Journal of Network and Systems Management
https://doi.org/10.1007/s10922-018-9482-z

OpenFlow Compatible Key-Based Routing Protocol: Adapting SDN Networks to Content/Service-Centric Paradigm

Adrian Flores-de la Cruz[1] · Pilar Manzanares-Lopez[1] · Juan Pedro Muñoz-Gea[1] · Josemaria Malgosa-Sanahuja[1]

Received: 24 March 2018 / Revised: 25 October 2018 / Accepted: 8 November 2018

Abstract

The host-to-host/content/service communication instead of the host-to-host communication offered by traditional Internet Protocol (IP) routing solutions has been demanded in the last few years. Nowadays, getting this type of communication directly at network level is an increasing demand in the framework of new networking scenarios, such as Internet of Things and data center scenarios. Inspired by Key-Based Routing (KBR) solutions which, in conjunction with Distributed Hash Tables, have offered a way of providing content-sharing solutions in overlay networks on the top of the Internet for years now, we propose OFC-KBR (OpenFlow Compatible Key-Based Routing) solution. OFC-KBR is a key-based routing solution directly implemented at network layer that makes use of the potential of Software Defined Networking. In this solution, end-points are identified by virtual identifiers. These virtual identifiers are obtained from a descriptive textual name, whose format is not fixed and can be defined depending on the requirements of the service that is going to use the proposed OFC-KBR solution. OFC-KBR is totally compatible with the current OpenFlow standard and can co-exist with other L2/L3 protocols. The proposal has been implemented and evaluated by simulation considering real topologies.

Keywords Distributed Hash Tables · Switching · Non-IP routing

✉ Pilar Manzanares-Lopez
pilar.manzanares@upct.es

Adrian Flores-de la Cruz
adrian.flores@upct.es

Juan Pedro Muñoz-Gea
juanp.gea@upct.es

Josemaria Malgosa-Sanahuja
josem.malgosa@upct.es

[1] Department of Information Technologies and Communications, Universidad Politecnica de Cartagena, Campus Muralla del Mar s/n, 30202 Cartagena, Spain

Published online: 14 November 2018

1 Introduction

The host-to-host/content/service communication instead of the host-to-host communication offered by traditional IP routing solutions has been demanded in the last few years. In this sense, the concept of Information-Centric Networking (ICN) [1] proposes the change from a host-centric paradigm to a content-centric network architecture. Under its umbrella, Named Data Networking (NDN) [2] is one of the most popular projects. NDN decouples contents from locations and offers content retrieval using two types of packets (Interest and Data packets). NDN builds an in-network caching architecture, where nodes maintain three data structures (Content Store, Pending Interest Table and Forwarding Information Base), and defines adequate routing mechanisms to respond to content searches.

The idea of decoupling end-points from locations without requiring a new network architecture is an increasing demand in the framework of new networking scenarios. Traditional routing protocols provided by the Internet Protocol allow each host to send data packets to other machines by knowing their IP addresses. This type of routing has been sufficient for most of the applications on the Internet. However, in recent years, applications developed in new network scenarios require a different type of routing. Cloud computing applications such as web caching solutions based on In-Memory Key-Value Storage (IMKVS) [3], data center applications based on the use of keys to identify data and users [4], or the Internet of Things (IoT) scenario where a great variety of devices are interconnected, require that routing tasks are associated to end-host names (or any other virtual identifier) instead of IP addresses.

Our proposal, OFC-KBR (OpenFlow Compatible Key-Based Routing), offers a routing solution in which end-points are identified by virtual identifiers (keys) and not just by IP addresses. In our implementation, the virtual identifier is obtained from the host name, but it is just a proposal. The definition of the host identifier could be changed depending on the application of OFC-KBR routing solution.

Unlike traditional key-based routing solutions that are defined at application level, the differentiating aspect of our proposal is the fact that OFC-KBR is completely implemented at network level, without the need of any additional infrastructure. This is possible thanks to the use of SDN (Software Defined Networking) architecture [5], which maintains a global view of the network and offers a wide control of the network devices. Using the well-known OpenFlow protocol [6], the network elements are configured to be able to route the packets adequately. An SDN controller (or a distributed set of controllers) obtains and inserts in the flow table of the network elements the required set of entries to offer this key-based routing service.

The fact that the proposed solution does not require any modification of the OpenFlow standard nor the OpenFlow-based network elements enables OFC-KBR to co-exist with traditional L2/L3 protocols. For example, flow table entries associated with the OFC-KBR solution can co-exist with flow table entries that allow network elements the routing based on standard IP or MAC addresses.

Packet classification is the key function of network devices in order to execute any routing protocol. Routers and switches operate by matching header fields of incoming packets against a set of rules and priorities. According to the matching

rule with the highest priority, a given action is performed on the packet, such as forwarding to a specific output interface, dropping the packet, or altering the headers or payload. Packet classification is also required to implement many other network functions, such as filtering, security, load-balancing and virtual networks.

Nowadays, the use of the 5-tuple ruleset (source IP address, destination IP address, source port number, destination port number, and protocol type) is not sufficient. Advanced network functions require a larger and more complex ruleset. For example, being able to express ranges of values in any of the matching fields is a desirable utility to allow the creation of advanced routing rules. In this regard, IP routers are Longest Prefix Match (LPM) compliant and therefore, it is not possible to specify some ranges without interfering the matching procedure.

Fortunately, SDN technology offers a powerful and flexible control of network devices, and OpenFlow switches offer a wide ruleset of 15 fields and the definition of different interconnected flow tables to perform packet classification. In this work, we implement a packet classification based on the definition of value ranges using the potential offered by OpenFlow.

The remainder of the paper is organized as follows. Sect. 2 presents some related works. Section 3 reviews the lookup mechanism of Chord and the prefix extension technique, two technologies closely connected with our proposal. In Sect. 4, the OFC-KBR solution is described in detail. The evaluation of the proposal is presented in Sect. 6. Section 7 suggests a useful enhancement of the OpenFlow standard to improve the performance of the solution. Finally, Sect. 8 concludes the paper.

2 Related Work

The concept of Key-Based Routing (KBR) was very common among the scientific community at the beginning of this century when structured P2P (peer-to-peer) networks were developed (Chord [7], Pastry [8], Tapestry [9] or Kademlia [10]). These networks offer a routing service at application level to send a message, with an associated key, to a single node, responsible for that key. In [11] the authors developed a common API with the ability needed to interact with the KBR service offered by any structured P2P network. These are as follows: *void route (key k, msg m)*, which routes a message, *m*, towards the responsible key node *k*; *void forward (key k, msg m, nodehandle nesthopnode)*, which is invoked from the structured P2P network layer of each node that forwards message *m* during its routing; *void deliver(key k, msg m)*, which is invoked from the structured P2P network layer of the node that is responsible for key *k* upon the arrival of message *m*.

Over the previous interface, the services offered by different kinds of applications can be implemented: Distributed Hash Tables (DHT) [12], Decentralized Object Location and Routing (DOLR) [13] and group anycast/multicast (CAST) [14]. The operations of these applications are implemented using the previous API to interact with the KBR service. For example, a DHT, which implements a simple store and retrieve functionality, where the value is always stored at the live overlay node(s) to which the key is mapped by the KBR layer, provides two operations: (1) *put*

key,value), and (2) *value = get(key)*. Both of them are implemented using the *void route (key k, msg m)* interface.

In recent years, several applications developed in new networking scenarios also use the KBR concept. For example, in cloud computing applications, there are web caching solutions based on In-Memory Key-Value Storage (IMKVS) [3]. The basic commands that can be executed in these applications are: *void set (key, value)*: ask a specific server to store a given value identified by a given key; *value get (key)*: retrieve the value identified by the given key stored in a specific server. In [3] authors propose a Two-Phase Load Balancer for IMKVS system. The proposed solution aimed to reduce the load on both servers and network in a scalable and easy to deploy way, using a NFV (Network Function Virtualization) architecture. The second phase index of the load balancer is based on the Chord network, but other structured P2P networks could be used. Beyond this, many applications running in data center networks use keys to identify data or users, such as MapReduce [15], Amazon Dynamo [16], Microsoft Dryad [17], and Facebook Photos [18]. CamCube [19] is a prototype built by Microsoft to implement KBR in data centers. In addition to this, in [4] authors present a new data center network architecture, named Space Shuffle (S2), which also supports KBR. Finally, the Internet of Things scenario, where a great variety of devices are interconnected, also uses KBR to alleviate overhead caused by flooding [20].

The idea of using the OpenFlow protocol in the context of information-centric networks has been applied in previous works. For example, in [21], the Openflow protocol is used to manage a sensor information network. There, based on context similarity, sensors are grouped into clusters, even if they are physically distant. Each cluster has its own SDN controller that performs the local processing for the cluster. Besides this, the local controllers are organized in the top-tier overlay in a Chord network and they form a logical controller for the entire network. In this work, the controllers (called sinks) are the destination point of the packet flows. Therefore, the proposal offers a specific solution to implement the sending of messages from a large group of sensors towards a certain sink. However, this solution is not suitable to implement solutions that require named-based routing service.

Information-Centric Networking (ICN) is a concept that relies on location-independent naming of data to support data movement and replication [22]. This architecture requires a scalable name resolution system (NRS) to translate the object IDs into network addresses. Data are then accessed through these addresses. An NRS architecture can be a tree, a DHT or hybrid, and it is composed by NRS servers. In Tree-based NRS the root performing global name resolution can become a bottleneck. To alleviate this bottleneck, some proposals build a DHT-based system at the root level. For example, the MDHT system [23] is a heterogeneous DHT that consists of multiple separate resolution domains that can each use a different structured P2P network internally such as Chord or Pastry.

The closest proposal to our work is VIRO (Virtual Id ROuting) [24]. The key idea behind VIRO is the introduction of a topology-aware, structured virtual id (vid) space onto which both physical identifiers (e.g., Ethernet MAC addresses) as well as higher layer addresses/names (e.g., IPv4/IPv6) are mapped. Taking advantage of such a topology-aware and structured vid space, VIRO employs a KBR algorithm

to build routing tables, look up objects (names, addresses, vids, etc.) and forward packets.

VIRO and our proposal OFC-KBR differ in some aspects that affect significantly the implementation of the systems. As it is detailed throughout the paper, the distributed name resolution mechanism and the forwarding procedure of OFC-KBR do not require any additional infrastructure nor any modification of the OpenFlow-based network elements. In addition, OFC-KBR solution has been implemented without the need of any modification of the OpenFlow protocol.

By contrast, to be able to implement the address-name resolution defined by VIRO, each host-node needs to maintain a local table that maps the physical identifiers (pid) to virtual identifiers (vid). On the other hand, VIRO implementation requires that each VIRO node implements a local SDN controller to realize VIRO packets forwarding functions in addition to a remote SDN controller responsible for the management plane of VIRO.

VIRO forwarding is done by using the destination vid (via vid prefix matching) and a forwarding directive. For that reason, they have to redefine the structure of the standard Ethernet frame to be able to include the vids and forwarding directive fields in the frame. The standardized matching operations, header fields and the allowable actions defined by OpenFlow protocol are tied to Ethernet/IP/TCP protocol. Therefore, the forwarding of VIRO, that requires to match source-vids, destination-vids and fd, cannot be implemented using the standard match-action functions of OpenFlow. Due to the fact that VIRO has been implemented using Open vSwitch (OVS) [25], the authors needed to modify and extend the match and the actions of the Open vSwitch user and kernel spaces to be able to implement their solution.

As said before, unlike VIRO, our proposed OFC-KBR solution is fully compatible with the current OpenFlow standard. Its implementation has not required any modification or extension of the OpenFlow protocol nor OpenFlow OVSs. In Sect. 7 we describe how a simple extension of the set of actions allowed by the OpenFlow standard would help to reduce the number of flow table entries in the network devices. However, this OpenFlow modification is not not mandatory for the implementation of OFC-KBR.

According to OpenFlow standard, values of matching fields of a flow entry can only be wildcarded (match any value) and in some cases bitmasked. As it will be described in detail in the following sections, our proposal is based on the obtaining of numerical intervals that will be used to configure adequately the routing tables of network elements. However, the obtained value ranges cannot always be expressed by bitmasked values.

To solve this problem, our proposal uses a range representation technology known as prefix expansion [26]. The keystone of this representation is turning each range into a set of subranges that can be bitmasked and, therefore, that can be included in an OpenFlow-based flow table by a flow entry. Applying this representation, the worst-case expansion ratio with ranges whose endpoints are w-bits numbers is 2w-2 entries per range.

In terms of flow entries, a more efficient solution to implement range search in OpenFlow switches is proposed in Orange [27], an OpenFlow adaptation of PIDR (Point Intersection Disjoint Ranges) [28]. PIDR is an algorithm for TCAMs

(Ternary Content-Addressable Memories) that can be easily implemented in hardware to solve the search operation problem for disjoint ranges. Most of the functionality of Orange can be implemented using OpenFlow tables but it also depends on the use of a comparison table. The main drawback of Orange is the fact that this comparison table is not implemented using an OpenFlow table. In fact, what we propose in this work is a way to implement comparison tables just by using OpenFlow tables, because a direct way to compare two values is by using ranges.

3 Review of Required Technologies

In this section, we present a review of Chord protocol [7] and prefix expansion technology [26]. Both existing technologies have been used in the definition of our proposal.

3.1 Lookup Mechanism of Chord

The basic principle of DHT protocols is the association between nodes and objects, and the construction of distributed routing structures to locate the objects efficiently.

In Chord, nodes and object identifiers are located in a circle formed by 2^m values ($id \in [0, 2^m - 1]$). Each node is responsible for a set of keys. In particular, a key k is assigned to the node whose identifier is equal to k, or, if it does not exist, it is assigned to the first node clockwise k. That node is called the key successor. For the routing strategy, each node n maintains its predecessor (the counterclockwise node in the ring), its successor (the clockwise node in the ring) and a list of extra nodes called finger table. The finger table is composed of m entries, where i-th table entry associates key $n + 2^i$ with the key successor.

When node n searches for a key k, it checks if k is located between itself and its successor. If so, the successor is the searched node. Otherwise, it checks its finger table from bottom to top to locate the immediate predecessor of k. Then, node n requires that node to deal with the lookup. This procedure is recursively done by different nodes until the node responsible of key k is found. The number of nodes that must be contacted to find a successor of a given key is $O(log_2 N)$, where N is the current number of active nodes. Figure 1 shows a lookup example in a Chord system, where $m = 6$ bits.

3.2 Range Representation Based on Prefix Expansion

OpenFlow specification 1.3 [29] defines that switches must support the match fields listed in Table 1. Some of them can be arbitrary bitmasked. However, OpenFlow does not define the assignment of range values that cannot be expressed just by a bitmasked range.

To solve this problem, the range representation technology known as prefix expansion [26] could be used. The keystone of this representation is turning each range that cannot be expressed by a bitmasked value into a set of subranges that

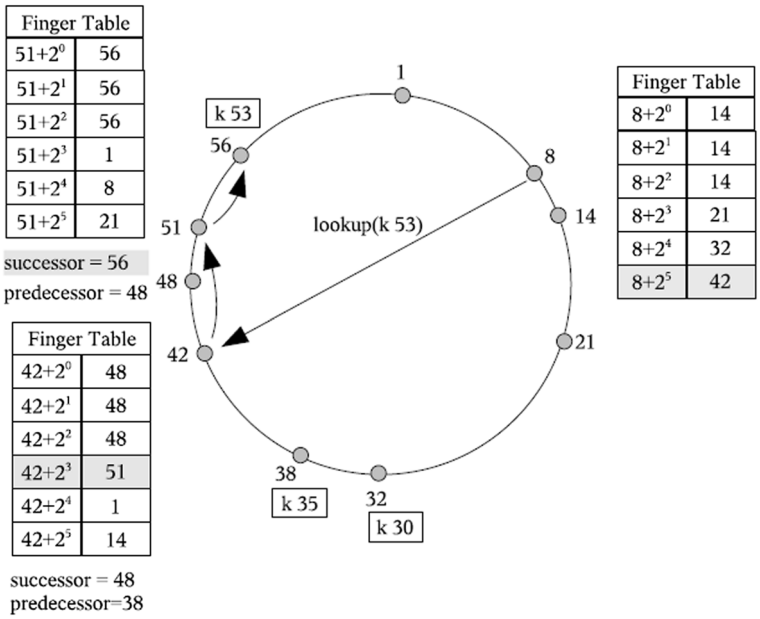

Fig. 1 A lookup example in a Chord system, where m = 6 bits. Node 8 is looking for an object whose key-identifier is 53

Table 1 Match fields. From OpenFlow 1.3 specification [29]

Field	Bits	Description
OXM_OF_IN_PORT	32	Ingress port
OXM_OF_ETH_DST	48	Ethernet destination port. Can arbitrary bitmask
OXM_OF_ETH_SRC	48	Ethernet source port. Can arbitrary bitmask
OXM_OF_ETH_TYPE	16	Ethernet type of the OpenFlow packet payload, after VLAN tags
OXM_OF_IP_PROTO	8	IPv4 or IPv6 protocol number
OXM_OF_IPV4_DST	32	IPv4 destination address. Can be subnet mask or arbitrary bitmask
OXM_OF_IPV4_SRC	32	IPv4 source address. Can be subnet mask or arbitrary bitmask
OXM_OF_IPV6_DST	128	IPv6 destination address. Can be subnet mask or arbitrary bitmask
OXM_OF_IPV6_SRC	128	IPv6 source address. Can be subnet mask or arbitrary bitmask
OXM_OF_TCP_SRC	16	TCP source port
OXM_OF_TCP_DST	16	TCP destination port
OXM_OF_UDP_SRC	16	UDP source port
OXM_OF_UDP_DST	16	UDP destination port

can be bitmasked and, therefore, that can be included in an OpenFlow-based flow table by a flow entry.

The algorithm to obtain the subranges from a certain range is described in Table 2. It consists in obtaining the Longest Common Prefix (LCP) of the range, and the extended LCPs (ELCPs) named 0-ELCP and 1-ELCP. The ELCPs are

Table 2 Algorithm to obtain the bitmasked ranges using prefix expansion [26]

```
01: Input: [s,t] = [s_0 s_1 s_2 .... s_{k-1}, t_0 t_1 t_2 .... t_{k-1}]
02: Output: a set of valid ranges
03: function bitmasked_ranges(s,t):
04:    obtain the LCP=p_0 p_1 .... p_{p-1}
05:    using 0-ELPC, determine the subrange [p_0 p_1 .... p_{p-1} 0 s_{p+1} .... s_{k-1}, p_0 p_1 .... p_{p-1} 01...1]
06:     if the subrange can be bitmasked:
07:       valid subrange identified
08:     else:
09:       new subrange [s',t']=[p_0 p_1 .... p_{p-1} 0 s_{p+1} .... s_{k-1}, p_0 p_1 .... p_{p-1} 01....1]
10:       bitmasked_ranges(s',t')
11:    using 1-ELPC, determine the subrange [p_0 p_1 .... p_{p-1} 10....0, p_0 p_1 .... p_{p-1} 1 t_{p+1} .... t_{k-1}]
12:     if the subrange can be bitmasked:
13:       valid subrange identified
14:     else:
15:       new subrange [s',t']=[p_0 p_1 .... p_{p-1} 10....0, p_0 p_1 .... p_{p-1} 1 t_{p+1} .... t_{k-1}]
08:       bitmasked_ranges(s',t')
```

calculated from the LCP by adding one more bit. For a LCP P, P0 is 0-ELCP and P1 is 1-ELCP.

Then, by using the corresponding 0-ELCP, 1-ELCP and the start and end values, the original range is split into two subranges. For each subrange, the process is repeated until a set of bitmasked subranges are obtained.

As an example, the range $[32, 54] = [00100000, 00110110]$ can be expressed by the following set of bitmasked ranges: $[32, 47] = [0010 ****]$, $[48, 51] = [001100 **]$, $[52, 53] = [0011010 *]$ and $[54] = [00110110]$. Consequently, the original range could be included in a flow table by the insertion of four bitmasked flow entries.

4 OpenFlow Compatible Key-Based Routing Protocol

In this paper, we propose OFC-KBR (OpenFlow Compatible Key-Based Routing), a key-based routing solution directly implemented at network layer that makes use of the potential of Software Defined Networking. Unlike other solutions, there is no need to use caches or additional databases on the network elements to store information about end-hosts. In addition, all the required functionality is fully compatible with the current SDN OpenFlow standards, and it is not required to modify nor extend the OpenFlow switch architecture. Finally, OFC-KBR protocol can coexist with traditional L2/L3 protocols.

Unlike traditional IP routing protocols that offer a host-to-host communication, where host and routers are identified by IP addresses, this solution is based on the definition of virtual identifiers (vids). Each network element, and each host, content, or service involved in the communication will be identified by a virtual identifier. To simplify we will refer to hosts, contents, and services as end-points.

4.1 Virtual Identifier Assignment

Virtual identifiers (vids) of network elements and end-points are defined as L-bits strings that comprise two parts of L/2 bits. In the case of a network element, the first L/2 bits are the result of applying a hash function to its OpenFlow datapath identifier and the last L/2 bits are set to 1. In the case of end-points, the first L/2 bits are the result of applying the hash function to the switch datapath to which the end-point is connected, and the last L/2 bits are the result of applying the hash function to its name. The definition of the name of an end-point is not fixed. It could be a flat or a hierarchical textual name, depending on the requirements of the service that is going to use the proposed OFC-KBR solution.

In our implementation of the OFC-KBR proposal, we have used the 32-bits of the IPv4 address field to contain the virtual identifier, which allows the use of OFC-KBR in networks of up to 2^{16} network elements. If larger networks were required, the 128-bits IPv6 address field could be used.

Figure 2 shows the physical topology of a SDN network and the corresponding virtual identifier assignment. For the sake of clarity and simplicity in representation, vids are 10 bits long. By way of example, we consider that end-points are hosts. In this way, the virtual identifier of host $h1$ is 1010100110. The first 5 bits would correspond to the hash value of the datapath identifier of switch $dp3$, and the last 5 bits would correspond to the hash value of the textual name associated to $h1$. The virtual identifier of switch $dp3$ is 1010111111. The first 5 bits would correspond to the hash value of the datapath identifier and the last 5 bits are set to 1.

4.2 Key-Based Routing Information

OFC-KBR is inspired by the lookup mechanism of DHT solutions. Although it could be considered any of the existing DHT solutions, we have chosen as a reference the Chord protocol [7] due to its simplicity and widespread use in research. A Chord node maintains its own table formed of L entries (finger table), a successor and a predecessor, and uses this information to perform the lookup process as summarized in Sect. 3.1.

In our work, we have taken inspiration from the lookup mechanism of Chord (a P2P overlay solution) to implement an in-network routing solution. However, we do not implement a finger table nor the Chord lookup procedure directly. A design requirement of our proposal determines that OFC-KBR must be totally compatible with the current architecture of OpenFlow switches. Thus, network elements will only make use of the OpenFlow tables to perform the routing decisions. Therefore, it is necessary to translate the finger table content that would be defined in node by Chord, into a set of flow table entries that will be stored in a network element, and to convert the Chord lookup procedure into an OpenFlow entry matching-based operation. The match fields of each entry determine a range of virtual identifiers, and the action fields indicate what to do when a packet that is destined to a value included in this range is received.

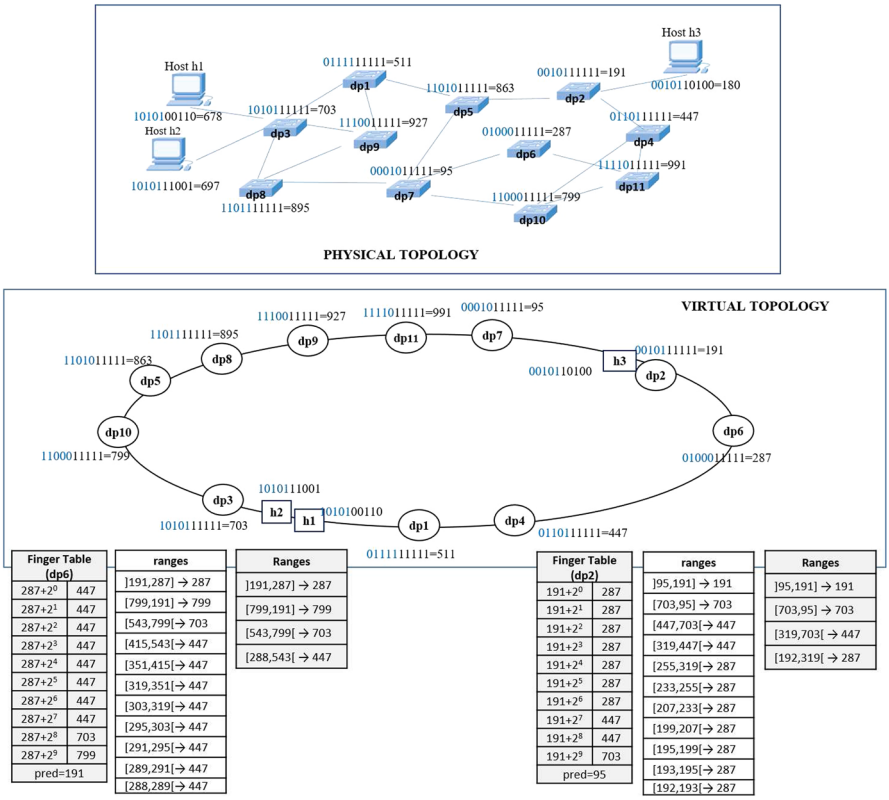

Fig. 2 The upper part of the figure shows the physical topology of an SDN network and the corresponding vid assignment, where vids are 10 bits long. The intermediate part of the figure shows the corresponding virtual topology according to OFC-KBR. Boxes in the lower part of the figure show, as an example, the obtained ranges corresponding to switches dp2 and dp6

The SDN controller will use its global knowledge of the network to perform this translation, and will configure the flow table of each network element accordingly. Figure 3 details the set of ranges that are obtained for a network element whose virtual identifier is n and whose finger table would be the left part of the figure. $ft(n+2^i)$ refers to the value of the finger table entry $n+2^i$, that is, the vid of the network element responsible of value $n+2^i$, and $\rightarrow x$ corresponds to the action *send to switch x*.

As an example, turning again to Fig. 2, the boxes at the bottom of the figure show the ranges and corresponding actions obtained for network elements $dp2$ ($vid = 0010111111b = 191d$) and $dp6$ ($vid = 0100011111b = 287d$). As can be seen, some different ranges involve the same action. Depending on the obtained ranges and actions, they could be merged. In the example, the obtained ranges can be reduced to just 4.

Once the equivalent ranges and the associated actions are obtained, the controller will instruct each network element to install the adequate flow entries. The current

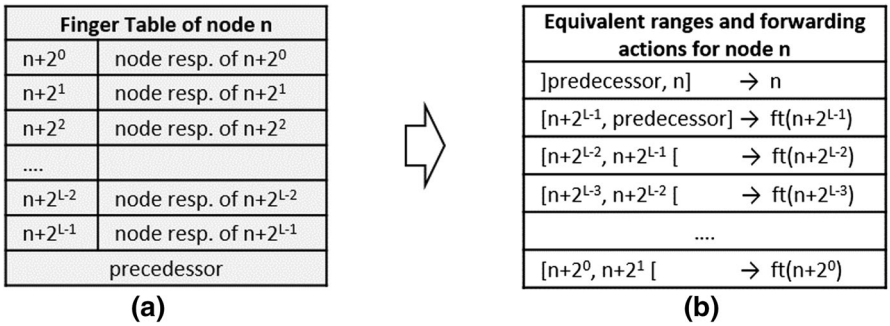

Fig. 3 Table (a) shows the finger table of a node. Table (b) shows the obtained ranges and the corresponding actions. $f.t.(n+2^i)$ refers to the value corresponding to finger table entry $n+2^i$, that is, the switch responsible of the vid $n+2^i$

implementation of OFC-KBR uses the IPv4 address fields to include the vid values. Consequently, the destination IPv4 field is the matching field of the flow entries that are used for routing tasks.

Due to the fact that some obtained ranges could not be expressed by just a bitmasked value, the prefix expansion algorithm (Sect. 3.2) will be used. This algorithm offers an efficient solution fully compatible with current OpenFlow switch architectures to express any obtained range as a limited set of bitmasked subranges, each of which will correspond to a flow entry.

As can be deduced from table (b) in Fig. 3, the minimum number of ranges obtained by the SDN controller for a certain network element is 2. This case happens when all the entries of the finger table of the node point to the succesor of the node. The maximum number of ranges for a switch is $L+1$. This case happens when each entry of the finger table of the node points to a different node.

In this work, we have developed a mathematical model that let us know the probability that the number of ranges obtained for a certain switch is k depending on the total number of network elements. We considered interesting to develop this study to be able to verify the feasibility of our proposal, and to be able to check if the number of ranges remains at limited and acceptable values. The mathematical model, which is detailed in "Appendix A", has been validated by simulation. In addition, it has been used to compare the results obtained in Sect. 6.

4.3 Virtual and Physical Paths

Location of network elements in the virtual topology depends on their virtual identifiers, and it is not linked to the physical topology. Therefore, virtual paths may do not be identical to physical paths.

Turning again to Fig. 2, as an example, consider the third range obtained for *dp*6 ([543, 799[→ 703). The action '*send toward 703 (dp3)*' has to be translated into an OpenFlow-compatible action. Concretely, the action should indicate the adequate output port to forward the packet. However, there is not a direct physical connection between *dp*6 and *dp*3.

A simple solution might be sending the packet through the output port determined by the optimal physical route to the destination. Due to the fact that the controller has a complete knowledge of the network topology, the optimal physical route can be obtained applying an adequate routing algorithm, such as Dijkstra. In the former example, the action associated to the range would be to forward to *dp*7, because the shortest route toward dp6 is through that switch. However, this solution is not ideal because loops may appear. After receiving the message, *dp*7 will use its own flow table to forward the packet. If the matching flow entry in *dp*7 determines that the message has to be sent to a network element whose shortest physical path passes through *dp*6, a loop is created.

To solve this problem, we propose the creation of a tunnel for the physical path associated to a virtual space hop using the MPLS technology [30], as described in Fig. 4. Each time the SDN controller inserts the flow entries corresponding to a certain range in a switch, it also inserts flow entries in all the switches forming the physical path between the switch and the destination of the virtual space hop. Actions of the flow entries installed on the source switch are the setting of an MPLS label into the packet and the forwarding through the adequate output port. On each intermediate switch except the last before the destination, the controller installs just a flow entry where the match fields are the packet type in order to detect MPLS packets and the MPLS label value. The action will be the forwarding throughout the adequate output port. Finally, on the last switch before the destination, the controller installs a flow entry whose matching fields are the packet type and the MPLS label value, and the actions will be two: first, to delete the MPLS label and then to forward the packet through the adequate output port.

5 Joining of End-Points and Distributed Name Resolution Mechanism

OFC-KBR protocol defines how network elements are able to route traffic towards the network element responsible of any virtual identifier. However, the objective of the proposal is to offer host/service/content-centric communication. Therefore, it is necessary to define a joining procedure for end-points and also, as it is going to be argued later, a name resolution service. Both mechanisms

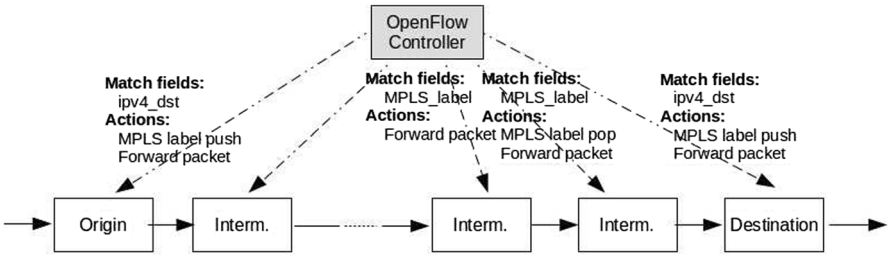

Fig. 4 Physical path implementation using MPLS technology

rely on the definition of the virtual identifiers of end-points and network elements, and make use of the DHT principles that substantiate OFC-KBR.

5.1 Joining Procedure

The joining of end-points to the network is solved by the definition of JOIN packets. In our implementation of the OFC-KBR solution, a JOIN packet is a packet whose source key is the virtual identifier of the joining end-point, and the destination key is the virtual identifier of the switch to which the end-point is connected. A joining end-point will send a JOIN packet to the directly connected switch.

Any network element stores in the flow table an entry that allows them to identify the joining packets, based on the source key field and destination key field. The action associated to this entry is the sending of the packet towards the controller by a Packet_In.

Once the controller receives the Packet_In, it will install the adequate flow table entries in certain network elements that will be necessary to implement the name resolution service, as detailed next.

5.2 Distributed Name Resolution Mechanism

A premise of our system is the fact that it is only known the name of the end-points, but not their virtual identifiers. This feature will be useful to adapt our routing solution to scenarios where hosts are mobile devices, because the name of the end-points do not depend on their physical location.

Therefore, it is required a mechanism that deals with the resolution of the name of end-points into their virtual identifiers. We define a distributed name resolution mechanism that makes use of flow table entries installed in the switches during the joining process by the controller.

Thus, after receiving a Packet_In containing a JOIN packet, the controller will install a flow entry in this switch that generated the Packet_In, and another flow entry in the switch that will be in charge of solving the name resolution. Using DHT terminology, this last switch will be the network element responsible of the virtual identifier formed by the last $L / 2$ bits of the joining end-point (that is, the hash name) and the rest of bits set to 0. In our implementation, based on Chord, this switch will be the network element whose virtual identifier is the next existing clockwise value in the virtual identifier space.

The flow table entry installed in the switch that generated the Packet_In enables the sending of any received message destined to the direcly connected end-point. The entry indicates the sending through the adequate output port. On the other hand, the flow table entry installed in the other switch implements the solving of the name resolution and instructs that switch as a forwarder towards the destination, by means of the corresponding actions.

5.3 Example of Joining and Name Resolution

This section illustrates the processes described above. Referring to the network topology shown in Figs. 2 and 5 describes the joining of hosts $h1$ and $h3$, and name resolution process associated to the sending of a data message from $h1$ to $h3$, and from $h3$ to $h1$.

Host $h1$ (vid = 1010100110) is connected to the switch $dp3$ (vid = 101010000). Therefore, both values are used to create the JOIN message. Thanks to the first flow entry stored in switch $dp3$, whose matching fields are src_ipv4 = 10101 ***** and dst_ipv4 = 1010111111, the JOIN message is identified and forwarded to the controller as a Packet_In message.

The controller installs a new flow entry in $dp3$ by means of a Flow_Mod message. Due to this entry, any packet received by $dp3$ and destined to $h1$ will be forwarded to $h1$ throughout the adequate port (in this case, it is p_{out}).

On the other hand, messages sent by any host towards $h1$ will be destined to 0011000000 (the concatenation of the hash of the name of $h1$ and 0s). Therefore, to implement the name resolution service, the controller installs a flow entry in $dp6$, the switch of the network responsible of that key. The flow table entry will

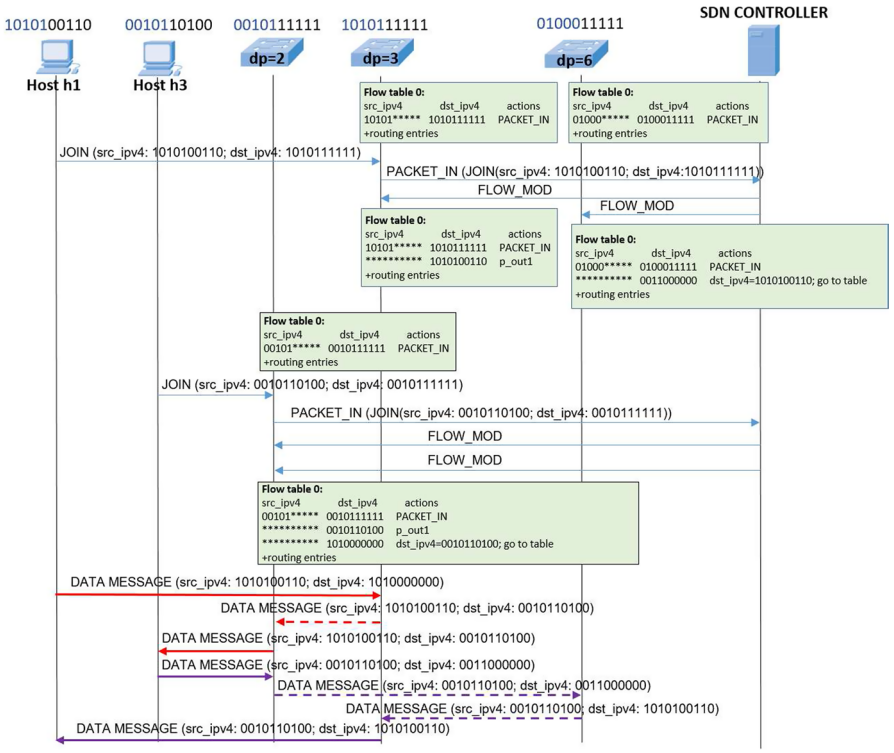

Fig. 5 This timeline describes the joining of $h1$ and $h3$, and the name resolution involved in the sending of messages from $h1$ to $h3$ and from $h3$ to $h1$

match any message destined to 0011000000 and will perform the translation of the received destination key to the virtual identifier of $h1$ (1010100110). Then, $d6$ will forward the packet toward the destination using its OFC-KBR routing entries.

Similarly, the joining of host $h3$ is shown. Next, the timeline describes the sending of a message from $h1$ to $h3$ (and from $h3$ to $h1$). In the first case, the destination key of the message is 10100000, that is the key obtained just from the name of $h3$. Using the OFC-KBR routing entries, the data message reaches the switch responsible of the solving the name resolution, in this case, itself. Then, the data message is forwarded using the OFC-KBR routing entries towards $h3$. In the second case, the destination key of the message is 0011000000, that is the key obtained just from the name of $h1$. Using the OFC-KBR routing entries, the data message reaches the switch responsible of the solving the name resolution, in this case, $dp6$. Finally, the data message is forwarded using the OFC-KBR routing entries towards $h1$.

6 Performance Evaluation

This section evaluates the performance of the proposed routing solution OFC-KBR. We conducted our simulations using Mininet [31], a testbed for Software Defined Networking. The solution has been implemented using the RYU controller [32]. We have used Mininet to generate three existing network topologies, from the Internet topology zoo [33]. The topologies Switch-L3 (39 nodes and 62 links), DFN (56 nodes and 87 links) and GTSCe (148 nodes and 192 links) are showed in Fig. 6a–c, respectively.

6.1 Finger Table Conversion into Ranges

OpenFlow Compatible Key-Based Routing (OFC-KBR) relies on the conversion of the Chord finger tables into intervals that will be used to fill the OpenFlow tables of the network nodes. In this section, we are going to evaluate this conversion. For each

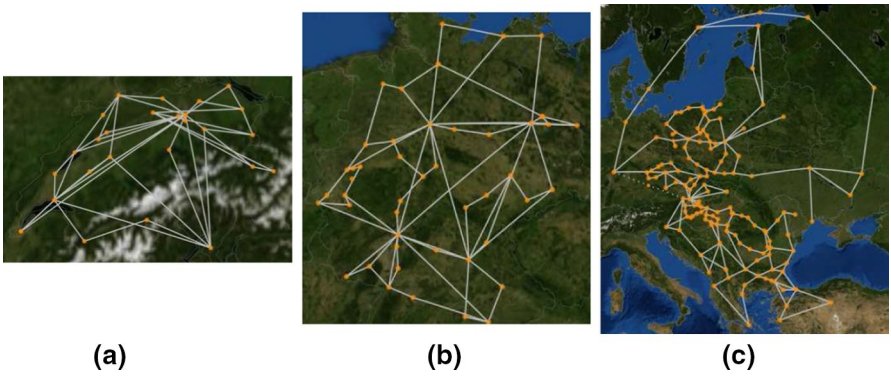

Fig. 6 Topologies. SwitchL3 (39 nodes and 62 links), Dfn (56 nodes and 87 links) and GtsCe (148 nodes and 192 links). **a** SwitchL3. **b** Dfn. **c** GtsCe

network topology, 10 simulations have been carried out. By definition, the virtual identifier of each node is obtained from its OpenFlow datapath identifier by applying a hash function. Each simulation has been carried out assigning a different set of OpenFlow datapath identifiers to the nodes.

Figure 7a shows the probability distribution of the number of ranges obtained for each node obtained from the simulations. Figure 7b shows the theoretical values obtained using the mathematical model described in "Appendix A". As expected, as the number of nodes in the topology increases, the chart moves to the right, giving rise to a higher mean number of entries for each node. As the number of nodes increases, the probability that the ranges obtained directly from the entries of the finger table can be merged is smaller, because the entries point to different nodes. In particular, the mean value of the number of ranges is 6.5743 for Switch-L3 topology (39 nodes), 7.1447 for DFN topology (56 nodes), and 8.5351 for GTSCe topology (148 nodes). These mean values are similar to $log_2(N)$, where N is the number of nodes.

6.2 OpenFlow Table Occupation

Each obtained interval, which associates a range of values with a forwarding action, should be used to directly insert an entry into the OpenFlow table of the corresponding switch. However, as explained in Sect. 3.2, this is not possible using the current OpenFlow standard. Instead, OFC-KBR makes use of the prefix expansion algorithm to obtain a limited number of bitmasked subranges from each range.

Figure 8a shows the mean number of bitmasked subranges corresponding to the obtained intervals after the conversion, for each node of the Switch-L3 topology. In the graph, it is also plotted the minimum and the maximum number of bitmasked subranges corresponding to the simulations. Similarly, Figs. 9a and 10a show the results for the DFN and GTSCe topologies, respectively. From the mean number of obtained ranges and the mean number of bitmasked subranges for each

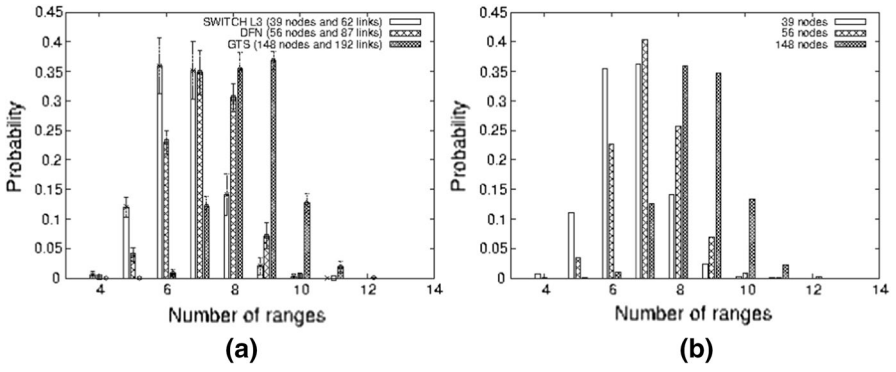

Fig. 7 Probability distribution of the number of ranges obtained for a node. L = 32 bits. **a** By simulation using Mininet [31]. **b** Using the mathematical model described in "Appendix A"

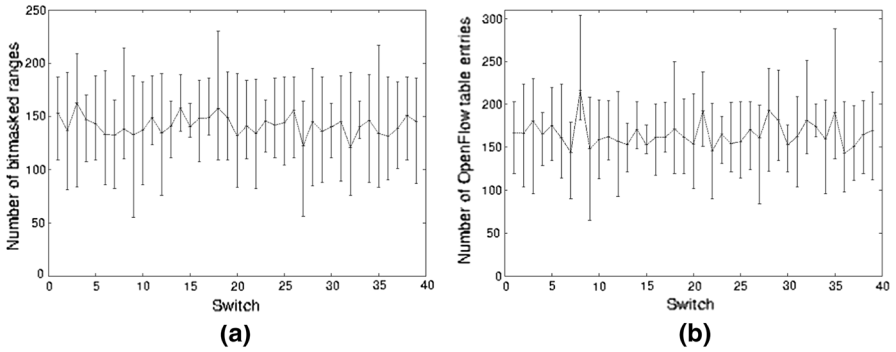

Fig. 8 SwitchL3 topology (39 nodes and 62 links). **a** Number of bitmasked ranges. **b** Number of OpenFlow table entries

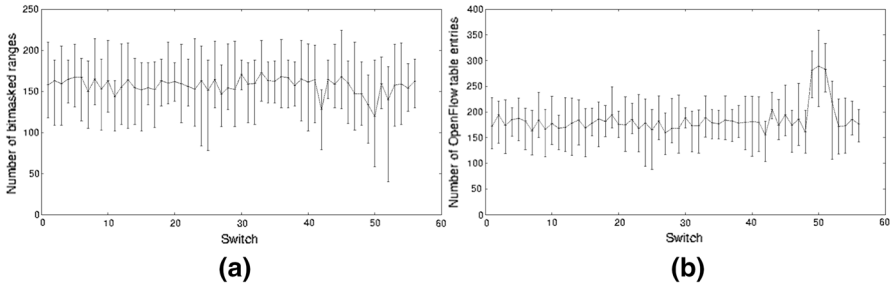

Fig. 9 DFN topology (56 nodes and 87 links). **a** Number of bitmasked ranges. **b** Number of OpenFlow table entries

node, it can be calculated the expansion factor as the ratio of both values. The obtained expansion factors corresponding to the topologies are 21.550, 22.005 and 22.603.

At this point, we can compare the obtained results with the number of entries required by Orange [27], an alternative solution to implement range search in OpenFlow switches, cited in Sect. 2. According to this work, the number of entries required to implement n ranges is $2n + 2L + 1$, where n is the number of ranges and L is the number of bits (in our case $L = 32$). Applying this expression, the mean number of entries required using Orange is $2 * \lceil 6.5743 \rceil + 2 * 32 + 1 = 79$ ranges for Switch-L3, 81 ranges for DFN and 83 ranges for GTSCe, respectively. However, the term $2L + 1$ corresponds to a comparison table that, as said before, cannot be implemented using an OpenFlow table. In fact, the implementation of a comparison table in OpenvSwitch is just what we have solved.

With respect to the results, we would like to underline that the obtained expansion factor is much lower than 62, the maximum expansion factor of the prefix expansion technique (that is, $2L - 2$).

Finally, we are going to evaluate the final occupation of the OpenFlow tables. As can be seen in Figs. 8b, 9b and 10b, the number of OpenFlow entries in each

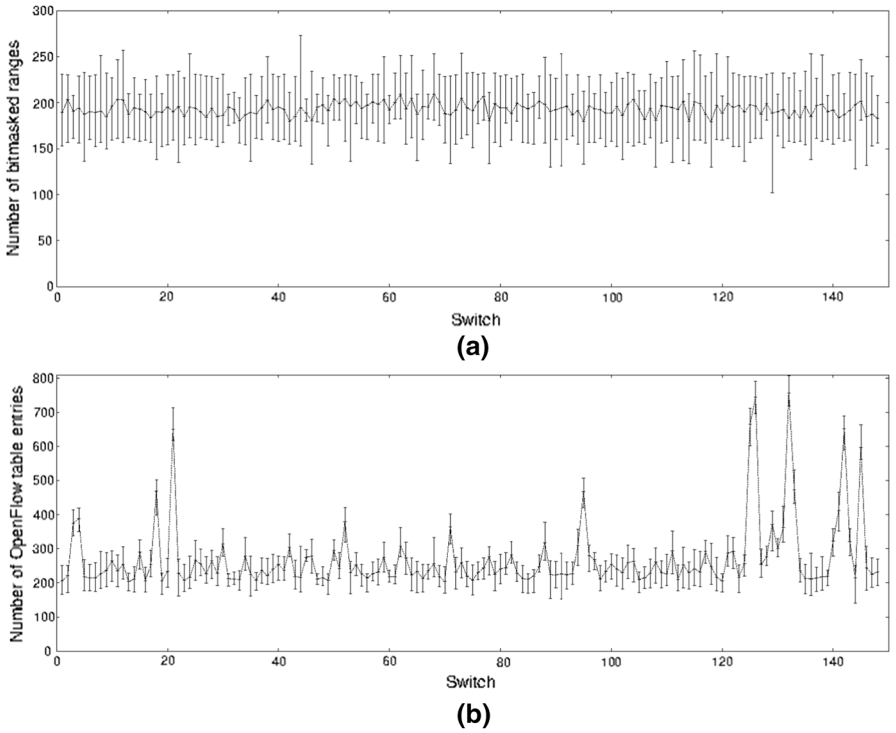

Fig. 10 GtsCe topology (148 nodes and 192 links). **a** Number of bitmasked ranges. **b** Number of OpenFlow table entries

switch is higher than the number of bitmasked ranges. This is due to the fact of transforming the virtual paths into physical paths, as described in Sect. 4.3. As expected, the increment between the number of ranges and the number of entries depends on the position and the connectivity of the switch. For example, in Fig. 8b switch 8 shows the most remarkable increment. Analyzing the Switch-L3 topology, this node corresponds to Zurich ETH, that has the highest connectivity degree. It is likely that the physical path corresponding to a virtual path goes through this point. The same behavior can be observed in the other two topologies. In any case, the obtained results are acceptable considering that a routing table can support from 750 to few thousands of rules. Nonetheless, in Sect. 7 we describe some proposals that would allow to reduce the number of required entries, if the OpenFlow technology evolved adequately.

6.3 Network Performance

To evaluate the network performance we will consider the routing optimization and the network throughput, as described in the following sections.

6.3.1 Routing Optimization

The routing optimization is measured by means of the routing stretch. Routing stretch (RS) is defined as the ratio of the number of physical hops using OFC-KBR and the number of physical hops using the Shortest Path (SP) between any two switches i and j, as defined in Eq. (1):

$$RS_{ij} = \frac{L_{ij}^{OFC-KBR}}{L_{ij}^{SP}} \tag{1}$$

Considering that the maximum path length of Chord is $log_2(N)$, the average of the maximum routing stretch can be obtained as follows:

$$\overline{RS_{max}} = \frac{log_2(N) * \overline{L^{SP}}}{\overline{L^{SP}}} = log_2(N) \tag{2}$$

First row of Table 3 shows the average routing stretch corresponding to the simulations. It can be seen that the obtained values are always higher than 1 (the ideal value). As in most of the key-based applications, this is the price to be paid for using a DHT solution. However, the RS is limited by $log_2(N)$.

6.3.2 Routing Throughput

The stability of the proposed solution and also the network throughput depends directly on the performance of the switches. To achieve a high network throughput, switches have to route incoming packets as fast as possible. There are two factors that have a significant influence on this performance: the switching architecture and the lookup delay. However, as far as our proposal concerns, only the lookup delay has to be taken into account.

The lookup delay can be defined as the time that a switch takes to find a match entry in the flow table. As hardware-based Ethernet switches and IP routers, OpenFlow switches take advantage of the TCAM technology to fetch a wildcard entry in the only one clock cycle, which is the minimum possible. In fact, a TCAM memory works as a cache of the flow table storing the most recently used entries, and updates the list when a mismatch occurs.

In a network virtualized (NV) scenario, the lookup process has to be done necessarily by software. In this case, Open vSwitch (ovs) is the most outstanding solution,

Table 3 Routing stretchs

	Switch-L3	DFN	GTSCe
$\overline{RS}$	2.748	2.9278	4.2038
$\overline{RS_{max}}$	5.2854	5.8074	7.2095

since it has been designed to minimize the lookup delay [34]. To minimize the lookup delay, an Open vSwitch has a specialized kernel module that it is defined as kernel-based datapath. When a packet arrives on a port, the kernel module processes it by extracting its flow key and looking it up in the flow table. If there is a matching flow, it executes the associated actions. If there is no match, it queues the packet to userspace for processing. In this process, the kernel module caches an entry handle further packets of the same type. This kernel module speeds up the lookup process by means of using a tuple space search classifier and flow caches [25]. The throughput and latency of OpenvSwitch have been accurately studied in [35], showing that Open vSwitch is the best option to reach high network throughput in NV scenarios.

As described above, as traffic is generated by hosts and the network elements forward the packets, entries are cached in the kernel module of the OpenvSwitches. By way of example, we are going to evaluate the kernel memory consumption by the following experiment in the implemented OFC-KBR system. We consider that there is a host connected to each switch. Each host randomly selects a destination among the rest of hosts in the network and then transmits a packet flow of 8 packets/second for 30 s. Figure 11 shows the results for SwitchL3 and DFN topologies. As can be seen, although the number of OpenFlow table entries is higher (see Figs. 8, 9), the number of entries that are used and, consequently are cached is substantially lower.

7 Proposed Expansion of OpenFlow Standard to Improve the OFC-KBR Performance

A key and distinguishing feature of OFC-KBR solution is the fact that the proposal is totally compatible with the current version of the OpenFlow standard. This fact determines and conditions some performance results such as the occupation of the OpenFlow tables. In this section, we are going to describe how this parameter could be improved if the OpenFlow standard evolved adequately allowing some simple additional actions.

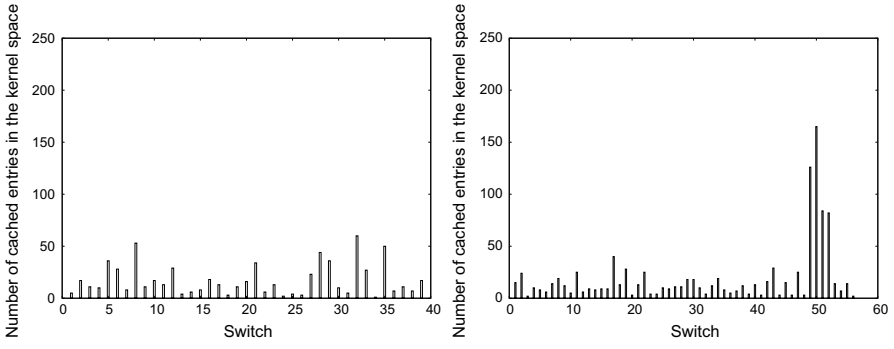

Fig. 11 Number of cached entries in the kernel space. As commented in Sect. 6.2, the increase of entries of switches 49 to 52 with respect to the rest of switches of the DFN topology is due to the position and the high connectivity degree of these switches. **a** SwitchL3 topology. **b** DFN topology

The relatively high number of entries of the OpenFlow tables of switches is caused by the technical impossibility of expressing most of the obtained ranges by just a bitmasked value. However, this problem is solved when the virtual identifier of the switch is 0. As can be deduced from table-(b) shown in Fig. 3, if $n = 0$, the last $(L - 1)$ obtained ranges can always be expressed by a bitmasked value each.

Using this feature, the number of entries in the OpenFlow tables could be reduced if, once the controller obtains the ranges and the forwarding actions for a switch, it makes a translation of the ranges, by subtracting the vid of the switch to each range limit. This subtraction would be equivalent to the shifting of the node (and range values) until the vid of the switch that matches the initial value 0. The shifting or subtraction is applied to the ranges but not to the actions.

If the ranges that are going to be used by a switch to forward a message are obtained as described, when the switch receives a message to be routed toward a destination, it will have to subtract its vid to the value of the destination before finding out the adequate range that determines the actions to be applied. Once the adequate flow entry matches, the associated actions will include adding the vid value that was subtracted. Each switch belonging to the logical path will do the same.

To be able to implement this proposal, which reduces the number of flow entries significantly, it should be necessary to modify the OpenFlow protocol to broaden the set of actions that can be associated to a flow entry. In particular, the set of new required actions should allow the modification of the source or destination IP fields by means of the adding or subtracting a certain value that should be included in the message next to the action as a parameter.

Some parts of the network could be optical and, at the same time, compliant with the Software Defined Optics (SDO) paradigm. In this case, there are lots of OpenFlow expansions available to SDO [36]. However, most of them are focused on managing the circuit switching mode, where the management is done by tuning transceivers and selecting wavelength but not by setting values in a flow table. Therefore, our proposed expansion should be applied in those points where the optical nodes perform packet -or burst of packets- switching rather than circuit switching. This mainly happens in the optical access network points, where the traffic granularity is still affordable for processing at optical rates. Fortunately, some of the solutions proposed in the literature for optical packet switching [37], are managed throughout the standard OpenFlow protocol and therefore, they are ready to incorporate our expansion directly.

8 Conclusions

In this paper, we have proposed a non-IP key-based routing mechanism that can co-exist with L2/L3 protocols. Among its main objective, the proposed OFC-KBR solution wants to extend SDN networks to non-host centric paradigm. To reach this objective, we have proposed a routing protocol inspired by KBR-DHT solutions, widely deployed in overlay networks. A KBR-DHT solution is a distributed system that stores (key, value) pairs, and where any participating node can efficiently retrieve the value associated with a given key.

In order to adapt this kind of mechanisms to be executed in a network level implementation, it is necessary that switches are able to make routing decisions based on numerical ranges. However, current OpenFlow switches are only able to make decisions based on matching and bitmasking. In this paper, it has been proposed an OpenFlow compatible mechanism to define ranges using several bitmasks, making it possible the implementation of the Chord routing mechanism.

We have evaluated the performance of our proposal in different topologies from real networks, showing that the implementation is viable. Alternatives are also proposed that reduce the size of the routing tables at the cost of modifications to the form of the OpenFlow switches.

Acknowledgements This research has been supported by the AEI/FEDER, UE Project Grant TEC2016-76465-C2-1-R (AIM). Adrian Flores de la Cruz also thanks the Spanish Ministry of Economy, Industry and Competitiveness for the FPI (BES-2014-069097) pre-doctoral fellowship.

Appendix A

Mathematical Model

Let us suppose a Chord network of size 2^L, and composed of N nodes. Without loss of generality, it is assumed that the vid assigned to one of the nodes is 0. We are interesting in calculating the probability distribution of the number of ranges obtained from the finger table of a node, assuming that the rest of the $N-1$ nodes are randomly distributed in the vid space $[0, 2^L - 1]$.

By definition, the finger table of a node is made up of L entries and the predecessor. The information of the finger table can be used to obtain $L+1$ ranges and the corresponding actions. The entry i associates the range $[2^i, 2^{i+1}[$ with first node clockwise 2^i in the virtual space. If more than one consecutive entries of the finger table point to the same node, these ranges can be merged into a single range. Under these assumptions, we are going to obtain the probability that the number of different ranges obtained for a node is k, where $2 \leq k \leq L+1$ when the total number of nodes in the network is N.

From now on, the interval $[2^i, 2^{i+1}[$ defined from the finger table of node 0 (the reference node) before merging will be called the i-th interval. If the first node clockwise 2^i (that is, finger table value associated to this entry) is not within the ith interval, we define this i-th the interval as an empty interval. As can be seen, an i-th interval is an empty interval if any of the existing N nodes are located within the associated interval. On the other hand, an i-th interval is a non-empty interval if one or more than one of the N nodes are located within the interval. Thus, all the consecutive empty intervals will be merged with the following non-empty interval. As a conclusion, if there are $k-1$ non-empty intervals and the rest are empty intervals, then, the finger table will produce k different ranges. As shown in Fig. 3, the range $]predecessor, n]$ that points to the own node will always be present.

Therefore, there will be k different ranges if the distribution of the N nodes in the virtual space allows node 0 to have $k-1$ non-empty intervals and $l-(k-1)$

empty intervals. However, due to the fact that each i-th interval has its own capacity (2^i), not all the $k-1$ combinations of $k-1$ non-empty intervals are valid. In these not valid cases, the total capacity is less than $N-1$, and it is not possible to distribute the $N-1$ nodes among them.

As an example, consider $l=6$, $N=10$, and $k=3$. There are 15 different combinations of 2 non-empty intervals and 4 empty intervals, but some of them are not valid. For example, if the non-empty intervals are the 0-th and 1-st intervals, the capacity is $2^0+2^1=3$, not enough to locate $N-1=9$ nodes. The same happens with the combinations 0, 2, 0, 3, 0, 4, 0, 5, and 1, 2. In the end, only 9 out of 15 combinations have enough capacity to locate N-1 nodes: 1, 3, 1, 4, 1, 5, 2, 3, 2, 4, 2, 5, 3, 4, 3, 5, and 4, 5.

Now, let us consider a valid combination of intervals of the above example (2, 3). Both intervals have the capacity to locate up to $2^2+2^3=12$ nodes. Therefore, there are 4 possibilities to distribute the $N-1=9$ nodes within the intervals: $n_2=1, n_3=8$, $n_2=2, n_3=7$, $n_2=3, n_3=6$ and $n_2=4, n_3=5$.

Consequently, among all the valid combinations of intervals, it should also be considered those that meet the following equation:

$$\Gamma \in i,j,\ldots,k | n_i+n_j+\cdots+n_k=N-1 \tag{3}$$

where n_i is the number of nodes in the interval i-th. Thus, the probability that the finger table of a node causes k different ranges is:

$$P(k)=\sum_{\Gamma} P(n_1)P(n_2)P(n_{k-1}) \tag{4}$$

where $P(n_i)$ is the probability that n_i of the total number of nodes $N-1$ are located in the interval i-th (with capacity to hold up to l_i nodes).

Finally, the probability can be easily obtained using combinatorial numbers as follows:

$$P(k)=\sum_{\Gamma} \frac{\binom{l_1}{n_1}\binom{l_2}{\cdots}\binom{l_{k-1}}{n_{k-1}}}{\binom{2^l-1}{N-1}} \tag{5}$$

Validation

The mathematical model has been validated using a simulation tool programmed in C language. The simulator considers the reference node (node 0) and randomly selects other $N-1$ nodes in a virtual of 2^L-1 values $(]0,2^L-1])$. Then, the simulator calculates the number of different ranges associated to node 0 (the reference node) accordingly to the procedure proposed in OFC-KBR solution, taking into account the existing nodes. This simulation is repeated 10^6 times in order to reach steady state probabilities.

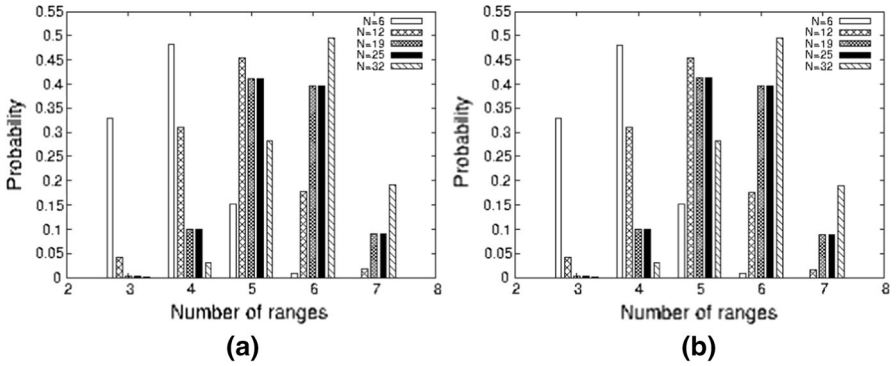

Fig. 12 Virtual space=$[0, 2^6 - 1]$ and N = 6, 12, 19, 25 and 32, corresponding to 10%, 20%, 30%, 40% and 50%. **a** Probability distribution of the number of ranges obtained by simulation. **b** Probability distribution of the number of ranges obtained using the mathematical model

Figure 12a shows the probability distribution of the number of ranges associated to a node, obtained by simulation, when $L = 6$ (that is, a virtual space $= [0, 2^6 - 1]$, and the number of nodes is $N = 6, 12, 19, 25$ and 32 , which corresponds to 10%, 20%, 30%, 40% and 50% of the capacity of the network.

As can be seen, as the number of nodes (N) increases, the probability distribution moves to the right. As the number of nodes increases, the probability that the ranges obtained directly from the entries of the finger table can be merged is smaller, because the entries point to different nodes. In particular, in the present study, the mean number of ranges are 3.8, 4.8, 5.4, 5.8 and 6.2 for $N = 6, 12, 19, 25$ and 32, respectively.

On the other hand, Fig. 12b shows the probability distribution of the number of ranges associated to a node, obtained by the mathematical model. The exact match between both figures clearly validates the model.

References

1. Ahlgren, B., Dannewitz, C., Imbrenda, C., Kutscher, D., Ohlman, B.: A survey of information-centric networking. IEEE Commun. Mag. **50**(7), 26–36 (2012)
2. Zhang, L., Estrin, D., Burke, J., Jacobson, V., Thornton, J.D., Smetters, D.K., Zhang, B., Tsudik, G., Claffy, K., Krioukov, D., Massey, D., Papadopoulos, C., Abdelzaher, T., Wang, L., Crowley, P., Yeh, E.: Named data networking (NDN) project, PARC Technical Report NDN-0001 (2010)
3. Trajano, A.F.R., Fernandez, M.P.: Two-phase load balancing of in-memory key-value storages using network functions virtualization (NFV). J. Netw. Comput. Appl. **69**, 1–13 (2016)
4. Yu, Y., Qian, C.: Space shuffle: a scalable, flexible, and high-performance data center network. IEEE Trans. Parallel Distrib. Syst. **27**(11), 3351–3365 (2016)
5. McKeown, N.: Software-defined networking. In: INFOCOM Keynote Talk, vol. 17, no. 2, pp. 30–32 (2009)
6. McKeon, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., Turner, J.: OpenFlow: enabling innovation in campus networks. ACM SIGCOMM Comput. Commun. Rev. **38**(2), 69–74 (2008)

7. Stoica, I., Morris, R., Liben-Nowell, Karger, D.R., Kaashoek, M.F., Balakrishnan, H.: Chord: a scalable peer-to-peer lookup service for internet applications. IEEE ACM Trans. Netw. **11**(1), 17–32 (2003)
8. Rowstron, A., Druschel, P.: Pastry: scalable, distributed object location and routing for large-scale peer-to-peer systems. In: Proceedings of IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), pp. 329–350 (2001)
9. Zhao, B.Y., Huang, L., Stribling, J., Rhea, S.C., Joseph, A.D., Kubiatowicz, J.: Tapestry: a resilient global-scale overlay for service deployment. IEEE J. Sel. Areas Commun. **22**(1), 41–53 (2004)
10. Maymounkov, P., Mazires, D.: Kademlia: A peer-to-peer information system based on the XOR metric. In: Proceedings of the First International Workshop on Peer-to-Peer Systems, pp. 53-65 (2002)
11. Dabek, F., Zhao, B., Druschel, P., Kubiatowicz, J., Stoica, I.: Towards a common API for structured peer-to-peer overlays, In: Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS03), pp. 33–44 (2003)
12. Dabek, F., Kaashoek, M.F., Karger, D., Morris, R., Stoica, I.: Wide-area cooperative storage with CFS. In: Proceedings of the 8th ACM Symposium on Operating Systems Principles (SOSP), pp. 202–215 (2001)
13. Hildrum, K., Kubiatowicz, J.D., Rao, S., Zhao, B.Y.: Distributed object location in a dynamic network. In: Proceedings of the 4th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA), pp. 41–52 (2002)
14. Castro, M., Druschel, P., Kermarrec, A.M., Rowstron, A.I.T.: Scribe: a large-scale and decentralized application-level multicast infrastructure. IEEE J. Sel. Areas Commun. **20**(8), 1489–1499 (2006)
15. Dean, J., Ghemawat, S.: Mapreduce: simplified data processing on large clusters. Commun. ACM **51**(1), 107–113 (2008)
16. DeCandia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., Sivasubramanian, S., Vosshall, P., Vogels, W.: Dynamo: Amazons highly available key-value store. ACM SIGOPS Oper. Syst. Rev. **41**(6), 205–220 (2007)
17. Isard, M., Budiu, M., Yu, Y., Birrell, A., Fetterly, D.: Dryad: distributed data-parallel programs from sequential building blocks. In: Proceedings of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems, pp. 59–72 (2007)
18. Beaver, D., Kumar, S., Li, H.C., Sobel, J., Vajgel, P.: Finding a needle in haystack: Facebooks photo storage. In: Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation, pp. 1–8 (2010)
19. Abu-Libdeh, H., Costa, P., Rowstron, A., OShea, G., Donnelly, A.: Symbiotic routing in future data centers. ACM SIGCOMM Comput. Commun. Rev. **41**(4), 51–62 (2011)
20. Sankaran, S., Sridhar, R.: Modeling and analysis of routing in IoT networks. In: Proceedings of the International Conference on Computing and Network Communications, pp. 649–655 (2015)
21. Rahmani, R., Rahman, H., Kanter, T.: On performance of logical-clustering of flow-sensors. Int. J. Comput. Sci. Issues **10**(5), 1–13 (2013)
22. Xylomenos, G., Ververidis, C.N., Siris, V.A., Fotiou, N., Tsilopoulos, C., Vasilakos, X., Katsaros, V., Polyzos, G.C.: A survey of information-centric networking research. IEEE Commun. Surv. Tutor. **16**(2), 1024–1049 (2014)
23. DAmbrosio, M., Dannewitz, C., Karl, H., Vercellone, V.: MDHT: a hierarchical name resolution service for information-centric networks. In: Proceedings of ACM Workshop on Information-Centric Networking, pp. 7–12 (2011)
24. Dumba, B., Mekky, H., Jain, S., Sun, G., Zhang, Z.L.: A virtual Id routing protocol for future dynamics networks and its implementation using the SDN paradigm. J. Netw. Syst. Manag. **24**(3), 578–606 (2016)
25. Pfaff, B., Pettit, J., Koponen, T., Jackson, E., Zhou, A., Rajahalme, J., Gross, J., Wang, A., Stringer, J., Shelar, P., Amidon, K., Casado, M.: The design and implementation of Open vSwitch. In: Proceeding of the 12th USENIX Symposium on Networked Systems Design and Implementation, pp. 117–130 (2015)
26. Srinivasan, V., Varghese, G., Suri, S., Waldvoguel, M.: Fast and scalable layer four switching. ACM SIGCOMM Comput. Commun. Rev. **28**(4), 191–202 (1998)
27. Schiff, L., Afek, Y., Bremler-Barr, A.: ORange: Multi field OpenFlow based range classifier. In: Proceedings of the ACM/IEEE Symposium on Architectures for Networking and Communications Systems, pp. 63–73 (2015)

28. Panigrahy, R., Sharma, S.: Sorting and searching using ternary cams. IEEE Micro **23**(1), 44–53 (2003)
29. OpenFlow switch specification: Version 1.3.5. https://www.opennetworking.org/software-defined-standards/specifications/. Accessed 24 Oct 2018
30. Rosen, E., Viswanathan, A., Callon, R.: RFC 3031: multiprotocol label switching architecture (2001)
31. Mininet Team. Mininet: an instant virtual network on your laptop (or other PC). http://mininet.org/. Accessed 24 Oct 2018
32. Ryu SDN Framework Community. Build SDN agilely. https://osrg.github.io/ryu/index.html. Accessed 24 Oct 2018
33. Knight, S., Nguyen, H.X., Falkner, N., Bowden, R., Roughan, M.: The internet topology zoo. IEEE J. Sel. Areas Commun. **29**(9), 1765–1775 (2011)
34. Linux Foundation Collaborative Projects: Open vSwitch. https://www.openvswitch.org. Accessed 24 Oct 2018
35. Emmerich, P., Raumer, D., Gallenmller, S., Wohlfart, F., Carle, G.: Trhougput and latency of virtual switching with Open vSwitch: a quantitative analysis. J. Netw. Syst. Manag. **26**(2), 314–338 (2018)
36. Thyagaturu, A.S., Mercian, A., McGarry, M.P., Reisslein, M., Kellerer, W.: Software defined optical networks (SDONs): a comprehensive survey. IEEE Commun. Surv. Tutor. **18**(4), 2738–2786 (2016)
37. Lee, S.S.W., Li, K.Y., Wu, M.S.: Design and implementation of a GPON-based virtual OpenFlow-enabled SDN switch. J. Lightwave Technol. **34**(10), 2552–2561 (2016)

Adrian Flores-de la Cruz received the Telecommunication Engineering degree in 2014 from Universidad Politecnica de Cartagena (UPCT), Spain. He is a research worker at the Department of Information and Communication Technologies of the UPCT since 2015 and his research interest is focused on Software-Defined Networks.

Pilar Manzanares-Lopez received the Engineering degree in Telecommunications in 2001 from the Technical University of Valencia (UPV), Spain. She has worked as associated professor at the Department of Information Technologies and Communications (UPCT) since 2001, where she obtained her Ph.D. in 2006. Her research work includes Software Defined Networking, P2P networks and distributed systems.

Juan Pedro Muñoz-Gea received the M.S. and Ph.D. in Telecommunication Engineering from Universidad Politecnica de Cartagena (UPCT), Spain. He works as associated professor in the Department of Information and Communication Technologies at UPCT. His research interests are focused on software-defined networking and large-scale distributed networks.

Josemaria Malgosa-Sanahuja received the degree in Telecommunication Eng. in 1994 from the Technical University of Catalonia (Spain). In November 2000, he received the Ph.D. degree in Telecommunication from the University of Zaragoza. In September 1999, he joined the Universidad Politecnica de Cartagena as associated professor. His interests include switching technologies and distributed systems.