The Budgeted Maximum Coverage Problem in Partially Deployed Software Defined Networks

Binayak Kar, Eric Hsiao-Kuang Wu, *Member, IEEE*, and Ying-Dar Lin, *Fellow, IEEE*

***Abstract*—Due to the large installed base of distributed legacy networks, software defined networking (SDN) nodes may be required to coexist with legacy nodes to form hybrid networks. It has been shown that such a hybrid network has better performance than a pure legacy network due to smarter transmission scheduling. Despite such advantages, limited budgets continue to hinder the rapid adaptation of SDNs. Only a part of the network can be upgraded at a time especially for large-scale networks. In this paper, we define the minimum percentage of SDN nodes in a path, and paths with at least one SDN node, as the hop coverage and path coverage, respectively. We intend to evaluate the relationship between cost and coverage in the partially deployed SDNs. We formulate SDN node selection as four optimization problems with *hop/path coverage* and *cost* as objectives and constraints, respectively, and vice-versa. We propose two heuristic solutions: 1) maximum number of uncovered path first (MUcPF) and 2) maximum number of minimum hop covered path first (MMHcPF), to these NP-hard problems. Through a MATLAB experiment, we show that MUcPF is significantly better in terms of economy and efficiency to establish a hybrid path between every pair of hosts in the network. In particular, it required 5%–15% less investment to achieve 100% *path coverage* compared to other algorithms. The results show the *coverage* consistency of MMHcPF on each individual path along with gains in terms of cost and efficiency. It takes 5%–20% less investment to achieve certain *hop coverage* target compared to other existing algorithms.**

***Index Terms*—Partially deployed SDN, SDN coverage, hybrid path, deployment cost.**

I. Introduction

SOFTWARE Defined Networking (SDN) [1]–[4] is a form of networking, which separates the control plane from the data plane. The data plane is maintained in individual devices, whereas the control plane is maintained in a logically centralized controller. In SDN, the network is typically controlled and managed by one or several controllers that offer a global view of the overall network and enable the administrators to dictate to the switches how the forwarding plane should handle the network traffic. Since the SDN-controller implements the centralized route control for SDN, it can provide smarter transmission scheduling to improve the link utilization [5] and the network throughput [6]. Because of the global view of the network, it manipulates the network resources dynamically and reduces the resource utilization [7] for traffic manipulation. Research shows that SDN can reduce the administrative task and maintenance cost by automation of operation due to its programmable network and cheaper hardware. It can make the network more efficient and scalable [8] by reducing the deployment cost of new devices and applications.

Most of the existing works on SDN are considered on pure SDN, where all switches of a network are controlled by the centralized controller. Despite several benefits [9], rapid adoption of SDN remains challenging. One of the most important reasons is limited budget. Upgrading a large network at once is a major task. Overnight adoption of SDN in the whole network may disrupt the network already in use, because after pure SDN deployment, if there is any kind of implementation problem [10], it would affect the network connectivity. Meanwhile, this emerging SDN architecture is not established enough to replace the legacy network as a whole. However, at the present time, SDN has several limitations and research challenges [11] concerning reliability [12], security [13], quality of services, etc. For example, congestion in the control plane may lead to quality of service issues [14]. There are also other challenges, such as compatibility [15], since the northbound application program interfaces (APIs) are not standardized. In the case of multiple controllers, APIs of one controller may not be compatible with another controller. Therefore, we need a mechanism to adopt the new technology along with protecting the existing investment.

To resolve the above issues, it is necessary to deploy SDN incrementally. This transition from legacy to SDN is likely to span several years [6], [18]. The hybrid model can achieve the qualitative diverse tradeoffs by making them convenient for various transition strategies and long-term network designs [23]. Therefore, SDNs are incrementally introduced to the legacy network, which will help to evaluate its practicality. After such partial deployment, the benefits of SDN potentially extend over the whole network through the SDN-enabled nodes. This new type of network is called a partially deployed SDN or hybrid SDN. We will use both the names throughout this paper. In this hybrid network, only the SDN-enabled switches are controlled by the centralized controller other switches function as legacy switch. An example of hybrid-SDN is shown in Fig. 1. The nodes 2, 9, and 13 are the SDN-enabled switches controlled by

Manuscript received February 15, 2016; revised June 17, 2016; accepted July 5, 2016. Date of current version September 30, 2016. This work was supported in part by Ministry of Science and Technology, Taiwan, and also in part by Chunghwa Telecom. The associate editor coordinating the review of this paper and approving it for publication was F. De Turck.

B. Kar and E. H.-K. Wu are with the Department of Computer Science and Information Engineering, National Central University, Chung-Li 32001, Taiwan (e-mail: binayakar@wmlab.csie.ncu.edu.tw; hsiao@csie.ncu.edu.tw).

Y.-D. Lin is with the Department of Computer Science, National Chiao Tung University, Hsinchu 300, Taiwan (e-mail: ydlin@cs.nctu.edu.tw).

Digital Object Identifier 10.1109/TNSM.2016.2598549

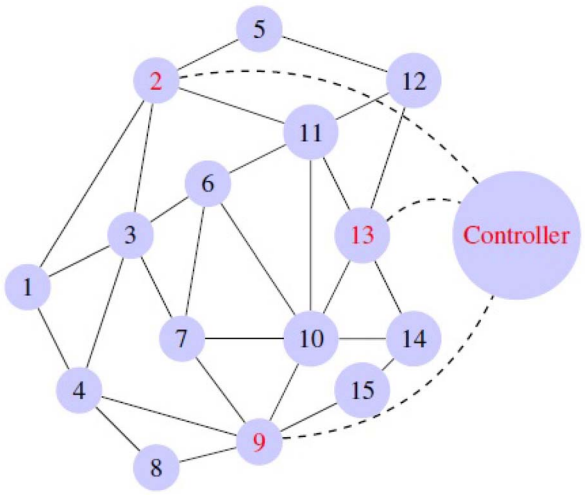


Fig. 1. Partially deployed SDN.

the SDN controller and other nodes are legacy switches or non-SDN node.

Definition 1: The communication path from a source to a destination is classified into 3 categories depending upon the number of SDN nodes in the path.

1) Legacy path: a path is called a legacy path, if it contains zero number of SDN nodes.

2) Pure SDN path: a path is called a pure SDN path, if all the nodes in the path are SDN-enabled.

3) Hybrid path: a path is called a hybrid path, if it contains both non-SDN and SDN-enabled nodes.

In Fig. 1, for a (4-12) *source-destination* pair, a number of paths exist. Let us consider three paths A (4-1-2-5-12), B (4-3-6-11-12) and C (4-9-10-13-12). Path B is a legacy path, as all the nodes in this path are non-SDN nodes, whereas path A and C are hybrid paths because they contain one (node 2) and two (node 9 and 13) SDN-enabled nodes, respectively. In this paper, the pure SDN paths and hybrid paths are restricted to only shortest paths of the network.

Definition 2: The path is divided into two categories based on *SDN coverage*.

1) SDN covered path: a path is said to be an SDN covered path, if there exists at least one SDN-enabled node in that path. All pure SDN and hybrid paths are *SDN covered path*.

2) SDN uncovered path: a path is said to be an SDN uncovered path, if all the nodes in that path are non-SDN nodes. All legacy paths are *SDN uncovered path*.

In Fig. 1, path B (4-3-6-11-12) is an SDN uncovered path as all the nodes in this path are non-SDN nodes, whereas path A (4-1-2-5-12) and path C (4-9-10-13-12) are SDN covered path as these paths are hybrid paths.

In hybrid SDN, the network can gain advantages of SDN while protecting the existing investment up to a certain extent. This hybrid network is a transition of the network from legacy to pure SDN. During this SDN up-gradation of the network, there are a few issues we need to consider like: "*How many switches should be upgraded to SDN switch in a network? Which switch should be upgraded and why?*" Leven *et al.* [16] presents a new architecture and methodology "Panopticon" for planning and operating hybrid SDN network switches. In Panopticon, they explain that, if we deploy one SDN switch in a communication path, all the flows in that path will get the key benefits of the centralized controller. The paths that pass even at least one SDN-enabled node can get the logically-centralized programmatic interface for orchestration [22]. The centralized architecture reduces the network management burden from the legacy network as the new network gets the SDN benefits such as dynamic policy enforcement [3], consistent policy update [19], network debugging [20], and customization of network behavior among others. For example, in a critical traffic condition, the controller can enforce rules for certain input packets, to attain the lowest possible delays [23]. The centralized controller can dictate the SDN-enabled switches to drop/buffer/forward the packets based on the policy requirement. Regardless of the path the flows travel, deploying one SDN node in each path controller can instruct the switches to take appropriate action if needed. Moreover, the traffic which traverses more than one SDN-enabled node may gain a greater advantage, like load balancing [21], by enabling further customized forwarding decisions. Agrawal *et al.* [24] proposed a hybrid SDN controllers optimization problem for traffic engineering, where they minimize the maximum link utilization. In the traditional traffic engineering scenario, the flows are always routed along the shortest paths. But in the SDN nodes, the controller will take the routing decision. It can centrally control the flows and redirect them to alternative paths. As a result, the delay and packet loss at the SDN links are minimized, since the delay and packet loss are the increasing function of link utilization [25]. So the performance of the network increases by increasing the number of SDN nodes and SDN links. Here, SDN links mean the links connected to the SDN nodes. This implies that if the number of SDN nodes in a path increases, the number of SDN links in that path also increases, and as a result, the performance of that path will improve.

For data center networks, SDN node selection is not a significant issue [26], as the network has a predefined structure like *fat-tree*. But networks such as backbone network, where the network does not have a specific structure, selection of nodes for SDN deployment is still a big issue. While selecting a new node for SDN deployment, various factors need to be considered, such as number of hosts, number of switches, switch's position, deployment cost, switch's priority, capacity and its impact on the network. Considering these factors, we analyze the SDN *deployment cost* and *coverage* issue in depth. To know the *SDN coverage* impact of nodes, on the whole network as well as each individual path, we further classify this *SDN coverage* into two categories, such as *path coverage* (*Pcoverage*), coverage of the paths of the whole network and *hop coverage* (*Hcoverage*), coverage of each individual paths of the network.

Pcoverage: If there exists, at least one SDN-enabled node in the path, the path is *Pcoverage*. More simply, all the hybrid paths are *Pcoverage*. If one or more paths pass through a single *SDN-enabled node*, then all those paths are *Pcoverage*.

Pcoverage of a network is the ratio of the number of hybrid paths to total paths in the network. In this paper, we considered total paths to mean the total number of shortest paths as we considered only the shortest paths for SDN deployment. We assume only one shortest path between every source-destination pair. For a 100% *Pcoverage* network, every path should have at least one SDN-enabled node.

Hcoverage: *Hop coverage* is defined by the percentage of nodes within the path being SDN-enabled (i.e., in a path, the number of SDN nodes that are deployed). More simply, it is the ratio of the number of SDN nodes in a hybrid path to the total number of nodes in that path. For example, if there are 2 SDN-enabled nodes present in a 10-node path, then their *Hcoverage* is 20%. In Fig. 1, *Hcoverage* of path A, B, and C are 20%, 0%, and 40%, respectively. To get a 10% *Hcoverage* network, all the paths of the network should have minimum 10% *Hcoverage*. Here all paths mean all shortest paths in the network.

In this paper, we try to select more suitable nodes for SDN deployment to transit the networks into efficient partially deployed software defined networks. More specifically, we consider the problem of finding the nodes that give more *coverage* benefit with minimum *deployment cost* in partially deployed SDNs. The objective of this paper is to develop a scheme that will be able to select a set of nodes in any network for SDN deployment with a certain cost limit to achieve the targeted *coverage*. Our novel contribution is summarized as follows:

1) First, we define the *SDN coverage* scenario and related deployment issues, including *deployment cost* that has not been discussed in previous works. Then we classify the *SDN coverage* scenario in a partially deployed SDN environment.
2) We address the budgeted maximum *coverage* problems in partially deployed SDN and formulate the maximum *coverage* minimum *cost* hybrid SDN optimization problems both for *path coverage* and *hop coverage*, and prove them NP-hard.
3) We propose two efficient heuristic algorithms, Algorithm 1 (MUcPF) for *path coverage* and Algorithm 2 (MMHcPF) for *hop coverage*, to solve the formulated problems. By MATLAB experiment, we demonstrate that our algorithms achieve the maximum *coverage* with minimum investment.

The rest of the paper is organized as follows. We highlight some hybrid SDN related works in Section II. We formulate the budgeted *coverage* optimization problems in Section III and prove them as NP-hard. Based on the formulation we designed two heuristic algorithms in Section IV. We evaluate the performance in Section V. Finally, we provide conclusions in Section VI.

II. Related Works

In this section, we will review and discuss the work that has already been accomplished to investigate the hybrid SDN networks. The literature study reveals that there are numerous works that have been reported on pure SDN.

TABLE I
Related Works on Hybrid SDN

Name	Traffic Engineering	Budget	Coverage
B4[6]	✓	×	×
TE-SDN[24]	✓	×	×
Panopticon[16]	✓	✓	×
OSTE[32]	✓	×	×
DC-SDN[33]	✓	×	×
This paper	✓	✓	✓

Nascimento *et al.* [17] discussed the routing control platforms in the context of OpenFlow and introduced a controller-centric hybrid-networking model and presented the design of the RouteFlow Control Platform (RFCP) along with the prototype implementation. H-SDN [27] tells that a partially SDN deployed environment would allow service providers to maintain their policy and help to expand audience through existing infrastructure connectivity and services. HybNET [28] presents an automation management framework for hybrid networks. It provides a centralized configuration interface to build virtual networks using virtual LANs. Here an abstraction of the physical network topology is taken into account by a centralized controller that applies a path finder mechanism, in order to calculate network paths and program the OpenFlow switches via REST interfaces. Open Source Hybrid IP/SDN (OSHI) [29] combines Quagga for OSPF routing and SDN capable switching devices (e.g., OpenvSwitch) on Linux to provide backward compatibility for supporting incremental SDN deployments. Kobayashi *et al.* [30] shares their SDN deployment experience in various campuses. The work [31] studies network updates in incrementally deployed SDNs. They develop the machinery to realize anomaly-free updates of hybrid networks. Orion [34] presents a hybrid hierarchical control plane for large-scale networks that can effectively reduce the computational complexity of an SDN control plane. Hybrid SDN can also play a key role in energy saving in the network. Wang *et al.* [35] discussed the energy consumption of the network using SDN. They proposed an optimal hybrid model for energy saving by switching off the SDN-enabled switches when not in use.

Table-I lists some papers based on hybrid SDN and their major contributions. Jain *et al.* [6] presented the design, implementation, and evaluation of B4, a private WAN connecting Google data centers across the planet. B4 took an SDN architecture using OpenFlow to control relatively simple switches. The centralized traffic engineering service drove links to near 100% utilization while splitting application flows among multiple paths to balance capacity against application demands. Levin *et al.* [16] systematically tackle the problem of how to partially deploy SDN-enabled switches into existing legacy networks and reap the benefits of the network. They present Panopticon, an architecture, and methodology for planning and operating networks that combine traditional switches and deployed SDN switches. The proposed architecture includes policy configurations, troubleshooting and maintenance tasks establishing transitional

TABLE II
LIST OF COMMONLY USED NOTATIONS AND VARIABLES

Notation/Variable	Description
s_i	Source node of path *i*.
t_i	Destination node of path *i*.
P	Set of shortest paths.
Q	Matrix form of P.
p_i	Shortest path from s_i to t_i.
V	Set of nodes.
X	Set of SDN nodes.
Y	Set of hybrid paths.
x_i	SDN node *i*.
y_i	Hybrid path *i*.
h_i	Set of nodes in path *i*.
sh_i	Set of SDN nodes in path *i*.
$Dcost_i$	*Deployment cost* of node v_i.
$Target_{Dcost}$	Target SDN node *deployment cost*.
$Target_{Pcoverage}$	Target *path coverage*.
$Target_{Hcoverage}$	Target *hop coverage*.

networks (SDN and legacy) in structures called Solitary Confinement Trees. Agarwal *et al.* [24] focus on incrementally deployed SDNs and discuss the effective use of SDNs for traffic engineering. They formulated the SDN controller optimization problem for traffic engineering and developed a Fully Polynomial Time Approximation Schemes (FPTAS) for solving this problem. OSTE [32] explores the traffic engineering in an SDN/OSPF hybrid network. They proposed a novel algorithm to minimize the maximum link utilization. Caria *et al.* [33] uses a *divide and conquer* method to partition the OSPF domain to sub-domains by selecting suitable SDN nodes for traffic engineering. The above-discussed articles in the Table I have focused on the traffic engineering issue of the hybrid network. However, article [16] has discussed the cost factor along with TE. The *SDN coverage* issue in a hybrid network, especially in the transition of legacy to pure SDN, the impact of SDN nodes on the network as well as each individual path has not been addressed before. In this paper, we will address both the *path coverage* and *hop coverage* issues along with *cost*.

III. PROBLEM FORMULATION

A. Variable Declaration

In this section, in Table II, we declared the variables that we have used to formulate the optimization problems. The variables s and t stand for the source and destination nodes respectively. In the path i, s_i is the source and t_i is the destination, where $s_i \neq t_i$. For two different paths i and j, there are three scenarios for the source-destination pairs: 1) $s_i \neq s_j$ & $t_i \neq t_j$, 2) $s_i = s_j$ & $t_i \neq t_j$ and 3) $s_i \neq s_j$ & $t_i = t_j$. If we have k numbers of hosts (s or t) in the network and each host is connected with a unique edge node, then the shortest path set P contains $k(k-1)/2$ numbers of paths. Assuming only one shortest path exists between every pair of hosts. As most of the advanced routing like OSPF in traditional network follows shortest path, we consider the shortest path nodes for SDN up-gradation. V is the set of nodes in the network, where each element of V must be a node in at least one shortest path. X and Y are the set of SDN nodes and hybrid paths respectively. h_i is the set of nodes and sh_i is the set of SDN nodes in the path i. *Dcost* is the *deployment cost* of nodes; we assume, the *deployment cost* of each node are equal. The sum of the *deployment cost* is the *total deployment costs* and $Target_{Dcost}$ is the investment boundary. Similarly, $Target_{coverage}$ is the objective *coverage* to achieve.

B. Problem Formulation

In this section, we will formulate our optimization problem using the variables described in Table-II.

Let $P = \{p_1, p_2, \ldots, p_m\}$ be the set of shortest paths and $V = \{v_1, v_2, \ldots, v_n\}$ be the set of nodes in the shortest paths of the network, where $m \leq n$, i.e., number of shortest paths always less than or equal to number of nodes present in the shortest paths of the network, as we are assuming only one shortest path between every "source-destination" pair. P can be represented as a *0-1 matrix* Q $[m \times n]$, *where* $Q_{ij} = 1$; *if* v_j *is a node present in path* p_i, *0 otherwise*. Here i represent the path set and j represents the node set. Let $X = \{x_1, x_2, \ldots, x_n\}$ be the set of SDN nodes, *where* $x_j = 1$ *if* v_j *is an SDN node, 0 otherwise*. $Y = \{y_1, y_2, \ldots, y_m\}$ be the set of paths covered by at least one SDN node, *where* $y_i = 1$ *if* $\exists$ *an* $x_j = 1$, 0 *otherwise*.

We classify our problem into two broad categories: given *cost* as a constraint, maximize the *coverage* and given *coverage* as a constraint, minimize the *cost*.

1) Maximize the Coverage With Cost as a Constraint: We are given a cost limit for the SDN deployment in total and *deployment cost* of each individual node. The objective here is to find suitable nodes for SDN deployment for maximum *coverage* such that the *total deployment cost* never exceeds the *target deployment cost*. If $Dcost_k$ is the *deployment cost* of node v_k, then *total deployment cost* is $Total_{Dcost} = \sum_{k=1}^{K} Dcost_k$, where $K \leq n$, subject to $Total_{Dcost} = Target_{Dcost}$. In the next two subsections, we will define in detail this *coverage* problem with *cost* constraints both for *Pcoverage* and *Hcoverage* respectively.

a) Maximize the Pcoverage with cost constraint: Path coverage means deploying, at least one SDN node in the path. Maximizing *path coverage* means maximizing the number of *hybrid paths*. Thus, we

$$\textit{maximize} \quad \frac{\sum_{i=1}^{m} y_i}{\sum_{i=1}^{m} p_i} \tag{1}$$

$$\textit{subject to} \quad \sum_{k=1}^{K} Dcost_k \leq Target_{Dcost}, \quad K \leq m, \tag{2a}$$

$$y_i \leq \sum_{j=1}^{n} Q_{ij} x_j, \quad \forall i, \tag{2b}$$

$$Dcost_k \geq 0, \quad \forall k, \tag{2c}$$

$$x_j, y_i \in \{0, 1\}. \tag{2d}$$

Our objective is defined as the ratio of the total number of hybrid paths to the total number of shortest paths in the network as presented in Equation 1. The Equation 2 presents

the set of constraints to our objective. The first inequality in Equation 2a ensures that the total sum of *deployment cost* is always less than the *target deployment cost* and the total number of SDN nodes is always less than or equal to the total number of paths. The second inequality in Equation 2b ensures that there should be at least one SDN node in each hybrid path. The third inequality in Equation 2c ensures that the *deployment cost* of any node is non-negative. The fourth inequality in Equation 2d ensures that the variables are in binary form.

b) Maximize the Hcoverage with cost constraint: Let $h_i = \{h_{i1}, h_{i2}, \ldots, h_{in}\}$ be the set of nodes in the path p_i, where $h_{ij} = 1$; if it is a node in the path p_i, *0 otherwise.* $sh_i = \{sh_{i1}, sh_{i2}, \ldots, sh_{in}\}$ be the set of SDN-nodes in the path p_i, where $sh_{ij} = 1$; *if* $x_j = 1$, 0 *otherwise*. The objective here is to maximize the *hop coverage* in each path such that each path will get a minimum percentage of *coverage* irrespective of their number of nodes. Thus, we

$$maximize \quad Min\left(\frac{\sum_{j=1}^{n} sh_{ij}}{\sum_{j=1}^{n} h_{ij}}\right), \tag{3}$$

$$subject\ to \quad \sum_{k=1}^{K} Dcost_k \leq Target_{Dcost}, K \leq n, \tag{4a}$$

$$y_i \leq \sum_{j=1}^{n} Q_{ij} x_j, \forall i, \tag{4b}$$

$$Dcost_k \geq 0, \forall k, \tag{4c}$$

$$x_j, y_i, h_{ij}, sh_{ij} \in \{0, 1\}. \tag{4d}$$

In this optimization problem, our objective is to maximize the minimum ratio of the total number of SDN nodes and the total number of nodes in the shortest paths in the network (shown in Equation 3). The first constraint in Equation 4a ensures that the total sum of *deployment cost* should not exceed the *target deployment cost* and the total number of SDN nodes are always less than or equal to the total number of nodes in the network. The second inequality in Equation 4b ensures that there should be at least one SDN node in each hybrid path and the third inequality in Equation 4c ensures that the *deployment cost* of any node is non-negative. Equation 4d ensures that all variables are in binary form.

2) Minimize the Cost With Coverage as a Constraint: This is the dual form of "maximize *coverage* with *cost* as a constraint" problem, because here, we are given the *deployment cost* of each node and a minimum coverage target to achieve. Our objective is to find the suitable nodes for SDN deployment such that $Total_{Dcost}$ is minimized and *total coverage* must be greater than or equal to the *target coverage*. In the next two subsections, we will define this minimum *cost* problem with *coverage* as a constraint for both *Pcoverage* and *Hcoverage* respectively.

a) Minimize the cost with Pcoverage constraint: If $\sum_{i=1}^{m} p_i$ is the total number of paths and $\sum_{i=1}^{m} y_i$ is the total number of paths covered by at least one SDN node, then we can define the problem as,

$$minimize \quad \sum_{k=1}^{K} Dcost_k, K \leq m, \tag{5}$$

$$subject\ to \quad \frac{\sum_{i=1}^{m} y_i}{\sum_{i=1}^{m} p_i} \geq Target_{Pcoverage}, \tag{6a}$$

$$y_i \leq \sum_{j=1}^{n} Q_{ij} x_j, \forall i, \tag{6b}$$

$$Dcost_k \geq 0, \forall k, \tag{6c}$$

$$x_j, y_i \in \{0, 1\}. \tag{6d}$$

This problem is the dual form of the problem "maximize the *path coverage* with *cost* as a constraint". Here our objective in Equation 5 is to minimize the *total deployment cost*, where the total number of deployments should not exceed the total number of paths. The first constraint in Equation 6a ensures that the ratio of total hybrid paths and total paths must be greater than or equal to the target *path coverage*. The second inequality in Equation 6b ensures that there should be at least one SDN node in each hybrid path. The third inequality in Equation 6c ensures that the *deployment cost* of any node is non-negative. The fourth inequality in Equation 6d ensures that all variables are in binary form.

b) Minimize the cost with Hcoverage constraint: If $\sum_{j=1}^{n} h_{ij}$ is the total number of nodes for each path i and $\sum_{j=1}^{n} sh_{ij}$ is the total number of SDN-nodes in the path i, then we can define the problem as,

$$minimize \sum_{k=1}^{K} Dcost_k, K \leq n, \tag{7}$$

$$subjectto \quad \frac{\sum_{j=1}^{n} sh_{ij}}{\sum_{j=1}^{n} h_{ij}} \geq Target_{Hcoverage}, \tag{8a}$$

$$\sum_{j=1}^{n} h_{ij} \geq \sum_{j=1}^{n} sh_{ij}, \forall i, \tag{8b}$$

$$y_i \leq \sum_{j=1}^{n} Q_{ij} x_j, \forall i, \tag{8c}$$

$$Dcost_k \geq 0, \forall k, \tag{8d}$$

$$x_j, y_i, h_{ij}, sh_{ij} \in \{0, 1\}. \tag{8e}$$

This problem is the dual form of the problem "maximize the *hop coverage* with *cost* as a constraint". Here our objective in Equation 7 is to minimize the *total deployment cost*, where the total number of deployments should not exceed the total number of nodes in the network. The first constraint in Equation 8a ensures that the ratio of SDN nodes to total nodes of any path must not be less than the *target hop coverage* and the second constraint in Equation 8b ensures that the total number of SDN nodes in a path should not exceed the total number of nodes in that path. The inequality in Equation 4c ensures that there should be at least one SDN node in each hybrid path and the inequality in Equation 4d ensures that the *deployment cost* of any node is non-negative. Equation 4e ensures that all variables are in binary form.

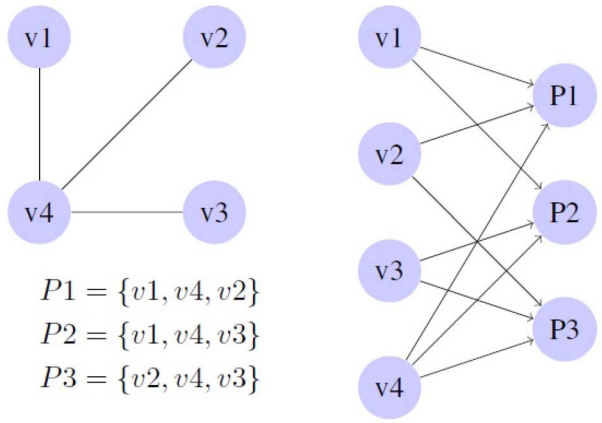


Fig. 2. Network with four nodes and three paths, describes the node-path mapping.

C. Problem Analysis

In the previous subsection, we proposed four optimization problems, where, in two problems our objective is to maximize the *coverage,* where *cost* is a constraint. Similarly, in the other two problems our objective is to minimize the *cost,* where *coverage* is a constraint. But all these optimization problems are proposed on a hybrid SDN network with maximum *coverage* and minimum *cost* concept or it's dual. In this section, we will prove that the maximum *coverage* minimum *cost* optimization problem is NP-hard.

Theorem 1: The optimal SDN deployment in the network for maximum *coverage* with minimum *cost* is NP-hard.

Proof: To prove the maximum *SDN coverage* with minimum *cost* is NP-hard, we use the technique of reduction. Consider a set of nodes $V = \{v_1, v_2, \ldots, v_n\}$, where each node has a *deployment cost* associated with it, i.e., $\{Dcost_i\}_{i=1}^{n}$. The nodes are present over set of paths $P = \{p_1, p_2, \ldots, p_m\}$, where each path j has certain path length $\{len_j\}_{i=1}^{m}$ and SDN weight (i.e., number of SDN node present in the path) $\{sw_j\}_{j=1}^{m}$. So the *SDN coverage* of paths can be represented as$\{cov_j\}_{j=1}^{m}$, where $cov_j = \frac{sw_j}{len_j}$.

The example shown in Fig. 2 has four nodes in the network {*v1, v2, v3, v4*}, and three paths {p_1, p_2, p_3}. If we consider v_1-v_4-v_2 as path p_1, v_1-v_4-v_3 as path p_2 and v_2-v_4-v_3 as path p_3, then path length of each path will be three, i.e., $len_1 = len_2 = len_3 = 3$. If we deploy v_1 as the SDN node in the network, then $sw_1 = 1$; $sw_2 = 1$ and $sw_3 = 0$. Again if we deploy v_4 as an SDN node, then $sw_1 = 2$; $sw_2 = 2$ and $sw_3 = 1$. So the *SDN coverage* of all 3 paths after two nodes deployment are $cov_1 = 2/3$; $cov_2 = 2/3$ and $cov_3 = 1/3$. If $V_S \subseteq V$ is the set SDN nodes in the network our objective is total cost of SDN node deployment in the network does not exceed the given budget i.e., $Target_{Dcost}$ and *total coverage* of paths is maximized.

Definition 3: A function $f : \delta_1 \rightarrow \delta_2$ is called a *mapping reduction* from A to B iff

a) For any $\alpha \in \delta_1, \alpha \in A$ iff $f(\alpha) \in B$.

b) f is a computable function.

Intuitively a mapping reduction from A to B says that a computer can transform any instance of A into an instance of B such that the answer to B is the answer to A.

The budgeted maximum coverage problem [36] defined as "*A collection of sets* $S = \{s_1, s_2, \ldots, s_m\}$ *with associated cost* $\{c_i\}_{i=1}^{m}$ *is defined over a domain with associated weight* $\{w_i\}_{i=1}^{n}$. *The goal is to find a collection of sets, s.t. the total cost of elements in* S^l *does not exceed a given budget L, and the total weight of elements covered by* S' *is maximized.*" This problem is NP-hard. By mapping the variables of this NP-hard problem to the variables of our optimization problem, we have,

$$\left\{ \begin{array}{c} S \rightarrow V \\ S^{'} \rightarrow V_S \\ c \rightarrow Dcost \\ w \rightarrow cov \end{array} \right\} \tag{9}$$

By definition 3 and Equation 9, we can map and reduce the NP-hard problem to our optimization problem. Hence, our optimization problem is NP-hard. ■

IV. Solution Approach

In this section, we will propose two heuristic coverage algorithms, one for *path coverage* and other for *hop coverage* to select nodes in any random topology for SDN deployment.

A randomly generated weighted graph is considered and a set of nodes in the graph is selected as *edge nodes* randomly, which are connected to the hosts in our topology. The shortest path algorithm to find the shortest paths from each *edge node* to every other *edge node* in the graph (assume ∃ only one shortest path between each pair of hosts) is applied. We eliminate the *long-path-nodes* (nodes which are not belonging to any shortest path) from the graph as we will not consider them for SDN-deployment. We only consider the nodes which are in the shortest path for SDN deployment, because most of the advanced routing like OSPF in traditional networks follows a shortest path. Selecting the nodes in the shortest path will minimize the hop count. Now we have a new graph which contains only the nodes and edges which are belonging to any of the shortest paths of the network. From the new graph, let's compute the *0-1 adjacency matrix* (B) [37] and *0-1 path matrix* (P), where P is a matrix that represents which nodes are in which path. *If* j is a node in the path i, then $P_{ij} = 1$, *else* 0.

The set of inputs we have taken for these algorithms are adjacency matrix (B), path matrix (P), X (node set), Y (path set). In the set X, if the value of x_i is 1, $\forall i$, then the node is an SDN node, else non-SDN node. In the set Y, if the value of y_j is 0, $\forall j$, then the path is a legacy path else hybrid path. Initially, all the nodes are non-SDN nodes and all paths are legacy paths. We assume the *deployment cost* of all nodes is equal.

A. MUcPF Algorithm

The *MUcPF algorithm* executes the heuristic *path coverage* problem. Its objective is to select the node for SDN deployment, which will cover the maximum number of uncovered paths first. The *MUcPF* algorithm works as follows: We first generate a node path SDN matrix ($np_{SDN_{mtx}}$). It is a matrix

sdn	$minSDNpath_1$	$\cdots$	$minSDNpath_{k1}$	$weight$	$degree$
1	1	$\cdots$	1	-1	-1

(a) MUcPF node structure

sdn	$minCov_1$	$\cdots$	$minCov_{k1}$	$maxLen_1$	$\cdots$	$maxLen_{k2}$	$weight$	$degree$
1	1	$\cdots$	1	-1	$\cdots$	-1	-1	-1

(b) MMHcPF node structure

Fig. 3. Node structure for different coverage algorithms. (a) Node structure for heuristic *path coverage* algorithm. (b) Node structure for heuristic *hop coverage* algorithm.

Algorithm 1 MUcPF Algorithm

Input: *B, P, limit*; X={$x_1, x_2, \ldots, x_n$}; Y={$y_1, y_2, \ldots, y_m$}; x_i=1, if v_i is an SDN node else 0; y_j=1, if p_j is the path having at least one SDN node else 0; $Dcost_i$.
Initialization: x_i=0, $\forall i$; y_j=0, $\forall j$; $Dcost_i$=1, $\forall i, x_i \in X$.
Algorithm:
$np_{SDN_{mtx}} = generateNpSDNmatrix(P^{'}, X, Y)$
$nodes = struct(num, sdn, minSDNpath_1, \ldots, minSDNpath_{k1}, wt, deg)$
$nodes = assignValuetoNodes(nodes, B, P, np_{SDN_{mtx}})$
while $totalSDN_{Dcost} \leq limit$ **do**
  $nodes_{sorted} = nestedSortStructure(nodes, \{sdn, minSDNpath_1, \ldots, minSDNpath_{k1}, wt, deg\})$
  for $i = 1 \rightarrow m$ **do**
    if $nodes_{sorted}(i).num \leq 0$ **then**
      Display ('*Error!*');
    else
      $deployedSDNnode = nodes_{sorted}(i).num$;
      $path_{numSDN} = NumofSDNinPath(P, X, deployedSDNnode)$
      $np_{SDN_{mtx}} = updateNpSDNmtx(P^{'}, path_{numSDN}, X, Y)$
      $nodes = updateNodesValue(nodes, np_{SDN_{mtx}}, X)$
      $totalSDN_{Dcost} = \sum_{i=1}^{n} x_i Dcost_i$;
      $totalSDN_{path} = \sum_{j=1}^{m} y_j$;
      $path_{coverage} = totalSDN_{path}/m$;
    end
  end for
end while
Output: $path_{coverage}$

that represents whether a node belongs to a particular path or not and how many SDN nodes are present in each path. We create a node structure (*nodes*), as given in Fig. 3a, which consist of $k_1 + 4$ fields (where $k_1 \leq m$); i.e., node numbers (*num*), nodes are SDN node or not (*sdn*), node's hybrid path in ascending order (*minSDNpath*), weight (*wt*) and degree (*deg*) of the nodes. Let us consider an example to understand the meaning of *node's hybrid path*. Let there be 5 paths (p_1, p_2, p_3, p_4, p_5) and the number of SDN nodes in $p_1 = 2$, $p_2 = 0$, $p_3 = 1$, $p_4 = 0$ and $p_5 = 1$. Out of the 5 paths, if 3 paths (p_1, p_2, p_3) pass through a node r, then r's $minSDNpath_1 = 0(p_2)$, $minSDNpath_2 = 1(p_3)$ and $minSDNpath_3 = 2(p_1)$. For the paths that are not passing through the node r, their *minSDNpath* value for r will set to ∞ irrespective of number of SDN nodes in that path. Since p_4 and p_5 are not passing through r, so, r's $minSDNpath_4 = \infty$ (for p_4) and $minSDNpath_5 = \infty$ (for p_5). After creation of the structure, values are assigned to the structure from the node path SDN matrix and the condition for the constraint (*limit*) is checked, which is given as an input to run the algorithm. We use multiple levels of structural sorting to sort the structure *nodes* (*see* Fig. 3a: Here '1' represents that the fields are sorted in ascending order and '-1' for descending order). The selected new node for SDN deployment is named as *deployedSDNnode* and is used to compute the number of SDN nodes in each path ($path_{numSDN}$). After updating the node path SDN matrix, we compute the *total deployment cost* ($totalSDN_{Dcost}$) and *path coverage* ($path_{coverage}$). This process will continue for each new deployment until we achieve our target.

B. MMHcPF Algorithm

The heuristic *hop coverage* problem is solved in the *MMHcPF algorithm*. Its objective is to select the node for SDN deployment that covers the maximum number of minimum *hop coverage* paths first. The *MMHcPF* algorithm works as follows: initially, we create a structure named *path* and assign the values to the structure by the function *assignValuetoPath*(). This structure consists of four fields; i.e., the path number (*no*), path length (*len*), number of SDN nodes in each path (hop_{sdn}) and *hop coverage* of each path (hop_{cov}). Then, we generate two matrices such as a node path matrix (np_{mtx}) and a *path coverage matrix* (pc_{mtx}). The node path matrix is a matrix that represents a node belonging to the paths and the length of those paths. The *path coverage matrix* represents which nodes belong to a path and what is the *hop coverage* of each path. We create another structure named *nodes* and assign value to it using the function *assignValuetoNodes*(). This structure consist of k_1+k_2+4 fields, as given in Fig. 3b, (where $k_1, k_2 \leq m$); i.e., node numbers (*num*), nodes are SDN-enabled nodes or not (*sdn*), *node paths hop coverage* (*minCov*) in ascending order, *node path length* (*maxLen*) in descending order, weight (*wt*) and degree (*deg*) of the nodes. Let us discuss the two terms "*node path hop coverage*" and "*node path length*" with one example. Consider five paths (p_1, p_2, p_3, p_4, p_5). For a path p_i, $\forall i = \{1 \ldots 5\}$, h_i is the number of SDN nodes, l_i is the path length and *hop coverage* $c_i = h_i/l_i$. Let the number of SDN node and length of the paths are $p_1 : h_1 = 3, l_1 = 7$; $p_2 : h_2 = 2, l_2 = 9$; $p_3 : h_3 = 4, l_3 = 8$; $p_4 : h_4 = 2, l_4 = 7$; $p_5 : h_5 = 3; l_5 = 9$. *Hop Coverage* of 5 paths are $p_1 : c_1 = h_1/l_2 = 0.42$; $p_2 : c_2 =$

Algorithm 2 MMHcPF Algorithm

Input: B, P, $limit$; $X=\{x_1, x_2, \ldots, x_n\}$; $Y=\{y_1, y_2, \ldots, y_m\}$; $x_i=1$, if v_i is an SDN node else 0; $y_j=1$, if p_j is the path having at least one SDN node else 0; $Dcost_i$.
Initialization: $x_i=0, \forall i$; $y_j=0, \forall j$; $Dcost_i=1, \forall i, x_i \in X$.
Algorithm:
$path = struct(no, len, hop_{sdn}, hop_{cov})$
$path = assignValuetoPath(path, P, Y)$
$np_{mtx} = generateNodePathMtx(P^{'}, X, Y)$
$pc_{mtx} = generatePathCovMtx(path, P^{'}, X, Y)$
$nodes = struct(num, sdn, minCov_1, \ldots, minCov_{k1}, maxLen_1, \ldots, maxLen_{k2}, wt, \deg)$
$nodes = assignValuetoNodes(nodes, B, P, np_{mtx}, pc_{mtx})$
while $totalSDN_{Dcost} \leq limit$ **do**
  $pc_{mtx} = updatePathCovMtx(pc_{mtx}, path, P^{'}, X, Y)$
  $nodes = updateNodesValue(nodes, pc_{mtx}, X)$
  $nodes_{sorted} = nestedSortStruct(nodes, \{sdn, minCov_1, \ldots, minCov_{k1}, maxLen_1, \ldots, maxLen_{k2}, wt, deg\})$
  for $i = 1 \rightarrow m$ **do**
    if $nodes_{sorted}(i).num \leq 0$ **then**
      Display ('*Error!*');
    else
      $deployedSDNnode = nodes_{sorted}(i).num$;
      $totalSDNnode = \sum_{i=1}^{n} x_i$;
      $totalSDN_{Dcost} = \sum_{i=1}^{n} x_i Dcost_i$;
      $path.hop_{sdn} = \{\sum_{i=1}^{n} sh_{1i}, \ldots, \sum_{i=1}^{n} sh_{mi}\}$;
      $path.hop_{cov} = Min(\frac{\sum_{i=1}^{n} sh_{1i}}{\sum_{i=1}^{n} h_{1i}}, \ldots, \frac{\sum_{i=1}^{n} sh_{mi}}{\sum_{i=1}^{n} h_{mi}})$;
    end
  end for
end while
Output: $path.hop_{cov}$

$h_2/l_2 = 0.22$; $p_3 : c_3 = h_3/l_3 = 0.5$; $p_4 : c_4 = h_4/l_4 = 0.28$; $p_5 : c_5 = h_5/l_5 = 0.33$. Out of these 5 paths, if 3 paths (p_1, p_2, p_3) pass through a node r, then r's $minCov_1 = 0.22$($path\ p_2$), $minCov_2 = 0.42$($path\ p_1$) and $minCov_3 = 0.5$($path\ p_3$). Similarly r's $maxLen_1 = 9$($path\ p_2$), $maxLen_2 = 8$($path\ p_3$) and $maxLen_3 = 7$($path\ p_1$). The paths that do not pass through the node r, the *minCov* value of the node for those paths are set to be ∞ and *maxLen* values are set to be 0. So r's $minCov_4 = \infty$ ($path\ p_4$), $minCov_5 = \infty$ ($path\ p_5$), $maxLen_4 = 0$ ($path\ p_4$) and $maxLen_5 = 0$ ($path\ p_5$). After checking the condition of the constraint (*limit*), we update the *path coverage matrix* and the structure *nodes* and apply $k1+k2+3$ level of structural sorting to sort the structure *nodes* (*see* Fig. 3b: Here '1' represents that the fields are sorted in ascending order and '-1' for descending order). The new node is selected *deployedSDNnode*, for SDN deployment. Finally, we compute the *total* SDN *deployment cost* ($totalSDN_{Dcost}$), and *hop coverage* of each path ($path.hop_{cov}$). For each new SDN deployment, the process will continue until a satisfactory condition is attained.

The above discussed two algorithms are designed to "maximize the *coverage,* where *cost* is a constraint." i.e., *MUcPF* is used to maximize the *Pcoverage,* where *cost* is a constraint, and *MMHcPF* maximizes the *Hcoverage,* where *cost* is a constraint. However, we can use these algorithms to solve "minimize the *cost,* where *coverage* is a constraint" problems both for path and hop by correctly changing the objective and constraint values.

TABLE III
MATLAB EXPERIMENT PARAMETERS

Item	Description/Value
Graph type	*Partially mesh*
Weighted matrix selection method	*Random*
Source-destination selection	*Random*
Number of shortest path between *source-destination* pairs	1
Deployment cost of each node	1
Capacity of each node	1
Value of $k1$	5
Value of $k2$	5

V. PERFORMANCE EVALUATION

In this section, we will discuss the experiment setup which is used in this paper to evaluate our proposed algorithms. In this experiment, we considered multiple partially meshed networks where the network does not have a predefined structure for SDN node deployment. Through this experiment, we demonstrated the *SDN coverage* with respect to *cost* and vice versa for each individual deployment of the network. We compared the performance of our algorithms with 3 other algorithms.

We compared our heuristic algorithms with 3 other algorithms: 1) Weighted *coverage* algorithm (*wcov*) [33]: Weight of the node will be given as a priority for SDN deployment, where weight means the number of shortest path passes through that node. However, [33] used this method differently by considering common nodes between sub-domains; 2) Degree *coverage* algorithm (*dcov*) [32], [35]: Degree of the node will be given as a priority for SDN deployment; 3) Random *coverage* algorithm (*rcov*): The nodes are selected randomly independently of traffic matrix. As we discussed in the previous section, we are not considering the *long-path-nodes* in the topology for SDN deployment. If we consider the *long-path-nodes*, it may lead to performance degradation of random *coverage* algorithms and degree *coverage* algorithms.

A. Experiment Setup

To compare the performance of the above-discussed algorithms, we have used MATLAB. Table-III shows the details of the experiment parameter used in the simulated scenario for this work. For this simulation, we have considered partially mesh network. First, we randomly generate a number of matrices of variable size and select the edge nodes (source and destination nodes) randomly. We apply the Dijkstra shortest path algorithm to evaluate the shortest path between every pair of source-destination, assuming one shortest path between every "*source-destination*" pair. The *deployment cost* and capacity of each node in our experiment are assumed to be equal. We used the *nestedSortStruct*() method in our algorithms for structural sorting. The values of $k1$ and $k2$ in our heuristic algorithms are set to be 5 for this experiment.

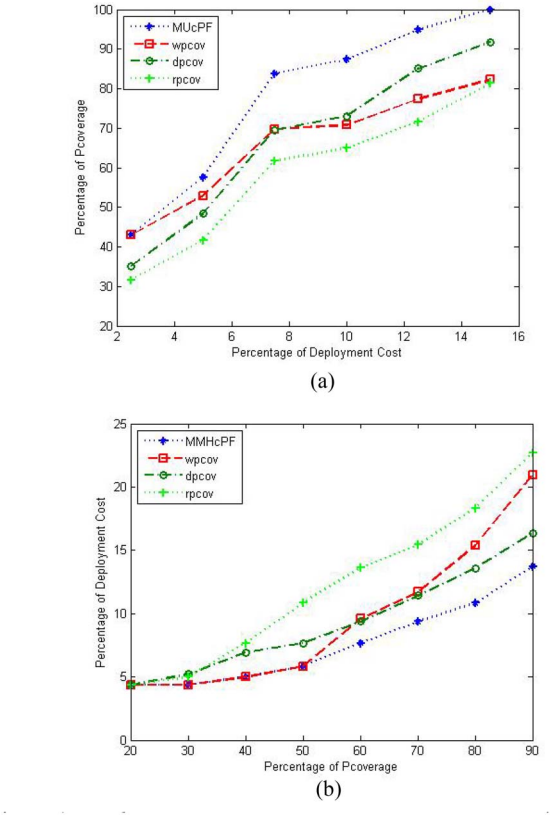


Fig. 4. *Path coverage* performance of randomly generated topologies. (a) Maximum *path coverage* where *cost* as a constraint. (b) Minimum *cost* where *path coverage* as a constraint. *MUcPF* has better performance than other algorithms.

B. Path Coverage Performance

Fig. 4 shows the performance of *path coverage* algorithms under multiple topologies. Fig. 4a presents results of maximum *path coverage*, with *cost* as a constraint and Fig. 4b shows the result of minimum *deployment cost* with *path coverage* as a constraint. The graphs in Fig. 4a clearly show the percentage of *path coverage* of heuristic algorithm (*MUcPF*) is always more than the *weight path coverage* (*wpcov*), *degree path coverage* (*dpcov*) and *random path coverage* (*rpcov*) algorithms for each percentage of deployment. Initially, the percentage of *coverage* of the *MUcPF* algorithm and *wpcov* are equal. As *deployment cost* increases, the percentage of *coverage* in *wpcov* decreases compare to the *MUcPF* method. Since the selection priority of *wpcov* is weight of the nodes, it may select the node from a path, which is already a hybrid path and the selected node may belong to a less number of legacy paths or zero legacy paths. In *dpcov*, the priority is degree, where it will choose the node that has highest degree but may or may not cover maximum number of legacy paths. There may be some nodes with more degree but not belonging to any legacy path. In the Fig. 4a, two curves *wpcov* and *dpcov* cross each other at some point. This is purely dependent upon the node selection method of the respective curves. As the *wpcov* priority is weight, initially, the selected node covers multiple legacy paths, but after certain deployment it may chose the node that will cover less legacy paths (i.e., after 10% deployment) or it may select a node with higher weight but not belonging to any legacy path (i.e., 7.5% to 10% deployment). Initially, *wpcov* covers more legacy paths than *dpcov*. At 7.5% deployment, both have the same *coverage*, where two curves meet. After 7.5% deployment, *dpcov* covers more legacy paths than *wpcov*. Fig. 4a clearly shows that by using the heuristic method with 8% investment, we can achieve 84% of *coverage*, whereas in *wpcov* and *dpcov* only 65% and *rpcov* 60%. Similarly to achieve 100% *coverage MUcPF* takes only 15% of *cost* whereas other algorithms required more. In Fig. 4b, the percentage of *deployment cost* in *MUcPF* is always less than *dpcov* and *rpcov* for each *coverage* constraint. But the *deployment cost* of *wpcov* is almost equal with *MUcPF* up to 50% of *path coverage*, and after that the *deployment cost* of the heuristic algorithm is relatively less than *wpcov*. This result is due to the selection priority of the *weighted coverage* algorithm.

Fig. 5 shows intermediate results of path coverage algorithms. We describe these results as intermediate results because it is based on a single topology (results of Fig. 4 were based on multiple numbers of topologies). In Fig. 5a and 5b, we display the SDN effect with the *path coverage*. The *SDN effect* shows the number of paths to get access to the controller by an SDN node. The "*total SDN effect*" is the summation number of paths in the SDN nodes and its capacity. It varies from algorithm to algorithm depending upon the selection of nodes for SDN deployment and their capacities. For example, suppose we have a network N and we deploy three different types of SDN nodes of capacity x, y, z in the network by applying three different types of algorithms, say A_1, A_2, and A_3. In A_1: deployed SDN nodes are x_1, y_1, and z_1, where each of the nodes belongs to a single path. Therefore, the *total SDN effect* in the case of A_1 is $x{\cdot}1 + y{\cdot}1 + z{\cdot}1$. In A_2: deployed nodes are x_2, y_2, and z_2, where one path passes through x_2 and two paths pass through y_2 and z_2. Therefore, the *total SDN effect* in the case of A_2 is $x{\cdot}1 + y{\cdot}2 + z{\cdot}2$. Similarly, in A_3: deployed nodes are x_3, y_3, and z_3, where x_3 and z_3 belong to two paths each and three paths pass through y_3. Therefore, the *total SDN effect* in this case is $x{\cdot}2 + y{\cdot}3 + z{\cdot}2$. Since we assume all switches in our experiment are equal types having equal capacity, so the curves in Fig. 5a of different algorithms are closer to each other. As *dpcov* and *wpcov* are selecting the nodes that cover multiple paths as a priority, so their *total SDN effect* is quite similar to the *MUcPF* algorithm. The node selection process in *rpcov* is unpredictable as the selection is random. Hence, its *SDN effect* is less than others. In the node selection process of the *MUcPF* algorithm, it is giving priority to the nodes that cover the maximum number of uncovered paths. So the percentage of *path coverage* is getting better after a certain number of deployments. Fig. 5b shows that the *path coverage* of the *MUcPF* algorithm improves after the third deployment compared to other algorithms. The difference between *path coverage* and *SDN effect* is by deploying a new SDN node in a *hybrid path,* if the newly upgraded node does not cover any new *uncovered path,* then the *path coverage* of the network remains constant but the *total SDN effect* will increase. For example, in Fig. 1, if we upgrade node 5 in the *hybrid path* (4-1-2-5-12) of the network to the SDN node, there will be no

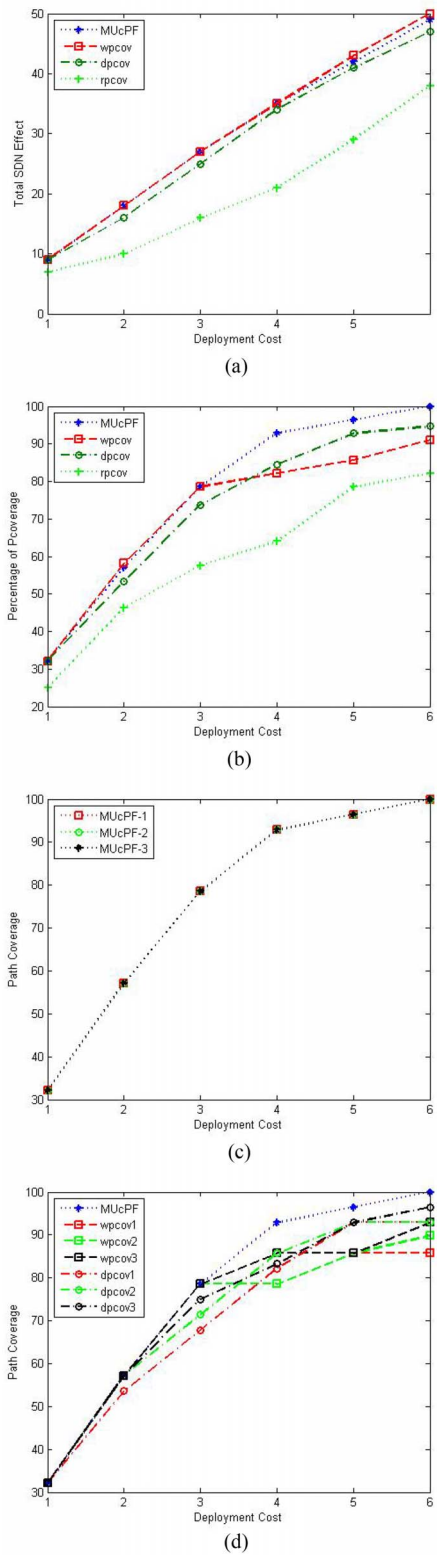


Fig. 5. Intermediate results of *path coverage* performance on a single topology. (a) *Total SDN effect* vs *deployment cost*, (b) *Path coverage* vs *deployment cost*. A comparison between *SDN-effect* and *path coverage* with respect to cost. (c) Results of 3 different execution of the *MUcPF* algorithm. (d) A performance consistency comparison of *MUcPF* with other algorithms. The heuristic algorithm is the most consistent algorithm.

improvement in the *Pcoverage*, but the *total SDN effect* will increase. Fig. 5c and 5d show the performance consistency of *path coverage* algorithms. We run three algorithms (*MUcPF, wpcov, dpcov*) on a single topology, multiple times. All the results of three algorithms are presented together in Fig. 5d, where the results of *wpcov* and *dpcov* vary repeatedly and the *MUcPF* method always gives the same result, as depicted in Fig. 5c. This result is due to the node selection process of *MUcPF* algorithm, which applies multiple levels of sorting to select the nodes. Here we have applied 8 ($k1 = 5$) levels of sorting. If we increase the $k1$ value, we can get more accurate results. But in *wpcov* and *dpcov*, two levels of sorting is applied for node selection due to their node structure. So the probability of node selection is quite random as multiple nodes have the same priority according to their node structure. The *coverage* percentage of the algorithms improve, when they select a node that covers a newly uncovered path and remain constant when selected nodes do not cover any new legacy path. However, the *MUcPF* algorithm always selects a node that covers not only a newly uncovered path but also the maximum number of uncovered paths.

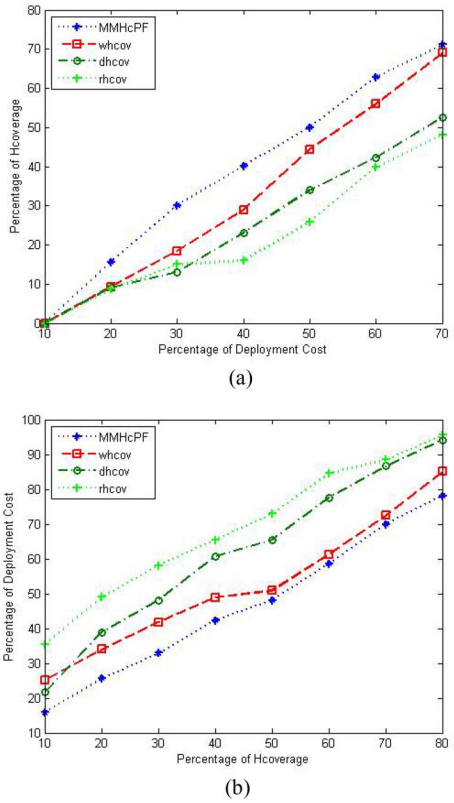


Fig. 6. *Hop coverage* performance under randomly generated topologies. (a) Maximum *hop coverage* where *cost* as a constraint. (b) Minimum *deployment cost* where coverage as a constraint. *MMHcPF* has better performance compared to other algorithms.

C. Hop Coverage Performance

Fig. 6 presents the performance of *hop coverage* algorithms. Fig. 6a shows the results of maximum *hop coverage*, where *cost* is a constraint and Fig. 6b shows the results of minimum *deployment cost,* where *hop coverage* is a constraint. Fig. 6a clearly shows the percentage of *hop coverage*, weight c*overage* (*whcov*), random coverage (*rhcov*) and degree *coverage* (*dhcov*) are always less than the percentage of *hop*

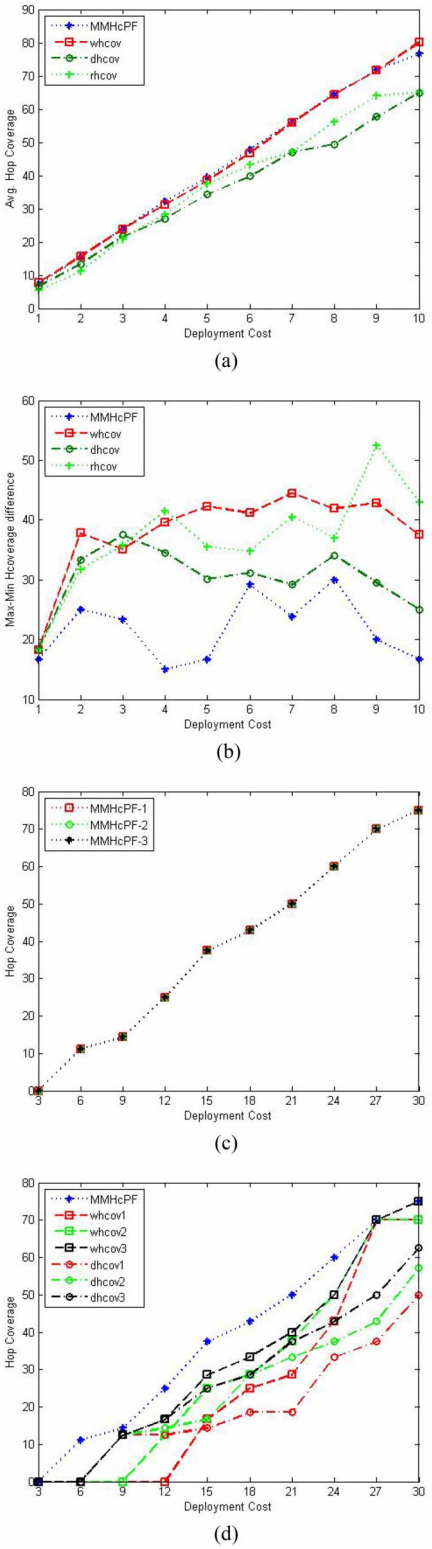


Fig. 7. Intermediate results of *hop coverage* performance on a single topology. (a) *Average hop coverage* vs *cost*, (b) *Max∼Min hop coverage* difference vs *cost*. *MMHcPF* is the most uniform algorithm. c) Results of 3 different execution of the *MMHcPF* algorithm. (d) A performance consistency comparison of the heuristic algorithm with other algorithms. *MMHcPF* is the most consistent algorithm.

coverage of the heuristic algorithm (*MMHcPF*) with respective *deployment cost*. This is because *MMHcPF* applies multiple levels of sorting compared to other algorithms and selects the node that covers the maximum number of less/zero covered paths. In Fig. 6b, the percentage of *deployment cost* of *rhcov* and *dhcov* are always greater than *MMHcPF* for each percentage of *hop coverage*. But the *deployment cost* of the *MMHcPF* algorithm is less than *whcov* at the initial stage, i.e., up to 50% of *coverage*; the *deployment cost* of the *MMHcPF* algorithm is relatively less. As the percentage of *coverage* increases, the two graphs come closer to each other due to the node selection priority of the weighted *hop coverage* algorithm.

Fig. 7 shows intermediate results of *hop coverage* algorithms. In Fig. 7a and 7b, we map the *average hop coverage* of each algorithm with its uniformity. The term "average hop coverage" means, if hop coverage of path p1 is 25% and p2 is 75%, then the average hop coverage of two paths is 50%. The term "uniformity" means all paths should cover uniformly. For example, if there are three paths (p_1, p_2, p_3) in a network, we apply two *hop coverage* algorithms (A_1, A_2) to it. In A_1: *hop coverage* of p_1, p_2 and p_3 are 50%, 0% and 100%, respectively; their *average hop coverage* is 50% and *Max∼Min hop coverage* difference is 100. In A_2: *hop coverage* of p_1, p_2 and p_3 are 40%, 50% and 60%, respectively; their *average hop coverage* is 50% and *Max∼Min hop coverage* difference is 20. Although the "*average hop coverage*" of both the algorithms is equal, we still say that A_2 is more uniform than A_1. Fig. 7a shows that as the deployment increases, the average *hop coverage* of *rhcov* and *dhcov* grow less than *MMHcPF*. But the *average hop coverage* of *MMHcPF* and *whcov* are quite similar. Although the *average hop coverage* of *MMHcPF* and *whcov* are similar, Fig. 7b clearly shows that the *Max∼Min hop coverage* difference of the heuristic algorithm is less than all other algorithms. Hence, the heuristic algorithm is more uniform than other algorithms. Fig.7c and 7d show the performance consistency of *hop coverage* algorithms. We run three algorithms (*MMHcPF, whcov, dhcov*) on a single topology, multiple times. All results of the three algorithms are presented together in Fig. 7d, where the results of *whcov* and *dhcov* vary repeatedly and the *MMHcPF* method always gives the same result as seen in Fig. 7c, because during the node selection process in the *MMHcPF* algorithm, we apply multiple levels of sorting to select the node. Here, we have applied 13 ($k1 = 5$, $k2 = 5$) levels of sorting. If we increase the $k1$ and $k2$ values, we can get more accurate results. But in *whcov* and *dhcov*, two levels of sorting are applied as the node structure contains only two fields, i.e., *sdn* and *weight/degree*. So the probability of node selection is quite random, as multiple nodes have the same priority according to their node structure. The *hop coverage* percentages of paths of the algorithms improve, when they select a node that belongs to a less percentage hop covered path. However, the *MMHcPF* algorithm always selects a node that covers not only a less percentage of hop covered path, but also the maximum number of minimum uncovered paths.

VI. Conclusion

In this paper, we analyzed the *SDN coverage* and *cost* issues in partially deployed SDNs. By formulating budgeted maximum coverage optimization problems, both for

Pcoverage and *Hcoverage*, which are proved to be NP-hard. Then, two heuristic algorithms, *MUcPF* and *MMHcPF* are proposed to solve the *path coverage* and *hop coverage* problems respectively. Results show that, from *path coverage* performance point of view, *MUcPF* is the most economical algorithm to deploy hybrid paths in the network. It takes only 15% investment to achieve 100% *path coverage*, whereas other algorithms take 20-30%. The intermediate results show that *MUcPF* is more consistent and has better SDN effect, as it covered the maximum number of uncovered paths compared with other algorithms. From *hop coverage* scenario, the results show with minimum investment *MMHcPF* can achieve maximum *hop coverage*, as compared to other existing algorithms, as it selects the node that covers the maximum number of uncovered/minimum covered paths. Intermediate results show *MMHcPF* is not only the most consistent but also a uniform algorithm compared to other existing algorithms. *Max∼Min Hcoverage* difference in *MMHcPF* is only 20-30%, whereas, in other algorithms, it varies from 25-55%.

In our future work, we will consider switches having variable cost and capacity to make the deployment more realistic. We will consider dynamic topology for SDN deployment, where nodes are added and deleted regularly.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions.

REFERENCES

[1] N. McKeown *et al.*, "Openflow: Enabling innovation in campus networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, 2008.

[2] N. McKeown, "Software-defined networking," in *Proc. INFOCOM Keynote Talk*, vol. 17. Rio de Janeiro, Brazil, 2009, pp. 30–32.

[3] M. Casado *et al.*, "Ethane: Taking control of the enterprise," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 4, pp. 1–12, 2007.

[4] A. Tootoonchian and Y. Ganjali, "HyperFlow: A distributed control plane for OpenFlow," in *Proc. Internet Netw. Manag. Conf. Res. Enterprise Netw.*, San Jose, CA, USA, 2010, p. 3.

[5] C.-Y. Hong *et al.*, "Achieving high utilization with software-driven WAN," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 15–26, 2013.

[6] S. Jain *et al.*, "B4: Experience with a globally-deployed software defined WAN," in *Proc. ACM SIGCOMM Comput. Commun. Rev.*, Hong Kong, 2013, pp. 3–14.

[7] U. Holzle. (2012). *Opening Address: 2012 Open Network Summit*. [Online]. Available: http://www.opennetsummit.org/archives/apr12/hoelzle-tue-openflow.pdf

[8] S. H. Yeganeh, A. Tootoonchian, and Y. Ganjali, "On scalability of software-defined networking," *IEEE Commun. Mag.*, vol. 51, no. 2, pp. 136–141, Feb. 2013.

[9] M. Ashton, "Ten things to look for in an SDN controller," White Paper, 2013. [Online]. Available: www.necam.com/Docs

[10] S. Sezer *et al.*, "Are we ready for SDN? Implementation challenges for software-defined networks," *IEEE Commun. Mag.*, vol. 51, no. 7, pp. 36–43, Jul. 2013.

[11] M. Jammal, T. Singh, A. Shami, R. Asal, and Y. Li, "Software defined networking: State of the art and research challenges," *Comput. Netw.*, vol. 72, pp. 74–98, Oct. 2014.

[12] V. Yazici, M. O. Sunay, and A. O. Ercan, "Controlling a software-defined network via distributed controllers," in *Proc. NEM Summit, Implementing Future Media Internet Towards New Horiz.* Istanbul, Turkey, Oct. 2012, pp. 16–21.

[13] C. D. Marsan, "IAB panel debates management benefits, security challenges of software-defined networking," *IETF J.*, vol. 8, no. 2, pp. 9–10, Oct. 2012.

[14] H. E. Egilmez, S. T. Dane, B. Gorkemli, and A. M. Tekalp, "Openqos: Openflow controller design and test network for multimedia delivery with quality of service," in *Proc. NEM Summit, Implementing Future Media Internet Towards New Horiz*, Istanbul, Turkey, 2012, pp. 22–27.

[15] J. O. Mikola, "Utilizing northbound API of the SDN stack in Web-based network management," M.S. thesis, Dept. Inf. Technol., Tampere Univ. Technol., Tampere, Finland, 2014.

[16] D. Levin, M. Canini, S. Schmid, and A. Feldmann, "Panopticon: Reaping the benefits of incremental SDN deployment in enterprise networks," in *Proc. USENIX Annu. Tech. Conf. (USENIX ATC)*, Philadelphia, PA, USA, 2014, pp. 333–345.

[17] M. R. Nascimento *et al.*, "Virtual routers as a service: The routeflow approach leveraging software-defined networks," in *Proc. 6th Int. Conf. Future Internet Technol.*, Seoul, South Korea, 2011, pp. 34–37.

[18] S.-Y. Wang, C.-C. Wu, and C.-L. Chou, "Constructing an optimal spanning tree over a hybrid network with SDN and legacy switches," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, Larnaca, Cyprus, 2015, pp. 502–507.

[19] M. Reitblatt, N. Foster, J. Rexford, C. Schlesinger, and D. Walker, "Abstractions for network update," in *Proc. ACM SIGCOMM Conf. Appl. Technol. Architect. Protocols Comput. Commun.*, Helsinki, Finland, 2012, pp. 323–334.

[20] N. Handigol, B. Heller, V. Jeyakumar, D. Maziéres, and N. McKeown, "Where is the debugger for my software-defined network?" in *Proc. 1st Workshop Hot Topics Softw. Defined Netw.*, Helsinki, Finland, 2012, pp. 55–60.

[21] R. Wang, D. Butnariu, and J. Rexford, "OpenFlow-based server load balancing gone wild," in *Proc. Hot-ICE*, Boston, MA, USA, 2011, vol. 11, p. 12.

[22] D. Levin, M. Canini, S. Schmid, and A. Feldmann, "Incremental SDN deployment in enterprise networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 473–474, 2013.

[23] S. Vissicchio, L. Vanbever, and O. Bonaventure, "Opportunities and research challenges of hybrid software defined networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 2, pp. 70–75, 2014.

[24] S. Agarwal, M. Kodialam, and T. V. Lakshman, "Traffic engineering in software defined networks," in *Proc. IEEE INFOCOM*, Turin, Italy, 2013, pp. 2211–2219.

[25] R. Morris, "TCP behavior with many flows," in *Proc. IEEE Int. Conf. Netw. Protocols*, Atlanta, GA, USA, 1997, pp. 205–211.

[26] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, "Hedera: Dynamic flow scheduling for data center networks," in *Proc. NSDI*, vol. 10. San Jose, CA, USA, 2010, pp. 281–296.

[27] M. Mendonca, K. Obraczka, and T. Turletti, "The case for software-defined networking in heterogeneous networked environments," in *Proc. ACM Conf. CoNEXT Student Workshop*, Nice, France, 2012, pp. 59–60.

[28] H. Lu *et al.*, "HybNET: Network manager for a hybrid network infrastructure," in *Proc. Ind. Track 13th ACM/IFIP/USENIX Int. Middleware Conf.*, Beijing, China, 2013, pp. 1–6.

[29] S. Salsano, P. L. Ventre, L. Prete, G. Siracusano, and M. Gerola, "OSHI—Open source hybrid IP/SDN networking (and its emulation on mininet and on distributed SDN testbeds)," in *2014 Third European Workshop on Software Defined Networks*, 2014, pp. 13–18.

[30] M. Kobayashi *et al.*, "Maturing of OpenFlow and software-defined networking through deployments," *Comput. Netw.*, vol. 61, pp. 151–175, Mar. 2014.

[31] S. Vissicchio, L. Vanbever, L. Cittadini, G. Xie, and O. Bonaventure, "Safe updates of hybrid SDN networks," UCL Belgium, Tech. Rep., 2013. [Online]. Available: http://hdl.handle.net/2078.1/134360

[32] Y. Guo, Z. Wang, X. Yin, X. Shi, and J. Wu, "Traffic engineering in SDN/OSPF hybrid network," in *Proc. IEEE 22nd Int. Conf. Netw. Protocols (ICNP)*, Raleigh, NC, USA, 2014, pp. 563–568.

[33] M. Caria, T. Das, A. Jukan, and M. Hoffmann, "Divide and conquer: Partitioning OSPF networks with SDN," in *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 2015, pp. 467–474.

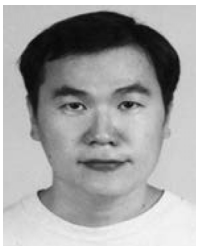
[34] Y. Fu *et al.*, "A hybrid hierarchical control plane for flow-based large-scale software-defined networks," *IEEE Trans. Netw. Service Manag.*, vol. 12, no. 2, pp. 117–131, Jun. 2015.

[35] H. Wang, Y. Li, D. Jin, P. Hui, and J. Wu. "Saving energy in partially deployed software defined networks," *IEEE Trans. Comput.*, vol. 65, no. 5, pp. 1578–1592, May 2016.

[36] S. Khuller, A. Moss, and J. S. Naor, "The budgeted maximum coverage problem," *Inf. Process. Lett.*, vol. 70, no. 1, pp. 39–45, 1999.

[37] C. D. Godsil, G. Royle, and C. Godsil, *Algebraic Graph Theory*, vol. 207. New York, NY, USA: Springer, 2001.

Binayak Kar received the B.Eng. degree in computer science and engineering from Utkal University, Bhubaneswar, India, in 2006, and the M.Tech. degree in computer science and engineering from International Institute of Information Technology (IIIT), Bhubaneswar, India, in 2010. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Information Engineering, College of Electrical Engineering and Computer Science, National Central University, Taiwan. His research interests include network security, cloud computing, software defined networking, and network function virtualization.



Eric Hsiao-Kuang Wu received the B.S. degree in computer science and information engineering from National Taiwan University, Taipei, Taiwan, in 1989, and the M.S. and Ph.D. degrees in computer science from the University of California, Los Angeles, in 1993 and 1997, respectively. He is currently a Professor of Computer Science and Information Engineering with the Department of Computer Science and Information Engineering, National Central University, Chung-Li, Taiwan. His research interests include wireless networks, mobile computing, and broadband networks. He is a member of the Institute of Information and Computing Machinery.

Ying-Dar Lin received the Ph.D. degree in computer science from the University of California at Los Angeles in 1993. He is a Distinguished Professor of Computer Science with National Chiao Tung University, Taiwan. He was a Visiting Scholar with Cisco System, San Jose, from 2007 to 2008. Since 2002, he has been the Founder and the Director of Network Benchmarking Laboratory, which reviews network products with real traffic and has been an approved test laboratory of the Open Networking Foundation (ONF) since 2014. He also cofounded L7 Networks Inc., in 2002, which was later acquired by D-Link Corp. He currently serves on the editorial boards of several IEEE journals and magazines. He published a book entitled *Computer Networks: An Open Source Approach*, with R.-H. Hwang and F. Baker (McGraw-Hill, 2011). His research interests include network security and wireless communications. His work on multihop cellular was the first along this line, and has been cited over 750 times and standardized into the IEEE 802.11s, the IEEE 802.15.5, the IEEE 802.16j, and 3GPP LTE-Advanced. He is an IEEE Distinguished Lecturer from 2014 to 2017 and an ONF Research Associate.