

Using Modified Floyd-Warshall Algorithm to Implement SPB IEEE 802.1aq Protocol Simulator on NS-3

Samuel A. Ajila*, Yoonsoon Chang

Department of Systems and Computer Engineering Carleton University, K1S 5B6, Ottawa, ON Canada.

* Corresponding author. Tel.: 1-613-5202600 ext. 2673; e-mail: ajila@sce.carleton.ca

Manuscript submitted April 22, 2016; accepted August 4, 2016.

doi: 10.17706/ijcce.2017.6.1.1-18

Abstract: Ethernet has evolved to support various network topologies while maintaining its backward compatibility and simplicity. Virtualization of the provider's Ethernet network enables support for fine-grained services for different users. Spanning Tree Protocol (STP) meets these properties but, still could benefit from improvements on utilization and convergence time. Shortest Path Bridging (SPB, IEEE 802.1aq) has been developed to overcome the shortcomings of STP. This paper presents the design and implementation of an SPB simulator for NS-3. The modified version of Floyd-Warshall algorithm is used to compute routes. Multicast and unicast communications are simulated in SPBM (SPB Mac-in-Mac) mode to show the simulator's capability. The results prove that the communication maintains the crucial property of SPB; congruency between multicast and unicast, and symmetry between forward and backward paths. The traffic route selected among candidate paths with the same cost is in accordance with the SPB standard. The contribution of this work is a powerful simulator that can be used to conduct experiments without the usual cost attached to the physical implementation.

Key words: Ethernet, cloud, bridge, switch, virtualization.

1. Introduction

Since Ethernet was released to the market of local area network (LANs), it has been successfully and widely deployed. The main strengths of Ethernet are simplicity and backward compatibility. The plug-and-play feature of Ethernet simplifies network configuration. This feature is desirable especially in cases where user locations are dynamically decided. Backward compatibility makes Ethernet switches more attractive compared to other competing LAN technologies [1].

In cloud data centers it is desirable to minimize network configuration while supporting large numbers of switches. Virtual machine instances in a data center should be located dynamically in terms of time and space. A cloud data center consists of large numbers of switches and servers. Cost efficiency of switches plays major role in constructing a cloud data center and Ethernet fulfills this requirement sufficiently [2].

Since networks have to evolved to demand more functionality, Ethernet has evolved to support virtualization. First, Virtual LAN (VLAN) emerged to support Ethernet virtualization. Its frame contains VLAN tags [3]. Second, Ethernet supports service identification for individual users. The traffic for each user is assigned the unique number in the provider network. This identifier called I-SID [4] enables a provider to offer a fine-grained service to individual users. Finally, Ethernet has evolved to support more sophisticated routing algorithms. Ethernet at its conception only supported broadcast communication through shared medium. Spanning Tree Protocol (STP) [5] introduced a new control scheme to support redundant paths in

Ethernet domain. However, STP still does not fully utilize all links in a network because of loop formation. Shortest Path Bridging (SPB) [6] addresses the shortcomings of STP through a sophisticated control scheme and routing algorithm.

Though Shortest Path Bridging (SPB) is emerging and there is currently no open source SPB network simulator. The absence of an open source SPB simulator is a major obstacle for many research institutions. First, Ethernet bridges running SPB protocol are still not affordable. Second, constructing real life experimental network for research prototyping requires major efforts relative to simulation. Thus, the contribution of this research work includes a thorough analysis of the state of the art; the design and implementation of SPB simulator on the open source, discrete event network simulator NS-3 which can be used by the research community, and a modified Floyd-Warshall all-pairs shortest paths algorithm in order to compute routes in a simulation. Our modified algorithm is designed to utilize a meshed network efficiently, which is particularly relevant to data center networks.

The rest of this paper is organized as follows. Section 2 gives the background and related protocols. Section 3 describes the design of the architecture of our simulator. Section 4 describes the SBP control and data planes. Section 5 gives the conclusion.

2. Background and Related Work

2.1. Legacy Ethernet Protocols

Shortest Path Bridging (SPB) addresses the problems of legacy Ethernet network. There are problems in both control plane and data plane. In control plane, minimizing convergence time after topology changes is a major challenge. In data plane, inefficiency of link utilization is the main problem.

Switches in an Ethernet network share the same physical medium. The routing system of an Ethernet network was still based on the flood-and-learning mechanism. Massive amounts of frames flooding in the network may cause broadcast storms, effectively melting down the network. To prevent broadcast storms while permitting such a flood-and-learning mechanism, the Spanning Tree Protocol (STP) was designed [5]. Redundant paths in an STP network are disabled to suppress loop formation. A spanning tree topology is activated in the network and the topology appears as replica of the shared medium. The STP locates each end point on a spanning tree hence is a single point of failure. This simple connectivity is not on the optimal path between two end points. To guarantee the loop free status at all times, any topology change shuts down all connectivity on the spanning tree until the new tree has converged [7]. Though this shutdown period is no longer than a few seconds, it is still not acceptable.

The main problem of STP's data plane is inefficient use of links in a network. First, to avoid loop formation in STP region, some of the links in a network must be disconnected. Another problem is that traffic within an STP region does not follow the optimal path frequently. Since an STP region is a replica of shared medium, it does not provide a sophisticated routing mechanism. Fig. 1 demonstrates the problems of STP data plane. First, the link between switch B and D is disabled to prevent loop formation. If the link is enabled, there is the loop $A \Rightarrow B \Rightarrow D \Rightarrow A$. Second, traffic between switch C and E, does not follow the optimal path $C \Rightarrow B \Rightarrow D \Rightarrow E$.

Shortest Path Bridging (SPB) resolves this problem of STP's data plane. In an SPB region, all links can be used and none need be blocked for loop prevention [1]. All switches in a SPB region share a global topology view. Based on the global topology view, each switch configures itself to carry traffic on the optimal path.

As seen in Fig. 1, Spanning Tree Protocol (STP) does not construct optimal topologies and its convergence time is unacceptable. Shortest Path Bridging (SPB) emerges to overcome this shortcoming [8]. SPB, as its name suggests, offers optimal paths between each end point. The switches in the same SPB region share the same topology information. In contrast to STP, SPB does not require topology information exchanges

through the flood-and-learn mechanism. Switches in the Shortest Path Bridging (SPB) domain build Link State Databases (LSDB) in which global topology information is stored. A LSDB is periodically updated through Intermediate Systems to Intermediate Systems (IS-IS) protocol [9]. A topology change also triggers LSDB updating. Since a SPB enabled switch responds to the event of topology changes without a flood-and-learn process, its convergence time is faster than Spanning Tree Protocol (STP). All switches in a SPB region share the same topological information and unlike STP, all links in a SPB network can be utilized. Thus, it is possible to compute the optimal path in the given topology [7], [10].

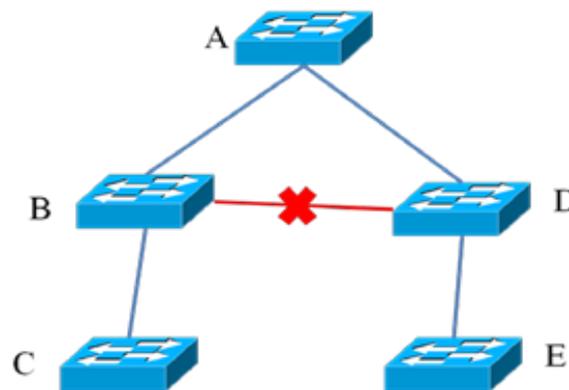


Fig. 1. STP problems.

2.2. Virtualization and Fine-Grained Service of Ethernet

Virtualization of an Ethernet network is dividing the network as multiple logical domains without regard to physical topology of the network. This approach has numerous benefits. First, it reduces the size of a broadcast domain. Second, it enhances the security of the network by blocking unknown traffic. Third, it makes it possible to direct traffics through pre-assigned routes. Virtualized Ethernet LANs are called Virtual LANs (VLANs) [3]. VLAN inherits all these benefits from virtualization.

In cloud network, where virtual machine instances are relocated dynamically, minimizing the overhead from managing virtual machines locations is necessary. VLAN can add, remove and relocate virtual machines using software configuration. Therefore, VLAN techniques can be used to minimize overhead. Shortest Path Bridging (SPB) adopts the VLAN concept so it has the same ability as VLAN. Each frame in SPB networks has the slot for VLAN identifier. This slot is called Backbone VLAN ID (B-VID).

SPB identifies a service instance. A service instance is a group of Backbone Edge Bridges (BEBs) that support a given customer's VLANs [11]. Every service instance is assigned I-component Service Instance Identifier (I-SID). I- SID came from Provider Backbone Bridge (PBB) [4], IEEE 802.1ah, frame header as well as B-VID. Physical or logical User Network Interface (UNI) ports are assigned I-SID. BEBs are located at the border between a SPB network and a customer network as shown in Fig. 2. For example, a customer frame with a VLAN identifier reaches to a UNI port at a BEB. The frame passes through the UNI port and arrives at I-component which encapsulates the frame with an I-SID matched with the customer VLAN identifier as well as Backbone Source Address (B-SA) and Backbone Destination Address (B-DA). This encapsulated frame travels across a SPB network and reaches another BEB with the B-DA. Finally, it is de-capsulated at an I-component linked to a UNI port.

NS-3 [12] is a discrete-event network simulator. It is an open source project and successor of the previously most popular open source network simulator NS-2 [13]. NS-3 consists of core modules and additional libraries of simulation models. Core modules are event scheduler, tracing system, attribute system, core network models (such as node, device, channel and application). Additional libraries come

from NS-3 community. These libraries are used to support new protocols or hardware models.

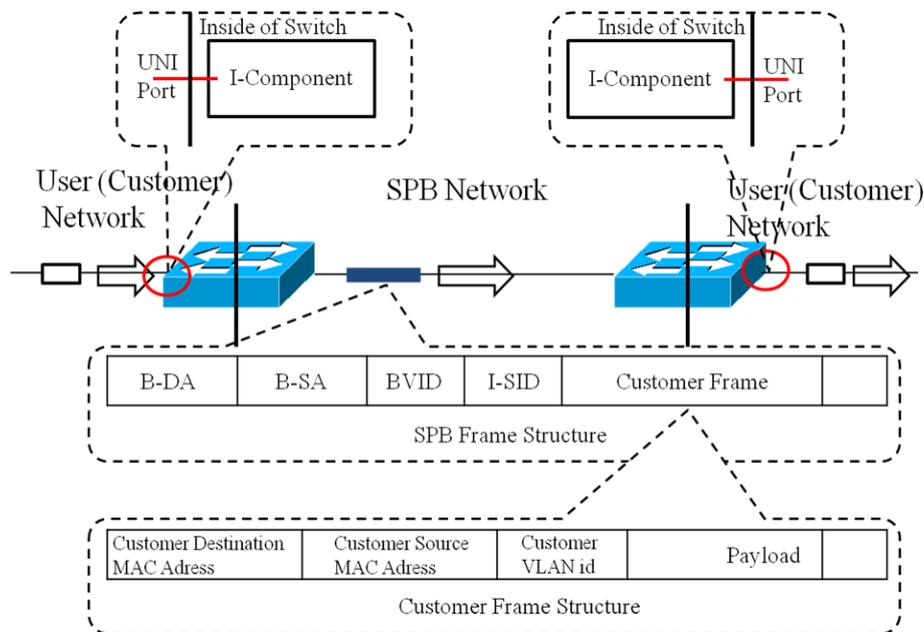


Fig. 2. En-/De-capsulation UNI port.

With NS-3, researchers can develop experimental protocols and simulate them on arbitrary simulation topologies. Simulation results can be collected as text files. NS-3 supports a file format used in many networking tools like TcpDump and Wireshark. TcpDump and WireShark - packet analyzers allow the user to intercept and display TCP/IP and other packets being transmitted or received over a network to which the computer is attached. Since NS-3 source code is open, researchers can develop a tracing system generating specific file format used to draw diagram, graph and animations. Its design concept and goals are referenced in [14], [15].

NS-3 also supports python and C++ binding but it can run script written in C++. In contrast to when NS-2 was released, current computing hardware is much faster than the past. So NS-3 does not need to follow NS-2 architecture any more. In NS-3, a script is main function of executable file and core parts are linkable libraries to the executable file. NS-3 supports both static and dynamic linking.

2.3. NS-3 Topology

A In the real world, a computer network consists of hosts and links. NS-3 offers software simulation models of hosts and links. The model of a host in NS-3 is called node and its corresponding C++ class is "Node". A link is represented as the "Channel" class in NS-3 simulation. Since a host in the real world consists of several components, and likewise a node in NS-3 has several components. A host, which can be a server or a router, has software parts and hardware parts. A network interface card (NIC) is an interface between a host and a link. A frame or packet comes in or goes out of NIC is managed by software protocol handlers. These protocol handlers also manage data from upper layers and user applications.

In NS-3, a node is an abstraction of a host in the real world, and it consists of protocol handlers and application code. The NIC model of NS-3 is "NetDevice" class. NS-3 offers IPv4 and IPv6 protocol stacks. These stacks interact with TCP/UDP socket layers (Fig. 3). A user application model of NS-3 is "Application" class.

Fig. 4 demonstrates how NS-3 models are combined. The nodes in Fig. 4 have OnOffApplication and Sink classes as an application model. Their protocol stacks consist of UDP/IP and ARP classes with respect to

transport and network layers models. CsmaNetDevice and CsmaChannel classes are selected as their link and physical layer models respectively.

OSI Layers	Corresponding NS-3 models
Applications	UDP echo, OnOffApplication, sink
Transport Layer	UDP, TCP
Network Layer	IPv4/IPv6 Global(OSPF), Static Routing..
Link Layer	PointToPoint, CSMA, 802.11
Physical Layer	Wired channel (CSMA, Point to Point) , Wireless channels (802.11a, LTE..)

Fig. 3. NS-3 models.

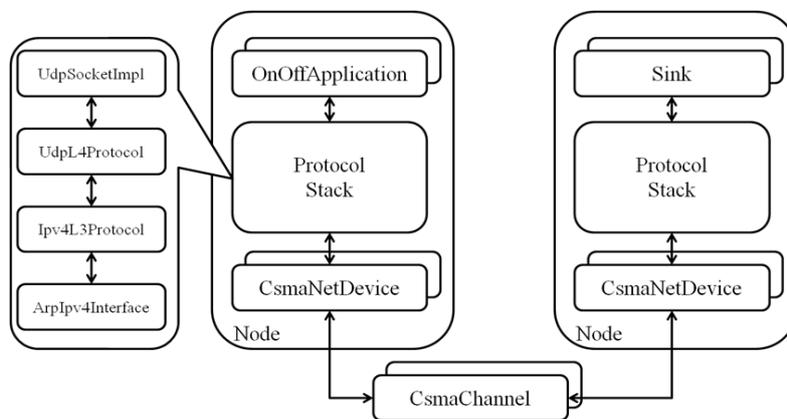


Fig. 4. Nodes with internet stack and CSMA channel in NS-3.

3. Architecture of Our Simulator

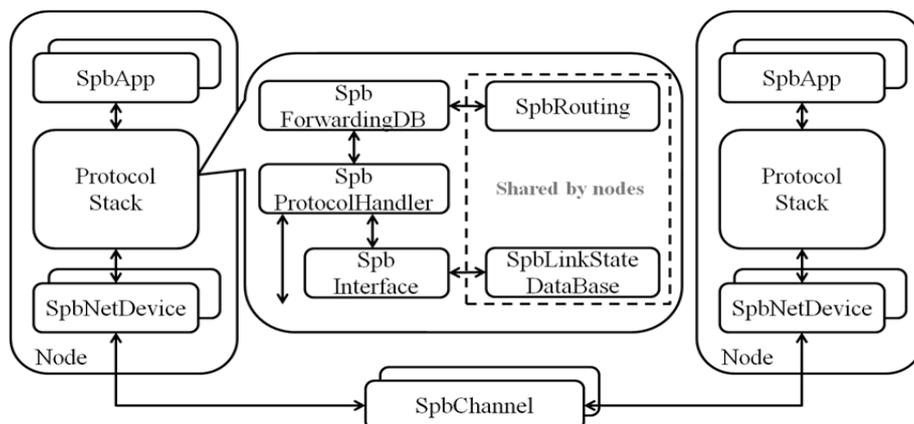


Fig. 5. Nodes with SPB stack and SPB channel.

Fig. 5 presents our architecture of SPB nodes. An SPB node models the bridge capable of running the SPB protocol. Channel is the model of a link. SpbRouting and SpbLinkStateDataBase are singleton objects shared by all SPB nodes. In the real world, an SPB switch may have its own SPB link state database. A link state database in a SPB node is synchronized with other databases installed in other nodes unless there is a miss-synchronization. Therefore, in most cases link state databases in the same SPB domain are identical. Current simulation design does not consider miss-synchronization because the objective of our design is to

offer a base SPB model for future extension. In contrast to a normal Internet node, SPB node's protocol handler communicates with SpbNetDevice directly since the SPB is a layer 2 protocol. The following is the description of each module.

- **SpbLinkStateDataBase:** This is the hash table where all references to each SpbInterface are saved. Its key is a nodal B-MAC address which identifies a node. Therefore, a query with a nodal B-MAC returns the SpbInterface on the node. This is a singleton object which has only one instance in the simulator. It is a global database shared by every node in the simulator. In a real world situation, a SPB enabled switch has its own Link State Database (LSDB) and the LSDB is synchronized with other LSDBs installed in other switches.
- **SpbInterface:** This is the interface to the control plane module for a node. It offers APIs for accessing **Intermediate System to Intermediate System (IS-IS) sub-TLVs** for SPB. Through this module, we can set or get sub-TLV's values and manage the sub-TLVs. For example, we can assign an I-SID on a node in order to register the node in the specific group represented by the I-SID. In a real world situation each SPB switch (represented by a node in simulation) constructs IS-IS sub-TLVs and exchanges the "digest" with neighboring switches. After exchanging the digest, a node in a SPB network can build LSDB. A digest is a summary of information like node identifiers and link costs.
- **SpbApp:** The model of a packet source and sink. It generates a packet and forwards it to **SpbProtocolHandler** to send a packet to other nodes. It also consumes the packets whose destination is matched by the node on which it is installed.
- **SpbProtocolHandler:** This module extracts the SPB headers of received frames from SpbNetDevice. It queries **the forwarding (filtering) database (FDB)** of the node to select the proper output ports.
- **SpbNetDevice:** The software/hardware model of the network driver of SPB and the input/output **interface** of a frame to/from outside of the node. This is the same as output port of a bridge. It is inherited from NS-3 base class Net-Device. It encapsulates the packet received from layer above the current layer. When it receives a frame from the SpbChannel, it forwards the frame to SpbProtocolHandler.
- **SpbChannel:** The model of a SPB link. It has two references of SpbNetDevice as endpoints of the link. The weight of the link is **user** configured.
- **SpbForwardingDB:** The model of a forwarding database of the SPB enabled by the bridge. When SpbProtocol-Handler and SpbRouting objects query SpbForwardingDB, SpbForwardingDB returns the reference of the table entry or the output port referencing the SpbNetDevice object.
- **SpbRouting:** This module is in charge of computing the path and populating the forwarding databases on nodes according to the result of the computation.

3.1. SPB Node

The following is the brief description of how a node handles the received frame:

1. When a node receives a frame from outside, the frame arrives at SpdNetDevice. A frame is a pointer value referencing the NS3::Packet object.
2. SpbNetDevice calls the callback function registered at the node.
3. The node looks for the correct handler for the packet type.
4. If it is a SPB packet, the node calls SpbProtocolHandler.
5. SpbProtocolHandler extracts SPB header from the received packet.
6. SpbProtocolHandler queries SpbForwardingDB to know the output port number. Every SpbNetDevice has corresponding output port number. If the output port number is '0', SpbProtocolHandler moves the packet to SpbApp. Otherwise, the packet is moved to corresponding SpbNetDevice

When a node send a frame:

1. SpbApp generates a packet and forwards it to SpbProtocolHandler with I-SID and traffic type. There are two traffic types, unicast and multicast.
2. SpbProtocolHandler queries SpbForwardingDB and SpbInterface. First, SpbProtocolHandler queries SpbInterface to get BVID corresponding to I-SID received from SpbApp. Second, it queries SpbForwardingDB with BVID and Nodal MAC address of the node.
3. SpbProtocolHandler checks the outport referencing one of the SpbNetDevice installed in the node. SpbProtocolHandler adds SPB header to the packet and moves it to SpbNetDevice.
4. SpbNetDevice corresponding to the output port sends the frame to connected SpbChannel.

4. SPB Control and Data Planes

This section presents the design of our control plane. In general, the term control plane refers to a part of the network architecture that collects the information of a network topology, and performs the routing calculations required to direct traffic. The information collected from the control plane is used to build a Forwarding Database (FDB). Since Shortest Path Bridging (SPB) has evolved from the latest Ethernet technology the SPB control plane not only deals with the traditional LAN topology information, but also the service and virtual LAN identifiers. The important thing to consider is that an SPB switch encapsulates a customer frame into its frame when it receives the customer frame, and thus the SPB control plane is isolated from a customer network.

4.1. IS-IS for Control Plane Protocol

The Intermediate System to Intermediate System (IS-IS) link state protocol is the base protocol used to control Shortest Path Bridging (SPB) network [6]. The basic mechanism of link state protocol is that every node, either routers or switches, have the same view of the network topology. The Hello protocol of IS-IS is used to learn about adjacent nodes and maintain adjacencies between neighboring nodes. A flooding protocol (link state update) is used to exchange information such as link metrics, path identifiers and group memberships. This information about nodes, links, paths and memberships is globally unique in a single Shortest Path Bridging (SPB) network domain. A network topology discovered by IS-IS is represented as a graph consisting of nodes and connecting links.

IS-IS is also an interior gateway protocol which is used within a single administrative network. IS-IS provides the distributed database for each SPB node in the same network. The information about nodes, links, path and memberships are exchanged between nodes in the same SPB network, and this data is used to populate the database at each node. Based on the database, each node in the network computes the forwarding path to other nodes independently and based on the forwarding decision, each SPB nodes produces forwarding tables (FDB) by using its database.

IS-IS can work in OSI network layer to discover a network topology. Another link state and interior gate protocol Open Shortest Path First (OSPF) carries information over IP protocol. On the other hand, IS-IS does not depend on some network address format (such as IP addresses). Since SPB is a layer 2 routing protocol for an interior network, thus IS-IS fulfils the requirements of SPB [17].

4.2. IS-IS Extension for SBP

Shortest Path Bridging (SPB), IEEE 802.1aq, extends the normal Intermediate System to Intermediate System (IS- IS) Protocol Data Unit (PDU) to carry SPB network information. This information includes node identifiers, link metrics, and adjacencies represented by SPB network terms. SPB can run in parallel with other network layer protocols. Therefore, SPB requires Network Layer Protocol Identifier (NLPID) and the

NLPID value 0xC1 is assigned to SPB. A node advertising NLPID value 0xC1 in IS-IS Hello (IIH) protocol can be a member of a SPB network.

IS-IS extension for SPB is augmented with a small number of Type Length Value (TLV) and sub-TLVs. SPB supports two different modes of operation. In SPBM (SPB MAC) mode, a group of service is identified by MAC address. In SPBV (SPB VID) mode, a group of services is identified by I-SIDs (Ethernet Services Instance Identifier). Our scope in this work is limited to SPBM mode.

Control Protocol of SPB and extensions to IS-IS consists of four major parts. The first part is Hello (IIH) protocol extensions. IS-IS Hello (IIH) Protocol is used to detect neighbor nodes that are capable of running SPB protocol. IIH is also used to exchange and digest information. The information holds two bridge identifiers (bridge priority & bridge sysID) and link metrics of neighboring nodes sharing the same link. The associations between Virtual LAN IDs (VID) and equal cost tie-breaking algorithms are also included in a digest. The digest information is used to validate the commonality of Link State Databases (LSDB) in the same SPB domain. Since IIH is a digest of topology information, other IS-IS extensions for SPB carry the full information. Thus the SPB simulator uses IIH only for detecting neighboring nodes.

The Second part is SPB Instance (SPB-Inst) sub-TLV which carries nodal information with the following characteristics:

- **SPSourceID:** 20-bit value of nodal nick name. It forms multicast destination address (DA).
- **Bridge Priority:** This 16-bit value combined with 4 bytes System ID forms the Bridge Identifier.
- **Number of Trees:** It is set to the number of VLAN-ID tuples.
- **VLAN-ID tuples:** It consists of 4 bytes ECT-Algorithm, 12-bit Base VID, 12-bit SPVID and U, M, A flag bits. When an SPB bridge advertised an ECT-Algorithm with Base VID in the same tuple, the ECT-Algorithm is applied on the given Base VID. Base VID is used for SPBM mode and SPVID is used for SPBV mode.

The third part is the adjacency information extensions. Its corresponding sub-TLV is SPB link Metric sub-TLV. This sub-TLV consists of:

- **SPB-Link Metric:** 24-bit unsigned number. Administrative weight of a link. Smaller number indicates lower weight. When two neighbor nodes advertise different SPB-Link Metric value on the same link, the maximum value is considered as the right value of the given link.
- **Port Identifier:** 16-bit. The standard IEEE port identifier.

The fourth part is service information extensions and it is carried in SPBM Service Identifier and Unicast Address (SPBM- SI) sub-TLV. It consists of:

- **B-MAC address:** A unicast address of the node. Each SPB node has the unique nodal Backbone MAC address. This may address a port. When multiple B-MACs are used, this TLV must be repeated per B-MAC. The SPB simulator so far supports only one nodal B-MAC.
- **Base VID:** This Base VID also appeared as VLAN-ID tuples in SPB-Inst sub-TLV. Thus, B-MAC address, Base VID, and ECT-Algorithm are all associated.
- **I-SIDs:** 24-bit group identifier. This I-SID set is assigned the Base VID in the same SPBM-SI sub-TLV. If two different nodes advertise the same I-SID, intermediate nodes between the two nodes will configure themselves to carry traffic generated from the two different nodes. The intermediate nodes will create Filtering Database (FDB) for unicast and multicast addresses.

4.3. Control Plane Software Architecture

Fig. 6 shows the class diagram of sub-TLVs and link state database (LSDB). SpbInterface inherits "ns3::Object." An "ns3::Object" instance can be aggregated to a node instance. SpbLinkStateDataBase is a hash table with keys corresponding to node MAC addresses and values containing SpbInterface pointers referencing the node with the MAC address. Since a node's MAC address is a unique value identifying a node,

we can use it as the key value of SpbLinkStateDataBase. Here is the brief description of each class:

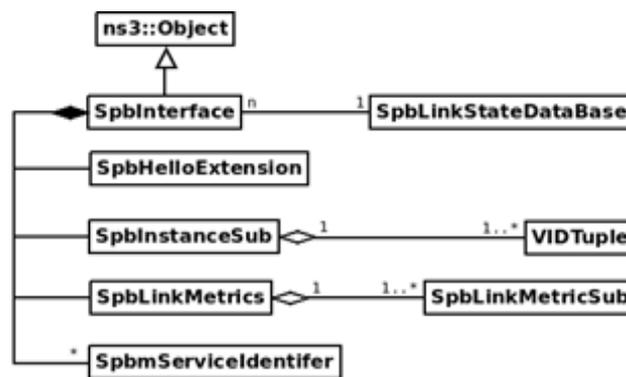


Fig. 6. Control plane architecture.

- **SpbHelloExtension:** Corresponding to Intermediate System to Intermediate System (IS-IS) Hello protocol extensions (IIH). IIH consists of three different sub- TLVs. However, in our simulations, we simplify IIH since we assume that link state database is always synchronized correctly. This class is used for detecting adjacent nodes.
- **SpbInstanceSub:** corresponds to SPB Instance sub- TLV. It can include multiple VIDTuple classes.
- **SpbLinkMetrics:** corresponds to SPB Link Metric sub- TLV. It can include multiple SpbLinkMetricSub.
- **SpbmServiceIdentifier:** corresponds to SPBM Service Identifier and Unicast Address sub-TLV
- **VIDTuple:** It consists of flag bits and a set of ECT- Algorithm, Base VID and SPVID. SPVID is used only for SPBV (SPB Q-in-Q) mode. A SPB node may be assigned multiple B-VIDs so the SpbInstanceSub class can include multiple VIDTuple classes.
- **SpbLinkMetricSub:** Since a SPB node is attached at least one link, the SpbLinkMetrics class needs multiple SpbLinkMetricSub in order to represent each link.

4.4. Routing Algorithm Based on LSDB

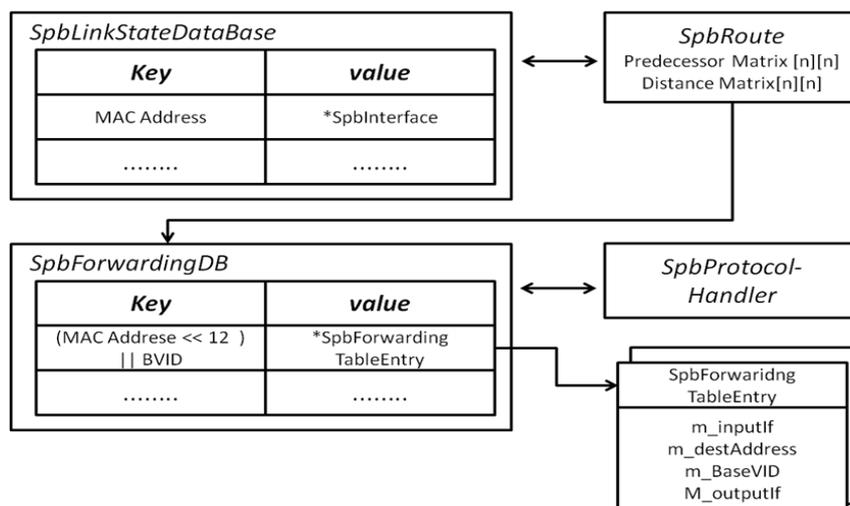


Fig. 7. SPB routing architecture.

Fig. 7 demonstrates the routing architecture of the SPB simulator. Below is the brief description of each class.

- **SpbRoute:** computes the path and populates filtering database of all nodes.

- **SpbForwardingDB**: corresponds to a filtering database of a SPB bridge. It has the hash table of which key value is a concatenation of MAC address and BVID. Once a frame arrives at a node, SpbProtocolHandler extracts destination MAC address and BaseVID from the frame. Next, SpbProtocolHandler combines them to make the key. After that, SpbProtocolHandler queries SpbForwardingDB with the key and get the reference of SpbForwardingTableEntry.
- **SpbForwardingTableEntry**: corresponds to each entry of a filtering database. An entry of a filtering database consists of input port, destination MAC address, Base VID, and output ports.
- **SpbProtocolHandler**: queries the SpbForwardingDB to get the output port when a frame is received from SpbApp or SpbNetDevice.

In a meshed Ethernet network such as a data center, there may exist multiple equal cost paths. Managing equal cost paths is the key to maximizing network capacity. In our case, we modified the Floyd-Warshall all-pairs shortest paths algorithm (Algorithm 1) [18] to manage equal cost paths. Time complexity of Floyd-Warshall algorithm is $O(N^3)$, where N is the number of nodes. In a meshed network where m and n are number of edges and nodes respectively, the number of edges is larger than the number of nodes. A single-source shortest path algorithm considers the number of edges (e.g. Bellman-Ford and Dijkstra's algorithms). Therefore, the complexity of a single-source algorithm does not scale in a meshed network. For example, if we run Bellman-Ford algorithm to compute routes among all nodes, its complexity equals to $N * O(NM) = O(N^2M)$. All-pairs shortest path algorithm offers better performance in a meshed network.

To compute paths, we use two n by n matrices. The matrix distance $[n][n]$ contains the distances between nodes in the simulation. For example, if distance $[i][j]$ is 10, the weight of the path from i to j is equal to 10. Another matrix predecessor $[n][n]$ contains the predecessor to node j on a shortest path from i to j . In other words, predecessor is the intermediate node. A predecessor matrix value is a 64-bit Bridge Identifier which is the concatenation of the bridge priority and the bridge system id. A bridge system id is numerical form of a MAC address of a node.

Lines number 9 and 10 both create path from i to j . A Path is a list of 64-bit Bridge Identifiers. The difference between lines 9 and 10 is that line 9 creates the path passing through k . Line number 10 joins two paths, one is from i to k and another one is from k to j . If path i to k is on a shortest path and path k to j is, then path, i to j , is also on a shortest path. Computing a shortest path has optimal substructure. Thus, we guarantee that path i to j passing through k is also on a shortest path [19]. Line 11 compares two paths using ECT-Algorithm. There are 16 different ECT-Algorithms. The least significant 1 byte of an ECT-Algorithm is used to XOR on each path.

Shortest Path Bridging (SPB) has the tie-breaking mechanism [20]-[22] to prioritize the equal cost path. Each node advertises the costs of the attached links. These costs are presented in SPB Link Metric sub-TLV. The sum of the link costs on the path is equal to the cost of the path. If equal cost paths exist between two end points, the path with smaller hop counts has the priority. If there are more than two paths with the same link cost and hop counts, the default tie-breaking mechanism picks up the path traversing the intermediate node with the lower Bridge Identifier [17], [23]. Mesh network such as a data center may have multiple paths with the same link cost and the hop counts. This SPB tie-breaking mechanism guarantees diversity [24]-[26].

Algorithm 1 - Modified Floyd-Warshall All-pairs Shortest Path

```

1: Initialize distance[n][n]
2: Initialize predecessor[n][n]
3: for k = 0 to n do
4:   for i = 0 to n do
5:     for j = 0 to n do
6:       if distance[i][j] > distance[i][k]+distance[k][j] then

```

```

7:      distance[i][j] = distance[i][k] + distance[k][j]
8:      else if distance[i][j]=distance[i][k]+distance[k][j] then
9:          Path1 = BuildPath from i to j
10:         Path2 = BuildPath from i to j passing through k
11:         if Path2 has higher priority then
12:             predecessor[i][j] = predecessor[k][j]
13:         end if
14:     else
15:         Do nothing
16:     end if
17: end for
18: end for
19: end for

```

Line 11 compares two equal cost paths. A path with lower hop counts has the higher priority. Since a path is a list of bridge identifiers, the number of elements in the list is equal to the hop counts including the source and the destination.

Thus, the smaller size of the list, the path has the higher priority. If two paths have the same hop counts then, ECT- Algorithm value is XORed with the paths. The path with the smaller result has the higher priority. For example, let's assume that path1 has sequence of bridge identifiers 0, 1, 4 and 5; path2 has sequence of 0, 2, 4 and 5. If path1 and path2 are XORed with 0xFF, path2 produces smaller number. In that case, path2 has higher priority hence it would be selected.

Algorithm 2 - Populating SPB Nodes

```

1. n = total number of nodes in path
2. for i = 0 to n - 1 do
3.     for j = i + 1 to n do
4.         ForwardPath = Build Path from node[i] to [j]
5.         Install Unicast Entry for ForwardPath
6.         Install Multicast Entry for ForwardPath
7.         BackwardPath = Build Path from node[j] to [i]
8.         Install Unicast Entry for BackwardPath
9.         Install Multicast Entry for BackwardPath
10:    end for
11: end for

```

Congruency between unicast and multicast, and symmetry between backward and forward paths makes populating a filtering database simple. We used Algorithm 2 to populate the node on a path. This algorithm takes a path, a SPB service identifier, and I-SID as input.

4.5. Simulation Setup

SPB simulator has been implemented on NS-3.15 running on Ubuntu Linux 12.04 system. Fig. 8 is the experimental network topology. Each node has Id and MAC address. Id is a reference number to identify each node in the simulation. In this experiment, node 0, node 4 and node 5 are grouped with the same ISID 1. T bit and R bit are set to 1 and 0 respectively. Every link in the simulation has the same weight 1. Italic numbers near the links are port numbers. Fig. 9 is the part of the simulation script building the experimental topology of Fig. 8. Fig. 10 shows the part of script for grouping nodes 0, 4, and 5 with the same ISID. Each node in the simulation has nodal B-MAC address of which is node id plus one.

Line 43 (Fig. 9) creates 6 nodes. Line 47 installs the SPB protocol stack on all nodes. From line 50 to 70, all nodes aggregate constant position mobility model. This mobility model fixes the position of a node. SetPosition function takes X and Y coordination of each node. From line 72 to 80, links between nodes are created. Line 117 calls SPB helper API BuildAdjacency(nodes). This call fills up the control plane data of each node. Lines 120 and 121 build Link State Database (LSDB). Line 124, 125 build SpbRouting object and

connect the object to the LSDB. SpbRouting object computes the route with the LSDB. Line 127 to 132 set up ECT-Algorithm index, I-SID and BVID. Nodes 0, 4 and 5 are grouped together from line 134 to 137. Line 139 assigns the ECT-Algorithm index, I-SID and BVID to the group of nodes 0, 4 and 5. Line 142 computes the route for the group. Line 144 prints out Forwarding Database (FDB) in all nodes. From line 146 to 153, SpbApplication in node id 0 sends multicast traffic to its group member sharing the same I-SID from time 1 second to 2 second.

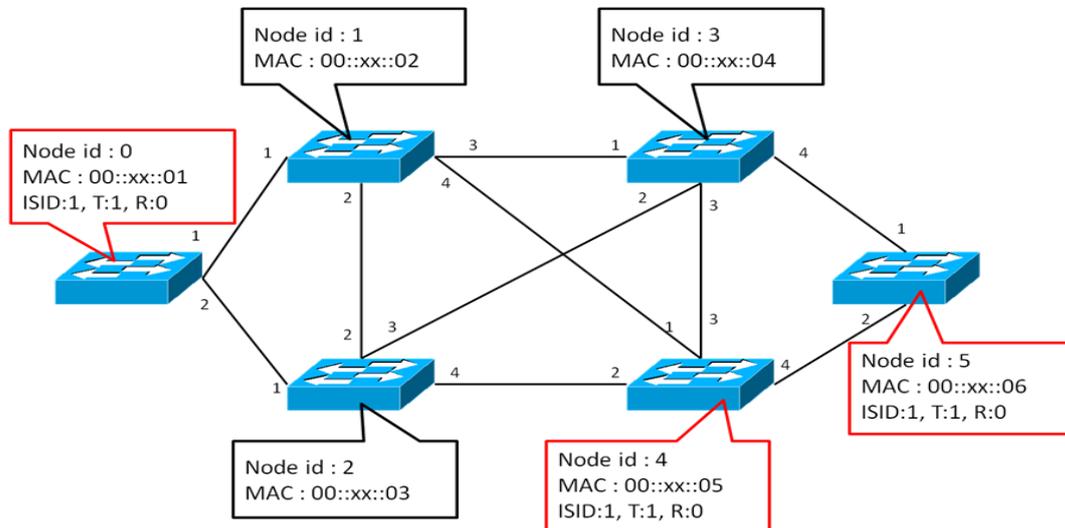


Fig. 8. Experimental topology.

```

41 /* create nodes */
42 NodeContainer nodes;
43 nodes.Create (6);
44
45 SpbHelper spbHelper;
46
47 spbHelper.InstallSpbStack (nodes);
48
49 #if 1 // Congruency Test
50 MobilityHelper mobility;
51 mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
52 mobility.Install (nodes);
53
54 Ptr<MobilityModel> mm = nodes.Get(0)->GetObject<MobilityModel> ();
55 mm->SetPosition( Vector (100, 500, 0));
56
57 mm = nodes.Get (1)->GetObject<MobilityModel> ();
58 mm->SetPosition (Vector (300, 300, 0));
59
60 mm = nodes.Get (2)->GetObject<MobilityModel> ();
61 mm->SetPosition (Vector (300, 700, 0));
62
63 mm = nodes.Get (3)->GetObject<MobilityModel> ();
64 mm->SetPosition (Vector (500, 300, 0));
65
66 mm = nodes.Get (4)->GetObject<MobilityModel> ();
67 mm->SetPosition (Vector (500, 700, 0));
68
69 mm = nodes.Get (5)->GetObject<MobilityModel> ();
70 mm->SetPosition (Vector (700, 500, 0));
71
72 spbHelper.ConnectNodes (nodes.Get (0), nodes.Get (1));
73 spbHelper.ConnectNodes (nodes.Get (0), nodes.Get (2));
74 spbHelper.ConnectNodes (nodes.Get (1), nodes.Get (2));
75 spbHelper.ConnectNodes (nodes.Get (1), nodes.Get (4), 1);
76 spbHelper.ConnectNodes (nodes.Get (1), nodes.Get (3));
77 spbHelper.ConnectNodes (nodes.Get (2), nodes.Get (4));
78 spbHelper.ConnectNodes (nodes.Get (2), nodes.Get (3));
79 spbHelper.ConnectNodes (nodes.Get (4), nodes.Get (3));
80 spbHelper.ConnectNodes (nodes.Get (4), nodes.Get (5));
    
```

Fig. 9. Topology in script.

```

117 spbHelper.BuildAdjacency (nodes);
118
119 // build Link State Data Base
120 SpbLinkStateDataBase* spbLSDB = new SpbLinkStateDataBase ();
121 spbLSDB->BuildDataBase();
122
123 // Connect LSDB and Routing Protocol
124 SpbRouting* spbDataPlane = new SpbRouting ();
125 spbDataPlane->SetLSDB (spbLSDB);
126
127 uint8_t ect = 2;
128 uint32_t isid = 1;
129 uint8_t T = 1;
130 uint8_t R = 0;
131 uint32_t ISID = (T << 31) | (R << 30) | isid;
132 uint16_t bVid = 103;
133
134 NodeContainer group;
135 group.Add (nodes.Get (0));
136 group.Add (nodes.Get (4));
137 group.Add (nodes.Get (5));
138
139 spbHelper.MakeGroup (group, ect, bVid, ISID);
140
141 // Floyd-Warshall with ISID
142 spbDataPlane->CalculateRoute (ISID);
143
144 spbDataPlane->PrintFDB ();
145
146 Ptr<SpbApp> srcApp = DynamicCast<SpbApp> (nodes.Get (0)->GetApplication (0));
147 srcApp->SetStartTime (Seconds (1.0));
148 srcApp->SetStopTime (Seconds (2.0));
149 srcApp->EnableSending (true);
150
151 #if 1
152 srcApp->SetTrafficType (true); // True : multicast, False : Unicast
153 srcApp->SetISID (isid);

```

Fig. 10. Grouping in script.

4.6. Simulation Result

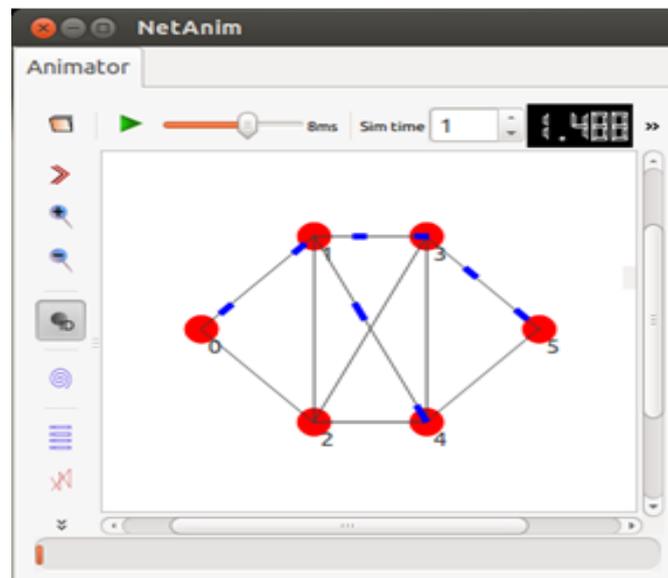


Fig. 11. Multicast when ECT 1.

Our experiment is designed to validate congruency of Shortest Path Bridging (SPB) routing and capacity to select the highest priority path among the equal cost paths. A simulation generates an XML output file. The XML output file is read through the NetAnim program that produces a graphical result. Since NS-3 only

supports graphical result for point to point (P2P) type channel, we use the same interface for P2P. Unicast and Multicast Paths between group members must be identical. From Fig. 8 nodes 0, 4 and 5 are in the same group. This group uses Equal Cost Algorithm (ECT). Each Equal Cost Algorithm has a 1 byte masking value. The masking value is XORed on a path identifier which is the System IDs. When ECT index is 1, then the corresponding mask value is also 0. Priorities between nodes follow the order of node Id: $0 > 1 > 2 > 3 > 4 > 5$. Because we set up the same bridge priorities to all nodes and 4-byte System ID of a node is numerical form of MAC address. Thus, the lower the system ID value, the higher the node priority. Under these constraints, unicast path between nodes 0 to 4 follows nodes 0, 1, 4 in that order. Another unicast path between 0 and 5 is 0, 1, 3, and 5 in order. When multicast is congruent with unicast path, traffic from node 0 must follow node 1 and propagate through ports number 3 and 4. Graphical result is shown in Fig. 11.

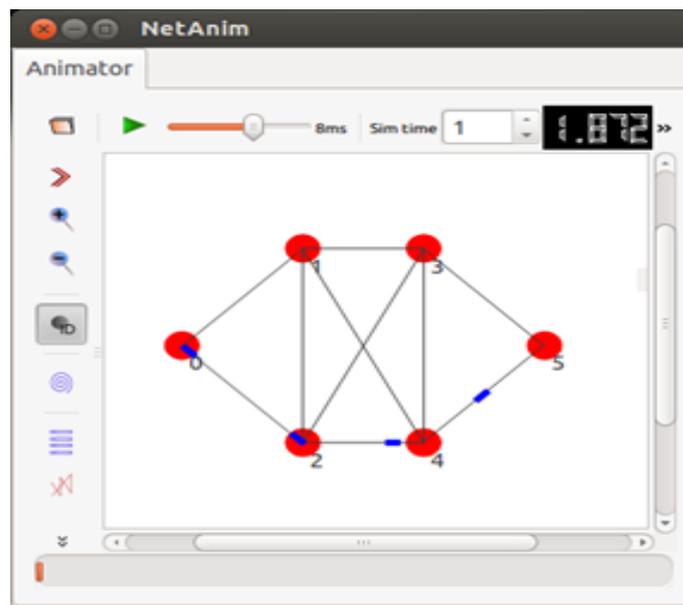


Fig. 12. Multicast when ECT 2.

++++ Distance Matrix +++++						++++ Distance Matrix +++++							
	0	1	2	3	4	5		0	1	2	3	4	5
0:	0	1	1	X	X	X	0:	0	1	1	2	2	3
1:	1	0	1	1	1	X	1:	1	0	1	1	1	2
2:	1	1	0	1	1	X	2:	1	1	0	1	1	2
3:	X	1	1	0	1	1	3:	2	1	1	0	1	1
4:	X	1	1	1	0	1	4:	2	1	1	1	0	1
5:	X	X	X	1	1	0	5:	3	2	2	1	1	0

++++ Predecessor Matrix +++++						++++ Predecessor Matrix +++							
	0	1	2	3	4	5		0	1	2	3	4	5
0:	X	0	0	X	X	X	0:	X	0	0	1	1	3
1:	1	X	1	1	1	X	1:	1	X	1	1	1	3
2:	2	2	X	2	2	X	2:	2	2	X	2	2	3
3:	X	3	3	X	3	3	3:	1	3	3	X	3	3
4:	X	4	4	4	X	4	4:	1	4	4	4	X	4
5:	X	X	X	5	5	X	5:	1	3	3	5	5	X

Fig. 13. Initial and final matrix when ECT is 1.

Before running all-pairs source shortest algorithm, distance matrix and predecessor matrix are shown in Fig. 13 left hand side. Each row indicates starting node while each column indicates ending node. For

example, distance from node 5 to 0 is X because the shortest path algorithm has not yet computed the distance from node 5 to 0. After running the algorithm, the result is shown at right hand side of Fig. 13. Distance from node 5 to 0 is 3 and the predecessor node from node 5 to 0 is node 4. After matrix computation FDBs on all nodes are installed.

Fig. 14 shows filtering database (FDB) of all nodes in the simulation. The capital U and M means unicast and multicast respectively. In and Out columns present port numbers. 'in' column shows input ports and 'out' column shows output ports. Port number 0 indicates the node itself. A Unicast entry of FDB does not consider input ports. Nodes 0, 4 and 5 have three multicast entries. Each multicast entry holds destination B-MAC address of which consists 20-bit SPSourceID and I-SID assigned to them. SPSourceID comes from B-MAC address of nodes 0, 4 and 5.

When ECT index is 2, its corresponding mask value is 0xFF [6]. The masking value 0xFF toggles bits in System IDs therefore, priorities between nodes follow the order Node Id: 5 > 4 > 3 > 2 > 1 > 0. Under these constraints, unicast path between nodes 0 to 4 follows nodes 0, 2, 4 in order. Another unicast path between 0 and 5 is 0, 2, 4, and 5 in order. When multicast is congruent with unicast path, traffic from node 0 must follow node 2 and 4. Result is shown in Fig. 12.

```

Current Node ID : 0
FDB entry size is 6
  in : Destination Mac : BVID : out
U X : 00:00:00:00:00:01 : 103 : 0,
U X : 00:00:00:00:00:05 : 103 : 1,
U X : 00:00:00:00:00:06 : 103 : 1,
M 0, : 03:00:01:00:00:01 : 103 : 1,
M 1, : 03:00:05:00:00:01 : 103 : 0,
M 1, : 03:00:06:00:00:01 : 103 : 0,
Current Node ID : 1
FDB entry size is 6
  in : Destination Mac : BVID : out
U X : 00:00:00:00:00:01 : 103 : 1,
U X : 00:00:00:00:00:05 : 103 : 3,
U X : 00:00:00:00:00:06 : 103 : 4,
M 1, : 03:00:01:00:00:01 : 103 : 3, 4,
M 3, : 03:00:05:00:00:01 : 103 : 1,
M 4, : 03:00:06:00:00:01 : 103 : 1,
Current Node ID : 2
FDB entry size is 0
  in : Destination Mac : BVID : out
Current Node ID : 3
FDB entry size is 4
  in : Destination Mac : BVID : out
U X : 00:00:00:00:00:01 : 103 : 1,
U X : 00:00:00:00:00:06 : 103 : 4,
M 1, : 03:00:01:00:00:01 : 103 : 4,
M 4, : 03:00:06:00:00:01 : 103 : 1,
Current Node ID : 4
FDB entry size is 6
  in : Destination Mac : BVID : out
U X : 00:00:00:00:00:01 : 103 : 1,
U X : 00:00:00:00:00:05 : 103 : 0,
U X : 00:00:00:00:00:06 : 103 : 4,
M 1, : 03:00:01:00:00:01 : 103 : 0,
M 0, : 03:00:05:00:00:01 : 103 : 1,
M 4, : 03:00:06:00:00:01 : 103 : 0,
Current Node ID : 5
FDB entry size is 6
  in : Destination Mac : BVID : out
U X : 00:00:00:00:00:01 : 103 : 2,
U X : 00:00:00:00:00:05 : 103 : 1,
U X : 00:00:00:00:00:06 : 103 : 0,
M 2, : 03:00:01:00:00:01 : 103 : 0,
M 1, : 03:00:05:00:00:01 : 103 : 0,
M 0, : 03:00:06:00:00:01 : 103 : 2,
    
```

Fig. 14. FDBs when ECT is 1.

5. Conclusion

The cloud data center network requires certain properties. The first property is virtualization. The infrastructure of the cloud data center is shared with multiple customers and each customer requires different levels of services. Without network virtualization, it is hard to offer different classes of services. Individual customer traffic has to be identified in the cloud data center for delivering customized services. The second property is acceptable convergence time. The cloud data center network has to quickly respond to network events. A cloud data center runs almost all type of application. Some of the application may need real time stream of data. In order to meet real time constraints, the convergence time after network changes should be minimized. The third property is higher utilization. Network traffic in a cloud data center should take diverse paths to achieve higher utilization. Fourth property is a minimum configuration. Traffic in a

cloud data center changes dynamically. Human intervention to the dynamicity may not be practical. Finally, cost of deploying network should be economical.

Ethernet meets all the requirements described above. It was developed to minimize a network configuration. Its plug-and-play feature can migrate hosts to different locations without configuration. Virtual machine instance in cloud data center may require frequent migrations. The plug-and-play mechanism would reduce the efforts following virtual machine migrations.

Ethernet has evolved to support virtualization and IEEE has developed standard for virtual LAN 802.1q. Though Ethernet offered simple configuration and virtualization, it was still not suitable for the cloud data center. First, its convergence time was unacceptable. Second, its link utilization was not higher enough. Spanning Tree Protocol (STP) has been widely used for Ethernet networks but, it still has the two problems. Shortest Path Bridging (SPB) has been developed to overcome those two shortcomings.

Convergence time after network changes depends on how fast information is delivered to nodes participating in the same network. STP broadcasts information to build up new topology when network changes occur.

Broadcasting takes certain amount of time which is not acceptable for real time applications. While STP uses broadcasting, SPB uses the link state database holding information. Nodes in a network retrieve information from link state databases immediately after network change thus SPB offers acceptable convergence time for real time applications.

Suboptimal link utilization is another problem for applying STP to the cloud data center. In contrast to STP, SPB utilizes links higher in a data center. First, SPB does not require disconnecting one of the links in order to prevent loop formation. Second, traffic in SPB network always takes the optimal path in terms of link cost. While customer networks are isolated from provider networks, network providers apply different services or policies on customer traffic. This situation occurs at the cloud data center also. IEEE 802.1q encapsulates customer traffic and it assigns different service identifiers on the customer traffic. This gives network providers the ability to enable different levels of a service agreement for each customer.

SPB employs all the features that the cloud data center network requires and there is need a simulator for SPB network. Since NS-3 has been widely adapted to research community, we designed and implemented the SPB simulator on NS-3. However, SPB protocol requires brand new administration systems which won't fit into the legacy telecom infrastructure.

This paper presents the design and implementation of the SPB simulator. We proposed a modified Floyd-Warshall all-pairs shortest paths algorithm in order to compute routes in a simulation. This algorithm is designed to utilize a meshed network efficiently, which is particularly relevant to data center networks. The SPB simulator we proposed here is still in the early stage. In order to fully utilize the capability of the simulator, a graphical result must be supported. Simulating User Network Interface (UNI) port and SPBV (SPB VID) mode would extend the capability. We strongly believe that the SPB simulator would help the research community. Future research areas where the SPB simulator can be used are described in [20] and [27].

References

- [1] Allan, N. B. D. (2012). IEEE 802.1q in a Nutshell: Antecedents and technology, exploiting multiple paths in SPB. *802.1q Shortest Path Bridging: Design and Evolution the Architect's Perspective*, New York, IEEE Press, p. 29.
- [2] Allan, N. B. D. (2012). The data center and general enterprise application. *802.1q Shortest Path Bridging Design and Evolution*, IEEE Press, pp. 152-155.
- [3] Draft Standard, 802.1Q/D10. (1997). *IEEE Standards for Local and Metropolitan Area Networks: Virtual*

Bridged Local Area Networks. Copyright by the Institute of Electrical and Electronics Engineers, Inc.

- [4] IEEE Standard 802.1ah. (2008). *IEEE Standard for Local and metropolitan area networks — Virtual Bridged Local Area Networks Amendment 7: Provider Backbone Bridges*. IEEE.
- [5] IEEE Standard, 802.1D. (2004). *IEEE Standard for Local and Metropolitan Area Networks*. Media Access Control (MAC) Bridges, IEEE.
- [6] Draft Standard. (2012). *IEEE 802.1aq/D4.6 Media Access Control (MAC) Bridges and Virtual Bridged Local Area Networks- Amendment XX: Shortest Path Bridging*. IEEE.
- [7] Allan, N. B. D. (2012). *802.1aq Shortest Path Bridging Design and Evolution*. IEEE Press, p. 3.
- [8] Allan, N. B. D. (2012). Why SPB looks as it does, history. *802.1aq Shortest Path Bridging: Design and Evolution The Architect's Perspective*. New York, IEEE Press, pp. 52-60.
- [9] Allan, N. B. D. (2012). Why the SPB control plane looks as it does. *802.1aq Shortest Path Bridging: Design and Evolution: The Architect's Perspective*. New York, IEEE Press, pp. 110-115.
- [10] ISO/IEC. (2002). Information technology-Telecommunications and information exchange between systems — Intermediate System to Intermediate System intra-domain routing information exchange protocol. ISO 8473.
- [11] Ashwood-Smith, P. (2010). *Shortest Path Bridging IEEE 802.1aq Tutorial and Demo*. NANOG 50.
- [12] NS-3. (2013). From <http://www.nsnam.org>
- [13] Nsnam. (November 5, 2011). From http://nsnam.isi.edu/nsnam/index.php/Main_Page
- [14] Henderson, T. R. (2008). Network simulations with the ns-3 simulator. *SIGCOMM Demonstration*.
- [15] Henderson, T. R. (2006). NS-3 project goals. *Proceeding of the 2006 Workshop on NS-2: The IP Network Simulator*. New York, NY: ACM.
- [16] Issariyakul-Teerawat. H. E. (2011). *Introduction to Network Simulator NS2*. Springer.
- [17] Fedyk, D. (2012). *IS-IS Extensions Supporting IEEE 802.1aq Shortest Path Bridging*.
- [18] Cormen, T. H. (2009). *Introduction to Algorithms*, 3rd Edition, Cambridge: MIT Press.
- [19] Cormen, T. H. (2009). 25 all pairs shortest paths. *Introduction to Algorithms*, 3rd Edition, MIT Press, p. 686.
- [20] Allan, N. B. D. (2012). *802.1aq Shortest Path Bridging: Design and Evolution The Architect's Perspective*. New York: IEEE.
- [21] Farkas, Z. A. J. (2009). Performance analysis of shortest path bridging control protocols. *Proceedings of IEEE GLOBECOM*.
- [22] Allan, D. (July 2012). Intelligent load balancing for shortest path bridging. *IEEE Communications Magazine*, 50, 163-167.
- [23] Allan, D. (September 2008). Provider link state bridging. *IEEE Communications Magazine*, 46, 110-117.
- [24] Allan, D. (October 2010). Shortest path bridging: Efficient control of larger ethernet networks. *IEEE Communications Magazine*, 48, 128-135.
- [25] Allan, N. B. D. (2012). Why the SPB control plane looks as it does, Convergence Behaviors. *802.1aq Shortest Path Bridging: Design and Evolution: The Architect's Perspective*, New York, IEEE Press, pp. 82-87.
- [26] Chen K., et al. (July-August 2011). Survey on routing in data centers: Insights and future directions. *IEEE Network*, 25, 6-10.
- [27] IEEE Standard 802.1Q. (2005). *IEEE Standard for Local and Metropolitan area networks-Virtual Bridged Local Area Network*. IEEE.

Samuel A. Ajila is currently an associate professor of engineering at the Department of Systems and Computer Engineering, Carleton University, Ottawa, Ontario, Canada. He received B.Sc. (Hons.) degree in

computer science from University of Ibadan, Ibadan, Nigeria and Ph.D. degree in computer engineering specializing in software engineering and knowledge-based systems from LORIA, Université Henri Poincaré – Nancy I, Nancy, France. His research interests are in the fields of software engineering, cloud computing, and big data analytics and technology management.

Yoonsoon Chan received his M.A.Sc degree in electrical and computer engineering from Carleton University, Ottawa, Canada. Currently he is a network engineer at CENX, Ottawa, Canada.