MλT: A Multicast Protocol with QoS Support

J. Ignacio Alvarez-Hamelin and Pierre Fraigniaud
LRI, Univ. Paris Sud, 91405 Orsay, France.
{ihameli,pierre}@lri.fr http://www.lri.fr/~{ihameli,pierre}

***Abstract*— This paper describes the new multicast protocol MλT, supporting QoS requirements. MλT is dedicated to any QoS parameter λ that is either additive (e.g., delay), or convex (e.g., available bandwidth). MλT constructs a multicast tree connecting a group of users such that, for any user x, the QoS λ supported by the path from the root of the tree to x is optimized under some constraints of bounded control traffic. Simulations on the UUNET network topology, as well as simulations on network topologies obtained by INET-3.0, show that MλT performs in average significantly better than QoSMIC, RSP strategies (e.g., CBT and PIM-SM), and greedy strategies.**

I. Introduction

Multicast protocols are the basic support for multi-users applications (e.g., white-board, video-on-demand, video-conferences, etc.) in which more than two users, possibly hundreds, interact. In most of the existing protocols, communications are performed through trees connecting all the users sharing the application [1], [2]. Several methods have been proposed for constructing multicast trees offering good cost vs. performance ratio, the main difficulty arising from the dynamics of the users group (i.e., users can join and leave the group at their convenience). On-line decentralized protocols can be roughly classified as *shortest paths* (e.g., CBT [3], and PIM-SM [4]), or *greedy* (e.g. QoSMIC [5], and YAM [6]). In the former case, the multicast tree is composed of the union of the (possibly reverse) shortest paths connecting some specific node (called "source" or "core") to every member of the group. In the latter case, a node joining a group is connected to its closest node in the current tree. Some of these latter protocols support QoS requirements by considering several possible routes connecting the joining node to the tree, and selecting the most appropriate one according to the considered QoS parameter (minimizing more than a single QoS parameter is a much more difficult task [7]).

In this paper, we describe the new multicast protocol MλT (for *Minimal λ-Tree*). Its periodic activation allows to maintain a multicast tree that spans the group of users, and such that a specified QoS parameter λ along the route from the core to each member is optimized. The optimization of the QoS is performed under some constraints of bounded control traffic. More precisely, if a route in the tree does not have the optimal QoS, then the length of a route with better QoS is longer than a given threshold. The lengths of the routes are controlled by two parameters ϱ and r whose values are fixed *a priori*. MλT only considers routes from the core s to each member x whose length is at most

$$\varrho \cdot d(s,x) + r \tag{1}$$

where $d(x,y)$ denotes the distance (i.e., number of hops) between two nodes x and y. MλT applies to any QoS parameter that is either additive or convex. More precisely, let $\lambda(e)$ be the QoS parameter corresponding to link e. The QoS parameter of a route $R=(e_1,e_2,\dots,e_k)$ must satisfy either

$$\lambda(R)=\sum_{i=1}^{k}\lambda(e_i) \quad \text{(additive version)} \tag{2}$$

or

$$\lambda(R)=\max_{1\le i\le k}\lambda(e_i) \quad \text{(convex version).}$$

The additive version corresponds to, e.g., the delay, whereas the convex version corresponds to, e.g., the available bandwidth (by setting $\lambda = 1/\text{available bandwidth}$). When two routes R_1 and R_2 are to be compared, MλT selects the one that has the smallest value for λ, i.e., the route $R \in \{R_1,R_2\}$ such that

$$\lambda(R)=\min\{\lambda(R_1),\ \lambda(R_2)\}. \tag{3}$$

Given a tree T, the QoS of T is denoted by $\lambda(T)$, and its value is recursively defined by

$$\lambda(T)=\max_{1\le i\le k}\big(\lambda(T_i)+\lambda(e_i)\big)$$

in the additive version, and by

$$\lambda(T)=\max_{1\le i\le k}\max\{\lambda(T_i),\ \lambda(e_i)\}$$

in the convex version, where T is the union of k subtrees $T_1,\dots,T_k$ whose roots are connected to the root of T by the links $e_1,\dots,e_k$, respectively.

Roughly speaking, MλT explores all routes between the core and the group members, whose lengths do not exceed the threshold specified by Equation 1. Then MλT selects the routes so that to obtain a tree T whose QoS $\lambda(T)$ is minimum among all trees constructed with routes whose length satisfy Equation 1. A more detailed description of MλT is given in Section II.

We have performed simulations on Internet topologies: UUNET and others, generated by INET-3.0 [8]. We have compared the performances of MλT with those of QoSMIC, RSP, and Greedy. Obviously, the larger ϱ and r, the better it is in term of QoS because more routes are taken under consideration. However, the control traffic resulting from the exploration of a large portion of the network counterbalances the benefit of this exploration. There is an inherent trade-off between the length of the routes (i.e., the values of ϱ and r), and the amount of control traffic. We show that the structure of the Internet topology allows MλT to perform significantly better than QoSMIC, RSP, and Greedy, even for small values of ϱ and r, that is for a limited increase of the control traffic. These simulations are presented in Section III.

Finally, Section IV contains some concluding remarks.

II. The multicast protocol MλT

MλT is built upon a multiple-path routing protocol which maintains a table D_v at each router v, such that $D_v[i, x]$ is the minimum length of the path connecting node v to node x when leaving v through the i-th interface. For instance, at the inter-domain level, BGP [9] stores this information at every node. At intra-domain level, RIP [10] and OSPF [11] do not store this information directly, but slight modifications of these protocols would allow to keep track of it. Indeed RIP and OSPF just discard all paths that are not the shortest. However, longer paths are considered during the setting of the routing tables, and one could easily keep track of the information about these paths for the setting of the tables D_v's.

Based on the multiple-path routing protocol, MλT is divided into three phases.

1) Exploration phase. This phase starts at the root s of the tree. The root operates like a *core* node or a *rendezvous* point. It uses the current list of members of a multicast group M, and starts an exploration of the network whose goal is to traverse, for each member $x \in M$, all the paths from s to x whose length is at most $\varrho \cdot d(s, x) + r$.
2) Selection phase. This phase consists of selecting, among all paths obtained during the exploration phase, the ones which minimize the QoS parameter λ.
3) Construction phase. During this phase, the routing tables of the multicast group M are set, so that all packets follow the paths selected during the selection phase.

The formal description of these three phases is given in Table I (only the additive version is described, the convex one can be trivially obtained by modifying lines 27 and 29). The current node is denoted by v, and its neighbors by u_i. L is a list, and $L[i]$ is its i-th element. $L \leftarrow L + y$ is the operation which adds a new element y into the list L. Similarly, $L \leftarrow L - y$ is the operation consisting of removing the element y from the list L. The list I stored at v contains, for each destination x, the next node in the selected path from v to x. MλT uses three types of messages:

1) Messages M_{expl} are used during the exploration phase. A message M_{expl} is a triplet (T, D, L), where T is a list of nodes belonging to the multicast group, D is the list of all distances $d(s, x)$ for all $x \in T$, and L is the list of the already visited nodes. We assume that T and D are in the same order, that is $D[k]$ is the distance corresponding to node $T[k]$.
2) Messages M_{sel} are used during the selection phase. A message M_{sel} is a triplet (S, Λ, L), where S is a list of nodes belonging to the multicast group, Λ is the list of QoS parameters corresponding to the nodes of S, and L is the list of nodes in the path from the root s to the current node v.
3) Messages M_{cons} are used during the construction phase. A message M_{cons} is a pair (C, L), where C is the list of nodes that the current node v connects to the multicast tree, and L is the list of already visited nodes.

MλT starts at the root s (i.e., initially, $v = s$ in the algorithm of Table I) with an exploration message $M_{expl} = (T, D, L)$, where T contains all members of the group, D contains all distances $d(s, x)$ for all members of the group, and $L = \emptyset$. The current node v sends an exploration message $M_{expl} = (T^{(i)}, D^{(i)}, L + v)$ on link i connecting v to its neighbor u_i. The lists $T^{(i)}$ and $D^{(i)}$ are set up as follows. Node $x \in T$ is added to $T^{(i)}$ if $|L| + \mathcal{D}_v[i, x] \leq \varrho \cdot d(s, x) + r$, where $|L|$ is the number of elements in the list L. Indeed, $|L| + \mathcal{D}_v[i, x]$ is at least the length of any route following the nodes in L, and going from v to x after leaving v by its ith interface. The exploration phase completes at a node z receiving the list $T = \{z\}$, i.e. z is the last element of the current list T.

When the exploration phase completes at node v, the selection phase begins at v. The current node v sends the message $M_{sel} = (S, \Lambda, L)$ with $S = \{v\}$ and $\Lambda = \{0\}$ to the node w from which it received the exploration message containing the list L. When a node v has received all the messages $M_{sel} = (S^{(i)}, \Lambda^{(i)}, L)$ from all its neighbors, it computes the QoS parameter of the message $M_{sel} = (S, \Lambda, L)$ which will be sent to w. The list Λ is set up using Equations 2 and 3, i.e. by adding the QoS value $\lambda(v, u_i)$ to $\Lambda^{(i)}[j]$, and by selecting the path which minimizes the QoS parameter λ among the values $\Lambda^{(i)}[j]$ for all i's. The messages M_{sel} converge to the root s which makes the final selection. Then, the construction phase starts.

During the construction phase, the root s sends the messages $M_{cons} = (C^{(i)}, \{s\})$ through its ith interface, where $C^{(i)}$ is the set of nodes with minimum QoS parameter. Every node which receives a construction message computes the lists $C^{(i)}$, and construct the corresponding message $M_{cons}^{(i)}$, by selecting, for each of its interfaces, the list of destination nodes that must be reached through this interface. Once the construction phase is completed, the tree is ready to transmit packets among the multicast users.

Remark. In a normal use, it is necessary to run MλT regularly in order to take into account the possible modifications of the group, and to capture the dynamic nature of the traffic.

Lemma 1: MλT does not have loops.

Proof: A loop implies that a message M_{expl}, M_{sel} or M_{cons} traverses twice the same node. In the case of a message $M_{expl} = (T, D, L)$, the current node v is added to the list L of visited nodes, and v checks L in order to not send a message M_{expl} to members of this list (see instructions 3 and 4). Thus, there cannot be loops related to the exploration messages. A selection message $M_{sel} = (S, \Lambda, L)$ is sent to the parent w of the current node v. By construction of the list L, this parent is unique. Thus, there cannot be loops related to the selection messages. Finally, a construction message $M_{cons} = (C, L)$ is sent to a subset of nodes in the list contained in the exploration

Algorithm 1

```
1  **Exploration phase:**
2  /* Activated when $v$ receives an exploration message $M_{expl} = (T, D, L)$ from $w$, or when $v$ is the root $s$ of the tree.*/
3      $L' \leftarrow L + v$;
4      For every $v$'s neighbor $u_i \notin L$ do
5          $T^{(i)} \leftarrow \emptyset$; $D^{(i)} \leftarrow \emptyset$;
6          For every member $x \in T$ do /* say $x = T[j]$ */
7              If $|L| + \mathcal{D}_v[i, x] \leq \varrho \cdot D[j] + r$ then
8                  $T^{(i)} \leftarrow T^{(i)} + T[j]$;
9                  $D^{(i)} \leftarrow D^{(i)} + D[j]$;
10         If $T^{(i)} \neq \emptyset$, then send $(T^{(i)}, D^{(i)}, L')$ to $u_i$;
11     If ($T^{(i)} = \emptyset$ for all $i$) then
12         $S \leftarrow \emptyset$; $\Lambda \leftarrow \emptyset$;
13         If $v$ is a group member then
14             $S \leftarrow S + v$;
15             $\Lambda \leftarrow \Lambda + 0$;
16         Send $M_{sel} = (S, \Lambda, L)$ to $w$;
17 **End exploration phase.**
18
19 **Selection phase:**
20 /* Activated when $v$ has received a message $M_{sel}^{(i)} = (S^{(i)}, \Lambda^{(i)}, L)$ from all the neighbors through which it sent an exploration message $(T^{(i)}, D^{(i)}, L)$.*/
21     $S \leftarrow \emptyset$; $\Lambda \leftarrow \emptyset$;
22     If $v$ is a group member then
23         $S \leftarrow S + v$; $\Lambda \leftarrow \Lambda + 0$;
24     For every group member $x$ do
25         If $S^{(i)} \neq \emptyset$ then
26             $S \leftarrow S + x$;
27             $\lambda \leftarrow \min_i(\lambda(v, u_i) + \Lambda^{(i)}[j])$ for all pairs $(i, j)$
                   such that $S^{(i)}[j] = x$;
28             $\Lambda \leftarrow \Lambda + \lambda$;
29             $I \leftarrow I + u_k$ where $k$ and $j$ satisfy $\lambda = \lambda(v, u_k) +$
                   $\Lambda^{(k)}[j]$ and $S^{(k)}[j] = x$;
30     If $v \neq s$ then
31         $L' \leftarrow L - w$;
32         Send $M_{sel} = (S, \Lambda, L')$ to $w$;
33     Else
34         For every neighbor $u_i$ do
35             $C \leftarrow \emptyset$;
36             If there exists $j$ such that $u_i = I[j]$ then
37                 $C \leftarrow C + S[j]$;
38             If $C \neq \emptyset$ then send $M_{cons} = (C, \{s\})$ to $u_i$;
39 **End of selection phase.**
40
41 **Construction phase:**
42 /* Activated when node $v \neq s$ receives a message $M_{cons} = (C, L)$ */
43     For every neighbor $u_i \neq w$ do
44         $C' \leftarrow \emptyset$;
45         For each node $x \neq v$ in $C$ do
46             If (there exists $j$ such that $x = S[j]$ and $u_i = I[j]$)
                   then
47                 $C' \leftarrow C' + S[j]$;
48         If $C' \neq \emptyset$ then send $M_{cons} = (C', L + v)$ to $u_i$;
49 **End of construction phase.**
```

TABLE I
THE MλT PROTOCOL

message M_{expl}. Since the exploration messages do not induce loops, the construction messages neither. ■

Lemma 2: MλT terminates.

Proof: The root s starts with the list of all members of the multicast group. Upon reception of an exploration message, the current node adds a node x to the list T in the message M_{expl}. This message will be sent to the neighbor u_i if and only if the distance $1 + d(u_i, x)$ is at most $|L| + \varrho \cdot d(s, x) + r$. At every stage, the same calculation is performed, using the original distance starting from the root plus the number of nodes already traversed (i.e., the number of elements in the list L). Therefore, since the number of nodes traversed before arriving at a node is bounded (cf. Lemma 1), the exploration phase terminates. The selection phase terminates because the messages eventually converge at the root. The construction phase terminates because the list C is updated after checking $v \neq T[i]$ (see instruction 45). ■

Using the two previous lemmas, one can prove the following.

Theorem 1: MλT performs correctly, i.e., MλT returns a tree T routed at s such that (1) all group members are in T, (2) the distance in T between s and any group member x is at most $\varrho\, d(s, x) + r$, and (3) the QoS of the route in T from s to any group member x is minimal among all routes in the network with length at most $\varrho\, d(s, x) + r$.

Due to lack of space, the proof is omitted, but is straightforward by construction of the algorithm in Table I.

The next section presents simulation results on MλT. In these simulations, the multiplicative parameter ϱ was set to 1, and we have studied the impact of the additive parameter r. Having $\varrho > 1$ gives more flexibility to find alternative routes between nodes far apart in the network. However, since we performed simulations on networks with relatively small diameter, the influence of ϱ was difficult to capture. Hence, we mainly focused on the influence of r.

III. SIMULATIONS

We have performed simulations on realistic network topologies: the backbone topology of UUNET (UUNET was constructed after the *Mapnet* project from *CAIDA* association[1]), and a topology generated by INET-3.0 [8]. UUNET is perhaps the largest commercial network whose topological characteristics are freely available. The UUNET backbone contains 129 nodes, its diameter (i.e., the maximum distance between two nodes) is 7, and its mean node-degree is roughly 5.

INET-3.0 is a well know Autonomous System's topology generator, based on the information collected from the BGP tables. The smallest topology generated by INET-3.0 has 3,037 nodes. The topology that we generated has diameter 8, and its mean node-degree is roughly 3.

Clustering coefficient. The average clustering coefficient is an important parameter for the characterization of network topologies. The clustering coefficient of node x is the ratio between the number of links connecting the k neighbors of

[1]`http://www.caida.org/tools/visualization/mapnet/`.

x, and the maximum possible number of such links, i.e., $k(k-1)/2$. The average clustering coefficient of a network is simply the average of the clustering coefficients of its nodes. It measures the probability for two neighbors to be connected. In other words, the average clustering coefficient is an estimator of the percentage of triangles (cycles of length 3) between the nodes.

The average clustering coefficient of UUNET backbone is roughly 0.25. The average clustering coefficient of our topology generated by INET-3.0 is roughly 0.20. Therefore, for both topologies, the percentage of triangles is high, that is, given an edge, the probability that an alternative path of length 2 exists between its two extremities is high. This property is generally satisfied in Internet [12] (clustering coefficient of 0.24 at the autonomous systems level, and 0.03 at the router level – this latter value is large if we note that it is averaged over $320,105$ nodes).

Experimental scenarios. In order to test the behavior of MλT, we have generated two sets of scenarios. Each scenario is characterized by the way the values of the parameter λ at each link were generated. One scenario generates integer values of λ's uniformly at random in $[1, 100]$. The other generates integer values of λ's in $[1, 100]$, so that the probability of choosing small values is higher. More precisely, the probability of choosing $\lambda = l$ is proportional to $1/l$. This type of situation is characteristic of the link-congestion, that is, few of the links are overloaded, whereas most of the links are not. In UUNET, we have generated 1000 scenarios, that is 1000 different setting of the λ's of the UUNET links. In INET-3.0, we have generated 100 scenarios only because of the large size of the network, which induced time-consuming simulations.

We have then performed several experiments. For each experiment, we have chosen the multicast group members uniformly at random among all network's nodes. Next, we have computed the multicast tree T corresponding to this group, according to the considered multicast protocol. Finally, we have computed the parameter $\lambda(T)$ of the tree T.

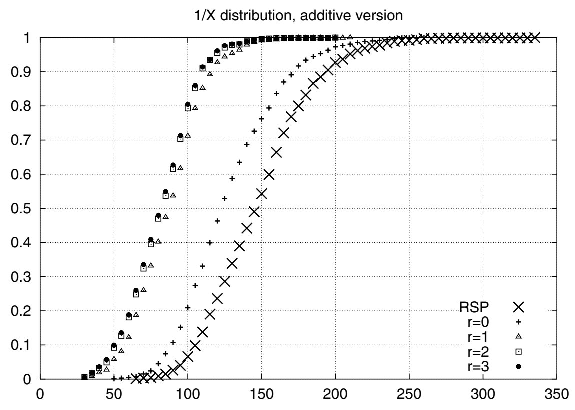
The quality of a multicast protocol is estimated by the following cumulative distribution function:

$$F(\ell) = \text{Prob}(\lambda(T) \leq \ell), \qquad (4)$$

where $\text{Prob}(\lambda(T) \leq \ell)$ is the probability that $\lambda(T)$ is at most ℓ, i.e., the percentage of multicast trees T such that $\lambda(T) \leq \ell$. Thus, a protocol A performs better than another protocol B if the cumulative distribution function of A increases more rapidly than the cumulative distribution function of B.

Message control vs. performances of MλT. We have first studied the impact on MλT of the additive parameter r. We have performed simulations in UUNET for both additive and convex variants. We have considered $r \in \{0, 1, 2, 3\}$. The results are displayed on Figure 1. We have displayed five cumulative distributions, one for the each of the four considered values of r, and one for the RSP protocol (the latter for the purpose of comparison).

We first observe that, even for $r = 0$, MλT performs better than RSP. For instance, in the additive case, 50% of the trees



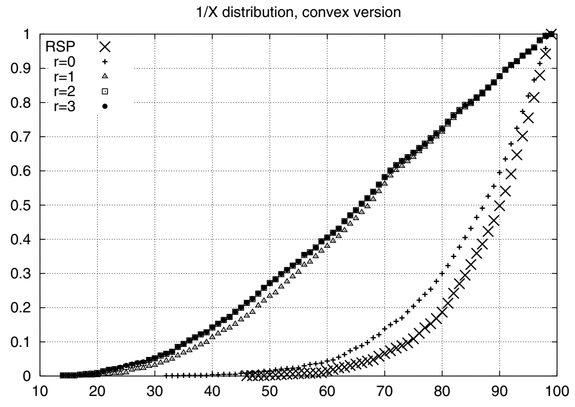


Fig. 1. Performances of MλT as a function of r in UUNET, for a group of 20 members

T constructed by RSP satisfy $\lambda(T) \leq 145$, whereas 50% of the trees T constructed by MλT satisfy $\lambda(T) \leq 125$. This phenomenon is due to the existence of several shortest paths in UUNET. For $r = 1, 2$, and 3, we observe that 50% of the trees T constructed by MλT satisfy $\lambda(T) \leq 87$, $\lambda(T) \leq 80$, and $\lambda(T) \leq 80$ respectively. We have observed a similar behavior of MλT in the convex case. That is, we have 50% of the trees T constructed by RSP that satisfy $\lambda(T) \leq 89$, whereas 50% of the trees T constructed by MλT satisfy $\lambda(T) \leq 86$, $\lambda(T) \leq 66$, $\lambda(T) \leq 64$, and $\lambda(T) \leq 64$, for $r = 0, 1, 2$, and 3, respectively. (Note that the curves for $r = 2$ and $r = 3$ are perfectly identical). These experiments demonstrate the following.

1. On one hand, the main improvement of MλT is obtained when increasing r from 0 to 1. Increasing r from 1 to 2, or from 2 to 3 has a relatively limited impact.

2. On the other hand, Table II displays the average number of messages generated by MλT and the average number of links used in the trees as a function of r. This table shows that the number of control messages increases dramatically with r.

We have observed a similar behavior for a uniform distribution of the λ's.

As a conclusion, the use of MλT with $r = 1$ is a good trade-off between performances and cost. Therefore, throughout the rest of the paper, all experiments are performed with $r = 1$.

Version	r	Nb. of messages	Nb. of links
additive	0	83	27
convex	0	83	27
additive	1	975	33.8
convex	1	975	40.4
additive	2	10134	34.2
convex	2	10134	48.6
additive	3	89857	34.6
convex	3	89857	52.5

TABLE II

AVERAGE NUMBER OF MESSAGES AND LINKS AS A FUNCTION OF r.

MλT vs. RSP, Greedy, and QoSMIC in UUNET. We have compared MλT with other construction techniques for multicast trees. Recall that QoSMIC [5] performs as Greedy but, to select the path with the highest QoS, QoSMIC also takes into account the QoS of the shortest paths from a joining node x to all nodes currently in the tree (we have considered *local search* only).

Results are displayed on Figure 2. We first observe that the behavior of RSP and Greedy are quite similar, independently of the variants (additive or convex) and of the distribution of λ.

In general QoSMIC performs better than RSP and Greedy, but for the convex variant applied to a uniform distribution of the λ's. In this latter case, QoSMIC performs actually worse than Greedy. This is due to the following situation. When a new member x joins the group, Greedy connects x to the closest (in number of hops) node in the current tree. In the same situation, QoSMIC considers several connecting points in the current tree, and chooses the one, say y, such that the QoS of the path x, y (here the sum of λ's for all edges of the path) is minimum. However, y can be far away from the core of the tree, and this implies that the depth of the resulting tree can be much larger than the depth of the tree constructed by Greedy. In the additive variant, a long path in the tree T will yield a large value for $\lambda(T)$. This phenomenon can also be observed when only few λ's have large value, but only in a limited extend. In the convex variant, this phenomenon was never observed in our simulations.

In all four cases of Figure 2, MλT always performs significantly better than QoSMIC. Actually, the benefit obtained from using MλT instead of QoSMIC is larger than the benefit obtained from using QoSMIC instead of RSP. Moreover, Table III shows that number of control messages exchanged for the construction of the tree by MλT is less than 3 times larger than the number of control messages of QoSMIC, whereas the number of control messages of QoSMIC is more than 10 times larger than the number of control messages of RSP. Therefore, for all practical instances in which QoSMIC performs better that RSP, the use of MλT would bring even more benefits.

The reason for such a good behavior of MλT is the already mentioned high average clustering coefficient of UUNET. The large number of triangles in UUNET insures the existence of a large number of different paths of length at most $d(x, y) + 1$ between two arbitrary nodes x and y. Therefore, hot spots can

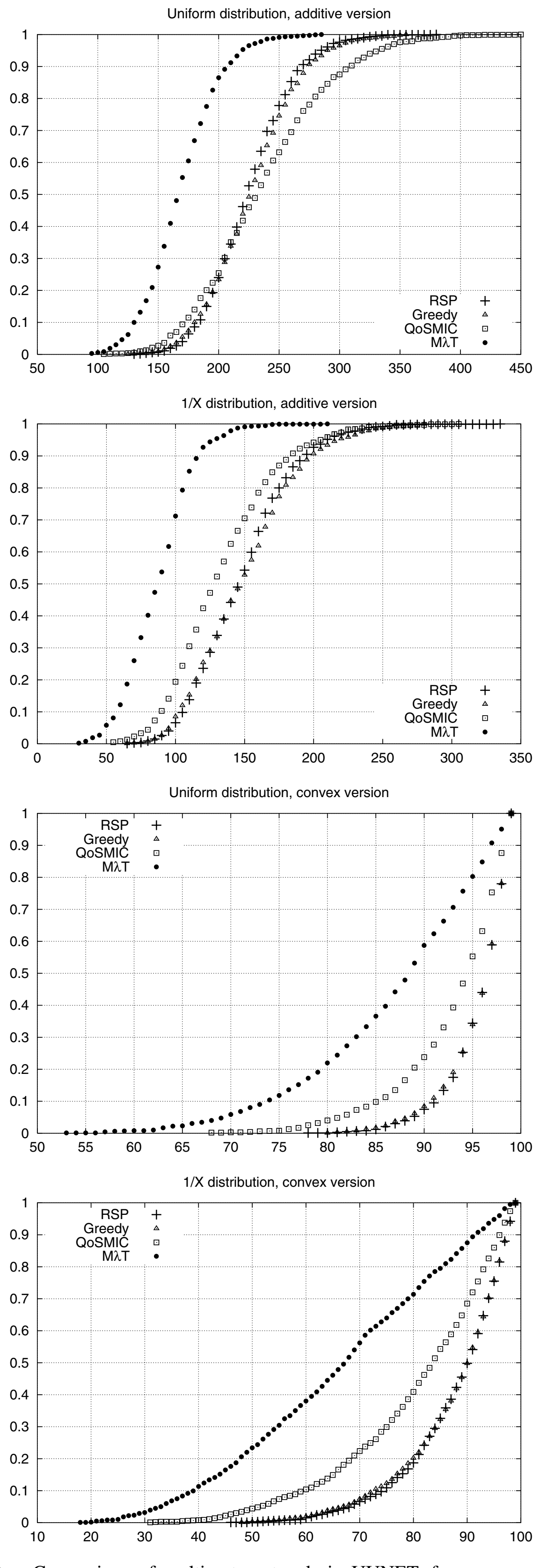


Fig. 2. Comparison of multicast protocols in UUNET, for a group of 20 members

	UUNET		INET-3.0	
Protocol	Messages	Links	Messages	Links
RSP	27	27	33	33
Greedy	314	27	4450	27
QoSMIC	357	29	8029	31
MλT, $r = 1$	975	34	2991	42

TABLE III

AVERAGE NUMBER OF MESSAGES AND LINKS FOR DIFFERENT PROTOCOLS

be easily overcame thanks to MλT, even for $r = 1$.

MλT vs. RSP, Greedy, and QoSMIC in INET-3.0. We have performed the same experiments in a topology generated by INET-3.0 (cf. Figure 3). Again, the large average clustering coefficient of the topologies generated by INET-3.0 explains why MλT performs significantly better than all its three competitors. Moreover, Table III shows that the number of control messages produced by MλT is smaller than for Greedy, and significantly smaller than for QoSMIC. This is due to the greedy technique used to construct the multicast tree in both Greedy and QoSMIC. This greedy technique explores the network at successive distances $1, 2, 3, \ldots$ to eventually find a connecting point for a new member. Now, the number of nodes of the topology generated by INET-3.0 is more than 20 times larger than the number of nodes of UUNET. Moreover, the INET-3.0 topology is very dense (large average degree and high clustering coefficient). Finally, since the group members were selected uniformly at random among all nodes, they are far away from each others in average. Therefore, the distance between a new member and the current tree is expected to be large, yielding a large number of message for the connection.

IV. CONCLUSION

This paper has described the new multicast protocol MλT. Our first sets of simulations indicates that MλT scales better than QoSMIC, in the sense that the number of control messages of MλT in large networks (e.g., the one generated by INET-3.0) is smaller than the one of QoSMIC. Obviously, MλT must be periodically activated to maintain a dynamic QoS. To add new members, MλT must however be activated on the whole tree. A more appropriate solution would consist to interleave the use of MλT with the use of an on-line aggregation protocol such as RSP.

Acknowledgments: J. Ignacio Alvarez-Hamelin is also from the Facultad de Ingeniería of Universidad de Buenos Aires, and is supported by the ALPHA program CORDIAL. Pierre Fraigniaud is with CNRS, and is supported by the European Union's Fifth RTD Framework Program DAVID (Data and voice integration over DWDM).

REFERENCES

[1] C. Diot, W. Dabbous, and J. Crowcorff, "Multipoint Communication: A Survey of Protocols, Functions, and Mechanisms," *IEEE Journal on Seleted Areas in Communications*, vol. 15, no. 3, pp. 277–290, 1997.

[2] W. Mostafa and M. Singhal, "A taxonomy of multicast protocols for Internet applications," *Computer Communications*, vol. 20, pp. 1448–11 457, 1998.

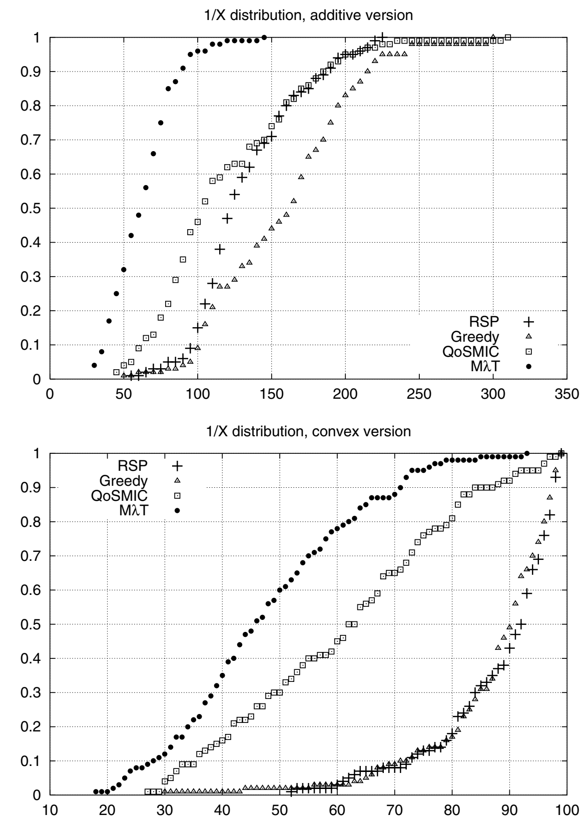


Fig. 3. Comparison of multicast protocols in INET-3.0 (3037 nodes), for a group of 20 members

[3] T. Ballardie, P. Francis, and J. Crowcroft, "Core based tree (CBT): an architecture for scalable inter-domain multicast routing," in *proceedings of SIGCOMM*. ACM press, 1993, pp. 85–95.

[4] D. Estrin, D. Farinacci, A. Helmy, D. Thaler, S. Deering, M. Handley, V. Jacobson, C. Liu, and L. Wei, "RFC-2362: Protocol Independent Multicast-Sparse Mode (PIM-SM) ," IETF, Tech. Rep., 1998.

[5] M. Faloutsos, A. Banerjea, and R. Pankaj, "QoSMIC: Quality of Service sensitive Multicast Internet protoCol," in *SIGCOMM*, September 1998, vancouver BC.

[6] K. Carlberg and J. Crowcroft, "Building Shared Trees using a one-to-many joining mechanism," in *ACM SIGCOMM Computer Communication Review*, January 1997, pp. 5–11.

[7] Z. Wang and J. Crowcroft, "Quality-of-service routing for supporting multimedia applications," *IEEE Journal of Selected Areas in Communications*, vol. 14, no. 7, pp. 1228–1234, 1996.

[8] J. Winick and S. Jamin, "Inet-3.0: Internet topology generator," Department of EECS, University of Michigan, Tech. Rep. UM-CSE-TR-456-02, 2002.

[9] Y. Rekhter, "RFC-1771: A Border Gateway Protocol 4 (BGP-4)," IETF, Tech. Rep., March 1995.

[10] G. Malkin, "RFC-2453: RIP Version 2," IETF, Tech. Rep., November 1998.

[11] J. Moy, "RFC-2178: OSPF Version 2," IETF, Tech. Rep., July 1997.

[12] A. Vazquez, R. Pastor-Satorras, and A. Vespignani, "Internet topology at the router and autonomous system level," *Physical Review*, vol. 65, no. 066130, 2002.