NETWORKED SYSTEMS GROUP
DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF KAISERSLAUTERN

Diploma Thesis

Design and Evaluation of a QoS Routing Protocol for Reservation-based Networks

Jiarui Xiao

29.09.2008

Design and Evaluation of a QoS Routing Protocol for Reservation-based Networks

Diploma Thesis

Networked Systems Group
Department of Computer Science
University of Kaiserslautern

Jiarui Xiao

Date	:	29.09.2008
Supervisors	:	Prof. Dr. Reinhard Gotzhein
		Dipl.-Inf. Philipp Becker
		Dipl.-Inf. Thomas Kuhn

Table of Contents

List of Figures

1. Introduction and Motivation

Wireless communication is the transfer of information without physical cables or “wires”. Instead, some forms of energy like radio frequency, infrared light and laser light are used as communication medium in wireless communication. Due to the nature of wireless communication, it provides many possibilities such as broadcast communication and the support for mobile communication, which are difficult or impossible to realize with wired communication. Since the medium is shared by more than one transmission in wireless networks, we must face some new problems such as transmission interference.

With the increasing need for and the development of wireless technology, different portable devices are brought into market, such as cellular phones, personal digital assistants (PDA) and laptops. All these devices need the support of wireless communication. Therefore, various kinds of wireless communication networks have been developed, such as cellular networks [Ra02], wireless LANs (WLAN) [Wl97], Bluetooth networks [Ha00] and WiMax [Wm04]. Cellular networks, WLANs, and WiMax are infrastructure based networks (WLAN can be ad hoc only with peer-to-peer mode), where the communication is controlled by some designated central devices. Bluetooth technology supports ad hoc connection of nodes, but there are many limitations, since Pico net [Ha00] is the standard network topology of Bluetooth. For instance, neighboring nodes can communicate with each other, when they are both slave nodes. Different from all the networks above, mobile ad hoc networks (MANETs) [Mu04] are distributed, self-configuring networks, which have become a popular research subject in recent years. MANETs don’t require centralized control and each node can work as a router. Such networks are useful in military or emergency aid domain, where an infrastructure could not be supported.

MacZ [HIS07], a MAC layer for MANETs, was developed in the domain of Ambient Intelligence [AmI] by Networked Systems Group [NSG] in the department of computer science at the University of Kaiserslautern. MacZ supports tick synchronization in a slotted medium and it is designed to support QoS by offering 2 possibilities to access the medium. One of them is the contention-based medium access with priorities. In this method, the network nodes compete with each other for the privilege to use the medium. Besides this, MacZ supports also contention-free method. With this method the medium resource can be reserved for communication. The data frames can only be sent after the bandwidth needed for this transmission is reserved. The assignment of priorities and reservation are not parts of MacZ. So in order to realize medium reservation, a protocol named Black Burst Reservation Protocol (BBRP) [Xi07] was developed. BBRP is based on MacZ and uses the services provided by MacZ. It completes the contention-free method of MacZ and provides bandwidth reservation for the higher layers. This protocol is able to establish and maintain reservations in multi-hop scenario and the range of the reservation is adjustable.

In MANET, routing is one of several problems for data transfer. Many routing protocols (e.g. AODV [Pe03], DSDV [Pe94], TBP [Ch99] and AQOR [Xu03]) were introduced in this domain. Some routing protocols use “best effort” strategy, which means that there are no guarantees for the data transfer and all nodes compete for the medium. Some other protocols use QoS (Quality-of-Service) routing. In QoS routing, the goal is to find a route satisfying one or more QoS constraints, such as bandwidth or delay.

Compared to other networks, designing a QoS routing protocol in MANETs is much more complex. Besides problems such as exposed and hidden terminal problem [Bh94] that can occur in most wireless networks, the mobility of the nodes results in more unpredictable connection breakups or interferences and therefore causes even more data collisions or QoS violations. In order to resolve these problems, a QoS routing protocol is developed in this thesis, which is based on MacZ and BBRP. This routing protocol supports 2 QoS constraints simultaneously: bandwidth and delay. In this routing protocol, the route discovery is performed in a contention-based period of medium, while the data is transferred in a contention-free period. To the best of our knowledge, this is quite different from most routing protocols, which based only on one kind of medium. This routing protocol is tightly interlocked with BBRP and integrates the functions for resource reservation and QoS maintenance. The route discovery phase includes two sub-phases, route request and route reply phase. The reservation of resource is done during the route discovery phase. Some route maintenance mechanisms are deployed in this routing protocol including QoS violation detection and QoS route recovery.

This routing protocol is specified using “Specification and Description Language (SDL)” [ITU99] with the modeling tool Telelogic TAU 4.6 [TEL]. With the help of SDL an executable system can be built and simulated. The “Message Sequence Charts” [MSC] are used in the design phase and the simulation of the routing protocol.

The chapters of this thesis are organized as follows. In chapter 2 we introduce MANETs and routing protocol for MANETs. The best-effort routing is first mentioned and then we focus on the QoS routing. In chapter 3 we present the conceptual design of our routing protocol with algorithms. In chapter 4 this routing protocol is specified with SDL. Then in chapter 5 we give results from performed simulations. In chapter 6 some other routing protocols in this area are introduced. Finally, we summarize this thesis and provide an outlook of future work.

2. Routing Protocols

2.1. MANETs

In wired networks the network nodes are connected by physical wires. The network topology doesn't change very often and the network nodes are stationary. In wireless networks, especially in MANETs, the situation has changed. We summarize it as the following aspects:

Firstly the medium is shared. The communication medium in wireless networks is shared by multiple transmissions, which means the transmission channel is not as reliable as in wired networks. Transmissions may interfere with each other and the packet delivery error happens more often. Shared medium denotes also that members of the network may need to compete for the privilege to access the medium, while in wired networks the medium between network members is separate. But shared medium has also its advantages. For instance, the broadcast transmission is natural in MANET while it is more complicated to realize in wired networks.

Secondly the network nodes are mobile. They can move completely independent and randomly, which means the information of the network topology changes constantly and needs to be updated frequently among the nodes. The link lifetime is a lot shorter when compared to wired networks. A break of connection can happen at any moment. The mobility of network nodes is one of several important advantages for wireless networks against wired networks.

Thirdly MANET is infrastructureless. There are mainly two types of wireless networks, infrastructure based networks and infrastructure free networks. Infrastructure based networks have base stations that connect the mobile user, and similar as in wired networks the routing is done by these central routers. When a mobile node moves out of the range of its base station, this node must find another base station to connect to. A handover is happened between the old and new base stations. Since this handover should be seamless for the mobile nodes, there is problem if a mobile node moves too fast, which means that the base station does not have enough time to detect the exit of the mobile node and to hand it over to another base station. Infrastructure free networks do not have such base stations. MANET is an infrastructure free network because as an ad hoc network, it is difficult to provide any form of centralized control. MANET can be set up spontaneously and its members can change dynamically. Most mobile nodes in MANETs cannot connect directly to all other nodes in the network. Therefore, each node can serve as a router. Two unconnected nodes can communicate with each other by means of deploying the nodes between them as routers. Packets sent from the source node will be forwarded to the destination by these routers.

Fourthly is the limited resource. Unlike in infrastructure networks the mobility of network nodes becomes a deciding factor in MANETs. But such mobile devices normally have lower computational power, memory and energy compared to the devices in the wired networks (e.g. desktop computers). This is especially fatal when QoS assurances are provided, since QoS routing protocols consume often more resources than best-effort routing protocols due to the extra information being exchanged and stored between the network members.

Due to their advantages against wired networks and other wireless networks under some circumstances, MANETs became an interesting research subject during recent years and many protocols were presented in this domain.

2.2. Routing Protocol for MANETs

In order to send data to a node, which is more than 1 hop away from the sender in multi-hop network, intermediate nodes have to be found to forward data. The process of finding such intermediate node is called *routing*. A *routing protocol* handles actions related to routing, which is specified in the MAC layer and network layer. As stated above, MANETs have the following features: shared medium, node mobility, lack of centralized control and limited resource. Therefore many widely used routing protocols cannot be applied to MANETs directly and numerous routing protocols designed for MANETs are published in recent years. These routing protocols can be classified according to several criteria. These criteria describe the working mechanisms and features of routing protocols.

1. Network state: local, global or aggregated

   Network state is the information to describe the whole network or part of it. It covers position information, topology information like links between nodes and QoS metric information like bandwidth. For each node, storing only its own network state can reduce the memory consumption, but it could be not enough to perform route discovery, because the information about neighboring nodes may be necessary to find a route in some route protocols . Keeping global state in every node is good for finding a route, but it requires massive information storage and information exchange for large network and thus reduces the scalability. As a compromise, aggregated state, e.g. the state of all 1 hop neighboring nodes, could be gathered in each node to find a suitable route without causing massive message exchange.

2. Propagation timing of network state modification: proactive or reactive

   Proactive routing is also called table-driven routing, since the network state is stored in distributing routing tables. The periodical exchange of network state increases the network communication and the information storage, but it provides the routes directly in source nodes. Reactive routing is also called on-demand routing, which means that the current network state is updated only when needed. Hence in reactive routing, the latency time is higher to find a route comparing to proactive routing, but less communication and storage overheads are produced and the information about the changes of network topology or link failures are more quickly propagated.

3. Route establishment: source routing, distributed routing or hierarchical routing

   With source routing the entire route is known to the source node and is included in the data packet. Each intermediate node takes the route information from the packet and then forwards the packet to the next hop. Hierarchical routing divides the nodes into different groups and processes routing within a group or between groups with divide and conquer-paradigm. Distributed routing is a hop-by-hop routing, which means each intermediate node decides independently which neighboring nodes should receive the packet.

4. Addressing : unicast or multicast

   In unicast routing only one destination node will receive packets, which are intentionally sent to it by source node. In multicast routing, multiple nodes are planned to receive packets.

In general, routing protocols in MANETs have the following three steps: route discovery, route selection and route maintenance.

1. Route discovery

A route is defined as a list of nodes (n_1, n_2, …, n_m) with $m \geq 2$, where n_1 is the source node and n_m is the destination node. There must be a link between n_i and n_{i+1} with $1 \leq i \leq m-1$. Route discovery starts at source node. For source routing, routes are initiated at the source node. That means each network node should have sufficient information about the entire network. That could cause a large amount of information exchange and the exchange process could take a long time. Besides, source routing supports no local route repair. For the reasons above source routing is not suitable for MANETs with high mobility. Instead distributed routing is used more often in MANETs.

Many protocols discover the routes in a reactive or hybrid (reactive and proactive) manner. In proactive routing the reaction on route breakup is quite slow. That is not efficient in MANETs where the change of network topology and route breakup could happen at any moment. Therefore reactive routing is more suitable for MANETs,

2. Route selection

   Route selection is required when more than one route from the source to the destination are found. There are a number of route selecting algorithms, from simple shortest path algorithms to complex quality controlled algorithms. DSR [Jo07] uses a shortest path algorithm and AODV [Pe03] utilizes the route with lowest delay. Some protocols use also link-stability based algorithms, which separates links into stable links and transient links. Stable links are favored to be used and transient links serve as a backup. The Quality of Service (QoS) is also considered by many protocols, which is discussed in chapter 2.3.

3. Route maintenance

   Due to the dynamic nature of MANETs, the mobility of nodes reduces the stability of connection. Therefore route maintenance is considered as a vital aspect to judge the quality of a route protocol. It is targeted at reducing the communication disruption time. Route maintenance can be divided into 2 categories: proactive route maintenance and reactive route maintenance.

   Proactive route maintenance tries to predict route failure, while reactive route maintenance is initiated only after a failure occurred. A route failure is defined as the summary of link failure and QoS violations. We focus here on the link failure, while QoS violations are discussed in chapter 2.3. In [Me04], the author presents a proactive route maintenance mechanism based on battery power and signal strength estimation. If a mobile node discovers that route critical incidents like signal strength weakening are mostly likely to happen, it sends a Route Change Request (RCR) message to the source node about the nature of the problem. Reactive route maintenance mechanisms are present in many routing protocols for MANETs. Link failure triggers a message to the source node about the dead link.

Until now we are talking about Best-Effort routing, where no guarantee of QoS exists. Sometimes users of computer networks have some performance demand on the connections. For example, in internet telephony the minimum required bandwidth for connections should be assured. For this situation we need the QoS routing. In Figure 2.1 we assume that node S want to communicate with D using internet telephony. A route with the shortest path is selected in Figure 2.1A with Best-Effort routing. But in order to keep the quality of speech, a throughput of 5 bandwidth units is required along the route. In Figure 2.1B the shortest path cannot fulfill the bandwidth requirement and another qualified route should be found. This can be achieved by QoS routing protocols.

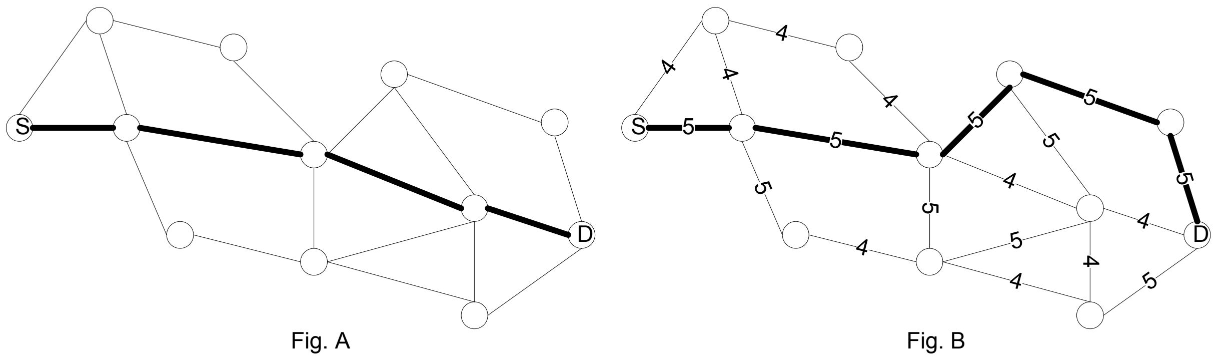


Figure 2.1: Best-Effort Routing vs. QoS Routing

2.3. QoS Routing Protocols in MANETs

Quality of Service (QoS) is the collective term used to determine the degree of satisfaction of a user of the service. QoS routing involves adding mechanisms to control the routing process and sometimes a prediction of the performance of routes based on previously gathered statistics (e.g. route break prediction based on the information of signal strength). Besides the common attributes of routing protocols in MANETs discussed in chapter 2.2, QoS routing protocols need to fulfill QoS constraints and to reserve resources. Therefore we discuss QoS routing protocols regarding the following aspects:

1. QoS constraints

   QoS metrics can be divided into different groups like additive metrics, multiplicative metrics and concave metrics. These groups of metrics are defined as follows:

   Let $m(n_i,n_j)$ be a metric for link (n_i,n_j) and $p = (n_1,n_2,\dots n_m)$ be a path between node n_1 and n_m

   Additive: $m(p) = \sum_{i=1}^{m-1} m(n_i, n_{i+1})$

   Multiplicative: $m(p) = \prod_{i=1}^{m-1} m(n_i, n_{i+1})$

   Concave: $m(p) = \min(\{m(n_i,n_{i+1}): 1 \le i \le m-1\})$

   A QoS constraint is defined as a lower or upper bound to a QoS metric. A QoS routing protocol must support routing with at least one QoS constraint. There are numerous important constraints like loss probability (multiplicative), cost and delay jitter (additive), bandwidth (concave) and end-to-end delay (additive). QoS routing protocols support single or multiple constraints. As described in [Wa96], it's very hard to find an optimal route satisfying multiple constraints concurrently, because this is of complexity NP. For this reason, most routing protocols try to find routes satisfying all constraints instead of searching for the optimal one.

2. QoS route discovery

   During the route discovery, the QoS constraints should be checked along the route. Concave metrics (e.g. bandwidth) constraints can be checked hop by hop locally. To verify additive metrics (e.g. delay) constraints or multiplicative metrics (e.g. loss probability) constraints, the information from the preceding nodes of the route is needed. When a QoS constraint cannot be fulfilled in an intermediate node, the route discovery in this node will be stopped locally.

Due to the shortage of network resources, it is hard to find a qualified route in a single path. Some protocols try to use a multi-path method. Multi-path route means that the data stream can be divided into many sub-streams and is reunited later. In this situation the concave QoS constraint (e.g. bandwidth) is also divided into many sub-constraints, which are more likely to be satisfied than the original one. The additive and multiplicative QoS constraints remain unchanged for each sub-stream. In general the multi-path solution is more complicated than the single-path, because extra control and maintenance mechanisms are required for the separation and reunion of sub-streams.

3. Resource reservation

   In order to satisfy QoS constraints along a route, resources have to be reserved along this route. A QoS routing protocol can either use its own reservations functionality or deploy other resource reservation protocols. Due to the nature of wireless networks, resources like bandwidth are shared by all the neighboring nodes. That makes the reservation more complicated, because the reservation has to be propagated to other nodes in the network. The reservation can be done before or after the route is selected. If the reservation is done after the route selection, a prereservation mechanism can be deployed to assure the resources are still available when they are reserved later. But generally a prereservation will not be propagated to the neighboring nodes.

4. QoS route selection

   QoS route selection is required when more than one route satisfying the QoS constraints is found. The selection is often based on one criterion, like the largest available bandwidth or the shortest end-to-end delay. The selection can be performed at the source node or at the destination node, which depends on the working mechanism of routing protocol.

5. QoS route maintenance

   Besides the link failure explained in chapter 2.2, QoS violation should also be considered by the QoS routing protocols. Due to the nature of MANETs like mobility, change of network topology and change of available resources, it is very difficult to meet a QoS constraint in all cases. Some routing protocols support soft QoS [Ch99], which means that there may exist transient time periods when the required QoS is not guaranteed. During this time a new satisfying route should be discovered. For example, if the destination node finds that the data packets arrive too late and exceed the maximum delay constraint, this node can choose to tolerate this or start a route discovery to find a new satisfying route.

QoS routing processes with different metrics deliver usually different results. In Figure 2.2A one route with the minimum bandwidth constraint 5 is found, while in Figure 2.2B three routes fulfill the maximum delay constraint 8. If a QoS routing with multiple constraints (bandwidth > 5 and delay < 8) is required based on the network information in Figure 2.2A and B, then there is no single-path solution satisfying both constraints. But there are still multi-path solutions.

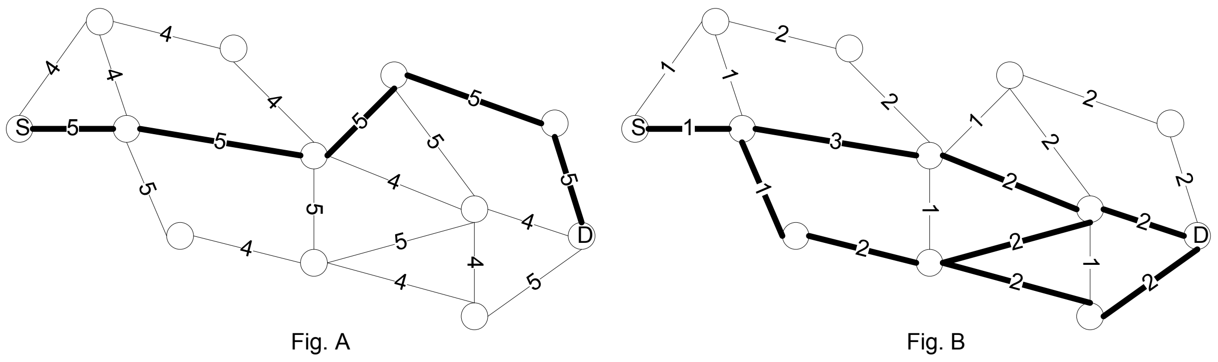


Figure 2.2: QoS routing with different metrics: (A) bandwidth and (B) delay

QoS routing may require the co-work of different layers. For resource reservation, a reservation protocol may be needed. For route recovery, a QoS MAC layer supporting contention-free and contention-based medium accesses may be required. There are some QoS MAC protocols developed in this domain like RT-MAC [BIM99], DFC-PC [DC99] and MacZ [HIS07].

2.4. MacZ

MacZ [HIS07] is a MAC layer based on medium (time) slotting and tick synchronization. MacZ aims at providing QoS in ambient ad hoc and sensor networks. The mobile nodes in these networks have scarce resources, e.g. energy, memory and computing power. Therefore the resource saving mechanisms should be considered in the design.

Black-Burst [Mo07], a special collision-resistant transmission mechanism, is used to support the synchronization over multiple hops. The tick synchronization between nodes is based on a relative reference tick. After the black burst synchronization of the network is done in the beginning, resynchronizations are performed periodically to clear the clock skews between different nodes. With the black burst (re-)synchronizations, all nodes have a common basis of time. MacZ guarantees the accuracy and robustness of synchronization against the changes of network topology. The medium is divided into macro slots of the same length. A macro slot is allocated between 2 synchronization ticks. Each macro slot is divided into a fixed number of micro slots of equal length. There are different kinds of micro slot groups (virtual slot regions) in each macro slot. They are Sync for synchronization slots, idle for unused slots, contention-free for slots based on contention-free medium access with reservations and contention-based for slots based contention-based medium access with priorities. These regions don't overlap with each other and the allocation of them is adjustable, but it is for all macro slots the same. Figure 2.3 presents an example of medium slotting and allocation of these virtual slot regions.

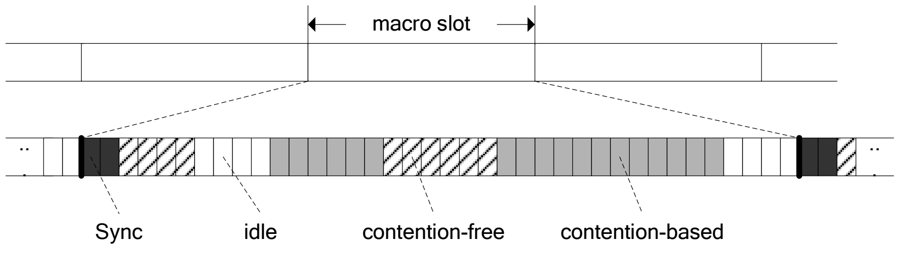


Figure 2.3: Medium slotting in MacZ [HIS07]

In contention-based transmission mode with priorities, the network nodes compete for the right to transmit over the medium. A mechanism like "Distributed Coordination Function" (DCF) in WLAN is deployed. In contention-based slot region, there is an arbitration phase with “carrier-sense multiple access with collision avoidance” (CSMA-CA) before the transmission of data frames. This arbitration phase is used to reduce the possibility that the nodes send data simultaneously. After this arbitration phase, a node will get the privilege to send. A priority value will be assigned by the higher layer as an attribute of each frame, which has a direct impact on the local sending order of frames. In contention-based transmission it is difficult to guarantee QoS because the actual transmission time is unknown and it will change when the network state changes.

In contention-free transmission mode with reservations, micro slots are used as reservation units. A reservation refers to a set of micro slots. The contention-free transmission mode assumes that the data will only be sent after the reservation succeeds. There is no arbitrations phase here like in contention-based transmission. MacZ provides services for the reservation protocol to reserve micro slots. When simultaneous reservations happen, a reservation collision occurs. Different reservations should use different micro slots within a predefined range and that should be guaranteed in the reservation protocol. With contention-free transmission mode with reservations, it is possible to guarantee QoS.

In order to reserve the micro slots in contention-free slot region, a reservation protocol called Black Burst Reservation Protocol (BBRP) [Xi07] was developed.

2.5. Black Burst Reservation Protocol (BBRP)

BBRP is based on MacZ and responsible for reservations of micro slots. BBRP supports decentralized reservations without a specific central master node or coordinator. In each node the information of reserved micro slots is maintained and stored. BBRP provides services for the higher layer to fulfill their demands (e.g. required bandwidth to number of micro slots to be reserved). If there are enough free micro slots available, the request will be accepted. Otherwise it will be rejected. Black burst frames are used in BBRP to spread reservation information. Each reservation has a deadline of several macro slots. When the deadline of a reservation expires, this reservation turns invalid and is released. This ensures that unused reservations are removed automatically. In order to keep the reservation valid, black burst frames should be periodically sent and propagated. The reservation of micro slots can be spread over multi-hops by sending more than one black burst frame in these micro slots. The Time-To-Live mechanism is used here, which means a node receiving N black bursts will propagate N-1 black bursts. In BBRP, the node, that wants to reserve a free micro slot, will send 2 black bursts in this slot, since any 2-hop neighbors can potentially interfere with the transmission within this micro slot according to the “hidden station” and “exposed node” problems [Bh94]. Figure 2.4 presents an example of this Time-To-Live mechanism. Node S sends 2 black bursts in the reserved micro slot to its direct neighbors. After receiving these black bursts, these neighbors mark this micro slot as reserved and send a single black burst to their direct neighbors in the same micro slot during the next macro slot.

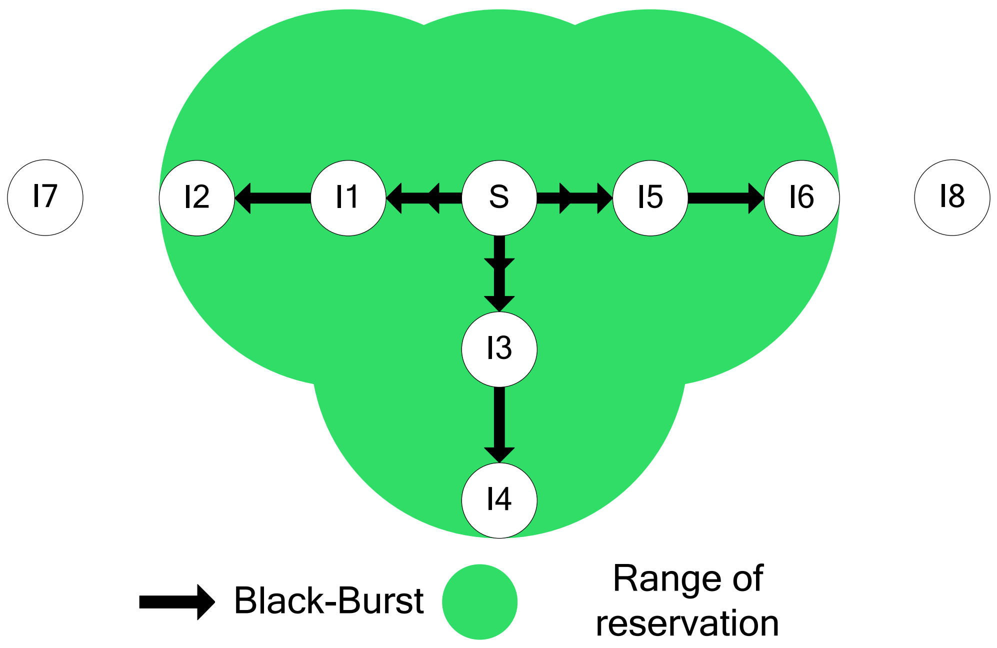


Figure 2.4: Time-To-Live flooding of black bursts

Different kinds of information need to be transmitted in a micro slot: the reservation information, the payload data and the acknowledge information for this data. Therefore we have 2 different kinds of frames in one micro slot, black burst frame and data frame. There are 2 kinds of data frame, application data frame and acknowledge data frame (ACK). In order to propagate the reservation of a micro slot to the 2-hop neighbors, 2 black burst frames are sent in this micro slot.

Having 2 positions of black burst frame in one micro slot is necessary. When 2 black burst frames overlap with each other in the medium, no collision will happen, since no meaningful data is transmitted in black bursts. If only one position of black burst frame instead of two is used, we get some problems. Firstly the propagation cannot stop, since the receiver of black burst cannot decide by itself when it should stop forwarding this black burst. Secondly after receiving the first black burst in a micro slot, the node cannot detect the following black burst in this micro slot, since the black burst frame received in this micro slot of last macro slot needs to be propagated in the same position. That means that even when the sender stops sending black burst to release the reservation, the other involved nodes cannot be informed.

Figure 2.5 shows an example structure of a micro slot. It begins with the black burst region, in which the black bursts are sent. There are 2 positions in this black burst region, which are reserved for 2 black burst frames. Then the payload data from application is sent in the application data region. In the end, ACK is sent to acknowledge the application data, when the data frame directed to it is correctly received.

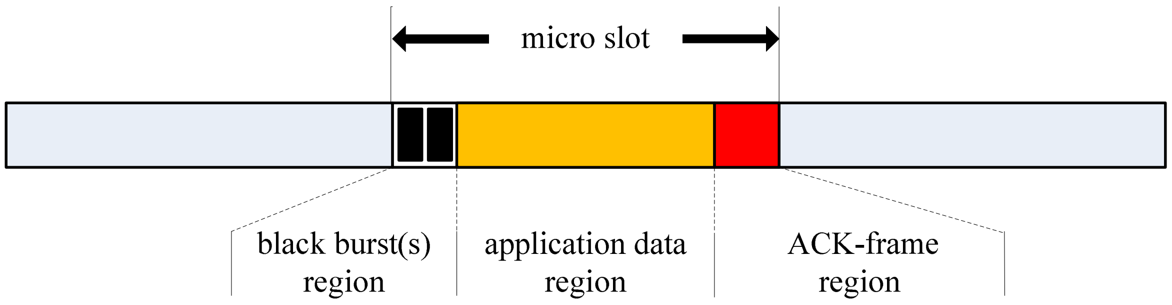


Figure 2.5: Structure of micro slot

Due the mobility of MANET, simultaneous reservations of the same slot could happen. Since black bursts are collision resistant, this could be detected only after the data collision at the receiver. The sender detects possible collision by the repeated absence of ACK-frames in the same micro slot. When the transmission failure happens several times, new random free slots will be chosen to replace the original faulty micro slots. Naturally it is possible that both involved sending nodes choose the same micro slots again and another reservation collision happens. But the possibility of encountering this situation become smaller and smaller. The changes of micro slots are reported to the routing protocol and may result in reroutings.

BBRP can be applied in multi-hop scenarios, where the source node and the destination node are not directly connected. The route between the source node and the destination node is assumed known. Each intermediate node serves as reservation creator and receiver. As shown in Figure 2.6, node I1 creates reservation in slot 5. Node I2 receives data in slot 5 and creates reservation in slot 9 and so on.

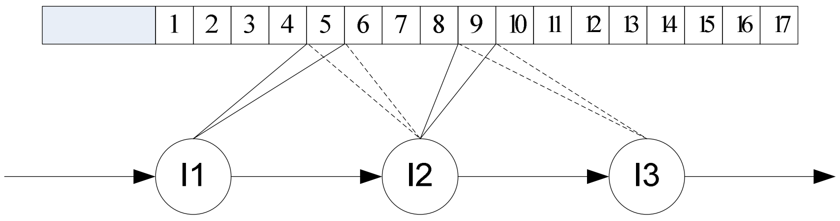


Figure 2.6: Hop by hop reservations

3. System Design

3.1. Design Goals and Challenges

Based on MacZ (indirectly) and BBRP (directly), a QoS unicast routing protocol is needed to find a QoS constraint route from source to destination. As presented in Figure 3.1, this protocol should use the functions offered by BBRP and provide its services to the higher layers. Among the protocols mentioned in chapter 6, AQOR [Xu03] with temporary reservations is the most suitable one to integrate with BBRP, which is based also on temporary resource reservations. The reservations in BBRP need to be renewed periodically with the propagation of black bursts. But AQOR cannot be directly integrated with BBRP because AQOR assumes that if the delay constraint can be fulfilled in the route discovery phase, it can also be fulfilled in the data transmissions phase, because both phases are based on contention-based medium access. However, in our design the data packets are sent in the contention-free period to meet the QoS demands. Route discovery could be done in both contention-based and contention-free period. We choose the contention-based period for the following reason. We assume that the entire route discovery in contention-based periods spends less time than in contention-free periods with a proper medium partition, since a message in the contention-free period can only be transmitted as soon as micro slots are reserved for it. A message could wait for nearly one macro slot before being sent out and it makes the route discovery very slow. However the concept of temporary reservation in AQOR can be utilized for designing a new QoS routing protocol to adapt to BBRP and MacZ.

A new protocol is developed based on the idea of AQOR. Two important QoS metrics, bandwidth and end-to-end delay, are taken into consideration. Resource reservation is in the charge of BBRP. Like in AQOR, all the reservations are temporary and will be released after their deadlines expire.

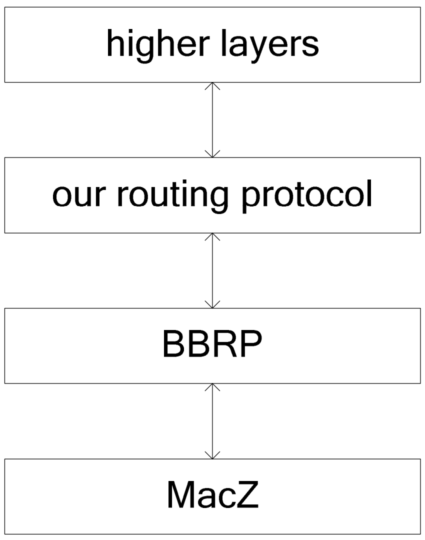


Figure 3.1: Position of routing protocol in the system

In the following sections, we discuss this routing protocol based on the following aspects: route discovery, resource reservation, route selection, route maintenance.

3.2. QoS network status maintenance

Network status information is useful for every node to maintain the local topology, traffic and mobility information. This information is used in this protocol in QoS route discovery, QoS violation detection and route recovery.

Every node in this network is required to send a "Hello" message periodically, informing the nodes within the distance of 2 hops of its existence (s. Figure 3.2). Every node manages a neighbors list named "NetworkStates" with the address of each neighbor, the distance to this neighbor (1 or 2 hops) and the reservation information of this neighbor. Without receiving any packet from one neighbor for T_{lost} period of time indicates that the link to this node is broken. The break of a link triggers route recovery when this link is part of any route in use. The "Hello" message additionally carries the resource reservation information, which is useful in the route discovery phase. A concrete example of the "Hello" message distribution is illustrated in chapter 3.4.

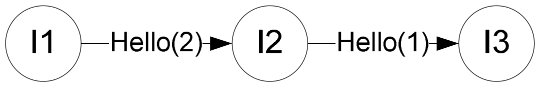


Figure 3.2 The propagation of Hello message to its neighbors within 2 hops with TTL = 2

3.3. Route Discovery

Route discovery phase is divided into two subphases: route request phase and route reply phase. We use limited flooding of route request packets (RREQ) in the first subphase and unicast transmission of route reply packets (RREP) in the second subphase. Figure 3.3 shows an example for the propagation of RREQ and RREP packet. In Figure 3.3A, source S broadcasts the RREQ packets through the network. Several of them reach destination D. In Figure 3.3B, node D chooses one of the requested routes and sends a RREP packet back to node S.

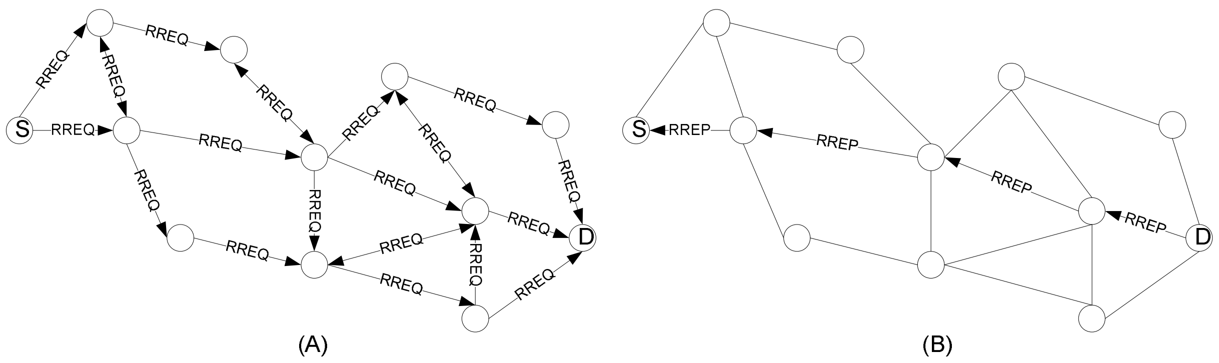


Figure 3.3: Example for the propagation of RREQ and RREP

In this chapter we use MSC (Message Sequence Charts) diagrams [MSC] to describe the mechanisms of our routing protocol. An example illustrated in Figure 3.4 describes the network topology and slots reservation situation of these MSCs in the following chapters.

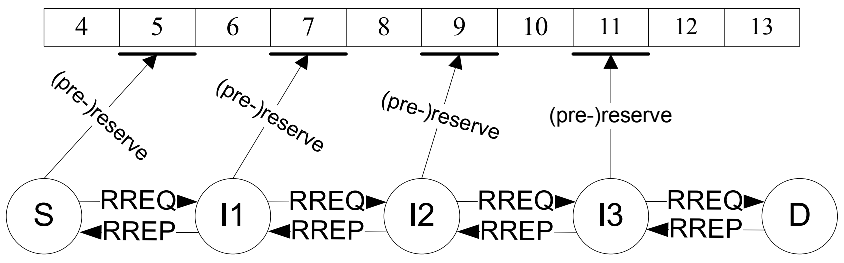


Figure 3.4: An example of RREQ flooding, RREP reply and reservations along the route

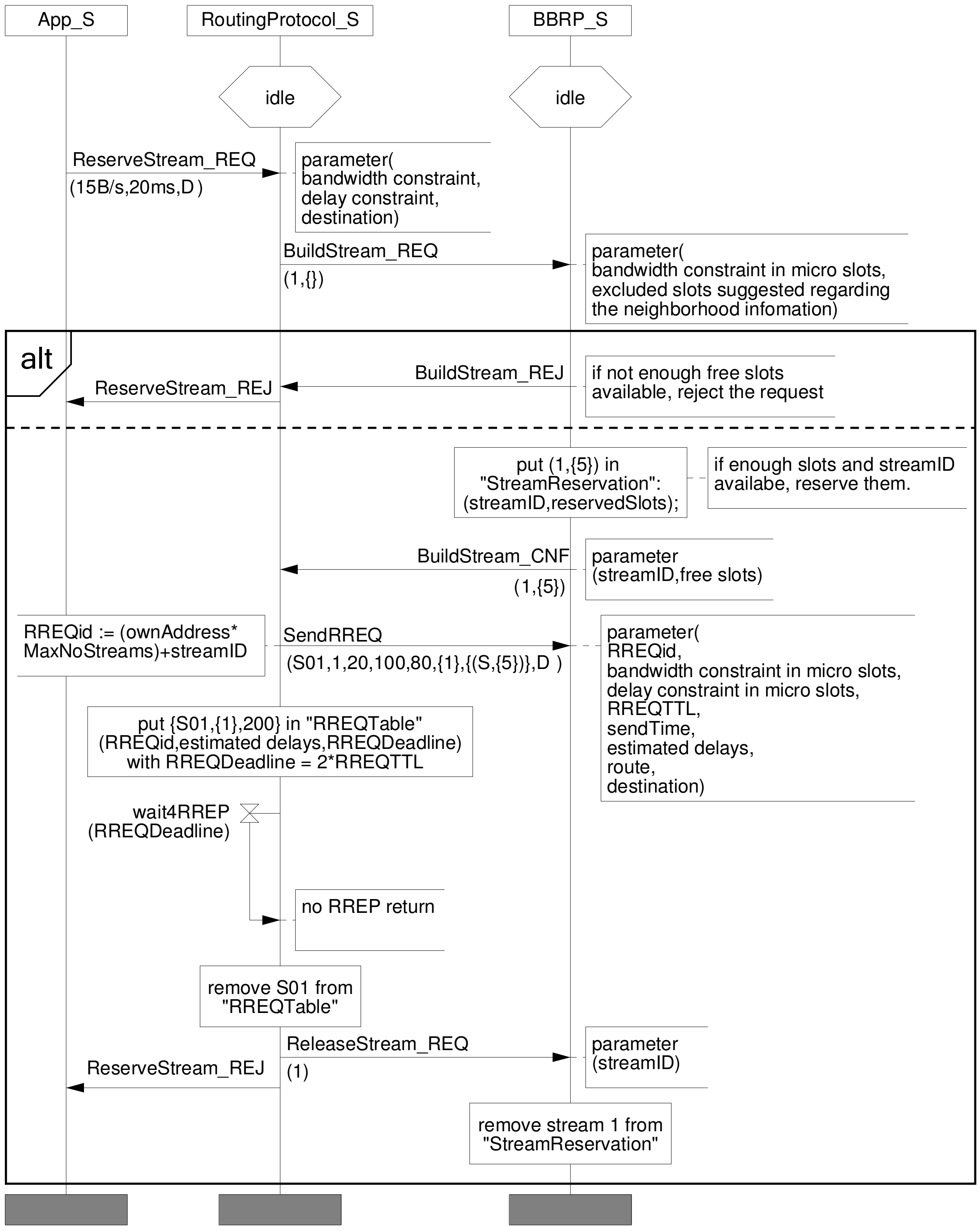
3.3.1. Route Request

A source node receives the QoS requirement from the application layer to find a route to the destination node *destination* with bandwidth constraint b_{min} and delay constraint d_{max}. After that, the source node will broadcast a RREQ packet, if bandwidth b_{min} is locally available. If there is not enough bandwidth available, a rejection will be sent to the application. The application will handle this rejection. For example, the application can repeat sending the old request or create a new request with compromised QoS constraints. The test of the availability of bandwidth is based on the local and neighborhood reservation information. When all the neighbors do not have enough bandwidth, no RREQ packets need to be broadcasted. Otherwise the source node will broadcast the RREQ packet, even when it is known that some nodes in the neighborhood cannot propagate this RREQ. The RREQ packet contains the following information:

1. *RREQid*	the RREQ identifier
2. b_{min}	bandwidth constraint in number of slots to reserve
3. d_{max}	delay constraint in number of slots to reserve
4. *RREQTTL*	the propagation time limit for RREQ in number of micro slots, “TimeToLive” of RREQ
5. *sendTime*	the number of the sending micro slot of RREQ in preceding node
6. d_{est}	set of currently estimated delays for each slot to reserve, $(d_1, d_2, \dots d_b)$ with $b = b_{min}$
7. *route*	set of travelled nodes and the prereserved slots at these nodes
8. *destination*	address destination node

RREQTTL is initialized with (*routeTTLunit / DMicroSlot*). *routeTTLunit* is a value predefined by the routing protocol (e.g. 100 ms) and is configurable. *DMicroSlot* is the duration of a micro slot and is defined by MacZ. *RREQTTL* denotes the remaining time of the validity of this RREQ packet. It decreases during the propagation of RREQ packets and when *RREQTTL* is smaller than 0, this RREQ packet is invalid and will not be propagated. *sendTime* denotes the number of the current sending micro slot of RREQ packets and will be updated when a RREQ packet is propagated. Having *RREQTTL* and *sendTime*, the duration of propagation of RREQ packets can be controlled. After broadcasting a RREQ packet, the source node sets a timer *wait4RREP* and waits for a RREP packet. The value of this timer is 2* *RREQTTL*, which means that the route should be discovered within 2 times the maximum duration of route request phase (*RREQTTL*). If no RREP packet for this RREQ packet is received within this period, the previous reservation for this RREQ packet will be released.

The details are illustrated in Figure 3.5. The part separated by the dashed lines describes the situation of alternatives. If a RREP packet with a route arrives in time, the route discovery finishes and the route will be used for data transmission. This will be discussed in chapter 3.3.3.



App_S
RoutingProtocol_S
BBRP_S
idle
idle
ReserveStream_REQ
(15B/s,20ms,D)
parameter(bandwidth constraint, delay constraint, destination)
BuildStream_REQ
(1,{})
parameter(bandwidth constraint in micro slots, excluded slots suggested regarding the neighborhood infomation)
alt
BuildStream_REJ
ReserveStream_REJ
if not enough free slots available, reject the request
put (1,{5}) in "StreamReservation": (streamID,reservedSlots);
if enough slots and streamID availabe, reserve them.
BuildStream_CNF
(1,{5})
parameter (streamID,free slots)
RREQid := (ownAddress* MaxNoStreams)+streamID
SendRREQ
(S01,1,20,100,80,{1},{(S,{5})},D)
parameter(RREQid, bandwidth constraint in micro slots, delay constraint in micro slots, RREQTTL, sendTime, estimated delays, route, destination)
put {S01,{1},200} in "RREQTable" (RREQid,estimated delays,RREQDeadline) with RREQDeadline = 2*RREQTTL
wait4RREP (RREQDeadline)
no RREP return
remove S01 from "RREQTable"
ReleaseStream_REQ
(1)
ReserveStream_REJ
parameter (streamID)
remove stream 1 from "StreamReservation"

Figure 3.5: Behavior of the source node in RREQ flooding

When an intermediate node receives a RREQ packet, it will first check the following conditions:

1. Let *microProMacro* be the number of micro slots pro macro slot,

   ((*current micro slot* ≥ *sendTime*) AND (*current micro slot* – *sendTime*) ≤ *RREQTTL*))

   //in the same macro slot

   OR

   ((*current micro slot* < *sendTime*) AND (*current micro slot* + *microProMacro* – *sendTime*) ≤ *RREQTTL*))

   //enter another macro slot

   This condition means the duration of routing request process has not exceeded the limit and the RREQ packet is still valid.

2. $\max\{ d_{est}(\mathrm{i})+d_{loc}(\mathrm{i}) \mid 1 \leq \mathrm{i} \leq b_{min}\} \leq d_{max}$ (d_{loc}(i) denotes the local delay caused by slot switching, e.g. receiving slot 5 and sending slot 7 causes a local delay of 2 slots.). This condition means that the current estimated delays have not exceeded the delay constraint.

3. There are enough free local micro slots to reserve.

4. This intermediate node has not already received other RREQ packets with the same ID or has received a RREQ packet with the same ID but with higher delay values than the current RREQ packet, which guarantees that this RREQ packet carries the route with the locally best QoS in terms of accumulated delays.

If one of these conditions is not fulfilled, the RREQ packet will be locally consumed instead of being propagated. Otherwise, the RREQ packet will be rebroadcasted. In order to reduce the delay for the route, micro slots just after the receiving micro slot are favored to choose as sending micro slots. For example, if the intermediate node receives data in micro slot 5, then it is better to propagate it in slot 6 than 7. Some information in the RREQ packet (e.g. *sendTime*, *RREQTTL*, d_{est}, *route*) has to be modified before the propagation. After propagating a RREQ packet, a timer *wait4RREP* will be set in the intermediate node in order to wait for the RREP packet. The value of *wait4RREP* is set as 2**RREQTTL*. When no RREP packet for this RREQ packet is received within this period, the previous prereservation will be released. The details are illustrated in Figure 3.6. The part separated by the dashed lines describes the situation of alternatives. If a RREP packet with a route arrives in time, the intermediate node will forward the RREP packet back to source. This will be discussed in chapter 3.3.3.

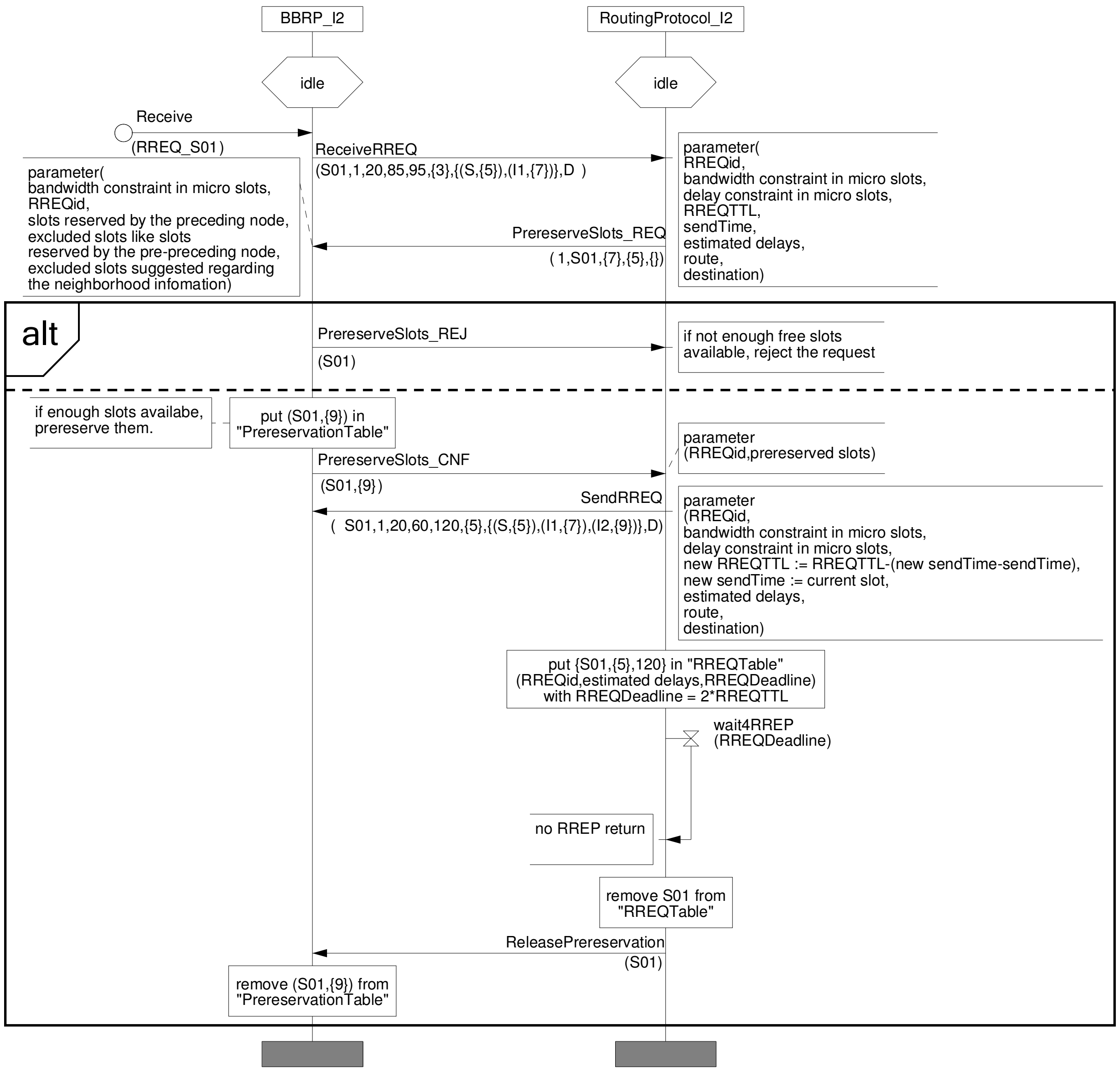

BBRP_I2
RoutingProtocol_I2
idle
idle
Receive
(RREQ_S01)
ReceiveRREQ
(S01,1,20,85,95,{3},{(S,{5}),(I1,{7})},D)
parameter(
RREQid,
bandwidth constraint in micro slots,
delay constraint in micro slots,
RREQTTL,
sendTime,
estimated delays,
route,
destination)
parameter(
bandwidth constraint in micro slots,
RREQid,
slots reserved by the preceding node,
excluded slots like slots
reserved by the pre-preceding node,
excluded slots suggested regarding
the neighborhood infomation)
PrereserveSlots_REQ
(1,S01,{7},{5},{})
alt
PrereserveSlots_REJ
(S01)
if not enough free slots
available, reject the request
if enough slots availabe,
prereserve them.
put (S01,{9}) in
"PrereservationTable"
parameter
(RREQid,prereserved slots)
PrereserveSlots_CNF
(S01,{9})
SendRREQ
(S01,1,20,60,120,{5},{(S,{5}),(I1,{7}),(I2,{9})},D)
parameter
(RREQid,
bandwidth constraint in micro slots,
delay constraint in micro slots,
new RREQTTL := RREQTTL-(new sendTime-sendTime),
new sendTime := current slot,
estimated delays,
route,
destination)
put {S01,{5},120} in "RREQTable"
(RREQid,estimated delays,RREQDeadline)
with RREQDeadline = 2*RREQTTL
wait4RREP
(RREQDeadline)
no RREP return
remove S01 from
"RREQTable"
ReleasePrereservation
(S01)
remove (S01,{9}) from
"PrereservationTable"

Figure 3.6: Behavior of the intermediate node in RREQ flooding

When the first RREQ packet arrives at the destination node, it will be accepted and stored in the local list “RouteStorage”. After this, a timer *wait4OtherRREQs* is set in order to wait for other RREQ packets with the same ID. The value of *wait4OtherRREQs* is set as *RREQTTL*. After this timer expires, no RREQ packets with this ID will be accepted. The destination node will select from the collected routes the one with the best estimated delay and reply with a RREP packet. The details are illustrated in Figure 3.7.

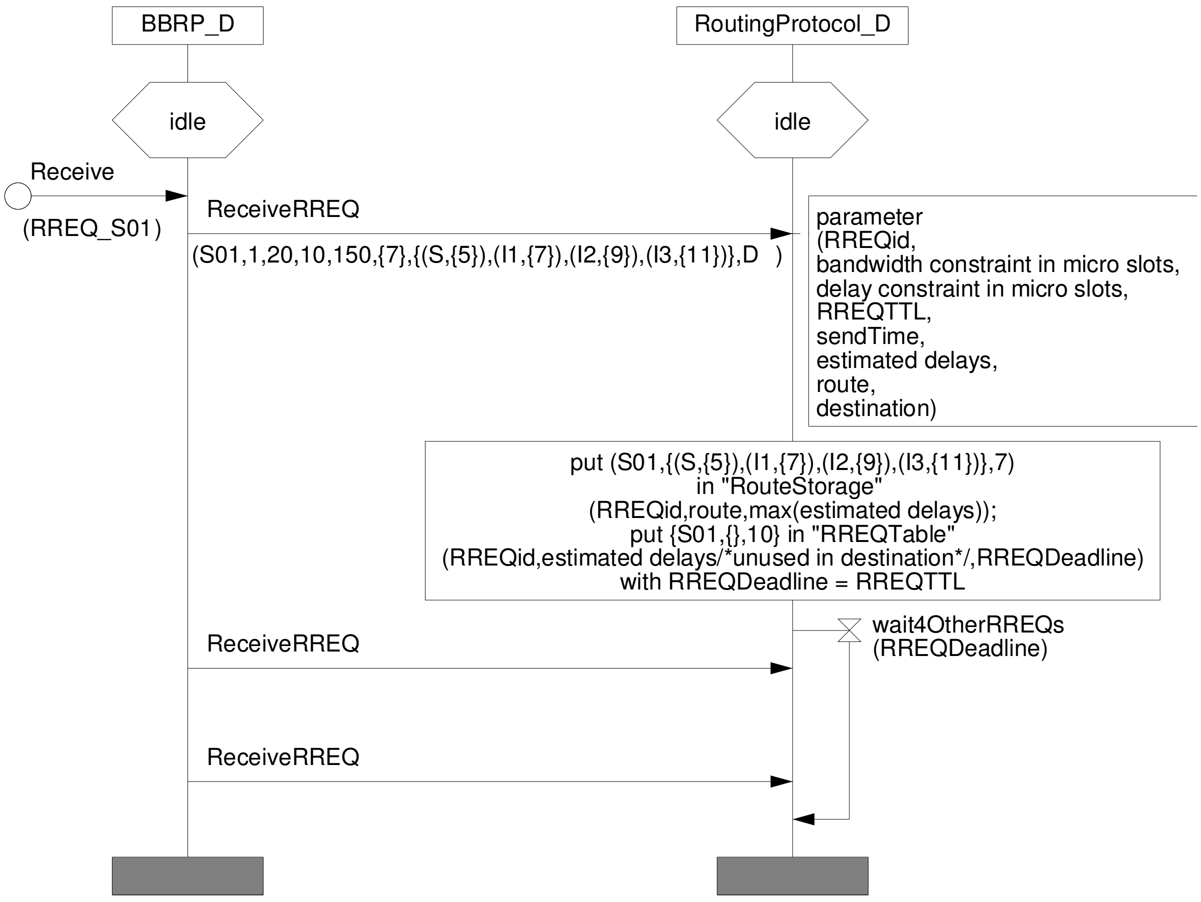

BBRP_D
RoutingProtocol_D
idle
idle
Receive
(RREQ_S01)
ReceiveRREQ
(S01,1,20,10,150,{7},{(S,{5}),(I1,{7}),(I2,{9}),(I3,{11})},D)
parameter
(RREQid,
bandwidth constraint in micro slots,
delay constraint in micro slots,
RREQTTL,
sendTime,
estimated delays,
route,
destination)
put (S01,{(S,{5}),(I1,{7}),(I2,{9}),(I3,{11})},7)
in "RouteStorage"
(RREQid,route,max(estimated delays));
put {S01,{},10} in "RREQTable"
(RREQid,estimated delays/*unused in destination*/,RREQDeadline)
with RREQDeadline = RREQTTL
wait4OtherRREQs
(RREQDeadline)
ReceiveRREQ
ReceiveRREQ

Figure 3.7: Behavior of the destination node in REEQ flooding

3.3.2. Limited RREQ Flooding

Unlimited RREQ flooding will create unnecessary communication overhead and increase the entire route discovery time. Therefore we use TimeToLive to limit flooding. As described in chapter 3.3.1, *RREQTTL* is initialized as the maximum RREQ propagation time. During the propagation of RREQ packets, the value of *RREQTTL* decreases following the formula:

$$\text{new } RREQTTL := \text{old } RREQTTL - (\text{new } sendTime - \text{old } sendTime)$$

When it is smaller than 0, the RREQ packet will not be propagated. It is used to limit the propagation range of RREQ packets and to define the wait timer of each node.

In order to limit flooding further, a QoS comparison mechanism as used in QuaSAR[04] is applied to each flow, which means only RREQ packets with lower estimated delays will be propagated in the intermediate nodes. In order to achieve it, the *RREQid* parameter is used in the RREQ packet as an identifier. Each node has a *RREQTable* for saving *RREQid* and its estimated delays. When a RREQ packet arrives at an intermediate node, this node checks first if another RREQ packet with the same *RREQid* arrived before. If so, the comparison of estimated delay values between the new and old RREQ packets is made. If one of the estimated delay values of the new RREQ packet is lower than the corresponding one in *RREQTable*, the delay values in *RREQTable* will be updated and the new RREQ packet will be propagated. Otherwise it will be consumed and no change happens in *RREQTable*.

For instance in Figure 3.8, source node S tries to find a route with the lowest delay to destination D. First in Figure 3.8A, node S propagates RREQ packets with the reservation of micro slot 5 to its neighbors I1 and I2. Then in Figure 3.8B, the node I1 and I2 broadcast RREQ packets with reservation of slot 13 and slot 7 separately. I2 receive a RREQ packet from I1 with a higher delay value (13–5)+1 = 9, comparing to the value (7–5)+1 = 3 in the *RREQTable*, so this RREQ packet will be ignored. I1 receives a RREQ packet from I2 with a lower delay value 3. After getting the free slot 8 from BBRP, the estimated delay (8–7)+3 = 4 is still lower than the value 8 registered in the *RREQTable* (s. Figure 3.8C). Therefore this RREQ packet will be accepted and rebroadcasted. Finally the destination collects all the received routes and replies to the one with the lowest delay with a RREP packet. In Figure 3.8D the route {S, I2, I1, D} has the lowest delay: 4 micro slots.

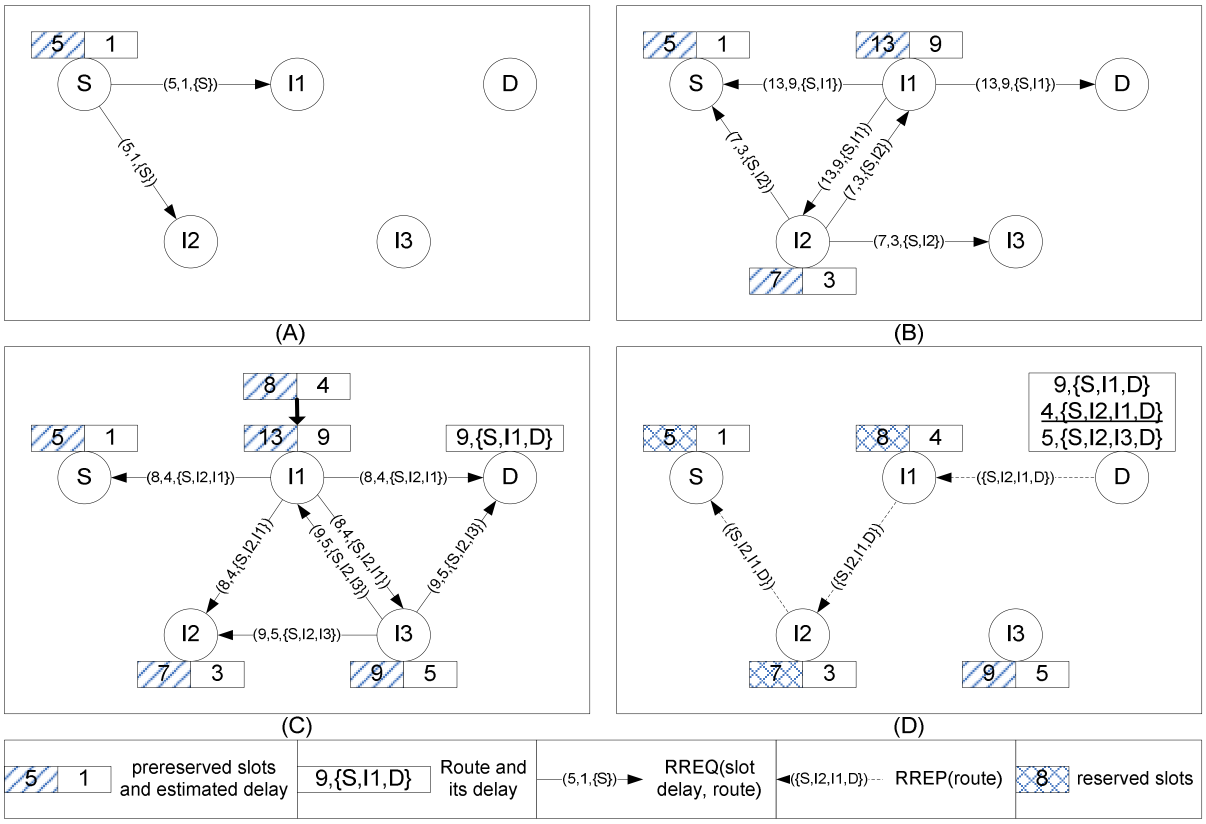


Figure 3.8: Propagation of RREQ

3.3.3. Route Reply

After collecting several RREQ packets before the expiration of timer *wait4OtherRREQs*, destination chooses the one with the shortest estimated delay and sends a RREP packet back to the source node along the reverse path of the corresponding RREQ packet in the contention-based period. A RREP packet contains the entire route and the reservations in each node. During the propagation of the RREP packet, the *RoutingTable* (*slotIn, slotOut, nextHop*) in each intermediate node is also updated with the information carried by the RREP packet and the prereserved micro slots turn their states into *reserved*. When a prereserved micro slot cannot be reserved during the route reply phase, the intermediate node will try to find a replacement without causing any delay violation, which is defined as follows:

Let *microProMacro* be the number of micro slots pro macro slot, b the bandwidth constraint in number of slots, $C = (c_1, \ldots, c_b)$ the slots reserved in the current node, $P = (p_1, \ldots, p_b)$ the ones in the preceding node, $S = (s_1, \ldots, s_b)$ the ones in the succeeding node and $c_i \in C$ $(1 \leq i \leq b)$ a unreservable slot. In order to keep the original delay, if the succeeding node is not the destination, the replacement slot r_i must be chosen in interval (p_i, s_i) if $s_i > p_i$ or in interval $(p_i, microProMacro - 1) \cup (0, s_i)$ if $s_i < p_i$. If the succeeding node is the destination, where no reservation is made, then $r_i \in (p_i, c_i)$ if $c_i > p_i$ or $r_i \in (p_i, microProMacro - 1) \cup (0, c_i)$ if $c_i < p_i$.

The details of slot replacement are described in Figure 3.10. If such a micro slot does not exist, this RREP packet will not be forwarded in this case. The source node and some intermediate nodes will not receive any RREP packet, so their timers *wait4RREP* will expire and the previously (pre)reserved resources will be released. Some intermediate nodes, which have already received the RREP packet, will release the reservations according to the reservation release mechanism in BBRP. As defined in BBRP, a reservation will be released after several macro slots without renewing the deadline of this reservation. When the RREP packet arrives at the source node, the *RoutingTable* will also be updated and the routing protocol informs the application of the success of the route discovery. That means the end of route discovery phase and the application can now send data packets. The propagation mechanisms of RREP packets in the destination, intermediate and source nodes are illustrated in Figure 3.9, Figure 3.10 and Figure 3.11. The parts separated by the dashed lines describe the situation of alternatives.

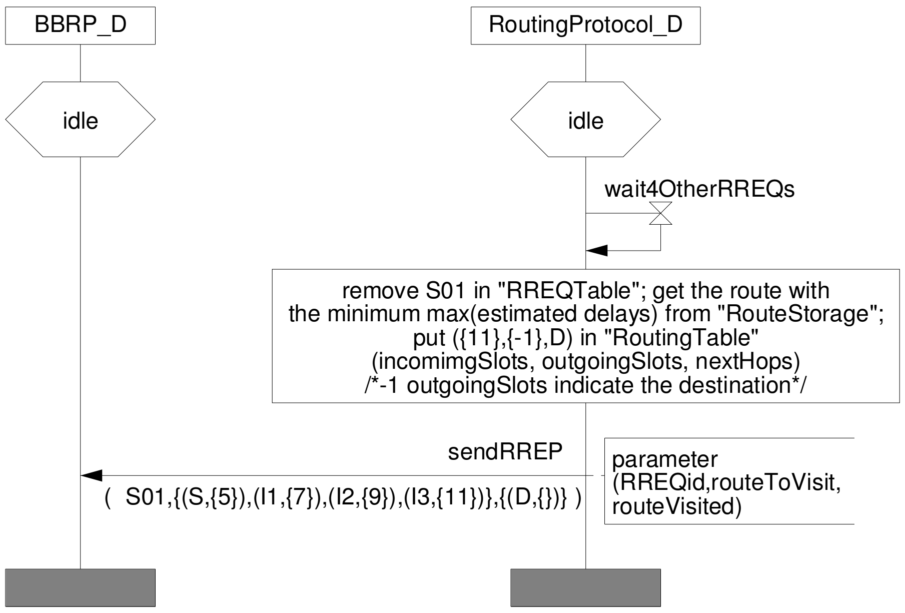

BBRP_D
RoutingProtocol_D
idle
idle
wait4OtherRREQs
remove S01 in "RREQTable"; get the route with the minimum max(estimated delays) from "RouteStorage"; put ({11},{-1},D) in "RoutingTable" (incomimgSlots, outgoingSlots, nextHops) /*-1 outgoingSlots indicate the destination*/
sendRREP
(S01,{(S,{5}),(I1,{7}),(I2,{9}),(I3,{11})},{(D,{})})
parameter (RREQid,routeToVisit, routeVisited)

Figure 3.9: Behavior of the destination node in unicast RREP transmission

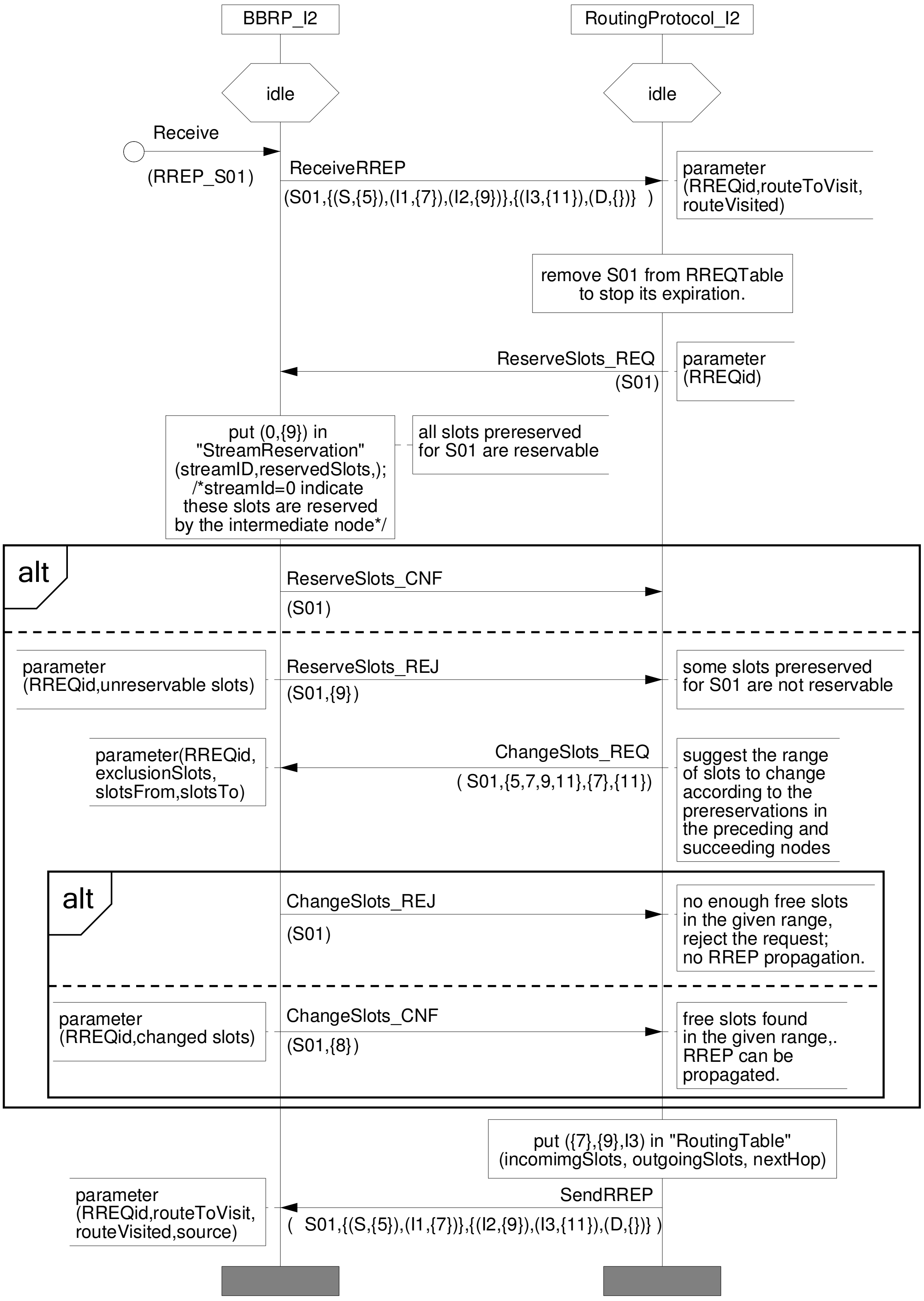
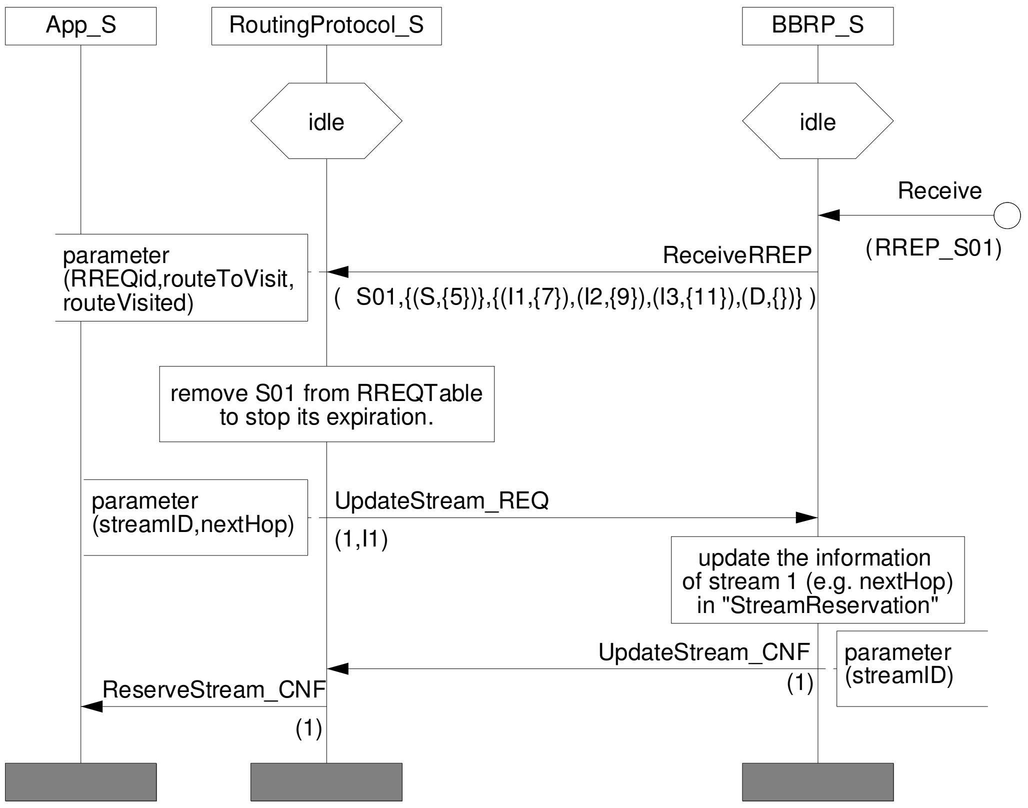

BBRP_I2
RoutingProtocol_I2
idle
idle
Receive
(RREP_S01)
ReceiveRREP
(S01,{(S,{5}),(I1,{7}),(I2,{9})},{(I3,{11}),(D,{})})
parameter (RREQid,routeToVisit, routeVisited)
remove S01 from RREQTable to stop its expiration.
ReserveSlots_REQ
(S01)
parameter (RREQid)
put (0,{9}) in "StreamReservation" (streamID,reservedSlots,); /*streamId=0 indicate these slots are reserved by the intermediate node*/
all slots prereserved for S01 are reservable
alt
ReserveSlots_CNF
(S01)
parameter (RREQid,unreservable slots)
ReserveSlots_REJ
(S01,{9})
some slots prereserved for S01 are not reservable
parameter(RREQid, exclusionSlots, slotsFrom,slotsTo)
ChangeSlots_REQ
(S01,{5,7,9,11},{7},{11})
suggest the range of slots to change according to the prereservations in the preceding and succeeding nodes
alt
ChangeSlots_REJ
(S01)
no enough free slots in the given range, reject the request; no RREP propagation.
parameter (RREQid,changed slots)
ChangeSlots_CNF
(S01,{8})
free slots found in the given range,. RREP can be propagated.
put ({7},{9},I3) in "RoutingTable" (incomimgSlots, outgoingSlots, nextHop)
parameter (RREQid,routeToVisit, routeVisited,source)
SendRREP
(S01,{(S,{5}),(I1,{7})},{(I2,{9}),(I3,{11}),(D,{})})

Figure 3.10: Behavior of the intermediate node in unicast RREP transmission



App_S
RoutingProtocol_S
BBRP_S
idle
idle
Receive
(RREP_S01)
ReceiveRREP
(S01,{(S,{5})},{(I1,{7}),(I2,{9}),(I3,{11}),(D,{})})
parameter (RREQid,routeToVisit, routeVisited)
remove S01 from RREQTable to stop its expiration.
parameter (streamID,nextHop)
UpdateStream_REQ
(1,I1)
update the information of stream 1 (e.g. nextHop) in "StreamReservation"
UpdateStream_CNF
(1)
parameter (streamID)
ReserveStream_CNF
(1)

Figure 3.11: Behavior of the source node in unicast RREP transmission

3.4. Resource Reservation

The reservation of micro slots and the propagation of reservations are achieved by BBRP. There are 3 internal states for each micro slot in BBRP: *free*, *prereserved* and *reserved*.

Definitions:

- A micro slot is *free* iff no 2-hop neighbor has reserved this slot.
- A micro slot is *prereserved* iff this slot is allocated by a RREQ packet but the corresponding RREP packet has not yet arrived to confirm the reservation.
- A micro slot is *reserved* iff the reservation is confirmed by the RREP packet and the slot can be used for data transmission.

As the RREQ packet propagates from source to destination, the slot state is changed from *free* to *prereserved* in the intermediate nodes. This information will be stored locally in BBRP and the reservation will not be propagated to the neighboring nodes. As the RREP packet propagates from the destination to the source, the corresponding slots change their states from *prereserved* to *reserved*. The reservation of these slots will now be propagated into the network.

As illustrated in Figure 3.12, source S sends a RREQ packet to the intermediate node I1 after prereserving micro slot 5. Then I1 prereserves micro slot 7 and forwards the RREQ packet to I2 and so on. After the RREQ packet arrives at the destination, a RREP packet is sent to answer it, if the route in this RREQ packet is selected by the destination. The bandwidth reservation along the route is finished when the RREP packet reaches the source.

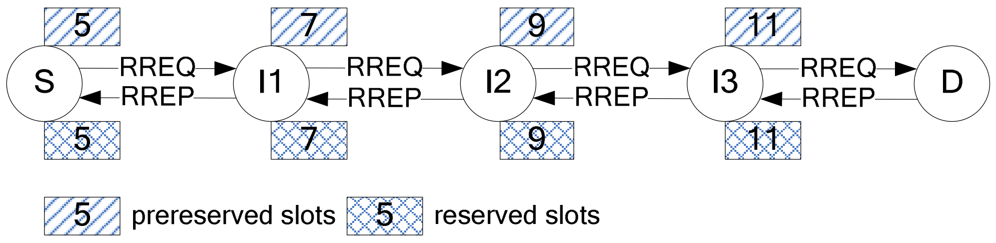
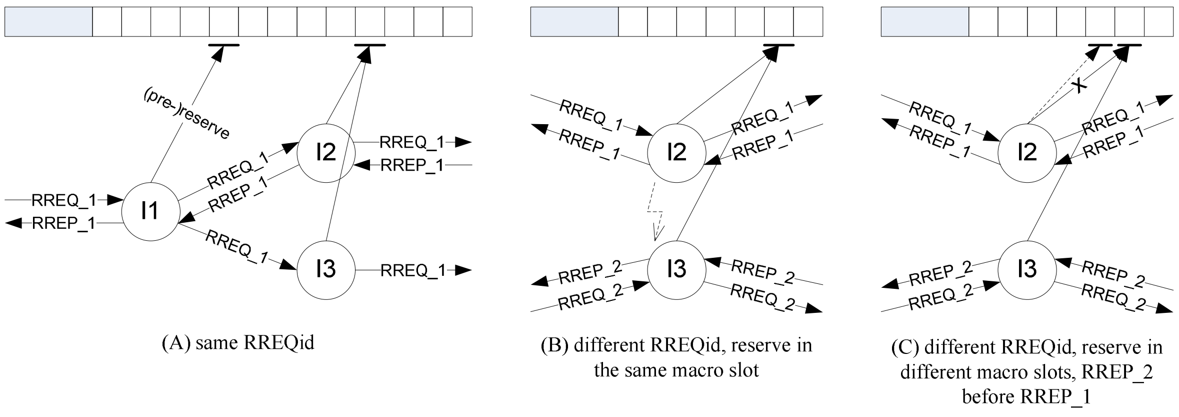


Figure 3.12: Prereservation in route request phase and reservation in route reply phase

Not all RREQ packets can reach destination. Some of them are consumed by the intermediate nodes because of different reasons like the expiration of time limit. Therefore bandwidth prereserved for such routes must be released. This is realized by the reservation release mechanisms of BBRP with expiration of a predefined timer. Each micro slot prereserved in the route request phase has a deadline (timer *wait4RREP*). If the deadline expires, the corresponding micro slot will be released by BBRP (the state changes from *prereserved* to *free*) without informing the routing protocol. Any RREP packet, which arrives after the expiration of timer *wait4RREP*, will be ignored.

With the addition of the *prereserved* state the reservation stays local in each node during the RREQ flooding. Therefore it is possible that two neighboring nodes reserve the same slot. If the reservations in these two nodes belong to the same RREQ flooding, no collision will happen because there could be only one node of them to be chosen as part of the route selected at the destination. If the reservations belong to two different RREQ flooding, there are 2 possibilities. The first one is that both nodes reserve this micro slot in the same macro slot. This can be detected later in the data transmission phase and this will be handled by BBRP with the replacement of free slots. No error is detected when the states are changed from *prereserved* to *reserved* and when the reservations are propagated. The second one is that both nodes reserve this micro slot in different macro slots, which means one of both neighboring nodes, who reserves this micro slot later, can detect the error. After finding out that a prereserved slot cannot be reserved in the route reply phase, the node will try to find a replacement without violating the delay constraint. That means the replacement micro slot must be located between the one reserved by the preceding node and the one by the succeeding node (s. chapter 3.3.3). In Figure 3.13, three examples of the situations described above are presented. In Figure 3.13A, the intermediate nodes I2 and I3 prereserve the same slot for the RREQ packet with the same ID (RREQ_1). Afterwards I2 was selected as part of the route with the propagation of RREP. The state of this slot in I2 turns from *prereserved* to *reserved* and in I3 the state turns from *prereserved* to *free* after the expiration of timer. In Figure 3.13B, the intermediate nodes I2 and I3 prereserve the same slot for RREQ packets with different IDs (RREQ_1 and RREQ_2). If both nodes are selected as part of its route and this slot is reserved by the RREP packets during the same macro slot, the interfering reservations will be detected when the collisions in the data transmission occur. In Figure 3.13C, the same slot is reserved by RREP_1 and RREP_2 in different macro slots, RREP_2 before RREP_1. After receiving RREP_1, node I2 checks its reservation map and finds out that the slots to be reserved were already reserved by some other nodes (e.g. I3). Then I2 finds a replacement slot which causes no delay violation and propagates RREP_1.



(pre-)reserve
RREQ_1
RREP_1
I1
I2
I3
(A) same RREQid
RREQ_2
RREP_2
(B) different RREQid, reserve in the same macro slot
(C) different RREQid, reserve in different macro slots, RREP_2 before RREP_1

Figure 3.13: Prereservation of the same slots in neighboring nodes

There are several rules to follow in the selection of micro slots to prereserve in the RREQ flooding phase.

- Do not choose the micro slots prereserved by the preceding 2 hops in RREQ packets to avoid the interference of reservation.
- Choose the micro slots with the smallest possible index difference to make the delay as low as possible.

But sometimes following the 2 rules above is not enough. For instance in Figure 3.14, the micro slots 1, 2, 3 are free in the node I1 and I2 and in the node I3 only the micro slots 1, 2. The RREQ packet is propagated along these nodes and slots are prereserved. According to the rules, micro slot 1 is prereserved in I1 and micro slot 2 in I2. When the RREQ packet arrives at I3, no micro slot is available. But, if I1 and I2 choose slot 3 and 2, there would still be a free slot (slot 1) remaining at I3, which means the RREQ could be propagated further.

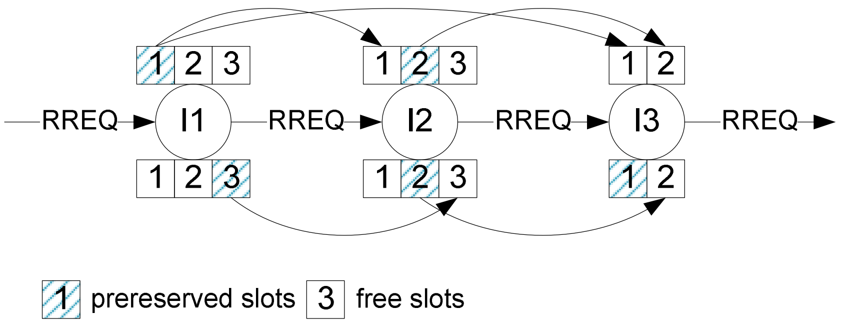


Figure 3.14: The propagation stops due to the unthoughtful strategy of slot selection

The cause of this problem is the lack of neighborhood information. In the example above, if I1 and I2 are informed before the prereservation that there are only 2 free micro slots (slot 1, 2) remaining at I3, they could choose not to reserve both of them. Therefore we add another rule in the selection of micro slots to prereserve:

- Let b_{min} be the bandwidth constraint in number of micro slots.

  If the destination is a 1-hop neighbor, consider only the local reservation information.

  If the destination is a 2-hop neighbor, try not to prereserve the free micro slots of all the 1-hop neighbors with [b_{min}, $2*b_{min}$) free slots remaining.

If the destination is neither 1-hop nor 2-hop neighbor, try not to prereserve the free micro slots of all the 2-hop neighbors with [b_{min}, 3*b_{min}) free slots and all the 1-hop neighbors with [b_{min}, 2*b_{min}) free slots remaining.

Any neighbor with less than b_{min} slots cannot propagate RREQ in all cases. Therefore the reservation situation of these nodes needs not to be considered. We use in the third rule "…try not to…" because sometimes the current node can not find enough free slots in order to fulfill the third rule. In such cases, the current node must prereserve these slots to flood the RREQ and the neighbors with scare bandwidth can only be ignored. With the third rule, the problem described in Figure 3.14 can be solved. This rule is only practicable with the help of the neighborhood information maintenance. The "Hello" message mentioned in chapter 3.2 was used to carry this information, which is stored in the neighborhood table *NetworkStates* and is periodically updated (s. Figure 3.15). The "Hello" message is sent by BBRP of sender node after collecting the necessary information and is received and used by routing protocol of receiver node.

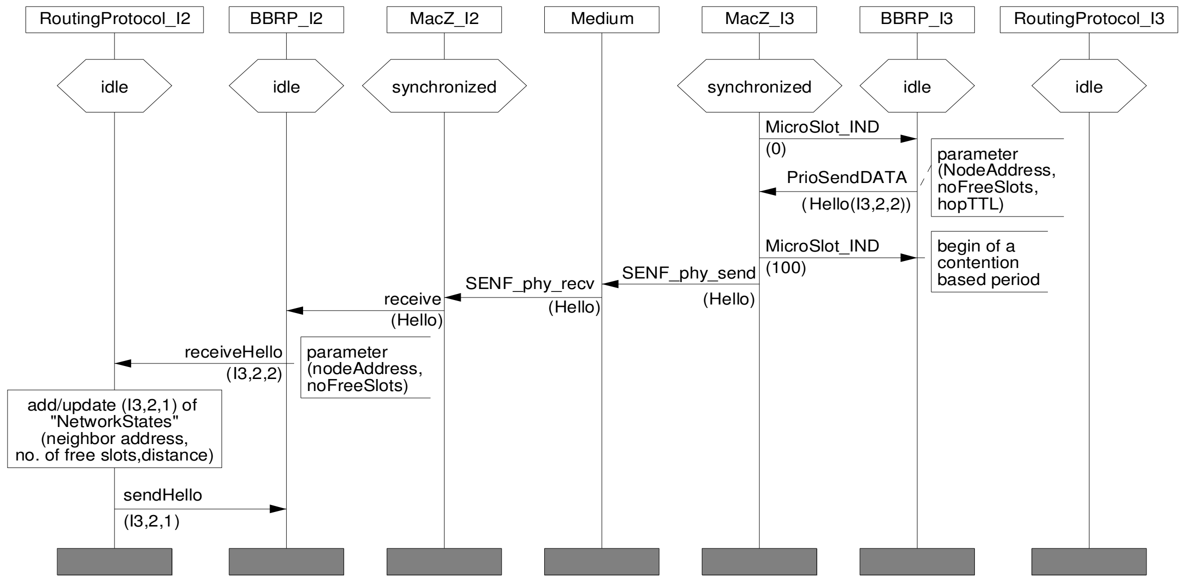


Figure 3.15: Sending and Reception the "Hello" message

During the data transmission, each reservation in the intermediate node has a deadline (predefined value of number of macro slots), which is renewed as long as this micro slot is still in use. We assume that the data packet received in micro slot *slotIn* in an intermediate node will be sent in micro slot *slotOut*. After receiving a reservation renewal frame in *slotIn*, the deadline of *slotOut* will be renewed. After the source finishes the data transmission, the micro slots reserved for this transmission stream must be released. The release is done in the source node, the intermediate node, the destination node, respectively. At the source, the application sends a release message to BBRP to release the resource explicitly. At the intermediate node and the destination, the reserved micro slots are released hop by hop implicitly after their deadlines expire. Any information concerning these micro slots in the routing table will also be removed.

3.5. Route Selection

In this protocol routes are selected by destination nodes. Any RREQ arriving at the destination node denotes a route with fulfilled bandwidth and delay constraint. Then the destination chooses the one with the smallest delay value. There are 2 main control mechanisms, bandwidth control and end-to-end delay control, which support the selection and guarantee that the selected route meets the QoS demands.

3.5.1. Bandwidth Control

Bandwidth is mapped to number of micro slots. The routing protocol informs BBRP of number of micro slots to reserve and BBRP replies with the IDs of reserved micro slots or rejects if not enough slots are available. If the reservation is rejected, the intermediate node will stop propagating the RREQ packet locally.

3.5.2. End-to-end Delay Control

During the route request phase, the estimated delays are calculated in each intermediate node and compared with the bandwidth constraint d_{max}. If any one of the current estimated delays exceeds d_{max}, the intermediate node will stop forwarding RREQ packet.

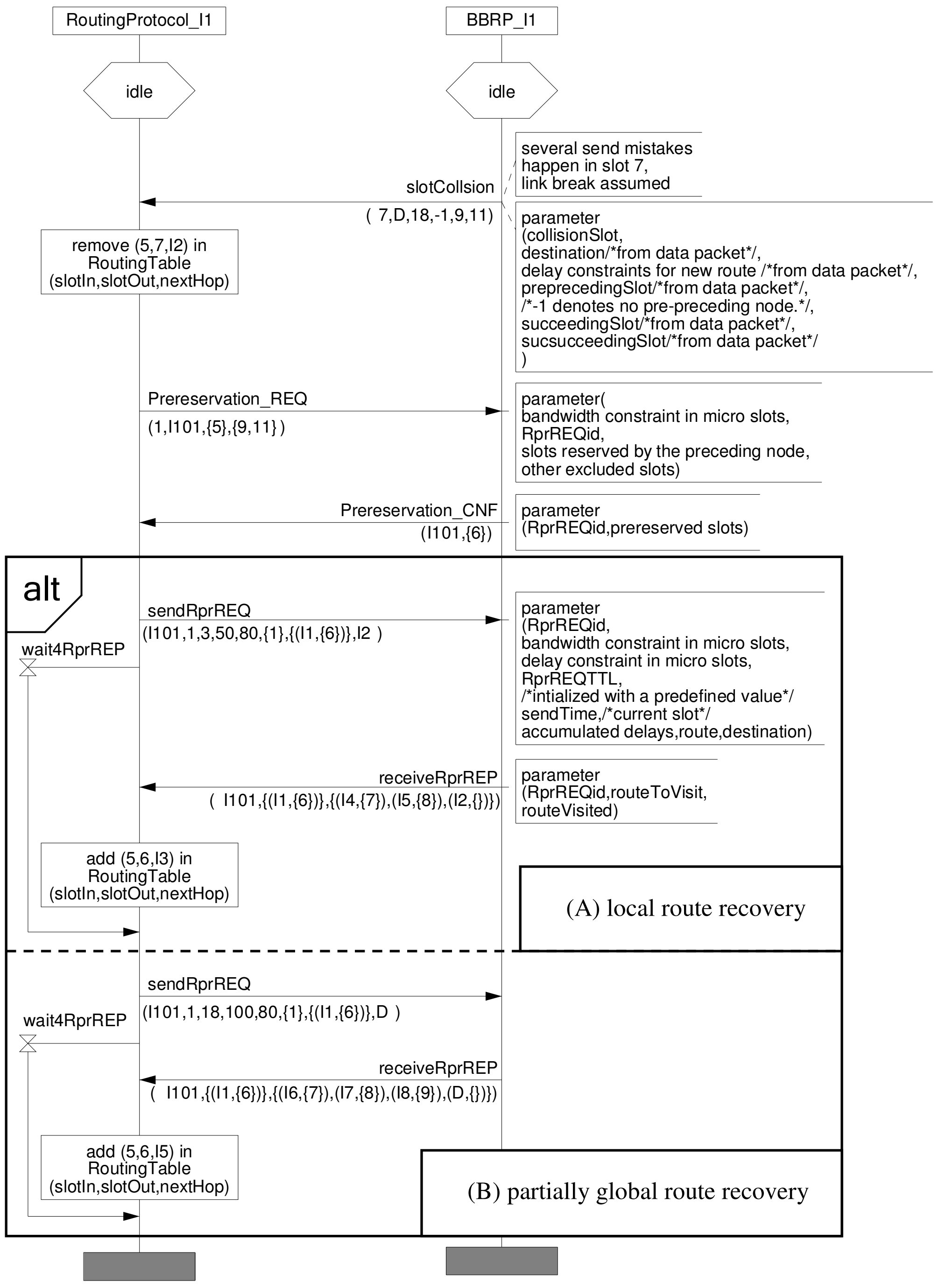
3.6. Route Maintenance

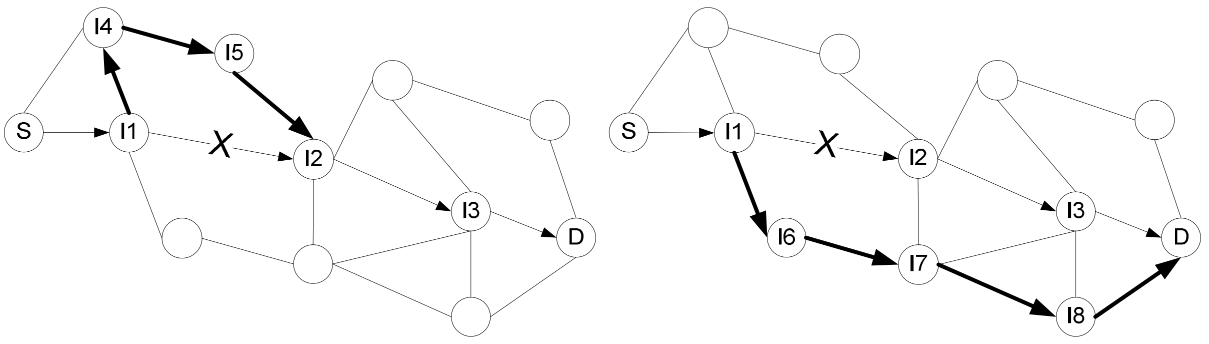
There are two parts in QoS violation detection: bandwidth violation detection and delay violation detection. After detecting that the same QoS violation happens several times successively, some recovery mechanisms will be triggered to repair them.

3.6.1. Bandwidth Violation Detection and Recovery

Bandwidth violation can happen due to route breaks or transmission interferences caused by node mobility. It is first detected by BBRP when the node fails to receive ACK packet after sending out the data in the reserved micro slot. When it happens several times (number of times is defined by BBRP), BBRP will offer a new free micro slot to the routing protocol and replace the old one in the routing table. When BBRP changes micro slots several times (number of times is defined by routing protocol) and there are still no ACKs replied, a route break is assumed.

There are 2 possibilities to repair the route. One possibility is to repair the route locally. That means we find a new route to bridge the broken point. The advantage of this method is that the other parts of the route can be reused. Another possibility is to repair the route partially globally. A rerouting will be started at the last working node and to find a new route between this node to the destination. The necessary information (e.g. delay constraint for the new route, destination address) for rerouting can be obtained by the data packet. Figure 3.16 presents the examples of locally (A) and partially globally (B) route recovery. In our route protocol both will be implemented and deployed in different networks. Networks with large maximum diameter or with relative stationary nodes tend to use the local variant, while networks with small maximum diameter or high mobility tend to use the partially global variant.


RoutingProtocol_I1
BBRP_I1
idle
idle
several send mistakes happen in slot 7, link break assumed
slotCollsion
(7,D,18,-1,9,11)
parameter (collisionSlot, destination/*from data packet*/, delay constraints for new route /*from data packet*/, prepreceding Slot/*from data packet*/, /*-1 denotes no pre-preceding node.*/, succeedingSlot/*from data packet*/, sucsucceedingSlot/*from data packet*/)
remove (5,7,I2) in RoutingTable (slotIn,slotOut,nextHop)
Prereservation_REQ
(1,I101,{5},{9,11})
parameter(bandwidth constraint in micro slots, RprREQid, slots reserved by the preceding node, other excluded slots)
Prereservation_CNF
(I101,{6})
parameter (RprREQid,prereserved slots)
alt
sendRprREQ
(I101,1,3,50,80,{1},{(I1,{6})},I2)
wait4RprREP
parameter (RprREQid, bandwidth constraint in micro slots, delay constraint in micro slots, RprREQTTL, /*intialized with a predefined value*/ sendTime,/*current slot*/ accumulated delays,route,destination)
receiveRprREP
(I101,{(I1,{6})},{(I4,{7}),(I5,{8}),(I2,{})})
parameter (RprREQid,routeToVisit, routeVisited)
add (5,6,I3) in RoutingTable (slotIn,slotOut,nextHop)
(A) local route recovery
sendRprREQ
(I101,1,18,100,80,{1},{(I1,{6})},D)
wait4RprREP
receiveRprREP
(I101,{(I1,{6})},{(I6,{7}),(I7,{8}),(I8,{9}),(D,{})})
add (5,6,I5) in RoutingTable (slotIn,slotOut,nextHop)
(B) partially global route recovery



(A) local route recovery

(B) partially global route recovery

Figure 3.16: two types of route recovery

3.6.2. Delay Violation Detection and Recovery

Delay violation denotes that the actual data transfer time exceeds the allowed maximum d_{max}. It can be caused if the reserved micro slot is changed. Delay violation is detected at the side of destination. Naturally the delay violation can be detected earlier in an intermediate node. But the problem is if this intermediate node should still forward this data. Considering the soft QoS [Ch99], to disobey QoS for transient time is acceptable. Therefore the delay violation test is performed only in the destination for each data packet. Another reason to check the delay only at the destination is that the destination can decide if the delay violation is tolerable. Each data packet will carry a variable named *RemainingDelay* representing the remaining delay and its value decreases along the route. When a data packet arrives at the destination, it exceeds the delay constraint if its *RemainingDelay* < 0. If several successive data packets exceed the delay limit, the destination will flood a route update packet (RUPD) to source in contention-based transmission mode. After receiving a RUPD, source will start a new route discovery process.

4. Specification of routing protocol and modifications in BBRP

In this chapter we describe the specification of our routing protocol and some modifications of the reservation protocol BBRP, which was specified in the project thesis [Xi07]. Firstly the structure of the routing protocol and new BBRP is presented. Then the signals between routing protocol and other layers are defined. After that, some data types and data structures as well as some frame types used in our specification are explained. Some routing information storages like *RoutingTable* are also described. In the end we explain the changes made in BBRP to adapt to the routing protocol and MacZ.

4.1. Architecture

As illustrated in Figure 4.1, the routing protocol lies between the higher layers (e.g. application) and reservation protocol BBRP. It is responsible for route discovery and route maintenance. All routing information is stored in routing protocol. The reservation protocol BBRP lies between routing protocol and MacZ. There are 2 components in BBRP, reservation manager and multiplexer. The black burst reservation component defined in the project thesis [Xi07] does not exist anymore due to some restructuring of MacZ. Its functionality is realized by the service layer of MacZ now. However the application can also communicate direct with BBRP and some actions like sending and receiving the application data can be performed without routing protocol (in the signal lists (App2RM) and (RM2App) in Figure 4.1). Some information obtained from MacZ (e.g. the current micro slot and the MAC address of this node) is useful for both BBRP and routing protocol. Therefore the multiplexer forwards the signals direct to these components (in the signal lists (Mux2RM) and (Mux2RT) in Figure 4.1).

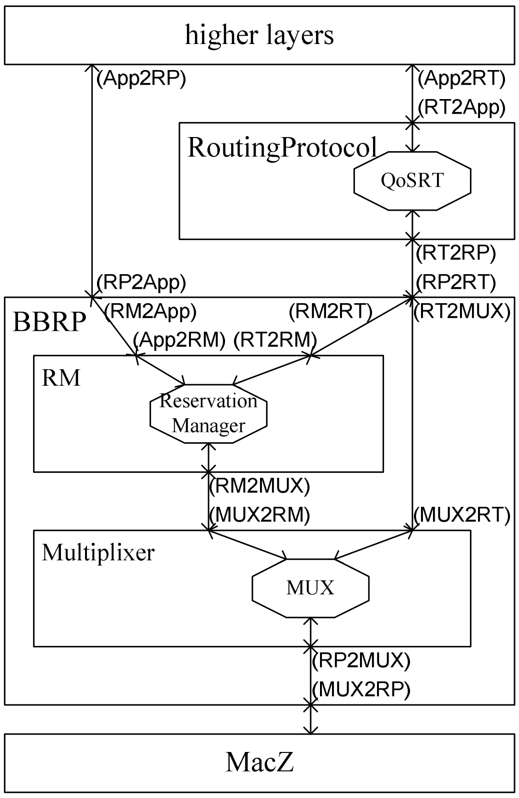


Figure 4.1 Structure of routing protocol and reservation protocol

4.2. Signals between routing protocol and other components

Now the signals between the routing protocol and other components are described. First we explain the signals between the routing protocol and the higher layers, which denote the signal lists (App2RT) and (RT2App) in Figure 4.1.

4.2.1. Signals between routing protocol and higher layers

Higher layers   Routing protocol (App2RT)

- Signal **ReserveStream_REQ** (

  bandwidthMin: Integer,

  delayMax: Integer,

  destination: AddressType)

  This signal is used to request the routing protocol for reserving a stream to the node *destination* with the QoS constraints: minimum bandwidth *bandwidthMin* and maximum delay *delayMax*.

Routing protocol   Higher layers (RT2App)

- Signal **ReserveStream_REJ**

  This signal replies the signal **ReserveStream_REQ** and rejects the request of reservation. The rejection can be triggered in 3 situations:

  - There are no free *streamID* to assign (The maximum number of *streamID* is 256).
  - There are no free micro slots to reserve.
  - The RREP packet does not come in time (within 2**RREQTTL*).

- Signal **ReserveStream_CNF** (*streamID*: Integer)

  This signal confirms the signal **ReserveStream_REQ** with the ID of reserved stream *streamID*. After receiving this signal, the higher layers (e.g. application) can start to send the application data.

Now we explain the signals between the routing protocol and the reservation protocol, which denotes the signal lists (RT2RP) and (RP2RT) in Figure 4.1.

4.2.2. Signals between routing protocol and reservation protocol

Routing protocol   BBRP (RT2RP)

- Signal **BuildStream_REQ** (

  bandwidthMin: Integer,

  exclusionSlotsTry: List_MicroSlot)

This signal is used to check if there are enough free micro slots locally to reserve. The parameter *bandwidthMin* denotes the bandwidth constraint in number of micro slots. The parameter *exclusionSlotsTry* contains the micro slots, which try not to be reserved. The selection of these micro slots is based on the reservation information of the neighboring nodes, which is stored in *NetworkStates*.

- Signal **SendRREQ** (*rreq*: Octet_StringFixed)

  This signal sends the encoded RREQ packet to BBRP and the packet will be sent with the priority-based method in the contention-based period.

- Signal **PrereserveSlots_REQ** (

  bandwidthMin: Integer,

  RREQid: Integer,

  slotsRsrvPreNode: List_MicroSlot,

  exclusionSlots: List_MicroSlot,

  exclusionSlotsTry: List_MicroSlot)

  This signal is used to ask the BBRP for prereserving some free micro slots. The request for prereservation happens in the intermediate nodes of a RREQ flooding after receiving a RREQ packet. The parameters *bandwidthMin* and *RREQid* denote the bandwidth constraint and the ID of the RREQ packet. The parameter *slotsRsrvPreNode* contains the micro slots reserved by the preceding node. The parameter *exclusionSlots* contains the micro slots to be excluded by selecting micro slots in BBRP, such as the slots reserved by pre-preceding node. The parameter *exclusionSlotsTry* contains the micro slots to be excluded regarding the neighborhood information. If there are not enough free slots to prereserve, the micro slots in *exclusionSlotsTry* can be prereserved.

- Signal **PrereserveSlots_RLS** (*RREQid*: Integer)

  This signal is used to inform BBRP to release the prereserved slots. The parameter *RREQid* is the ID of the RREQ packet and also the ID of the prereservation.

- Signal **PrereserveSlots_ACK** (*RREQid*: Integer)

  This signal is used to inform BBRP that the prereservation is accepted. The parameter *RREQid* is the ID of the RREQ packet and also the ID of the prereservation.

- Signal **ReleaseStream_REQ** (*streamID*: Integer)

  This signal is used to request BBRP to release the reservation. The parameter *streamID* is the ID of the reservation.

- Signal **SendRREP** (

  rrep: Octet_StringFixed,

  nextHop: AddressType)

This signal carries the encoded RREP packet, which will be sent by BBRP in the contention-based transmission mode. The parameter *rrep* is the encoded RREP packet and the parameter *nextHop* is the address of the next intermediate node to propagate the RREP packet.

- Signal **ReserveSlots_REQ** (*RREQid*: Integer)

  This signal is used to request BBRP to reserve the previously prereserved slots. The parameter *RREQid* is the ID of the previous RREQ packet and also the ID of the prereservation.

- Signal **ReleasePrereservation** (*RREQid*: Integer)

  This signal is used to request BBRP to release the previously prereserved slots. The parameter *RREQid* is the ID of the previous RREQ packet and also the ID of the prereservation.

- Signal **UpdateStream_REQ** (

  streamID: Integer,

  nextHop: AddressType)

  This signal is used to update the reservation information (e.g. *nextHop*) stored in BBRP. This signal is sent after receiving the RREP packet at the source node. The parameter *streamID* is the ID of the reservation and the parameter *nextHop* is the information to be updated.

- Signal **ChangeSlots_REQ** (

  RREQid: Integer,

  exclusionSlots: List_MicroSlot,

  slotsFrom: List_MicroSlot,

  slotsTo: List_MicroSlot)

  This signal is used to request BBRP to find a replacement slot for the unprereservable slot. The parameter *RREQid* is the ID of the previous RREQ packet and also the ID of the prereservation. The parameter *exclusionSlots* contains the slots to be excluded by the selection of free slots. The parameter *slotsFrom* and *slotsTo* defines the range of the selection.

- Signal **Routing_REP** (

  slotOut: Integer,

  nextHop: AddressType)

  This signal is used to reply the routing request signal **Routing_REP** (*slotIn*: Integer) from BBRP. The parameter *slotOut* denotes the outgoing slot for the incoming slot *slotIn*. The parameter *nextHop* is the address of the next intermediate node to propagate the data packet.

- Signal **Routing_REJ**

This signal rejects the routing request signal **Routing_REP** (*slotIn*: Integer) from BBRP. It indicates that there is no entry for *slotIn* in the routing table.

- Signal **ReleaseSlots_CNF** (*slotsOut*: List_MicroSlot)

  This signal replies the signal **ReleaseSlots_REQ** (*slotsIn*: List_MicroSlot) from BBRP. The parameter *slotsOut* is the outgoing slots for the incoming slot *slotsIn*.

- Signal **SendHello** (

  nodeAddress: AddressType,

  numFreeSlots: Integer,

  hopTTL: Integer)

  This signal is used to propagate the Hello message. This signal is encoded in BBRP into HELLO packet, which is sent in the contention-based transmission mode. The parameter *nodeAddress* denotes the address of the sender node. The parameter *numFreeSlots* is the number of free slots of sender. The parameter *hopTTL* is used to limit the range of the propagation of Hello message. The *hopTTL* is initialized with 2 and decrements 1 after travelling a node. When the *hopTTL* equals 0, the Hello message will not be propagated.

- Signal **NetworkStates_REQ** (

  requesterAddress: AddressType,

  replierAddress: AddressType,

  hopTTL: Integer)

  This signal is used to request the reservation information of a neighbor node within 2 hops. This signal is encoded in BBRP into NetworkStates_REQ packet, which is sent in the contention-based transmission mode. The parameter *requesterAddress* and *replierAddress* denote the addresses of the requester node and the replier node. The parameter *hopTTL* is used to limit the range of the propagation of the NetworkState_REQ message. The *hopTTL* is initialized with 2 and decrements 1 after travelling a node. When the *hopTTL* equals 0 or the NetworkState_REQ message arrived at the replier node, the message will not be propagated.

BBRP Routing protocol (RP2RT)

- Signal **InitializeMw** (*ownAddress*: AddressType)

  This signal is used to initialize the routing protocol with the address of local node *ownAddress*.

- Signal **MicroSlot_IND** (*slot*: Integer)

  This signal indicates the beginning of micro slot *slot*.

- Signal **BuildStream_CNF** (

  streamID: Integer,

reservedSlots: List_MicroSlot)

This signal replies the signal **BuildStream_REQ** with the assigned ID of reservation *streamID* and the reserved micro slots *reservedSlots*.

- Signal **BuildStream_REJ**

  This signal rejects the signal **BuildStream_REQ**. It can be caused when the number of reservations reaches the maximum limit or there are not enough free slots to reserve.

- Signal **ReceiveRREQ** (*rreq*: Octet_StringFixed)

  This signal indicates that a RREQ packet is received. The parameter *rreq* is the encoded RREQ packet.

- Signal **PrereserveSlots_CNF** (

  RREQid: Integer,

  prereservedSlots: List_MicroSlot)

  This signal is used to reply the signal **PrereserveSlots_REQ** with the assigned ID of prereservation *RREQid* and prereserved micro slots *prereservedSlots*.

- Signal **PrereserveSlots_REJ** (*RREQid*: Integer)

  This signal is used to reject the signal **PrereserveSlots_REQ** with the ID of prereservation *RREQid.*

- Signal **ReleaseStream_CNF** (*streamID*: Integer)

  This signal is used to reply the signal **ReleaseStream_REQ** when the reservation *streamID* is released successfully.

- Signal **ReceiveRREP** (*rrep*: Octet_StringFixed)

  This signal indicates that a RREP packet is received. The parameter *rrep* is the encoded RREP packet.

- Signal **UpdateStream_CNF** (*streamID*: Integer)

  This signal replies the signal **UpdateStream_REQ** with the ID of reservation *streamID.*

- Signal **ReserveSlots_CNF** (*RREQid*: Integer)

  This signal replies the signal **ReserveSlots_REQ** with the ID of prereservation *RREQid.*

- Signal **ReserveSlots_REJ** (

  RREQid: Integer,

  unreservableSlots: List_MicroSlot)

This signal rejects the signal **ReserveSlots_REQ** with the ID of prereservation *RREQid* and the prereserved micro slots *unreservableSlots*, which cannot be reserved.

- Signal **ChangeSlots_REJ** (*RREQid*: Integer)

  This signal rejects the signal **ChangeSlots_REQ** with the ID of prereservation *RREQid* because not enough micro slots can be found to replace the slots *unreservableSlots*.

- Signal **ChangeSlots_CNF** (

  RREQid: Integer,

  changedSlots: List_MicroSlot)

  This signal replies the signal **ChangeSlots _REQ** with the ID of prereservation *RREQid* and the reserved micro slots *changedSlots*, which replace the slots *unreservableSlots*.

- Signal **Routing_REQ** (*slotIn*: Integer)

  This signal requests the routing protocol for the outgoing slot *slotOut* and next intermediate node to propagate *nextHop* regarding the incoming slot *slotIn*.

- Signal **ReleaseSlots_REQ** (*slotsIn*: List_MicroSlot)

  This signal is used to request the routing protocol to remove the entries in the routing table, which have the incoming slots *slotsIn*.

- Signal **ReceiveHello** (

  nodeAddress: AddressType,

  numFreeSlots: Integer,

  hopTTL: Integer)

  This signal indicates that a Hello message is received. The parameter *nodeAddress* is the address of the sender of the Hello message. The parameter *numFreeSlots* is the number of free slots in node *nodeAddress*. The parameter *hopTTL* is used to limit the range of the propagation of Hello message. The *hopTTL* is initialized with 2 and decrements 1 after travelling a node. When the *hopTTL* equals 0, the Hello message will not be propagated.

- Signal **NetworkStates_CNF** (

  requesterAddress: AddressType,

  replierAddress: AddressType,

  freeSlots: List_MicroSlot,

  hopTTL: Integer)

This signal is used to reply the signal **NetworkStates_REQ**. This signal is encoded in BBRP into NetworkStates_CNF packet, which is sent in the contention-based transmission mode. The parameter *requesterAddress* and *replierAddress* denote the addresses of the requester node and the replier node. The parameter *freeSlots* contains all the free slots in replier node. The parameter *hopTTL* is used to limit the range of the propagation of the NetworkState_CNF message. The *hopTTL* is initialized with 2 and decrements 1 after travelling a node. When the *hopTTL* equals 0 or the NetworkState_CNF message arrived at the requester node, the message will not be propagated.

4.3. Data types and structures

In this thesis, different data types and data structures are defined, which are based on the predefined SDL data types and data structures. They are created to adapt to our routing protocol. Some of the following data types and structures are system-defined and some are user- defined.

- Octet

  A tuple of 8 bits.

- Octet_StringFixed

  An array of data type Octet with fixed maximum length.

- AddressType

  A 4-byte MAC address for every node.

- List_MicroSlot

  A list, which is used to store micro slots and has the maximum length MaxResSlotLength.

- Node with the attributes NodeAddress and SlotsToReserve

  A data structure, which represents an intermediate node in a route. The attribute NodeAddress is the address of the node and the attribute SlotsToReserve denotes the micro slots that should be (pre)reserved in this node.

- List_Node

  A list, which is used to store routes and has the maximum length MaxNumNodes.

- SlotOut_NextHop with the attributes SlotOut and NextHop

  A data structure, which stores the outgoing slot and the address of next propagation hop.

- Map_Routing

  A hashmap, which uses the incoming slots as key to store the SlotOut_NextHop values. It has the maximum length MaxResSlotLength.

- Delays_Deadline with the attributes Delays, Deadline, DeadlineMode

A data structure, which stores the information of RREQ packets. The attribute Delays denotes the estimated delays for a RREQ packet. The attribute Deadline denotes the deadline for a RREQ packet. The attribute DeadlineMode denotes the type of deadline, which can be 0, 1 or 2. The value 0 indicates that this deadline is set at the source node to wait for the return of the RREP packet. The value 1 indicates that the deadline is set at the intermediate node to wait for the return of the RREP packet. The value 2 indicates that the deadline is set at the destination node to wait for other RREQ packet with the same ID.

- Map_RREQTable

  A hashmap, which uses the ID of the RREQ packet as key to store the Delays_Deadline value. It has the maximum length MaxNoRREQs.

- Node_States with the attributes NoFreeSlots, Deadline and Distance

  A data structure, which stores the information of a neighboring node within 2 hops. The attributes NoFreeSlots denotes the number of free slots in this node. The attributes Deadline denotes the remaining time of the existence of this node. The attributes Distance denotes the distance between the local node and the neighboring node, which can be 1 or 2.

- Map_NetworkStates

  A hashmap, which uses the address of a node as key to store the Node_States value. It has the maximum length MaxNoNeighbors.

- Route_Delay with the attributes Route and Delay

  A data structure, which stores the routes and the estimated delay values of each route.

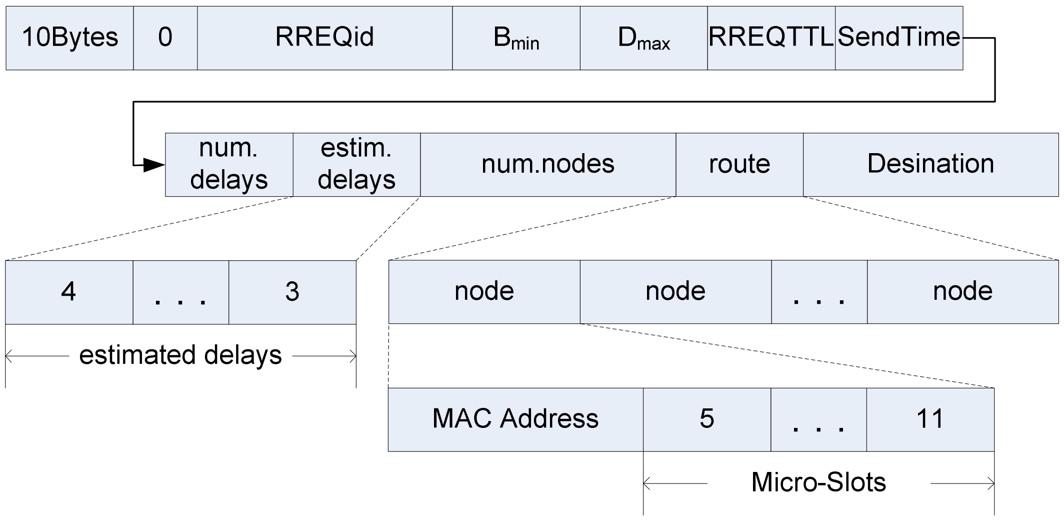
- Map_RouteStorage

  A hashmap, which uses the ID of the RREQ packet as key to store the Route_Delay value. It has the maximum length MaxNoRREQs.

4.4. Frame format

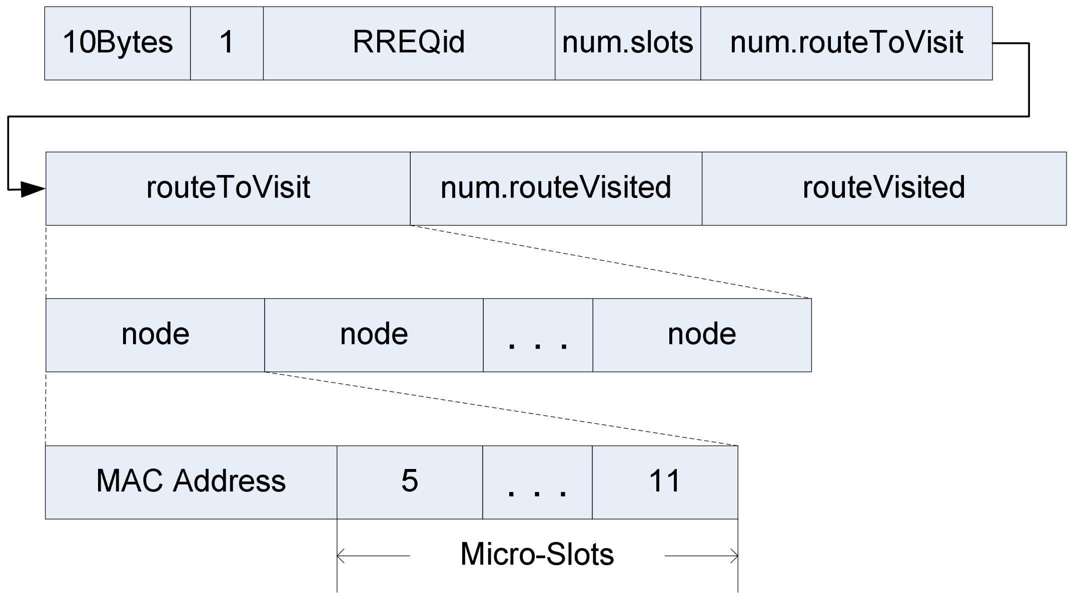
There are 5 different types of frames in our routing protocol. The first 10 bytes of each frame are reserved by MacZ and the maximum length of the entire frame, as defined in MacZ, is 121 bytes.

- RREQ



The RREQ frames are propagated in the route discovery phase to find a suitable route. It is sent in the contention-based transmission mode. Firstly it is the frame ID (1 byte) 0x00. Then follow the ID of the RREQ packet RREQid (4 bytes), the bandwidth constraint B_{min} (2 bytes), the delay constraint D_{max} (2 bytes), the remaining propagation time RREQTTL (2 bytes) and the number of sending micro slot SendTime (2 bytes). Then follow a list estim.delays with the estimated delays for each reserved micro slot (2 bytes for each delay) and the length of the list num.delays (2 bytes). Then follow a list of travelled nodes route and the length of the list num.nodes (2 bytes). For each node in route, the MAC address of this node (4 bytes) and the (pre)reserved micro slots (2 bytes each slot) are stored. In the end of a RREQ packet is the address of the destination node (4 bytes).

- RREP



The RREP frame is propagated unicast from the destination node to the source node to reserve the prereserved slots in the intermediate nodes and to reply the RREQ packet. It is sent in the contention-based transmission mode. Firstly it is the frame ID (1 byte) 0x01. Then follow the ID of the RREQ packet to reply RREQid

(4 bytes) and the number of reserved micro slots in each node num.slots (2 bytes). Then follow 2 lists of nodes routeToVisit and routeVisited as well as their lengths num.routeToVisit (2 bytes) and num.routeVisited (2 bytes). The list routeToVisit stores the nodes, which have not visited by this RREP packet. The list routeVisited stores the nodes, which are already visited by this RREP packet. For each node in routeToVisit and routeVisited, the MAC address of this node (4 bytes) and the (pre)reserved micro slots (2 bytes each slot) are stored. No micro slot is stored for the destination node.

- DATA

10Bytes	2	id	destination	SendTime

Remaining Delay	Prepre Slot	SucSlot	Application Data

The DATA frame is used to send the application data. It contains also some information for the routing maintenance. It is sent in the contention-free transmission mode. Firstly it is the frame ID (1 byte) 0x02. Then follow the ID of the application data packet id (1 byte) and the address of the destination node destination (4 bytes). Then follow the number of sending micro slot of this frame SendTime (2 bytes), the remaining delay constraint RemainingDelay (2 bytes), the slot reserved for this data packet in the pre-preceding node PrepreSlot (2 bytes) and the slot reserved for this data packet in the succeeding node SucSlot (2 bytes). In the end of DATA frame is the application data. The value of RemainingDelay decreases during the propagation of data packet. If it is smaller than 0 at the destination, there is a delay violation for this data packet.

- HELLO

10Bytes	3	NodeAddress	Num.Free Slots	Hop TTL

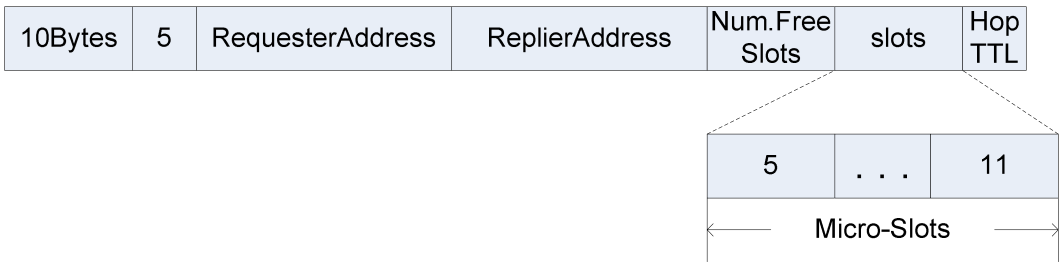
The HELLO frame is used to broadcast the local reservation information to the neighboring nodes and to show the existence of nodes. It is sent in the contention-based transmission mode. Firstly it is the frame ID (1 byte) 0x03. Then follow the address of the sender node NodeAddress (4 bytes), the number of free slots of the sender node Num.FreeSlots (2 bytes) and the number of the remaining propagation hops HopTTL (1 byte).

- NetworkStates_REQ

10Bytes	4	RequesterAddress	ReplierAddress	Hop TTL

The NetworkStates_REQ frame is used to request a neighboring node (replier) to send its reservation information back to the requester node. It is sent in the contention-based transmission mode. Firstly it is the frame ID (1 byte) 0x04. Then follow the address of the requester node RequesterAddress (4 bytes), the address of the replier node ReplierAddress (4 bytes) and the number of the remaining propagation hops HopTTL (1 byte).

- NetworkStates_CNF



The NetworkStates_CNF frame is used to reply NetworkStates_REQ frame. It contains the local reservation information of the replier. It is sent in the contention-based transmission mode. Firstly it is the frame ID (1 byte) 0x04. Then follow the address of the requester node RequesterAddress (4 bytes), the address of the replier node ReplierAddress (4 bytes). Then follow the list of free slots of the replier node slots (2 bytes each slot) and the number of the remaining propagation hops HopTTL (1 byte).

4.5. QoSRT

QoSRT is the main component to perform different routing mechanisms, e.g. route discovery and route maintenance. In order to perform those mechanisms, much information is needed. Therefore there are 4 hashmaps created in QoSRT to store different routing information. They are *RREQTable*, *NetworkStates*, *RouteStorage* and *RoutingTable*.

4.5.1. *RREQTable*

The *RREQTable* is defined with the data type Map_RREQTable. The capacity of *RREQTable* is predefined with the constant MaxNoRREQs. MaxNoRREQs denotes the maximum acceptable number of RREQs with different IDs. The key for each entry is the RREQid. Each entry of *RREQTable* contains the estimated delays and the deadline for the RREQid. A new entry is added into *RREQTable* after receiving a RREQ packet with a new RREQid. An entry is updated after receiving a RREQ packet with the same RREQid and lower estimated delays. The deadline of each entry decreases 1 in the beginning of every micro slot. An entry is removed after receiving a RREP packet with the same RREQid or after the deadline of this entry expires (equals 0).

After the deadline of an entry expires, different processes will be performed according the type of deadline, which is also stored in *RREQTable* as attribute DeadlineMode. When the DeadlineMode equals 0, which means that the deadline is set by the source node to wait for the RREP packet, the previously reserved slots for this RREQid must be released. When the DeadlineMode equals 1, which means that the deadline is set by an intermediate node to wait for the RREP packet, the previously prereserved slots for this RREQid must be released. When the DeadlineMode equals 2, which means that the deadline is set by the destination node to wait for the other RREQ packets with the same RREQid as this entry, the destination node chooses the route with the smallest estimated delays and sends a RREP back to source.

4.5.2. *NetworkStates*

The *NetworkStates* is defined with the data type Map_NetworkStates. The capacity of *NetworkStates* is predefined with the constant MaxNoNeighbors. MaxNoNeighbors denotes the maximum acceptable number of neighboring nodes within 2 hops. The key for each entry is the NodeAddress of the neighboring node. Each entry of *NetworkStates* represents a neighboring node along with its reservation information (e.g. number of free slots), position information (e.g. distance to the local node) and a deadline. A new entry is added into *NetworkStates* after receiving a Hello message with a new NodeAddress. The Hello message is periodically sent by each node. An entry is updated after receiving a Hello message with the same NodeAddress as this entry. The deadline of each entry decreases 1 in the beginning of every micro slot. An entry is removed after the deadline of this entry expires (equals 0), which means that the link to this neighbor is broken.

The neighborhood information stored in *NetworkStates* is useful in both route discovery phase and route maintenance phase. In the route discovery phase, if a neighboring node contains very few free slots, the local node will try to avoid reserving these free slots in order to propagate the RREQ packet further. In the route maintenance phase, if the link to a neighbor is broken, the routing protocol will check if there is any route built on this link. If so, the routing protocol will try to repair this route.

4.5.3. *RouteStorage*

The *RouteStorage* is defined with the data type Map_RouteStorage. The capacity of *RouteStorage* is predefined with the constant MaxNoRREQs. MaxNoRREQs denotes the maximum acceptable number of RREQ packets with different RREQid. The *RouteStorage* is used only in the destination node. The key for each entry is the RREQid. Each entry of *RouteStorage* contains the route and the estimated delay for this route (the maximum value of the estimated delays of slots). A new entry is added into *RouteStorage* after receiving a RREQ packet with a new RREQid. An entry is updated after receiving a RREQ packet with the same RREQid as this entry and a lower estimated route delay. An entry is removed after the deadline of this RREQid in *RREQTable* (DeadlineMode = 2) expires and a RREP with RREQid will be sent with the route of this entry.

4.5.4. *RoutingTable*

The *RoutingTable* is defined with the data type Map_Routing. The capacity of *RoutingTable* is predefined with the constant MaxResSlotLength. MaxResSlotLength denotes the maximum number of micro slots to reserve. The key for each entry is the incoming slot *slotIn*. Each entry of *RoutingTable* contains the outgoing slot *slotOut* and the next propagation node *nextHop*. A new entry is added into *RoutingTable* after receiving a RREP packet with the incoming and outgoing slots. An entry is updated after the route was repaired due to a route break or QoS violation. An entry is removed after the deadline of the incoming slot in BBRP expires and the incoming and outgoing slots will be released by BBRP.

The main functionality of *RoutingTable* is routing. The data packets received in the incoming slots will be sent in the corresponding outgoing slots. By that means the data packet can be transmitted over multiple hops. The *slotOut* of the destination node is -1.

4.6. Modification in BBRP to adapt to QoSRT and MacZ

There are several modifications in BBRP in order to match the routing protocol and MacZ. The modifications are made due to the following reasons:

- The BBRP specified in [Xi07] assumes that the route is available before the reservation in multi-hop scenario. It means that the route discovery and reservation are separately made. In our routing protocol, the route discovery and reservation are made simultaneously.
- In order to separate the functionality of MacZ and BBRP, some of the functionality of BBRP is integrated into the service layer of MacZ. It includes the propagation of black bursts, the addition or removal of a reservation and the addition or removal of micro slots into or from a reservation.
- A new internal state *prereseved* is added for each micro slot. Therefore a new information storage has to be created to store this reservation information.

4.6.1. New signals between BBRP and MacZ

Now the new signals between BBRP and MacZ are described. First we explain the signals sent from BBRP to MacZ.

BBRP MacZ

- Signal **delRes** (*streamID*: Octet)

  This signal is used to remove the reservation with ID *streamID*.

- Signal **resStream** (

  streamID: Octet,

  streamType: Octet)

  This signal is used to start a new reservation with ID *streamID*. The second parameter *streamType* is currently not in use and can be set as constant 1.

- Signal **resAddSlot** (

  slot: Integer,

  streamID: Octet,

  unused: Octet)

  This signal is used to add a slot *slot* into a reservation with ID *streamID*. The third parameter is currently not in use and can be set as constant 1. This reservation is permanent and can only be released explicitly by the signal **resRemSlot.**

- Signal **resAddSlotTemp** (

  slot: Integer,

  streamID: Octet,

interval: Octet)

This signal is used to add a slot *slot* into a reservation with ID *streamID*. This reservation is temporary contrary to the signal **resAddSlot**. It will be released implicitly after a number of macro slots defined by the third parameter *interval***.**

- Signal **resRemSlot** (*slot*: Integer)

  This signal is used to remove a slot *slot* from a reservation.

Now we explain the new signal sent from MacZ to BBRP.

MacZ BBRP

- Signal **slotBusy** (*slot*: Integer)

  This signal indicates that the slot *slot* is already reserved.

4.6.2. *PrereservationTable*

The new storage *PrereservationTable* is created to store the prereservation information. The *PrereservationTable* is defined with the data type Map_PrereservationTable. The capacity of *PrereservationTable* is predefined with the constant MaxResSlotLength. MaxResSlotLength denotes the maximum number of micro slots to reserve. The *PrereservationTable* is used only in the intermediate nodes. Each entry in *PrereservationTable* represents a prereservation. The key for each entry is the RREQid. Each entry of *PrereservationTable* contains a list of prereserved slots. A new entry is added into *PrereservationTable* after receiving a RREQ packet with a new RREQid. An entry is updated after receiving a RREQ packet with the same RREQid as this entry and a lower estimated route delay. An entry is removed after receiving a RREP packet with the same RREQid or after the deadline of this entry expires (DeadlineMode = 1).

5. Results and Evaluation

First we assume that the durations of one macro slot and one micro slot are 1000 millisecond and 1 millisecond, which means there are 1000 micro slots pro macro slot. The indexes of micro slot are numbered from 0 to 999. The medium is partitioned as illustrated in Figure 5.1. Since there is a synchronization phase in the beginning of each macro slot, we assume that the micro slots from 0 to 5 are used for the synchronization. The micro slot intervals [6,50), [100,150), …, [900,950) are contention-free periods based on reservation. The micro slot intervals [50,100), [150,200), …, [950,1000) are contention-based periods based on priorities.

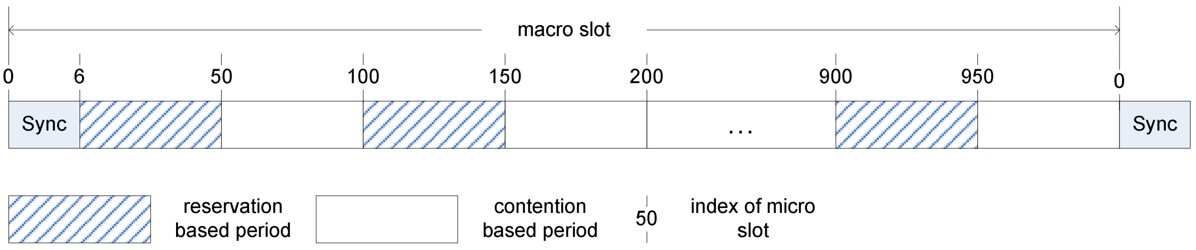


Figure 5.1 Medium partition

5.1. Scenario 1

We start with a simple scenario. There are 3 nodes in the network. The source node S tries to find a route to destination node D with the bandwidth constraint 30 Bytes/s and delay constraint 20 milliseconds. We assume that 2 micro slots are needed to fulfill the bandwidth requirement. The delay constraint is mapped into 20 micro slots according to the duration of one micro slot defined above. The addresses of node S, node I and node D are 100, 200 and 300. In the beginning all slots are free.

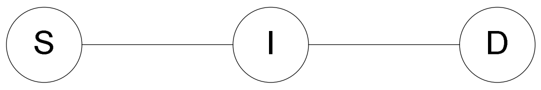


Figure 5.2 The network topology of scenario 1

In Figure 5.3 the simulation of scenario 1 is illustrated. There are 6 parts which are compared with the MSCs in the system design in chapter 3.

- Part 1 is compared with Figure 3.5.
- Part 2 is compared with Figure 3.6.
- Part 3 is compared with Figure 3.7.
- Part 4 is compared with Figure 3.9.
- Part 5 is compared with Figure 3.10.
- Part 6 is compared with Figure 3.11.

The behaviors of both sides in the 6 comparisons above are the same.

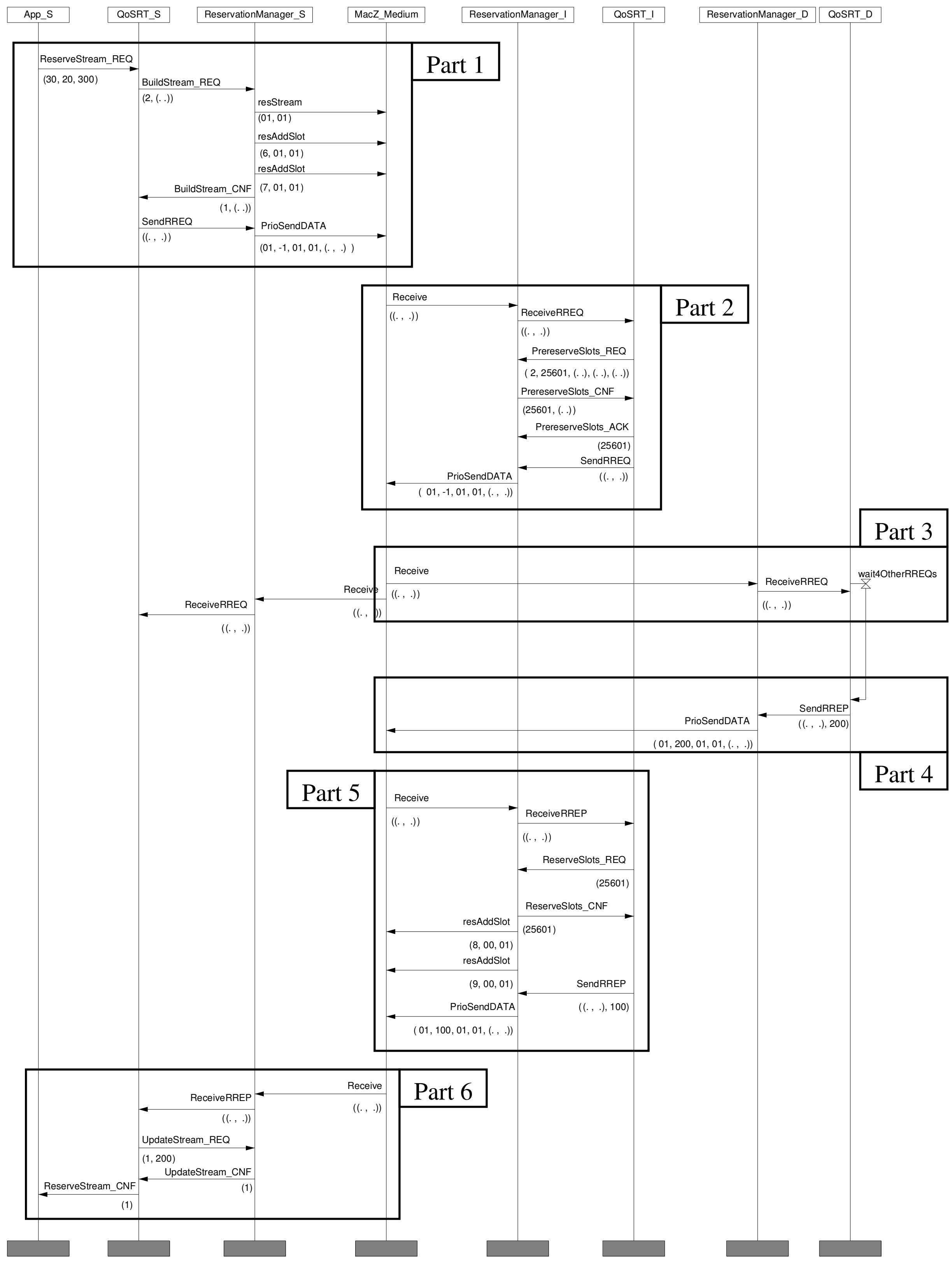


Figure 5.3 Simulation of scenario 1

5.2. Scenario 2

Now we have a more complicated scenario. There are 5 nodes in the network. The source node tries to find a route to destination node D with the bandwidth constraint 30 Bytes/s and delay constraint 30 milliseconds. We assume that 2 micro slots are needed to fulfill the bandwidth requirement. The delay constraint is mapped into 30 micro slots according to the duration of one micro slot defined above. The addresses of node S, node I1, node I2, node I3 and node D are 100, 200, 300, 400 and 500. In the beginning the slots 8, 9, 10 and 11 are reserved in node I1.

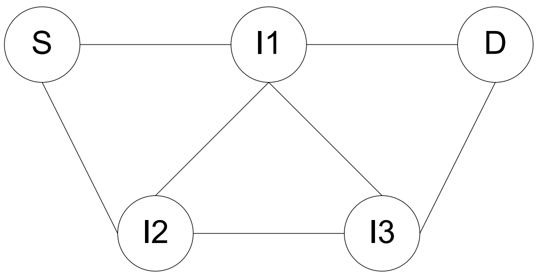


Figure 5.4 The network topology of scenario 2

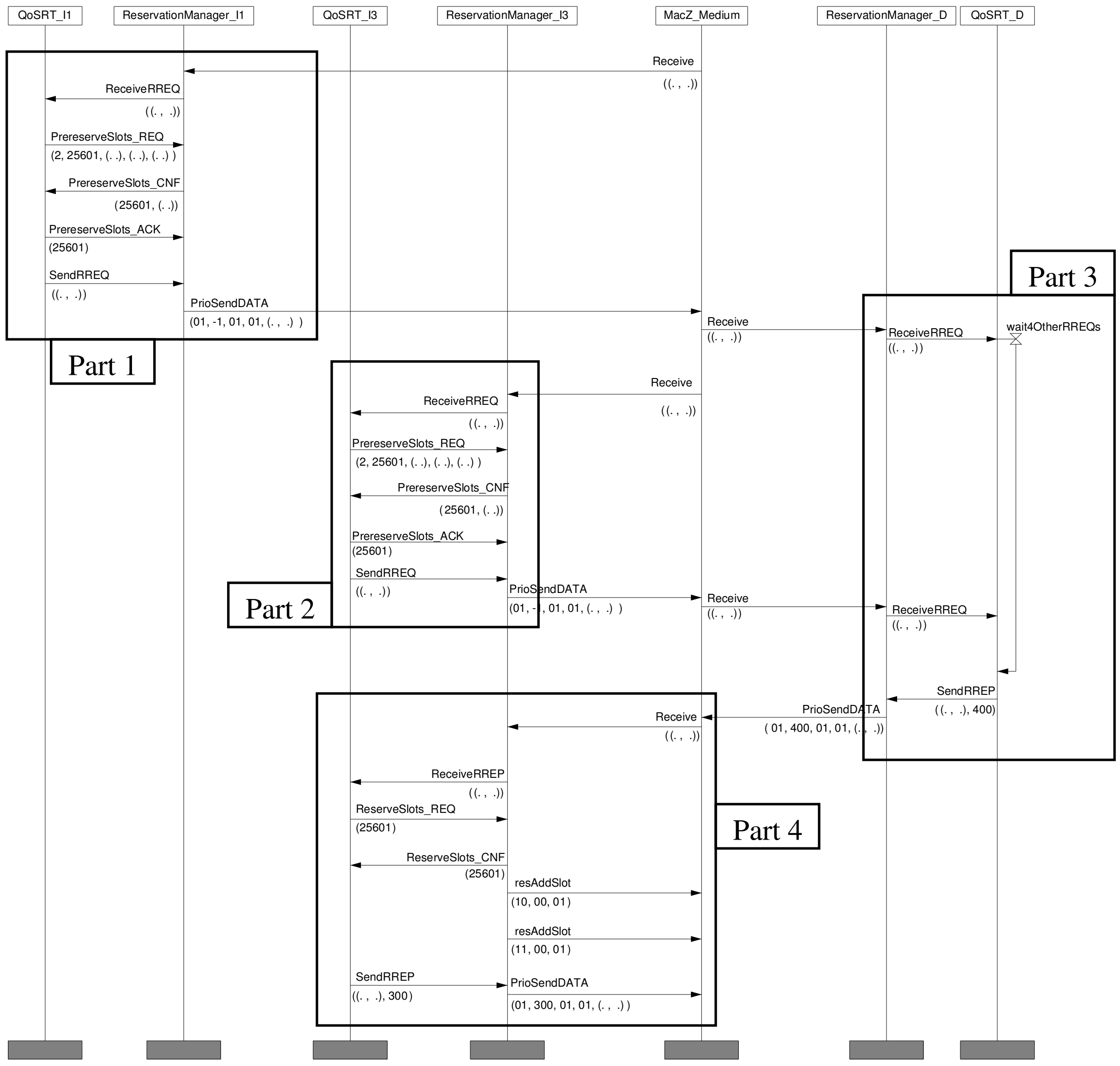


Figure 5.5 Simulation of scenario 2

In Figure 5.5 the simulation of scenario 2 is illustrated. In part 1 and part 2 of the figure, node I1 and node I3 propagate 2 RREQ packets to the destination node. After the timer wait4OtherRREQs expires in part 3 in the figure, the destination node sends a RREP packet to the node I3. Then the RREP packet is propagated from the node I3 to the node I2 in part4 in the figure. That means the best route is (S, I2, I3, D). The route (S, I1, D) is not chosen, because the slots 8 to 11 are reserved in I1. The node I1 can only choose slots 12, 13, while the node I3 chooses slots 10, 11. Therefore the route (S, I2, I3, D) has a lower estimated delay than the route (S, I1, D). The result of simulation is the same as the design.

6. Related Work

Now we introduce several different QoS routing protocols and discuss their working mechanisms and their strengths and weaknesses.

6.1.1. Ticket Based Probing (TBP)

Ticket Based Probing [Ch99] is a distributed ticket-based unicast routing protocol. Tickets are used to grant the intermediate node the right to search further and meanwhile limit flooding of the route request packets.

1. QoS constraints

   TBP tries to solve 2 routing problems: delay-constrained least-cost routing and bandwidth-constrained least-cost routing.

2. Route discovery

   There are two kinds of tickets during route discovery: yellow and green ticket. Yellow tickets are used to find a feasible route satisfying the bandwidth or delay constraints, while the green tickets try to find the most suitable route, which means the route with the lowest cost. When a source node wants to find a QoS route, it must first decide how many yellow and green tickets should be sent to its neighboring node. The more tickets the source node sends, the more probably a satisfying route can be found, but also the more time could discovery process take and more bandwidth is needed.

   The intermediate nodes forward more yellow tickets to their neighbors that better satisfy delay or bandwidth constraint and more green tickets to their neighbors that have lower cost links. If one of constraints is not satisfiable, this intermediate node sets the ticket as invalid and sends it to destination node. That means: # incoming tickets = # outgoing tickets.

   Figure 6.1 shows an example of tickets propagation in TBP. The source node S initiates 3 tickets (2 yellow and 1 green). These tickets are divided into 2 probes. The probe, which is sent to intermediate node I, contains 1 yellow and 1 green ticket. This probe is split into 2 probes in node I, with both probes having one ticket. All the tickets reach the destination node D in the end.

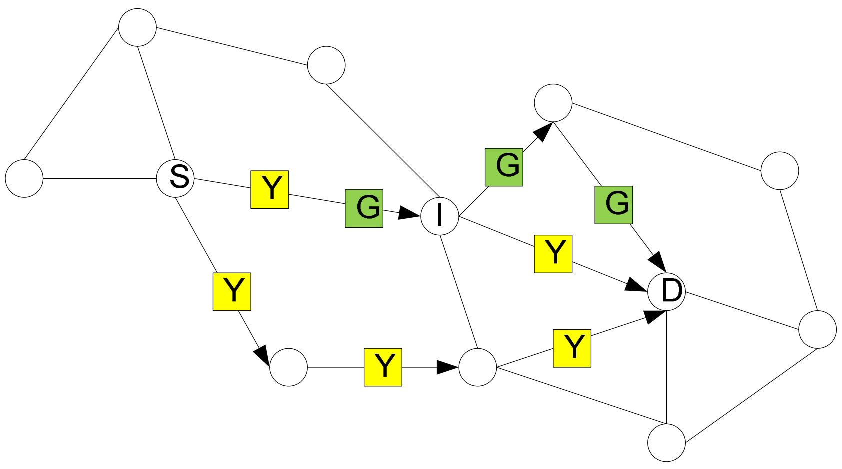


Figure 6.1: Tickets spread from source to destination in TBP

3. Route selection

   When multiple valid tickets arrive at destination node, the one with the lowest cost will be selected as primary route and the others are secondary routes.

4. Resource reservation

   After the primary route is selected, a confirmation will be sent back to the source along this primary route. Resource reservation is in the meantime done hop-by-hop. If resources cannot be reserved due to the change of network state, the confirmation will turn back to the destination and the destination will choose a secondary route as the primary route. Unused reserved resources are released with timeouts.

5. Route maintenance

   TBP supports several route maintenance mechanisms, such as broken route detection, rerouting by source node, path redundancy and path repairing at the breaking point.

   A broken route is detected using neighbor discovering protocol. When a node detects a broken route, it sends a path-breaking message to the source node. Then the source node reroutes the connection to another feasible path. The resources reserved by the original route will be released in both implicit (with timeout) and explicit (with resource releasing message) ways. Another approach to solve the route break problem is to use redundant route. As stated in route selection section, more than one route candidates can be found. Then there are three possible redundancy schemes to use: the first-, second- and third-level redundancy schemes. In first-level redundancy, multiple routes will be used for one connection; data packets will be sent in each route. In second-level redundancy, multiple routes will be established for one connection; data packet will be sent in the primary route; the resources in secondary routes will be reserved but not used. In third-level redundancy, it is similar to the second-level except that the resource in secondary routes will not be reserved. After the primary route is broken, current resource availability will be check in the secondary routes. If some of the secondary routes are still usable, one is selected as the new primary path. Otherwise rerouting will be activated. After finding a route break, route repairing can be processed instead of global rerouting. Route repairing shifts the data traffic to a neighbor node, where the QoS constraints can still be met.

TBP uses number of tickets to control the route discovery phase and avoids flooding the network. The simulation presented in [Ch99] shows that TBP can work with imprecise state information. But still in order to update the state information for every node, a proactive protocol like DSDV [Pe94] must be deployed. In each node a routing table storing all these state information will be periodically updated. This increases the communication and storage overhead and makes TBP not so scalable with network size.

6.1.2. Quality of Service-aware Source initiated Ad-hoc Routing (QuaSAR)

Medidi and Vik define in [Me04] a distributed unicast routing protocol named QuaSAR.

1. QoS constraints

   QuaSAR provides the application layer different QoS classes constructed with the following QoS metrics: bandwidth, latency, signal strength and battery power.

2. Route discovery

   The route discovery phase in QuaSAR is divided into 2 sub-phases, QoS Route Request and QoS Route Reply.

In Route Request phase, a QoS route request packet (QRREQ) is flooded into the network. In order to avoid unnecessary broadcasting, every intermediate node that has already propagated a QRREQ will only rebroadcast the second QRREQ if its QoS is better and the length of its route is not bigger than 2 times the predefined minimum route length. The classification of QoS will be defined with a QoS metric precedence rule, which is by default defined as follows: battery power > signal strength > bandwidth > latency.

In Route Reply phase, destination node will first wait for a threshold of arriving QRREQs and reply the one having the best QoS metric with a QoS route reply packet (QRREP). After that, any QRREQ with better QoS metric will also be replied.

3. Resource reservation

   The source reservation is done in the Route Request phase and QoS will not be updated during the Reply phase, because QoS doesn't change significantly during this time and the QoS update consumes extra battery power and costs extra wait time.

4. Route selection

   QuaSAR collects route statistics in route discovery phase and uses these statistics in the Route Reply phase to find an optimal route according to these metrics: available bandwidth, latency, signal strength and battery power. The route choosing algorithm uses QoS metric precedence to choose between routes and the ranking of the metrics can be changed by the applications depending on their different needs.

5. Route maintenance

   QuaSAR supports both reactive and proactive route maintenance mechanisms. The reactive part is similar to the one in AODV [Pe03] or DSR [Jo07]. The proactive part is based on preemptive detection of battery power and signal strength. After some critical incidents like signal strength weakening, battery power depletion or memory shortage are discovered, QuaSAR sends a Route Change Request packet (RCR) back to source informing about the cause of the problem.

QuaSAR considers battery power and signal strength during the route discovery and route maintenance phase. Therefore unstable nodes are avoided to be used and route breaks can be earlier prepared for. But in order to achieve this, a proactive process has to be used.

6.1.3. On-Demand Delay-Constraint Unicast Routing Protocol (ODRP)

Zhang and Mouftah propose in [Zh05] a delay-constrained unicast routing protocol named ODRP.

1. QoS constraints

   In ODRP end-to-end delay between source and destination node is considered as the only QoS constraint.

2. Route discovery

   ODRP deploys first a proactive distance vector algorithm to establish and maintain routing tables, storing the shortest path from local node to all other nodes in the network. When a delay-constraint route is required, the source node sends a probe directly to the destination along the shortest path. After receiving the probe, the destination node returns an ACK packet back to the source if this path satisfies the delay constraint.

If the probe doesn't meet the constraint, the destination initiates a reactive, directed flooding of RREQ Packets. Intermediate node will forward RREQ packets if the number of hops travelled is below a predefined limit and the path delay is below the delay constraint value. If the first RREQ reaches the source and satisfies the delay constraint, the source node can accept this route or wait for a while to collect more candidates.

3. Resource reservation

   ODRP assumes that a media access protocol is deployed to resolve media contention and support resource reservation at the MAC layer. In the proactive discovery phase, the resources are reserved during the reply phase with ACK. In the reactive discovery phase, the resource reservations along the route are done by the first data packet sent by the source.

4. Route selection

   If the route suggested by the proactive phase is feasible, it will be used. Otherwise the source node chooses the route with the shortest delay from those, which are proposed in the reactive phase.

5. Route maintenance

   Upon detecting a route break, the upstream node of the break point sends a Route Error packet (RERR) up to the source node; the downstream node of the break point sends a Route Release packet down to the destination. After receiving the RERR packet, the source node will reinitiate route discovery process.

ODRP is based mainly on the proactive process to find the qualified route. The reactive process will only be deployed when the proactive process failed, which means ODRP has all the common drawbacks of proactive routings like massive communication and storage overhead.

6.1.4. Ad hoc QoS On-demand Routing (AQOR)

AQOR [Xu03] is a hybrid distributed unicast QoS routing protocol, which is based on a contention medium.

1. QoS constraints

   AQOR finds routes that satisfy the bandwidth and end-to-end delay constraint.

2. Route discovery

   AQOR use a proactive process to maintain the neighborhood information. Each node sends out "Hello" packet once per second to its direct neighbors, in order to announce its existence and exchange traffic information like bandwidth. Without receiving any packets from this neighbor for T_{lost} period indicates that the link to a neighbor is down.

   In AQOR, the route discovery is on-demand with the propagation of route request and route reply packets between source and destination. The source broadcasts a route request packet with the minimum bandwidth constraint B_{min} and the maximum end-to-end delay constraint T_{max}, when the destination is not in the source's neighborhood list. Upon receiving a route request packet, the node has to decide whether this request could be accepted with a bandwidth admission decision process. The currently available bandwidth can be estimated on the fly with the aggregated traffic in the channel and the raw data rate of the node according to the algorithms presented by the authors.

If the request is acceptable, this node turns into *explored* state and the request will be rebroadcasted. The state *explored* remains $2T_{max}$; all later reply packets will be ignored. To prevent unnecessary propagation, each request packet has a maximum propagation time.

The destination replies every incoming request packets. During the reply phase, the intermediate node checks the bandwidth again and changes its state from e*xplored* to *registered*. The state *registered* lasts for $2T_{max}$.

To prevent possible loops during route exploration, AQOR uses route sequence number for each flow. Sequence number will be increased successively for each route control packet. The route control packet can only be consumed by a node with lower sequence number (except RREQ in destination). Upon consuming a packet, sequence number of this node will be replaced with the one of the packet.

3. Resource reservation

   AQOR uses the temporary reservation. That means in the route discovery phase, every state of node remains only $2T_{max}$. In the data transfer phase, the reservation is active for period $T_{interval}$. Therefore no explicit reservation release is needed. For application with long silent suspension session in the flow, light-weighted dummy data packets should be sent in order to keep the reservation valid.

4. Route selection

   In AQOR, the route is chosen in source node after the route discovery phase. Source node waits maximum $2T_{max}$ for the reply packet. The first arrived one will be accepted. If no replies come in time back to source, try the route discovery later again or turn down the flow.

5. Route maintenance

   The delay violation detection is destination oriented, which means, if the destination receives n consecutive data packets whose delays exceed T_{max}, the QoS recovery will be started. The route recovery will also be triggered, if the destination fails to receive data packets of a reserved flow before its reservation timeout ($T_{interval}$).

   The basic neighbor lost detection (Hello message) is also used to spot network partitions or route failures. When a node detects that the downlink node on a reserved route is lost, it sends a route error packet, with its current sequence number, to the corresponding uplink node along the route. The reserved bandwidth of the flow will be released at the nodes in this route.

   After the QoS violations are detected, the destination will broadcast an unsolicited route reply packet, also called route update, back to the source. Upon receiving the first in-time route update packet, the source switches the flow concerned to the reverse route on which the update arrives.

Since all the reservations in AQOR are temporary, the release of connections is done automatically and no additional mechanism is needed. To further reduce the control overhead caused by flooding of route request packets, AQOR can work with some location aided routing protocols.

7. Conclusions and Future Work

In this thesis a QoS routing protocol was developed, which is based on MacZ and BBRP. It is a hybrid distributed unicast routing protocol. This routing protocol supports two QoS constraints simultaneously, bandwidth constraint and delay constraint. The route discovery phase includes two sub-phases, route request and route reply phase. During the route request phase, the RREQ packets are flooded in the network to find a suitable route to the destination node. During the flooding of the RREQ packets, the prereservations are made by BBRP in the intermediate nodes. In order to make the flooding more efficient, several mechanisms are introduced to limit the unnecessary flooding and to enlarge the necessary flooding. These mechanisms include the integration of propagation time limit *RREQTTL*, the QoS comparison mechanisms and the thoughtful selection of the micro slots to reserve. After a suitable route is found at the destination node, a RREP packet is sent unicast back to the source node along the route. The RREP packet reserves the previously prereserved slots in each intermediate node. After receiving the RREP packet, the source node can start to send the application data. Some route maintenance mechanisms are deployed in this routing protocol including QoS violation detection and QoS route recovery. This routing protocol includes also network maintenance mechanism, which is useful in route discovery phase and route maintenance phase.

The specification of this routing protocol was built in SDL. Various data types, data structures and frame types were defined. The interface between routing protocol and other components were also explained. The functionality of routing protocol was specified with the utilization of these data types, data structures and signals. Some modifications were made in BBRP in order to adapt to the routing protocol and MacZ.

The simulation was performed to test the correctness of the specification. The scenarios were tested with the help of the simulator in TAU [TEL]. A MSC diagram was generated for each scenario. After comparing the MSCs between the simulation and the design, the mistakes made in the specification were found and corrected.

The routing protocol needs still to be improved in the future. There are several possible optimizations. In the route discovery phase, more mechanisms can be integrated to adjust the flooding of RREQ packets. More neighborhood information can be stored in order to prevent the unnecessary RREQ propagation earlier. Some mechanisms such as signal strength detection and node positioning detection can also be integrated in the route maintenance phase to detect the route break earlier.

CD

This CD contains the routing protocol specified in SDL with Telelogic Tau.

Bibliography

[AmI] Ambient Intelligence homepage of TU Kaiserslautern.
http://www.eit.uni-kl.de/AmI/

[Bh94] V. Bharghavan, "MACAW: A Media Access Protocol for Wireless LAN's", *Proceedings of the conference on Communications architectures, protocols and applications*, 1994

[BIM99] R.O. Baldwin, N.J. Davis IV and S.F. Midkiff, "A real-time Medium Access Control protocol for ad hoc wireless local area networks", *Mobile Comput. Commu. Rev.3*, Page 20-27 ,1999

[Ch99] S. Chen and K. Nahrstedt, "Distributed Quality-of-Service Routing in Ad Hoc Networks", *IEEE Journal on Selected Areas in Communications*, 17(8), August 1999

[DC99] D.J. Deng and R.S. Chang, "A priority scheme for IEEE 802.11 DFC access method", *IEICE Trans. Commun. 1E82-B*, Page 96-102, 1999

[Ha00] J. Haartsen and S. Mattisson, "BLUETOOTH—A New Low-Power Radio Interface Providing Short-Range Connectivity", *Proceedings of the IEEE Special Issue on Low-Power RF Systems*, 2000

[HIS07] P. Becker, R. Gotzhein and T. Kuhn, "MacZ – A Quality-of-Service MAC Layer for Ad hoc Networks", *Proceedings of the 7th International Conference on Hybrid Intelligent Systems* (HIS 2007), 2007

[ITU99] ITU-T RECOMMENDATION Z.100 (11/99): Specification and Description Language (SDL). International Telecommunication Union (ITU), 1999

[Jo07] D. Johnson, Y. Hu and D. Maltz, http://tools.ietf.org/html/rfc4728 DSR Specification 2007

[Me04] S.R. Medidi and K.H. Vik, "Quality of service-aware source-initiated ad-hoc routing", *Sensor and Ad Hoc Communications and Networks,* 2004

[Mo07] T. Kuhn and J.I. de Irigon, "An Experimental Evaluation of Black Burst Transmissions", *MobiWAC 2007*, Oct. 2007

[MSC] Message Sequence Charts, http://www.sdl-forum.org/MSC/

[Mu04] C. S. R. Murthy and B. S. Manoj, *Ad Hoc Wireless Networks: Architectures and Protocols*. Prentice Hall, 2004

[NSG] Networked Systems Group, http://vs.cs.uni-kl.de/

[Pe94] C. E. Perkins and P. Bhagwat, "Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers", *ACM SIGCOMM'94, Conference on Communications Architectures, Protocols and Applications*, Page 234-244, 1994

[Pe03] C. E. Perkins, E. M. Belding-Royer and S. R. Das, "Ad hoc On-Demand Distance Vector (AODV) Routing", *Mobile Ad Hoc Networking Working Group, IETF*, 2003

[Ra02] T. Rappaport, *Wireless Communications: Principles and Practice*. Prentice Hall, 2002, ch.10

[TEL] Telelogic Tau SDL Suite: http://www.telelogic.com/products/tau/sdl/index.cfm

[Wl97] IEEE Computer Society LAN MAN Standards Committee, “Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications”, IEEE std. 802.11-1997, 1997

[Wa96] Z. Wang and J. Crowcroft, “Quality-of-Service Routing for Supporting Multimedia Applications”, *IEEE Journal of Selected Areas in Communications*, 14(7), Page 1228-1234, 1996

[Wm04] *IEEE Standard for Local and Metropolitan Area Networks*, IEEE Computer Society and the IEEE Microwave Theory and Techniques Society Std. 802.16, 2004

[Xi07] J. Xiao, “Entwurf und Spezifikation eines Reservierungsprotokolls basierend auf MacZ”, project thesis, 2007

[Xu03] Q. Xue and A. Ganz, “Ad hoc QoS on-demand routing (AQOR) in mobile ad hoc networks”, *Journal of Parallel and Distributed Computing*, 63(2), Page 154-165, 2003

[Zh05] B. Zhang and H. T. Mouftah, “QoS routing for wireless ad hoc networks: problems, algorithms and protocols”, *IEEE Commun. Mag*, Oct. 2005.