IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 1, 2021, accepted June 6, 2021, date of publication June 10, 2021, date of current version June 18, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3088288

Simplified Stream Reservation Protocol Over Software-Defined Networks for In-Vehicle Time-Sensitive Networking

SANGJIN NAM, HYOGON KIM, (Member, IEEE), AND SUNG-GI MIN
Department of Computer Science and Engineering, Korea University, Seoul 02841, Republic of Korea

Corresponding author: Sung-Gi Min (sgmin@korea.ac.kr)

This work was supported by the National Research Foundation of Korea (NRF) grant through the Korea Government (MSIT) under Grant 2020R1A2C3011888.

ABSTRACT As the automotive industry adopts Ethernet for in-vehicle network technology, automotive Ethernet has to handle bounded delay traffic. IEEE std 802.1Q introduces the stream reservation protocol (SRP) to reserve resources for time-sensitive traffics. SRP uses bridge-by-bridge propagation for its messages and registers VLAN ID and group address before making Audio-Video (AV) stream reservations. In this study, we propose a simplified SRP scheme over the Software-defined Network (SDN) for in-vehicle bridged networks. The proposed scheme exploits the topology and bridge information of bridged networks. A centralized stream reservation procedure in the scheme replaces the bridge-by-bridge propagation for SRP reservations. The reservation procedure does not require VLAN ID and group address registration. We implemented a prototype of the proposed scheme and showed that reserved flows are well protected when links of bridges are overloaded.

INDEX TERMS In-vehicle communication, software-defined networking (SDN), stream reservation protocol (SRP), time-sensitive networking (TSN).

I. INTRODUCTION

The automotive industry has recently adopted automotive Ethernet [1], [2] for in-vehicle communication. Demands for high-speed communication link and bounded delay traffic have accelerated the adoption of automotive Ethernet in vehicles. To handle bounded delay traffic, Ethernet bridges and end devices, such as Electronic Control Units (ECUs) should support Time-Sensitive Networking (TSN) [3] functionalities. IEEE std 802.1Q [4] introduces the stream reservation protocol (SRP) and the traffic scheduler, such as the Credit-based Shaper (CBS) and Enhanced Transmission Selection (ETS) for TSN.

However, SRP-enabled end devices require numerous steps to make a time-sensitive Audio-Video (AV) stream. SRP first requires end devices to register a VLAN membership and group address for a time-sensitive Audio-Video (AV) stream. Multiple VLAN Registration Protocol (MVRP) and Multiple MAC Registration Protocol (MMRP) are used for VLAN membership and group address registration, respectively. Next, end devices send SRP domain announcements for the SRP domain of the AV stream using Multiple Stream Reservation Protocol (MSRP). After that, end devices may announce their AV stream using MSRP. Other end devices may join announced AV streams. Moreover, MVRP, MMRP and MSRP, with the exception of the SRP domain announcement, use bridge-by-bridge propagation. A bridge in the path of an AV stream may either modify or merge MSRP messages before forwarding them. Fig. 1(a) shows required messages before end devices make or receive AV stream announcements, and an AV stream announcement in the standard SRP.

Software-Defined Networking (SDN) [5]–[7] is a network paradigm to separate the control plane and the data plane of a network. In the control plane, an SDN controller hosts controller applications. Each controller application manages operations of several network nodes to support a specific network functionality. Network nodes in the data plane forward packets using flow rules installed by the controller applications instead of the Filtering Database (FDB). In addition, controller applications running over the same SDN controller may share their information with each other using a common

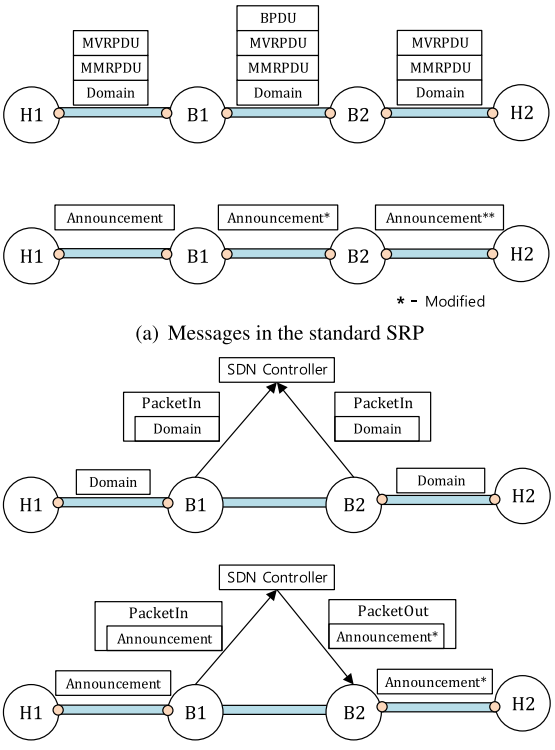
The associate editor coordinating the review of this manuscript and approving it for publication was Rentao Gu.

database, such as the unified data repository in the 3GPP 5G system [8].

Several previous proposals [9], [10] applied SDN concepts to handle SRP reservations. In [9], the new ''ForwardSRP'' OpenFlow control message is introduced and it still uses bridge-by-bridge propagation. [10] explains what the TSN controller module in an SDN controller needs and which protocols can be used to control TSN functionalities in the SDN. However, it is unclear which protocol is used for the stream reservation and there is no mention of how to process the stream reservation at the TSN module.

We propose a simplified SRP scheme over SDN for in-vehicle bridged networks. The proposed scheme runs the MSRP controller application to control stream reservations. The proposed scheme uses the OpenFlow protocol [11] and MSRP messages without any modification and bridge-by-bridge propagation. It obtains the topology and bridge information from the common data repository to build the shortest path trees (SPTs) for AV stream reservations. The SPTs are used to directly forward the AV stream announcements and reservation requests. They are also used to centrally verify the resource availability on the paths of the AV streams and reserve the resources of the AV streams at bridges in the controller application. Unlike the stream reservation that is done by bridge-by-bridge propagation, the proposed scheme does not reserve resources at any bridge if any bridge on the path cannot satisfy the traffic specification of the AV stream. In addition, end devices do not need to register VLAN IDs and group addresses for AV streams. The scheme replaces bridge functions which need VLAN ID and group address registrations by using the controller application. Fig. 1(b) illustrates required messages before SRP-enabled end devices make or receive AV stream announcements and an AV stream announcement propagation in our scheme.

The rest of this paper is organized as follows. Section II discusses some previously proposed SRP over SDN schemes. Section III describes the IEEE 802.1Q SRP operation. Section IV describes the overall design of the proposed architecture. Section V presents the operation of the proposed scheme in detail. Section VI shows the results taken from the prototype we implemented in order to prove that our scheme can operate well. Finally, Section VII concludes this paper.



(a) Messages in the standard SRP

(b) Messages in our scheme

FIGURE 1. Messages before making or receiving AV stream announcements.

II. RELATED WORKS

Several schemes [9], [10] were previously proposed to combine the TSN and the SDN in bridged networks. In [9], the proposed scheme handles SRP over SDN for an in-car network. It uses the new ''ForwardSRP'' OpenFlow control message to forward SRP messages to the SDN controller. When a bridge receives an SRP message, the SRP message is encapsulated in the ''ForwardSRP'' message and the bridge forwards it to the SDN controller. After the SDN controller processes the SRP message, it sends the SRP message back to the bridge. The bridge floods the SRP message to next bridges. Therefore, the proposed scheme needs bridge-by-bridge propagation. In addition, it did not mention about how to process the joining requests and how to make resource reservation for an AV stream at a bridge.

The proposed scheme in [10] handles the dynamic stream reservation over SDN. The proposed scheme describes the conceptual centralized architecture for the TSN. It explains the conceptual structure of the TSN controller module and kind of protocols required to interact with end devices and bridges. In the scheme, messages for stream reservations are exchanged between end devices and the controller using user-only protocol. However, [10] suggests that the user-only protocol may use a REST API concept but doesn't describe a specific protocol. It also does not describe how to map the path between a talker and listeners and how to process the REST API messages at the TSN controller module.

III. IEEE 802.1Q-2018 STREAM RESERVATION PROTOCOL

SRP is defined in IEEE std 802.1Q-2018. SRP is based on Multiple Registration Protocol (MRP) [12], so it is called MSRP in the standard. MSRP supports an uni-directional multicast stream. As a result, there must be a single sender, called a talker, and multiple receivers, called listeners. A talker is an end station that sends an AV traffic and listeners are end stations that consume the AV traffic. The members of the AV stream must belong to the same VLAN and join the same multicast group over the VLAN. MSRP utilizes MVRP and MMRP for this purpose. All SRP-enabled end devices must broadcast SRP domain announcements to notify their presence and SRP domains which they support.

MSRP uses "One Pass With Advertising" (OPWA) concept [13]. A talker announces an AV stream by sending a talker declaration over the VLAN. The talker declaration contains the declaration type, accumulated latency, and traffic specification. The declaration type may change at a bridge and the local latency at the bridge is added to the accumulated latency field in the talker declaration. The talker declaration is propagated to other bridges according to the bridge-by-bridge propagation rule defined at the MSRP module in the bridge. By default, the talker declaration is propagated through ports over the VLAN. If talker pruning is used at the bridge, the talker declaration is propagated through ports over the VLAN at which the destination MAC address in the talker declaration is registered.

If a listener is interested in the AV stream, it joins the AV stream by sending a listener declaration to the nearest bridge. The listener declaration is reversely propagated along the path which the talker declaration passed through. During reverse propagation, each bridge makes the resource reservation at its output port, which is used by the AV stream, and merges the event of listener declaration with events of the other listener declarations. When the talker receives a listener declaration with a *Ready* or *Ready failed*, it commences the transmission of the traffic of the AV stream.

IV. SIMPLIFIED SRP OVER SDN ARCHITECTURE FOR AUTOMOTIVE TSN

We assume that the MSRP controller application can obtain network topology information gathered by the Link Layer Discovery Protocol (LLDP) [14] and the path latency between a pair of edge bridges, which is measured by LLMP [15], from the common data repository. We also assume that SPTs are built whenever a network topology change is detected by LLDP. In this paper, the assumptions are made to focus on the MSRP operation.

The proposed architecture consists of the following network entities:

- Electronic control units (ECUs)
- OpenFlow-enabled bridges
- the SDN controller

The SDN controller hosts the MSRP controller application. Fig. 2 shows the architecture example of the proposed scheme.

A. ECU

An ECU refers to all of potential embedded devices within a vehicle. MSRP-capable ECUs act as talkers and/or listeners in in-vehicle bridged networks. All MSRP messages are carried to the nearest bridge to which ECUs are connected.

B. OpenFlow-ENABLED BRIDGES

An OpenFlow-enabled bridge is capable of handling the OpenFlow protocol. At every edge bridge, which is connecting an ECU directly in bridged networks, all MSRP messages sent from ECUs are forwarded to the SDN controller

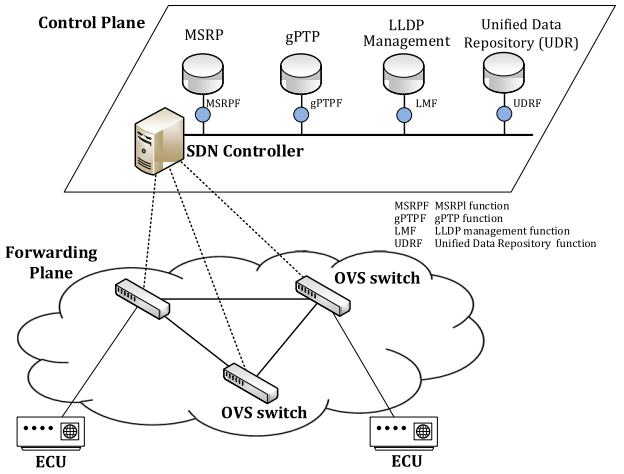


FIGURE 2. Simplified MSRP over SDN Architecture.

using a *PacketIn* message. The receiving bridge and port ID are recorded in the *PacketIn* message. If a bridge receives a *PacketOut* message containing an MSRP message from the controller, the bridge forwards the MSRP message in the *PacketOut* message to the recorded port specified in the *PacketOut* message.

Each bridge classifies incoming packets using flow rules in its flow tables. Each flow rule has criteria, such as VLAN ID and destination MAC address. If there is a matching flow rule for an incoming packet, the bridge applies the action in the flow rule to the packet. By the action, for example, the output port and queue for the incoming packet are selected. In this paper, queues on output ports of bridges used at flow rules employ CBS as a traffic scheduler for SR classes.

C. THE SDN CONTROLLER

The SDN controller is the central control point of bridged networks. It hosts several controller applications for generic Precision Time Protocol (gPTP), LLDP, LLMP and MSRP. It connects OpenFlow-enabled Bridges via the OpenFlow protocol and when the SDN controller receives a *PacketIn* message, it dispatches the message to its controller applications.

D. THE MSRP CONTROLLER APPLICATION

The MSRP controller application processes all MSRP messages between ECUs and edge bridges in in-vehicle bridged networks. The application subscribes *PacketIn* messages, of which the *EtherType* is $0 \times 22EA$ (SRP). After processing, if needed, it sends a *PacketOut* message containing an MSRP message to an edge bridge connecting an ECU.

The controller application constructs an SPT of which the root node is the edge bridge using the topology and bridge information gathered by LLDP. A node in the SPT points to the corresponding bridge and has connection points. Each connection point consists of a bridge ID and a port ID. It presents an output port in the bridge and a branch in the SPT. A connection point connecting an ECU directly is

called an edge connection point. The controller application identifies all connection points to reach an ECU from a root bridge using the SPT of the root bridge. This makes a reservation for an AV stream to be centrally processed in the controller application. In order to make the reservation for the AV stream, the controller application installs the flow rule for the AV stream into a bridge where each connection point belongs to. Then it reserves resources for the AV stream at each connection point.

For the MSRP message propagation, the controller application perceives the data plane as a single abstract bridge. Fig. 3 illustrates a conceptual bridge over the data plane. The controller application propagates MSRP messages directly to ECUs by sending *PacketOut* messages through abstract ports. Abstract ports represent edge connection points in SPTs. The controller application records information of receiving or sent MSRP messages with abstract ports they passed through. By using this information, the controller application checks which ECUs connected to abstract ports are in a same SRP domain, locates the abstract port towards the talker for an AV stream, retrieves information for the AV stream, such as VLAN ID and destination MAC address of the AV stream, and merges events of listeners for the AV stream without involving all connection points with the exception of abstract ports.

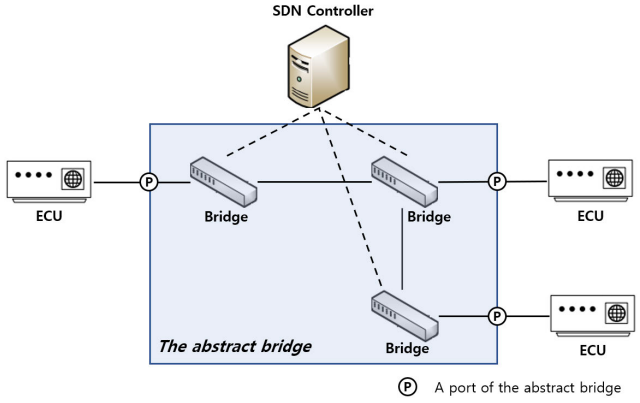


FIGURE 3. The abstract bridge over the data plane.

The controller application does talker pruning over the abstract bridge using SRP domain announcements, instead of using group address over the VLAN. A talker declaration from the talker is propagated to ECUs in the same SRP domain where the talker belongs.

As a result, MVRP and MMRP which use bridge-by-bridge propagation are not needed since a reserved AV traffic is forwarded at OpenFlow-enabled bridges by flow rules installed by the controller application instead of using a Filtering Database (FDB) and the controller application does talker pruning using SRP domain announcements in our scheme. Moreover, there seems to be only one bridge, the abstract bridge; in fact, multiple bridges might exist in the data plane. MSRP messages are exchanged between ECUs and the abstract bridge. Therefore, bridge-by-bridge propagation does not occur in our scheme. In addition, since there is only the abstract bridge regardless of the number of bridges in the data plane, the number of MSRP messages needed is not dependent by the number of bridges in the data plane. Fig. 1 illustrates these properties. In the Fig. 1(a), there are 5 messages between B1 and B2 for the standard SRP while there is no message in the Fig. 1(b). These properties make the data plane more efficient and this advantage is maximized when there is a large number of bridges in the data plane.

Data structures for topology and bridge information are as follows:

- SptTable: It contains all *SPTs* in bridged networks for edge bridges. An *SPT* is retrieved with the bridge ID of its root.
- SPT: An *SPT* is a multicast tree which connects a root bridge to all MSRP-enabled ECUs. The multicast tree consists of a set of *Paths*.
- Path: A *Path* is a shortest route from the root bridge to an ECU in an *SPT*. Each *Path* consists of a path ID, an active path flag, a path latency and a list of connection points. The edge connection point connecting the ECU is used for the path ID.
- BridgeList: Each entry contains a *BridgeInformation* for a bridge. The entry is retrieved with the bridge ID contained in a connection point.
- BridgeInformation: A *BridgeInformation* consists of a Bridge ID, a *StreamTable* and a *PortTable*. It contains all resource information to verify resource availability and to reserve resources for AV streams at a bridge.
- StreamTable: Each entry contains a *StreamInformation* for an AV stream at the bridge. The entry is retrieved with the stream ID of an AV stream.
- StreamInformation: A *StreamInformation* consists of its stream ID, a flow ID, a port map and the talker declaration. A flow ID is the ID of the flow rule installed at the bridge for the AV stream. The port map indicates all ports used by the AV stream and the talker declaration is used to extract the resource reservation parameters for the AV stream at the bridge.
- PortTable: Each entry contains a *PortInformation* for a port on the bridge. The entry is retrieved with the port ID contained in a connection point.
- PortInformation: A *PortInformation* consists of its port transmission rate and the *TrafficClassTable*.
- TrafficClassTable: Each entry contains a *TrafficClassInformation* for a traffic class at the port of the bridge. The entry is retrieved with the priority of the AV stream.
- TrafficClassInformation: *TrafficClassInformation* consists of its class number mapped with priority of an AV stream and all parameters for CBS [18].

Data structures for MSRP operations are as follows:

- SrDomainTable: It records all SRP domain announcements made by MSRP-enabled ECUs. Each entry stores the edge connection point from which its SRP domain announcement is received, the VLAN ID, an SR class

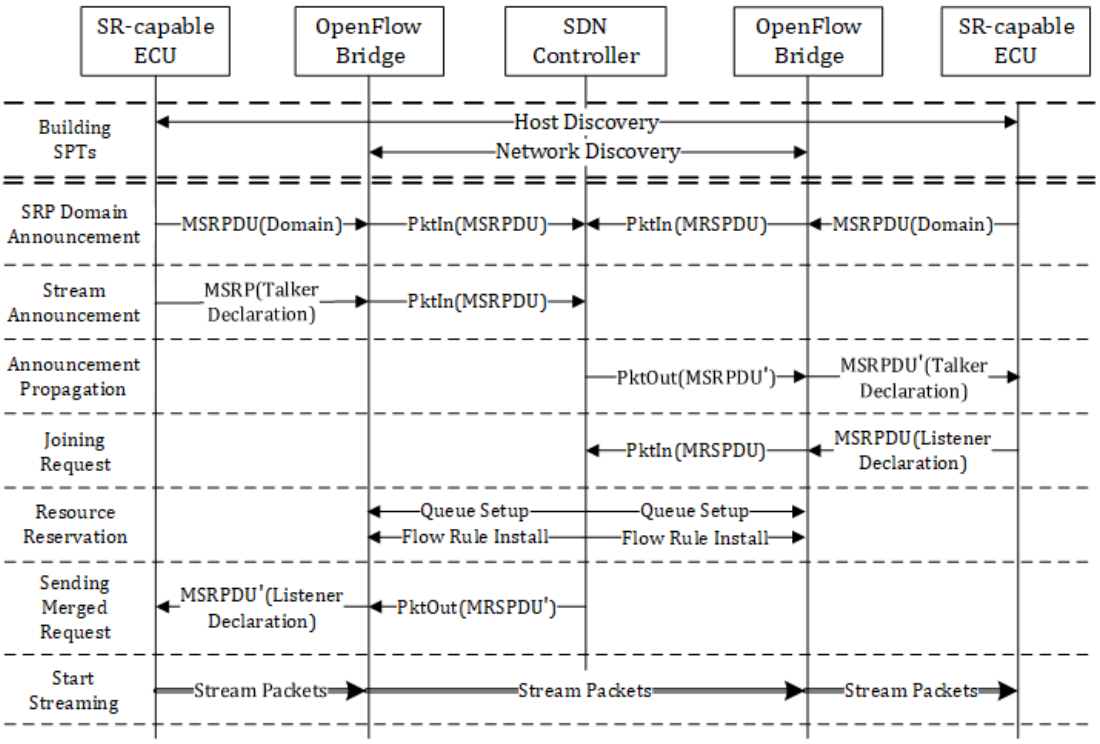


FIGURE 4. The proposed stream reservation phases.

and the priority of the SR class. If more than one SRP domain is announced from an ECU, multiple entries are created for the edge connection point toward the ECU. It is used for talker pruning.

- TalkerRegisterTable: It records the receiving connection point of a talker declaration made by MSRP-enabled ECUs with the stream ID of the talker declaration. The receiving connection point is called the talker connection point.
- TalkerDeclarationTable: It records all talker declarations, which are propagated through edge connection points toward MSRP-enabled ECUs. Each entry consists of the stream ID of a talker declaration, the edge connection point used to propagate the talker declaration, and the talker declaration. The edge connection point is called the declaring connection point.
- ListenerDeclarationTable: It records events of listener declarations. It is used to merge events of all listener declarations sent by candidate members of the AV stream. Each entry consists of a stream ID of the listener declaration and a list of *Listener*.
- Listener: A *Listener* consists of the edge connection point from which the listener declaration is received and its final event. The final event is decided by the *availability check* procedure.

V. THE PROPOSED STREAM RESERVATION MECHANISM

Fig. 4 illustrates the AV stream reservation phases in our scheme.

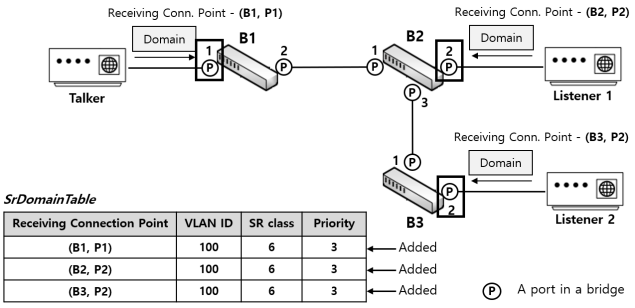


FIGURE 5. SRP domain announcement processing.

A. SRP DOMAIN ANNOUNCEMENT

An MSRP-enabled ECU must announce its SRP domains by sending SRP domain announcements. When the edge connection point receives an SRP domain announcement, it is forwarded to the SDN controller application using *PacketIn* message. The controller application records the edge connection point in the *PacketIn* message, VLAN ID, the SR class and the priority in the SRP domain announcement into the *SrDomainTable*.

Fig. 5 shows an example of the *SrDomainTable*. In the figure, three MSRP-enabled ECUs announces the same VLAN ID (100), SR class (6) and priority (3).

B. AV STREAM ANNOUNCEMENT

A talker announces an AV stream by sending a talker declaration to the nearest bridge. When the talker declaration arrives at the controller application, the controller application

executes the *talker declaration propagation* procedure. The talker declaration propagation procedure processes the talker declaration as follows:

1) It looks up the *SrDomainTable* using the receiving connection point of the talker declaration, the VLAN ID and the priority in the talker declaration. If a matching entry exists, it extracts the SR class value from the matching entry. Otherwise, it ignores the talker declaration, since it cannot identify the SRP domain where the traffic of the new AV stream can be sent.
2) It adds an entry to the *TalkerRegisterTable*. The entry stores the stream ID and the receiving connection point.
3) It searches the *SPT* by looking up the *SptTable* with the bridge ID in the receiving connection point.
4) It executes the *talker pruning* procedure for each *Path* in the *SPT* with the extracted SR class, the VLAN ID, and the priority of the talker declaration. If it returns the status code ''true'', the *Path* is added to the *list of not pruned Path*.
5) It executes the *talker declaration path propagation* procedure for each *Path* in the *list of not pruned Path* using the extracted SR class and the talker declaration.

The *talker pruning* procedure checks whether the input *Path* must be pruned using the input VLAN ID and priority as follows:

1) It checks whether the input *Path* connects the talker. If so, it returns ''false''.
2) It looks up the *SrDomainTable* using the last connection point in the list of connection points in the input *Path*. If a matching entry has the same VLAN ID, SR class, and priority with those of the input VLAN ID, SR class and priority, it returns ''true''. Otherwise, it returns ''false''.

The *talker declaration path propagation* procedure updates the talker declaration to reflect the path characteristics and verifies the resource availability on the path as follow:

1) For each connection point in the *Path*, it executes the *availability check* procedure with the talker declaration. At any connection point, if it returns the status code ''unreservable'', the declaration type of the talker declaration is changed to ''Talker Failed''.
2) It adds the latency of the *Path* to the accumulated latency field of the talker declaration.
3) It records the modified talker declaration with the stream ID of the talker declaration and the last connection point in the *Path* into the *TalkerDeclarationTable*.
4) It forwards the modified talker declaration to listeners connected at the last connection point in the *Path*.

The *availability check* procedure checks resource availability on the input connection point with the input talker declaration as follows:

1) It locates the *BridgeInformation* in *BridgeList* using the bridge ID of the input connection point.
2) It looks up the *StreamTable* of the *BridgeInformation* using the stream ID in the input talker declaration to find *StreamInformation*.
3) If there is a matching entry, it accesses the *PortMap* in the entry. Then, it checks whether the bit value of the port ID position in the *PortMap* is one. If so, the port is already reserved for the AV stream; thus, it returns the status code ''reserved''.
4) It locates the *PortInformation* in the *PortTable* of the *BridgeInformation* using the port ID of the input connection point.
5) It locates the *TrafficClassInformation* in the *TrafficClassTable* of the *PortInformation* using the priority of the input talker declaration.
6) It checks whether the traffic class at the port has available bandwidth requested by the input talker declaration. If so, it returns the status code ''reservable''. Otherwise, it returns the status code ''unreservable''.

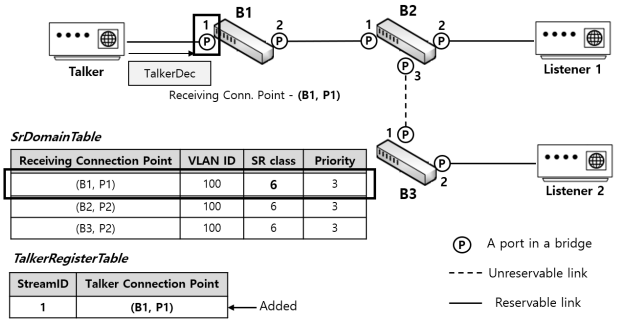
Fig. 6 shows an example of the *talker declaration path propagation* procedure. In the figure, it is assumed that the talker and listeners are in the same SRP domain. The SRP domain uses the VLAN ID (100), the SR class (6) and the priority (3). A talker sends a talker declaration which contains the declaration type (Talker Advertise), the stream ID (1), the VLAN ID (100), and the priority (3).

In Fig. 6(a), the talker declaration sent by the talker arrives on the edge connection point (B1,P1). It is forwarded to the controller application and the talker declaration propagation procedure handles the forwarded talker declaration. It looks up *SrDomainTable* with (B1, P1), the VLAN ID (100), and the priority (3) in the talker declaration. Then it extracts the SR class value (6) and records the stream ID (1) and the receiving connection point (B1, P1) to the *TalkerRegisterTable*.

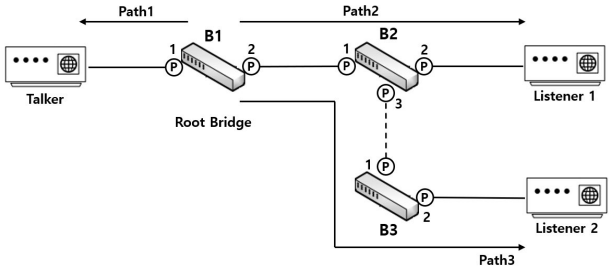
Then it searches the SPT rooted at the bridge B1. Fig. 6(b) illustrates the found SPT. The SPT consists of three *Paths*. The first *Path*, PATH1, connects the talker, the second *Path*, PATH2, connects the listener 1, and the third *Path*, PATH3, connects the listener 2.

The last connection point in PATH1 is (B1,P1) and it connects the talker; thus, PATH1 is pruned. The last connection point of PATH2 is (B2,P2), and a matching entry for the connection point has the same SRP domain information for the input SR class (6), the VLAN ID (100), and the priority (3) of the talker declaration. Therefore, PATH2 belongs to the SRP domain (100, 6, 3). PATH3 also belongs to the same SRP domain. They are not pruned. Fig. 6(c) shows the result of the talker pruning for each *Path* in the *SPT*.

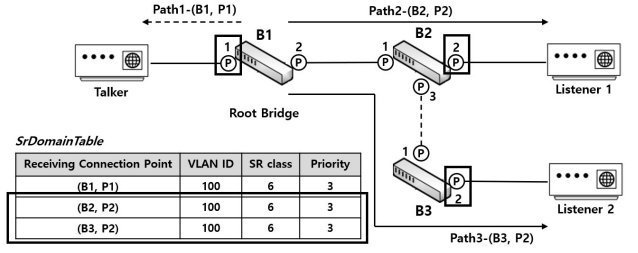
Connection points in PATH2 are (B1,P2) and (B2,P2). The bridge B1 has no *StreamInformation* for the stream ID (1) in its *StreamTable* and (B1,P2) has available bandwidth for the AV stream. Bridge B2 also has no *StreamInformation* for the stream ID (1) in its *StreamTable* and (B2,P2) has available bandwidth for the AV stream. Therefore, the procedure only adds the latency of PATH2 to the talker declaration. Then, the procedure records the stream ID (1), the last connection point (B2,P2) and the modified talker declaration into the *TalkerDeclarationTable*. Finally, the procedure forwards the modified talker declaration to the listener 1 via the last



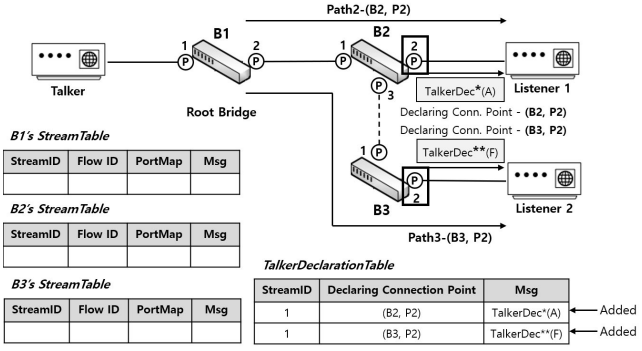
(a) Registration of the talker connection point



(b) Identification of the *SPT*



(c) Talker pruning



(d) Availability check and forwarding

FIGURE 6. **Talker declaration propagation procedure.**

connection point (B2,P2) by sending a *PacketOut* message. Connection points in PATH3 are (B1,P2), (B2,P3) and (B3,P2). Bridge B1, B2 and B3 have no *StreamInformation* for the stream ID (1) in their *StreamTable*. However, the connection point (B2,P3) cannot supply the resource for new AV stream. Therefore, the procedure adds the latency of PATH3 to the talker declaration and changes the declaration type of the talker declaration to ''Talker Failed''. The modified talker declaration is then stored with the last connection point (B3,P2) into the *TalkerDeclarationTable* and forwarded via the last connection point (B3,P2). Fig. 6(d) illustrates this process.

C. JOINING THE AV STREAM

A listener may join a new AV stream by sending a listener declaration of which the event is ''Ready'' to the nearest bridge. When the listener declaration arrives at the controller application, the controller application executes the *listener declaration propagation* procedure. It processes the listener declaration as follows:

1) Let the *listener declaration event* variable be that of the received listener declaration.
2) It checks whether the receiving connection point had sent the corresponding talker declaration. If any matching entry exists in the *TalkerDeclarationTable*, it processes the listener declaration. Otherwise, it ignores the listener declaration. Let *corresponding talker declaration* variable be the corresponding talker declaration.
3) It finds the talker connection point from *TalkerRegisterTable* using the stream ID of the listener declaration. The talker connection point is used to locate the *SPT* rooted at the bridge identified by the bridge ID in the talker connection point. It selects the *Path* in the SPT, matching the last connection point in the *Path* with the receiving connection point.
4) It reverses the list of connection points in the *Path*, then executes the *availability check* procedure for each connection point in the reversed list with the *corresponding talker declaration*. If the returned status code is ''reservable'', it adds the connection point to the *candidate list*. If the result is ''unreservable'', it changes the *listener declaration event* to ''Asking failed''. If the result is ''reserved'', it **terminates** this procedure because all remaining connection points in the reversed list are already reserved by another listener of the AV stream.
5) If the final *listener declaration event* is ''Asking failed'', it does not reserve resources at any connection points in the *candidate list*. Otherwise, it executes the *reservation* procedure for each connection point in the *candidate list* with the *corresponding talker declaration*. The *reservation* procedure performs actual reservation for the AV stream at the connection point using the *corresponding talker declaration*.
6) It searches a list of *Listener* in the *ListenerDeclarationTable* using the stream ID of the listener declaration. If the list is not found, it creates a list of *Listener* with the stream ID of the listener declaration.
7) It creates new *Listener* entry with the receiving connection point and the *listener declaration event*, and appends it to the list of *Listener*.
8) It merges all events in the list of *Listener*.
9) It overwrites the event in the listener declaration with the merged event and forwards the modified listener declaration to the talker through the talker connection point found at Step 2.

The *listener declaration event* indicates whether or not the *Path* to the listener can completely provide the AV stream to the listener. The *candidate list* traces all connection points that must be newly reserved in the *Path* to provide the AV stream to the listener. In bridge-by-bridge propagation, a bridge in a path reserves resources for an AV stream even though another bridge on the same path is unable to reserve the resources for the AV stream. This is caused by the lack of path sequence information over the distributed architecture. This partial reservation problem is prevented in our scheme.

The *reservation* procedure makes a reservation at the input connection point with the input talker declaration as follows:

1) It looks up the *StreamInformation* with the stream ID of the input talker declaration as described in Step 1-2 of the *availability check* procedure. If the matching entry doesn't exist, it creates a new entry and initializes the entry with the stream ID of the input talker declaration and the input talker declaration. The *PortMap* of the entry is initialized as all zeros and the flow ID of the entry is also zero.
2) It sets the bit value at the position of port ID in the input connection point in the *PortMap* to one.
3) It installs the OpenFlow flow rule for the AV stream at the bridge indicated by the bridge ID in the input connection point. The OpenFlow flow rule is created using the VLAN ID, the priority and the destination MAC address which are taken from the input talker declaration as well as all active port IDs taken from the *PortMap*. It updates the flow ID in the entry found or created in Step 1 with the ID of the created flow rule.
4) It locates the *TrafficClassInformation* with the priority of the input talker declaration as described in Step 1, Step 4, and Step 5 of the *availability check* procedure. Then, it updates the *TrafficClassInformation* with the bandwidth requested by the input talker declaration.
5) It sends a CBS parameter update request to the bridge indicated by the bridge ID of the input connection point. The CBS parameter update request consists of a port ID, a priority, and CBS parameters. The port ID is the port ID of the input connection point and the priority is taken from the input talker declaration. CBS parameters are taken from the *TrafficClassInformation*. When the actual bridge receives the request, it executes the linux *tc* command [19] with the interface name, the queue ID and CBS parameters. The interface name and the queue ID are mapped from the port ID and the priority in the request, respectively.

Fig. 7 shows an example of the *listener declaration propagation* procedure. In the figure, the stream ID is one and the event of the listener declaration is"Ready".

In Fig. 7(a), the listener declaration sent by listener 1 arrives at the edge connection point (B2,P2). The procedure finds an entry in the *TalkerDeclarationTable* for stream ID (1) and the receiving connection point (B2,P2). It finds the talker declaration sent at Fig. 6(d).

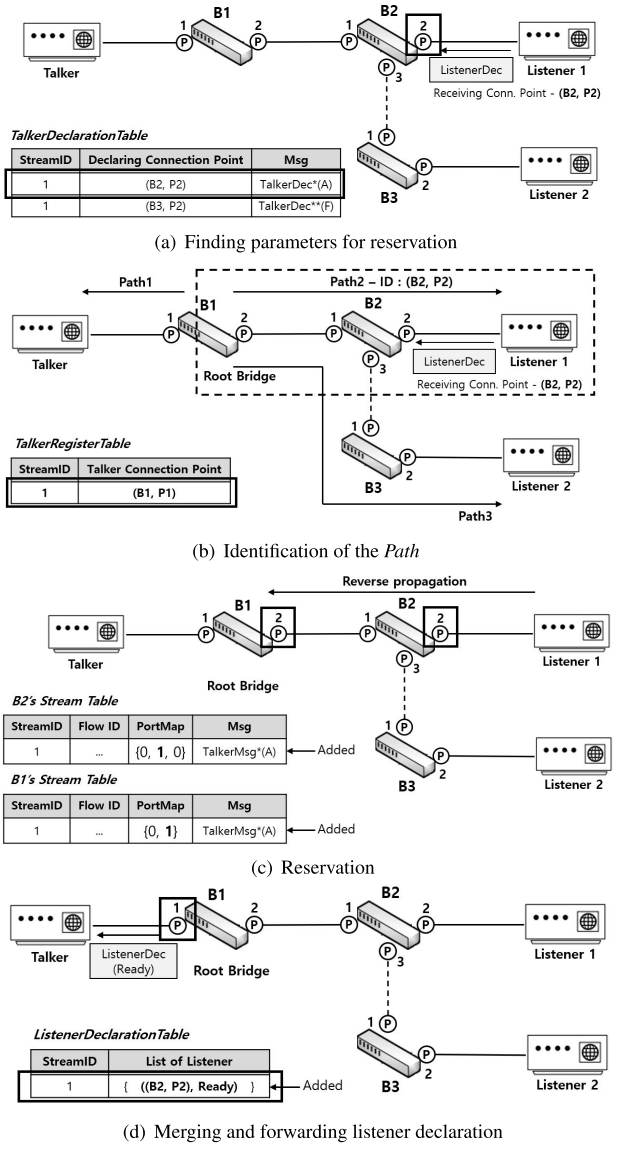


FIGURE 7. Listener declaration propagation processing.

The procedure also finds the talker connection point (B1,P1) in the *TalkerRegisterTable* with stream ID (1). The talker connection point (B1,P1) is used to locate the *SPT* in *SptTable*. It locates PATH2 in the *SPT*, since the last connection point in PATH2 matches with the receiving connection point (B2,P2). Fig. 7(b) illustrates these operations.

The procedure reverses the list of connection points, (B1,P2) and (B2,P2) in PATH2. Then, it executes the *availability check* procedure with the reversed list. As a result, the *listener declaration event* still has the initial event "Ready" and the *candidate list* contains (B2,P2) and (B1,P2). It executes the *reservation* procedure which creates an entry in the *StreamTable* of the *BridgeInformation* for each connection point in the *candidate list*. Fig. 7(c) illustrates the *reservation* procedure.

Finally, the procedure creates a new entry in the *ListenerDeclarationTable* with the stream ID (1), since there is no

entry for the stream ID (1). Then it appends a new *Listener* entry which contains the receiving connection point (B2,P2) and the *listener declaration event* "Ready" to the list of *Listener*. Then it merges all events in the list. The received listener declaration is updated with the merged event and sent to the talker via the talker connection point (B1,P1). Fig. 7(d) shows these operations.

VI. EXPERIMENTS

We have implemented a prototype MSRP controller application over the Open Network Operating System (*ONOS*) SDN controller. The prototype uses *Mininet* [16] to install MSRP-enabled hosts (Hi) and OpenFlow-enabled bridges (Bi). In our experiments, there are four hosts and four bridges. Each bridge acts as an Open Virtual Switch (*vSwitch*) [17]. Bridges interact with the *ONOS* SDN controller using the OpenFlow protocol version 1.4 [11]. The current OpenFlow version does not support QoS control for the Credit Based Shaper (CBS) [18]. Since our experiments use *Mininet*, we used the Linux *tc* command directly to control CBS parameters of ports on bridges instead of sending a CBS parameter update request. The *tc* command is executed by the MSRP controller application. All programs run over Linux Ubuntu 18.04 over the Virtual Box Virtual Machine (VM) at a PC with AMD Ryzen-5 2600 CPU with 48GB Memory.

The bandwidth of output ports on hosts and bridges is limited by the Hierarchy Token Bucket (HTB) [19], [20]. In our experiments, each link speed is limited to 10Mbps. Each scheduler at an output port uses the priority queue discipline with a single queue, which has 3 bands. Band 0 and 1 are used by SR class A and B, respectively. They are scheduled by the CBS. Band 2 is used by the best effort traffic.

In our experiments, there are two streams. One is an AV stream from a talker (H1) to potential listeners (H2 and H3). The other one is a best effort traffic from H4 to H2. We implemented MSRP talker and listener applications. The measured data rate of the AV traffic from the talker was 6Mbps. *Wireshark* is used for the measurement. The data rate of the best effort traffic generated by Linux *iperf* was 7Mbps. If two streams exist at the same time, the connection point (B1,P2) is overloaded. Fig. 8 shows the architecture used for our experiments. We ran the experiments 20 times.

Experiment steps are as follows:

1) All MSRP-enabled hosts (H1, H2, H3) send same SRP domain announcements.
2) At the beginning of an experiment (t=0), H1 announces its AV stream and H4 starts the best effort stream.
3) After 5s (t=5), H2 joins the AV stream. As soon as H1 receives the joining request, it starts the AV stream.
4) After 5s (t=10), H3 joins the AV stream.
5) After 20s (t=30), H2 leaves the AV stream.
6) After 5s (t=35), H3 leaves the AV stream.

We measured the processing time for each MSRP messages after receiving at the application. Initially, when the controller application receives each MSRP message, it takes a very long time to process. But when it processes the following MSRP messages by re-executing MSRP host applications, the MSRP message processing times decrease significantly. The initial processing and average re-processing times are (8.622ms and 0.117ms), (2.959ms and 0.371ms), (23.746ms and 4.43ms), (10.416ms and 4.735ms) for an SR domain announcement, a talker declaration, listener declarations to join and leave the AV stream, respectively. Table 1 presents the results of the measured processing time, with the exception of the initial processing time.

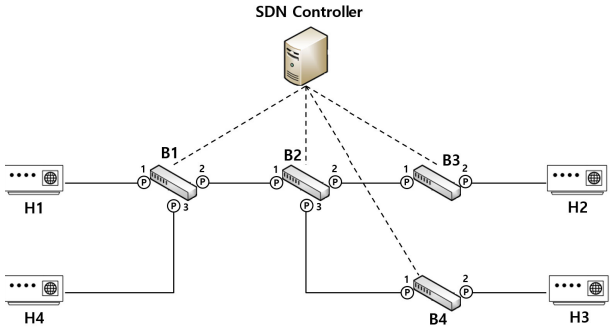


FIGURE 8. The architecture of the experiments.

TABLE 1. Table for measured processing time.

Message type	Mean	Standard deviation
Domain	0.117ms	0.023ms
Talker Declaration	0.371ms	0.043ms
Join Listener Declaration from H2	4.430ms	1.011ms
Leave Listener Declaration from H2	4.735ms	0.687ms
Join Listener Declaration from H3 when the Stream to H2 is reserved	2.868ms	0.376ms
Leave Listener Declaration from H3 when the Stream to H2 is reserved	2.814ms	0.360ms

The processing time of a listener declaration for H3 and the processing time of a listener declaration for H2 are different in our experiments. Since H2 joins the AV stream before H3, the processing for the joining request from H3 only reserves connection points (B2,P3) and (B4,P2) which are not reserved yet. The reservation at (B1,P2) is shared with H2, so this reservation is omitted for H3.

Fig. 9 shows the result of our experiments. It shows the receiving rates of two streams at H2 and H3. H2 receives both streams and H3 receives only the AV stream. Before H2 joins the AV stream, it receives only the best effort stream. As soon as it joins the AV stream, it receives both streams. The AV stream is congested at links between S1 and H2, so the receiving rate of the best effort stream is steeply dropped to 2Mbps. The receiving rates of the AV stream at H2 and H3 are stable at 6Mbps. The receiving rate of the best effort traffic is still dropped after H2 leaves the AV stream, since H3 has not left the AV stream. After all listeners leave the AV stream, the receiving rate of the best effort stream is restored to its sending rate.

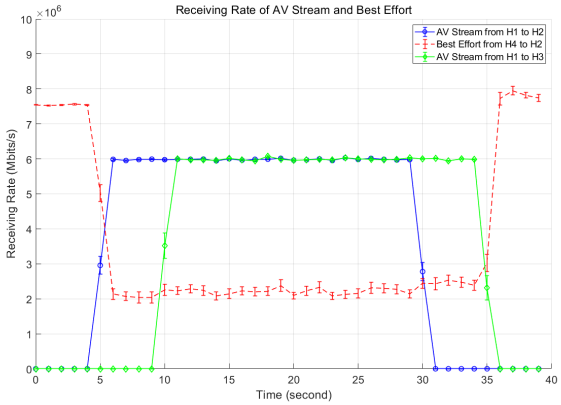


FIGURE 9. Traffic receiving rates at Host 2 and Host 3.

We experienced some mismatched rates in our experiments. We assumed that since *Mininet* is a virtual test bed and development environment, its performance is affected by underlying workloads. The best effort traffic sent by *iperf* with 7Mbps parameter exceeds its maximum rate by about 0.5Mbps. We measured the data rate of the best effort traffic using *Wireshark* and it was 7.5Mbps. The data rate of the best effort traffic surges to about 7.8 Mbps from t=36 to t=38 since our work conserving scheduler at (B1,P2) flushes its queued packets at band 2, used by best effort traffic, after the reserved bandwidth of band 0 is freed at t=35.

In addition to bandwidth measurements for each traffic, we also measured end-to-end latency for each traffic at H2. The architecture illustrated in Fig. 8 is used for this measurement. The AV traffic from H1 and the best-effort traffic from H4 use 7Mbps. H1 announces its AV stream and H2 joins the AV stream in advance. In this measurement, H1 does not send the AV traffic to H2 as soon as it receives a listener declaration with "Ready". After that, we manually turn on the best-effort traffic generator program at H4 and subsequently, the AV traffic generator program at H1.

Note that our experiment environment is controlled by software, not by hardware. Even though speeds of links in Fig. 8 are restricted to 10Mbps by the HTB, their original link speeds are much higher than 10Mbps. When we measured the 1-link propagation time of an 1512-byte frame, the average propagation time was about 2μs. However, when links are overloaded, their transmission timings are controlled by the HTB. Therefore, the maximum interfering time is calculated with 10Mbps, which is the HTB link speed. In the experiment, two traffics use 1512-byte frames. As a result, the maximum interfering time is about 1.2ms. Since this value is dominant compared with the propagation time and we assumed that the processing time is also extremely low, we roughly calculated the 1-link maximum delay bound using only the maximum interfering time. The maximum delay bound is 4.8ms since there are four links between H1 and H2.

Fig. 10 shows the result of the measurement. The red line in the figure represents the maximum delay bound calculated above. In this measurement, the AV traffic generator program operated at t=110ms. We measured latency between t=60ms and t=160ms. Between t=60ms and t=110ms, only the best-effort traffic exists so that the average of the measured latency is extremely low (9μs). However, the measured latency of the best-effort traffic steeply increases after t=110ms. This is because the best-effort traffic is delayed by the AV traffic. Since class A has the highest priority, the AV traffic can be selected first if the CBS of class A has enough credits. While the latency of the best-effort traffic increases, the measured latency of the AV traffic is lower than the maximum delay bound and the average of the measured latency is obtained as 1.3ms.

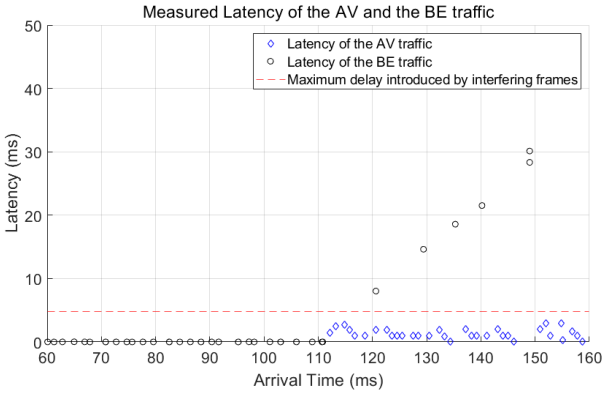


FIGURE 10. Latency of the AV traffic and the best-effort traffic measured at H2.

We ran an additional experiment to prove that our scheme prevents partial reservation problem well. In the experiment, H4 is also an MSRP-enabled host and acts as a talker. It announces its AV stream which uses 6Mbps. H1 also announces an AV stream which uses 6Mbps. In this experiment, there are two additional links from B1 to B2 and from B1 to B3. When B1 receives the AV traffic from H1, these links relay the AV traffic from H1 to B2 and B3 unconditionally. If the AV traffic from H1 to H2 is partially reserved at connection points (B2, P2) and (B3, P2), the AV traffic will be forwarded towards H2 at B2 and B3. Fig. 11 illustrates the experiment architecture to test partial reservations at B2 and B3.

The experiment was conducted 20 times and the experiment steps are as follows:

1) All MSRP-enabled hosts (H1, H2, H3, H4) send same SRP domain announcements.
2) At the beginning of an experiment (t=0), H1 and H4 announce their own AV streams. Also, for the experiment, H1 sends its AV traffic although H1 has not received any joining request.
3) After 5s (t=5), H3 joins the AV stream from H4.
4) Right away, H2 tries to join the AV stream from H1.
5) After 10s (t=10), H3 leaves the AV stream from H4 and the additional links from B1 to B2 and B3 are disabled.
6) Then, H2 tries to rejoin the AV stream from H1.

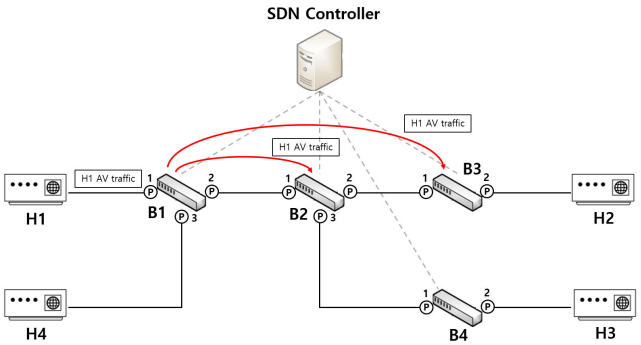


FIGURE 11. The architecture to test partial reservations at B2 and B3.

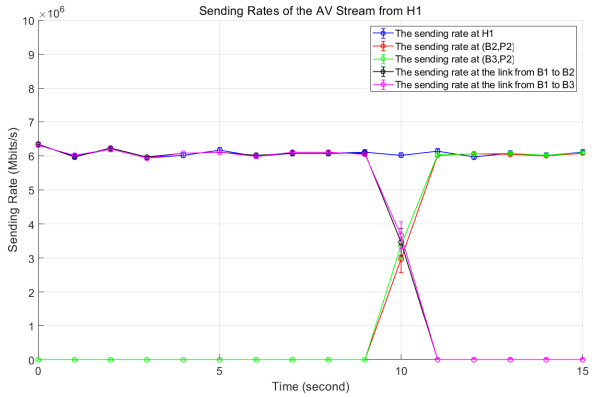


FIGURE 12. Traffic sending rates at H1, the additional links, (B2,P2) and (B3,P2).

At t=10, the additional links are disabled since the AV traffic from H1 is forwarded twice at B2 and B3 if a reservation is done and they are not disabled.

Fig. 12 illustrates the five sending rates of the AV traffic from H1 at H1, the additional links, (B2,P2) and (B3,P2). In the figure, the two sending rates at (B2,P2) and (B3,P2) are fixed as zero even after H2 tries to join the AV stream from H1 (t=5). It shows that AV traffics arrived at B2 and B3 through the additional links were not forwarded at B2 and B3 since there was no flow rule. This proves that there is no partial reservation problem in our scheme. If there were partial reservation problems at B2 and B3, flow rules for the AV stream from H1 would have been installed at B2 and B3 so that sending rates at (B2,P2) and (B3,P2) would not be zero. In addition, we also showed that H2 can join the AV stream from H1 even though it initially failed to join the AV stream. H2 successfully joins the AV stream from H1 at t=10 because (B1,P2) is reservable after H3 leaves the AV traffic from H4.

VII. CONCLUSION

The proposed MSRP scheme simplifies the time-sensitive stream establishment over bridged networks. It exploits the centralized topology and bridge information collected at the SDN controller. It centrally processes MSRP messages, so bridge-by-bridge propagation is not needed. Central processing can detect partial path reservation for a listener. In addition, since talker pruning is done by the central controller application and AV traffics are forwarded at each bridge using flow rules without FDB, VLAN ID and destination address registration for AV streams are not necessary. The experimental results show that the proposed scheme is feasible for real-time stream reservation algorithm in in-vehicle bridged networks.

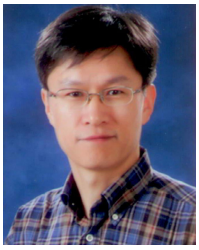
In the future, we plan to include a recovery algorithm against link failure. When a link in bridged networks fails, new *SPTs* in the *SptTable* must be reconstructed, and old paths between a talker and listeners that are affected by the failure must be compared with the new paths in order to update output port reservation of bridges included in the old and new paths. Note that in our scheme, MSRP state information related to AV streams (*SrDomainTable*, *TalkerRegisterTable*, *TalkerDeclarationTable* and *ListenerDeclarationTable*) is stored at the central controller application and the information doesn't contain any connectivity state for non-edge connection points. Therefore, as long as connectivity between MSRP-enabled ECUs and edge bridges are maintained, the controller application can restore paths for an AV stream between the talker and listeners without additional MSRP messages after new *SPTs* are rebuilt.

REFERENCES

[1] T. Steinbach, K. Müller, F. Korf, and R. Rollig, "Demo: Real-time Ethernet in-car backbones: First insights into an automotive prototype," in *Proc. IEEE Veh. Netw. Conf. (VNC)*, Dec. 2014, pp. 133–134.
[2] *IEEE Standard for Ethernet*, IEEE Standard 802.3-2018, (Revision of IEEE Std 802.3-2015), New York, NY, USA, 2018.
[3] *The IEEE Website*. (Apr. 15, 2021). [Online]. Available: https://1.ieee802.org/tsn/
[4] *IEEE Standard for Local and Metropolitan Area Networks—Bridges and Bridged Networks*, IEEE Standard 802.1Q-2018, (Revision of IEEE Std 802.1Q-2014), New York, NY, USA, 2018.
[5] D. Kreutz, F. M. V. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proc. IEEE*, vol. 103, no. 1, pp. 14–76, Jan. 2015.
[6] P. Berde, M. Gerola, J. Hart, and Y. Higuchi, "ONOS: Towards an open, distributed SDN OS," in *Proc. 3rd Workshop Hot Topics Softw. Defined Netw.*, 2014, pp. 1–6.
[7] J. Medved, R. Varga, A. Tkacik, and K. Gray, "OpenDaylight: Towards a model-driven SDN controller architecture," in *Proc. IEEE Int. Symp. World Wireless, Mobile Multimedia Netw.*, Jun. 2014, p. 19.
[8] *Technical Specification Group Core Network and Terminals; 5G System; Principles and Guidelines for Services Definition*, document ETSI, 3GPP TS 29.501 V15.08, 3GPP, Nov. 2020.
[9] T. Hackel, P. Meyer, F. Korf, and T. C. Schmidt, "Software-defined networks supporting time-sensitive in-vehicular communication," in *Proc. IEEE 89th Veh. Technol. Conf. (VTC-Spring)*, Apr. 2019, pp. 1–5.
[10] M. Böhm, J. Ohms, M. Kumar, O. Gebauer, and D. Wermser, "Dynamic real-time stream reservation for IEEE 802.1 time-sensitive networks with OpenFlow," in *Proc. 8th Int. Conf. Appl. Innov. IT (ICAIIT)*, Mar. 2020, pp. 1–6.
[11] *The Open Networking Foundation (ONF) Website*. (Apr. 15, 2021). [Online]. Available: https://www.opennetworking.org/software-defined-standards/specifications/
[12] *IEEE Standard for Local and Metropolitan Area Networks—Virtual Bridged Local Area Networks—Amendment 07: Multiple Registration Protocol*, Standard IEEE 802.1ak-2007, New York, NY, USA, 2007.
[13] S. Shenker and L. Breslau, "Two issues in reservation establishment," in *Proc. Conf. Appl., Technol., Archit., Protocols Comput. Commun. (SIGCOMM)*, Cambridge, MA, USA, 1995, pp. 14–26.
[14] *IEEE Standard for Local and Metropolitan Area Networks—Station and Media Access Control Connectivity Discovery*, Standard IEEE 802.1AB-2016, New York, NY, USA, 2016.

[15] Y. Li, Z.-P. Cai, and H. Xu, "LLMP: Exploiting LLDP for latency measurement in software-defined data center networks," *J. Comput. Sci. Technol.*, vol. 33, no. 2, pp. 277–285, Mar. 2018.
[16] *The Mininet Website*. (Apr. 15, 2021). [Online]. Available: http://mininet.org/
[17] *The Open vSwitch Website*. (Apr. 15, 2021). [Online]. Available: http://www.openvswitch.org/
[18] The Michael Kerrisk Website. (Apr. 15, 2021). *CBS—Credit Based Shaper (CBS) Qdisc*. [Online]. Available: http://man7.org/linux/man-pages/man8/tc-cbs.8.html
[19] The Michael Kerrisk Website. (Apr. 15, 2021). *TC—Show/Manipulate Traffic Control Settings*. [Online]. Available: https://man7.org/linux/man-pages/man8/tc.8.html
[20] *The TLDP Website*. (Apr. 15, 2021). *Classful Queueing Disciplines*. [Online]. Available: https://tldp.org/HOWTO/Adv-Routing-HOWTO/lartc.qdisc.classful.html

SANGJIN NAM received the B.S. degree in computer science from Korea University, South Korea, in 2020, where he is currently pursuing the Ph.D. degree in computer science and engineering. His research interests include the future internet, vehicular networks, QoS, mobility protocol, and software-defined networking.



HYOGON KIM (Member, IEEE) received the B.S. and M.S. degrees in computer engineering from Seoul National University, Seoul, South Korea, in 1987 and 1989, respectively, and the Ph.D. degree in computer and information science from the University of Pennsylvania, in 1995. From 1996 to 1999, he was a Research Scientist with Bell Communications Research (Bellcore). He is currently a Professor with Korea University. His research interests include wireless communication, vehicular networking, the Internet of Things (IoT), and mobile computing.

SUNG-GI MIN received the B.S. degree in computer science from Korea University, Seoul, South Korea, in 1988, and the M.S. and Ph.D. degrees in computer science from the University of London, in 1989 and 1993, respectively. From 1994 to 2000, he was with the LG Information and Communication Research Center. From 2000 to 2001, he was a Professor with the Department of Computer Engineering, Dongeui University, Busan, South Korea. Since 2001, he has been a Professor with the Department of Computer Science and Engineering, Korea University. His research interests include wired/wireless communication networks, mobility protocols, network architectures, QoS, and mobility management in the future networks.