Quality of Service on Ad-hoc Wireless Networks

PhD Thesis

Rafael Paoliello Guimarães

Barcelona, May 2008

Quality of Service on Ad-hoc Wireless Networks

Barcelona, May 2008

Universitat Politècnica de Catalunya
Departament d'Arquitectura de Computadors

Quality of Service on Ad-hoc Wireless Networks

PhD Thesis

Rafael Paoliello Guimarães

PhD Advisor:

Prof. Dr. Llorenç Cerdà Alabern	Universitat Politècnica de Catalunya, Spain

Members of the PhD tribunal:

Prof. Dr. Ulf Körner	Lund University, Sweden
Prof. Dr. Davide Careglio	Universitat Politècnica de Catalunya, Spain
Prof. Dr. Matteo Cesana	Politecnico di Milano, Italy
Prof. Dr. Josep Solé-Pareta	Universitat Politècnica de Catalunya, Spain
Prof. Dr. David Remondo-Bueno	Universitat Politècnica de Catalunya, Spain

Contents

List of Figures

List of Tables

Preface and Acknowledgments

First of all I would like to strongly thank my wife, Aline, for being at my side all these years that took me to finish this PhD thesis. Thanks for being so patient and comprehensive, mainly on that (possibly) boring moments when I tried to explain to you what was my PhD about.

I would also like to thank my parents and my sister for always believing on me and being there whenever I needed them. Mom, dad, "Lucinha": I did it!

Thanks to my parents-in-law, my sister-in-law and my grandma-in-law.

Thanks to my advisor, Llorenç, for all the productive discussions we have had at the course of my PhD. Thanks for helping me so much to make this happen.

Thanks to the CompNet professors for all the support since the beginning of my PhD up to the last moment. Thanks Jorge and José Maria.

Thanks also to all great friends I have had the pleasure to know during these years in the DAC and that were always there to make their comments and give me some precious hints: Raimir, Rubén, Julián, Johann, David, José Luis, Juán Iván, Jesús, Carlinhos and many others I won't be able to list now...

Thanks to all other friends that were also there for me outside the University, specially those that I met through the Association of Brazilian Students and Researchers in Catalunya (APEC).

Thanks to all other professors at DAC, who were always so kind and to the administration people, who were always in the backstage providing all necessary means...

Finally, thanks to CAPES - Brazil for my scholarship on the last years of my PhD, the European project WIDENS for my scholarship on the first years and the European NoE EuroNGI, the Ministry of Education of Spain and the Generalitat de Catalunya for the indirectly provided grants.

Abstract

Over the last years, Mobile Ad-hoc Networks (MANETs) have captured the attention of the research community. The flexibility and cost savings they provide, due to the fact that no infrastructure is needed to deploy a MANET, is one of the most attractive possibilities of this technology. However, along with the flexibility, lots of problems arise due to the bad quality of transmission media, the scarcity of resources, etc. Since real-time communications will be common in MANETs, there has been an increasing motivation on the introduction of Quality of Service (QoS) in such networks. However, many characteristics of MANETs make QoS provisioning a difficult problem.

In order to avoid congestion, a reservation mechanism that works together with a Connection Admission Control (CAC) seems to be a reasonable solution. However, most of the QoS approaches found in literature for MANETs do not use reservations. One reason for that, is the difficulty on determining the available bandwidth at a node. This is needed to decide whether there are enough resources to accommodate a new connection.

This thesis proposes a simple, yet effective, method for nodes in a CSMA-based MANET to compute their available bandwidth in a distributed way. Based on this value, a QoS reservation mechanism called BRAWN (Bandwidth Reservation over Ad-hoc Networks) is introduced for multirate MANETs, allowing bandwidth allocation on a per flow basis. By multirate we refer to those networks where wireless nodes are able to dynamically switch among several link rates. This allows nodes to select the highest possible transmission rate for exchanging data, independently for each neighbor.

The BRAWN mechanism not only guarantees certain QoS levels, but also naturally distributes the traffic more evenly among network nodes (i.e. load balancing). It works completely on the network layer, so that no modifications on lower layers are required, although some information about the network congestion state could also be taken into account if provided by the MAC (Medium Access Control) layer. The thesis analyzes the applicability of the proposed reservation mechanism over both proactive and reactive routing protocols, and extensions to such protocols are proposed whenever needed in order

to improve their performance on multirate networks.

On mobile scenarios, BRAWN also achieves high QoS provisioning levels by letting the nodes to periodically refresh QoS reservations. This extension of the protocol for mobile nodes is referred as BRAWN-R (BRAWN with Refreshments).

Summarizing, the outstanding features of the reservation mechanism proposed by this thesis are: (i) Multirate, i.e. it allows wireless nodes to choose among different transmission rates, in order to accommodate to different channel conditions. (ii) Targeted to CSMA-based wireless MAC protocols, e.g. 802.11. (iii) Reservation based, allowing the network nodes to pro-actively protect ongoing QoS flows, and applying an effective CAC. (iv) Adaptive to topology changes introduced by the mobility of the nodes, re-routing QoS flows to more efficient paths. (v) Feasible and simple to implement over existing MANET routing protocols (as it is shown by the prototype presented at the end of the study).

List of Publications

International Journals

1. R. Guimarães, Ll. Cerdà, J.M. Barceló, J. García, M. Voorhaen, C. Blondia. Quality of Service through Bandwidth Reservation on Multirate Ad-hoc Wireless Networks. Accepted for publication by *Elsevier Ad-hoc Networks Journal*, 2008. Available online: `http://dx.doi.org/10.1016/j.adhoc.2008.04.002`.

International Conferences and Workshops

1. R. Guimarães, Ll. Cerdà. *Adaptive QoS Reservation Scheme for Ad-hoc Networks*. Wireless Systems and Network Architectures in Next Generation Internet: Third International Workshop of the EURO-NGI NoE, Sitges, Spain, June 2006. Published in Lecture Notes in Computer Science, v. 4396, p. 102-112, May 2007;

2. R. Guimarães, Ll. Cerdà. *Improving reactive routing on wireless multirate ad-hoc network*. 13th European Wireless Conference, Paris, France, April 2007;

3. R. Guimarães, Ll. Cerdà. *Qualidade de serviço em redes ad-hoc através de reserva distribuída de recursos*. XII Seminario Apec, Barcelona, Spain, v.1. p. 325-334., May 2007;

4. R. Guimarães, Ll. Cerdà. *Bandwidth Reservation On Wireless Networks with RTS/CTS Signalling*. 12th European Wireless Conference (EW2006), Athenes, Greece, April 2006;

5. Ll. Cerdà, R. Guimarães, J. Morillo, J.M. Barceló, J. García, A. Perez-Neira, M. Realp, M. Voorhaen, C. Blondia, R. Knopp, N. Nikaein. *QoS Management in WIDENS terminodes*. Invited Paper for the 2nd Workshop on Trends in Radio Resource Management, Barcelona, Spain, November 2005;

6. Ll. Cerdà, M. Voorhaen, R. Guimarães, J.M. Barceló, J. García, J. Morillo, C. Blondia. *A Reservation Scheme Satisfying Bandwidth QoS Constraints for Multirate Ad-Hoc Networks*. IST Mobile and Wireless Communications Summit (IST Summit), Dresden, Germany, June 2005;

7. Ll. Cerdà, M. Voorhaen, R. Guimarães, J.M. Barceló, J. García, C. Blondia. *A Reservation Scheme Satisfying Bandwidth QoS Constraints for Ad-Hoc Networks*. 1st EuroNGI Workshop on Mobility and Wireless, Dagstuhl, Germany, June 2004. Published in Lecture Notes in Computer Science, vol. 3427, p. 176-188, February 2005.

National Conferences

1. R. Guimarães, J. Morillo, Ll. Cerdà, J.M. Barceló, J. García. *Redes Inalámbricas de Emergencia Pública: el Proyecto WIDENS*. XV Jornadas Telecom I+D 2005, Spain, November 2005.

Technical Reports

1. R. Guimarães, Ll. Cerdà. *BRAWN: Bandwidth Reservation over Ad-hoc Wireless Networks*. Technical Report (UPC-DAC-RR-XCSD-2008-3). UPC, Barcelona, Spain, March, 2008;

2. R. Guimarães, Ll. Cerdà. *An Enhanced Bandwidth Reservation Scheme for Ad-hoc Networks*. Technical Report (UPC-DAC-RR-2006-42). UPC, Barcelona, Spain, November 2006;

3. R. Guimarães, Ll. Cerdà. *Bandwidth Reservation On Wireless Networks*. Technical Report (UPC-DAC-RR-XCSD-2005-4). UPC, Barcelona, Spain, June 2005;

4. R. Guimarães, J. Morillo. *QoS on Routing: the WIDENS case*. Technical Report (UPC-DAC-RR-2005-35). UPC, Barcelona, Spain, June 2005;

5. R. Guimarães, J. Morillo, Ll. Cerdà, J.M. Barceló, J. García. *Quality of Service for Mobile Ad-hoc Networks: an Overview*. Technical Report (UPC-DAC-2004-29). UPC, Barcelona, Spain, July 2004;

Chapter 1

Introduction

1.1 Background and Motivation

Over the last years, the telecommunications world has been facing a huge revolution, with the creation of new applications and the fact that they are rapidly spread all over the world. On the other hand, there is the wireless "boom", i.e., the development of mobile wireless devices that are more and more powerful and cheaper at the same time. The convergence between these two realities (new applications, mainly real-time multimedia applications and the wireless world) are the focus of many researches, since it is no longer the future, it is our present, and although it is happening now, we still need to solve many issues in order to achieve the main goal: the so called pervasive computing, or the omnipresence of computers, where virtual applications will be present everywhere and computing infrastructures will be inherent of humans.

Wireless technologies play an important role on such scenario. The flexibility they provide can not be replaced by any other current technology. The possibility for nodes to move, free of cables and all over the world, for them to be connected without being physically plugged is the key to the success of pervasive computing. It is the key to the future.

Moreover, the possibility to communicate to places where cables are not able to reach, or even places where trespassing a cable is financially infeasible. Moreover, the ability to rapidly deploy networks that do not depend on any pre-existent infrastructure may be very useful on disaster areas, for public safety. Fire fighters, doctors, policemen can all communicate to each other and exchange vital information (telemedicine, video surveillance etc) by using this kind of networks. The entertainment industry may also take advantage from this technology, since short duration wireless networks may be deployed

by, for example, a group of friends that wish to play a game.

Many other scenarios that reinforce the importance that wireless networks have nowadays, and that they may have in a near future, can be presented: vehicular networks, sensor networks etc. Due to the huge impact that wireless networks may have on our future, it is not difficult to imagine that these networks should provide a minimum level of quality of service (QoS) for some applications that may run on the top of it, such as multimedia applications (voice over IP, video conference etc).

This work is focused on analyzing the need for QoS on a specific kind of wireless networks, the so-called ad-hoc wireless networks, i.e., wireless networks that need no infrastructure to be deployed, where all nodes that integrate the network collaborate to make communication possible. Our objective here is to introduce the concept of ad-hoc networks, present an overview of proposed solutions for QoS guaranteeing on such networks, discussing their applicability, their advantages and disadvantages, and finally present our proposal and how it differs from the pre-existent solutions.

1.1.1 Ad-hoc Wireless Networks

A Ad-hoc Wireless Network (or wireless ad-hoc network) may be defined as follows:

"A wireless ad-hoc network, also known as IBSS - Independent Basic Service Set, is a computer network in which the communication links are wireless. The network is ad-hoc because each node is willing to forward data for other nodes, and so the determination of which nodes forward data is made dynamically based on the network connectivity. This is in contrast to older network technologies in which some designated nodes, usually with custom hardware and variously known as routers, switches, hubs, and firewalls, perform the task of forwarding the data. Minimal configuration and quick deployment make ad hoc networks suitable for emergency situations like natural or human-induced disasters, military conflicts, emergency medical situations etc." – Wikipedia [8]

Differently from infrastructured wireless networks, where a fixed network access point is responsible for intermediating every communication that takes place in the network, a Ad-hoc Wireless Networks node should somehow dynamically discover to which nodes it is able to communicate directly (its neighbors) and how to reach nodes to which it can not communicate directly (nodes that are not in its transmission range). Nodes in such a network should cooperate in order to allow communication to take place. They should act as hosts and routers at the same time, so that whenever a node is not able to directly reach another one, data flows through intermediate nodes until it reaches the destination.

Specific routing protocols are needed for this to happen, as we shall comment below.

Ad-hoc Wireless Networks are, thus, self-creating, self-organizing and self-administrating networks. A few examples of its applications are:

a) A group of friends may establish a short duration network for exchanging data or playing electronic games;

b) A team of firefighters may deploy a network for communicating to each other on an area that was completely destroyed (where no infrastructure was left);

c) Sensors may be spread by plane over a forest or a farm and they may spontaneously establish a network, so that measurements may be obtained from every sensor;

d) A military unit may deploy such a network in the battlefield, since they are not able to rely on the enemy's telecommunication infrastructure;

e) Space operations, undersea operations etc.

1.1.2 Research Challenges

Along with the flexibility provided by Ad-hoc Wireless Networks, a whole set of new research challenges arise. The possible mobility of nodes, the bad quality of transmission media, the scarcity of resources and many other problems have been capturing the attention of researchers over the last years. We may summarize the main research challenges on Ad-hoc Wireless Networks as follows:

a) Mobility: the possible random mobility of nodes with varying speeds and directions adds complexity to the majority of the common network problems such as addressing, routing and quality of service (QoS) support. Such type of Ad-hoc Wireless Network where nodes move are also known as Mobile Ad-hoc NETworks (MANETs) [62][43];

b) Dynamic Changing Topology: the fact that mobile nodes may move independently from each other makes the network topology to be in constant change. Node failures, poor channel conditions and interferences may also cause topology to be time-varying. A node can experience frequent topology changes during a session [60][64][69];

c) Imprecise State Information: link state information used for QoS support may constantly change due to nodes mobility and channel conditions [25][24];

d) Bandwidth constrains: since channel conditions are very poor when compared to wired networks, congestion can take place very easily on such networks [66] [44];

e) Energy constrains: many nodes that are part of such networks may rely on batteries, if this is the case, saving power is an important issue [32][34];

f) Scalability: solutions should not introduce to much overhead in order to maintain the scalability of the network. Mainly due to the constant change of network topology, the bandwidth and energy constraints, this issue is more challenging on such networks [46][30].

These challenges are responsible for many problems that are still open issues, such as effective routing, effective medium access control (MAC) mechanisms, power management, mobility management and, the focus of this study, QoS support. Moreover, the higher the density of nodes in the network, the more complex the scenario is. Also, nodes movement together with varying channel conditions may cause routing information to quickly become obsolete, causing a necessity of frequent control information exchange. At the same time, Ad-hoc Wireless Networks should provide reliable communication, its availability should be maximized and a minimum degree of QoS should be provided.

Based on these issues, one may notice that any proposed solution for ad-hoc routing and QoS should cope with the following requirements:

a) Low overhead: signaling should not consume too much bandwidth, since resources are scarce on this kind of networks. Protocols should be as lightweight as possible;

b) Adaptive solutions: algorithms should adapt to network conditions, on a intent to maximize routes lifetime when changes in topology, network traffic and radio conditions take place;

c) Robustness: proposals should be robust to network failures, they should overcome the failure of a given node and avoid high congestion on delimited regions.

1.2 Routing on Ad-hoc Wireless Networks

Due to the unpredictable location and possible mobility of mobile nodes, classical routing protocols used on wired networks are not suitable for Ad-hoc Wireless Networks. Some specific routing protocol were thus defined for ad-hoc networks taking into account their particularities. These protocols may be classified as proactive and reactive protocols.

Proactive protocols are characterized by the fact that every mobile node maintains routes to all destinations all the time. In order to do that, nodes periodically exchange topology control messages so that every node has a "complete" (although not always fresh enough) view of the network topology all the time. The Internet Engineering Task Force (IEFT) currently maintains two standard proactive routing protocols, OLSR (Optimized Link State Routing Protocol) [26], and TBRPF (Topology Dissemination Based on Reverse-Path Forwarding) [58].

On the other hand, reactive protocols do not try to have a topology view of the network. Instead, nodes launch a route discovery procedure whenever needed. That means that, only when a route is needed, control signaling is exchanged in order to find it. Although this approach minimizes control messaging overhead, the route discovery procedure takes longer. The reactive protocols that are standardized by IETF are DSR (Dynamic Source Routing) [48] and AODV (Ad hoc On-demand Distance Vector) [56]. Currently, there has been a great effort on trying to standardize a new protocol, DYMO (Dynamic MANET On-demand) [23].

All these protocols have been analyzed and compared in several papers (with exception to DYMO, since it is still very recent). The main conclusion of these comparisons is that none of them is the best for all environments. Depending on several aspects - such as mobility, network load, network diameter, etc - one protocol may behave better than another.

1.3 QoS Approaches

In order to obtain QoS (Quality of Service) on an Ad-hoc Wireless Network, it is not sufficient to provide basic routing functionalities [61] [54]. As discussed before, other aspects should also be taken into account, such as bandwidth constraints, generally due to a shared media, dynamic topology, since nodes are mobile and the topology may change and power consumption due to limited batteries.

For wired networks there are basically two approaches to provide QoS: over-provisioning and network traffic engineering. Over-provisioning consists on offering a huge amount of resources such that the network can accommodate all the demanding applications. Such an approach, although possible to be implemented on a wired environment, is infeasible when wireless links are used, due to the scarcity of resources. On the other hand, network traffic engineering classifies ongoing connections and treats them according to a set of established rules. Two QoS architectures based on traffic engineering have been proposed

by the IETF: Integrated Services (IntServ) and Differentiated Services (DiffServ).

IntServ is based on a reservation-oriented approach where applications request for the QoS parameters they need. The Resource reSerVation Protocol (RSVP) has been proposed by IETF to setup resource reservations for IntServ. Bandwidth and buffer space, for example, may be explicitly reserved for a given application flow, so that delays may be controlled. This approach, however, has not been well accepted by network providers and router vendors, since all routers should implement not only RSVP, but also Call Admission Control (CAC) mechanisms, per-flow forwarding and flow state control. That would make routers even more complex than they already are. Besides this, it would only work if all routers implemented IntServ.

DiffServ, on the other hand, is a reservationless method. It is based on the classification of flows into a limited number of service classes (according to their QoS requirements). Routers are then only required to differentiate among a few service classes (instead of several flows). The IPv4 TOS octet or the IPv6 Traffic Class octet is used to tag a packet as belonging to a particular QoS class.

In general, the specific aspects of wireless networks make these wired-based QoS models not appropriate for Ad-hoc Networks. IntServ/RSVP may require unaffordable storage and processing for mobile nodes, and a great signaling overhead. Diffserv on the other hand, although being a lightweight model, presents an organization in customers and service providers that does not fit the distributed nature of Ad-hoc Wireless Networks. This have motivated numerous QoS proposals specifically targeted to Ad-hoc Wireless Networks.

1.4 Existent QoS Mechanisms for Ad-hoc Wireless Networks

Although many proposals have been published in the last few years, there are still lots of open issues related to QoS provisioning in Ad-hoc Wireless Networks. Different approaches have been proposed trying to enhance the reliability of such networks and, although Ad-hoc Wireless Networks differ from wired networks in many aspects as we could see in our previous discussion, most of these proposals are still inspired on DiffServ or IntServ.

1.4.1 DiffServ Inspired Proposals

Load-Balancing Schemes

The simplest QoS mechanisms that have been proposed for ad-hoc networks can be considered the load-balancing schemes. Two of these proposals are [50] and [17], the former has been proposed for AODV and the latter for OLSR. The basic idea behind these approaches consists on letting the nodes estimate the available bandwidth. This is done by forcing nodes to measure the transmission time of the packets and its activity periods. Using AODV, an additional field is added to the Route Requests packets (RREQ) to propagate the measurements when a new route is searched. This information is taken into account by the destination before sending the Route Reply (RREP) packet, so that several RREQs are received and a RREP is sent only over the less congested path. In the case of OLSR, nodes propagate the available bandwidth together with the topology to the rest of the network. This information is used by the Shortest Path First (SPF) algorithm when searching for a new route.

Courtesy Piggybacking

The Courtesy Piggybacking [53] is a proposal that intends to avoid the bandwidth starvation suffered by low priority traffic on service differentiated systems. In these systems, whenever high priority traffic is intense, the low priority traffic may not be transmitted at all, since it keeps waiting for the "never ending" transmission of high-priority traffic.

The idea of this proposal is to piggyback low priority traffic into the high priority traffic packets whenever there is a free space. This free space may occur when a MAC frame is not completely filled by the high priority data – this may happen when data is fragmented and the last fragment is shorter than the MAC frame or when available high priority data is not enough to fill a MAC frame. Whenever this happens, this unused "free space" may be used to piggyback low priority traffic.

This approach is completely independent of the service differentiation scheme and the routing algorithm used. It is designed in a cross-layer way, so that the MAC layer must have access to network layer information to fill its frame with low priority data.

SWAN

The SWAN Project [68] proposes a feedback-based mechanism to provide soft real-time services and service differentiation on stateless ad-hoc wireless networks. It uses rate control for UDP and TCP best-effort traffic and admission control on the sender for UDP real-time traffic.

Instead of depending on signaling and state information, SWAN uses feedback information from the network. By measuring MAC delays, it automatically configures the rate control mechanism and, by measuring the rate of real-time flows that pass through its neighbors, it evaluates the amount of bandwidth that is still available for new real-time connections, configuring thus the admission control.

Whenever a node suffers from QoS degradation, it marks every forwarded packet with an Explicit Congestion Notification (ECN) flag. The destination of a packet marked with ECN should notify the source of the flow, so that it blocks transmission or adapts it to the new conditions.

SWAN is a simple and effective solution. By avoiding signaling, it simplifies the whole architecture and provides a solution that, although not being able to guarantee the QoS needs of each flow for the whole session, provides a differentiation between real-time and best-effort, prioritizing the former.

CEDAR

Most routing algorithms designed for ad-hoc networks assume that every node behave as edges of the flows (source and destination) and as routers. This means that every node must maintain the state of the network and must exchange this information with every other node. In proactive algorithms, this information is exchanged periodically while in reactive algorithms, it is exchanged on demand.

Trying to avoid all this overhead, the *Core Extraction Distributed Ad hoc Routing* (CEDAR) [67] algorithm proposes the election of a core network that is responsible for all the route computation.

A set of nodes is dynamically elected to form the core of the network, so that each of them maintains the local topology of the nodes that belong to its domain. The core nodes propagate information about bandwidth availability on the stable links of the core network and keep information about dynamic and low-bandwidth links. By doing this, all route computations are restricted to the core nodes.

Whenever a node needs to establish a connection to another one, it contacts the core node of its domain. This core node computes a core path to the destination domain and uses this core path as a directional guideline for the establishment of a short stable admissible QoS route from the source to the destination.

1.4.2 IntServ Inspired Proposals

Not many QoS mechanisms for Ad-hoc Wireless Networks are inspired on the IntServ architecture. In fact, at first sight, the IntServ architecture seems to be very heavy-weighted for being used on ad-hoc networks. And that is exactly why all proposals that fall into this category are, in fact, lightly inspired by IntServ. They are usually based on "soft reservation" of resources and this is done through a simplified control signaling and also by avoiding the need of too much flow state information on each node.

Quality of Service for Ad hoc On-Demand Distance Vector

The idea of resource reservation on Ad-hoc Wireless Networks, although not so explored, has been envisioned by the authors of AODV, for example. In the [55] document, they propose a standard QoS extension for their routing protocol (note that this approach is based on, and is suitable only for, AODV) that should be included in the Route Request (RREQ) and Route Reply (RREP) messages, which are exchanged during the route discovery procedure.

A node will become a hop on the route only if it can meet the requirements specified in the RREQ. If, once the route is already established, a node realizes that the QoS requirements can not be sustained for a certain flow, the node must originate an ICMP QOS_LOST message back to the source.

There are two similar mechanisms for guaranteeing maximum delay and minimum available bandwidth on a path. For guaranteeing delay, every time a node receives a RREQ, it subtracts the NODE_TRAVERSAL_TIME (which is the time required by a node to process the RREQ) from the delay value carried by the RREQ. If the result is negative, the packet is discarded, since the delay requirement can not be accomplished for this route. For guaranteeing bandwidth, the value carried by the RREQ is compared to the available link capacity. If the available link capacity is lower, the packet is discarded. When the destination node replies with a RREP, each node forwarding the RREP compares the bandwidth field in the RREP and its own link capacity and maintains the minimum of the two in the Bandwidth field of the RREP before forwarding the RREP.

Cansever et al

Other studies, such as [19], focus their attention on developing models for computing the available bandwidth on a node by the knowledge of as few information as possible about the on-going traffic on the neighborhood. Such works, although not proposing concrete mechanisms, provide analytical tools for such matter. In fact, as we will further see, our QoS proposal was developed inspired on this work.

The authors look for the formula to estimate the available bandwidth in an ad-hoc network using shared links. To do so, each node should do the following calculation:

$$MUB_i = C_i - \sum_j l_{ij}, \forall j \in \text{Neighborhood of } i \tag{1.1}$$

where MUB_i means the maximum unused bandwidth, C_i is the capacity of the node and l_{ij} is the total traffic between nodes i and j.

But, since the traffic between neighbors of a node also interfere, these traffics must also be taken into consideration to calculate the maximum available bandwidth (MAB_i), what leads us to:

$$MAB_i = MUB_i - \sum_j \sum_k l_{jk}, \forall j \in \text{Neighborhood of } i, \forall k \in \text{Neighborhood of } j \tag{1.2}$$

The MAC protocol must support regulated access to the media and also random access (CSMA-CA) for about 10% of the time. In the random access period, all nodes broadcast their *MUB* and their local bandwidth requests. Now that all nodes are aware of their neighbors traffic demands, a simple algorithm may allocate time slots among the neighbors in proportion to their demands.

When using a reactive routing protocol, such as AODV or DSR, the *MAB* may be used to elect a path that fulfills the QoS needs of a flow. The Route Request (RREQ) messages check the available bandwidth to be sure that the flow may pass through the node (if not, the RREQ is discarded). During the reverse path establishment (Route Reply), the resources may be then reserved.

INSIGNIA

As we said, few are the concrete proposals of mechanisms inspired by IntServ. INSIGNIA [52] is probably the first of them. It consists on an in-band signaling protocol in contrast with out-of-band signaling protocols like RSVP. This means that the QoS signaling information is encapsulated into data packets, making this approach "lightweight". This implies that there are no special packets for doing the signaling. INSIGNIA is just the signaling protocol and a routing protocol, such as DSR, AODV, OLSR or TBRPF, is still needed.

INSIGNIA supports fast flow reservation, restoration and adaptation algorithms that are specifically designed to deliver adaptive real-time service in Ad-hoc Wireless Networks. It encapsulates control signals in an IP option of every data packet which is called INSIGNIA option.

Ad hoc QoS on-demand routing (AQOR)

Another interesting example of IntServ inspired proposals is the Ad-hoc QoS On-demand Routing protocol (AQOR) [72]. It proposes a resource reservation-based routing and signaling algorithm that tries to provide quality of service support, in terms of bandwidth and end-to-end delay.

This scheme, however, provides a superficial analysis of the bandwidth consumed by a connection and the computation of the available bandwidth for the establishment of new connections in a given node. It also does not take into account the multirate capability of current networks

1.4.3 Other proposals

Some other proposals can not be classified as being inspired by neither DiffServ nor IntServ. The Flexible QoS Model for MANETs [70], for example, consists on combining both approaches - IntServ and DiffServ.

Flexible QoS Model for MANETs (FQMM)

FQMM [70] is a QoS model specifically designed for MANETs that combines both IntServ and Diffserv mechanisms. Basically, it proposes a hybrid provisioning scheme that combines the per-flow granularity of IntServ and per-class granularity of DiffServ, and a

relative and adaptive traffic profile to maintain consistent differentiation between traffic types and keep up with the dynamics of the network.

Trying to exploit the best of both approaches, FQMM provides QoS differently according to the traffic priority. Per-flow provisioning is given for high-priority traffic while per-class provisioning is given for other traffic priorities. Classification is made at the source node and QoS provisioning is made on every node along the path.

1.5 Outline

This thesis is organized as follows:

Chapter 2: Before introducing the QoS reservation mechanism, we discuss the use of traditional reactive routing protocols (such as AODV) on Multirate Ad-hoc Wireless Networks. Most of currently used MAC protocols, such as 802.11b [10], 802.11a [9] and 802.11g [12], allow the use of different transmission rates. However, in order to fully use the multirate capabilities on an ad-hoc wireless network, the routing protocol should also be aware of this information. There is no point on being able to transmit at so many different rates on the MAC layer, if at the end, the routing protocol always chooses routes based only on hop count. However, although it seems to be very important taking multirate into account at the routing layer, there are not many publications that deal with this issue. In this chapter we propose an efficient solution for the election of high throughput paths through the use of reactive routing protocols. This proposal was presented in [39].

Chapter 3: After adapting a reactive routing protocol for the multirate environment, in this chapter we introduce our reservation mechanism, which we call Bandwidth Reservation on Ad-hoc Wireless Networks (BRAWN). Our mechanism is based on an end-to-end bandwidth reservation protocol that works together with a Connection Admission Control (CAC) algorithm. Although most of the QoS proposals found in the literature for Ad-hoc Wireless Networks do not use reservations, we consider that this approach may be efficient for avoiding network congestion. One reason for the scarcity of works that deal with resource reservation on Ad-hoc Wireless Networks is the difficulty on determining the available bandwidth at a node. This is needed to decide whether there are enough resources to accommodate a new connection. We thus propose a simple, yet effective, method to compute the available bandwidth at a node in AWNs. We then use this method as a basis for a bandwidth reservation QoS

mechanism. Our proposal not only guarantees certain QoS levels, but also naturally distributes the traffic more evenly among network nodes (i.e. load balancing). It works completely on the network layer, so that no modifications on lower layers are required, although some information about the network congestion state could also be taken into account if provided by the MAC (Medium Access Control) layer. Our mechanism takes into account the multirate capability of wireless networks, i.e., it considers that wireless nodes are able to choose among several modulation schemes, providing different transmission rates, in order to accommodate to different channel conditions. We provide a set of *QoS constraints* that must be satisfied for the ongoing QoS flows to consume an overall bandwidth at any node smaller than or equal to a certain threshold. We applied our reservation scheme to both the *Optimized Link State Routing Protocol* (OLSR) [26] and the *Ad hoc On-demand Distance Vector* (AODV) [56] routing protocols and launched a set of simulations that shows the feasibility of our scheme for guaranteeing the QoS requirements of accepted flows. The proposal was preliminarily presented in [22] and [21] and a more complete view can be seen in [41]. Some variations of the proposed scheme may be also seen in [36] (taking RTS/CTS into account), [37] (using feedback from the MAC layer) and [38] (a enhancement on the available capacity computation).

Chapter 4: In the case that nodes move, topology changes in the network may cause connections that were previously accepted by the CAC not to have their QoS requirements guaranteed after a while. Moreover, even if QoS can still be guaranteed over a given path, topology changes may cause more efficient paths to show up, and being able to use them may optimize the use of network resources. Therefore, in the presence of movement, the QoS mechanism should be made adaptive. This could be achieved, for example, by periodically refreshing reservations, so that the network is constantly re-validating the admission control and searching for better routes for previously established connections. In this chapter we discuss how to introduce this behavior in BRAWN and the performance gains that may be obtained on mobile scenarios.

Chapter 5: A real implementation for the BRAWN mechanism is presented for the Linux operating system. This implementation was briefly described in [42] and [20].

Chapter 6: We present a brief review of the concepts presented by this PhD thesis, providing a final discussion on the proposed solutions, their pros and cons.

Chapter 2

Routing in Multirate Networks

2.1 Introduction

Wireless communications have been spread all over the world during the last years. The majority of the commercially available wireless devices are based on the IEEE 802.11 standards family. Most of them, such as 802.11b [10], 802.11a [9], 802.11g [12] and, more recently, 802.11n [71] allow the use of different transmission rates.

The election of which transmission rate should be used depends on the wireless medium conditions. The worse the channel quality, the stronger the code that should be used and, consequently, the lower the achieved transmission rate. Since channel quality is directly related to distance between nodes, we may say that usually, the closer two nodes are from each other, the higher the transmission rate used between them.

In 802.11a (and also in 802.11g), for example, the set of possible data transmission rates are 6, 9, 12, 18, 24, 36, 48, 54 Mbps while 802.11b supports 1, 2, 5.5 and 11 Mbps. In order to exploit this capability, some Medium Access Control (MAC) mechanisms are required. The Auto Rate Fallback (ARF) [49] protocol was the first to deal with this issue. Implemented on the Lucent WaveLAN-II wireless cards, the sender increases (or decreases) the transmission rate to be used in future transmissions based on the successes (or failures) in the previous ones. Some mechanisms, like the one implemented in the Atheros AR5000 chipset for 802.11a [28], are based on throughput comparison. A small fraction of the data to be transmitted (around 10%) is sent using the next higher rate and the next lower rate than the current one. At the end of a given decision window, the transmission performances over the three rates are compared and the best one is chosen for future transmissions. Finally, other mechanisms, such as the Receiver Based Auto

Rate (RBAR) [45] protocol, are based on Signal to Noise Ratio (SNR) measurements. The receiver measures the quality of the channel when it receives a Request To Send (RTS) message and selects the appropriate rate to be used under these conditions. It then informs the sender the rate to be used for data transmission through the Clear To Send (CTS) message.

However, in order to fully use the multirate capabilities on a wireless ad-hoc network, the routing protocol should also be aware of this information. There is no point on being able to transmit at so many different rates on the MAC layer, if at the end, the routing protocol always chooses routes based only on hop count. Traditional routing protocols, like the Ad-hoc On-demand Distance Vector (AODV) [56] or the Optimized Link State Routing Protocol (OLSR) [26], usually elect this kind of path, where the minimization in the number of hops causes the election of long range links over short range ones. If short range links were elected, although the number of hops would increase, higher transmission rates could be used, and the overall performance of the network could be significantly improved. However, although it seems to be very important to take multirate into account in the routing layer, there are not many publications that deals with this issue.

In proactive routing protocols (like OLSR), the solution for this problem is quite straightforward. Since each node knows the (almost) entire network topology, information about link rates would be enough to choose an efficient path. In [16], the authors propose a routing metric that is able to maximize the achievable throughput on chosen paths. However they only implement it on a proactive protocol and no further comments on how to do so on reactive protocols are made.

Reactive protocols (like AODV) do not have any previous information about the network topology, they choose their routes by flooding the network with Route Request messages trying to reach the destination node. This makes the problem much more complex, as we will discuss in further sections. Providing a simple and yet efficient solution is not trivial.

In this chapter, we propose an efficient solution for the election of high throughput paths through the use of reactive routing protocols. More specifically, we propose that each node keeps track of its 1-hop neighborhood topology, using a proactive approach for choosing the route in the neighborhood, and a reactive approach for choosing the route towards distant nodes. We believe that reactive routing can provide better response to the constant changes in the topology of a mobile ad-hoc network, while monitoring the 1-hop neighborhood may improve routing decisions and should not be a problem even when mobility is not so low. Furthermore, the knowledge of the 1-hop neighborhood may also be useful for other mechanisms that may improve the network overall performance,

such as efficient route repairing and controlled flooding [47][73]. In a previous work [35], we have presented our mechanism and some preliminary analysis. In this chapter, we analyze it deeply, comparing its performance to the proposals presented at [31], [14] and [13], which also deal with multirate reactive routing.

Although we focus our attention to the AODV protocol throughout the chapter, the proposed mechanism can be applied to any reactive routing protocol (as long as it is based on the exchange of Route Request / Route Reply messages). Moreover, although we deal with choosing high throughput paths on multirate networks, our proposal is more general, in the sense that it deals with routing through minimum cost paths, no matter what the cost represents. That means that our proposal could also be used for searching paths that minimize different metrics (link rate, link delay, available bandwidth, link stability, loss probability etc).

The chapter is organized in 5 additional sections. In the next section we discuss the problem of using traditional reactive protocols on multirate ad-hoc networks more deeply and the difficulty of applying a metric for taking link rate into account on these routing protocols. In section 2.3 a brief overview on the already existing solutions for the problem is presented, discussing their pros and cons. In section 2.4 we present our proposal, by making modifications on the reactive routing election process in order to take transmission rates into account. In section 2.5 we show through simulations the overall improvement that can be obtained when using our proposal under different scenarios, when compared to the standard reactive routing mechanism and to other related works. Finally, we present some conclusions in section 2.6.

2.2 Motivation

Wireless ad-hoc networks are usually composed by portable nodes – notebooks, palmtops or even mobile phones. This portability also brings an important issue: mobility. This is a key factor in ad-hoc networks. The mobility of the nodes causes the topology of the network to constantly change. Keeping track of this topology is not an easy task, and may consume too much resources in signaling. Reactive routing protocols were designed for these environments. They are based on the idea that there is no point on trying to have a picture of the entire network topology, since it will be constantly changing. Instead, whenever a node needs a route to a given destination, it initiates a route discovery process on the fly, for finding out a path.

This kind of protocols (which has AODV as its major example) is usually based on flooding

the network with Route Request (RREQ) messages. The source node broadcasts a RREQ with a time-to-live equal to 1, i.e., a broadcast limited to its 1-hop neighborhood. Each RREQ is uniquely identified through a sequence number, so that the first copy of a RREQ received by a node is processed, while duplicate messages are discarded. When a node receives the first copy of a given RREQ, it records the address of the node that sent the message, establishing thus a reverse route. When the first RREQ reaches the desired destination, a Route Reply (RREP) message is generated and sent back to the source node through the recorded reverse path, confirming then a path from the source to the destination.

This kind of protocol is very effective on single-rate networks. It usually minimizes the number of hops of the chosen path. However, on multirate networks, the number of hops is not as important as the throughput that can be obtained on a given path.

In figure 2.1, for example, if node A wants to transmit to node E and a reactive protocol is used to find a path, the elected path would be A-C-E. Node A would broadcast a RREQ, which would be received by B and C. Node B would re-broadcast the RREQ, that would be discarded by C (since it has previously received a copy of this RREQ from A). Node C would broadcast the RREQ and it would reach E (as well as D). Node E would then reply with a RREP, that would cross node C and reach A. The path would then be established.

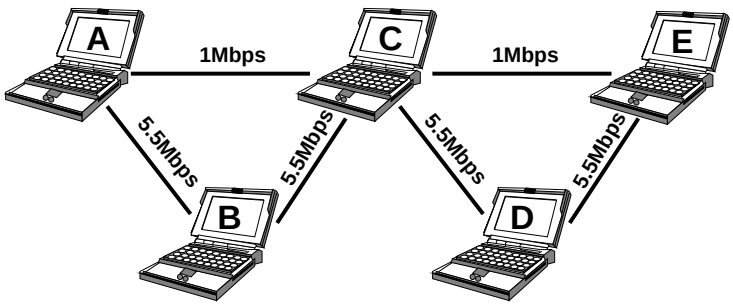


Figure 2.1: An example of a multirate wireless ad-hoc network

It is not very difficult to notice that, in this case, the path A-B-C-D-E, although being longer, would have been a better choice. Data would be transmitted using a 5.5Mbps rate, instead of 1 Mbps. This simple example shows that traditional routing protocols do not cope with the multirate network requirements. We should, therefore, take transmission rate into account when choosing the path towards a given destination, using it as a routing metric.

2.2.1 Taking Link Rate into Account

There are already some proposals that use the transmission rate between wireless nodes as a routing decision metric. The Medium Time Metric (MTM) [16], for example, establishes a link cost for each transmission rate, which is computed through the analysis of how much time it takes to transmit a 1500 bytes packet on 802.11. The link costs for several transmission rates are presented in table 2.1.

Table 2.1: MTM metrics

Transmission rate	MTM link cost
11.0 Mbps	5
5.5 Mbps	7
2.0 Mbps	14
1.0 Mbps	25

The implementation of this metric on a proactive routing protocol is very straightforward. Since each node already knows the topology of the network, it should only add this metric to each link and compute the less costly route towards the desired destination. Nevertheless, on reactive protocols, the problem becomes more complex.

The main issue that arises when dealing with reactive protocols, such as AODV, is the fact that nodes discard duplicate copies of received RREQ messages, so that not every possible path is taken into account in the route discovery procedure. In the example depicted by figure 2.1, the best path from node A to C in terms of throughput would be through B, however, the RREQ sent by B only reaches C after the one sent by A, what causes it to be discarded. Due to this, the maximum throughput path (A-B-C-D-E) is never taken into account, and a less effective route is established through A-C-E.

2.3 Related work

Since the problem presented in the last section resides in the fact that duplicate copies of received RREQ are discarded, a simple solution would be not to discard them. Instead, nodes would accumulate the link cost on each retransmission of the RREQ message and, whenever a duplicate RREQ is received, it would be retransmitted if its accumulated link cost is lower than the cost of all previously received RREQs. If this is the case, the RREQ would be re-broadcasted and the reverse path would be updated. The destination node would not reply the first received RREQ as it is done in traditional reactive routing, but

it would instead wait for a certain period of time, or for a given number of RREQs, and then it would reply the one with the lowest cost.

This solution was proposed by [31] for being applied to AODV and a similar solution was proposed by [14] using the DSR (Dynamic Source Routing) protocol [48] for the MIT Roofnet Mesh Network Project [29]. This idea is also the basis of the on-demand mode of the Hybrid Wireless Mesh Protocol (HWMP), proposed as the default routing protocol for the 802.11s standard [13]. Although these proposals are simple, they heavily increase the number of RREQ messages on the network.

Ad-hoc networks with a dynamic behavior could suffer from performance degradation due to avalanches of RREQs. Notice that this increment in the number of broadcasted RREQs is concentrated in a very short period of time (during the route discovery procedure). The occurrence of these RREQ bursts could significantly increase the number of collisions among copies of the same RREQ. Since broadcast transmissions are not acknowledged in 802.11, many RREQs could be lost and the route discovery procedure could not perform well. Furthermore, as the number of nodes in the network increases, so will the number of duplicate RREQs that are transmitted.

A totally different approach is proposed by [74]. It deals with the multirate issue completely in the Medium Access Control (MAC) layer. The MAC layer hides from upper layers the existence of low throughput links, by selectively filtering received frames. As a result, on the top one could use any standard routing protocol and high throughput paths would always be elected. However, hiding topology information from the routing layer may not always be a good solution. Although solving multirate path election issues, it may significantly degrade the performance of mechanisms such as rapid route repair, or any other schemes which performance is directly related to the amount of topology knowledge that a node has. The authors of [65] propose a similar method where an intermediate layer is created between the network and link layers to deal with multirate. Also in this proposal, the fact that the decision is not taken by the routing layer may have a negative impact on other mechanisms.

Our proposal, which works completely on the network layer, is based on the hybrid routing concept, i.e. it acts proactively when dealing with nodes that are in the neighborhood and reactively when dealing with the rest of network (nodes that are farther away). Some previously proposed routing protocols, such as the Zone Routing Protocol (ZRP) [63], are based on this concept, however none of them focus on solving the multirate routing issues.

2.4 Our proposal

In order to avoid an increase in the number of transmitted RREQs over the network, we propose that every node keeps track not only of their 1-hop neighborhood (what is already done by most of the existing ad-hoc routing protocols through the periodic exchange of HELLO messages), but also of the topology of these neighbors. That means that a node should know the links that exist between its neighbors. Notice that even on highly dynamic networks, this information is not difficult to be maintained, since a node is aware of any change on its 1 hop neighborhood very quickly.

Once a node is aware of the topology of its 1-hop neighborhood, the RREQ/RREP procedure can take place with minor changes. Whenever a node receives and processes a RREQ, it may compute the best path (it terms of throughput) towards the node that sent him the RREQ message, or towards any other node before in the path (if it is more efficient not to pass through the previous node). After computing this part of the path, the complete path information is updated in the RREQ message and it is re-broadcasted. When the first RREQ reaches the destination, a RREP is sent to the source following the path recorded in the request.

2.4.1 Keeping track of the 1-hop neighborhood topology

In order to keep track of the topology of the 1-hop neighborhood, nodes should include a list of their 1-hop neighbors (nodes from which they receive HELLO messages) in the HELLO messages that they periodically broadcast together with the link cost towards each of the neighbors. This link cost is computed based on the link rate (see table 2.1).

By receiving HELLO messages from every neighbor, a node is able not only to have a complete view of the 1-hop neighborhood topology, but also to know its 2-hop neighbors and their connectivity with the 1-hop neighborhood (in order to have a complete 2-hop topology, it would be necessary also to know the links among the 2-hop neighbors). Figure 2.2 shows an example of the topology map that can be built by node A, using the information we propose to be carried by HELLO messages.

2.4.2 The multirate route discovery procedure

Once the nodes know the complete 1-hop topology and a partial 2-hop topology, the route discovery procedure can be modified in order to retrieve not the minimum hop path, but

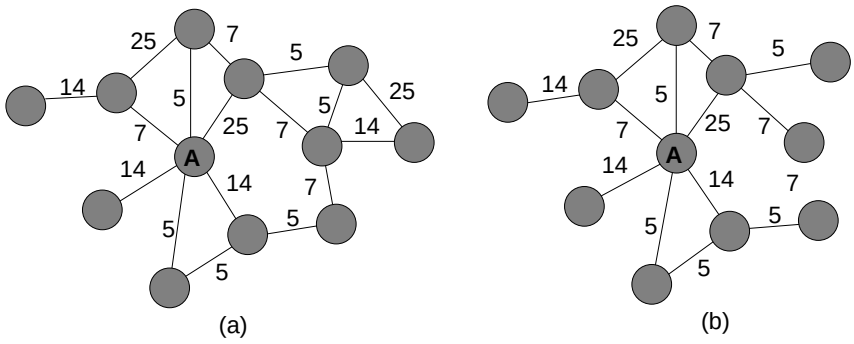


Figure 2.2: (a) Complete ad-hoc network topology with link costs (b) Partial topology known by node A due to HELLO messages

the maximum throughput one.

The first step toward achieving this objective is to extend the RREQ message by introducing a list of nodes and link costs that represents a maximum throughput path from the source node to the node that received the message. Every node that receives the RREQ completes this list using their 1-hop topology knowledge in order to create a complete path from the source to the destination.

The route discovery procedure works as follows:

1. The source node broadcasts a RREQ message to its 1-hop neighbors.

2. Each node that receives the RREQ message computes the maximum throughput (minimum cost) path to the last node through which the RREQ passed.

3. The node includes the maximum throughput path it computed in the RREQ message by introducing the IP address of the nodes between the current node and the previous one together with the link cost to go from one node to another. In figure 2.3(a) for example, when node C receives a RREQ from node A, it computes that the maximum throughput path towards A is passing through node B. So it includes the IP address of A with 5 as the link cost (to reach A from node B), and then the IP address of B with 7 as the link cost (to reach node B from the current node C).

4. Finally, when the RREQ reaches the destination node, it replies with a RREP that should follow the path included in the RREQ, which represents the maximum throughput path from the source to the destination (see figure 2.3(b)).

In the ad-hoc network depicted by figure 2.3, if node S wants to find a route towards node D, it broadcasts a RREQ message that, at each intermediate node, receives a list

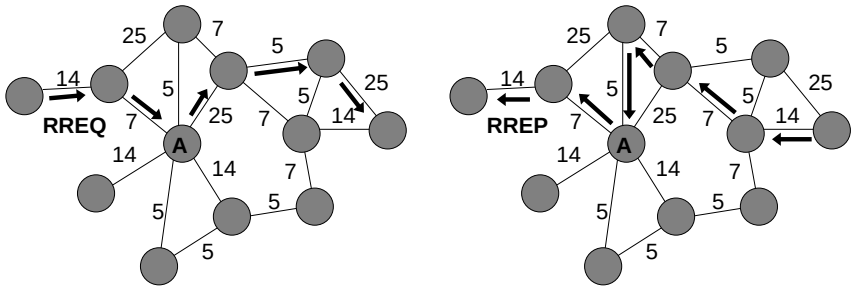


Figure 2.3: (a) Path followed by the 1^{st} RREQ to reach the destination (b) Path followed by the RREP

of nodes that represents a candidate path. In this example, at each intermediate node through which a given RREQ message passes, the list of nodes that it carries is updated as follows. Notice that we are only dealing with the RREQ that first reaches the destination node.

at E: *S 14*

at A: S 14, *E 7*

at C: S 14, E 7, *A 5, B 7*

at K: S 14, E 7, A 5, B 7, *C 5*

at D: S 14, E 7, A 5, B 7, C 7, *J 14*

In fact, an intermediate node may not only insert new nodes in the candidate path included in the RREQ message, but it may also replace existing hops of the path by other hops that it considers more efficient. This may happen whenever the node is able to reach another node that is in the path included in the RREQ message with a lower cost than the one presented in the RREQ. This optimization is only possible due to the fact that the routing protocol is the entity responsible for the multirate routing decisions, having a complete (or almost complete) knowledge of the topology and link costs. If routing decisions like that were left to lower layers (like in [65] or [74]), such optimizations would not be possible.

Notice that the cost towards a given intermediate node can be obtained by summing the links costs from the last node in the list up to the the desired node. In the last example, the cost for node J to reach A is $7 + 7 + 5 = 19$.

Still in the example, when node D receives the RREQ, it checks that the minimum cost for reaching node K (the sender of the RREQ) is 19 (through node J). That value summed

to 5 ($19 + 5 = 24$), would be the cost for reaching C through J and K (that path could be elected by just including nodes J and K in the RREQ). However, since the node has a knowledge of its complete 1-hop neighborhood (and partial 2-hop) it is also able to check that it is less costly to reach node C through J only. In this case, the cost would be of $14 + 7 = 21$. Thus, D decides for the minimum cost path and removes C with a cost of 5 from the RREQ and introduces C with a cost of 7 and J with a cost of 14.

By doing this procedure, we can not guarantee that the minimum cost path is finally elected but, at least, we can guarantee that the chosen path, will perform better than the minimum hop path. In order to guarantee the election of the minimum cost path, we should allow the re-broadcast of RREQs (as proposed by [31], [14] and [13]), however we think that the collateral effect of such solution (high increase on the number of RREQs) is a very high price to pay. As we will show in the simulations, our mechanism provides a better trade-off between performance and overhead.

2.4.3 Using Multi-Point Relays

Since we propose that nodes keep track of their complete 1-hop neighborhood and partial 2-hop neighborhood, it seems to be a good idea to take advantage of this information to improve the mechanism performance even more. Through the use of Multi-Point Relays (MPR), a concept introduced by the OLSR (Optimized Link State Routing) [26] protocol, we are able to significantly reduce the amount of signaling traffic needed to compute the multirate routes.

The basic idea of Multi-Point Relays is to minimize the number of flooding messages (such as RREQs) in the network by avoiding redundant retransmissions in the same region. Each node in the network selects a set of nodes in its 1-hop neighborhood as its MPRs. These selected nodes, and only them, are responsible for retransmitting its broadcast messages. Nodes in the 1-hop neighborhood which are not MPRs, should receive and process every broadcast message from the node, but should not retransmit them. If we correctly choose the MPR set so that every node in the 2-hop neighborhood may be reached through at least one of the MPRs, every node in the network may be reached using less broadcast messages (see example in figure 2.4).

In our mechanism, each node uses its MPR set using the election algorithm proposed by [26]. By using this technique, we may significantly reduce the number of RREQs in the network, avoiding even more the probability of collisions among copies of the same RREQ and, therefore, enhancing the overall performance of the system.

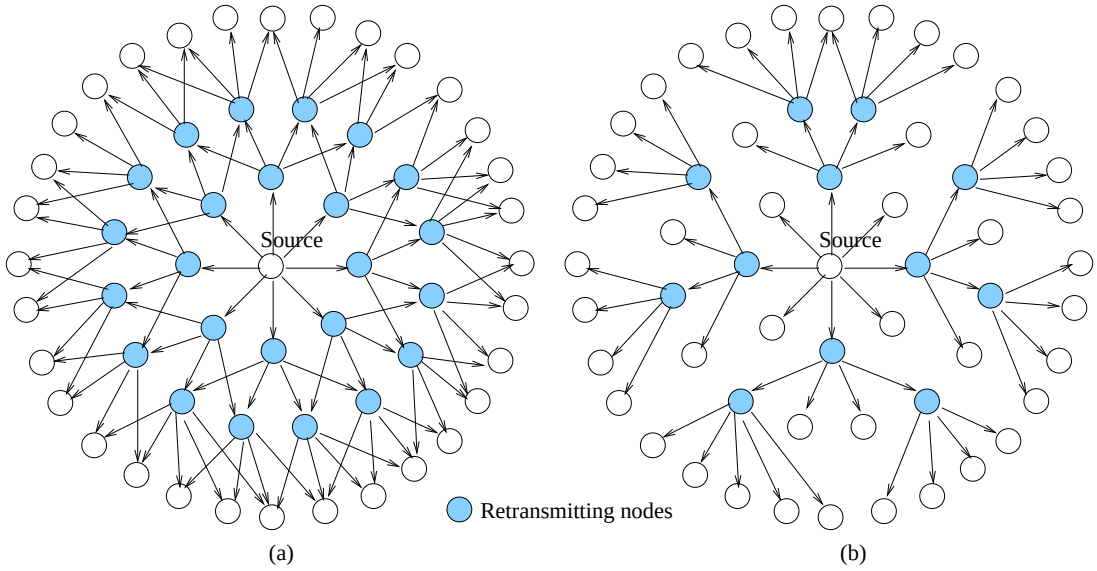


Figure 2.4: (a) 24 retransmission are needed to reach 3 hops while (b) using MPRs, only 12 are needed [57]

2.5 Simulation Results

We have modified the AODV implementation provided by the network simulator ns-2 [6] version 2.30 and launched several simulations for validating our proposal and checking its overall performance. We compared the results obtained with our proposal (which we call Multirate AODV, or MR-AODV), the standard AODV implementation and a modified version of AODV that re-broadcasts RREQs whose accumulated link costs are lower than the previous copies of the same RREQ (as proposed by [31], [14] and [13]). We refer to this last strategy as Rebroadcast. For all these simulations, we used the parameters listed in table 2.2.

Table 2.2: Parameters used in simulations

Parameter	Value
MAC Protocol	802.11 with multirate
Propagation Model	Two Ray Ground
Transmission Rates	1, 2, 5.5 and 11 Mbps
Transmission Ranges	115, 90, 70 and 50 meters[1]
Carrier Sensing Range	200 meters
Simulation Time	500 seconds
Simulation Area	Square of 500×500 meters

[1]According to the ORiNOCO 802.11b PC card specification for a semi-open environment[1].

The first simulation used a simple topology composed of 25, 36, 49, 64, 81 and 100 nodes disposed in a regular matrix configuration (5x5, 6x6, 7x7, 8x8, 9x9 and 10x10 respectively). Then, an FTP connection was established between the node in upper left corner and the one in the bottom right. In figure 2.5 it is possible to see the topologies that were simulated. In all of them, we represented the transmission ranges using each available transmission rate.

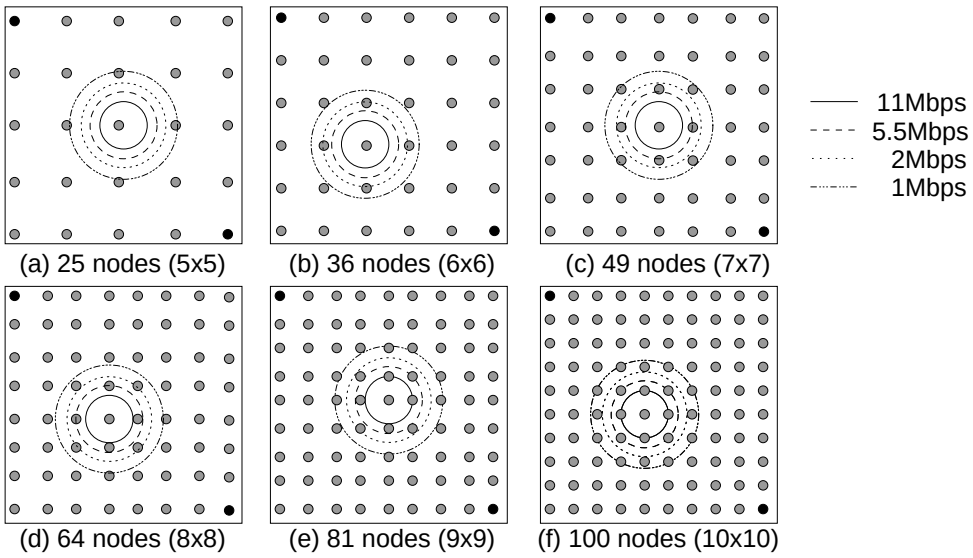


Figure 2.5: Network topologies used in the first simulation. The black nodes are the source and destination of the FTP connection.

Figure 2.6 shows the throughput obtained when using our proposal with and without the use of MPRs, the standard AODV and the Rebroadcast solution. As we may see, up to the 49-nodes scenario, when the density of nodes is low, the difference between the four mechanisms is almost inexistent.

Figures 2.7 to 2.10 show the amount of packets transmitted using each transmission rate in the 6 simulated scenarios, when using AODV, Rebroadcast and MR-AODV without and with MPRs respectively. These graphs help us understand the throughput comparison provided by figure 2.6. In the first scenario (25 nodes), a node can only reach its neighbors by using the lowest transmission range (1Mbps), the impossibility of using another transmission rate causes that the use of any multirate mechanism does not improve the performance of the network. The same happens in the second scenario (36 nodes), when a node can reach all its neighbors using 2Mbps.

In the third scenario, half of the neighbors can only be reached by using 1Mbps (neighbors in the diagonals) while the other half can also be reached by using 2Mbps (neighbors above, below, in the right and in the left). Although at first sight, it would be logical that AODV would choose the path that goes directly through the diagonal (the lowest

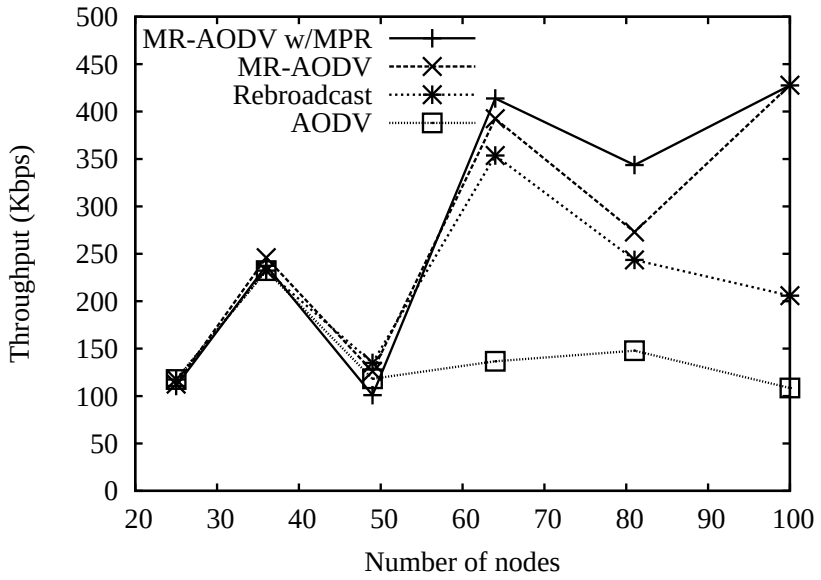


Figure 2.6: Throughput against the number of nodes

hop path), we may see in figure 2.7 that AODV chooses paths that contain about 2/3 of 1Mbps links (diagonal links) and 1/3 of 2Mbps links (horizontal/vertical links). With both Rebroadcast and MR-AODV with MPRs, only 1Mbps links are chosen. The explanation for such behavior in both mechanisms is similar. As we may see in table 2.1, two hops at 2Mbps is more costly than one hop at 1Mbps ($2 \times 14 > 25$). That makes the path composed of only diagonal links not only the shortest path but also the costless one (figure 2.11).

When using MR-AODV without MPRs, however, half of the links that compose the chosen path are 1Mbps and half are 2Mbps. As we have already commented our mechanism not always chooses the best path since, like AODV, once a node receives a RREQ, it discards its subsequent copies. In this case, the RREQ forwarded by node A is not the first RREQ received by node B, so it is discarded. That makes the link A-C (2 Mbps) to be chosen as part of the path (instead of A-B, that uses 1Mbps). The same happens with nodes D, E and F. Notice that, when using MPRs, the number of RREQs in the network decreases significantly (figure 2.12), reducing the probability of not choosing the best route.

In fact, looking at figures 2.5(c), (d), (e) and (f), we can see that using 1Mbps links, a node may achieve neighbors that are farther away, what results in paths with a lower number of hops. That is what should happen with AODV in these scenarios. However, due to the just described AODV behavior, some links with higher rates (and also lower ranges) are chosen, causing some packets to be sent using these higher rates – e.g., 2Mbps in the scenario with 49 nodes and 5.5 Mbps in the scenario with 64 nodes (notice that

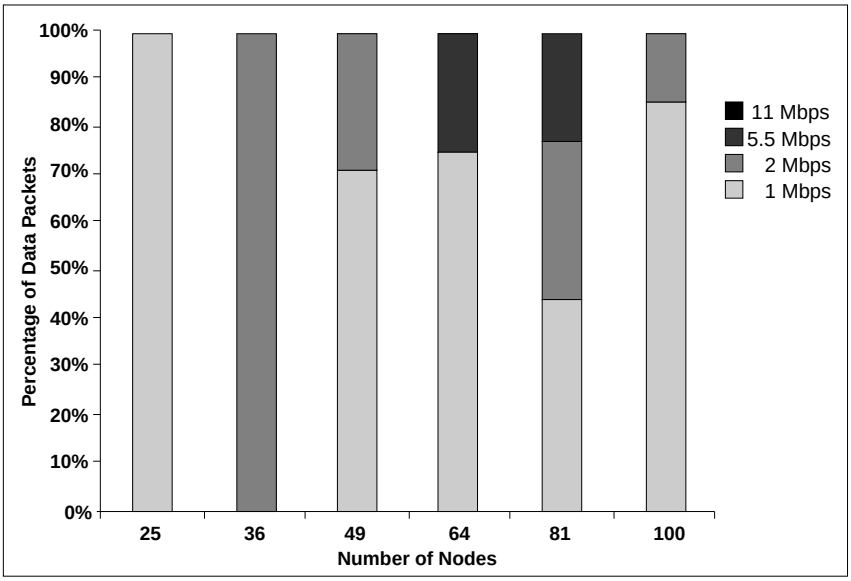


Figure 2.7: Percentage of data packets transmitted with each transmission rate against the number of nodes when using AODV

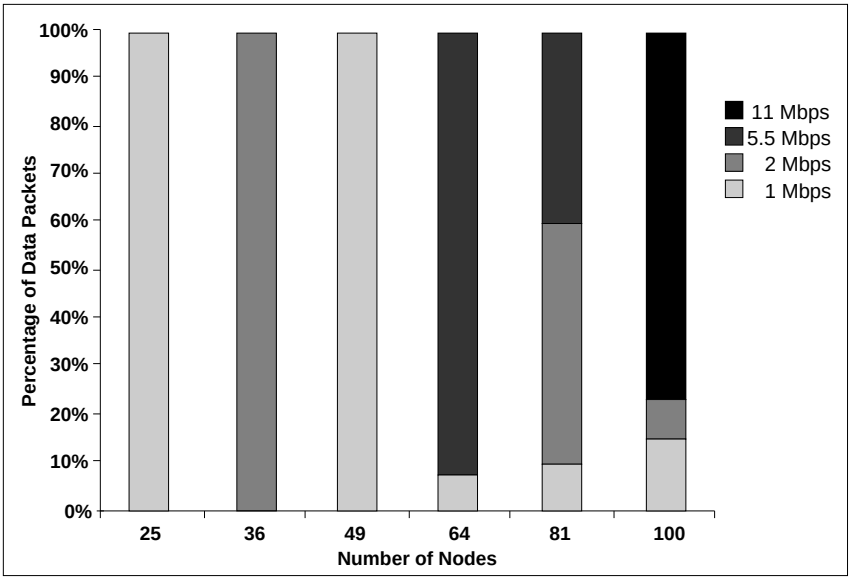


Figure 2.8: Percentage of data packets transmitted with each transmission rate against the number of nodes when using Rebroadcast

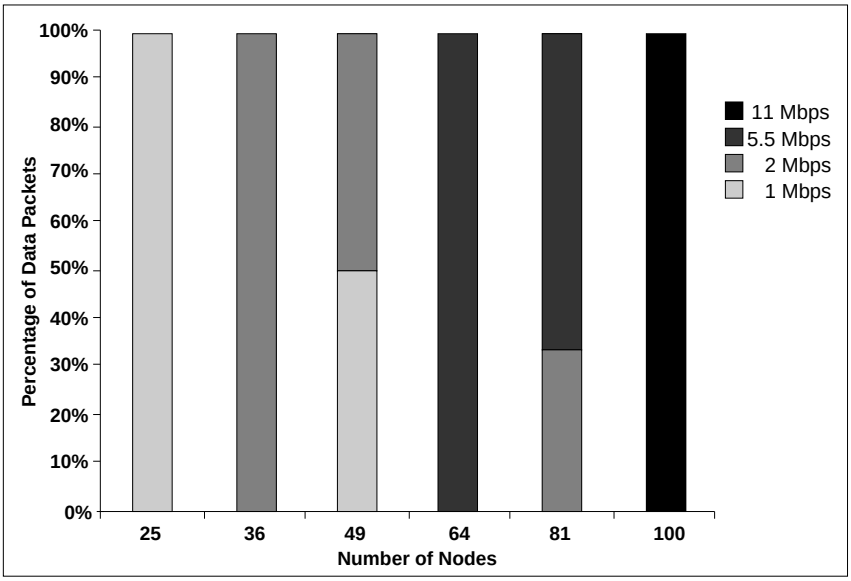


Figure 2.9: Percentage of data packets transmitted with each transmission rate against the number of nodes when using MR-AODV without MPRs

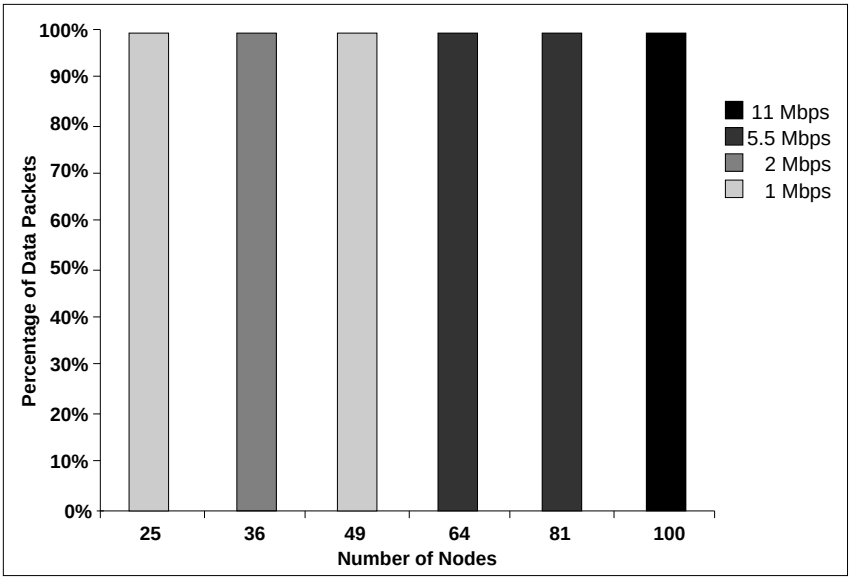


Figure 2.10: Percentage of data packets transmitted with each transmission rate against the number of nodes when using MR-AODV with MPRs

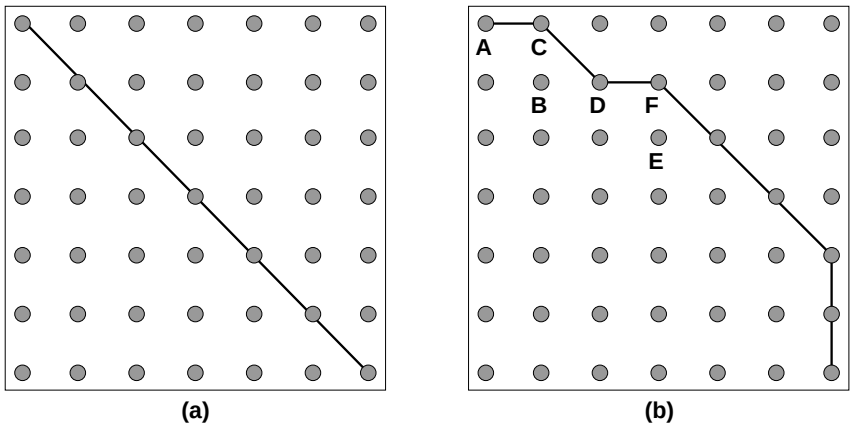


Figure 2.11: (a) Shortest and also minimum cost path and (b) MR-AODV path for the 3rd scenario (49 nodes)

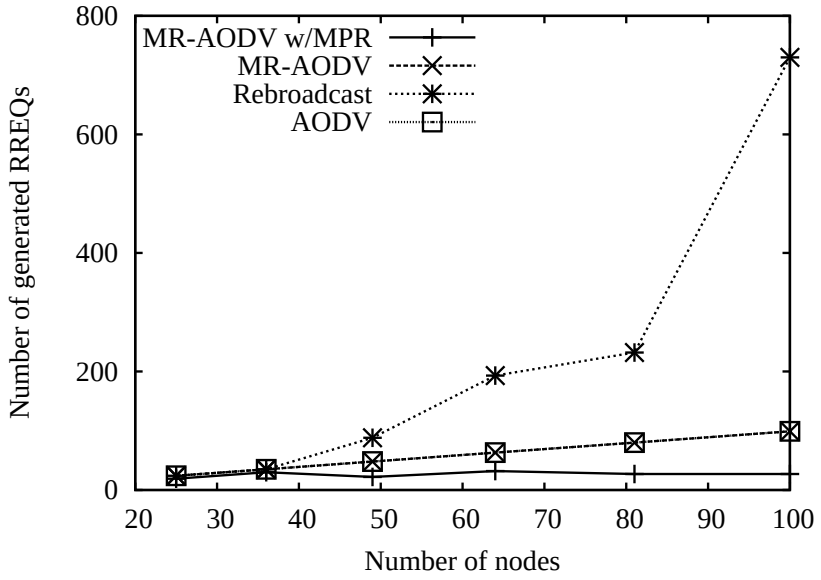


Figure 2.12: Number of transmitted RREQs against the number of nodes

in the latter, all neighbors that can be reached using 2Mbps can also be reached using 5.5Mbps).

When using MR-AODV, however, high rate links are preferred in most of the times. That makes it choose 5.5Mbps links in the scenario with 64 nodes (notice that in these scenarios, depicted by figures 2.5(d) and (e), a node can not reach any neighbor using 11Mbps). In the scenario with 81 nodes, some links that compose the chosen path use 5.5Mbps while others use 2Mbps when not using MPRs and only 5.5Mbps links are chosen when using MPRs. That happens due to the fact that, according to the used metric (table 2.1), two hops at 5.5Mbps have exactly the same cost as one hop at 2Mbps ($2 \times 7 = 14$). Finally, in the scenario with 100 nodes, the higher rate links are always chosen.

Notice that the same did not happen in the Rebroadcast mechanism. When using this solution, the number of RREQs in the network rapidly increases, what causes avalanches of RREQs and many collisions. Consequently, not always the higher throughput routes are chosen.

In a second simulation, we have fixed the number of nodes in 80 and randomly positioned them inside a 500m $\times$ 500m region. We have then varied the number of simultaneous FTP connections from 5 up to 25. For each number of FTP connections, we have launched 50 simulations with different random node positions.

Figures 2.13, 2.14 and 2.15 show the average throughput gain (and the respective standard deviation) obtained by using MR-AODV with MPRs, without MPRs and Rebroadcast respectively when compared to the standard AODV. As we may see, the average throughput gain we obtained when using our proposal stands between 20% and 50% in all scenarios (between 30% and 50% when using MPRs), while the Rebroadcast mechanism had a maximum gain of about 20% and decreased its performance as the number of FTP connections increased. As expected, in many situations, using the Rebroadcast mechanism lead us to a performance decrease in respect to AODV, since the avalanche of RREQs prevented us from finding better paths.

In a few particular cases (22 out of the 250 launched scenarios when using MR-AODV with MPRs and 45 out of the 250 scenarios when not using MPRs) we had a performance decrease when compared to AODV (of 0.14%, 0.57%, 1.10%, 1.17%, 1.38%, 2.03%, 2.50%, 2.69%, 2.84%, 3.09%, 3.50%, 7.03%, 7.19%, 7.38%, 8.90%, 9.05%, 9.06%, 9.62%, 9.99%, 10.41%, 11.50% and 15.27% when using MR-AODV with MPRs). We may see that due to this, the standard deviation interval crosses the unity line once when using MPRs and twice when not using them. We may also notice that, when using the Rebroadcast mechanism, the standard deviation interval always crosses the unity line, since we have plenty of scenarios with a performance decrease when compared to AODV. In most of these scenarios, this mechanism generated too many RREQs so that, instead of helping us find better routes, it caused too many collisions and decreased the overall performance of the network.

If we check the worst case when using MPRs (throughput decrease of 15.27% on a 5 FTP connections scenario), we may see that the path chosen by MR-AODV for all 5 established FTP connections is less costly than the ones chosen by standard AODV (see tables 2.3 and 2.4.) We may also see that just one connection had a throughput decrease when MR-AODV was used (in bold in table 2.4), while the other four had their throughput increased.

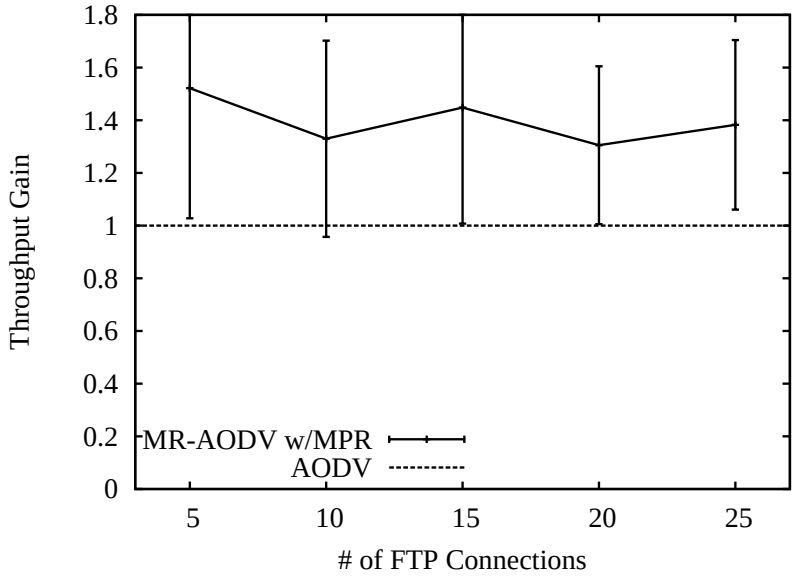


Figure 2.13: Average throughput gain of MR-AODV with MPRs against the number of FTP connections

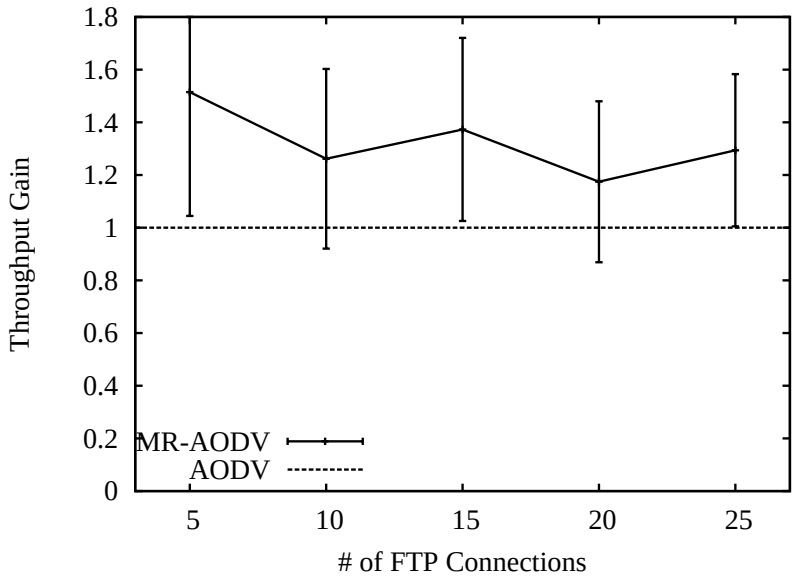


Figure 2.14: Average throughput gain of MR-AODV without MPRs against the number of FTP connections

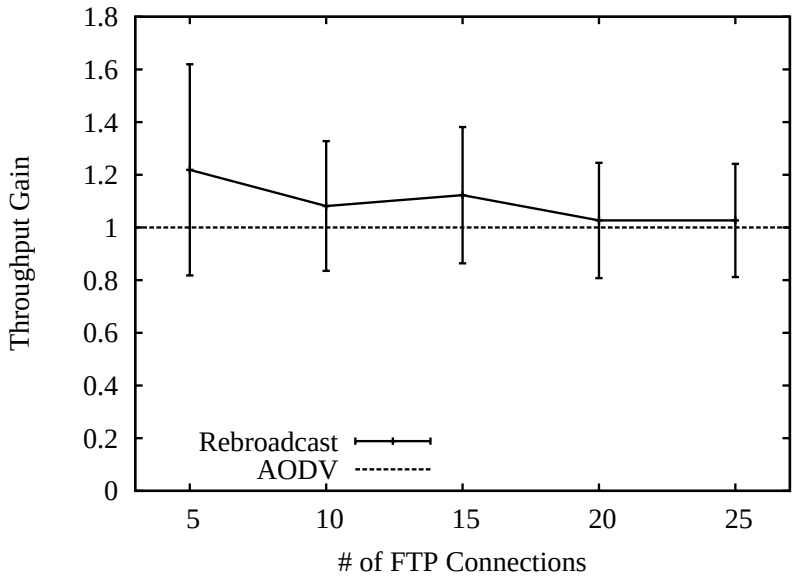


Figure 2.15: Average throughput gain of Rebroadcast against the number of FTP connections

Table 2.3: Paths chosen by AODV in one of the scenarios with 5 FTP connections

Path	Cost	Throughput
$0 \rightarrow 18 \rightarrow 79$	50	369.25Kbps
$1 \rightarrow 78$	25	742.07Kbps
$2 \rightarrow 54 \rightarrow 40 \rightarrow 77$	44	4.48Kbps
$3 \rightarrow 76$	7	3.23Mbps
$4 \rightarrow 75$	5	56.89Kbps

Figures 2.16 and 2.17 show the network topology for this worst case scenario and the paths that were chosen for the 5 FTP connections by AODV and MR-AODV with MPRs respectively. Quickly analyzing these figures, it is possible to see that node 54 is an intermediate of 2 FTP connections when using MR-AODV and of just one when using AODV. This increases the amount of traffic transmitted by node 54, which is in the carrier sensing range (200 meters, see table 2.2) of nodes 3 and 76. As a result, these 2 FTP connections cause interference with the FTP connection between 3 and 76, lowing down its throughput when using MR-AODV with MPRs.

The choice of a path that, in this case, decreases the throughput of another connection happened due to the fact that the used metric – the MTM metric – does not take into account the current state of the network. The MTM metric considers that an 11Mbps link is always better that a 5.5Mbps, for example. However, that is not always true. A link with a lower throughput may be a better choice if it is less congested. That suggests

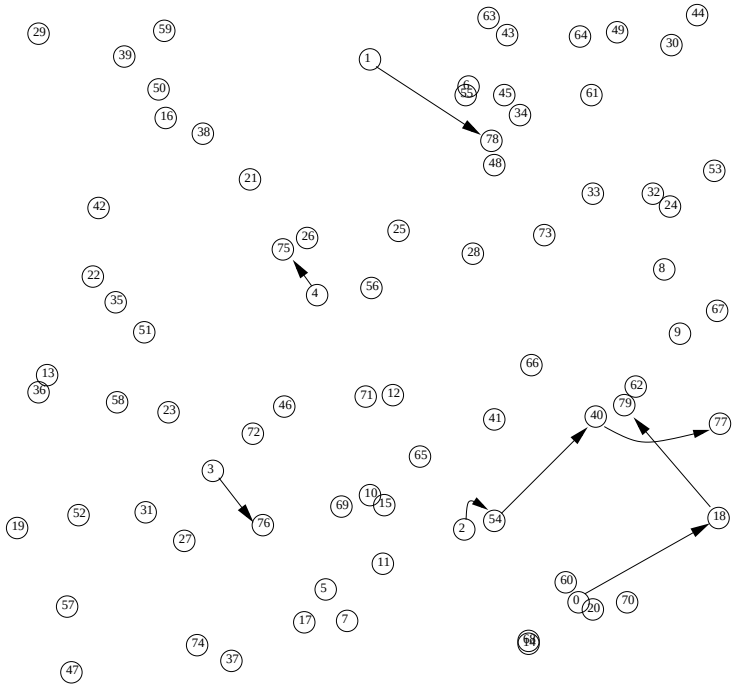


Figure 2.16: Topology of the worst case scenario showing paths established by AODV for the 5 FTP connections

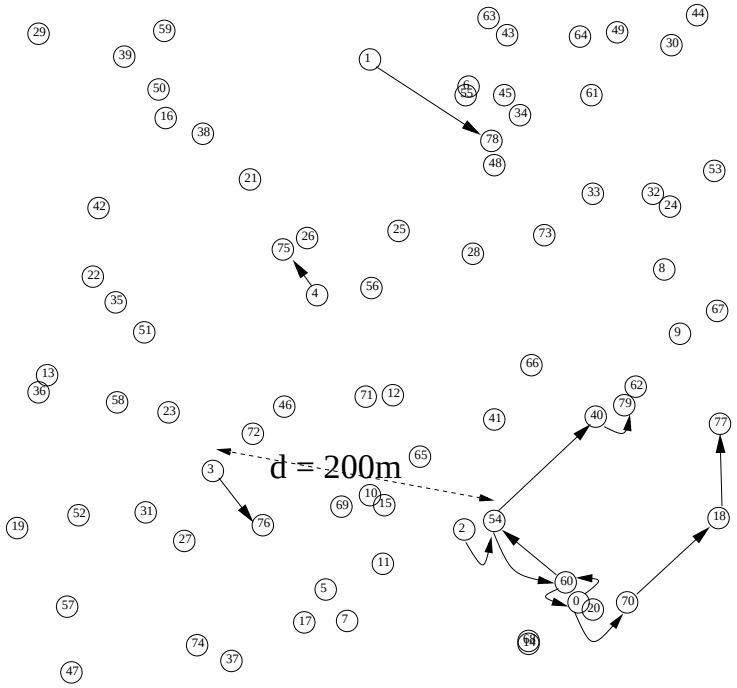


Figure 2.17: Topology of the worst case scenario showing paths established by MR-AODV with MPRs for the 5 FTP connections

Table 2.4: Paths chosen by MR-AODV with MPRs in one of the scenarios with 5 FTP connections

Path	Cost	Throughput
$0 \rightarrow 60 \rightarrow 54 \rightarrow 40 \rightarrow 79$	42	405.08Kbps
$1 \rightarrow 78$	25	1.05Mbps
$2 \rightarrow 54 \rightarrow 60 \rightarrow 0 \rightarrow 70 \rightarrow 18 \rightarrow 77$	43	60.30Kbps
$3 \rightarrow 76$	**7**	**1.84Mbps**
$4 \rightarrow 75$	5	389.42Kbps

that a metric that takes not only link rate but also network congestion into account would better distribute the load and, consequently, improve the overall network performance.

Figures 2.18 to 2.21 show the percentage of packets sent using each of the available transmission rates. We may see that, when using MR-AODV (with and without MPRs), a greater percentage of data packets is transmitted using higher transmission rates, what lead to a performance increase in the majority of the cases, as shown by figures 2.13 and 2.14. More than 85% of all transmitted data packets used links with the two greater transmission rates with MR-AODV (both with and without MPRs), while with Rebroadcast about 70% of the packets used these rates and with AODV this percentage was of only about 55%.

2.6 Final Remarks

In this chapter we have proposed modifications in the behavior of traditional reactive protocols in order to better work on multirate wireless ad-hoc networks. Using our proposal, reactive protocols are able to use the transmission rate as a routing metric. By doing that we were able to elect high throughput paths without significantly increase the signaling.

We have conducted some simulations that show the effectiveness of our proposal when applied to a particular reactive routing protocol (AODV). Through these simulations we could see that our proposal outperforms both the traditional routing protocols and previous proposals that were based on the re-broadcast of RREQs, by choosing paths that significantly increase the overall throughput of data packets.

Finally, we may stress that although we used transmission rates as the routing metric for MR-AODV, our mechanism could also work with any other metric, such as mean delay, link stability or available bandwidth. In fact, the simulation results for some specific

cases suggested that a metric that takes into account not only link rates but also network congestion could improve the overall performance even more.

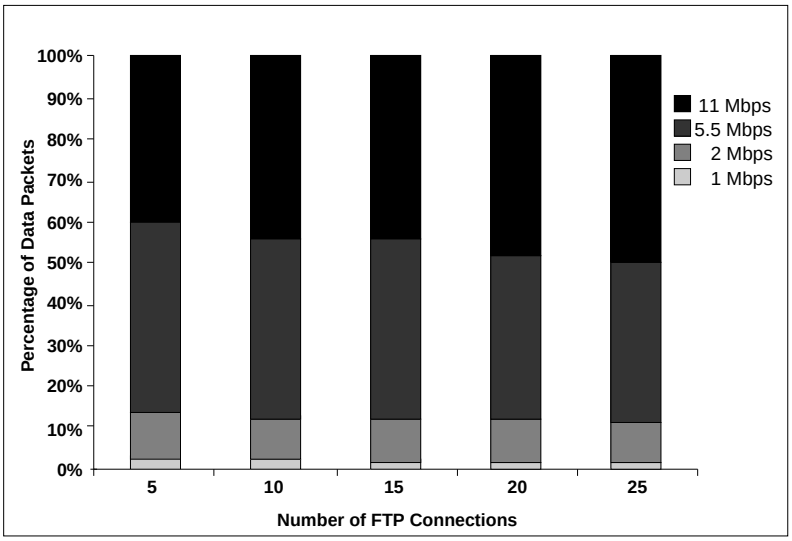


Figure 2.18: Percentage of data packets transmitted with each transmission rate against the number of nodes when using MR-AODV with MPRs

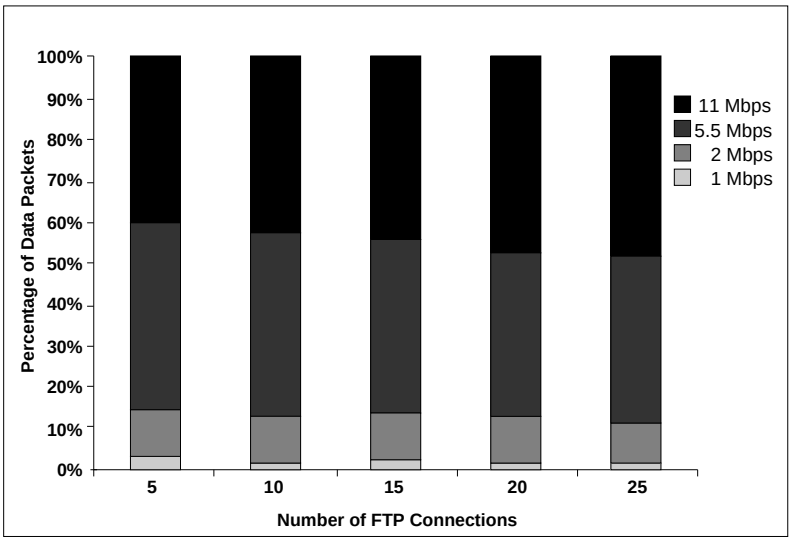


Figure 2.19: Percentage of data packets transmitted with each transmission rate against the number of nodes when using MR-AODV without MPRs

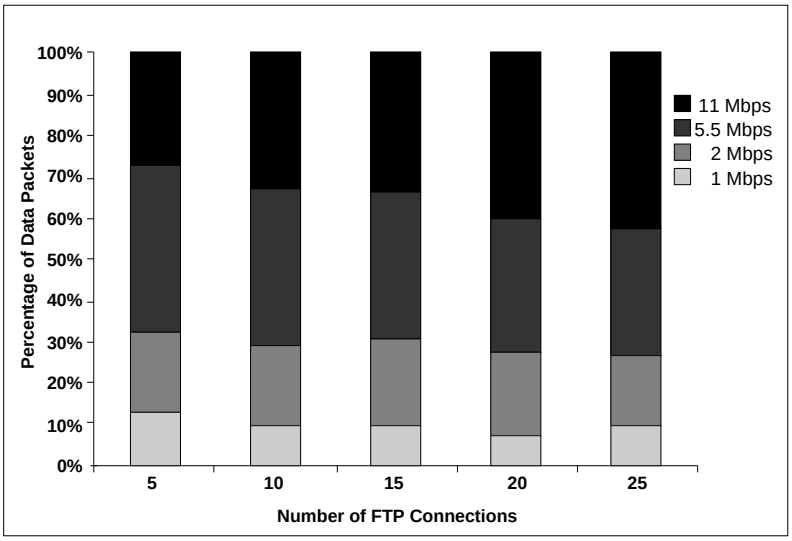


Figure 2.20: Percentage of data packets transmitted with each transmission rate against the number of nodes when using Rebroadcast

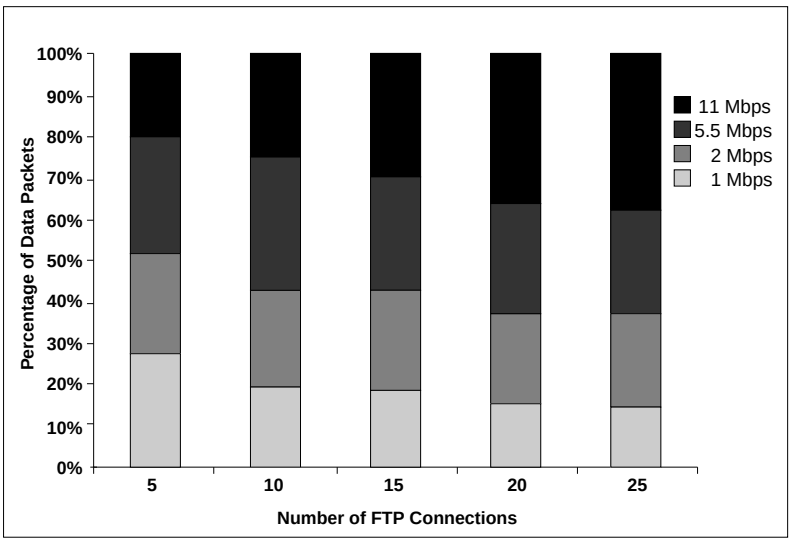


Figure 2.21: Percentage of data packets transmitted with each transmission rate against the number of nodes when using AODV

Chapter 3

A Reservation-based Approach

3.1 Introduction

Over the last years, Ad-hoc Wireless Networks (AWNs), have captured the attention of the research community. The flexibility and cost savings they provide, due to the fact that no infrastructure is needed to deploy a AWN, is one of the most attractive possibilities of this technology. However, along with the flexibility, lots of problems arise due to the bad quality of transmission media, the scarcity of resources, etc.

Since real-time communications will be common in AWNs, there has been an increasing motivation on the introduction of Quality of Service (QoS) in such networks. However, many characteristics of AWNs make QoS provisioning a difficult problem.

Due to the shared media and multihop characteristics of AWNs, it is known that its capacity can be surprisingly low [44]. Consequently, congestion may easily occur, provoking losses and high end-to-end delays. In order to avoid congestion, a reservation mechanism that works together with a Connection Admission Control (CAC) seems to be a reasonable solution. However, most of the QoS approaches found in literature for AWNs do not use reservations. One reason for that, is the difficulty on determining the available bandwidth at a node. This is needed to decide whether there are enough resources to accommodate a new connection.

In this chapter we propose a simple, yet effective method to compute the available bandwidth at a node in AWNs. We use this method to propose a reservation based QoS mechanisms. Our proposal not only guarantees certain QoS levels, but also naturally distributes the traffic more evenly among network nodes (i.e. load balancing). It works completely on the network layer, so that no modifications on lower layers are required,

although some information about the network congestion state could also be taken into account if provided by the MAC (Medium Access Control) layer.

Our mechanism takes into account the multirate capability of wireless networks, i.e., it considers that wireless nodes are able to choose among several modulation schemes, providing different transmission rates, in order to accommodate to different channel conditions. We provide a set of *QoS constraints* that must be satisfied for the ongoing QoS flows to consume an overall bandwidth at any node smaller than or equal to a certain threshold. Along the chapter we shall refer to this threshold as Q. It may be understood as the percentage of time that the channel can be busy at any given node, because it is transmitting, receiving or listening to traffic that belongs to QoS flows. We propose a set of CAC rules that, upon the assumptions listed in the following subsection, can satisfy the *QoS constraints*.

Finally, we apply our reservation scheme to the *Optimized Link State Routing Protocol* (OLSR) [26] although it could be applied to other ad-hoc routing protocols as well (see [36] for a reference on how to apply such a mechanism to the *Ad-hoc On-demand Distance Vector* routing protocol - AODV [56]). In [22] and [21] we have presented preliminary studies of the protocol. The results show the feasibility of our scheme for guaranteeing the QoS requirements of accepted flows.

3.1.1 Our proposal

We treat the problem of achieving end-to-end bandwidth reservation. Our mechanism, which we call BRAWN (Bandwidth Reservation over Ad-hoc Wireless Networks), is based on the computation of the available bandwidth seen by a given node and the use of this value to verify whether new flows can still be routed through this node.

Our scheme is based on the following assumptions: (i) QoS-aware applications are able to request the appropriate bandwidth when establishing a connection. (ii) The nodes know the capacity of the wireless links that is available for QoS flows. Besides this, we assume that the MAC used is able to isolate traffic classes, in such a way that QoS traffic has priority over non-QoS traffic (we could, for instance, use 802.11e). This allows nodes to fix the previously introduced Q threshold. (iii) A pure Carrier Sensing Medium Access (CSMA) protocol is used. Thus, whenever a node is transmitting, all its neighbors will remain silent. Through the chapter we shall refer as neighbors each pair of nodes that are in the receiving range of each other. Note that we are not considering a MAC using RTS/CTS, although it could be easily supported, as we proposed in [36]. (iv) Nodes are able to reach all their neighbors through broadcast packets.

Of course, the previously described assumptions are not exact in real wireless networks. For instance, the available capacity for QoS traffic may be influenced by non-QoS traffic and other network conditions. To cope with that, a conservative value shall be used for Q, or it may be made adaptive, as we proposed in [40]. Furthermore, changes in the network conditions, which can be very frequent in AWNs, make the information used by nodes to compute the available bandwidth to be uncertain. Therefore, after a flow is accepted, its QoS parameters (end-to-end delay, packet loss, etc.) should be constantly monitored in order to react to congestion. This could be done by re-routing or even dropping some of the involved flows. We will not deal with these issues, in order to keep the chapter focused on the reservation mechanism.

Note that a reservation mechanism approach is more appropriate for wireless ad-hoc networks with fixed nodes (e.g. wireless mesh networks [15]) or where mobility is not very high (e.g. pedestrian networks). If nodes constantly move with high speeds (vehicular networks, for instance), changes on the topology are very frequent, thus, the reserved path should be constantly updated. For this reason we use the term AWN (Ad-hoc Wireless Networks) and not MANET (Mobile Ad-hoc NETworks). MANET is commonly used in literature to remark the mobility characteristic of the AWN under consideration.

3.2 How much bandwidth is available for reservations?

The BRAWN mechanism is based on the computation of the available bandwidth (AB) by each node in the network in a distributed way. By knowing its available bandwidth, a node is able to accept or reject a new reservation. So, the first step we should take in order to define our mechanism is to compute the AB of each node.

If we want to compute the amount of bandwidth that is available for a given node to use for new reservations, we should first investigate the amount of bandwidth that is already being consumed by active flows. By knowing this value, we may just subtract it from the total bandwidth dedicated to QoS traffic in order to obtain the currently available bandwidth.

The first issue that we should notice is that a transmission between two nodes does not consume bandwidth only from these nodes but also from the whole neighborhood, since no other neighbor is able transmit at the same time (at least using the same channel) in order to avoid collisions. In fact, the exact knowledge of which nodes suffer the interference

of a given transmission depends directly on the MAC protocol that is being used. For this reason, we assume the use of a Carrier Sensing based (CSMA-like) protocol for the analysis that we will present throughout the chapter.

In order to know how much bandwidth is available for a node to use, we must take into account all transmissions that directly affect its opportunities to transmit. In the case of a CSMA-based wireless MAC protocol, the bandwidth of a node is consumed whenever:

case 1) It transmits data to a neighbor;

case 2) One of its neighbors is transmitting data (if the node senses that the medium is being used, it remains in silence);

Representing this in an analytical way, we may state that the load impact of all transmissions on a node i is given by:

$$l_i = \left| \underbrace{x_i}_{Case\ 1} \cup \underbrace{\bigcup_{j \in \mathcal{N}_i} x_j}_{Case\ 2} \right| = \left| \bigcup_{j \in \mathcal{N}_i^+} x_j \right| \tag{3.1}$$

where:

- l_i is the load impact of all transmissions (in bps) on node i;
- x_i is the total traffic (in bps) that node i wants to transmit (either if node i is the source of the traffic or if it is just forwarding);
- $\mathcal{N}_i$ is the set of neighbors of node i;
- $\mathcal{N}_i^+$ is the set of neighbors of node i and node i itself;
- The union operator $\cup$ represents a "time-based union", i.e., intersections represent parts of the transmissions that takes place simultaneously. See figure 3.1 for an example of the appliance of this operator over two transmissions that overlap in time.

The formula derived before can be generalized for wireless multirate networks, i.e., networks where nodes can communicate to each other at different transmission rates, depending on the wireless medium conditions. To do so, all these values that were represented in bps above must be normalized, dividing them by the transmission rate used:

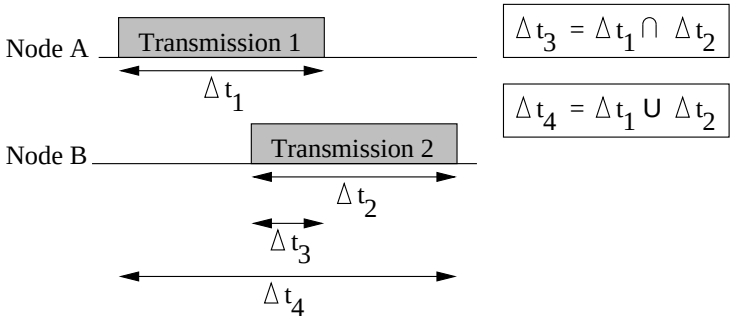


Figure 3.1: Example of "time-based" union and intersection operators

$$L_i = \left| \bigcup_{j \in \mathcal{N}_i^+, \forall k} x_{jk}/v_{jk} \right| \tag{3.2}$$

where:

- L_i is the normalized load impact on node i. From now on, we will consider only the multirate case (since the single-rate can be seen as a particular case of a multirate network, where all transmission rates are the same). We shall use capital letters for referring to normalized values.
- x_{jk} is the total traffic (in bps) that node j wants to transmit to node k.
- v_{jk} is the transmission rate used between nodes j and k.

Since the equation is normalized, if the node is not overloaded, L_i should be a value between 0 and 1.

The use of the union operator states that some transmissions in the neighborhood may overlap in time. This can happen in CSMA-based networks whenever these transmissions do not interfere with each other, as shown by figure 3.2. In this example, transmissions a and b can overlap in time.

Once we have computed the load impact on each node of the ad-hoc network and after defining the amount of normalized bandwidth dedicated to QoS traffic as Q, we are able to state the following *QoS constraint* that should be respected in order to provide QoS guarantees for real-time flows:

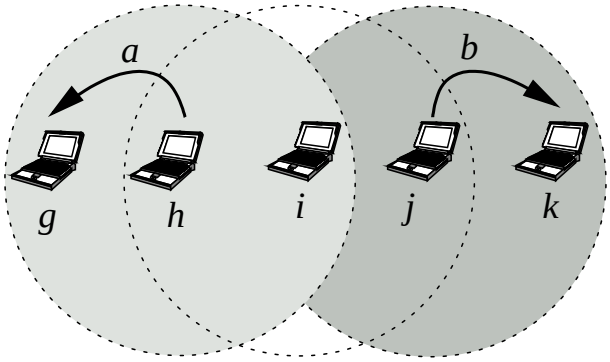


Figure 3.2: Simultaneous transmissions in the neighborhood of node i

$$L_i \leq Q, \forall i \in \mathcal{S} \qquad (0 \leq Q \leq 1) \tag{3.3}$$

Where $\mathcal{S}$ is the set of nodes that are transmitting or receiving QoS traffic, i.e. the nodes having at least one QoS reservation. In the rest of the thesis we shall refer $\mathcal{S}$ as the QoS set. By guaranteeing condition 3.3, we can guarantee that the channel occupancy due to the QoS traffic observed by any node of the QoS set is never greater than Q. This condition should guarantee that there is enough capacity to accommodate all QoS flows.

Note that Q can be understood as the percentage of time that the channel can be busy at any node, because either it is transmitting or receiving traffic that belongs to the QoS flows. We shall assume that the MAC is able to restrict non-QoS traffic, such that the normalized capacity Q will be always available for QoS traffic. This could be achieved e.g. using 802.11e, or 802.11 with some additional mechanism, e.g. SWAN [68], that regulates non-QoS traffic. Of course, due to collisions, impact of non-QoS traffic and other reasons, the amount of normalized capacity Q available for QoS traffic may vary. To cope with that, a conservative value shall be used for Q, or it may be made adaptive, as we proposed in [40].

3.3 The Basis of BRAWN

As previously mentioned, BRAWN is based on the computation of the available bandwidth (AB) in each node of the network. The goal of our *bandwidth reservation mechanism* is to provide rate allocation (e.g. peak or sustainable rate) and, at the same time, remain as simple as possible. The solution should provide QoS and yet introduce as little overhead as possible in the network. In order to do that, it should only make use of the information

about its 1-hop neighborhood. Since most of the available ad-hoc routing protocols already provide 1-hop signaling, e.g. HELLO messages, any additional information that may be necessary can be piggybacked on these signaling messages.

In order to provide a simple mechanism that is feasible to implement, some simplifications must be done. The first of them is related to the computation of the load impact on each node of the ad-hoc network. The use of the union operator, as shown by equation 3.2, is not possible, since a node has no idea of the "degree of simultaneity" of the transmissions on the neighborhood. For this reason, we simplify the equation by using a simple sum instead, since it is always more restrictive than using the union (figure 3.1), what still guarantees the QoS requirements. Thus, the load on a node i will be computed as:

$$L_i = \left| \bigcup_{j \in \mathcal{N}_i^+} X_j \right| \approx \sum_{j \in \mathcal{N}_i^+} X_j \tag{3.4}$$

where:

$$X_j = \sum_{\forall k} \frac{x_{jk}}{v_{jk}} \tag{3.5}$$

is the normalized amount of QoS traffic that node j wants to transmit (either if node j is the source of the traffic or if it is just forwarding). In BRAWN each node would reserve bandwidth for this traffic, thus, X_j can also be interpreted as the total reserved bandwidth at node j. Using a sum in equation 3.4 to represent a union may be pessimistic in some scenarios. In [33] the approximation given by equation 3.6 has been proposed.

$$\left| \bigcup_{j \in \mathcal{N}_i^+, \forall k} X_j \right| \approx \sum_{j \in \mathcal{N}_i^+} X_j - \sum_{j,k \in \mathcal{N}_i^+} |X_j \cap X_k| \tag{3.6}$$

However, in 802.11-like networks two transmissions cannot overlap in time whenever either the sender or the receiver of one transmission is a neighbor of either the sender or the receiver of the other one. Therefore, in order to accurately compute which intersections from equation 3.6 are not null, a node would need individual information about every flow in the neighborhood, so that it would be able to identify those that may take place

simultaneously. Exchanging this information would introduce too much overhead in the protocol. Consequently, we have considered equation 3.4 as a convenient approximation for the load demand.

3.3.1 The Available Bandwidth in each node

Once each node is able to compute the load demand on itself, this value can be used to establish which part of the total bandwidth dedicated to QoS connections is still available for reservations. By using just the information locally known by a node (the pre-established Q value and the computed load impact), we define a new value that represents this availability for new flows to be established, which we call the Maximum Available Bandwidth (MAB).

$$MAB_i = Q - L_i \tag{3.7}$$

This value is simply the amount of bandwidth available for QoS flows minus the amount of bandwidth already consumed under the point of view of this node, i.e., its load impact. By looking at equation 3.3 it is quite simple to notice that we may re-write the QoS constraint using this new value.

$$MAB_i \geq 0, \forall i \in \mathcal{S} \tag{3.8}$$

However, knowing the local MAB of a node is not enough for the node to decide if new flows can be accepted. This is because the available bandwidth of a given node i is also affected by transmissions of its two-hop nodes that have one of the neighbors of i as a receiver. In figure 3.3, for example, the transmission from g to h only causes an impact on the computation of MAB_g and MAB_h, although when it takes place, node i is not allowed to transmit (notice, however, that $MAB_i = 1$). That means that a node also needs to take into account its neighbors restrictions.

We, thus, propose to estimate what we call the Available Bandwidth of a node i (AB_i) as the minimum value of the MABs in its QoS set neighborhood:

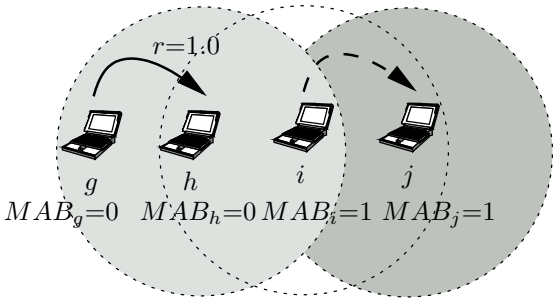


Figure 3.3: The Maximum Available Bandwidth and restrictions imposed by neighbors

$$AB_i = \min\{MAB_j\}, j \in \mathcal{N}_i^+ \cap \mathcal{S} \tag{3.9}$$

This value can also be understood as a more complete view of the node about the impact of new transmissions on the neighborhood. It is, in fact, the amount of bandwidth available for new transmissions over a given node.

Now, the QoS constraint given by equation 3.8 can be rewritten in terms of the available bandwidth as we state in the following theorem:

Theorem A. *Guaranteeing that the AB given by equation 3.9 of every node that takes part in a reserved path is non-negative, is equivalent to guaranteeing that the MAB of every node of the QoS set is non-negative.*

See the proof of this theorem in appendix A. In other words, the QoS constraint given by equation 3.8 can be rewritten as:

$$AB_i \geq 0, \forall i \in \textit{reserved paths} \tag{3.10}$$

Summing up, BRAWN requires that the nodes know the normalized amount of traffic (X_j) and the maximum available bandwidth (MAB_j) of their neighbors belonging to the QoS set ($\mathcal{S}$). These values should be periodically exchanged among neighbors belonging to $\mathcal{S}$. Nodes that do not belong to $\mathcal{S}$ would compute the MAB_j, which could be needed in the CAC of future QoS reservations, but they would not send it. Each node i uses X_j to compute the load L_i using equation 3.4, and MAB_i using equation 3.7. Finally, the available bandwidth (AB_i) is computed using equation 3.9.

3.3.2 Call Admission Control

After defining the distributed mechanism to compute the *available bandwidth* at each node of the network, we will use this value to decide whether a new connection of r bps fits or not in a given node.

The first step toward the definition of a Call Admission Control (CAC) is realizing which transmissions cannot take place while a node i is transmitting towards a node j. As we have discussed before, if we are using a CSMA-like protocol, none of the i's neighbors nor the j's neighbors are allowed to transmit while i is transmitting to j. Therefore, the following CAC should be checked in every node along a candidate path:

$$AB_i \geq \left| \bigcup_{y \in \left(\left(\mathcal{N}_i^+ \cup \mathcal{N}_j^+ \right) \cap path \right)} \frac{r}{v_y} \right| \tag{3.11}$$

where

- i represents the current node in the path;
- j represents the next node in the path (to which i will transmit);
- r represents the bandwidth required by the new connection;
- v_y is the transmission rate from node y toward its next hop in the path.

In this case, just like in the load demand computation (equation 3.4), we use a simple sum approximation for the union operator.

$$AB_i \geq \sum_{y \in \left(\left(\mathcal{N}_i^+ \cup \mathcal{N}_j^+ \right) \cap path \right)} \frac{r}{v_y} \tag{3.12}$$

See the proof that this CAC condition guarantees the QoS constraint presented by equation 3.10 in appendix B.

Notice that in the case that nodes move, topology changes in the network may cause connections that were previously accepted by the CAC not to have their QoS requirements guaranteed after a while. Moreover, even if QoS can still be guaranteed over a given path,

topology changes may cause more efficient paths to show up, and being able to use them may optimize the use of network resources. Therefore, in the presence of movement, the QoS mechanism should be adaptive. This could be achieved e.g. by periodically refreshing reservations, so that the network is constantly re-validating the admission control and searching for better routes for previously established connections.

3.4 Exemplifying BRAWN's behavior

In order to better understand the behavior of the BRAWN mechanism, we will take a step-by-step look at the ad-hoc network example depicted by figure 3.4. In this simple example all links between mobile nodes are 5 Mbps. Assume that in this network there is an established reservation for a QoS flow of 1 Mbps following the path $MN_A \rightarrow MN_B \rightarrow MN_E \rightarrow MN_F$. For simplifying the example, we shall also assume that the reserved capacity for QoS traffic is $Q = 1$.

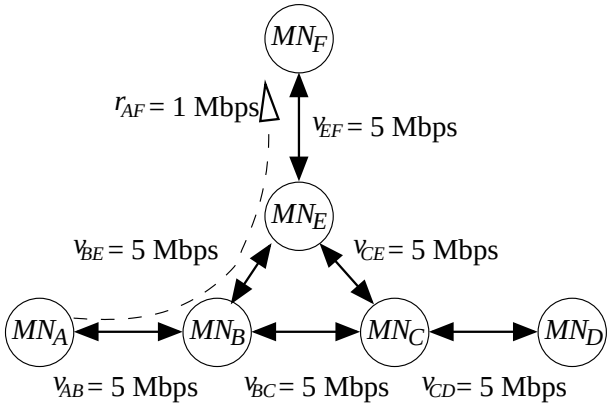


Figure 3.4: Network topology

The row X_i in table 3.1.(a) shows the normalized amount of traffic that would be advertised by the nodes. Upon receiving theses values, each node would compute the MAB_i shown in the corresponding row of the table. For instance, MN_B would receive $X_A = 0.2$, $X_C = 0.0$ and $X_E = 0.2$. Since $X_B = 0.2$, it would compute $MAB_B = 0.4$. Finally, upon receiving the MAB from their neighbors, nodes would compute the AB_i given in the table. Note that nodes MN_C and MN_D would not advertise their MAB, because they do not belong to the QoS set.

Assume that after this, node MN_C wishes to establish a new QoS flow of $r_{CD} = 2$ Mbps with node MN_D. The following CAC conditions would be checked: $AB_C \geq 0.4$ and $AB_D \geq 0.4$ (2 Mbps / 5 Mbps = 0.4). Thus, the flow would be accepted, and the values of X_i, MAB_i and AB_i would change as shown in table 3.1.(b).

Table 3.1: Parameters computed by nodes using BRAWN. With flow r_{AF} (a), and with flows r_{AF} and r_{CD} (b).

		MN_A	MN_B	MN_C	MN_D	MN_E	MN_F
(a)	X_i	0.2	0.2	0.0	0.0	0.2	0.0
	MAB_i	0.6	0.4	0.6	1.0	0.6	0.8
	AB_i	0.4	0.4	0.4	1.0	0.4	0.6
(b)	X_i	0.2	0.2	0.4	0.0	0.2	0.0
	MAB_i	0.6	0.0	0.2	0.6	0.2	0.8
	AB_i	0.0	0.0	0.0	0.2	0.0	0.2

We may intuitively check that, after accepting the flow r_{CD}, the available bandwidth computed by the nodes is correct: Whenever one of the nodes MN_A, MN_B, MN_C and MN_E send a packet, all the others in this set must remain silent. Since altogether send 5 Mbps, which is the link capacity, their available bandwidth is 0. Node MN_E must be silent whenever MN_B, MN_C or MN_F transmit. Since nodes MN_E, MN_B and MN_C transmit altogether 4 Mbps, the available bandwidth at node MN_F is 1-4/5 = 0.2. Similarly, we can derive that the available bandwidth at node MN_D is also 0.2.

3.4.1 Comparing BRAWN to AQOR

Among the previously proposed protocols, AQOR is the solution that most resembles that of BRAWN. In this section we use the previous example to compare BRAWN and AQOR in terms of the calculation of the available bandwidth.

In AQOR the authors define $B_{self}(I)$ as the total traffic transmitted or received at a node I. B_{self} is periodically exchanged between neighbors. Then, the available bandwidth ($B_{available}$) is computed as:

$$B_{available}(I) = B - \sum_{J \in N(I)} B_{self}(J) \tag{3.13}$$

where B is the maximum transmission bandwidth (5 Mbps in the above example), and $N(I)$ is the neighborhood of node I. Table 3.2 shows the B_{self} and $B_{available}$ values that would be computed by the nodes using AQOR in figure 3.4 (we did not use normalized values, as it was done in BRAWN, since AQOR was not defined for multirate networks). Note that AQOR would estimate an available bandwidth of only 1 Mbps at node MN_C, while we have seen before that a new flow r_{CD} = 2 Mbps could be accepted at node MN_C. Nevertheless, if the flow r_{CD} were accepted, the nodes would update B_{self} and

$B_{available}$ as shown in table 3.2.(b). Note that the value B_{self} would become negative at node MN_C, meaning that there has been an over-reservation of resources. In fact, the authors of AQOR have reported that the more traffic is sent in the neighborhood, the more conservative is the estimation of the available bandwidth. Therefore, we conclude that BRAWN is able to estimate the available bandwidth much more accurately.

Table 3.2: Parameters computed by nodes using AQOR. With flow r_{AF} (a), and with flows r_{AF} and r_{CD} (b).

		MN_A	MN_B	MN_C	MN_D	MN_E	MN_F
(a)	B_{self} (Mbps)	1	2	0	0	2	1
	$B_{available}$ (Mbps)	3	2	1	5	2	3
(b)	B_{self} (Mbps)	1	2	2	2	2	1
	$B_{available}$ (Mbps)	3	0	-1	3	0	3

3.5 Implementation Issues

BRAWN can be integrated into many routing protocols proposed for AWNs. In this section we explain how we have integrated it in one protocol that uses a reactive approach, AODV, and another one that uses a proactive approach, OLSR.

3.5.1 Integrating into AODV

In order to take advantage of the multirate characteristics of current networks, we integrated BRAWN into the modified version of AODV that supports the election of routes taking link rate into account, the MR-AODV protocol proposed in chapter 2.

As previously discussed, MR-AODV provides a neighbor discovery mechanism based on the periodic broadcast of HELLO messages. These messages are broadcasted to the one-hop neighborhood and, by receiving them, a node is able to be aware of its neighbors. BRAWN makes use of these messages by piggybacking on them the information that a node should have about its neighbors (X_j and MAB_j) in order to compute the load impact (L_i) and the available bandwidth (AB_i), as seen in equations 3.4 and 3.9.

The detailed changes required to integrate BRAWN into MR-AODV are the following:

(i) HELLO messages are modified such that each node i advertises X_i, MAB_i, AB_i and AB_j ($\forall j \in \mathcal{N}_i$) to its neighbors. This is all the information that node i's neighbors need to compute equation 3.4 and to eventually perform the CAC for some of its

neighbors that take part in the chosen path but that do not forward the RREQ message (see the behavior of MR-AODV in section 2.4).

(ii) Each node i collects these QoS HELLO messages from its neighbors to compute AB_i according to equation 3.9.

(iv) In order to find a route that meets the QoS requirements, we modified the MR-AODV route selection algorithm to find a shortest hop path that has enough bandwidth (the shortest-widest path) to meet these requirements.

(v) Bandwidth reservation at intermediate nodes is done by adding the required bandwidth in the already existing RREQ and RREP messages, so that routes that do not meet these requirements are excluded from the routing election process.

As previously explained, each node gathers from HELLO packets the necessary information for performing the CAC. We modified the default route selection algorithm from MR-AODV so that it is able to compute a route for QoS flows that meets bandwidth requirements and delivers a shortest-widest path. The CAC is performed during this route computation in order to remove intermediate nodes that do not have enough available resources.

In fact, since not every node that takes part in a route forwards a RREQ (remember that in MR-AODV, when a node receives a RREQ it computes the highest throughput path towards the last hop, and includes possible intermediate nodes into the final computed path), nodes should be able to compute the CAC not only for them, but also for nodes in the path that did not forward RREQ messages. That may be achieved by piggybacking on the RREQ the *AB* information for each node that takes part in the path, so that whenever a node receives a RREQ it may check if the flow fits not only based on its own *AB*, but also based on the *AB* of previous hops.

Note that although it seems too much information to piggyback on a single signaling message, each of these values (X, *MAB* and *AB*) can be stored on just a few bits depending on the desired granularity (if 16 bits are used, we may represent up to 64Mbps in units of 1Kbps, for example).

Figures 3.5 and 3.6 show our proposal for the extensions of the HELLO and RREQ/RREP AODV messages respectively.

BRAWN Extension for the HELLO message:

- *Type*: Type of AODV extension. We may use any unused type number for the BRAWN HELLO extension, 250 for example;

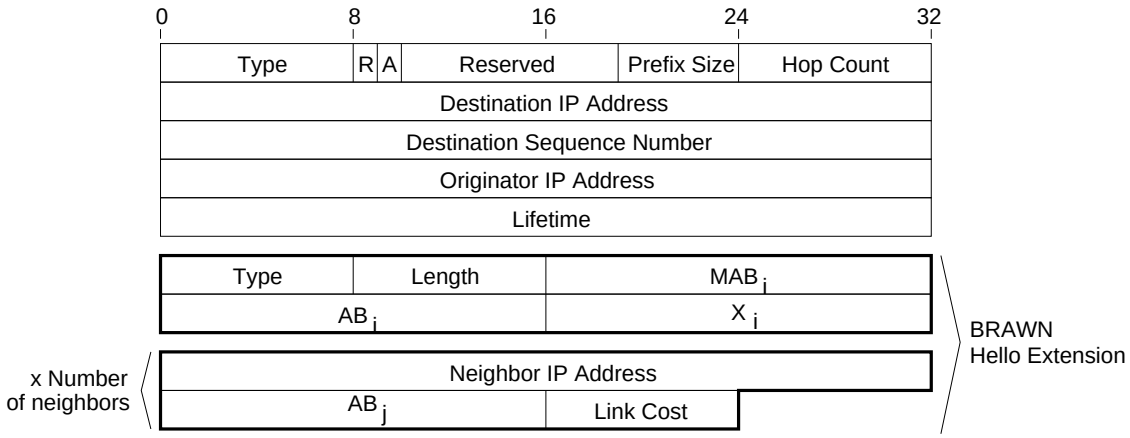


Figure 3.5: Extension proposed for the AODV HELLO message

- *Length*: Length of the AODV extension;
- MAB_i: Normalized Maximum Available Bandwidth computed by the node (equation 3.7);
- AB_i: Normalized Available Bandwidth computed by the node (equation 3.9);
- X_i: Normalized sum of all traffic generated/forwarded by the node (equation 3.5);
- *Neighbor IP Address:* IP Addresses of all 1-hop neighbors;
- AB_j: Normalized Available Bandwidth of all 1-hop neighbors;
- *Link Cost*: Link cost from the node towards all 1-hop neighbors;

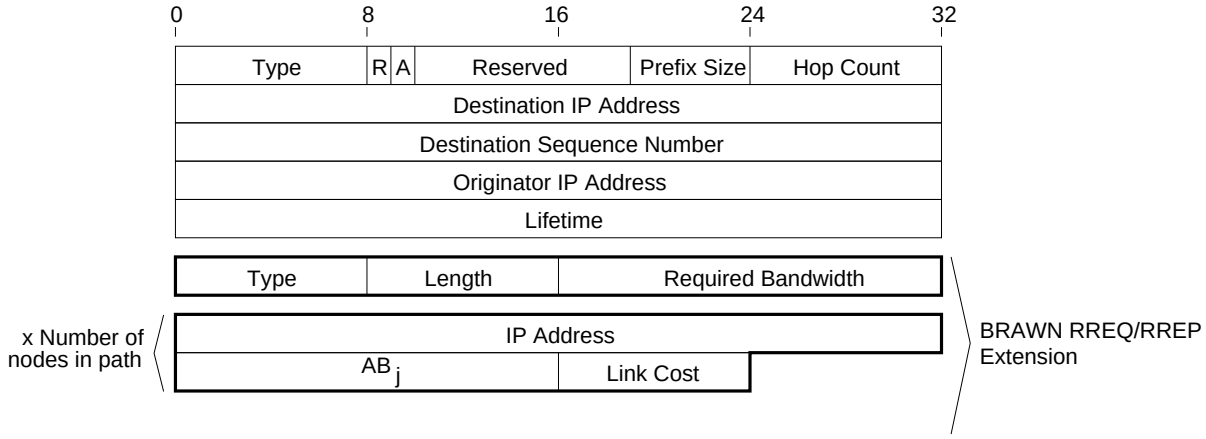


Figure 3.6: Extension proposed for the AODV RREQ and RREP messages

BRAWN Extension for both RREQ and RREP messages:

- *Type*: Type of AODV extension. We may use any unused type number for the BRAWN RREQ/RREP extension, 251 for example;
- *Length*: Length of the AODV extension;
- *Required Bandwidth*: Bandwidth required for the reservation of a new flow (in Kbps);
- *IP Address*: IP Addresses of every node in the candidate path (used for path accumulation);
- AB_j: Normalized Available Bandwidth of every node in the candidate path;
- *Link Cost*: Link cost of every hop in the candidate path (as proposed in section 2.4);

3.5.2 Simulation Results with MR-AODV

We have added our reservation scheme into an AODV implementation and then simulated its behavior using ns-2 [6] version 2.30.

Simulations were run using the scenario described as follows:

- MAC: 802.11 with multirate.
- Multirate parameters: for a distance less than 50 meters the rate is 11 Mbps; for a distance between 50 and 90 meters the rate is 2 Mbps, according to the ORiNOCO 802.11b PC card specification for a semi-open environment[1].
- Carrier sensing range: 200 meters (around 2.2 × 2 Mbps transmission range).
- CBR connections sending 500 bytes packets with a 32 kbps rate.
- 20 to 100 nodes randomly placed over a square of 300 × 300 meters.
- 20 flows are initiated (one each 15 seconds) between random pairs of nodes.
- The simulation time is 400 s, including a 10 s startup period that gives AODV the time to exchange routing information before applications start.
- Nodes do not move.

The first evaluation we should do is the election of the value for the QoS parameter Q, since it should not be chosen arbitrarily. Using a simulation setup with 40 nodes and varying Q from 10% to 27.5% (0.1 to 0.275) in steps of 2.5%, we have investigated the effect of this parameter on the end-to-end delay and packet loss measurements. Note that we have run 10 simulations with different node positions for each value of Q, so that the results shown below represent the average value with a confidence interval.

Results are summarized in figures 3.7 and 3.8: the former presents the average 99.9 end-to-end delay percentile of the flows for different values of Q while the latter presents the percentage of packets that were lost.

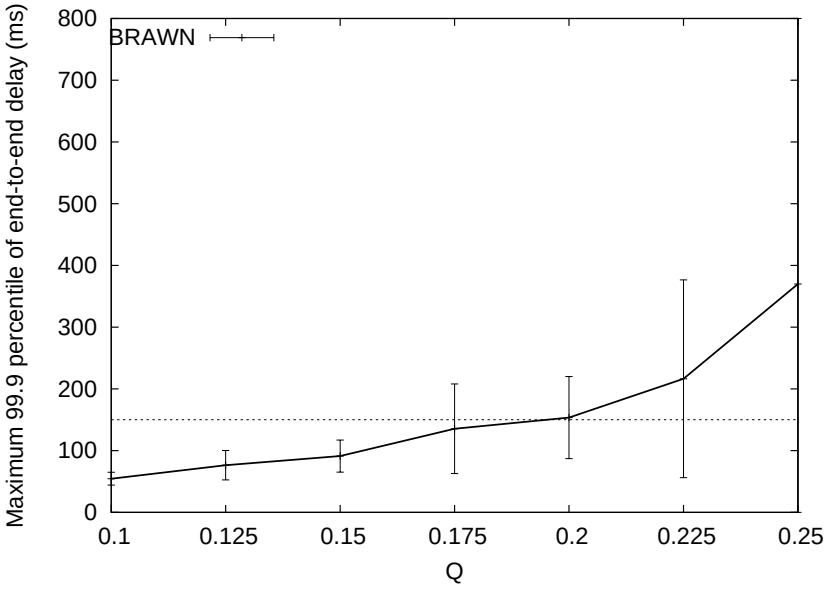


Figure 3.7: 99.9 end-to-end delay percentile for different values of Q

These graphs show that while loss is not a such problem in these scenarios, the end-to-end delay achieves undesirable values for interactive multimedia applications (preferred to be lower than 150ms [2]) when Q is greater than 20%. We have, thus, chosen Q to be equal to 20% for the following simulations.

Having chosen a value for the Q parameter, we have compared the BRAWN mechanism to a scenario where no QoS is supported (using the MR-AODV protocol). By varying the network density, we were able to observe the number of flows that were accepted by the CAC of the BRAWN mechanism and the number of flows that could be accepted by MR-AODV before the network nodes reached an occupancy of 25% (about the same one allowed by BRAWN for $Q = 20\%$, when considering the overhead generated by IP and ethernet headers, by MR-AODV and by the reservation signaling). See figure 3.9 for these results.

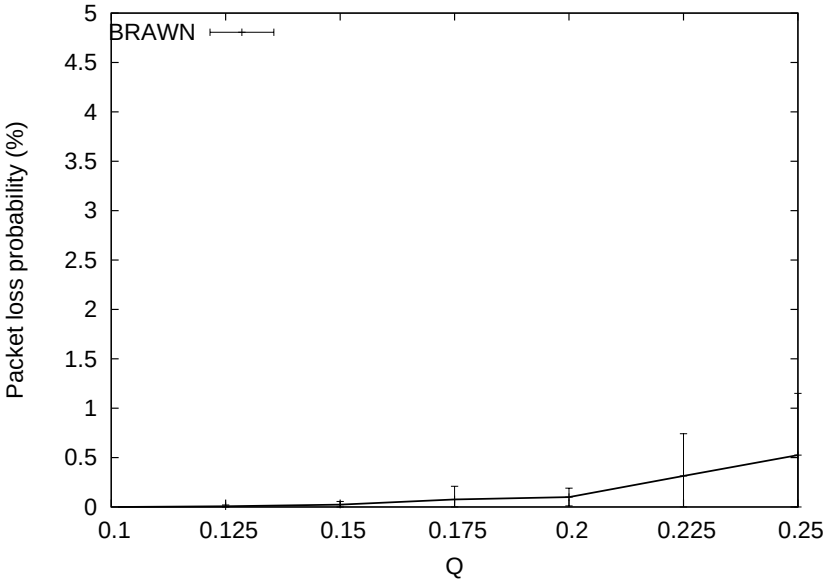


Figure 3.8: Packet loss for different values of Q

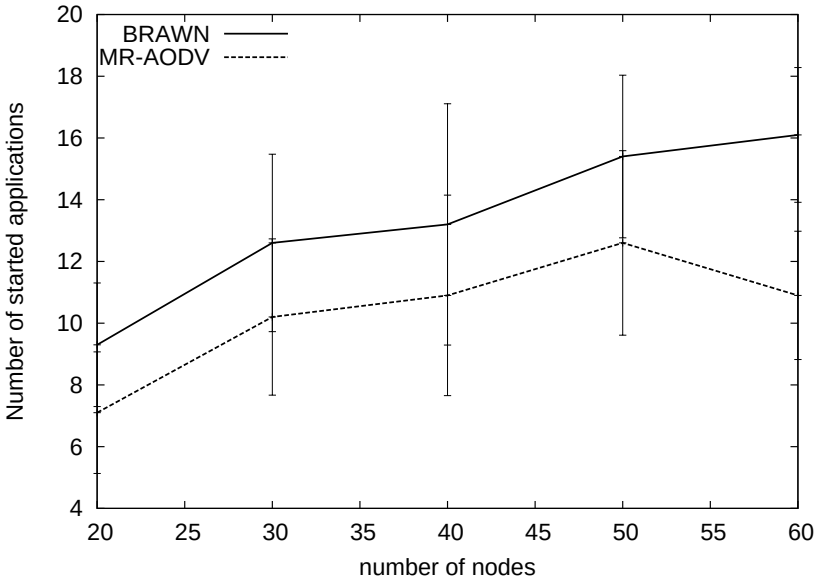


Figure 3.9: Number of "accepted" flows for different network densities

As we have previously discussed, BRAWN naturally distributes the traffic more evenly through the network, since whenever a node is about to reach congestion, it is no longer used in new paths. That makes new flows to be routed through (possibly longer) paths that avoid potential congestion areas. Due to this behavior, BRAWN is able to accept more flows than MR-AODV when using this "occupancy cut-off"

This cut-off, however, was artificially introduced into MR-AODV in order to compare the load-balancing that is provided by our mechanism. In its standard behavior, MR-AODV

never rejects new flows, even when the network is already congested. This behavior can be easily seen in figures 3.10 and 3.11.

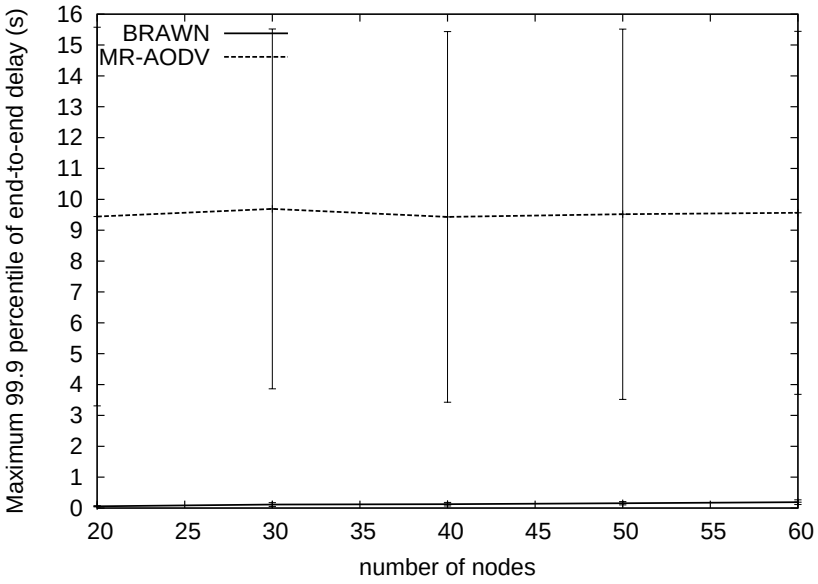


Figure 3.10: 99.9 end-to-end delay percentile (in seconds) for different network densities

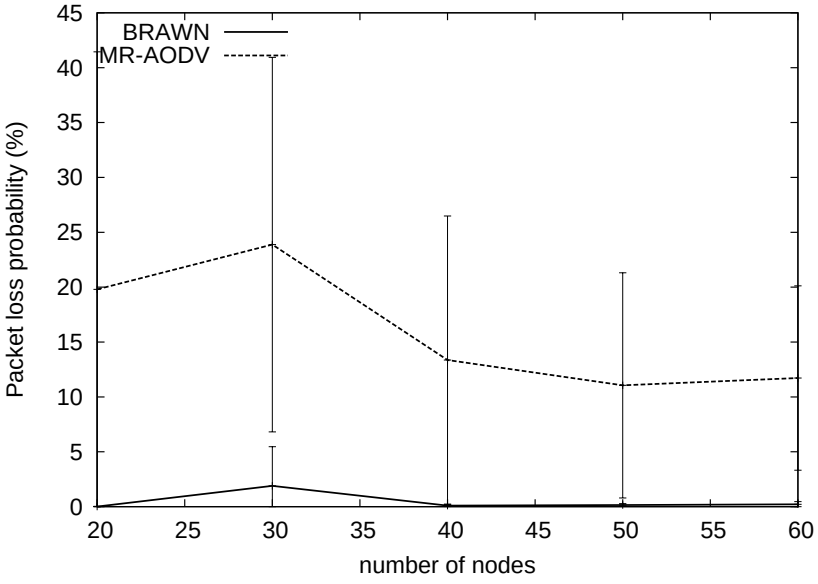


Figure 3.11: Packet loss probability for different network densities

The first figure shows the average value of the 99.9 percentile end-to-end delay suffered by flows when using BRAWN or MR-AODV. With BRAWN, delays are always under an acceptable limit, since the admission of new flows is limited by the mechanism and routes are well spread through the network. When using MR-AODV, however, delays are very high and increase as the network density gets higher.

The second figure shows the packet loss probability. By using our mechanism, losses are almost insignificant, while the lack of control of MR-AODV leads to high packet loss rates.

In order to provide a better understanding of the behavior of the mechanism, below we present some figures that analyzes the dynamics of BRAWN on a single run with 40 nodes deployed on random positions. Since results on a single simulation may be influenced by many random factors (e.g. position of the nodes), we first repeated the simulation 500 times, each of them using different node placements in order to guarantee that this single simulation is a representative scenario, i.e., the number of accepted flows in this single simulation is close to the average number of accepted flows in many different simulation runs. Figure 3.12 shows the results we obtained for these 500 repetitions. BRAWN accepts an average of 14 connections.

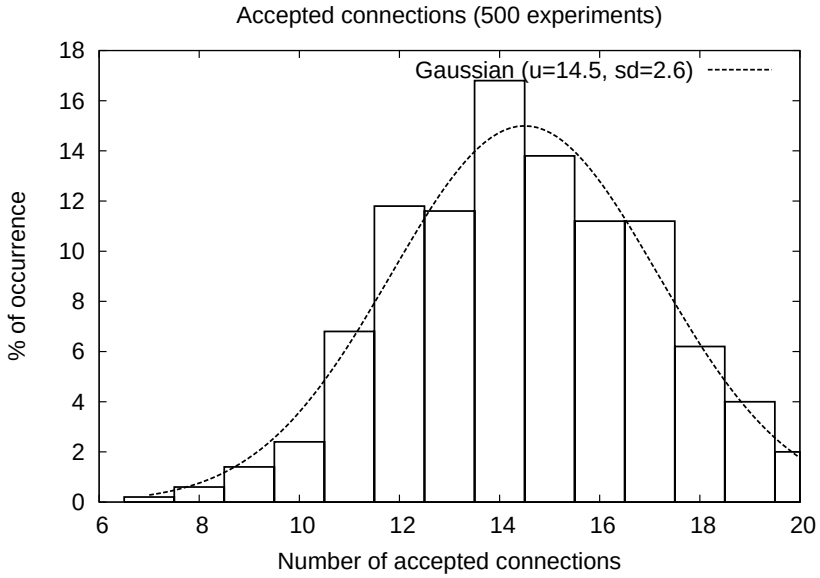


Figure 3.12: Probability of accepting a given number of connections when using BRAWN

We then launched a single scenario where 14 connections were accepted to be analyzed. Figure 3.13 shows the evolution of the connections that were established when using each protocol on this specific scenario. Note that all connections are established with MR-AODV while only 14 with BRAWN (the others are blocked).

Following the analysis of the same scenario, we are able to notice that end-to-end delays and packet loss increase significantly around 250 seconds of simulation. At this moment the MAC gets congested and requirements can no longer be guaranteed for the previously established flows when using MR-AODV. Figure 3.14 depicts the maximum end-to-end delay of CBR packets, and figure 3.15 depicts the maximum percentage of packets lost by connections, measured in intervals of 1 second. These figures show us that BRAWN is

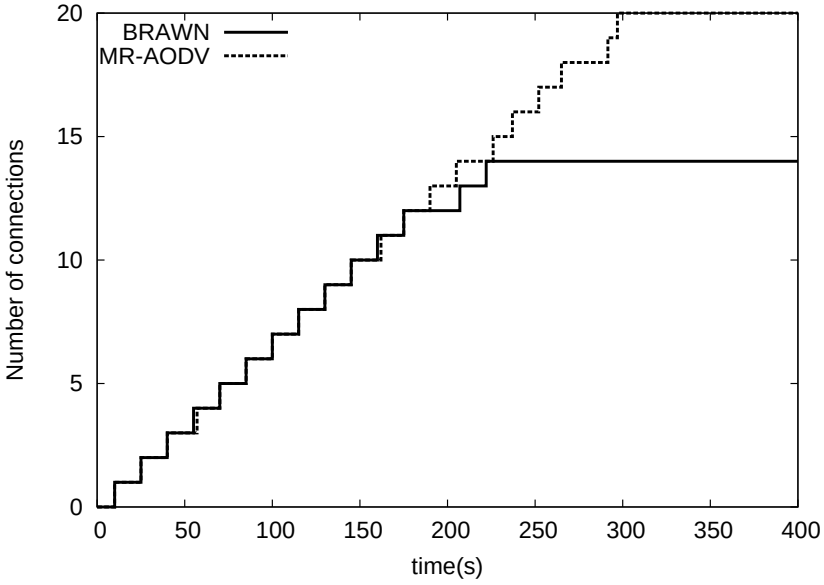


Figure 3.13: Connection setup

not only successful in avoiding network congestion, but also in avoiding packet losses and increased delays. Compared to BRAWN, MR-AODV behaves much worse, since it looses up to 20% of the packets at some instances.

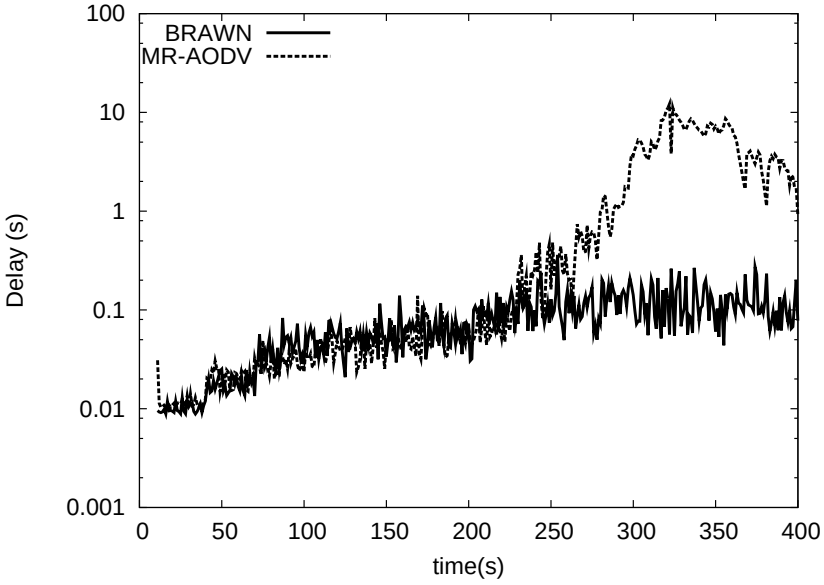


Figure 3.14: Maximum delay

It is also interesting to know how many connections are suffering from congestion. Figures 3.16 and 3.17 show the transmission delay Complementary Cumulative Distribution Function (CCDF), i.e. Prob{transmission delay $> x$}, for all established connections with MR-AODV and BRAWN. Figure 3.16 shows us that using MR-AODV, about half

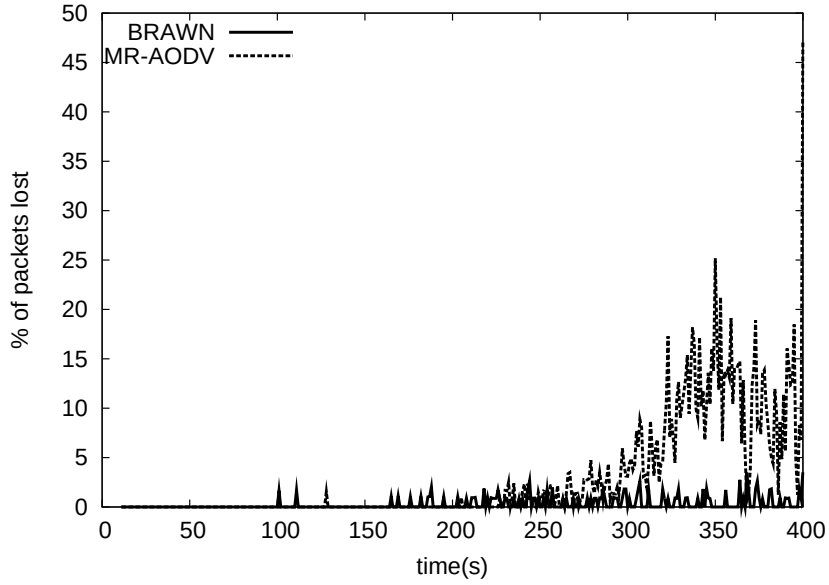


Figure 3.15: Maximum loss

of the flows have a 10% chance of having at least a 1 second delay. With BRAWN on the other hand the majority of the flows have end-to-end delays smaller than 150 ms with a probability higher than 99.9%.

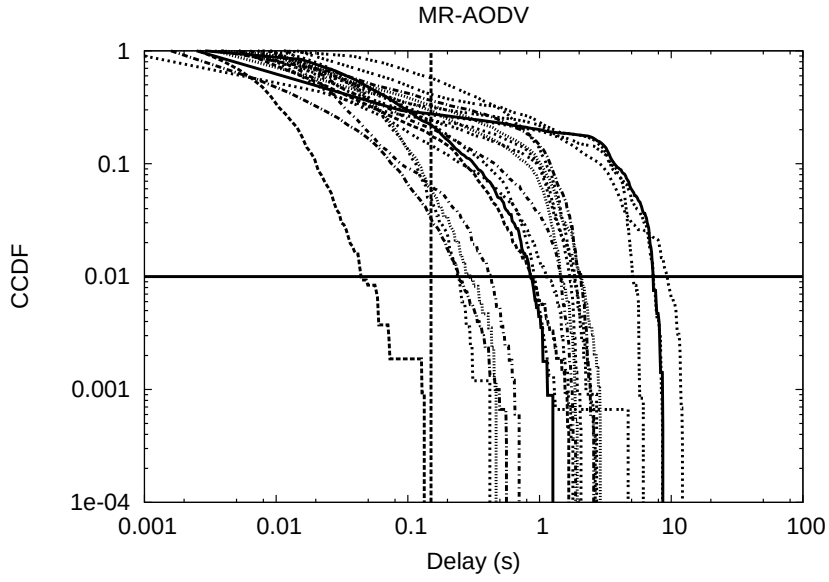


Figure 3.16: MR-AODV delay histogram

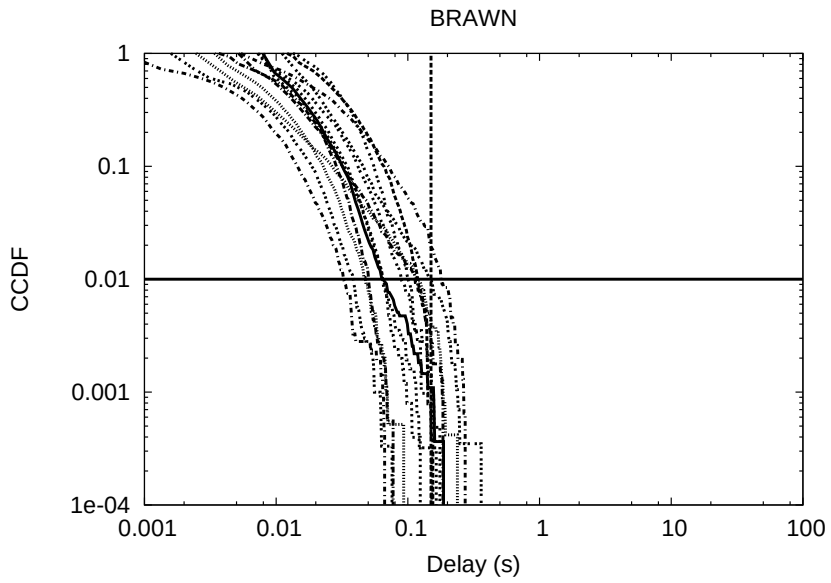


Figure 3.17: BRAWN delay histogram

3.5.3 Integrating into OLSR

OLSR provides a neighbor discovery mechanism based on the periodic broadcast of HELLO messages. These messages are broadcasted to the one-hop neighborhood and, by receiving them, a node is able to be aware of its neighbors. BRAWN makes use of these messages by piggybacking on them the information that a node should have about its neighbors (X_j and MAB_j) in order to compute the load impact (L_i) and the available bandwidth (AB_i), as seen in equations 3.4 and 3.9.

The detailed changes required to integrate BRAWN into OLSR are the following:

(i) OLSR HELLO messages are modified such that each node i with QoS reservations advertises X_i and MAB_i to their neighbors. This is all the information that node i's neighbors need to compute equation 3.4.

(ii) Each node i collects these QoS HELLO messages from their neighbors to compute AB_i according to equation 3.9.

(iii) OLSR TC messages are modified to also advertise AB_i and v_{ij} of each of node i's MPR selectors. By doing that, each node has knowledge of the network topology and the bandwidth available in the network.

(iv) In order to find a route that meets the QoS requirements, we modified the OLSR route selection algorithm to find a shortest hop path that has enough bandwidth (the shortest-widest path) to meet these requirements. Since TC messages also

The following pseudo algorithm describes how the CAC is integrated into the OLSR route selection algorithm:

(1) Add all one hop neighbors registered as symmetric to the routing table with a hop-count of 1 AND FOR WHICH THE CAC ALLOWS THIS ROUTE (SEE EQUATION 3.12).

(2) For each symmetric one-hop neighbor, add all two hop neighbors registered on that neighbor that has:

- not already been added to the routing table;
- a symmetric link to the neighbor;
- BEEN ALLOWED BY THE CAC (SEE EQUATION 3.12).

These Entries are added with a hop-count of two and next-hop as the current neighbor. Set n equal to two.

(3) Then, for every added node N in the routing table with hop-count n add all entries from the TC set where:

- the originator in the TC entry is N;
- the destination has not already been added to the routing table;
- THE CAC DETERMINED THAT ENOUGH RESOURCES ARE AVAILABLE ALONG THE ROUTE (SEE EQUATION 3.12).

New entries are added with a hop count of n+1 and next-hop as the next-hop registered on N's routing entry.

(4) Increase n with one and do step 3 over until there are no entries in the routing table with hop-count equal to n or if a route to the destination was found.

Figure 3.18: Integration of the CAC Algorithm in OLSR

advertise AB_i and v_{ij}, the originating node has sufficient information to decide if enough resources are available (see equation 3.12).

(v) Bandwidth reservation at intermediate nodes is done through the exchange of Reservation Request / Reservation Reply messages previously to sending data packets.

The CAC in OLSR

The OLSR routing protocol uses an optimized version of the Dijkstra algorithm to compute shortest path routes to all the other nodes in the network. However the QoS routing

in BRAWN needs to find a route towards a specific destination that meets the bandwidth requirements of the new flow and at the same time avoids a QoS violation should the flow be allowed. Both goals are accomplished by applying the CAC algorithm as shown in section 3.3.2.

We modified the default route selection algorithm from OLSR so that it is able to compute a route for QoS flows that meets bandwidth requirements and delivers a shortest-widest path. The CAC is performed on each new link that is added to the node's topology map so that links that would result in the QoS constraint being broken somewhere along the route will no longer be taken into account by the routing algorithm. As previously explained, each node gathers from HELLO and TC packets the necessary information to perform the CAC.

Figure 3.18 shows a pseudo code for the route computation algorithm. It is based on the route computation algorithm defined by OLSR, to which we added especially important parts for the QoS routing decisions (in uppercase). While this algorithm can compute shortest-widest paths to all other nodes in the network, we are only interested in one to the requested destination. After this route is found, the reservation signaling reserves resources along the path.

3.5.4 Reservation signaling interaction

Reservation of requested bandwidth is done by sending a reservation request (*ResvReq*) from the source towards the destination (Figure 3.19). On receiving such a request an intermediate node determines the next hop for the QoS flow, which also involves checking that the QoS constraint is not being violated.

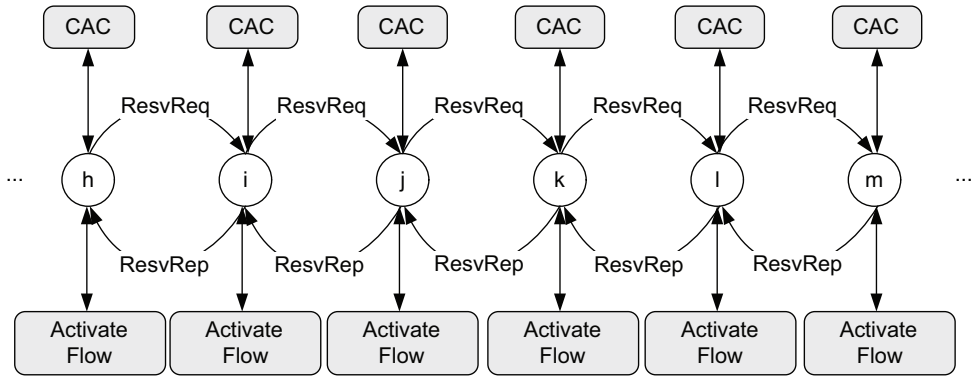


Figure 3.19: Reservation Signaling

If the destination is reached, it sends back a positive reply towards the source using the reverse path. On seeing the reply coming through, the intermediate nodes installs the

QoS route into their active routing table.

Should it occur that the QoS constraint would be violated in one of the intermediate nodes, then a negative reply is returned, containing new information on the availability of bandwidth at this intermediate node. Using this information the source can try to reserve resources along another path, if it exists.

By performing the reservation this way, we still stay close to the OLSR philosophy, where the link state routing algorithm is only used to find a suitable next hop towards the destination. On the one hand the source can use the CAC to locally decide if it is necessary to block the flow, on the other hand all nodes work together in a distributed manner to decide the best path towards the destination. The idea behind this is that, although all nodes in the network obtain information about the topology and the available bandwidth, this information might not be up to date since the topology signaling is only performed periodically and also happens at a lower rate than the HELLO signaling related to the local neighborhood.

Simulations of the BRAWN mechanism integrated to the NRL OLSR implementation [3] in ns-2 may be seen in [41].

Proposed extensions and new signaling messages format

According to the required modifications that were previously described, the new proposed formats for the HELLO and TC messages are depicted by figures 3.20 and 3.21 respectively. QoS information is carried by using the *Link Code* 250, which is not used by standard OLSR and is silently discarded by any node that does not recognize the code, i.e., any node that does not implement our proposed QoS extensions.

Furthermore, two additional signaling messages are proposed: Reservation Request and Reservation Reply. The proposed formats for these two messages are depicted by figures 3.22 and 3.23.

BRAWN Extension for the HELLO message:

- *Link Code*: Link Code of the OLSR extension. We may use any unused type number for the BRAWN HELLO extension, 250 for example;
- *Link Message Size*: Length of the OLSR extension;
- MAB_i: Normalized Maximum Available Bandwidth computed by the node (equation 3.7);

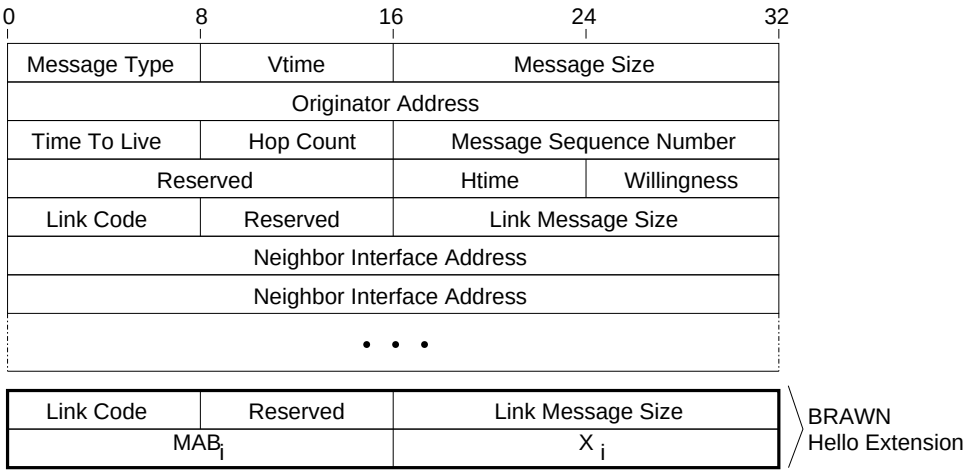


Figure 3.20: Extension proposed for the OLSR HELLO message

- X_i: Normalized sum of all traffic generated/forwarded by the node (equation 3.5);

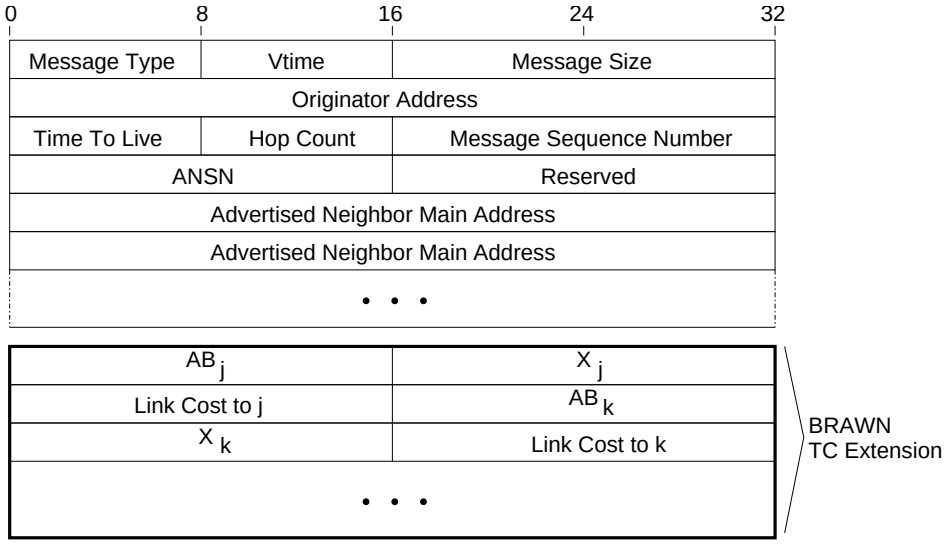


Figure 3.21: Extension proposed for the OLSR TC message

BRAWN Extension for the TC message:

- AB_j, AB_k, etc: Normalized Available Bandwidth of all its MPR selectors;
- X_j, X_k, etc: Normalized sum of all traffic generated/forwarded by all its MPR selectors;
- *Link Cost to j, k, etc*: Link cost from the node towards all its MPR selectors;

Proposed Reservation Request message:

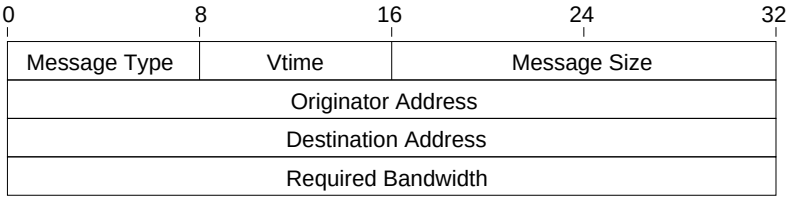


Figure 3.22: Proposed OLSR Reservation Request message

- *Message Type*: Number that uniquely identifies the message type. An unused value should be used, 250 for example;
- *Vtime*: Validity time of the message (see [26] for further information);
- *Message Size*: Size of the message (in bytes);
- *Originator Address*: IP Address of the source node;
- *Destination Address*: IP Address of the destination node;
- *Required Bandwidth*: Bandwidth required for the reservation of a new flow (in bps);

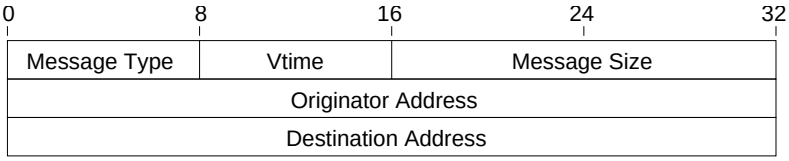


Figure 3.23: Proposed OLSR Reservation Reply message

Proposed Reservation Reply message:

- *Message Type*: Number that uniquely identifies the message type. An unused value should be used, 251 for a positive reply and 252 for a negative reply, for example;
- *Vtime*: Validity time of the message (see [26] for further information);
- *Message Size*: Size of the message (in bytes);
- *Originator Address*: IP Address of the source node;
- *Destination Address*: IP Address of the destination node;

3.6 Final Remarks

In this chapter we have described a bandwidth reservation scheme for ad-hoc networks that satisfies the following QoS constraint: "The load demand offered to the wireless media by the QoS traffic observed at any node in a path that is about to be established $\leq Q$". Parameter Q is dimensioned in a way that delays are acceptable for QoS connections. Our reservation scheme is designed for networks where nodes can communicate to neighbors using different transmission rates depending on channel conditions (multirate ad-hoc networks) and only requires that nodes know the normalized bandwidth reservation and maximum available bandwidth of their neighbors. These quantities can be easily advertised by means of HELLO packets. We also give a CAC rule that nodes should apply to new connections requiring QoS.

We have described how to integrate our reservation scheme with the OLSR routing protocol and we have implemented it using the ns-2 simulator. We have then simulated MR-AODV with and without our reservation scheme. The following items summarize our findings:

- Ad-hoc networks can easily become congested by QoS traffic (differently from TCP, this kind of traffic typically does not provide congestion control mechanisms).
- Congestion can easily extended to most of the network introducing high delays and losses, damaging, thus, most of the connections that requires QoS.
- Our reservation scheme provides a feasible way to avoid congestion, guaranteeing, thus, QoS requirements to ongoing connections.

3.7 Appendix A: Proof of Theorem A

Theorem A. *Guaranteeing that the AB given by equation 3.9 of every node that takes part in a reserved path is non-negative, is equivalent to guaranteeing that the MAB of every node of the QoS set ($\mathcal{S}$) is non-negative.*

Proof. The computation of the *MAB* of a given node takes into account only information about transmissions performed by the node and its 1-hop neighbors (see equation 3.7). Consequently, a transmission between two nodes only impacts the *MAB* of the 1-hop neighborhood of the sender.

By observing this, we can state that whenever a new real-time flow is established over the network, only nodes that takes part in the flow path and their 1-hop neighborhood are affected by these new transmissions. All the other nodes throughout the network see no changes on their *MAB*. Thus, if they were non-negative before the flow was established, they would remain like that afterwards.

Then:

$$\begin{aligned} AB_i \geq 0, \forall i \in resvd\ paths \Leftrightarrow \\ \min_{j \in \mathcal{N}_i^+} \{MAB_j\} \geq 0, \forall i \in resvd\ paths \Leftrightarrow \\ MAB_j \geq 0, \forall j \in \mathcal{N}_i^+, \forall i \in resvd\ paths \end{aligned} \tag{3.14}$$

So, considering that in the beginning all nodes of the QoS set have non-negative *MAB*s and that nodes that are not in the 1-hop neighborhood of reserved paths do not see any changes on their *MAB*, we can conclude that:

$$\begin{aligned} MAB_j \geq 0, \forall j \notin \mathcal{N}_i^+, \forall i \in resvd\ paths \Leftrightarrow \\ MAB_j \geq 0, \forall j \in \overline{\mathcal{N}_i^+}, \forall i \in resvd\ paths \end{aligned} \tag{3.15}$$

By using the results of 3.14 and 3.15:

$$\begin{aligned} MAB_j \geq 0, \forall j \in \left(\mathcal{N}_i^+ \cup \overline{\mathcal{N}_i^+}\right), \forall i \in resvd\ paths \Leftrightarrow \\ MAB_i \geq 0, \forall i \in \mathcal{S} \end{aligned} \tag{3.16}$$

So, as we were willing to demonstrate:

$$AB_i \geq 0, \forall i \in resvd\ paths \Leftrightarrow MAB_i \geq 0, \forall i \in \mathcal{S} \tag{3.17}$$

□

3.8 Appendix B: Proof that the CAC Guarantees the QoS Constraint

The use of the CAC proposed in equation 3.12 guarantees that the QoS constraint defined in equation 3.8 is respected in every condition. In fact, as we demonstrate below, it is guaranteed in all cases when using a simple sum approximation for the union operator.

Proof. As we have previously demonstrated, in order to guarantee the QoS constraint presented in equation 3.8, we can limit ourselves to guaranteeing the condition presented by equation 3.10. Thus, we just need to demonstrate that the proposed CAC guarantees that after accepting a new flow of r bps, every node in the flow path present a non-negative *AB*, i.e., all the nodes in the flow path and their 1-hop neighborhood belonging to the QoS set, present a non-negative *MAB*. Since we are only concerned with the nodes belonging to the QoS set, in the following we shall refer to only this set of nodes.

Nodes in the flow path: for a given node i in the path, we want to guarantee that its *MAB* is non-negative in the moment t_1 just after the acceptance of the new flow (t_0 represents the moment just before the acceptance).

$$
\begin{aligned}
&MAB_i(t_1) \geq 0 \Leftrightarrow \\
&Q_i - L_i(t_1) \geq 0 \Leftrightarrow \\
&Q_i - \sum_{y \in \mathcal{N}_i^+} X_y(t_1) \geq 0
\end{aligned}
$$

Since the only new transmissions from t_0 to t_1 are the ones due to the accepted flow, we have:

$$
Q_i - \underbrace{\sum_{y \in \mathcal{N}_i^+} X_y(t_0)}_{=L_i(t_0)} - \sum_{y \in \left(\mathcal{N}_i^+ \cap path\right)} \frac{r}{v_y} \geq 0
$$

We then have:

$$MAB_i(t_0) \geq \sum_{y \in \left(\mathcal{N}_i^+ \cap path\right)} \frac{r}{v_y} \tag{3.18}$$

So, in order to be correct, the CAC must guarantee equation 3.18. Our CAC, in simple sum approximation form, expressed by equation 3.12 guarantees the following condition:

$$AB_i(t_0) \geq \sum_{y \in \left(\left(\mathcal{N}_i^+ \cup \mathcal{N}_j^+\right) \cap path\right)} \frac{r}{v_y} \tag{3.19}$$

Since we know that $AB_i(t_0) \geq MAB_i(t_0)$, we may also say that the CAC guarantees that:

$$MAB_i(t_0) \geq \sum_{y \in \left(\left(\mathcal{N}_i^+ \cup \mathcal{N}_j^+\right) \cap path\right)} \frac{r}{v_y} \tag{3.20}$$

And finally, since equation 3.18 is more restrictive than equation 3.20 (note the additional terms in the former, as well as the sum over a more restricted set of nodes), we may say that the CAC satisfies the desired conditions.

Nodes in the 1-hop neighborhood of the flow path: There are basically two different cases that should be taken into account:

- Node n that is a neighbor of a node i in the path and all its other neighbors in the path are in the neighborhood of i or j (considering that i transmits to j).
- Node whose neighbors in the path are "more spread".

In the first case, we have that:

$$\begin{aligned} &MAB_n(t_1) \geq 0 \Leftrightarrow \\ &MAB_n(t_0) \geq \sum_{y \in \left(\mathcal{N}_n^+ \cap path\right)} \frac{r}{v_y} \end{aligned} \tag{3.21}$$

Since we know that

$$AB_i(t_0) \leq MAB_n(t_0)$$
$$\left(\mathcal{N}_n^+ \cap path\right) \subset \left(\left(\mathcal{N}_i^+ \cup \mathcal{N}_j^+\right) \cap path\right)$$

and since the CAC guarantees equation 3.19, we are also able to guarantee the condition expressed by equation 3.21.

Since transmissions that take place outside the neighborhood of the sender and the transmitter may overlap in time, the second case may, in fact, be seen as a combination of several non-correlated occurrences of the first case. So, if the CAC guarantees the QoS constraints for the first case, it will guarantee for the second as well.

□

Chapter 4

Introducing Mobility

4.1 Introduction

The model presented in the previous chapter is well suited for static networks. Once nodes start moving, their transmission rates to neighbors may vary and, consequently, their *AB* may vary as well. It is not difficult to realize that, in the presence of mobility, topology changes in the network may cause flows that were previously accepted by the CAC not to have their QoS requirements guaranteed after a while. Moreover, even if QoS can still be guaranteed over a given path, topology changes may cause more efficient paths to show up, and being able to take advantage of them may optimize the use of network resources. Consequently, when nodes move, the QoS mechanism should be made more adaptive. It should somehow capture the dynamic behavior of the network and reflect this into the management of the ongoing reservations.

Therefore, we propose modifying BRAWN to cope with mobility by doing a periodical refreshment of the QoS reservations, inspired on the route refreshment proposed by LUNAR (Lightweight Underlay Network Ad-hoc Routing Protocol) [27]. We shall refer to this extension of our protocol as *BRAWN-R* (BRAWN with Refreshments).

By doing that, the network will constantly re-validate the flows' admission control and search for better routes for the previously established connections. Most of the proposed reservation mechanisms for ad-hoc networks (see [52] and [72], for example) periodically check if the established route is still valid (i.e., if no link has broken along the way or if the end-to-end delay is too high). If the route is still up and providing acceptable end-to-end delays, nothing is done, otherwise they search for a new route, executing the CAC procedure once again. That may be a good solution for trying to guarantee the continuity

of the flows and a minimum QoS, however, besides being a bit complex to implement (due to the need of constantly monitoring the end-to-end delay of all ongoing connections), this kind of solution ignores the possibility of switching flows to better paths that may have shown up.

4.2 What should we do? Periodically refresh...

Based on these observations, we propose reservations to be guaranteed only for a given period. That means that once a new flow is accepted into the network, it is no longer stopped just in the case that some intermediate link breaks or if the end-to-end delay increases when using the chosen path. Instead, it is stopped if, after a given period of time, the network is no longer able to provide the required QoS through any possible path.

Inspired on the periodical route refreshment proposed by the LUNAR routing protocol (where every path is rebuilt from scratch every 3 secons, even if everything is fine with the current path), we propose that reservations have a time to live. After this period of time, the source node starts a new reservation process (in fact, a re-reservation). This process, very similar to the original reservation, is responsible for refreshing the flow's previous reservation and for guaranteeing its QoS requirements for a new period. By doing that, we periodically adapt the flow's reservation to the current state of the network without the need for additional reservation maintenance procedures and link repair actions, reducing the complexity of the protocol. If the current path can no longer support the previously agreed QoS requirements (due to the decrease of some link transmission rate, for example), a new path may be found for the flow. Moreover, if the path characteristics have not changed at all, but a new path that provides a better use of the network has become available, it may be used (a new node may show up between the source and the destination, for example).

However, in order to cause as little interference as possible in the flow transmission, a node should not be prevented to transmit while its reservation is being refreshed. It should also not compete with new flows that are trying to reserve resources for the first time. A node in the reservation refresh process, should have priority over the other ones. It should not happen that a node is trying to refresh its reservation and it is not only not able to find a better path, but it also looses its previously reserved resources to a new flow.

In order to cope with all these design requirements, we have extended the BRAWN protocol explained in chapter 3 such that the QoS flows are periodically refreshed. We shall

refer to it as BRAWN-R (BRAWN with Refreshments). In BRAWN-R, nodes that are in the path of a given flow trigger the following *Reservation Refresh Process* whenever the flow is about to timeout ($\Delta t_{refresh} = t_{timeout} - t_{refresh}$):

1. Subtract the normalized bandwidth used by this node for transmitting / forwarding the flow from X_i (total amount of normalized bandwidth transmitted / forwarded by this node to every neighbor). This new value, which we call $X_i^{refresh}$, should also be advertised to neighbors;

2. Compute the Maximum Available Bandwidth of a node using the $X_i^{refresh}$ values that were advertised by its neighbors as well as its own local $X_i^{refresh}$. This new value, called $MAB_i^{refresh}$, should also be advertised to neighbors;

3. Compute the Available Bandwidth of a node using these $MAB_i^{refresh}$. This new value is called $AB_i^{refresh}$ and should be used by the CAC for flows that are trying to refresh their reservations. For new flows, the CAC should keep using the original AB_i;

4. $\Delta t_{reserv} = t_{timeout} - t_{reserv}$ before the timeout of the flow, the source node starts a new reservation refresh process for this flow, signaling that this is an ongoing connection;

5. Nodes that receive the reservation request for this flow, compute the CAC using $AB_i^{refresh}$, since it does not take into account flows that are in the refresh process. Notice that at this moment, a new route is searched in the network. If the previously used route is still efficient, it has a good probability to be re-elected, otherwise a better path may be used. When receiving reservation requests from new flows, however the original AB_i is used as proposed by equation 3.12. By doing that, flows in the refresh process have more resources available for reservations than new flows and, thus, have a kind of priority over them for the reservation refresh over a new path (or over the same path that was previously being used);

6. When a flow reaches its timeout, its route is erased from the routing table of the nodes from the old path that do not take part in the recently elected path so that, if a flow has been re-routed to a more efficient path during the refresh process, intermediate nodes from the old path are able to release the corresponding resources.

Figure 4.1 shows a timeline that depicts the behavior of the mechanism.

In the example below (figure 4.2), flow 1 has originally been routed from node g to k through the path $g \rightarrow h \rightarrow i \rightarrow j \rightarrow k$. However, after a while, node i starts moving away

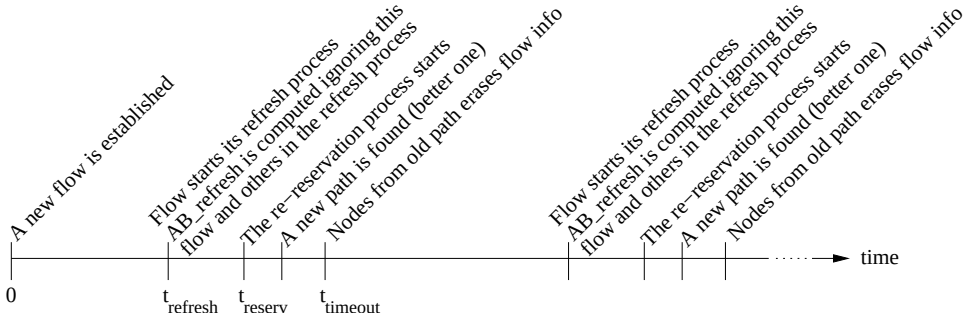


Figure 4.1: Timeline of the periodic refresh of the ongoing reservations

from the flow, being eventually out of the range of nodes h and j. Before that happens, during a refresh process, even if QoS requirements are still fulfilled, the route re-discovery procedure re-routes the path through n. This happens due to the fact that links $h \rightarrow i$ and $i \rightarrow j$ have their transmission rates decreased because of the increasing distance of node i and, at a given moment, routing through n may become more efficient. This reservation re-route through n anticipates a possible link break (which certainly occurs if node n keeps moving).

If, afterwards, node i moves back to its original position (between nodes h and j), although the QoS requirements may still be fulfilled by the path $g \rightarrow h \rightarrow n \rightarrow j \rightarrow k$, during the reservation refresh process, the flow takes advantage of the fact that node i showed up again between nodes h and j providing a more efficient link and changes its path to $g \rightarrow h \rightarrow n \rightarrow j \rightarrow k$, optimizing the use of the network resources.

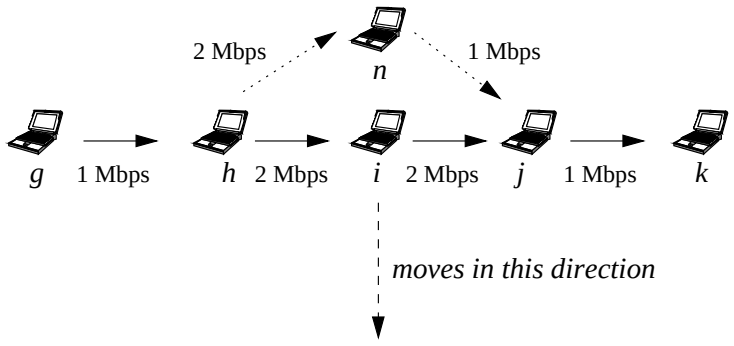


Figure 4.2: An example of a re-route (through node n) that occurs due to the movement of node i

Notice that in this example, even if another new flow 2 was trying to establish a connection through this network, flow 1 (that was previously established) would have its reservation guaranteed (at least on nodes g, h, j and k), even during the refresh process, due to the fact that its CAC uses $AB^{refresh}$, while flow 2 would use the original AB (which still takes flow 1 into account).

As we may see, our mechanism effectively re-routes flows through better throughput paths whenever they are available, what may not only optimize the use of the network, but also anticipate link breaks, providing the desired QoS for the dynamic ongoing flows.

4.3 Implementation Issues

Just like the original BRAWN protocol, BRAWN-R can be easily integrated into many routing protocols proposed for AWNs. In this section we explain how we have integrated it in both AODV and OLSR, providing, thus, an example of integration to a reactive protocol and to a proactive protocol, respectively.

4.3.1 Integrating into AODV

The CAC in BRAWN-R works in a very similar way to the original BRAWN protocol. The only difference is that whenever the node is dealing with a flow under the reservation refresh process, $AB_i^{refresh}$ should be used, while the original AB_i value should only be used for new reservations. Due to this similarity, every message that included BRAWN-related values, should now also include their $refresh$ equivalents. This means that:

(i) HELLO messages should not only advertise X_i, MAB_i, AB_i and AB_j ($\forall j \in \mathcal{N}_i$) but also $X_i^{refresh}$, $MAB_i^{refresh}$, $AB_i^{refresh}$ and $AB_j^{refresh}$ ($\forall j \in \mathcal{N}_i$).

(ii) Each node i collects these QoS HELLO messages from its neighbors to compute AB_i according to equation 3.9 and its equivalent $AB_i^{refresh}$.

(iv) Bandwidth reservation at intermediate nodes is done by adding the required bandwidth in the already existing RREQ and RREP messages. Moreover, every node through which the message passes appends its AB or $AB^{refresh}$ (so that the CAC may be correctly executed), depending on the verification of the state of the flow — if it is a new flow or if it is in the reservation refresh process.

Figures 4.3 and 4.4 show our proposal for the extensions of the HELLO and RREQ/RREP AODV messages respectively.

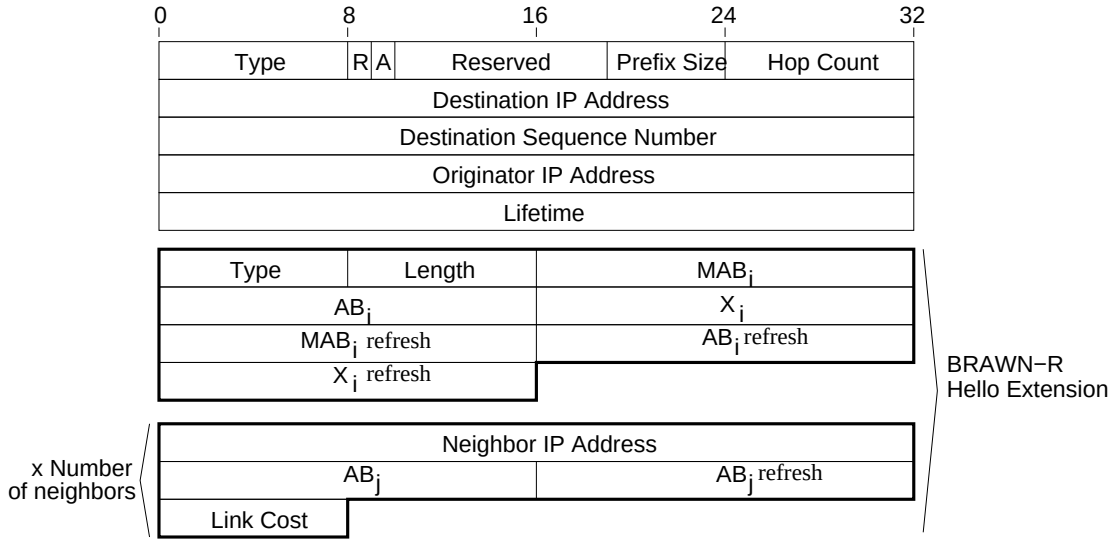


Figure 4.3: Extension proposed for the AODV HELLO message

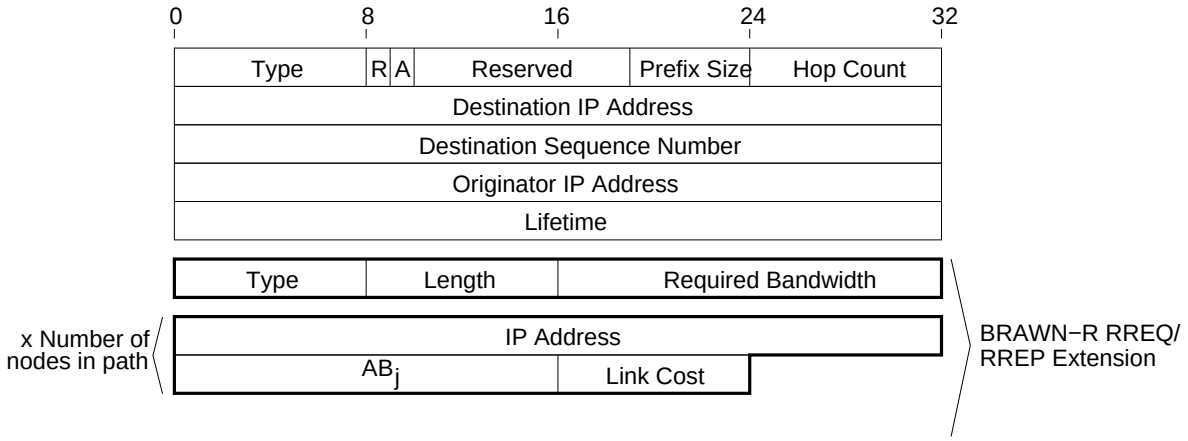


Figure 4.4: Extension proposed for the AODV RREQ and RREP messages

4.4 Simulation Results with AODV

We have added the described refresh mechanism into our BRAWN implementation and compared its results to the original BRAWN mechanism, where a reservation is only re-routed if a link break occurs. For all these simulations, we have used the parameters listed in table 2.2.

We have implemented BRAWN-R over the AODV routing protocol available in the ns-2 simulator [6] version 2.30 and launched simulations using the parameters listed in table 2.2. Results were compared to the BRAWN mechanism (without mobility support) and also to MR-AODV, where no reservations are made. The simulation code is available at `http://research.ac.upc.edu/CompNet/software/brawn`.

[1] According to the ORiNOCO 802.11b PC card specification for an open environment[1].

Table 4.1: Parameters used in simulations (ns-2 version 2.30)

Parameter	Value
MAC Protocol	802.11 with multirate
Propagation Model	Two Ray Ground
Transmission Rates	1, 2, 5.5 and 11 Mbps
Transmission Ranges	550, 400, 270 and 160 meters[1]
Carrier Sensing Range	1000 meters
Mobility Model	Random Waypoint
Simulation Time	600 seconds
Simulation Area	Square of 1500×1500 meters
$t_{refresh}$, t_{reserv}, $t_{timeout}$	50, 55 and 60 seconds

We have firstly simulated the scenario of the simple example previously explained (see figure 4.2). In this simulation, node i started moving down after 120 seconds of simulation. It keeps moving at 5 m/s for 120 seconds and it remains stopped out of the other nodes's range for another 120 seconds. Then, it moves back to its original position at 5 m/s, where it remains for the final 120 seconds.

As depicted by figure 4.5, when using the original BRAWN mechanism (with no refresh mechanism), the QoS reservation initially goes through node i, since it is part of the highest throughput path. However, as node i starts moving, it eventually gets out of range (that happens around 210 seconds). It takes a while for the routing protocol to be aware of the link break — the standard AODV link break detection mechanism is used, i.e., after three consecutive HELLO losses, the link is considered to be broken. After realizing the link break, BRAWN re-establishes the flow through the alternative path (through node n).

When using BRAWN-R, however, the source node periodically searches for the current highest throughput path with enough resources. By doing that, BRAWN-R is able to anticipate link breaks, re-routing the QoS flow before node i gets out of range. Figure 4.6 shows that around 170 seconds, the flow is re-routed through node n. At this moment, links $h \rightarrow i$ and $i \rightarrow j$ reduce their transmission rate to 1 Mbps (because of the increasing distance) and, due to that, the path through n becomes a better choice. Note that this happens much before the actual link break, which only takes place 40 seconds after.

The QoS flow is only re-routed through node i around 450 seconds. At this moment, links $h \rightarrow i$ and $i \rightarrow j$ increase their transmission rate back to 2 Mbps, making this path to be again a better choice over the one through node n.

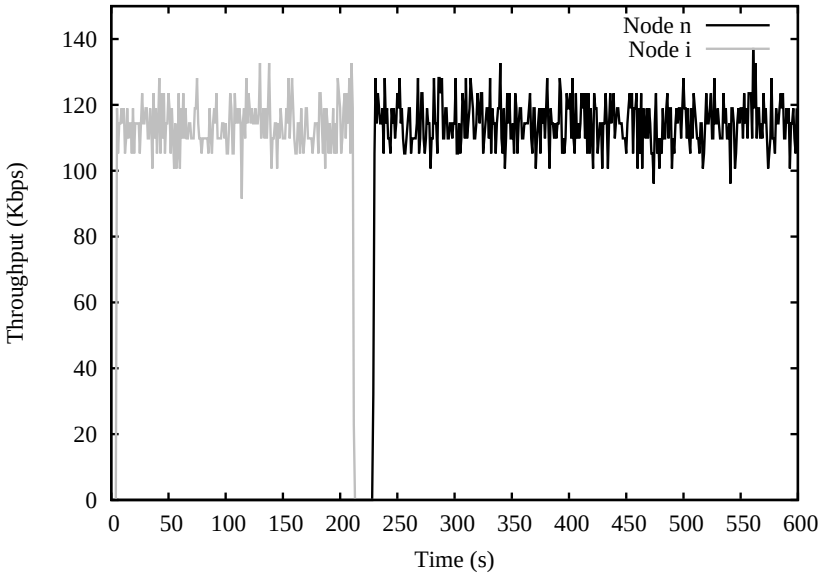


Figure 4.5: Throughput of nodes i and n when using the original BRAWN mechanism

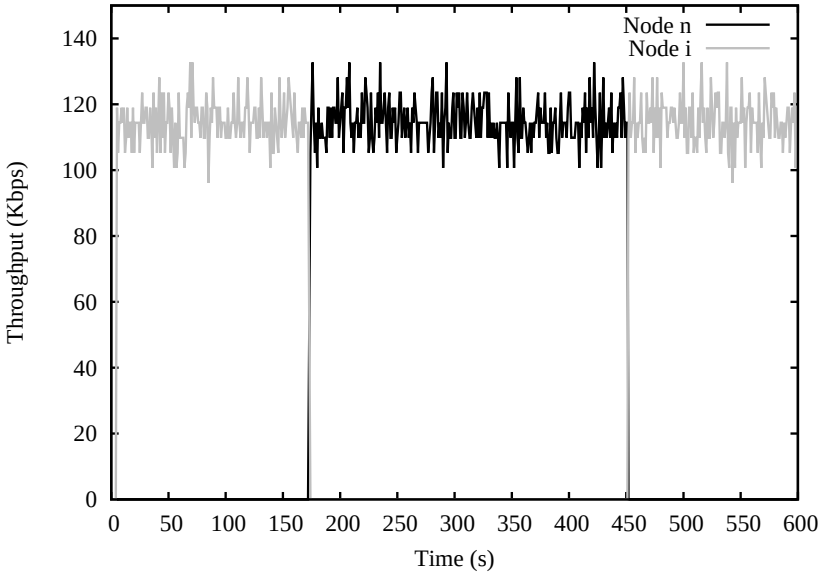


Figure 4.6: Throughput of nodes i and n when using BRAWN-R

Figure 4.7 confirms the results presented by the previous graphics. It shows that, exactly when there is a link break using the original BRAWN mechanism and until a new path is not found by the routing protocol, data packet are completely lost. When using BRAWN-R, the QoS flow is re-route before the link break occurs, avoiding data packet losses to take place.

By analyzing figure 4.8, that shows the end-to-end delay when using both mechanisms, it is possible to verify that, when using BRAWN-R, the QoS flow is re-routed exactly

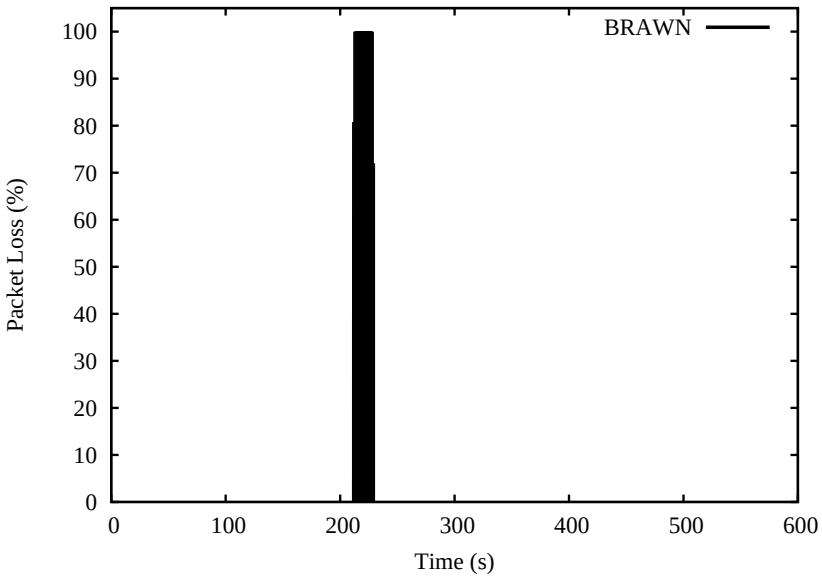


Figure 4.7: Packet losses when using the original BRAWN (when using BRAWN-R no losses where verified)

when links $h \rightarrow i$ and $i \rightarrow j$ decrease their transmission rate. This may be noticed by the fact that end-to-end delays get higher every time the link rate reduces so, when using the original BRAWN it increases to around 20 milliseconds at 170 seconds of simulation, while the use of the refresh mechanism makes the end-to-end delay remain around 18 milliseconds due the re-route of the flow through node n. When node i moves back to its original position, the delay decreases again when using the refresh mechanism, since the QoS flow is once again re-routed through i (the same does not happen when using the original BRAWN mechanism).

On a second scenario, we have randomly positioned 40 nodes that move according to a Random Waypoint model. We have established 20 CBR flows, of 32 Kbps each, among random pairs of nodes. We have varied the speed of the nodes from 0 m/s (non-moving nodes) to 10 m/s in order to capture the mechanism's behavior under different degrees of mobility. Finally, we have evaluated several performance aspects of the system. For each node speed, we have launched 20 different simulations using different initial node positions and different mobility patterns.

Figure 4.9 shows the total throughput of the network when using MR-AODV, BRAWN and BRAWN-R. One may notice that, when using MR-AODV, the network gets congested due to the absence of an admission control mechanism. Although the total throughput of the network is higher when using MR-AODV, all the other graphs show the low performance of the flows in this case. When using the original BRAWN mechanism, however, the network is able to provide quality of service. Moreover, due to the adaptable nature

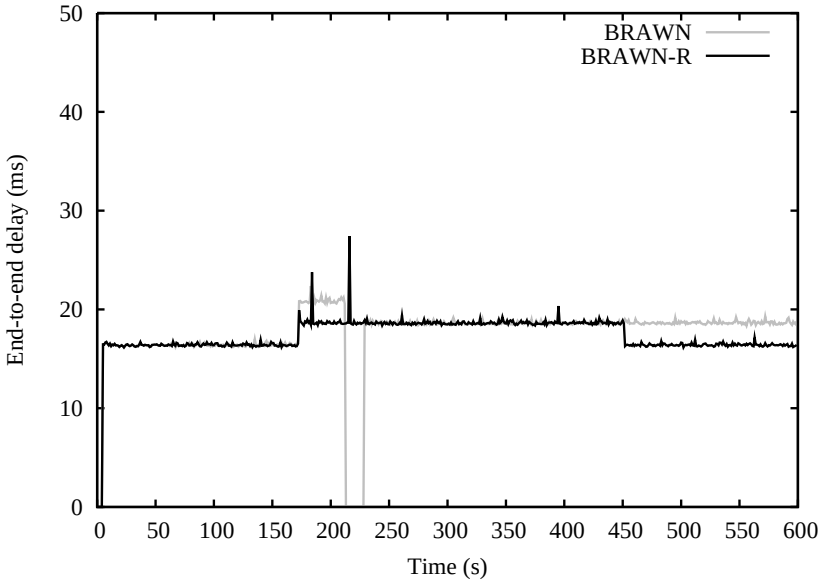


Figure 4.8: End-to-end delays when using the original BRAWN and when introducing the refresh mechanism

of the refresh mechanism used by BRAWN-R, it performs even better than BRAWN in mobile scenarios. The refresh mechanism constantly searches for better routes for the flows, allowing them to adapt more effectively to the network topology changes while still guaranteeing QoS.

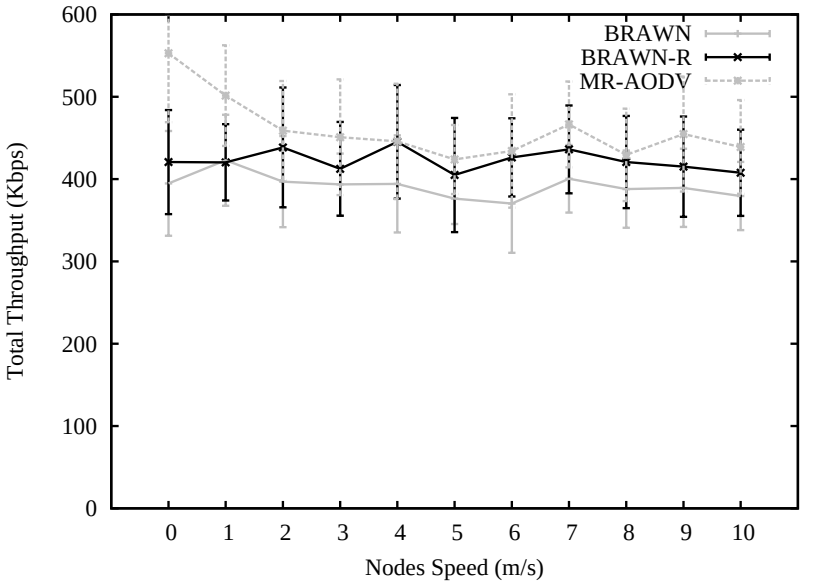


Figure 4.9: Throughput of the network when using BRAWN-R, BRAWN and MR-AODV

In fact, flows suffer due to the mobility of the nodes even when the refresh mechanism is used, since not always the QoS constraints can be guaranteed for the whole period of time between two consecutive reservation refreshes. However, Figure 4.10 shows that the

impact of the mobility is much lower when using BRAWN-R. Around 98% of the time the flows are able to transmit at at least 90% of the reserved rate (90% × 32 kbps = 28.8 kbps). With BRAWN, the amount of time at which at least 90% of the reservation is guaranteed is significantly lower, but yet much higher than with MR-AODV.

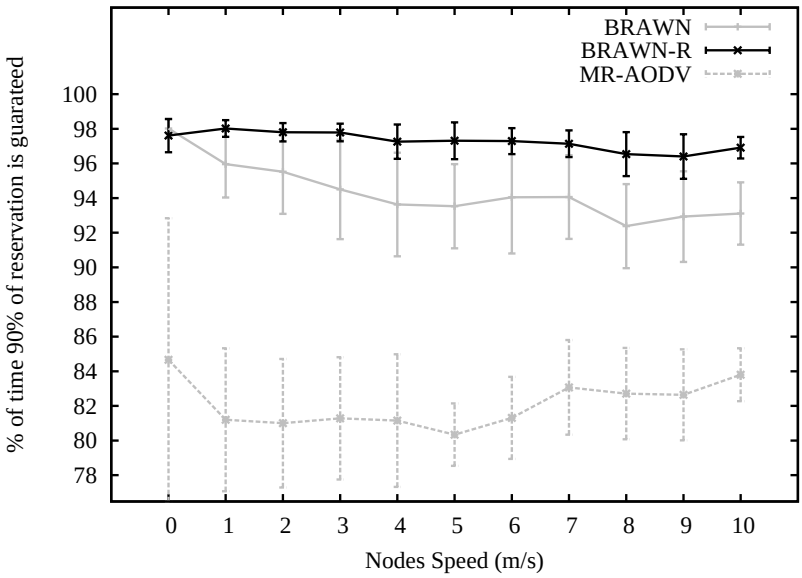


Figure 4.10: Percentage of time that at least 90% of the reserved bandwidth is being guaranteed for the reserved flows

Figure 4.11 confirms the inability of BRAWN to deal with the constant change of the network topology caused by the mobility of the nodes. With BRAWN-R, packet losses are significantly reduced, remaining below 2% when nodes move up to 4 m/s and reaching at most 5% for higher speeds scenarios. Using BRAWN, however, packets are lost more frequently, reaching a probability of 15% when nodes speed is 10 m/s. For MR-AODV, losses are much higher (up to 30%).

While the reduction of packet losses is a good indicative of the adaptability of BRAWN-R, it is also interesting to check the amount of packets that arrived at the destination after the maximum tolerable limit. For realtime multimedia applications, we have adopted a maximum tolerable end-to-end delay of 150 ms (as recommended by the ITU [2] for interactive multimedia applications). We may check that when using BRAWN-R, for node speeds up to 2 m/s, almost every packet arrive in time at their destination and the percentage of late packets remains below 5% for higher speed scenarios. When using BRAWN, however, the fact that the characteristics of the QoS route may change over time (being quickly unsuitable for the previously accepted reservation) makes a greater amount of packets to arrive late at their destination. This becomes more critical as the speed of the nodes increases (Figure 4.12).

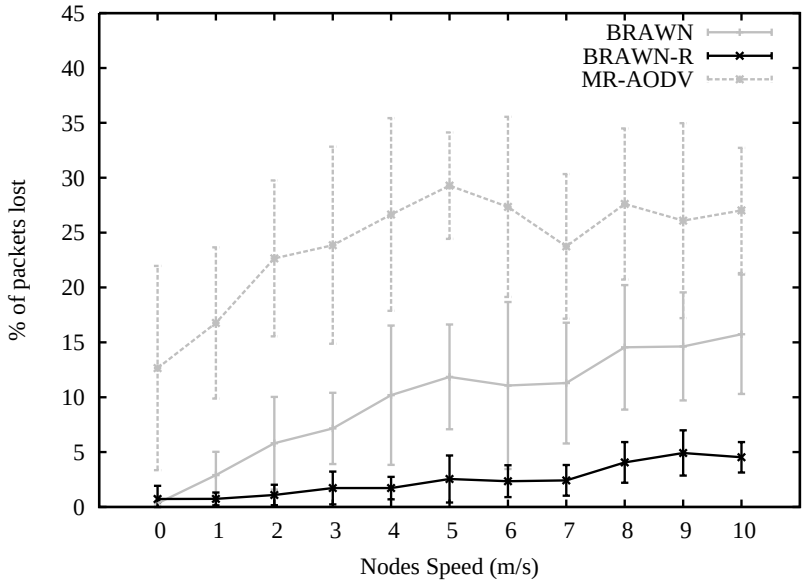


Figure 4.11: Percentage of packets lost when using BRAWN-R, BRAWN and MR-AODV

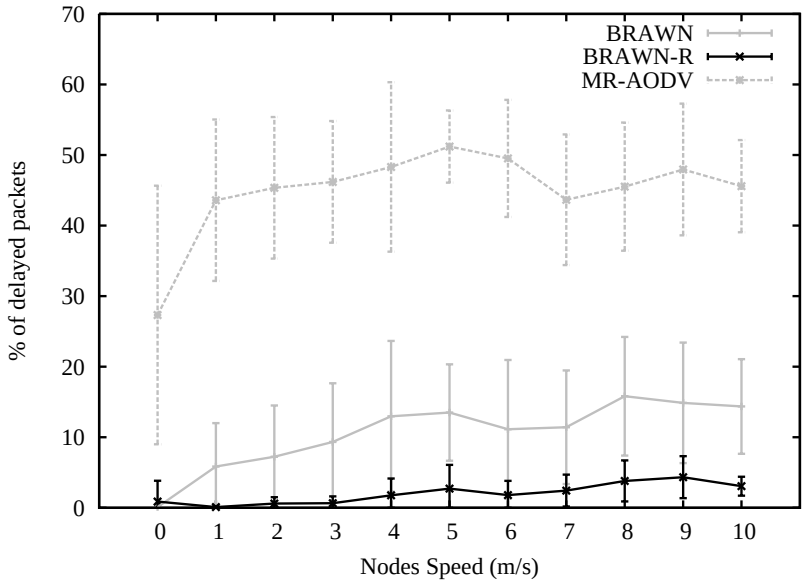


Figure 4.12: Percentage of packets which end-to-end delay is greater than the maximum tolerable limit (150ms)

Figures 4.13 and 4.14 show the signaling overhead of the three mechanisms (MR-AODV, BRAWN and BRAWN-R). One may notice that, although BRAWN-R signaling packets carry more information than BRAWN and MR-AODV signaling packets (see sections 4.3 and 4.3.1 for the specifications of the signaling packets), the overall signaling traffic impact of BRAWN-R is a bit lower than BRAWN and MR-AODV in most scenarios. When nodes do not move, both BRAWN-R and BRAWN introduce more signaling traffic than MR-AODV — one may get the same picture either if comparing the number of

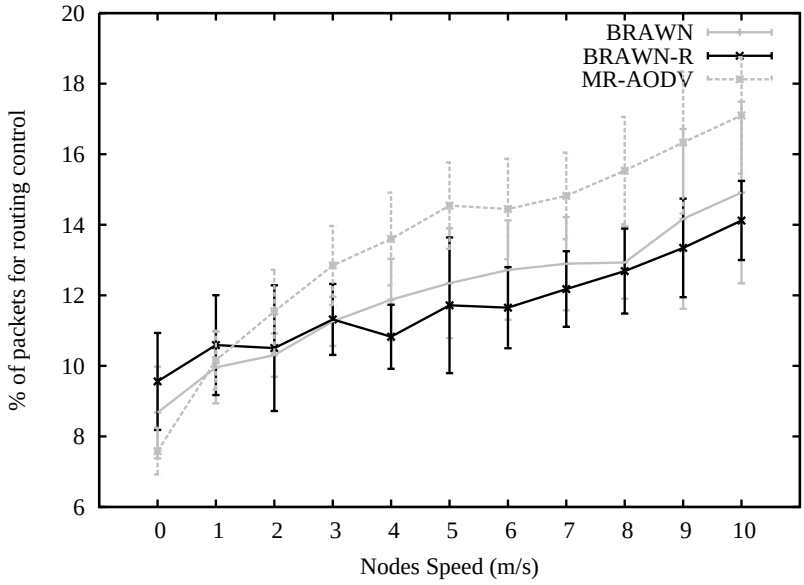


Figure 4.13: Percentage of packets in the network that are in fact routing signaling messages

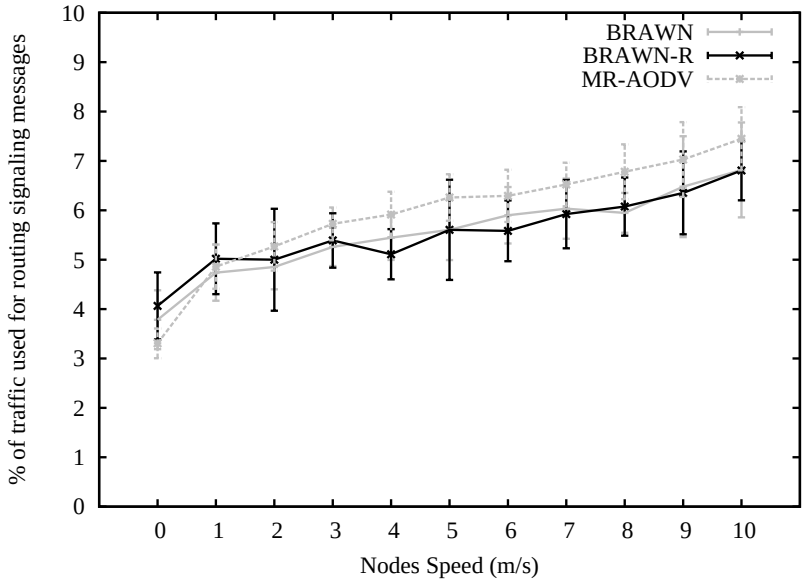


Figure 4.14: Percentage of the overall network traffic that is used for routing signaling messages

transmitted signaling packets or if comparing at the overall bandwidth consumed by these signaling packets. In one hand, when there is no mobility, both BRAWN and MR-AODV establish paths that are valid through all the simulation time, while BRAWN-R periodically send refresh messages, consuming more resources. In the other hand, when nodes move paths are eventually broken and new paths are frequently searched when using both BRAWN or MR-AODV. In these scenarios, although BRAWN-R makes use of larger

signaling packets, its periodical refreshes anticipates link breaks, avoiding additional path re-election mechanisms to be triggered. Whenever there is movement, the overall overhead introduced by BRAWN-R remains below the overhead introduced by both BRAWN or MR-AODV.

4.4.1 Integrating into OLSR

The integration of BRAWN-R to OLSR is quite straightforward, i.e., it may be easily performed by following the steps presented in the previous chapter for the original BRAWN mechanism (see section 3.5). Just like with AODV, the only difference between integrating BRAWN-R and BRAWN to OLSR is that the equivalent $refresh$ values should also be appended to the signaling messages. That means that:

(i) OLSR HELLO messages should be modified such that each node i with QoS reservations advertises not only X_i and MAB_i but also $X_i^{refresh}$ and $MAB_i^{refresh}$.

(ii) Each node i collects these QoS HELLO messages from their neighbors to compute AB_i according to equation 3.9 and its equivalent $AB_i^{refresh}$.

(iii) OLSR TC messages should be modified to advertise not only AB_i and v_{ij} of each of node i's MPR selectors, but also $AB_i^{refresh}$. By doing that, each node has knowledge of the network topology and the bandwidth available in the network for new flows and also for flows in the reservation refresh process.

(v) Bandwidth reservation at intermediate nodes is still done through the exchange of Reservation Request / Reservation Reply messages previously to sending data packets (see section 3.5).

Proposed extensions and new signaling messages format

According to the required modifications that were previously described, the new proposed formats for the HELLO and TC messages are depicted by figures 4.15 and 4.16 respectively. QoS information is carried by using the *Link Code* 251, which is not used by standard OLSR and is silently discarded by any node that does not recognize the code, i.e., any node that does not implement our proposed mobility-aware QoS extensions.

Furthermore, two additional signaling messages are proposed (just like for the original BRAWN mechanism): Reservation Request and Reservation Reply. The proposed formats for these two messages are depicted by figures 4.17 and 4.18.

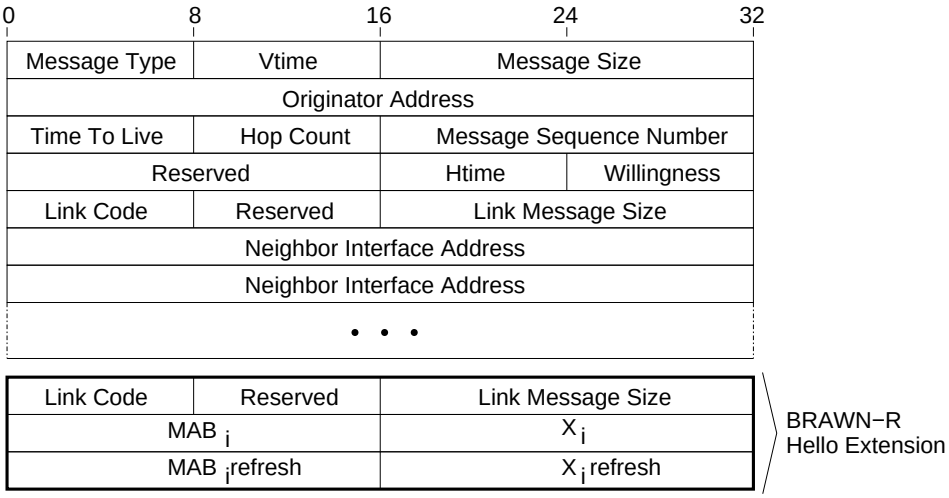


Figure 4.15: Extension proposed for the OLSR HELLO message

BRAWN-R Extension for the HELLO message:

- *Link Code*: Link Code of the OLSR extension. We may use any unused type number for the BRAWN-R HELLO extension, 251 for example;
- *Link Message Size*: Length of the OLSR extension;
- MAB_i: Normalized Maximum Available Bandwidth computed by the node for new flows (equation 3.7);
- $MAB_i^{refresh}$: Normalized Maximum Available Bandwidth computed by the node for flows in the reservation refresh process;
- X_i: Normalized sum of all traffic generated/forwarded by the node (equation 3.5);
- $X_i^{refresh}$: Normalized sum of all traffic generated/forwarded by the node subtracting the flows in the reservation refresh process;

BRAWN-R Extension for the TC message:

- AB_j, AB_k, etc: Normalized Available Bandwidth of all its MPR selectors for new flows;
- $AB_j^{refresh}, AB_k^{refresh}, etc$: Normalized Available Bandwidth of all its MPR selectors for flows in the reservation refresh process;
- X_j, X_k, etc: Normalized sum of all traffic generated/forwarded by all its MPR selectors;

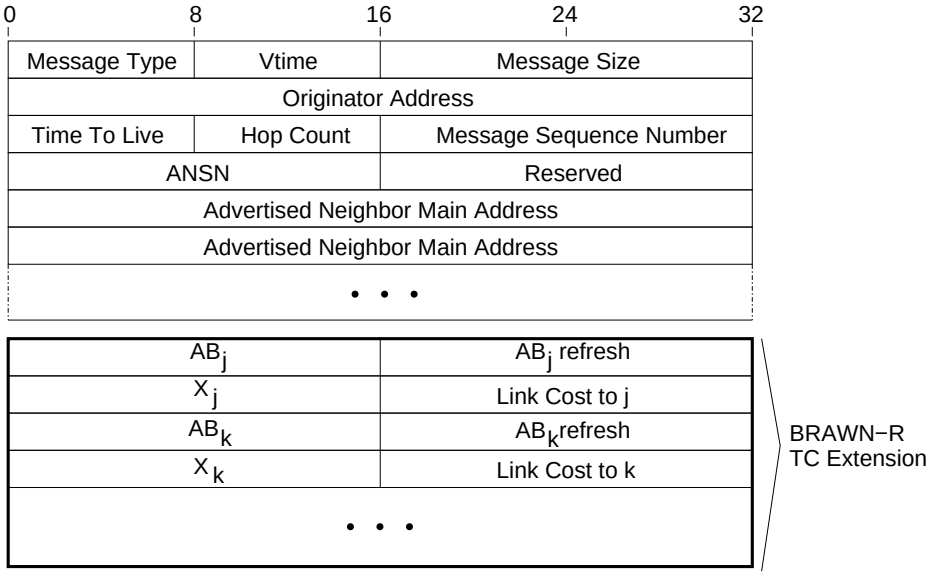


Figure 4.16: Extension proposed for the OLSR TC message

- $X_j^{refresh}, X_k^{refresh}, etc$: Normalized sum of all traffic generated/forwarded by all its MPR selectors subtracting the flows in the reservation refresh process;
- *Link Cost to j, k, etc*: Link cost from the node towards all its MPR selectors;

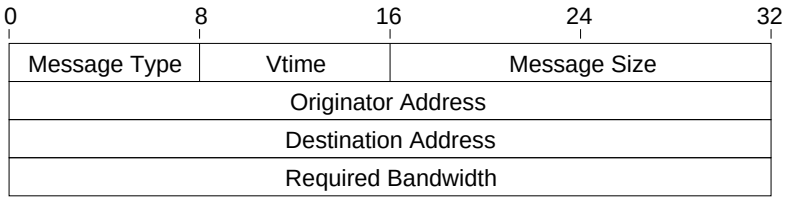


Figure 4.17: Proposed OLSR Reservation Request message

Proposed Reservation Request message:

- *Message Type*: Number that uniquely identifies the message type. An unused value should be used, 250 for example;
- *Vtime*: Validity time of the message (see [26] for further information);
- *Message Size*: Size of the message (in bytes);
- *Originator Address*: IP Address of the source node;
- *Destination Address*: IP Address of the destination node;
- *Required Bandwidth*: Bandwidth required for the reservation of a new flow or for the reservation refreshment of an ex(in bps);

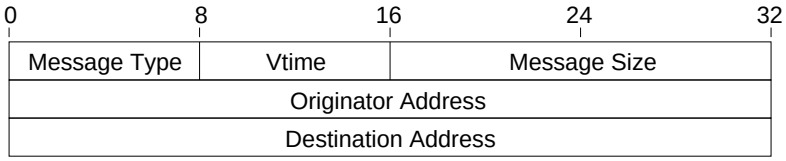


Figure 4.18: Proposed OLSR Reservation Reply message

Proposed Reservation Reply message:

- *Message Type*: Number that uniquely identifies the message type. An unused value should be used, 251 for a positive reply and 252 for a negative reply, for example;
- *Vtime*: Validity time of the message (see [26] for further information);
- *Message Size*: Size of the message (in bytes);
- *Originator Address*: IP Address of the source node;
- *Destination Address*: IP Address of the destination node;

4.5 Final Remarks

In this chapter we have dealt with the possible mobility of the nodes. Since the original BRAWN mechanism, proposed in chapter 3 was based on the establishment of reservations according to the instantaneous resources availability, changes in the network could cause previously established reservations not to be guaranteed after a while. This behavior usually lead to a low performance of BRAWN under the presence of mobility.

Due to this, we proposed BRAWN to periodically refresh reservations in order to take possible changes into account in the reservation mechanism. Flows in the periodical refreshing process (pre-established flows that are trying to renew their reservation) have priority over flows that are trying to establish a new reservation. By doing that, we are able to adapt our mechanism to the dynamic behavior of the network, anticipating link breaks and making a better use of the network resources. Simulation results confirm that the refreshing mechanism, which we have called BRAWN-R, improves the adaptiveness of the protocol, reducing packet losses and delays with a low overhead cost.

Chapter 5

A Prototype for BRAWN

5.1 The context

A simplified version of the reservation mechanism proposed in this work was implemented for the European Project WIDENS (WIreless DEployable Network System) [7].

WIDENS was a cooperative project involving European industries and universities that was supported by the European Commission under the IST Framework Programme 6. The overall objective of the project was to design, prototype and validate a high data-rate, rapidly deployable and scalable wireless ad-hoc communication system with QoS support for future public safety, emergency and disaster applications.

In order to attend all these requirements, the project proposed a system for an easily deployable IP ad-hoc wireless network in the absence of infrastructure, that used some of the well known wireless networks standards and proposed adaptations and changes to: *(i)* better adequate them to the typical scenarios of this kind of networks, and *(ii)* introduce QoS support. The project also intended to disseminate its results to the Mobility for Emergency and Safety Applications (MESA) standardization project [4].

On this kind of scenarios there is a great necessity to support many different types of applications: from file transfers and database queries, where there are no QoS requirements, to videoconferences, audioconferences and video surveillance, that are very sensible to delays and jitters, and that usually need a guaranteed minimum bandwidth to properly work.

In order to support all the QoS requirements of such applications, the WIDENS ad-hoc network is composed of nodes that implement the following elements:

- A DLC/MAC/PHY based on 802.11 that provides reliable communication mechanisms with quality of service;
- A modified version of the OLSR routing protocol that supports the election of routes based on the BRAWN mechanism;
- Network management components, that allow the configuration of the node for each particular operation;
- Security components, that guarantees the confidentiality, authenticity and reliability of the information and the robustness of the system;
- Group applications and services that would be used by rescue teams during operations.

The MAC layer that was developed for the project is based on the 802.11e, 3GPP and HiperLAN/2 standards, in order to provide QoS. The network is organized in clusters, each of them coordinated by a special node called "cluster head". This node manages the cluster resources, assigning transmission opportunities for every node under its control, i.e. every node in the cluster. By doing that, it is possible to guarantee a deterministic QoS in transmissions.

On the top of this MAC layer, we have implemented the BRAWN mechanism, by integrating it into the OLSR protocol. In this chapter, we briefly describe the PHY/MAC on the top of which the BRAWN mechanism was deployed and then describe the QoS mechanism implementation, the design choices, used tools and the protocol formal specification through the use of State Machines and Message Sequence Charts (MSCs).

5.2 A Brief Introduction to the WIDENS PHY/MAC

One of the goals of the WIDENS project was the development of new QoS-aware physical (PHY) and medium access control (MAC) layers, related to the research activities carried out in it [18].

The WIDENS MAC looks like an enhanced 802.11e [11]. The MAC is time-slotted and synchronized by special nodes, called *Cluster Heads* (CH), which play the role of the hybrid coordinator in 802.11e networks. Nodes associate themselves with the cluster head after synchronization. They are able to associate with more than one cluster head, allowing the interconnection of clusters through these so-called *relay nodes*.

The role of the cluster head is to regulate traffic within the cluster by scheduling transmission opportunities based on traffic volume measurements signaled by terminals and on network layer QoS reservations. When a WIDENS terminode wants to reserve bandwidth towards another node, it issues a layer 3 reservation request (through an application API, which will be detailed later) and this request eventually reaches the MAC layer. Layer 2 then negotiates with the cluster head the reservation of the requested resources in the cluster. The cluster head answers with a layer 2 reservation reply. From then, the cluster head issues transmission opportunities to the WIDENS terminode to accomplish the aforementioned QoS reservation. During transmission opportunities, terminodes can schedule their traffic queues over physical layer resources using reconfigurable scheduling policies that satisfy different QoS scenarios and based on wideband channel measurements with respect to their destinations.

The software/hardware architecture for the WIDENS prototype is based on the proven PC-based real-time software radio architecture, used e.g. by the FP5 IST Mobydick [5] platform. In the WIDENS implementation, the nodes consist of:

- 5MHz channels, TDD RF front-Real-time data acquisition system;
- Real-time software (RTLinux) development environment for fully-reconfigurable PHY/MAC;
- Dual-antenna (TX-RX) capability;
- IPv4/IPv6 interconnect;
- WLAN interoperability (using commercial Wi-Fi hardware).

5.3 The Prototype

In order to implement the QoS features that an implementation of the BRAWN mechanism should offer, we have develop several modules, with very specific functionalities, that work together (figure 5.1).

The Reservation Module, for example, is responsible for managing every reservation request and the release of resources when they are no longer being used. This module directly communicates with the routing protocol (a modified version of OLSR) that, together with the CAC mechanism, verifies the availability of a route towards the destination that may provide the required QoS. Once a route is found, a flow identifier is assigned to this reservation and the protocol is started in order to confirm the resource reservations in every node along the elected path.

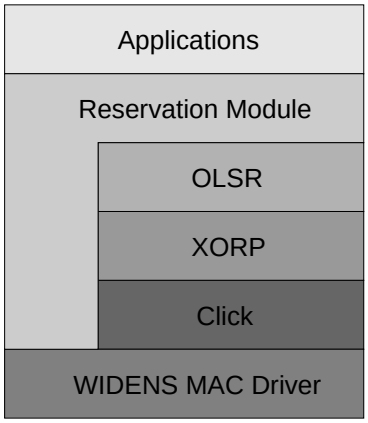


Figure 5.1: Main components of a WIDENS terminode

After the establishment of the path, all the forwarding is done based on the pair (source IP address, flow identifier), which must be unique in the entire network. Then, OLSR installs a route in the forwarding tables, which were implemented using the Click tool [51] (as discussed in the next section), and every packet generated by the application towards the desired destination is tagged with this flow identifier.

In the following section we will describe each module of the WIDENS node network layer, focusing on their main characteristics and justifying the election of the tools that were used for the implementation.

5.3.1 Click Modular Router

The Click Modular Router tool [51] is a platform for the rapid and flexible development of routers based on a set of modules that are called "elements". These elements may be seen as objects with very specific packet processing functions: queues, classifiers, generators, etc.

The idea behind Click is to build a router through a chain of elements that determines the path that the packet will follow inside the node. The graph that defines the connection between elements is obtained through a configuration file.

The original Click system shared the Linux interruption structure and device manager, and the overhead introduced by these mechanisms limited the system overall performance, consuming around 5 microseconds from each 13 microseconds required for processing a packet in a 700Mhz Pentium III [51]. However, although this should not be a problem for wireless systems due to the low transmission rates involved, for higher rate links Click may replace the interruption mechanism by another one based on polling, which was

shown to be much more efficient. So, although using a high level programming language for the construction of a router using Click, the global efficiency of the system is not an issue and, in some cases, it may even behave better than the traditional Linux forwarding mechanism.

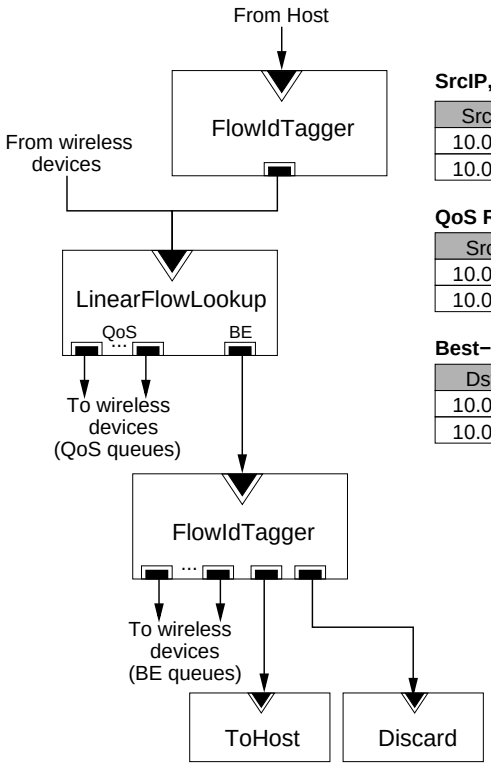
Usually, the processing of a packet starts in a special element that generates or stores it, passes through a series of elements and connections and ends in another element that consumes or stores packets. We may classify Click elements in groups according to their functionalities:

- *Packet sources*: responsible for generating packets, getting them from the network (FromDevice), from a file (FromDump), creating them from specific data (InfiniteSource, RatedSource) or creating them from random data (RandomSource);
- *Packet destinations*: eliminate packets from the system, by dropping them (Discard, TimedSink) or sending them to the network (ToDevice) or to a file (ToDump);
- *Packet modifiers*: change packet data (DecIPTTL, SetIPAddress, SetRandIPAddress);
- *Packet verifiers*: keep statistics about packets (Counter), or verify their integrity (CheckLength, CheckIPHeader);
- *Forwarding elements*: choose where packets should go to, based on forwarding algorithms that do not depend on the packets (Switch, RoundRobinSwitch), that depend on general characteristics of the packet flow (Meter, PacketMeter) or on the verification of the packets content (Classifier, HashSwitch, LookupIPRoute);
- *Storage elements*: store packets in the memory for using them afterwards (Queue, FrontDropQueue);
- *Scheduling elements*: select packets from one or more possible packet sources (RoundRobinSche, PrioSched);
- *Information elements*: implement language extensions (AddressInfo, ScheduleInfo) or interact with the out-of-band configuration (ControlSocket).

Each Click element is one subclass of the *Element* C++ class, which has about 20 virtual functions. *Element* provides default implementations for many of these functions, so that a great part of the subclasses only overload a few of them. Once all necessary elements

are implemented, one may specify the router through the programming/configuration language offered by Click.

However, although Click provides a rich elements library, for implementing the BRAWN mechanism we had to create an additional set of elements that allowed us to forward packets based on their QoS flow identifiers. The following elements were implemented to fulfill our specific needs. Figure 5.2 shows a graphical simplified configuration for the QoS packet forwarding, focusing on the key elements, while 5.3 shows the equivalent textual configuration.



SrcIP,SrcPort,DstIP,DstPort,Protocol –> FlowID

SrcIP	SrcPort	DstIP	DstPort	Protocol	FlowID
10.0.0.1	19003	10.0.0.3	6060	UDP	30
10.0.0.1	19007	10.0.0.5	3023	TCP	31

QoS Routing Table

SrcIP	FlowID	NextHop	Output
10.0.0.1	30	10.0.0.2	1
10.0.0.7	7	10.0.0.11	2

Best−Effort (BE) Routing Table

DstIP	Prefix	NextHop	Output
10.0.0.3	32	10.0.0.6	1
10.0.0.5	32	10.0.0.2	2

Figure 5.2: Simplified view of the Click elements configuration in each node

- *FlowIdTagger* (packet modifier): adds the correct flow identifier into every packet generated by applications that have requested bandwidth reservation for QoS communication (figure 5.4). Based on a table, the element maps source and destination IP addresses as well as source and destination ports and the used transport protocol into a 20 bit flow identifier (for being compatible with the 20 bit Flow Label field of the IPv6 header [59]). The entries in this table are updated by the routing protocol through a control socket whenever a new QoS route is added. The only packets that are processed by this element are the ones generated by the node, i.e. packets that arrive from upper levels and need an identifier to be correctly forwarded. Packets

```
KernelTun(10.0.0.1/32, MTU 1500) // Get packets generated by the node
   -> FlowIdTagger()             // Tag QoS packets with their flow ID
   -> _flow_rt;                  // Send them to the QoS routing table

FromDevice(wifi0_0) -> (...)     // Get packets received by device wifi0_0
   -> _flow_rt;                  // Then send to the QoS routing table
FromDevice(wifi0_1) -> (...)     // Get packets received by device wifi0_1
   -> _flow_rt;                  // Then send to the QoS routing table
FromDevice(wifi0_2) -> (...)     // Get packets received by device wifi0_2
   -> _flow_rt;                  // Then send to the QoS routing table
FromDevice(wifi0_3) -> (...)     // Get packets received by device wifi0_3
   -> _flow_rt;                  // Then send to the QoS routing table

// Routing tables (search routing entry and modify packet's next hop)
_flow_rt :: LinearFlowLookup();   // The QoS routing table (flow based)
_standard_rt :: LinearIPLookup(); // Standard routing table (best-effort)

// QoS Routing table outputs
_flow_rt[0] -> _standard_rt; // Output 0: no route found, goto best-effort
_flow_rt[1] -> (...)         // Output 1: send to device wifi0_0
   -> ToDevice(wifi0_1);     //      (QoS queue 1 of device wifi0)
_flow_rt[2] -> (...)         // Output 2: send to device wifi0_1
   -> ToDevice(wifi0_2);     //      (QoS queue 2 of device wifi0)
_flow_rt[3] -> (...)         // Output 3: send to device wifi0_2
   -> ToDevice(wifi0_3);     //      (QoS queue 3 of device wifi0)
_flow_rt[4] -> (...)         // Output 4: send to device wifi0_3
   -> ToDevice(wifi0_4);     //      (QoS queue 4 of device wifi0)
_flow_rt[5] -> (...)         // Output 5: destination is the host itself
   -> ToHost;

// Best-effort Routing table outputs
_standard_rt[0] ->  -> (...) // Output 0: send to device wifi0_0
   -> ToDevice(wifi0_0);     //      (Best-effort queue of device wifi0)
_standard_rt[1] -> (...)     // Output 1: destination is the host itself
   -> ToHost;
_standard_rt[2] -> Discard;  // Output 2: destination unknown
```

Figure 5.3: Simplified textual version of the Click elements configuration in each node

that do not match to any entry of the mapping table are not modified and are transmitted as best-effort.

- *LinearFlowLookup* (forwarding element): after a packet is generated and passes through FlowIdTagger, it must be forwarded to the next hop of its route towards the destination. The forwading is based on the pair (source IP address, flow identifier), that uniquely identifies a flow in the network. By searching a forwarding entry

in this table, LinearFlowLookup transmits the packet through one of its outputs (each output is connected to a different output device, which may represent different network interfaces or different queues of a single interface for different service classes). This element also presents a best-effort output for those packets that do not have a flow identifier. These packets are delivered to another element (a Click standard element) that forwards them to the best-effort queues of the network interfaces. The same procedure also applies to packets that are received by the node (by the lower layers) and that must be forwarded for the destination node.

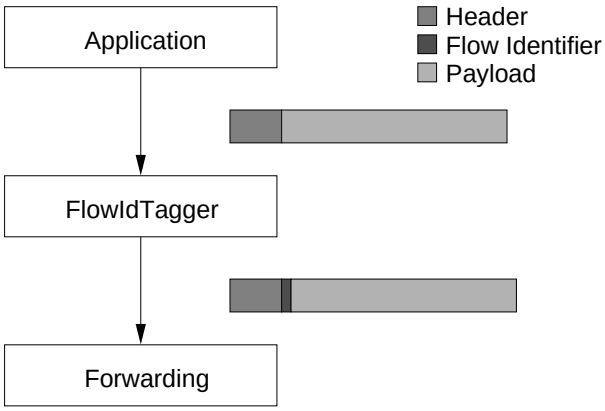


Figure 5.4: Adding a flow identifier into data packets

The use of Click has significantly simplified the implementation process of the forwarding mechanism, since implementing the elements is much easier than making the equivalent changes in the standard Linux IP stack.

5.4 The Reservation Module

On the top of the previously described modules (that compose the forwarding plane), we have implemented a module responsible for managing the reservations that are requested by applications. This module provides an API (table 5.1) that allows applications to reserve and release network resources for a given flow. This API is presented in the form of a C library that should be linked with the application code (see the complete documentation of the API in Appendix A).

In order to avoid that resources assigned to flows that no longer exist remain reserved, besides offering the possibility to explicitly release resources, the reservation module constantly monitors applications that required a reservation so that, if a flow connection is closed, resources are automatically released.

Table 5.1: Reservation Module API description

Function	Description
widens SetReservation	Allows the establishment of a new reservation. The application must provide the destination IP address, source and destination ports, transport protocol, socket descriptor that will be used and required bandwidth. As a result, the function returns whether it was possible to make a reservation or not. If it was possible, the application gets a session identifier, otherwise it gets an error description.
widens SetReservation AndOpenSocket	Reserves resources and opens a new socket for transmitting data related to the reservation, at once. The only difference from the previous function is that it is not needed to inform the source port.
widens ReleaseReservation	Releases a previously established reservation. The session identifier must be informed.
widens RegisterCallback	Registers an application's callback function, so that it may called whenever something happens with one of its reservations. By doing that, the application is informed about reservations that are released due to the fact that resources are no longer available, for example.

In order to offer these features for the applications, we have implemented the BRAWN signaling and Connection Admission Control mechanisms. Although this module is independent from the routing protocol (OLSR, in this prototype), there is an intense collaboration between them, since all the reservation process depends on the chosen path between the source and the destination nodes. For this reason, the communication between these modules is very frequent and it is basically done through the exchange of four messages:

- *getRoute*: gets a route towards the desired destination that accomplishes the requested QoS requirements);
- *getAlternativeRoute*: does the same after trying to reserve a path and failing;
- *removeRoute*: informs the routing protocol that a given reservation was released;
- *linkDown*: used by the routing protocol to inform the reservation module that a given link went down.

For implementing the reservation establishment control, we have specified some messages that are exchanged between nodes that will take part in a path between the source and the destination (table 5.2).

Table 5.2: Reservation protocol messages

Message	Description
L3ReservationRequest	Requests the establishment of a reservation to the next node in the path towards the destination. Informs all QoS requirements.
L3ReservationReply	When the reservation request arrives in the destination node, it confirms the reservation and sends this message back to the source. If an intermediate node has not enough resources, it may send this message to the source refusing the reservation.
L3ReleaseRequest	Requests the release of the reservation to the next node in the path towards the destination.
L3Failure	If a node does not have enough resources anymore or if a link has broken, this message is sent back to the source.
L3Refresh	Whenever there is no traffic for a given flow, the source node sends signaling packets for keeping the reservation alive, since after a while without receiving packets for a given flow, the nodes release its reservation.

These messages are exchanged between the nodes that take part in a QoS route, allowing the flow-based routing/forwarding offered by the other modules (OLSR and Click respectively) to work properly.

Finally, in order to have a complete understanding of the dynamic behavior of the reservation module, we have specified state diagrams for the source node (figure 5.5), the intermediate nodes (figure 5.6) and the destination node of a flow (figure 5.7). These diagrams, together with the message sequence charts presented in the next section, were the basis for the C++ implementation of this module. Transitions in the diagrams are represented by an arrow and a pair of values A/B, where A is the event that triggered the transition and B is the signaling produced by the transition.

In the source node, initially there is no reservation established (the node is in the $RESERV\_CLOSED$ state). Whenever it receives a *SetReservation* request from the upper layers (*widensSetReservation* or *widensSetReservationAndOpenSocket* API call), it searches for a suitable path with the help of the routing protocol (OLSR) and sends an *L3Reservation Request* to the next node in the path, if the CAC admits this new flow. The node enters in the $RESERV\_SENT$ state, until it receives an *L3ReservationReply* that confirms the reservation, what causes the node to enter the $RESERV\_OK$ state. If, instead, it did not receive a reply in time, it would increment the Request Counter (*sequence_request*) and

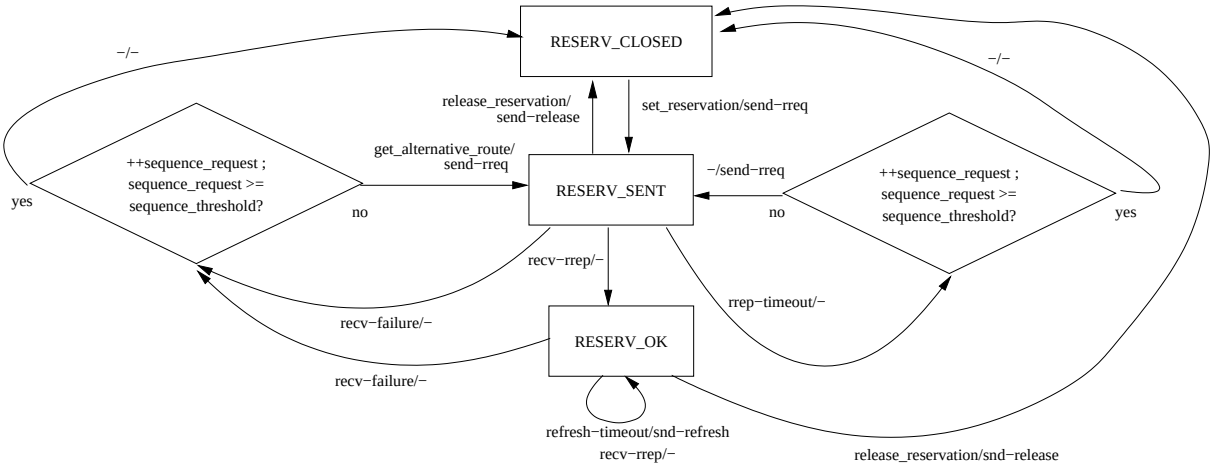


Figure 5.5: State diagram of a source node

send another *L3ReservationRequest*. At any moment that the node receives a negative *L3ReservationReply* or an $L3Failure$, it increments the Request Counter and tries to establish a route through an alternative path. If the Request Counter surpasses a given threshold (*request_threshold*), the node gives up trying to establish a reservation for this flow, alerts the application and goes back to the $RESERV\_CLOSED$ state. Also, at any moment that the node receives a *release_reservation* signal from upper layers (through the *widensReleaseReservation* API call), it sends and *L3ReleaseRequest* and goes to the $RESERV\_CLOSED$ state, releasing all previously reserved resources.

Intermediate nodes have a similar behavior. However, besides the fact that these nodes should also forward every received message, they should periodically receive an $L3Refresh$ signal, in order to stay in the $RESERV\_OK$ state. Moreover, there is an additional state, called $RESERV\_FAILURE$, in which the node stays for a while after the reservation fails (due to a timeout or explicit reception of an $L3Failure$ message). In this state, if the node receives QoS packets to forward, or $L3Refresh$, it sends back an $L3Failure$, informing the previous nodes that the reservation was released due to a failure. If it receives an *L3ReservationRequest*, however, it re-starts the reservation procedure, forwarding the request and going to the $RESERV\_SENT$ state.

Finally, the destination node has only two states: $RESERV\_CLOSED$ and $RESERV\_OK$. Whenever it receives an *L3ReservationRequest* and the CAC admits the new flow, it goes from the $RESERV\_CLOSED$ state to $RESERV\_OK$. Whenever the reservation times-out or upon receiving an *L3ReleaseRequest*, the node goes back to the $RESERV\_CLOSED$ state.

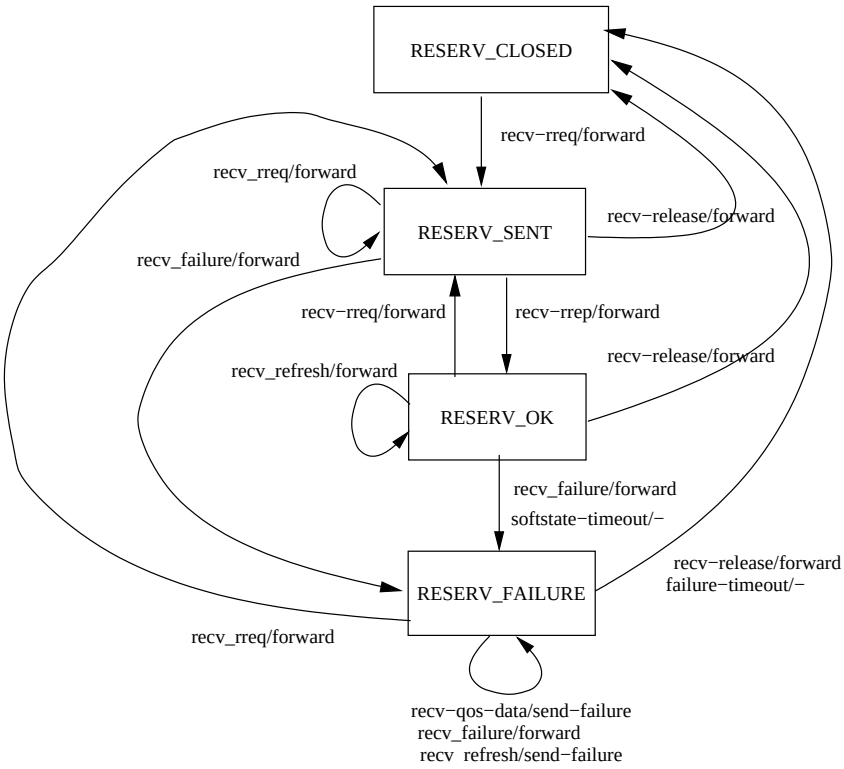


Figure 5.6: State diagram of an intermediate node

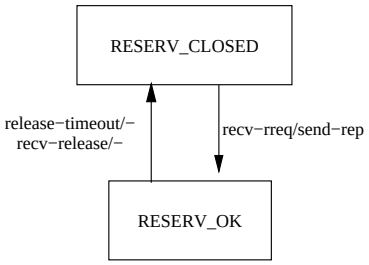


Figure 5.7: State diagram of a destination node

5.5 The Protocol Specification

The state diagrams presented above provide us the basic behavior of each node that takes part in one QoS reservation. However, in order to have a better understanding of the interaction among the several modules that compose the network layer of a WIDENS terminode and their interaction with the MAC layer, we present a set of Message Sequence Charts (MSCs). These MSCs describe the overall behavior of the protocol implementation.

In the following MSCs we refer to the aforementioned modules as:

L5 is the application layer;

RvSgM is the Reservation Signaling Module (or simply Reservation Module) which is the module responsible for managing all reservations. It provides an API for applications to make reservations;

RtM is the Routing Module, in this case a modified implementation of the OLSR protocol;

CAC is the Connection Admission Control module, it works tightly coupled with routing;

L2 is the MAC layer manager module, responsible for the lower layer reservation;

FP is the forwarding plane, implemented by using the Click Modular Router tool.

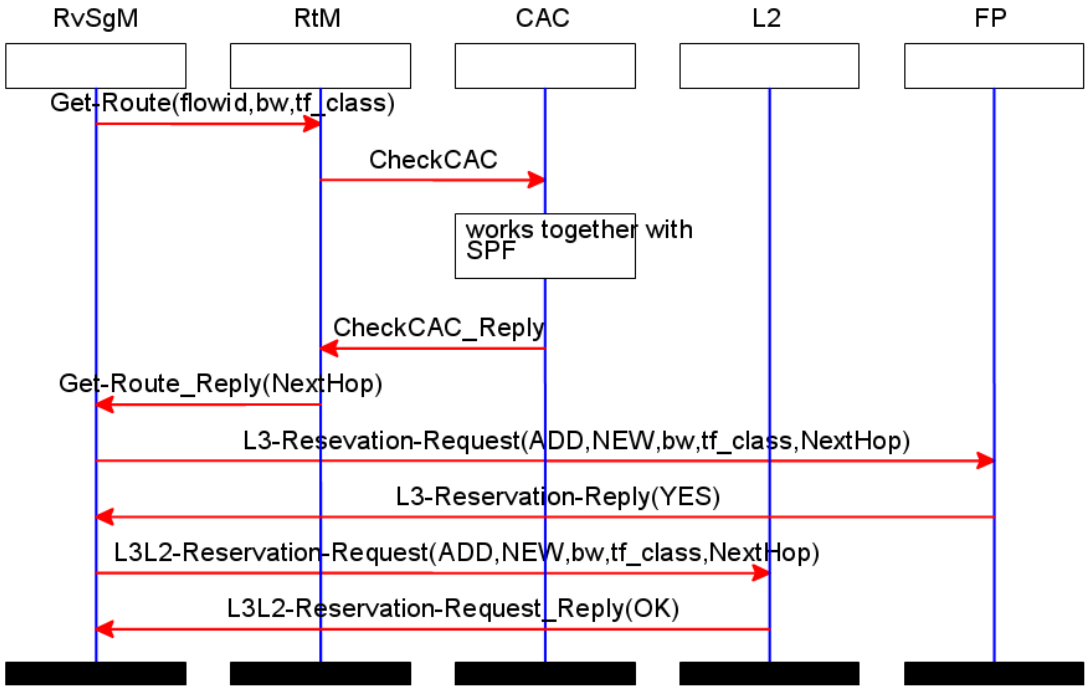


Figure 5.8: Reservation procedure in one node when everything goes right

Whenever an application needs to establish a reservation for realtime flows, it requests the Reservation Module through the previously described API. The Reservation Module then manages the whole reservation process. Initially, it communicates directly with the routing protocol, which, in turn, with the aid of the Call Admission Control mechanism, verifies the availability of a route towards the destination that is able to satisfy the QoS requirements. Once OLSR finds a QoS route, a flow identifier is assigned to the connection by the Reservation Module. Then, the Reservation Module confirms the reservation with the local MAC layer, installs a new entry in the forwarding table and every packet generated by the application to the desired destination is marked with this flow identifier. From this point, all the forwarding is based on the pair source IP address and flow

identifier (a combination that is unique in the entire network). Finally, the node sends an Reservation Request message towards the next node in the elected path in order to confirm the new reservation. In each node, a similar procedure takes place. If all the chosen nodes are able to provide the desired QoS, a Reservation Reply is eventually received by the source node, completing the reservation procedure (figure 5.8).

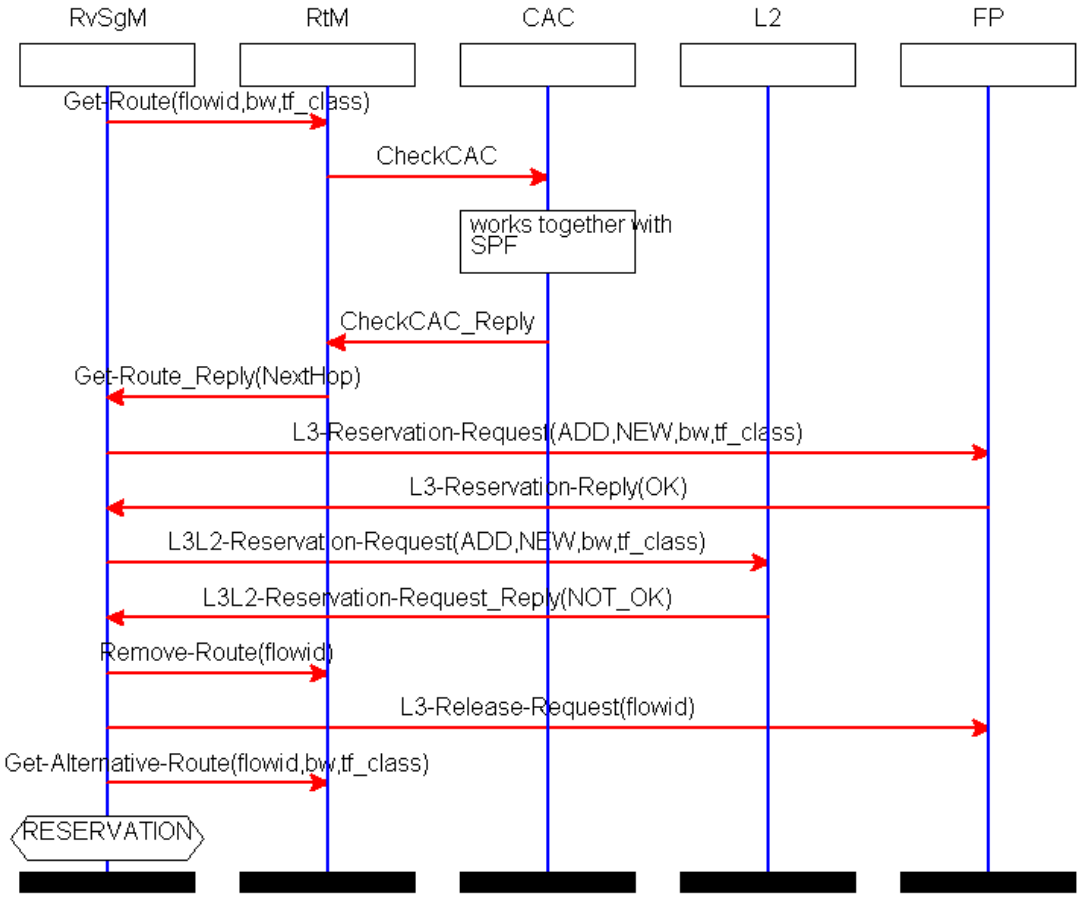


Figure 5.9: Reservation procedure when another node in the path refuses the new flow

If any chosen intermediate node, however, refuses the establishment of this new reservation – this may happen due to the fact that the topology information that the source node routing protocol has is not up to date – the Reservation Module warns the routing protocol and asks for an alternative path, restarting the process (figure 5.9).

Finally, it may also happen that a node refuses the new flow. That may happen due to the fact that the routing protocol is not aware of any path that is able to provide the required QoS (figure 5.10) or due to the fact that the MAC layer is not able to provide the required QoS – the information in Layers 2 and 3 are therefore not consistent (figure 5.11). In these cases, the Reservation Module informs the application of the impossibility to reserve the desired resources.

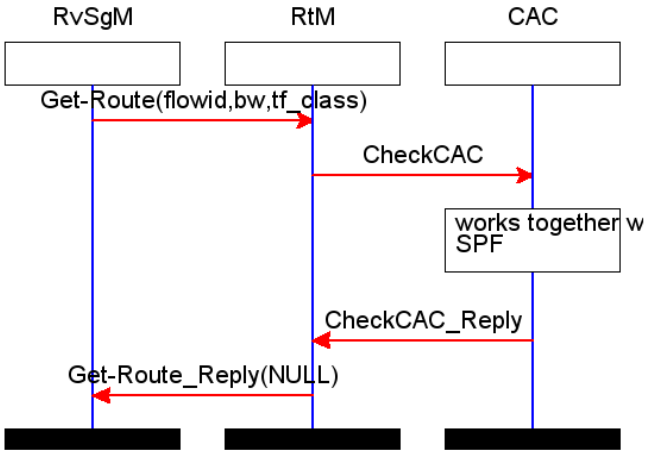


Figure 5.10: Reservation procedure when the routing protocol refuses the new flow

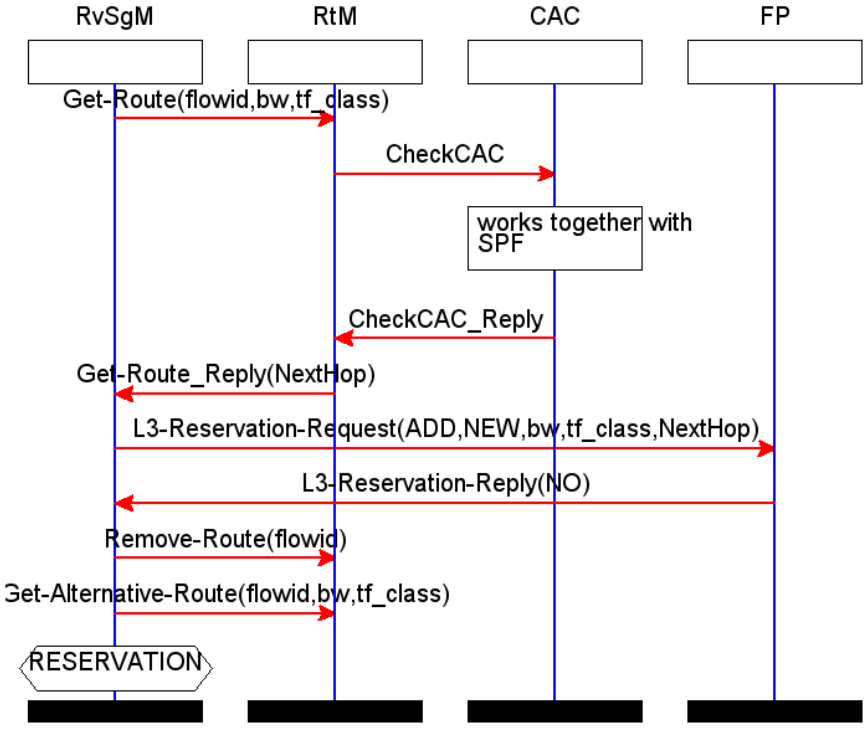


Figure 5.11: Reservation procedure when the MAC layer refuses the new flow

In the case that a new flow is accepted by every node in the path, the reservation is established and realtime data is sent. Whenever the data flow finishes, the application may explicitly request the release of the reserved resources (figure 5.12). In this case, the Reservation Module informs the MAC layer and the routing protocol of the release request and send a Release Request message to the next node in the flow path, so that a similar release procedure may take place in every node along the route to the flow destination.

It may happen, however, that a link down takes place and new paths must be found for

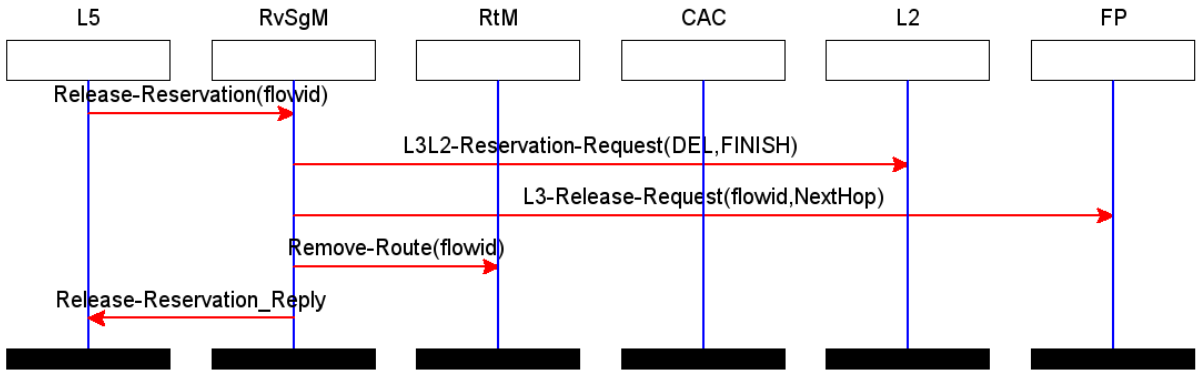


Figure 5.12: Releasing a pre-established reservation

all the reservations that use this link. Three different cases should be considered: the link down affects the source node of a flow (figure 5.13), the link that went down is between an intermediate node of a flow and the next hop in the path (figure 5.14) and the link that went down is between an intermediate node and the previous hop in the path (figure 5.14).

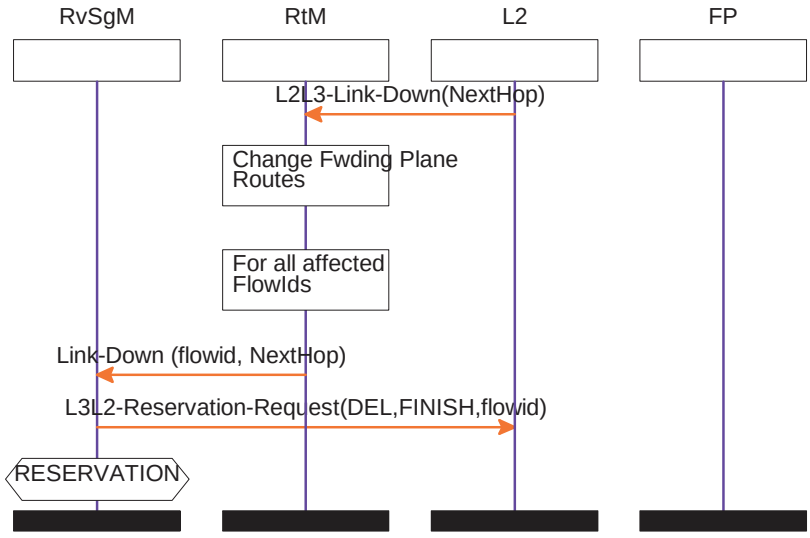


Figure 5.13: Link Down on the source node of a flow

For the first case, the routing protocol is informed by the MAC layer of the fact that the link went down. It then removes the forwarding plane entries that uses this link (since they are now useless) and informs the Reservation Module of what happened. The Reservation Module releases resources in the local MAC layer and starts a new reservation procedure, trying to find new routes for every affected flow. If there are any flows that could not be re-routed, a LinkDown message is sent to the application through the previously registered Callback Function, so that the application may stop generating data.

If, however, the link down happens on an intermediate node, besides all the release and

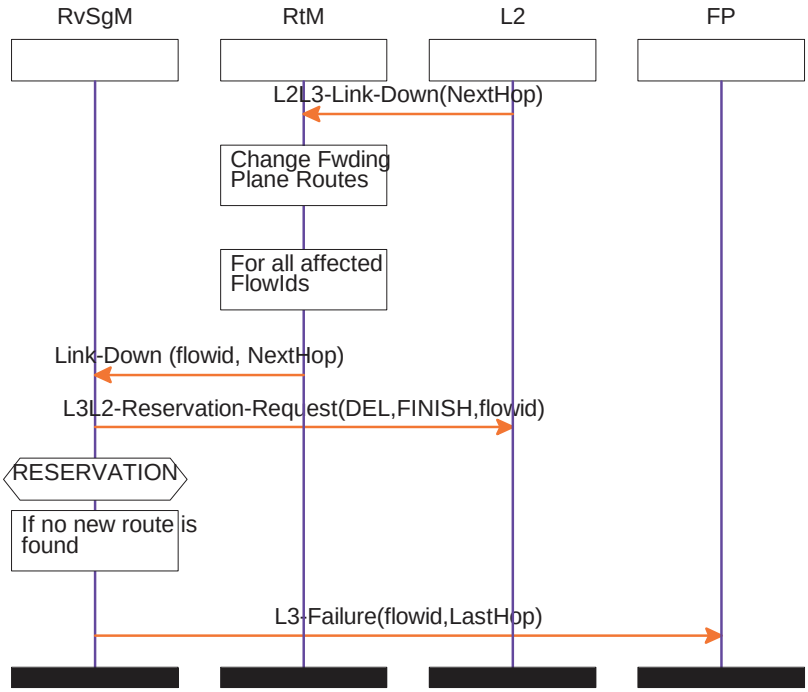


Figure 5.14: Link Down on an intermediate node of a flow

re-reservation process, a Failure message should be sent to the source node of all affected flows. This Failure message causes previous nodes to release their resources and the source node to try finding an alternative path for the flow.

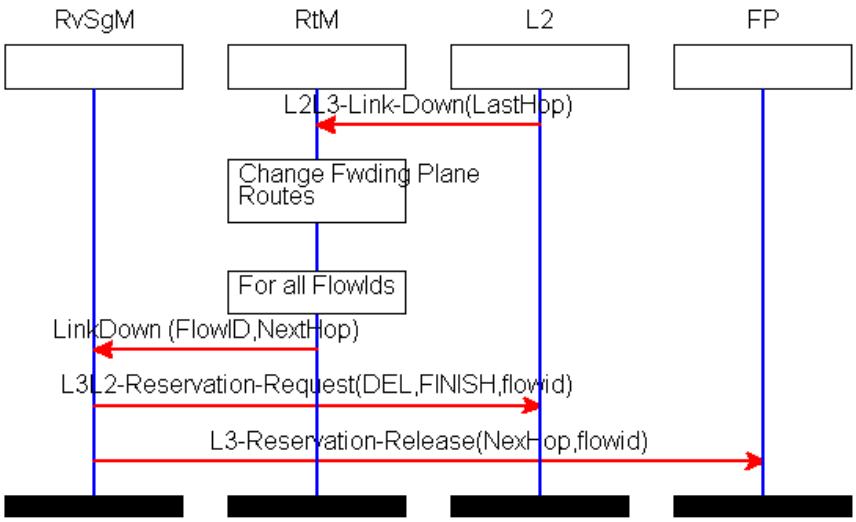


Figure 5.15: Link Down on the last hop of an intermediate node of a flow

Finally, if the link that went down is between an intermediate node and its previous hop in the path, the only possibility is to release all local resources and inform every node in the path toward the destination that resources should also be released.

Note that a link down on intermediate nodes will trigger this last two actions simultaneously. The node just before the link break will release resources and inform every node towards the source node, while the node just after the link break will release resources and inform all subsequent nodes to do the same.

5.6 Appendix A: QoS API Documentation

```
#include <netinet/in.h>

#include <sys/types.h>
```

Defines

- #define **TCP_PROTOCOL** 6
- #define **UDP_PROTOCOL** 17
- #define **ICMP_PROTOCOL** 1

Typedefs

- typedef void(∗ **widens_callback_func_ptr**)(uint32_t, char ∗)

Functions

- int **widensSetReservation** (struct sockaddr_in dst, u_short srcport, u_short protocol, int fd, u_long PeakDataRate, u_long MeanDataRate, u_long MinDataRate, u_long MaxBurstSize, u_long DelayBound, u_char QosClass, uint32_t ∗session_id, char ∗error)
- int **widensSetReservationAndOpenSocket** (struct sockaddr_in dst, u_short protocol, u_long PeakDataRate, u_long MeanDataRate, u_long MinDataRate, u_long MaxBurstSize, u_long DelayBound, u_char QosClass, uint32_t ∗session_id, char ∗error)
- void **widensReleaseReservation** (uint32_t session_id)
- void **widensRegisterCallback** (**widens_callback_func_ptr** func)

5.6.1 Define Documentation

#define TCP_PROTOCOL 6

#define UDP_PROTOCOL 17

#define ICMP_PROTOCOL 1

5.6.2 Typedef Documentation

typedef void(∗ widens_callback_func_ptr)(uint32_t, char ∗)

Callback function.

Type of the function that should be registered in order to be informed about relevant events related to ongoing connections

5.6.3 Function Documentation

int widensSetReservation (struct sockaddr_in *dst*, u_short *srcport*, u_short *protocol*, int *fd*, u_long *PeakDataRate*, u_long *MeanDataRate*, u_long *MinDataRate*, u_long *MaxBurstSize*, u_long *DelayBound*, u_char *QosClass*, uint32_t ∗ *session_id*, char ∗ *error*)

Tries to establish a new QoS reservation

Parameters:

dst IP Address and port of the destination node

srcport Port that will be used by the source node for this connection

protocol Transport protocol that will be used by the source node for this connection

fd The file descriptor of the socket that will be used by this connection

PeakDataRate Peak Data Rate

MeanDataRate Mean Data Rate

MinDataRate Minimun Data Rate

MaxBurstSize Maximun Burst Size

DelayBound Delay Bound

QosClass QoS Class that will be used

session_id Session Identifier that will be generated and returned

error if an error occurs, contains its textual description

Returns:

1 if reservation was successfully done, 0 otherwise

int widensSetReservationAndOpenSocket (struct sockaddr_in *dst*, u_short *protocol*, u_long *PeakDataRate*, u_long *MeanDataRate*, u_long *MinDataRate*, u_long *MaxBurstSize*, u_long *DelayBound*, u_char *QosClass*, uint32_t * *session_id*, char * *error*)

Tries to establish a new QoS reservation and opens/binds the socket

The socket that is opened by this function is already bound to a dynamic local port, so no additional bind should be done.

Parameters:

dst IP Address and port of the destination node

protocol Transport protocol that will be used by the source node for this connection

PeakDataRate Peak Data Rate

MeanDataRate Mean Data Rate

MinDataRate Minimun Data Rate

MaxBurstSize Maximun Burst Size

DelayBound Delay Bound

QosClass QoS Class that will be used

session_id Session Identifier that will be generated and returned

error if an error occurs, contains its textual description

Returns:

the socket descriptor if reservation was successfully done, 0 otherwise

void widensReleaseReservation (uint32_t *session_id*)

Releases a given reservation that will no longer be used

Parameters:

session_id Session Identifier

void widensRegisterCallback (widens_callback_func_ptr *func*)

Registers a callback function to deal with sessions that are closed

Whenever a session is closed (and resources are therefore release) for any given reason other than the explicit requisition from the source node, the application is warned by calling a callback function that should be registered through this primitive. This callback function expects two arguments, the first one is the session_id of the closed session (uint32_t) and the second one is the reason why the session was closed (char $*$).

Parameters:

func A pointer to the callback function

Summary and Outlook

The aim of this thesis was to study how QoS flows may be routed through heterogeneous multirate ad-hoc wireless networks and propose a mechanism that could control the admission of flows as well as route them based on the availability of resources. To achieve this goal, several steps have been followed: *i)* analyzing the problem of reactive routing on multirate ad-hoc wireless networks, *ii)* analyzing the problem of bandwidth allocation for QoS flows on these networks, *iii)* proposing strategies for adapting the QoS route to topology changes caused by a possible mobility of the nodes, and *iv)* building a simplified prototype for the proposed solutions.

A detailed study of previous works that dealt with similar issues has been carried out, contributing as a starting point for the work presented by this thesis. This background information was manly composed by previous proposals for routing on multirate ad-hoc wireless networks as well as QoS mechanism for this kind of networks.

The first contribution of the thesis is the reactive routing mechanism for multirate ad-hoc wireless networks. This mechanism provides a great enhancement on the overall performance of reactive routing mechanisms on multirate networks. On such a network, traditional routing mechanisms usually minimize the number of hops, resulting on routes composed by long range, and consequently low throughput, links. Our mechanism provides a simple and very effective way of using the transmission rate as a routing metric without a significant increase of the signaling message overhead. Previous proposals were very inefficient, hugely increasing the number of signaling messages, what sometimes could even lead to a performance decrease when compared to traditional routing protocols. Although we used transmission rates as the routing metric, our mechanism could also work with many other metrics, such as mean delay, link stability or available bandwidth.

The results of this work were published in the Proceedings of the 13th European Wireless Conference (EW2007), April 2007 [39].

Based on these modifications on the reactive routing mechanism of AODV (which we called MR-AODV — MultiRate AODV), the thesis presented an effective, and yet simple,

mechanism for supporting QoS through bandwith reservation for multirate ad-hoc wireless networks. Based on the following QoS constraint: "*The load demand offered to the wireless media by the QoS traffic observed at any node in a path that is about to be established* $\leq Q$", the mechanism was integrated not only to MR-AODV, but also to the OLSR routing protocol, since it is completely independent of the chosen routing paradigm (reactive or proactive). The Q parameter is dimensioned in a way that the end-to-end delays of the QoS flows are below an acceptable threshold. In order to guarantee the constraints of the established QoS flows, a CAC rule was proposed for new connections requiring QoS. As a result of the simulations that were conducted using our proposal (which we called BRAWN — Bandwidth Reservation on Ad-hoc Wireless Networks), we may point out the following findings:

- Ad-hoc networks can easily become congested by QoS traffic (differently from TCP, this kind of traffic typically does not provide congestion control mechanisms).
- Congestion can easily extended to most of the network introducing high delays and losses, damaging, thus, most of the connections that requires QoS.
- Our reservation scheme provides a feasible way to avoid congestion, guaranteeing, thus, QoS requirements to ongoing connections.

The preliminary results of this second part of the work were first published in the Proceedings of the IST Mobile and Wireless Communications Summit, June 2005 [22] and presented in the 1st EuroNGI Workshop on Mobility and Wireless (revised and published afterwards in the Lecture Notes in Computer Science) [21]. A more complete view of the work was accepted for publication in the Elsevier Ad-hoc Networks Journal in 2008 [41]. Some variations of the proposal were also published in the Proceedings of the 12th European Wireless Conference, May 2006 [36] (taking RTS/CTS into account), presented in the Third International Workshop of the EURO-NGI NoE and revised and published afterwards in the Lecture Notes in Computer Science [37] (using feedback from the MAC layer) and presented in the EuroNGI Workshop on QoS and Traffic Control, December 2005 [38] (a enhancement on the available capacity computation).

Although being efficient on static network topologies, i.e., when the topology does not vary on time, once nodes start moving, not all pre-established QoS reservations can still be guaranteed. Based on these observations, we have applied a mobility extension to our mechanism, calling it BRAWN-R (BRAWN with Refreshments). BRAWN-R was shown to perform well on the presence of mobility by periodically refreshing reservations in order to take possible changes into account in the resource allocation mechanism. The

BRAWN-R mechanism, one of the main contributions of this thesis, is able to adapt QoS reservations to the dynamic behavior of the network, anticipating link breaks and making a better use of network resources. Simulation results confirm that the refreshing mechanism improves the adaptiveness of the protocol, reducing packet losses and delays at a low overhead cost.

The results of this work on mobility have been presented on a paper submitted for an international conference. At the time of writing this thesis, the reviewing process was not yet finished.

As a final contribution of the thesis, we have implemented a simplified version of the proposed QoS reservation mechanism. Several modules with very specific functionalities were developed to run in collaboration in order to implement BRAWN, assuring that our proposal is, therefore, feasible.

This prototype was described in several deliverables of the WIDENS project [7] and presented on an invited paper published in the Proceedings of the 2nd Workshop on Trends in Radio Resource Management [20] and on another paper published in the Proceedings of the National Conference "*XV Jornadas Telecom I+D 2005*".

Acronyms

AB	Available Bandwidth
AODV	Ad hoc On-demand Distance Vector
API	Application Programmable Interface
AQOR	Ad-hoc QoS On-demand Routing protocol
ARF	Auto Rate Fallback
AWN	Ad-hoc Wireless Network
BRAWN	Bandwidth Reservation on Ad-hoc Wireless Networks
CAC	Connection Admission Control
CBR	Constant Bit Rate
CCDF	Complementary Cumulative Distribution Function
CEDAR	Core Extraction Distributed Ad hoc Routing
CSMA	Carrier Sense Multiple Access
CTS	Clear To Send
DiffServ	Differentiated Services
DLC	Data Link Control
DSR	Dynamic Source Routing
DYMO	Dynamic MANET On-demand
ECN	Explicit Congestion Notification
FTP	File Transfer Protocol
FP	Forwarding Plane
FQMM	Flexible QoS Model for MANETs
HWMP	Hybrid Wireless Mesh Protocol
IBSS	Independent Basic Service Set
ICMP	Internet Control Message Protocol
IEEE	Institute of Electric and Electronic Engineers
IETF	Internet Engineering Task Force
IntServ	Integrated Services
IP	Internet Protocol
ISO	International Organisation for Standardisation
ITU	International Telecommunications Union
ITU-T	ITU Telecommucation sector
LUNAR	Lightweight Underlay Network Ad-hoc Routing
MAB	Maximum Available Bandwidth
MAC	Medium Access Control
MANET	Mobile Ad-hoc Network

MESA Mobility for Emergency and Safety Applications

MPR Multi Point Relay

MR-AODV Multi-Rate AODV

MSC Message Sequence Chart

MTM Medium Time Metric

NS Network Simulator

OLSR Optimized Link State Routing

QAODV Quality of Service for Ad hoc On-Demand Distance Vector

QoS Quality of Service

RBAR Receiver Based Auto Rate

RREP Route Reply

RREQ Route Request

RSVP Resource reSerVation Protocol

RtM Routing Module

RTS Request To Send

RvSgM Reservation Signaling Module

SNR Signal to Noise Ratio

SPF Shortest Path First

TBRPF Topology Dissemination Based on Reverse-Path Forwarding

TC Topology Control

TCP Transmission Control Protocol

TDMA Time Division Multiple Access

TOS Type of Service

UDP User Datagram Protocol

WIDENS WIreless DEployable Network System

ZRP Zone Routing Protocol

Index

Bibliography

[1] Datasheet for Orinoco 11b Client PC Card. Available: http://www.proxim.com/learn/library/datasheets/11bpccard_A4.pdf.

[2] ITU-T Study Group 12. http://www.itu.int/ITU-T/studygroups/com12.

[3] NRLOLSR Implementation from the NRL (Naval Research Laboratory). Available: http://pf.itd.nrl.navy.mil/projects.php?name=olsr.

[4] The MESA Project. http://www.projectmesa.org.

[5] The Moby Dick Project. http://www.ist-mobydick.org.

[6] The Network Simulator ns-2. Available: http://www.isi.edu/nsnam/ns.

[7] The WIDENS Project. http://www.widens.org.

[8] Wikipedia - The Free Encyclopedia. http://www.wikipedia.org.

[9] IEEE Std 802.11a, Supplement to Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer in the 5 GHZ Band, 1999.

[10] IEEE Std 802.11b, Supplement to Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: Higher-speed Physical Layer Extension in the 2.4 GHz Band, 1999.

[11] IEEE Std 802.11e/D3.0. (Draft Supplement to IEEE Std 802.11). Medimum Access Control (MAC) Enhancements for Qualiy of Service, May 2002.

[12] IEEE Std 802.11g/D8.2, Draft Supplement to Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: Further Higher-Speed Physical Layer Extension in the 2.4 GHz Band, 2003.

[13] A. Joshi et al. HWMP Specification, 2006. Available: `https://mentor.ieee.org/802.11/public/06/11-06-1778-01-000s-hwmp-specification.doc`.

[14] D. Aguayo, J. Bicket, and R. Morris. SrcRR: A High Throughput Routing Protocol for 802.11 Mesh Networks. Technical report, 2003. Available: `http://pdos.csail.mit.edu/~rtm/srcrr-draft.pdf`.

[15] Ian F. Akyildiz, Xudong Wang, and Weilin Wang. Wireless mesh networks: a survey. *Computer Networks*, 47(4):445–487, March 2005.

[16] B. Awerbuch, D. Holmer, and H. Rubens. The Medium Time Metric: High Throughput Route Selection in Multirate Ad Hoc Wireless Networks. *Technical Report: Department of Computer Science, Johns Hopkins University*, Oct 2004.

[17] H. Badis and K. Al Agha. QOLSR, QoS routing for Ad Hoc Wireless Networks Using OLSR. *European Transactions on Telecommunications*, 15(4):427–442, 2005.

[18] A. Boukalov and V. Conan. Wireless Deployable Network System for Future Public Safety - WIDENS Project. In *Proceedings of the E^2R Workshop on Reconfigurable Mobile Systems and Networks Beyond 3G*, 2004.

[19] D. H. Cansever, A. M. Michelson, and A. H. Levesque. Quality of service support in mobile ad-hoc IP networks. In *Proceedings of the IEEE Military Communications Conference*, Oct 1999.

[20] Ll. Cerdà, R. Guimarães, J. Morillo, J.M. Barcelo, J. García, A. Perez-Neira, M. Realp, M. Voorhaen, C. Blondia, R. Knopp, and N. Nikaein. QoS Management in WIDENS terminodes. In *Invited Paper for the 2nd Workshop on Trends in Radio Resource Management*, November 2005.

[21] Ll. Cerdà, M. Voorhaen, R. Guimarães, J.M. Barcelo, J. García, and C. Blondia. A Reservation Scheme Satisfying Bandwidth QoS Constraints for Ad-Hoc Networks. *Lecture Notes in Computer Science (LNCS)*, 3427:176–, Feb 2005.

[22] Ll. Cerdà, M. Voorhaen, R. Guimarães, J.M. Barcelo, J. García, J. Morillo, and C. Blondia. A Reservation Scheme Satisfying Bandwidth QoS Constraints for Multirate Ad-Hoc Networks. In *IST Mobile and Wireless Communications Summit*, Jun 2005.

[23] I. D. Chakeres and C. E. Perkins. Dynamic MANET On-demand Routing Protocol (draft), May 2007.

[24] S. Chakrabarti and A. Mishra. QoS issues in ad hoc wireless networks. *IEEE Communications Magazine*, 39(2):142–148, Feb 2001.

[25] S. Chen and K. Nahrstedt. Distributed quality-of-service routing in ad hoc networks. *IEEE Journal on Selected Areas in Communications*, 17(8):1488–1505, Aug 1999.

[26] T. Clausen and P. Jacquet. RFC 3626: Optimized Link State Routing Protocol (OLSR), Oct 2003.

[27] C. Tschuding et al. LUNAR: a Lightweight Underlay Network Ad-hoc Routing Protocol and Implementation. In *Proceedings of the Next Generation Teletraffic and Wired/Wireless Advanced Networking (NEW2AN'04)*, Feb 2004.

[28] I. Haratcherev et al. Automatic IEEE 802.11 rate control for streaming applications. *Wireless Communications and Mobile Computing*, 5(4):421–437, Jun 2005.

[29] J. Bicket et al. Architecture and Evaluation of an Unplanned 802.11b Mesh Network. In *Proceedings of the 11th ACM International Conference on Mobile Computing and Networking (MobiCom '05)*, Cologne, Germany, Aug 2005.

[30] T. Kosch et al. The scalability problem of vehicular ad hoc networks and how to solve it. *IEEE Wireless Communications*, 13(5):22–28, Oct 2006.

[31] Z. Fan. High throughput reactive routing in multi-rate ad hoc networks. *Electronic Letters*, 40(25):1591–1592, Dic 2004.

[32] L. M. Feeney and M. Nilsson. Investigating the energy consumption of a wireless networkinterface in an ad hoc networking environment. In *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 3, pages 1548–1557, 2001.

[33] Y. Gao, D. Chiu, and J. C. S. Lui. Determining the End-to-end Throughput Capacity in Multi-Hop Networks: Methodology and Applications. In *Proceedings of the SIGMetrics/Performance'06*, Jun 2006.

[34] A. J. Goldsmith and S. B. Wicker. Design challenges for energy-constrained ad hoc wireless networks. *IEEE Wireless Communications*, 9(4):8–27, Aug 2002.

[35] R. Guimaraes and Ll. Cerda. Improving Reactive Routing on Wireless Multirate Ad-hoc Networks. In *Proceedings of the 13th European Wireless Conference*, Apr 2007.

[36] R. Guimarães and Ll. Cerdà. Bandwidth Reservation On Wireless Networks with RTS/CTS Signalling. In *Proceedings of the 12th European Wireless Conference (EW2006)*, April 2006.

[37] R. Guimarães and Ll. Cerdà. Adaptive QoS Reservation Scheme for Ad-hoc Networks. *Lecture Notes in Computer Science*, 4396:102–112, May 2007.

[38] R. Guimarães and Ll. Cerdà. An Enhanced Bandwidth Reservation Scheme for Ad-hoc Networks, November 2007.

[39] R. Guimarães and Ll. Cerdà. Improving reactive routing on wireless multirate ad-hoc network. In *Proceedings of the 13th European Wireless Conference*, April 2007.

[40] R. Guimarães and Ll. Cerdà. *Wireless Systems and Mobility in Next Generation Internet: Third International Workshop of the EURO-NGI Network of Excellence*, volume 4396 of *Lecture Notes in Computer Science (LNCS)*, chapter Adaptive QoS Reservation Scheme for Ad-hoc Networks, pages 102–112. Springer-Verlag, March 2007.

[41] R. Guimarães, Ll. Cerdà, J.M. Barcelo, J. García, M. Voorhaen, and C. Blondia. Quality of Service through Bandwidth Reservation on Multirate Ad-doc Wireless Networks. *Elsevier Journal of Ad Hoc Networks (to appear)*, 2008. Available: `http://dx.doi.org/10.1016/j.adhoc.2008.04.002`.

[42] R. Guimarães, J. Morillo, Ll. Cerdà, J.M. Barcelo, and J. García. Redes Inalámbricas de Emergencia Pública: el Proyecto WIDENS. In *XV Jornadas Telecom I+D 2005*, November 2005.

[43] L. Guolong, G. Noubir, and R. Rajaraman. Mobility models for ad hoc network simulation. In *INFOCOM 2004. Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies*, 2004.

[44] P. Gupta and P. R. Kumar. The Capacity of Wireless Networks. *IEEE Transactions on Information Theory*, 46(2):388–404, mar 2000.

[45] G. Holland, N. Vaidya, and P. Bahl. A Rate-adaptive MAC Protocol for Multi-hop Wireless Networks. In *Proceedings of the 7th ACM Annual International Conference on Mobile Computing and Networking (MobiCom)*, pages 236–251, Sep 2001.

[46] L. Huang and T. H. Lai. On the scalability of IEEE 802.11 ad hoc networks. In *Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking and computing*, pages 173–182, 2002.

[47] I. Joe and S. G. Betsell. MPR-based Hybrid Routing for Mobile Ad-Hoc Networks. In *Proceedings of the 27th Annual IEEE Conference on Local Computer Networks*, pages 7–12, Nov 2002.

[48] D. B. Johnson, D. A. Maltz, and Y. Hu. DSR-draft: The Dynamic Source Routing Protocol for Mobile Ad Hoc Netowrks (DSR), Apr 2003.

[49] A. Kamerman and L. Monteban. WaveLAN II: A high-performance wireless LAN for the unlicensed band. *Bell Labs Technical Journal*, pages 118–133, Summer 1997.

[50] M. Kazantzidis, M. Gerla, and S. Lee. Permissible throughput network feedback for adaptive multimedia in AODV MANETs. In *Proceedings of the IEEE International Conference on Communication*, Jun 2001.

[51] E. Kohler. The Click modular router (PhD Thesis - MIT), November 2000.

[52] S. Lee, G. Ahn, and A. T. Campbell. Improving UDP and TCP performance in mobile ad hoc networks with INSIGNIA. *IEEE Communications Magazine*, 39(6):156–165, Jun 2001.

[53] W. Liu and Y. Fang. Courtesy Piggybacking: Supporting Differentiated Services in Multihop Mobile Ad Hoc Networks. In *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, March 2004.

[54] N. Nikaein and C. Bonnet. A Glance at Quality of Service Models for Mobile Ad Hoc Networks. In *16eme Congres DNAC (De Nouvelles Architectures pour les Communications)*, Dec 2002.

[55] C. Perkins and E. Belding-Royer. Quality of Service for Ad hoc On-Demand Distance Vector Routing (work in progress), Oct 2003. draft-perkins-manet-aodvqos-02.txt.

[56] C. Perkins and S. Daas E. Belding-Royer. RFC 3561: Ad hoc On-Demand Distance Vector Routing, Jul 2003.

[57] A. Qayyum, L. Viennot, and A. Laouiti. Multipoint relaying: An efficient technique for flooding in mobile wireless networks. In *Proceedings of the 35th Annual Hawaii International Conference on System Sciences*, 2001.

[58] M. Lewis R. Ogier, F. Templin. RFC 3684: Topology Dissemination Based on Reverse-Path Forwarding (TBRPF), Feb 2004.

[59] J. Rajahalme, A. Conta, B. Carpenter, and S. Deering. RFC 3697: IPv6 Flow Label Specification, Mar 2004.

[60] R. Rajaraman. Topology control and routing in ad hoc networks: a survey. *ACM SIGACT News*, 33(2):60–73, Jun 2002.

[61] T. B. Reddy, I. Karthigeyan, B. S. Manoj, and C. S. R. Murthy. Quality of service provisioning in ad hoc wireless networks: a survey of issues and solutions. *Ad Hoc Networks*, 4(1):83–124, 2006.

[62] A. K. Saha and D. B. Johnson. Modeling mobility for vehicular ad-hoc networks. In *Proceedings of the 1st ACM international workshop on Vehicular ad hoc networks*, 2004.

[63] P. Samar, M. R. Pearlman, and Z.J. Haas. Independent zone routing: an adaptive hybrid routing framework for ad hoc wireless networks. *IEEE/ACM Transactions on Networking*, 4(12):595–608, Aug 2004.

[64] P. Santi. Topology control in wireless ad hoc and sensor networks. *ACM Computing Surveys (CSUR)*, 37(2):164–194, 2005 Jun.

[65] Y. Seok, J. Park, and Y. Choi. Multi-rate Aware Routing Protocol for Mobile Ad Hoc Networks. In *Proceedings of the IEEE Vehicular Technology Conference*, volume 3, pages 1749–1752, Spring 2003.

[66] G. Sharma, R. Mazumdar, and B. Shroff. Delay and Capacity Trade-Offs in Mobile Ad Hoc Networks: A Global Perspective. *IEEE/ACM Transactions on Networking*, 15(5):981–992, Oct 2005.

[67] R. Sivakumar, P. Sinha, and V. Bharghavan. CEDAR: A COre-Extraction Distributed Ad Hoc Routing Algorithm. *IEEE Journal on Selected Areas in Communications*, 17(8):1454–1465, Aug 1999.

[68] A. Veres, G. Ahn, A.T. Campbell, and L. Sun. SWAN: Service Differentiation in Stateless Wireless Ad Hoc Networks. In *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, Jun 2002.

[69] Y. Wang. *Topology Control for Wireless Sensor Networks*, pages 113–147. Springer US, 2008.

[70] H. Xiao, W. G. Seah, A. Lo, and K. C. Chua. A flexible quality of service model for mobile ad hoc netowrks. In *Proceedings of the IEEE Vehicular Technology Conference*, May 2000.

[71] Y. Xiao. IEEE 802.11n: enhancements for higher throughput in wireless LANs. *IEEE Wireless Communications*, 12(6):82–91, Dic 2005.

[72] Q. Xue and A. Ganz. Ad hoc QoS on-demand routing (AQOR) in mobile ad hoc networks. *Journal of Parallel and Distributed Computing*, (63):154–165, 2003.

[73] Y. Yi, M. Gerla, and T. J. Kwon. Efficient Flooding in Ad hoc Networks: a Comparative Performance Study. In *Proceedings of the IEEE International Conference on Communications*, volume 2, pages 1059–1063, May 2003.

[74] S. Zou, S. Cheng, and Y. Lin. Multi-rate Aware Topology Control in Multi-hop Ad Hoc Networks. In *Proceedings of the IEEE Wireless Communications and Networking Conference*, volume 4, pages 2207–2212, Mar 2005.