2009 3rd IEEE International Conference on Digital Ecosystems and Technologies

Dynamic Real-Time IP-based Publish-Subscribe Group Collaboration using Multicast Label Filters

Jaipal Singh[1], Member, IEEE, Prakash Veeraraghavan[2] and Samar Singh[2], Senior Member, IEEE

[1]Digital Ecosystems and Business Intelligence Institute, Curtin University of Technology, Perth, Australia,
e-mail: j.singh@cbs.curtin.edu.au
[2]Department of Computer Science and Computer Engineering, Latrobe University, Melbourne, Australia,
e-mail: {p.veera, s.singh}@latrobe.edu.au

***Abstract*—Internet based enterprise level collaboration tools enable organisations to make decisions faster and more accurately with less effort. However, these tools provide limited real-time group collaboration within and across organisations. This paper proposes a novel IP network layer filter architecture that provides efficient and scalable real-time group collaboration between the required entities within an organisation. This proposed network architecture uses a label filter mechanism to improve privacy, scalability and bandwidth for one-to-one, one-to-many and many-to-many real-time communication on an existing multicast group in an organisation.**

***Index Terms*—Collaborative work, multicast transmissions, network filters.**

I. Introduction

In recent years, businesses have increasingly used the Internet to implement their business processes within an organisation as well as share information with partner organisations. They have started adopting web-based Enterprise level collaboration tools for providing better workplace collaboration and automation of enterprise business processes.

These collaboration tools use a web-based interface to provide features like document/content management, file sharing, real-time data updates, text chat, application/document sharing and threaded discussions. While the information is provided in real-time, the recipient will access the information at a different time. These asynchronous (not real-time) information sharing tools provide very limited real-time collaboration among these various users.

This paper described an innovative publish-subscribe network architecture that provides efficient real-time group communication in an enterprise. Our proposed architecture provides a dynamic group collaboration infrastructure that can be built on and used by digital ecosystems (DES) applications. The proposed architecture will reduce the end user computation overheads and optimise the use of network bandwidth for delivering data packets between all users collaborating through the organisation's network infrastructure. This architecture will provide a novel filtering architecture for private communication on a shared network path so that the users will have more privacy and flexibility when using the organisation's network for real-time collaboration.

This paper is organised in 5 sections. Section I gives a brief introduction to the problem domain. Section II provides an overview of the network layer routing protocols. Section III describes the proposed network layer group collaboration architecture. Section IV provides simulation analysis of the proposed architecture. Section IV concludes this paper.

II. Overview of Network Level Group Communication

Publish-Subscribe systems are used to provide more efficient group level communication on a network. They are categorised as subject-based or content-based, where the subscribers will receive information in a timely manner based on a particular subject or specific content respectively [1]. This paper uses an IP multicast group architecture to implement both the subject-based and content-based publish-subscribe model.

Regular Internet communication uses a point-to-point (unicast) protocol where a data packet instance is created for each sender-recipient pair. Multicast routing protocols were introduced as a more efficient method for group communication over an IP network. The multicast protocol creates a logical network tree represented by a multicast group address that anonymously connects all nodes that want to receive the same communication. The sender will address packets to the multicast address and every node that subscribes to this group (address) will receive these packets. For group communication, multicast is more efficient that unicast since only a single instance of the data packet will be transmitted on every link in the multicast tree [2].

There are two categories of multicast network protocols, source-based trees (SBT) and shared trees (ShT). A SBT protocol is used for a multicast group that only has one sender and the receivers are densely grouped together in the network. If multiple nodes on the SBT want to send messages, N multicast groups have to be formed where N is the number of senders. The IPv4 multicast specification allows only 2^{24} multicast addresses. This limited address space presents a huge scalability issue and limits the number of multicast groups that can be created [3]. A sparsely populated multicast group can use either SBT or ShT protocol. Unlike SBT, ShT can have multiple senders on the same tree. ShT overcomes the scalability problem of SBT since all nodes on the ShT can transmit packets using the same tree. However, ShT might incur more delay than SBT since all traffic has to go through a central core router [4, 5].

In terms of applications, a multicast group is ideal for distributing subject-based communication through the publish-subscribe model. The SBT protocol is ideal for one way

978-1-4244-2346-0/09/$25.00 ©2009 IEEE

communication such as distributing software updates or content to all receivers in the network. The ShT protocol can be used for providing interactive collaboration such as interactive video conferencing among many users for a particular subject.

This paper will use a ShT multicast protocol as the network routing protocol. A ShT protocol is selected since it is built to support one-to-one, one-to-many, and many-to-many communication [6]. Unfortunately, the ShT multicast protocol has scalability issues due to limited IP address allocations and does not provide any privacy since all members in the group can send and receive the communication packets. The ShT protocol is not suited for content-based delivery since the receivers join the multicast group based on the subject. Previous researchers have proposed using filters on a multicast tree to improve scalability, privacy and implementing content-based distribution, but these methods still have some drawbacks like processing delays and high network computation costs [7].

A. Multicast filters to provide privacy, efficiency, scalability and content-based distribution

The subject-based subscription model requires that a multicast group be classified into subjects and the receivers will join a group based on the subject it is interested in. A content-based subscription model allows a node to filter information to only what it wants to receive within a subject-based group. This publish-subscribe model approach not only requires a classification method for multicast groups but it still has the problem of creating one multicast group for one subject matter. The creation of a new multicast group for every subject or content delivered to receivers is not a feasible option.

Filtering multicast communication is a method to reduce the number of multicast groups in the network. This is usually done through brokers that filter the packets before it reaches the receivers. There are three methods to filter multicast packets, namely precision reduction, transmit to all brokers and multi-hop routing [8].

The group approximation algorithms and network flooding are methods that reduce precision but improve multicast scalability by generalising members into large groups so that the number of groups can be reduced. This approach is not suitable for distributing sensitive information in a network.

In order to improve precision, many multicast groups with receivers interested in the same subject/content are clustered together under one or a collection of broker nodes. A copy of the same data packets will be sent to all the brokers, which in turn will multicast the packets to all the multicast groups under them.

Multi-hop routing is a hierarchical approach to the clustered multicast group method. The data source will send packets to brokers close to it, who in turn will forward the packets to brokers near to them, and so forth until all brokers receiver the packets. These brokers will then multicast the packets to the groups located under them.

A multicast routing filter that routes packets to interested receivers on the same multicast tree would reduce network wastage and scalability since only paths that contain members that want to receive the files will get the packet and no new multicast tree needs to be formed. Filters will also allow for better privacy since only interested parties will receive the communication instead of everyone in the group.

The simplest filter would be based on distance. A multicast packet could be restricted by either roundtrip delay or hop count so only members within the restricted range will receive the packet. This method is not exact as nodes outside the range will not receive the packets while nodes that do not want to receive these packets but are within the required range will receive them. This becomes even more difficult as the multicast tree topology will change as new members join and old members leave a multicast group.

A hierarchical label [9] filter provides a more exact method of filtering by giving a label to routers directly connected to a receiver node in a multicast tree. The label specifies the routers position on the multicast tree relative to a core router. Each router will build a hierarchical label tree by using its parents label as a prefix to its own label. A sender will address packets by the labels of the interested receivers and these packets will be routed along paths that connect these receivers to the sender. Such a scheme does not require applications to understand IP since a label is used to identify receivers. However, this method is limited to one-to-one communication and the labels will increase according to the depth and branches of the multicast tree. Levine and Garcia-Luna-Aceves [9] states that at least three bits are required for each integer that comprises a label.

In this paper, we are interested in network-based filters on a ShT multicast tree. We do not look at SBT since the nature of such a tree only allows one sender for a group. In an enterprise, a truly dynamic and collaborative environment will have multiple senders with multiple receivers. Since ShT multicast is an open group protocol, every member will receive the communication. Our proposed network layer architecture will ensure that packets are only sent to the subset of receivers that are interested in the communication and not to the rest of the nodes on the tree.

III. SHARED TREE MULTICAST LABEL FILTER

The proposed architecture described below is an extension of work previously done in [10, 11] for providing a label filter architecture for use by mobile nodes communicating through a shared multicast tree. The label filters improve scalability and network bandwidth for many corresponding nodes to communicate privately with a mobile node over a shared multicast tree. Due to space constrains, interested readers can view these papers for details about the original mobile multicast label filter architecture.

This paper proposes using a static label filter to route packets along the multicast tree to members that are authorised to receive this private subgroup communication.

While MPLS is a label switching protocol, the proposed architecture is completely independent from the MPLS architecture [12]. The proposed architecture works only in an IP-based network and can easily be implemented in the ex-

isting routers through the "Router Policy Engine". Thus, the real-time implementation of the proposed algorithm does not require any router upgrade. However, the MPLS architecture requires all participating routers to implement the MPLS protocol.

MPLS is inherently implemented as a unicast protocol although a proposal for MPLS in a multicast environment can be found in [13]. MPLS routes packets by switching the labels in the packet as it is transmitted over the network. The proposed label filter architecture only uses one static label to identify the path leading to all the recipients. No label switching is involved and thus no extensive table lookups are required.

The label filter architecture is limited to a multicast tree and cannot be used to route packets beyond the multicast tree. This reduces the complexity of label management while improving scalability since the labels are localised to one multicast tree.

The proposed label filter architecture can effectively utilise the allocated QoS of the existing multicast tree. The proposed subgroup can be constructed in linear time whereas constructing a new multicast tree with the prescribed QoS either through the conventional multicast shared tree protocol or using MPLS [13] takes quadratic time in terms of the message and time complexity. This fact is explained in [11].

In this paper, the nodes will initially join the multicast group using the core-based tree (CBT) [14] multicast routing protocol although the filter architecture can be used with any shared tree multicast protocol. In addition to a core router, the shared tree will also need a label manager/broker to manage any new communication requests by group members. This label manager can be collocated with the core (or rendezvous point) router or any other router on the tree. How this label manager is elected will not be covered in this paper. The main purpose for the label manager is to provide admission control for the member nodes that want to communicate privately. Once a request is approved, the manager will provide a label for the path used in the communication. The manager will also set any limitations to the communication like session duration and QoS requirements.

Fig. 1 shows the messages sent when setting up a label filter on the multicast tree. Like regular multicast control messages, the label filter control messages are segregated into messages sent to the end node (local membership) and messages sent among the multicast on-tree routers (label management). The segregation of the messages allows the end-node messages to piggy-back with existing multicast control messages.

The Comm-Init message is used to inform the first hop on-tree router that a node wants to communicate privately with one or more nodes on the same tree. The requested node indicates its approval or rejection to a communication request by sending a Comm-Resp message to its first hop on-tree (leaf) router. The Comm-Resp message is also used to send an acknowledgement once a label path is created. An on-tree router sends Comm-Info messages that inform the end nodes about any communication requests, label groups it is connected to, the decision of the label manager regarding a communication request and whether the label has been created successfully. The Comm-Qery message queries the leaf router for every public subgroup label address in the shared multicast tree.

The Comm-Reqt message is used to request the creation of a label group and inform nodes of the creation of the group. Source routing is used to send a Comm-Ackn message on the reverse path taken by the Comm-Reqt message. A Comm-Ackn message is sent to recipient nodes leaf routers to indicate permission to create or join the label group. The Comm-Labl message is used to create a label along the routers connecting all the nodes interested in filtering their communication. The Comm-Renw message is used to extend the duration of a label group while the Comm-Clse message is used to tear down the label group if the communication ends earlier than the expiration time.

Fig. 2 gives an example of a shared multicast tree with four node, i.e. nodes A, B, C and D where nodes A, B and C want to setup a new communication between themselves. The on-tree (OT) router OT1 is the multicast tree core and label manager. Node B is requesting the creation of a label where the label will be created from node B or the label manager depending on whether node B requests a public or private group label. In this paper, we will only show the creation and joining of nodes in a private group label. A private subgroup will only enable those that need to communicate with each other to join a subgroup, thus improving security, privacy and reduce information overload. Those that are not authorised to communicate in a subgroup will be prevented from doing so. The process for creating a public

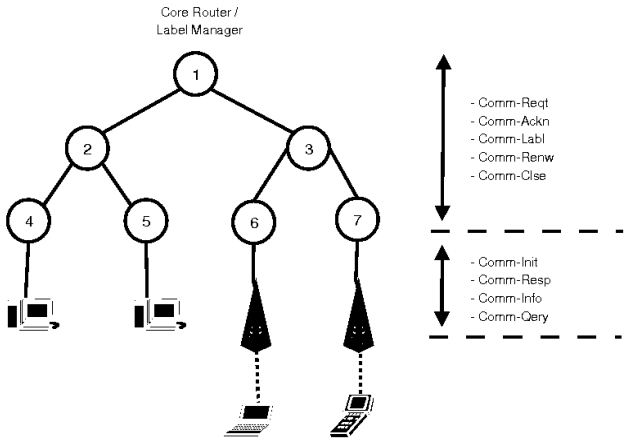

Fig. 1 Control messages used to setup a label filter

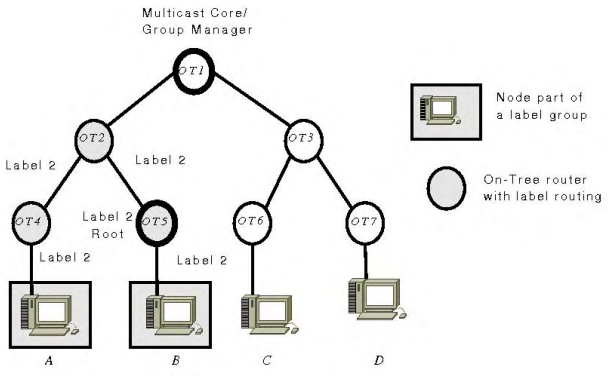

Fig. 2 Multicast and label groups

group label is shown in [15].

A. Creating Private Group Labels

The private group label operation is explained in this section. For illustrative purposes, a private subgroup label between node B and node A will be created.

When the initiating node has prior knowledge of the nodes joining the label group and wants to only allow selected nodes to join this group, the leaf router for the initiating node will be the central router or root for this label group. The label will first be created on this leaf router and travel outwards on every path connecting the root to all the other nodes interested in communicating using the label. The steps required for creating the label group are detailed below:

1. The initiating node will inform its leaf router to request the creation of a private label by sending a Comm-Init message to the leaf router. This message will contain the initiating node's address and the names of the intended recipients along with other information regarding the creation of the label group. The leaf router will multicast a Comm-Reqt packet to all leaf routers and the label manager to request permission from the label manager to create the private label and to find the path to the recipient nodes. In the example network in fig. 2, node B will send the Comm-Init message to router OT5 which in turn will multicast a Comm-Reqt message. This process is illustrated in fig. 3. The detailed algorithm to create a label filter is shown in fig. 4.
2. Once the label manager receives the Comm-Reqt message, it will check the tree policy to allow a creation of a private label group based on the request made by the initiating node. If the multicast group cannot allow the creation of a label group, the manager will send a Comm-Ackn message with the failure flag set on the reverse path taken by the Comm-Reqt message otherwise it will send a Comm-Ackn message with the success flag set. A successful Comm-Ackn message will contain the computer names of the approved parties for the subgroup communication, any limits on such a communication, the label (Label 2) to be used to forward the packets and a token which gives the originating leaf router authorisation to create the label in the multicast tree. Once the initiating node receives the Comm-Ackn from the label manager, it will wait until it receives a successful Comm-Ackn from the recipient nodes before creating the label. While the label manager is processing the Comm-Reqt message, every leaf router in the multicast tree will also receive the same Comm-Reqt message. The leaf routers will send a Comm-Info message to every node on the tree informing them about the request to create a label group. The nodes (nodes C, D) whose names are not listed in the Comm-Info message will drop the packet while each node that is listed will send a Comm-Resp message indicating whether it accepts or rejects the joining of this label group. The leaf router for node A will send a Comm-Ackn indicating interest in joining this subgroup back to the originating router (OT5) based on the value of the Comm-Resp message. If the originating node (node B) does not receive a Comm-Ackn from the manager or recipient nodes, it will multicast the Comm-Reqt message again. If the originating node gets a Comm-Ackn message indicating failure, it will do nothing further.
3. The leaf router (OT5) of the initiating node (node B) will create the private label approved by the manager from itself to the leaf routers (OT1) of all the recipients (node A) that accepted the label group request. The leaf router will send a Comm-Labl message on the reverse path used by the Comm-Ackn messages in step 2. The routers that receive the Comm-Labl message will create an entry in their label routing table. Each table entry records four values: the label and port number for incoming packets and the label and port number for outgoing packets. For security purposes, the label will only be created if the Comm-Labl message contains a token created by the la-

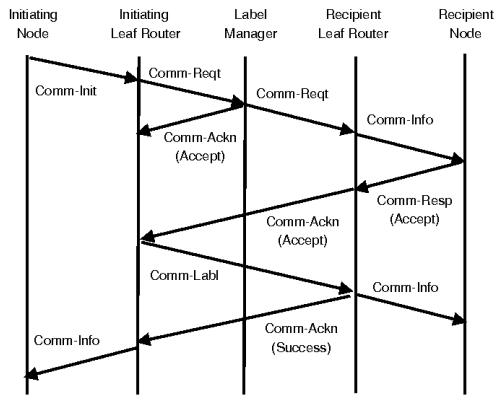

Fig. 3 Private Group Label creation process

```
1  initiating node multicasts label creation request on multicast shared tree;
2  switch (the Node receiving label creation request) do
3      case (Label Manager)
4          if (Tree.Resources == available) and (Tree.Policy == Enable) then
5              Generate Comm-Ackn (Success) with label and token;
6          else
7              Generate Comm-Ackn (Failure);
8          end
9          Transmit Comm-Ackn on reverse path of Comm-Reqt;
10     end
11     case (Requested node)
12         if (Node wants to join the label sub-tree) then
13             Generate Comm-Ackn (Success);
14         else
15             Generate Comm-Ackn (Failure);
16         end
17         Transmit Comm-Ackn on reverse path of Comm-Reqt;
18     end
19     otherwise
20         Route packet to next router;
21     end
22 end
23 foreach (Router on identified label path) do
24     if (received packet is Comm-Ackn (Success)) then
25         Create label soft state;
26         Set soft state timer;
27     end
28     if (timer ≤ expiry time) and (receives label creation token) then
29         Change soft state label to permanent virtual label;
30     else
31         Flush soft state label;
32     end
33     Transmit packet to next router on label path;
34 end
35 if (timer ≈ label session expiry) then
36     initiating node sends renewal message (Comm-Renw) to label manager;
37     if (Label Manager extends label session) then
38         initiating node updates routers with new session time;
39     else
40         label path is torn down;
41     end
42 end
43 if (initiating node ends communication prematurely) then
44     initiating node tears down label path (Comm-Clse);
45 end
```

Fig. 4 Algorithm to create a label filter path

bel manager.

4. Once the leaf router (OT1) of a recipient node (node A) creates the label, it will send a Comm-Info message to the node informing it about the successful creation of the label path. The leaf router will also send a Comm-Ackn message using the label to inform the originating leaf router (OT5) that the path label has been successfully created.
5. Any of the nodes (node A, B) in the private label group (label 2) can send a packet with the label so that only the nodes which are part of the subgroup will receive messages. A node will send an IP packet encapsulated with a link layer frame which uses the label for switching within the multicast tree. The end host will decapsulate this packet to receive the original IP packet. These nodes can still send a regular (not private) multicast packet to all members on the shared tree by not encapsulating the IP packet with a label.

B. Joining Private Group Labels

The interaction within and between enterprises is dynamic, where departmental units and team members join and leave one or more online groups. Fig. 5 shows the operations for a new node joining an existing private label group. We use the example from fig. 2 where node C joins the existing private label subgroup (label 2) to illustrate this process.

In the case of private subgroups, the interested node (node C) will not be able to query the network regarding existing private subgroups since this subgroup is private and information about it will not be published on the multicast tree. The node will need to know about the private subgroup label by some other means, such as communicating directly with the subgroup owner (node B). Once the node knows that the subgroup exists and finds out the subgroup label address (label 2), it will initiate a label join request as detailed below:

1. The joining node C will send a Comm-Init message to its leaf router (OT6) indicating it wants to join an existing label subgroup. The Comm-Init message will include the label address of the private subgroup the node wants to join and the address of the joining node. The leaf router will send a Comm-Reqt message to the label manager (OT1) to receive permission to join the subgroup.
2. The label manager will perform two tasks after receiving the Comm-Reqt from the joining leaf router. It will look up its label registry and forward the Comm-Reqt to the label subgroup's originating node (node B). The Comm-Reqt message will be sent to the originating node's leaf router which in turn will send a Comm-Info regarding the joining node's request to the originating node. While the label manager is awaiting the reply from the originating node, it will also check the tree policy to see if the joining node meets any policy requirements to join the label subgroup. One important requirements is the path QoS resources of the joining node if QoS is required for the private subgroup communication.
3. The originating node B may or may not allow the joining node to become a member of the label subgroup. This decision is sent to the originating node's leaf router (OT5) in a Comm-Resp packet. The leaf router will send a Comm-Ackn to the label manager. There are a few outcomes based on the decisions of the originating node and label manager which will allow the joining node to join the label subgroup:
   (a) Both the label manager and originating node allow the node to join the subgroup. The label manager will send a successful Comm-Ackn message with a temporary token for the joining node to create a label between itself and the existing label subgroup.
   (b) The originating node allows the joining of the subgroup but the joining node fails the policy requirements (such as QoS) for the subgroup. The label manager will send a Comm-Ackn message indicating the permission to join but it can do so only if the path meets the QoS requirements. The joining node will have to rejoin the multicast tree on another path that meets the QoS requirements. The joining node will have to send another Comm-Init to the leaf router and the leaf router will send a Comm-Reqt to the label manager to join the label subgroup. Once the new path meets the tree and subgroup policy, the label manager will send a successful Comm-Ackn with a temporary token to the leaf router for it to create a label to the subgroup.
   (c) The label manager allows the new join request but the originating node disallows the request. The label manager will send a negative Comm-Ackn to the originating node's leaf router. The leaf router will inform the joining node using a Comm-Info message regarding the unsuccessful request.
   (d) Both the label manager and originating node do not allow the new node to join the label subgroup. The label manager will inform the joining node of its failure to join the subgroup the same way as described above.

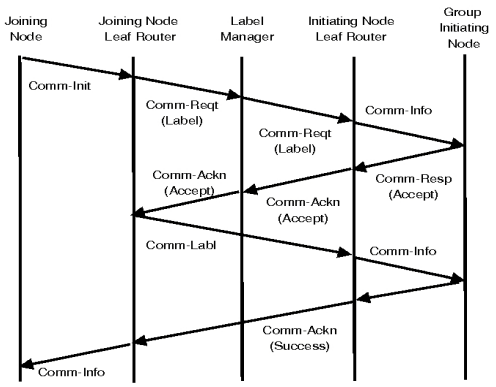

Fig. 5 Private group node joining process

IV. ANALYSIS

This section analyses the savings of network resources using the proposed label filter architecture. We have proven in [11] that creating a label filter subgroup uses less messages and is faster than creating a new CBT multicast tree.

The level of network traffic is used as an indicator of network resource usage and scalability for concurrent group

communication on a shared multicast tree. For the purposes of this paper, the transmission medium will not affect the results of the analysis.

The proposed label filter architecture was simulated on two network topologies consisting of 15 to 150 nodes using the NS-2 network simulator [16]. A subgroup of nodes in these networks use VoIP to communicate privately themselves. Fig. 6 measures the packets sent on a multicast tree with and without implementing label filtering. It shows the average number of packets transmitted on a shared multicast tree when 3 nodes communicate privately among themselves. Regular multicast uses 60.98% more packets compared with using a label filter. The label filter reduces the average packet duplication by 45.24% and maintains privacy among the participants at a network level.

Our experiments in fig. 6 and fig. 7 show that the number of packets sent on the multicast tree will increase significantly on a shared multicast tree without a label filter. This is because a multicast protocol will distribute the packets on all paths on the tree until all nodes attached to the tree receives the packets. The number of packets sent on the multicast tree is proportional to the number of senders and the size of the multicast tree as shown in fig. 7.

With the label filter, the sender nodes will transmit packets only to the receiver nodes instead on all network paths on the multicast tree. In the case of multiple source nodes, the number of packets sent on the multicast tree will increase since the number of source nodes has increased. However, the total number of packets transmitted over the multicast tree when using a label filter is reduced by as much as 36.79% to 75.48%, with a 95% CI. This reduction in network traffic is important since the network resources can be utilised for other connections in addition to providing private communication between different subgroup parties on a shared multicast tree.

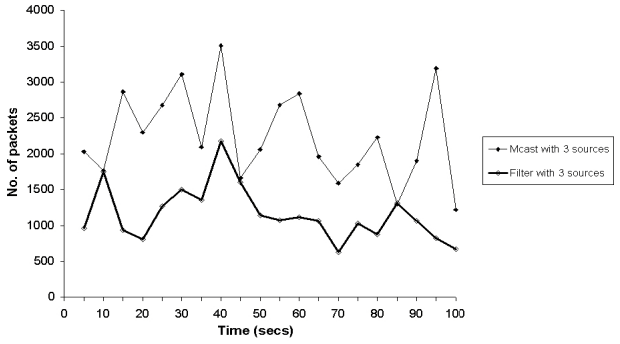

Fig. 6 Average number of packets transmitted by 3 senders to members in a private subgroup in a 15 member multicast tree.

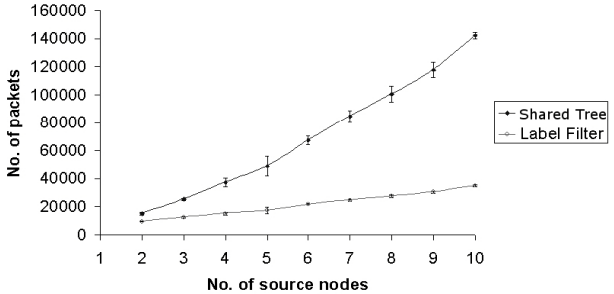

Fig. 7 Average number of VoIP packets transmitted among many source nodes in a private subgroup

The initial simulation results show that the proposed label filter architecture reduces network bandwidth usage and provides privacy during subgroup communication in an enterprise multicast tree. The label filter architecture can provide both subject-based and content-based filtering as a subgroup can be formed to cater for both instances. In future papers, this work will be extended to compare the bandwidth usage of the proposed label filter architecture against other publish-subscribe systems.

V. Conclusion

This paper presents a new scalable network filter architecture using labels on a multicast shared tree. This allows a member node on the tree to perform one-to-one, one-to-many or many-to-many private communications with a subset of nodes within the same group. The proposed label filter architecture provides more efficient group communication by reusing an existing multicast shared tree for private subgroup communication, hence improving scalability by reducing the number of IP multicast addresses used. Other publish-subscribe architectures using multicast technology are not as scalable as they still aim to connect multiple multicast trees using a broker middleware. The proposed architecture also provides better use of available bandwidth and network resources like QoS as shown by the simulated experiments. This will enable multicast communication to be more widely deployed in the current Internet environment for supporting group collaboration, particularly in DES environments, without high network wastage.

VI. References

[1] G. Banavar, T. Chandra, B. Mukherjee, J. Nagarajarao, R. E. Strom, and D. C. Sturman, "An efficient multicast protocol for content-based publish-subscribe systems," in *Proc 19th IEEE Intl. Conf. on Distributed Computing Systems*, 1999, pp. 262-272.

[2] M. Handley, I. Kouvelas, T. Speakman, and L. Vicisano, "Bidirectional protocol independent multicast (BIDIR-PIM)," IETF, RFC 5015, October 2007.

[3] D. Meyer and P. Lothberg, "GLOP addressing in 233/8," IETF, RFC 3180, September 2001.

[4] L. Wei and D. Estrin, "The trade-offs of multicast trees and algorithms," in *Proc. 4th Intl. Conf. on Computer Comm. and Netw.*, 1995, pp. 150-157.

[5] P. Paul and S. V. Raghavan, "Survey of multicast routing algorithms and protocols," in *Proc. 15th Intl. Conf. on Computer Communication*, India, 2002.

[6] B. Quinn and K. Almeroth, "IP multicast applications: Challenges and solutions," IETF, RFC 3170, September 2001.

[7] S. Langerman, S. Lodha, and R. Shah, "Algorithms for efficient filtering in content-based multicast," in *Proc. 9th Annual European Symposium Algorithms*, Denmark, 2001, pp. 428-439.

[8] L. Opyrchal, M. Astley, J. Auerbach, G. Banavar, R. Strom, and D. Sturman, "Exploiting IP multicast in content-based publish-subscribe systems," in *Proc. IFIP/ACM Intl. Conf. on Distributed Syst. Platforms*, USA, 2000, pp. 185-207.

[9] B. N. Levine and J. J. G. Luna-Aceves, "Improving internet multicast with routing labels," in *Proc. Intl. Conf. on Network Protocols*, 1997, pp. 241-250.

[10] J. Singh, P. Veeraraghavan, and S. Singh, "Implementing label filters on a shared tree mobile multicast architecture," in *Proc. IEEE Intl. Conf. on Netw.*, Malaysia, 2005.

[11] J. Singh, P. Veeraraghavan, and S. Singh, "Performance of a shared tree multicast label filter architecture," in *Proc.IEEE Intl. Conf. on Netw.*,Malaysia, 2005.

[12] U. Black, *MPLS and label switching networks*, 2nd ed. New Jersey, USA: Prentice Hall, 2002.

[13] D. Ooms, B. Sales, W. Livens, A. Acharya, F. Griffoul, and F. Ansari, "Overview of IP multicast in a multi-protocol label switching (MPLS) environment," IETF, RFC 3353, August 2002.

[14] A. Ballardie, "Core based trees (CBT version 2) multicast routing: Protocol specification," IETF, RFC 2189, September 1997.

[15] J. Singh, P. Veeraraghavan, and S. Singh, "Network based filtering architecture on a shared multicast tree," in *Proc. IASTED Intl. Conf. on Networks and Communication Systems*, Thailand, 2006, pp. 390-395.

[16] The Vint Project, "The ns network simulator," Available from: http://nsnam.isi.edu/nsnam/index.php [Accessed 12 January 2009].