Heterogeneous Multicast in Heterogeneous QoS Networks

Jens Schmitt[1], Frank Zdarsky[1], Martin Karsten[1], and Ralf Steinmetz[1,2]

1: Darmstadt University of Technology
Merckstr. 25 • 64283 Darmstadt • Germany

2: GMD IPSI
Dolivostr. 15 • 64293 Darmstadt • Germany

Email: {Jens.Schmitt, Frank.Zdarsky, Martin.Karsten, Ralf.Steinmetz}@KOM.tu-darmstadt.de

Abstract

Supporting heterogeneous receivers in a multicast group is of particular importance in large internetworks as, e.g., the Internet due to the large diversity of end-system and network access capabilities. Furthermore, it is the nature of large-scale internetworks which makes homogeneous Quality of Service (QoS) support unrealistic at least for the middle-term future. Therefore, we investigate in this paper the implications of differing multicast models in heterogeneous QoS networks. In particular, we approach the problem an edge devices faces when mapping a heterogeneous QoS multicast from an overlaid QoS system onto a system providing only a homogeneous QoS multicast. The generic solution technique we propose for this problem is called foresting. The idea of foresting is to support a heterogeneous multicast by a forest of homogeneous multicast trees. We develop different foresting algorithms and compare them by extensive simulations.

1 Introduction

Taking into account multicast for network QoS systems is considered very important because many distributed applications that require QoS from the underlying network also require group communication mechanisms. In particular, many multimedia applications like, e.g., large-scale video-conferencing, computer-supported collaborative working (CSCW), or distributed multi-player games have fairly stringent QoS requirements on the one hand and need efficient network support for multicast transmissions, on the other hand.

1.1 Heterogeneous QoS Multicast

The support of heterogeneous QoS requests within a single multicast group can, combined with heterogeneous transmission facilities, be very useful to give various receivers (e.g., in multimedia application scenarios) exactly the presentation quality they desire and which they and the network resources towards the sender are able to handle. Such transmissions demand that the data to be forwarded can be somehow distinguished so that, e.g., the base information of a hierarchically coded video is forwarded to all receivers while enhancement layers are only forwarded selectively.

1.2 Heterogeneous QoS Networks

Heterogeneity with regard to network QoS seems inevitable in large-scale internetworks as, e.g., the Internet. As a consequence *edge devices* between different network QoS systems are required to mediate between the differen features and capabilities of these. In this paper, we investigate a technique that can be applied at edge devices if different multicast models are used in network QoS systems.

1.3 The Foresting Technique

If a QoS system that supports a homogeneous QoS multicast model is overlaid onto a QoS system that supports a heterogeneous QoS multicast model, there is obviously no problem. However, if a heterogeneous QoS multicast must be overlaid on a system that solely supports a homogeneous QoS multicast model, then there is an obvious mismatch that must be mediated by edge devices between the QoS systems. The technique that is proposed to deal with this problem is called *foresting*. The idea of foresting is to build a heterogeneous multicast tree from a collection of homogeneous multicast trees. In contrast to an actual heterogeneous QoS multicast foresting involves data duplication for links that are shared between multicast trees within the QoS system supporting a homogeneous QoS multicast model. In order to ease the discussions on the foresting techniques, we further on call homogeneous multicast trees *homMCT*. Furthermore, we distinguish the sending and the receiving edge devices for a multicast transmission as subnet-senders and -receivers or, simply as sender and receiver.

1.3.1 Application of Foresting

There are several scenarios where foresting as a generic technique is applicable.

RSVP/IntServ over ATM This is probably the most prominent instance of a heterogeneous QoS system where foresting is applicable. The problem is to find a collection of point-to-multipoint VCs from which the heterogeneous RSVP multicast tree (the part which is in the ATM network) is being constructed.

RSVP/IntServ over DiffServ Since the philosophy of DiffServ is to keep core routers simple it conflicts with heterogeneous transmission as these require very complex filtering functionality in routers. Therefore, it is likely to expect a DiffServ-based system not to support a heterogeneous QoS multicast model. However, this means that a heterogeneous QoS system where an RSVP/IntServ- is overlaid on a DiffServ-based system requires the application of foresting.

Hierarchical RSVP/IntServ A backbone provider may choose to disallow heterogeneous reservations despite using RSVP/IntServ as its QoS architecture due to missing filtering functionality in its high-speed routers. RSVP/IntServ-based access providers connected to the backbone provider may leave the heterogeneous QoS features turned on in their networks since their routers might be operating at a throughput that still allows for filtering to take place. Again, at the edge devices between the access and backbone providers, foresting techniques can bridge this gap.

1.4 Outline

In the next section, we present existing approaches to foresting for one particular instance of the general problem: RSVP/IntServ over ATM. In Section 3, the foresting problem is given a more detailed investigation, particularly with regard to its inherent complexity. Different foresting strategies are then presented and motivated by some numerical examples in Section 4. However, in order to obtain a better understanding of the performance of foresting heuristics, we also perform large-scale simulations in Section 5, before we then discuss some related work in Section 6 and draw some conclusions in Section 7.

2 Existing Approaches for RSVP over ATM

One instance of the heterogeneous over homogeneous multicast problem is for RSVP/IntServ over ATM and as such it has been taken up by the ISSLL (Integrated Services over Specific Link Layers) working group within the IETF. In particular, the following basic models to support heterogeneous RSVP/IntServ reservations over an ATM subnetwork have been proposed in [1]:

Full Heterogeneous Model In the full heterogeneous model, point-to-multipoint VCs are provided for all requested QoS levels plus an additional point-to-multipoint VC for best effort receivers. This leads to a complete preservation of the heterogeneity semantics of RSVP but can become very expensive in terms of resource usage since a lot of data duplication takes place.

Homogeneous Model In the homogeneous model solely one point-to-multipoint QoS VC is provided for all receivers including the best-effort receivers. The QoS VC is dimensioned with the maximum QoS being requested. This model is very simple to implement and saves VC space in comparison to the full heterogeneous model but may waste a lot of bandwidth if the resource requests are very different.

Another, quite different architecture for mapping RSVP/IntServ onto ATM is proposed in [2]:

Quantized Heterogeneous Model This model supports a limited number of QoS levels, including the best-effort class, for each RSVP multicast session. Each QoS level maps into one point-to-multipoint VC. While this proposal is an improvement over the very rigid models proposed by ISSLL, it says nothing about how to allocate the supported QoS levels for an RSVP multicast session. That means the concrete foresting strategy is left open to an edge device.

3 The Foresting Problem

In this section, we want to look at the inherent complexity of foresting by first taking a static view on the problem and then briefly discussing the implications for the dynamic problem as it has to be faced.

3.1 Static View

In the static case, it is assumed that all subnet-receivers and their requests are known beforehand. Assume we have N different resource request messages arriving at an ingress edge device. Suppose the receivers can be ordered by the size of their QoS requests (if that is reasonably possible, e.g., by regarding only their bandwidth requirements) and denote them from r_1 to r_N, i.e., r_1 is the highest and r_N the lowest request. That means if we define $q(r_i)$ as QoS requested by receiver r_i, then it applies that $\forall i,j$ with $i < j$: $q(r_i) > q(r_j)$. Call R the set of all subnet-receivers, $R = \{r_1,..., r_N\}$ and let

$f(S,q)$ = cost for a homMCT with QoS q from the subnet-sender to all $r \in S$;

$c(S) = f(S, q(r_{min}))$ for $S \subseteq R$, with *min* being the minimum index of all $r_i \in S$.

That means $c(S)$ represents the cost to set up a homMCT for a given set of subnet-receivers with differing QoS requirements, where the homMCT is dimensioned for the maximum QoS request (which is represented by the element with the minimum index in the set of subnet-receivers).

Call $p = \{R_1,...,R_n\}$ a partition of R, if $R_1 \cup \ldots \cup R_n = R$ and $\forall i,j$: $R_i \cap R_j = \varnothing$. Thus, the foresting problem is:

Find p of R such that $\sum_{i=1}^{n} c(R_i)$ is minimized.

Such a partition is called a cost-optimal partition, p^{opt}.

Note that $p = \{R\}$ is the homogeneous model whereas $p=\{\{r_1\},..., \{r_N\}\}$ is the full heterogeneous model from Section 2. To assess how difficult the foresting problem is, consider the size of the partition space, $S_P(N)$:

$$|S_P(N)| = \begin{cases} \sum_{k=0}^{N-1} \binom{N-1}{k} |S_P(N-k-1)| & \text{if } N > 1 \\ 1 & \text{if } N = 0, 1 \end{cases}$$

This recursive formula can be explained by the observation that all partitions can be viewed as having r_1 and a k-elementary subset of the remaining (N-1) receivers as one homMCT and for the remaining homMCTs of the (N-k-1) receivers, we have $|S_P(N\text{-}k\text{-}1)|$ alternatives (per definition). Some example values of $|S_P(N)|$ are given in Table 1.

N	2	3	4	5	6	7	8	9	10	15
$\|S_P(N)\|$	2	5	15	52	203	877	4140	21147	115975	$1.38\text{x}10^9$

Table 1: Growth of the partition space.

It is obvious that for a high number of different reservation requests, the partition space becomes too large to be searched exhaustively while for smaller numbers, this should still be possible. Keep in mind that N is the number of different reservation requests which should be bounded by the number of scaling levels the data transmission system is able to support.

3.2 Dynamic View

In the dynamic case, which is the actual problem, we need to investigate foresting strategies when the set of different receivers is changing in time, i.e., instead of R, we now have R^t with discrete time steps t = 0,1,2,...Thus, we can view the search for the cost-optimal partitions of R^t as a series of static case foresting problems which, however, have a certain relationship.

4 Heuristics for Foresting

As we have illustrated in the preceding section, the foresting problem represents a difficult problem, therefore we directly chose to look for heuristic techniques to approach it, since computationally intensive algorithms are prohibitive as foresting decisions lie on the control path of network QoS systems.

Here, we distinguish between static and dynamic algorithms. Static algorithms only look at the problem at a certain point in time and compute a certain partition independent of the current partition. Hence, they need to be repeated whenever the set of receivers changes in some way. Dynamic algorithms consider the current partition and try to evolve it in a way such that “good” partitions result.

4.1 Static Heuristics

4.1.1 Simple Static Heuristic

A greedy algorithm that operates on the sub-space of ordered partitions is given in Figure 1. With link bandwidth consumption of a set of receivers, we mean the sum of bandwidth consumptions per link for the homMCT which would be built from the ingress edge device to the subnet-receivers while the rest of the notation is analogous to the definitions in Section 3 (with `V` and `H` as auxiliary sets of subnet-receivers and brackets instead of subscripts).

```
k = j = 1; V = R;
WHILE (V NOT empty) DO  //for all receivers
  R[k] = r[j];// start new homMCT
  V = V - r[j];
  L' = INFINITY;
  WHILE (V NOT empty) AND (L < L') DO
  j++;
  H = union(R[k], r[j]);
  L = link bandwidth consumption of H;
  L' = link bandwidth consumption of R[k] +
      link bandwidth consumption of {r[j]};
  IF (L <= L')
    R[k] = H; // adding successful
    V = V - {min V};
  ELSE
    j--; // start new partition
    k++;
```

Figure 1: Greedy algorithm.

The heuristic that is essentially applied by that greedy algorithm is to group together adjacent requests, where adjacency is defined with respect to topology and resource requirements (note that it is assumed that the receivers are ordered as described in Section 3). This is due to the observation that it makes little sense to have very different (with respect to their reservations) receivers in the same homMCT if they are far apart from each other because that would waste a lot of bandwidth for the part of the homMCT that is unique to a receiver with low resource requirements. Obviously, the complexity of the greedy algorithm is $O(N)$ since for every receiver there is one decision in which subpartition to place it. This is, of course, a very low complexity compared to the state complexity of the problem as it has been presented in Section 3.

4.1.2 Numerical Example

To show what results can be achieved with this simple algorithm, consider the example network in Figure 2 which represents a model of the topology of the NSFNet (National Science Foundation Network) backbone as of 1995 [3]. Let us suppose that the following reservations have been issued by the subnet-receivers: `r[1]`=10 Mb/s, `r[2]`=8 Mb/s, `r[3]`=4.5 Mb/s, `r[4]`=3 Mb/s and `r[5]`=2 Mb/s. Applying the algorithm to the example network gives the partition: GA={{`r[1]`,`r[2]`},{`r[3]`,`r[4]`},`r[5]`}, with $L(GA)$=118

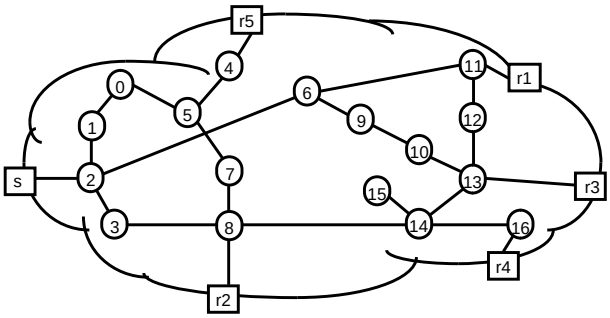

Figure 2: NSF backbone as example network.

as the sum of link bandwidth consumption of the three homMCTs (using classical minimum spanning trees for the computation of the homMCTs as an example routing strategy). Compare this to the full heterogeneous model from, *FH*={r[1],...,r[5]}, with *L*(*FH*)=129, or to the homogeneous model, *H*={{r[1],...,r[5]}}, with *L*(*H*)=170. So, *GA* saves about 30% link bandwidth inside the underlying QoS system relative to *H*. Actually (as a total enumeration shows), *GA* is the optimal partition (with respect to link bandwidth consumption).

The greedy algorithm, of course, does not guarantee an optimal solution. Consider, for example, that r[3]=5 Mb/s and everything else unchanged. Then the algorithm gives *GA*={{r[1],r[2],r[3]},r[4],r[5]} with *L*(*GA*)=130 but the optimal partition *O*={{r[1],r[2]}, {r[3],r[4]}, r[5]} has *L*(*O*)=122 (note that *L*(*FH*)=132 and *L*(*H*)=173 for this configuration).

4.1.3 Improved Static Heuristic

While for the examples above, only ordered partitions were optimal, it should be noted that this is not necessarily the case as the simple example in Figure 3 shows. Suppose that r[1]=9 Mb/s, r[2]=5.5 Mb/s and r[3]=3 Mb/s. Then the greedy algorithm gives *GA*={{r[1]},{r[2]},{r[3]}} with *L*(*GA*)=64.5 whereas the optimal partition is *O*={{r[1],r[3]},{r[2]}} with *L*(*O*)=61.5 (*L*(*FH*=*GA*) = 64.5, *L*(*H*)=63). This pathology of the simple static heuristic is due to not taking into account the relative topological position of the subnet-receivers.

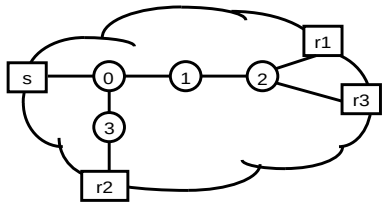

Figure 3: Example of an unordered optimal partition.

An improvement of the simple heuristic is based on sorting the subnet-receivers not only according to their resource requirements but also with respect to the distance between them. The algorithm in Figure 4 generates such an ordering of requests. In this algorithm, r[i].B denotes the capacity

```
sorted = FALSE;
WHILE NOT sorted
  sorted = TRUE;
  FOR i = 1 TO N-2
    IF (H(r[i],r[i+1])*(r[i].B - r[i+1].B) >
        H(r[i],r[i+2])*(r[i].B - r[i+2].B)
      exchange r[i+1] and r[i+2];
      sorted = FALSE;
```

Figure 4: Distance-oriented sorting algorithm.

requested by subnet-receiver r[i] and H(r[i],r[j]) denotes the number of links that are shared between the shortest path from subnet-sender to receiver r[i] and r[j]. It is assumed again that the receivers are initially sorted by their resource requests. After the sorting of receivers is done in this fashion, the simple greedy algorithm is applied again to do the partitioning of receivers into homMCTs. Applying this improved algorithm to the example given above now results in the partition *DGA*={{r[1],r[2]}, r[3]}. *DGA* is equal to *O*, and thus also achieves *L*(*DGA*)=61.5. For the example given in Figure 2, the distance-oriented greedy algorithm results in the same partition as the simple version, which means that the distance-oriented algorithm cannot guarantee optimality, either. The complexity of the distance-oriented greedy algorithm is $O(N^2)$ since the computation of the shortest paths (e.g., using Dijkstra's algorithm) and the sorting algorithm are $O(N^2)$ operations whereas the partitioning is $O(N)$.

4.2 Dynamic Heuristic

A disadvantage of applying the static heuristics independently to the series of static case problems is that relationships between successive partitions are not taken into account. This might lead to a large number of membership changes for receivers. To remedy this problem the dynamic heuristic given in Figure 5 follows the rationale to search for a partition in the "neighborhood" of the existing partition.

```
let rnew be a new a receiver;
k = n+1;
M = {rnew};
Lmin = link bandwidth consumption of M;
FOR i = 1 TO n DO //for all existing homMCTs
  H = union(R[i], rnew);
  Linc = link bandwidth consumption of H -
     link bandwidth consumption of R[i];
  IF (Linc <= Lmin) // cheaper join found
    k = i;
    M = H;
    Lmin = Linc;
R[k] = M;
```

Figure 5: Centralized dynamic heuristic algorithm.

The dynamic algorithm tries to incrementally add a new receiver to one of the existing homMCTs or to set up a new homMCT if this is "cheaper" with respect to link bandwidth consumption. The results of the dynamic heuristic are dependent on the order in which receivers make their reservations. Consider the example network in Figure 2 again. Suppose the receivers issue their requests (as in Section 4.1.2 with r[3]=5 Mb/s) in descending order of capacity demands. The resulting partition is the one that is also generated by the static greedy algorithm However, if the receivers are assumed to issue their reservations in order of ascending capacity, the resulting partition is becomes *D*={r[1],{r[2],r[3],r[4]},r[5]} with *L*(*D*)=124.
While this is not the optimal partition (*L*(*O*)=122), it is better than the partition calculated by the static greedy heuristic (L(GA)=130).

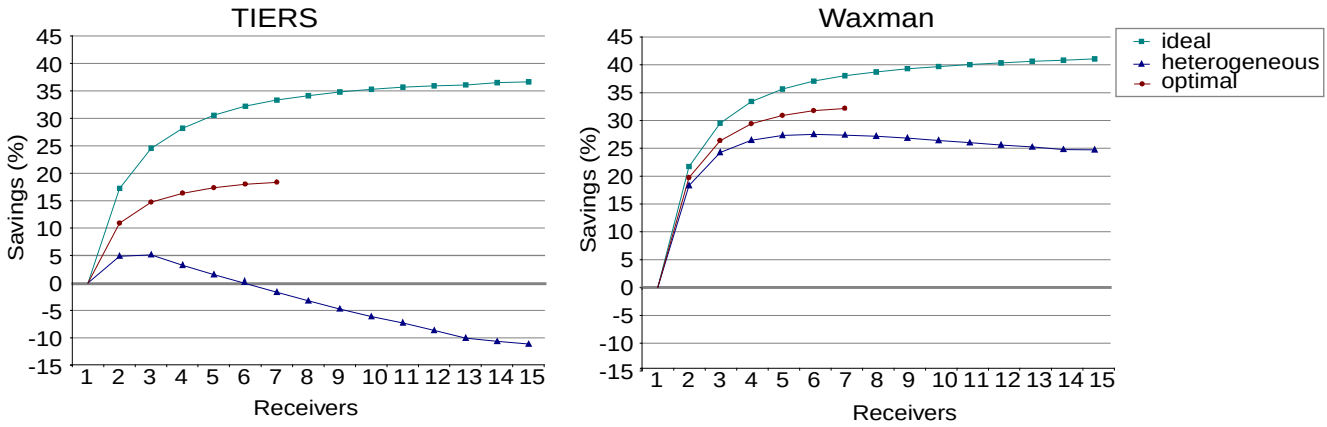

Figure 6: Bandwidth savings relative to homogeneous model.

5 Simulations

While in the preceding sections several foresting algorithms have been presented and some illustrative examples have been used to give an intuition of how these might perform, we now present more thorough results from extensive simulations of the foresting problem.

5.1 Simulations Setup

All of the simulation experiments are based on topologies produced by Waxman [4] and TIERS [5] topology generators. For both methods three topologies have been generated. For the hierarchical network models produced by the TIERS generator, the topologies vary in the number of nodes in the core and access levels as well as in the degree of edge redundancy that is introduced within and between the levels. For the flat networks produced by the Waxman generator, the topologies vary in the number of nodes and the density of edges to nearby as well as distant nodes. All simulations are repeated until the length of the 95% confidence interval falls below 0.1% of the sampled value.

In order to compare different strategies, the homogeneous model described in Section 2 is often used as reference value for the outcome of foresting strategies.

5.2 Simulative Experiments

Experiment 1: Possible Savings in Different Topologies

In this first experiment, it shall be analyzed which savings are principally possible by using intelligent foresting strategies instead of the very simple schemes proposed in Section 2. We compare the optimal foresting strategy, i.e., one that always computes the currently optimal partition whenever a change occurs, with the homogeneous and heterogeneous model. Furthermore, we evaluate the resource consumption of these with the “ideal” situation where the underlying QoS system would offer a heterogeneous multicast model. In every simulation, 8 nodes are chosen randomly, each one of them acting as sender in one multicast group and as a potential receiver in the 7 others. Additionally 8 nodes are chosen exclusively as potential receivers in all multicast groups, so that the dynamic reservation scenarios range from 1 to 15 receivers per group. However, due to the high computation times for searching the whole partition space, the optimal foresting algorithm has to be restricted to only calculate partitions for up to 7 receivers. Per tick of the simulation clock, every receiver has the chance of sending a request for adding, dropping, or changing a reservation to all of the multicast groups. The probabilities for doing so were set to 0.2, 0.15, and 0.4 respectively. The amount of requested capacity in existing or modified requests is selected in steps of 0.5 Mb/s up to a maximum of 10 Mb/s from a uniform random distribution.

Figure 6 shows the sampled results for the different strategies with respect to link bandwidth consumption for both kinds of topologies. The x-axis represents the number of receivers with bandwidth reservations while the y-axis indicates how much link bandwidth could be saved by the different approaches relative to the homogeneous model. As can be seen, the different topologies lead to very different results. While in the flat topologies, the optimal foresting strategy can save up to 33% over the homogeneous model, it only saves up to 18% in the hierarchical topologies. On top of that, it can be observed that the full heterogeneous model performs well in flat topologies whereas in the hierarchical topologies, it performs even worse than the homogeneous model. Interestingly, the ideal scenario with a heterogeneous multicast model offered by the underlying QoS systems is not so sensitive to the topologies.

Experiment 2: Performance of the Heuristics

Now, let us take a closer look at the performance of the heuristics that have been introduced in the preceding section. So, one issue is how much of the savings these heuristics actually achieve and another issue is how they compare to each other. The same settings as in the previous experiment are used. Figure 7 shows the simulation results for all of the heuristics (again relative to the homogeneous model).

Besides the two static heuristics based on the greedy algorithm of Section 4.1 which, however, differ in their sorting

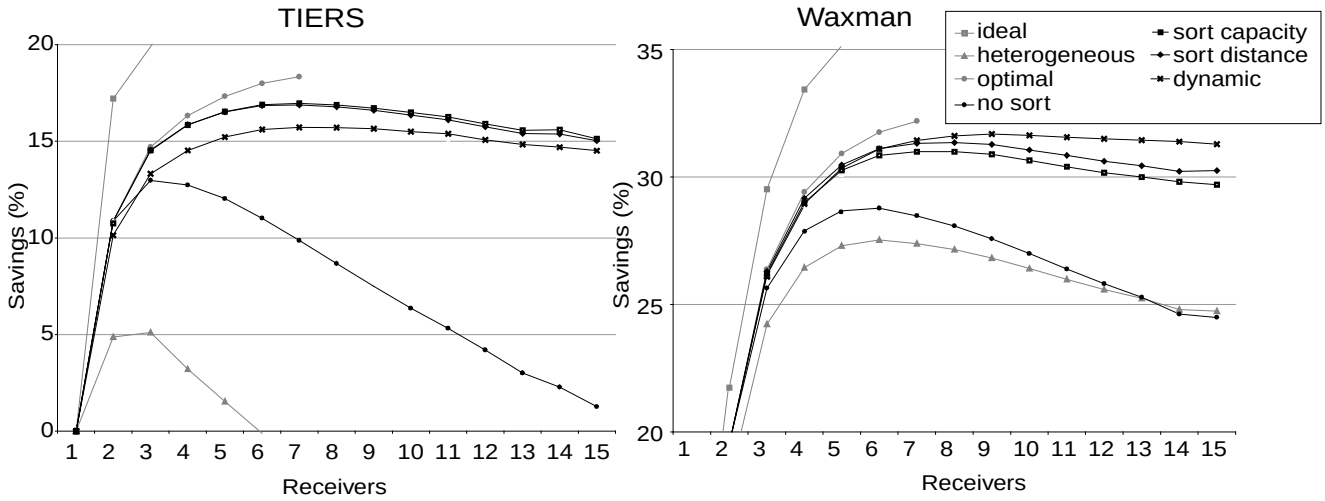

Figure 7: Results of the heuristics.

of receivers, we also use a static heuristic based on the greedy algorithm that does not sort the receivers at all. This is done in order to find out how the sorting affects the quality of the static heuristics. In both types of topologies, the heuristics perform pretty well as they only remain 2-3% below the savings of the optimal foresting strategy. With an increasing number of receivers, these savings decrease slightly. The static heuristic without any sorting of the receivers performs just a little bit better than the full heterogeneous model in flat topologies, and shows the same rapid decrease for hierarchical topologies. Therefore, the sorting of receivers, either with respect to the requested capacity or with respect to distance and capacity, is a good choice. The higher effort for the distance-sorting does not seem to pay off. The dynamic heuristic performs considerably better in flat topologies where it even outperforms the static heuristics. However, in hierarchical topologies, they perform worse than the static heuristics with sorting.

6 Related Work

To our knowledge, there is actually no work that treats the foresting problem in the generic sense we have done in this paper. The only other work we are aware of that is directly comparable to ours is [6] which looks at the special instance of foresting for RSVP/IntServ over ATM. The authors of [6] developed an almost identical version of the greedy static heuristic from Section 4.1. Yet, [6] provides no rationale for choosing the static greedy algorithm by comprehensive simulations or analysis of the design space for foresting heuristics as we have done in this paper. As described in Section 2, the work in [2] surrounding the quantized heterogeneous model is related to ours but in a complementary way. [2] gives the protocol support that is required for a foresting strategy to be implemented in an RSVP/ATM edge device based on their special solution for RSVP/IntServ over ATM.

7 Conclusions

In this paper, we have dealt with a specific control path problem in heterogeneous QoS systems: the provision of heterogeneous QoS multicast over a QoS system that only supports a homogeneous QoS multicast model. We have called the generic technique to deal with this situation foresting. We have analyzed the foresting problem, and showed that it is a very complex problem in general. Hence, we have derived heuristic techniques to deal with the problem. By some numerical examples, we have motivated these heuristics, and showed that they have a considerable potential to save resources when compared to standard approaches. To reinforce and verify these observations, we have conducted large-scale simulations. These simulations have shown that up to 30% bandwidth savings can be achieved by clever, yet, still simple foresting heuristics when compared to standard models of dealing with the problem.

References

[1] L. Berger, E. Crawley, S. Berson, F. Baker, M. Borden, and J. Krawczyk. A Framework for Integrated Services with RSVP over ATM. Informational RFC 2382, August 1998.

[2] L. Salgarelli, A. Corghi, H. Sanneck, and D. Witaszek. Supporting IP Multicast Integrated Services in ATM Networks. In *Proceedings of SPIE Voice and Video '97*, SPIE, November 1997.

[3] J. Jamison and R. Wilder. vBNS: The Internet Fast Lane for Research and Education. *IEEE Communications Magazine*, 35(1), January 1997.

[4] B. M. Waxman. Routing of Multipoint Connections. *IEEE Journal of Selected Areas in Communication*, 6(9):1617–1622, December 1988.

[5] M. B. Doar. A Better Model for Generating Test Networks. In *Proceedings of IEEE GLOBECOMM'96, London, England*, pages 86–93. IEEE, November 1996.

[6] D. Lee and K. Kim. Virtual Circuit Connection Method for RSVP Multicasting Supporting Heterogeneous Receivers. *IEE Electronic Letters*, 34(15):1474–1476, July 1998.