content location information, the controller of the content requester will construct the transmission paths. It's needed to point out that the path cost here could be defined flexibly according to application types. For example, the online streaming applications should select paths with high available bandwidth and low jitter, whereas the interactive applications should select paths with low latency, etc. Algorithm 3 only provides the optimized target selection capability of SD-ICN, and the detailed cost definition as well as cost calculation is beyond the scope of this paper. Such an issue will be discussed in a future work.

4. Typical Communication Types of SD-ICN

Before data transmission, SD-ICN uses the controller to select the best communication peers based on the content name. SD-ICN can support typical communication types such as unicast, multicast, mobility and multi-homing, load balancing and multipath transmission effectively. In this part, we will analyze the implementation mechanism of typical communication types in SD-ICN.

4.1 Unicast

In unicast communication mode of SD-ICN, if the content requester and publisher are located in the same edge service subnet, then they will use content name based routing directly. The controller is responsible for determining the route by sending the specific flow processing rules to the related OpenFlow switches. If the content requester and publisher are located in

different edge service subnets, then the data will traverse two edge service subnets and the CTN. Such a process has already been discussed in Section 3.

During the communication process, the source host only needs to send out its content request, and does not need to know in which host and ESN the content is located. All the content lookup and mapping operations are handled by the cooperation of the controllers and the OFG gateways. Moreover, when there are multiple replicas in the network, the controller can select the best one according to the predefined criteria. The communication peer can also be changed dynamically during the communication process, which is also handled by the controllers and is transparent to the end host.

4.2 Multicast

Multicast is a network function that facilitates efficient information dissemination. Based on the content name routing in edge service network, SD-ICN can support content multicast in the granularity of different ESNs.

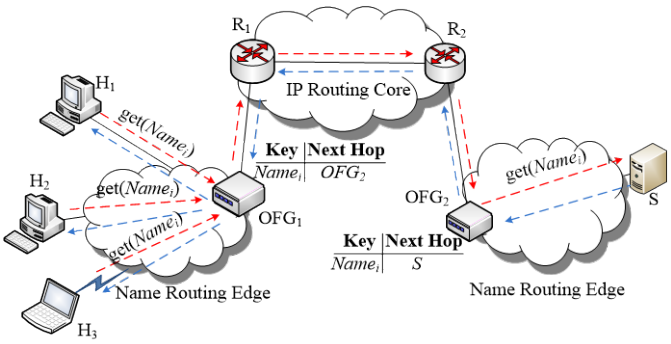

Fig. 5. multicast in SD-ICN

As shown in **Fig. 5**, host H_1, H_2 and H_3 in ESN_1 are all requesting content $Name_i$. When these request packets arrive at the OFG_1, it will ask the controller to handle such requests. The controller finds that ESN_2 has the content, and then it will notify the OFG_1 to encapsulate only one packet with source IP address IP_{OFG1} and destination IP address IP_{OFG2}. After receiving the response IP packet, OFG_1 will extract the content packet and send this packet to H_1, H_2, and H_3 separately. When hosts in multiple ESN_s request the same content in server S, S could only send one copy to OFG_2, and let OFG_2 encapsulate a packet for each ESN. In this manner, the edge service subnet based multicast mechanism is implemented, and the traffic load in the wide area network is decreased.

4.3 Mobility and Multihoming

Since SD-ICN uses the content name based routing in the edge service network, and the end hosts do not care the location of contents, the mobility of end hosts will not affect the ongoing communications. With the dynamic mapping of content name and the OFG address, SD-ICN can support the mobility of both content requester and content publisher effectively. Meanwhile, the switching among different interfaces of the multi-homing node can be seen as a special type of mobility, and it can also be handled by SD-ICN effectively.

Suppose hosts H_1 and H_2 are located in ESN_1 and ESN_2, and the original communication path is $<H_1, OFG_1, OFG_2, H_2>$. When H_2 moves to ESN_3, H_2 will firstly register to controller C_3 in ESN_3. Then C_3 will notify controller C_1 in ESN_1 to update the mapping relationship. Finally $<H_1, OFG_1, OFG_3, H_2>$ will be used as the new data transmission path. In this manner, the goal of seamless handover during node mobility can be achieved.

Such a feature is vital for improving the data transmission performance. The following two issues are solver effectively: the limitations of mobility support in IP network and the inability to handle content publisher mobility effectively because of route aggregation in current ICN proposals. By supporting the interface switch of multi-homing hosts, SD-ICN can ensure seamless host switch among WiFi and 3G interfaces according to the traffic and network performance status.

4.4 Load Balancing and Multipath Data Transmission

SD-ICN uses content name based routing mechanism in the edge service network, and let the controller be responsible for the dynamic mapping between the content name and the core transmission network IP address. Thus during the mapping process, a dynamic load balancing and multipath transmission mechanism can be implemented according to the content location status.

For load balancing, suppose a server S_1 in ESN_1 is selected as the content downloading source, if the transmission rate is decreased due to the overload of S_1 or ESN_1, the controller can adjust the mapping between the content name and the OFG, and switch to another server S_2 to download the content. Such a switch is accomplished by the controller dynamically, and is totally transparent to the end host.

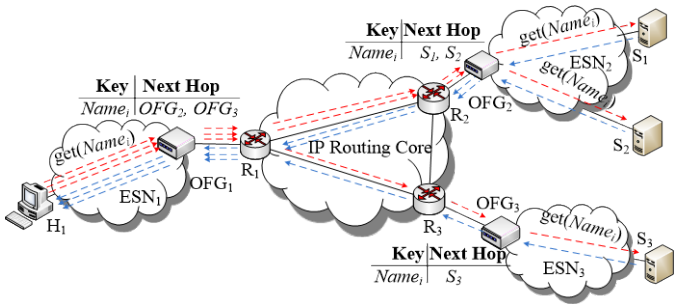

Fig. 6. multipath data transmission in SD-ICN

By using the feature of dynamic mapping between the content name and the IP address, SD-ICN can also achieve the multipath efficient data transmission aims. As shown in **Fig. 6**, host H_1 in ESN_1 request the content $Name_i$, and when such a request arrives at the OFG_1, it selects to download the content from S_1 in ESN_2, and thus forms the $<H_1, OFG_1, OFG_2, S_1>$ transmission path. If the bandwidth of such a path cannot satisfy the application requirement, OFG_1 can select S_2 in ESN_2 and S_3 in ESN_3 as the communication peers too, and form three parallel data transmission paths $<H_1, OFG_1, OFG_2, S_1>$, $<H_1, OFG_1, OFG_2, S_2>$, and $<H_1, OFG_1, OFG_3, S_3>$ simultaneously. The content data is scheduled among the three paths, so that the data transmission efficiency can be increased. Besides, the end host only sees the content name, and it does not care how many paths are used to transmit the data. Thus, the number of transmission paths can be adjusted dynamically according to the content distribution and network performance status.

The multipath data transmission in SD-ICN is fundamentally different form that of MPTCP. MPTCP utilizes multiple paths for data transmission for the same TCP connection between two ends, while multipath transmission in SD-ICN could involve multiple servers, rather than a fixed one. Besides, the implementation of multipath transmission is transparent for the applications. Compared with other information centric networks such as NDN and Sofia, SD-ICN utilizes a dual space structure that separates edge processing and core transmission, which has a better scalability.

5. Prototype Implementation and Evaluation

5.1 Prototype Implementation

The controller in SD-ICN is implemented based on the POX controller, whereas the OFG gateway is implemented by enhancing the OpenFlow switch. Finally, the end host is modified so as to support the name based communication.

The main components of the controller and switch are shown in **Fig. 7**. The OpenFlow switch maintains the flow table, which supports the flexible routing policies defined by the controller. The OFG gateway has an additional packet encapsulation component apart from the standard OpenFlow switch components. This component is responsible for data encapsulation of packets transmitted among different edge service subnets. The controller manages the topology, path, and cache in its subnet and DHT structure with other subnets. With the cooperation of different controllers, the content replica can be optimized according to the request distribution.

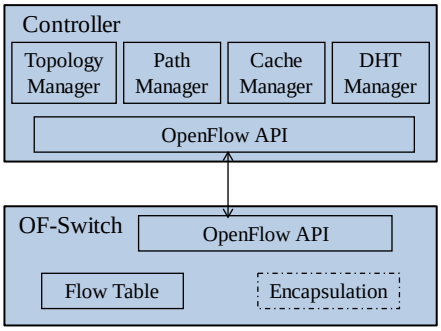

Fig. 7. structure of controller and of-switch in SD-ICN

The main goal of this paper is to demonstrate the feasibility of SD-ICN. Thus, we use the raw socket programming method to generate the content requesting packet. The format of the header is similar to that of the IP packet, but the semantic of the two formats is different. For example, we use the 64-bit source and destination address fields to represent the hash value of the content name, and we use the protocol type field to identify the ICN request and response packet.

After receiving a packet, the OpenFlow switch identifies the ICN packet based on the protocol type field, and then it uses the 64-bit content name hash value and the protocol type to match its flow table. If a matching entry is found, then the packet will be forwarded based on the entry; otherwise the packet will be forwarded to the controller. For the content request packet, the controller finds the best replica of the requested content based on the DHT structure, and sends back the flow entries to the switch. The flow entries include an entry for the request packets and an entry for the corresponding response packets. Since the forwarding path for the content response packet will be constructed when handling the corresponding content request packet, match entries will be found in the specific OpenFlow switch for the content response packet. The switch will then forward packets (either request packets or response packets) to the appropriate destination host (if located in the same edge service subnet) or to the OFG gateway (if located in different edge service subnet). The OFG gateway is responsible for encapsulating packets and sending them to the right destination edge service subnet's OFG gateway through the core transmission network. And finally the destination edge service subnet's OFG gateway extracts the ICN packets and sends them to the right end host.

To validate the feasibility of the SD-ICN service model, we implement a prototype as shown in **Fig. 8**. The prototype includes three edge networks connected by our campus IP network. Each edge network includes three OpenFlow switches, one OFG gateway, one local controller and a few end hosts. We implement the OFG and the DHT name resolution system for each ESN. We also implement the raw socket programming based video streaming and file sharing demonstration applications in the end host. The experiments are repeated 15 times and the result is the average of these experiments.

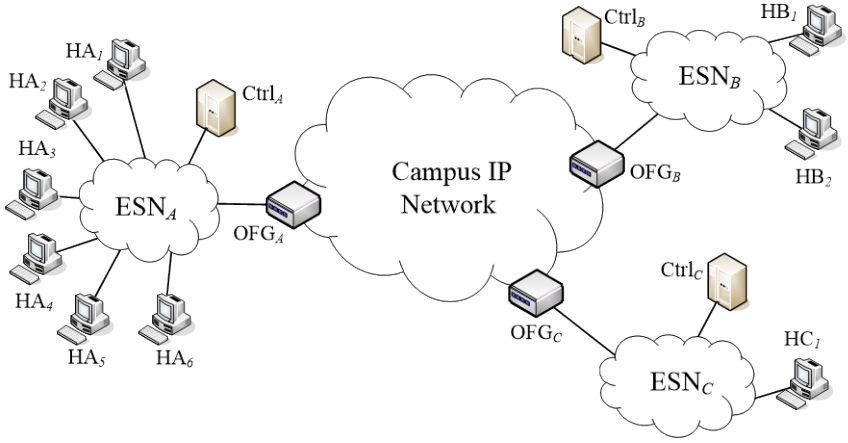

Fig. 8. experiment environment of SD-ICN

All OpenFlow switches are 64-bit Linux PCs with five Ethernet ports, running the OpenFlow 1.0.0 Stanford reference software implementation [20]. It supported 100 Mbps throughput without dropping packets, which is sufficient for our experiments. For our prototype, other hardware OpenFlow switches can also be used. But in the future work, some modules such as an improved content name based routing mechanism may be added to the switches. Thus the software based implementation is a better choice. The OFG gateway is implemented based on the OpenFlow 1.0.0 Stanford reference software implementation, and modules such as packet encapsulation and IP packet forwarding are added. The POX OpenFlow controllers are used, and also Python modules are developed for local name resolution, global DHT name resolution, flow table manipulation, etc. Controllers in different edge service subnets form a chord DHT structure, and content names are registered in the chord structure.

5.2 Experiment Analysis

5.2.1 Multicast

Firstly we give a test on the multicast capability of SD-ICN. Host HB_1 in ESN_B registers a 2 Mbps high-definition video streaming V_i in the network, and Hosts HA_1, HA_2, HA_3, HA_4, HA_5 and HA_6 in ESN_A all request to watch such a video streaming at different intervals. When the first request arrives at the OFG_A, OFG_A will query $Ctrl_A$ for the mapping relationship of V_i and the global IP address, and will then encapsulate the request packet to process the wide area communication. When the other requests arrive at OFG_A, OFG_A can duplicate the received streaming packets, and then send them to HA_1, HA_2, HA_3, HA_4, HA_5 and HA_6. If host HA_1 leaves during the multicast process, then $Ctrl_A$ will capture such an event, and select another request from other host randomly as the new representative request. The upload and download rates are calculated every 8 seconds.

Fig. 9 shows the comparison of the streaming server's upload rate and the sum of hosts' download rate. As the number of hosts joining the multicast process increases, the sum of the download rate that each host acquired also increases accordingly. However, during the entire process, the upload rate of the streaming server remains constant. Such an experiment illustrates that SD-ICN is an attractive architecture to support multicast, and thus provides an efficient solution for live video streaming, multiparty video conference, and IPTV.

Fig. 9. transmission rate comparison of server and hosts

5.2.2 Multihoming

To validate the multihoming supporting capability of SD-ICN, we let host HB_1 in ESN_B register a video streaming application V_i in the network, and then HA_1 in ESN_A requests such a streaming by name. HA_1 has two network interfaces: a wired one and a wireless one. The link bandwidths of these interfaces are restricted to be 2 and 1 Mbps respectively through the *Linux TC* tool. During the communication process the wired link that has a higher speed is preferred if possible, and the data transmission rate is also calculated every 8 seconds.

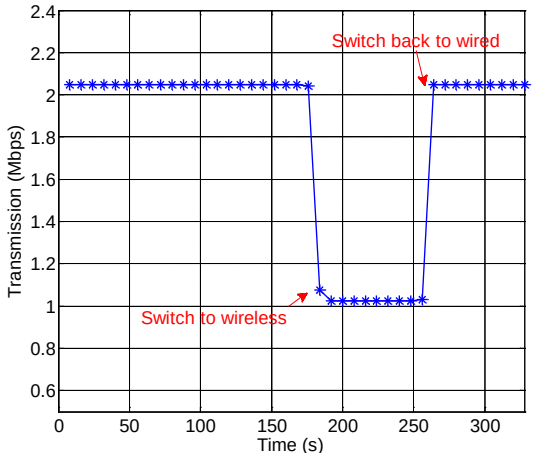

Fig. 10. multihoming switch data transmission

Firstly, the streaming is transferred from host HB_1 to host HA_1 via the wired link at approximately 2 Mbps. Then, the wired cable of host HA_1 is unplugged. As shown in **Fig. 10**,

the SD-ICN detects the failure of the wired network interface and switches to the low speed available wireless interface by notifying the controller about the change of the transmission interface. All these operations are transparent from the perspective of the application. Afterwards, the wired cable of HA_1 is re-plugged. The SD-ICN switches back to the wired link without interrupting the transfer. Our experiment results demonstrate that SD-ICN can support multihoming efficiently, and thus provide a useful way of utilizing the increasing number of ubiquitous multihoming devices.

5.2.3 Load Balancing and Multipath Data Transmission

During the load balancing capability evaluation process, we let HB_1 and HB_2 in ESN_B and HC_1 in ESN_C register a large file F_i in the network, and then HA_1 in ESN_A downloads such a file by name. The data transmission rate is recorded every 8 seconds. The upload rates of HB_1, HC_1, and HB_2 are restricted to be 2048, 2048, and 1536 Kbps respectively through the *Linux TC* tool.

Firstly HA_1 downloads F_i from HB_1. Afterwards, we restrict the access bandwidth of HB_1 to 1 Mbps. When detecting the transmission rate decreasing, the controller selects HC_1 as the new server; thus HA_1 requires F_i from host HC_1. A few moments later, we also restrict the access bandwidth of HC_1 to 1 Mbps. The controller then selects HB_2 as the new server. **Fig. 11** presents the downloading bandwidth variation during such a process. The trend shows that as the bandwidth changes, SD-ICN can always help HA_1 select the best download sources dynamically.

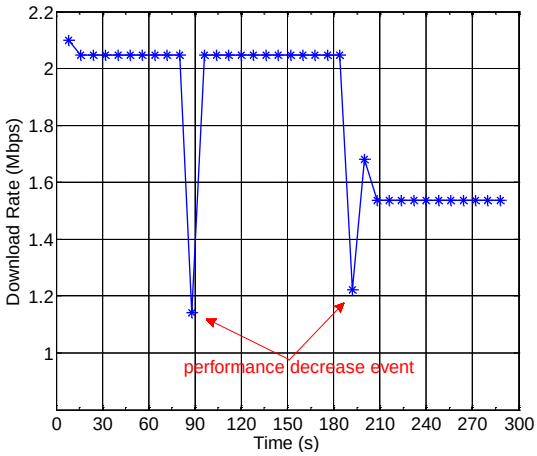

Fig. 11. load balancing in SD-ICN

To validate the multipath data transmission capability, we let HB_1 and HC_1 register a large file F_i in the network and adjust their access bandwidths between 200 and 1000 Kbps dynamically. The download rate is recorded every 8 seconds. When HA_1 requires the content F_i, the controller of ESN_A will map the content name F_i to the OFG IP addresses of both ESN_B and ESN_C. Accordingly, HA_1 can download data from HB_1 and HC_1 simultaneously.

Fig. 12 shows the download rate of HA_1 and upload rates of HB_1 and HC_1 with multipath data transmission mechanism. The trend shows that as the upload rates of HB_1 and HC_1 vary, the download rate of HA_1 also varies correspondingly. In addition, the download rate HA_1 is approximately equal to the sum of the upload rates of HB_1 and HC_1. Compared with the single data source downloading in traditional TCP/IP architecture, SD-ICN can improve the data downloading rate effectively.

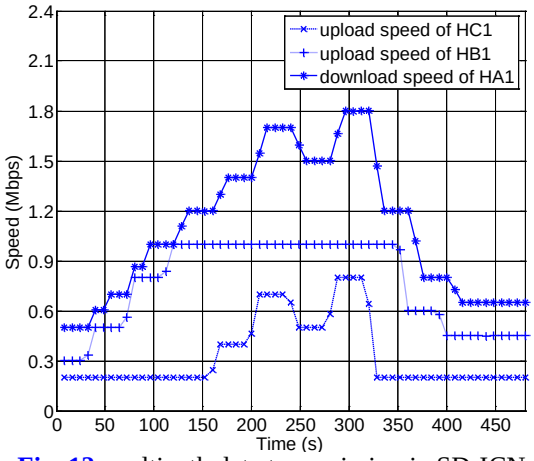

Fig. 12. multipath data transmission in SD-ICN

6. Conclusions

To solve the deficiencies such as incremental deployment, scalability, and mobility support of existing ICN proposals, we propose a wide area deployable software defined information centric networking service model SD-ICN in this paper. SD-ICN utilizes the dual space architecture that separates the edge service network and the core transmission network. In the edge service network, the enhanced SDN technology is used to implement the intelligent data routing and caching functions. In the core transmission network, the high speed transmission IP functions are reserved. The controller cooperation based distributed name resolution system and the OFG gateways are implemented to solve the global name resolution problem. Finally, we implement a prototype that includes three edge service subnets and use the wide area campus network as the core transmission platform. Our experiment results show that SD-ICN can be deployed in a scalable and incremental manner without the need to modify the core IP network. SD-ICN can also support unicast, multicast, mobility and multihoming, load balancing and multipath transmission effectively. In future work, we will validate the feasibility of SD-ICN in a larger testbed, and the optimization of path selection and caching in SD-ICN is another important issue that needs to be considered further.

Acknowledgements

This work supported by the State Key Development Program for Basic Research of China under Grant No.2012CB315806, the National Natural Science Foundation of China under Grant No.61379149 and No.61103225, and Jiangsu Future Networks Innovation Institute Prospective Research Project on Future Networks under Grant No.BY2013095-1-06.

References

[1] G. Xylomenos, C. Ververidis, V. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. Katsaros, and G. Polyzos, “A survey of information centric networking research,” *IEEE Communications Surveys & Tutorials*, vol. 16, no. 2, pp. 1024 - 1049, May, 2014. Article (CrossRef Link)

[2] C. Dannewitz, D. Kutscher, B. Ohlman, S. Farrell, B. Ahlgren, and H. Karl, “Network of Information (NetInf) – an information-centric networking architecture,” *Computer Communications*, vol. 36, no. 7, pp. 721 - 735, April, 2013. Article (CrossRef Link)

[3] V. Jacobson, D. Smetters, J. Thornton, M. Plass, N. Briggs, R. Braynard, "Networking Named Content," *ACM CoNext*, December 1 - 4, Rome, Italy, 2009. Article (CrossRef Link)
[4] P. Jokela, A. Zahemszky, C. Rothenberg, S. Arianfar, and P. Nikander, "LIPSIN: line speed publish/subscribe inter-networking," *ACM SIGCOMM*, August 17 - 21, Barcelona, Spain, 2009. Article (CrossRef Link)
[5] Q. Wu, Z. Li, J. Zhou, H. Jiang, Z. Hu, Y. Liu, and G. Xie, "SOFIA: toward service-oriented information centric networking," *IEEE Network*, vol. 28, no. 3, pp. 12 - 18, May 2014. Article (CrossRef Link)
[6] D. Kim, J. Kim, Y. Kim, "Mobility support in content centric networks," *ACM SIGCOMM ICN Workshop*, August 17, Helsinki, Finland, 2012. Article (CrossRef Link)
[7] L. Kleinrock, "History of the internet and its flexible future," *IEEE Wireless Communication*, vol. 15, no. 1, pp. 8 - 18, February, 2008. Article (CrossRef Link)
[8] J. Rexford, C. Dovrolis, "Future internet architecture: clean slate versus evolutionary research," *Communications of the ACM*, vol. 53, no. 9, pp. 36 - 40, September 2010. Article (CrossRef Link)
[9] S. Fayazbakhsh, Y. Lin, A. Tootoonchian, A. Ghodsi, T. Koponen, B. Maggs, K. Ng, V Sekar, and S Shenker, "Less pain, most of the gain: incrementally deployable ICN," *ACM SIGCOMM*, August 12 - 16, Hong Kong, China, 2013. Article (CrossRef Link)
[10] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling Innovation in Campus Networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, No. 2, pp. 69 - 74, April 2008. Article (CrossRef Link)
[11] C. Dannewitz, M. Ambrosio, and V. Vercellone, "Hierarchical DHT-based name resolution for information-centric networks," *Computer Communications*, vol. 36, no. 7, pp. 736 - 749, April 2013. Article (CrossRef Link)
[12] A. Detti, N. Melazzi, S. Salsano, and M. Pomposini, "CONET: a content centric inter-networking architecture," *ACM SIGCOMM ICN Workshop*, Toronto, Ontario, Canada, August 19, 2011. Article (CrossRef Link)
[13] C. Tsilopoulos, G. Xylomenos, Y. Thomas, "Reducing forwarding state in content-centric networks with semi-stateless forwarding," *IEEE Infocom*, Toronto, Canada, April 27 - May 2, 2014. Article (CrossRef Link)
[14] D. Syrivelis, G. Parisis, D. Trossen, P. Flegkas, V. Sourlas, T. Korakis, and L. Tassiulas, "Pursuing a software defined information-centric network," *European Workshop on Software Defined Network*, Darmstadt, Germany, October 25 - 26, 2012. Article (CrossRef Link)
[15] S. Salsano, N. Blefari-Melazzi, A. Detti, G. Morabito, and L. Veltri, "Information centric networking over SDN and OpenFlow: architectural aspects and experiments on the OFELIA testbed," *Computer Networks*, Vol. 57, no. 16, pp. 3207 - 3221, November, 2013. Article (CrossRef Link)
[16] M. Mukerjee, D. Han, S. Seshan, and P. Steenkiste, "Understanding Tradeoffs in incremental deployment of new network architectures," *ACM CoNEXT*, Santa Barbara, USA, December 9 - 12, 2013. Article (CrossRef Link)
[17] D. Han, A. Anand, F. Dogar, B. Li, H. Lim, M. Machado, et al., "XIA: architecting a more trustworthy and evolvable internet," *ACM SIGCOMM Computer Communication Review*, Vol. 44, no. 3, pp. 50 - 57, July, 2014. Article (CrossRef Link)
[18] M. Drmota, R. Kutzelnigg, "A Precise Analysis of Cuckoo Hashing," *ACM Transactions on Algorithms*, vol. 8, No. 2, April 2012. Article (CrossRef Link)
[19] S. Rhea, B. Godfrey, B. Karp, J. Kubiatowicz, S. Ratnasamy, S. Shenker, I. Stoica, and H. Yu, "OpenDHT: A Public DHT Service and Its Uses," *ACM SIGCOMM*, Philadelphia, Pennsylvania, USA, August 21–26, 2005. Article (CrossRef Link)
[20] OpenFlow 1.0.0 Specification [Online]. Article (CrossRef Link).

Changyou Xing received the Ph.D. degree from PLA University of Science and Technology, Nanjing, China, in 2009. He is currently an associate professor at PLA University of Science and Technology, Nanjing, China. His research interests include software defined networking, network measurement, network modeling.

Ke Ding received the M.S. degree from PLA University of Science and Technology, Nanjing, China, in 2005. He is currently a Ph.D. candidate at PLA University of Science and Technology. His research interests include software defined networking, network architecture.

Chao Hu received the Ph.D. degree from PLA University of Science and Technology, Nanjing, China, in 2013. He is currently an assistant professor at PLA University of Science and Technology. His research interests include software defined networking, distributed computing.

Ming Chen received the Ph.D. degree from Nanjing Institute of Communication Engineering, China, in 1991. He is currently a professor at PLA University of Science and Technology, Nanjing, China. He held visiting position at Columbia University in 1999. His research interests include network architecture, network measurement, future networking.

Bo Xu received the Ph.D. degree from PLA University of Science and Technology, Nanjing, China, in 2011. He is currently an assistant professor at PLA University of Science and Technology. His research interests include software defined networking, distributed computing.