QoS Scalable Tree Aggregation

Joanna Moulierac, Alexandre Guitton

QoS Scalable Tree Aggregation

Joanna Moulierac and Alexandre Guitton

IRISA/University of Rennes I, 35 042 Rennes Cedex, France
{joanna.moulierac, alexandre.guitton}@irisa.fr

Abstract. IP multicast is not widely deployed over Internet. One of the reasons which prevents its deployment is multicast forwarding state scalability and control explosion. In this paper, we propose an algorithm called Q-STA (QoS Scalable Tree Aggregation) which reduces the number of forwarding states by allowing several groups to share the same tree (the less trees, the less forwarding states). Q-STA performs fast aggregations by evaluating few trees for each aggregation while increasing the number of accepted groups. Q-STA builds trees by using links having a low utilization in order to achieve load balancing. Moreover, groups are aggregated to trees having a minimum number of links in order to minimize the network load. By extensive simulations, we show that Q-STA outperforms the previous algorithm.

Keywords: multicast, tree aggregation, QoS, Aggregated Multicast.

1 Introduction

Multicast is becoming increasingly popular with the emergence of multimedia group applications: audio-video conferencing, Internet video-games or Internet TV. However, the deployment of multicast is limited mainly due to multicast forwarding state scalability and control explosion [1,2]. In traditional multicast [3], each group is assigned a multicast address corresponding to a tree delivery structure. This multicast address is stored in each on-tree router. Consequently, the number of forwarding states increases with the number of trees and then with the number of groups. As the number of groups becomes large, the number of forwarding states increases, slowing down the overall traffic. Tree aggregation is a recent approach to deal with this problem. In this paper, we focus on tree aggregation with groups having bandwidth requirements.

Tree aggregation. Tree aggregation consists in having several groups sharing the same delivery tree within a domain. In this way, less trees are maintained in the network. Consequently, the number of forwarding states in routers is decreased, together with the control overhead of messages used for tree maintenance.

To achieve tree aggregation, ingress routers of a domain add to the multicast messages for a group g a label l specific to the domain. Egress routers restore

the original group address of the packets that are forwarded outside the domain. Tree aggregation is achieved by assigning the same label l to several groups. In this way, the routers of the domain have only a forwarding entry by label and not by group.

Related work. The first algorithm achieving tree aggregation is Aggregated Multicast (AM), described in [4,5]. In AM, each time a new group arrives, candidate trees for the aggregation are searched in the multicast tree set. A candidate tree covers the new group and satisfies a given bandwidth constraint. Among all the candidate trees, the one minimizing the bandwidth wasted is chosen. If no candidates are found, a native tree for the group is built and added to the multicast tree set.

The algorithm Aggregated Multicast using Bi-directional trees (BEAM), described in [6], can be seen as a distributed version of AM. In BEAM, each arriving group is assigned a core router. Each core router is in charge of aggregating the groups with trees of its own multicast tree set. If a core is incapable of finding a candidate tree, it requests the other cores to aggregate the new group. As it is shown in [6], BEAM balances the load compared to AM.

The algorithm Aggregated Multicast Based on Tree Splitting (AMBTS) is an algorithm recently proposed in [7]. It introduces a new concept: several sub-trees are used for a group. All these sub-trees are shared by several groups. Using AMBTS reduces the complexity of finding a (sub-)tree aggregation for a new group.

These three algorithms, AM, BEAM and AMBTS, do not deal with bandwidth constraints. The algorithm Aggregated QoS Multicast (AQoSM), described in [8], is a framework based on AM to support QoS multicast. In the model of AQoSM, each link of the network is assigned a capacity and each group is assigned a bandwidth requirement. The goal of AQoSM is to aggregate groups to trees while respecting the bandwidth requirements.

In this paper, we propose a new algorithm called QoS Scalable Tree Aggregation (Q-STA), based on the framework of AQoSM. First, Q-STA performs faster aggregations than AQoSM by evaluating few trees while AQoSM evaluates all the trees of the multicast tree set each time a new group arrives. Second, Q-STA builds trees that use links having a low utilization. Thus, trees are candidates for further aggregations and the load of the network is balanced. Third, the groups are aggregated to trees having a minimum number of links in order to spare the network resources. We show that Q-STA evaluate less trees and accepts more groups than AQoSM.

Plan. In Section 2, we describe the tree aggregation principles together with Q-STA algorithm. In Section 3, we describe several metrics for tree aggregation. In Section 4, we run extensive simulations to compare AQoSM and Q-STA. Then, we analyze the results and show that Q-STA outperforms AQoSM.

2 Tree aggregation and Q-STA algorithm

In this section, we first describe tree aggregation and then tree aggregation with bandwidth constraints. Then, we describe our algorithm Q-STA.

2.1 Tree aggregation

Tree aggregation reduces multicast forwarding states. The key idea of tree aggregation is to force several groups sharing the same delivery tree.

A centralized entity, the tree manager, is responsible of maintaining the trees used by the groups and assigning trees to new groups. The tree manager knows the network topology, the available bandwidth on the links and the multicast groups with their assigned trees and their bandwidth requirements.

We present tree aggregation on Fig. 1. Three groups are depicted: g_1 (with members in routers A, D, E, G), g_2 (with members in routers A, B, E, G), and g_3 (with members in routers A, E, G). Without tree aggregation, three trees t_1, t_2 and t_3 are needed for the three groups g_1, g_2 and g_3. With tree aggregation, only one tree t is needed for these three groups. There is only one forwarding state in routers A, B, C, D, E and G instead of three without aggregation. The router F has no forwarding state with tree aggregation whereas F has one forwarding state without tree aggregation.

Tree aggregation reduces the number of maintained trees together with the number of forwarding states in routers. We proposed an algorithm called Scalable Tree Aggregation (STA) [9] achieving fast aggregations.

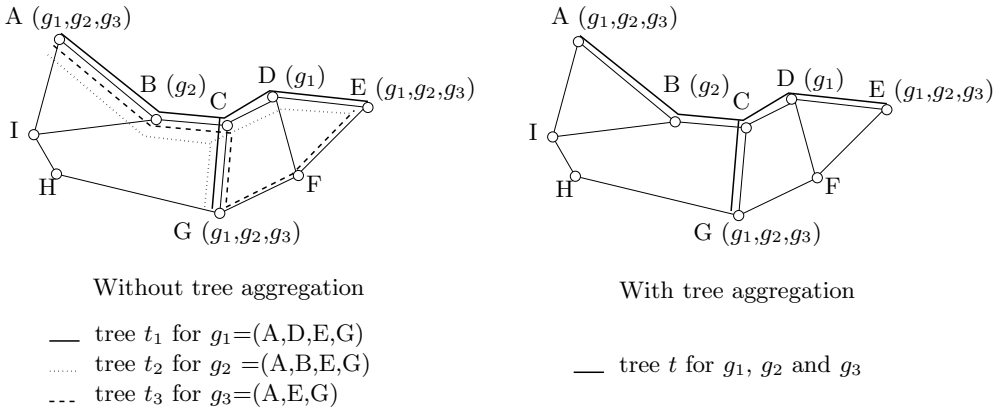

Fig. 1. The groups g_1, g_2 and g_3 can share the same tree t.

2.2 Tree aggregation with bandwidth constraints

We focus on tree aggregation when groups have bandwidth requirements and links have limited bandwidth capacity. When a group g is accepted, bandwidth

is reserved along the links of its tree. A group g is refused when the bandwidth requirement of g exceeds the available bandwidth of the network. In this case, the tree manager cannot find any tree covering g with enough available bandwidth on the links.

2.3 Q-STA algorithm

Q-STA is a proposition that achieves tree aggregation with bandwidth constraints. Q-STA is based on AQoSM framework and Q-STA is an extension of our proposition STA [9]. Q-STA performs fast aggregations, build efficient trees for new groups and aggregate groups to good trees.

Main ideas of Q-STA:

1. *Q-STA performs fast aggregations.* Q-STA evaluates only the trees that are the most likely to be candidates whereas AQoSM evaluates all the trees of the multicast tree set each time a new group arrives.
2. *Q-STA builds efficient trees.* Each time a new group g arrives, Q-STA builds a native tree maximum available bandwidth. Thus, the links are not overloaded unnecessarily and the tree can be candidate for further aggregations. Moreover, the trees use links having a low utilization and then load balancing is achieved.
3. *Q-STA aggregate groups to good trees.* Q-STA aggregates the group g to the tree covering g with minimum links and that has enough available bandwidth. Thus, the network resources are spared.

Description of Q-STA. In Q-STA, the trees of the multicast tree set T are sorted by cost (*i.e.*, the number of links of the trees) : $T = \{T_1, T_2, T_3...\}$ where a tree t in $T_i \subset T$ has a cost i (*i.e.*, t has i links).

When a new group g with bandwidth requirement $b(g)$ arrives, Q-STA computes a native tree t_g for g with maximum available bandwidth. If t_g cannot satisfy the bandwidth requirement $b(g)$, then g is refused and the algorithm stops.

Otherwise, Q-STA examines the trees of cost between $|g| - 1$ and the cost of t_g in the increasing order of cost: trees of T_i are examined before trees of T_{i+1}. The first tree that can cover g and that has enough available bandwidth considering $b(g)$ is chosen for g. In this way, the tree chosen for g is the one utilizing the least links possible.

If no such tree can be found, no aggregation is made and t_g is added in the set T_{c_g} of the multicast tree set T, where c_g is the cost of t_g. Q-STA is presented on Alg. 1 in more details.

Q-STA on an example. On Fig. 2, a domain with 9 routers is represented and a new group g arrives (with members in A, E, G). A tree t' of cost 4 is in the set $T_4 \subset T$ and a tree t of cost 5 is in the set $T_5 \subset T$. A native tree t_g with

maximum available bandwidth is computed for g. We suppose that t_g, of cost 5, satisfies the bandwidth requirement of g. Then, the trees of T are evaluated in the increasing order of their costs from $|g|-1=2$ to 5. T_2 and T_3 are empty sets. When examining T_4, Q-STA detects that t' cannot cover g. When examining T_5, Q-STA detects that t can cover g and has enough bandwidth available. t is chosen for an aggregation of g.

Algorithm 1 Q-STA

Require: The network $G=(V,E)$, the multicast tree set $T=\{T_1,T_2,..\}$, a group g with a bandwidth requirement $b(g)$
Ensure: a tree t for g satisfying $b(g)$ or g is refused
  compute a native tree t_g for g with maximum available bandwidth
  if t_g does not satisfy the requirement $b(g)$ **then**
    g is refused
  end if
  $found \leftarrow false$ and $i \leftarrow |g|-1$
  while not($found$) **and** $i \leq c_g$ (c_g is the cost of t_g) **do**
    while not($found$) **and** there is a tree t in T_i not evaluated yet **do**
      if t covers g **and** t has enough available bandwidth **then**
        $found \leftarrow true$ **and** $t_{agg} \leftarrow t$
      else
        t is evaluated
      end if
    end while
    $i \leftarrow i+1$
  end while
  if $found$ **then**
    aggregate g to t_{agg}
  else
    add t_g in $T_{c_g} \subset T$
  end if

3 Metrics

We compare the performances of the algorithms according to five metrics: the number of accepted groups, the percent of accepted groups per bandwidth requirement, the number of maintained trees, the aggregation ratio and the number of evaluated trees.

3.1 Number of accepted groups

As links have limited bandwidth capacities, some groups may be refused. A group g is refused when the tree manager cannot find any tree covering g with enough available bandwidth on the links. The *number of accepted groups* is the

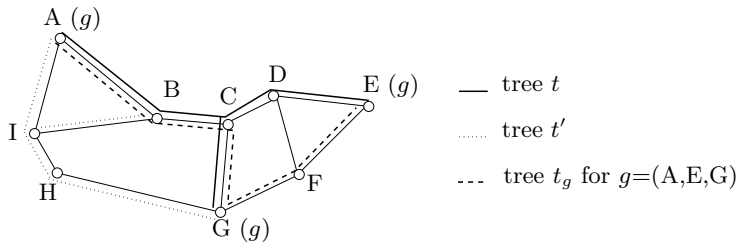

Fig. 2. A new group g can use the tree t.

number of group requests minus the number of refused groups. It is denoted by $|G|$, where G is the set of accepted groups.

3.2 Percent of accepted groups per bandwidth requirement

The number of accepted groups is not sufficient to evaluate the performance of an algorithm since groups may have different bandwidth requirements. Indeed, an algorithm that would systematically reject high-bandwidth groups would accept a large number of low-bandwidth groups.

The *percent of accepted groups per bandwidth requirement* is a new metric that evaluates the fairness of an algorithm. Given a group requirement b, we define the percent of accepted groups for b as:

$$100.\frac{|G(b)|}{|\mathbf{G}(b)|},$$

where $|G(b)|$ is the number of accepted groups having a bandwidth requirement of b and $|\mathbf{G}(b)|$ is the number of group requests having a bandwidth requirement of b. The percent of accepted groups per bandwidth requirement has to be close for all the bandwidth requirements in order to have a fair algorithm.

3.3 Number of maintained trees

The scalability of multicast depends on the number of forwarding states in routers. This number is directly related to the number of maintained trees. In PIM-SM, the number of trees is equal to the number of accepted groups. With tree aggregation, several groups can share the same tree and consequently the number of trees is reduced. The *number of maintained trees* is equal to $|T|$ where T is the multicast tree set of the accepted groups.

3.4 Aggregation ratio

As the number of concurrent groups in the network is not the same for all the algorithms, the number of maintained trees is not sufficient to evaluate the performance of a tree aggregation algorithm. Then, the *aggregation ratio* is defined as $\frac{|T|}{|G|}$, where T is the multicast tree set and G is the set of accepted groups. The lower the tree aggregation ratio, the better the algorithm.

3.5 Number of evaluated trees

To find a good aggregation, tree aggregation algorithms evaluate several trees to determine the right candidates. The more trees are evaluated, the slower the algorithm. Thus, the number of evaluated trees is a good metric to compare running time of the algorithm.

With AQoSM, all the trees of T are evaluated each time a new group g arrives (the number of evaluated trees $ev(g)$ for a group g is equal to $|T|$). With Q-STA, only trees having a cost between $|g| - 1$ and the cost of a native tree for g (having maximum available bandwidth) are evaluated. Moreover, Q-STA aggregates g to the first candidate found while AQoSM determines the right candidate after having evaluated all the trees. We define the *number of evaluated trees* as $\sum_{g \in \mathbf{G}} ev(g)$, where $\mathbf{G}$ is the set of the group requests and $ev(g)$ is the number of evaluated trees for a group g.

4 Simulation analysis

We conducted several simulation experiments on the graph Abilene. Abilene is the backbone of Internet 2 and is constituted of 11 nodes and 14 edges. We choose to assign 1 Gb/s as the capacity dedicated to multicast for each link of Abilene. In this way, we take into account the use of the links by unicast communications. Note that the real link capacity of links of Abilene is 10 Gb/s.

We compared three algorithms: PIM-SM, AQoSM with its bandwidth threshold equal to 0 (in order to minimize the network load) and Q-STA[1]. PIM-SM neither performs tree aggregation nor avoid congested links. In other words, PIM-SM does not take into account the available capacity on links to compute trees. The groups and their bandwidth requirements were the same for the three algorithms. 50 % of the group requests were low-bandwidth (10 Kb/s), 30 % of the group requests were medium-bandwidth (100 Kb/s) and the remaining 20 % of the group requests were high-bandwidth (1 Mb/s). To obtain groups that are realistic, we implemented the node weighted model [11] with 80 % of the nodes having a weight of 0.2 and 20 % of the nodes having a weight of 0.6. In this model, nodes having a large weight have an high probability of being members of groups. We varied the number of groups from 1000 to 10000. Each plot on the figures is the average of 100 simulation scenarios.

4.1 Number of accepted groups

Figure 3 shows the number of groups accepted by PIM-SM, AQoSM and Q-STA. These three algorithms accept the first 5000 groups because the network capacity is sufficient. After the first 5000 groups, PIM-SM starts refusing a large number of groups: it accepts only 800 of the next 5000 groups. Since PIM-SM does not take into account the available bandwidth of links, a group is refused as soon

[1] The program of Q-STA can be found at [10]

as its native tree uses saturated links. AQoSM accepts more groups than PIM-SM. This behavior is expected since AQoSM takes into account the available bandwidth of links: AQoSM builds trees satisfying the bandwidth requirements of the groups. Q-STA has better performances than both PIM-SM and AQoSM: Q-STA accepts 25% more groups than PIM-SM and 20% more groups than AQoSM. Indeed, Q-STA builds trees using links having a low utilization (as Q-STA builds trees with maximum available bandwidth) and then balances the load of the network. Then less links are congested and more groups can be accepted.

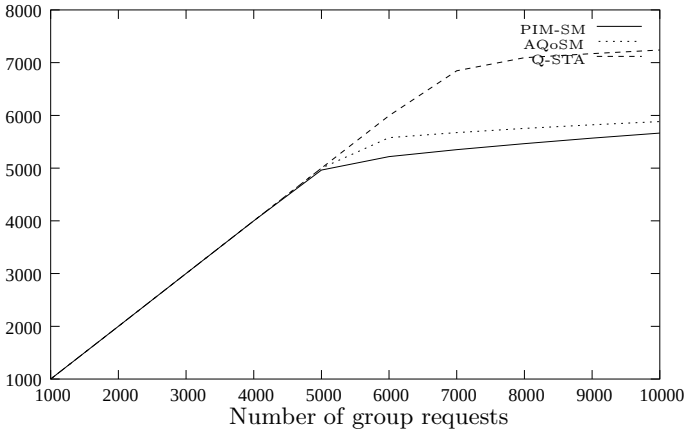

Fig. 3. Number of accepted groups.

4.2 Percent of accepted groups per bandwidth requirement

As said previously, the number of accepted groups is not sufficient to evaluate an algorithm if groups have different requirements. Figure 4 plots the percent of accepted groups considering their requirements: low-bandwidth groups ($b(g) =$ 10 Kb/s), medium-bandwidth groups ($b(g) =$ 100 Kb/s) and high-bandwidth groups ($b(g) =$ 1 Mb/s). As it can be seen, the three algorithms are fair: they accept groups independently of their bandwidth requirements. Unfair algorithms would have refuse high-bandwidth groups in order to accept more low-bandwidth groups[2]. As described in 4.1, Q-STA accepts significantly more groups than PIM-SM and AQoSM independently of their bandwidth requirements.

4.3 Number of maintained trees

To be scalable, a tree aggregation algorithm has to reduce the number of maintained trees. Figure 5 shows the number of maintained trees by the three al-

[2] Note that the fairness of AQoSM was not studied in [8]. We believe, however, that it is an important metric.

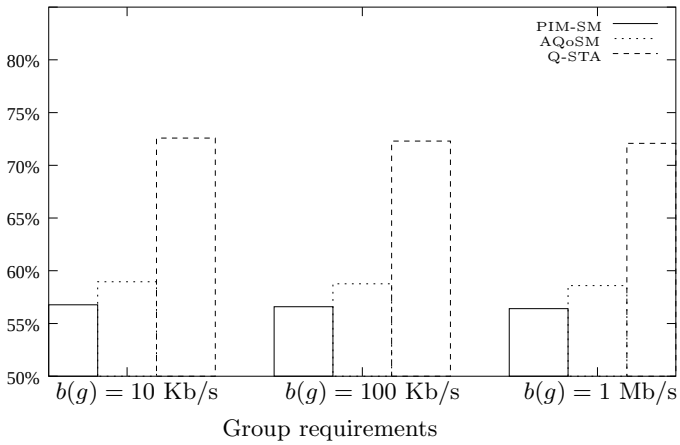

Fig. 4. Percent of accepted groups per bandwidth requirement

gorithms. As expected, the number of maintained trees for PIM-SM is equal to the number of accepted groups by PIM-SM. Both AQoSM and Q-STA algorithms builds less than 300 trees. Thus, AQoSM and Q-STA performs scalable tree aggregation. However, it can be noticed that Q-STA builds around 50 more trees than AQoSM. This behavior allows Q-STA to accept more groups. Q-STA prefers building new trees with few links rather than aggregating groups to trees with too many links, which would congest unnecessarily the links.

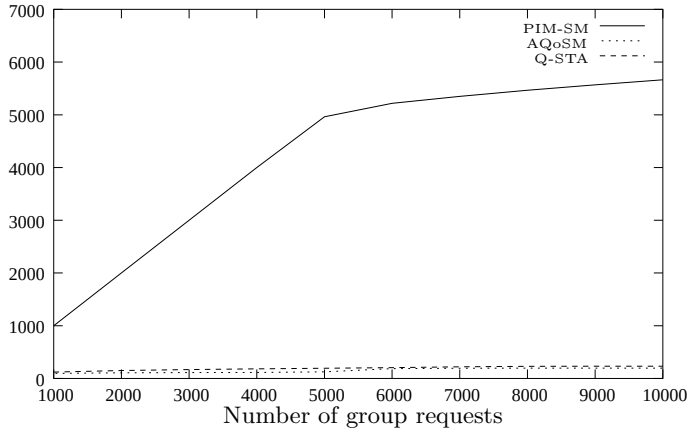

Fig. 5. Number of maintained trees.

4.4 Aggregation ratio

The number of maintained trees depends on the number of accepted groups. Q-STA builds around 50 more trees than AQoSM but handles around 1350 more groups. Therefore, the aggregation ratio is an important metric. Figure 6 displays the aggregation ratio of AQoSM and Q-STA. The aggregation ratio of PIM-SM is not plotted since it is equal to 1. Before 6000 groups, AQoSM has a lower aggregation ratio than Q-STA: AQoSM builds less trees for the same number of groups. Instead of aggregating at all cost as AQoSM does, Q-STA balances the load by building more trees. After 6000 groups, Q-STA is slightly better than AQoSM, because Q-STA accepts more groups than AQoSM without building new trees.

If we consider the number of maintained trees and the aggregation ratio, AQoSM and Q-STA behave in the same way by building significantly less trees than PIM-SM (300 trees instead of 5700 trees).

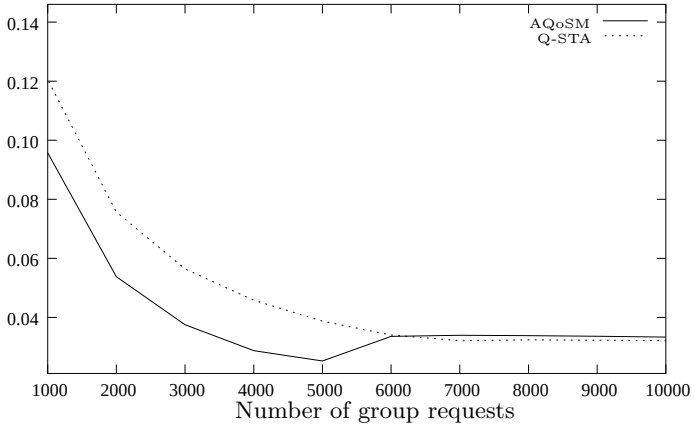

Fig. 6. Aggregation ratio.

4.5 Number of evaluated trees

Our last metric is the number of evaluated trees, which impacts the running time of the algorithm. In AQoSM, all the maintained trees are evaluated each time a new group arrives. In Q-STA, the trees are sorted by their cost and trees of cost i are stored in the set T_i. Instead of evaluating all the sets T_i for a group g, in the worst-case, only the sets from $|g| - 1$ to the cost of the native tree t_g (with maximum available bandwidth) are evaluated. Additionally, Q-STA chooses as a candidate for g the first tree that can cover g: less trees are evaluated. Figure 7 shows the number of evaluated trees by AQoSM and Q-STA. Q-STA evaluates 45% less trees than AQoSM.

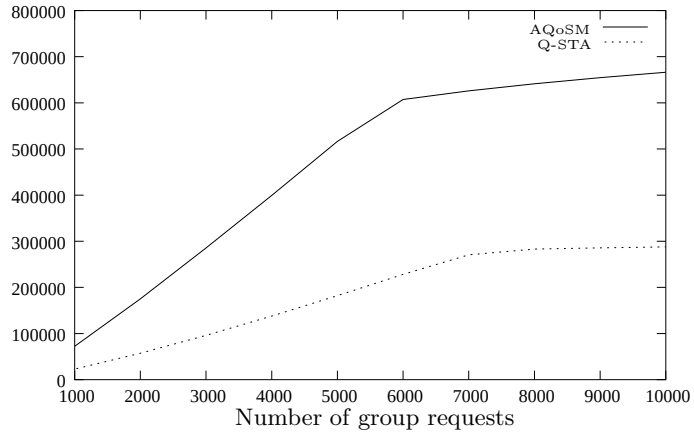

Fig. 7. Number of evaluated trees.

4.6 Summary of the simulation results

With our simulations, we show that the ideas behind Q-STA give good results:

1. By building trees maximizing the available bandwidth, Q-STA balances the load of the network and consequently accepts more groups than AQoSM (see Fig. 3 and Fig. 4).
2. Q-STA builds slightly more trees than AQoSM but much more groups are accepted (see Fig. 5 and Fig. 6). However, Q-STA and AQoSM behave in the same way considering the number of maintained trees and the aggregation ratio.
3. By evaluating only trees that are likely to be candidates (*i.e.*, by examining trees of cost close to the cost of the native tree for a group), Q-STA evaluates less trees and then runs faster than other algorithms (see Fig. 7). The trees that are not evaluated for a group g cannot be candidates: no tree with cost below the lower bound given by Q-STA can cover g and trees with cost above the higher bound given by Q-STA waste bandwidth unnecessarily.

Finally, Q-STA is a scalable algorithm that performs fast aggregations and accepts significantly more groups than the existing algorithm AQoSM. Moreover, Q-STA performs fair aggregations by accepting groups independently of their bandwidth requirements.

5 Conclusion

In this paper, we proposed a new algorithm Q-STA that deals with tree aggregation with bandwidth constraints. Tree aggregation reduces the number of forwarding states in routers by having several groups sharing the same tree within a domain. We compare the performances of our algorithm with the previous algorithm AQoSM by extensive simulations. Q-STA performs faster aggregations

than AQoSM by evaluating less trees. Q-STA builds as few trees as AQoSM and Q-STA outperforms AQoSM by accepting more groups independently of their bandwidth requirements. Indeed, Q-STA builds trees using few links and uses links having a low utilization. This behavior achieves load balancing. Moreover, we show that tree aggregation does not induce an overhead of the network load compared to PIM-SM. Finally, Q-STA is an efficient algorithm for tree aggregation with bandwidth constraints.

References

1. Almeroth, K.: The Evolution of Multicast: From the MBone to Inter-Domain Multicast to Internet2 Deployment. IEEE Network **14** (2000) 10–20
2. Wong, T., Katz, R.: An Analysis of Multicast Forwarding State Scalability. In: IEEE International Conference on Network Protocols (ICNP). (2000)
3. Deering, S.E.: Multicast Routing in a Datagram Internetwork. PhD thesis, Stanford University (1991)
4. Cui, J.H., Kim, J., Maggiorini, D., Boussetta, K., Gerla, M.: Aggregated Multicast — A Comparative Study. In: IFIP Networking. (2002)
5. Cui, J.H., Kim, J., Maggiorini, D., Boussetta, K., Gerla, M.: Aggregated Multicast — A Comparative Study. Special issue of Cluster Computing: The Journal of Networks, Software and Applications (2003)
6. Cui, J.H., Lao, L., Maggiorini, D., Gerla, M.: BEAM: A Distributed Aggregated Multicast Protocol Using Bi-directional Trees. In: IEEE International Conference on Communications (ICC). (2003)
7. Liu, Z.F., Dou, W.H., Liu, Y.J.: AMBTS: A Scheme of Aggregated Multicast Based on Tree Splitting. In: IFIP Networking. (2004) 829–840
8. Cui, J.H., Kim, J., Fei, A., Faloutsos, M., Gerla, M.: Scalable QoS Multicast Provisioning in Diff-Serv-Supported MPLS Networks. In: IEEE Globecom. (2002)
9. Guitton, A., Moulierac, J.: Scalable Tree Aggregation with a Large Number of Multicast Groups. In: Submitted to IEEE International Conference on Communications (ICC). (2004)
10. Q-STA: (2004) http://www.irisa.fr/armor/lesmembres/Guitton/recherche/qsta/qsta-en.html.
11. Fei, A., Cui, J.H., Gerla, M., Faloutsos, M.: Aggregated Multicast with Inter-Group Tree Sharing. In: 3rd International Workshop on Networked Group Communications (NGC). (2001)