Multicast Support in DiffServ Using Mobile Agents

Mohamed El Hachimi, Abdelhafid Abouaissa, and Pascal Lorenz

Many multicast applications, such as video-on-demand and video conferencing, desire quality of service (QoS) support from an underlying network. The differentiated services (DiffServ) approach will bring benefits for theses applications. However, difficulties arise while integrating native IP multicasting with DiffServ, such as multicast group states in the core routers and a heterogeneous QoS requirement within the same multicast group. In addition, a missing per-flow reservation in DiffServ and a dynamic join/leave in the group introduce heavier and uncontrollable traffic in a network. In this paper, we propose a distributed and stateless admission control in the edge routers. We also use a mobile agents-based approach for dynamic resource availability checking. In this approach, mobile agents act in a parallel and distributed fashion and cooperate with each other in order to construct the multicast tree satisfying the QoS requirements.

Keywords: Mobile agents, multicast, DiffServ, QoS.

I. Introduction

The phenomenal growth of real-time multicast applications requires scalable and efficient network support. The differentiated services (DiffServ) architecture was developed to provide quality of service (QoS) and achieve scalability by avoiding complexity in the core routers. The simplicity of the core routers also causes fundamental problems in conjunction with the provision of an IP multicast in a DiffServ domain. It appears to be a difficult task to support a multicast in a DiffServ domain while eliminating the multicast tree state in the core routers. Some solutions have been proposed in [1] and [2] to reduce the multicast tree state in the core routers.

Another issue is how to support heterogeneous QoS requirements within the same group. This is due to the fact that different multicast receivers in one multicast group may require a different level of QoS. Some proposals have been suggested in [1] and [3] to counter this problem.

The difficulty to reserve previously suitable resources for a multicast group arises from the fact that in a multicast, receivers can dynamically join and leave a multicast group at anytime. Consequently, no previous reservation can be done for the multicast group in a DiffServ domain. When a new receiver joins an IP multicast group, the corresponding multicast routing protocol (e.g., distance vector multicast routing protocol, protocol independent multicast—dense mode, or protocol independent multicast—sparse mode) creates a situation where a new sub-tree, which connects the new receiver to the already existing multicast tree, expands the multicast tree. As a result of tree expansion and a missing per-flow traffic profile in the core routers, the new receiver will implicitly affect the currently provided QoS level of the other receivers (with correct reservations) if the additional amount of resources consumed by the new part of the multicast tree are not taken into account. This is called the neglected reservation sub-tree problem (NRS) [3].

Manuscript received Apr. 13, 2004; revised Dec. 18, 2004.

Mohamed El Hachimi (email: m.elhachimi@uha.fr), Abdelhafid Abouaissa (email: a.abouaissa@ uha.fr), and Pascal Lorenz (phone: + 33 03 89 20 23 66, email: lorenz@ieee.org) are with IUT, University of Haute Alsace, France.

In this paper, we propose a solution for the NRS problem. The advantage of the proposal presented consists of using a distributed stateless admission control in the edge routers in order to check resource availability before a join request is accepted, instead of using the centralized control entities [2], [4]. In addition, since DiffServ core routers can not support any explicit signaling, a mobile-agents-based approach is used here.

Mobile software agents provide a new and useful paradigm for distributed computing. A mobile agent is defined as a small program that can move between two nodes. It works autonomously toward a goal and interacts with other agents and its environment. The use of mobile agents allows the edge routers to take stateless admission control for multicast traffic in a distributed and dynamic fashion. The expansion of the existing multicast tree consists then of checking resource availability before a join request is accepted and consequently finding a feasible best QoS-compliant path. This is decided by a cooperative and parallel task of the mobile agents launched by the edge router upon receiving a join request.

The rest of this paper is organized as follows. Section II describes related work and motivations. In section III we present a set of simulations in order to prove the NRS problem. In sections IV, V, and VI, we discuss our solution in detail and present the proposed algorithm. In section VII, we present a performance study and simulation results. Finally, our conclusions and future work are presented.

II. Related Work

Recently, there have been several proposals for a multicast in DiffServ. In this section, we present some background on multicasting in a DiffServ domain. We classify and review briefly a number of proposals and highlight the needs that motivate our work.

The approaches proposed in [2] and [4] consist of using a centralized management entity (tree manager or multicast broker). Moreover, this entity must have detailed knowledge of the current multicast tree topologies and resources available in order to make admission control decisions and determine the involved branching points. In addition, it must treat the join and leave messages for all active groups in the domain.

DSMCast (DiffServ MultiCast) [1] uses an encapsulation-based approach. This approach does not require any state information in the core routers. Rather, it introduces an additional header because it embeds the multicast information within the multicast packet itself. The other weakness is that the edge router has to keep detailed knowledge of the current multicast tree topology and corresponding DSCP (differentiated service code point) for each link.

In [5], a probe-based approach is used to convey at the edge routers the information that the network is congested and that a new flow cannot be accepted. This proposal is based on an extension of protocol independent multicast–sparse mode, in which the routing topology is known from unicast routing. Consequently, multicast tree construction doesn't take into account that resources in other paths may be released and are therefore available while resources in the domain change dynamically.

III. Neglected Reservation Sub-tree Problem

In order to clarify the existence of the NRS problem, we provide a set of simulations made using the NS2 simulator. The NRS problem could occur in two different cases: when the branching point to extend the multicast tree is a core router or when it is an edge router.

1. The Branching Point is a Core Router

In this case, since the core router is usually not equipped with metering or policing functions it will not recognize any excess amount of traffic and will forward the new multicast flow. If the latter belongs to a higher priority service, such as Expedited Forwarding (EF) in section III.1.A and Assured Forwarding (AF) in section III.1.B, the result is that the bandwidth of the aggregate is higher than the aggregate's reservation and it will steal bandwidth from lower priority services. The additional amount of EF/AF without a corresponding reservation is forwarded together with the aggregate that has a reservation. Figure 1 describes the topology used to prove this result when the branching point is a core router in each of two cases: when the additional multicast flow is EF and when it is AF.

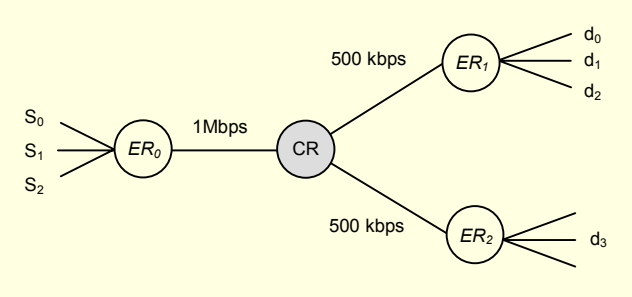

Fig. 1. The branching point is a core router (CR).

A. Additional EF Multicast Flow

The sender S_0 generates two shaped EF flows of 200 kbps each (packets of 64 bytes, constant in size) and sends them to multicast groups G_1 and G_2. Senders S_1 and S_2 send an AF traffic flow of 200 kbps and a best effort (BE) traffic flow of 100 kbps toward d_1 and d_2, respectively. One static profile is installed in the ingress edge router ER_0 with a 2×200 kbps EF

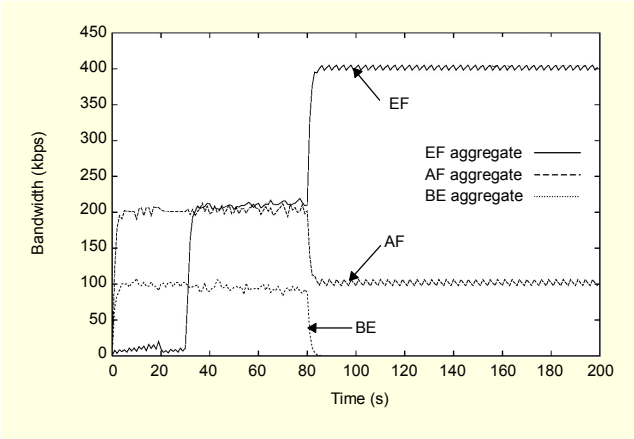

Fig. 2. Bandwidth sharing between CR and ER_1.

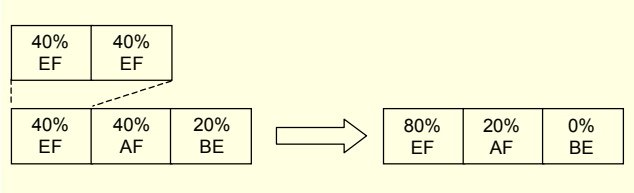

Fig. 3. Resulting share of bandwidth.

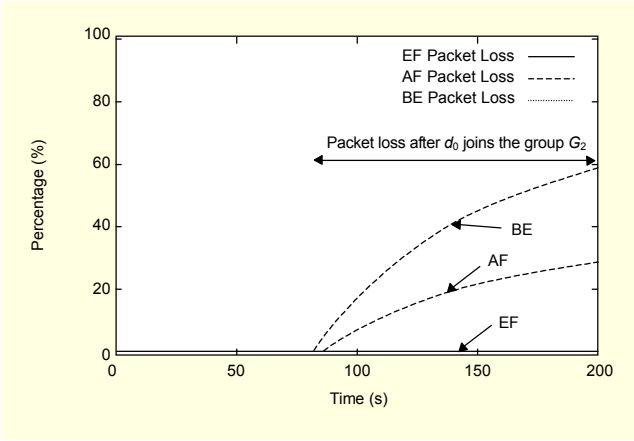

Fig. 4. Packet loss between CR and ER_1.

aggregate.

At 30 seconds, host d_3 joins G_2. At 60 s, host d_0 joins G_1. Those joins are made using a reservation for the group toward the sender. At 80 s, host d_0 joins G_2 as shown in Fig. 2. The last join creates an additional flow. This results in no packet losses for EF as long as the resulting aggregate is not higher than the output link bandwidth. Because of its higher priority, EF gets as much bandwidth as needed. As a result, there is no restriction for EF, but other services will be extremely disadvantaged by this use of non-reserved resources. Their bandwidth is stolen by the new additional flow as is shown in Figs. 2 and 3. In this case, the additional 40% EF traffic caused by the last join preempts resources from the AF traffic, which in turn preempts resources from the BE traffic, shown in Fig. 2, resulting in 20% packet losses for the AF aggregate and a complete loss for the BE traffic as shown in Figs. 2 through 4.

In all simulations, the percentage of packet loss is defined as (the total packet loss of the aggregate / total packets of the aggregate transmitted) × 100. Figure 4 shows that the packet losses of both BE and AF appear at 80 s, corresponding to the time when d_0 joins G_2.

B. Additional AF Multicast Flow

The sender S_0 generates two shaped AF flows of 200 kbps each (packets of 1500 bytes, constant in size) and sends them to multicast groups G_1 and G_2. Senders S_1 and S_2 generate an EF traffic flow of 200 kbps and a BE traffic flow of 100 kbps toward d_1 and d_2, respectively. One static profile is installed in the ingress edge router ER_0 with a 2 × 200 kbps AF aggregate.

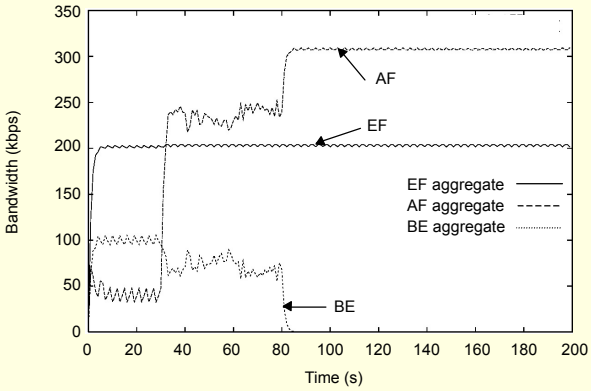

Fig. 5. Bandwidth sharing between CR and ER_1.

At 30 s, host d_3 joins group G_2. At 60 s, host d_0 joins group G_1. These joins are made using a reservation for the group toward the sender. At 80 s, host d_0 joins group G_2 as shown in Fig. 5.

Figures 5 and 6 show that other services with even lower priority can be reduced in their quality (in this case the best-effort traffic is discarded completely). In addition, other traffic such as AF can also be disadvantaged. As shown in the example, the services aggregate causing the problem (AF) can itself be affected by packet loss (20% of the AF aggregate is discarded).

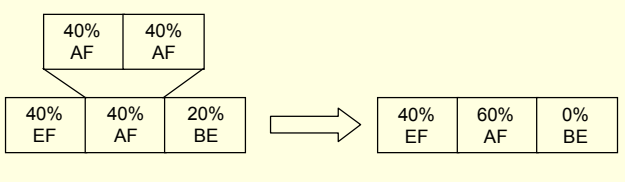

Fig. 6. Resulting share of bandwidth.

In Fig. 7, a small packet appears at 50 s, which corresponds to the time when d_0 joins G_1. This is due to multicast traffic flooding G_2. At 80 s, AF packet loss appears and BE packet loss increases.

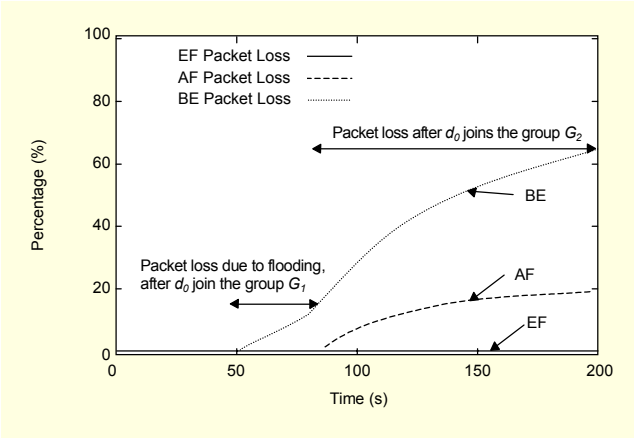

Fig. 7. Packet loss between CR and ER_1.

2. The Branching Point is an Edge Router

In contrast with the core routers, the policing component in the egress router can recognize any excess amount of traffic and then discard packets until the traffic aggregate conforms to the traffic contract. But while discarding packets, the router cannot identify the responsible flow (because of a missing flow classification functionality at this level), and therefore it randomly discards packets whether they belong to a correctly reserved flow or not. As a result, there will no longer be any service guarantee for the reserved flows. Figure 4 describes the topology used to prove this result when the branching point is an edge router in each of two cases: where the additional multicast flow is EF and when it is AF.

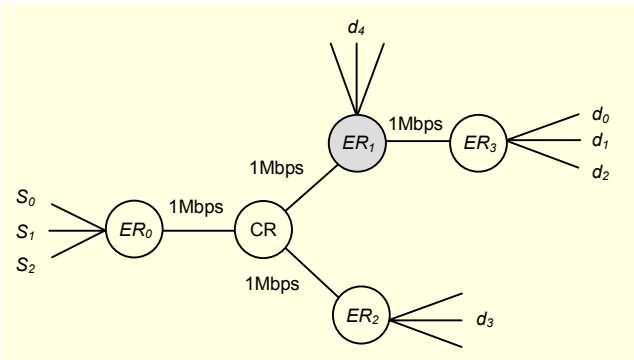

Fig. 8. The branching point is an edge router (ER_1).

A. Additional EF Multicast Flow

The sender S_0 generates two shaped EF flows of 200 kbps each (packets of 64 bytes, constant in size) and sends them to multicast groups G_1 and G_2. Senders S_1 and S_2 send an AF traffic flow of 200 kbps and a BE traffic flow of 100 kbps toward d_1 and d_2, respectively. In the egress edge router ER_1, one profile has been installed for the output link to ER_3, permitting a 200 kbps EF aggregate.

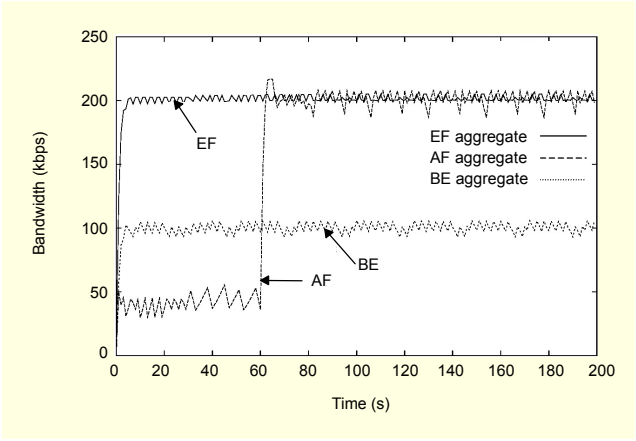

Fig. 9. Bandwidth sharing between ER_1 and ER_3.

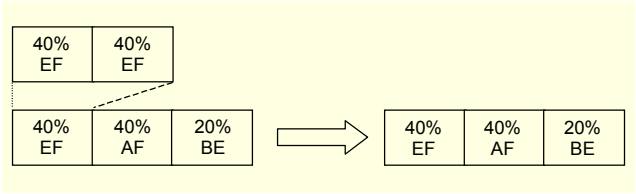

Fig. 10. Resulting share of bandwidth.

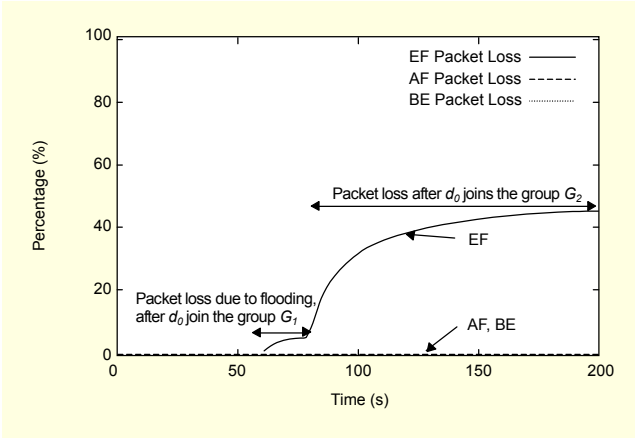

Fig. 11. Packet loss between ER_1 and ER_3.

At 30 s, host d_4 joins G_2. At 60 s, host d_0 joins G_1. Those joins are made using a reservation for the group toward the sender. At 80 s, host d_0 joins G_2. Figures 9 and 10 show the resulting share of bandwidth in the case where EF is used for the flow causing the NRS problem, assuming that the

additional traffic would use another 40% of link bandwidth. Figures 9 and 10 illustrate that the resulting EF aggregate (80% of the outgoing link bandwidth) is throttled down to its originally reserved 40%. In this case, the amount of dropped EF bandwidth is equal to the amount of excess bandwidth. From Figs. 10 and 11, it is clear that the complete EF aggregate is affected by packet loss. The other services, e.g., AF or BE, are not disadvantaged.

In Fig. 11, a small packet loss appears at 60 s when d_0 joins G_1. This is due to multicast traffic flooding G_2. At 80 s, EF packet loss continues increasing.

B. Additional AF Multicast Flow

The sender S_0 generates two shaped AF flows of 200 kbps each (packets of 1500 bytes, constant in size) and sends them to multicast groups G_1 and G_2. Senders S_1 and S_2 send an EF traffic flow of 200 kbps and a BE traffic flow of 100 kbps toward d_1 and d_2, respectively. In the egress edge router ER_1, one profile has been installed for the output link to ER_3, permitting up to 200 kbps AF. One static profile is installed in the ingress edge router ER_0 with a 2×200 kbps AF aggregate.

At 30 s, host d_4 joins G_2. At 60 s, host d_0 joins G_1. Those joins are made using a reservation for the group toward the sender. At 80 s, host d_0 joins G_2 as shown in Fig. 12. Only the join of host d_0 to G_2 has no admitted reservation.

Figures 12 and 13 show the resulting share of bandwidth in the case when AF is used for the flow, causing the NRS problem (assuming that the additional traffic would use another 40% of link bandwidth). Figures 12 and 13 illustrate that the resulting AF aggregate (80% of the outgoing link bandwidth) is throttled down to its originally reserved 40%. Figures 13 and 14 show that AF is now affected by discards and that the remaining services will get their guarantees. In either case, packet losses are restricted to the misbehaving service class by the traffic meter and policing mechanisms in the edge routers.

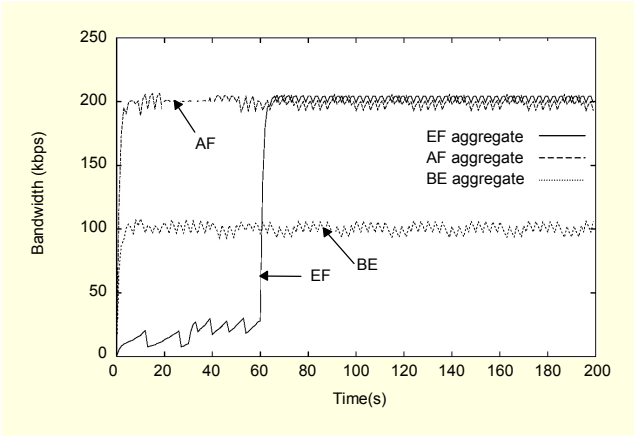

Fig. 12. Bandwidth sharing between ER_1 and ER_3.

Moreover, the latter problem occurs only in egress edge routers because they are normally responsible to make sure that no more traffic leaves the DS domain than the following ingress edge router will accept. Therefore, those violations of service level agreements will be already detected and processed in the egress edge routers. In Fig. 14, a small packet loss appears at 60 s when d_0 joins G_1. This is due to multicast traffic flooding G_2. At 80 s, AF packet loss continues increasing.

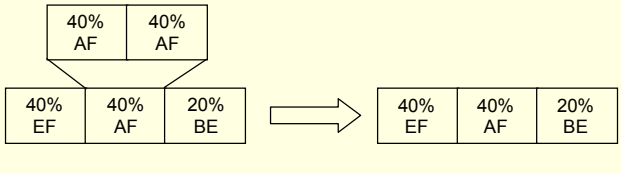

Fig. 13. Resulting share of bandwidth.

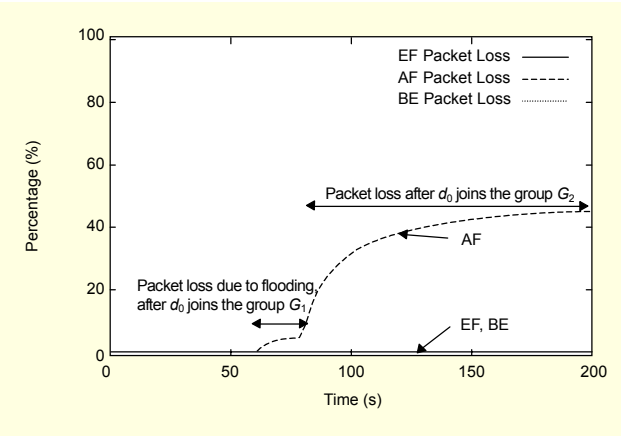

Fig. 14. Packet loss between ER_1 and ER_3.

IV. Description of the Proposed Solution

Since no previous reservation can be done for the multicast group (dynamic join and leave), the proposed approach consists of using a distributed stateless admission control in the edge routers. It is distributed because this increases dependability of the network by avoiding the point of failure of the centralized control entities [2], [4] and stateless because no per-flow reservation state can be employed in DiffServ. In addition, since DiffServ core routers cannot support any explicit signaling, we use a mobile-agent-based approach to decide whether a new member can or cannot be accepted in the domain depending on resource availability. Then, the multicast tree is expanded by a new sub-tree, which connects the new receiver. The best QoS path finding algorithm used in this article is based on a colony of mobile agents deployed for the actual discovery of QoS-compliant routes. The other benefits of this scheme are that the multicast tree construction doesn't depend on the unicast routing

protocol since the mobile agent carries the list of visited nodes and current link states. This achieves scalability by significantly reducing the communication overhead of constructing a multicast tree, providing a good resource management mechanism by selecting the best QoS-compliant path to connect the new member to the existing multicast tree. Therefore, it doesn't require global link-state information or global topology knowledge.

V. QoS Path Finding Using Mobile Agents

The process of adding a new branch to the multicast tree starts when an edge router receives a request to join a multicast group. If the edge router is part of the group already, the connection is established locally. If the edge router is not part of the group, a feasible path searching algorithm is employed to connect the new member to the available tree.

The feasible QoS path finding algorithm proposed here consists of using a coordinated colony of mobile agents launched from the edge router. The proposed algorithm is inspired from the algorithm proposed in [6], where a similar mobile-agent-based approach is used to establish a multipoint-to-point tree in order to decide on the merging point in MPLS. The algorithm proposed here is used to establish a point-to-multipoint tree (multicast tree).

Rule 1. An agent clones itself only on links satisfying the bandwidth required [6], [7].

Rule 2. An agent is allowed to continue its travel if the distance carried is less than or equal to the one previously recorded (this is applied for agents coming from the same home node and looking for the on-tree nodes in the same multicast tree).

The agent is allowed to travel only on links that meet the QoS constraints required. First, this allows finding only QoS-compliant routes and second, it reduces the number of mobile agents. When an agent reaches a node, the agent clones itself into the same number of copies as the number of QoS outgoing links of the node. During its travel, an agent carries the list of visited nodes and the local variable (hops). This variable is used as the cost and is incremented, arriving in an intermediate node. The cost value is used by each node to allow any agent having traveled a more minimum partial distance than the one previously recorded by a different agent coming from the same home node and looking for the same multicast tree to continue its travel toward the destination. An agent carried a larger distance is discarded at any intermediate node as soon as this condition is detected. The distance here is the number of hops, but it could be any other function. This process will reduce the overhead created by mobile agents.

We deduce two important results in this process: First, all possible QoS paths are founded. Second, the routes found are cycle free since if an agent returns to an earlier visited node, it will carry a cost larger than the previous cost recorded during its first visit to the same node. Therefore, it will be discarded (state = dies). The mobile agent stops cloning itself, even reaching the source of the multicast tree or an intermediate on the tree node. In this case, the agent doesn't clone itself. Rather, it changes its state (state = backtracking) and travels back to its home node following back the founded path.

Finally, the home node has the list of all the shortest paths from itself to the multicast tree that comply with QoS constraint requirements. Then, a selection process allows for finding the best QoS-compliant route using the cost carried by the agents. This procedure allows for optimizing the cost of the multicast tree. Then, the agent carrying the best cost (hops) changes its state (state = constructing) and establishes the path.

1. Algorithm

```
// Initialization
Agent.visit_list = NULL
Agent.home_node = home
Agent.mcast_group = group
Agent.required_QoS = QoS
Agent.hops = 0
Agent_state = Searching
//Terminate if the source or an on-tree node reached
while (cur_node = (mcast_src or on-tree node)) do
   Agent.visit_list.push(cur_node)
   Agent.hops+ = 1    //increment the distance traveled
   For each (i ∈ neighbors of cur_node)
      Agent.clone() through QoS-compliant links
   If Agent reach cur_node first time
      cur_node.push(Agent)
   Else If Agent.hops <= cur_node[Agent].hops
      If Agent.hops < cur_node[Agent].hops
         cur_node[Agent].hops = Agent.hops
      continue navigation
   Else Agent.state = dies
done
Agent.state = Backtracking
while (Agent.visit_list≠NULL) do
   //remove last node from visit_list and move to it.
   dest = Agent.visit_list.pop()
   Agent.move(dest)
done
Agent.state = Constructing
while (Agent.visit_list≠NULL) do
   Createmulticastentry(S,G)incurrent_node
   //remove first node from visit_list and move to it.
   dest = Agent.visit_list.pop()
   Agent.move(dest)
done
```

VI. Connection Establishment

We will examine how the new path is established after the searching phase has been completed.

1. Since all paths previously discovered are QoS compliant, the edge router selects the best candidate according to the cost collected by the set of backward agents.
2. The edge router sends a constructing agent to the selected candidate. The constructing agent traverses the path in the opposite direction and establishes a routing state along the selected path.
3. When the selected candidate (source or on-tree router) receives the constructing agent, it starts transmitting data packets on the newly set-up path towards the edge router.

VII. Leaving a Group

A leave request is sent through the same protocol that communicated the join request. Whenever a router receives a leave request from a host, it removes the link from the distribution tree and checks whether it has become a leaf of the related tree. If so, it sends an agent "prune" up the tree and removes the state for the tree from its database, thereby ceasing to be an on-tree router for that tree.

VIII. Simulation Results

1. Random Graph Generation

To ensure that the simulations of the effects of the different routing algorithms are fairly evaluated, random graphs with low average degrees are constructed. The nodes are randomly connected with the probability function:

$$P(u,v) = \lambda \exp(\frac{-d(u,v)}{\rho L}), \quad (1)$$

where $d(u,v)$ is the distance between nodes u and v, and L is the maximum possible distance between any pair of nodes. The parameters λ and ρ with a range of (0,1] can be modified to create the desired network model. For example, a large value for λ gives nodes with a high average degree, while a small value for ρ increases the density of shorter links relative to longer ones. In our simulation, λ and ρ are set to 0.25 and 0.2, respectively, to simulate a large network such as the Internet. We use the bandwidth of the link between nodes u and v as the cost of the edge. The bandwidth capacity of each edge is randomly generated in [0,10] Mb.

2. Performance Analysis

The algorithm previously described in section V.1 has been implemented using an NS2 simulator. Topologies with 100 nodes and different maximum branching degrees were used in the simulations. The state of each directed link is randomly generated. A dynamic scheme was created and implemented to simulate a number of random receivers requesting to either join or leave an existing multicast tree. Each point in Figs. 15 and 16 is the result of 500 simulations. In each simulation run, a source and a set of 10 multicast receivers were randomly generated. The receiver's QoS requirement is fixed for each link's success ratio (what percent of links meet the new receiver QoS requirements). For each data point plotted, we run the simulation 100 times. We have mainly focused on the success ratio per join as a measure of performance, which is defined as follows:

$$Success\ ratio = \frac{Number\ of\ \ join\ requests\ accepted}{Total\ number\ of\ \ join\ requests} \quad (2)$$

Figure 15 compares the success ratio per join of the algorithm described in section V.1 using topologies with different maximum branching degrees (MBDs).

We provide a case study on how the MBD affects performance. As shown in Fig. 15, a larger MBD results in a better success ratio. This is due to the fact that with a high MBD we have a greater chance to find a feasible path. Consequently, this introduces a larger overhead as shown in Fig. 16. Then, a good tradeoff between the success ratio and the agent's overhead would be required and the concept of MBD could be used in the algorithm shown in section V.1 in order to optimize the overhead.

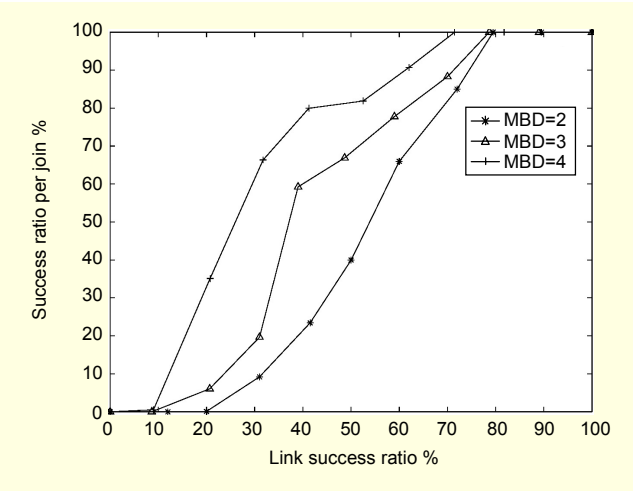

Fig. 15. Success ratio per join using topologies with different maximum branching degrees.

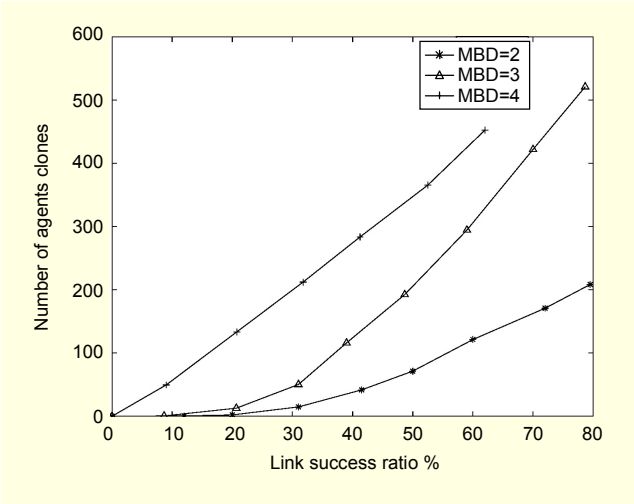

Fig. 16. Mobile agent's overhead.

IX. Conclusion

In this paper, we present a set of simulations in order to prove the neglected reservation sub-tree problem. To solve this problem, we propose a distributed stateless admission control to avoid the point of failure of centralized control entities [2], [4]. In addition, since no explicit signaling can be used in DiffServ, a mobile agents approach is proposed in this paper. Mobile agents work in parallel and cooperate in order to find a feasible path from the new member to the multicast tree. The algorithm doesn't depend on the unicast routing protocol and requires no intermediate routers to exchange a link's state information.

Future work will focus on how to reduce the number of agents in the network and the bandwidth consumed by them. More efficient techniques have to be investigated and tested. Complex methods might imply an increased computational complexity of the agent algorithms. A deeper investigation and trade-off analysis would be required to establish an adequate equilibrium between these important factors.

References

[1] A. Striegel and G. Maniaran, "A Scalable Protocol for Member Join/Leave in DiffServ Multicast," *Proc. IEEE Local Computer Networks (LCN)*, Florida, USA, Nov. 2001, pp.395-404.

[2] A. Striegel, A. Bouabdallah, H. Bettahar2, and G. Manimaran, "EBM: A New Approach for Scalable DiffServ Multicasting," *Proc. NGC*, Sept. 2003, pp.131-142.

[3] R. Bless and K. Wehrle, "Group Communication in Differentiated Services Networks," *IQ Workshop at CCGRID 2001*, May 2001, pp. 618-625.

[4] J-H. Cui, J. Kim, A. Fei, M. Faloutsos, and M. Gerla, "Scalable QoS Multicast Provisioning in Diff-Serv-Supported MPLS Networks," *Proc. of IEEE Globecom2002*, vol. 21, no. 1, Taiwan, Nov. 2002, pp.1459-1464.

[5] G. Bianchi, N. Belfari-Mellazi, G. Bonafede, and E. Tintinelli, "QUASIMODO: Quality of Service-Aware Multicasting over DiffServ and Overlay Networks," *IEEE Network*, vol. 17, no. 1, Jan./Feb. 2002, pp.38-45.

[6] S. Gonzalez and V. Leung, "QoS Routing for MPLS Networks Employing Mobile Agents," *IEEE Network*, May/June 2002, pp.16-21.

[7] K. Oida and M. Sekido, "ARS: An Efficient Agent-Based Routing System for QoS Guarantees," *Comp. Commun.*, vol. 23, 2000, pp. 1437-1447.

[8] M. Faloutsos, A. Banerjea, and R. Pankaj, "QoSMIC: Quality of Service Sensitive Multicast Internet Protocol," *Proc. SIGCOMM'98*, Sept. 1998, pp.144-153.

[9] Shigang Chen, Klara Nahrstedt, and Yuval Shavitt, "A QoS-Aware Multicast Routing Protocol," *IEEE J. on Selected Areas in Comm.*, vol. 18, issue. 12, Dec. 2000, pp. 2580-2592.

[10] Z. Li and P. Mohapatra, "QoS-Aware Multicast Protocol Using Bounded Flooding (QMBF) Technique," *ICC*, vol. 25, no. 1, Apr. 2002, pp.1259-1263.

[11] G. Manimaran and A. Striegel, "A Survey of QoS Multicasting Issue," *IEEE Network*, June 2002, pp.82-87.

[12] J. Hou and B. Wang, "Multicast Routing and its QoS Extension: Problems, Algorithms, and Protocols," *IEEE Network*, vol. 14, no. 1, Jan./Feb. 2000, pp.22-36.

[13] G. Di Caro and M. Dorigo, "Mobile Agents for Adaptive Routing," *Proc. of the 31st Int'l Conf. on System Sciences (HICSS-31)*, IEEE Computer Society Press, vol. 7, 1998, pp. 74-83.

[14] Network Simulator, http://www.isi.edu/nsnam/ns.

Mohamed El Hachimi received the PhD degree in computer science from Franche-Comté University, France in 2004. His main research interest is in QoS, DiffServ, MPLS, and multicasting.

Abdelhafid Abouaissa is an Associate Professor at the University of Haut Alsace, in Colmar France. He received the BS degree from Technical University of Wroclaw, Poland, in 1995, and the MS degree from Franche-Comte University of Besanon, France, in 1996. He obtained the PhD at Technical University of Belfort, France in January 2000. His interests include multimedia synchronization, group communication systems, Ad-Hoc, MPLS, DiffServ, and QoS management.

Pascal Lorenz received the PhD degree from the University of Nancy, France. Between 1990 and 1995 he was a Research Engineer at WorldFIP Europe and at Alcatel-Alsthom. He is a Professor at the University of Haute-Alsace and is responsible for the Network and Telecommunication Research Group. His research interests include QoS, wireless networks, and high-speed networks. He was the Program and Organizing Chair of the IEEE ICATM'98, ICATM'99, ECUMN'00, ICN'01, ECUMN'02 and ICT'03, ICN'04 conferences and Co-Program Chair of ICC'04. Since 2000, he has been a Technical Editor of the IEEE Communications Magazine Editorial Board. He is the Secretary of the IEEE ComSoc Communications Systems Integration and Modelling Technical Committee. He is a Senior Member of IEEE, a member of many international program committees and has served as a guest editor for a number of journals including Telecommunications Systems, IEEE Communications Magazine and LNCS. He has organized and chaired several technical sessions and given tutorials at major international conferences. He is the author of 3 books and 135 international publications in journals and at conferences.