Efficient Multicast Algorithms for Scalable Video Coding in Software-Defined Networking

Jang-Ping Sheu, Che-Wei Chang, Yeh-Cheng Chang
Department of Computer Science
National Tsing Hua University
Hsinchu, 30013, Taiwan
sheujp@cs.nthu.edu.tw, hatsunerika@gmail.com, jas1123kimo@gmail.com

Abstract- **Software-Defined Networking (SDN) is a new approach to design, build and manage computer networks. Multicast is used to transmit the same video file to different users. In this paper, we propose two multicast algorithms to solve the multicast problem in SDN environment. Both algorithms consider the balance of bandwidth utilization and communication delay between the source and clients. Simulation results show that our algorithms can improve the network bandwidth utilization and successful rate of the multicast requests than the previous works.**

Keywords—Bandwidth utilization, multicast, routing algorithms, software-defined networking

I. INTRODUCTION

Software-Defined Networking (SDN) is a new approach for designing, building and managing of networks, which offers flexibility and programmability by decoupling the control plane from the data plane when compared to traditional networks [1]. A central controller at the control plane includes the network topology; network devices only have data plane, which is controlled under the central controller [2, 4]. OpenFlow [3] is the first standard communication interface between control plane and data plane of SDN architecture.

MPEG H.264 is a popular video compression system and dominates the JPEG for image compression [4]. The H.264/AVC has a new technology named scalable video coding (SVC) [5]. In SVC, each video frame is encoded to a base layer and multiple enhancement layers [6]. We can deliver different video qualities to different clients based on their demand. Base layer is a basic quality of a video. A client receiving more enhancement layers can largely increase the video quality of the client [7]. In this paper, we study the problem of multicast of video in SDN with SVC technology.

Multicast is the communication between a single source and multiple destinations in a network with focus on to minimize the network load and the data flow stored in routers or switches. We can use multicast gainfully in video transmission, online gaming, and distributed computing [8]. There are two main schemes to establish a multicast tree, source based tree (SBT) and share tree. SBT computes the shortest paths between the source and all destinations to establish a multicast tree from source to destinations [9, 10]. In shared tree, only one tree needs to be established for a group which is shared by all the sources and destinations within that group. The shared tree approaches can be classified into Core-based tree algorithms [11] and Steiner Tree based algorithms [12, 13].

With the traditional network architecture, we want to establish a multicast spanning tree, i.e. a multicast tree without loop. The multicast generation tree has been formulated as computing a directed Steiner tree of minimal cost [12]. In [13], the authors proposed a shared-tree-based multi-source multicast routing protocol to establish the multi-source multicasting in Mobile Ad Hoc Networks. The multicast routing algorithm as presented in [14] to minimize the size of routing tree in SDN based data-center system. A video multicast framework is proposed in [9] using SVC for TDMA-based Wireless Mesh Networks, where receivers have their own video demand quality. In [10], the application-network cross-layer design is interpreted as an incorporation of application intelligence into the network, and proposed an application-oriented multicast protocol.

In [15], the authors proposed three methods to generate a data aggregation multicast tree in wireless sensor networks. In the first scheme, Center at Nearest Source (CNS), the source which is nearest the sink acts as the aggregation point. All other sources send their data directly to this source which then send the aggregated information to the sink. In the second scheme, Shortest Path Tree (SPT), each source sends its data to the sink along the shortest path between the source and sink. Then combine these shortest paths to form a multicast aggregation tree. In the last scheme, Greedy Incremental Tree (GIT), the multicast aggregation tree is built sequentially. At the first step the tree consists of only the shortest path between the sink and the nearest source. At each following step, the next source closest to the current tree is connected to the multicast tree.

In this paper, we propose algorithms to solve the multicast video problem in SDN environment. We assume that there is a source server that wish to send an SVC video to a number of clients. Each client has its demand of video quality. The central controller of a SDN has the entire network topology, traffic cost, and the remaining bandwidth of each switch link. We propose two algorithms to solve the multicast tree problem. The objective of the first algorithm is to minimize the communication delay of source to clients under the delay-time constraint of each client. The objective of the second algorithm is to find the maximum bottleneck bandwidth (MBB) from source to each client with the minimal delay path. The

978-1-4673-6782-0/15/$31.00 ©2015 IEEE

bottleneck bandwidth of a path is the minimal residual bandwidth of the edges in the path. Simulation results show that our algorithms have better link utilization and multicast successful rate than previous multicast algorithms in [15].

II. Preliminary

2.1 Network Model

In our multicast system, each client has the base layer of the same video sent from the source server. However, a client can request enhancement layers to improve its video quality, but consumes more network bandwidth to deliver the enhanced packets to the client. We assume there are m layers of SVC video quality, a base layer and m-1 enhancement layers. Let w_1 be the request video bandwidth with only the base layer and w_{i+1} represent the request video bandwidth with base layer and i more enhancement layers, for $1 \le i \le m$-1. For example, in Fig. 1, w_3 means the bandwidth with base layer and two more enhancement layers.

We consider an SDN has a set of n switches and a central controller. In our network model, a multicast group contains a source server and a number of clients. Each switch link has a remaining bandwidth value. An SDN network can be represented as a weighted graph $G = (V, E)$ in which V is a set of vertices and E is a set of edges interconnected vertices in V. Each vertex in V represents a switch in SDN and each edge in E represents a switch link in SDN. We use the notation (u, v) to indicate an edge between vertex u and vertex v. For example, in Fig. 1, S is the source server which can send an SVC video with different qualities to four clients c_1, c_2, c_3, and c_4 through 15 switches. For each edge $(u, v) \in E$, $b(u, v)$ denotes the remaining bandwidth of the switch link. Each client c_i has a bandwidth request based on its requested video quality. In Fig. 1, $b(s_1, s_2)$ =14 and the bandwidth requests of the clients c_1, c_2, c_3, and c_4 are 8, 8, 5, and 2, respectively.

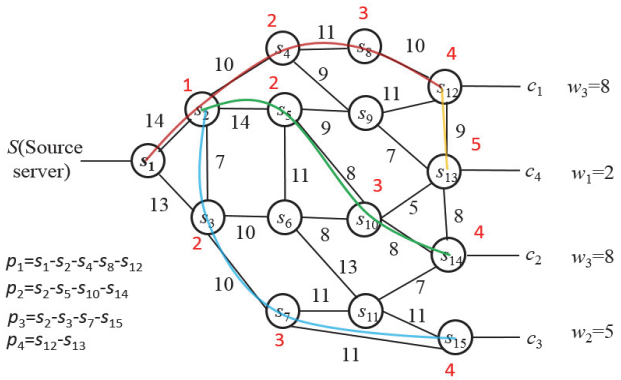

Fig. 1 An example of using MPCA-k in a switching network.

2.2 Problem Formulation

In the first algorithm, we want to find a minimum communication delay (shortest path) from source to each client. If we have multiple paths with the same communication delay, we select the path which has the MBB. In order to reduce the total communication cost, we may not select the shortest path as the multicast path between source and its clients. The total communication cost represents the total number of switch links used to forward a multicast request. Thus, additional k hops of a path from the source to each client other than the shortest path is acceptable. That is, if there exist some paths from the source to some clients, we can join a new client to the existed paths to reduce the total communication cost if the hop count from the source to the client is smaller than or equal to the maximum allowable delay cost. The maximum allowable delay cost of a client is the sum of k and the cost of the shortest path from source to the client. Thus, the new established path from source to the new client may be not the shortest one and k is named the extra delay-time constraint. In the second algorithm, we want to establish a multicast tree with the MBB. For each client, we will find a path with the MBB. If more than one path has the same MBB, we choose the one with the shortest path.

III. Our Proposed Algorithms

In this section, we present two algorithms to solve the multicast problem. First, we present a Minimal Path Cost Algorithm with the extra delay time k (MPCA-k) in section 3.1 to minimize the communication cost from source to clients under the additional delay-time constraint. Second, we present a Maximum Bandwidth Utilization Algorithms with Shortest Path (MBUA-SP) in section 3.2 to maximize the link utilization.

3.1 Minimal Path Cost Algorithm-k (MPCA-k)

In MPCA-k, we first sort the clients by their bandwidth requests in decreasing order. Without loss of generality, we assume n clients are sorted in the order c_1, c_2, c_3, …, c_n. For example in Fig. 1, we sort the clients as c_1, c_2, c_3, and c_4 according to their bandwidth requests. Then, remove the edges in the original network whose residual bandwidth is smaller than the first client bandwidth requirement. Later, find a shortest path (hop count) by the breadth first search (BFS) algorithm [16] from source to the first client. If there are more than one shortest path, select the path with MBB. For each vertex $v \in V$, $m(v)$ denotes the MBB of the path from source S to v. Initially, set $m(S) = \infty$ and for each non-source vertex $x \in V$, $m(x) = 0$. In our BFS algorithm, for each u, $v \in V$, when we find outgoing edges of u to v, if $m(v) < \min\{m(u), b(u, v)\}$, the value of $m(v)$ will be updated as $\min\{m(u), b(u, v)\}$ because we can reach v through u with a larger MBB than without through u. For example, in Fig. 1, for each vertex $x \in V - \{s_1\}$, $m(x) = 0$, $m(s_1) = \infty$. At the first iteration, we find outgoing edges of s_1 and update $m(s_2) = 14$ and $m(s_3) = 13$. Next, find outgoing edges of s_2 and s_3, then update their neighboring switches. We finish the algorithm when all reachable switches are visited. Note that, the red number above each switch represents number of hop counts from source to this switch.

After the BFS algorithm, the three paths s_1-s_2-s_4-s_8-s_{12}, s_1-s_2-s_4-s_9-s_{12}, and s_1-s_2-s_5-s_9-s_{12} have the same minimal hop count from S to c_1. We choose the path s_1-s_2-s_4-s_8-s_{12} which has the largest MBB=10 as the initial multicast tree. Let P denote the set of switches in the selected shortest path. For example, in Fig. 1 P is equal to $\{s_1, s_2, s_4, s_8, s_{12}\}$. Let t_i be the multicast tree including clients c_1, c_2, … c_i, for $1 \le i \le n$. Thus, the shortest path from source to c_1 is t_1. Next, we remove the edges in the original network whose residual bandwidth is smaller than the second client bandwidth requirement. Then,

use the BFS algorithm to count the shortest hop count from c_2 to each switch in the set P. Later, we connect c_2 to the nearest switch in P if the length of c_2 to the switch plus the length of the switch to source is no longer than the maximum allowable delay-time of the client c_2. For example, in Fig. 1, we can find that the shortest path cost from c_2 to each switch in $P = \{s_1, s_2, s_4, s_8, s_{12}\}$ are 4, 3, 4, 3 and 2. If the extra delay-time $k = 2$, we can connect c_2 to s_{12} which has the shortest path from c_2 to the existed tree t_1 because the delay-time from S through s_{12} to c_2 is 6 no larger than the maximum delay-time of the client c_2.

However, if the extra delay-time $k = 1$, we will connect c_2 to switches s_2 or s_8. Once the path from c_2 to set P is determined, combine the path to t_1 and get a new multicast tree t_2. The set P is updated to the set of switches used in t_2. For example, in Fig. 1 if $k = 1$ and s_2 is selected, the P is updated to $\{s_1, s_2, s_4, s_5, s_8\ s_{10}, s_{12}, s_{14}\}$. Finally, repeat the above steps to find the multicast tree t_n. If we cannot find a path from source server to any client, we return no path and deny this request. In Fig. 1, we find four multicast paths for clients c_1, c_2, c_3, and c_4 in $p_1 = s_1$-s_2-s_4-s_8-s_{12}, $p_2 = s_2$-s_5-s_{10}-s_{14}, $p_3 = s_2$-s_3-s_7-s_{15}, and $p_4 = s_{12}$-s_{13}, respectively.

In the following, we evaluate the time complexity of MPCA-k. First, the time complexity of sorting the clients in decreasing order is $O(n \log n)$ time, where n is the number of clients. The time complexity to eliminate the edges whose residual bandwidth is smaller than the selected client's request bandwidth is $O(|V|)$. Next, we use BFS algorithm to find the shortest path from the client to all switches in set P. The time complexity of this step is equal to $O(|V|+|E|)$. The time complexity used to update the switch links in the multicast tree is $O(|V|)$. The above procedure is repeated n times, the time complexity of MPCA-k is $O(n(|V|+|E|))$.

3.2 Maximum Bottleneck Bandwidth Utilization Algorithm with Shortest Path (MBUA-SP)

In this subsection, we propose a Maximum Bandwidth Utilization Algorithms with Shortest Path (MBUA-SP) to maximize the link utilization. We can modify the Dijkstra's one-to-all shortest paths algorithm [16] to find the MBB from source server S to all switches in our network. In the Dijkstra's algorithm, we modify its *relax* operation as follows. For each vertex $v \in V$, $m(v)$ denotes the MBB of the path from source s to v. Initially, we set s_1 connected to source S as a visited switch and set the other switches as unvisited ones. In our example, $m(s_1)$ is set to ∞ and for each non-source vertex $v \in V$, $m(v)$ is set to 0. When a vertex $u \in V$ is visited, we *relax* all of the outgoing edges of vertex u as follows. Assume (u, v) is one of the outgoing edges of u. If $m(v) < \min\{m(u), b(u, v)\}$, the value of $m(v)$ will be updated to $\min\{m(u), b(u, v)\}$ because we can reach v through u with a larger MBB path than without through u. In the next step, visit an unvisited switch x that has the maximum $m(x)$ and mark it as visited. After performing $|V|$ iterations of *relax* operation, we get MBB of all switches.

In Fig. 2, for example, the first visited switch s_1 connected to the source server S, we set $m(s_1) = \infty$. After *relax* the outgoing edges of s_1, we have $m(s_2) = 14$ and $m(s_3) = 13$. Note that, the red number above each switch represents the MBB from source to this switch. Since s_2 has the maximum MBB, we visit and *relax* its outgoing edges. We have $m(s_4) = 10$ and $m(s_5) = 14$. After repeating 15 *relax* operations, the MBB of each switch is as shown in Fig. 2.

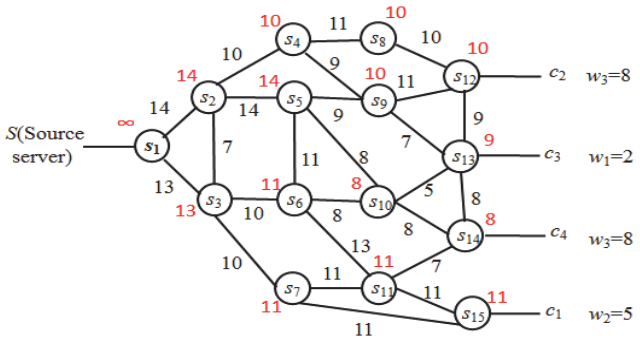

Fig. 2 The MBB path from source S (s_1) to all switches.

Next, we sort the n clients in decreasing order according to the MBB of their connected switches. Without loss of generality, assume that the n clients are sorted in the order c_1, c_2, $c_3 \ldots c_n$. Then each client finds a shortest path from source to the client such that the MBB of the shortest path is no less than the MBB of the switch connected to the client. For example, in Fig. 2, the MBB of a shortest path from source to client c_2 must no less than 10. This can be done as follows. If we want to find a shortest path from source to client c_i, we can remove the switch links whose residual bandwidth is smaller than the MBB of the client c_i. Then use BFS algorithm to find a shortest path from S to client c_i. We can iterate the above steps from $i = 1$ to n. Let t_i be the multicast tree including clients c_1, $c_2 \ldots c_i$. After finding the shortest path of client c_i, merge this path to t_{i-1} to become t_i, for $2 \le i \le n$. After establishing a multicast tree, we update the residual bandwidth in all edges of the network. The final result of Fig. 2 is shown as in Fig. 3. The time complexity of the modify Dijkstra's algorithm is $O(|V|^2)$. The sorting algorithm spends $O(n \log n)$ time. The time complexity of all clients to find their shortest paths to source S is $O(n(|V|+|E|))$. Thus, the total time complexity of MBUA-SP scheme is $O(|V|^2 + n(|V|+|E|))$.

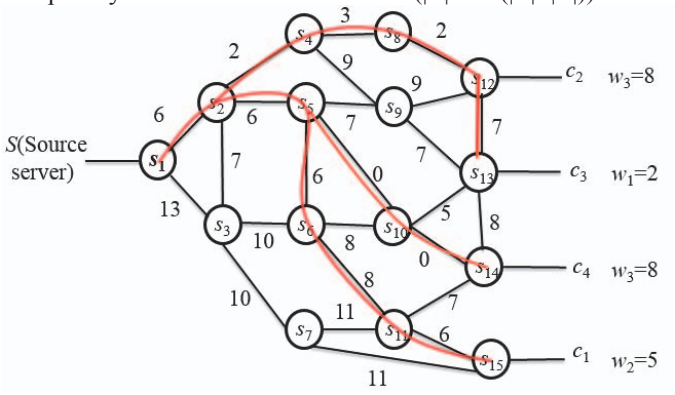

Fig. 3 A multicast tree of using MBUA-SP in Fig. 3.

IV. Performance Evaluation

In this section, we evaluate the performance of our proposed algorithm through simulations. In our simulation, we create a network with 200 switches. Each multicast request has a number of clients which request a chosen source server for the same video files. The source server is randomly selected from 10 different servers in the network. The number of

clients are 20 to 90. Our video files have four kinds of bandwidth requirement: 40 MB/s, 100 MB/s, 160 MB/s, and 200 MB/s. All links have the same initial bandwidth 1 GB/s. The multicast tree is established by the SDN controller.

We measure the performance of the proposed schemes according to the following criteria. (1) Average hop counts of a multicast request: The lower average hop counts imply the lower packet transmission delay. (2) Average successful rate of a request: The average successful rate is the ratio of number of successful requests/total requests. (3) Average link utilization: After all of the multicast requests are accepted and routed, we compute the average utilization of the network switches. The lower link utilization means the lower multicast successful rate. We compare the performance of our proposed algorithms MPCA-*k* and MBUA-SP with the previous works SPT and GIT [15].

4.1 Simulation Results

For the MPCA-*k* scheme, the average hop counts would increase as the *k* increase. Thus, the larger *k* value has the higher average hop counts. However, the larger *k* value has the higher link utilization. Fig. 4 shows the average hop counts from source server *S* to each client for 200 multicast requests in our simulations with different average switch degree. The degree of a switch represents the number of neighboring switches of the switch. For all algorithms, the average hop counts decreases as the average switch degree increases. We find that the MPCA-0 without extra delay time and SPT have the shortest average hop counts from *S* to each client. The MBUA-SP, MPCA-4, and GIT algorithms have the intermediate communication delay.

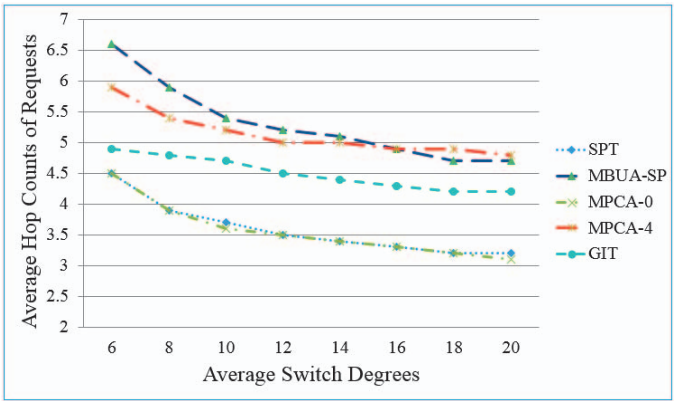

Fig. 4 Average hop counts of multicast requests vs average switch degrees.

Fig. 5 shows the average multicast successful rate of 200 multicast requests with various switch degrees. For all algorithms, the multicast successful rate increases along with the average switch degree. MPCA-4 has the highest multicast successful rate because it allow us to forward packet without through the shortest path. However, MPCA-4 has higher communication delay than other schemes except the MBUA-SP as shown in Fig. 4. The MBUA-SP always choose the shortest path from each client to *S* with MBB restriction. Thus, the multicast successful rate of MBUA-SP is higher than MPCA-0, GIT, and SPT. The GIT successively adds next nearest source to the existing tree whose successful rate is better than MPCA-0 and SPT. Although the MPCA-0 only choose the shortest path from each client to source *S*, it also select the one with MBB if there is more than one shortest path. So, the request successful rate of MPCA-0 is better than SPT. Although the SPT has the shortest communication delay, it has the least multicast successful rate.

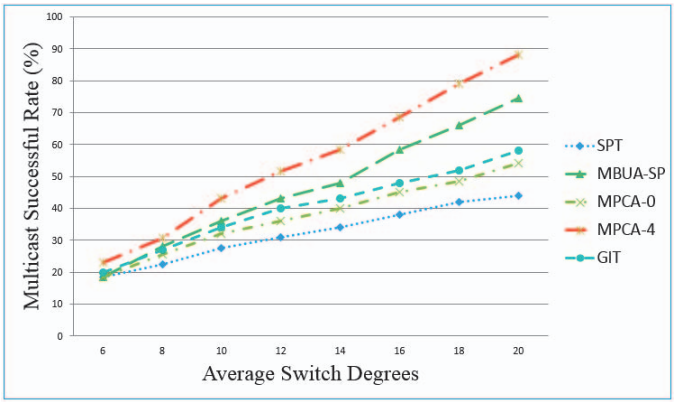

Fig. 5 Average multicast successful rate vs average switch degrees.

Fig. 6 shows the accumulated rejection rate with the number of requests varied from 40 to 200. The average switch degree of the simulation is 15. The average rejection rate is the ratio of rejected requests to the total requests. We can see that the multicast rejection rate increases with the number of multicast requests. The accumulated rejection rate of the algorithms from low to high are MPCA-4, MBUA-SP, GIT, MPCA-0, and SPT. This is because MPCA-4 and MBUA-SP schemes consider not only the shortest path but the maximum bottleneck bandwidth (MBB).

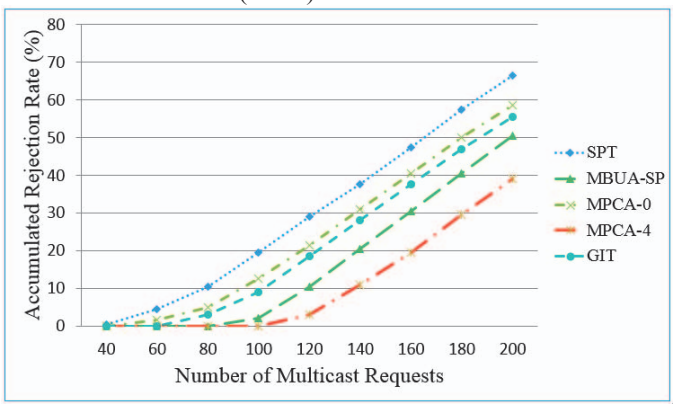

Fig. 6 Accumulated rejection rate with the number of 200 multicast requests

Fig. 7 shows the average link utilization of multicast with various average switch degrees for 200 multicast requests. The link utilization is the ratio of total consumed bandwidth to the total network bandwidth. If a multicast request is rejected, the consumed bandwidth of the request is zero. Since the SPT chooses a shortest path from *S* to each client, its link utilization is the lowest among all algorithms. The link utilization of MPCA-0 is higher than SPT. The link utilization of MBUA-SP and MPCA-4 is larger than 80%. Since the MBUA-SP chooses the shortest path from each client to source *S* under the MBB constraint, it has the best link utilization.

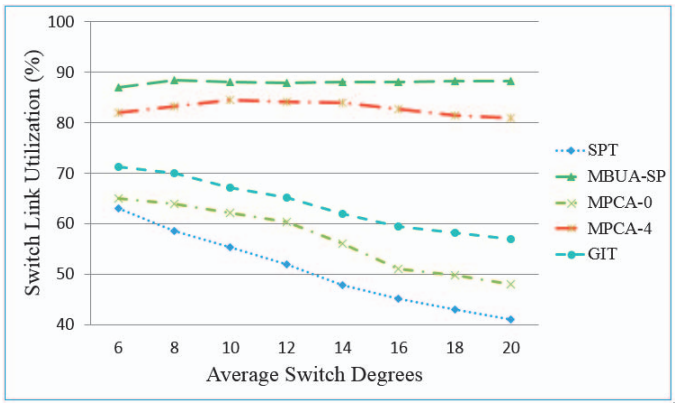

Fig. 7 Average link utilization of multicast requests vs average switch degrees.

V. Conclusion

In this paper, we presented two efficient multicast algorithms in SDN. In MPCA-*k* algorithm, the average hop counts, link utilization, and multicast successful rate increase when the extra delay time *k* increases. Since *k* is a tunable variable, we can adjust our MPCA-*k* algorithm according to the requirements of the network. For MPCA-0, its average hop counts from source to each client is near to SPT, but the performance of link utilization and multicast successful rate is better than SPT. For MPCA-4, it has the highest multicast successful rate. The MBUA-SP algorithm builds the multicast tree under the MBB constraint which has the best link utilization. The communication delay of MBUA-SP is a little higher than SPT and GIT, but the multicast successful rate and bandwidth utilization is much higher than the SPT and GIT algorithms.

References

[1] Y. Kanizo, D. Hay, and I. Keslassy, "Palette: Distributing Tables in Software-Defined Networks," *IEEE International Conference on Computer Communications*, Turin, Italy, April 2013.

[2] H.-E. Egilmez, S.-T. Dane, K.-T. Bagci, and A.-M. Tekalp, "OpenQoS: An OpenFlow Controller Design for Multimedia Delivery with End-to-End Quality of Service over Software-Defined Networks," *Asia-Pacific Signal & Information Processing Association Annual Summit and Conference* (APSIPA ASC), pp. 1-8, December 2012.

[3] T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, and J. Rexford, "Openflow: Enabling Innovation in Campus Networks," *ACM SIGCOMM Computer Communication Review archive,* Vol. 38 No. 2, pp. 69-74, April 2008.

[4] T. Stutz and A. Uhl, "A Survey of H.264 AVC/SVC Encryption," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 22 No. 3, pp. 325-339, March 2012.

[5] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the Scalable Video Coding Extension of the H.264/AVC Standard," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 17, No. 9, pp. 1103-1120, September 2007.

[6] M. Xing, S. Xiang, and L. Cai, "A Real-Time Adaptive Algorithm for Video Streaming over Multiple Wireless Access Network," *IEEE Journal on Selected Areas in Communications*, Vol. 32, No. 4, pp. 795-805, April 2014.

[7] J.-P. Sheu, C.-C. Kao, S.-R. Yang, and L.-F. Chang, "A Resource Allocation Scheme for Scalable Video Multicast in WiMAX Relay Networks," *IEEE Transactions on Mobile Computing*, Vol. 12, No. 1, pp. 90-104, January 2013.

[8] Z. Yan, J.-H. Lee, S. Shen, and C. Qiao, "Novel Branching-Router-Based Multicast Routing Protocol with Mobility Support," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 24, No. 10, pp. 2060-2068, October 2013.

[9] J.-B. Hwang and C.-Y. Lee, "Effective Video Multicast Using SVC with Heterogeneous User Demands over TDMA-Based Wireless Mesh Networks," *IEEE Transactions on Mobile Computing*, Vol.12, No.5, pp. 984-994, May 2013.

[10] X.-H. Tian, Y. Cheng, and B. Liu, "Design of a Scalable Multicast Scheme with an Application-Network Cross-Layer Approach," *IEEE Transactions on Multimedia*, Vol.11, No. 6, pp. 1160-1169, October 2009.

[11] T. Ballardie, P. Francis, and J. Crowcroft, "Core Based Trees (CBT) an Architecture for Scalable Inter-Domain Multicast Routing," *ACM SIGCOMM* 1993, September 13-17, 1993, San Francisco, CA, USA. 1993

[12] S. Ramanathan, "Multicast Tree Generation in Networks with Asymmetric Link," *IEEE/ACM Transactions on Networking*, Vol. 4, No. 4, pp. 558-568, August 1996.

[13] F. Sato, "An Efficient Shared-tree-Based Multi-Source Multicast Routing Protocol in Mobile Ad Hoc Networks," *International Conference on Advanced Information Networking and Applications Workshops*, pp. 377-382, May 2009.

[14] A. Iyer, P. Kumar, and V. Mann, "Avalanche: Data Center Multicast Using Software Defined Networking," *Sixth International Conference on Communication Systems and Networks* (COMSNETS), pp. 1-8, January 2014.

[15] B. Krishnamachari, D. Estrin, and S. B. Wicker, "The Impact of Data Aggregation in Wireless Sensor Networks," *International Conference on Distributed Computing Systems Workshops* (ICDCSW), pp. 575-578, Vienna, Austria, July 02-05 2002.

[16] H.-T. Cormen, E.-C. Leiserson, L.-R. Rivest, and C. Stein, "Introduction to Algorithms," Third Edition, The MIT Press, 2009.