

QoS Routing in Ad Hoc Networks Using QOLSR With no Need of Explicit Reservation

Hakim Badis^{*†}, Ignacy Gawędzki^{*} and Khaldoun Al Agha^{*†}

^{*}LRI Laboratory, University of Paris XI, 91405 Orsay, France

[†]INRIA Laboratory, Rocquencourt, 78153 Le Chesnay, France

Abstract—Multimedia applications often require guaranteed quality of service (QoS) and resource reservation, which has raised a number of challenging technical issues for routing. We have proposed the QOLSR protocol, a QoS routing over OLSR protocol introducing metrics such as bandwidth and delay that are more appropriate than the hop distance. When using the QOLSR protocol, there are cases in which a source node continuously changes a flow's next hop in response to the change of available bandwidth on its path and cannot tell apart the traffic induced by itself from traffic generated by other nodes. This article describes a way to achieve QoS routing without using explicit reservation mechanisms and gives new distributed solutions to the oscillation and collision of flows. We can guarantee that the flows take the appropriate paths and avoid interferences with other existing flows. The performance of our distributed algorithm is extensively investigated by analyses, examples and simulations in both static and dynamic networks. Our results show that the gain achieved by our proposal represents an important improvement in mobile wireless ad hoc networks.

I. INTRODUCTION

A wireless ad hoc network is a special type of wireless network that does not have a wired infrastructure to support communication between the wireless nodes. In multi-hop ad hoc networks, communication between two nodes that are not direct neighbors requires relaying of messages by some intermediate nodes. Each node acts as a router as well as a communication end-point.

Many modern network applications, such as transmission of multimedia data, real time collaborative work, and interactive distributed applications, require QoS provisions to work properly, hence the question of QoS routing in ad hoc networks, which is an intensively studied subject. Most routing protocols for mobile Ad hoc networks (MANETs) [1], such as OLSR [2], AODV [3], DSR [4], are designed without explicitly considering the QoS of the routes they find. QoS routing requires not only to find a route from a source to a destination, but a route that satisfies the end-to-end QoS requirement, often given in terms of bandwidth, delay or loss probability. Quality of service is more difficult to achieve in ad hoc networks than in their wired counterparts, because the wireless bandwidth is shared among adjacent nodes and the network topology changes unpredictably as the nodes move. This requires extensive collaboration between the nodes, both to establish the route and to secure the resources necessary to provide the QoS. We have proposed the QOLSR protocol in [5]–[7], which is an enhancement of the OLSR routing protocol to support multiple-metric routing criteria.

QoS parameter reservation is more difficult in proactive than reactive protocols. In the proactive case, the routing protocol periodically updates the reachability information in the nodes' routing tables. Thereby a route is immediately available when needed. Proactive protocols use an adaptive system of routing based on periodic exchange of control messages. There may be various kinds of control messages: those which are sent locally (broadcasted to one-hop neighbors) to enable a node to discover its local neighborhood (as HELLO messages in the QOLSR protocol); and those which are sent to the entire network to allow the distribution of the topology and QoS information to all the nodes (as TC messages in the QOLSR protocol).

Our QOLSR protocol suffers from some defects related to QoS information diffused in TC messages. In this article, we describe the problems of oscillation and collision of flows and we propose innovative distributed solutions backed with analyses and examples. We can guarantee that the flows take the appropriate paths and avoid interferences with other existing flows. Paths selected by source nodes satisfy the end-to-end bandwidth and delay requirements.

This paper is organized as follows. Section II presents our QOLSR protocol, which is an enhancement of the OLSR routing protocol to support multiple-metric routing criteria. The oscillation problem and the proposed solution in the QOLSR protocol are presented in Section III. In section IV we identify the flow collision problem and the backoff-based solution. Finally, we present our conclusions.

II. THE QOLSR PROTOCOL

OLSR is a proactive routing protocol, which inherits the stability of a link state algorithm and has the advantage of having the routes immediately available when needed thanks to its proactive nature. The OLSR protocol uses a kind of Dijkstra's shortest path algorithm to provide optimal routes in terms of number of hops. QOLSR is an enhancement of the OLSR routing protocol that supports multiple-metric routing criteria. A detailed description of the protocol can be found in [5]–[7].

Let $G = (V, E)$ be the network with $|V|$ nodes and $|E|$ arcs and $p = (i, j, k, \dots, q, r)$ a directed path. For the Hard Real time applications we use a shortest path algorithm. For delay or jitter metric, each arc (i, j) in the path p is assigned a real number $\text{del}(i, j)$. When the arc (i, j) is inexistent or j is not a MPR of i (Referring to the OLSR routing mechanism), then

$\text{del}(i, j) = \infty$. Let $\text{del}(p) = \text{del}(i, j) + \text{del}(j, k) + \dots + \text{del}(q, r)$. The routing problem is to find a path p^* between i and r so that $\text{del}(p^*)$ is the minimum. In such a case, we use the well-known Dijkstra routing algorithm. For bandwidth metric, each arc (i, j) in the path is assigned a real number $\text{bw}(i, j)$. When the arc (i, j) is inexistent or j is not a MPR of i , $\text{bw}(i, j) = 0$. Let $\text{bw}(p) = \min\{\text{bw}(i, j), \text{bw}(j, k), \dots, \text{bw}(q, r)\}$. The routing problem is to find a path p^* between i and r that maximizes $\text{bw}(p^*)$. In order to implement such a metric, we use a variant-Dijkstra algorithm.

For the Soft Real time applications, We consider the Delay and Bandwidth Constrained Least Hop path problem (DB-CLH). Given two constants, the minimum bandwidth Δ_{bw} and the maximum delay Δ_{del} . The Delay and Bandwidth Constrained Least Hop path problem (DBCLH) is to find a path p from i to r minimal for a hop-count, satisfying $\text{del}(p) \leq \Delta_{\text{del}}$ and $\text{bw}(p) \geq \Delta_{\text{bw}}$. The formal description is: $\min\{\text{hop}(p) : p \in P(s, t) \text{ and } \text{del}(p) \leq \Delta_{\text{del}} \text{ and } \text{bw}(p) \geq \Delta_{\text{bw}}\}$ where $P(s, t)$ is the set of paths from the source node s to the destination node t , and $\text{hop}(p)$ is the hop-count. It is clear that the problem with two additive and one concave metrics is *NP-Complete* [8]. We have proposed an efficient heuristic based on the Lagrange Relaxation method to resolve this problem [9]. If there is no path that satisfies the constraints, we use the Best Effort method. Each node in the ad hoc network makes its decision (on which next hop to use) using information in incoming packet and its topology table. To avoid the recomputation of next hops for every one of them, each incoming packet must be mapped into some traffic class; all packets in the same class get the same treatment. The choice of a class may be based on the contents of the packet's header.

III. OSCILLATION PROBLEM IN THE QOLSR PROTOCOL AND THE PROPOSED SOLUTION

Consider the graph presented in Figure 1.

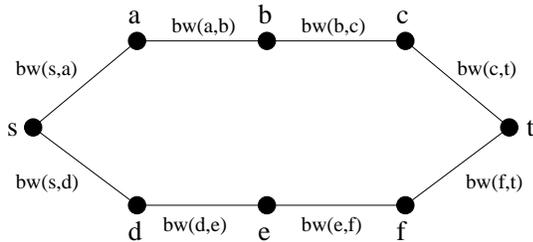


Fig. 1. Oscillation problem

There are two routes from the source node s to the destination t .

Let the raw capacities of the paths $p_1 = (s, a, b, c, t)$ and $p_2 = (s, d, e, f, t)$ be respectively $c(p_1) = \min\{c(s, a), c(a, b), c(b, c), c(c, t)\}$ and $c(p_2) = \min\{c(s, d), c(d, e), c(e, f), c(f, t)\}$

Let the used capacities of the paths $p_1 = (s, a, b, c, t)$ and $p_2 = (s, d, e, f, t)$ be respectively

$u(p_1) = \max\{u(s, a), u(a, b), u(b, c), u(c, t)\}$ and $u(p_2) = \max\{u(s, d), u(d, e), u(e, f), u(f, t)\}$ such that $u(p_2) > u(p_1)$. We suppose that node s sends traffic data with average rate Δ_s such that $u(p_1) + \Delta_s > u(p_2)$. It follows that $u(p_2) + \Delta_s > u(p_1)$. Path p_1 is the best path which fulfills the capacity requirement from s to t and s therefore selects a as the next hop. As a result, node s sends its data traffic to node a . Consequently, the used capacity of p_1 is $u(p_1) + \Delta_s$. Since $u(p_1) + \Delta_s > u(p_2)$, node s will select path p_2 as the best path, so the next hop becomes d . When s sends its traffic data afterwards, the used capacity of p_2 becomes $u(p_2) + \Delta_s > u(p_1)$. Obviously, s will continuously change the next hop in response to the change of capacity on its path. Node s cannot tell apart the traffic induced by itself from traffic generated by other nodes. We call this the *oscillation* problem.

To avoid it we propose the following solution (Oscillation-Avoidance 1: OA1): When a node detects a change in network conditions (via HELLO and TC messages), it checks for each QoS flow, either generated locally or forwarded from another node, whether the respective next hops can be kept without breaking the flows' QoS requirements. This is achieved by applying the following distributed rule: each node accounts the amount of traffic sent for each flow and removes the resulting rate from the used capacity of every arc on the previously calculated path starting from the current next hop towards the destination. Then it checks whether the path is still satisfying the QoS requirement; if not, a new one is calculated starting from the current node using basic QOLSR.

For each known QoS flow, every node must remember not only the next hop but also all the subsequent hops on the path towards the destination for this rule to be usable. This information can be stored in the topology tuples, however, for performance considerations, it would be much better to keep it on a separate topology graph. This graph is maintained along with the topology tuples and is much more useful for routes calculation and retrieval of known paths.

Discussion

Information about forward section of the path is crucial for a node to decide if the next hop is still valid for each flow. If a node keeps information about the next hop alone, it loses the ability to distinguish the part of used capacity that is induced by the given flow from other ones.

We have previously proposed a distributed method for oscillation avoidance that uses global topology information in addition to the next hop alone to decide whether to change the path.

The method is similar to OA1, but since information about the previously calculated path is not available, the current node removes the used capacity from the used capacity of each arc of every feasible path starting from the next hop towards the destination. Of course if an arc's used capacity is lower than the actually used capacity for the current flow, the paths to which this arc belongs are clearly not the ones of interest and are ignored. If at least one feasible path remains, the next hop

is kept as is, otherwise a new path is calculated starting from the current node.

Theorem 1: This rule avoids the *oscillation* problem.

Proof: We use the *enumeration proof* that generates and then analyzes all possible cases for a problem or some aspect of a problem.

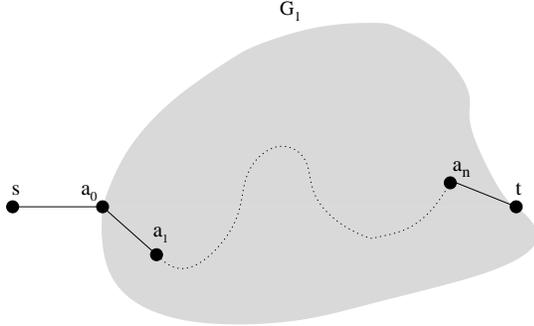


Fig. 2. A subgraph G_1 betw. a_0 and t

Let p be the best path selected by the source node s to the destination node t that fulfills the QoS requirement (capacity and delay) in a graph $G = (V, E)$ using the QOLSR protocol. Let $p = (a_1, \dots, a_{i-1}, a_i, a_{i+1}, \dots, a_n)$ and $G_1 = (V \setminus \{s\}, E \setminus \{(s, a_0)\})$ a subgraph of all paths between a_0 and t not containing s . Node s keeps only the next hop a_0 in its routing tables and has no information about how the next hop will further forward the traffic. The only information available to s is that its traffic will be forwarded by a_0 . Let suppose that the traffic is forwarded to t along path p and that the used of path (p, t) is ℓ (see figure 2). We suppose that node s sends traffic with an average rate of Δ_s and that ξ is the set of arcs that are correlated (they share the same capacity) with arcs of path p .

a) Suppose that the new used capacity of (p, t) is $\ell + \Delta_s$, which means that the path (p, t) is forwarding only traffic from s and there is no other traffic on any arc of ξ . Using the OA1 rule, s removes Δ_s from every arc in G_1 and checks if at least one paths is satisfying the QoS requirement from a_0 . The used capacity of path (p, t) is $\ell + \Delta_s$ during the transmission. But by subtracting Δ_s from each arc of p , we guarantee that there exists at least one path (actually (p, t)) satisfying the QoS requirement (Δ_s). Consequently, node s keeps a_0 as the next hop.

b) Suppose that the new used capacity of (p, t) is $\ell + \Delta_s + \tau$. τ is introduced by some other traffic on arcs of p or ξ . Using the OA1 rule, the used capacity of path (p, t) will be $\ell + \tau$. If $\ell + \tau < u(p, t) - \Delta_s$, s keeps a_0 as the next hop. Otherwise, by subtracting Δ_s from the used capacity of every arc of G_1 , a path other than (p, t) satisfying the QoS requirement (Δ_s) exists if and only if there is another path between a_0 and t . This case, however, shows a weak point in the OA1 rule: a source node considers all traffic beyond the next hop as the same. This leads to routing equivalent to best effort. In order to improve performance, we present a way to take packet loss into account in the next section.

Unfortunately, this method does work in some specific situations. Consider the cases depicted in figures 3 and 4: node s_2 is unable to distinguish between the two situations whereas it should act differently in each case. If figure 3, s_2 should choose to route the flow through e , but the oscillation avoidance rule tells it to keep routing through a .

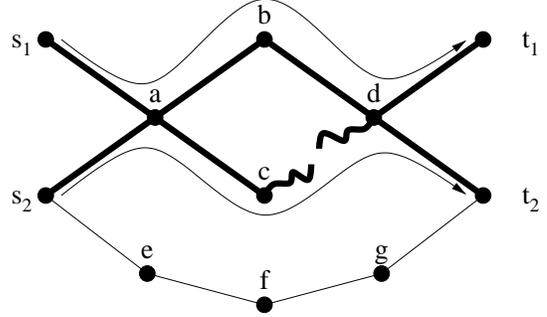


Fig. 3. A problematic example

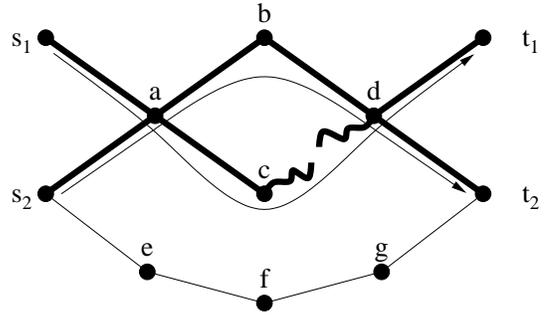


Fig. 4. A second problematic example

In the first case, s_2 should find the alternate path, whereas in the second case, it should keep the same next hop and s_1 should stop the flow.

IV. THE FLOW COLLISION PROBLEM AND THE BACKOFF-BASED SOLUTION

If two distinct nodes initiate some QoS traffic based on the same knowledge of the network, they might choose paths that share common arcs, possibly overflowing the arcs' capacity. We call the phenomenon *flow collision*. Figure 5 shows a simple case of flow collision. The use of the oscillation-avoidance rule does not help, as each node would keep sending traffic along the same paths, since they cannot see by what amount the capacity is overflowed.

To estimate the amount of overflow of the arcs' capacities, we propose to take into account the packet loss ratio on each arc. As a source node can block the transmitting process when too much traffic is sent, an intermediate node has no other choice than to drop excess traffic when its buffers are full. Thus, the excess bandwidth of an arc could be estimated using the proportion of lost data. This information is accounted for each link by the forwarding nodes and is diffused in the TC

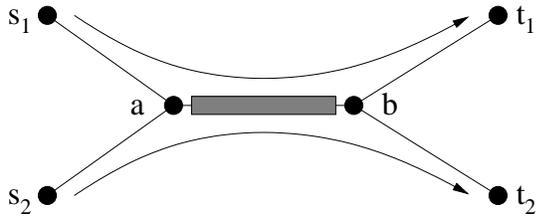


Fig. 5. Simple case of flow collision.

messages. This way, the amount of overflow is known to each node in the network and enables then to take turns to find alternative paths (OA2 rule: OA1 rule plus taking into account the lost packet ratio).

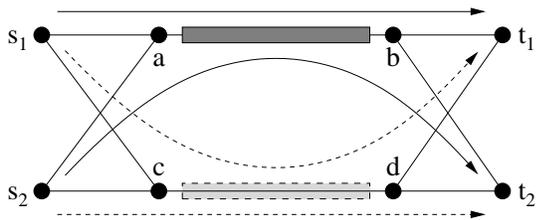


Fig. 6. Oscillation by the collision-avoidance rule.

Nevertheless, the new rule alone is not enough to prevent flows from colliding: there is no way that nodes of two colliding flows behave differently in response to arc overflow. In the case of figure 6, both flows will block, although at least one of them could use the path. Consider now another example, depicted in figure 7. Supposing that initially both flows share arc (a,b) , they would both choose another path, and may choose paths with yet another shared arc, say (c,d) and fall into a completely equivalent situation. This shows how the collision-avoidance rule may introduce, quite paradoxically, another form of oscillation.

Therefore, some asymmetry has to be introduced to enable nodes of colliding flows to take different decisions regarding route change. This asymmetry can be achieved with the use of our proposed backoff-based solution. The idea is that the first node in the flow's current path that precedes an overflowed arc and that has an alternate route satisfying the flow's QoS requirement should be considered for route switching. This node should initialize a random backoff timer and maintain the current next hop as long as the timer has not expired. At timer expiration, the node applies OA2 again and if the flow keeps colliding, then the node calculates a new path using standard QOLSR, instead of keeping the next hop. The node responsible for the switch is chosen using the following method: each node applies the OA2 rule and notifies the next hop if it is determined to have an alternate route, otherwise it waits for a notification from its preceding node. On reception of a notification, a node applies the OA2 rule if it has not done so already and notifies its next hop if it is determined to have an alternate route and so on. If the application of OA2 gives a regative answer regarding alternate routes from the next

hop, the current node is the one that should switch the flow. It then extracts the source address of the notification message which is the address of the preceding node on the current path and removes this node from the topology for alternate path calculation. Removal of the preceding node from the topology avoids finding paths that would route the flow back to it.

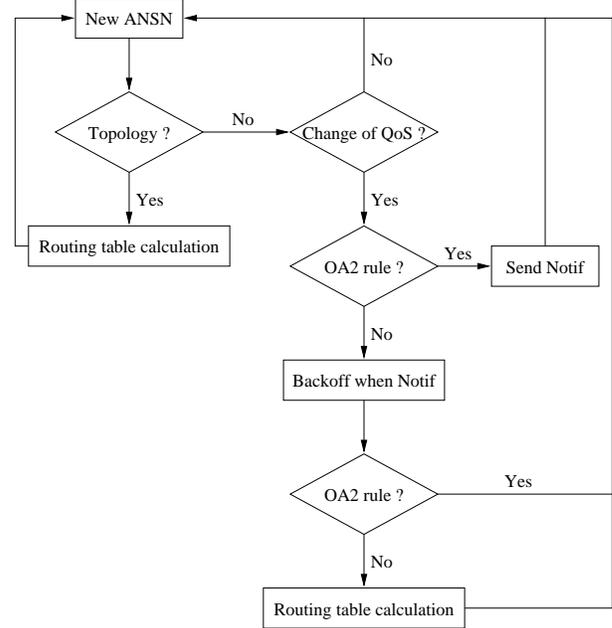


Fig. 7. The proposed QoS model using QOLSR.

Figure 7 represents the proposed QoS model. Each TC message contains a Advertised Neighborhood Sequence Number (ANSN) which helps keeping track of topology and QoS changes. Upon reception of a TC message with a new ANSN, if the topology has changed, then the routing tables have to be recalculated using plain QOLSR methods. Otherwise, if the state of QoS of the network has changed for a given flow, then the OA2 rule must be applied to decide whether the path must be changed. If there exists at least one feasible path from the next node on the current path to the flow's destination, then a change of path must be performed by some node downstream. To inform these that the current node maintains its next hop for the current flow, a notification message is sent to the next node. When a node receives the notification, it applies the OA2 rule if not done previously and behave exactly as the sender of the notification if a feasible path exists from the next node. Otherwise, it means that this node has to resolve the collision itself.

When a node is in charge of resolving a collision, it selects a random backoff timer value and delays further actions. Upon timer expiration, the OA2 rule is applied again to decide whether the current path has been freed by the colliding flow, in which case the collision is resolved. If the flow is still colliding, then the node calculates the alternate path, not using the current next hop. To avoid sending the flow back through the previous node, the source address of the

notification message is extracted and the link to the previous node is removed during path calculation.

The size of the backoff window must be set to reflect the priorities of flows, since a larger window increases the probability of selecting a timer value larger than that of colliding flows and consequently decreases the probability that the flow will have to switch routes.

If two colliding flows select timer values too close to each other to let the later one witness the switch, then we can say that the collision resolution failed and has to be performed once more. Moreover, it may also happen that a switching flow collides again with another flow on the alternative path. Then the "older" flow should obviously be prioritized over the more recent one. Therefore, each time a collision occurs after route switching, the size of the backoff window has to be doubled.

V. FURTHER IMPROVEMENTS

For collision resolution, other flow parameters can be taken into account to set the initial size of the backoff window. For example, if a flow's loss probability has to remain low, the backoff timer should be shorter to avoid losing too much packets during the resolution process while the other flow's timer decreases. The same idea can be applied to delay and bandwidth, since they are both strongly affected by capacity overflow of arcs.

VI. CONCLUSIONS

We have described the problems of oscillation and collision of flows and proposed new distributed rules to guarantee that the flows take the appropriate paths and avoid interferences with existing flows. Our proposed rules are based on analyses and examples. Simulations will follow shortly. Simulation results will show the performance of the QOLSR including our distributed rules comparing with the OLSR and standard QOLSR protocols.

REFERENCES

- [1] "<http://www.ietf.org/html.charters/manet-charter.html>."
- [2] T. Clausen and P. Jacquet, "Optimized Link State Routing Protocol," *RFC 3626*, October 2003.
- [3] C. Perkins, E. M. Royer and S. R. Das, "Ad Hoc On-Demand Distance Vector (AODV) routing," *RFC 3561*, July 2003.
- [4] D. Johnson and A1, "The Dynamic Source Routing Protocol for Mobile Ad hoc Networks (DSR)," *In IETF Internet Draft, draft-ietf-manet-dsr-09.txt*, April 2003.
- [5] A. Munaretto, H. Badis, K. Al Agha and G. Pujolle, "A Link-state QoS Routing Protocol for Ad Hoc Networks," *In the proceedings of IEEE MWCN2002, Stockholm, Sweden*, September 2002.
- [6] H. Badis, A. Munaretto, K. Al Agha and G. Pujolle, "QoS for Ad hoc Networking Based on Multiple Metrics: Bandwidth and Delay," *In the proceedings of IEEE MWCN2003, Singapore*, October 2003.
- [7] H. Badis, K. Al Agha, A. Munaretto and G. Pujolle, "Optimal Path Selection in a Link State QoS Routing," *In the proceedings of IEEE VTC'04 spring, Italy*, May 2004.
- [8] F. Kuipers, P. Van Mieghem, T. Korkmaz and M. Krunz, "An Overview of Constraint-Based Path Selection Algorithms for QoS Routing," *IEEE Communications Magazine*, vol. Vol.40, No.12, December 2002.
- [9] H. Badis and K. Al Agha, "A Distributed Algorithm for Multiple-Metric Link State QoS Routing Problem," *In the proceedings of IEEE MWCN2003, Singapore*, October 2003.