Journal of Network and Computer Applications 153 (2020) 102505

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

The source-multicast: A sender-initiated multicast member management mechanism in SRv6 networks

Weihong Wu[a], Jiang Liu[a,*], Tao Huang[b]

[a] State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, China
[b] Purple Mountain Laboratories, China

ARTICLE INFO

Keywords:
Multicast
SRv6
Member management
BIER

ABSTRACT

The multicast is an efficient approach for one-to-many communications. However, multicast has not been widely deployed due to the lack of the perceptibility and operability to the receivers as well as the excessive status in the routers. In the typical multicast, the sender cannot operate the members in the group, which makes the multicast unable to meet the newly emerging demand, such as the sender-initiating, the reliability, and the security. Thus we put forward the source-multicast as a novel member management mechanism in SRv6 (Segment Routing IPv6) networks. We define some new SRv6 segments and the corresponding processes in different network entities. We present the Packet-Control Mode to provide the operability to the sender. We leverage the Bit Index Explicit Replication (BIER) as the transmission underlay to optimize the transmission. The source-multicast enhances the multiple-destinations ability to the IP, which allows the efficient multicast-like transmission pattern as well as the fine-grained member management. Some experiments are conducted in order to measure the processing latency in source-multicast and to find out the efficient conditions of source-multicast. The results indicate that the source-multicast is suitable for the scenarios having a large number of sessions, the bounded number of receivers, as well as the small scale multicast tree.

1. Motivation

The current network services are mostly deployed based on the end-to-end logic, because of the widely used TCP/IP. Along with the network developing, the services require more novel network features to meet emerging demands. There are more and more scenarios requiring the network to provide one-to-many services. For instance, the video conference, the IoT (Internet of Things), the distributed file system as well as the data center.

The video conference requires each participant receiving the same real-time video content simultaneously. Besides, the video conference should support the proactively dialing as well as the member verification. In IoT, the firmware update is often executed divisionally. The server needs to push the software to the part of the selected terminals. The distributed file system such as the HDFS (Hadoop File System) (Borthakuret al.) calls for alto-frequent multicast, in order to acquire the data from the alternative servers. And in the data center, according to (Zhu et al., 2017), the new demands contain the small group, reliability, sender-initiating, efficiency, and robustness.

The conventional IP multicast can provide efficient one-to-many transmission. It utilizes the class D addresses to identify one group of receivers. The traffic destined to the multicast address can be received by all the hosts which have joined the group. The intermediate routers can replicate the packets to the related downstream nodes, which ensures that in each link there only exists one piece of content. The commonly utilized routing protocols for multicast contain the PIM (Protocol Independent Multicast) (Deering et al.,) and the MOSPF (Multicast OSPF) (Moy). The member management protocols aim at maintaining the members in each group, such as the IGMP (Fenner, 2236) in IPv4 and the MLD (Deering et al., 1999) in IPv6.

The member management protocols have a correlation with deployment and performance. Regarding the IGMP, it requires each facility to maintain status for each session. Thus the excessive sessions in the networks would consume the cache resources in the routers very quickly. So the typical multicast is not suitable for scenarios with a large number of sessions. On the other hand, the member management can determine the processing in the routers, which can indirectly affect the performance. The IGMP has an efficient transmission, by reading the multi-

* Corresponding author.
E-mail addresses: wwh_bupt@foxmail.com (W. Wu), liujiang@bupt.edu.cn (J. Liu), htao@bupt.edu.cn (T. Huang).

https://doi.org/10.1016/j.jnca.2019.102505
Received 22 May 2019; Received in revised form 26 October 2019; Accepted 3 December 2019
Available online 24 December 2019

cast address to determine the replicating and forwarding. However, the member inquiring in IGMP needs the feedback from all the edge nodes, which would be inefficient if the session scale is small.

Besides, the typical IP multicast cannot satisfy the new multicast demands. The IP multicast can provide efficient transmission but it cannot provide the receiver-perceptibility and the receiver-operability to the sender. The IP multicast is designed on the basis of the end-to-group logic, in which the sender cannot perceive the receivers' information. The deficiency of the perceptibility and the operability can bring about some limitations to the typical IP multicast, for instance:

- **Initiative**: The sender cannot initiate the session to the assigned receivers.
- **Security**: The sender does not have the authority to process the session joining request.
- **Neutrality**: The session is established on the user plane, while the session management is executed by the network plane. Thus the sender cannot operate the receivers.
- **Efficiency**: All the members status are maintained on the routers. And each session requires one piece of multicast status.

Thus considering the new multicast demands, we put forward the source-multicast to extend the multicast logic to end-to-group-to-multi-end. The former half end-to-group can provide the multicast-like transmission, which can reduce the redundant content efficiently. The latter half group-to-multi-end can provide the unicast transmission, which the perceptibility and the operability.

Fig. 1 illustrates the logics of different transmission patterns in the one-to-many communication scenario. The unicast has the end-to-end logic. The sender must send the content to all the receivers respectively one by one, no matter in layer 3 or layer 2. The multicast has the end-to-group logic. The sender only needs to provide one piece of content to a group address. Each receiver can get the content from joining the group. The routers and the switches take charge of the content replication. The source-multicast has the end-to-group-to-multi-end logic. The sender provides one piece of content to a virtual group and informs the receivers' information to the routers. The edge gateways can replicate the content and forward it to all the receivers through unicast.

The remainder of this paper is as follows. In Section 2 we introduce some fundamental mechanisms related to the source-multicast. And in Section 3, we sketch some related works for multicast member management. The overview of the source-multicast is in Section 4. Subsequently, we elaborate on the design of the source-multicast in Section 5. Then we present the simulation evaluation in Section 6. Finally, we draw a conclusion and organize some future works in Section 7.

2. Fundamental introduction

2.1. SRv6

The source-multicast is designed based on the SRv6 (Segment Routing IPv6) (Filsfils et al.,) data plane. The SRv6 is the implementation of the SR (Segment Routing) (Filsfils et al., 2015) in IPv6. The SR can bring centralized control and programmability to the label-based switching. The SR combines the SDN (Software Defined Networks) and the label together. In SRv6, one segment stands for one endpoint or one specific action. The segment list can express the behaviours in each transmission path, including the route and the actions in specific routers. The segment list is carried in the Segment Routing Header (SRH) (Leddy et al.,). The SRH is derived from the IPv6 route header with the new Routing Type 4. The SRv6 programmability can introduce the layer 3 new features to the network, which is the basis of the source-multicast.

2.2. SRv6 multicast

The SRv6 natively supports the multicast through the SRv6 SIDs, called the Spray. In Spray, the network is divided into the SRv6 domain and the multicast domain. The SIDs in the SRH can specify the unicast-based routes in the SRv6 domain as well as the group in each multicast domain. The Spray is a suitable transmission mechanism for multicast in SRv6. But it can only provide the transmission matching the SRv6 mode, without the functional enhancement.

The TreeSID (Filsfils et al., 2019) is an approach derived from the Spray, which can support the control plane in SRv6 to compute the tree from the root node to a set of leaf nodes. The TreeSID has the SDN-like mechanism, which requires the controller to compute the tree for each session.

2.3. BIER

In order to optimize the transmission, we leverage the BIER (Bit Indexed Explicit Replication) (Wijnands et al., 2018) for multicast transmission. We classify the source-multicast as the member management mechanism because the routing and the forwarding belong to the BIER underlay category. The BIER is a novel approach for multicast transmission which can effectively reduce the status maintained in the router. It utilizes the Set Identifier(SI) and a BitString as the match fields. Each edge routers in BIER is identified by a determined bit position called the BFR-id (Bit-Forwarding Router ID). For instance, the BitString 101 identifies 2 routers, namely the router with the identifier 100 and the routers with 001. Thus in BIER, one BitString can express the multiple edge end routers, which can guide the intermediate routers to replicates the packets explicitly. The BIER-supported router is called the Bit-Forwarding Router (BFR). The ingress router of the BIER domain is called the Bit-Forwarding Ingress Router(BFIR) while the egress router is the Bit-Forwarding Egress Routers(BFERs).

In the BIER domain, all the BFRs maintain the Bit Index Routing Tables (BIRT) and the Bit Index Forwarding Tables (BIFT) towards all the possible BFERs (also the edge routers). Thus, in BIER, the table status in the routers is only related to the network topology. Besides, because the BIRTs are towards the edge endpoint, thus the unicast routing protocols can be utilized too. In order to eliminate the multicast storm and the loop, the BFR must modify the BitString after each matching process to prevent the redundant replication.

Fig. 2 illustrates an example of BIER processing in the BFR. The BFIR sends a packet with the BitString 0101. The BFR1 can replicate the packet to B and D. The BitString towards B is set to 0100 while the BitString to D is set to 0001. The modification of the BitString can prevent the downstream nodes from replicating the packet back to the upstream, which can ensure the route is the tree-like.

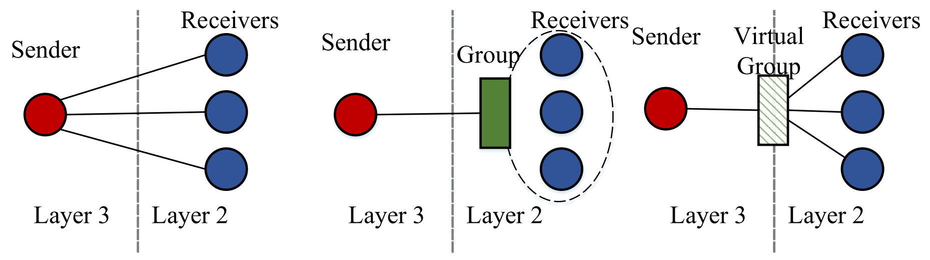
3. Related works

There exist some works for enhancing multicast member management. Most of them were presented for reliable multicast and the sender initiated multicast (SIM), because both of the reliability and the SIM require the receivers' information for packet recovery and session establishment respectively. These works have different mechanisms to provide the receivers' information to the sender. According to their principles, these works can be classified into 3 categories, namely the receiver-based, the agent-based as well as the protocol-based.

3.1. Receiver-based

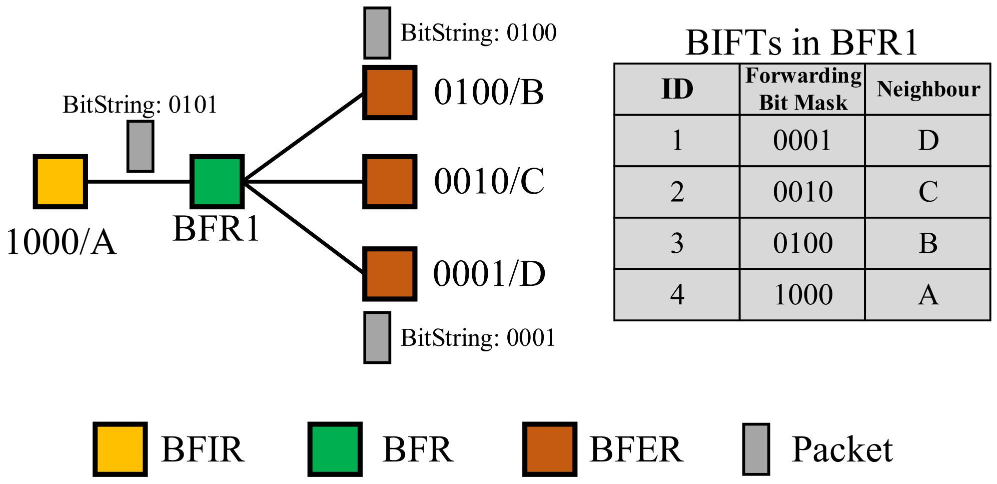
Sally Floyd proposed the SRM (Scalable Reliable Multicast) (Floyd et al., 1995), a reliable multicast framework for light-weight sessions in 1995. This framework exploits the NETBLT (Clark et al., 1987) as the multicast transport layer protocol, whereby providing the receiver-based packet recovery. The NETBLT can make the sender get the receivers' information directly, without the other interaction.

The RMTP (Reliable Multicast Transport Protocol) (Lin and Paul, 1996) aims at providing the sequenced and lossless delivery for mul-



(a) The IP unicast logic (b) The typical multicast logic (c) The source-multicast logic

Fig. 1. The logic of different approaches.



BIFTs in BFR1

ID	Forwarding Bit Mask	Neighbour
1	0001	D
2	0010	C
3	0100	B
4	1000	A

Fig. 2. An example of BIER processing. The match procedures in BFR1 is as follows. 1) Read the first (rightmost) bit, get 1. 2) Match ID 1, get the downstream D. 3) Set the BitString towards D to 0001&0101 = 0001. 4) Set the BitString in processing to %0001&0101 = 0100. 5) Read the second bit, get 0, bypass. 6) Read the third bit, get 1. 7) Match ID 3, get the downstream B. 8) Set the BitString towards B to 0100&0100 = 0100. 9) Set the BitString in processing to %0100&0100 = 0000, and processing ends.

ticast. The packet recovery in RMTP is based on positive acknowledgement (ACK). It groups the receivers into several hierarchical local regions. And for each region, a Designed Receiver (DR) is assigned to it in order to process the ACKs and to retransmit packets. The DR in RMTP acts as the proxy between the sender and the receivers, working for the member management.

The ARM(Active Reliable Multicast) (Lehman et al., 1998) was an NACK (negative acknowledge) based packet recovery scheme for reliable multicast. In ARM, the packet loss detection is executed by the receivers. If there happens the loss, the receiver sends an NACK message towards the sender for notifying the recovery. The related routers will cache and aggregate the multiple NACKs into one to ensure the unicity of the NACK. The routers also can get the subscription information of the receivers so that they can retransmit the packets to the required links.

These receiver-based mechanisms can provide the receivers' information only if the receivers have already been in the group. Thus, these works are suitable for the reliable multicast because the transmission and the recovery both work after the session has been established. However, the SIM requires the sender to determine the receivers ahead of the session. So these works are incapable of the SIM demands.

3.2. Agent-based

Most of the agent-based mechanisms are conducted based on the Software Defined Networks (SDN) because the controller in SDN can manage the networks globally.

Li proposed the RDCM (Reliable Data Center Multicast) for data center. The RDCM (Li et al., 2014) utilizes the OpenFlow (McKeown et al., 2008) controller to process the multicast message. In RDCM, all the IGMP messages should be redirected to the controller. So the controller can get the overall group members' status. Moreover, the RDCM has the peer-driven packet recovery mechanism, by retransmitting the repair packets through the P2P unicast.

Zhang presented the ECast (Zhang et al.,) for reliable multicast recovery in OpenFlow networks. They designed the elastic area multicast(EAM), in which the nodes are grouped into several elastic EAM sub-trees. Each EAM tree contains the receivers that require the same packets repairing. The sender need only retransmit the repair packets to the EAM tree, without perceiving each receiver.

The agent-based mechanisms can offload the management overhead to the selected agents. But the network is usually neutral for the users, while the agent is often a network management facility. Thus in most cases, the administrator would not allow the sender to operate the receivers through the agent. The control plane is often strictly transparent for the users, for the purpose of ensuring security and stability.

3.3. Protocol-based

The protocol-based mechanisms are mostly the sender-based too. The sender can get the receivers' information actively through the enhanced protocols.

The Single Connection Emulation (SCE) (Talpade and Ammar, 1995) introduces a single connection emulation sublayer between the single transport layer and the multicast network layer. The SCE can provide a reliable connection destined to a multicast address through the TCP. The SCE agents the multicast connection to fool the TCP layers into believing that there exist multiple single connections.

The M/TCP (multicast extension to Transmission Control Protocol) (Visoottiviseth et al., 2001) was designed to provide the SIM by means of modifying the TCP protocol to carry the individual receiver's address. The sender can attach the receivers' addresses list into the M/TCP header. During the TCP three-way handshake period, the receivers' information can be informed to the sender.

Jeacle and Crowcroft presented the TCP-XM (Jeacle and Crowcroft, 2005) to provide the unicast-enabled reliable multicast by embedding the multicast capability within the TCP. They also modified the TCP to encapsulate the group IP into the TCP header. The utilization of TCP-XM in hybrid multicast (Jeacle et al., 2005) and Grid environment (Jeacle and Crowcroft) were also proposed.

Zhu and his team designed a new transport protocol called MCTCP (Zhu et al., 2016) for data center reliable multicast. It's a TCP-based protocol so that it's also connection-oriented. The sender can establish the connection through the unicast, and the multicast address can be written in the SYN (synchronous) option field by the sender. Once the receivers get the SYN, they get the multicast address and then can join in it.

Most of these works are derived from the TCP. The handshake allows the interacting of the multicast information and the receivers' information. Thus, the SIM and the reliability can be introduced into multicast through these protocols. But the multicast is completely a layer 3 behaviour while the TCP is the layer 4 protocol. Thus, the protocol-based mechanisms narrow the usage of these approaches, such as the video which is often based on connectionless transmission.

4. Design overview

In this section, we explain the overview of the design for source-multicast, including the category of the routers, the principle of the Packet Control Mode, and the content of the design.

In source-multicast, the sender takes charge of the member management, including assigning the receivers when initiating a session. Because of the BIER, the multicast routing is needless, and the unicast IGP (Interior Gateway Protocol) can be leveraged for routing. Therefore, in our design, we do not discuss the routing algorithm.

All the routers in source-multicast are categorized based on their functions. We name the Source Control Router (SCR) for the layer 3 gateway router of the sender. Meanwhile, the gateway routers of the receivers are called the Access Control Router (ACR). The intermediate routers are called the Transit Routers (TR). From the BIER perspective, the SCR has the BFIR functions while the ACRs contain the BFERs functions. Because the BIER can provide the tree-like routing like the typical multicast, thus the SCR is also the root node of the tree while the ACRs are the leaves. An example of routers categories is illustrated in Fig. 3.

The design of the source-multicast contains 3 aspects, namely the SRv6 Enhancement, the Session Initiating as well as the member management.

According to Fig. 1, the source-multicast has end-to-group-to-multi-end logic. Thus, in order to address all the multiple ends, the sender must encapsulate all the addresses to be operated in the sent packets. We enhance the SRv6 to design some new SIDs (Segment Identifier) for carrying the addresses and indicating the operations. These new SIDs are explained in Section 5.1.

We introduce the stateful forwarding (Zhu et al., 2015) into the source-multicast. The source-multicast has the Packet-Control Mode, that is all the status operating should be triggered by the special packets which contain the operating information. The operations for the status can be creating, adding, deleting and acknowledging. Each operation must be numbered with one sequence for identifying. And each operation can be accomplished through multiple packets, according to the number of the addresses. The cache resource occupied by the status is measured in Section 6.2.

Considering the security, all the operations are agented by the SCR. The operations expressed in the packets is actually the operation requests. Only the authenticated sender's operating packets can be processed by the SCR. The SCR will re-express the operations to the ACRs with a special verification. The ACRs can only response the operations

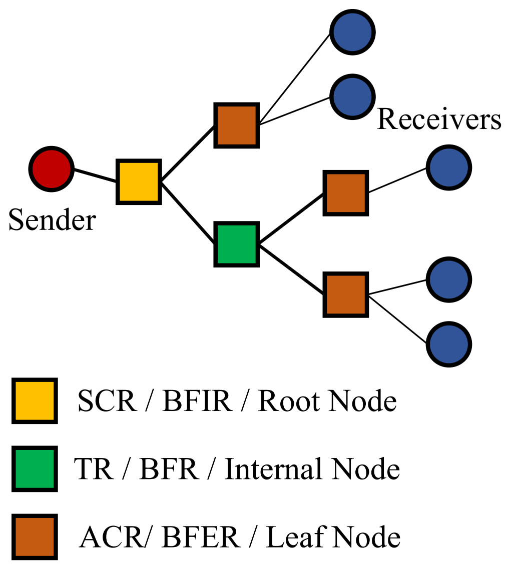


Fig. 3. An example of routers categories: The Sender's layer 3 (SRv6) gateway is the SCR, as well the BFIR and the root node of the multicast tree. A router is an ACR as long as there is at least one receivers belonging to it. The ACR is also the BFER and the leaf node. The other routers are the TRs, as well as the BFRs and the internal nodes.

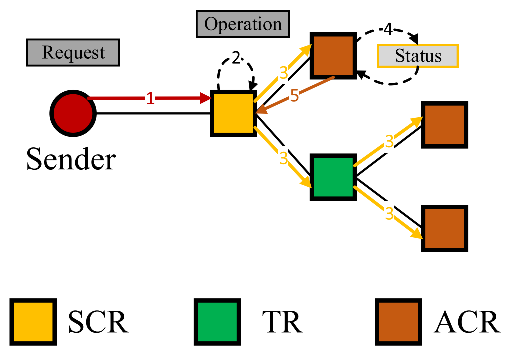


Fig. 4. The Packet-Control Mode principle: In this example, the sender is the initiator, the SCR is the agent, and the ACRs are the executor. 1) The Sender transmits the operation request in the packet. 2) The SCR authenticates the sender and converts the request to the operation with the verification. 3) The SCR transmits the operating packet to all the related ACRs in the form of BIER. 4) The ACRs execute the operation and operate the status accordingly. 5) The ACRs send back the acknowledgement to inform the accomplishment.

with the verification. Fig. 4 illustrates a Packet-Control Mode principle example.

The source-multicast allows the sender to initiate a session. The sender should send some operating packets for creating the status in the routers for the session. These packets can trigger different processes in SCR, TRs, and ACRs. The processes and their threads are explicated in Section 5.2.

In source-multicast, the member management is essentially the status modification in the routers. The sender can add or delete the members by providing the operating packets to the SCR. In Section 5.3 we introduce the member management procedures.

The stateful forwarding requires more resources in the routers because the routers need to maintain more operating status. Thus we leverage the BIER for transmission. The BIER can decrease the status in the routers, especially for the routing status. The overall resource occupation is measured and analysed in Section 6.

5. Design

5.1. SRv6 enhancement

In SRv6, the SID stands for the SRv6 label which can express an endpoint or some actions in the SRv6 domain. The SID has the prefix to express their fundamental functions, such as the End and Transit (Filsfils et al., 2019). We design 3 prefixes for source-multicast, namely MS (Members Status), AL (Address List), and BR (BIER). The MS notates the members status, which is utilized to indicate the operations concerning the status. The AL SIDs works for handling the address list in which carries the receivers' addresses. The BR is the implementation of BIER in source-multicast.

We design 6 SIDs called the SM SIDs for source-multicast. Each SM SID has the particular semantics and some of which can boot the process in the router for source-multicast processing. Besides, we modify the standard SRv6 End SID processing in order to accommodate the source-multicast. In Table 1 there shows the SM SIDs as well as the modified End.

The End is the basic function of SRv6, whose procedure is expressed in (Filsfils et al., 2019). In general the End SID is a 128-bit IPv6 address. We modify its procedure to accommodate the source-multicast. The detection of the SM SIDs in the SRH can boot the process for source-multicast, as illustrated in A.1.

The SM.Start is the start-tag of the SM SIDs list. In other words, the SM.Start can indicate the existence of the SM SIDs. It requires the uniqueness in the domain. All the intra-domain source-multicast sessions have the same SM.Start, although they may be initiated by different users.

The SM.Identifier corresponds to the source-multicast session one by one, mainly used to identify different sessions. It must be unique in the domain too. We utilize the MAC address as the uniqueness basis, appending 16-bit source port, 16-bit destination port, 32-bit Unix timestamp (Ritchie and Thompson, 1978) as well as a 16-bit random value. The structure of the SM.Identifier is shown in Fig. 5. The SM.Identifier can also trigger the process as illustrated in A.2.

The MS.Header controls the members status operation. Its structure is demonstrated in Fig. 6. The Operation Type indicates the expressed operation, containing the Address Notification, the Address Addition, the Address Deletion, and the Acknowledgement. The Verification Mark is the certificate for members status operation. The Verification Mark is generated by the SCR and transparent for all the users. The Operation SEQ indicates the operation sequence number, because one operation may be accomplished through multiple packets. We reserve some SEQs for special operations, shown in Table 2. These operations will be further explained in Section 5.2.

Each ACR can be represented by 1 specific bit position in BR.BitString. Thus the BR.BitString implies the multiple selected ACRs of one session. We treat each multicast as one BIER domain so that the sub-domain id is not required. In source-multicast, each BFER just utilizes the BFR-id to identify itself without the SI. So that, the source-multicast can support the multicast with a maximum of 127 ACRs. The calculating of the BitString can be realized through the approaches mentioned in (Wijnands et al., 2018), such as the BIER MVPN (Multicast Virtual Private Network) (Rosenet al., 2018).

Table 1
The enhanced SIDs.

SID	Function	Explanation
End	Information Carrier Process Trigger	Boot the source-multicast process
SM.Start	Information Carrier	The start-tag of the SM SIDs list in SRH
SM.Identifier	Information Carrier Process Trigger	The identifier of one certain source-multicast session
BR.BitString	Information Carrier	The BitString of BIER
AL.Header	Information Carrier	The header of the address list
AL.Header.AB	Information Carrier	The header of one address block
MS.Header	Information Carrier	The header for members operation

Table 2
The special operation SEQ.

Operation	SEQ
Address Notification	0xff
Active Leaving	0x0f
ACR Removal	0xf0

The AL.Header carries the control field for the address list. The structure of AL.Header is presented in Fig. 7. The Address Number indicates the number of addresses related to one operation. The AB Number is the number of the Address Block (AB), which will be further explained in Section 5.2. The End Address Number indicates the sequence number of the last address in one packet. For example, assuming that the sender has 100 addresses to operate, if there are 40 addresses that have been already handled and the current packet contains only 10, the End Address Number should be set to 50 in the AL.Header. The Address Type includes the IPv4, IPv6, IPv4 Prefix, IPv6 Prefix, and Hybrid. The Unused field is reserved for future works.

The AL.Header.AB is the control header for one Address Block. Its format is shown in Fig. 8. The Address Number indicates the number of the addresses belonging to this address block. The Address Type is the same as the AL.Header. But in one Address Block, the addresses' type must be the same hence the Hybrid type is invalid in AL.Header.AB.

5.2. Session initiating

The source-multicast allows the sender to initiate a session. In this section, we elaborate on the processes of the session initiating in the order of the network entities through which the packet passes. Because of the Packet-Control Mode, the sender should firstly generate some operating packets to inform the networks to establish the session-related status. After the status establishment, the sender can transmit the content datagram. We call the status establishment process the Address Notification, while the latter the Content Transmission. The Address Notification is the key point of the design.

5.2.1. Sender

The IP is connectionless, thus in IP networks, the sender can send datagrams to the receiver as long as knowing the destination IP, without acquiring the receiver's status. The source-multicast also has the similar connectionless logic that the sender can initiate the session as long as it knows the receivers' address and it has been permitted by the administrator. So, before the session initiating, we assume that the sender has already known the addresses of all the receivers.

The sender constructs the operating packets for Address Notification. The packets are SRv6-based, which contains the SRH and the SM SIDs. Fig. 9 shows the protocols stack of the originally sent packets. The SIDs in SRH have an inverted order, for example, the SM.Start is located in the last of the SRH but parsed first.

The SM.Start is defined by the administrator, and any intra-domain unique 128-bit value can be determined to be the SM.Start.

The SM.Identifier value is set accordingly with the actual conditions. The MAC address, the transport ports and the Unix time can be read directly from the OS (Operating System).

The values in MS.Header are set for Address Notification. The Operation Type is set to the Address Notification. The Operation SEQ is 0xff according to Table 2. The Verification Mark is set randomly just for filling because it is invalid for the sender.

The AL.Header and the AL.Header.AB are determined based on the receivers' addresses. Following the AL.Header.AB is the address list in

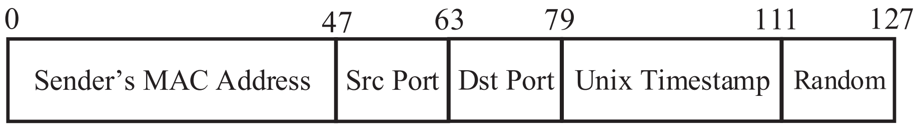


Fig. 5. The SM.Identifier structure.

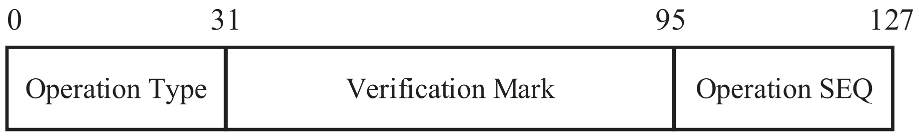


Fig. 6. The MS.Header structure.

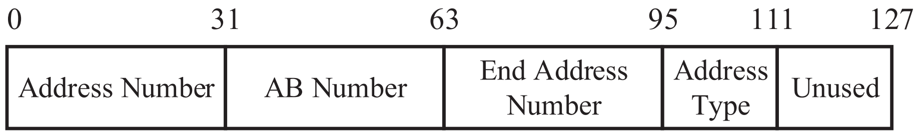


Fig. 7. The AL.Header structure.

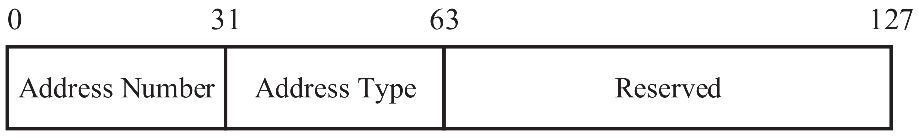


Fig. 8. The AL.Header.AB structure.

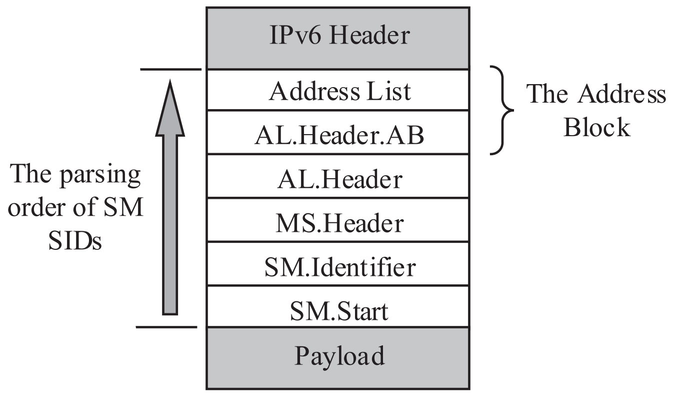


Fig. 9. The SM SIDs stack in SRH: The SRv6 requires the SIDs to be encapsulated inversely. The first parsed SID is located in the innermost of the SRH.

which carries the receivers' addresses. If the addresses cannot be carried in one packet, the sender should accomplish the Address Notification through multiple packets. And, these multiple operating packets have the same Operation SEQ in the MS.Header.

The IPv6 DA (Destination Address) is set the SCR. The address of the SCR is an End SID too, thus the modified procedures can help trigger the SM SIDs processing. In the SRH, the SL (Segment Left) points to the SM.Start, which means that the SM.Start is the next active SID.

5.2.2. SCR

The SCR is the entity reacting to the operations requested by the sender. The SCR is the starting point of the layer 3 domain. When the operating packets arrive at the SCR, the IPv6 DA is the first parsed SID, which is an End SID too. According to A.1, during the End processing, the SL can guide the SCR to parse the SM.Start.

When the SCR confirms the SM.Start, it can verify request based on the configuration according to the source IP. If the sender is permitted, the sender boots the Address Notification Process.

Fig. 10 illustrates the Address Notification Process diagram. It contains 2 threads, namely the Main Thread for the status establishment and the BR Thread for BIER transmission.

Main Thread-MS.Header: The Main Thread is triggered by the SID SM.Start with the default state INIT. Then the SCR decrements the SL to parse the next SID. The next SID is the MS.Header according to Fig. 9. When the SCR is processing the MS.Header, the state of Main Thread is PROCESS. The values in MS.Header can indicate the operations. Firstly, the SCR generates a random value as the Verification Mark, and saves it in the memory in the form of a tuple called the Status Tuple. Then the SCR starts the procedure Read. Finally, it triggers the BR Thread and the MS.Header procedure ends.

BR Thread-Init: The BR Thread is booted by the Main Thread. The initial procedure of the BR thread is Init, whose state is set to the INIT_WAIT. The BR Thread requires the Main Thread passing the AL SIDs (AL.Header, AL.Header.AB, address list) for routing processing. The Init is an On-Hold procedure which is until when the AL SIDs are started to be passed.

Main Thread-Read: When the Process procedure is Read, the SCR parses the required SIDs for the subsequent operations. For the Main Thread, the AL SIDs, the MS.Headr, and the SM.Identifier are required because the Main Thread will construct the packet later. The SM.Identifier will be appended to the Status Tuple if it does not exist in the Memory. For the BR Thread, it requires the AL SIDs for BIER-based operation. After the Read procedure, the Main Thread skips to procedure Input, while the state is set to PROCESS_WAIT.

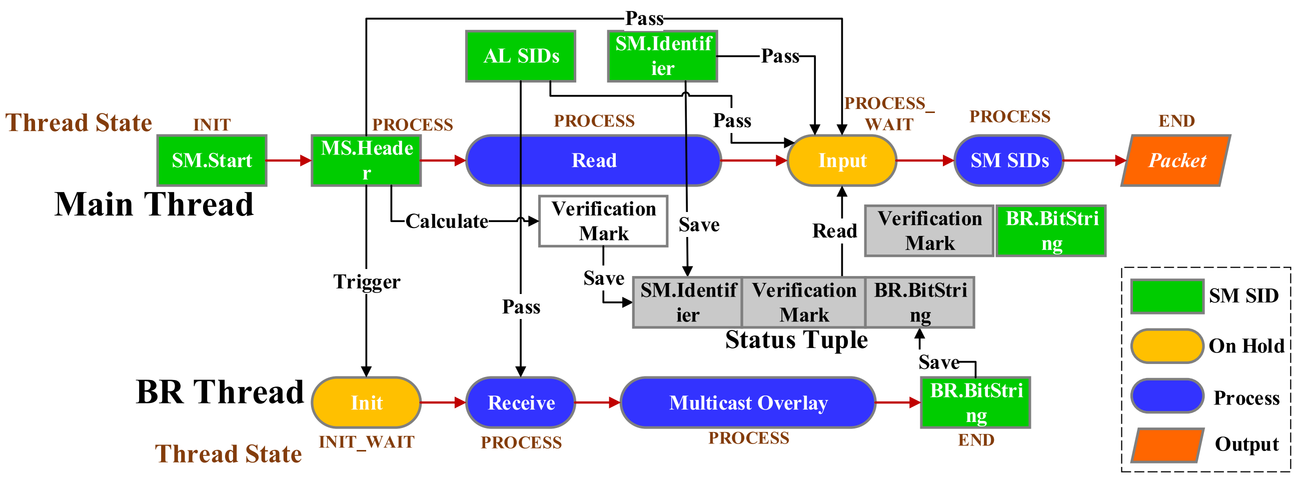


Fig. 10. The SCR process in Address Notification.

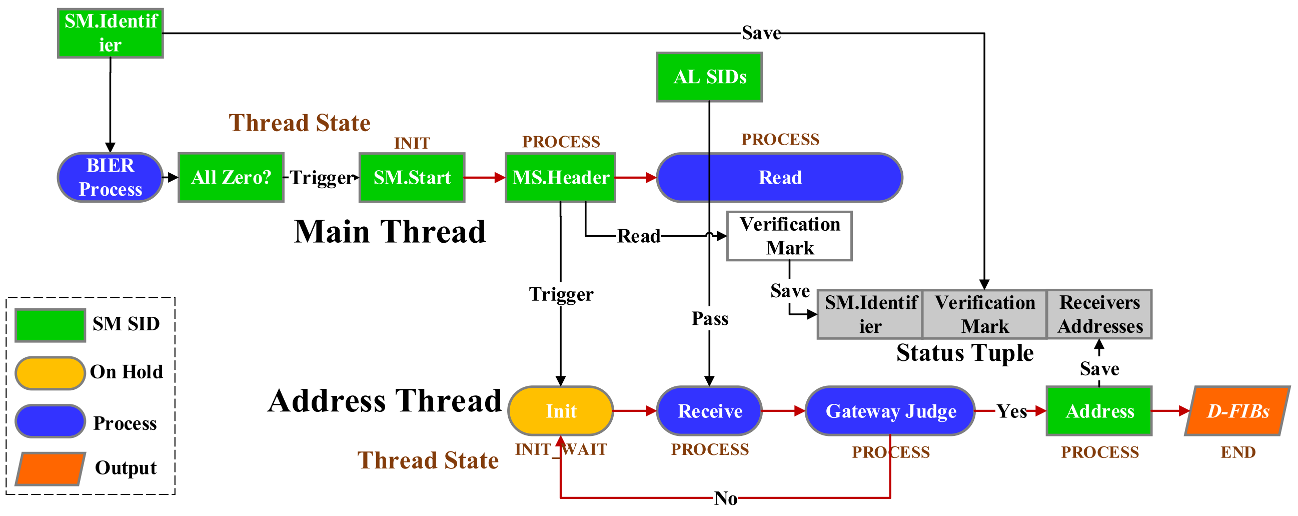


Fig. 11. The ACR process in Address Notification.

BR Thread-Receive: The Receive procedure starts at the AL SID comes. The Receive procedure can accept and parse the AL SIDs and then get multiple destinations. After that, the BR Thread starts Multicast Overlay procedure.

BR Thread-Multicast Overlay: The Multicast Overlay mainly aims at determining the ACRs according to the multiple destinations. Once the ACRs have been determined, the BR Thread can generate the BR.BitString based on the ACRs. Then it saves the BR.BitString into the Status Tuple. Until now the Status Tuple construction is accomplished. Afterwards, the BR Thread ends.

Main Thread-Input: The Input is an On-Hold procedure which should be until when all the required parameters have been successfully accepted. These parameters should be obtained both from the Main Thread and the Status Tuple. From the Main Thread, it can get the MS.Header, the SM.Identifier, and the AL SIDs. From the Status Tuple, the SCR gets the Verification Mark and the BR.BitString.

Main Thread-SM SIDs: After getting all the required parameters, the Main Thread can reconstitute the output packet. The Main Thread constructs the SM SIDs in the SRH with the acquired parameters in Input, according to the sequence shown in Fig. 9. In particular, the BR.BitString is inserted ahead of the SM.Start, which will be parsed after the SM.Start.

Main Thread-Packet: When the SRH has been constructed, the Main Thread can generate the packet with the IP payload, i.e. the upper layer protocols and data. The payload stays unchanged during the process. In particular, the output packet's IPv6 DA is set to the SM.Identifier.

After generating the output packet, the Main Thread ends. The SCR should process the packet transmission. Because the IPv6 DA is the SM.Identifier, thus the SCR should process it according to A.2. The SM.Identifier can help the SCR parsing the BR.BitString, and then determining the replication and forwarding on the basis of the standard BIER processing.

5.2.3. TR

When the packets arrive at the TR, the IPv6 DA SM.Identifier can trigger the transmission processing. Because the TR only transmits the packets in the multicast way, thus it would not operate the SM SIDs except reading the BR.BitString.

The transmission processing is the same as the SCR, that is the procedures shown in A.2. It parses the BR.BitString, and matches the BIER FIBs to determine the output ports and the corresponding modified BR.BitString.

5.2.4. ACR

The ACR processing also starts at parsing the SM.Identifier. Thus, the ACR should process the packet like the TR, i.e. the transmission processing. After the processing, the BR.BitString would be set to all-zero, which can trigger the Main Thread to process the SM SIDs. Fig. 11 demonstrates the processing for Address Notification in the ACRs.

Main Thread-MS.Header: The MS.Header is the first PROCESS procedure in the thread. The ACR firstly reads the Verification Mark from the MS.Header, and saves it into the Status Tuple. Then the Main Thread Triggers the Address Thread. And finally, it jumps to Read procedure.

Main Thread-Read: The Read procedure is similar to the SCR. In this procedure, the ACR parses the SRH for required SM SIDs including SM.Identifier for Main Thread and AL SIDs for Address Thread. The SM.Identifier will be saved into the Status Tuple correspondingly. When the Read is accomplished, the Main Thread ends.

Address Thread-Init: The Init is an On Hold procedure with the state INIT_WAIT. If there exists at least one unprocessed AL SID (passed from the Main Thread) in this thread, it jumps to Receive procedure.

Address Thread-Receive: The Receive procedure takes charge of the AL SIDs acquiring and parsing. It utilizes a pointer to locate the currently parsed address. After it passes the currently parsed address to the next procedure, it moves the pointer to the next address.

Address Thread-Gateway Judge: The Gateway Judge procedure can determine whether the address being processed belongs to the ACR. If it belongs to the ACR, it will be passed to the next procedure. If not, the thread turns back to the Init to acquire the next address.

Address Thread-Address: The Address procedure can acquire the address parameter which belongs to this ACR. This procedure firstly saves the address into the Status Tuple corresponding to the SM.Identifier. Subsequently, it can generate some FIBs for replicating the packet, removing the SRH, and constructing the IP header destined to the addresses in the Status Tuple. We call these FIBs the D-FIBs (Distributing FIBs). A working instance of D-FIBs is shown in Fig. 12.

The Address Notification operation may be accomplished through multiple packets because one packet might not be able to carry all the addresses. When all the ACRs have generated the corresponding D-FIBs, the Address Notification can be regarded as being accomplished. The sender requires the feedback from the network in order to terminate the Address Notification packets generating. The ACRs should feedback the acknowledgement messages to the sender. The acknowledgement is introduced in Section 5.4.1.

5.2.5. *Content Transmission*

When the sender receives the acknowledgement feedback from all the related ACRs, it can stop sending the Address Notification operating packets. Then the sender can start the Content Transmission to transmit the data.

Because all the related routers have built the Status Tuple, thus the address list is no longer needed in the Content Transmission. The originally sent packets are destined to the SCR, which only contain 2 SM SIDs, namely the SM.Identifier and SM.Start. The edge network can transmit the packet to the SCR based on layer 2. In the SCR, the BR.BitString should be inserted between the SM.Start and the SM.Identifier. The TR can replicate and transmit the packet explicitly based on the BR.BitString. The ACR can distribute the packets' payload to the receivers through unicast based on the SM.Identifier and the D-FIBs.

5.3. *Member management*

In the SCR, the Status Tuple maintains the correspondence among the SM.Identifier, the Verification Mark, and the BR.BitString, in order to verify the operation and provide the route indication (BR.BitString). In the ACRs, the Status Tuple contains the correspondence among the SM.Identifier, the Verification Mark, and the Receivers Addresses, which can replicate the payload and re-encapsulate them in the form of unicast.

Therefore, the Status Tuples in the ACRs are the intuitive expression of the member management. Any member management operation should be finally accomplished with the ACRs' Status Tuple modification.

5.3.1. *Sender*

The member management operations in source-multicast are initiated by the sender with the Packet-Control Mode.

The source-multicast can support both the active and passive member addition. The sender can specify the additional receivers without the pre-interaction. And the join request can also be issued by the receiver, and it can join in the group after the sender authentication.

The members may leave the group actively, passively, or accidently. If one member leaves the group actively, it should send a message to inform the sender. Then the sender can remove the status from the corresponding ACR. The member can also be removed by the sender passively. And if one member leaves the group by accident such as an outage, the sender should detect the leaving and update the status accordingly.

All the operations are expressed by the operating packets which are also treated as the requests in the SCR. The operating SM SIDs can be carried together with the content. In other words, the member management operations would not interrupt the Content Transmission.

The operating packet has the same protocol stacks as the Address Notification. The MS.Header can express the operation type while the AL SIDs contain the addresses to be operated.

5.3.2. *SCR*

The SCR can agent the member management operation request in the same form as the Address Notification. Because the addresses change may result in the route change, thus the SCR should check the route to determine whether to modify the BR.BitString.

Fig. 13 illustrates the process and the threads in the SCR for member management. The Main Thread is similar to the Address Notification. The procedures with the same names have the same processing. Compared to the Address Notification, the difference is that the calculation of the Verification Mark is no longer needed. The Main Thread works mainly for inserting the Verification Mark into the SRH so that the ACRs can execute the operations legally.

In the Main Thread, the MS.Header procedure can trigger the BR Thread. The BR Thread has one more procedure compared to the Address Notification (Fig. 10), which is the Update Judge after the procedure Multicast Overlay. The other procedures stay the same.

BR Thread-Update Judge: The Update Judge procedure can judge whether the BR.BitString in the Status Tuple should be updated. If the BR.BitString needs updating, it means that there should exist some ACRs need to be appended or removed. On this occasion, the BR Thread modifies the BR.BitString in the Status Tuple accordingly. If the BR.BitString does not need the modification, the BR.Thread ends. And then the SCR starts the transmission processing.

The packets after the SCR processing should be inserted with the Verification Mark and the BR.BitString. The Verification Mark can ensure the ACRs to response Status Tuple operations. And the BR.BitString can help to replicate and to forward the packets to all the ACRs, including the newly appended.

The TR for member management work only for transmitting the data to the ACRs, which is the same as Section 5.2.3. Thus in this section, we omit the introduction of TRs.

5.3.3. *ACR*

Modifications to the Status Tuple in the ACRs are the implementation of member management. For the members joining, the members can join the group by applying or be assigned by the sender. The ACRs should add the additional addresses into the Status Tuple. Particularly, if the ACR is newly appended in this session, it should create the Status Tuple similar to Address Notification. For the members leaving, the members can leave the group actively, passively, or accidently. The

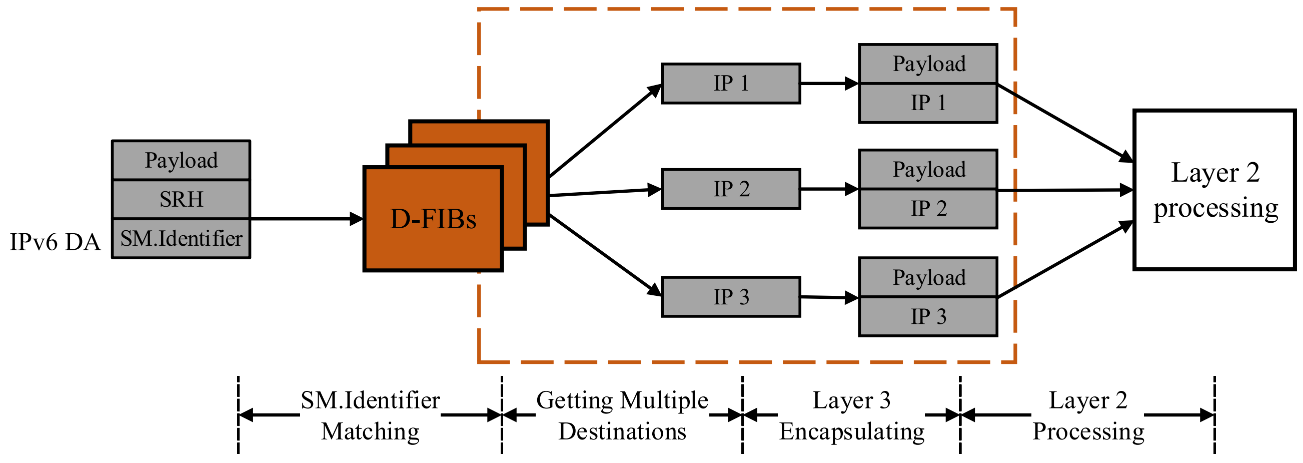


Fig. 12. An instance of the D-FIBs processing: The Input for the D-FIBs is the packet with the SM SIDs. 1) The ACR matches the SM.Identifier to the D-FIBs. 2) It gets the multiple destination IP addresses from the matching operations results. 3) It replicates the payload. 4) It encapsulates the packets with the addresses and the replicated payload (without the SRH). 5) It passes these packets to the layer 2 processing.

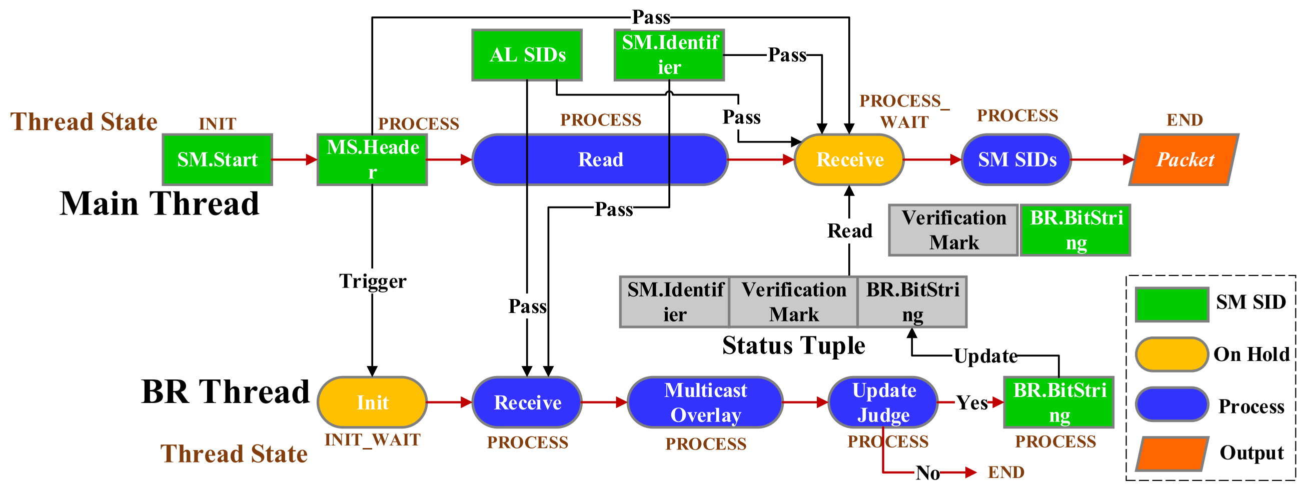


Fig. 13. The SCR process in member management.

sender should modify the Status Tuples in the ACRs for removing the related status. Besides, if one ACR no longer contains any receivers, it should take measures to inform the SCR in order to leave itself out of the session.

The ACRs includes the original ACR and the appended ACR in member management. When the operation is being processed, if the ACR contains the session-related Status Tuple, it's an original ACR. Otherwise, it's an appended ACR that needs to create the Status Tuple. The original ACR and appended ACR have different processes. For member remove operation, all the related ACRs are the original ACRs.

Fig. 14 demonstrates the process in the original ACR. The Main Thread takes charge of the SID parsing and legality verification. The Address Thread works for updating of the Status Tuple and the D-FIBs. The all-zero BR.BitString after the transmission processing can trigger the Main Thread. The parsing of the SM.Start is the first procedure with the state INIT. The Main Thread jumps to the MS.Header procedure immediately.

Main Thread-MS.Header: The MS.Header procedure firstly read the Verification Mark from the MS.Header. And then it selects the stored Verification Mark in the Status Tuple based on the SM.Identifier. Then it passes the two marks to the next procedure Verify in the form of parameters.

Main Thread-Verify: The Verify procedure can verify the legality of the operating packets. If the two Verification Marks are the same, it starts the Read procedure and boots the Address Thread. If they are not the same, the ACR ignores this operating packet and ends the Main Thread.

Main Thread-Read: The Read procedure can parse the AL SIDs and pass them to the Address Thread. The AL SIDs contain the addresses that need to be operated. Meanwhile, the SM.Identifier and the Operation Type in MS.Header are both passed to the Address Thread either. After the SIDs passing, the Main Thread ends.

The Address Thread is triggered by the Verify procedure in Main Thread. The Address Thread starts at the Init procedure, with the state INIT_WAIT. It contains 5 procedures namely Init, Receive, Gateway Judge, Address, and D-FIBs.

Address Thread-Receive: The Receive procedure can get the AL SIDs, the SM.Identifier, and the Operation Type from the Main Thread. Then it passes the acquired SIDs to the next procedure Gateway Judge.

Address Thread-Gateway Judge: Because the operating packet can be transmitted to all the related ACRs, thus the ACR needs to judge whether the addresses belong to itself. All the belonging addresses will be passed to the next procedure, while the others are omitted.

Address Thread-Address: The Address procedure can update the Status Tuple with the specified operation. The SM.Identifier can help to index the Status Tuple. And the Operation Type can specify the operation, adding or deleting the address content.

Address Thread-D-FIBs: Once the Status Tuple has been updated, the corresponding D-FIBs will be updated accordingly, too. The Address Thread can create or delete the D-FIBs based on the addresses. After the D-FIBs updating, the member management operation accomplishes.

As shown in Fig. 14, when the MS.Header utilizes the SM.Identifier as the index in the Status Tuples, the ACR would not find any matched Status Tuple if the ACR is an appended ACR. Thus, for these ACRs they should adopt the different processes for the member management.

The process shown in Fig. 11 is leveraged for the appended ACR. If the SM.Identifier cannot match any Status Tuple, the ACR executes the same process as the Address Notification. It creates the Status Tuple and saves the SM.Identifier, the Verification Mark, and the addresses into it. However, although the appended ACRs in member management have the same process to the Address Notification, they have different acknowledgement procedures, which is explained in Section 5.4.2.

5.3.4. *Leaving detection*

If the sender leaves the session actively, it should send a message to the sender to inform the leaving through unicast. The message contains the SIDs the SM.Identifier, the SM.Start, and the MS.Header. The SM.Identifier can be gotten from the sender when joining. And the MS.Header indicates the operation Active Leaving, with the Operation SEQ 0x0f as shown in Table 2. When the sender receives the message, it can remove the member from the group.

The receivers may leave the group by accident. Thus, in order to reduce the waste of resources, the source-multicast must detect the accidental leaving.

If one receiver leaves the group by accident, the sender would not receive the leaving message. Thus the sender should take measures to judge the members' activity. Because different session types may have different management plans, thus we transfer the right of judging the accidently leaving to the sender. The applications should send messages to all the members periodically for inquiring. If the receiver is active, it should feedback the inquiring message to declare the activity. The sender can determine the policy for judging the activity based on the feedback messages. If the sender confirms that one certain receiver has left, it can remove this receiver from the session.

5.3.5. *ACR removal*

For one ACR, if all the belonging receivers have left from the session, it should inform the sender to remove it. The ACR sends the leaving message to the SCR through unicast. It contains the SIDs SM.Identifier, the SM.Start, and the MS.Header. The MS.Header contains the Verification Mark from the Status Tuple, as well as the Operation SEQ 0xf0 as defined in Table 2.

When the SCR receives an ACR leaving message, it should recalculate the BR.BitString and update its Status Tuple. Specifically, the SCR will feedback an acknowledgement message to the ACR. When the ACR receives the acknowledgement, it can destruct the Status Tuple and delete the related content. The acknowledgement processing is explained in Section 5.4.3.

5.4. *Operation acknowledgement*

The source-multicast has the Packet-Control Mode. The Address Notification, the member management, the member leaving as well as the ACR removal are all treated as the operations. They have similar operating packet formats but different target and acknowledgement mechanisms. The SCR is the initiator of the operations while the ACRs are the final executions. Thus the acknowledgement messages are mainly processed in the SCR.

5.4.1. *Address Notification*

The Address Notification is a special member management operation. The SCR notifies all the related ACRs to create the Status Tuple for the session. The acknowledgement of the Address Notification is handled by the SCR too.

When an ACR accomplishes the Status Tuple creation, it generates a packet destined to the SCR for acknowledgement. The packet contains SIDs the SM.Start, the MS.Header, the SM.Identifier as well as the BFR-id of this ACR (in the form of BR.BitString), in the order of parsing. The acknowledgement packets are transmitted through unicast. The SM.Identifier is set to corresponding the session. The Operation Type in the MS.Header is set to Acknowledgement. The Verification Mark can be read from the Status Tuple. And the Operation SEQ is set to equal to the operation's, that is 0xff for Address Notification.

When the SCR receives the acknowledgement packet from a certain ACR, the SCR can learn the ACR's BFR-id from the BR.BitString. The Verification Mark can be utilized to verify the legality. Fig. 15 illustrates the flow diagram of the acknowledgement processing in SCR. When receives the first acknowledgement message for the Address Notification, the SCR will create a temporary BitString (Temp-BitString) in the Status Tuple to record the messages. The Temp-BitString is initiated to all-zero. When receives one acknowledgement message, the Temp-BitString is updated to the OR result of current Temp-BitString and the ACR's BFR-id. After each updating, the SCR should compare the Temp-BitString to the BR.BitString. If they are the same, which means all the ACRs have accomplishes the Status Tuple creation, the SCR can send an acknowledgement message to the sender to inform the Address Notification accomplishment. The acknowledgement message contains the SID SM.Start, the SM.Identifier, and the MS.Header without the Verification Mark. When the sender receives this message, it can start the Content Transmission.

5.4.2. *Members join & leave*

The members can join or leave the group in different ways. However, regardless of how members requests are created, the sender is the initiator of the corresponding operations. Thus the sender should take charge of the acknowledgement messages of typical member management. When one ACR accomplishes the Status Tuple modification, it sends an acknowledgement message to the sender through unicast. The message contains the SID SM.Start, MS.Header, SM.Identifier, and the AL SIDs. The Operation Type is the Acknowledgement. The Verification Mark is the value saved in the Status Tuple. The Operation SEQ is set corresponding to the operation. And the AL SIDs contains the addresses which are involved in this ACR.

All the acknowledgement messages are destined to the sender, thus they will definitely encounter the SCR. The process in the SCR mainly aims at erasing the Verification Mark to a useless value. After the processing, the SCR forwards the message to the sender keeping the SIDs unchanged except the Verification Mark.

When the sender receives the acknowledgement message, it can get the accomplished addresses. Because these addresses have been processed by the ACR, thus the sender can stop encapsulating them in the operating packet. The member management operation completes if the sender receives all the addresses acknowledgement message.

5.4.3. *ACR removal*

The ACR removal is a special operation because its initiator and executor are both the ACR. When the ACR removal is triggered, the ACR should send a message to the SCR as introduced in Section 5.3.5. The ACR needs the feedback from the SCR for deleting the Status Tuple.

When the SCR accomplishes the BR.BitString updating, it sends back the acknowledgement message to the corresponding ACR through unicast. The message contains only 3 SM SIDs namely the SM.Start, the MS.Header, and the SM.Identifier. The Operation Type is the Acknowledgement while the Operation SEQ is the 0xf0 which indicates the

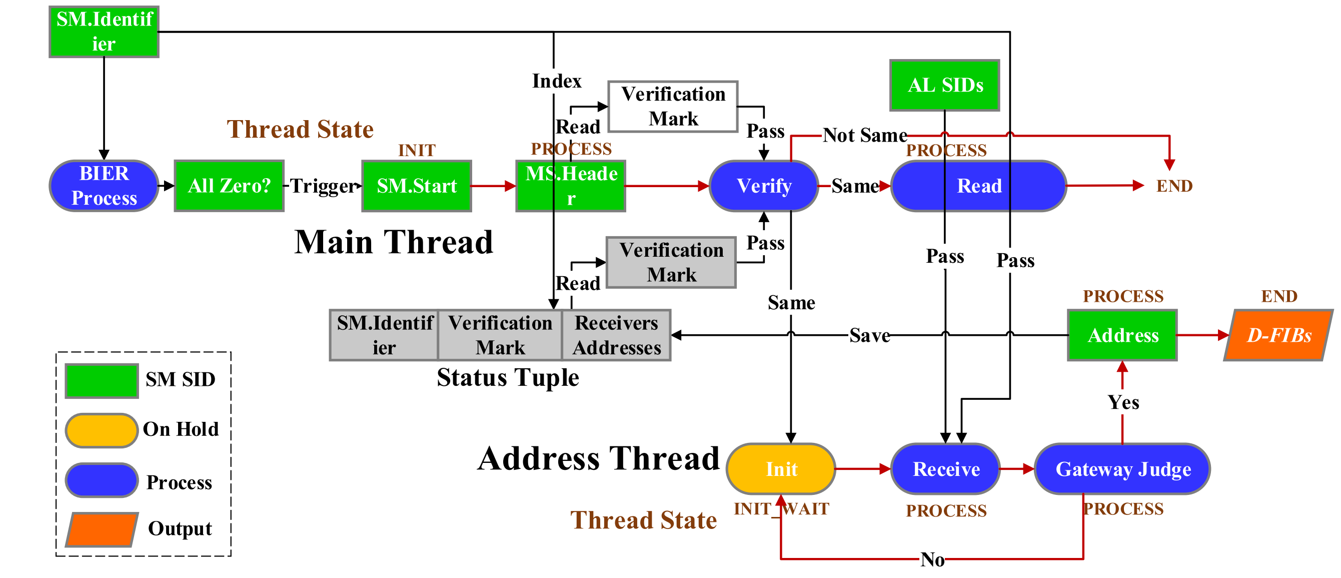


Fig. 14. The original ACR process in member management.

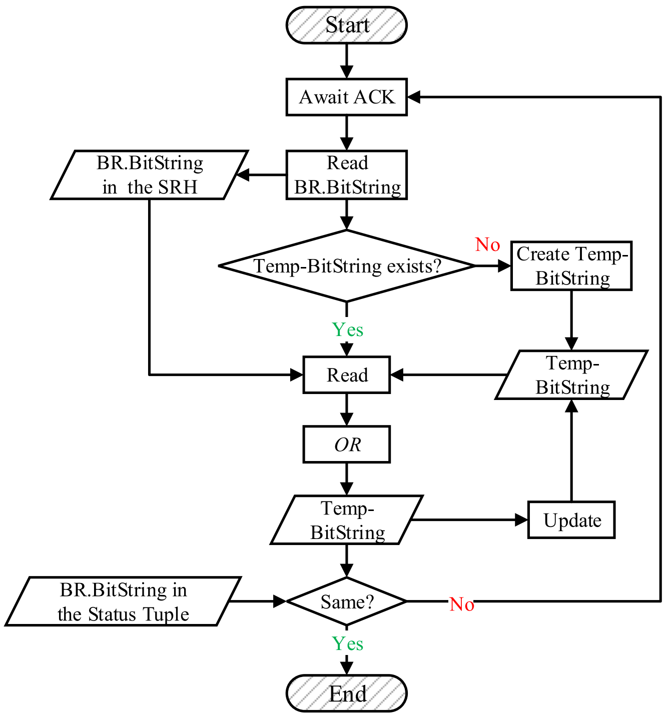


Fig. 15. The flow diagram of the acknowledgement processing in SCR.

ACR Removal. The SM.Identifier can identify the session. The ACR can delete the Status Tuple accordingly once it receives the acknowledgement feedback.

6. Performance evaluation

In this section, we introduce the experiments for performance evaluation of the source-multicast. These experiments include the emulations of the processing latency as well as the simulations of the macro performance. These experiments are all conducted on a DELL R730 server which has two Intel(R) Xeon(R) E5-2609 CPUs, 128 GB RAM as well as 1 TB SSD storage.

These experiments are carried out in 3 aspects.

- **Processing Latency:** The processing latency of source-multicast, including the Address Notification and Member Management in the SCR and the D-FIBs processing.
- **Resource Occupancy:** The occupancy of the cache resource caused by the status to be maintained.
- **Session Initiating Overhead:** The transmission overhead in the routers for session initiating.

Regarding the Processing Latency, because the source-multicast threads process more than the conventional networks. So we conducted some experiments based on P4 (Bosshart et al., 2014) and BMv2 (Consortiumet al., 2018) to evaluate the processing delay of the processing. The P4 is a protocol-independent language allowing the developer to define the network processing. And the BMv2 is the corresponding virtual switch for P4.

For the Resource Occupancy, because of the Status Tuples, the source-multicast calls for more cache resources compared to the typical IGMP. However, the import of the BIER can reduce the cache resources for routing tables. We performed some mathematical simulations on MATLAB for comparing the resource occupancy to the other approaches.

For the Session Initiating Overhead, the source-multicast allows the SIM. The session initiating requires the routers to process more interaction messages. We mathematically simulated the session initiating process on MATLAB for measuring the overhead in the routers, and compared the source-multicast with the other approaches.

6.1. Processing latency

The emulations evaluate the processing latency of the threads which can output the packets, including the Main Threads of the SCR in Address Notification and Member Management, and the D-FIBs processing. Because the processing in ACR only contains the status operation, thus we omit the ACR threads in the emulations.

We implement the SCR threads through P4-16. In this emulation, the processing in BR Thread is taken charge by the control plane. So the program contains the Main Thread only. The program contains 5 tables for source-multicast, and 1 table for latency recording. Table 3 shows the tables as well as their details including the match key, the functions, and the covered procedures. The tables can realize the Main Thread in the SCR functionally, and the BR.BitString is specified by the control plane through the Apache Thrift (2017). The table *replicate* can

Table 3
The tables in the P4 program.

Name	Key	Function	Covered Procedure
generate_vm	NULL	Generate the random Verification Mark and insert it in the MS.Header	MS.Header
save_sid	NULL	Save the SIDs into the P4 Register	Read
insert_bit	NULL	Insert the BR.BitString into the SRH	Input, SM SIDs
replicate	SM.Identifier	Replicate the packet to the specific group	D-FIBs
D_FIBs	Egress Port	Reconstruct the IPv6 Header as a unicast packet according to the Status Tuple	D-FIBs
write_time	NULL	Get the processing time and write it as the source MAC address in the packet	NULL

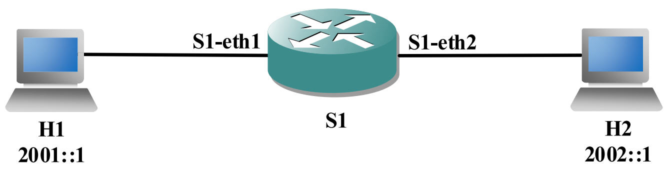


Fig. 16. The topology in this emulation.

help to replicate the packet to all the specific ports, and the *D_FIBs* can reconstruct the IPv6 header to the unicast packet, according to the Status Tuple. Specifically, the Status Tuple is realized by the P4 Register, a stateful element that can store the specific data (P4 16 language specification).

The processing time is measured internally in the BMv2 switch. The *v1model.p4* in the P4-16 utilizes the variable *deq_timedelta* to notate the time spent in the queue. The table *write_time* can read this variable for each packet, and write the value as the source MAC addresses. We capture the packets from the output ports and can also get the processing time by reading the MAC address.

The experiments are embedded in the Ubuntu 16.04 with a kernel 4.14.0, which internally supports the SRv6. The topology is illustrated in Fig. 16, organizes by the Mininet 2.3.0d1 (Kaur et al., 2014) with the BMv2 enhancement. The H1 is the sender while the H2 is the receiver. The H2 exists as a destination, only for H1's successfully sending the SRv6 packets. The emulation contains 3 experiments namely the Main Thread in SCR for Address Notification, the Main Thread in SCR for Member Management, and the D-FIBs processing. We also measure the simple IPv6 DA based matching as the comparisons.

In H1, all the IPv6 packets destined to H2 should be encapsulated with the SIDs shown in Fig. 9. For each experiment, the H1 sends 10000 packets in total, with 0.01s time interval. The S1 processes these packets according to the threads, and output the packets to H2 through S2-eth2. Specifically, in D-FIBs experiment, for each packet the S1 would output 2 packets through S1-eth1 and S1-eth2 respectively, by replicating the payload. And the S1-eth2 is the later output port. We capture the output packets from S1-eth2, and analysis the latency results.

Fig. 17 illustrates the average processing latency of each experiment, and Table 4 shows the relative increase on the basis of IPv6. Because of the performance limitation of BMv2, the results may be higher than real devices. The average latency in IPv6 experiment is 11.2831µs and is set as the base of the comparison. The Address Notification thread has the longest latency 12.2290µs, with the 8.3833% latency increasing. The Management and the D-FIBs both have a close latency to IPv6, which are 11.3631µs and 11.3353µs, with the increasing by 0.7090% and 0.4626% respectively.

The results indicate that the Address Notification is the main cost of the source-multicast. The latency of Address Notification is higher than the Member Management while they have similar procedures. According to the results and the threads, we can infer that the Verification Mark calculation is the main factor affecting the latency performance. The IPv6 experiment can represent the most basic processing in IPv6 data plane. Regarding the time cost, the main cost in source-multicast is the Address Notification in the SCR, with 8.3833% latency increasing. And for the other processing, the latency of source-multicast is less than 1% more than IPv6.

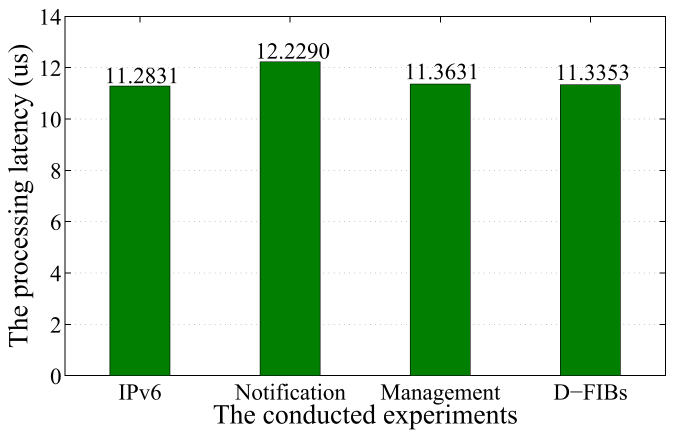


Fig. 17. The average processing latency of each experiment.

Table 4
The relative increase on the basis of Inport

Experiment				
Item	IPv6	Notification	Management	D-FIBs
Latency (μs)	11.2831	12.2290	11.3631	11.3353
Relative Increase		8.3833%	0.7090%	0.4626%

We can conclude that processing latency in the source-multicast is acceptable compared to the typical IPv6 processing. The main latency is caused in Address Notification while it only exists when starting the session. During the session, the processing latency in source-multicast is close to the IPv6, whose difference is less than 1%. So on the whole, we can treat the latency performance in source-multicast at the same level to the IPv6.

6.2. Cache resource for status

As mentioned in Section 3, the different multicast approaches have different mechanisms in transmission and member management. Both of the transmission and the member management need the cache resources. The transmission requires the tables stored in the cache, while the member management entity needs to maintain the members status.

The simulation program can generate the topologies randomly, with the parameter nodes number. Then it selects a random sub-tree from the topology as the multicast tree. Based on this tree, the program can simulate different approaches for their cache resources occupancy. We utilize variable C to notate the resource occupancy. The C is dimensionless, which is the counter of the status entry. For instance, in MLD, if there are 10 multicast tables in a router, the C of this router is 10. In particular, the C of the ACR is equal to the number of addresses in the

Table 5
The status in different approaches.

Approach	Transmission Status Factor	Members Status Factor
source-multicast	Possible ACRs	Receivers
Typical Multicast	Sessions	Edge Router
RDCM	Sessions	Edge Router
M/TCP	Sessions	Receivers

Status Tuple in source-multicast.

The simulation results are affected by multiple factors, for example, the network size, the number of the sessions as well as the number of the receivers. We ran the simulation for these factors respectively.

6.2.1. *Factors: network size and session*

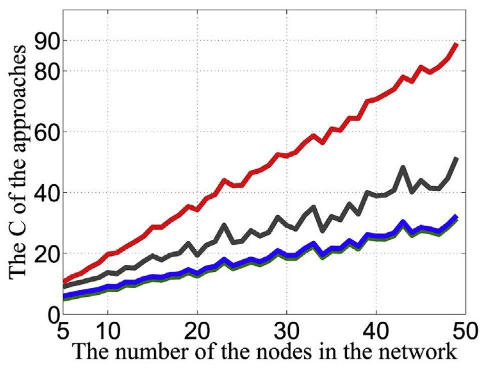
We utilize the number of the nodes in the network to value the network size. We select the typical multicast, RDCM, and the M/TCP as the comparison, which are enumerated in Table 5. We test the C of each approach under different network size and the number of the sessions.

The experiment can generate topologies randomly. Based on the each topology, the program can select the edge nodes. The edge nodes mean that they have the possibility of having receivers. And then from the edge nodes, the program selects the ACRs which are the representative of the session. For each ACR, we set that there only exists 1 receiver in this experiment. Based on these conditions, the programme can simulate the approaches and get the results. We get 1000 pieces of data for each approach under the conditions of a different number of sessions, and then calculate the average C in different network size.

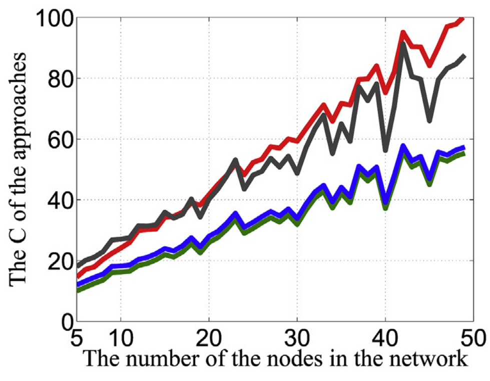
Fig. 18 illustrates the simulation results for the network size and the sessions. Because of the randomness, the curves have multiple peaks. And Table 6 shows the average Relative-C (R–C) from the source-multicast perspective. The R–C is defined in Equation (1). The positive value means that the source-multicast costs more while the negative value means that the source-multicast performs better.

$$Relative - C = \frac{C_{source-multicast} - C_{comparison}}{C_{comparison}} \tag{1}$$

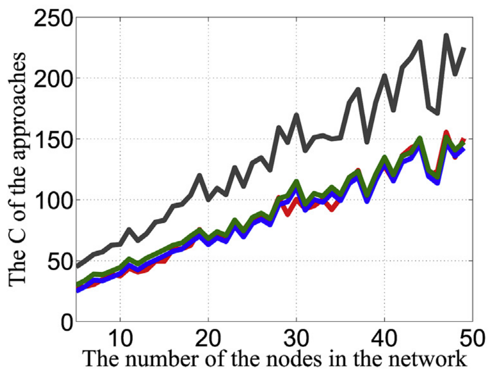
If there exists only 1 multicast session, the source-multicast calls for most overhead for the status maintenance. On average, the R–C of source-multicast is respectively 176.70%, 161.58%, and 74.19% larger than the other approaches. But as the number of the sessions grows, the R–C of source-multicast becomes smaller. When the sessions are more than 5, the source-multicast calls for the least C compared to others, which can also be inferred from Fig. 18(b), (c), and 18(d). The R–Cs are respectively −18.38%, −23.08%, and −48.66% when the number of the sessions is 10.



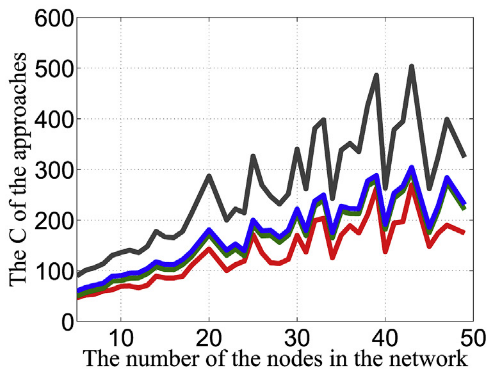
(a) Session Number = 1



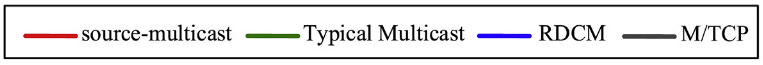
(b) Session Number = 2



(c) Session Number = 5



(d) Session Number = 10



(e) The Legend of (a), (b), (c), and (d)

Fig. 18. The C in different network size and sessions.

Table 6
The Average Relative-C of source-multicast.

Comparison			
R–C			
Sessions	Typical Multicast	RDCM	M/TCP
1	176.70%	161.58%	74.19%
2	70.14%	60.49%	8.53%
5	1.12%	−4.48%	−35.59%
10	−18.38%	−23.08%	−48.66%

We can conclude from this experiment that the source-multicast is more suitable for the scenarios which contain plentiful sessions. In this experiment, 5 can be treated as a critical value. The source-multicast is more resource-saving than others if the number of the sessions exceeds 5.

6.2.2. *Factors: receiver and session*

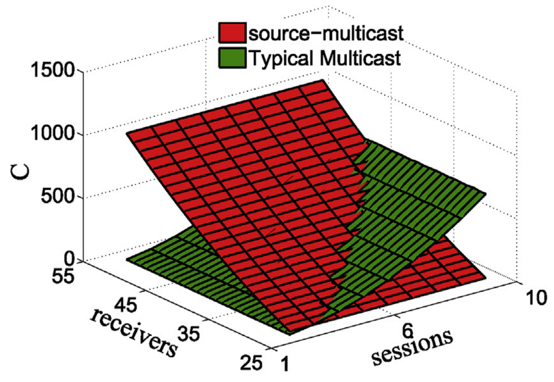
The C of the source-multicast is receiver-related because it utilizes the unicast at the end of the network. In the previous experiment, we set each ACR contains only 1 receiver. In this experiment, we evaluate the C under the influence of the number of sessions and the receivers.

The experiment contains 2 variables, namely the number of the sessions and the number of the receivers in total. The topologies are generated randomly that fixedly contains 25 nodes. Based on each topology, the experiment can evaluate the C of each approach in the case of different sessions and receivers. We conduct the experiment 100 times, and calculate the average C on each condition. We get 97200 pieces of data in total and demonstrate the result in Fig. 19.

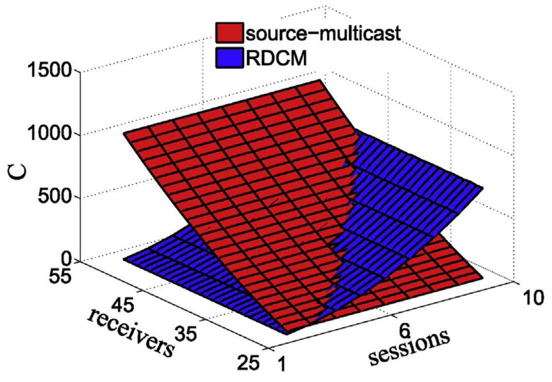
Fig. 19(a), (b), and 19(c) RDCM, and the M/TCP respectively. These 3 subfigures have a similar trend. With the number of the sessions increasing, the C of the source-multicast stays the same while the others' become larger. On the other hand, the source-multicast is receiver-related, thus if there are a huge number of receivers, the source-multicast requires more cache resources compared to the others.

We project the intersection lines in Fig. 19(a), (b), and 19(c) to the receivers-session plane, as illustrated in Fig. 19(d). We define these projections as Performance Boundary Lines (PBL). The PBL can indicate the conditions that the corresponding approach would be more cost-saving. In Fig. 19(d), the red-shadowed area which is above the 3 PBLs is the cost-saving area of the source-multicast. If the condition, including the number of receivers and sessions, locates inside this area, the source-multicast requires less cost than the others.

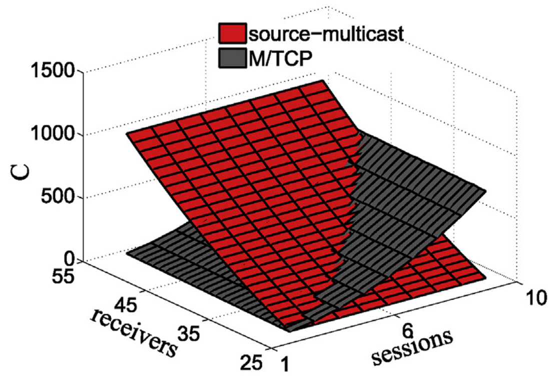
The cost-saving area of the source-multicast is restricted by the receivers. The number of the sessions does not affect the source-multicast but can produce lots of overhead as far as the other approaches. From the simulations, we can conclude that source-multicast is suitable for the scenarios having bounded-scale receivers and unlimited sessions. The bounded-scale receivers mean that the number of the receivers has the upper bound, such as the intra-area scenarios including the campus and the residence.



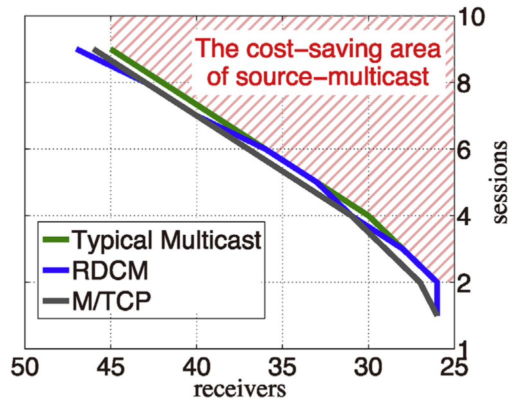
(a) Typical Multicast



(b) RDCM



(c) M/TCP



(d) The PBLs and the cost-saving area

Fig. 19. The C in different sessions and receivers.

Table 7
The conditions of the experiment.

Approach	Start	End
source-multicast	Address Notification	ACK accomplishes
MLD with PIM-DM	No status in the network	All the receivers have joined in the group
MLD with PIM-SM	No status in the network	All the receivers have joined in the group
M/TCP	TCP 3-way handshake	Receivers get the group information
MCTCP	TCP 3-way handshake	The MST (Multicast Spanning Tree) has been built

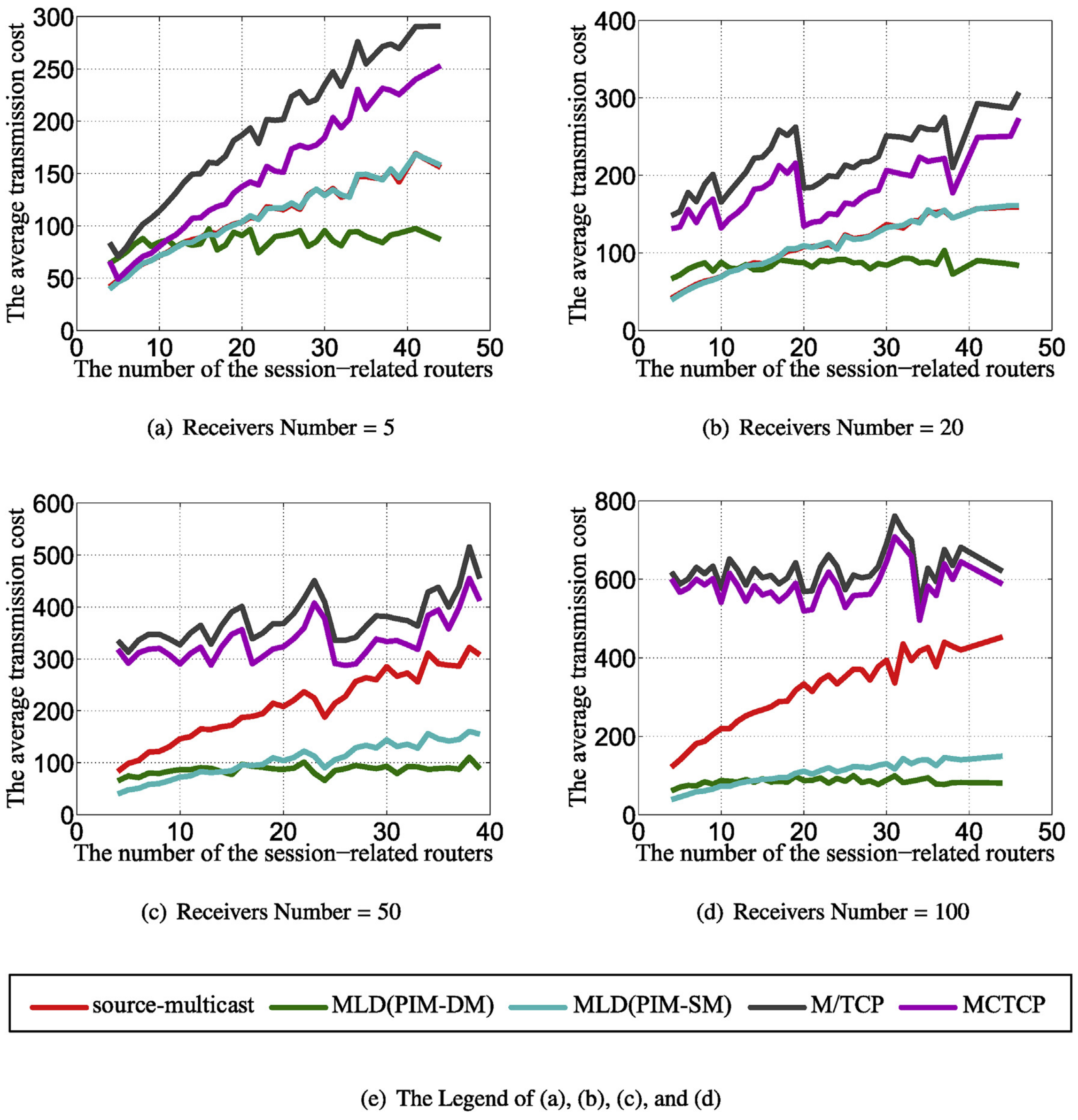


Fig. 20. The transmission cost in different multicast size and receivers.

6.3. *Processing overhead for session initiating*

In source-multicast, the session initiating requires some interaction messages, which can produce some overhead. In this simulation, we conduct some experiments to measure the cost of transmitting the interaction messages for session initiating.

In this simulation, the topologies are generated as the weighted graphs, in which the links have the transmission cost. The cost is set randomly to a maximum of 10 for each link. And the topology contains at most 50 nodes. We select some approaches as comparisons. Table 7 illustrates these approaches as well as their Start and End conditions. The MLD with PIM-DM (Adams et al., 2004) and PIM-SM (Fenner et al., 2006) are both belonging to the typical multicast. We calculate the cost for only 1 time during the 3-way handshake for M/TCP and MCTCP. Because the other handshaking is not relevant to the session initiating. The routing in M/TCP and MCTCP is set to the smaller value of PIM-SM and PIM-DM, thus the result curves of them may contain the sharp points.

We select the multicast-related routers and the number of the receivers as the factors. This simulation contains 4 experiments with

Table 8
The Average Relative-Cost of source-multicast.

Comparison				
R-Cost				
Receivers	MLD PIM-DM	MLD PIM-SM	M/TCP	MCTCP
5	24.36%	0.06%	−43.21%	−27.14%
20	26.97%	0.26%	−51.22%	−40.36%
50	143.51%	100.52%	−43.97%	−37.01%
100	273.09%	201.11%	−49.72%	−46.29%

thenumber of the receivers 5, 10, 50, and 100 respectively. In each experiment, the program generates 1000 topologies and records the transmission cost for each approach. Then we calculate the average cost under the different numbers of multicast-related routers in each experiment.

The results are demonstrated in Fig. 20 and Table 8. We utilize the average Relative-Cost (R-Cost) to value the results, as defined in Equation (2).

$$Relative-Cost=\frac{Transmission\_Cost_{source-multicast}-Transmission\_Cost_{comparison}}{Transmission\_Cost_{comparison}} \quad (2)$$

In Fig. 20, the abscissa of these sub-figures is the number of session-related routers while the ordinate is the average transmission cost for transmission under the same conditions. Fig. 20(a), (b), (c), and (d) show the results of the number of the receivers in 5, 20, 50, and 100 respectively.

The M/TCP and the MCTCP produce more cost than the others because they are both TCP-based which have a unicast-based handshake. But on the other hand, the cost can provide reliability in transmission.

The source-multicast has the close cost to the MLD with PIM-SM because the Address Notification is spanning-like. The R-Costs of for MLD with PIM-SM are 0.06% and 0.26%, corresponding to the number of the receivers 5 and 20, as illustrated in Fig. 20(a) and (b). The source-multicast is receiver-based, thus with the number of receivers increasing, the source-multicast produces more transmission cost, as shown in Fig. 20(c) and (d). If the number of the receivers is larger than 50, the R-Cost of the MLD (include both PIM-DM and PIM-SM) is larger 100%. If the receivers scale is small, the source-multicast has a very close cost compared to the MLD. But it can bring some new features to the network and user, as introduced in Section 1. The selection of these approaches needs further trade-off according to the demands of the sessions.

These conducted experiments measure the applicable conditions of the source-multicast. It's suitable for the scenarios which have a large number of sessions, the bounded number of receivers, as well as the small multicast tree. It has the potential to be utilized in some intra-area networks to allow the users to deploy the multicast sessions by themselves. Such as the cloud, the campus, the industry as well as the residence.

The emulations in Section 17 measures the processing latency in source-multicast regarding the processing in SCR and the D-FIBs. The results indicate that the source-multicast has the same level of latency performance as the typical IPv6, in which the main processing latency cost is in Address Notification, increasing about 8%. The simulations in Section 6.2 simulate the cache occupation caused by the maintained status. The results show the cost-saving conditions of source-multicast concerning the transmission, namely the large number of sessions and the small number of receivers. The simulation in Section 6.3 shows the overhead in session initiating. The results are consistent with the conclusion in Section 6.2, which is the source-multicast is suitable for the scenarios with vast of sessions.

7. Conclusion

In this paper, we propose the source-multicast based on the SRv6 data plane. We firstly design the SM SIDs for source-multicast. Then we design the principle and the processes in the different entities for source-multicast. The source-multicast allows the sender to initiate and operate the session. We conduct some experiments to evaluate the performance, and make comparisons. The results indicate that the latency in source-multicast is at the same level as the typical IPv6. The results also illustrate the conditions under which the source-multicast is more cost-saving, which are the large number of sessions and the small number of receivers.

Compared to the conventional multicast, the source-multicast can introduce some new features into the network such as the initiative, the sender-based operation as well as the privacy, at the small expense of processing latency. And the BIER-based transmission solution can optimize the resource occupation in the network. Regarding the SRv6 multicast, according to (Zhang et al., 2018), the configuration or instantiation could bring about vast overhead if the sessions are massive. Thus the SRv6 multicast has the limitation in the massive sessions.

The performance evaluation for source-multicast indicates the potential real applications. For instance, according to TR 22.821 (3GPP and Feasibility Study on LAN Support in 5G, 2018), the 5G LAN should support the "use case on creating and joining private multicast communication". In each 5G LAN group, the number of receivers is often bounded. The 5G LAN may also contain the massive sessions due to that the users have the right to create the multicast communications. The source-multicast also suits the enterprise network in which containing multiple intra-group file sharing. In addition, the initiative scenarios are also the potential applications of source-multicast, like the multi-user online video, requiring the initiative calling and the receivers indication.

Meanwhile, there are still some works remain for this scheme. The processing in source-multicast is stateful, so most of the potential applications are internal scenarios. We are considering enhancing the source-multicast with the stateless design, to make it fit the large scenarios like the backbone. The backbone is the typical use case of SRv6, thus expanding the stateless mechanism can facilitate the utilization of source-multicast in the backbone.

Acknowledgements

The work described in this paper is supported by National Key R&D Program of China (Project Number: 2018YFB1800501), National Natural Science Foundation of China (Project Number: 61671086) Consulting Project of Chinese Academy of Engineering (Project Number: 2016-XY-09), and Research on the Development Strategy of Future Network Operating System (Project Number: 2019-XY-5).

Appendix A. The Detailed Procedure of the SM SIDs

Some abbreviations are illustrated in the following. The NH stands for the next-header field in the IPv6 header. The SL means the segment left, which indicate the currently active segment in SRH. N is utilized to represent an SRv6 node and S stands for the destination address. The DA means the destination address. The pseudo code style refers to the SRv6 programming draft (Filsfils et al., 2019).

Appendix A.1. End

When N receives a packet whose IPv6 DA is S and S is a local End, N does:

```
1:  if NH=SRH and SL > 0
2:    if SRH[SL] is an SM.Start SID
3:      read SRH[SL-1]
4:      if SRH[SL-1] is an MS.Header
5:        boot the SCR process
6:      else
7:        drop the packet
8:      end if
9:    else
10:     decrement SL
11:     update the IPv6 DA with SRH[SL]
12:     FIB lookup on the updated DA
13:     forward accordingly to the matched entry
14:   end if
15: else
16:   drop the packet
17: end if
```

Appendix A.2. SM.Identifier

When N receive a packet whose IPv6 DA is S and S is a local SM.Identifier, N does:

```
1:: if SRH[SL] is an SM.Start SID then
2:    read SRH[SL-1]
3:    if SRH[SL-1] is an BR.BitString SID then
4:      match the BIER FIBs with BR.BitString
5:      if BR.BitString is all-zero after processing then
6:        boot the ACR process
7:      else
8:        replicate and forward accordingly to the matched entry
9:      end if
10:   else
11:     drop the packet
12:   end if
13: else
14:   drop the packet
15: end if
```

Appendix B. Supplementary data

Supplementary data to this article can be found online at https://doi.org/10.1016/j.jnca.2019.102505.

References

3GPP, Feasibility Study on LAN Support in 5G, 06 2018. Technical Report (TR) 22.821, 3rd Generation Partnership Project (3GPP), Version 14.2.2. https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId3281.

Adams, A., Nicholas, J., Siadak, W., 2004. Protocol Independent Multicast-Dense Mode (Pim-dm): Protocol Specification (Revised). Tech. rep..

D. Borthakur, etal., Hdfs Architecture Guide, Hadoop Apache Project 53.

Bosshart, P., Daly, D., Gibb, G., Izzard, M., McKeown, N., Rexford, J., Schlesinger, C., Talayco, D., Vahdat, A., Varghese, G., et al., 2014. P4: programming protocol-independent packet processors. Comput. Commun. Rev. 44 (3), 87–95.

Clark, D.D., Lambert, M.L., Zhang, L., 1987. Netblt: a high throughput transport protocol. In: ACM SIGCOMM Computer Communication Review, vol. 17. ACM, pp. 353–359.

Consortium, P.L., et al., 2018. Behavioral Model (Bmv2).

Deering, S., Fenner, W., Haberman, B., 1999. Multicast listener discovery (mld) for ipv6. Tech. rep..

S. Deering, D. Estrin, D. Farinacci, V. Jacobson, C.-g. Liu, L. Wei, Protocol Independent Multicast (Pim): Protocol Specification, Internet Draft RFC.

W. Fenner, Rfc2236: Internet Group Management Protocol, November.

Fenner, B., Handley, M., Holbrook, H., Kouvelas, I., 2006. Protocol Independent Multicast-Sparse Mode (Pim-sm): Protocol Specification (Revised). Tech. rep..

Filsfils, C., Nainar, N.K., Pignataro, C., Cardona, J.C., Francois, P., 2015. The segment routing architecture. In: Global Communications Conference (GLOBECOM), 2015 IEEE. IEEE, pp. 1–6.

Filsfils, C., Garvia, P.C., Leddy, J., Voyer, D., Matsushima, S., Li, Z., Feb. 2019. SRv6 Network Programming, Internet-Draft Draft-Filsfils-Spring-Srv6-Network-Programming-07, Internet Engineering Task Force, Work in Progress. https://datatracker.ietf.org/doc/html/draft-filsfils-spring-srv6-network-programming-07.

Filsfils, C., Parekh, R., Bidgoli, H., Zhang, Z.J., Jul. 2019. SR replication policy for P2MP service delivery, internet-draft draft-voyer-spring-sr-p2mp-policy-03. Internet Engineering Task Force, work in Progress. https://datatracker.ietf.org/doc/html/draft-voyer-spring-sr-p2mp-policy-03.

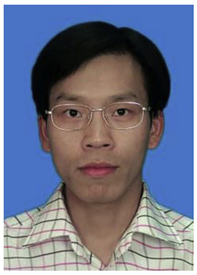
C. Filsfils, J. Leddy, D. Voyer, D. Bernier, D. Steinberg, R. Raszuk, S. Matsushima, D. Lebrun, B. Decraene, B. Peirens, etal., Srv6 Network Programming, Internet-Draft.

Floyd, S., Jacobson, V., McCanne, S., Liu, C.-G., Zhang, L., 1995. A reliable multicast framework for light-weight sessions and application level framing. Comput. Commun. Rev. 25 (4), 342–356.

K. Jeacle, J. Crowcroft, Reliable Ligh-Speed Grid Data Delivery Using Ip Multicast.
Jeacle, K., Crowcroft, J., 2005. Tcp-xm: unicast-enabled reliable multicast. In: Computer Communications and Networks, 2005. ICCCN 2005. Proceedings. 14th International Conference on. IEEE, pp. 145–150.
Jeacle, K., Crowcroft, J., Barcellos, M.P., Pettini, S., 2005. Hybrid reliable multicast with tcp-xm. In: Proceedings of the 2005 ACM Conference on Emerging Network Experiment and Technology. ACM, pp. 177–187.
Kaur, K., Singh, J., Ghumman, N.S., 2014. Mininet as software defined networking testing platform. In: International Conference on Communication. Computing & Systems (ICCCS), pp. 139–142.
J. Leddy, S. Matsushima, D. Voyer, Ipv6 Segment Routing Header (srh).
Lehman, L.-W.H., Garland, S.J., Tennenhouse, D.L., 1998. Active reliable multicast. In: INFOCOM98. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, vol. 2. IEEE, pp. 581–589.
Li, D., Xu, M., Liu, Y., Xie, X., Cui, Y., Wang, J., Chen, G., 2014. Reliable multicast in data center networks. IEEE Trans. Comput. 63 (8), 2011–2024.
Lin, J.C., Paul, S., 1996. Rmtp: a reliable multicast transport protocol. In: INFOCOM96. Fifteenth Annual Joint Conference of the IEEE Computer Societies. Networking the Next Generation. Proceedings IEEE, vol. 3. IEEE, pp. 1414–1424.
McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., Turner, J., 2008. Openflow: enabling innovation in campus networks. Comput. Commun. Rev. 38 (2), 69–74.
J. Moy, Mospf: Analysis and Experience.
P4 16 language specification. URL https://p4.org/p4-spec/docs/P4-16-v1.0.0-spec.html.
Ritchie, O., Thompson, K., 1978. The unix time-sharing system. The Bell System Technical Journal 57 (6), 1905–1929.
Rosen, E.E., et al., 2018. Multicast Vpn Using Bier, Draft-Ietf-Bier-Mvpn-11, Internet Engineering Task Force. Internet-Draft, pp. 1–17.
Talpade, R., Ammar, M.H., 1995. Single connection emulation (sce): an architecture for providing a reliable multicast transport service. In: Distributed Computing Systems, 1995., Proceedings of the 15th International Conference on. IEEE, pp. 144–151.
Thrift, A., 2017. Apache Thrift.
Visoottiviseth, V., Mogami, T., Demizu, N., Kadobayashi, Y., Yamaguchi, S., 2001. M/tcp: the multicast-extension to transmission control protocol. In: Proceedings of ICACT, Citeseer.
I. Wijnands, G. J. Shepherd, C. J. Martin, Bit indexed explicit replication, uS Patent App. 15/827,084 (Mar. 22 2018).
Zhang, Z.J., Wijnands, I., Dolganow, A., Geng, L., Aug. 2018. Multicast in SR Networks, Internet-Draft Draft-Zzhang-Pim-Sr-Multicast-00, Internet Engineering Task Force, Work in Progress.
X. Zhang, M. Yang, L. Wang, M. Sun, An openflow-enabled elastic loss recovery solution for reliable multicast, IEEE Systems Journal.
Zhu, S., Bi, J., Sun, C., Wu, C., Hu, H., 2015. Sdpa: enhancing stateful forwarding for software-defined networking. In: Network Protocols (ICNP), 2015 IEEE 23rd International Conference on, IEEE, pp. 323–333.
Zhu, T., Wang, F., Hua, Y., Feng, D., Wan, Y., Shi, Q., Xie, Y., 2016. Mctcp: congestion-aware and robust multicast tcp in software-defined networks. In: Quality of Service (IWQoS), 2016 IEEE/ACM 24th International Symposium on. IEEE, pp. 1–10.
Zhu, T., Feng, D., Wang, F., Hua, Y., Shi, Q., Xie, Y., Wan, Y., 2017. A congestion-aware and robust multicast protocol in sdn-based data center networks. J. Netw. Comput. Appl. 95, 105–117.

Weihong Wu, received his B.S. degree in communication engineering from Beijing University of Posts and Telecommunications (BUPT), Beijing, China, in 2016. He is currently working towards his Ph.D. degree in network engineering in State Key Laboratory of Network and Switching Technology, BUPT. His research interests include the novel multicast protocol and the deterministic networking.

Jiang Liu, received his B.S. degree in electronics engineering from Beijing Institute of Technology, Beijing, China, in 2005, the M.S. degree in communication and information system from Zhengzhou University, Zhengzhou, China, in 2009, the Ph.D. degree from Beijing University of Posts and Telecommunications, Beijing, China, in 2012. He is currently an associate professor in Beijing University of Posts and Telecommunications. His current research interests include network architecture, network virtualization, software defined networking (SDN), information centric networking (ICN), and tools and platforms for networking research and teaching.



Huang Tao, received his B.S. degree in communication engineering from Nankai University, Tianjin, China, in 2002, the M.S. and Ph.D. degree in communication and information system from Beijing University of Posts and Telecommunications, Beijing, China, in 2004 and 2007 respectively. He is currently the professor in Beijing University of Posts and Telecommunications. His current research interests include network architecture, Routing and Forwarding, network virtualization.