

# A Case for Dense-Mode Multicast Support in MPLS

Mozafar Bag-Mohammadi<sup>†</sup>, Siavash Samadian-Barzoki<sup>†</sup>,  
Mohammad Nikoopour<sup>±</sup>, Nasser Yazdani<sup>‡</sup>, Naser Rezaee\*  
*Router Lab., University of Tehran, Tehran, Iran*  
<sup>†</sup>{mozafarb, s.samadian}@ece.ut.ac.ir, <sup>±</sup>mnikpur@modares.ac.ir,  
<sup>‡</sup>yazdani@ut.ac.ir, \*n\_rezaee@isc.iranet.net

## Abstract

*In this paper, we propose a new method for dense-mode multicast support in an MPLS domain. Our method uses a unique identifying label for multicast distribution tree. It consumes no label from unicast label space. Label swapping is not required at all, since we use just one label for each multicast tree. This approach results in at least 50% memory savings in LSRs (Label Switch Routers). In our previous method, the tree labels are assigned centrally in the MPLS domain. However, they are assigned in a partially distributed manner by LLAs (Local Label Assigners) in the new scheme. Multicast label ranges are assigned to LLAs proportional to their needs. There is no need for explicit on-demand signaling in this approach. The proposed solution significantly reduces the control messages overhead in comparison with previous work.*

## 1. Introduction

Multicast capability is an important feature in data networks and is evolving into a highly demanded service being offered by nearly all major ISPs (Internet Service Providers). Multicast benefits include reducing the transmission overhead on the sender and network and decreasing data delivery delay for all destinations. Most of multicast applications usually have Quality of Service (QoS) requirements and, thus, the constraints of QoS provisioning should be also considered while supporting multicast communications.

Several techniques have been proposed by Internet Engineering Task Force (IETF) for QoS support in Internet. One of the approaches, Multiprotocol Label Switching (MPLS) is being considered as a scalable solution [1]. MPLS technology enhances IP packet forwarding capability, combining fast ATM switching with the benefits of IP routing. There are many unresolved issues regarding IP multicast support in

MPLS domains [2] and it seems there is no dominant solution. Meanwhile, multicast support is not provided in MPLS architecture and it is left for further study [1].

Almost all multicast routing protocols construct a multicast tree for data distribution purposes. The tree requires state maintenance in routers, which contradicts the stateless nature of IP networks. On the other hand, MPLS uses a pre-established LSP (Label Switching Path) that implies keeping state in switching devices. This behavior is similar to the IP multicast. Many researchers believe that multicast and MPLS are two complementary technologies and merging these two technologies will enhance performance and provide better QoS support for multicast communication [3][4].

Existing multicast routing proposals in MPLS use a separate label for each branch of the multicast tree. The labels are selected from a label space common between multicast and unicast traffics. As a result, the label assignment process to a multicast traffic is not a trivial task. In our architecture for multicast support in MPLS named GAM (General Architecture for Multicasting in MPLS), we identify a multicast tree by a unique Tree Label (*TL*) [7]. Multicast senders use *TL* to send data packets to the receivers. Consequently, label swapping is not required in multicast packet switching.

Based on the GAM architecture, we proposed the first MPLS broadcast mechanism and extended it to support dense mode multicast group communication in MPLS [6]. This proposal called CLB-DM (Centralized Label Binding – Dense Mode), employs a central node named BLAC (Broadcast Label Assignment Center) in order to bind a label to the multicast tree. The explicit broadcast of *TL* assignment and release messages from BLAC in CLB-DM installs the appropriate forwarding states in each LSR. This prevents the malicious LSRs from using unauthorized *TL*s without co-ordination with BLAC. Unfortunately, CLB-DM produces a large amount of control messages when it floods the network with its *TL* assignments and releases. Besides, there is at least one RTT (Round Trip Time) delay between the

sender and BLAC to achieve the *TL* before multicast data distribution. This is not desirable particularly when a Label Edge Router (LER) receives the multicast data from outside the MPLS domain.

In this paper, we present a partially distributed mechanism for dense-mode group communications in an MPLS domain which also comply with GAM. In this method named DLB-DM (Distributed Label Binding – Dense Mode), a central node (BLAC) partitions the multicast label space among some domain entities. Then, each entity assigns a *TL* from the associated label space segment to an incoming unlabeled multicast flow. DLB-DM significantly reduces the number of control messages and the load on BLAC compared to CLB-DM. In addition, the label assignment is faster and data flooding is more efficient. The trade-off is that DLB-DM uses multicast label space inefficiently since the label range of each domain entity may not be used completely. In addition, more states are needed at each LSR compared to CLB-DM.

Section 2 contains a brief overview of GAM. DLB-DM is mainly covered in section 3. Section 4 presents the simulation results. Related work is discussed in section 5, and finally, we conclude in section 6.

## 2. GAM

In dense-mode multicast protocols, a distribution tree is constructed for each (Source, Group) pair. In GAM, we identify the tree using a tree label (*TL*). This label is also used to flood multicast data by the source. As a result, all multicast packets belonging to the same flow are tagged with that label which is not changed in the MPLS domain for the lifetime of the multicast session [7]. The two important issues of GAM are:

- **Uniqueness of the *TL*:** The architecture must include a binding mechanism, which assigns a unique *TL* to each tree in an MPLS domain. The centralized version of this process is introduced as CLB-DM in [6]. The current paper proposes a partially-distributed version for the same task.
- **Differentiation of multicast and unicast label spaces:** When an LSR receives a labeled packet, it must be able to distinguish whether the packet is a unicast or multicast. The reason is that the two different packets use different forwarding tables, processing and label binding mechanism.

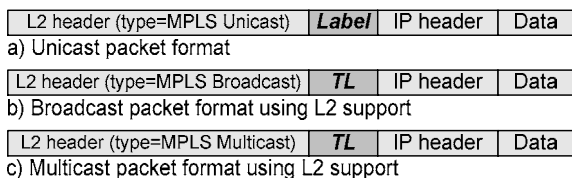


Figure 1. Packet formats using layer 2 support

## 2.1 Using layer 2 support

As the first solution, we suggest defining a specific value in layer 2 headers to indicate whether the label space in the MPLS header is unicast, multicast or broadcast. This value already exists in PPP (*Protocol* field type 0281 hex for MPLS unicast and type 0283 hex for MPLS multicast), Ethernet and IEEE 802.3 (ethertype value 8847 hex for MPLS unicast and value 8848 hex for MPLS multicast) [9][10]. Fig. 1 shows the MPLS packet formats using this solution.

## 2.2 Using 2-Level Label Stack

If the first solution was not possible, then we use a two-level label stack for each multicast or broadcast data packet in an MPLS domain. Therefore, we define two globally specific reserved label values among MPLS labels (*Z* and *Y*) to sit at the top of packet's label stack, as illustrated in Fig. 2. In this figure, labels *Z* and *Y* are used to identify multicast and broadcast data packets, respectively. The second 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.

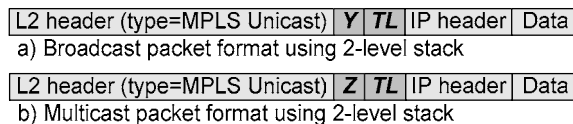


Figure 2. Packet formats using 2-level label stack

## 3. DLB-DM mechanism

DLB-DM uses the same packet formats as CLB-DM and GAM to differentiate various types of packet. However, the label binding in DLB-DM is distributed and the multicast packet distribution is partly different.

### 3.1 Partially distributed mechanism for label binding

The functionality of BLAC (Broadcast Label Assignment Center) is different in DLB-DM in comparison with CLB-DM, where it was responsible to assign a unique *TL* or release it in response to an explicit request of a sender. In DLB-DM, BLAC centrally partitions the whole multicast label space among *LLAs* (Local Label Assigners) proportional to their needs. *LLAs* are those nodes in an MPLS domain where the potential multicast senders are local to them. After this partitioning, the authority of *TL* assignment to a multicast sender from the specified label range is

given to the corresponding *LLA*. Therefore, when an unlabeled multicast flow reaches an *LLA*, it assigns an unused *TL* from its own label range to that flow. As a result, the label assignment process is no longer centralized and it is locally accomplished at *LLAs* in a distributed manner. This eliminates the BLAC assignment and release messages, which were flooded in the network in CLB-DM [6].

In this paper, we consider two different kinds of MPLS domain. The first category is a domain, which contains no multicast senders/receivers except at the LERs. Backbone networks are examples of such domains. In the second category, some LSRs in the domain are connected to LANs consisting of many hosts, which may contain some multicast senders/receivers. An example of this category is a stub network. In the first category, LERs inject the multicast traffic into the MPLS domain only from outside sources. Therefore, *LLAs* are the LERs of the MPLS domain. Since the multicast senders may reside in MPLS domain LANs for the second category, there must be an *LLA* at each connected cluster of LANs in addition to the LERs. Hence, the multicast traffic source is external in the first category and is either internal or external in the second one.

For each LAN that connects more than one LSR, we must choose one of the LSRs to be responsible for *TL* assignments. For the sake of simplicity, we propose that the querier of IGMP (Internet Group Management Protocol) [14] be also the *LLA* of the LAN. The following subsections describe the protocol details for the second category. The case for backbone MPLS domains will be presented in the last subsection.

**3.1.1 Label Space Partitioning.** In DLB-DM, BLAC partitions the multicast label space among all *LLAs*. Hence, it must know the required amount of *TLs* at each *LLA* to assign sufficient label range. There are two kinds of *LLAs*: LER-*LLAs* and LAN-*LLAs*. To estimate the required number of *TLs* at a LAN-*LLA*, we assume it must be proportional to the LAN size. Therefore, we consider the prefix corresponding to each LAN as a representative measure for the LAN size. One can also think of other criteria such as finding the exact host count on the LAN. We avoid such complexities for the sake of simplification.

Estimation of the required *TLs* for LER-*LLAs* is not a trivial task. The reason is the unknown pattern of incoming multicast flows from external sources. Therefore, we propose that BLAC partitions the multicast label space into two different subspaces. One subspace is then partitioned equally among LER-*LLAs* and the other one is partitioned among the LAN-*LLAs* proportional to their LAN size. We define a special parameter  $\alpha < 1$ , which is the ratio of potential external

multicast senders to all potential multicast senders for an MPLS domain. We suggest this parameter to be set statically by the administrator of the domain. It can also be set through a dynamic algorithm. BLAC partitions the multicast label space as follows:

$$Subspace_1 = \alpha \cdot (2^{20} - 16) \quad (1)$$

$$Subspace_2 = (1 - \alpha) \cdot (2^{20} - 16) \quad (2)$$

$$|Label\_Range\_LER_i| = \left\lceil \frac{Subspace_1}{N_{LERs}} \right\rceil \quad (3)$$

$$N_{Hosts} = \sum_{i=1}^{N_{LANs}} 2^{32-PL_i} \quad (4)$$

$$|Label\_Range\_LAN_i| = \left\lceil \frac{2^{32-PL_i}}{N_{Hosts}} \cdot Subspace_2 \right\rceil \quad (5)$$

*Subspace<sub>1</sub>* and *Subspace<sub>2</sub>* in equations (1) and (2) are the portions of the multicast label space dedicated to LER-*LLAs* and LAN-*LLAs* respectively. In these equations, the multicast label space includes  $2^{20}-16$  labels. The reason is that values 0 to 15 among  $2^{20}$  possible labels are reserved [9][6]. Equation (3) calculates the size of the label range for each LER-*LLA* where  $N_{LERs}$  stands for the number of LERs in the domain. *Label\_Range\_LER<sub>1</sub>* starts at 16 and *Label\_Range\_LER<sub>i</sub>* starts after *Label\_Range\_LER<sub>i-1</sub>*. Equation (4) estimates the number of possible hosts in all LANs of the domain.  $N_{Hosts}$  and  $N_{LANs}$  in this equation are respectively the number of possible hosts and the number of LANs in the domain. As stated before, we estimate  $2^{32-PL_i}$  as the number of possible hosts in *LAN<sub>i</sub>* where  $PL_i$  is the prefix length of that LAN. Finally, the size of *Label\_Range\_LAN<sub>i</sub>* is computed in equation (5), proportional to estimated size of *LAN<sub>i</sub>*. *Label\_Range\_LAN<sub>1</sub>* starts after last LER label range and *Label\_Range\_LAN<sub>i</sub>* starts after *Label\_Range\_LAN<sub>i-1</sub>*. The more possible hosts on the LAN, bigger is label range assigned to it.

So far, we assumed that BLAC knows *LLAs* and their related information. This can be trivially achieved using any link state unicast routing protocol such as OSPF [12], or it should be collected from *LLAs* otherwise. In the latter case, LER-*LLAs* and LAN-*LLAs* are responsible for notifying BLAC about their existence and the prefix corresponding to LAN-*LLAs*. This is done reliably through a TCP/T [15] connection. This protocol is a modification of TCP with little overhead on connection establishment and closing and it is useful for having a reliable transaction with few messages. To inform BLAC about *LLA* failure, this connection must use the keep-alive mechanism existing in TCP/T. The connection is set up whenever *LLA* is booted up or selected again in the LAN.

Upon receiving the information about a new *LLA*,

BLAC computes the new label ranges and inserts the new information in a special table named LRT (Label Range Table). Each entry of this table consists of an *LLA* IP address, the label range assigned to it and corresponding prefix length for LAN-*LLAs* or 0 for LER-*LLAs* (see Fig. 3). Entry addition and deletion changes the label ranges in all entries of the table. Then, the new table is broadcasted to inform the *LLAs* about the new changes in the label ranges. However, this rarely happens. It occurs when a new *LLA* is added to the network or one is changed. LRT is also broadcasted periodically at regular intervals. *LLAs* recognize the label range, which is assigned to them through LRT broadcasting from BLAC. They can use the corresponding label range to assign *Tls* until the next time on which LRT is broadcasted. All LSRs store a copy of the advertised LRT which is used to validate the *Tl* of labeled multicast packets (See section 3.2).

$IP\ of\ LER-LLA_j$	0	$LABEL\ RANGE_j$
:	:	:
$IP\ of\ LER-LLA_M$	0	$LABEL\ RANGE_M$
$IP\ of\ LAN-LLA_j$	$PL_j$	$LABEL\ RANGE_{j+M}$
:	:	:
$IP\ of\ LAN-LLA_N$	$PL_N$	$LABEL\ RANGE_{N+M}$

**Figure 3. LRT structure.**

An *LLA* failure - which can be detected through its TCP/T connection keep-alive messages - causes the corresponding entry in BLAC's LRT to go in a reserved state. If the total amount of entries with reserved state in LRT passes a specified threshold, BLAC deletes them and computes the label ranges in all entries again. A timer is set for each reserved entry as well. The entry is deleted when this timer is expired and the corresponding *LLA* is down. If the *LLA* comes up before the timer expires, it establishes a new TCP/T connection. This time BLAC finds the corresponding *LLA* entry in LRT and the entry's state is changed to the normal mode. This way LRT can tolerate transient failures in some *LLAs*.

We define another reserved label value  $X$  in addition to  $Y$  and  $Z$ . When BLAC broadcasts a message in the domain, it places label  $X$  at top of the message label stack. The first message from BLAC then creates a tree in the same way that the multicast trees are constructed for the multicast senders (see section 3.2). This tree spans all LSRs in the domain and provides the label switching mechanism for BLAC broadcasts. As a result, all LSRs (except the leaf ones) have an entry in their MFTs with label  $X$  as the *Tl*. The tree construction method and MFT structure will be described in subsection 2. As stated in [9], since label values 4-15 are reserved and may be assigned by IANA, based on IETF consensus, we propose label

values 4-6 for labels  $X$ ,  $Y$  and  $Z$ .

To protect the broadcast of LRT against loss, an LSR, which sends a control message to another, must set a timer. The control messages should be acknowledged by the receiving LSR to the sender. The sender LSR will retransmit the control messages at regular intervals until reception of the acknowledgements. OSPF (Open Shortest Path First) routing protocol [12] uses the same mechanism for the reliable flooding of its link state messages.

If BLAC crashes, although existing *LLA* operations remain unaffected, no new *LLA* can be formed in the domain. Therefore, BLAC is a single point of failure and it must be fail-proof. We envisage heavily loaded BLAC to be implemented in the form of a computing cluster connected by a fault-tolerant and load-balancing middleware infrastructure. Other protocols like those in [3] and [13] use the same approach. The BLAC selection process is described in [6]. Therefore, we omit it here and assume that a fixed BLAC is selected in the domain for the rest of the paper.

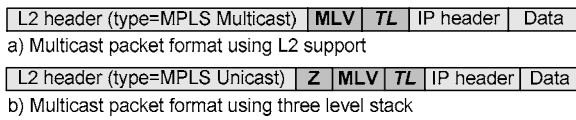
**3.1.2 On-Demand Label Assignment at *LLAs*.** After the processes in the previous subsection are accomplished, all *LLAs* know the label range that can be used by the multicast senders local to them. The first multicast packet that belongs to a new (Source, Group) pair on a LAN, triggers the LAN-*LLA* to search its label range to find an unused *Tl* and assign it to the multicast tree rooted at the sender. It advertises this *Tl* with a *Bla* (Broadcast Label Assignment) message in the LAN to inform other LSRs connected to that LAN to use it. *LLA* knows the existence of other connected LSRs from unicast routing protocols and can eliminate this message if there are no other LSRs. When a new external multicast flow reaches an LER-*LLA*, it simply searches its label range to find an unused *Tl* and assigns it to the tree rooted at the LER-*LLA*.

The Minimum Label Value (*MLV*) from the label range corresponding to each *LLA* is reserved for extension. If no unused label is available in the label range except *MLV*, the *LLA* extends its multicast label space by  $2^{20}$ -16 labels using one more level in the label stack. Figure 4 shows the multicast packet formats in this case. The packet formats are similar to those in section 3.1 except that they are extended using the *LLA*'s *MLV*. This means that the *LLA* picks up an unused *Tl* from the 3<sup>rd</sup> level extended label space. This way, the size of the *LLA*'s label range does not limit the number of possible senders local to it. The trade-off is that the number of levels in the label stack increases.

Using a 3 level label stack, there can be a minimum of  $2^{20}$ -16 multicast trees rooted from each *LLA*. We believe that this number is much more than sufficient for each *LLA* and so, it is useless to extend the levels

beyond 3. If the 3<sup>rd</sup> level label space is unlikely exhausted, the LAN-LLA sends a *BLE* (Broadcast Label Exhausted) message to the sender. An LER-LLA discards the new multicast traffic silently in this situation. In the LAN case, it is up to the sender to try again later when it received a *BLE* message.

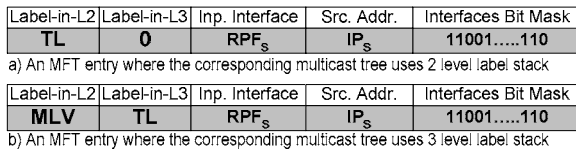
As the *BLA* message may be lost in the LAN, we need a mechanism to protect it. Therefore, other LSRs in the LAN set a timer on data arrival to a random value less than a threshold. If a timer expires and no *BLA* message is received, the LSR sends a *BLR* (Broadcast Label Request) message to the LAN-LLA. The LAN-LLA sends the *BLA* message in response. Since only one *BLA* is sufficient in the multi-access link, other LSRs in the LAN suppress their timers when they see the *BLR* message.



**Figure 4. Packet formats in DLB-DM when the label range corresponding to an LLA is exhausted.**

### 3.2 Multicast Packet Distribution

By this time, the corresponding LLA has assigned the appropriate *TL* to the new multicast flow. To start the flooding phase of the protocol, it sends the labeled data packets to its child interface(s) due to the RPB (Reverse Path Broadcasting) scheme [11]. Actually, we use the revised version of RPB implemented in DVMRP [5]. DVMRP tries to avoid sending unnecessary packets to neighbors who will then generate prune messages due to the failed RPF check.



**Figure 5. MFT structure in each LSR in DLB-DM**

Fig. 5a illustrates an entry in the MFT of an LSR. We have revised the structure of this table compared to the CLB-DM [6]. In this figure *RPF<sub>s</sub>* is the RPF (Reverse Path Forwarding) link towards *S*. The *IP<sub>s</sub>* field is the address of the multicast sender. This field is used to compute new value of *RPF<sub>s</sub>* when the unicast route to the sender changes. The *Interfaces Bit Mask* field in each entry indicates the branches of the corresponding tree among the LSR interfaces. Upon MFT entry insertion, this bit mask is filled with value 1 where the interfaces are the LSR child links. This guarantees the flooding phase of the protocol to forward the packets based on RPB algorithm.

Each MFT entry also consists of two 20-bit labels *Label-in-L2* and *Label-in-L3*. Figs. 5a and 5b illustrate the two kinds of entries in MFT where the LLA uses 2 or 3 levels in the label stack. In Fig. 5a, *Label-in-L3* is set to 0 which is considered as NULL value and *Label-in-L2* is the *TL*. Since the usable label space starts at 16 [9], no LLA would label a multicast flow with value 0. Therefore, this value in the MFT means that there is no more multicast label in the 3<sup>rd</sup> level label stack. *Label-in-L2* in Fig. 5b contains the *MLV* (Minimum Label Value) and *Label-in-L3* includes *TL* (see section 3.1.2).

When a labeled multicast packet arrives at an intermediate LSR, the label is searched in the MFT. If it was not found, the LSR considers it as the first packet of a multicast stream. Hence, the LSR checks the label in its LRT to find out if the label is an *MLV*. If it was, the LSR knows that the label in the next level of the label stack is *TL*. Otherwise, it considers the label itself as *TL*. For the LAN-LLA, the matched LRT entry is used to validate the *TL* (or *MLV* in the case of extension) due to security reasons. In this case, an LSR finds the longest prefix match between the packets source IP address and the IP address of LAN-LLA. If the value is greater than the corresponding *PL* (Prefix Length) of the LAN-LLA, the check is passed. Then, it performs the well-known RPF check [11] to check whether the packet is received on the correct interface. The LSR discards the packet if the RPF check or *TL* check fails. If both checks succeed, the LSR forwards the packet to all its child interfaces according to the RPB algorithm [11]. Then, the LSR insert the appropriate entry in its MFT to forward the future multicast data packets of the same flow. Subsequent data packets match an entry in MFTs and are forwarded accordingly.

The first multicast data packet installs the corresponding entry in MFTs of all LSRs in the flooding phase of the protocol. It also triggers the *prune* messages from the non-member leaf LAN-LLAs [11] or the non-member LER-LLAs. When an LSR receives a *prune* message from an interface, it searches its MFT to find the corresponding entry. Then, it turns off the appropriate bit in *Interfaces Bit Mask* field of the entry to prune the interface from the tree. Prune timer expiration or reception of a *graft* message on the same interface turns on the same bit. This way, DLB-DM builds the same multicast tree as DVMRP [5].

### 3.3 DLB-DM in Backbone

The previous subsections considered an MPLS domain, which contains some LANs, and the multicast senders/receivers are either internal or external. This is not the case for backbone networks where the multicast senders and receivers are only external to the domain.

As a result, the group members in these networks reside only on the edge of the network where LERs are located. Hence, in backbone networks, *LLAs* are only LER-*LLAs* and  $\alpha=1$  is used for label space partitioning (see section 3.1.1).

## 4. Simulation Results

To show the effectiveness of the proposed mechanism, we have evaluated the MFT size and the number of control messages in comparison with related work and our previous work, using NS-2 environment [16]. We considered network size and activity rate of multicast sources as the two main parameters that affect the performance of our protocol. The activity rate is assumed as the percentage of *LLAs* that start or stop sending multicast data in one period of LRT distribution in DLB-DM. Of course, one can consider other criteria for the activity rate definition. However, our conclusion from the simulation results is independent of this definition.

We used two sets of experiments, one in stub networks and another in backbone networks. To check the independency of the simulation results from the network topology, we used two different flat random graph-generating models that are considered close to real network topologies [17][18]. These models namely, Locality and Doar-Leslie, are part of GT-ITM network topology generator [17]. We fixed the average node degree of the generated random graphs approximately at 3.5 and 5.5 for stub and backbone networks respectively. The backbone network size is fixed at 1000 nodes and the stub network sizes change from 10 to 195 nodes. One should note that using flat topology generators, it is not practical to generate connected random graphs with average node degrees less than 6 in moderate-sized graphs (e.g., 1500 nodes). Besides, it seems that the average node degree of 3.5 is a realistic choice for small-sized graphs [17][18].

### 4.1 Stub Networks

Fig. 6 shows the comparison between the MFT size of CLB-DM, DLB-DM, Acharya method [19] and Zhang method [20] using Doar-Leslie flat random graph models. We take Acharya and Zhang methods into account for our comparison since to the best of our knowledge; they are the only other existing approaches that support dense-mode multicast in MPLS. However, these methods use one label per tree branch in their MFT structures. The difference between them is that Acharya method uses variable length entries while Zhang method replicates common information in its table entries to achieve fixed length entries. The MFT

structure in CLB-DM and DLB-DM uses fixed size entries.

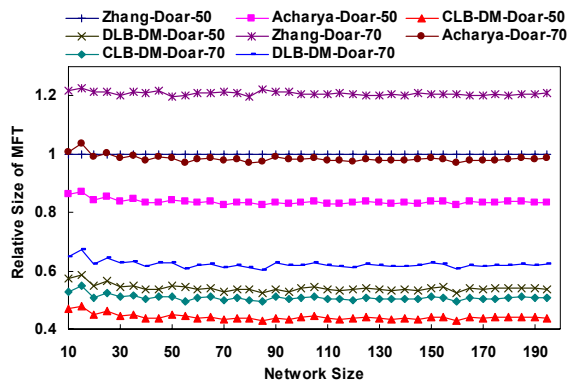


Figure 6. MFT size comparison based on network size in Doar-Leslie flat random topology model

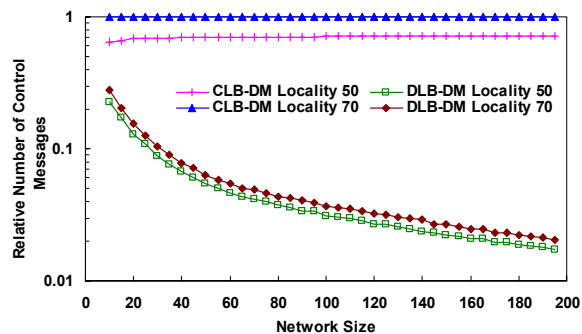


Figure 7. The control messages overhead comparison based on network size in stub networks with Locality flat random topology model

The MFT size of each method is evaluated when the group size is 50% and 70% of network nodes, making up two different plots for each method. For example DLB-DM70 shows the MFT size in DLB-DM when 70% of the network nodes are group members. We built a multicast tree and computed the summation of MFT sizes on all tree nodes to have a measure for MFT size of each method. Each point of the plot is the result of 20 different tree constructions in 10 different random graphs. This produces 200 different runs for each point. The identities of the group members are selected randomly for each run. Finally, all the plots are normalized to Zhang50.

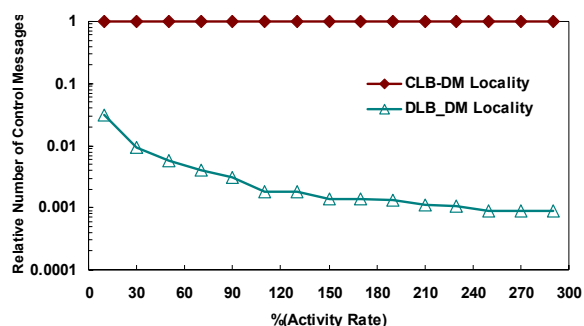
As it can be seen in Fig. 6, CLB-DM has the smallest MFT size while Zhang has the biggest of all. DLB-DM and Acharya reside between these two extremes with less MFT size for DLB-DM. Even DLB-DM70 has about 20% smaller MFT size than Acharya50. Obviously, the growth in group size results in bigger trees which leads to larger MFT size. But, the relative results are roughly independent of the network size. Note that the LRT size is not considered in this comparison since one LRT is stored at each LSR for all

multicast trees in the domain and thus the overhead is divided by the number of multicast trees. The results with Locality network model are the same and omitted from here due to space limit.

Fig. 7 shows the relative comparison between the number of control messages in CLB-DM and DLB-DM in logarithmic scale. In each network size, we selected 50% and 70% of the nodes as *LLAs*. Then, we considered 100% activity rate for this number of *LLAs*. All the plots are normalized to CLB-DM70. The results show that DLB-DM reduces the control messages overhead dramatically in the network. The network size growth sharply increases the difference. The differences between DLB-DM50 and DLB-DM70 are the result of the increase in number of *LLAs*, which increases the number of TCP/T connections. Furthermore, the growth of *LLA* count, increases the number of activities in the network which is the basis for the differences between CLB-DM50 and CLB-DM70. The network model used in Fig. 7 is Locality while the results with Doar-Leslie network model shows exactly the same behavior with slightly more divergence between CLB-DM and DLB-DM.

## 4.2 Backbone Networks

The effect of activity rate on the number of control messages in CLB-DM and DLB-DM is illustrated in Fig. 8 in logarithmic scale. We generated 10 flat networks with 1000 nodes and executed 10 simulation runs on each topology. At each run, 300 nodes are selected randomly among each topology as LERs. The session initiations and terminations are also randomly distributed among these LERs for each run. The results are normalized to CLB-DM. The simulations for backbone networks confirm the same relative superiority of DLB-DM against CLB-DM in terms of control messages overhead as observed in stub domains. Again, the divergence between DLB-DM and CLB-DM is a little more with Doar-Leslie model.



**Figure 8. The control messages overhead comparison based on activity rate in backbone networks with Locality flat random topology model**

## 5. Related Work

The first operational prototype for label switching IP multicast consists of a Unix workstation and an ATM switch [21]. This LSR was a switch/router that was capable of forwarding multicast data using PIM-SM [22] in IP layer and p2mp (point-to-multipoint) connections in ATM. The established tree in layer 3 is mapped to a p2mp tree in layer 2 in their LSR. Although they have chosen PIM-SM in their implementation, the approach works also for PIM-DM and DVMRP. Ooms et al. in [23] and [2] 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 [22] and CBT [24] have explicit Join messages, which could carry the label mappings in MPLS. This approach called piggy-backing method is described in [25]. 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 protocol messages are described in [23][2].

Reference [19] suggests that labels be assigned on a per-flow (source, group) basis in a traffic-driven fashion. A traffic-driven label distribution method is introduced in [20] and a dense-mode multicast routing protocol is proposed there. In these proposals, label binding and distribution is done at each LSR, which introduces extra delay in the tree construction. In contrast, label binding in DLB-DM is faster due to its local label binding mechanism. In addition, DLB-DM consumes fewer labels when the label pool is common between interfaces in an LSR. The MFT size in our mechanisms is also much smaller than them as shown in simulation results.

To make multicast traffic suitable for aggregation, the approach in [4] converts p2mp LSP setup to multiple p2p (point-to-point) LSP problems. The protocol assumes multicast members are present only at edge routers. When the groups are dense, this method results in an inefficient usage of the network resources. A new method for sparse mode multicast support is proposed in [3]. The method uses a centralized entity to calculate the tree based on *Join* and *Prune* messages of receivers. Reference [26] also proposes a simple and inefficient method to implement PIM-SM in ATM based MPLS networks.

Work in [27] addresses the required extensions to MPLS signaling protocols, RSVP-TE (Resource Reservation Protocol with Traffic Engineering

extensions) and LDP (Label Distribution Protocol), to support MPLS network multicasting functionalities.

We suggested the first MPLS broadcast scheme and extended it to support dense-mode group communication in MPLS [6]. To provide scalable QoS multicast support, [8] proposes a new architecture, called AQoSM (Aggregated QoS Multicast). AQoSM can support QoS multicast scalably in DiffServ supported MPLS networks since it maintains MPLS-trees that serve multiple groups. This aggregated approach results in some extra traffic in the network since an aggregated tree may be leaky for some groups. The reason is that the set of the group members and the tree leaves are not always identical.

## 6. Conclusion

We have proposed DLB-DM, which is a partially distributed protocol for dense-mode multicast support in MPLS. DLB-DM has dramatically less control messages overhead than CLB-DM. Simulation results of the proposed mechanism show a great performance benefits over related work and our previous work. The protocol includes some noticeable features such as no requirement for label assignment signaling. We consider the following issues as future work:

- Adaptation of the  $\alpha$  parameter in DLB-DM to the traffic pattern of LER-LLAs and LAN-LLAs to balance the label partitioning process to the requirements of each LLA category dynamically. The estimation of the number of hosts in a LAN can also be changed to achieve exact count.
- Study of QoS considerations in DLB-DM to reserve the appropriate network resources for multicast flows.
- We will study the application of GAM in sparse-mode multicast in future since most of the current multicast applications need this kind of group communication.

## 7. References

- [1] E.Rosen, A.Viswanathan, R.Callon, "Multiprotocol label switching architecture", RFC 3031, Jan. 2001
- [2] D.Ooms, et al., "Overview of IP multicast in a Multi-Protocol Label Switching (MPLS) environment", RFC 3353, Aug. 2002
- [3] A. Boudani, B. Cousin, "A new approach to construct multicast trees in MPLS networks", Proc. Of ISCC 2002, pp. 913 - 919, July 2002
- [4] B. Yang and P. Mohapatra, "Edge router multicasting with MPLS traffic engineering", Proc. Of ICON 2002, Aug. 2002
- [5] D.Waitzman, C.Partridge, S.Deering, "Distance Vector Multicast Routing Protocol", RFC 1075, Nov.1988
- [6] S.Samadian-Barzoki, M.Bag Mohammadi, N.Yazdani, "A mechanism for MPLS broadcast and its extension for multicast dense-mode support in MPLS", Proc. Of ICOIN 2003, LNCS 2662, Korea, Feb. 2003
- [7] S.Samadian-Barzoki, M.Bag-Mohammadi, M.Nikoopour, N.Yazdani, "An Architecture for Multicast Routing Protocol Support in MPLS", Proc. Of IST 2003, Isfahan, Iran, Aug. 2003.
- [8] J.-H. Cui, J. Kim, A. Fei, M. Faloutsos, M. Gerla, "Scalable QoS multicast provisioning in Diff-Serv-Supported MPLS networks", Proc. Of IEEE Globecom2002, Taiwan, Nov. 2002
- [9] E. Rosen, et al., "MPLS label stack encoding", RFC 3032, Jan. 2001
- [10] "Protocol Numbers and Assignment Services", <http://www.iana.org/numbers.html>
- [11] S.E.Deering, D.R.Cheriton, "Multicast routing in datagram internetworks and extended LANs", ACM Trans on Computer Systems, Vol.8, No.2, May.1990
- [12] J. Moy, "OSPF version 2", RFC 2328, Apr. 1998
- [13] S. Keshav, S. Paul, "Centralized multicast", Proc. of IEEE ICNP, 1999
- [14] B. Cain, et. al., "Internet Group Management Protocol, Version 3", RFC 3376, Oct. 2002
- [15] W.S.Stevens, "TCP/IP illustrated", Vol. 3, Addison-Wesley Pub., 1996
- [16] "The Network Simulator - ns-2", [www.isi.edu/nsnam/ns/](http://www.isi.edu/nsnam/ns/)
- [17] E. W. Zegura, K. Calvert, S. Bhattacharjee., "How to model an Internetwork.", Proc. of IEEE Infocom '96, San Francisco, CA
- [18] E.W.Zegura, K.L.Calvert, M.J.Donahoo, "A quantitative comparison of graph-based models for Internet topology", Journal of IEEE/ACM Trans. on Networking, Dec. 1997.
- [19] A.Acharya, F.Griffoul, F.Ansari, "IP multicast support in MPLS", IEEE Proc. on ATM Workshop, 1999
- [20] Z. Zhang, K. Long, W. Wang, S. Cheng, "The new mechanism for MPLS supporting IP multicast", Proc. Of APCCAS 2000.
- [21] P. Dumortier, et al., "IP multicast shortcut over ATM: A winner combination", IEEE Globecom'98
- [22] S.Deering, et al., "The PIM architecture for wide-area multicast routing", IEEE/ACM Trans. on Networking, Vol.4, No.2, April 1996
- [23] D.Ooms, W.Livens, "IP multicast in MPLS networks", Proc. of IEEE HPSR 2000.
- [24] A.Ballardie, "Core Based Trees (CBT Version 2) multicast routing - Protocol specification", RFC 2189, Sep.1997
- [25] 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
- [26] J. Cho, M. Y. Chung, "A simple method for implementing PIM to ATM based MPLS networks", Proc. Of IEEE ICON, p.p. 362-365, Oct. 2001
- [27] J.-M. Chung, M. A. S. Benito, G. Y. Cho, P. Rasiah, H. Chhabra, "MPLS multicasting through enhanced LDP and RSVP-TE control", Proc. Of MWSCAS-2002, Volume: 3, p.p. 93-96, 2002