Open access • Book Chapter • DOI:10.1007/3-540-45546-9_6

An Overlay Tree Building Control Protocol — Source link

Laurent Mathy, Roberto Canonico, David Hutchison

Institutions: Lancaster University

Published on: 07 Nov 2001 - Lecture Notes in Computer Science (Springer)

Topics: Spanning tree, Multicast, Network topology and Transmission Control Protocol

An Overlay Tree Building Control Protocol[⋆]

Laurent Mathy[1], Roberto Canonico[2], and David Hutchison[1]

[1] Lancaster University, UK
{laurent, dh}@comp.lancs.ac.uk
[2] University Federico II, Napoli, Italy
roberto.canonico@unina.it

Abstract. TBCP is a generic Tree Building Control Protocol designed to build overlay spanning trees among participants of a multicast session, without any specific help from the network routers. TBCP therefore falls into the general category of protocols and mechanisms often referred to as Application-Level Multicasting. TBCP is an efficient, distributed protocol that operates with partial knowledge of the group membership and restricted network topology information. One of the major strategies in TBCP is to reduce convergence time by building as good a tree as possible early on, given the restricted membership/topology information available at the different nodes of the tree. We analyse our TBCP protocol by means of simulations, which shows its suitability for purpose.

1 Introduction

It is well known that the research community has proposed several basic models and a plethora of supporting protocols for multicast in the Internet, none of which has been ultimately deployed on the very large scale of the whole Internet [3]. Making multicast deployment a more difficult task is the fact that even the network layer service of the Internet is in the process of being (at least partially) updated, due to the introduction of a new version of the Internet Protocol (IPv6).

Both the versions of the IP protocol support multicast transmission of datagrams to several receivers [2], i.e. they reserve a set of addresses to identify groups of receivers and they assume that network routers are able to replicate packets and deliver them to all entities registered as members of a group. To carry out such a job, a number of complementary control and routing protocols must be deployed both in the end systems and in the network routers. Such deployment of new software in the network routers (i.e. changes to the network infrastructure) is a major hurdle to ubiquitous deployment, and is responsible for very long rollout delays.

This situation has led the research community to propose mechanisms and protocols to build overlay spanning trees among hosts in a multicast session

[⋆] This work was supported by the EC IST GCAP (Global Communication Architecture and Protocols) Project.

(and possibly service nodes, which are hosts placed strategically in the network to facilitate and optimise the construction of these overlays). Obviously, the distribution of multicast data with such overlay spanning trees is less efficient than in native IP multicast mode, but this relative performance penalty must be contrasted with the easy and speed of deployment offered by the overlay technique.

Until recently, the main paradigm enabling multicast in the Internet was the dual concept of group and associated group address. However, these concepts alone lead to an open service model for IP multicast where anybody can send to a group and that therefore presents serious scalability, security and deployment concerns [3]. This is why a restricted Internet multicast model called Source-Specific Multicast (SSM) has been proposed [8][5]. SSM is based on the concept of multicast channel (i.e. a pair (source address, multicast address)) and the fact that only the channel source can send on the channel. This new model for multicast service, because of its simplicity and elegance, has attracted widespread support within both the academic and industrial communities.

Unfortunately, this SSM model breaks many of the mechanisms and protocols that have been proposed for reliable multicast communications (e.g. [6]), due to the lack of a multicast backchannel from the receivers to the rest of the mutlicast group. It should be noted that such a problem is not specific to an SSM multicast model, as it can also manifest itself on asymmetric satellite link with terrestrial returns, some firewall configurations and some proprietary multicast deployment schemes (e.g. UUnet multicast deployment).

This observation, along with the fact that Tree-based ACKs (TRACKS) [16] [15][14] appear to be an excellent candidate for scalable, (near) real-time reliable multicast communications, argues for overlay multicast spanning trees to be used for control purposes in a reliable multicast scenario.

In this paper, after briefly reviewing techniques that have been proposed to build both multicast data and control trees, we present our Tree Building Control Protocol (TBCP) which is capable of efficiently building both data and control trees in any network environment. Finally, we analyse the performance and characteristics of the overlay trees built by TBCP.

2 Related Work

The concept of tree has proved to be a particularly well suited and efficient dissemination structure for carrying both data and control information in group communications.

Even if, and when, IP multicast becomes widely deployed, the need for tree-based control structures will remain in a reliable multicast scenario. The nodes of such control trees must understand and take part in the control protocols used. These nodes can either be the end-hosts (sender and receivers), service nodes, network routers implementing the appropriate control functions, or any combination of these.

The best way to build control trees that are congruent to the IP multicast data distribution tree is, of course, to support the construction of the control tree within the network. Pretty Good Multicast (PGM) [4] builds control trees based on the concept of *reverse path*. PGM control trees are not overlay trees because internal tree nodes are PGM routers. For that reason, PGM is very efficient.

In [11], overlay trees are built based on *positional reachcasting*. Reachcasting consists of sending packets to the nearest member host in any direction of a multicast routing tree and relies on multicast routers supporting the notion of nearest-host routing. The overlay trees built by with this technique are optimal.

Unfortunately, the above mentioned tree building techniques require modifications to the network infrastructure and are yet to be widely deployed.

Techniques to build overlay trees without relying on special router features, apart from multicast routing, have also been proposed. TMTP [16] uses *expanding ring searches (ERS)* to discover tree neighbours. However, ERS does not perform well in asymmetrical networks, so that the notion of *unsolicited neighbour announcements* has been introduced in TRAM [10]. TMTP and TRAM both require multicast support in the network and will not operate in an SSM context.

More recently, techniques for building overlay spanning trees without depending on multicast routing support in the network have been proposed. We do not attempt an exhaustive review of this area, but rather try and present the most significative proposals.

ALMI [13] is representative of centralised techniques, where a *session controller* has total knowledge of group membership as well as of a the characteristics of a mesh connecting the members (this mesh improves over time). Based on this knowledge, the session controller builds a spanning tree whose topology is distributed to all the members. However, as centralised approach do not scale, distributed techniques are required to support groups larger than a few tens of members.

In Yoid [7], a prospective receiver learns about some tree nodes from a rendezvous point and chooses one of them as its parent. The place in the tree where a new member joins may not be optimal and a distributed tree management protocol is employed to improve the tree over time. However, for scalability reasons, the convergence time to optimality can be rather slow. Yoid hosts also maintain an independent mesh for robustness.

In Narada [1], the hosts first build a random mesh between themselves, then a (reverse) shortest path spanning the mesh. For robustness, each host gains full knowledge of the group membership (Narada is targetted towards small multicast groups) through gossiping among mesh neighbours, and this knowledge is used to slowly improve the quality of the mesh.

Overcast [9] is another approach aimed at an infrastructure overlay. It is specifically targeted towards bandwidth efficient overlay trees. Overcast is a tree-first approach building *unconstrained* trees (i.e. tree nodes do not limit the number of children they serve).

3 Tree Building Control Protocol (TBCP)

Building an overlay spanning tree among hosts presents a significant challenge. To be deployable in practice, the process of building the tree should not rely on any network support that is not already ubiquitous. Furthermore, to ensure fast and easy deployment, we believe the method should avoid having to rely on "its own infrastructure" (e.g. servers, "application-level" routing protocols, etc.), since the acceptance and deployment of such infrastructure could hamper the deployment of the protocol relying on it. Consequently, one of our major requirements is to design a tree building control protocol that operate between end-systems exclusively[1], and considers the network as a "black-box". End-systems can only gain "knowledge" about the network through host-to-host measurement samples.

From a scalability point of view, it is also unrealistic to design a method where pre-requisite knowledge of the full group membership is needed before the spanning tree can be computed. This is especially true in the context of multicast sessions with dynamic group membership. For robustness and efficiency, as well as scalability, it is also preferable to avoid making use of a centralised algorithm.

Our goal is therefore to design a distributed method which builds a spanning tree (which we also call the *TBCP tree*) among the hosts of a multicast session without requiring any particular interactions with network routers nor any knowledge of the network topology, and which does so with only partial knowledge of the group membership.

TBCP is a tree-first, distributed overlay spanning tree building protocol, whose strategy is to place members in a (near) optimal position at joining time.

Compared with the proposals discussed in the previous section, TBCP can be viewed as complementing Yoid, in the sense that TBCP could advantageously replace the rather "random bootstrap" approach in Yoid. Unlike Narada, TBCP does not require full membership knowledge and is not a mesh-first protocol. Also, because of the fundamental strategy in TBCP, tree convergence time should be much faster than in Yoid and Narada.

The general approach to building the tree in TBCP is similar to the one followed in overcast. However, TBCP is not restricted to bandwidth optimised trees and, maybe more importantly, TBCP trees are explicitly constrained (each host fixes an upper-limit on the number of children it is willing to support).

3.1 TBCP Join Procedure

In TBCP, the root of the spanning tree (e.g. main sender to the group) is used as a rendez-vous point for the associated TBCP tree, that is new nodes "join" the tree at the root. Hence a TBCP tree can be identified by the (S,SP) pair, where S is the IP address of the TBCP tree root, and SP the port number used by

[1] We want to emphasise that infrastructure nodes are not considered as a pre-requisite in our approach. However, should such nodes exist, TBCP *could*, of course, be used to interconnect them.

the root for TBCP signalling operations. This information is the only advertised information needed to join the TBCP tree.

Each TBCP entity (including the tree root) fixes the maximum number of “children” it is willing to accommodate in the spanning tree. This value, called the *fanout* of the entity, allows the entity to control the load of traffic flowing on the TBCP tree that it will handle. The TBCP Join procedure is a “recursive”

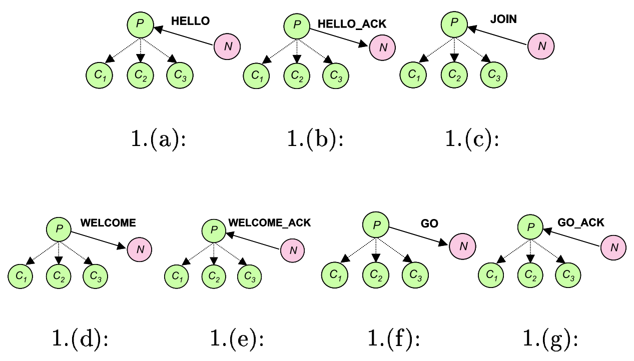


Fig. 1. TBCP Join procedure messages

mechanism and works as follows:

1. A newcomer N contacts a candidate parent P with a HELLO message, starting with the tree root S (figure 1.(a)).
2. P sends back to N the list of its existing children C_i in a HELLO_ACK message (figure 1.(b)), starts a timer T_0 and wait for a JOIN message from N (figure 1.(a). This timer is needed because, for consistency reasons, P cannot accept any new HELLO message until it has finished dealing with the current newcomer.
3. N estimates its “distance” (i.e. takes measurement samples) from P and all C_is and sends this information to P in a JOIN message (figure 1.(c)). Note that if P has not received this JOIN message within its timer T_0, P sends a RESET message to N, meaning that N needs to restart the procedure from stage 1.
4. P finds a place for N, by evaluating all possible “local” configurations having P as parent and involving $C_i \cup N$ (see figure 2). A score function is used to evaluate “how good” each configuration is, based on the distance estimates among P, N and the C_is.
5. Depending on which configuration P has chosen, the following occurs:
   (a) if N is accepted as a child of P, P sends a WELCOME message to N, which N acknowledges immediatly (figures 1.(d) and 1.(e)). The join procedure is then completed for N.

(b) if either N or any of P's children (say C_j) is to be redirected to one of P's children (say C_k), that node is sent a GO(C_k) message (figure 1.(f)) and starts a join procedure (stage 1) with C_k assuming the role of P and C_j the one of N. When N receives such a message, it acknowledges this immediately with a GO_ACK message (figure 1.(g)). However, in order not to disrupt the flow of data for an already established node, C_j is given a time T_1 to find its new place in the tree.

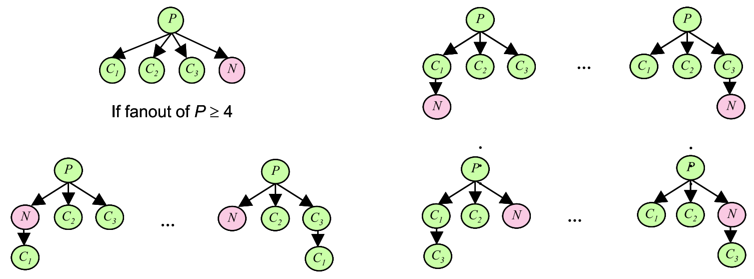


Fig. 2. Local configurations tested.

Notice that a Join procedure is not exclusively performed by TBCP entities that have not joined the tree yet. But, even a TBCP Entity that has already joined the tree may be forced to start a Join procedure, to find a new place in the tree. It should also be noted that the algorithm always finishes, as GO messages always send a TBCP entity (and the associated TBCP subtree whose root is the corresponding TBCP entity) *down* the TBCP tree.

Additional Rules for Tree Construction In order to improve the efficiency (and the shape) of the control tree, a hierarchical organization of receivers in "domains" is also enforced, so that receivers belonging to the same domain belong to the same sub-tree.

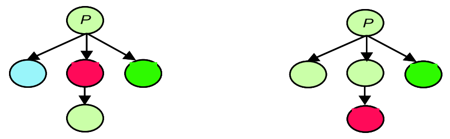
Receivers declare their *domainID* when connecting to a new candidate parent (the *domainID* is a 32 bits identifier, e.g. IP address && netmask). The tree root can then elect a *domain root* node for each domain. For instance, the first node joining from a given domain can be elected as the domain root of its domain. Domain root nodes find their place in the tree with the same mechanism described in the previous section, starting from the tree root.

When a new node wants to join the tree, it is immediately redirected by the tree root to its domain's domain root with a GO message. The Join procedure described in the previous section then starts when the node sends the HELLO message to its domain root.

The following 2 constraints are also enforced, to keep all the nodes of the same domain "clustered":

1. Rule 1: A node P will discard configurations in which a node from its own domain is a child of a node from a different domain.
2. Rule 2: To keep domain roots as high as possible in the tree (i.e. as close as possible to the tree root), configurations in which a node P keeps more than one node from its own domain as children, and sends a domain root of a different domain as a child of one of its children, are discarded.

Figure 3 illustrates discarded configurations, where domains are represented by "colour codes".



3.(a): Rule 1.

3.(b): Rule 2.

Fig. 3. Discarded configuration due to rule violation.

3.2 Cost Function and Measurements

In section 3.1, we have seen that a score is computed for each "local" configuration involving a parent node, its children and a "newcomer". These scores are obtained by applying to the local configurations a score function whose inputs are the distances among the nodes involved and/or any other relevant "metric" (e.g. domainIDs, node fanouts, etc.). Because the score function is used in each local decision taken by the TBCP nodes in the tree, the score function influences the final (i.e. global) shape of the tree. Of course, what constitutes a good shape for an overlay tree depends on the purpose for which the tree is built. Likewise, the notion of "distance" between tree nodes, used as input to the score function, can be represented by any metric (or set of metrics) of interest (e.g. delay, throughput, etc.) depending on the problem at hand.

Since score functions and distance evaluation mechanisms depend on the purpose of the tree, for flexibility, they are considered as "modules" rather than part of the core TBCP protocol.

For example, for an overlay used for reliable multicast, the following could be selected:

– distance $D(i, j)$ = round trip time (RTT) between node i and node j *along the tree*[2];

[2] Hence, if node i is the parent of node k that, in turn, is the parent of node j, then $D(i, j) = D(i, k) + D(k, j)$.

– score function = $max_{\forall M \in \{C_i\} \cup N} D(P, M)$, where $\{C_i\}$ is the set of P's children and N the newcomer.

The chosen configuration is the one with the smallest score. We therefore see that our strategy is to try and maximise the "control responsiveness" between nodes.

4 Simulation Results

We have simulated (with NS-2 [12]) our algorithm, with the score function proposed in section 3.2, in order to evaluate their suitability for purpose. At this stage, because we are mainly concerned with the characteristics of our trees, no background traffic is used and hop counts can therefore be used as distance metric instead of RTTs.

We have used topologies composed of:

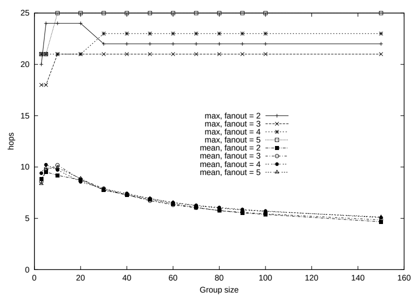
– a core network of 25 routers, with each router connected to any other with a random probability of 10%;
– 15 stub networks of 10 routers each, with each router connected to any other router in the same stub with a random probability of 5%;
– each stub network is connected to the core by a single link, with each stub connected to a different core router.

Furthermore, the nodes running our algorithm (root/receivers) are "hosts" connected to stub routers only, with at most one receiver host connected to one stub router. The hosts are distributed randomly in the stub networks which determine domains. We therefore see that these topologies correspond to a worst-case scenario, as the distance between receivers and between stub networks is minimum 3 hops and there is no tendency to "cluster" the participating hosts. The results obtained should therefore indicate upper-bounds results.

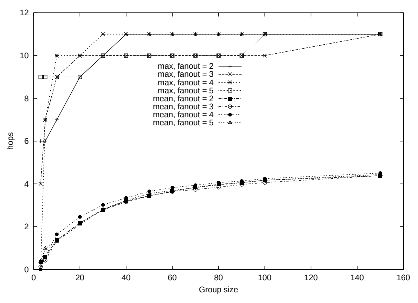
We have simulated scenarios where all the TBCP entities have an identical fanout of respectively 2,3,4 and 5.

For each value of the fanout, groups of 3, 5, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100 and 150 receivers have been incrementally tested on a same topology, that is 3 receivers joined the tree, then 2 were added, then 10, etc. This scenario was repeated 10 times (with a different topology each time) for each fanout value.

Since, in the context of this paper, we are interested in the performance of the TBCP trees for control purposes, the mean and maximum distance measured between any receiver (i.e. node) in the tree and its parent, as well as the distance between any receiver and its domain root (see section 3.1), are of prime concern. These are depicted in figure 4. In figure 4.(a), we see that as the size of the group increases, the mean distance between nodes and their parents decreases. This is because the larger the group is, the more "clustering" appears, and the algorithm is thus efficient at exploiting such clustering among receivers. The maximum distance observed between a node and its parent is due to the "interconnection" of different domains and depends on the topology.



4.(a): Mean and maximum distance from parent.



4.(b): Mean and maximum distance from domain root.

Fig. 4. Distances measuring the "locality" efficiency of the TBCP trees.

Figure 4.(b) shows that control operations confined within a domain would see small response times, with a mean distance between any node in a domain and its domain root (the root of the subtree covering the domain) increasing with a small slope. The position of the domain root within the domain has of course a great influence on the node-domain root distances, especially the maximum distance. Because in these simulations, the domain root was chosen to be the first node of a domain to join the tree and was therefore randomly placed within a domain, the maximum values in figure 4.(b) are therefore likely to represent worst-case scenarios.

The mean and maximum distances measured between any receiver and the tree root is depicted in figure 5. This figure represents the performance observed

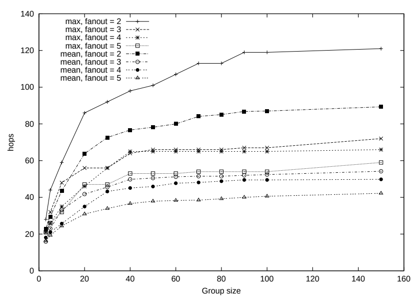
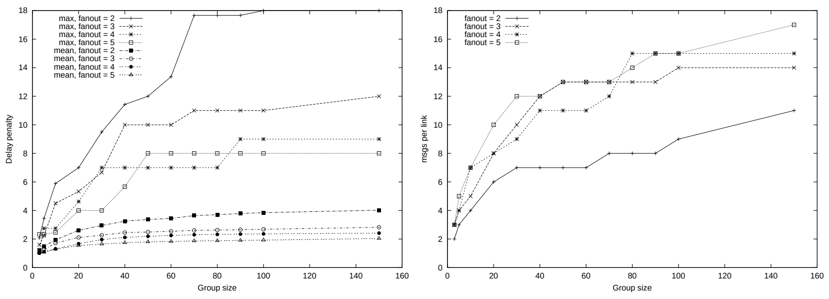


Fig. 5. Mean and maximum distance from (tree) root.

when traversing the tree from root to leaves. The distances between the root and the leaf nodes is less critical in the control scenario considered here than if the overlay tree were to be used for data transfers. As we expect, the values observed stress the importance of the value of the fanout in reducing overall delay along the tree, with the mean delay for a fanout of 2 being (roughly) 1.5 times the maximum delay for a fanout of 5.

The tree can also be "globally" characterized by investigating mean and maximum delay ratios and link stresses. The delay ratio for a receiver is defined as the ratio of (the delay from the root to this receiver along the tree) to (the delay from the root to this receiver with "direct" (unicast) routing). The link stress is simply a count of how many replicates of the same data appear on a link, when that data is "multicast" along the TBCP tree. These are depicted in figures 6. Figure 6.(a) shows that, on average, the distance between the tree root



6.(a): Mean and maximum delay ratios.

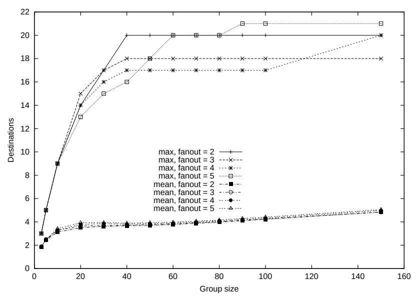
6.(b): Maximum link stress.

Fig. 6. Global characteristics of tree.

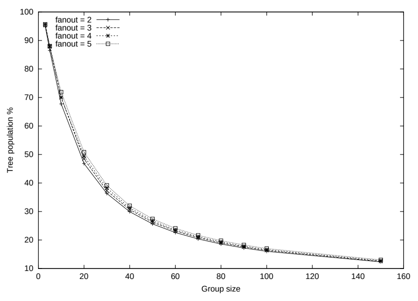
and any node along the tree is on average about 2 to 4 times the direct distance between the same nodes, which shows that the algorithm, and score function, exhibits a tendency to grow the trees "in width". The figure also shows that the value of the fanout has a dramatic effect on the maximum delay penalty observed on the trees.

Figure 6.(b) shows the expected reduction in load on network links when an overlay tree is used as opposed to a "reflector" which uses unicast communications from the root to each receiver. Indeed, in the reflector case, the maximum link stress is equal to the group size, as the link directly out of the root must carry all the data packets.

Finally, figures 7.(a) and 7.(b) illustrate the measurement sampling overhead of the protocol (in terms of the number of nodes sampled) during the *initial* joining period for a newcomer (i.e. from the moment the newcomer issues a JOIN and it receives a WELCOME message). The overhead measured is therefore proportional to the joining latency of a new node. Figure 7.(a) shows that the



7.(a): Mean and maximum measurement samples.



7.(b): Mean measurement samples in % of population.

Fig. 7. Measurement sampling overhead.

maximum number of nodes sampled is kept well below the number of nodes in the tree and is, indeed, very reasonable. It also shows that the average number of measurement samples taken by newcomers shows a sub-linear increase in terms of the group size.

Figure 7.(b) shows the percentage of the already existing population of the tree that is sampled by a newcomer. The shape of these curves should not be any surprise. Indeed, only nodes that are siblings of the nodes constituting the branch between the root and the final place of the newcomer are sampled in addition to nodes in this branch. Therefore, the more branches that are added to the tree, the bigger the proportion of the tree population will be ignored by each move of the newcomer along its branch.

These results presented in figure 7 show the scalability of the proposed tree building mechanism.

5 Conclusions

We have described an efficient and scalable TBCP algorithm/protocol to build such an overlay spanning tree. Our TBCP protocol does not rely on any special support from routers, nor on any knowledge of the network topology.

Because our algorithm builds the tree through a series of local decisions involving only a small subset of nodes for each decision, a joining node need only to get to know a few members of the multicast groups. Also, because our algorithm is decentralised, several new members can be in the process of joining the tree simultaneously without incurring additional overhead, which is a good scaling property for this type of protocols.

In the light of our simulation results, we believe our proposal constitutes a good candidate for scalable overlay tree building. We have also designed a leave procedure which, because of the lack of space, was not presented in this paper.

References

1. Y-H. Chu, S. Rao, and H. Zhang. A Case for End System Multicast. In *ACM SIGMETRICS 2000*, pages 1–12, Santa Clare, CA, USA, June 2000.
2. S. Deering and D. Cheriton. Multicast Routing in Datagram Internetworks and Extended LANs. *ACM Trans. on Comp. Syst.*, 8(2):85–110, May 1990.
3. C. Diot, B. Levine, B. Lyles, H. Kassem, and D. Balensiefen. Deployment Issues for the IP Multicast Service and Architecture. *IEEE Network*, 14(1):78–88, Jan/Feb 2000.
4. D. Farinacci, A. Lin, T. Speakman, and A. Tweedly. Pretty Good Multicast (PGM) Transport Protocol Specification. Internet Draft draft-speakman-pgm-spec-00, IETF, Jan 1998. Work in progress.
5. B. Fenner, M. Handley, H. Holbrook, and I. Kouvelas. Protocol Iindependent Multicast - Sparse Mode: Protocol Specification (revised). Internet Draft draft-ietf-pim-sm-v2-new-02, IETF, Mar 2001. Work in Progress.
6. S. Floyd, V. Jacobson, C. Liu, S. McCanne, and L. Zhang. A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing. *IEEE/ACM Trans. Network.*, 5(6):784–803, Dec 1997.
7. P. Francis. Yoid: Extending the Internet Multicast Architecture. Technical report, ACIRI, Apr 2000. www.aciri.org/yoid.
8. H. Holbrook and D. Cheriton. IP Multicast Channels: EXPRESS Support for Large-scale Single-source Applications. *ACM Comp. Comm. Reviews*, 29(4):65–78, Oct 1999.
9. J. Jannotti, D. Gifford, K. Johnson, F. Kaashoek, and J. O'Toole. Overcast: Reliable Multicasting with an Overlay Network. In *USENIX OSDI 2000*, San Diego, CA, USA, Oct 2000.
10. M. Kadansky, D. Chiu, and J. Wesley. Tree-based Reliable Multicast (TRAM). Internet Draft draft-kadansky-tram-00, IETF, Nov 1998. Work in Progress.
11. B. Levine and JJ. Garcia-Luna. Improving Internet Multicast with Routing Labels. In *IEEE Intl. Conf. on Network Protocols (ICNP'97)*, pages 241–250, Atlanta, USA, Oct 1997.
12. Ns-2 Network Simulator. http://www.isi.edu/nsnam/ns/.
13. D. Pendarakis, S. Shi, D. Verma, and M. Waldvogel. ALMI: an Application Level Multicast Infrastructure. In *3rd USENIX Symposium on Internet Technologies*, San Francisco, CA, USA, Mar 2001.
14. B. Whetten, D.M. Chiu, M. Kadansky, and G. Taskale. Reliable Multicast Transport Building Block for TRACK. Internet Draft draft-ietf-rmt-bb-track-01, IETF, Mar 2001. Work in progress.
15. B. Whetten and G. Taskale. An Overview of Reliable Multicast Transport Protocol II. *IEEE Network*, 14(1):37–47, Jan 2000.
16. R. Yavatkar, J. Griffioen, and M. Sudan. A Reliable Dissemination Protocol for Interactive Collaborative Applications. In *ACM Multimedia'95*, San Francisco, CA, USA, Nov 1995.