FFI-rapport 2009/01765

Enhancements to the Narrowband Waveform (NBWF) network simulator

Bjørnar Libæk, Tore J. Berg and Jan Erik Voldhaug

Forsvarets forskningsinstitutt/Norwegian Defence Research Establishment (FFI)

10 June 2010

FFI-rapport 2009/01765

1088

P: ISBN 978-82-464-1780-6
E: ISBN 978-82-464-1781-3

Keywords

Modellering og simulering

Mobile ad-hoc nettverk

Taktisk kommunikasjon

Medium aksesskontroll

Trådløs kommunikasjon

Approved by

Torunn Øvreås	Project Manager
Eli Winjum	Forskningssjef
Vidar S. Andersen	Director

English summary

NATO (SC/6-AHWG/2) is currently in the effort of standardising the Narrowband Waveform (NBWF) for increased inter-operability among NATO coalition forces at the tactical level. This implies agreeing on communication protocols in all layers of the OSI stack. The requirements state that the targeted networks should be capable of serving simultaneous voice and data utilising a 25 kHz bandwidth in the UHF and VHF bands.

As a member of the working group, FFI is working on a proposal for link layer protocols for the NBWF, designed to operate above the waveform's physical layer proposed by CRC Canada. To assist in the process of developing the protocols, FFI has designed a discrete event network simulator especially for this purpose. The main objective of this document is to describe this simulator. The link layer is first described by means of a reference model, before the details about the simulator implementation are presented. In addition to the link layer, this includes the modelling of the physical layer, the voice traffic generators and the statistical output capabilities of the simulator.

Sammendrag

I NATO (SC/6-AHWG/2) arbeides det for tiden med å standardisere en smalbånds bølgeform for å oppnå økt interoperabilitet mellom nasjoner i internasjonale operasjoner, på taktisk og stridsteknisk nivå. Dette innebærer enighet om kommunikasjonsprotokoller i alle lag av OSI-modellen. I henhold til kravene til den nye bølgeformen, skal det være støtte for samtidig tale og data over en radiokanal med 25kHz båndbredde i UHF og VHF båndene.

Som medlem av arbeidsgruppa har FFI tatt på seg arbeidet med å utarbeide forslag til protokoller på linklaget. Protokollene skal fungere sammen med det fysiske laget foreslått av CRC Canada. For å støtte dette arbeidet har FFI utviklet en hendelsesdrevet nettverkssimulator. Hovedmålet med dette dokumentet er å beskrive denne simulatoren. En referansemodell presenteres, og linklaget beskrives ved hjelp av denne. I tillegg gis det detaljert informasjon om hvordan de ulike modulene er implementert. Dette innebærer linklags protokoller, fysisk lag og talegeneratorer på applikasjonslaget. Det gis også en oversikt over hvilke muligheter simulatoren har for å produsere statistikk.

Contents

1 Introduction

1.1 Background

The Narrowband Waveform (NBWF) is currently under NATO standardization by the UHF/VHF ad-hoc working group (AHWG/2 – SC/6). As a member of the group, FFI’s main dedication is to propose link layer protocols for the new waveform. This includes the definition of a set of services that meets the stated requirements, as well as solutions on how these can be realized in a multi-hop tactical radio network. The link layer is designed to operate optimally along with the waveform’s physical layer provided by CRC Canada [1].

FFI has developed a discrete event network simulator in order to study the behaviour and assess the performance of the proposed protocols and mechanisms. The main objective of this document is to describe this simulator. This includes both the underlying simulator design, as well as the implementation of the proposed link layer protocols.

This document may be seen as a sequel of [2], which described the initial design of the simulator. The evolved design is to a large degree based on that document. However, significant changes have been made to the link layer specification since then, forcing the simulator design to adapt.

The proposed link layer solutions are documented in [3], and a high-level technical overview of the link layer design is given in [4]. However, because the protocol specification is subject to continuous revising, and due to simplifications in the simulator model, the described link layer in this document to some extent deviates from the actual specification. For this reason, this document starts with a brief overview of the assumed link layer specification.

1.2 Organization of this document

Before turning the focus to the simulator implementation, section 2 gives an overview of the link layer reference model and the link layer specification assumed in the rest of the document. In section 3, the simulator software and overall design is presented, describing the different entities involved. Further, section 4 explains how the physical layer is modelled and implemented, while the link layer implementation is described in section 5. The last two sections describe the voice traffic generators and the statistical output parameters, respectively.

2 Link layer specification

This section gives an overview of the link layer specification, which is detailed in [3]. The NBWF reference model is presented, and the protocols are described. Section 5 gives a more detailed description of the simulator implementation.

The proposed MAC protocol is TDMA (Time Division Multiple Access) based and the nodes use dynamic reservations to allocate time slots on-demand. The primary focus is to ensure good quality for the MV (multicast voice) application, while also being able to serve IP traffic from the network layer. The protocol is designed to operate in multi-hop environments. A key feature is the ability to relay MV beyond the radio range of the source node using relaying at layer 2. The current version of the specification only supports dedicated voice relays, while a future goal is *automatic voice relaying*, where any node can take the role as a voice relay. The TDMA frame structure only allows two simultaneous voice relays. Note that relaying of IP packets (forwarding) are handled at layer 3 in a store-and-forward manner, thus the limitation on number of hops does not apply here.

2.1 TDMA frame structure

A TDMA frame consists of N time slots, numbered from 0 through N-1. Each slot S_i , $0 \leq i < N$, occurs at repetitive intervals equal to the TDMA frame length. The TDMA frame structure (number and length of time slots) is a result of optimising for 2.4 kbps MELPe (enhanced Mixed Excitation Linear Prediction) over the slowest mode provided by the NBWF PHY specification. As this specification is subject to frequent revisions, it has been decided that the version of the NBWF simulator described in this document is the C2 mode of the 2nd edition of the PHY specification [5]. This mode provides a 16 kbps data rate using CPM (continuous phase modulation). The simulator will be upgraded to use the N1 mode (20kbps) according to [1] in the near future.

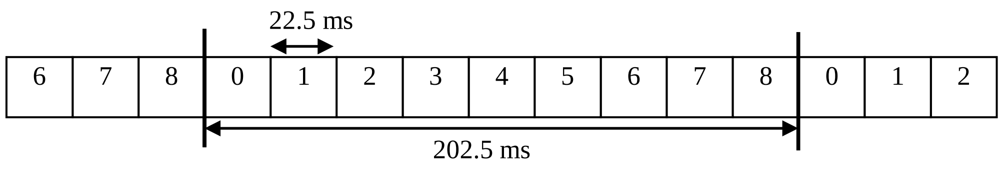


Figure 2.1 TDMA frame structure

Figure 2.1 shows the selected frame structure, consisting of 9 time slots, each of 22.5ms length, forming a 202.5ms frame. Refer to [3] for details about the choice of these parameters.
The link layer specification also describes a super frame structure, where a number of TDMA frames form a super frame. This provides the possibility to allocate a fixed capacity to each node in the network, using time slots reserved for super frame in a round-robin fashion. For the remainder of this document, it is assumed that the two last time slots are reserved for the super frame scheme.

2.1.1 Slot types

The TDMA frame structure also defines a set of slot types as illustrated in Figure 2.2. Different rules apply to different slots: First of all, in order to prevent collisions between MV signalling messages and data traffic, a number of MVS (Multicast Voice Signalling) time slots are exclusively available for MV signalling. This number depends on how many simultaneous MV relays are allowed. To be able to serve the actual voice transmissions, twice as many time slots are needed. Because MV is favoured over data, an additional slot type is introduced, named DU (Dual Use). These slots follow the MVS slots, and are available for both MV and data transmission. However, if a node has reserved DU slots for data transmission, and another node initiates a MV connection, the first node must stop using the DU slots and continue its transmissions in the GU (General Use) slots only. The figure below illustrates the TDMA frame layout and the different slot types in the cases of none, one and two relays, respectively. The GU and DU slots are available for data traffic using contention access, while the SF (Super Frame) slots are used by the super frame mechanism.

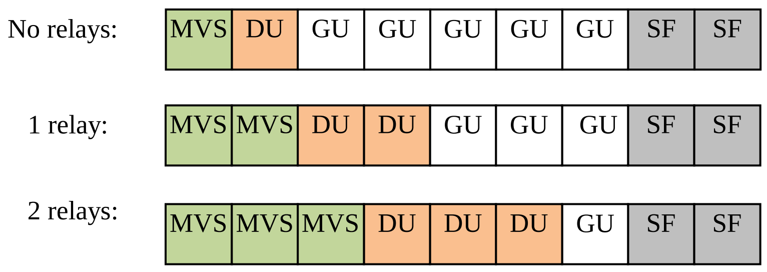

No relays:
MVS DU GU GU GU GU GU SF SF
1 relay:
MVS MVS DU DU GU GU GU SF SF
2 relays:
MVS MVS MVS DU DU DU GU SF SF

Figure 2.2 Slot types

2.2 Dynamic TDMA

Primarily, the nodes have to reserve time slots on-demand using the *reservation protocol*. There is also the possibility to transmit in free (non-reserved) time slots using a contention scheme. The latter is primarily intended for short data packets, while the reservation mechanism is for voice, transparent data and longer data transmissions. A third option is to transmit in the statically reserved super frame slots as described in the previous section. Due to PHY overhead (8-9 ms in each transmission), it is more efficient to let one transmission span several time slots. This is termed *slot merging.* A merged slot is simply a collection of consecutive time slots used for one PHY transmission. All of the three access methods (dynamic reservation, contention and super frame allocation) may use merged slots. In principle, a merged slot may cross TDMA frame borders, but the current reservation scheme does not allow this.

2.2.1 Reservations

A node may reserve one or several consecutive (merged) time slots dynamically for one-hop transmissions. During the lifetime of such a *reservation*, the node that made the reservation, the *reservation originator* (RO), is allowed to transmit in the reserved slot range. It is important that the originator’s transmissions do not collide with other transmissions at the destination nodes, so no other nodes within the reservation area can send in the given slot range during the reservation lifetime. In this respect, the destination nodes, or *recipients*, of the originator's transmission, may either be a single unicast destination, several members of a multicast group or a relay node, all of which must be within the originator's transmission range (i.e. one-hop neighbours). The aforementioned reservation area in which no other nodes are allowed to send is termed the *reservation domain*. More specifically, we define the reservation domain to include all the one-hop neighbours of the originator, as well as all two-hop neighbours that are within the transmission range of any of the recipients (as defined above). Figure 2.3 shows an example topology and identifies the reservation domain for that specific scenario.

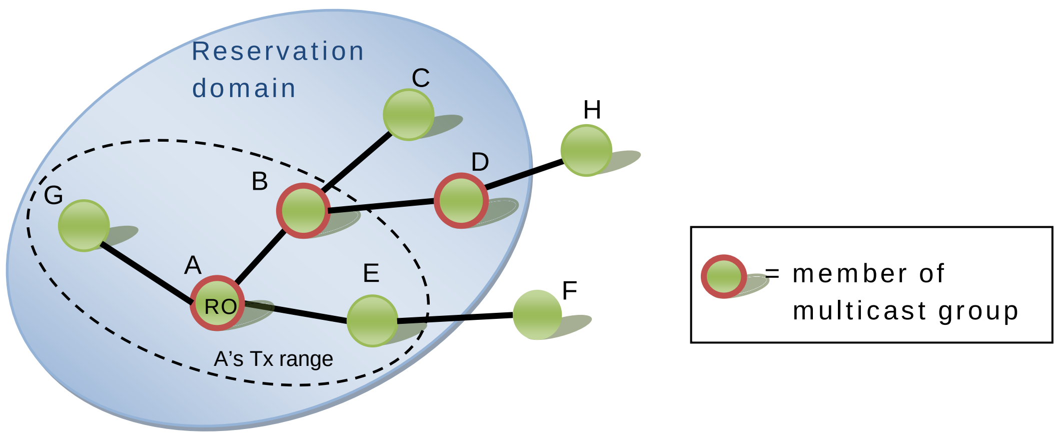


Figure 2.3 Reservation domain. Node A is the reservation originator (RO). Node B is the only recipient node, because it is the only multicast destination within A's transmission range. All nodes inside the blue area are included in the reservation domain, and must therefore be informed about A's reservation.

The reservation procedure is governed by the reservation protocol, whose objective is to ensure that the reservation is known by all nodes in the reservation domain. Details about the reservation protocol are given in section 2.4.

As stated above, reservations are done on a single-hop basis. If an end destination node is more than one hop way from the source (e.g. node D in Figure 2.3), one reservation must be done for every hop along the path, with each intermediate relaying node being a reservation originator. For IP data, reservations are made on a per packet basis, and are not allowed to overlap in time. This means that the reservation for the second hop cannot be initiated before the first hop reservation has been terminated. Relaying of data packets is performed at layer 3, so the link layer only provides single-hop transfer of data packets.
For multi-hop MV however, the link layer provides relaying using up to two voice relays. The *relay mechanism* provides an efficient way of establishing a sequence of reservations parallel in time, in addition to establishing a forwarding path at layer 3. See section 2.4.4 for a more detailed explanation of voice relaying.

Each node stores information about known slot reservations in its *reservation database (ResvDB)*. Database updates can either be caused by local events, or by the reservation protocol. Obviously, the different node's ResvDBs may be unsynchronized for long periods, due to delay in the reservation protocol, hidden node situations or lost signalling messages. Figure 2.4 shows an example ResvDB at node B in the given topology.

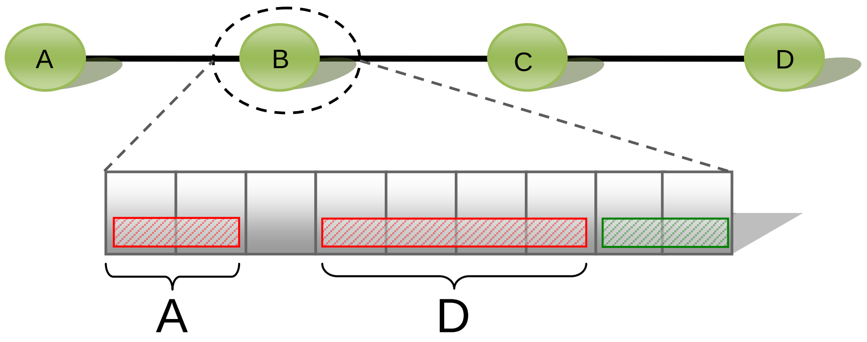


Figure 2.4 Reservation view seen from node B

Slot start	#Slots	Originator address	#Hops away	Resv. status
0	2	A	1	RESERVING
3	4	D	2	RESERVED
7	2	Super frame	-	RESERVED

Table 2.1 Node B's Reservation DB

The table contains one record for each known reservation. It stores information about the concerned slots, the originator of the reservation, and its distance to this node (B). Additionally, a reservation status field indicates whether the reservation is confirmed (RESERVED), or if there is an ongoing reservation period (RESERVING). Note that one specific time slot may be included in more than one reservation, due to spatial reuse. For instance, a node in the middle of a chain topology may have knowledge of reservations in both directions, reserving the same time slots. Also note that the database format is an implementation issue, and the above is merely an example.

2.2.2 Contention scheme

The description of the random access contention mechanism is subject to further study and is not implemented in the current version of the simulator.

2.2.3 Super frame access

The nodes are given access to the SF slots in a round robin manner. The protocol needed is subject to further study.

2.3 NBWF reference model (link layer focus)

The NBWF reference model was first introduced in [2], and is based on the OSI model [6].

This document makes significant updates to the NBWF reference model as presented in [2], including new and redefined SAPs (Service Access Point) at the link layer, the introduction of the management plane and handling of connection-oriented packet data.

2.3.1 Overview

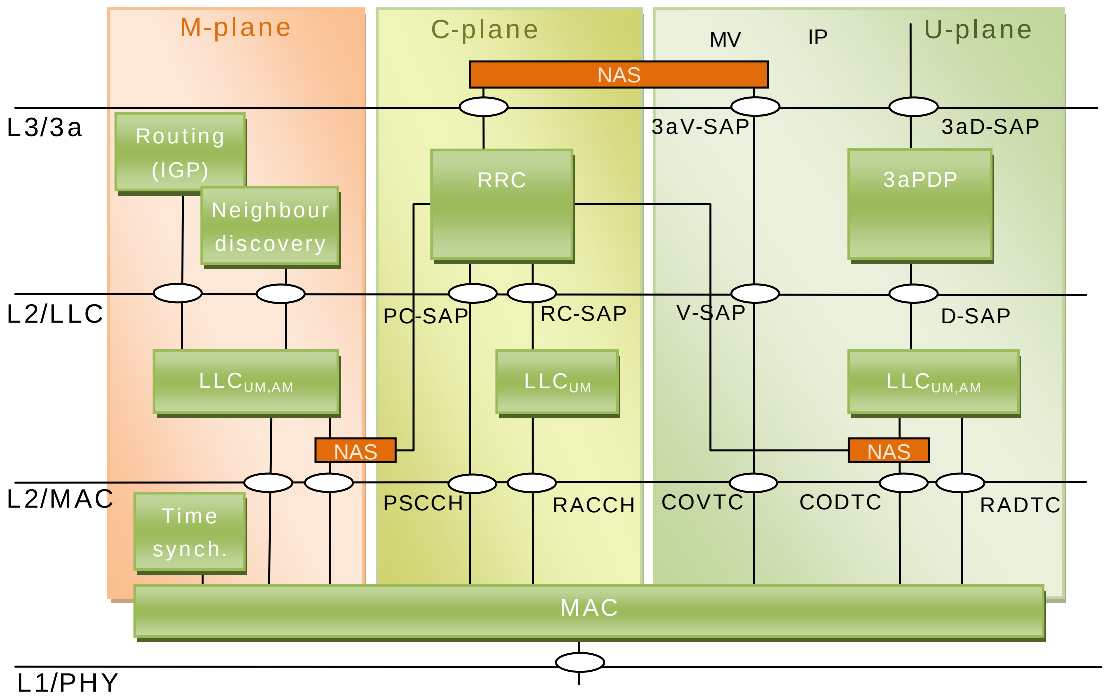


Figure 2.5 NBWF reference model. The green boxes represent protocols entities. The white ellipses represent SAPs (Service Access Points).

The control plane (C-plane) handles signalling related to the establishment and release of MAC and LLC connections, while the user plane (U-plane) handles forwarding of user (application) generated traffic. The management plane (M-plane) handles all types of management functionality, e.g. routing updates, TDMA synchronization etc. As seen in the figure, the three planes are terminated at the MAC layer, where all traffic is merged into the TDMA frame structure. Note that the M-plane functionality is not discussed further in this document, but is included in the figure to contribute to a complete understanding of the NBWF protocol stack.

As seen in the figure, the RRC (Radio Resource Control) is a central entity in the model. Its main responsibility is executing the reservation protocol in order to establish and release connections requested by U-plane or M-plane entities locally, or by peer entities when requesting relaying services. The RRC is also responsible for local and distributed admission control, as well as maintaining the ResvDB.

The NAS (None Access Stratum) separates the service primitives associated with setup and release of connections from the actual data to be sent. As an example, if the MV application issues a connect request (push-to-talk (PTT)), the primitive is redirected to the RRC which initiates the connection setup procedure, while the actual MELPe stream is passed right down to the U-plane LLC entity.

The following gives a brief description of the SAPs (Service Access Point) involved. The services are described in more detail in the following sections. The LLC layer provides four services to the upper layer, reached through the following SAPs:

- V-SAP (Voice-SAP): This is used by the application layer to send application generated voice packets over a previously established LLC-Voice-Connection. An LCID (Local Connection IDentifier) is passed as ICI (Interface Control Information).
- D-SAP (Data-SAP): This is used by the network layer to send connection-less data packets.
- RC-SAP (Random Access Control-SAP): This is used by the RRC to send random access control traffic.
- PC-SAP (Predetermined Slot Control-SAP): This is used by the RRC to send control messages in explicitly specified time slots.

LLC reaches the MAC layer services through five different SAPs:

- *PSCCH (Predetermined Slot Control Channel):* This SAP gives the RRC the opportunity to specify in which time slot a signalling message should be transmitted (e.g. CC messages).
- *RACCH (Random Access Control Channel):* This SAP is used by the RRC to send signalling messages using the contention scheme (e.g. CR for data traffic).
- *COVTC (Connection Oriented Voice Traffic Channel):* This SAP is used to send multicast (and unicast) voice packets in dynamically reserved time slots. (In addition, the transparent data service will use this SAP, but this is subject to further work and is left out of this document).
- *CODTC (Connection Oriented Data Traffic Channel):* This SAP handles transmission of U-plane data packets in dynamically reserved time slots.
- *RADTC (Random Access Data Traffic Channel):* This SAP is used for contention based transmission of short data packets.

2.3.2 MAC associations

According to the OSI standard, an *(N)-association*, where N denotes the layer, is a cooperative relationship among *(N)-entity-invocations*. In the context of the NBWF reference model, an (N)-entity-invocation is the instantiation of a protocol entity in a radio node. As an example, two NBWF radio nodes each hosts a MAC-entity-invocation (i.e. a MAC module), they communicate using the MAC-protocol and establish MAC-associations between each other. In the following, when using the term *entity*, we actually refer to an *entity-invocation*.

The reservation protocol is used in order to establish, maintain and terminate two types of MAC-associations, named *MAC-Reservations* and *MAC-2H-Reservations*, respectively:

- A *MAC-Reservation* is a one-to-many association, where the MAC entity in the *Reservation Originator* (RO) node is associated with the MAC entity in one or more *Reservation Neighbour* (RN) nodes. The set of RNs includes all nodes within radio range of the RO (i.e. all one-hop neighbours).
- A *MAC-2H-Reservation* is a one-to-many association between the MAC entity in an RN node and the MAC entity in one or more *Reservation 2-Hop Neighbour* (R2N) nodes. The R2Ns are the RO's two-hop neighbours included in the reservation domain.

Using the same example as in section 2.2.1, Figure 2.7 illustrates the roles (RO, RN and R2N) of all the nodes in the reservation domain and the associations between them. Note that none of the nodes outside the reservation domain is part of any association (by definition).

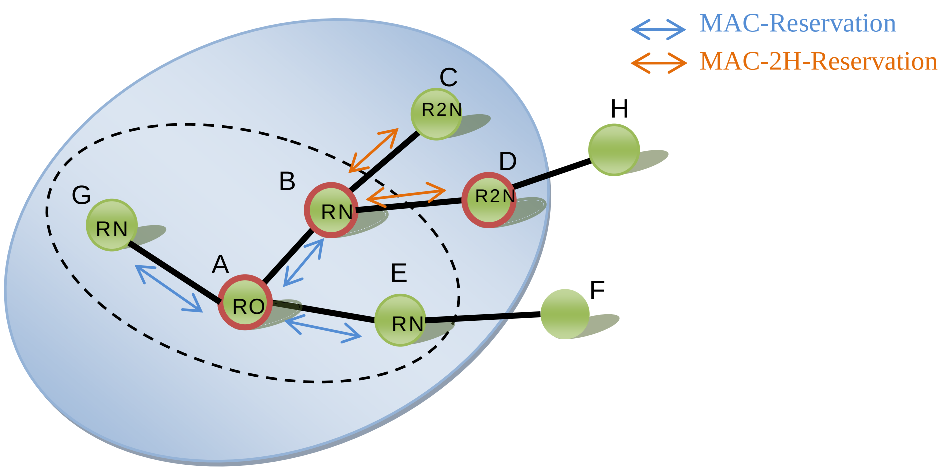


Figure 2.6 MAC associations within the reservation domain.

Each MAC-Reservation is identified with a globally unique reservation identifier (RID), which is included in all signalling messages. The same RID is used for the associated MAC-2H-Reservations (see Figure 2.7).

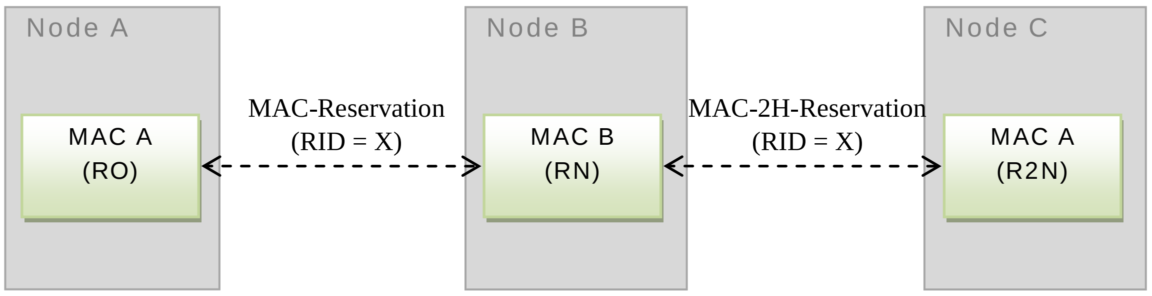


Figure 2.7 MAC-Reservation and MAC-2H-Reservation associations are relationships between the MAC entities in the different nodes in the reservation domain. The RID identifies the reservation, and is the same for all associations within the reservation domain.

A MAC-Reservation alone may be used to provide a single-hop one-way unreliable multi-endpoint connection oriented service to the upper layer (LLC). It is unreliable in the sense that no error recovery is performed, but the fact that transmissions occur in reserved time slots lowers the probability of collisions considerably. It is multi-endpoint, as all nodes within radio range may be in the set of recipients, given by the associated destination multicast address (provided as a parameter). Note that the MAC-Reservation is also used for unicast services, if the destination MAC address is a unicast address.

2.3.3 MAC voice services

This section describes the two MAC services used by the upper layer to establish, maintain and release a single-hop MAC-Connection. The services are reached through the MAC SAPs as described in Figure 2.5.

2.3.3.1 MAC connection-oriented voice service (U-plane)

The MAC voice service is a connection-oriented service used for transmission of voice packets in reserved slots. Connection establishment is carried out using the PSCCH SAP, resulting in a MAC-Connection in voice-mode. The voice SDUs are then delivered through the COVTC SAP during the data transfer phase.
This service is realized by establishing a MAC-Reservation as illustrated in Figure 2.8, thus having exactly the properties described above. An FCS (Frame Check Sequence) protects the MAC PCI from bit-errors, intentionally leaving the payload unprotected because of the error correcting/concealing properties of MELPe. The reservation establishment is handled by the RRC upon request by the layer 7 entity. An LCID common to the LLC and MAC layer is allocated for the reservation, which is used in all communication between the two layers in order to specify which connection to use. Inside each node, there is a one-to-one relationship between the LCID and the RID, and MAC needs to handle this mapping.

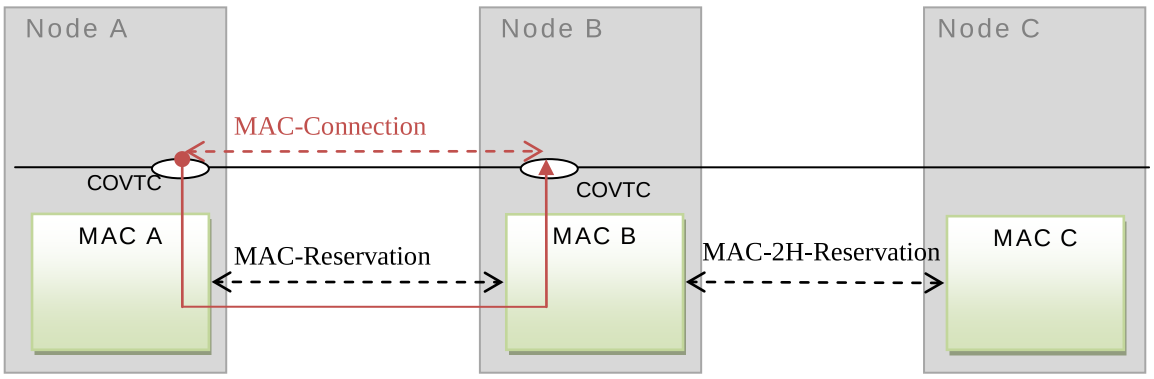


Figure 2.8 A MAC Connection in voice mode

2.3.3.2 MAC predetermined slot control traffic service (C-plane)

This service is reached by the LLC through the PSCCH SAP (C-plane). Some control messages must be sent in specific time slots, and cannot be sent using the MAC random access control traffic service. These are typically part of the reservation protocol. The time slot to be used must be attached as ICI.

2.3.4 MAC packet data services

This section describes the MAC layer services that provides connection-oriented and connection-less transmission of data (IP) packets. The upper layer (LLC) decides if connection-oriented or connection-less service is to be used.

2.3.4.1 MAC connection-oriented packet data service (U-plane)

The connection-oriented packet data service is used for transmission of data packets in reserved slots. Connection establishment is initiated using the MAC random access control traffic service in the C-plane (see section 2.3.4.3), resulting in a *data-mode MAC-Connection* in the U-plane, as illustrated in Figure 2.9. The data SDUs are then delivered through the CODTC SAP during the data transfer phase.

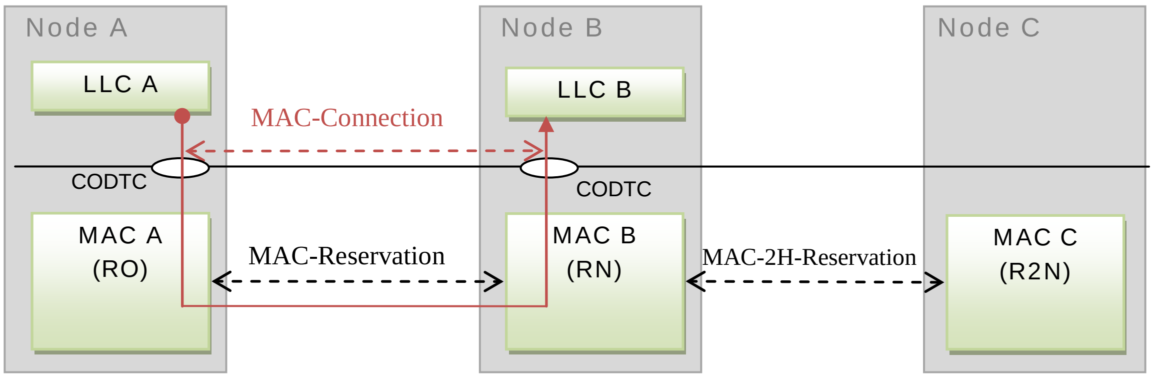


Figure 2.9 A MAC-Connection in data-mode

In the data transfer phase, a data-mode MAC-Connection has almost the same properties as voice-mode. I.e. it is a single-hop one-way unreliable multi-endpoint connection oriented service. The main difference between voice-mode and data-mode lies in the reservation establishment phase, as will be described later. (Segmentation and ARQ functionality is handled by the LLC layer). Another important difference is that the FCS now covers the entire MAC PDU.

The connection-oriented packet data service also provides return channel functionality, which can be used by the LLC layer to send feedback reports within the reserved time slots (i.e. without establishing a new reservation in the opposite direction).

2.3.4.2 MAC connection-less packet data service (U-plane)

This service is reached through the RADTC SAP in the U-plane, providing a random access data service for the LLC layer. It is a connection-less one-way single-hop unreliable unicast or multicast service. The service is limited to payloads that can be carried within a single MAC SDU, whose size is limited by the number of available consecutive time slots in the TDMA frame. The FCS covers the entire MAC PDU, thus not allowing bit-errors in the payload. MAC uses a contention based access mechanism for this service. The contention mechanism is not yet specified and is not discussed further in this document.

2.3.4.3 MAC random access control traffic service (C-plane)

This service is reached by the LLC through the RACCH SAP (C-plane), and is used by the LLC to send random access control messages. This service is similar to the connection-less data service provided through the RADTC SAP (U-plane). However, control traffic takes precedence over data traffic both in terms of scheduling (internally in a node) and the contention mechanism. Currently, the only application of this service is CR messages for the connection-oriented data.

2.3.5 LLC connection-oriented voice service

The LLC voice service extends the MAC voice service with U-plane buffering of outgoing voice SDUs. The LLC layer provides an *LLC-Voice-Connection* to the upper layer reachable through the V-SAP, and there is a one-to-one mapping between a MAC-Connection (in voice-mode) and an LLC-Voice-Connection as illustrated in Figure 2.10. Thus, the LLC-Voice-Connection inherits all properties of the underlying single-hop MAC-Connection, and their holding times are also identical.

Note that there is currently no LLC protocol entity for voice in the U-plane, as no PCI is added at the LLC layer. The only functionality is buffering of outgoing voice SDUs. The exact buffering solution is however subject to further work. The current approach is that the concatenation is performed at layer 7, so that one LLC SDU contains a number of voice frames. An alternative, more flexible solution is that one LLC SDU (U-plane voice) contains exactly one MELPe frame, and the LLC entity concatenates a number of SDUs (i.e. 9 according to the current specification) and builds an LLC PDU. The latter method has the advantage that it is possible to initially transmit a smaller number of MELPe frames if available at the start of the reserved slot, without needing to wait until 9 frames are buffered. This reduces the experienced voice delay, but it also requires the adding of PCI at the LLC layer, and consequently introducing a LLC protocol entity.

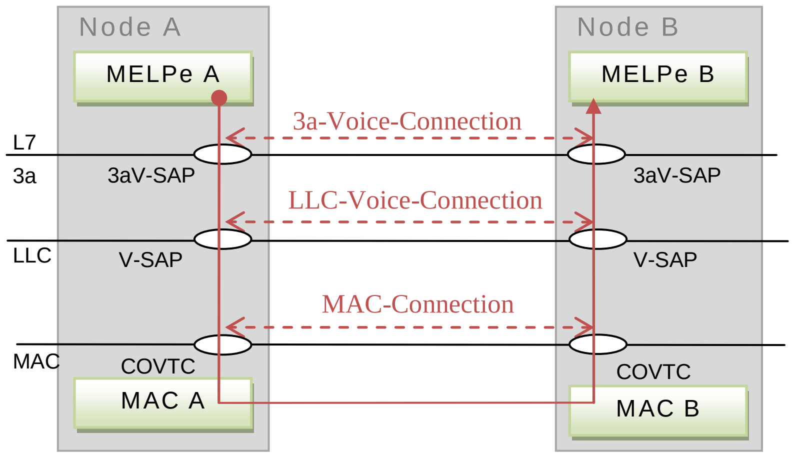


Figure 2.10 A single-hop LLC Voice Connection

2.3.6 LLC packet data service

The LLC layer offers a single hop connection-less unicast data service to the upper layer through the D-SAP. The LLC entity may decide to either send the SDU over a MAC connection (data-mode) or using the MAC connection-less data service. The decision is taken based on the SDU's size and priority. Note that in either case, the service provided by LLC to the upper layer is connection-less. If using a MAC Connection, an *LLC-Data-Association* between the U-plane data LLC entities in the source and destination nodes is established as described in Figure 2.11. When using the connection-oriented delivery method, the LLC entities perform segmentation and optionally ARQ. If using ARQ, the feedback reports are sent over the return channel provided by the MAC connection-oriented data service.

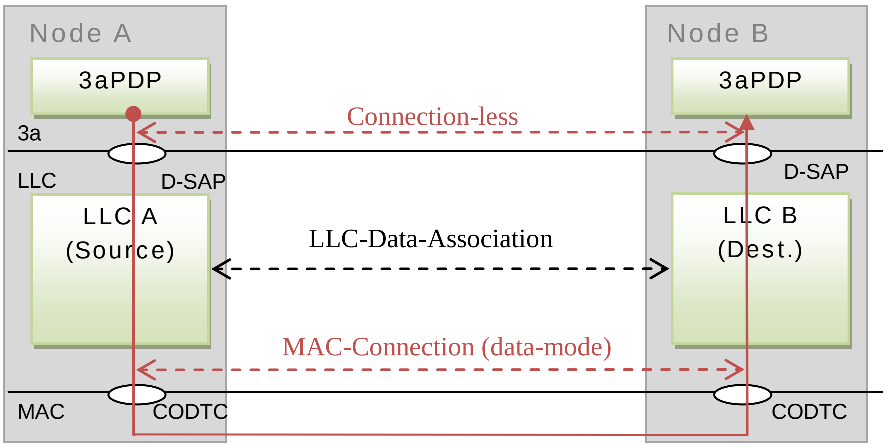


Figure 2.11 An LLC Data Association when LLC uses the MAC connection-oriented data service.

2.3.7 C-plane LLC services

2.3.7.1 LLC random access control service

This service lets the RRC entity send random access control packets through the RC-SAP. Priority queuing and lifetime control are performed. This is only used by CR messages for the connection-oriented data service.

2.3.7.2 LLC pre-scheduled control service

This gives direct access to the *pre-determined slot control traffic service* provided by the MAC layer through the PC-SAP. It is used only by the RRC entity, which therefore needs to be “TDMA aware”. It is directly mapped to the MAC predetermined slot control traffic service (PSCCH) and adds no functionality.

2.3.8 3a connection-oriented voice service

This is the connection-oriented multicast voice service provided to the layer 7 voice application through the 3aMV-SAP. The 3a voice service primarily extends the LLC voice service with multi-hop capabilities. The 3a layer provides a *3a-Voice-Connection* to the upper layer, possibly spanning multiple hops using voice relays. A 3a-Voice-Connection is identified by the End-to-end Connection ID (ECID). This is a globally unique identifier created by the PTT-node. Connection setup and release primitives from the application are directed to the C-plane RRC entity through the NAS (Non Access Stratum), while the U-plane 3aV-SAP is used during the data transfer phase. The RRC uses the reservation protocol to establish and release both MAC-Connections and LLC-Voice-Connections. The reservation protocol is described in section 2.4.

In order to provide multi-hop MV, the 3a layer provides voice relaying functionality. The RRC is responsible for establishing and maintaining *3a-Voice-Relay-Associations* in the U-plane. A 3a-Voice-Relay-Association is simply a mapping between an incoming and an outgoing LLC-Voice-Connection (see Figure 2.12). In the data-transfer phase, incoming LLC SDUs are simply forwarded to the associated outgoing connection. Note that there is no 3a protocol in the U-plane for MV traffic.

2.4 Reservation protocol

The reservation protocol's main objective is to distribute information about a reservation (as defined in section 2.2.1) to all nodes in the reservation domain in order to avoid collisions. The robustness is challenged by link errors, hidden nodes and topology changes. At the same time, it is important that the reservation setup delay is kept at acceptable levels for real-time applications.

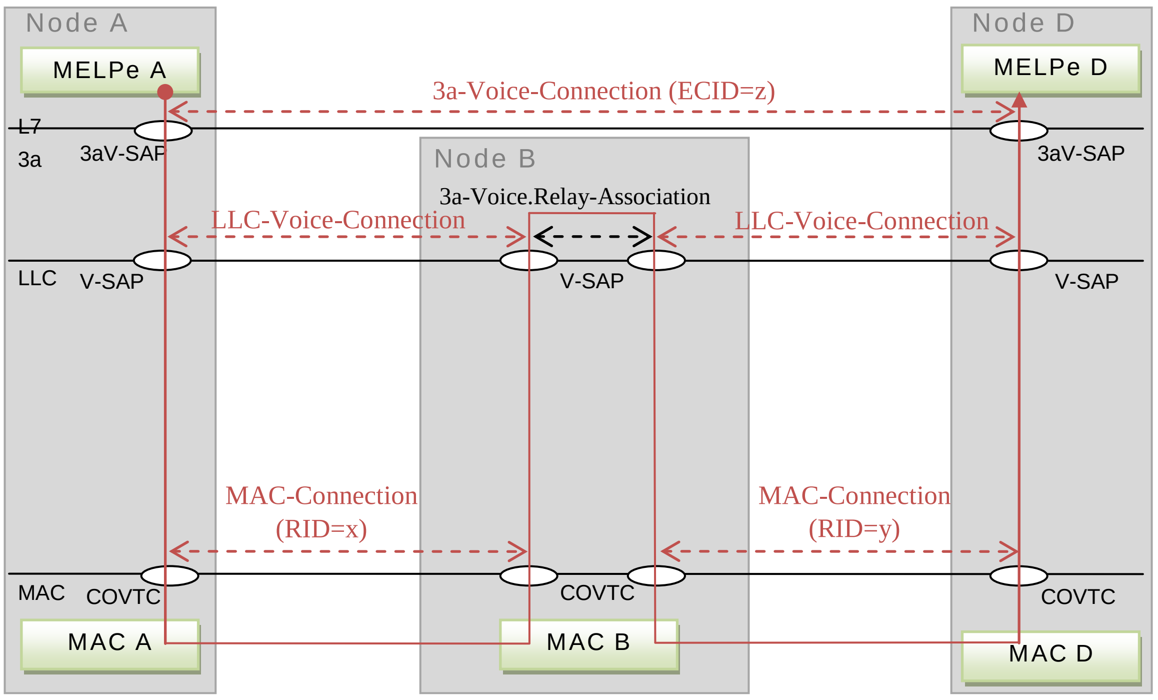


Figure 2.12 A two-hop LLC-Voice-Connection

The basic operation of the protocol is as follows: The RO broadcasts a *Connect Request* (CR) signalling message to all its one-hop neighbours, which then takes the role as RNs. This establishes the multi-endpoint MAC-Reservation association between the RO and the RNs. A selected set of the RNs respond with a *Connect Confirm* (CC) message, indicating whether the reservation was accepted (positive CC) or not (negative CC). This *CC-set* is selected by the RO such that all two-hop neighbours inside the reservation domain will see at least one CC. The CC has two primary functions:

- Information about the reservation is distributed to the two hop neighbours, in order to reduce collisions with the user data sent in the reserved slots. This establishes the MAC-2H-Reservation between the RN and the R2Ns.

- Avoid overlapping reservation domains, in the cases where the CC-node already knows of a reservation of the same time slot(s). In this case, the CC-node should send a negative CC.

Note that topology information about the two-hop neighbourhood is required to select the CC-set. This information must be collected by a neighbour discovery protocol, most likely shared with the layer 3 routing protocol. Likewise, when the originator wants to release the reserved time slots, it broadcasts a *Disconnect Request* (DR) message, and the nodes included in the CC-set , the *CC-nodes*, respond with a *Disconnect Confirm* (DC) in order to notify all two-hop neighbours.

Section 2.4.2 and 2.4.3 will describe the reservation establishment and release procedures, respectively. After that, the MV relay establishment is described.

2.4.1 The CC-schedule

The CC-schedule is a central concept of the reservation protocol. It specifies when (in which slot) each CC-node should transmit its CC. It is important that the CCs don't collide with transmissions from known reservations. It is also important that all the participating nodes (the originator and all the 1-hop neighbours) are consistent in their perception of the CC-schedule. The CC-set is included in the CR PCI, but this only states the sequence of the CCs, and not which slots to use. As explained in section 2.3, a MAC-Connection is in either *voice-mode* or *data-mode*, and the main difference between the two modes is the way the setup procedure is performed. While the voice mode uses a fixed CC-schedule, the data mode uses a flexible CC-schedule where CCs can be sent in any free slot.

2.4.1.1 Voice-mode

When establishing a voice-mode MAC-Connection, signalling messages are primarily sent in the MVS slots. This protects against collisions with data traffic. The originator of the reservation sends its CR in one of the MVS slots, say slot 1. The CCs for this reservation should then be sent in slot 1 and slot 1+m of the next TDMA frame, where m is the number of MVS slots. In addition, if the originator requests a relay, the relayed CR will contain a piggybacked CC (relays are always included in the CC-set). Figure 2.13 gives an example when m=2. Here, the maximum size of the CC-set is 5 and 4 for the first and second CR, respectively. Note that the CCs sent in the MVS slots are more probable to succeed, so the most important CC-nodes (e.g. covering the highest number of two-hop neighbours) should be placed first in the CC-set.

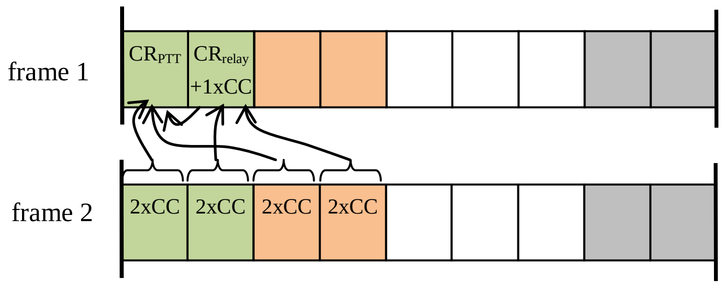


Figure 2.13 CC-schedule (voice-mode). Refer to Figure 2.2 for color codes.

2.4.1.2 Data-mode

In data-mode, only the DU and GU slots are available for signalling messages (see Figure 2.2). The CR is sent using a contention scheme, and may be sent anywhere in the available slot range. The CC-schedule starts at the beginning of the first available time slot after the end of the transmitted CR. In principle, an unlimited amount of CCs may be included in the CC-set, but the PCI format enforces a maximum number to be included. Note that for unicast, only a single CC is required, and this could potentially be sent right after the reception of the CR. For multicast however, slot boundaries should probably be obeyed.

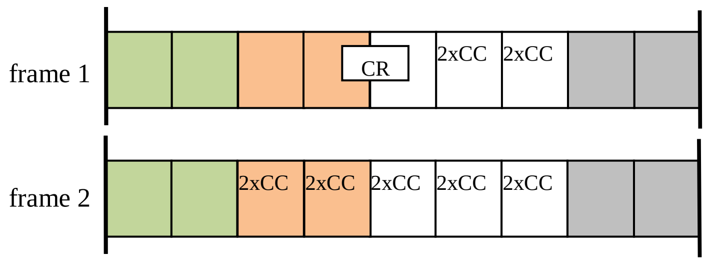


Figure 2.14 CC-schedule (data-mode). Refer to Figure 2.2 for color codes.

2.4.2 Reservation establishment

2.4.2.1 Reservation originator (RO)

The reservation establishment phase is initiated when the originator node broadcasts a CR signalling message to all its 1-hop neighbours. For MV (voice-mode MAC-Connection), this is sent in the first MVE slot (see section 2.4.4), while for connection-oriented data (data-mode), it is sent in a free slot using a contention scheme. Before sending the CR, the originator calculates the CC-set and the CC-schedule. The CR contains the CC-set (and possibly additional info about the CC-schedule in case the flexible variant is used), information about which time slots that are to be reserved, and the globally unique Reservation ID (RID). After sending the CR, the originator sets the *CC-timer* to fire at the end of the CC-schedule period, when all CCs are expected to have arrived. When this timer goes off, the originator is able to decide whether the reservation was successful or not. If successful, the reservation is marked as active, and the data transmission can start in the next occurrence of the reserved slot(s). Otherwise, the reservation is cancelled, and the upper layer is notified about the reservation failure.

The success criterion depends on whether the connection is in voice-mode or data-mode. When setting up a voice-mode connection, which typically has multiple recipients (as defined in section 2.2.1), the protocol does not require that the RO receives any of the CCs. This allows (partial) successful connection establishment when some of the targeted CC-nodes have moved out of the transmission range. This is a likely situation because of the limitations on the frequency of routing updates in narrowband networks

In data-mode however, when setting up a unicast data connection, it is essential that the destination node, the only CC-node, is reached. Therefore, in this situation, it is required that the RO receives the CC.

Multicast data connections are subject to further study, and are not discussed further in this document. It is however reasonable to assume that the multicast data service will be unreliable (no ARQ), so an approach similar to voice-mode is likely.

2.4.2.2 CC-nodes

When a node receives a CR and finds itself in the CC-set, it takes the role as a CC-node. It must then find out whether it should send a positive CC, a negative CC or no CC at all. The action to take depends on the existence of ongoing voice sessions as well as the originator's success criterion:

Data-mode (unicast)
In data mode, it is required that the CC is received by the RO. Consequently, there are no reasons for sending a negative CC. The choice is then between a positive CC and no CC. A positive CC is sent if the CC-node is not aware of any existing data reservations and if the requested slots are not currently in use by a voice connection. It is however possible to accept a data reservation during a voice session, if only GU slots are requested. If the above criteria are not met, the new data connection is not accepted, and no CC is sent.

Voice-mode
In voice-mode, the RO will abort the connection establishment procedure only if receiving a negative CC. The CC-node only accepts the new connection if it is not aware of any existing voice reservation. In this case, it must send a positive CC, in order to distribute the information about the reservation to the outer parts of the reservation domain (i.e. establishing associations with the R2Ns). A conflict arises however, when the CC-node cannot accept the new request due to an existing voice connection. On the one hand, if it chooses to send a negative CC, the transmission may disturb the ongoing voice connection. On the other hand, if it chooses not to send the negative CC, the RO of the new connection may conclude that the request was accepted, and the result is two parallel voice sessions with potentially high collision probability. This situation is by far worse than a single CC collision. However, the likelihood of this situation to occur depends on the distance between the existing and the new RO, which is known to the CC-node.

Figure 2.15 and Figure 2.16 show example scenarios when this distance is three and two hops, respectively. It is clear that the latter situation should be avoided, so in this situation node B should send a negative CC. When the distance is three hops however, the two connections may exist at the same time without causing collisions, but there is still a chance that the signals from node D will cause significant noise at node B. It is therefore not obvious whether the negative CC should be sent or not. This is a trade-off between utilisation and robustness and is a subject of simulation studies. *Table 2.2* summarises the CC-node's actions.

2.4.2.3 1-hop neighbours

All 1-hop neighbours, including the CC-nodes, must inspect/calculate the CC-schedule when receiving a CR, and make sure that they don't send any random access traffic that interferes with the CC's. They must also update their ResvDBs in order to avoid collisions in the reserved slots.

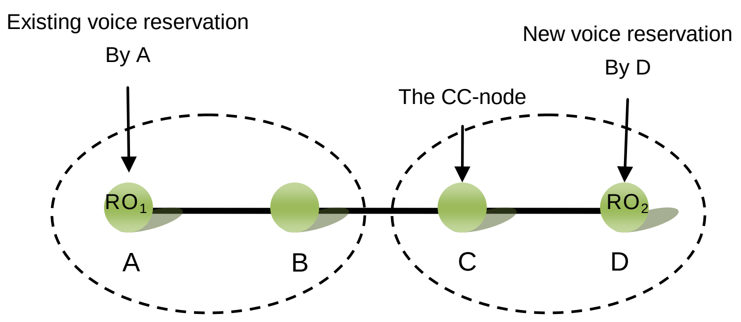


Figure 2.15 Distance between existing and new originator is three hops. This is a likely situation for the given topology, as node D cannot know about the existing reservation.

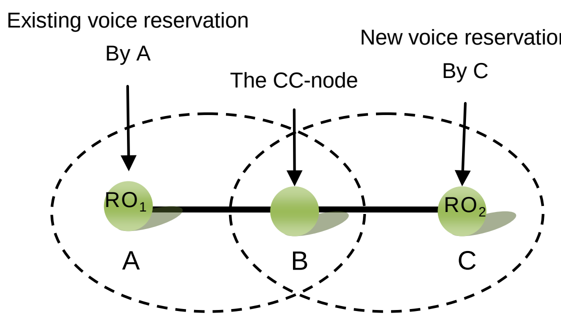


Figure 2.16 Distance between existing and new originator is two hops. This may happen if node C initially missed B's CC for the existing reservation, or if node B was not a CC node for the existing reservation. This is clearly an unwanted situation.

Existing voice reservations	**Action**
None	Send positive CC
Distance between ROs: 2	Send negative CC
Distance between ROs: 3	TBD

Table 2.2 CC-node's actions when receiving voice-mode CR

2.4.2.4 2-hop neighbours

When a node receives a CC without receiving the initial CR (using the RID), it takes the role as an R2N. This ensures that the node doesn't transmit in the reserved slots. The R2N role is given up either if explicitly signalled by the RN, or if a timer expires.

There is also a possibility that it missed the CR due to channel interference, and that the MV voice PDUs may actually reach the node. Whether the node should then become an RN or the PDUs should be ignored is not yet decided.

2.4.2.5 Recipients

The CR contains a destination address field, which can be either a unicast or multicast MAC address. This is the destination for the data/voice to be sent over the given MAC-Reservation (note that the CR is sent as broadcast, and therefore is picked up by all neighbours). If the node finds itself in the destination group, it takes the role as recipient. This involves preparing the LLC layer for the incoming data.

2.4.3 Reservation release

An established reservation may only be released by the originator itself, not by another node. (A data-mode reservation may however be pre-empted when a MV reservation is initiated, by implicitly reducing the number of slots for the data reservation. More specifically, it loses all the DU slots.)
Equivalent to the reservation establishment procedure, the intention of the release procedure is to notify all nodes in the reservation domain that the reservation is to be released and that the time slots will be ready for use by other nodes.
When the originator wants to release a reservation, it broadcasts a *Disconnect Request* (DR) message to its neighbours. The DR is transmitted in one of the reserved slots. (An alternative is to transmit the DR in a free slot in order to speed up the disconnect procedure, but this increases the likelihood of collisions). The CC-nodes (using the CC-set from the CR) transmit their *Disconnect Confirm* (DC) messages according to the DC-schedule, so that also 2-hop neighbours are notified. The DC-schedule (semantically equivalent to CC-schedule) always consists of the reserved slot range (that is to be released).
Additionally, a soft reservation mechanism is needed in order to time out reservations that for some reason have not been explicitly released. This may for instance happen when the DR message is lost, or if the connection establishment phase failed. Nodes that have registered a reservation in their ResvDBs, but do not receive any packets within a certain time delay, must consider the reservation released. If the originator does not have a data/voice packet available at the start of the reserved time slot, but it does not intend to release the reservation, it should transmit “keep-alive” messages in order to avoid reservation timeout at its neighbours.

2.4.4 Establishment of multi-hop LLC-Voice-Connections

When source and destination(s) are more than one hop away from each other, one reservation must be done for each hop, using different time slots. Each reservation thus exists parallel in time. MV requires short setup delays, typically in the order of 300-400ms. For this reason, it is important to perform the signalling as fast as possible, while also having a certain level of robustness.
Note that this does not apply to relaying of data (IP) packets, where only one reservation is allowed to exist at a time.

2.4.4.1 Dedicated MV relays

As specified in [3], a single voice flow transmitting MELPe at 2.4 kbps requires two time slots (merged) when using the chosen TDMA slot size (22.5 ms). Each relay requires two additional slots, so two relays are possible with the proposed TDMA frame structure. Recall that two slots have been statically reserved for super frame. For this reason, the current relay mechanism is limited to two relays. Also, to simplify the protocol, it is assumed that these are dedicated relays, and that all nodes are pre-configured to use them if they are within radio range. An alternative, more flexible solution would be to let the originator decide which node(s) to use as relay based on information from the layer 3 routing protocol, in order to reach as many recipients as possible. Such *automatic relaying* is clearly a wanted feature, but left for future versions of the MAC specification due to the additional complexity.

2.4.4.2 The multi-hop reservation procedure

Each originator (PTT-node and relay nodes) sends a CR, each reserving two time slots. As indicated in section 2.4.1.1, all the CRs are transmitted subsequently in the MVS slots.
Another important optimisation is the possibility for the relays to piggyback their CC for the previous hop on their next hop CR. With this follows the requirement that relays are always included in the CC-set. Due to the overlapping information in the two messages, relatively few bits must be added to the CR.
These optimisations ensure that a larger part of the reservation domain is informed about the reservation quickly, without interference from data traffic. If additional CCs are needed, they will be scheduled according to the fixed CC-schedule as described in section 2.4.1.1. If an existing data reservation occupies the DU slots, and if the originator of that reservation receives any of the CR or CC messages, it will immediately stop using (pre-empt) the reserved DU slots without initiating a disconnect procedure (the connection may still continue to use the GU slots).

Figure 2.17 illustrates the connection establishment procedure, in a simple chain topology consisting of four nodes including two MV relays. The shaded areas represent slot reservations. The vertical dimension of the area represents the slots that are reserved, while the horizontal dimension represents the reservation domain. The thick line located on one of the node’s time line indicates that this node is the RO of the reservation.
In this example, node B has reserved slots 3-6 before node A initiates the MV connection establishment procedure by transmitting CR_x in the first slot. Node R1 will operate as a relay node and as a CC-node, and send CR_y in the next slot with CC_x piggybacked. R2 will do likewise, and will require a CC from R1. When B receives CR_z from R2, it passively releases slots 3 through 5, leaving only a reservation for slot 6. The result of the procedure is three separate voice-mode reservations, each reserving different time slots and potentially having different reservation domains.

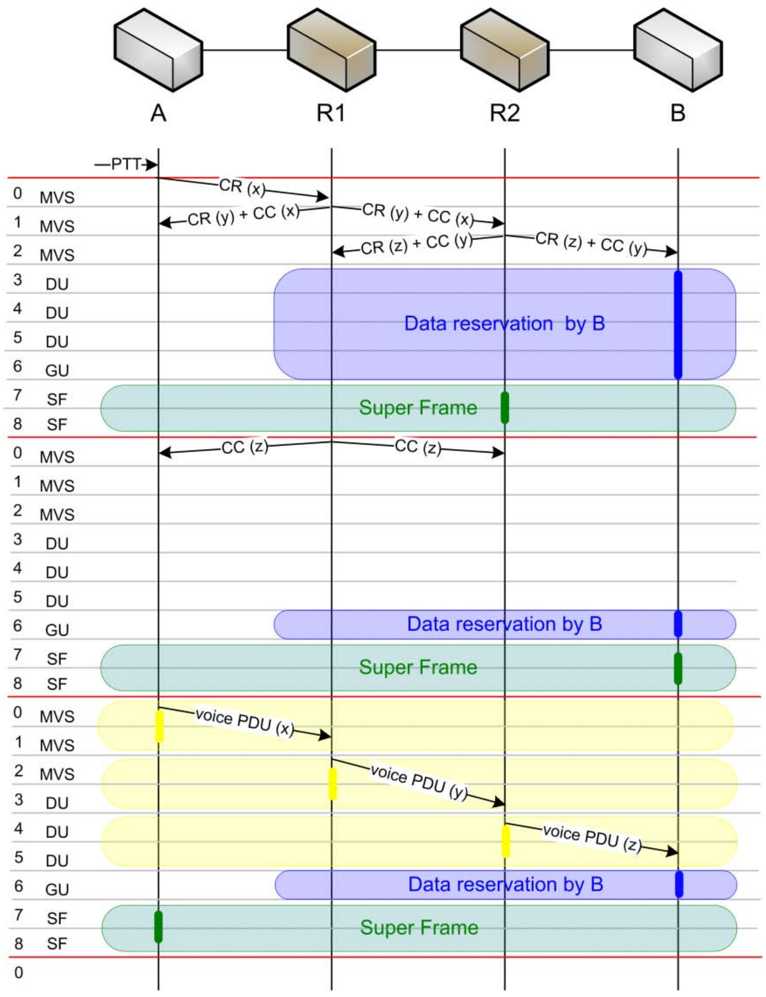


Figure 2.17 MV connection setup example with two MV relays and an existing data reservation. Letters within parenthesis denote RID.

2.5 Connection identifiers

It is important that the nodes are able to associate all incoming signalling messages (CR/CC/DR/DC) and U-plane PDUs with the correct MAC-Reservation. It is also important that the nodes are able to deduce that two different reservations in fact belong to the same 3a-Voice-Connection. For instance, if a node is within range of both the initiator and a relay, it will receive two CRs with different RIDs. If it is able to conclude that the second CR is a relayed version of the first, it can safely ignore the MV voice packets arriving on this reservation (or utilize the redundancy in case of corruption). For these reasons, all signalling messages should contain both the RID and the ECID. With this in mind, the ECID and RID is designed as described in the following.

2.5.1 End-to-end connection ID (ECID)

The End-to-end connection ID (ECID) uniquely identifies a 3a-Voice-Connection (as defined in the reference model, section 2.3.8), and is generated by the PTT-node at the LLC layer. To ensure that it is globally unique (i.e. different from ECIDs generated by other PTT-nodes), it should contain the PTT-node's Node ID. It is also important that the ECID of one connection is not mistaken as belonging to another connection (earlier or parallel) from the same PTT-node. For this reason, the ECID should also contain a serial number and/or the SAP (Service Access Point) identifier:

ECID
Initiator node ID
SAP ID
Serial number

Table 2.3 End-to-end Connection ID

2.5.2 Reservation ID (RID)

The single-hop Reservation ID (RID) uniquely identifies a MAC-Reservation, and is generated by the RO's MAC entity. To ensure that it is globally unique (i.e. different from RIDs generated by other ROs), it should contain the Node ID of the RO. To ensure that it is different from RIDs of earlier/parallel reservations originated by the same node, it should also include a serial number. However, since all messages must include the ECID, and the ECID is already globally unique, this can be used instead. Thus, the RID simply contains the originator's address and the ECID:

RID	
Originator Node ID	
Initiator Node ID	
SAP ID	ECID
Serial Number	

Table 2.4 Reservation ID

3 NBWF simulator overview

This section gives a brief overview of FFIs NBWF network simulator.

3.1 Development stages

A step-wise development model is chosen. Step 1 (specified in this document) includes basic MV functionality, whereas data link protocols, neighbour discovery, routing and automatic relaying are added as the protocol specifications advance. The primary goal of the first version is to study the performance of the reservation protocol when subject to random packet loss or collisions with background data traffic. Step 2 will involve adding the data link protocols (segmentation and ARQ functionality), before introducing routing and mobility models in step 3.

3.2 Simulator software

The NBWF network simulator is based on OMNeT++ v4, a discrete event simulation environment. OMNeT++ provides a component architecture where the basic building blocks are *simple modules*, communicating with each other through *gates*. The functionality of the simple modules are programmed using C++, and assembled into larger components (compound modules) using the high level language NED (NEtwork Description). OMNeT++ provides an extensive set of protocol implementations, ranging from physical layer to application layer protocols. For this project however, where the aim is to study new protocols on all layers, all simple modules have been written from scratch. The NBWF simulator has adopted the runtime system and statistics collecting tools from the OMNeT++ based open source projects oTWLAN and oProbe [7;8].

3.3 NBWF simulator modules

The NBWF simulator modules are organized as shown in Figure 3.1. As seen in the figure, the simulator design is closely related to the NBWF reference model described in section 2.3.1.

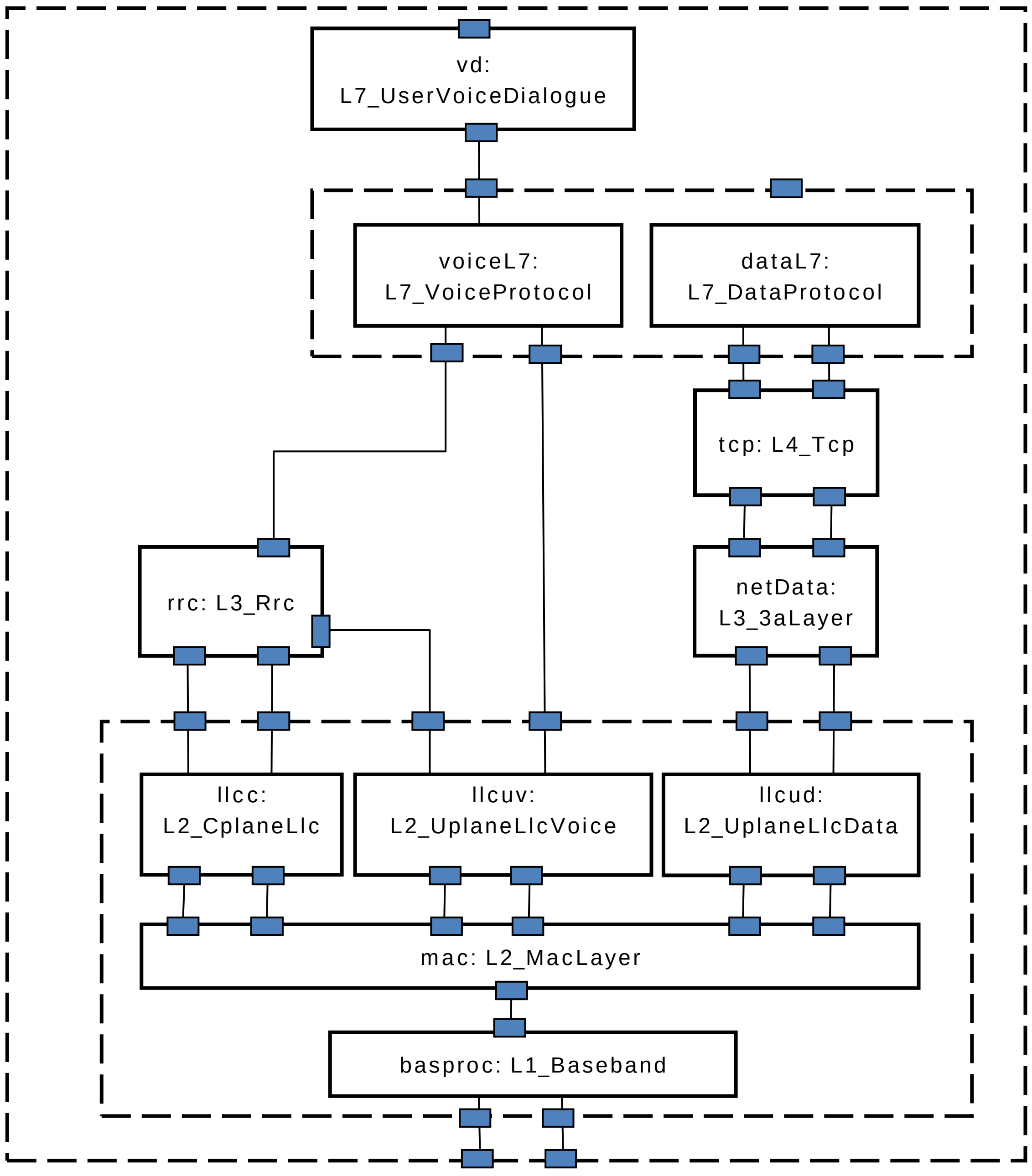


Figure 3.1 Simulator modules

4 Physical layer implementation

This section gives an overview of the NBWF radio [9] and explains how it is implemented in the simulator. The current implementation is based on the second edition of the CRC proposal, but is subject to modification according to the fourth edition in the near future. Section 4.3 presents some validation results of the simulator. Figure 4.1 shows the air frame format while Table 4.1 presents the basic radio parameters.

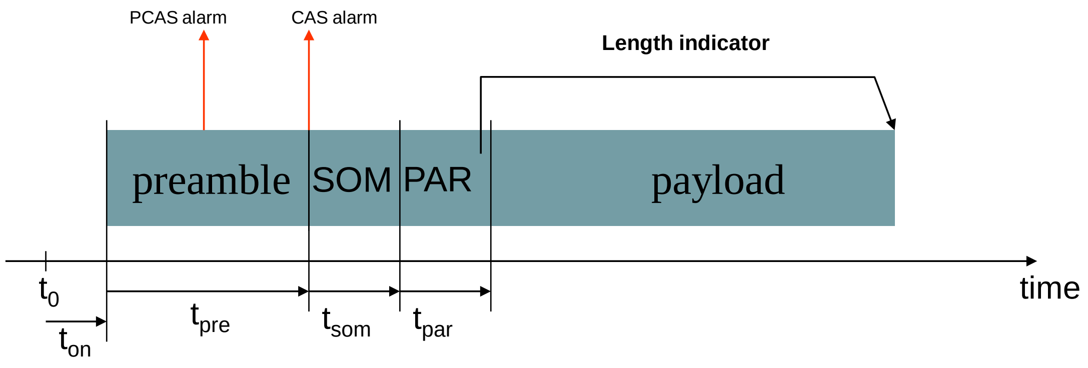


Figure 4.1 Physical layer frame format.

Name	Description	Value	Unit
$f_{payload}$	Payload transmission rate	16.025	kbits/s
t_{pre}	The length of the preamble field	3.328	msec
t_{pcas}	PCAS delay	$t_{pre}/2$	
t_{som}	The length of the SOM field	1.664	msec
t_{par}	The length of the PAR field	2.496	msec
t_{on}	Receive to transmit switching delay	1	msec
n_t	Thermal noise of the receiver	-119.85	dBm

Table 4.1 Physical layer parameters

4.1 Reception

A radio in the synchronisation search (SS) mode detects a transmission on the radio channel when a carrier sense (CAS) alarm is generated. If the signal-to-noise ratio (SNR) is sufficiently high, the radio may get a preliminary CAS (PCAS) alarm before the CAS. This is beneficial since a PCAS gives faster detection of a busy channel.

Figure 4.2 depicts the situation when all air frame sections are decoded successfully. The simulator is designed to minimise the signalling between the PHY and the MAC. The payload field of the air frame is therefore delivered to the MAC as a single unit carried by a *PHY-Data.indication* service primitive. Only three signals are sent to the MAC layer entity in this case. If PCAS succeed but no CAS alarm arrives, the baseband processor sends a *PHY-is-idle* signal at the CAS time instance (t_{on} + t_{pre}) and the radio enters the SS-mode. A necessary condition to detect a start-of-message (SOM) is that the CAS alarm has been received. In this case, the MAC entity has been informed by the *PHY-is-busy* signal, which has been sent either at the PCAS time or the CAS time instance. If the SOM fails, the baseband issues a *PHY-is-idle* signal and enters the SS-mode (Figure 4.3).

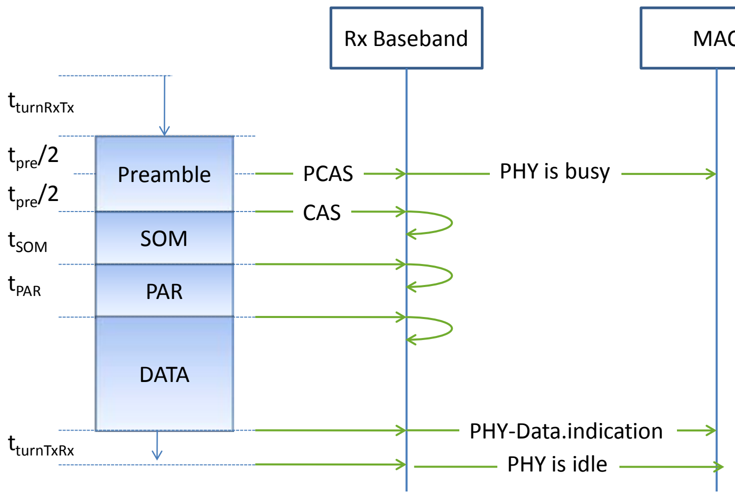


Figure 4.2 Incoming air frame. All air frame sections are detected successfully.

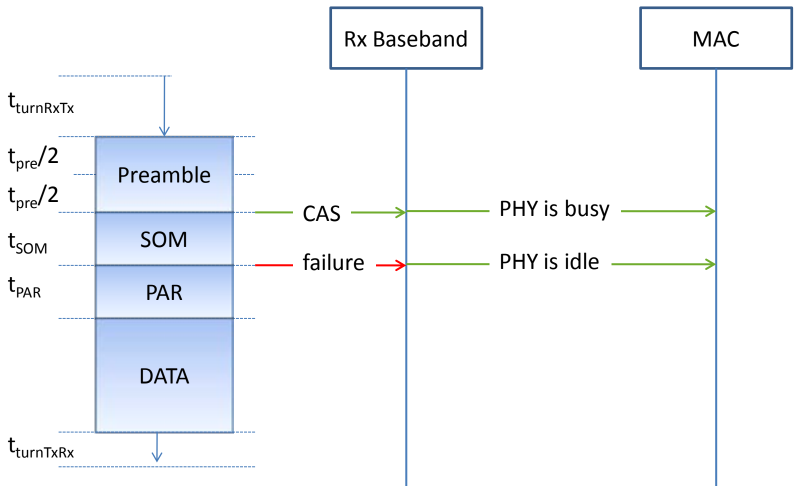


Figure 4.3 Illustration of a start-of-message (SOM) failure event.

4.2 Transmission

The MAC entity starts a transmission (Figure 4.4) by sending a *PHY-Data.request*. This service primitive contains the payload to be sent as a single radio burst. The baseband emits a *PHY-is-idle* signal when the radio has returned to the SS-mode.

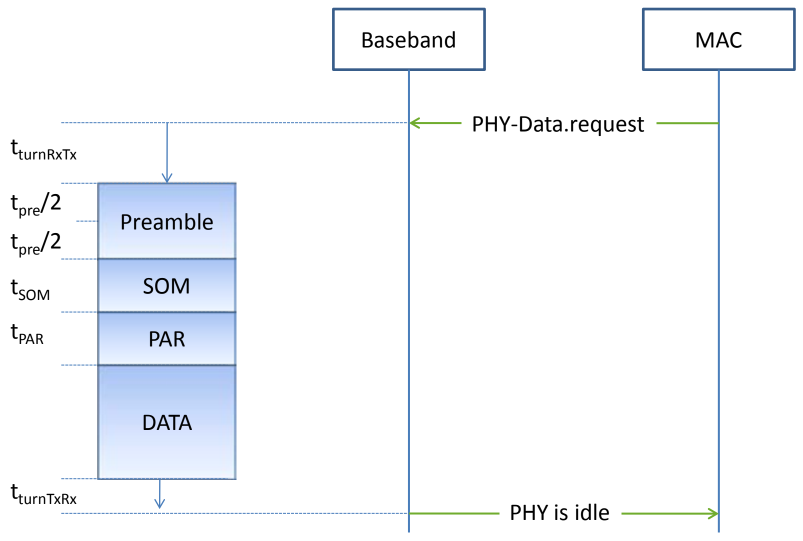


Figure 4.4   Outgoing air frame.

4.3 Validation

The purpose of this test set is to test the correctness of the physical layer time sequence diagram. Test traffic comes from layer 7 and is based on UDP traffic that does not request use of ARQ. The NBWF MAC layer protocol (module mac: L2_MacLayer in Figure 3.1) is replaced by a random access protocol (class L2_MacLayerTest1).

The MAC protocol uses an access delay function *D*, which is the sum of a fixed component and a random component, given by:

$$D(a_p,b_p) = a_p \cdot t_v + RandUniform[0, b_p \cdot t_v],\ a_p,b_p : \text{Integer} > 0,\ p = 0...3$$

where a_p is named the *priority delay* access factor, t_v is the *vulnerable period* and b_p determines the upper bound of the uniformly distributed random delay. t_v is the point in time where a receiver has the first opportunity to detect a busy channel. For the NBWF radio this becomes $t_v = t_{on} + t_{pre} = 4.328$ msec.

The MAC protocol handles four priority levels with the parameters:

$$\{a_0,a_1,a_2,a_3\} = \{6,5,4,3\} \quad \text{and} \quad \{b_0,b_1,b_2,b_3\} = \{100,50,30,20\}$$

We use a network containing two nodes only and with simplex deterministic traffic since this give an easy scenario to analyse. Node 0 transmits to node 1 using a deterministic packet arrival distribution with $\Lambda = 1.0$ packets/s and the fixed layer payload size 100 bytes. The PCI added by the protocol stack is 23 bytes and the air frame length becomes

$$t_{dt} = t_{on} + t_{pre} + t_{som} + t_{par} + 123 \cdot 8 / (16025) = 69.89 \cdot [\text{msec}]$$

We use $(a_p, b_p) = (6,100)$, one priority level only (p = 0) and ARQ is disabled. The power level is set sufficiently high to have an error-free radio channel.
The system operates at a low load level and only the MAC service time contributes to the average end-to-end delay, given by $t_v(a_p + b_p/2) + t_{dt}$, which equals to 312.26 msec. The minimum end-to-end delay is given by $a_p t_v + t_{dt}$ and the maximum is $(a_p + b_p)t_v + t_{dt}$.

Table 4.2 compares simulated results and theoretical results, and the table show excellent conformity between theoretical and simulated results.

Variable		**Value**
End-to-end delay [msec]	simulated	312.23 $\pm$ 0.0003
	analytical	312.26
End-to-end delay (min, max) [msec]	simulated	(95.86, 528.66)
	analytical	(95.86, 528.66)

Table 4.2 Comparison table between simulated and analytical results with ARQ disabled. Estimations of first order moments are presented as 99% confidence intervals.

5 Link layer implementation

The following description of the link layer implementation is primarily based on the specification outlined in section 2. Significant deviations from [2] will be emphasized.

5.1 Node internals

This section briefly describes the most important modules and entities involved. The functionality is primarily handled in the *L2_MacLayer* module, but the *L3_Rrc* module and the L2_Llc modules also play important roles.

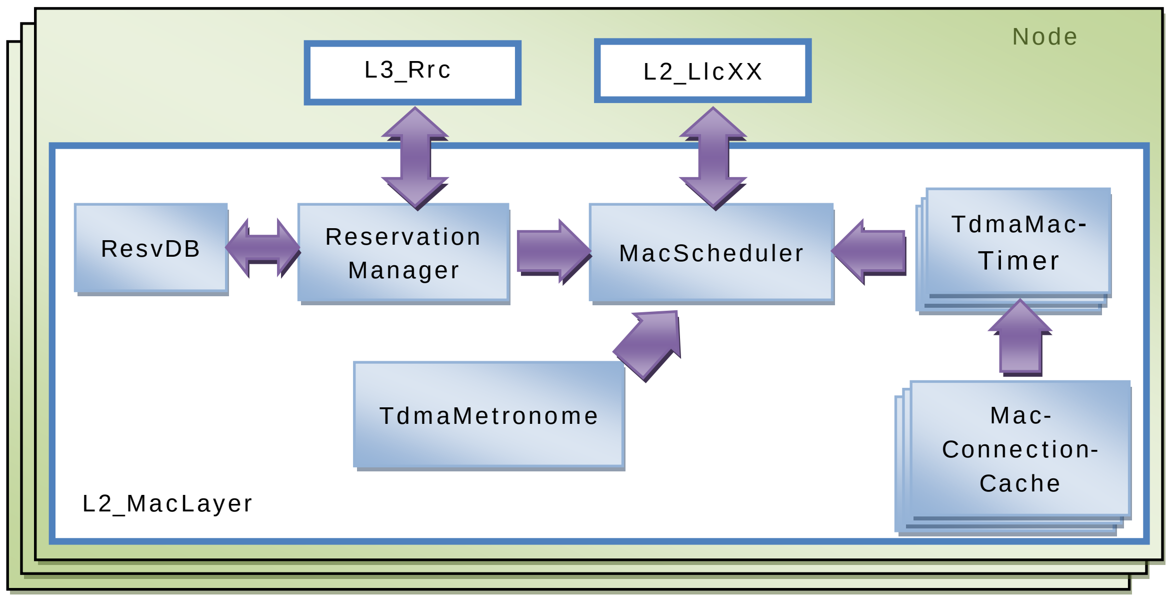


Figure 5.1 TDMA MAC modules overview. The arrows denote information flow.

5.1.1 MAC layer

5.1.1.1 TDMA metronome

The metronome generates the "heart beat" of the MAC module. Its main purpose is to notify the MAC Scheduler at the start of each time slot, using the globally defined TDMA frame layout. When activated by the simulator kernel, it calls the MAC scheduler's doScheduling() method. When this call returns, the metronome re-schedules itself to be activated at the start of the next time slot.

In order to save precious CPU cycles during a simulation, the metronome can be set to a sleep mode. The MAC Scheduler's doScheduling() method's return value indicates whether this node has any pending tasks (transmissions) or not. If none tasks is present, the metronome can enter a sleep state. In this state, the MAC scheduler will not be activated. To resume to the metronome's normal operation, its wakeup() method must be called. This is typically done when the LLC layer signals data availability, or when signalling messages (CR/CC) should be sent. When woken up, the metronome must perform a “fast forward” operation in order to calculate the current position in the TDMA frame structure.

The metronome also serves as an information base for the MAC module. It both provides querying functions describing the TDMA frame layout, as well as information about the current and future time slots. If the metronome is in sleep state when some of these are called, it may need to perform the fast forward operation (but it doesn’t need to wake up).

5.1.1.2 Reservation cache

The reservation cache was initially described as “connection cache” in section 4.1.1 of [2]. The basic semantics of this data structure is still the same, but additional state information has been added, and the connection cache objects now serve as *protocol elements* with state machines, rather than pure data storage. Figure 5.2 illustrates the class hierarchy of the reservation cache objects.

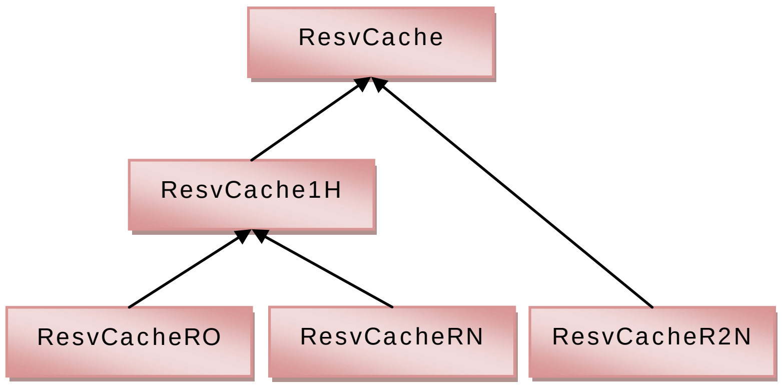


Figure 5.2 Reservation Cache class hierarchy

At the top level, the ResvCache super class contains the basic information related to a reservation, which is used to realize the reservation data base. The presence of a ResvCache object in a node, implies that the node is included in the related reservation domain. Each of the leaf classes implement one of the three roles a node may have in the reservation domain, namely RO, RN or R2N. Since the RN and RO share much of the functionality related to the reservation protocol (CC/DC schedule) and related to the internal signalling with LLC, the intermediate super class ResvCache1H (1-hop) was included. Each ResvCache1H object is assigned a local connection identifier, which is used for communication with the LLC layer. The objects are created during the connection setup phase, and are destroyed when the connections are disconnected.

5.1.1.3 Reservation manager and reservation database

The reservation manager handles all operations on the reservation database (ResvDB). It provides functions to query and update the database, and also knows about ongoing reservation procedures, preventing the MAC Scheduler from sending before the end of a CC-schedule. It is primarily used by the MAC Scheduler at the start of each time slot, in order to find the reservation status of the current slot. It is also used before sending a CR, to find out which time slots are available for reservation. In the current implementation, the ResvDB and the reservation cache are two separate data structures containing redundant information. It would be favourable to only have the reservation manager using the reservation cache structure, to avoid the extra overhead of maintaining the ResvDB, but this is left to future versions of the simulator. The ResvDB contains one record for each timeslot. Each record is on the following format:

```
class SlotReservation
{
   (...)
   int slotnumber;
   ResStatusType reservationStatus; // RESERVED, RESERVING, FREE
   MacGlobalConnectionId reservedByGcid;
   bool mergedWithPrevious;
};
```

This implementation supports reservations for one or several slots (merged), but it does not support multiple reservations of the same time slot, which is a likely demand in networks with many hops. If instead using the reservation cache, where each instance of the ResvCache class would represent a record in a virtual data base as described on section 2.2.1, this will not be an issue.

5.1.1.4 TDMA MAC timers

The MAC module contains a timer designed to schedule future events exactly at the beginning of a time slot. This is useful when a packet, buffered at MAC (typically a CR or CC), is scheduled for transmission a few time slots ahead, or when a connection needs a timeout in case expected packets did not arrive on time (e.g. CCs). Each TDMA MAC timer object has a reference to its owner. The owner's handler function is called when the timer expires. The current granularity of this timer is one mini slot (one half of a time slot).

5.1.1.5 MAC scheduler

As seen in Figure 5.1, the MAC scheduler is a central object in the MAC module. Its primary task is to select packets for transmission in the current slot. Additionally, it manages the TDMA MAC timers. The scheduler is activated by the metronome at the beginning of each time slot. It then checks and possibly activates the timers, before running the actual scheduling algorithm in order to decide if a transmission should be initiated in the current slot. After this, the scheduler indicates in its return value whether the metronome should go to sleep or not. Sleep mode is only entered if there are no buffered packets at LLC or MAC, and there are no pending TDMA MAC timers.

The MAC scheduling algorithm executed at the start of each time slot is outlined in Figure 5.3. It starts by checking if any timers should go off at this time. If true, they are executed by calling their handlers. The handlers may or may not decide to transmit a packet (typically from PSCCH). If PHY is currently sending (e.g. if this slot is merged with previous or any of the timer handlers decided to initiate a transmission), the scheduler terminates. Otherwise, the next actions depend on whether the slot is reserved or not. If the slot is reserved by this node, and the connection is active (all CC's have arrived), the scheduler issues a SDU request to LLC. If the connection was not yet activated, the CR may be retransmitted in this slot (only data connections). If the slot is not reserved, the scheduler initiates a contention attempt by sending an SDU request to one of the Random Access (RA) SAP's unless a CR could be retransmitted also here. Note that retransmissions of MV CR are not sent, because it would complicate the relay setup procedure even further.

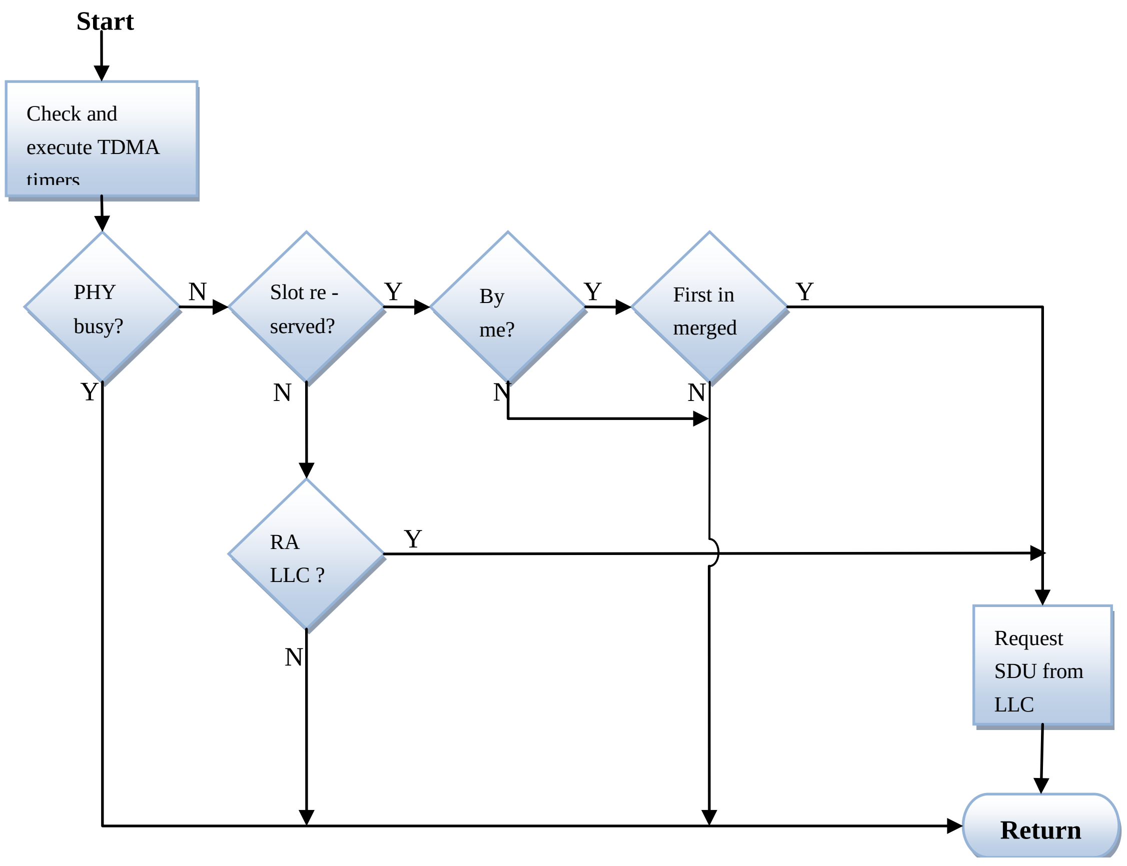


Figure 5.3   MAC Scheduler operation overview

5.1.1.6   MAC state machine

The MAC module is operated by the state machine illustrated in Figure 5.4. In addition to keeping track of the interactions with the upper and lower layers, it instructs the metronome to enter sleep mode when no actions are expected in near future.

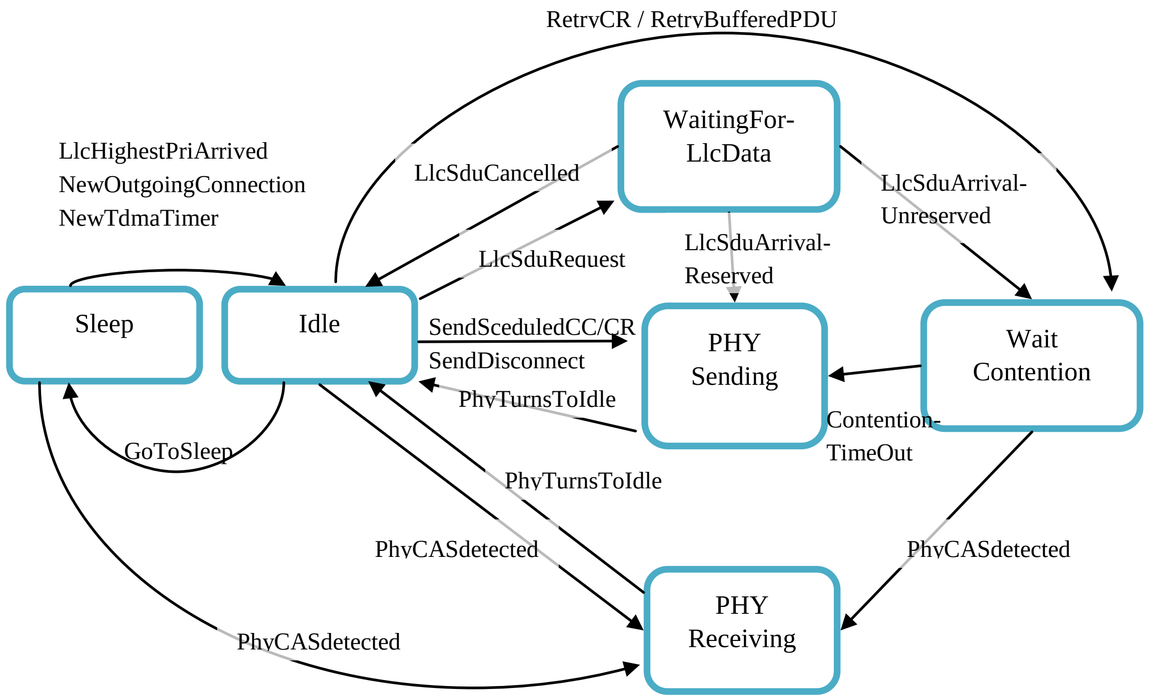


Figure 5.4 MAC state machine

5.1.2 LLC and interface to MAC

5.1.2.1 Overview

The LLC layer contains three modules: *L2_LlcCplane* handles random access control traffic, *L2_LlcUplaneVoice* handles user data from the MV application and *L2_LlcUplaneData* handles user data from data applications. Figure 5.5 shows an overview of the interfaces between the different LLC modules and the MAC module. For all interfaces except PSCCH, the MAC scheduler needs to issue an SDU request to the appropriate LLC module. However, because the scheduler makes its decision before sending the request, it needs to know the *data availability status* of the different modules. In the following, the functionality of three LLC modules are described along with a description of the inter layer signalling protocol.

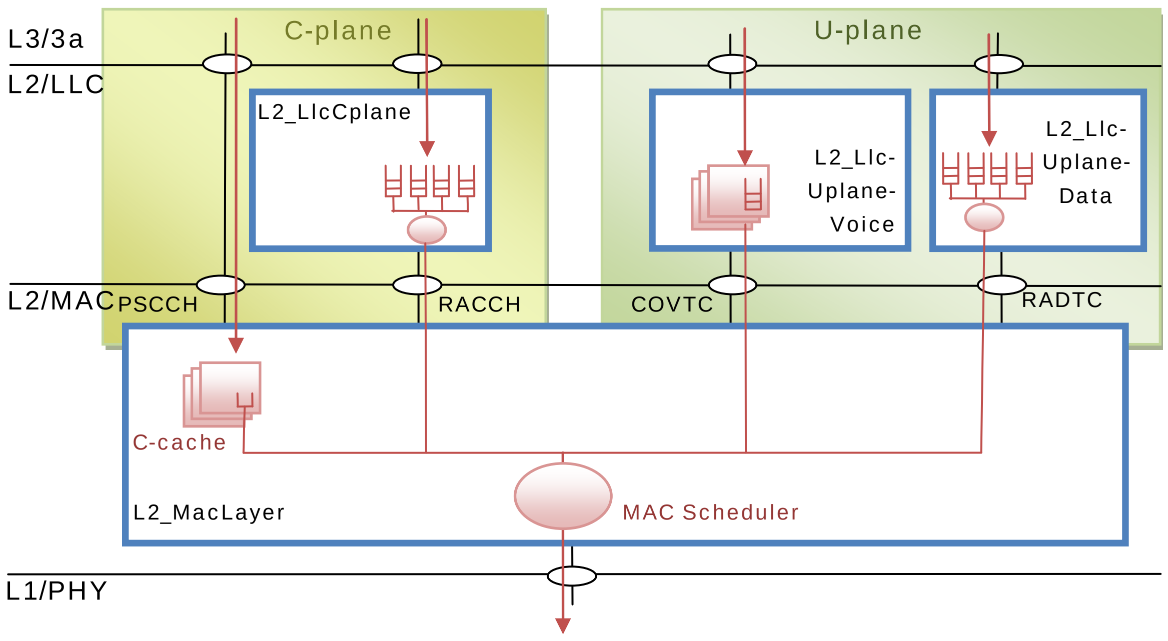


Figure 5.5 LLC MAC interface overview (outgoing packets)

5.1.2.2 L2_LlcCplane

LLC C-plane implements a LLC UM (Unacknowledged Mode) protocol. The only PCI carried is the lifetime of the SDU passed as ICI from upper layers. The module supports priority handling and lifetime control. Outgoing packets are queued in a strict priority queue, consisting of four FIFO queues and a strict priority scheduler. SDUs with expired lifetime are dropped at the front of the queue (i.e. after queuing). Incoming packets are decapsulated and sent upwards to the RRC without queuing.

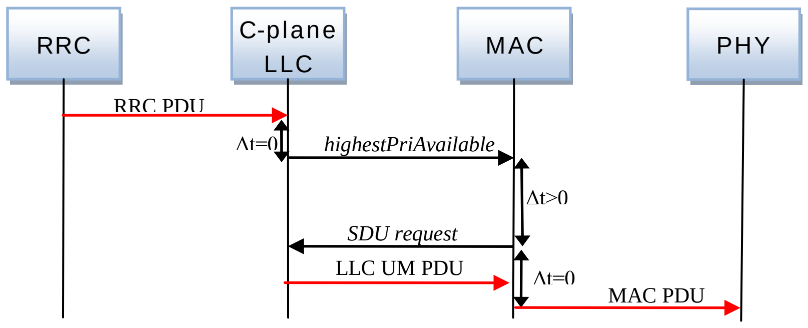


Figure 5.6 RACCH interface signalling

The signalling interface over the RACCH SAP relies on the LLC module to signal its *data availability status* to the MAC layer, using the *highestPriorityAvailable* value. This value is the priority of the highest priority packet in the queue, or -1 if the queue is empty. The value is passed to the MAC scheduler whenever the value changes, using the *highestPriAvailable* signalling message. This may happen when an SDU arrives from upper layer, after a PDU has been sent to MAC, or when lifetime control drops packets. When the scheduler eventually decides that a packet from this module can be sent, MAC issues an *SDU request*. The LLC module will immediately respond by sending the packet with the highest priority, possibly followed by a highestPriAvailable signal.

5.1.2.3 L2_LlcUplaneVoice

This module handles packets from established MV connections, for which the local terminal is either a sender or a receiver, or if this node is a relay. During the connection setup procedure handled in the control plane, this module is instructed (by the RRC) to create a connection object for the new connection, identified by the local connection id. Similar to the MAC reservation cache, the objects can be of type incoming or outgoing. Outgoing objects contain a single FIFO queue. The LLC U-plane Voice module operates in Transparent Mode (TM), meaning that no PCI is added before outgoing packets are handed to the MAC layer.

If a node operates as relay for a connection (MV only), both an incoming and outgoing connection cache is created, and a relay association is established. Later, when packets arrive on the incoming connection, they are placed in the outgoing connection cache FIFO queue.

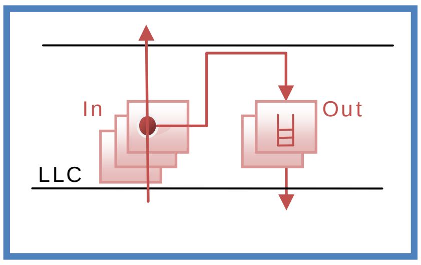


Figure 5.7 LLC u-plane voice buffering

The outgoing interface to the MAC layer is similar to the random access interfaces, based on an SDU request from MAC. However, in the current implementation, the MAC scheduler assumes that an established connection always has a packet available, so there is no need to signal *data availability status*. It may however happen, even for MV, that the LLC buffer is empty when receiving a SDU request from MAC.

If MAC has nothing to send in a reserved slot, it sends *a keep-alive* message to prevent neighbours to assume that the reservation is disconnected. This message contains the global connection id, and an empty SDU.

5.1.2.4 L2_LlcUplaneData

This module implements the LLC data service as described in section 2.3.6. An SDU can either be transmitted over a MAC connection over the CODTC SAP, or it can be transmitted using the random access contention scheme over the RADTC SAP. The module also provides segmentation and reassembly as well as ARQ functionality. At the time of writing, this functionality is under development and will not be described in detail here.

5.1.3 RRC layer and interface to link layer

5.1.3.1 Overview

The RRC (Radio Resource Control) module together with MAC implements the reservation protocol. In the current simulator implementation however, most of this functionality is handled in the MAC module, while RRC is left with controlling the LLC U-plane Voice module as well as being the point of contact between L7 and MAC. The U-plane LLC is instructed by RRC to create or delete connection objects for both outgoing and incoming connections when necessary.

The choice to let MAC handle the reservation protocol was made to make the interface between MAC and RRC simple. As the RRC module should control the “radio resources”, which in this case are the TDMA time slots, the RRC should have knowledge about the TDMA frame structure as well as the current position in the structure. Also, it would need access to the reservation data base. Much of the functionality now located in the MAC protocol elements would then be moved to the RRC module. However, even if this functionality was moved to RRC, MAC would still need access to these data structures, and the interface would get more complex. Another implication is that the header fields associated with the reservation protocol should then be carried by the RRC header instead of the MAC header. In future implementations, this alternative organization should be seriously considered, in order to simplify the MAC module and for better compliance with the reference model.

5.1.3.2 RRC-MAC interface

As indicated by the NBWF reference model (Figure 2.5), RRC may send PDUs in two different ways. Either through the LLC C-plane module, which queues the packets and sends over the RACCH SAP (contention based). This is typically used when sending CR for data reservations. Alternatively, RRC may send PDUs directly to the MAC module over the PSCCH SAP, with information about exact sending time appended as ICI. This is used for MV CRs and CCs, which should be transmitted in dedicated slots according to the reservation protocol. Note that in the current implementation, this ICI information is omitted, as all state information related to the reservation protocol is kept in the MAC reservation cache objects.

5.1.3.3 RRC-LLC U-plane interface

When L7 wants to initiate a connection, it sends a connect request to the RRC module. RRC instructs LLC U-plane Voice module to create a new connection object for the new connection, identified by a local connection id. The connect request is passed on to the MAC module, which makes the decision based on the current reservation database. If the requested resources were unavailable, a disconnect message is returned to the RRC, which instructs the LLC module to delete the connection objects, before notifying L7 about the failure.

5.2 TDMA implementation

As already described, the metronome and scheduler together forms the basis of the TDMA operation. The metronome wakes up the scheduler at the beginning of each time slot, which then decides if the node should transmit a packet in the current slot.

5.2.1 Frame structure

The frame structure is implemented as described in section 2.1.

5.2.2 Reservation DB

The reservation DB is implemented as described in section 2.2.1

5.2.3 Contention

A simple contention scheme is currently implemented, intended to be replaced by a more sophisticated scheme in the near future.

5.3 Reservation protocol implementation

This section gives a more detailed description of how the reservation protocol is actually implemented in the simulator. The implementation mostly follows the specification outlined in section 2, with some simplifications. The following explains how a voice-mode MAC-connection is requested (COVTC), but most of the behaviour is also valid for data-mode reservations (CODTC).

5.3.1 Reservation cache state machines

Figure 5.8 and Figure 5.9 shows the state machines of the RN and RO protocol elements respectively.

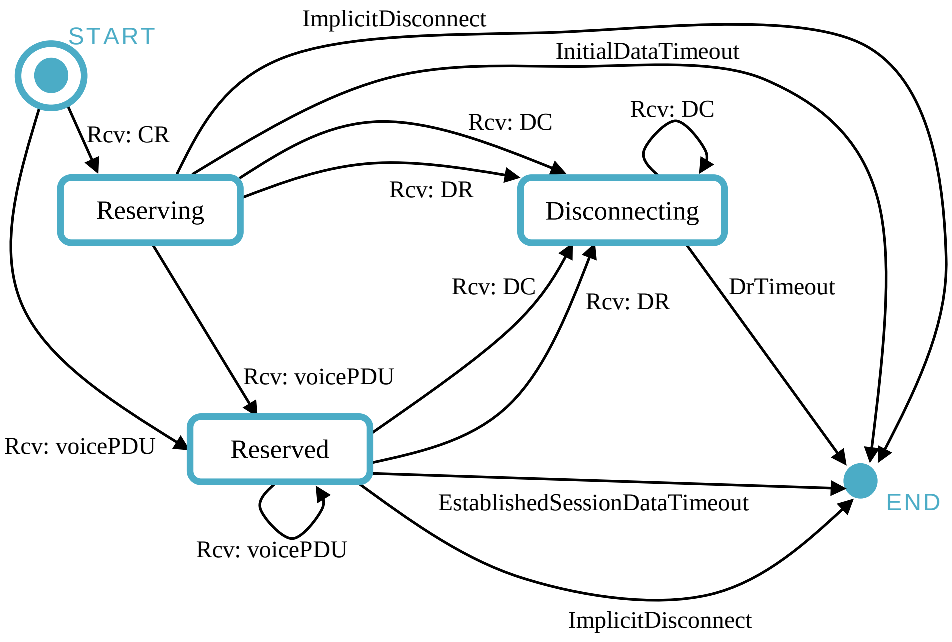


Figure 5.8 ResvCacheRN (Reservation 1-hop neighbor)

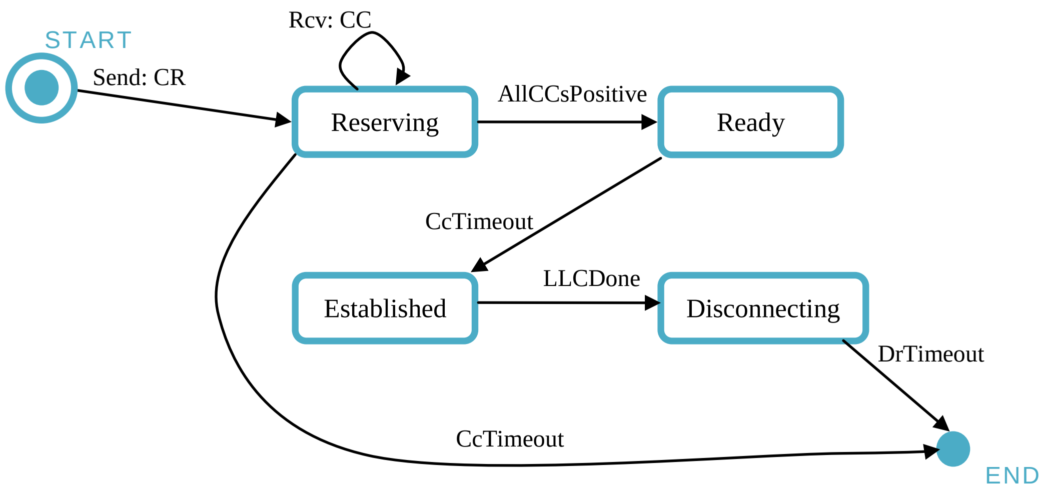


Figure 5.9 ResvCacheRO (Reservation Originator)

5.3.2 Connection ID

Connections are identified using RID and ECID as described in Table 2.3 and Table 2.4. All signalling messages and MV PDUs contain the RID (which includes the ECID) in their PCI. This facilitates simple and robust connection cache lookup when messages arrive.

5.3.3 Connection setup

The relay setup procedure as described in section 2.4.4 is implemented, using the slot organization illustrated in Figure 2.2. Two dedicated relays are assumed, so the TDMA frame starts with three MVS slots.

5.3.3.1 Initiator (PTT-node)

Normal operation

Figure 5.10 shows how connection setup is handled in the initiator node, when no irregularities occur. When receiving a CR from L7, RRC creates the CC-set and relay-set before passing the CR on to the MAC module. It also instructs LLC U-plane to create an outgoing connection object. MAC schedules the CR in the correct slot (the first slot in the frame according to the current protocol specification) and registers the incoming CC's. When the reservation period is over, the connection enters ESTABLISHED state, and the MAC scheduler will begin to issue SDU requests to LLC at the start of each occurrence of the reserved slot (every 202.5 ms).

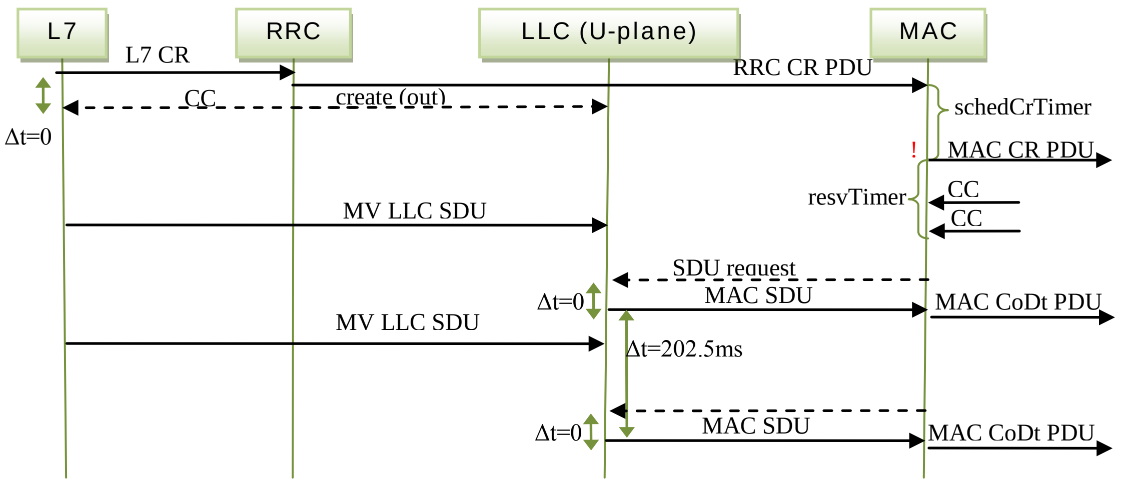


Figure 5.10 Connection setup in the initiator node. Normal operation. Red exclamation mark denotes a timeout.

Exceptions

Seen from the initiator, two exceptions may occur. In the first situation, the requested time slots may be occupied at the CR creation time as described in Figure 5.11. In this case, the CR will not be sent. Instead, a local DR is issued to the RRC, notifying that the resources are currently unavailable. No retransmission attempts will be done by the MAC layer. The RRC will in turn notify L7 and instruct LLC to delete the connection cache along with all queued data.
The second situation happens when the CC-conditions are not met, either if not enough positive CCs have arrived, or if a negative CC has arrived. This is illustrated in Figure 5.12. The exception handling is identical to the first case, only now triggered by the resvTimer.

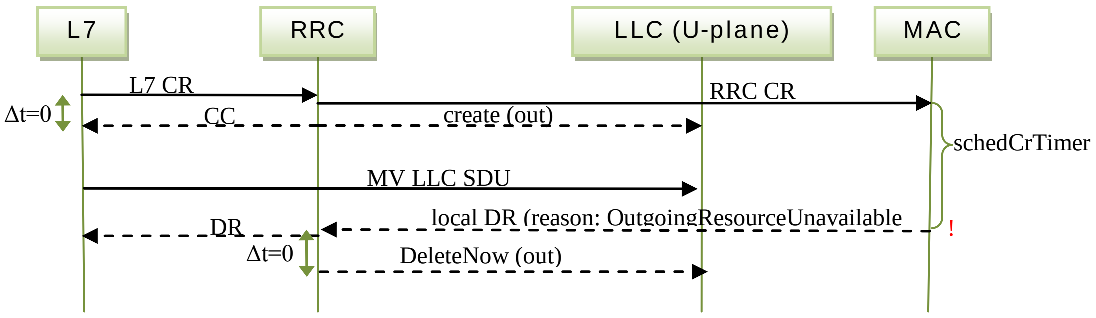


Figure 5.11 Connection setup in the initiator node. Operation when the slots are occupied. Red exclamation mark denotes a timeout.

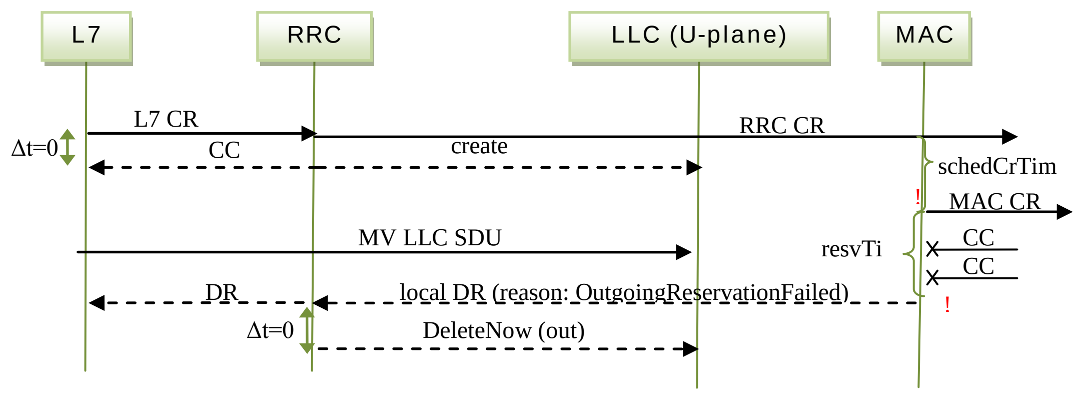


Figure 5.12 Connection setup in the initiator node. Operation when the CC conditions are not met. Red exclamation mark denotes a timeout.

5.3.3.2 1-hop neighbour

1-hop neighbours refer to all receivers of a CR, i.e., 1-hop away from the originator. Technically, this also includes relay nodes, but the relay operation is described separately in section 5.3.3.3.

Normal operation

Figure 5.13 and Figure 5.14 describe the setup operation in CC-nodes and other 1-hop neighbours respectively. When MAC receives a CR from a peer, it creates a new reservation cache for the incoming connection before passing the CR along to the RRC module. If this node was included in the CC-set, RRC will immediately send a CC down to MAC, indicating whether the new connection is accepted or not. If accepted, and if this node is a member of the destination multicast group, the CR is passed on to L7 and the LLC is instructed to create an incoming connection object. MAC stores the CC for later transmission, determined by this node's position in the CC-set and the CC-schedule algorithm (see section 5.3.5). When the first CoDt PDU arrives, the connection is moved to the RESERVED state (see Figure 5.8), before the payload is passed to LLC.

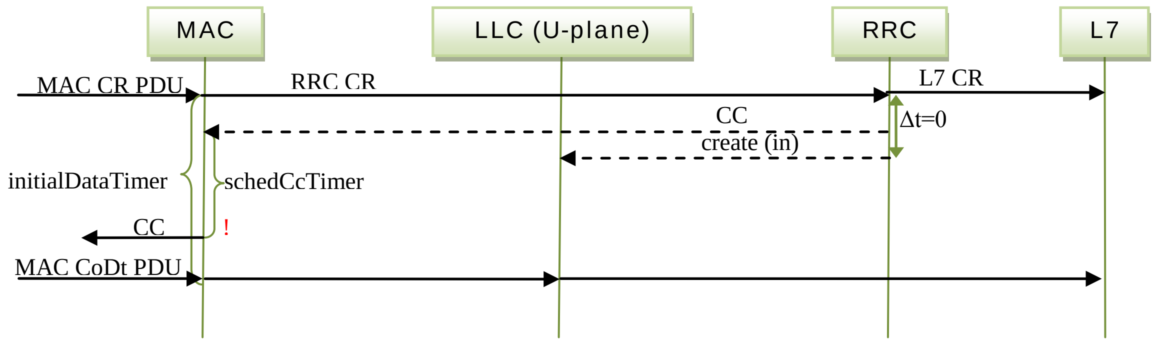


Figure 5.13 Connection setup in CC-nodes (not relays). Normal operation. Red exclamation mark denotes a timeout.

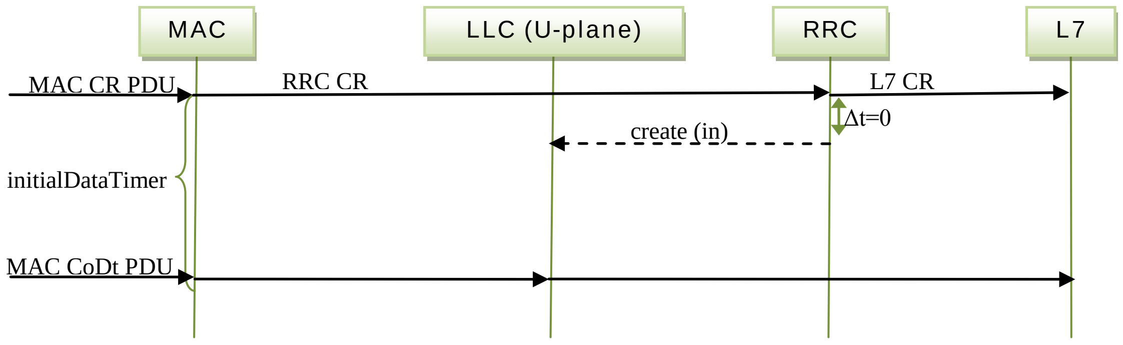


Figure 5.14 Connection setup in a 1-hop neighbor node (not CC-node). Normal operation.

Exceptions

As illustrated in Figure 5.15, if no data packets have arrived on a connection before the initialDataTimer goes off, the connection is discarded and a local DR is sent to RRC stating that the incoming connection failed. This situation occurs if the originator decided not to continue with the reservation due to missing/negative CCs. In the simulator, a passive disconnect mechanism is used in this case. This means relying on the initialDataTimer instead of sending a DR. Another situation where the timer will fire is if the data packets where corrupted during transmission (i.e. the MAC header checksum fails or PHY fails to detect the preamble due to noise.)

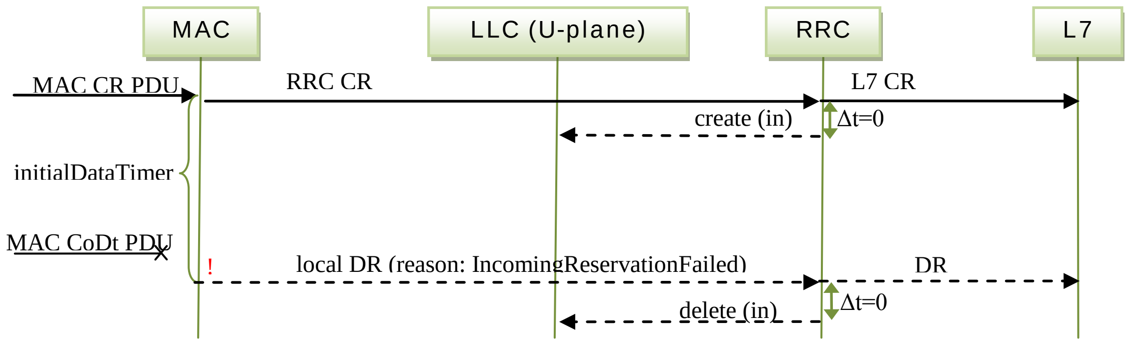


Figure 5.15 Connection setup in a 1-hop neighbor node (including CC-nodes). Operation when no data arrives before the initialDataTimer goes off. Red exclamation mark denotes a timeout.

5.3.3.3 Relay

Normal operation

When the RRC module receives a CR from MAC, and finds this node's address in the relay-set, it sets up this node as a relay[1]. This primarily involves instructing the LLC module to create both an incoming and an outgoing connection object, as well as establishing a relay-association between them. The RRC also creates a new CR in order to reserve the time slots for the next hop,

[1] Dedicated relays are used, but it is still the originator that decides which relays to use when building the CC-set. In fact, the relay-nodes need not to be aware that they are chosen as dedicated relays, as they will obey the instructions in the CR.This simplifies the transmission towards automatic relaying.

which is sent to the MAC module together with the CC for the previous hop. When it is time to send the CR, MAC "attaches" the CC on to the CR, in order to send both messages in one transmission (in 1 time slot). Later, when MV PDUs arrive, the LLC layer takes care of the relaying.
To summarize, the relay procedure is almost identical to the sum of the initiator and the CC-node procedures, but with some exceptions:

- The outgoing CR is triggered by an incoming CR instead of a CR from L7
- The relay association at LLC is now established.
- The CC for the incoming connection is piggybacked on the CR for the outgoing connection.

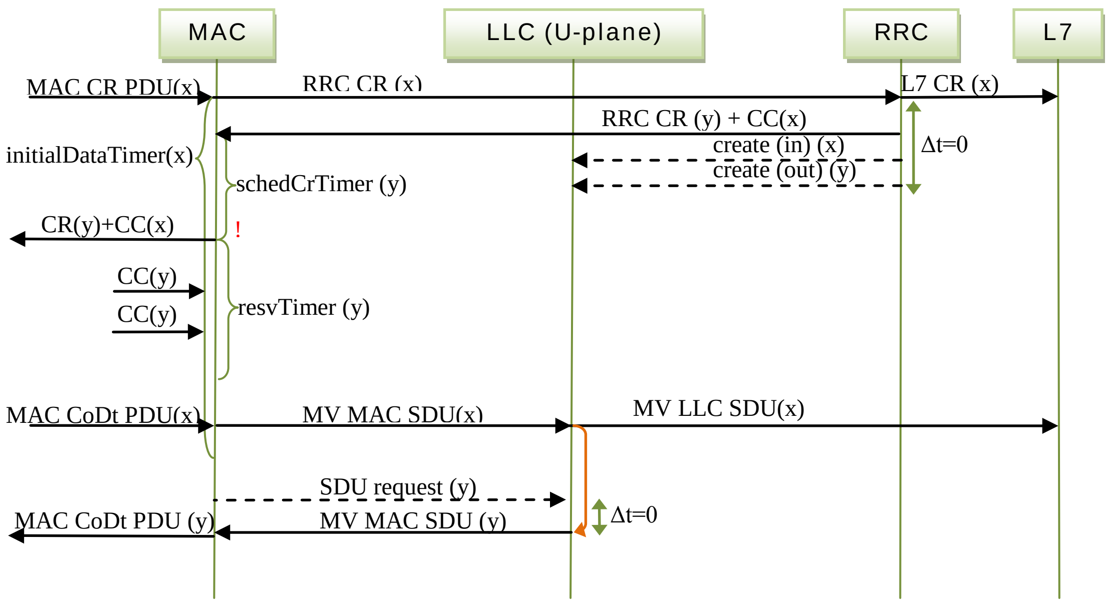


Figure 5.16 Connection setup in a relay node. Normal operation. The incoming connection (previous hop) has RID=x, while the relayed outgoing connection has RID=y. Red exclamation mark denotes a timeout.

Exceptions

Exceptions for the individual connections are handled as described above, and will not be repeated here. However, failure on one of the two connections may or may not affect the other: If the incoming connection fails, RRC will terminate the outgoing connection, and MAC will send a DR. If the outgoing connection fails however, the incoming connection is not affected, because there may be other receivers within range of the previous hop originator.

5.3.3.4 2-hop neighbours

Two hop neighbours are only able to hear CCs and DCs sent by the CC-nodes. In the current simulator implementation, MAC does not need to inform RRC about reservations two hops away, since the reservation database is located in the MAC module. When receiving a CC with a RID not seen earlier, MAC updates the reservation database, and starts a timer associated with the given slot (slot number is included in the CC PCI). The slot is released when the DC arrives, or when the timer goes off. The length of the timer interval should be long enough to cover a relatively long MV session. This is a compile time parameter, currently set to 20s.

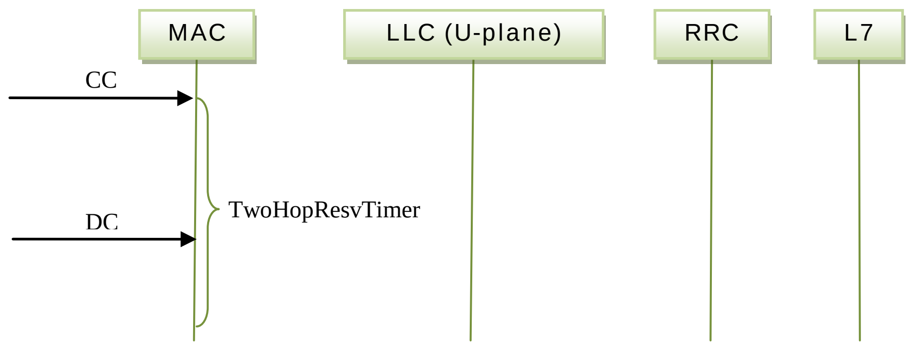


Figure 5.17 Operation in a 2-hop neighbour. Normal case

5.3.4 Connection release

The connection release procedure is implemented as described in section 2.4.3 with some modifications. The DR and DCs are transmitted in the reserved slots only. Thus, if more than one DC is requested, the next frame must be used.

5.3.4.1 Initiator (PTT-node)

Figure 5.18 describes the connection release procedure in the initiator node. After L7 has delivered its last MV PDU to LLC, it sends a DR to RRC. RRC then instructs LLC to terminate the outgoing connection if/when the buffer is empty, and it also sends a DR to MAC. MAC does not transmit the DR right away, but continues the data transmission operation until receiving the LLC done primitive from LLC. The DR is then sent in the next occurrence of the reserved slot, and the DrTimer is set in order to keep the slot marked as reserved until all DCs have been sent. Note that the originator ignores the incoming DCs. Their only function is to notify two-hop neighbours. Consequently, the initiator expects no incoming messages during the disconnect phase, and no exception handling is required.

5.3.4.2 1-hop neighbour

Normal operation

Figure 5.19 describes the disconnect procedure in a CC-node. The operation in other one-hop neighbours are identical except from the DC part (coloured blue in the figure). When receiving a DR from the originator, MAC immediately sets the DrTimer before passing the DR on to RRC. The DrTimer has the same purpose as in the initiator node; mark the reserved slots as free after the last DC should have been transmitted. If this node is in the CC-set (received earlier, in the CR), the position in the CC-set determines in which slot to transmit the DC. DCs are only sent in the reserved slot.

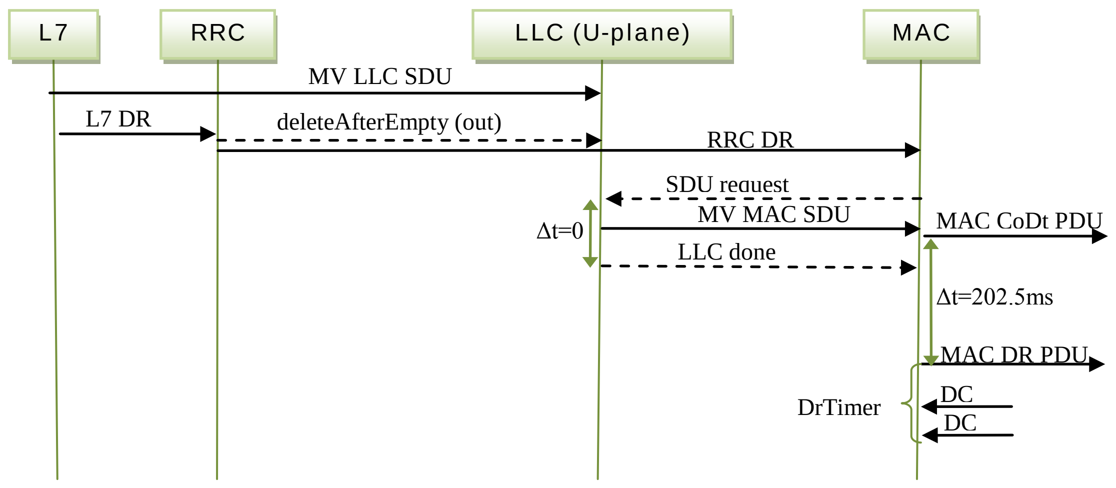


Figure 5.18 Connection release in initiator node. Normal operation.

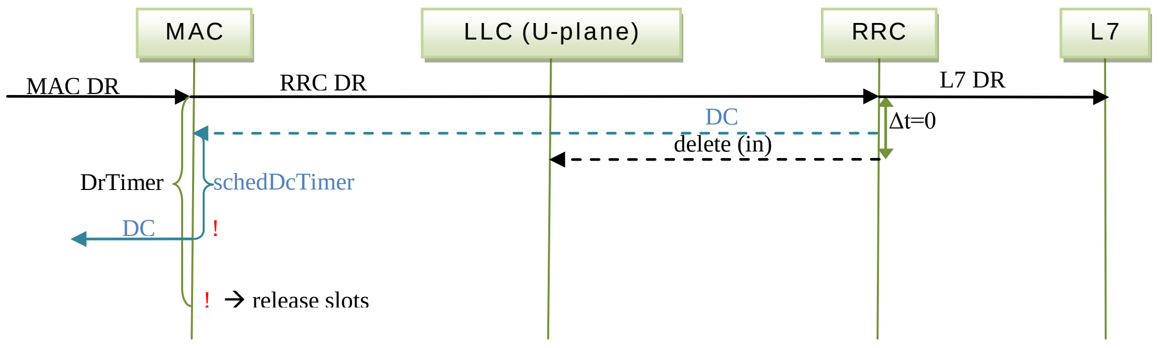


Figure 5.19 Connection release in a CC-node. Normal operation. The operation in other 1-hop neighbours is identical, accept from the DC part. Red exclamation mark denotes a timeout.

Exceptions

In order to handle the case where the DR is lost, the one-hop neighbours resets the SessionEstablishedTimer on each CoDtPDU arrival. If the timer goes off, the incoming connection times out and is considered disconnected. A local DR is sent to RRC, which will instruct the LLC to delete the incoming connection object.

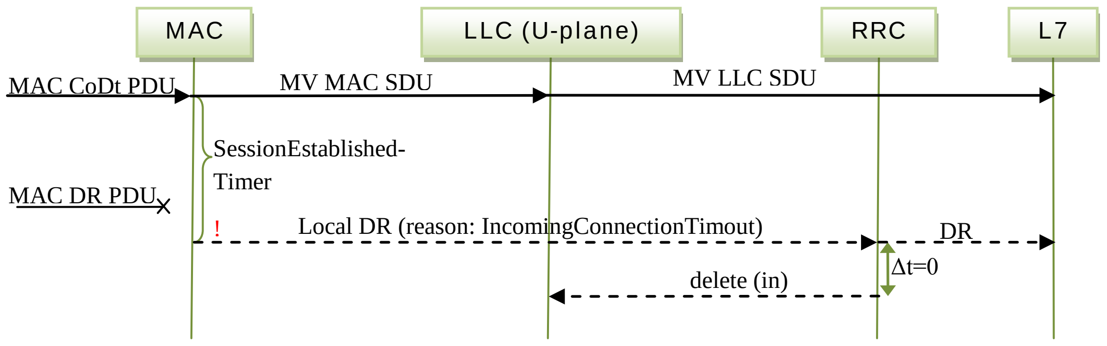


Figure 5.20 Connection release in a 1-hop neighbour node. Operation when no MV PDU or DR PDU arrives before the SessionEstablishedTimer goes off. Red exclamation mark denotes a timeout.

5.3.4.3 Relay

As opposed to the setup procedure, no optimization is done in order to speed up the release phase of a relayed connection. Each connection's disconnect procedure is performed separately, and no piggybacking is used. Figure 5.21 illustrates the disconnect procedure in a relay node. When RRC receives a DR from MAC, it remembers that this is the incoming part of a relayed connection, and initiates a disconnect procedure on the outgoing part. From here on, the two reservation procedures is performed separately, as described for CC-nodes (Figure 5.19) and initiators (Figure 5.18). Consequently, the only exception that needs to be handled is that the sessionEstablishedTimer fires because of corrupted CoDtPDUs or DR. As in the setup procedure, if the incoming connection fails, the outgoing connection will be terminated.

5.3.4.4 2-hop neighbour

For two hop neighbours, the reservation is released either when receiving a DR, or when the TwoHopReservationTimer goes off.

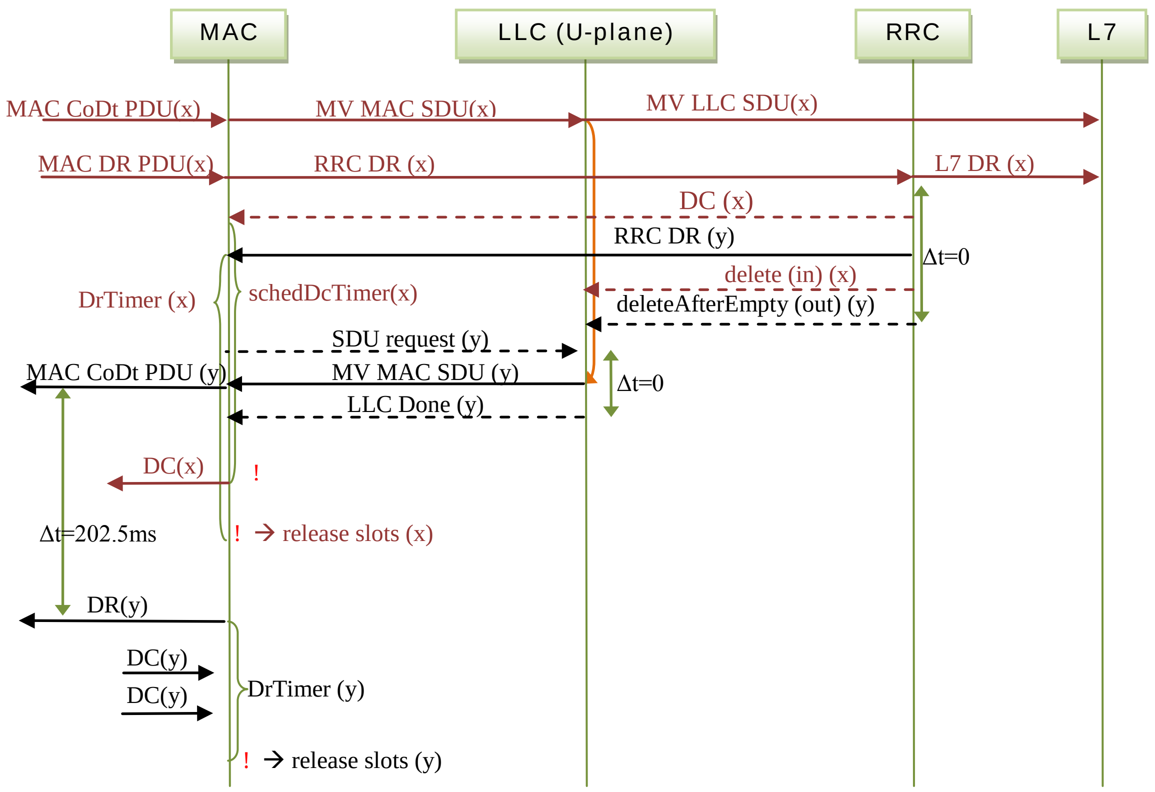


Figure 5.21 Connection release in a relay node. Normal case, combining the operation of an initiator and a 1-hop neighbour. Unlike the initiator, where the disconnect of the outgoing connection is triggered by DR from L7, the disconnect is now triggered when the previous hop connection is disconnected. The incoming connection (previous hop) has RID=x, while the relayed outgoing connection has RID=y. Red exclamation mark denotes a timeout.

5.3.5 CC selection algorithm

The RCC module is responsible for building the CC-set, and passing it to MAC as ICI along with the CR. As described in section 2.4, the CC-nodes should be selected so that all two hop neighbours in the reservation domain are covered with at least one CC. The algorithm implemented in the current simulator works as follows:

```
N₁ : set containing all one-hop neighbours of this node. Assumed populated
N₂ : set containing all two-hop neighbours of this node. Assumed populated
R  : set of dedicated relay nodes (MV only), assumed populated.
D  : set of all destination nodes (multicast members or unicast destination)
X  : set of all 1-hop neighbours through whom this node may reach at
     least one two-hop neighbour. (CC candidates)
Cₓ : set containing neighbours of node x, x∈X, which are also two-hop
     neighbours of this node.


//identify CC-node candidates (populate X and Cₓ)
for each node i∈N₁ {
    for each node j∈N₂ {
      if(isNeighbours(i,j) && (i∈D || i∈R) ) {
       <add i to X>
       <add node j to Cᵢ>
      }
    }
}

//eliminate redundant CC-nodes:
for each node a∈X {
    for each node b∈X, b≠a, b∉R {
       if(C_b – C_a  ≡ ∅) <remove b from X>
   }
}

//X now contains the unordered CC-set, including relays.
<order set based on importance,  relays always first>
```

5.3.6 Implicit disconnect

If a one-hop neighbour misses a DR, it eventually times out. The implicit disconnect feature is merely an optimization in the case where a new CR arrives from the same node owning the previous reservation. In this case, the receiver of this CR will locally disconnect the old reservation, trusting that the originator knows best. Likewise, if receiving a CoDt from the same node with a new RID (both the DR and the new CR was lost), an implicit disconnect is performed. A similar mechanism is used for two-hop neighbours.

5.3.7 Keep-alive

If the reservation period finishes quickly, there is a chance that L7 has not produced the first voice PDU when MAC is ready to send in the reserved slot. This is because L7 buffers voice samples in 202.5 ms before sending the first PDU to LLC. However, if not transmitting anything in the first slot, the receivers may time out (InitialDataTimer, described in 5.3.3.2). Likewise, if for some reason no SDU is available at the start of the reserved slot during the data transfer phase, the receivers may also time out (EstablishedSessionDataTimer). To prevent unwanted timeouts, the originator may choose to send keep-alive messages, in order to signal that the connection is still active. This is simply a MAC MV PDU with empty payload. Both the initiator (PTT-node) and relays may send keep-alive messages.

5.4 Data traffic implementation

5.4.1 LLC data protocol

The data protocol, with segmentation and ARQ functionality is currently under development and will not be described in detail here.

5.4.2 MAC contention access algorithm

A simple contention access algorithm is included in the simulator, and will be applied when transmitting random access control traffic (RACCH), and random access data traffic (RADTC). Currently, only CRs for data reservations are sent through the RACCH SAP. The algorithm does not differentiate between SDUs of different priority or type, as will the final algorithm. Another limitation with the implementation is that the contention window cannot span TDMA frame borders. Figure 5.22 shows the position of the contention window. When a node has a random access SDU to transmit, it draws a random start time within the window. If no preamble has been received during the waiting period, the SDU is transmitted. Note that TDMA slot borders are ignored within the window.

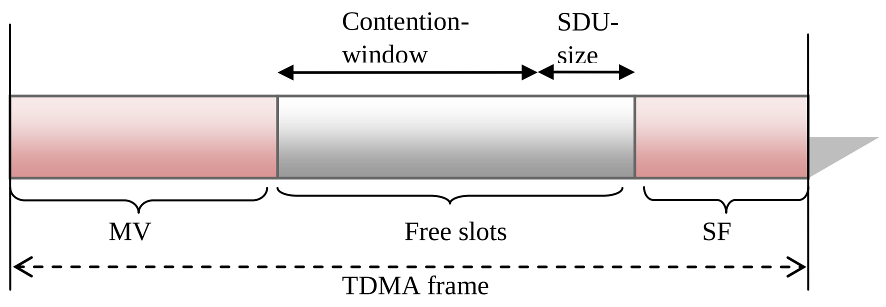


Figure 5.22 Contention window

6 Voice traffic generators

The multicast voice service is similar to the voice service provided by a legacy CNR-radio with the extension that voice traffic shall be relayed to destinations outside the radio range of the originator, if necessary. This section describes how human voice communications are modelled in the simulator. The terminology used is specified in the table below.

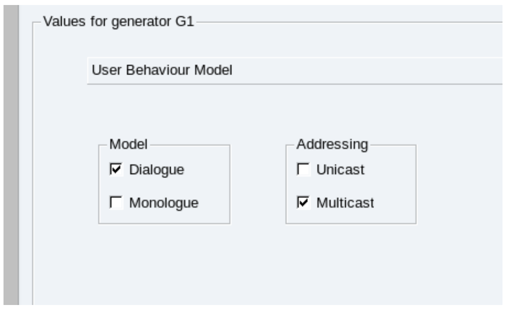
Term	**Description**
User Behaviour Model (UBM)	A user behaviour model specifies how two or more human users interact during a conversation.
Voice monologue	A one-way speech from one user to one or more other users served by the NBWF voice channel. A monologue encompasses one single PTT followed by a number of outgoing talk spurts. In contrast to a dialogue, the originator does not expect to receive any feedback from the recipient(s).
Voice dialogue	A two-way conversation between two users that can be considered to be a sequence of monologues. Only two users take part in the conversation while any number of other users may listen to the conversation. The latter is determined by the traffic pattern selected; unicast or multicast addressing.
Originator[2] A	The person who initiates the dialogue/monologue.
Responder B	The person with whom the originator A wants to talk.
Destination set Bset	The outgoing call shall be relayed to this node set. The *Bset* is included in the *connect.request* primitive sent to the local RRC entity and the set also includes the responder's address.
Multicast addressing	The *Bset* contains more than one element.
Unicast addressing	The *Bset* contains one element only.
***Call rate* [calls/hour]**	The number of calls per hour issued by the users. This is identical to the PTT rate of voice monologues and the rate of the *HelloB* for voice dialogues.

Table 6.1 Terminology

6.1 Implementation

The simulator has implemented the two user behaviour models *monologue* and *dialogue*, and provides the widgets in Figure 6.1 for configuration of traffic generators in the simulator.

[2] In the context of layer 7, the term *originator* is different from *reservation originator* at the MAC layer.

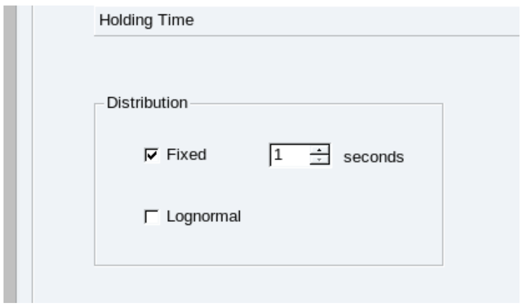

Values for generator G1

User Behaviour Model

Model: [✓ Dialogue] Monologue

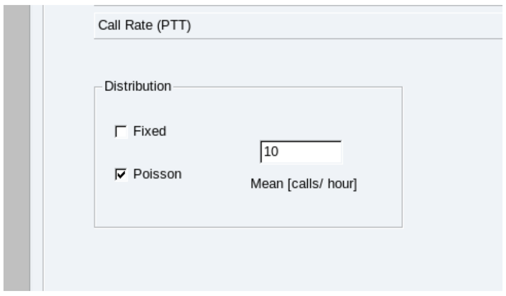
Addressing: Unicast [✓ Multicast]

User Behaviour Model


Holding Time

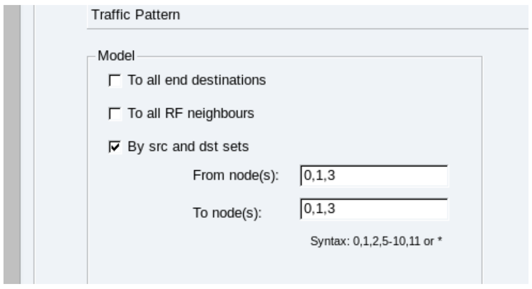
Distribution: [✓ Fixed] 1 seconds; Lognormal

Holding Time Distributions


Call Rate (PTT)

Distribution: Fixed [✓ Poisson]; Mean [calls/ hour]: 10

Call Rate Distributions


Traffic Pattern

Model: To all end destinations; To all RF neighbours; [✓ By src and dst sets]

From node(s): 0,1,3

To node(s): 0,1,3

Syntax: 0,1,2,5-10,11 or *

Traffic Pattern Model

Figure 6.1 The voice traffic generator's parameters provided by the simulator

The voice models are implemented by the two OMNeT++ modules named *L7_UserVoiceDialog* and *L7_VoiceProtocol*, see Figure 6.2. The main functionality of the first module is to implement a state machine which models a dialogue between two persons. The latter module generates the talk spurts as a MELPe traffic stream. Figure 6.3 illustrates how the L7_*VoiceProtocol* module establishes a connection by means of the RRC transfers the talk spurt and releases the connection. A more detailed description follows in the next section.

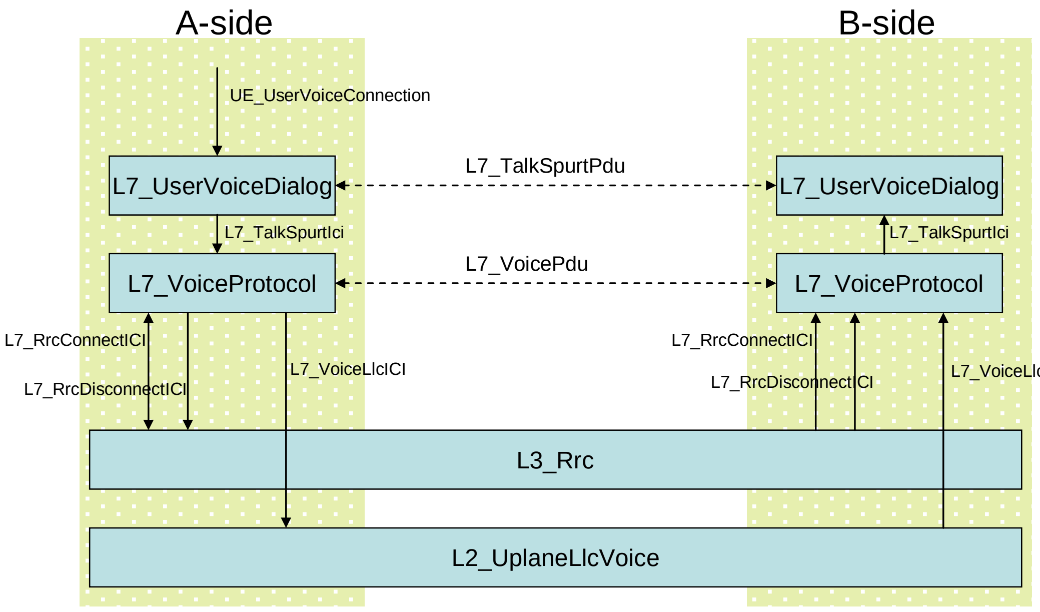


Figure 6.2   Overview of modules and data structures involved in voice handling

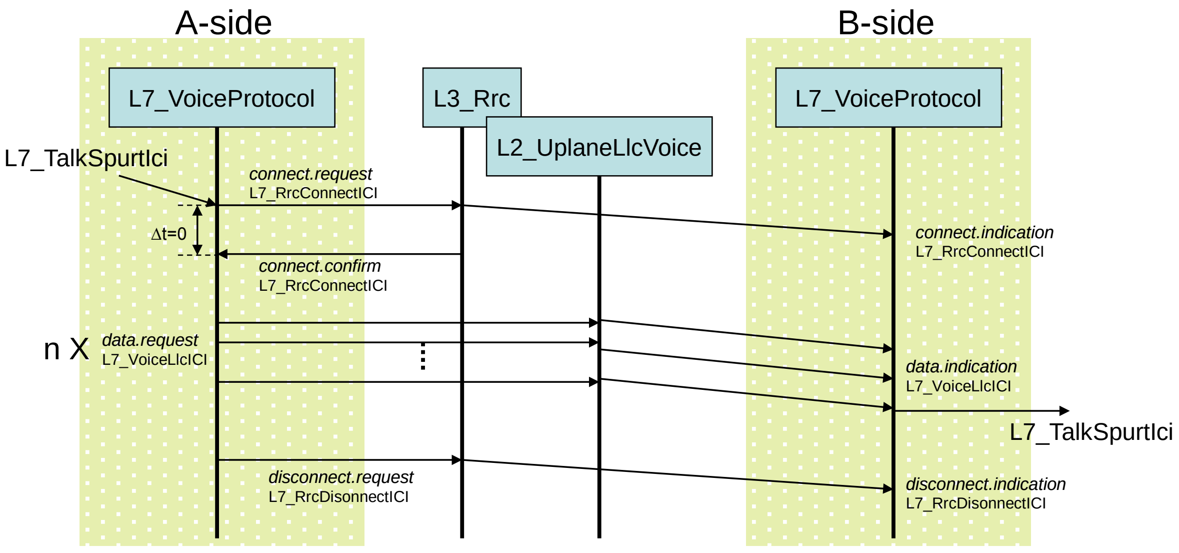


Figure 6.3   Time-sequence diagram for transferring a voice monologue

6.2 Voice monologue

The voice monologue (class *L7_UserVoiceMonolog*) is implemented mainly for debugging and testing. However, it may also become useful in simulation experiments. In contrast to the voice dialogue model, the originator in the voice monologue model does not expect a response from the remote side. The time-sequence diagram of a monologue is outlined in Figure 6.4. A new monologue event (PTT/HelloB) (class *UE_UserVoiceConnection*) is sent from the user environment module to the user voice dialogue module (class *L7_UserVoiceDialog*) where it is processed by the class *L7_UserVoiceMonolog*.

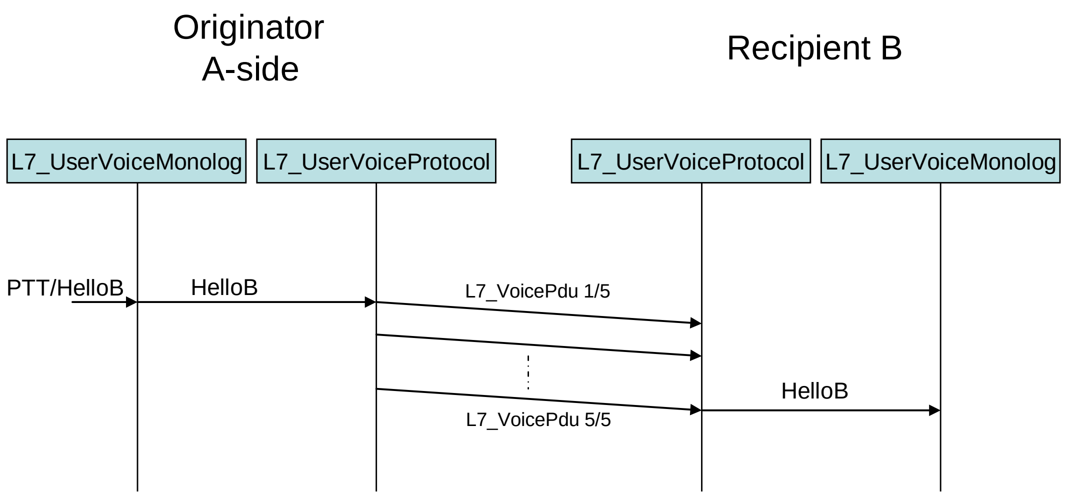


Figure 6.4 Example time-sequence diagram for serving a monologue

The ”HelloB” is carried by the message *L7_TalkSpurtPdu* (identical with the class *L7_TalkSpurtBase*) to the remote side where it is processed in the *L7_UserVoiceMonologue*.

The *L7_UserVoiceMonologue* can handle one and only one monologue at a time and is busy until the ”HelloB” has been sent. New voice calls (remote and local) are rejected until it becomes idle.

Connection setup and disconnection is handled by the *L7_UserVoiceProtocol*. The ”HelloB” is segmented at the sender side by the *L7_UserVoiceProtocol* and is reassembled at the receiving side by its peer entity. Each segment (class *L7_VoicePdu*) contains a sequence number field and a total number field (Figure 6.7). These fields are used by the receiving *L7_UserVoiceProtocol* to calculate the fraction of lost segments. This information, possibly with a partly reassembled talk spurt, is sent upwards to *L7_UserVoiceDialog* module where it is processed. The outcome of this process depends on the fraction of elements lost, which is the speech quality.

```
message UE_UserVoiceConnection // Local message UE_TrafficModule -> L7_UserVoiceDialog
{
fields:
  int    srcAddr           = Undefined;
  int    destAddr          = Undefined;  // This is the node that shall respond to
                                         // HelloB. destAddr is added to multicastSet
  int    multicastSet[];                 // if multicastSet.size() == 1 then unicast
                                         // else multicast;
  int    priority          = Undefined;
  double holdingTime       = Undefined;
  int    holdingTimeModel  = Undefined;  // Fixed, LogNormal,...
  bool   isDialog          = true;       // true >= Dialogue, false => Monologue
}
```

Figure 6.5 The message sent from the traffic generators to the L7_UserVoiceDialog module

```
message L7_TalkSpurtBase extends UE_UserVoiceConnection
// A base class used by the L7_UserVoiceDialog module

// TalkSpurtPdu for signalling L7_UserVoiceDialog <-> L7_UserVoiceDialog
// TalkSpurtIci for local message exchange L7_UserVoiceDialog <-> L7_VoiceProtocol.
// The end-destination needs to know the holdingTimeModel.
// The TalkSpurtPdu shall not increase the air frame size.

{
fields:
  int    talkSpurtType          = Undefined; // HelloA, HelloB,...
  double fractionOfElementsLost = Undefined; // range 0...1. Used by the receiving side
}
```

Figure 6.6 The data structure for a talk spurt

```
message L7_VoicePdu  // L7_VoiceProtocol <-> L7_VoiceProtocol
{
fields:
  // Layer PCI
  double startTime            = Undefined;  // The start time of the talkspurt
  double endTime              = Undefined;  // The end time of the talkspurt
  int connectionId            = Undefined;
  int noSegmentsInThisMonolog = Undefined;
  int segmentNumber           = Undefined;

  // Layer internal constants. Prefix const.
  int constPciLength  = 0;       // [bytes]

  // variables for statistics and dbg. Prex st.
}
```

Figure 6.7 The layer 7 voice Protocol Data Unit (PDU).

The simulator supports two talk spurt length distributions; fixed and truncated lognormal. The lognormal distribution (based on surveys done by e.g. Sharp et al. [10]) with parameters $\mu = 0$ and $\sigma = \sqrt{(2\ln(a))}$, where a is the desired expected value. The distribution is truncated to produce values ranging from 2 to 8 seconds.

The traffic model for the talk spurt duration is based on the 2400 bit/s Mixed-Excitation Linear Predictive enhanced (MELPe) encoder. 2.4 kbit/s MELPe operates at a sampling frequency of 8 kHz, and compresses blocks of 180 samples to 54 bits. The class *L7_MELPe* implements the MELPe and generates a speech stream of voice packets at the rate $1/t_{voice}$ where

$$t_{voice} = 9 \cdot t_{slot} \text{ and } t_{slot} = 22.5 \text{ msec}$$

The length of the voice packets sent down to the lower layer is $54 \cdot 9 = 486$ bits, which is 60.75 bytes. The simulator increases the length to 61 bytes. Each voice packet carries a sequence number and the number of fragments that constitutes the current monologue. The receiving *L7_MELPe* knows the exact duration of a monologue and calculates the fraction of segments lost. This number is used by the *L7_UserVoiceDialog* to determine if the voice quality is above a certain threshold *L7_UserVoiceDialog::shareOfPDUsNeeded*. If the test fails then the user is not able to interpret the voice content.

6.3 Voice dialogue

A voice dialogue is a sequence of voice monologues as outlined in Figure 6.8. The example below gives an example of how a dialogue is handled in the network.

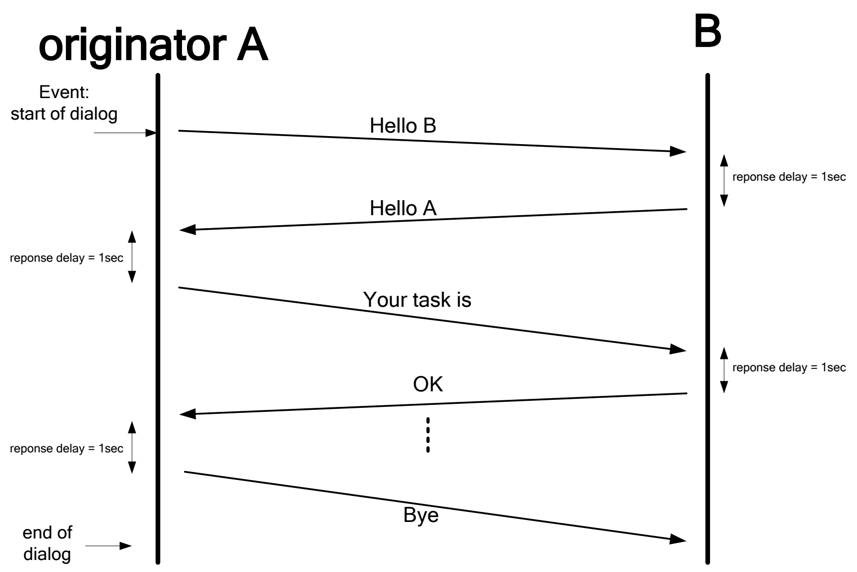


Figure 6.8 A conversation dialogue between persons A and B. The shown dialogue consists of exactly 5 talk spurts ("Hello B", "Hello A", "Your task is", "OK", and "bye".

6.3.1 Example

Consider the case where we want to model two different originators of voice where one uses multicast addressing. Then we specify the two parameter sets

Originator A = 2, Bset = {4} and λ = a1
Originator A = 0, Bset = {1,3,4,5} and λ = a2

When a dialogue establishment event occurs, a random destination B is taken from the *Bset*. The *Bset* is a model input parameter and by defining more than one element, different traffic patterns result at the network level. For the first parameter set above, the Bset contains one element {4} and hence results in a dialogue between persons 2 and 4 at a call rate given by a1.

For the latter parameter set, person 0 initiates dialogues at a frequency given by a2. Person 1,3,4 and 5 belongs to the multicast group, which means that one of them shall act as the responder B while the other shall listen only. The responder B is randomly drawn from the multicast set for each outgoing call. If we look at the snapshot A=0 and B=4 for the network in Figure 6.9, the dialogue between A and B will obviously lead to two-way layer 7 traffic between the nodes [6]. The nodes {1,3,5} are also involved but get one-way traffic only at layer 7 because the users shall listen only.

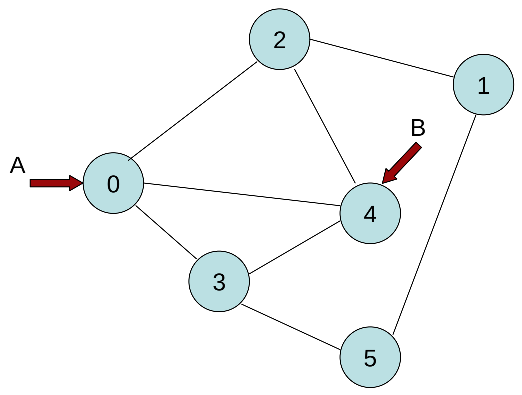


Figure 6.9 A snapshot when the originator A=0 and destination B=4 are to establish a dialogue.

6.3.2 State diagrams

The model is detailed by the two state diagrams in Figure 6.10 and Figure 6.11 which describe the behaviour of the two communicating parties. The model is enhanced with a timer to model impatient subscribers; neither A nor B wait forever for a response. For example, if *HelloA* is lost in the network, the originator A eventually times out and the dialogue fail. Another enhancement is to introduce a probability for having more than one *YourTaskIs/Ok* exchange. A human processing delay is added for analysing the incoming voice messages.

A's state diagram expresses a loop in state *wait2*, whose length is determined by the following algorithm. When the state *wait2* is entered, a loop counter is set to a random number in the range [0, N], where N is an integer constant. This loop counter is decremented for each round and the state is exited when the counter reaches zero. The number of *YourTaskIs/Ok* talk spurts is then limited to N+1.

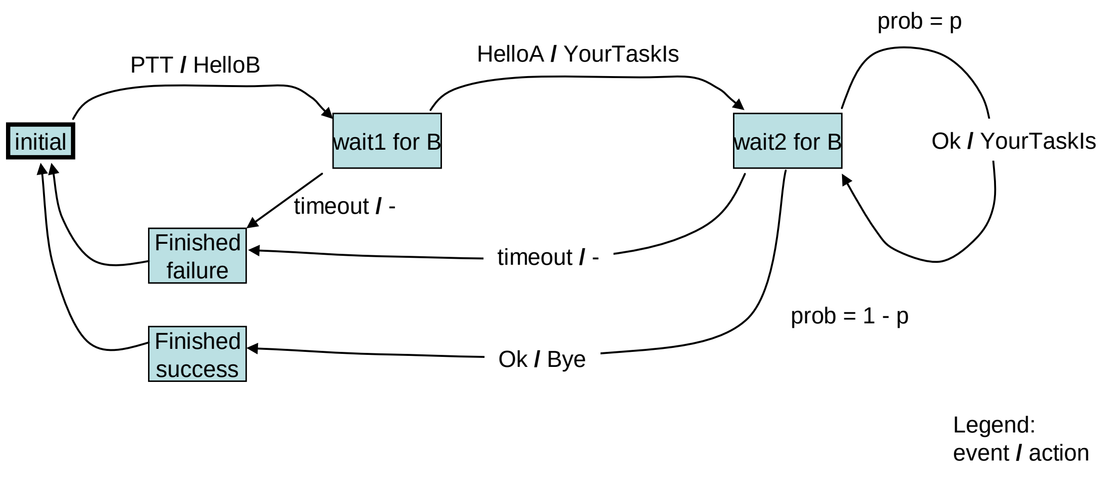


Figure 6.10 State diagram for the originator A forms a stochastic process.

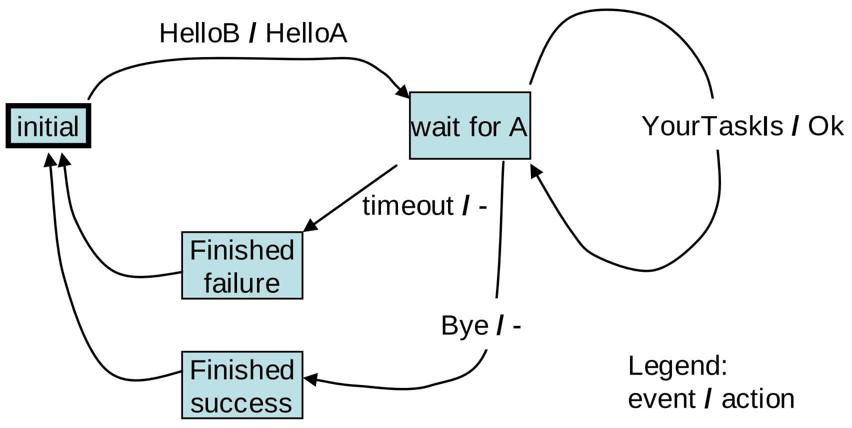


Figure 6.11 State diagram for the recipient B forms a stochastic process.

The figure below expresses the delay values for the state diagrams.

```
static const double shareOfPDUsNeeded = 1.0; // to understand the talkspurt, range 0...1
static const double pSendAnotherYourTaskIs = 0.0;

enum ConstantsInModule
{
  //! Constants for the log normal distribution [sec]
  ConstMinimumHoldingTime = 2, ConstMaximumHoldingTime = 8,

  //! Constants for the state diagrams in figure 2.3 and 2.4 [sec]
  ConstAWaitStateForB = 5, ConstBWaitStateForA = 5,

  //! A human user must process a sentance before giving a response.
      This is the delay in seconds.
  ConstUserResponseDelay = 1,
};
```

Figure 6.12 Numerical values used by the L7_UserVoiceDialog module

7 Statistics

In order to produce statistical output, the oProbe [8] software is integrated with the NBWF simulator. *Probes* are placed at specific positions in the source code, collecting samples during the simulation run. At the end of the simulation, estimates of mean of the sample distributions are available along with the surrounding confidence interval.
Section 7.1 describes all probes (and counters) currently defined in the simulator, while section 7.2 explains the concept of *matrix probes*.

7.1 Defined probes in the NBWF simulator

This section specifies the probes implemented in each module of the simulator.

7.1.1 Probes in the module L7_UserVoiceDialog

Dialogue length [s]
The time delay until the originator A returns to the initial state (Figure 6.10). Only dialogues ending successfully are included and samples are collected at the originator side only. A matrix probe is implemented for this probe name.

Dialogue failure probability
The fraction of the dialogues not completing successfully as defined in Figure 6.10. Samples are collected at the originator side only. A matrix probe is implemented for this probe name.

Dialogue rejection probability
The call rejection probability for calls issued by the local terminal. This probe records both dialogue events and monologue events. A call is rejected if there is an incoming call in progress, or if the local terminal issues a new call before the current call (dialogue/monologue) is completed.

FractionOfElelementsLost
A talk spurt is sent as a sequence of packets over the air interface and one or more may be lost. This probe measures the fraction of elements lost and counts both monologues and dialogues. A matrix probe is implemented for this probe name and is measured at B-side.

Talk spurt delay [sec]
Talk spurt delay is the latency time between the PTT event and the delivery of the last bit of the corresponding talk spurt. A certain number of packets must be received (*shareOfPDUsNeeded*) to be declared as successful, see the probe *FractionOfElelementsLost*. Talk spurts not fulfilling *shareOfPDUsNeeded*-threshold are rejected. This probe counts both monologues and dialogues. A matrix probe is implemented for this probe name. Measurements are taken at the B-side.

7.1.2 Probes in the module L7_VoiceProtocol

The *L7_VoiceProtocol* cannot differentiate between a monologue and a dialogue. Therefore all statistics taken within this module is the sum of both voice types.

crL7delay [sec]
This probe measures the CR PDU latency time from the originator to all of its recipients, that is, a single CR PDU may lead to many samples. A matrix probes is implemented for this probe name.

voiceL7pduDelay [sec]
This probe measures the end-to-end transit delay to the B-side for the individual packets of the talk spurts. A matrix probe is implemented for this probe name.

7.1.3 Probes in the module L3_Rrc

endToEndCRDelay
The delay t_x - t_0 , where t_x is the reception time of a CR, and t_0 is the time at which the CR was sent from the initiator's (PTT-node) RRC. This probe takes a sample each time an RRC module receives a CR from air.

7.1.4 Probes in the module L2_MacLayer

receiverSuccessRatio
This is an estimate of the overall probability of successful connection establishment (MAC-Connection in voice-mode) to a given member of a multicast group, regardless of the number of hops between the initiator (PTT-node) and the member. A connection establishment is defined as successful if the member both receives the CR and at least one PDU before the initialDataTimer expires.

7.1.5 Probes in the module L1_Baseband

phyNKTx
When the baseband processor receives a SOM-alarm, the number of ongoing transmissions in its neighbourhood is measured. A matrix probe is implemented for this probe name and the source address is set to the node address which caused the SOM-alarm.

snrPreamble [dB]
When the baseband processor gets a CAS-alarm, the current SNR level is calculated based on the ongoing transmissions in its neighbourhood. This value is sent to the probe *snrPreamble*. A matrix probe is implemented for this probe name and the source address is set to the node address which trigged the CAS-alarm.

snrPayload [dB]
When the baseband processor gets the end-of-payload alarm, the current SNR level is calculated based on the ongoing transmission in its neighbourhood. This value is sent to the probe *snrPayload*. A matrix probe is implemented for this probe name and the source address is set to the address of the sending node.

7.2 Matrix probes

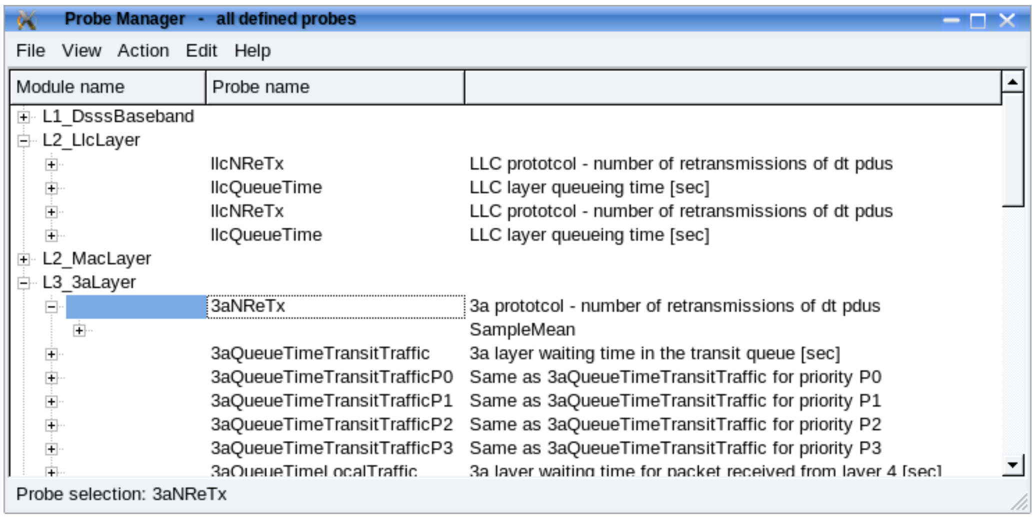
A probe is an object which accepts stochastic input data and produces statistics about the data. A matrix probe extends the functionality of the probe by supporting measurements over (source, destination)-pairs[3]. Thus, depending on its location within the simulation model, a matrix probe can be used to produce statistics on an end-to-end basis[4] or a per link basis[5]. A matrix probe is created in two steps:

1) Use the probe manager (Figure 7.1) to open the list of defined probes
2) Select Action->EnableMatrixProbe after selecting a probe.

Matrix probe data is now inserted and the next step is to configure the attributes:

1) Select View->MatrixProbes
2) Make a probe selection and
3) Then activate the Edit->Probe menu.

The matrix probe editor widget pops up (Figure 7.2) allowing the user to set the matrix probe parameters


Probe Manager - all defined probes
File View Action Edit Help

Module name	Probe name	
L1_DsssBaseband		
L2_LlcLayer		
	llcNReTx	LLC prototcol - number of retransmissions of dt pdus
	llcQueueTime	LLC layer queueing time [sec]
	llcNReTx	LLC prototcol - number of retransmissions of dt pdus
	llcQueueTime	LLC layer queueing time [sec]
L2_MacLayer		
L3_3aLayer		
	3aNReTx	3a prototcol - number of retransmissions of dt pdus
		SampleMean
	3aQueueTimeTransitTraffic	3a layer waiting time in the transit queue [sec]
	3aQueueTimeTransitTrafficP0	Same as 3aQueueTimeTransitTraffic for priority P0
	3aQueueTimeTransitTrafficP1	Same as 3aQueueTimeTransitTraffic for priority P1
	3aQueueTimeTransitTrafficP2	Same as 3aQueueTimeTransitTraffic for priority P2
	3aQueueTimeTransitTrafficP3	Same as 3aQueueTimeTransitTraffic for priority P3
	3aQueueTimeLocalTraffic	3a layer waiting time for packet received from layer 4 [sec]

Probe selection: 3aNReTx

Figure 7.1 Aprobe manager (class GUI_ProbeManager) displays information about probes organised according to the module they belong to. A click on the "+"-sign gives more detailed information

[3] Version 2.0 does not support batch means analysis of sampled data.
[4] For example end-to-end delays between node x and node y in figx
[5] For example, packet loss probability on the link x to y in figx

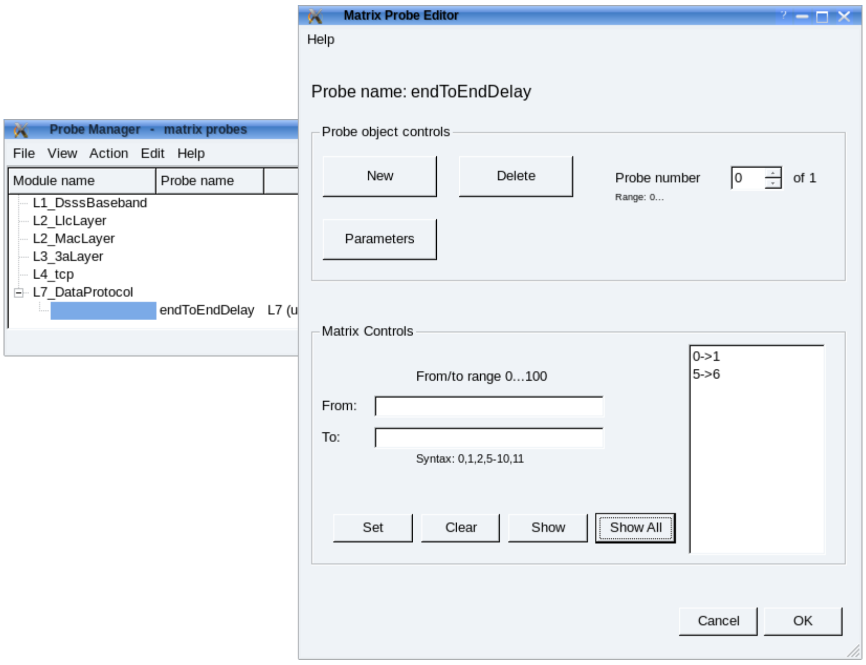
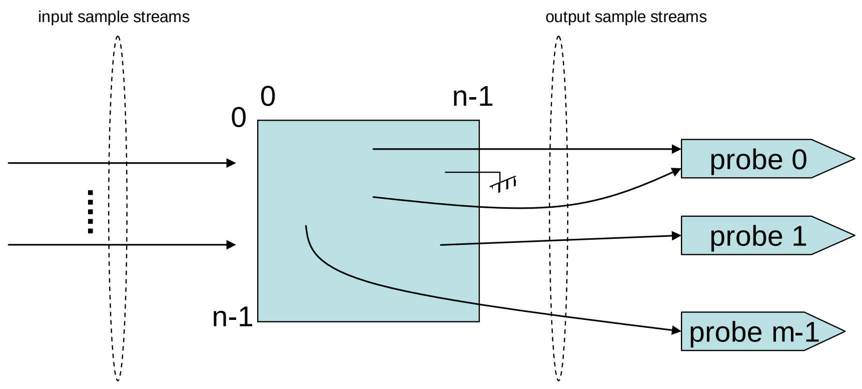


Figure 7.2 The matrix probe editor.

7.2.1 The sample matrix

This section describes the intention of the matrix controls section in the matrix probe editor widget. A matrix probe (class *PRB_ProbeMatrix*) uses one or more basic probe objects determined by the setup of its sample matrix. As illustrated by Figure 7.3, samples sent to a matrix probe are routed through a matrix where each element points to either a basic probe or NULL. A NULL pointer means no sampling for the corresponding (i, j)-pair. The content of this matrix is specified by the simulator's input data (file *setup/probeInFile.xml*).



Probes::sample (src, dst, statistics.crDelay, theSample);

Figure 7.3 The sample matrix is of order (n,n) where each elements points to a probe object, or a null pointer. The atomic modules send data to a matrix probe by using the OMNET_Probe::sample() function shown.

The following data structure describes the matrix probe attributes:

```
<probeInputFile>
  <defaults>
    <probeType>SampleMean</probeType>
    <alfaConfidenceCoefficient>0.9</alfaConfidenceCoefficient>
    <accuracy>0.49</accuracy>
    <trace>false</trace>
    <maxmin>false</maxmin>
  </defaults>
  <probe>
   <name>foo</name>
   <probeType>Terminating</probeType>
  </probe>
  <mprobe>
   <name>foo</name>
   <object>
     <sampleSets>
         <set from="2,5" to="5,2" />
         <set from="1,3" to="1,3" />
     </sampleSets>
     <probeType>SampleMean</probeType>
     <transientPeriodLength>100</transientPeriodLength>
   </object>
   <object>
     <sampleSets>
         <set from="1,4" to="4,1" />
     </sampleSets>
     <probeType>SampleMean</probeType>
     <transientPeriodLength>101</transientPeriodLength>
   </object>
  </mprobe>
</probeInputFile>
```

The probe module reads in matrix probe data from the same XML coded file (*probeInFile.xml*) as used for basic probes. Basic probes are identified by the <probe>-tag while matrix probes are identified by the <mprobe>-tag. A probe object is created for each <object>-section found, and the elements in the sample matrix that shall point to this object are determined by <set>-tags within the enclosing <sampleSets>-section. Elements not addressed are assigned a NULL-pointer and those (source, destination)-pairs are excluded from the input sample stream. Each object inherits the same proprieties as of the basic probe with the same name. The matrix probe editor (class *GUI_MatrixProbeEditor*) supports read and write operation on tags within the <mprobe>-section.

7.2.2 Usage

Consider the situation where you want to estimate the MAC connection setup delay (*MacCrDelay*) between node 1 and node 2 in Figure 6.9, and the end-to-end connect setup delay (*L7CrDelay*) between the nodes 3 and 4. Activate the matrix probe editor from the probe manager widget selecting the *L7CrDelay* probe as the input. Then the picture in Figure 7.2 pops up.

The probe object controls available are:

New: Creates a new probe object with a matrix of NULL pointers.

Delete: Deletes the current probe object which is identified by the number shown in the spin box to the right.

Parameters: Opens an editor for editing probe parameters.

The elements of the matrix belonging to the current probe object are manipulated from the following matrix controls:

From/To fields: Selects the (i, j)-elements in the matrix

Set: Inserts a pointer to the current probe object for all elements identified by from/to.

Clear: Inserts a NULL pointer for all elements identified by from/to

Show: Displays the elements identified by the current from/to fields

Show All: Displays all elements pointing to the current probe

Example: from/to = "1-2"/"1,5,7"
These input strings represents the (i, j)-elements {(1,5), (1,7), (2,5), (2,7)}. The diagonal always contains NULL pointers.

7.2.3 Implementation

The Model-View-Controller (MVC) pattern is used during the implementation of the probe manager (class *GUI_ProbeManager*). As the user brows around in the probe tree and alters data, the actual information about probe parameters is managed through the class *PRB_TreeModel* which inherits *QAbstractItemModel*. Functions to read/write from/to XML-files are placed in the class *OPROBE::XML_UserParameters*.

Appendix A MAC PCI

This is a description of the MAC protocol control information (PCI) assumed in the simulator. Note that this differs from the suggested PCI in the link layer specification document, because some of the functionality is omitted in the simulator. The fields included here are considered as required in order to comply with the assumed specification described in section 2.
All MAC PDUs are on the format specified in Table A.1. A common base header precedes a PDU type specific extension header. The PDU type is indicated in the base header (Table A.2). The following extension headers are defined:

- MAC Connect Request (Table A.3)
- MAC Connect Confirm (Table A.4)
- MAC Disconnect Request (Table A.5)
- MAC Disconnect Confirm (Table A.6)
- MAC Connection-oriented Data (Table A.7)
- MAC Connection-oriented Return Channel (Table A.8)
- MAC Connection-less Data (Table A.9)

MAC PDU		
Field	**Description**	**Size (bits)**
Base header	Mandatory. Includes information carried by all MAC PDU types.	60
Extension header	Mandatory	variable
Payload	Optional	variable
	Sum:	**variable**

Table A.1 MAC PDU format

MAC Base Header		
Field	**Description**	**Size (bits)**
PDU type	Indicates which extension header that follows the base header. Currently, 7 different PDY types are defined, so 4 bit should be more than enough.	4
Crypto IV		40
PaddingLength	Number of padding bytes. Padding is needed in order to fill the PHY interleaver. Interleaver length is signalled as ICI by PHY, so the PDU length is found by subtracting the padding. Worst case: pad 1 time slot at highest PHY mode (96kbps?): 96kbps*0.0225/8=270. => 9 bit	8
Source MAC address		8
	Sum:	**60**

Table A.2 MAC base header

MAC Connect Request (CR)		
Field	**Description**	**Size (bits)**
Base header		60
SAP requested	CODTC or COVTC. Part of ECID/RID	2
Connection serial number	ECID	6
Initiator MAC address	ECID	8
Priority		2
Range start	First slot number in reserved range	4
Range length	Number of slots in reserved range	4
Destination address	Unicast or Multicast destination address	8
CC-set	Max 5 CCs allowed, 8 bit each	40
Relay set length	This number of nodes in CC-set is requested as relays	3
CC-Schedule	Experimental	9
Following fields are needed for piggybacked CC:		
CC Destination Address	The originator address in the CR responded to by this CC	8
CC Range start	The range start of the CR responded to by this CC	4
CC Range length	The range length of the CR responded to by this CC	4
CC Accepted	0: negative CC, 1: positive CC	1
CRC-16	Covers entire PDU (small payload possible)	16
	Sum:	**177**

Table A.3 Extension header: MAC Connect Request

MAC Connect Confirm (CC)		
Field	**Description**	**Size (bits)**
Base header		60
Destination address	RID. The originator address in the CR responded to by this CC.	8
SAP requested	ECID. Same as in CR responded to by this CC	2
Connection serial number	ECID. Same as in CR responded to by this CC	6
Initiator MAC address	ECID. Same as in CR responded to by this CC	8
Accepted	0: negative CC, 1: positive CC	1
Range start	Same as in CR responded to by this CC	4
Range length	Same as in CR responded to by this CC	4
CRC-16		16
	Sum:	**109**

Table A.4 Extension header: MAC Connect Confirm

MAC Disconnect Request (DR)		
Field	**Description**	**Size (bits)**
Base header		60
SAP requested	ECID. Same as in CR	2
Connection serial number	ECID . Same as in CR	6
Initiator MAC address	ECID. Same as in CR	8
Reason	Indicate reason for disconnecting	2
DC-Schedule	Experimental	9
CRC-16		16
	Sum:	**103**

Table A.5 Extension header: MAC Disconnect Request

MAC Disconnect Confirm (DC)		
Field	**Description**	**Size (bits)**
Base header		60
Destination address	RID. Same as in DR responded to by this DC	8
SAP requested	ECID. Same as in DR responded to by this DC	2
Connection serial number	ECID. Same as in DR responded to by this DC	6
Initiator MAC address	ECID. Same as in DR responded to by this DC	8
Range start	Same as in DR responded to by this DC	4
Range length	Same as in DR responded to by this DC	4
CRC-16		16
	Sum:	**108**

Table A.6 Extension header: MAC Disconnect Confirm

MAC Connection-oriented Data (CoDt)		
Field	**Description**	**Size (bits)**
Base header		60
SAP requested	ECID. Same as in CR	2
Connection serial number	ECID. Same as in CR	6
Initiator MAC address	ECID. Same as in CR	8
Feedback request	Indicate in which TDMA frame return channel is available. 0: this frame, 1: next frame, and so on..	4
Long/short	0: CRC-16, covers only MAC PCI 1: CRC-32, covers entire MAC PDU	1
CRC-16/32	Depends on long/short field	16/32
	Sum:	**97/113**

Table A.5 Extension: MAC connection-oriented data

MAC Connection-oriented Return Channel (CoDtReturn)		
Field	**Description**	**Size (bits)**
Base header		60
SAP requested	ECID	2
Connection serial number	ECID	6
Initiator MAC address	ECID	8
CRC-16	Covers entire PDU	16
	Sum:	**92**

Table A.6Extension header: MAC connection-oriented return channel

MAC Connection-less Data (ClDt)		
Field	**Description**	**Size (bits)**
Base header		60
Destination address	Unicast or multicast destination MAC address	8
priority		2
Long/short	0: CRC-16, covers entire MAC PDU 1: CRC-32, covers entire MAC PDU	1
CRC-16/32	Depending on long/short bit	16/32
	Sum:	**97/113**

Table A.7 Extension header: MAC connection-less data

Appendix B Abbreviations

3aPDP	3a Packet Data Protocol
AM	Acknowledged Mode
ARQ	Automatic Repeat Request
CAS	Carrier Activity Sensor
CC	Connect Confirm
CNR	Combat Net Radio
CR	Connect Request
DC	Disconnect Confirm
DR	Disconnect Request
ECID	End-to-end Connection ID
FCS	Frame Check Sequence
ICI	Interface Control Information
L7	Layer 7 (ref. OSI model)
LCID	Local Connection ID
LLC	Link Layer Control
MAC	Medium Access Control
MELPe	Mixed Excitation Linear Prediction, enhanced
MV	Multicast Voice
NAS	Non Access Stratum
NBWF	Narrowband Waveform
OSI	Open System Interconnection
PCAS	Preliminary CAS
PCI	Protocol Control Information
PDU	Protocol Data Unit
PHY	Physical layer (ref. OSI model)
PTT	Push To Talk
R2N	Reservation 2-hop Neighbour
ResvDB	Reservation Database
RID	Reservation ID
RN	Reservation Neighbour
RO	Reservation Originator
RRC	Radio Resource Control
SAP	Service Access Point
SDU	Service Data Unit
SNR	Signal-to-Noise Ratio
SOM	Start Of Message
TDMA	Time Division Multiple Access
UM	Unacknowledged Mode
UBM	User Behaviour Model

References

[1] NATO C3B SC/6 - AHWG/2, "Technical standards for narrowband physical layer of the NATO network enabled communications waveform and VHF propagation models (Draft 4)","Mar.2010.

[2] T. J. Berg, "The Design of an Initial NBWF Network Simulator," FFI Report 2008/01921 (Unclassified), 2008.

[3] S. Haavik, "Link Layer Protocol Design for NBWF," FFI Report 2009/01895 (Unclassified), 2009.

[4] Vivianne Jodalen, Bjørn Solberg, and Svein Haavik, "NATO Narrowband Waveform (NBWF) - overview of link layer design," FFI Report 2010/01248 (Unclassified), 2010.

[5] NATO C3B SC/6 - AHWG/2, "Technical standards for narrowband physical layer of the NATO network enabled communications waveform and VHF propagation models (Draft 2)","Sept.2008.

[6] ITU-T X.200 and ISO/IEC 7498, "Open System Interconnection - Basic reference model,"1994.

[7] T. J. Berg, "oTWLAN (http://sourceforge.net/projects/otwlan/)," 2010.

[8] T. J. Berg, "oprobe (http://sourceforge.net/projects/oprobe/)," 2010.

[9] Bjørn Solberg, "Physical layer specification (edition 2.0) for the FFI NBWF OMNET network simulator," Unpublished Jan.2009.

[10] D.S.Sharp, N.Cackov, N.Laskovic, S.Qing, and L.Trajkovic, "Analysis of public safety traffic on trunked land mobile radio systems," IEEE Selected areas in communications, vol. 22, no. 7, pp. 1197-1205, Sept.2004.