


Article

Delay Bounded Multi-Source Multicast in Software-Defined Networking

Thabo Semong ^{1,2,*} , Kun Xie ¹, Xuhui Zhou ¹, Hemant Kumar Singh ¹ and Zhetao Li ³

¹ College of Computer Science and Electronics Engineering, Hunan University, Changsha 410006, China; xiekun@hnu.edu.cn (K.X.); zxh9682@126.com (X.Z.); hemant.china@outlook.com (H.K.S.)

² College of Science, Botswana International University of Science and Technology (BIUST), PO Box 552 Palapye, Botswana

³ College of Information Engineering, Xiangtan University, Yuhu, Xiangtan 411105, Hunan, China; liztchina@gmail.com

* Correspondence: semongt@biust.ac.bw; Tel.: +86-1887-4877-341

Received: 21 December 2017; Accepted: 19 January 2018; Published: 21 January 2018

Abstract: Software-Defined Networking (SDN) is the next generation network architecture with exciting application prospects. The control function in SDN is decoupled from the data forwarding plane, hence it provides a new centralized architecture with flexible network resource management. Although SDN is attracting much attention from both industry and research, its advantage over the traditional networks has not been fully utilized. Multicast is designed to deliver content to multiple destinations. The current traffic engineering in SDN focuses mainly on unicast, however, multicast can effectively reduce network resource consumption by serving multiple clients. This paper studies a novel delay-bounded multi-source multicast SDN problem, in which among the set of potential sources, we select a source to build the multicast-tree, under the constraint that the transmission delay for every destination is bounded. This problem is more difficult than the traditional Steiner minimum tree (SMT) problem, since it needs to find a source from the set of all potential sources. We model the problem as a mixed-integer linear programming (MILP) and prove its NP-Hardness. To solve the problem, a delay bounded multi-source (DBMS) scheme is proposed, which includes a DBMS algorithm to build a minimum delay cost DBMS-Forest. Through a MATLAB experiment, we demonstrate that DBMS is significantly more efficient and outperforms other existing algorithms in the literature.

Keywords: multicast; software-defined networking; multi-source; steiner tree; routing; bounded delay

1. Introduction

As a promising centralized control architecture, Software-defined networking (SDN) enabled by OpenFlow protocol, allows the control plane to be programmable for effective optimization of the network resources and support various network dynamics [1–3]. This is made possible by separating the control plane and data plane in a network into two main components; controllers (SDN-Cs) and forwarding elements (SDN-FEs) [4–7]. Forwarding elements in switches are flexible to support various forwarding rules for different traffic. One or more controllers are responsible for taking control plane from the switches and installs the corresponding forwarding policies through secured interfaces. Figure 1 illustrate the basic architecture of SDN. The separation of the application logic from the forwarding substrate greatly facilitates the deployment and operation of new services and enables SDN to react gracefully to changing network demands and modifications to the substrate topology [8]. OpenFlow is the most common protocol used in SDN networks. It enables the controller to communicate with all the Network Elements.

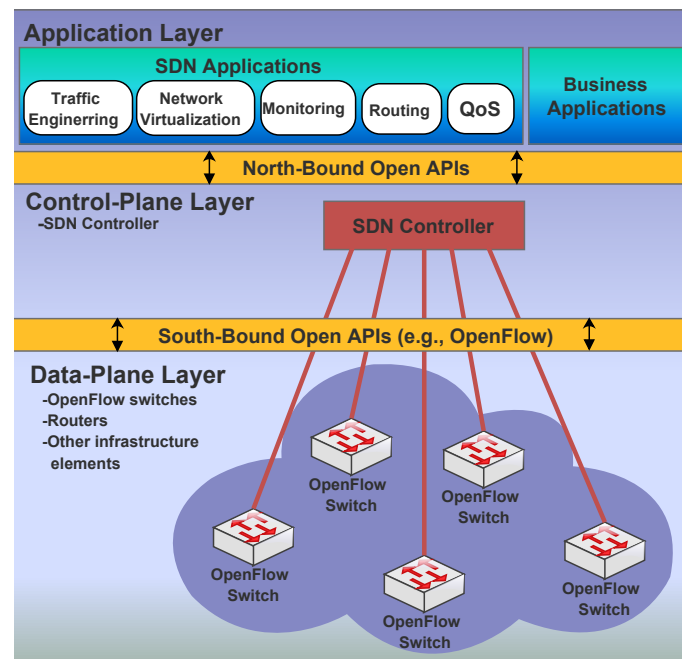


Figure 1. SDN Architecture.

Multicast benefits in saving the bandwidth consumption and reducing the load on the source by avoiding unnecessary duplication of transmissions among a group of independent unicast paths towards destinations, after multiplexing a shared multicast tree. Despite such bandwidth superiority, multicast has not been optimally utilized by the Internet during the past decades. Recently, it has achieved some successful network-level deployments in datacenter networks [9], enterprise networks and IPTV networks [10–13]. Research proposals dealing with the construction of multicasting trees that satisfy various constraints under single source scenario are abundant in the literature. For example, the problem of constructing a single source serving a single multimedia stream wherein minimum delay is desired can be solved efficiently by constructing a single source shortest path tree and pruning subtrees that do not have a destination node. In cases where the delay bound, delay variation bound, node degree bound, and others are desired the multicasting tree construction problems have been shown to be NP-hard [14,15].

One of the existing multicast routing standard protocols is Protocol Independent Multicast Sparse Mode (PIM-SM) [16], which connects the source and destination nodes through a shortest-path tree. PIM-SM presented a solution to the problem of routing multicast packets in large, wide-area internets. This approach uses constrained, receiver-initiated, membership advertisement for sparsely distributed multicast groups. In most multicast techniques, however, the characteristics of a multicast group, like a source node is known in advance. This case is referred to as a deterministic multicast. In many cases, however, the characteristics of a multicast transfer are not known in advance. It is not always necessary for the source of a multicast transfer to be in a specific location, as long as certain constraints are met. A major reason is the widely used content replica design for improving the robustness and efficiency in various networks, such as the content distribution network [17,18], IPTV networks and datacenter networks. In those networks, any node with a replica content can act as a potential source. When delivering a content file to multiple destinations, the source of such a multicast transfer can be any of the replica. This causes multi-source multicast due to the source flexibility. The SDN controller is able to determine which nodes have the replica as it constantly gets updated information regarding the network.

In this paper, we focus on the multicast transfer, the *delay bounded multi-source multicast*, which aims to provide and demonstrate the effectiveness of utilizing more than one source node in an SDN

environment. Given a network topology, a set of potential source nodes, a set of destination nodes and a delay bound, the DBMS scheme build *minimum delay cost trees* with a delay constraint from each potential source node to each destination node, which eventually forms a DBMS-Forest. *Minimum delay cost tree* in this paper refers to a tree that consists of minimum delay cost links from the source node to its multicast nodes. Combining the processes of tree building with a delay bound constraint and potential source node selection is novel and difficult to solve as these processes are closely related and impact each other.

We prove that DBMS problem is NP-Hard. A simple approach would be to divide DBMS as a set of single-source multicasts, each with a different source node. The SMT with the least delay cost among such multicasts just acts as the DBMS of the multi-source multicast. This way, however, suffers from the complexity of solving a set of NP-hard SMT problems. Moreover, the DBMS under non-single sources may have a total cost less than the picked best SMT with a single source. Thus, prior approaches, relying on traditional multicast, remain inapplicable to the DBMS proposed in this paper. To solve the DBMS problem efficiently, we propose a DBMS algorithm which consists of two sub algorithms. The algorithm selects one of the sources to establish routing paths towards its destination nodes. DBMS algorithm seeks shared nodes among such paths to enhance the possibility of aggregating more links. Thus, it can reduce the number of employed links while ensuring that the delivered content reaches each destination within an acceptable delay. Having multiple potential source nodes ensures that the congestion at the sources will be greatly reduced, on the other hand, the delay bound constraint is an important factor that ensures the quality of service. Our contribution in this work can be summarized as follows:

- The DBMS problem is formulated as a mixed integer linear programming (MILP) problem, and the problem is found to be an NP-hard problem.
- To solve the problem, we propose a DBMS algorithm to ensure that a destination node connect to the potential source node through a minimum delay cost path. Out of the source nodes set, the DBMS algorithm selects a particular potential source based on the total delay cost from the destination node. It ensures that a destination node connects to only one source. By selecting the source with less delay cost, our algorithm ensures that the overall delay cost of the DBMS-Forest is minimized. Based on the formulated MILP, our algorithm bound the end-to-end delay cost path from each source to its multicast nodes.
- We conducted extensive simulations to evaluate the proposed technique feasibility and usefulness and also compared the performance of our technique with the previous work in literature. The simulation results demonstrate the effectiveness of our solution.

The rest of this paper is organized as follows; In Section 2 the related work is reviewed and discussed. Section 3 presents the system architecture; other corresponding aspects, including Traffic Delay Cost Measurement, Topology Discovery and Multicast Routing Model, are also described. We present the problem description in Section 4. The DBMS algorithm is designed in Section 5. We present the performance evaluation and results of our experiments in Section 6, and finally concluding remarks are given in Section 7.

2. Related Work

In recent years, a lot of research has been devoted to the development of multicast routing algorithms to construct a delay constrained Steiner tree. S. Li et al. [19] considered an algorithm for constructing delay bounded minimum cost multicast trees. This idea and several other solutions were proposed in the literature. Based on their design objectives, the proposed solutions can be viewed as members of one of two possible classes. The first class solutions include algorithms which are designed to accommodate Internet environments. The second class is more based on algorithms which aim at reducing the cost of the multicast tree, while bounding the end-to-end delay.

Kompella et al. [20] gave the first heuristic for delay constrained minimum Steiner tree problem, normally referred to as KPP. This heuristic computes a constrained closure graph which takes approximation time of $O(|U||V|^3)$. When the link delays and the user specified delay bound (U) takes non-integer values, KPP multiplies out fractional values to get integer and V is the set of vertices. Following this approach, KPP is guaranteed to construct a constrained tree if one exists. The bounded shortest multicast algorithm (BSMA) is another delay-constrained minimum Steiner tree algorithm that is considered the best in terms of tree cost [21]. This algorithm iteratively replaces the edges in the tree until the tree cost cannot be further reduced. BSMA used k -shortest path algorithm to find lower cost edges. The time complexity for BSMA is $O(K|V|^3 \log|V|)$, where K may be very large in cases of large, densely connected networks and it may be difficult to achieve acceptable running time. The approaches of the above mentioned algorithms are limited to a single source on a normal network as opposed to multiple sources in SDN that we have considered in this paper. The utilization of the SDN controller and traffic engineering can make the above algorithms more effective as they are based on the traditional networks. A fast and effective algorithm which approximates the SMT problem using a minimum spanning tree (MST) was presented in [22].

Other algorithms based on greedy strategies achieve better approximation ratio of about 1.693 in [23] and 1.55 in [24]. These methods can incur considerably high time complexity than the MST approximation algorithm. Although the fast SMT algorithm does not achieve the optimal approximation ratio, it significantly saves more computation time than other algorithms. Therefore, it is more suitable for large multicast groups and large-scale networks than other algorithms. Such SMT algorithms, however, remain inapplicable to the DBMS problem proposed by this paper as they are based on a single source multicast. Fortunately, the emergence of SDN makes it possible to realize novel multicast protocols. A new SDN based reliable multicast routing algorithm by S. Shen et al. [24], called RAERA is a typical example. Given any multicast group, RAERA constructs a shortest-path and reliable multicast tree. It enables each destination to retrieve content from at least one reliable recovery node along the path towards the source if necessary.

Table 1 lists some papers in literature based on multicast routing on SDN and their major contributions. Authors in [3] studied the relationship between cost and coverage in the partially deployed SDNs. They deduced that after such partial deployment, the benefits of SDN potentially extend over the whole network through the SDN-enabled nodes. The most related research work involving routing with multi-source was done by Z. Hu et al. [25]. This work involved utilizing the aspect of uncertain source nodes, however it leaves out the important factor of traffic delay constraint that we study in this paper.

Table 1. Some Related Work on Multicast SDN.

Name	TE	Delay	Multi-Source	Focus
DBMC [19]	✓	✓	×	Delay-bound single source multicast
RAERA [24]	✓	✓	×	Recover-aware Steiner Tree
MUCPF [3]	✓	×	×	Partially deployed SDNs
P-MCF [25]	✓	×	✓	Multicast with uncertain sources
This paper	✓	✓	✓	Delay-bound multi-source multicast

Another closely related work was done on multi-source multicast in [26], which deals with delivering a stream from multiple sources to a group of destinations. For each multicast stream, the MMForest algorithm can establish a multicast forest to span all destinations and sources under the constraint of achieving the maximal residual bandwidth. The MMForest is designed based on the existing Widest-Path Forest algorithm, which prefers to select those links with higher residual capacity. This simply means that different multicast streams have to design their MMForests cooperatively. However, most multicast streams are usually independent. The MMForest method focuses on the design constraint of optimizing the residual bandwidth under a set of multicast streams. The MMForest

basic idea does not deviate much from the most straightforward multicast trees. The destination selects one path with the largest residual bandwidth towards all sources, and then merges those selected paths from all destinations. This method is highly probable to loose non-trivial opportunities to maximize the number of shared links among the individual paths.

3. Architecture

The design of the DBMS SDN system is illustrated in Figure 2. Four modules are designed for the DBMS scheme in the control plane.

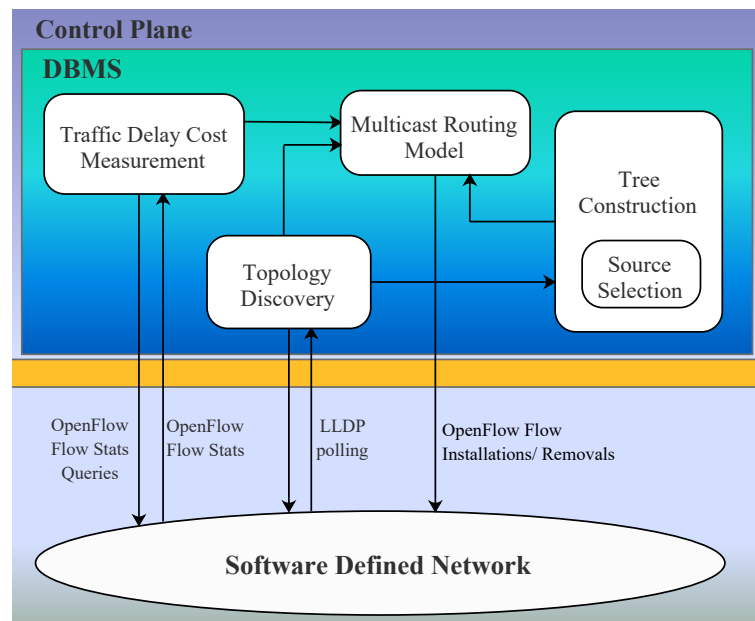


Figure 2. DBMS proposed Architecture.

Traffic Delay Cost Measurement Module: This module is tasked with implementing traffic delay cost on network links using tracking, and reports each link delay cost as defined in Section 4 every time a routing calculation is performed. Traffic delay cost is implemented through periodic polling of all switches in the network with OpenFlow queries (as defined in the OpenFlow v1.0 specification [27]), similar to the approaches presented in [28,29]. The controller reads the counters from all flows on all switches at a fixed measurement interval, analyzes the counters, and generates statistics to report to the network operators. The controller also dynamically adapts the rules based on the counter values from previous measurement intervals. In practice, this interval is limited by how quickly the controller can read the counters from the flows and generate the rules for the next interval. In our experiments, the interval ranges from seconds to minutes. Query times for each switch are randomly staggered, based on the time at which the switch first connected to the controller. This approach has known drawbacks, as continually polling the network introduces potentially unnecessary overhead in the control plane. This implementation could be improved by adopting a passive utilization tracking approach, such as the approach presented in [30]. This module maintains a map of least delay cost links keyed by switch DPID and port number, which the module queries when calculating total delay cost. When calculating link delay cost, it is assumed that all tracked links in the network have a uniform maximum bandwidth since there is no mechanism for querying the maximum bandwidth supported by a particular port in OpenFlow 1.0. This mechanism is provided in later versions of the OpenFlow specification.

Topology Discovery Module: This module implements topology discovery through Link Layer Discovery Protocol (LLDP) polling. By using the information obtained from the topology discovery module we can construct the network topology graph $G(V, E)$, where the node set V corresponds to

the switches and the link set E corresponds to the links. Then, the data that is relative to the topology graph G would be sent to the tree construction module to build up the tree.

Source Node Selection Module: This module work closely with the Tree Construction module since for a tree to be constructed the information about its root is very vital. This module is responsible for selecting a potential source node from the potential source nodes set $|S|$. After the topology has been discovered, this module will get the information about the destination nodes set $|D|$ and the potential source nodes set $|S|$. This information is then used to select the potential source node for each multicast nodes based on the MILP presented in Section 4 using Algorithm 3.

Multicast Routing Model Module: This module implements detection of multicast potential sources, multicast tree calculation, and installation/removal of OpenFlow rules to direct multicast traffic. Least delay path tree calculation is performed for each combination of multicast potential source IP address and destination IP address using DBMS-DelayPath Algorithm 1. Whenever a multicast receiver joins or leaves a multicast group for which a potential source has been identified, the multicast tree calculation and flow installation is performed. Routing is considered separately for each multicast group, and no flow aggregation between separate multicast groups is implemented.

4. Problem Formulation

In this section, we firstly look at the important observations about our DBMS scheme, thereafter we present the problem formulation and prove its NP-hardness. Lastly, we present a mixed integer linear programming model for the DBMS scheme.

Multicast is a natural approach of routing content from a source to a set of destination nodes. Unlike the unicast, it benefits in efficiently saving the bandwidth consumption and reducing the load on the source. The total cost of a multicast tree, which spans the single source and all destinations is widely used as the performance metric of a multicast. In most cases, the cost of multicast tree depends on the amount of links occupied without considering the diversity that exists among the network links. To minimize tree delay cost, given a multicast group, many approximation methods like in [19] have been proposed to construct the SMT. Given the same set of destinations, the delay of a multicast group is sensitive to the selection of the source. In many situations, the content replica design [18] can be used to ensure that the source of a multicast transfer does not necessarily have to be located in a specific location.

A simple method for exploiting such opportunities is to calculate the SMTs for multicast groups with same set of destinations but different sources. When multiple sources participate simultaneously, a more efficient multicast can be formed. The best picked SMT with a single source may not be the optimal selection when using multiple sources. Such a preferred design of having multiple sources participating simultaneously, however, cannot be achieved by the approaches in the literature, which are relying on a traditional single-source multicast. Those approaches remain inapplicable to the DBMS problem we are proposing in this paper. Since we focus on the delay bounded multi-source problem in the data forwarding plane, we assume that each switch connects to the controller with a direct link in this paper.

The topology of the network without the controller is modeled as a di-graph $G(V, E)$, where V denotes the set of nodes (switches), and E denotes the set of edges (links) among the nodes. Each link $e \in E$ is associated with a delay cost $\delta_{(u,v)}$ i.e., for link (u,v) . Given a destination set $D \in V$ and a potential source set $S \in V$, DBMS tries to deliver the content to the whole set of destinations in D from partial or even all potential sources in set S . There is a constraint that each destination in set D must just reach to one and only one potential source in set S . This simply means that when set S contains only one source, then the DBMS is actually equivalent to the deterministic multicast. This does not impose any constraint on the selection of used sources for each destination $d \in D$. Therefore, DBMS will still be a deterministic if all destinations access any $s \in S$ as their common source even if $|S| > 1$. The transmission delay cost of DBMS is basically dominated by the strategies of using such sources, hence, in the following paragraphs we explain the process of constructing a DBMS-Forest.

DBMS-Forest in this case is a product of minimum delay cost trees formed by the DBMS scheme. The DBMS scheme creates DBMS-Forest from trees spanning each destination $d \in D$ with only one source, such that the total delay of utilized links in each tree is minimized. Any pair of source nodes s_1 and s_2 appearing in a multicast tree must be isolated into different trees.

Isolating the source nodes ensures that no destination node can be reached by more than one source node, as this can increase the delay cost of the resultant DBMS-Forest. This will also reduce the congestion at each source. Thus, if there are k potential sources in S , then the DBMS-Forest will consists of k isolated trees, each of which roots at one of those k sources. The DBMS becomes the deterministic multicast when the source set contains only one element. Thus the process of constructing the DBMS-Forest is more similar to constructing a SMT, which is NP-hard problem in a general graph. However, the construction of DBMS-Forest is more challenging than the SMT problem in general setting, because it involves using multiple potential source nodes and also enforce the delay bound constraint to ensures the quality of service.

Theorem 1. *Given a DBMS problem with potential source set $|S|$ and a destination set $|D|$ in a network $G(V, E)$, the challenge of finding the paths for each source-destination pair such that the total delay cost among the links is a minimum is NP-Hard [31].*

Proof. We prove the NP-hardness of constructing our DBMS-Forest by giving a polynomial time reduction from SMT, which is an NP-hard problem. To solve this, the SMT problem for DBMS must be considered first, meaning we have to find an SMT for spanning all the destination nodes and source nodes of that DBMS. Clearly, the optimal solution of this SMT problem cannot be found within polynomial time. Note that $|S|^2$ source pairs exist in the optimal SMT and each source pair are connected by only one path. The path between any two sources contains at least one redundant link. A link is considered to be redundant only if each destination node can still reach at least one source node even after the removal of that link. Therefore, the SMT problem of a DBMS would be reduced to the DBMS-Forest by removing all the redundant links in the optimal SMT. The process of removing potential redundant links can be completed within polynomial time. Then, the remaining links in the original optimal SMT form a DBMS-Forest, where each destination connects to only one source. Therefore, the process of constructing the DBMS-Forest using our DBMS scheme is also NP-Hard. Note that the network flow techniques that was used to find heuristic solutions in [31] cannot be applied to our DBMS scheme as our scheme requires multicasting as explained in the coming sections. \square

Mixed Integer Linear Programming

We design a Mixed Integer Linear Programming (MILP) formulation for the DBMS problem. Let Nv denote the set of all neighbor nodes of node v in G , and u is in Nv if (u, v) is a link from u to v . Let $|S|$ denote the potential source set of the DBMS scheme, and $|D|$ denote the destination set. The DBMS scheme needs to ensure that there is only one path in each tree from every destination node $d \in D$ to only one source node $s \in S$ in output DBMS-Forest. Assuming that path P from the multicast source node $s \in S$ to the destination node $d \in D$ satisfy the delay bound δ . Given $G(V, E)$, S , D and δ , the DBMS scheme's task is to construct a DBMS-Forest, consisting of trees $T(V', E')$, such that $V' \subseteq V$, $E' \subseteq E$, and (i) $\delta(T)$ is minimum, (ii) $\forall d \in D \exists P$ such that P is a path from s to d in T , and $\delta(P) \leq \delta$. To achieve this goal, our DBMS scheme includes the following binary decision variables. All other major notations used in this paper are listed in Table 2.

Let binary variable $\omega_{(d,s)}$ denote whether a destination node d selects a source node s as the root node. Meaning, there is a path from d to s if $\omega_{(d,s)} = 1$. In this setting, let binary variable $\pi_{(d,u,v)}$ denote whether link (u, v) is in the path P from the destination node d to the source node s . Let binary variable $\varepsilon_{(u,v)}$ denote whether link (u, v) is in the output tree T . Intuitively, we should find the path from each destination d to just one source node s with $\omega_{(d,s)} = 1$. Thus, every link (u, v) in the path has $\pi_{(d,u,v)} = 1$.

Let binary variable $\delta_{(u,v)}$ denote a delay that a packet experiences across link (u,v) . Given the above notation, the delay experienced by a packet traversing a path P is defined as $\delta(P) = \sum \delta_{(u,v)} \forall (u,v) \in P$. The routing of the resultant DBMS-Forest with $\varepsilon_{(u,v)} = 1$ for every link (u,v) in the DBMS-Forest can be achieved by the union of the paths from all the destination nodes in D to at least one source node in S . Our goal is to find trees with minimum delay cost as defined in Equation (1) under the delay constraint defined in Equation (2). This means that, we need to solve the following problems;

Table 2. Notations.

Notations	Description
$G(V,E)$	Network topology G , with node set V and link set E
$S \in V$	potential source set
$D \in V$	Destination set
$s \in S$	Each potential source node in set S
$d \in D$	Each destination node in set D
(u,v)	A link from u to v
T	Tree
P	Path
$P_{(s,d)}$	A path from s to d
δ	Delay bound
$\delta_{(u,v)}$	Delay cost across link (u,v)
$\delta(T)$	Delay cost of tree T
$\delta(P)$	Delay cost across path P
Nv	The set of all neighbor nodes of node v in G
$\omega_{(d,s)}$	Denote whether a destination node d selects source node s as the root node
$\omega_{(d,s)} = 1$	There is a path from d to s
$\pi_{(d,u,v)}$	Whether link (u,v) is in the path from the destination d to source s
$\varepsilon_{(u,v)}$	Whether link (u,v) is in the output tree T

$$\min \delta(T) = \sum_{(u,v) \in E} \delta_{(u,v)} \times \varepsilon_{(u,v)} \quad (1)$$

where the weight $\delta_{(u,v)}$ denote the delay of link (u,v) , and $\forall d \in D$

$$\delta(P_{(s,d)}) = \sum_{(u,v) \in P_{(s,d)}} \delta_{(u,v)} < \delta \quad (2)$$

where, $P_{(s,d)}$ is a path on tree T from s to d and (u,v) is a link on $P_{(s,d)}$. The bounded delay requirement is additive over individual paths from every s to its destinations d on the tree T . To ensure the existence of such a tree which satisfies Equation (2), the following condition must hold, where $SDP(s, d)$ is the Shortest Delay Path from source s to destination d :

$$\forall d \in D : \sum_{(\delta) \in SDP(s,d)} \delta_{(u,v)} < \delta \quad (3)$$

Otherwise, a routing tree which can meet the delay bound δ does not exist.

Constraints

To find $\varepsilon_{(u,v)}$, our MILP formulation include the following constraints, which explicitly describe the routing principles for DBMS tree creations;

$$\sum_{s \in S} \omega_{(d,s)} = 1, \forall d \in D \quad (4)$$

For every destination node $d \in D$, the first constraint (4) ensures that there exist only one source node $s \in S$ such that there exists at least one path from d to s in the output tree T . Equations (5)–(7) impose constraints on finding the path from every destination d in D to its source s . More precisely, given any destination node d , s is the source of the path towards d .

$$\sum_{v \in N_s} \pi_{(d,s,v)} = 1, \omega_{(d,s)} = 1, \forall d \in D, \exists s \in S \quad (5)$$

Constraint (5) implies that only one link (s,v) from s to any neighbor node v needs to be selected with $\pi_{(d,s,v)} = 1$. At the same time, it ensures that there exists only one path from s to d , and any pair of source nodes in the output tree T is isolated. On the other hand, every destination node d is the flow destination.

$$\sum_{u \in N_d} \pi_{(d,u,d)} = 1, \omega_{(d,s)} = 1, \forall d \in D, \exists s \in S \quad (6)$$

Constraint (6) ensures that only one link (u,d) from any neighbor node u to d must be selected with $\pi_{(d,u,d)} = 1$.

$$\sum_{v \in N_u} \pi_{(d,u,v)} = \sum_{v \in N_u} \pi_{(d,v,u)} = 1, \omega_{(d,s)} = 1. \quad (7)$$

For any other node u in graph G , we can infer from constraint (7) that it is either located in the path P from s to d or not. If u is located in the path P , then u has one incoming flow in the path with one binary variable $\pi_{(d,v,u)} = 1$ and one outgoing flow in the path with one binary variable $\pi_{(d,u,v)} = 1$.

Otherwise, $\pi_{(d,v,u)} = 0$ and $\pi_{(d,u,v)} = 0$. It is important to note that, according to the objective function, $\pi_{(d,v,u)} = 1$ is set for just one neighbor node v so as to minimize the delay cost of outgoing tree T .

$$\forall d \in D, \exists s \in S, \forall u \in V, u \neq d, u \neq s \quad (8)$$

$$\pi_{(d,u,v)} \leq \varepsilon_{(u,v)}, \forall d \in D, \exists s \in S, \forall (u,v) \in E \quad (9)$$

The last 2 constraints (8) and (9) desires to find the routing of the outgoing tree, i.e., $\varepsilon_{(u,v)}$. More specifically, $\varepsilon_{(u,v)}$ must be 1 if link (u,v) is in the path between at least one pair of source node s and destination node d , i.e., $\pi_{(d,u,v)} = 1$. The output DBMS-Forest is the union of all the trees created.

5. DBMS Routing Algorithm

The DBMS problem is an NP-hard and cannot be solved in polynomial time. Therefore, in this section, we designed DBMS algorithm to solve the problem. The focus is on designing an efficient method to select a potential source node from the source node set S for the multicast nodes,

then construct minimum delay cost trees within a certain delay bound constraint for the DBMS-Forest. The quality of service for the tree construction is enforced by the delay bound constraint.

5.1. Transformation of the Problem

In order for us to solve our problem, we transform the multi-source multicast problem into an ordinary flow network with only one source node. Given a network $G(V, E)$, potential source nodes set S and a destination nodes set D , we introduce a virtual source node to which all the potential source nodes set S members connect to. The potential source nodes are all directly connected to the virtual source node with a delay cost $\delta(u, v) = \infty$. This will ensure that a minimum delay cost path P , from the virtual source node to any destination node d , passes through one of the potential source node s . That particular potential source node s on the minimum delay cost path P , becomes the source node of that particular destination node d .

Example 1. Consider Figure 3 as an illustration of how our DBMS-Forest in a small scale network of undirected graph is formed. Let $G(V, E)$ be the SDN network, where V denotes the set of nodes (switches), E denotes the set of links among the nodes and the virtual source node v . The potential source node set $|S| = (s_1, s_2, s_3, s_4)$ and destination node set $|D| = (a, b, \dots, l)$. Each link is weighted with a delay cost. Since SDN can coexist with any topology that is listed below it, in our example topology we only show SDN nodes (switches) without the controller for simplicity. The introduction of a virtual source node v on this topology enables us to convert the problem of multiple sources into an ordinary flow network with only one source node. All the potential source nodes are directly connected to the virtual source node v with a delay cost $\delta(u, v) = \infty$.

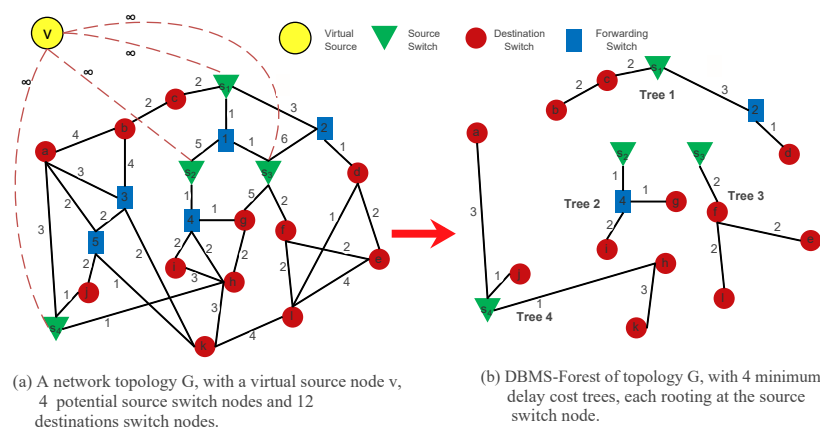


Figure 3. An example of how our DBMS scheme is able to create minimum delay cost trees that form the DBMS-Forest.

All minimum delay cost paths from the virtual source node v to any of the destination nodes have to pass through one of the potential source nodes. That potential source node will then be the ultimate source for that destination node. When our DBMS algorithm is implemented using this topology, the SDN-Controller is able to find the minimum delay cost path for each destination node that satisfy the delay bound δ . In Figure 3b the DBMS-Forest of topology in Figure 3a is displayed and each source node is isolated from other source nodes in the tree. The assumption is that the delay cost is symmetric, i.e., the values are equal for both directions of a link. The source node of each multicast tree becomes the root node. Our Algorithm 3 is able to construct the DBMS-Forest for any kind of topology that satisfy our routing constraints as long as all the necessary inputs are provided.

The main idea of DBMS-DelayPath algorithm (Algorithm 1) is to calculate the minimum delay cost as well as finding the minimum delay path between any two nodes. DBMS-DelayPath is based on the well known Floyd Algorithm. The algorithm firstly create a delay table and then iteratively find the

delay cost between the adjacent nodes (lines 1–9). The sequence table is then created with all the delay paths between any two nodes and the minimum delay (lines 11–18). In Algorithm 1, the complexity of all the iterations is $O(|V|) + O(|V|) + O(|V|^2) + O(|E|) + O(|V|^3) + O(|V|^4) = O(|V|^4)$. Thus the complexity of Algorithm 1 is $O(|V|^4)$. DBMS-MST Algorithm 2 is tasked with creating the minimum spanning tree. In Algorithm 2, the complexity of lines (5–10) is $O((|D| + 1)^2)$. Thus the complexity of Algorithm 2 is $O((|D| + 1)^2)$.

Algorithm 1: *Pseudocode for DBMS-DelayPath*

Data: Delay Table
Result: Minimum Delay and the delay path between any two nodes.

```

1 begin
2   Create a delay table  $D_k$  and sequence table  $S_k$  in  $k^{th}$  iteration
3    $d[i, j]$  is the distance between adjacent vertices
4   Fill each cell  $C[i, j]$  in  $D_k$  using the following conditions
5   if  $d[i, k] + d[k, j] < d[i, j]$ , then
6     | Fill the cell  $C[i, j]$  in  $D_k$  with the value  $d[i, k] + d[k, j]$  of  $D_{k-1}$  table.
7   end
8   else
9     | Fill the cell  $C[i, j]$  in  $D_k$  table with value  $d[i, j]$  of  $D_{k-1}$  table.
10  end
11  On the sequence table
12  if  $C[i, j]$  and  $C[j, i]$  in  $D_k$  and  $D_{k-1}$  are the same, then
13    | Fill  $C[i, j]$  and  $C[j, i]$  in  $S_k$  table with  $S_{k-1}$  value.
14  end
15  else
16    | At the end of the  $k^{th}$  loop, we get the required table with the minimum delay paths.
17  end
18  Return the minimum delay and the delay path between any two nodes
19 end

```

Algorithm 2: *Pseudocode for DBMS-MST*

Data: Array
Result: DBMS-Minimum Spanning Tree

```

1 begin
2   Create an array of size  $v$  and initialize it to NULL
3   Create a min Priority Queue  $H$  of size  $v$ 
4   Insert all vertices into  $H$  such that key value of starting vertex is 0 and other vertices is set to  $\infty$ 
5   if  $H$  is not empty, then
6     |  $u = \text{extractmin}(H)$ .
7   end
8   for every adjacent  $v$  of  $u$ , do
9     | if  $v \in H$ , then
10      | Update key values of  $v$  in  $H$  if delay of link  $(u-v) < u$ .
11    | end
12  end
13  Parrent[ $v$ ] =  $u$ 
14  Return minimum spanning tree
15 end

```

As we suggested in Section 1, the DBMS scheme can be treated as a set of deterministic multicasts, each with a different source. Accordingly, we select the SMTs with the least delay cost to form the DBMS-Forest of the DBMS scheme. This method suffers the complexity of solving a set of NP-hard SMT problems. The DBMS scheme involving multiple sources may cost less total delay cost than the picked best SMT; hence, the prior approaches for traditional multicast remain inapplicable to the proposed DBMS scheme.

For this reason, we propose the DBMS algorithm. Algorithm 3 summarizes the procedure of solving our MILP for Equations (1) and (2). In line 2, Algorithm 3 will create a set Q , then add the virtual source to that set. In line 5, Algorithm 3 will compare the delay cost of path P from the virtual source to each destination node with the input delay bound δ . The algorithm will add to set Q (lines 6–11) all the nodes that satisfy the delay bound, then use Algorithm 2 to find in set Q , the minimum delay cost from each potential source node $s \in S$ to each destination node $d \in D$ according to our MILP presented in Section 4.

Algorithm 3: Pseudocode for DBMS algorithm

Data: $G = (V, E)$, $|S|$, $|D|$ and the δ .

Result: A DBMS-Forest, formed by the minimum delay cost trees rooting at each potential $|S|$, the $|D|$ that do not satisfy the δ .

```

1 begin
2   Create Graph  $G$  with a virtual source node
3   Create set  $Q$  which include the virtual source node
4   Connect all nodes in set  $|S|$  to the virtual source node, with  $\delta(u, v) = \infty$ 
5   Call DBMS-DelayPath to get the minimum delay cost path between the virtual source node
   and  $|D|$ 
6   Compare the total delay cost of the path  $P$ , from the virtual source node to  $|D|$  with the
   delay bound  $\delta$ , according to equations (2) and (3)
7   if  $\delta(P) > \delta$ , then
8     | The path to the destination node does not satisfy the delay bound.
9   end
10  else
11    | Add nodes to set  $Q$ .
12  end
13  Call DBMS-MST on set  $Q$  to get the minimum spanning trees
14  Return the DBMS-Forest
15 end

```

5.2. Complexity Analysis for Algorithm 3

We analyse the time complexity of Algorithm 3 in the following steps. Firstly, we have to calculate the least delay cost paths from all destination nodes to all potential source nodes and among the $|D|$. The number of the least delay cost paths will be $(|S||D| + \frac{|D||D-1|}{2})$. The time complexity of calculating one delay cost path is $O(|V|^2)$ [32]; therefore, the time complexity for a complete graph would be $O((|S||D| + \frac{|D||D-1|}{2}) \times |V|^2)$. The time complexity of forming the minimum spanning tree T is $O((|D| + 1)^2)$. For creating the steiner tree, the time complexity is $O(|V|) + O(|V|) + O(|V|^2) + O(|E|) + O(|V|^3) + O(|V|^4) = O(|V|^4)$. The overall time complexity of our Algorithm 3 therefor is $O(|V|^4) + O((|D| + 1)^2) = O((|V| + 1)^4)$.

6. Performance Evaluations

In this section, we employed both small real networks and massive synthetic networks to evaluate our DBMS scheme by simulations. The implemented prototype was evaluated through network

emulation of representative topologies and workloads using Mininet [33], which is popularly used for emulating OpenFlow network environments. All switches in the Mininet network are implemented using real instances of OpenVSwitch v1.4.6, and links are implemented as paired virtual interfaces. From the perspective of the network controller, a Mininet emulated network is indistinguishable from a physical SDN network. All results presented here were produced by parsing the output logs of the network controller, hence, the same evaluation techniques could be applied to a physical SDN Network. The software components of the experiment are installed on a Lenovo Ideapad Y50-70 laptop with the specifications illustrated in Table 3. Table 4 illustrate the tools and technologies used to carry out the experiment.

Table 3. Hardware specification used for the experiment.

Hardware parts	Specification
CPU	Intel Core i7-4710HQ CPU 2.5 GHz
RAM	10 GB DDRII 1600 MHz
Storage	512 GB SAMSUNG SSD 6Gbps
Operating System	Microsoft Windows 10 64-bit

Table 4. Software Used in implementation of the experiment.

Tools and Technologies	Name	Version	Description
SDN Controller	Pox [34]	2.0.2	Python based open source OpenFlow/SDN Controller.
Simulator	Mininet [33,35]	2.2.1	An emulator for deploying large networks on the limited resources of a simple single Computer or Virtual Machine. Mininet has been created for enabling research in SDN and OpenFlow.
Traffic Generator	Inet [36]	3.0	It generates random networks with characteristics similar to those of the Internet from November 1997 to February 2002, and beyond. The generator should be used to generate network of no less than 3037 nodes, which was the number of ASs on the Internet in November 1997.
Development tools	C++	C++14	A general-purpose programming language with a bias towards systems programming.

6.1. Simulation Setup

As we have shown in Sections 4 and 5, the efficacy of the proposed algorithms does not rely on specific network factors, such as system topologies, source or destination nodes placement and request patterns. IBM ILOG CPLEX Optimizer [37] is adopted to solve the MILP model presented in Section 4. We simulate the routing algorithms with randomized configurations. Through this experiment, we demonstrate the DBMS scheme routing with respect to total delay cost, execution time, multicast nodes and delay bound for each individual deployment of the network. We compared the performance of our DBMS scheme with the following algorithms: (1) slightly modified recover aware edge reduction algorithm (RAERA) [24], which we renamed to mRAERA, (2) the shortest-path tree algorithm (SPT), (3) a Steiner tree (ST) algorithm [38], (4) KPP and (5) Integer Programming solver CPLEX, which finds the optimal solution of the DBMS scheme by solving the MILP formulation presented in Section

4. RAERA was slightly modified by introducing multi-sources since it was mainly based on single source multicast.

The source nodes set $|S|$, destination node set $|D|$ and delay bound (δ) will be given as a priority for SDN deployment [39]. The destination and source nodes sets are selected randomly independent of traffic matrix similar to in [3]. Table 5 shows the detail of the experiment parameters used in the simulated scenario for this paper.

Table 5. Experiment Parameters.

Item	Description/ Value
Graph type	Partially mesh
Network scale	4000 to 10,000 nodes
Weighted matrix selection method	Random
Source-destination selection	Random
Number of shortest delay path between source-destination	1
Deployment cost of each node	1
Number of <i>source nodes</i>	2 to 60
Number of <i>destination nodes</i>	5 to 100
Capacity of each node	1
Delay bound (δ)	40 ms to 9000 ms

6.2. Small Real Networks

Under this subsection, we implement our algorithm on real network called Biznet. In order to observe the performance of DBMS in total delay cost and execution time, we compare it with the solution of the MILP model which is solved using the IBM ILOG CPLEX Optimizer [37]. CPLEX is able to find optimal solutions for small instances for our MILP problem. Biznet is available on the internet topology zoo at <http://www.topology-zoo.org/dataset.html>. The potential source nodes set $|S|$ spans from 2 to 10 and destination nodes set $|D|$ from 5 to 13. We also compare the performance of our DBMS scheme with that of mRAERA, SPT and ST.

Figure 4a shows that in general, as the number of source nodes increases from 2 to 6, the execution time decrease for all the algorithms. The execution time that the SDN controller will take to create the DBMS-Forest reduces because more source nodes enables more destination nodes to connect to the suitable sources faster. As the number of source nodes increases, the probability of more destination nodes being closer to those source nodes becomes higher as well, and hence less links from source to destination node. Our DBMS scheme performs much better than the mRAERA, ST and SPT. The execution time of DBMS is very closer to the optimal solution generated by CPLEX.

We can deduce from Figure 4b that increasing the number of destination nodes increases the total cost delay. As the number of destination nodes increases from 5 to 13, more links are being added to the multicast trees, meaning that the trees generated will be larger. DBMS scheme still manage to find better minimum total delay cost compared to mRAERA, SPT and ST. The total delay cost solution generated by DBMS is very closer to the optimal solution. On the other hand, the total delay cost can be reduced by increasing the number of source nodes as demonstrated by Figure 5. DBMS scheme utilizes the delay bound effect to insure that only the destination-source nodes pair with the minimum delay cost are connected. As the number of source nodes increases from 2 to 10, the total delay cost for mRAERA, SPT and ST decrease as well, but at a slower frequency compared to the DBMS. This demonstrate the positive effects of DBMS on SDN multicast as the number of source node increases. The DBMS solution is very similar to the optimal solution in this regard and reduces the total cost by about 34% when compared to mRAERA.

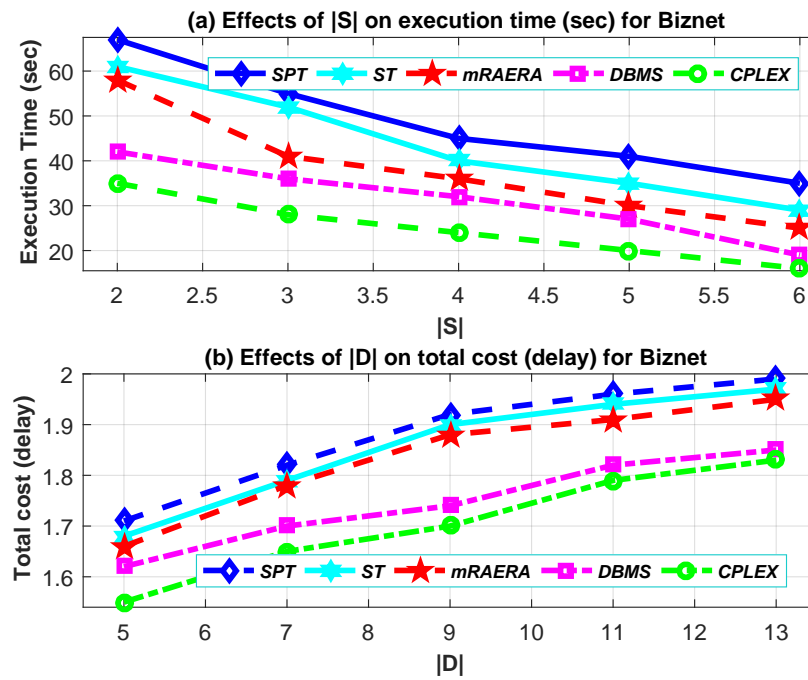


Figure 4. Impact of (a) the number of source nodes $|S|$, and (b) the number of destination nodes $|D|$ on Execution Time (in log-scale) and Total Cost (delay) respectively.

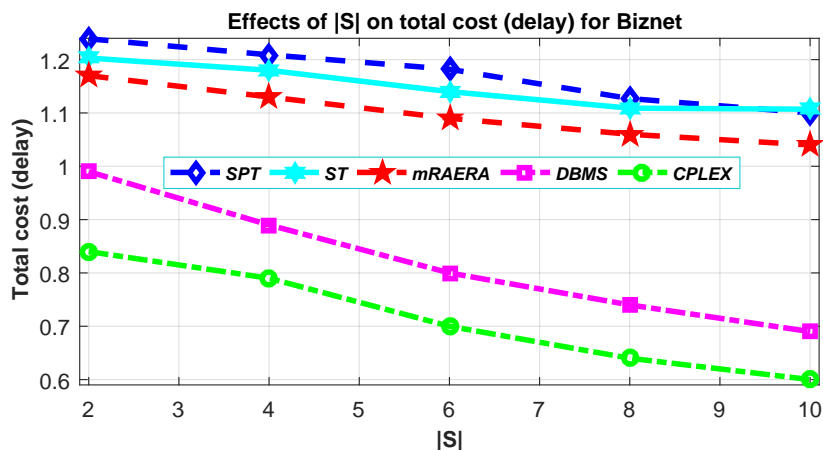


Figure 5. Impact of the number of source nodes $|S|$ on Total Cost (delay).

6.3. Large Synthetic Networks

In this subsection, we used Inet-3.0 [36] to generate large synthetic networks with a mean degree of 4 by default. We then applied the involved algorithms to evaluate the delay cost between every pair of neighbor nodes, assuming one shortest path between every neighbor pair. Network $|V|$ spans from 4000 to 10,000, potential source nodes set $|S|$ from 20 to 60, destination nodes set $|D|$ from 20 to 100 and delay bound (δ) from 20 ms to 9000 ms. All links are considered to be bi-directional with the capacity of 1Gbps for each direction. First, we randomly generate a number of matrices of variable size and select the edge nodes (source and destination nodes) randomly. The performance metrics include (1) total delay cost, (2) delay bound, (3) impact of the number of sources $|S|$ nodes and (4) impact of the number of network size [number of multicast nodes]. Figure 6a depict that the total delay cost for all the algorithms increases with the increase in the delay bound (δ). DBMS performs much better

than mRAERA and other algorithms in this regard. A higher (δ) means that a destination node d is allowed to connect to a source node s along a path with many links. A very low (δ) will result in a very low total delay cost, but with the disadvantage of many source-destination pair path delay cost failing to satisfy the (δ). DBMS performs at about 25% better than mRAERA, SPT, ST and KPP since it pays particular interest on selecting a minimum delay cost path based on the (δ).

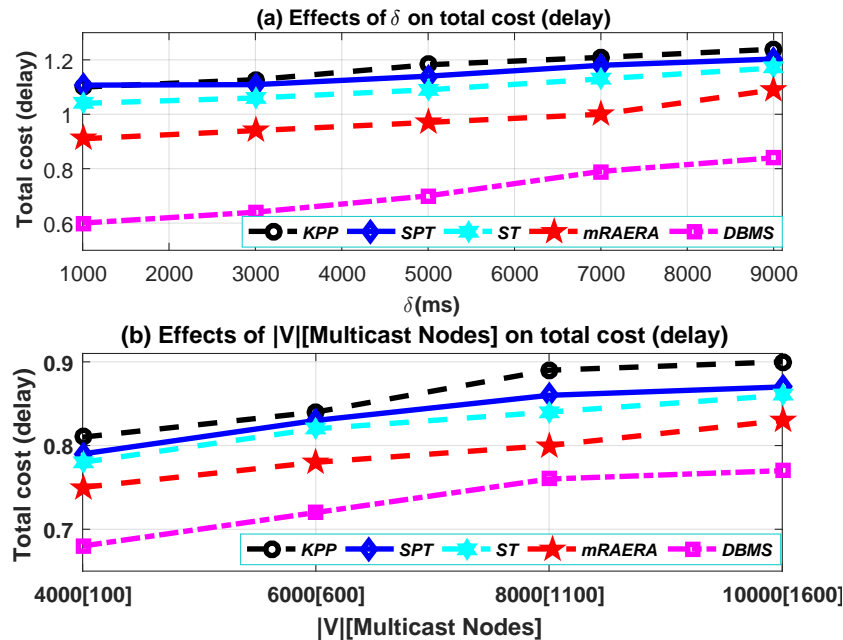


Figure 6. Impact of (a) the delay bound δ , and (b) the network size $|V|$ on the Total Cost (delay).

As the network size $|V|$ and the number of multicast node increases, so is the total delay cost as depicted by Figure 6b. As $|V|$ increases, the expectation is that $|S|$ and $|D|$ will also increase, hence the increase on the total cost delay for all the algorithms is not that huge. The efficacy of DBMS is well demonstrated as it still outperform the other algorithms. Figure 7a clearly indicates that the number of links decreases with the introduction of more source nodes under the entire algorithms investigated. Generally, more source nodes on the network ensure that trees are required to span fewer nodes in a path from source to destination. This result shows the effectiveness of increasing the number of source node on an SDN multicast as compared to having the usual single source multicast. DBMS produces less number of links as it was designed with the intension to utilize the multiple source concept as compared to the other algorithms. As the sources are disconnected into various trees on DBMS, the number of links is greatly reduced as demonstrated by Figure 3 on Section 5.

In Figure 7b the effects of the delay bound (δ) is clearly visible on the number of link created by all the algorithms. Increasing the delay bond (δ) allows a situation where the source-destination pair can be far away from each other, and hence spanning more links. Unlike mRAERA, SPT, ST and KPP, DBMS is designed to accommodate and responds to limited delay bound (δ). DBMS is the best performer on this regard as expected, since the number of links is closely related to the total delay cost.

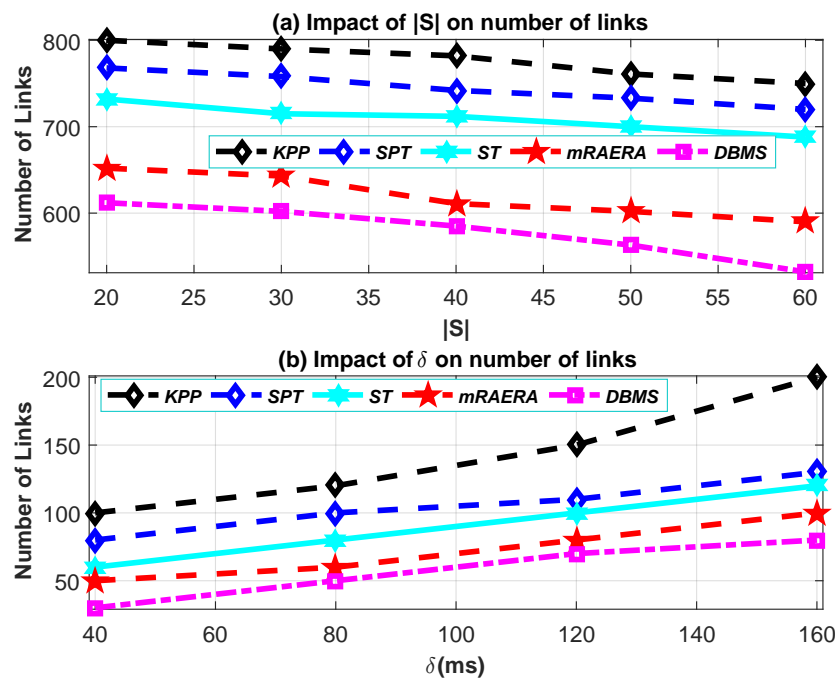


Figure 7. Impact of (a) the number of source nodes $|S|$, and (b) the delay bound δ on Number of Links.

6.4. The Running Time of Involved Algorithms

The time complexity of the algorithms depends not only on the network scale $|V|$, but also on the number of sources and destinations of a given multi-source multicast. In general, the number of sources and destinations is far fewer than the network scale $|V|$. In this section, we further evaluate the running time of our DBMS algorithm and the other involved algorithms in both small scale random networks and large synthetic networks. The network scale varies from 10 to 30 and from 40,000 to 10,000 for both small scale networks and large synthetic networks, respectively. The number of destinations of evaluated multi-source multicasts ranges from 5 to 13 on the small scale Biznet. The default number of sources is 5. For the large synthetic networks, we kept the number of destination nodes constant at 2000, and varied the number of source nodes from 20 to 60. We collected the running time of the algorithms under varied settings of evaluations, and report the average results in Tables 6 and 7 respectively. It is clear from Table 6 that as the number of the destination nodes increase the running time for all the concerned algorithms increases slightly as well. From Table 7 we can deduce that as the number of source nodes increases, the execution time reduces for all the concerned algorithms. Our DBMS scheme performs much better than the other algorithms in this regard.

Table 6. The execution time of the algorithms under varied settings on Biznet.

	$ D = 5, V = 10$	$ D = 7, V = 15$	$ D = 9, V = 20$	$ D = 11, V = 25$	$ D = 13, V = 30$
CPLEX	0.0029 s	0.00403 s	0.00582 s	0.00652 s	0.00948 s
DBMS	0.00712 s	0.00989 s	0.02113 s	0.03632 s	0.04714 s
mRAERA	0.01727 s	0.02981 s	0.04174 s	0.05352 s	0.07023 s
ST	0.04326 s	0.05532 s	0.06831 s	0.07982 s	0.09831 s
SPT	0.06523 s	0.08121 s	0.15735 s	0.18437 s	0.28571 s

Table 7. The execution time of the algorithms under varied settings on a large synthetic networks, ($K = 1000$).

	$ S = 20, V = 4 K$	$ S = 30, V = 5 K$	$ S = 40, V = 6.5 K$	$ S = 50, V = 8 K$	$ S = 60, V = 10 K$
DBMS	0.1973 s	0.17614 s	0.15121 s	0.1421 s	0.1188 s
mRAERA	0.260122 s	0.24104 s	0.2081 s	0.1924 s	0.1745 s
ST	0.2641 s	0.2581 s	0.21218 s	0.1942 s	0.1818 s
SPT	0.2732 s	0.27427 s	0.24692 s	0.20132 s	0.19548 s
KPP	0.30165 s	0.2715 s	0.249922 s	0.2182 s	0.20323 s

7. Conclusions

Recent studies on traffic engineering and multicast protocols on SDN mostly focus on single source multicast. This paper studies the problem of delay bounded multi-source multicast for SDN. The objective is to use multiple source to minimize the total delay cost, and hence increase transmission efficiency, utilizing the content replica design. Compared with unicast, multicast can naturally reduce the bandwidth consumption and reduce the load on the source. We first model our DBMS scheme into MILP problem and prove its NP-hard hardness. We then designed DBMS algorithm to solve the MILP model. Our DBMS algorithm is able to find the minimum delay cost path from each potential source node to its multicast nodes. We have done intensive simulations to evaluate the performance of our DBMS scheme, and our results demonstrate that DBMS scheme tremendously reduce the total delay cost when compared to other schemes. Our algorithm ensures that the execution time of our DBMS scheme is much lower than that of the modified-RAERA, STP, ST and KPP, and is very closer to the optimal solution. DBMS scheme has been tested for different network topologies, delay bounds, network sizes, $|S|$ and $|D|$ sets. These parameters do affect the simulation results to various degrees. However, DBMS performs well in all situations when compared to other algorithms.

Acknowledgments: This work was supported by the Department of Tertiary Education Financing (DTEF) in Botswana and the Botswana International University of Science and Technology (BIUST).

Author Contributions: Thabo Semong conceived and wrote the paper; Thabo Semong, Xuhui Zhou and Hemant Kumar Singh performed experiments; Kun Xie and Zhetao Li provided technical support and revised the paper.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

SDN	Software-Defined Network
DBMS	Delay Bounded Multi-Source
LLDP	Link Layer Discovery Protocol
TE	Traffic Engineering
PIM-SM	Protocol Independent Multicast Spare Mode
MILP	Mixed-Integer Linear Programming
SMT	Steiner Minimum Tree
BSMA	Bounded Shortest Multicast Algorithm
MST	Minimum Spanning Tree
RAERA	Recover Aware Edge Reduction Algorithm
DBMC	Delay-Bound Minimum-Cost
MUCPF	Maximum number of Uncovered Path First
E-MCF	Efficient-Minimum Cost Forest

References

1. Kitsuwat, N.; McGettrick, S.; Slyne, F.; Payne, D.B.; Ruffini, M. Independent Transient Plane Design for Protection in OpenFlow-Based Networks. *J. Opt. Commun. Netw.* **2015**, *7*, 264–275.
2. Huang, H.; Guo, S.; Li, P.; Liang, W.; Zomaya, A.Y. Cost minimization for rule caching in software defined networking. *IEEE Trans. Parallel Distrib. Syst.* **2016**, *27*, 1007–1016.

3. Kar, B.; Wu, E.H.K.; Lin, Y.D. The budgeted maximum coverage problem in partially deployed software defined networks. *IEEE Trans. Netw. Serv. Manag.* **2016**, *13*, 394–406.
4. McKeown, 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.* **2008**, *38*, 69–74.
5. Casado, M.; Freedman, M.J.; Pettit, J.; Luo, J.; McKeown, N.; Shenker, S. Ethane: Taking control of the enterprise. *ACM SIGCOMM Comput. Commun. Rev.* **2007**, *37*, 1–12.
6. Kuo, J.J.; Shen, S.H.; Yang, M.H.; Yang, D.N.; Tsai, M.J.; Chen, W.T. Service Overlay Forest Embedding for Software-Defined Cloud Networks. *arXiv* **2017**, arXiv:1703.09025.
7. Hampel, G.; Steiner, M.; Bu, T. Applying software-defined networking to the telecom domain. In Proceedings of the 2013 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Turin, Italy, 14–19 April 2013; pp. 3339–3344.
8. Hu, Y.; Wang, W.; Gong, X.; Que, X.; Cheng, S. Balanceflow: Controller load balancing for openflow networks. In Proceedings of the 2012 IEEE 2nd International Conference on Cloud Computing and Intelligent Systems (CCIS), Hangzhou, China, 30 October–1 November 2012; Volume 2, pp. 780–785.
9. Li, D.; Xu, M.; Liu, Y.; Xie, X.; Cui, Y.; Wang, J.; Chen, G. Reliable multicast in data center networks. *IEEE Trans. Comput.* **2014**, *63*, 2011–2024.
10. Mahimkar, A.A.; Ge, Z.; Shaikh, A.; Wang, J.; Yates, J.; Zhang, Y.; Zhao, Q. Towards automated performance diagnosis in a large IPTV network. *ACM SIGCOMM Comput. Commun. Rev.* **2009**, *39*, 231–242.
11. Guo, D.; Xie, J.; Zhou, X.; Zhu, X.; Wei, W.; Luo, X. Exploiting efficient and scalable shuffle transfers in future data center networks. *IEEE Trans. Parallel Distrib. Syst.* **2015**, *26*, 997–1009.
12. Cha, M.; Rodriguez, P.; Crowcroft, J.; Moon, S.; Amatriain, X. Watching television over an IP network. In Proceedings of the 8th ACM SIGCOMM Conference on Internet Measurement, Vouliagmeni, Greece, 20–22 October 2008; pp. 71–84.
13. Hei, X.; Liang, C.; Liang, J.; Liu, Y.; Ross, K.W. A measurement study of a large-scale P2P IPTV system. *IEEE Trans. Multimedia* **2007**, *9*, 1672–1687.
14. Vik, K.H.; Halvorsen, P.; Griwodz, C. Evaluating Steiner-tree heuristics and diameter variations for application layer multicast. *Comput. Netw.* **2008**, *52*, 2872–2893.
15. Banik, S.M.; Radhakrishnan, S.; Sekharan, C.N. Multicast routing with delay and delay variation constraints for collaborative applications on overlay networks. *IEEE Trans. Parallel Distrib. Syst.* **2007**, *18*, 421–431.
16. Deering, S.; Estrin, D.; Farinacci, D.; Handley, M.; Helmy, A.; Jacobson, V.; Liu, C.G.; Sharma, P.; Thaler, D.; Wei, L. Protocol Independent Multicast-Sparse Mode (PIM-SM): Motivation and Architecture. *Ann Arbor* **1996**, *1001*, 48109.
17. Chun, B.G.; Wu, P.; Weatherspoon, H.; Kubiawicz, J. ChunkCast: An Anycast Service for Large Content Distribution. In Proceedings of the 5th International Workshop on Peer-To-Peer Systems (IPTPS), Santa Barbara, CA, USA, 27–28 February 2006.
18. Presti, F.L.; Petrioli, C.; Vicari, C. Dynamic replica placement in content delivery networks. In Proceedings of the 13th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, Atlanta, GA, USA, 27–29 September 2005; pp. 351–360.
19. Li, S.; Melhem, R.; Znati, T. An efficient algorithm for constructing delay bounded minimum cost multicast trees. *J. Parallel Distrib. Comput.* **2004**, *64*, 1399–1413.
20. Kompella, V.P.; Pasquale, J.C.; Polyzos, G.C. Multicast routing for multimedia communication. *IEEE/ACM Trans. Netw.* **1993**, *1*, 286–292.
21. Zhu, Q.; Parsa, M.; Garcia-Luna-Aceves, J. A source-based algorithm for delay-constrained minimum-cost multicasting. In Proceedings of the INFOCOM'95. Fourteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Bringing Information to People, Boston, MA, USA, 2–6 April 1995; pp. 377–385.
22. Kou, L.; Markowsky, G.; Berman, L. A fast algorithm for Steiner trees. *Acta Inform.* **1981**, *15*, 141–145.
23. Robins, G.; Zelikovsky, A. Improved Steiner tree approximation in graphs. In Proceedings of the 2000 Citeseer Conference on Discrete algorithms (SODA), San Francisco, CA, USA, 9–11 January 2000; pp. 770–779.
24. Shen, S.H.; Huang, L.H.; Yang, D.N.; Chen, W.T. Reliable multicast routing for software-defined networks. In Proceedings of the 2015 IEEE Conference on Computer Communications (INFOCOM), Hong Kong, China, 26 April–1 May 2015; pp. 181–189.

25. Hu, Z.; Guo, D.; Xie, J.; Ren, B. Multicast routing with uncertain sources in software-defined network. In Proceedings of the 2016 IEEE/ACM 24th International Symposium on Quality of Service (IWQoS), Beijing, China, 20–21 June 2016; pp. 1–6.
26. Robins, G.; Zelikovsky, A. Minimum steiner tree construction. In *The Handbook of Algorithms for VLSI Physical Design Automation*; CRC Press: Boca Raton, FL, USA, 2009; pp. 487–508.
27. Consortium, O.S.; Heller, B.; Pfaff, B.; Talayco, D.; Erickson, D.; Gibb, G.; Appenzeller, G.; Tourrilhes, J.; Pettit, J.; Yap, K.K.; et al. OpenFlow Switch Specification Version 1.0.0. 2009. Available online: <http://www.archive.OpenFlow.org/documents/OpenFlow-spec-v1.0.0.pdf> (accessed on 12 July 2017).
28. Jose, L.; Yu, M.; Rexford, J. Online Measurement of Large Traffic Aggregates on Commodity Switches. In Proceedings of the 11th USENIX Conference on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services(Hot-ICE), Boston, MA, USA, 29 March 2011; USENIX Association: Berkeley, CA, USA, 2011; p. 13.
29. Tootoonchian, A.; Ghobadi, M.; Ganjali, Y. OpenTM: Traffic matrix estimator for OpenFlow networks. In *International Conference on Passive and Active Network Measurement*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 201–210.
30. Yu, C.; Lumezanu, C.; Zhang, Y.; Singh, V.; Jiang, G.; Madhyastha, H.V. Flowsense: Monitoring network utilization with zero measurement cost. In *International Conference on Passive and Active Network Measurement*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 31–41.
31. Kar, K.; Kodialam, M.; Lakshman, T.V. Minimum interference routing of bandwidth guaranteed tunnels with MPLS traffic engineering applications. *IEEE J. Sel. Areas Commun.* **2000**, *18*, 2566–2579.
32. Zhong, C.; Malinen, M.; Miao, D.; Fränti, P. A fast minimum spanning tree algorithm based on K-means. *Inf. Sci.* **2015**, *295*, 1–17.
33. Team, M. Mininet: An instant virtual network on your laptop (or other PC). Mininet. 2012. Available online: <http://mininet.org> (accessed on 1 May 2017).
34. Kaur, S.; Singh, J.; Ghumman, N.S. Network programmability using POX controller. In Proceedings of the ICCCS International Conference on Communication, Computing & Systems, Chennai, India, 20–21 February 2014; p. 138.
35. Kaur, K.; Singh, J.; Ghumman, N.S. Mininet as software defined networking testing platform. In Proceedings of the International Conference on Communication, Computing & Systems (ICCCS), Chennai, India, 20–21 February 2014; pp. 139–142.
36. Winick, J.; Jamin, S. *Inet-3.0: Internet Topology Generator*; Technical Report CSE-TR-456-02; University of Michigan: Ann Arbor, MI, USA, 2002.
37. ILOG, I. CPLEX Optimizer. *En ligne*. 2012. Available online: <http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer> (accessed on 14 June 2017).
38. Takahashi, H. An approximate solution for the Steiner problem in graphs. *Math. Japonica* **1980**, *24*, 573–577.
39. Caria, M.; Das, T.; Jukan, A. Divide and conquer: Partitioning OSPF networks with SDN. In Proceedings of the 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM), Ottawa, ON, Canada, 11–15 May 2015; pp. 467–474.



© 2018 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).