

Multicast Support for Collaborative Networking

L. A. Flynn
Bell Laboratories
Lucent Technologies
Whippany, NJ 07981
laflynn@lucent.com

H. P. Dommel
Computer Engineering Department
Santa Clara University
Santa Clara, CA 95053
hpdommel@scu.edu

ABSTRACT

Collaboration infrastructures are gradually coming of age, supporting real-time audio and video with a mixture of network core and edge based techniques to enable large-scale, multi-party communication. User participation in collaborative activities requires mechanisms to support dynamic group membership and maintenance of perpetually changing delivery topologies. In the past, multicast support has been studied within testbeds such as the MBone, and across overlay architectures. In this paper we tackle the problem of how network-native multicast can help to orchestrate the formation of virtual teams in group collaboration. We argue that user interaction can be understood as a series of constraint-driven information exchanges, for example by sharing objects based on access rights. A novel multicast protocol, Limited Core Fix (LCF), is introduced to efficiently maintain multicast delivery trees among distributed work teams. LCF is a receiver-centric mechanism to support qualification-based multicast routing and performs more efficiently than comparable existing protocols.

KEYWORDS: Multicast-supported collaboration, QoS multicast performance, tree repair protocols.

1. INTRODUCTION

A new generation of networked multimedia applications has emerged in recent years, bringing Internet radio, VPN videocasts [2], Video on demand and Internet television to end users. Along with such applications, collaborative networking fosters workgroup interactivity at increasing scale and is deemed to generate the next wave of “killer apps”. Multicast deployments in the MBone [14] and more recently in the Internet 2 [1] have lead to more sophisticated one-to-many communication protocols, whose primary strength is the significant reduction in bandwidth waste currently incurred through unicast transmissions. While the generality and complexity of MBone-type multicast has been an obstacle to wider acceptance and the prolif-

eration of group-aware applications, the lower price point of investments in large-scale multicast-capable routing infrastructures makes group communication services economically viable in relation to saved bandwidth from the use of multicast [8]. Although IP multicast per se has not been as widely deployed as expected, experimental developments in overlay networks [6] or webcasts [16] have spurred continued interest in the use of multicasting for various types of applications. Ultimately, the success of multicast will depend on the availability and acceptance of multi-party applications, whose traffic volume and Quality-of-Service (QoS) requirements demand more sophisticated transmission models than provided by the current IP infrastructure.

Multicast has been studied in many variants of link layer, routing (e.g., [7, 9]), reliable transport, and application-level protocols. The field of QoS-supported multicast is of particular interest to service providers and distributors of multimedia content or secure connectivity. Effective and flexible multicast is highly desirable for collaborative environments, where the number of simultaneous participants and applications fluctuates greatly, requiring loose consistency and weak group membership, absence of central servers, and global scalability through sub-partitioning and on-demand formation of transmission topologies [15].

The idea of *qualified multicast* [11] follows this premise of mapping QoS-routing for multicast into a qualification model, where user privileges, security levels or pay-per-view transmissions can be traced through qualifiers to enable transmission for some users and disable it for others. We define a qualified network as one in which only a subset of routers and links are permitted to handle specific “qualified” types of packets. Qualification is expressed through the evaluation of qualifiers in routers. The qualifier may be a security level or different attribute which determines if traffic may flow through a particular node. Qualifiers can be based on general metrics, traditional QoS parameters such as sampling resolution, latency, jitter, and other transmission characteristics, or they can express billing, security [12], or other application-relevant criteria. The formation and maintenance of qualified multicast trees repre-

sent a particular challenge in terms of routing and repair, since criteria for nodes to be on-tree or off-tree are both a function of group membership and the fulfillment of one or more qualifiers.

Such selective quality-oriented communication is especially desirable for collaborative work and interactive group communication, where many users cojoin in virtual teams for interspersed and intermittent shared activities. The contribution of this paper is a study on how multicast routing protocols can be used to solve this conditional distribution problem. In particular, we compare the efficiency or qualification-based routing protocols, with special regard for tree build and repair in terms of messaging overhead and flexibility to adapt the delivery infrastructure to changing receiver sets. We introduce the *Limited Core Fix* (LCF) multicast protocol which uses a novel depth-first-search technique to find new routes to a multicast source, in contrast to on-demand multicast protocols such as MAODV [3] and QoS MIC [17], which operate based on expanding ring search. LCF employs a receiver-initiated multicast repair method which uses a minimum of bandwidth in order to repair dense-mode qualified multicast routing trees, and can also be used to build multicast trees in sparse-mode.

The remainder of this paper is structured as follows. Section 2 reviews related work. Section 3 presents the inner workings of the LCF protocol. Section 4 presents a comparative analysis of LCF versus other on-demand multicast protocols. Section 5 concludes the paper and outlines future work.

2. BACKGROUND

Various protocols such as MAODV (Multicast Ad-Hoc On Demand Distance Vector), DVMRP (Distance-Vector Multicast Routing Protocol), FGMP (Forwarding Group Multicast Protocol), or ODMRP (On Demand Multicast Routing Protocol) have been introduced in the past to achieve efficient multicast routing for dynamic topologies [5]. A corresponding body of work exists for QoS-based multicast routing [4, 17].

Most existing multicast routing protocols waste significant bandwidth by requiring periodic dissemination of control messages to keep track of multicast routes and group membership. In contrast, MAODV aims to minimize messaging overhead by performing route discovery on-demand, mainly to due its target operation in ad hoc networks.

MAODV is a multicast extension of AODV (Ad-hoc On-Demand Distance Vector) and operates in a receiver-initiated manner, fabricating and maintaining a multicast

tree based on hard state. A multicast routing request is broadcast similar to the unicast route request, and the route reply propagates back from the nodes that are members of the multicast group.

A node that does not have a route and wishes to join a group broadcasts a route request (RREQ) with a limited Time-To-Live (*TTL*) as its hop radius. If the RREQ does not receive a response, it is repeated with a larger *TTL*. Nodes which have a routing entry for the multicast group respond with a reply (RREP) which indicates that particular node's distance from the multicast group leader. The joining node chooses the best RREP among the bids it receives, and sends a JOIN message along the chosen path. Nodes receiving the JOIN packet become part of the multicast forwarding tree.

In case of group reformation or link failures, MAODV repairs are initiated by the node directly below a link break. From the viewpoint of qualified multicasting this node is located below the node which has become unqualified. This node sends a RREQ marked with the last multicast route sequence number and the last hopcount to the multicast head before the break. The repair RREQ is also sent in an expanding ring search with increasing *TTL*. If the node does not receive a RREP within *RREQ_RETRIES* then a RREQ is sent with a *TTL* equal to the network diameter. MAODV uses the direct descendant of the new off-tree node as the node which performs repairs, as opposed to the stranded receiving nodes commencing repairs for themselves. Note that there may be a subtree below this node, and the repairing node itself may not be a multicast receiver. The MAODV repair path is required to go through this node, irrespective of whether the node continues to be an effective head of a subtree. Since by rule repairs do not commence from the orphans, it is possible that repeated expanding ring searches from a far-off node lead to long times for repairing the tree. In contrast, if a search starts directly from the perimeter of the nodes which need to rejoin, repairs could be much quicker. MAODV hence adds latency to repairs when not initiating them from the nodes which actually need to be repaired. The MAODV Internet draft describes the operation of MAODV in more detail [3].

The primary reason why we chose to compare LCF to MAODV is based on the fact that both protocols support and sustain the formation and reformation of multicast groups in intermittent or rapidly changing communication topologies. In addition, ad-hoc routing scenarios incur similar QoS issues [4], and collaboration across wireless links is of growing interest. Although we have conducted more extensive analyses and comparative simulations with other protocols, for the sake of space we limit our presentation here to a discussion of LCF vs. MAODV as a representative on-demand protocol.

N	→	Number of nodes in network.
PATH	→	Path request control packet.
F-PATH	→	Path-found control packet.
NACK	→	Negative acknowledgment (no path found).
ACK	→	Control packet acknowledgement.
T_x	→	Multicast routing table (T) for node x .
$g_{p:l:ts}^{a:m}$	→	Multicast entry (M) in routing table as tuple g, a, m, p, l, ts .
S_x	→	Multicast search table (S) for node x .
$s_{r:i:n}^{g:q:nq:ts}$	→	Search Entry (s) in the multicast routing table as tuple (g, q, nq, ts, r, i, n) .
l_u^v	→	Link (L) from u to v .
L.qualify	→	Qualification of the link with regards to some qualifier: 1 if both nodes are qualified and 0 otherwise.
M.seqNo	→	Sequence number (timestamp) of last F-PATH control packet received for the group.
M.parent	→	Current multicast parent as set in routing table.
A	→	Set consisting of instantaneous collection $\forall T_r, \forall V \in G$, i.e. state of all multicast routing tables within the network.
$A.path_g^n$	→	Path formed by following M.parent pointers for group g from node n using $T \in A$.
Orphan	→	Node o which presently has hosts attached wishing to receive multicast g AND a current $A.path_g^o$ which does not end at the multicast source AND node had a $M.parent$ in the past.
Source	→	Source of the multicast, or node which multicast sources forward their transmission to. Synonymous with core.

Table 1: LCF Nomenclature.

3. LIMITED CORE FIX MULTICAST

The innovative claim in LCF is that repairs are requested in a limited sequential search towards the core, while routing loops are prevented by requiring on-tree nodes to maintain a pathlist to the core. Table 1 summarizes the terms used in the description of LCF.

3.1 Stored Information

Each router x in LCF maintains a qualified multicast routing table T_x where each entry $g_{p:l:ts}^{a:n}$ is a tuple $[g, a, m, p, l, ts]$, with g denoting the group, p as the multicast parent (reverse next-hop to source), a list l of neighbors except for p which have the qualifier for the group and have not sent a prune since the last flood, a timestamp ts , an associated qualifier a , and a multicast group identifier m , which indicates whether this node has hosts which have joined the respective multicast group. Each router maintains one qualifier table per neighbor with a list of applicable qualifiers for

that neighbor. Each router x also maintains a multicast routing search table S_x where each entry $s_{r:i:n}^{g:q:nq:ts}$ is stored as a tuple $[g, q, nq, ts, r, i, n]$, where q is again at least one qualifier, nq denotes the neighbors queried, ts is the requester timestamp, r represents neighbor replies, i is the request originator Id, and n is the neighbor node to reply to.

3.2 Operation

Periodically exchanged qualification information among neighboring routers is used to update the neighbor qualification table. On-tree nodes have a list of all nodes on their reverse on-tree path to the core. Off-tree nodes simply have a standard unicast table irrespective of qualification status, plus a neighbor qualification table with information about qualification status of direct neighbors only.

LCF control packets (PATH, F-PATH, and NACK) are sent in a reliable manner over each link. The control packets are acknowledged by receiving routers. If the packet is not acknowledged within a timeout then the packet is resent. If the original path-requester does not receive a reply (F-PATH or NACK) within a timeout period then it resends the path-request to that neighbor. The on-tree node appends its own stored path to the core of path listed in the request packet, and returns a path-found (F-PATH) packet to its neighbor. The F-PATH packet is forwarded along the new tree path as listed in the packet itself, and as it is forwarded, the nodes make entries in their multicast routing table. At the time the original joining node receives the F-PATH, the on-tree path to the joining node has been constructed.

All nodes performing a search make an entry in their search table. If the request packet reaches a node with no other qualified neighbors, a negative acknowledgement (NACK) is returned to the requestor. If a node receives a NACK, it sends a path request to the qualified neighbor which has not yet been queried and has the shortest standard unicast path. If no unqueried qualified neighbor remains, this node sends a NACK to the neighbor which sent the original request. The request to join the tree only fails if the original node receives a NACK from each of its qualified neighbors. The join request succeeds if it reaches an on-tree node which includes the core.

When a neighbor is found unqualified, the multicast routing table is modified so that this neighbor is no longer a multicast parent or child. If a node is connected to a link or it is a direct descendant of a node no longer qualified or working, it must notify the subtree below using a reliable subcast. Such *orphaned* nodes send a path-request to their qualified neighbor with the shortest standard unicast path to the core.

3.3 Cooperation with two tree build methods

The Limited Core Fix method is able to work on trees built using either core supervision or LCF joins. Both build methods create a pathlist within on-tree routers.

3.3.1 Core-supervised tree build

A core-supervised tree build is performed if:

- The multicast core knows all joining nodes.
- The multicast core can make nodes qualified OR the multicast core has a qualification-based topology map.
- The multicast core has a unicast routing table.

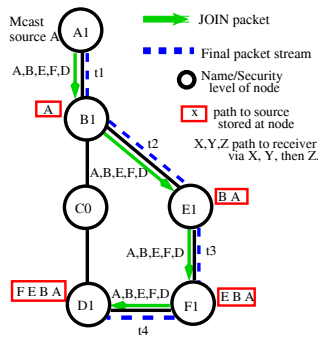


Figure 1: Core-supervised tree build.

Fig. 1 shows a scenario where the multicast core sends CAN-JOIN packets along a known path to each of the joining nodes, which causes nodes along the path to become part of the multicast forwarding tree and to store the multicast entry including the reverse path to the core. This illustration is an example of a core-supervised tree build, where all nodes on the multicast tree from receiver node D to multicast source A have stored their path to A.

The situation is analogous if the multicast operates for example with encrypted messages. Nodes apply directly for encryption or decryption keys from the multicast source, thus notifying the source of their identity [12]. While giving out the keys, the source gives a qualifier to the node. A multicast core can give a key to any trusted router and encrypt the data stream so that only nodes which have the correct current keys can decrypt the current data stream. If the core has a standard topology map irrespective of qualification status and trusts a subset of those routers, it is able to send CAN-JOIN packets along those paths to each of the joining nodes. The CAN-JOIN packet causes nodes along the path to store the multicast entry including the reverse path to core and encryption or decryption keys.

3.3.2 LCF sparse mode tree build

If each joining node performs an LCF join, a complete multicast tree can be built in a sparse-mode build style using LCF joins. Sparse-mode joins do not depend on floods and prunes. Rather, the multicast receivers initiate joins along a reverse path to the multicast core [10]. An LCF-only sparse mode tree build would leave exactly only the information necessary at each node for performing an LCF tree repair.

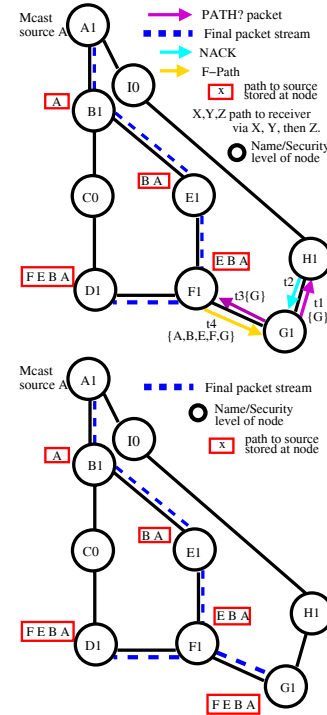


Figure 2: LCF sparse mode snapshot before and after join.

Fig. 2 shows an LCF sparse-mode join by node G before and after the join process. A node attempting to join the tree sends a path-request (PATH) to the qualified neighbor with the shortest standard unicast path to the core. Intermediate nodes follow the same search methodology, and add their own id (IP address) to the request pathlist when they forward the request. In the upper figure, node G's unicast table is qualification unaware. Hence node H is the first neighbor which the PATH packet is sent to, since H is on the shortest unicast path to the multicast source A.

However, H has no other qualified neighbors and therefore H returns a NACK. Next, node G tries node F as its second shortest unicast next-hop to the multicast source. Node F has a path to the multicast source stored, and sends the path in the F-PATH packet returned to G. Now, F starts to forward the multicast stream to G, and G stores its complete path to the source. The lower figure shows the resulting new

tree, with a path to the source in a memory structure at node G.

4. PERFORMANCE COMPARISON

4.1 Analytical Discussion

Since LCF is a receiver-initiated multicast method suitable for qualified multicast, we compare the performance of LCF to other sparse-mode multicast methods. LCF operates as a distributed multicast routing protocol, similarly to MAODV and QoSMIC, and as such it requires a small amount of memory and state to be stored at nodes participating in the protocol. Formation of receiver-initiated qualified multicast trees is similar to on-demand protocols in that a unicast reverse path to the multicast source is not known at the time the receiver initiates the join.

We compare the expanding ring search used in the MAODV on-demand protocol to the linear search of LCF, exploring network behavior based on the average shortest number of hops from any node to a node joining the multicast. In particular, we focus on join latency and bandwidth used. Table 2 compares the radius scope during expanding search. MAODV halts the search if the $radius_{35}$ search fails.

Radius	MAODV	Exponential
$radius_1$	1	1
$radius_2$	2	2
$radius_3$	4	4
$radius_4$	35	8
$radius_5$		16
$radius_6$		32
$radius_7$		64
$radius_8$		128
$radius_9$		256
$radius_{10}$		512
$radius_{11}$		1024

Table 2: Exponential radius with expanding radius search.

Table 3 shows how bandwidth used by MAODV was calculated, for an average distance of n links to reach a neighbor joined to the multicast and an average number of neighbors x . Due to MAODV's radial search, unsuccessful paths are traversed to their maximum search length on each path. If all paths for a given MAODV search radius are unsuccessful, then the MAODV search recommences with a larger radius. Therefore, note we count a cumulative bandwidth for larger than minimum radius MAODV searches. Here we count each link hop as one unit of bandwidth.

In general, with a uniformly distributed network, an average of $1/n$ nodes that are multicast members, and the average number of qualified neighbors x , then the radial searches are expected to reach the tree when the radius reaches a size of at least $\log_x(n)$. LCF takes an average of n link hops before its search hits the multicast tree. The expectation for bandwidth used by LCF is $n * 2$. Table 4 shows the latency

$$\begin{aligned}
 & \text{if}(x^{radius_1} \geq n) \\
 & \quad \text{thisBandwidth} = \sum_{i=1}^{radius_1} 2x * (1^i) \\
 & \text{elseif}(x^{radius_2} \geq n) \\
 & \quad \text{thisBandwidth} = \sum_{i=1}^{radius_2} 2x * (1^i) + \\
 & \quad \sum_{i=1}^{radius_1} 2x * (1^i) \\
 & \text{elseif}(x^{radius_3} \geq n) \\
 & \quad \text{thisBandwidth} = \sum_{i=1}^{radius_3} 2x * (1^i) + \\
 & \quad \sum_{i=1}^{radius_2} 2x * (1^i) + \sum_{i=1}^{radius_1} 2x * (1^i)
 \end{aligned}$$

Table 3: Bandwidth calculations for expanding ring search.

for the expanding radius search, where queuing latency is Q_l and average link latency is L_l .

$$\begin{aligned}
 & \text{[Perform with increasing radii:]} \\
 & \text{if } x^{radius_1} \geq n, \\
 & \quad \text{thisLatency} = (Q_l + L_l) * radius_1 * 2, \\
 & \text{elseif } x^{radius_2} \geq n, \\
 & \quad \text{thisLatency} = (Q_l + L_l) * radius_1 * 2 + \\
 & \quad (Q_l + L_l) * radius_2 * 2, \\
 & \text{elseif } x^{radius_3} \geq n \\
 & \quad \text{thisLatency} = (Q_l + L_l) * radius_1 * 2 + \\
 & \quad (Q_l + L_l) * radius_2 * 2 + \\
 & \quad (Q_l + L_l) * radius_3 * 2, \\
 & \text{elseif } x^{radius_4} \geq n, \\
 & \quad \text{thisLatency} = (Q_l + L_l) * radius_1 * 2 + \\
 & \quad (Q_l + L_l) * radius_2 * 2 + \\
 & \quad (Q_l + L_l) * radius_3 * 2 + \\
 & \quad (Q_l + L_l) * radius_4 * 2,
 \end{aligned}$$

Table 4: Latency calculations for expanding ring search.

In contrast, LCF performs a receiver-initiated search along only one path at a time. LCF searches linearly, which is very different from the expanding ring search with TTL limiting, which is 'circular' with a radius, as used in MAODV and QoSMIC. The linear approach of LCF directly trades latency of tree joins for very low bandwidth usage. For joining from a separated node, the LCF method makes a depth-first-search as opposed to a breadth-first-search as used in MAODV and QoSMIC. The benefit of using LCF is very low bandwidth usage. The cost of the QoSMIC and MAODV method would be a lower time-to-repair if the QoSMIC search radius were set to infinity, however searches are limited and repeat searches expand the radius using the QoSMIC method.

Figure 3 compares the latency of LCF to MAODV expanding ring search for a significant set of network scenarios. It shows that the expected latency of node join time versus the inverse of the average ratio of nodes joined to the multicast in the network generally results in a better latency for LCF. Negative latency means that the MAODV protocol stopped the search without locating the tree. Obviously the higher the ratio of joined nodes, the lower the join latency, all else equal. LCF has an absolute lower latency than the expanding ring search when the average number of qualified neighbors is 1. For cases such that the average multicast membership is 1 in 10 or greater, latency is clearly similar between methods even with higher average numbers of quali-

fied neighbors of 2, 3, and 4. Due to the exponential growth of the expanding ring search, and the fact that sequential searches increase latency cumulatively, it is interesting to note that within the joined ratios of 1 : 5 to 1 : 35, the expanding ring method for nodes with one or two neighbors has the same latency as LCF. However, LCF uses a higher latency in all cases but these. For average neighbors being one, when the membership is between 1 : 38 and 1 : 100 and lower, MAODV fails to rejoin the orphaned node to the tree.

Figures 3 and 4 show similar results, with the difference that for average neighbor being 1, the exponential expanding radius algorithm does not fail. Clearly a tradeoff for the extremely low bandwidth use of LCF is higher latency than exponential expanding ring search, unless the average number of qualified neighbors is 1 or the average multicast membership is 1 in 10 or greater.

The pathlength is expected to be n long for LCF, since $1/n$ is frequency of node membership in the multicast group. The expectation for the LCF latency can be trivially shown to be $N * (Q_l + L_l) * 2$, where Q_l is the queuing latency and L_l is the average link latency. For simpler calculation and since it does not make a difference for comparison purposes, we set $(Q_l + L_l) = 1$. The expected radius to find a multicast tree member using MAODV is $radius \geq \log_x(n)$. The bandwidth used for the MAODV process is x^{radius} , or $x^{radius(I)} + x^{radius(II)} + x^{radius(III)} + \dots$ etc., up until where $x^{radius(MMM)} \geq n$. Using MAODV, if the $radius_1 < \log_x(n)$ then it is expected to require multiple searches, and if the next search has $radius_2 \geq \log_x(n)$ then we expect to find a node on the multicast tree. The LCF depth-first-search has an expected bandwidth usage, or link traversal, of n before reaching the multicast tree, no matter what value x or n has.

Figure 5 shows that LCF uses very low bandwidth compared to exponential expanding ring search. This disparity becomes more pronounced as the average number of qualified neighbors increases. Figure 6 shows a closeup of the part of the graph depicting network membership in the multicast between 1:1 and 1:10. Spikes in the MAODV bandwidth happen when the TTL is raised to 35, causing a huge increase in the packet traffic. MAODV stops the search if it fails a radius-35 search.

4.2 Simulations

The purpose of this performance evaluation was to determine the effect on protocol performance of two values: average number of qualified neighbors, and average ratio of nodes part of the multicast. We calculated latency for nodes joining to the tree, and also bandwidth used for the process. Simulated network sizes varied from 10 to 100 node networks, increasing by tens. The percentage of nodes re-

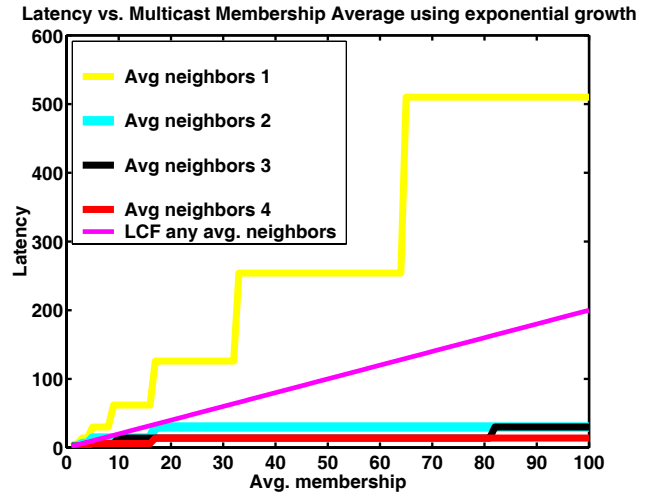


Figure 3: Latency of MAODV vs. LCF. Membership varies from 1:1 to 1:100. Average number of neighbors varies between 1 and 4.

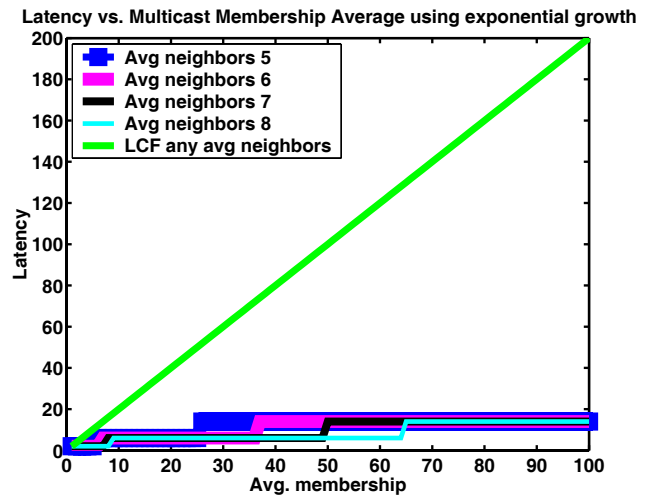


Figure 4: Latency of MAODV vs. LCF. Membership varies from 1:1 to 1:100. Average number of neighbors varies between 5 and 8.

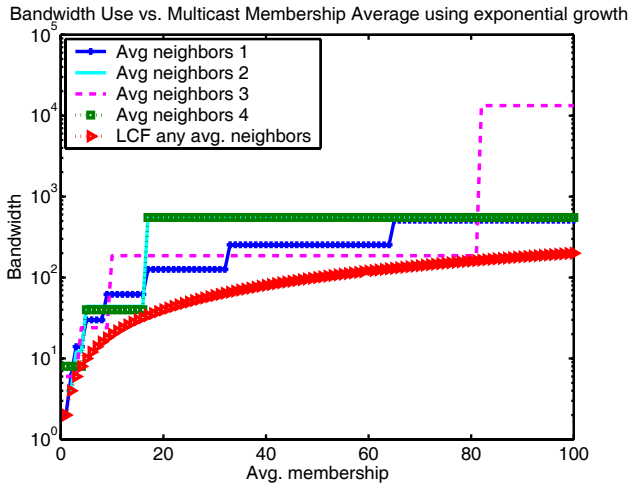


Figure 5: Bandwidth usage of MAODV vs. LCF. Membership varies from 1:1 to 1:100.

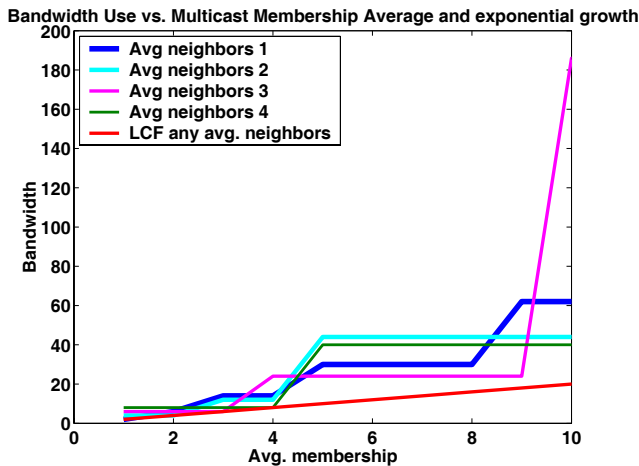


Figure 6: Bandwidth usage of MAODV vs. LCF. Membership varies from 1:1 to 1:10.

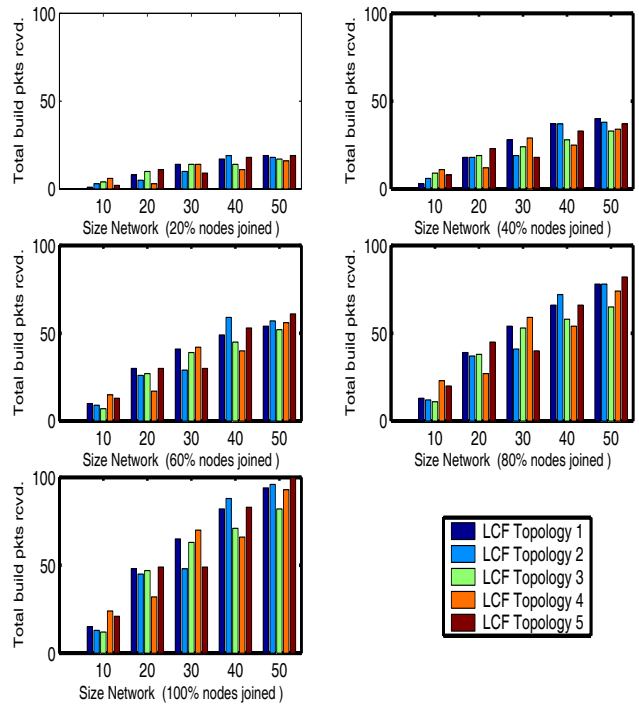


Figure 7: Total number of build packets received by all nodes

ceiving a multicast varied between 20, 40, 60, 80, and 100 percent of all network nodes. For each size of network, 5 different topologies were randomly generated using the Georgia Tech Internetwork Topology Model (GT-ITM) software [18]. For each size of network from 10 to 100 nodes, increasing by 10, 5 topology graphs were created. A total of 50 different topologies were used, created by the GT-ITM software. Links had randomly assigned bandwidth using a flat random graph with a geometric random distribution for generating edges. The multicast tree build and repair simulations ran for times of 11.75 and 50.0 seconds on the ns2 [13] clock. Multicast traffic was constant bit rate (CBR). All nodes were considered qualified, since networks where only N nodes are qualified can be considered to be a network of size N nodes as far as any of the protocols work. (None of these protocols send packets over any unqualified links, and thus no extra time or bandwidth would be required for any additional unqualified nodes existing in the network.) The choice of node which became disqualified was uniform random among joined nodes.

Figure 7 shows the total number of LCF control packets required to build the initial multicast tree before repair. This figure shows effects due to varying network configurations and for varying ratios of multicast membership. As the percentage joined grew larger, the total bandwidth used for building the LCF tree increases. Note that only approxi-

mately 2 packets per node are required to build the tree, in the worst case. Simulations also showed that LCF consumes very low bandwidth in the number of control packets sent.

5. CONCLUSIONS AND FUTURE WORK

Reflecting QoS natively in on demand routing computations and ultimately in more adaptive and intelligent distribution of content allows for new models of pricing and billing, establishment of secure communication channels, or differentiated multicast in collaboration. The choice of qualifiers may be made by providers, users or by software agents in applications. We have introduced a novel method to efficiently repair multicast routing trees, which is of particular interest for multimedia collaboration, where users group together ad hoc and at varying scale. The LCF protocol implements such tree repair using very low bandwidth independent of a particular tree topology, given that a pathlist exists within on-tree nodes. A performance analysis and simulation showed that LCF compares favorably to on-demand protocols in varying topologies, and always displays superior performance when a distance-vector or topology-map unicast table is used.

To our knowledge, few studies have been undertaken to illuminate the interplay between multicast routing using qualifiers and collaborative networking. LCF is scalable and well suited for collaboration infrastructures, where users aggregate dynamically in virtual teams and require group communication adhering to specific qualification constraints. Future work entails the development of a hierarchical version of LCF for greater scalability and introduction of strata in the formation of group communication trees. A hybrid protocol using flooding for periodic dense-mode tree repairs and LCF for intermittent repair between floods is also under development. We surmise that it is only a matter of time for the right combination of multicast support and popular interactive applications to emerge, before one-to-many and many-to-many communication models find their ultimate raison d'être.

REFERENCES

- [1] K. Almeroth. The Evolution of Multicast: From the Mbone to Interdomain Multicast to Internet2 Deployment. *IEEE Network*, Jan./Feb. 2000.
- [2] W. Arbaugh, J. R. Davin, D. J. Farber, J. M. Smith. Security for Virtual Private Intranets. *IEEE Computer*, V.31, N.9, 48–54, Sept. 1998.
- [3] E. M. Belding-Royer and C. Perkins. Multicast Operation of the Ad-Hoc On-Demand Distance Vector Routing Protocol. In *Proc. MOBICOM*, 207–218, 1999.
- [4] S. Chakrabarti and A. Mishra. QoS Issues in Ad Hoc Networks. *IEEE Communications Magazine*, 142–148, Feb. 2001.
- [5] X. Chen and J. Wu. Multicasting Techniques in mobile Ad hoc Networks. In *The Handbook of Ad hoc Wireless Networks*, CRC Press, 25–40, 2003.
- [6] Y. Chu, S. G. Rao, S. Seshan, and H. Zhang, Enabling Conferencing Applications on the Internet using an Overlay Multicast Architecture. In *Proc. ACM SIGCOMM*, San Diego, CA, Aug. 2001.
- [7] S. Deering and et al. Protocol Independent Multicast Version 2, Dense Mode Specification. *Internet Draft draft-ietf-idmr-pim-dm-05.txt*, May 1997.
- [8] C. Diot, B.N. Levine, B. Lyles, H. Kassem, and D. Balensiefen. Deployment Issues for the IP Multicast Service and Architecture. *IEEE Network*, V.14, N.1, 88–98, Jan. 2000.
- [9] D. Estrin and et al. Protocol Independent Multicast Version 2, Dense Mode Protocol Specification. *Internet draft draft-ietf-idmr-pim-dm-06.txt*, August, 1997.
- [10] D. Estrin and et al. Protocol Independent Multicast - Sparse Mode (PIM-SM) Protocol Specification. *RFC 2117*, June 1997.
- [11] L. Flynn and J.J. Garcia-Luna-Aceves. A Protocol for Building and Repairing Secure and Qualified Multicast Trees. In *Proc. 4th Multiconference on Systemics, Cybernetics and Informatics*, Orlando, FL, July 2000.
- [12] X. S. Li, Y. R. Yang, M. G. Gouda, and S. S. Lam. Batch Rekeying for Secure Group Communications. In *Proc. 10th ACM WWW*, Hong Kong, 525–534, Aug. 2001.
- [13] Network simulator - ns-2.
<http://www.isi.edu/nsnam/ns/>.
- [14] A. S. Thyagarajan and S. E. Deering. Hierarchical distance-vector multicast routing for the Mbone. In *Proc. ACM SIGCOMM '95*, Cambridge, MA, 60–66, 1995.
- [15] A. Tsirigos and Z.J. Haas. Multipath Routing in the Presence of Frequent Topological Changes. *IEEE Communications Magazine*, 132-138, Nov. 2001.
- [16] L. K. Wright and S. McCanne and J. Lepreau. A Reliable Multicast Webcast Protocol for Multimedia Collaboration and Caching. In *Proc. ACM Multimedia*, Marina del Rey, 21–30, 2000.
- [17] S. Yan, M. Faloutsos and A. Banerjee. QoS-Aware Multicast Routing for the Internet: The Design and Evaluation of QoSMIC. *IEEE/ACM Transactions on Networking*, V.10, N.1, 54–66, Feb. 2002.
- [18] A. Zegura, K. Calvert, and S. Bhattacharjee. How to Model an Internetwork. In *Proc. IEEE Infocom*, 594–602, March 1996.