

A Mechanism for MPLS Broadcast and Its Extension for Multicast Dense-Mode Support in MPLS ¹

Siavash Samadian-Barzoki[†], Mozafar Bag Mohammadi[†], and Nasser Yazdani[‡]

Router Laboratories, University of Tehran

[†]{s.samadian, mozafarb}@ece.ut.ac.ir

[‡]nasyaz@sofe.ece.ut.ac.ir

Abstract. Due to the increasing multicast applications and high desire for its deployment, it seems that any new technology should support multicast. However, MPLS technology which enhances IP packet forwarding capability by using layer 2 switching still does not offer any special solution for multicast support. The main difficulty in supporting dense-mode multicast protocols like DVMRP and PIM-DM is lack of a broadcast mechanism in MPLS. We propose a genuine broadcast mechanism for MPLS in this paper. First, the proposed mechanism is based on a central broadcast label assigner called BLAC (Broadcast Label Assignment Center). Then, we propose a new dense-mode multicast protocol for MPLS as an extension to our broadcast mechanism. Our method consumes fewer labels compared to the existing proposals supporting dense-mode multicast groups and has smaller forwarding tables.

1 Introduction

Multicast capability is a very important feature in existing data networks and supporting it considered as a highly valued services being offered by some Internet Service Providers (ISPs) [3]. There are many multicast applications such as audio and video conferences (like IETF sessions in the Mbone) [4][21], distance education and remote project meetings [12]. In many circumstances, an application must send the same information to many destinations. Using multicast transmission instead of unicast in these cases reduces the transmission overhead on the sender, on the network and saves the time taken for all destinations to receive the data [1].

MPLS technology which enhances IP packet forwarding capability by using layer 2 switching is being standardized in IETF [11] and manifests performance benefits in the backbone of Internet [13]. MPLS tries to tackle the problem of “how to integrate the best attributes of the traditional layer 2 and layer 3 technologies”. There are many unsolved problems for supporting IP multicast in a MPLS domain [7][8]. Multicast support is not defined in MPLS architecture and it is left for further study [11]. Any efficient solution for supporting multicast in MPLS will accelerate MPLS deployment in the future network.

We introduce a solution to the broadcast problem in MPLS network for the first time in this paper. Our solution is based on the RPF checking mechanism and multicasting in IP [1][17]. It assigns a label to a source of broadcast packet upon its request. The label is then, the identifier of the source’s broadcast packets and each LSR (Label Switch Router) uses this label to determine correct incoming interface based on RPF check. If the packet was from right interface, it is copied to the other interfaces of LSR. Broadcast label is assigned and released by a central node named BLAC (Broadcast Label Assignment Center). We propose to define a specific value in layer 2 headers (such as ethertype value in Ethernet and IEEE 802.3 or Protocols field in PPP) to indicate the broadcast label space in the MPLS header. If not possible, our second solution is a globally specific reserved label value to sit at top of the label stack while the source identifying label sits in the next level. In this way, the scheme consumes only identifying labels for broadcasting while unicast label space is remained intact by separating the broadcast and unicast label spaces.

Using the broadcasting mechanism, we devise a new dense-mode multicast routing protocol in MPLS environment which is like DVMRP [9]. This protocol consumes fewer labels; only one label from the multicast label space and no label from unicast label space, for a source tree than the existing proposals and has smaller forwarding tables.

¹ This work is partially supported by ITRC (Iranian Telecommunication Research Center).

The rest of paper is organized as follows. In the next section, we briefly introduce multicasting, specifically, DVMRP protocol, MPLS and difficulty of supporting multicast in MPLS. BLAC broadcast mechanism is discussed in section 3. We explain our proposal for MPLS dense-mode multicast routing protocol in section 4. Section 5 discusses some related issues of the proposed mechanisms. Section 6 discusses related work. Finally, we conclude in section 7.

2 Background and Problem Definition

2.1 Multicasting

Basically, multicast data is sent to a predefined user group which consists of members joining and leaving dynamically. Currently, a user group is specified by a class D address. A continuing challenge has been how to route data to the group members efficiently [1][3]. In some circumstances, the group members are densely distributed across the network. In this case, high percentage of nodes is group member. Protocols designed to deal with this situation are called dense-mode protocols. Examples of these protocols are DVMRP [9][3] and PIM-DM [3].

Dense-mode protocols use a data driven approach to construct multicast distribution tree. They send and store explicit prune states in response to unwanted multicast data. DVMRP (Distance Vector Multicast Routing Protocol) [9] is the first protocol introduced for multicast routing efficiently supporting dense mode groups [3]. For each sender S and group G a source specific tree (S, G) is constructed. Root of this tree is S and it is built from all shortest paths between S and the multicast group members. An example of such tree is shown in figure 1.

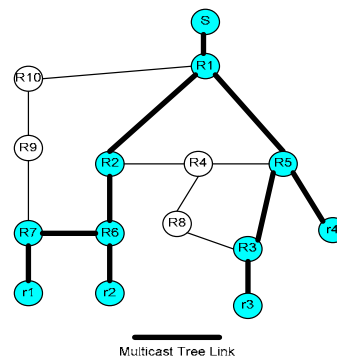


Fig. 1. Source tree

This tree is constructed by the mechanism called flood and prune. Multicast data is flooded in the network according to the RPF (Reverse Path Forwarding) broadcast mechanism [1][17]. Upon receiving this traffic, leaf nodes may transmit prune messages back toward the source if there were no group members on their directly attached hosts. These prune messages cause upstream nodes to remove all branches not leading to group members. This results in a source specific shortest path tree. A timer is also set to disable the pruning effect when expired. In this way, the flooding procedure is repeated periodically and the pruning is done again. This is needed to discover the newly joined members. It is also possible to graft a previously pruned branch by sending a Graft message upstream.

The main challenge in supporting dense mode protocols like DVMRP in a MPLS domain is lack of a broadcast mechanism. This mechanism is needed in the flooding stages of the algorithm. We explain our solution to this problem in the next section.

2.2 MPLS

MPLS tries to encounter the time consuming network layer lookup problem by tagging packets going to the same destination. Therefore, a label is used as an index into the routing table containing the next hop and outgoing label. In [13] efficiency of MPLS in the backbone is assessed. Reported results there show that for a fine designed MPLS node, there is 30% cut-off in the end-to-end delay of an IP packet. Figure 2 illustrates MPLS technology's main concepts.

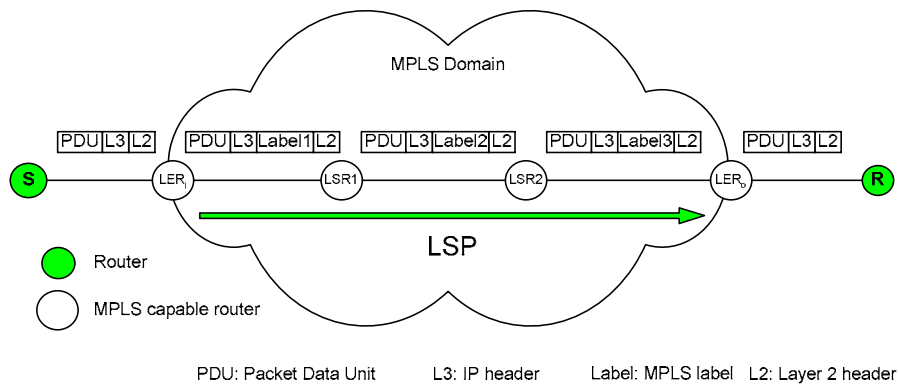


Fig. 2. MPLS concepts

MPLS domain is a portion of the network that consists of devices understanding MPLS. Label edge router (LER) is a device that sits at the edge of an MPLS domain and uses the routing information to assign labels to datagrams and forwards them into a MPLS domain. A LSR is a device that typically resides in the middle of a network and forwards datagrams only based upon labels. The specific path that a datagram travels through a network based upon the labels that are assigned to that datagram is called LSP (Label Switching Path). Forwarding Equivalent Class (FEC) is a set of all packets belonging to the same LSP. In MPLS, the assignment of a particular packet to a particular FEC is done just once at the ingress LER when the packet enters the network. The FEC to which a packet belongs is encoded in a short fixed length "label". In the MPLS forwarding paradigm, once a packet is assigned to a FEC, no further header analysis is done by subsequent LSRs and all forwarding are driven by labels. Labels are used as indexes into a table which specifies next hops, and outgoing labels. Existing labels are replaced with new ones, and packets are forwarded to next LSRs. In the egress LERs, packets are stripped from their MPLS headers and IP packets are forwarded to their next hops. This approach restricts the time consuming layer 3 lookup to MPLS edge routers and allows routers in the domain use exact match to find the output interfaces quickly [11].

2.3 Multicast Support Difficulties in MPLS

1. Multicast traffics has more than one outgoing interface, then, it may be forwarded on L2 in some branches and on L3 in others [7][8]. Therefore, multicast packet forwarding in MPLS domain can be performed in three ways as figure 3 shows. Our method requires mixed L2/L3 forwarding paradigm.

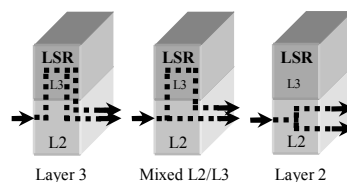


Fig. 3. MPLS forwarding mode

2. Flood & Prune multicast routing protocols have some characteristics which significantly differ from unicast routing protocols:

- Due to the Flood & Prune nature of the protocol, very volatile tree structures are generated. Solutions to map a dynamic L3 p2mp (point-to-multipoint) tree to a L2 p2mp LSP need to be efficient in terms of signaling overhead and LSP setup time. The volatile L2 LSP will consume a lot of labels throughout the network, which is a disadvantage when label space is limited [8]. Both of these problems are efficiently solved in our proposal. Tree generation in our method is native and supported implicitly in broadcast mechanism. Label consumption is minimized by using only one label from multicast label space to identify a multicast tree while leaving unicast label space intact.
- A router only creates state for a certain group when data arrives for that group. Routers also independently decide to remove state when timer expires. Thus, LSPs can not be pre-established as is usually done in unicast. To minimize the time between traffic arrival and LSP establishment, a fast LSP setup method is favorable [8]. In our method, there is no need to setup explicit LSPs for multicast traffic, except obtaining a broadcast label and therefore our method does not suffer from this problem.
- Since creation and deletion of a L3 route at each node is triggered by traffic, this suggests that the LSP associated with the route be setup and torn down in a traffic-driven manner as well. If the mixed L2/L3 forwarding capability is not supported, the traffic driven trigger requires a label distribution mode in which the label is requested by the LSRu (Downstream on Demand or Upstream Unsolicited mode) [8]. Our method is not using any LSP trigger method therefore it has none of these problems.

To get idea about other difficulties, the reader can refer to [7][8].

3 Broadcast Mechanism

For broadcasting data in MPLS, we need a means to identify broadcast packets. This can be done by defining a specific value in layer 2 headers (such as ethertype value in Ethernet and IEEE 802.3 or Protocols field in PPP) to indicate whether the label space in MPLS header is broadcast. If this solution was not possible, a globally specific reserved label value is needed among MPLS labels to sit at top of the packet's label stack. The latter solution, of course, consumes more network bandwidth with 4 bytes extra label. Although we strongly recommend the first solution, the rest of the paper is based on the second solution to achieve an independent method and when the first solution is possible for a special layer 2 technology, the needed conversions should be done appropriately.

In this way, each a LSR can examine the packet label and if it was the broadcast label, it starts broadcasting packet and forwarding it to all interfaces except the incoming one. Of course, this is a simple solution. Unfortunately, this way broadcasting is inefficient and broadcast traffic will congest network rapidly. Better methods are presented for IP broadcast, namely RPF, extended RPF and RPB (Reverse Path Broadcasting) [1][17]. These methods use packets' source addresses to filter extra packets and determine output interfaces for a right packet. We consider a packet as right packet if it has come on an interface used to forward the packet to its source (reverse path) based on the routing table. In MPLS, one can't guess source address of a packet by merely examining MPLS header. Then, the LSR must examine the IP header. To avoid this time consuming process, a one to one binding must be defined between source addresses and assigned label and all LSRs must be aware of this correspondence. Then, source of a packet can be identified from the corresponding assigned label in the header.

In our method, when a source S wants to broadcast some packets into the network, first it claims a broadcast label from BLAC (Broadcast Label Assignment Center). BLAC is an LSR which is responsible for assigning and releasing broadcast labels. Therefore, the source sends a *Broadcast_Label_Request* message toward BLAC. When BLAC node receives these messages it acts as follows:

```

IF ( $\exists$  LabelB  $\in$  Label_RangeB) THEN
  Send a Broadcast_Label_Assignment message(X, LabelB) destined to S
  to all interfaces;
  Label_RangeB  $\leftarrow$  Label_RangeB - {LabelB};
  Insert (IPS, LabelB) in TableAssigned_Labels ;
ELSE
  Discard the Broadcast_Label_Request message;
ENDIF

```

In the BLAC, if a broadcast label ($Label_B$) is available in the broadcast label space ($Label_Range_B$), it broadcasts the following message in the MPLS domain:

	Shim Header	Source Address	Destination Address	Payload
L2 header	X	$Label_B$	IP_S	Broadcast_Label_Assignment

Consequently, all LSRs in MPLS domain know that $Label_B$ is assigned to S. S can use this label to broadcast its data. Label X at top of the stack identifies BLAC broadcasted messages. When this message reaches S, it knows that $Label_B$ can be used for its data broadcasting. Therefore, it first pushes $Label_B$ in label stack of each broadcast packet followed by another label Y. Label Y has special meaning, and is used by all LSRs for broadcasting conventional broadcast packets. Finally, S broadcasts the resulting packets to all of its interfaces. Important fields of these broadcast packets are as follows:

	Shim Header	Source Address	Destination Address	Payload
L2 header	Y	$Label_B$	IP_S	$IP_{Broadcast}$ Broadcast Data

Notice that label Y is needed when we have no identifier in layer 2 header for broadcasting to prevent partitioning the label space in to two parts, one for unicast packets and another for broadcast ones. By using one more level in label stack, we have separated broadcast label space from unicast label space. Therefore, both can use label space independently at the expense of using more bandwidth. This larger packet size (4 bytes) has little effect on the network performance since broadcast event is less frequent. When an LSR receives a packet with a label Y, it knows that the next label in the stack identifies the source of packet. Therefore it acts as follows:

```

IF top label is Y THEN
  Pop the top label;
  IF  $Label_B$  exists in BFT THEN
     $iif_{RPF} \leftarrow BFT(Label_B).input\_interface;$ 
    IF  $iif \neq iif_{RPF}$  THEN
      Discard the packet;
    ELSE
      Push label Y at top of stack in the packet;
      Forward the packet to AllLinks - { $iif$ };
    ENDIF
  ELSE
    Push label Y at top of stack in the packet;
    Forward the packet to AllLinks - { $iif$ };
  ENDIF
ENDIF

```

If the first label in the stack is Y, then the second label ($Label_B$) is looked up in the BFT (Broadcast Forwarding Table). BFT maintains broadcast information in corresponding LSR. It is constructed by means of BLAC broadcast messages. Each entry of this table contains assigned label by BLAC and its corresponding source address. It also contains one of LSR interfaces which is used by that LSR to forward received packet toward the source. As in the basic reverse path forwarding algorithm [1][17], a LSR forwards a broadcast packet originating at source S if and only if it arrives via the shortest path from the LSR back to S (i.e., the "reverse path"). We named this interface as iif_{RPF} in the above pseudo code. Required steps for constructing BFT table are explained later in the text. If $Label_B$ is found in the BFT, then we perform RPF check. If the RPF check is succeeded, data will be copied to all interfaces except incoming one ($AllLinks - \{iif\}$). Otherwise, data will be silently discarded.

If incoming label doesn't exist in BFT, data will be copied to all interfaces except incoming one. This occurs when *Broadcast_Label_Assignment* message is received by S but it isn't reached this LSR yet. Therefore, the LSR will be possibly informed about it in future. There are other options to treat with unknown labels. One can simply drop the corresponding packets. Unfortunately, this reduces reliability of the broadcast mechanism. Another option that we selected is to broadcast the packet anyway and don't bother RPF check. This method wastes network bandwidth but increases reliability of broadcast algorithm.

In this case, the LSR will shortly be informed about label assignment, and returns to normal efficient broadcast mechanism. The final option is using information in the IP header and IP lookup to check RPF which increases processing time of broadcast packet.

When S no more needs the broadcast assigned label, it must inform BLAC to release that label. Therefore it sends a *Broadcast_Label_Release* message toward BLAC. BLAC deletes the corresponding label from the used label set. It must also inform all other LSRs in MPLS domain to delete it from their BFTs. Therefore, BLAC broadcasts a *Broadcast_Label_Release* message in the domain. Here is the pseudo code for BLAC when responding *Broadcast_Label_Release* message:

```

IF ( $\exists$  ( $IP_S$  ,  $Label_B$ ) in  $Table_{Assigned\_Labels}$ ) THEN
    Send a Broadcast_Label_Release message( $X$ ,  $Label_B$ ) destined to S to
    all interfaces;
     $Label\_Range_B \leftarrow Label\_Range_B + \{Label_B\}$ ;
    Delete ( $IP_S$  ,  $Label_B$ ) from  $Table_{Assigned\_Labels}$  ;
ELSE
    Discard the Broadcast_Label_Release message;
ENDIF

```

This algorithm is the same as suggested algorithm for processing of *Broadcast_Label_Request* in BLAC. The format of *Broadcast_Label_Release* message is as follows.

	Shim Header	Source Address	Destination Address	Payload	
L2 header	X	$Label_B$	IP_{BLAC}	IP_S	<i>Broadcast_Label_Release</i>

Since the message is broadcasted, requesting node S is informed about success of the label releasing procedure as well. Careful attention must be paid to the success of S requests because *Broadcast_Label_Request* and *Broadcast_Label_Release* messages may be lost in the network. *Broadcast_Label_Request* may also be dropped by BLAC in case of broadcast label shortage. Therefore S activates a timer initially set to $\alpha * RTT_{BLAC}$ when it sends a request toward BLAC. The timer is turned off after receiving the corresponding reply from BLAC. When the timer is expired, it resends the request in an exponential back-off manner if it has not received the reply message yet. Values of α and RTT_{BLAC} are determined base on simulation experiments. RTT_{BLAC} is the round trip time between S and BLAC.

Till now, we considered the BLAC address to be known by LSRs. The question is how a LSR distinguishes the BLAC address? We can have three solutions. In the first solution, the network administrator selects a LSR as BLAC node and configures its address statically on all MPLS domain LSRs. Obviously; this is not a good solution due to the unwanted miss-configuration and tremendous amount of work needed for changing BLAC node. In the second method, administrator configures a LSR as BLAC statically and BLAC broadcasts the following message periodically.

	Shim Header	Source Address	Destination Address	Payload	
L2 header	X	S=1	IP_{BLAC}	$IP_{Broadcast}$	BLAC_Address

When a LSR receives this message, it finds that it is a BLAC address advertisement message (S=1 in MPLS denotes the last label in label stack [18]). Therefore, it extracts BLAC address from it and adds the following entry to its BFT:

Label-in	Input interface	IP Address
X	Routing[IP_{BLAC}]	IP_{BLAC}

Routing[IP_{BLAC}] returns the output interface which is used to reach BLAC node from this LSR using IP lookup. As you see, there is no output interface field in the BFT. Output interfaces can be calculated as *AllLinks - Input interface*. Output label for a packet is the same as input label and so is not included in the table as well.

In the last solution which is like the monitor election in rings or root election in spanning tree for Ethernet [22], each LSR broadcasts a *BLAC_Address* message when it boots. When a LSR receives such a message it first performs RPF check. If the test succeeded, it compares address contained in the message with a currently stored BLAC address. The stored address is initially set to the LSR's own address. LSR

saves the new address as BLAC address and broadcasts *BLAC_Address* message when the new address is less than the old one. Otherwise, the packet is discarded. At last, this procedure converges and all LSRs will select a LSR with smallest IP address in the domain. The selected node is afterwards responsible to broadcast *BLAC_Address* messages periodically after $T_{\text{Broadcast_Address}}$ to refresh the corresponding BFT entry in LSRs. Each LSR has a timer that is reset to $\beta * T_{\text{Broadcast_Address}}$ when it receives *BLAC_Address* message from BLAC node. If the timer is expired, the LSR will broadcast a *BLAC_Address* message to candidate himself as BLAC. Values of β and $T_{\text{Broadcast_Address}}$ are determined from simulation experiments. They are affected by broadcast reliability and fault tolerance.

There are some considerations regarding the last method. First, each LSR can estimate its processing power and decide not participate the election process. This way, it is possible that no LSR participates in the election. Therefore, the administrator must select one of LSRs as default BLAC that starts election anyway to avoid this situation. Second, election mechanism can be also performed based on some other criterion instead of the smallest IP address and this must be expressed in the election messages.

We present the pseudo-code for a LSR when it receives a broadcast packet from BLAC.

```

IF top label is X THEN// X is a specific reserved broadcast label
  IF BFT(X) does not exist and S=1 THEN
     $iif_{RPF} \leftarrow \text{Routing}[IP_{BLAC}]$ ;
  ELSE IF BFT(X) does not exist and S=0 THEN
    Discard the packet;
    Return;
  ELSE
     $iif_{RPF} \leftarrow \text{BFT}(X).\text{input\_interface}$ ;
  ENDIF

  IF  $iif \neq iif_{RPF}$  THEN
    Discard the packet;
  ELSE IF S=0 THEN
    Pop the top label;

    IF Payload is Broadcast_Label_Assignment THEN
      Insert (LabelB , Routing[IPS] , IPS) in BFT;
    ELSE IF Payload is Broadcast_Label_Release THEN
      Delete (LabelB , Routing[IPS] , IPS) from BFT;
    ELSE
      Discard the packet;
      Return;
    ENDIF

    Push label X at top of stack in the packet;
    Forward the packet to AllLinks - {iif};
  ELSE IF Payload is BLAC_Address THEN
    Insert (X ,  $iif_{RPF}$  , IPBLAC) in BFT;
    Forward the packet to AllLinks - {iif};
  ELSE
    Discard the packet;
  ENDIF
ENDIF

```

On reception of a new *Broadcast_Label_Assignment* which satisfies the above code, the following entry will be added to BFT:

Label-in	Input interface	IP Address
Label _B	Routing[IP _S]	IP _S

We finish this section by an example of the broadcast mechanism as shown in figure 4. In figure 4a, S_1 and S_2 want to broadcast some packets into the network. Both nodes must first request a broadcast label from BLAC by sending a *Broadcast Label Request* message. BLAC assigns L_1 and L_2 to S_1 and S_2 respectively and broadcasts two *Broadcast Label Assignment* messages as depicted in figure 4b. Now, consider LSR_9 . It only accepts broadcast messages from its direct link to BLAC and rejects broadcast messages from LSR_3 due to RPF mechanism. Since the shown network is considered symmetric, the distances between two nodes in two different directions are the same. This is not the case in real networks and most paths are asymmetric [20]. Therefore the RPF interface must be determined in reverse direction in their graphs. Bidirectional arrows in the figure mean that broadcast message is sent in both directions of the link.

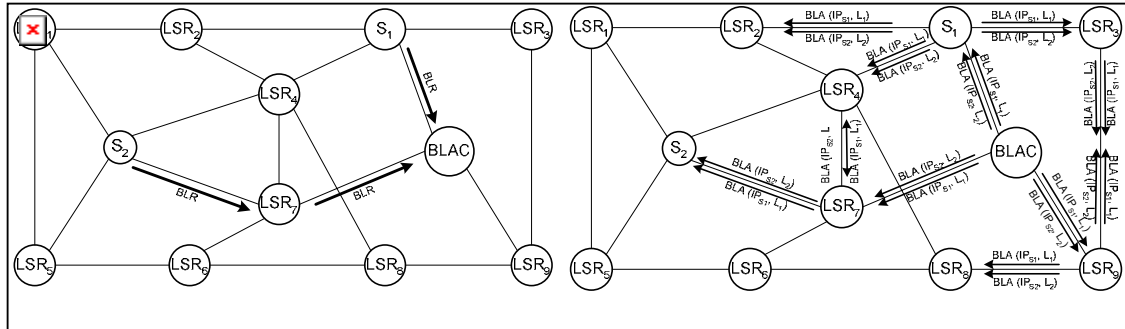


Fig. 4. Two steps of the broadcast protocol (a) broadcast label request by sources (b) label assignment by BLAC

Upon receiving *Broadcast Label Assignment* by S_1 and S_2 , they will broadcast their data using labels L_1 and L_2 accordingly. They will issue *Broadcast Label Release* messages once they are done with broadcasting (the same as fig. 4a). BLAC will broadcast *Broadcast Label Release* messages to other nodes in response (the same as fig. 4b).

BLAC can save network bandwidth and processing power by accommodating *Broadcast Label Assignment* and *Broadcast Label Release* messages in a single message when broadcasting instead of sending separate reply messages for separate requests. It waits for $T_{\text{Delayed_Response}}$ before responding. If other requests are received, it combines the reply messages.

4 Dense-Mode Multicast Protocol

Our broadcast mechanism is used to implement a dense-mode multicast protocol in MPLS. BLAC must assign a broadcast label to each (S, G) pair. Therefore, all messages must be changed so that they contain both source address, S, and multicast group address, G. When a source wants to send data to group G, it first, sends a modified *Broadcast Label Request* message toward BLAC. BLAC responds by a modified *Broadcast Label Assignment* message. As soon as the response is received, it adds following entry to its MFT (Multicast Forwarding Table):

Source Address	Group Address	Label-in	Input interface	Pruned entry
IP_S	IP_G	$Label_B$	$Routing[IP_S]$	NULL

In MFT, *Source Address* and *Group Address* fields together specify a multicast tree. One can also think of them as a FEC that identify a multicast tree. Meaning of *Label-in* and *Input interface* fields is as before. *Pruned entry* field is a pointer to PIT (Pruned Interfaces Table) where the first entry of pruned interfaces for the tree is located. Interface index of pruned interfaces and status of their timers are maintained in this table. In this way, we avoid variable length entry in MFT. Therefore, MFT can be efficiently implemented in proper hardware like CAM. PIT can be implemented in RAM. A simple linked list structure can be used for it. An example of MFT and PIT relationship is presented in figure 5.

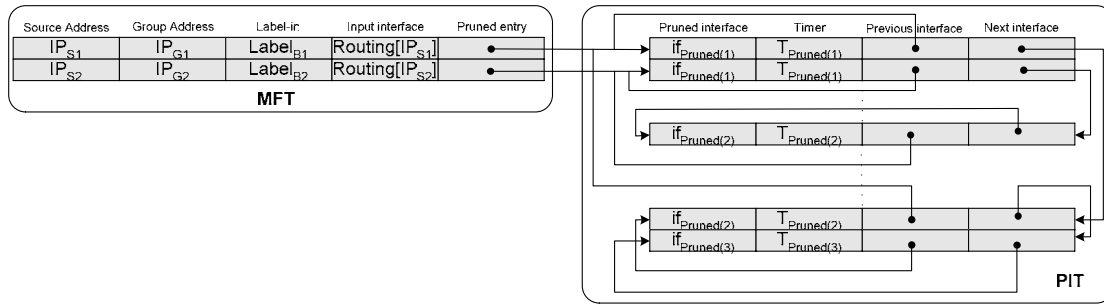


Fig. 5. Linked list structure for PIT

In dense mode multicast groups, number of pruned interfaces of each LSR is zero or a small fraction of total interfaces of a LSR. Therefore, maintaining state of pruned interfaces is more cost-effective than keeping state of tree links. Hence, the main advantage of PIT is its small memory requirements.

Next, we focus on steps needed for constructing PIT. Once source S acquires a broadcast label from BLAC, it adds it to its MFT. Rather than using reserved label Y for broadcasting, it uses another reserved label Z for broadcasting its multicast data. By this means, all other LSRs know that this packet is a multicast data packet. Also the multicast group address must be placed in *Destination Address* field in data packet. All other steps for processing these broadcast packets are like previously described procedures in broadcast algorithms except that this time we use MFT and PIT instead of BFT. The outgoing link set for a data packet is $AllLinks - \{iif\} - \{AllPrunedLinks\}$ in this case.

To avoid the extra label Z, we can do the same thing as the broadcast mechanism. We use the value in layer 2 headers when available to differentiate a multicast MPLS packet from unicast and broadcast. This value already exists in PPP (Protocol field type 0283 hex for MPLS multicast), Ethernet and IEEE 802.3 (ethertype value 8848 hex for MPLS multicast) [18][23]. We propose to define specific values for MPLS multicast in other layer 2 technologies as well where not available.

When a leaf LSR receives multicast packet of source S and it has no member for that group, it sends a *Prune* message toward S. Each LSR that receives this message from one of its interfaces, it examines PIT for that interface. If there exists such interface, resets associated timer. Otherwise, it is added to the PIT. If all interfaces of a LSR are in prune state, it forwards received *Prune* message toward S. In some occasions, a LSR sends a *Graft* message toward S, to restore pruned link. When a LSR receives such message from one of its interfaces, it deletes that interface from PIT.

5 Other Issues

We propose to use three reserved labels X, Y and Z in our mechanisms when there is not possible to have layer 2 supports. As described in [18], since label values 4-15 are reserved and may be assigned by IANA, based on IETF consensus, we propose label values 4-6 for our protocol use.

We use label stack of depth two, one for reserved label (X, Y or Z when needed) and another for broadcast label. Therefore, we may use 4 bytes more than conventional MPLS. This fact must be considered in path MTU discovery process in TCP. We have separated unicast, multicast and broadcast label spaces from each other. Hence, if a label is consumed in one of them, it has no impact on the others. Therefore, one can use $2^{20}-16$ labels in each label space if needed (values 0-15 are reserved [18]). This means that we can support $2^{20}-16$ sources of broadcast data and (S, G) multicast trees in an MPLS domain. Proposed multicast protocol is suitable for intra-domain use, and its extension beyond a MPLS domain through hierarchy of domains is subject to further study.

In this article we implemented RPF technique for broadcasting. Other more efficient techniques like RPB and extended RPF [17][1] can be implemented with a negligible changes. Layer 3 protocol is considered IPv4 in all of our procedures. When other layer 3 protocols are possible, the bottom of label stack must include a special reserved label for each layer 3 protocol as stated in [18]. This may require some small modification on our algorithms.

If BLAC crashes, although existing multicast groups remain unaffected, no new broadcast label can be assigned or released. Therefore, BLAC is a single point of failure and it must be fail-proof. It also can crash when it is overloaded by label requests. We envisage heavily loaded BLACs to be implemented in the form of a computing cluster connected by a fault-tolerant and load-balancing middleware infrastructure. Other protocols like ones in [14] and [19] use the same approach.

Another thing that needs further study is the impact of link failure on the operation of proposed mechanisms and the way we treat that. For example, in the event of link failure, S may not receive reply message from BLAC and repeats its request. Thus, BLAC can broadcast response again or unicast it toward S.

6 Related Works

The first operational IP multicast shortcut over ATM prototype for label switching IP multicast consisted of a Unix workstation and an Alcatel ATM switch [24]. This LSR was a switch/router that was capable of forwarding multicast data using PIM-SM in IP layer and p2mp connections in ATM. The established tree in layer 3 is mapped to a p2mp tree in layer 2 in their LSR. Their proposal needed mixed L2/L3 forwarding for correct operation. They decided to set up LSPs based on changes to the Multicast Forwarding Cache (MFC), which can be categorized as a traffic driven trigger [7]. The MFC is a cached subset of the Multicast Routing Table (MRT). MRT updates MFC when something changes in the routes. The MFC requests MRT for a certain multicast group when it experiences a cache miss for an incoming multicast packet. Although they have chosen PIM-SM as multicast routing protocol in their implementation, the approach works also for PIM-DM and DVMRP. However, the flood & prune nature dictates periodic cache misses due to timer expiration. This makes the approach inappropriate. In addition, they don't explain how they deal with label assignments.

Ooms et. al. in [7] and RFC3353 [8] present detailed framework for multicast support in MPLS. They explain many multicast related problems in MPLS and suggest solutions to some of them.

Protocols such as PIM-SM [2] and CBT [10] have explicit Join messages which could carry the label mappings. This approach is called piggy-backing method and described in [5]. Protocol messages must be changed properly in favor of MPLS. Implementation of their approach in case of dense-mode protocols like PIM-DM and DVMRP is inefficient since these protocols use no explicit messages for piggy-backing labels on them. The pros and cons of piggy-backing labels on multicast routing messages are described in [7][8].

Work in [6] suggests that labels be assigned on a per-flow (source, group) basis in a traffic-driven fashion. When a LSR detects a new multicast FEC, it invokes L3 routing to determine the outgoing interfaces. For each outgoing interface, it selects an unused label and binds that to the corresponding multicast tree branch. Their proposal has some disadvantages compared to ours. First, our proposal outperforms their work regarding MFT size, since they store (interface, label) pairs for each multicast tree branch at the LSR. The majority of interfaces in most LSRs are tree branches in dense-mode groups and this leads to larger MFT size in their approach. Second, they use variable length entries in their MFT. This makes hardware implementations more difficult. Third, label binding and distribution is done at each LSR which introduces extra delay in tree construction. The label binding in our method is done at BLAC and labels are distributed to all LSRs. When the binding reaches the source, it broadcasts its traffic according to new label binding, while other LSRs may still be waiting for this binding. Forth, we consume fewer labels when the label pool is common between interfaces in a LSR.

A traffic-driven label distribution method is introduced in [16] and a dense-mode multicast routing protocol is proposed. They have the same weaknesses as [6] compared to our proposal. The MFT size in their method is even larger than [6] since they replicate common information in their table entries to achieve fixed length entries.

Our proposal consumes more bandwidth in comparison with both [6] and [16]. First, in some LSRs, the labeled data traffic may be ready while the corresponding label binding information may still be on its way. We use the inefficient broadcast scheme in this case. An alternative remedy is to switch this traffic in layer 3 expending more processing power in the LSRs. Second, the extra 4 bytes specific label may exist at top of the stack which consumes bandwidth. This can be avoided as explained using layer 2 support. Third, BLAC broadcast messages consume bandwidth. This can be reduced using delayed responses.

To make multicast traffic suitable for aggregation, the approach in [15] converts p2mp LSP setup to multiple p2p LSP problems. In their protocol, the multicast trees branch only at the edge routers and use the MPLS tunnels set up by the core routers. They assumed multicast members are present only at edge routers. While this condition is not valid in general, the proposed method results in a bad usage of network resources when the groups are dense. The reason is that most LERs are group members in dense-mode and therefore there will be a separate LSP from each source to each receiver. These LSPs may have overlaps, thus several copies of the same data may be transmitted along the common links. Another main disadvantage of their scheme is the need for layer 3 operations at each LER on the traffic to separate the flows from each other when they diverge. This prevents end-to-end label switching of data and introduces extra delay between sender and receivers. In addition, it increases processing load of LERs. The unicast traffic is also disturbed since the existing LSPs between LERs are used, and the same layer 3 operations must be done on this traffic.

7 Conclusion

We propose a protocol for MPLS broadcast for the first time. This mechanism can be used in all applications and protocols requiring broadcast. A central node called BLAC is responsible for broadcast label assignment and release. Proposed mechanism has no overhead on unicast label space. We have shown its use in dense mode multicast support in MPLS. Our multicast protocol has many nice properties in comparison with other proposals available for dense mode groups. It consumes only one label for each multicast tree from multicast label space and no label from unicast label space. Also it has smaller multicast forwarding tables by storing an entry for each non-tree link instead of each tree link. Therefore, it consumes LSR memory conservatively. It poses no delay on multicast data distribution except for the label assignment phase in contrast with mechanisms existing so far.

References

1. S.E.Deering, D.R.Cherton, "Multicast Routing in Datagram Internetworks and Extended LANs", ACM Transactions on Computer Systems, Vol.8, No.2, May.1990
2. S.Deering, D.Estrin, D.Faranacci, V.Jacobson, C.G.Liu, L.Weil, "The PIM Architecture for Wide-Area Multicast Routing", IEEE/ACM Transactions on Networking, Vol.4, No.2, April 1996
3. K.C.Almeroth, "The Evolution of Multicast: From the Mbone to Inter-Domain Multicast to Internet2 Deployment", IEEE Network, Jan./Feb. 2000
4. S.Casner, S.Deering, "First IETF Internet Audiocast", ACM SIGCOMM, Computer Communications Review, Vol. 22, No. 3, July 1992
5. D.Farinacci, Y.Rekhter, E.C.Rosen, T.Qian, "Using PIM to Distribute MPLS Labels for Multicast Routes", IETF Draft, draft-farinacci-mpls-multicast-03.txt, Nov. 2000
6. A.Acharya, F.Griffoul, F.Ansari, "IP Multicast Support in MPLS", IEEE Proceedings on ATM Workshop, 1999
7. D.Ooms, W.Livens, "IP Multicast in MPLS Networks", Proceedings of the IEEE Conference on High Performance Switching and Routing, 2000
8. D.Ooms, B.Sales, W.Livens, A.Acharya, F.Griffoul, F.Ansari, "Overview of IP Multicast in a Multi-Protocol Label Switching (MPLS) Environment", RFC 3353, Aug.2002
9. D.Waitzman, C.Partridge, S.Deering, "Distance Vector Multicast Routing Protocol", RFC 1075, Nov.1988
10. A.Ballardie, "Core Based Trees (CBT Version 2) Multicast Routing - Protocol Specification", RFC 2189, Sep.1997
11. E.Rosen, A.Viswanathan, R.Callon, "Multiprotocol Label Switching Architecture", RFC 3031, Jan.2001
12. A.Watson, M.Sasse, "The Good, the Bad, and the Muffled : the Impact of Different Degradations on Internet Speech", Proceedings of the 8th ACM international multimedia conference, Oct. 2000
13. G. Liu, X. Lin, "MPLS Performance Evaluation in Backbone Network", IEEE International Conference on Communications (ICC 2002), Vol. 2, pp. 1179 - 1183, May 2002
14. A. Boudani, B. Cousin, "A New Approach to Construct Multicast Trees in MPLS Networks", Proceedings of the Seventh International Symposium on Computers and Communications (ISCC 2002), pp. 913 - 919, July 2002
15. B. Yang and P. Mohapatra, "Edge Router Multicasting with MPLS Traffic Engineering", IEEE International Conference on Networks (ICON 2002), Aug. 2002

16. Z. Zhang, K. Long, W. Wang, S. Cheng, "The new mechanism for MPLS supporting IP multicast", The 2000 IEEE Asia-Pacific Conference on Circuits and Systems (APCCAS 2000)
17. Y.K. Dalal, R. M. Metcalfe, "Reverse Path Forwarding of Broadcast Packets", Communications of the ACM, Vol. 21, No. 12, Dec. 1978
18. E. Rosen, D. Tappan, G. Fedorkow, Y. Rekhter, D. Farinacci, T. Li, A. Conta, "MPLS Label Stack Encoding", RFC 3032, Jan. 2001
19. S. Keshav, S. Paul, "Centralized Multicast", Proceedings of IEEE ICNP, 1999
20. V. Paxson, "End-to-End Routing Behavior in the Internet", Proceedings of SIGCOMM '96
21. M.R. Macedonia, D.P. Brutzman, "MBone Provides Audio and Video Across the Internet", IEEE Computer Magazine, Vol. 27, No. 4, pp. 30 -36, April 1994
22. L. L. Peterson, B. S. Davie, "Computer Networks: A Systems Approach", Morgan Kaufmann Publishers, 2nd Edition, 2000
23. "Protocol Numbers and Assignment Services", <http://www.iana.org/numbers.html>
24. P. Dumortier, D. Ooms, W. Livens, I. Girard, M. Ramalho, "IP Multicast Shortcut over ATM: A Winner Combination", IEEE Globecom 1998, Nov. 1998