Quality-of-Service Multicast Overlay Spanning Tree Algorithms for Wireless Ad Hoc Networks

Georgios Rodolakis[1], Cédric Adjih[1], Anis Laouiti[2], and Saadi Boudjit[3]

[1] INRIA Rocquencourt, France
Georges.Rodolakis@inria.fr, Cedric.Adjih@inria.fr
[2] GET/INT Evry, France
Anis.Laouiti@int-evry.fr
[3] Université Paris 13, France
saadi.boudjit@l2ti.univ-paris13.fr

Abstract. In this article, we explore modified versions of Multicast Overlay Spanning Tree algorithms (MOST) to support quality of service for wireless ad hoc networks. These algorithms ($\mathcal{Q}$-MOST) take into account the interferences due to radio transmissions and the residual capacity of the nodes in the network. Different algorithms are compared to the basic MOST algorithm. We show by simulations the ability and superiority of these algorithms to find spanning trees that connect all multicast group members with respect to the bandwidth requirements.

1 Introduction

Mobile ad hoc networks (MANETs) are one of the key research topics of the moment. A MANET can be defined as a set of mobile nodes that communicate using the wireless medium, and does not require any pre-existent infrastructure. The main challenge of MANET research is to offer connectivity between the participating nodes in a multi-hop manner by sharing the same wireless channel. Several unicast protocols have been proposed to address this issue.

However, the wireless medium has limited capacity and is a scarce resource, because it is a shared among nodes within an area. One possibility is the use of multicast, which fits the increase of the popularity of group communication (P2P, Web 2.0, ...), where several nodes exchange the same data among themselves. With multicast, every transmission from the source(s) will ultimately reach all the multicast group members, and this is achieved by using dedicated multicast structures and protocols that connect a set of nodes. This enables gains by decreasing the number of necessary transmissions. Indeed, for massively dense ad hoc networks and in the same setting as the result of Gupta and Kumar [1], it has been shown that multicast can offer a gain of $\Theta(\sqrt{n})$ over direct unicast to each of n destinations [2].

But decreasing the number of transmissions may not be sufficient since some multicast multimedia applications may also require quality of service guarantees in order to deliver real time data flows within the network. In particular, gains from multicast will delay the apparition of congestion, but might not prevent it in overloaded networks, resulting in packet loss and delays, that will have a negative impact on QoS applications. Avoiding congestion in a wireless ad hoc network is not simple, because of the issues of shared medium and interference. The question is then how to perform multicast in a way that avoids congestion, taking interferences into account.

In this article, we propose some answers to this question: we introduce multicast algorithms that are, first, based on admission control, so that they are able to detect and reject multicast groups that would create congestion, and, second, that can find multicast structures which avoid congested areas, and hence are more likely to be admissible. The multicast structure computed by the algorithms, is a shared multicast spanning tree, which is constructed as an overlay tree linking all group members. This approach has several important advantages. It implies that only nodes interested in taking part in the multicast communication would need to participate in the protocol operation. In addition, the overlay tree approach can offer high robustness and reliability for the multicast service [3]. Finally, when the number of group members is small compared to the network size, multicast overlay spanning trees achieve asymptotically optimal performance [2] and maximize the network capacity. Indeed, the key QoS metric is the capacity: every multicast group has a QoS requirement expressed as a total required bandwidth for the source(s) of the group. The algorithms we propose prevent congestion by considering the channel occupancy on each node on the paths and of other nodes within the interference area of the paths, during the construction of the multicast trees.

The rest of this paper is organized as follows: Section 2 provides background material from related work; Section 3 states the studied problem, and describes formally the QoS interference model; Section 4 introduces the different new algorithms; Section 5 evaluates and analyzes the algorithms by means of simulations; Section 6 discusses protocol aspects and future work; and Section 7 concludes.

2 Related Work

As discussed, our contribution is the introduction of quality of service for multicast in ad hoc networks. In this section, we first focus on quality of service in wireless networks, especially the concept of "residual capacity" of a node; we then detail multicast protocols and QoS for multicast.

Interference and QoS for Unicast in Ad Hoc Networks. Architectures of quality of service involve a complex set of features. They include the *semantics and QoS model* that quantify the performance for some metric of quality of service in some network model and may allow to check whether admitting a flow would result in acceptable performance, e.g. *admission control.* In this article, the focus is on the semantics and QoS model, whose main feature is *interference-awareness.* In wireless ad hoc networks, additional challenges are caused by the distinguishing features of the wireless medium, and a number of adapted QoS solutions have been proposed; see [4] for a recent survey. One of these features is the fact that the network operates on a shared medium, where the number of transmission opportunities should be carefully examined due to the issue of *interferences*. Essentially, two simultaneous transmissions by nearby emitters may interfere with each other, and as a result, may not be correctly received by their intended recipients. A solution is to ensure that no such concurrent transmissions will occur, and thus to avoid the *hidden terminal problem*. The *signal-to-noise ratio* (SNR) may be used to determine whether a set of transmissions may occur at the same time and still be successfully received, but a frequent simplification is to consider that two transmissions may occur simultaneously if the transmitters (and/or the receivers), are sufficiently far from each other [4].

With this assumption, several approaches exist, which we describe from the more complex for the wireless MAC and physical layers, to the less demanding:

• In slotted networks, where the transmission opportunity schedule is predetermined, a reservation algorithm can ensure that two nodes on the same channel and in the same area would never transmit on the same slot, as in [5]. For networks that are not using a slotted MAC layer, such as IEEE 802.11, the constraints of non-interference may still be expressed as conflict graphs, indicating which sets of transmitter-receiver pairs may transmit simultaneously, as in [6]. In both cases, this requires complex synchronization to be effectively implemented.
• Another approach is based on measurements: rather than pre-determining transmissions that would not interfere with each other, the idea is to measure the *medium occupancy* using carrier-sense features, and, based on this information, to accept only new flows for which the measured idle channel time on each node is sufficient, as in [7]. However, in practice the medium occupancy is not readily available for off-the-shelf equipment, hence estimates are often required, as proposed in [8]. As indicated in [8], the medium may be available for two neighbor nodes, but at different times, so in reality, they might not be able to communicate with each other: not only medium occupancy but also scheduling should be considered once again; however it was proposed to estimate the probability that such an event occurs (mismatch of idle medium intervals).
• The last approach is to ignore precise scheduling aspects, which are complexly linked to the wireless MAC and physical layers, and which are not robust with respect to mobility, and to only consider channel occupancy, of a node and of its neighbors. Then the *residual capacity* of a node, is defined as the minimum of the total time for which the channel is not occupied from the point of view of one node and the other nodes in its interference area. It is actually an upper bound of the transmission possibilities with scheduling.

This model can be used as an admission control algorithm: if there is not enough residual capacity, given the flows that are already present, a new flow is not accepted. [9] had shown that this admission control decision is an NP-complete problem, but [10] introduced efficient heuristics. This is the model and admission control that we use in this article, see Section 3.1 for details. Under a similar model, an entire QoS interference-aware version of the routing protocol

OLSR, was developed in [11], for unicast QoS flows; the channel occupancy is derived from a packet transmission counter of each node, and transmitted as an extension of OLSR messages. The soundness of the approach is validated in practice by simulations in [11] and experiments on real test-beds [12].

Multicast routing - Multicast & QoS. Many protocols have been proposed for multicast routing in mobile ad hoc networks. These protocols can be classified into two categories: tree based and mesh based protocols. While mesh based protocols use a single structure for each multicast group, tree based protocols can either use one tree per multicast group (shared tree) or create a separate one for each tuple (source, multicast group). Among the tree based multicast protocols we can cite MAODV [13], MOLSR [14], and MOST [3]. MAODV(Multicast Ad hoc On Demand Distance Vector) is an extension to the routing protocol AODV [15], and is a reactive multicast protocol. MOLSR (Multicast Optimized Link State Routing) is an extension to OLSR unicast routing protocol [16]. MOLSR uses the topology information given by OLSR to build a multicast tree for each tuple (source, group). MOST (Multicast Overlay Spanning Tree) [3] builds a shared overlay spanning tree between the members of the multicast group. As an example of mesh-based routing protocols, we mention On-Demand Multicast Routing Protocol (ODMRP) [17].

Multicast algorithm design for ad hoc networks is a complex problem in itself. One must take into account the characteristics of these networks such as the sensitivity of the shared wireless medium and the dynamicity of the nodes within the network. The main goal of the multicast protocols is to save bandwidth from unnecessary data transmissions. But maintaining a multicast structure implies additional overhead, and decreases the overall reliability. Using the same tree to forward multicast data for all the group members means that when a multicast packet is lost (notice that a multicast packet is not acknowledged by the receiver(s)) or when a link fails, a subset of the tree is prevented from receiving the multicast data. Mesh structures may offer more redundancy in some situations to cope with this problem. Multicast overlay structures use unicast tunnels and are less sensitive to packet loss than the basic multicast trees or meshes.

Most multicast protocols deal with multicast structure building, some of them try to improve the reliability, but only few of them are addressing the QOS requirements like QAMNET[18], QMR[19]. The basic idea of QAMNET is to extend existing approaches of mesh based multicasting and unicast QOS provisioning (the local capacity of a node is calculated as in SWAN [20]). QMR (QoS Multicasting Routing) is also a multicast mesh based protocol coupled with its own mechanism to evaluate the residual capacity locally on each node. In this protocol, nodes have to interact with their MAC layer to estimate the available bandwidth [21]. These protocols build a mesh structure in a reactive manner. Thus, there is no control neither a knowledge of the constructed mesh. In this case, the use of the overall network capacity is not optimized, and new nodes may be prevented from joining the multicast communicating groups. Both protocols try to estimate the residual bandwidth with different mechanisms, but, they do not consider the interferences problem efficiently during the bandwidth reservation. In fact, in QAMNET and QMR, bandwidth reservation is made locally on multicast mesh nodes. The neighboring nodes have to evaluate continuously the available bandwidth on their own, hence, their residual capacity is updated only when the multicast data flows start later on. Nodes transmissions from a same multicast group mesh may interfere in that case.

3 Methodology

In this section, we present the methodology we use to derive efficient algorithms for QoS multicast in multi-hop wireless networks. Our approach is based on the models and heuristics proposed in [9,10] concerning unicast routing. However, we present the models and the problem formulation in a slightly different manner, which allows us to generalize for the case of multicast communication and to propose improvements and performance estimates for the heuristics we consider.

3.1 Network and Interferences Model

As indicated in Section 2, we consider the interferences model that was introduced in [9]. We assume that a transmission from a node i interferes with all nodes within an interference zone, and equivalently, transmissions within this zone interfere with node i. This means that in order for a reception to be successful, it must be ensured that no other node within the interference zone is transmitting at the same time. We denote $\mathcal{I}(i)$ the set of nodes that are within the interference zone of node i. For example, if we assume at most two-hop interferences, $\mathcal{I}(i)$ will comprise the one-hop and two-hop neighbors of i, as well as the node i itself. In general, $\mathcal{I}(i)$ can be any set of nodes containing i. We also denote C_i the residual capacity of node i.

Let us consider two nodes i and j which can communicate directly. We will describe the effect on the residual capacities in the network, when a flow of x units of bandwidth is sent from i to j. All nodes in the interference zone of i will not be able to receive data in the same time that i is transmitting, and if a CSMA MAC protocol is used, such as IEEE 802.11, these nodes will not be allowed to transmit either. The model we consider here is consistent with this constraint, since we make no distinction between receiving and transmitting capacities. Similarly, transmissions from the nodes in the interference zone of node j will interfere with correct reception from j. These nodes will not be able to transmit data at the same time that j is receiving from i. Hence, when the flow x is injected, all nodes k in the interference zone $\mathcal{I}(i) \cup \mathcal{I}(j)$ (including i and j too) will have residual capacities updated to $C_k - x$. We note that our model is symmetric, in the sense that a flow from i to j has the same effect on the residual capacities as a flow from j to i.

Let us now consider a flow of x units of bandwidth following a route $\mathcal{R}$ from a source s to a destination t. We represent the route $\mathcal{R}$ as a set of links, *i.e.*, $\mathcal{R} = \{e_1, e_2, \ldots\}$. We will generalize the previous discussion to describe the residual capacities in this case. For the given route $\mathcal{R}$, we define the coefficients

λ_k corresponding to the number of route links (u, v) where at least one of the nodes u or v are located in the interference zone of node k:

$$\lambda_k = \sum_{(u,v)\in\mathcal{R},\ k\in\mathcal{I}(u)\cup\mathcal{I}(v)} 1 \quad . \tag{1}$$

These coefficients allow us to express the effective bandwidth which a flow x passing through the route $\mathcal{R}$ will consume in node k, due to interferences and/or transmissions by k. The effective bandwidth will depend on the route and, for a node k, it is equal to:

$$x_{\mathcal{R}} = \lambda_k x. \tag{2}$$

As a result, the residual capacities for all nodes in the network, after the flow x is accepted through the route $\mathcal{R}$, will be equal to: $C_k - \lambda_k x$.

3.2 Problem Statement

According to the model presented in the previous section, we can formulate the necessary conditions for the available capacities in the network, in order for a flow to be potentially accepted. Moreover, if a bandwidth reservation mechanism is in use, the effective bandwidth defined in (2) corresponds to the amount of bandwidth that each node would have to reserve for the given flow (we can include a factor to consider the MAC and scheduling overhead, as in [11]).

At first, we consider unicast routing. We present our formulation of the path with Residual Capacity problem (RC), defined and shown to be NP-complete in [9]. The objective is to find a route $\mathcal{R}$ in order to transmit x units of bandwidth from a source s to a destination t. The constraint is that the residual capacity of all nodes in the network must be larger (or equal) than 0. Using the previously presented notation, this can be expressed as follows:

$$\forall k : C_k - \lambda_k x \geq 0. \tag{3}$$

In other words, the capacity C_k must be larger than the total bandwidth which a flow x passing through the route $\mathcal{R}$ will consume in node k.

Using the same notation introduced in the previous section, we can formulate the problem of finding a multicast overlay spanning tree with residual capacity (MOST-RC), as a direct generalization of the previous case, which is obviously NP-complete too. An overlay tree corresponds to a set of routes (tunnels) between the multicast members. So, we can represent it as a set of links $\mathcal{T}$, in analogy to the route $\mathcal{R}$. For instance, a tree consisting of n tunnels $\mathcal{R}_i$ will be represented as $\mathcal{T} = \bigcup_{i=1..n} \mathcal{R}_n$, where the sources and destinations for the tunnels are chosen arbitrarily[1]. The objective is to find such an overlay tree spanning on all the multicast nodes, which can accept a flow of x units of bandwidth. Again, the constraint is that the residual capacity of all nodes in the network must be

[1] Due to the symmetry in the interference model, we do not need to consider a particular source in the tree, hence the arbitrary choices concerning the tunnel sources for the definition of the sets $\mathcal{R}_i$.

larger (or equal) than 0, as described in (3). In this case, the coefficients λ_k are defined with respect to a given tree $\mathcal{T}$, instead of a route $\mathcal{R}$, but their definition remains unchanged, *i.e.*, $\lambda_k = \sum_{(u,v)\in\mathcal{T},\ k\in\mathcal{I}(u)\cup\mathcal{I}(v)} 1$.

3.3 Heuristics

Since the problem we consider is NP-hard, we describe some heuristics that can be used in order to compute bandwidth-aware routes and overlay multicast trees. In this section, we provide a general discussion and the motivation for using these heuristics, as well as some arguments concerning the performance that can be achieved. A detailed description of the QoS multicast overlay spanning tree algorithms will be presented in Section 4.

The main goal of the heuristics we will describe is to find a route/tree that satisfies the capacity constraints (3). Moreover, it is desirable to find a solution that does not consume too many resources, so that future flow admission requests can be satisfied too. Hence, the heuristics take into account the available capacities in the network, so that nodes that have higher capacities are preferentially chosen as relays, while nodes that have low capacities are bypassed. We note that, while the heuristics can be used to compute in an efficient way a candidate route/tree, they offer no guarantee or definitive answer on whether a flow can be accepted in the network. Thus, the constraints in (3) must be checked after the computation, to verify whether the flow can actually be accepted.

From the definition of the coefficients λ_k we note that we only need to consider nodes in the route/tree, as well as the nodes in their interference zones. For all the remaining nodes we have $\lambda_k = 0$, and the residual capacities do not change. Hence, (3) is true if and only if:

$$\forall k \in \mathcal{I}(u) \cup \mathcal{I}(v),\ (u,v) \in \mathcal{T} : C_k \geq \lambda_k x, \tag{4}$$

where we took the case of the multicast tree for generality.

We can then write the following equivalent constraint formulation:

$$\min_{k\in\mathcal{I}(u)\cup\mathcal{I}(v),\ (u,v)\in\mathcal{T}} \frac{C_k}{\lambda_k} \geq x \tag{5}$$

According to the previous discussion, a candidate route for satisfying the given constraints will be the route that maximizes the route capacity in the left hand side of (5), or equivalently that minimizes the inverse capacity, *i.e.*,

$$\min \left\{ \max_{k\in\mathcal{I}(u)\cup\mathcal{I}(v),\ (u,v)\in\mathcal{T}} \frac{\lambda_k}{C_k} \right\}. \tag{6}$$

However, such a route may be too long and may consume too many network resources due to interferences. Consequently, the capacity of the network for accepting more flows could be unnecessarily reduced.

A better candidate route/tree can be found by the following heuristic. Each node k is associated with a weight $w_k = \sum_{j \in \mathcal{I}(k)} \frac{1}{C_j}$, *i.e.*, the sum of the inverse capacities of all nodes in the interference zone of k (including k). We can then define the weight of a route/tree as the sum of weights of the nodes in the route/tree. The heuristic consists in taking the route/tree with the minimum weight. Such a minimization can easily be performed with Dijkstra's algorithm or a classic spanning tree algorithm, for a route or an overlay tree respectively. This will result in avoiding nodes which have low capacities, or which interfere with other low capacity nodes. This approach was introduced in [10] for unicast routing, and it was found (via simulations) to achieve the best performance among other proposed approaches. We observe that the route/tree that we compute in this case minimizes the sum: $\sum_{k \in \mathcal{I}(u) \cup \mathcal{I}(v),\ (u,v) \in \mathcal{T}} \frac{\lambda_k}{C_k}$.

We can propose a variant of this heuristic, by considering weights $w_k = \sum_{j \in \mathcal{I}(k)} \frac{1}{(C_j)^n}$, where n can be any positive integer. We then define the following incremental algorithm: we start with $n = 1$ and if the route/tree discovery fails (*i.e.*, the capacity constraints are not satisfied) the value of n is increased. The procedure is repeated until the flow is accepted, or until n reaches a predefined upper threshold (in this case the flow cannot be admitted). In fact, larger values of n mean that low capacity nodes will have even larger weights and they will be chosen less often. So, we will compute longer routes/trees. As n tends to infinity, the computed route/tree will maximize the minimum residual capacity in all its interfering nodes, since we have that $\lim_{n \to \infty} \left(\sum_k \frac{1}{(C_k)^n} \right)^{\frac{1}{n}} = \max_k \frac{1}{C_k}$.

The above heuristics take into consideration the available capacities in the network, but they ignore the amount of bandwidth that is being requested by the current flow. Although this approach can be effective when the requested bandwidth is small compared to the residual capacities in the network, the performance can be improved by heuristics that adapt the route/tree computation to the particular amount of bandwidth requested each time. In fact, in the case of multicast flows, this adaptation is even more important, since a large number of nodes might be involved, and the flow may have a significant impact on the network conditions. As a result, better performance can be achieved if the node weights are updated during the algorithm execution, according to the new residual capacities. Let us assume that we use a greedy algorithm and the same definition of the node weights as before. If the weights are updated each time a node is added to the route/tree, the algorithm will minimize the sum[2]: $\sum_{k \in \mathcal{I}(u) \cup \mathcal{I}(v),\ (u,v) \in \mathcal{T}} \sum_{j=0..\lambda_k - 1} \frac{1}{(C_k - jx)^n}$. The route/tree we compute will then approach (as n tends to infinity) the route/tree which minimizes the least residual capacity in the network after the new flow has been admitted, *i.e.*, $\max \min_k (C_k - \lambda_k x)$. However, this optimization is achieved at the cost of an increase in the algorithm's running time complexity. In the next section, we discuss how a compromise can be found between the running time and the expected performance, with selective weight updates whenever a multicast node is added to the multicast overlay tree.

[2] We take $\frac{1}{C_k - jx} = \infty$ if $C_k - jx \leq 0$.

4 QoS Multicast Overlay Spanning Tree Algorithms

In this section, we present the algorithms which we use to construct multicast overlay spanning trees. All new algorithms are based on the spanning tree algorithm which is in use in the MOST protocol [3], combined with the heuristics we presented in Section 3.3. The performance of all algorithms described here will be compared via extensive simulations in the following section.

Basic MOST Algorithm. The algorithm in the MOST protocol computes a minimum spanning tree over all multicast nodes, by minimizing the size of the multicast tree in number of links/hops. An efficient algorithm for constructing such an overlay tree is presented in [3]. However, this algorithm does not take into consideration the available capacities in the network.

MOST with Unicast QoS. One possibility for improvement consists in taking an overlay tree computed by MOST and using unicast QoS routing independently for each individual tunnel in the overlay tree, as if each tunnel corresponded to a different unicast flow. This means that the overlay tree structure is not dependent on the available capacities in the network, since each multicast node will have exactly the same overlay neighbors as with the basic MOST algorithm. Nonetheless, some optimization is possible since the unicast tunnels will follow more appropriate paths, using the heuristics from [10].

Simple $\mathcal{Q}$-MOST Algorithm. It is possible to construct a better overlay tree, by taking into account the available capacities directly in the computation of the overlay tree structure. This can be achieved by using the defined weights, and by computing the overlay spanning tree that minimizes the sum of these weights. This minimization can be performed with algorithm 1, which is an adaptation of the basic MOST algorithm for our particular context. The difference consists in taking node weights instead of unit edge costs.

We denote $G(V, E)$ the network graph, where V is the node set, E is the edge set. Each node v is associated with a weight $W(v)$. We also denote S the set of multicast nodes. The array d associates each node with a distance to the multicast overlay tree, *i.e.*, $d[v]$ corresponds to the minimum distance of node v to the multicast nodes that are already part of the tree. This distance is initialized to $W[root]$ for the root node and to ∞ for all other nodes. The root node corresponds to the node with the smallest weight. The array *overlaypred* associates each node with an overlay predecessor multicast node. On the other hand, the array *pred* associates each node with its direct predecessor in the multicast tree (which is not necessarily a multicast node). These arrays need only be maintained during the computations, in order to construct the list of overlay routes *routeList*, which represents the complete multicast tree structure. The algorithm manages a set F of multicast nodes that have not been covered yet by the tree, and a min-priority queue Q which includes all nodes, with the priority attribute being equal to their distance d. In each iteration the algorithm chooses a node u with the smallest distance to the overlay tree (step 7), and

Algorithm 1: Efficient QoS Minimum Spanning Tree Algorithm

***Input*:** Graph $G(V, E)$, Multicast Node Set S, Node Weight Array W.
***Output*:** Overlay route List *routeList*

```
1.  for all $(v \in V)$ { $d[v] \leftarrow \infty$; $pred[v] \leftarrow NIL$; $overlayPred[v] \leftarrow NIL$;}
2.  $root \leftarrow \arg\min_i W[i]$
3.  $d[root] \leftarrow W[root]$;
4.  $Q \leftarrow V$;
5.  $F \leftarrow S$;
6.  while $(F \neq \varnothing)$ {
7.        $u \leftarrow$EXTRACT-MIN$(Q)$;
8.        if $(u \in S)$ {
9.              $d[u] \leftarrow W[u]$;
10.             DEL$(F, u)$;
11.             INSERT$(routeList, getRoute(overlayPred[u], u)$ }
12.       for each $(v \in adj[u])$ {
13.             if $(d[v] > d[u] + W[v])$ {
14.                   $d[v] \leftarrow d[u] + W[v]$;
15.                   if$(v \notin Q)$ { INSERT$(Q, v)$; }
16.                   if $(u \in S)$ $overlayPred[v] \leftarrow u$;
17.                   else $overlayPred[v] \leftarrow overlayPred[u]$;
18.                   $pred[v] \leftarrow u$; }} }
```

checks whether it is a multicast node (step 8). In this case, the node's distance is updated to $W[u]$ (because the node is added to the overlay tree) and it is removed from the set F. Afterwards, for each chosen node, steps 13−18 check its adjacent nodes on whether their distance can be improved, and update the predecessors appropriately, similarly to the basic MOST algorithm (*cf.* [3]).

The node weights are set according to the formula: $w_k = \sum_{j \in \mathcal{I}(k)} \frac{1}{C_j}$ for a node k. We call this algorithm variant *QOSMOSTsimple*. Similarly, we can use the weight definition $w_k = \sum_{j \in \mathcal{I}(k)} \frac{1}{(C_j)^n}$, where n is an integer. In this case, the algorithm is repeated with increasing values of n until the flow is accepted, or until n reaches a predefined upper threshold. This variant is denoted *QOSMOSTsimple-inc*. In the following, we denote N the number of nodes in the network, M the number of links, n the number of multicast members, and D the maximum number of nodes in an interference zone. The time needed to compute the weights for all nodes in the network, is $O(ND)$. Similarly, the complexity for checking whether the residual capacity constraints are satisfied is $O(TD)$, where T is the size of the multicast tree. Once the weights have been computed, the complexity of algorithm 1 is exactly the same as the basic MOST algorithm, *i.e.*, $O\left(n(N \log N + M)\right)$ in the worst case, and approximately equal to Dijkstra's algorithm on average (*cf.* [3]). Therefore, this particular algorithm has the important advantage of having low running time complexity. Moreover, we note that we do not need to take into account here the amount of bandwidth requested by a given flow. This means that this algorithm can be used to compute bandwidth-aware multicast overlay trees, which can be used regardless of the bandwidth requirements, in a group shared multicast tree protocol.

Improved $\mathcal{Q}$-MOST Algorithm. In case the QoS requirements are stricter, it is beneficial to propose an algorithm which can adapt to the precise amount of bandwidth requested by a given multicast flow. Therefore, an improved version of the $\mathcal{Q}$-MOST algorithm consists in updating the remaining capacities and the node weights whenever a multicast node is added to the overlay tree. More precisely, the following modifications must be made to algorithm 1, after step 11:

1. Update the residual capacities in the network (when the tunnel which connects the newly added multicast node to the tree has accepted flow x);
2. update the weights for all concerned nodes;
3. reset the distances of all nodes in the network to infinity (except multicast nodes that have already been added to the overlay tree).

Again, we define two variants of the improved $\mathcal{Q}$-MOST algorithm, depending on whether we use incremental weights or not: *QOSMOST* and *QOSMOST-inc*. According to the discussion in the previous section concerning weight updates, we note that the incremental algorithm will find an overlay tree which approaches the tree that maximizes the least residual capacity in the network, after the multicast flow is accepted. Hence, we expect to find a suitable multicast tree with respect to a flow request in almost all cases where such a tree exists. This performance gain comes at a (small) extra complexity cost. Once the node weights have been computed, the worst case complexity in both variants of the improved $\mathcal{Q}$-MOST algorithm is $O\left(n(N \log N + M) + TD\right)$, where T is the overlay tree size. However, the average case complexity is also higher due to the fact that the distances must be reset each time a multicast node is added to the tree.

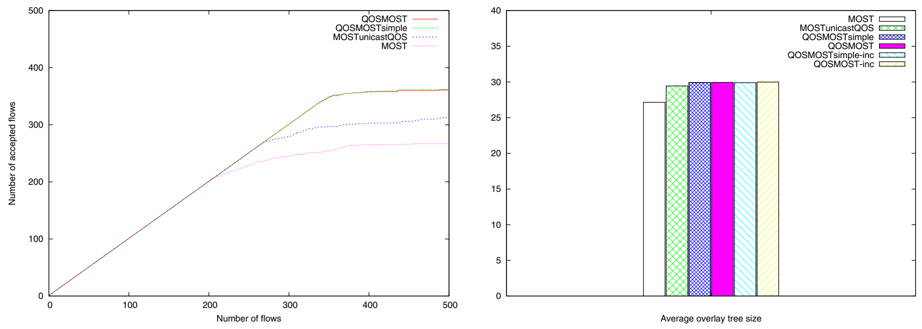
5 Simulations

In order to evaluate the performance of the variants of $\mathcal{Q}$-MOST, we perform extensive simulations, which are detailed in this section. The focus of the $\mathcal{Q}$-MOST algorithm is on wireless ad hoc networks. Such networks have been modeled as *unit disk graphs* of the plane, where two nodes are neighbors whenever their distance is lower than a fixed radio range. The simulator used was self-developed; the simulation parameters are given in table 1. In the simulations, the network considered is a square. The interference area of one node will be considered to be either the one-hop neighborhood, i.e. the area within range ρ; or the two-hop neighborhood (which is an approximation of the area within range $2 \times \rho$).

In the first scenario, 1000 nodes are randomly distributed, the interference area is the one-hop neighborhood, and every node has an initial capacity of 5000 units of bandwidth. The scenario repeatedly attempts to add more multicast trees, with a randomly selected group of 10 members, and each of the trees requiring a capacity of 10 units of bandwidth. The results are shown on Fig. 1 (we omit the plots corresponding to the incremental variants, because in this particular scenario they have exactly the same performance as the non-incremental

Table 1. Basic simulation parameters

Parameter	Value(s)
Network width L	$L = 1$
Number of nodes	200 or 1000
Range ρ	0.2 for 200 nodes, 0.1 for 1000 nodes
Position of the nodes	random uniform i.i.d



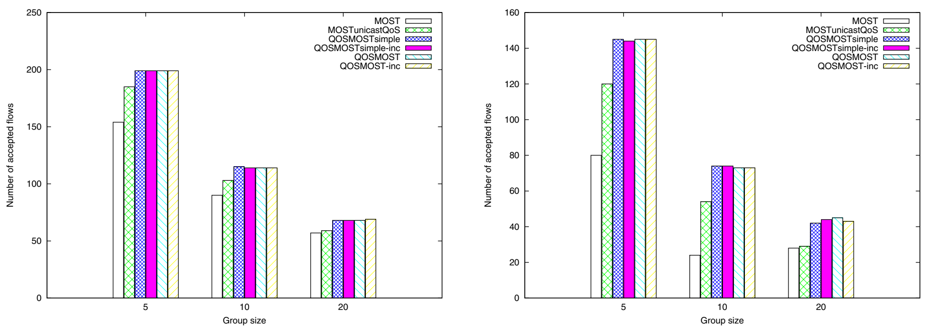
(a) Number of accepted groups (b) Average size of the accepted trees

Fig. 1. Admission control of 500 successive groups

algorithms). We first see on Fig. 1(a), the evolution of the accepted groups with the number of groups: on the x-axis, the network starts with 0 groups, and new groups are created randomly until 500 groups are reached. On the y-axis, the number of groups that are successfully added is displayed.

As one can see, both the simple $\mathcal{Q}$-MOST and improved $\mathcal{Q}$-MOST perform very similarly and clearly outperform the basic MOST algorithm and MOST with Unicast QoS, in spite of all the algorithms being simulated with the same groups. This result comes from the fact that, when congestion begins, while minimum spanning trees based on hop-count distances (MOST) are no longer necessarily accepted, the $\mathcal{Q}$-MOST algorithms are able to find trees with routes sidestepping from nodes with low residual capacity. MOST with Unicast QoS is able to do the same to some extent, but its performance on the scenario is only midway, illustrating that the fact that QoS for multicast is more than multicast using unicast QoS. Hence, the global optimization in the multicast tree structure performed by the $\mathcal{Q}$-MOST algorithms implies an important performance gain. Another interesting point is the sharper plateauing of the $\mathcal{Q}$-MOST algorithms (after about 350 admissions), showing that almost no group can be accepted after some congestion level is reached, hinting at the fact that these algorithms are efficient in finding possible trees when they exist. In Fig. 1(b), the average size of the final accepted overlay tree is shown for the various algorithms; it is the average total number of links in the tree. It appears that the size of the overlay tree of the $\mathcal{Q}$-MOST algorithm is not significantly larger than for basic

MOST; hence, although the heuristics avoid going through and near areas with low residual capacity, this is done in a adequate manner. Notice that using too large trees would have a negative performance impact on Fig. 1(a).



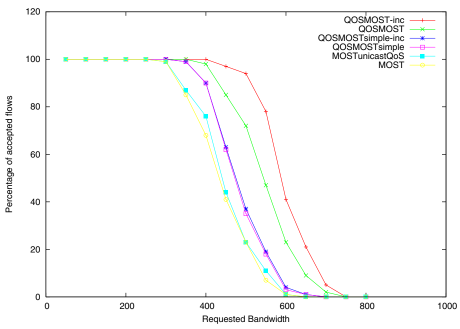
(a) With identical initial capacity (b) With initial capacity hole in the center

Fig. 2. Admission control of 500 successive groups for all algorithm

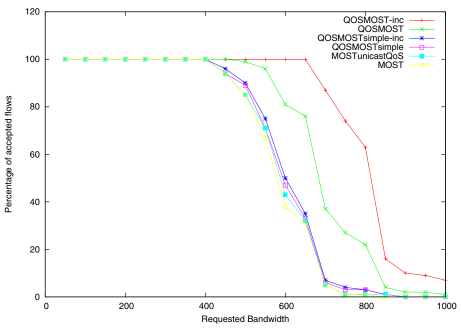
The second scenario is similar to the first one, except that the interferences area is the two-hop neighborhood, and that the number of members in each group varies: 5, 10 or 20. The results are represented on histograms in Fig. 2, which indicate the total number of accepted multicast groups after 500 attempts. Fig. 2(a) is for a network where all the nodes have the same initial capacity equal to 5000, whereas in Fig. 2(b), the nodes in a smaller square in the center of edge 0.5 have a halved initial capacity, that is 2500 units of bandwidth.

We see again that all the variants of $\mathcal{Q}$-MOST have a similar performance, considering the number of multicast groups that can be accepted, that MOST performs worse, and that MOST with Unicast QoS has intermediary performance. One significant result here, is comparing Fig. 2(a) and Fig. 2(b), it appears that the benefits of $\mathcal{Q}$-MOST over MOST increase, when the capacity is not uniform (for instance, when there is a capacity hole in the center). This last case corresponds to more realistic cases, where the center of the network is more likely to get congested, or where non-multicast traffic is not uniformly spread.

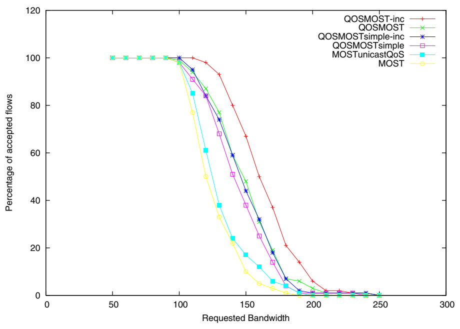
The last scenario focuses on the impact of the size of requests (requested bandwidth) on the performance. The two base scenarios are: either 1) with 200 nodes, or 2) with 1000 nodes. In addition, two sub-variants are tested, for different initial capacity assignments: 1) in the first, every node has an initial capacity selected uniformly at random in the range of 4000 to 5000 units; 2) in the second, every node has an initial capacity of 5000, except for nodes in the center square, a capacity "hole", which have a random capacity in the range of $2000 - 2500$ units. We present here the results when in the first variant we assume one-hop interferences, while in the second variant we assume two-hop interferences. For the four combinations, the scenario simulates the arrival of *one* group with a large bandwidth requirement, with 10 members. The metric



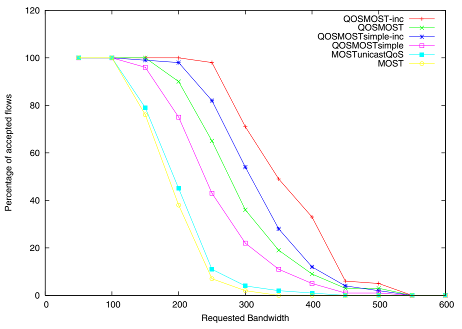
(a) 200 nodes, 1-hop interferences, no capacity hole



(b) 1000 nodes, 1-hop interferences, no capacity hole



(c) 200 nodes, 2-hop interferences, with capacity hole



(d) 1000 nodes, 2-hop interferences, with capacity hole

Fig. 3. Admission probability w.r.t. requested bandwidth (avg. of 100 requests)

is the average admission rate, based on 100 attempts with randomly selected members on the same topology and with the same bandwidth requirement. The results are on Fig. 3.

On these figures, one can see the admission rate: for instance, on Fig. 3(d), for a requested bandwidth equal to 250 units, the admission probability is 7% for MOST, compared to 99% for the iterative improved $\mathcal{Q}$-MOST. The results get better and better when the algorithms are more sophisticated and more interference aware; noticeably, unlike previous results from Fig. 3, this time the difference of performance between the variants of $\mathcal{Q}$-MOST are more clearly established: this comes from the fact that only one request is to be satisfied at a time, hence, algorithms cannot compensate by accepting later "easier" requests. The best $\mathcal{Q}$-MOST variant in all cases is the incremental improved $\mathcal{Q}$-MOST algorithm, which can nearly accept the double of the requested bandwidth compared to the basic MOST algorithm, in the more critical case of Fig. 3(d).

6 Protocol and Future Work

In the previous sections, we presented $\mathcal{Q}$-MOST as an algorithm, and evaluated its performance by simulation, assuming all information is available to every node. In practice, one needs to extend the QoS model that is used, in order to create a protocol with proper signaling. In reality, most of the ingredients necessary to construct such an actual protocol already exist, if the OLSR routing protocol is selected as the routing protocol of the ad hoc network: the multicast signaling and network interfacing are available with MOST [3], which in turns reuses implementations and ideas from MOLSR [14] - both have been implemented and tested in real test-beds. Similarly, the mechanisms and protocols for exchanging the necessary information about quality of service and residual capacities have already been designed in a QoS interference-aware version of OLSR [11], implemented, tested and validated on test-beds [12]. The two remaining difficulties are the following: the first one is that consistent information needs to be used for the multicast tree computation; this is easily overcome by using some kind of global counter/timing in order to decide which sets should be included in the calculation. The second is more general: the model only considers channel occupancy and not scheduling as indicated in Section 2, and although it has excellent performance in practice [12], precise probabilistic arguments such as in [8] would be interesting. Lastly, the exact specification of the protocol and its implementation are also subjects of future work.

7 Conclusions

In this article, we presented a family of algorithms, $\mathcal{Q}$-MOST, for multicast in ad hoc networks. The central feature of these algorithms is that they integrate quality of service constraints based on the concept of residual capacity, as well as interference-awareness, with a specific model of interferences. They perform admission control. They essentially compute a multicast tree, linking nodes of the group, satisfying the QoS interference constraint that the residual capacity must never go below zero in the network; in addition, the multicast tree is an overlay tree, hence two neighbors in the tree are linked by reliable multi-hop routes which are themselves interference-aware. The difficulty is that even unicast route calculation is NP-complete within this model, hence we have proposed several variants of the $\mathcal{Q}$-MOST algorithms based on efficient heuristics. We have experimentally shown via simulations the excellent behavior of these heuristics on several scenarios: for instance, compared to the basic MOST algorithm, they allow to admit noticeably more groups. Future work includes considering the impact of scheduling, and developing an actual protocol specification and implementation.

References

1. Gupta, P., Kumar, P.R.: Capacity of Wireless Networks. IEEE Trans. Inf. Theory 46(2), 388–404 (2000)
2. Jacquet, P., Rodolakis, G.: Multicast Scaling Properties in Massively Dense Ad Hoc Networks, SANSO, Fukuoka, Japan (2005)

3. Rodolakis, G., Meraihi Naimi, A, Laouiti, A.: Multicast Overlay Spanning Tree Protocol for Ad Hoc Networks, WWIC, Coimbra, Portugal (2007)
4. Hanzo, L., Tafazolli, R.: A Survey of QoS Routing Solutions for Mobile Ad hoc Networks IEEE Communications Surveys & Tutorials (2007)
5. Lin, C.R., Liu, J.-S.: QoS Routing in Ad Hoc Wireless Networks, IEEE Journal on Selected Areas in Communications, 1426–1438 (August 1999)
6. Gupta, R., Jia, Z., Tung, T., Walrand, J.: Interference-aware Qos Routing (IQRouting) for Ad-Hoc Networks. In: Proc. Globecom 2005, vol. 5 (November 2005)
7. Ge, Y., Kunz, T., Lamont, L.: Quality of Service Routing in Ad-Hoc Networks Using OLSR. In: Hawaii International Conference on System Sciences (January 2003)
8. Sarr, C., Chaudet, C., Chelius, G., Guerin-Lassous, I.: A Node-Based Available Bandwidth Evaluation in IEEE 802.11 Ad Hoc Networks. International Journal of Parallel, Emergent and Distributed Systems (July 2005)
9. Georgiadis, L., Jacquet, P., Mans, B.: Bandwidth Reservation in Multihop Wireless Networks: Complexity, Heuristics and Mechanisms, WWAN, Japan (2004)
10. Allard, G., Jacquet, P.: Heuristics for Bandwidth Reservation in Multihop Wireless Networks, INRIA Research Report RR-5075 (January 2004)
11. Nguyen, D., Minet, P.: Quality of Service Routing in a MANET with OLSR. Journal of Universal Computer Science (JUCS) 13(1), 56–86 (2007)
12. Nguyen, D., Minet, P., Adjih, C.: Quality of service for OLSR: Implementation and Measures on a Real Military MANET, 3rd OLSR Interop, Japan (October 2006)
13. Royer, E., Perkins, C.: Multicast Ad hoc On-Demand Distance Vector (MAODV) Routing, IETF, Intemet Draft: draft- ietf-manet-maodv-00.txt (2000)
14. Laouiti, A., Jacquet, P., Minet, P., Viennot, L., Clausen, T., Adjih, C.: Multicast Optimized Link State Routing, INRIA research report RR-4721 (2003)
15. Perkins, C., Belding-Royer, E., Das, S.: Ad hoc on-demand distance vector (AODV) routing, RFC 3561 (2003)
16. Clausen, T., Jacquet, P., Adjih, C., Laouiti, A., Mühletaler, P., Minet, P., Qayyum, A., Viennot, L.: Optimized link state routing protocol, RFC 3626 (2003)
17. Lee, S., Su, W., Gerla, M.: On demand multicast routing protocol in multihop wireless mobile networks, ACM/Baltzer Mobile Networks and Applications (2000)
18. Tebbe, H., Kassler, A.: QAMNet: Providing Quality of Service to Ad-hoc Multicast Enabled Networks, ISWPC, Thailand (2006)
19. Saghir, M., Wan, T.C., Budiarto, R.: Load Balancing QoS Multicast Routing Protocol in Mobile Ad hoc Networks, AINTEC, Bangkok, Thailand (2005)
20. Ahn, G., Campbell, A.T., Veres, A., Sun, L.: Supporting Service Differentiation for Real-Time and Best Effort Traffic in Stateless Wireless Ad Hoc Networks (SWAN), IEEE Transactions on Mobile Computing (September 2002)
21. Saghir, M., Wan, T.C., Budiarto, R.: QoS Multicast Routing Based on Bandwidth Estimation in Mobile Ad Hoc Networks, ICCCE, Malaysia (2006)