WAVE: A New Multicast Routing Algorithm for Static and Dynamic Multicast Groups

Ernst Biersack, Jörg Nonnenmacher
Institut Eurécom,
2229 Route des Crêtes,
06904 Sophia-Antipolis — France
e-mail: erbi@eurecom.fr

Abstract. We present a new multicast algorithm called *WAVE* for establishing source-specific multicast trees. WAVE meets multiple quality of service requirements (constraints) such as delay, cost, and available bandwidth, simultaneously. Simulation results show that WAVE performs very good in terms of delay and cost for both, static and dynamic multicast groups, when compared to the best multicast algorithms known.

1 Introduction

Many new applications in the area of multimedia such as teleseminars or distribution of news require multipoint connections.These applications also typically have complex quality of service (QOS) requirements concerning delay, cost, and bandwidth needed that must be taken into account (as constraints) by the multicast (MC) algorithm. Existing MC algorithms are only able to consider one or two constraints.

1.1 Notation

Before discussing the MC algorithms, we need to introduce some notation [1].

A network is represented as a graph $N = (V_N, E_N)$, where V_N is the set of nodes and

$E_N \subset V_N \times V_N$ is the set of edges. The average number of edges that depart from a node is referred to as *outdegree*. Over the set of edges we define the two functions delay $Delay$: $E_N \to R^+ \setminus \{0\}$ and cost $Cost$: $E_N \to \{1\}$. The delay and the cost of a path are defined as the sum of the delay or cost of all the edges of the path.

The multicast receivers are referred to as *MC group*, and $Q \in V_N$ is the source of the MC group (we assume that there is a single source in a MC group.)

The *multicast tree* $MCT_M = (V_M, E_M)$ with $V_M \subset V_N$ and $E_M \subset V_N \times V_N \subset E_N$ is a directed, acyclic subgraph of N with Q as root that connects all nodes in the MC group.

The cost of MCT_M is defined as the sum over the cost of its edges:

$$Cost_M = \sum_{e \in E_M} Cost(e) = |E_M| .$$

1.2 Shortest Path First (SPF)

One of the simplest MC algorithms is the shortest path tree (SPT) [1]. The MC tree for the SPT consists of the shortest paths -- in terms of delay -- from the sender Q to all receivers in the MC group. The shortest paths are established using the existing unicast routing algorithm. When a new receiver R joins a MC group, the sender Q determines the shortest path from Q to R. If the beginning of this path from Q to a node A is already in the MC tree, the MC tree needs only be extended by the shortest path from A to R. See figure 1, where the path from Q to R_3 overlaps from Q to A with the existing MC tree.

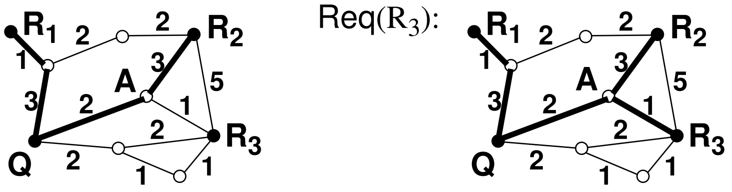


Figure 1: Shortest path tree.

While SPT minimizes the delay, it does not try to minimize the total cost of the MC tree.

1.3 Steiner Tree

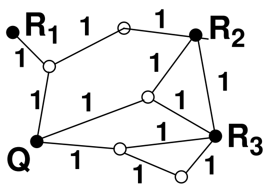
Another class of MC algorithms solve the Steiner Tree problem, which consists of computing the tree with the minimum cost that connects a subset M of the nodes. Minimizing the total cost of the MC tree goes at the expense of the delays from the sender to the members in the MC tree. The delays are much higher than for SPT.

Computing the Steiner Tree is an NP-complete problem. However, there exist good heuristics that run in polynomial time. One of them, proposed by Kou, Markowsky, and Berman (KMB) [3] computes trees that have approximately 5% higher costs than the cost for the minimal Steiner Tree.

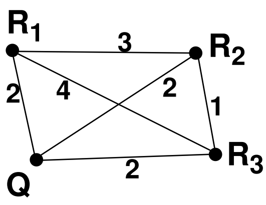
The KMB algorithm works as follows: (see figure 2)

- Starting from a graph G, a *complete* (every node is connected with every other node) graph G1 is constructed
- For G1, a minimal spanning tree T1 is constructed
- The edges in T1 are replaced by the shortest paths in G, which gives a subgraph G2.
- For G2, a minimal spanning tree T2 is constructed
- The branches in T2 that don't contain nodes that are members of the MC group are pruned.

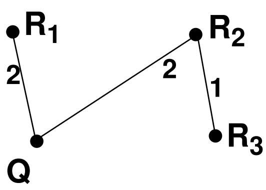
Graph G with MC group R_1, ..,R_3



Graph G1



Minimal spanning tree T1



Graph G2 after expansion

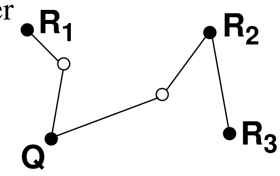


Figure 2: KMB algorithm.

1.4 WAVE

The basic principle of WAVE is very simple: When a node *R* wants to join a MC group, it sends a request Req to the source *Q*. Starting from *Q*, this request is propagated throughout the MC tree and answered (Rsp) by the nodes that received that request. A major advantage of WAVE is that complex QOS requirements can be taken into account since the path from the source *Q* to a new receiver is dynamically discovered. The Req and Rsp messages that are passed along the network can be used to dynamically collect and update information --such as delay, cost, or available bandwidth -- concerning the characteristics of the path. Each node that receives such a message can compare the QOS requested with the QOS characteristics of the path taken by this message. The message will only be forwarded if the path taken so far meets the QOS requirements. (In the following, we will for the sake of simplicity, only consider delay and cost as QOS requirements.) Each response received by *R* will have the form Rsp = (*n_id,cost,delay*), where *n_id* denotes the node that generated the response and the other two entries denote the cost and delay of the connection from the source *Q* to *R* via node *n_id*.

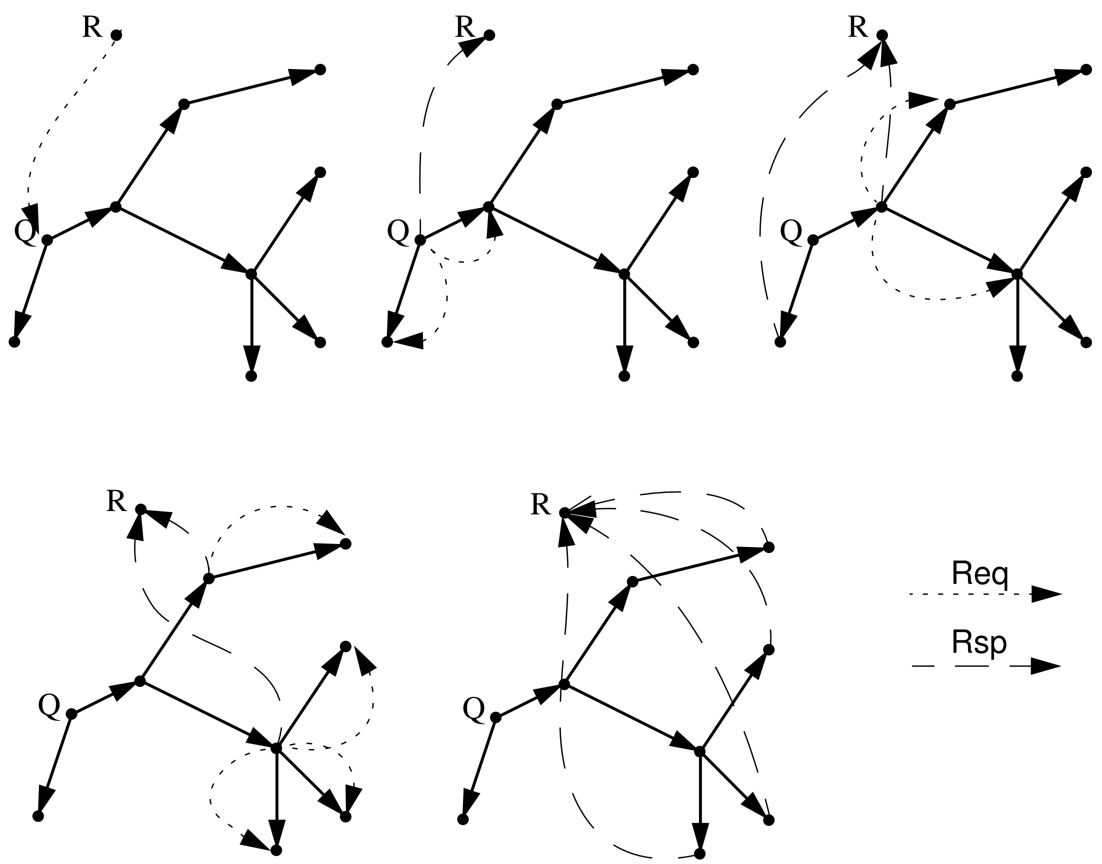


Figure 3: Requests and Responses in WAVE.

The basic steps to join a MC group are as follows (See figure 3 and figure 4):

- A new receiver *R* that wants to join the MC group contacts the source *Q* with a request Req.
- When the source or any other node in the MC tree receives a Req, it will send a reply Rsp to *R* along the shortest delay path.

- R will receive a set {Rsp} of responses from which it selects one Rsp = (*K,cost,delay*) (see below how). *R* then sends a connect to node *K* (*attachment node*) that generated this response.
- *K* will extend the MC tree from itself to *R* along the shortest path from *K* to *R*.

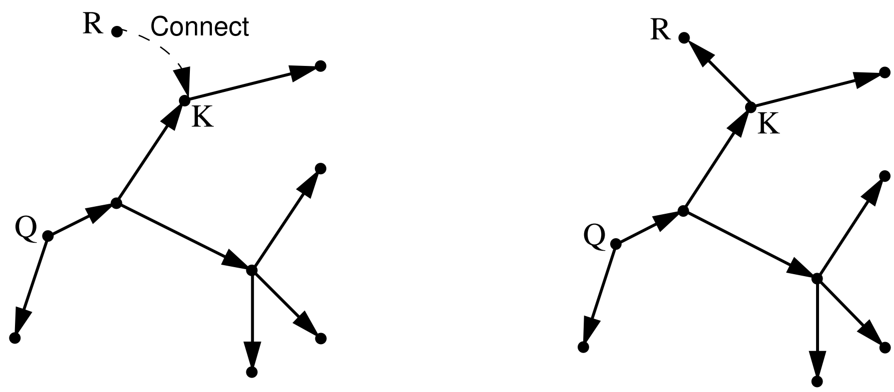


Figure 4: Selection of attachment node and extension of MC tree.

If every node in the MC tree generates are response, the number of responses will be proportional to the number of nodes in the MC tree. There are two situations where nodes can avoid producing a response or where the response is deleted by a later node.

- *Neighbor overlap*: When a node *K* produces a response, it checks if the link (K,K_s) over which *K* will send its response is already in the MC tree. If so, no response will be sent. It suffices that the neighbor node K_s generates/has generated a response. (See figure 5.)

Figure 5: Neighbor overlap.

- *Other overlap:* A response Rsp generated by *K* is sent along a path (K,K_1), ..., (K_l,K_m), where K_m is already in the MC tree. In this case, K_m must delete Rsp to ensure that the structure of the MC tree remains a tree. K_m itself will generate/has generated a response.

The results presented later in figure 11 indicate that suppressing responses whose path overlaps with the MC tree can reduce the number of responses by up 80%.

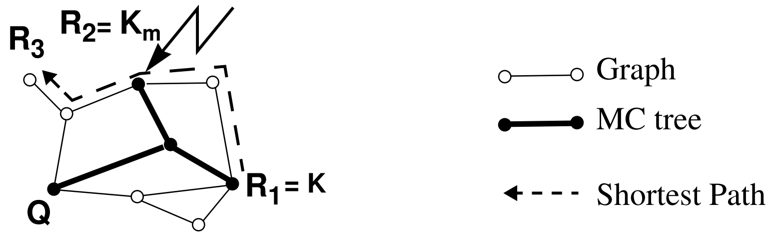


Figure 6: Other overlap.

In the following, we explain how the receiver selects the attachment node via which it will be connected to the MC tree.

Definitions

- $\Re \subset V_N$ defines the set of nodes in the MC tree from whom *R* has received a response.
- *MCA* denotes one of the three MC algorithms (WAVE, SPT, or KMB).
- $MCT_{MCA} = (V_{MCA}, E_{MCA})$ denotes the MC tree computed by MCA.
- The node $(K \in \Re)$ via which *R* will be connected to the MC tree is called **attachment node**.
- $Delay_{MCA}$ *(Q-->K)* denotes the delay from *Q* to *K* along the path in the MC tree MCT_{MCA}.
- $Delay_{sp}$*(K-->R)* denotes the delay along the shortest path from *K* to *R*.
- *Delay(Q-->K-->R)* denotes the delay from *Q* via *K* to *R* and is defined as *Delay(Q-->K-->R)* = $Delay_{MCA}$ *(Q-->K)* + $Delay_{sp}$*(K-->R)*.

Assumption

- The receiver *R* has received a set of responses {Rsp} = {(*K*, *Delay(Q-->K-->R)*, *Cost(K-->R)*, where $K \in \Re$ } from which he selects an attachment node $k \in \Re$.

Section of attachment node

- For each response $rsp \in \{Rsp\}$ with *rsp =(n_id,delay,cost)*, the receiver computes a **weighted cost** $WC(n\_id) = w_c*(cost/max\_cost) + w_d*(delay/max\_delay)$, where $w_c \in [0, 1]$, $w_d \in [0, 1]$ and *max_cost* and *max_delay* are the maximum cost and delay values over all responses {*Rsp*} received.
- The receiver calculates for all responses {*Rsp*} the weighted costs and selects the node $k \in \Re$ with the **minimal weighted cost**, i.e. $WC(k) \leq WC(K), \forall (K \in \Re)$.

The choice of the weights w_c, w_d allows the receiver to trade-off cost versus delay:

- For $w_c \in (0, 1]$ and $w_d = 0$, the receiver chooses among all the shortest-delay paths from *K* to *R* the path with the minimal incremental cost.
- For $w_c = 0$ and $w_d \in (0, 1]$, the receiver ignores cost and chooses the path with the shortest delay between *Q* and *R*. In this case, WAVE yields the same MC tree as SPF.

2 Performance Evaluation

We used simulation to compare the performance of WAVE against SPT and KMB and to evaluate the impact of the choice of w_c, and w_d on the performance.

2.1 Performance Metrics

The MC tree MCT_{MCA} has a set $\aleph$, $\aleph \subset V_{MCA}$ of receivers. $\aleph$ denotes the set of the MC group members.

- Cost of the MC tree is $Cost_{MCA} = |E_{MCA}|$
- Average delay from the source Q to any receiver R is
  $AvgRecvDelay_{MCA} = \frac{1}{|\aleph|}\sum_{R \in \aleph} Delay_{MCA}\langle Q \rightarrow R\rangle$
- Maximum delay from the source Q to any receiver R is
  $MaxRecvDelay_{MCA} = max_{R \in \aleph}\{Delay_{MCA}\langle Q \rightarrow R\rangle\}$

In the following, we define several ratios that allows us to relate the performance of WAVE and SPT (for delay comparisons) and WAVE and KMB (for cost comparisons) and indicate the *inefficiency* of WAVE with respect to the MC algorithm available.

- Cost ratio $CostR = \frac{Cost_{MCA}}{Cost_{KMB}}$
- Mean delay ratio $AvgRecvDelayR = \frac{AvgRecvDelay_{MCA}}{AvgRecvDelay_{SPT}}$
- Maximum delay ratio $MaxRecvDelayR = \frac{MaxRecvDelay_{MCA}}{MaxRecvDelay_{SPT}}$

To evaluate the overhead reduction in WAVE due to neighbor overlap and other overlap, we compute the ratio between the nodes (*requested*) that received a Req and the number of Rsp that arrived at the receiver (*answered*).

2.2 Simulation Environment

For our simulation, we applied the MC algorithms to a set of random networks. To generate these networks we use an approach introduced first by Waxman and later slightly modified by Wei&Estrin [4,5]. A random network is constructed by randomly placing its n nodes on a cartesian grid. The coordinates of the nodes are expressed as integers. To determine whether nor not to connect a pair of nodes *(u,v)* by an edge we evaluate the edge probability function $P_k(u,v)$ that is defined as

$$P_k(u, v) = \beta \cdot \exp\frac{-d(u, v)}{\alpha \cdot L}$$

where $d(u,v)$ is defined as the euclidean distance between (u,v), L is the maximum distance between any two nodes, α and β are parameters between $0 < \alpha, \beta \leq 1$. A large value for α increases the number of edges between nodes that are further apart, while a large value for β increases the outdegree. The delay of an edge is defined as $d(u,v)$.

For outdegrees that vary from 3 to 8, we produced 500 random networks each with 200 nodes.

To obtain a performance value, we fix all parameters and apply a MC algorithm to all 500 networks. Executing the MC algorithm for all 500 networks gives 500 samples for each performance metric. For the 500 samples, we then calculated the mean values and the 95% confidence intervals. The plots of the means are given with their confidence intervals.

When we consider the case that the MC group may evolve dynamically by nodes joining or leaving we use the function $P_A(k)$ introduced by Waxman [4]:

$$P_A(k) = \frac{\gamma(n-k)}{\gamma(n-k) + (1-\gamma)k}$$

where n denotes the total number of nodes, k the current number of receivers in the MC tree, and γ is a parameter between (0,1). γ represents the ratio #receivers/#nodes. For $\gamma = k/n$ we have $P_A(k) = 1/2$. To determine whether the next modification will be a join or leave, we compute a random number r, $0 \le r < 1$ to compare with $P_A(k)$. If $r > P_A(k)$, the modification is **leave** and randomly one of the receivers that will leave the MC group is determined. For $r \le P_A(k)$, the modification is **join** and a node is randomly selected as new receiver.

2.3 Results

We first consider scenarios where the MC group is **static**, i.e. the MC tree is constructed for a fixed group of receivers that does not change.

2.3.1 Choice of the weights

Before comparing the MC algorithms, we need to choose the values for the weights w_c, and w_d. In figure 7 and figure 8 we present the impact of the weights on the average cost and delay of the MC tree. The value of one weight is set to 1 while the value of the other weight varies between 0 and 1.

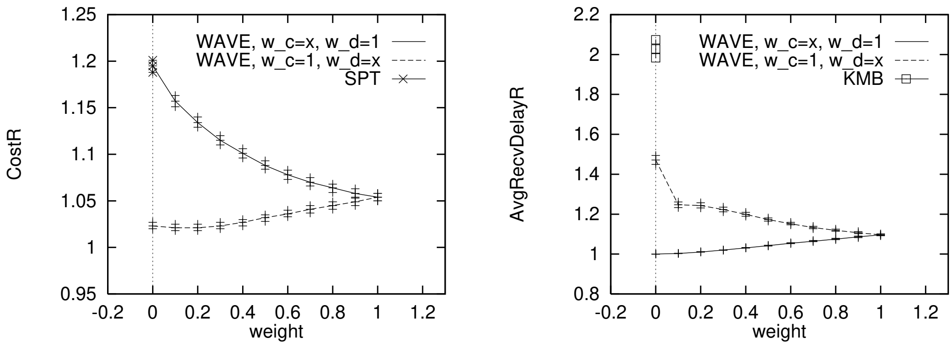


Figure 7: CostR and AvgRecvDelayR as a function of the weights (200 nodes, outdegree = 3, MC group size = 40).

The average cost of the MC tree for WAVE is between 2% (for $w_c = 1$ and $w_d = 0$) and 20% (for $w_c = 0$ and $w_d = 1$) higher than the cost of the MC tree for KMB. The higher the ratio between the cost weight and the delay weight, the closer the cost performance for WAVE approaches the cost performance of KMB. The delay performance of WAVE for $w_d > 0$ is at worst 22% higher than for SPT. An exception is $w_d = 0$, where the delay increases significantly. Even then, the delay performance of WAVE is still much better than for KMB where the average delay is about twice as high as for SPT. The values for MaxRecvDelayR, which are not plotted, are very similar to the ones for AvgRecvDelayR.

We are not only interested in the average delays but also in the distribution of the delays as presented in figure 8. While the x-axis gives that absolute delay values, the y-axis gives the number of receivers that experience a certain delay value. We can use the delay distribution to determine how many percent of the requests to connect to the MCT would fail if the delay constraint demands that the delay must be below a certain value. We see that the delay distributions for WAVE approach the delay distribution for SPT as the value of delay weight increases relative to the value for the cost weight. The delay distribution for WAVE and SPT has a much narrower shape than for KMB. The delay distribution for KMB has a long tail with delay values up to 12000, while the delay values for WAVE are never higher than 6000 for $w_d > 0$.

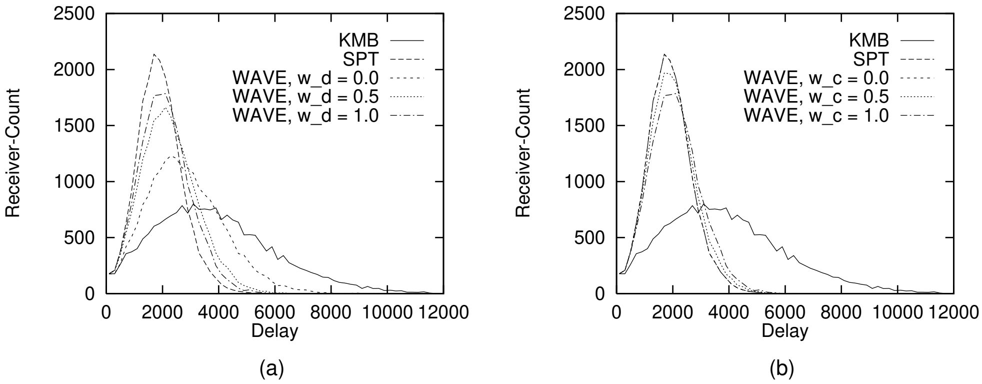


Figure 8: Delay histograms for (a) w_c=1 and (b) w_d=1
(200 nodes, outdegree = 3, MC group size = 40).

For w_d=1 and $w_c < 1$, the delay distributions for WAVE approximate the distribution of SPT closer than for $w_c = 1$ and $w_d < 1$. Therefore, for the following simulations, we fix w_d=1 and $w_c = 0.7$, in which case the delay and cost inefficiencies are both 1.07 compared to the best MC algorithm for either metric. (See figure 9).

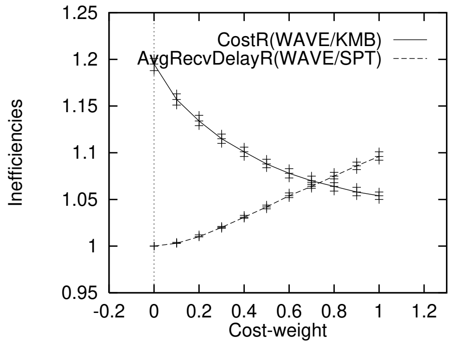


Figure 9: AvgRecvDelayR and CostR as a function of the cost weight
w_c for w_d=1 (200 nodes, outdegree = 3, MC group size = 40).

In summary, we can say that depending on the constraints (cost, delay) imposed, the weights for WAVE can be chosen in such a way that either the cost efficiency of KMB or the delay efficiency of SPT is achieved. An intermediate choice of the weights allows to achieve both, very good cost and delay efficiency.

2.3.2 Impact of MC group size and outdegree

We see in figure 10 the cost and delay efficiency as a function of the outdegree for two different MC group sizes of 5 and 40. The cost inefficiency for WAVE and SPT increases with increasing outdegree. This is due to the fact, that both, WAVE and SPT, use the shortest path to connect a new receiver to a node in the existing MC tree. As the outdegree increases the probability that at least part of this shortest path overlaps with the existing MC tree becomes lower (The decreasing number of neighbor overlaps in figure 11 for increasing outdegree corroborates this.) Since WAVE takes for $w_c > 0$ the cost of the new path into account, its cost inefficiency does increase slower than that of SPT.

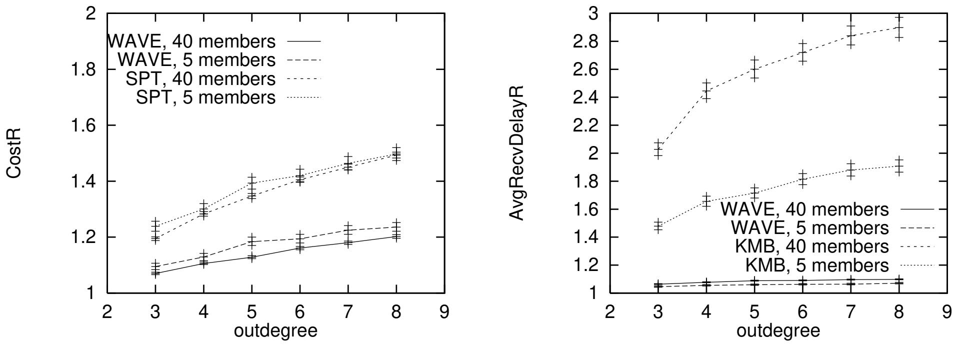


Figure 10: CostR and AvgRecvDelayR as a function of the outdegree for MC group sizes of 5 and 40 (200 nodes).

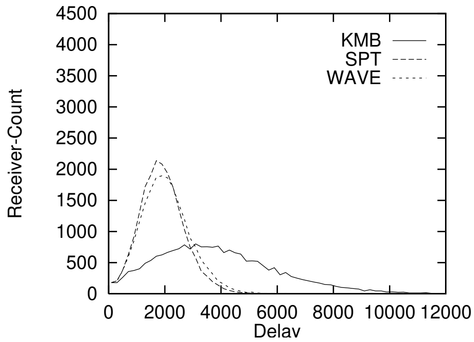
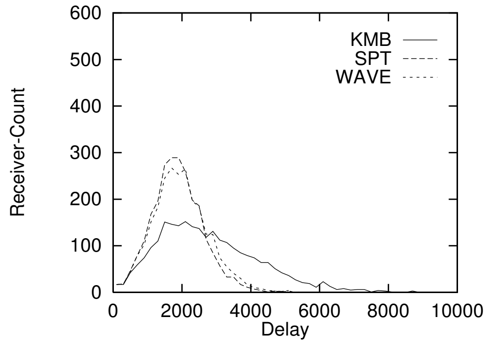
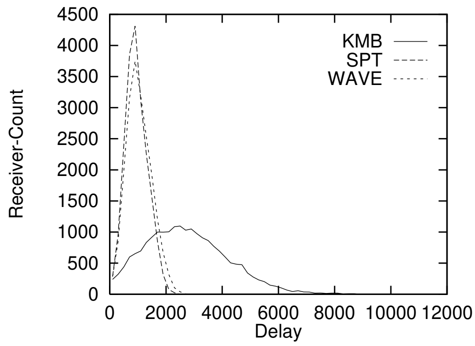
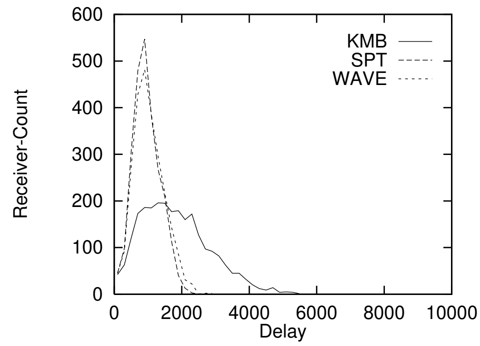
The delay inefficiency of WAVE stays the same, independent of the outdegree and MC group size. For KMB however, the inefficiency increases with the MC group size and with the outdegree because, as either one increases, KMB has more alternatives for constructing the minimal-cost MC tree. Therefore, the differences between the MC tree constructed by KMB as compared to SPT or WAVE become more pronounced with increasing MC group size or outdegree.

In Table 1 we see the distribution of the delays as a function of the outdegree for two different MC group size. The distributions for WAVE are very close to the ones for SPT. As the outdegree increases for a fixed MC group size, the delay distribution for WAVE and SPT becomes much narrower, i.e. the delays between the source and the receivers become smaller. For KMB, the reduction of the delay values is much less pronounced. A higher outdegree means richer connectivity, since each node has more neighbors. While WAVE and SPT use the richer connectivity to (predominately) optimize the delay, KMB optimizes the cost.

2.3.3 Overhead reduction due to neighbor overlap and other overlap

The following figure 11 shows the overhead incurred by WAVE. We see that there is a significant potential to reduce the overhead by eliminating Rsp messages that incur a neighbor overlap. Depending on the outdegree, the elimination of messages with neighbor overlap reduces the total number of messages that arrive at the receiver between 60% and 80%. The reduction is highest for low outdegrees. As the outdegree increases, more paths exist to connect a new receiver with the MC tree, therefore probability that the path chosen overlaps with the MC tree decreases. Also, the overhead reduction is more effective for larger MC groups for which the

Table 1: Delay histograms (200 nodes).

	MC group size 40	MC group size 5
outde-gree 3		
outde-gree 8		

probability that the shortest path overlaps with the MC tree increases.

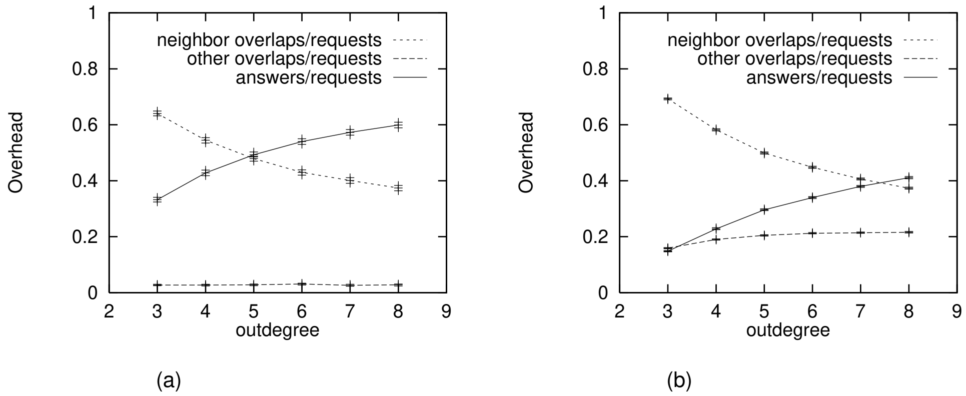


Figure 11: Answers received and overhead reduction for (a) MC group size = 5 and (b) MC group size = 40 (200 nodes).

2.3.4 Dynamic MC Groups

In many cases, the MC group changes when during the existence of a session new receivers join or leave the MC group. A MC algorithm should be able to allow for changes in the MC group without disrupting the communications -- by changing the paths -- between the source and existing members of the MC group. An algorithm such as KMB does not meet this requirement. Any change in the MC group members ship will require to recompute the complete MC tree. Changes in the MC tree therefore affect existing members. To compare WAVE and KMB with respect to the cost inefficiency, we recompute the MC tree using KMB after every 50 MC group modifications. The cost of the MC tree obtained for KMB was then compared with the cost of the MC tree for WAVE that was dynamically evolving with each modification. When interpreting the results, we therefore must keep in mind that this comparison is in some respect “unfair” towards WAVE because KMB is not able to “smoothly” grow the MC tree each time a change in the group membership occurs.

We see in figure 12 the cost and delay efficiency as a function of the outdegree for two different dynamic MC groups of size of 5 and 40. The cost inefficiency for WAVE increases slightly during the first 50 modifications and stays then at the this level for the remaining several hundred modifications. The average delay for WAVE is not affected at all by the modifications.

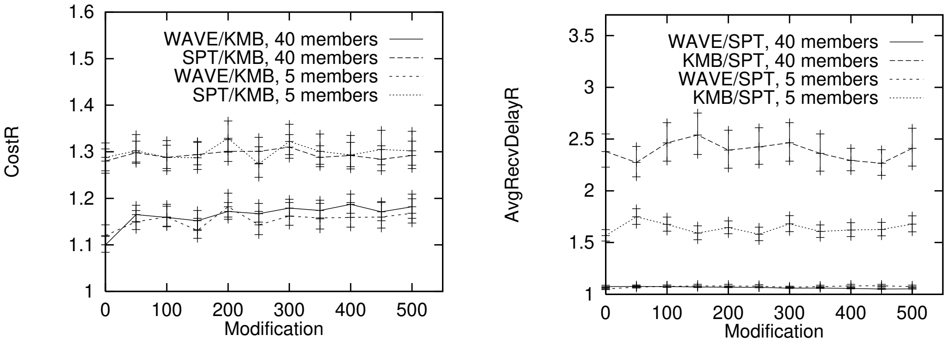
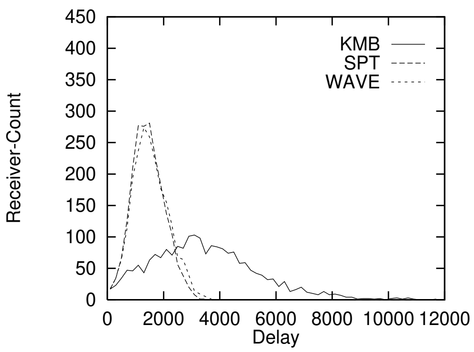
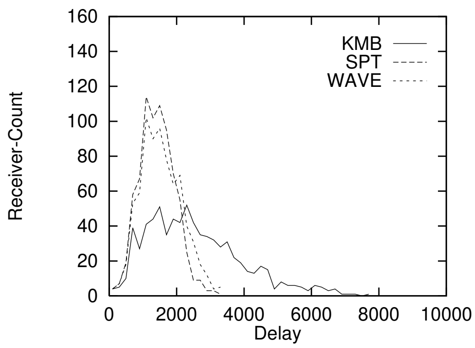
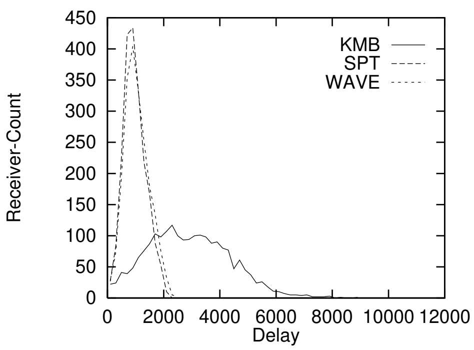
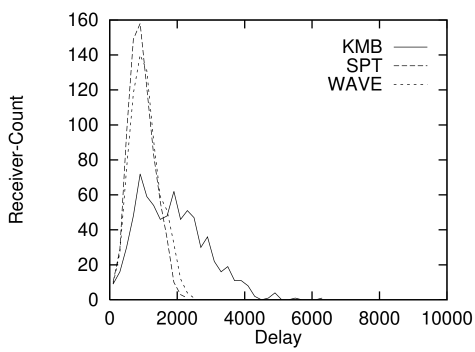


Figure 12: CostR and AvgRecvDelayR as a function of the number of modifi cation for MC group sizes of 5 and 40 (200 nodes outdegree=4).

In Table 2, we see the distribution of the delays as a function of the outdegree for two different MC group sizes after 500 modifications of the MC group (join and leave requests). The

Table 2: Delay histogram for different outdegrees and MC group sizes after 500 modifications.

	MC group size 40	MC group size 5
outde-gree 4		
outde-gree 8		

shape of the delay distribution for WAVE remains for all scenarios very close to the delay distribution of SPT that is not subject to any destination at all, because it extends for each new MC group member the MC tree by the shortest path from the source to the new member.

These results confirm that WAVE retains its excellent cost and delay properties for the important case where the MC group evolves dynamically.

3 Conclusion

WAVE is a flexible MC algorithm that allows to optimize the MC tree according to different criteria and achieves close to optimal performance. Depending on the constraint (cost, delay) imposed, the weights for WAVE can be chosen in such a way that either the cost efficiency of KMB or the delay efficiency of SPT is achieved. An intermediate choice of the weights allows to achieve both, very good cost and delay performance. The performance of WAVE is not affected, when the MC tree is subject to modifications.

Acknowledgements

We would like to thank John Matthew Doar for providing us with the simulation code for the KMB algorithm and Antoni B. Przygienda for comments on the presentation.

4 References

[1] T. H. Cormen, C. E. Leiserson and R. L. Rivest, *Introduction to Algorithms*, The MIT Press, 1990.

[2] M. Doar and I. Leslie, *How Bad is NaÏve Multicast Routing?* Proc. INFOCOM 1993, 82ff.

[3] L. Kou, G. Markowsky, and L. Berman, *A Fast Algorithm for Steiner Trees.* Acta Informatica, Vol. 15:141-145, 1981.

[4] B. M. Waxman, *Routing of Multipoint Connections.* IEEE J. Selected Areas in Communications, Vol. 6, No. 9, December 1988, p.1617 ff.

[5] L. Wei and D. Estrin, *The Trade-offs of Multicast Trees and Algorithms*. Proc. Int. Conf. on Computer Communications and Networks, 1994.

[6] R. Widyono, *The Design and Evaluation of Routing Algorithms for Real-time Channels,* University of California at Berkeley & International Computer Science Institute, TR-94-024, June 1994.