OPTIMIX:

OPTIMISATION OF MULTIMEDIA OVER WIRELESS IP LINKS VIA X-LAYER DESIGN

DELIVERABLE D3.3A
SPECIFICATION AND PRELIMINARY DESIGN OF DATA LINK LAYER

Contractual date of delivery to EC	M10 (31/01/2009)
Actual date of delivery to EC	30/01/2009
Revision date (if applicable)	01/05/09
Version number	1.1

Editor(s)	VTT
Participants	VTT, THALES, BME, CNIT

Deliverable nature and security level	**Report, PU**
Estimated work-load	
Total number of pages	49

Executive Summary

This document is the first specification for the work done in Task 3.3 (data link layer) in the OPTIMIX project. The specifications and work described in this document will be detailed further in follow-up documents of Task 3.3, namely D3.3b and D3.3c. The purpose of this document is to provide a fist phase specification and design of the OPTIMX data link layer functionalities. Besides the basic message passing mechanisms, the data link layer functionalities considered here include header compression techniques, point-to-multipoint transmission, and message scheduling techniques.

The role of the data link layer is to provide the basic medium access control and data link control mechanisms to enable data transfer across a physical link. At the sender side, the data link layer accepts packets from the network layer entity, encapsulates them into transmission frames, and submits them to the physical layer for transmission, at the same time managing the media access and controlling the data delivery. In the receiver, the data link layer receives messages from the physical layer, decapsulates them, and forwards packets containing payload data to the network layer. The data link layer also interacts with the physical layer to controlling the physical link error handling and data flow.

The implementation of the data link layer is technology specific. The OPTIMIX project focuses on two standards, that is, the IEEE 802.11 WLAN and IEEE 802.16 WiMAX. The aim is to obtain the specification for the basic data link layer services from these standards, as the basic services are not key issues in the project. The standard data link layer implementations are to be extended with enhancements for multimedia delivery and cross-layer signalling in the project.

This document gives an overview of the data link layer structures and functionalities supported by the IEEE 802.11 WLAN and IEEE 802.16 WiMAX standards. This description includes the basic message passing mechanisms supported by the standard with solutions to achieve reliability and Quality of Service (QoS). The standard data link layer implementations are discussed in the light of the OPTIMIX system and new interfaces are specified towards the cross-layer signalling frameworks utilized in the project. The standard data link layer solutions are also to be modified to support forwarding damaged frames to higher layers for processing, instead of just dropping them. Multimedia flows can benefit from the damaged frames to enhance the quality of the media stream compared to the case where such frames are dropped. Partial checksum mechanisms are considered at the data link layer to implement this feature.

The OPTIMIX project considers point-to-multipoint transmission of multimedia. In this document, the implementation of multicast in the data link layer is discussed. The focus is on addressing the bad support for multimedia flows in the multicast implementation defined by the IEEE 802.11 standard. Different solutions for the problem are analyzed.

The document considers also the interfaces and operation of the data link layer towards the cross-layer signalling framework identified for the OPTIMIX system and presented in more detail in the OPTIMIX Task 3.2 specifications document D3.2a. The use case considered here is the multiuser scheduling that aims to optimize transmission over the physical medium. In principle, the data link layer interfaces with the cross-layer signalling system to obtain the necessary feedback information for the scheduling operations as well as to provide data for other system entities regarding the data link status (e.g. frame transmission performance and buffer state information).

Robust header compression is a solution for reducing the protocol overhead introduced by network packetization to the useful video data. The issue is critical especially in the case of video transmission over constrained bandwidth wireless links. In this document, a solution of header compression for DCCP/IPv6 profile being compliant to RoHC standard is described.

This document also includes evaluation of WLAN throughput models. The modelling of WLAN data link level throughput is discussed based on measurements performed using IEEE 802.11b/g. The goal of the measurements and analysis is to establish a model for the throughput characteristics. Once obtained, such a model can be used in the development of an error detection/correction code later in the project.

Finally, the data link layer elements for the OPTIMIX simulation chain are presented. The basic data link layer implementation will base on the WLAN and WiMAX standards. The interface specifications defined in the OPTIMIX document D1.3 will be used in the design of the OMNeT++ modules and are to be detailed further as the implementation of the modules proceeds.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 Introduction

This document provides the first specification and design of the OPTIMX data link layer (DLL) functionalities. The role of the DLL within the OPTIMIX system is to provide the required mechanisms for transferring data and control information over the physical medium. The DLL mechanisms considered in the project are based on the IEEE 802.11 WLAN and IEEE 802.16 WiMAX technologies. The basic DLL functionalities are adopted from these technologies and extended with more advanced mechanisms to better support multimedia delivery. The main improvements considered for the DLL here include the use of partial checksum mechanisms for data frames, advanced data link level multicast, efficient multiuser queuing and scheduling solutions, and the Robust Header Compression (RoHC) in the context of the Datagram Congestion Control Protocol (DCCP). In addition to traditional data and control information delivery, the DLL is to be integrated as a part of the cross-layer control and feedback signalling framework of the OPTIMIX system, which necessitates the DLL to support the associated interfaces and information collection.

Chapter 2 provides an overview of the basic DLL functionalities associated with the IEEE 802.11 WLAN and IEEE 802.16 WiMAX technologies. The basic DLL mechanisms are required to provide a message passing service for the higher layers over the physical medium. The basic DLL services are not key issues in OPTIMIX, and they are thus implemented according to the IEEE 802.11 WLAN and IEEE 802.16 WiMAX standards. The chapter also discusses the integration of the DLL with the rest of the OPTIMIX system, including the interfaces towards the Media Independent Handover (MIH) and triggering frameworks presented in the OPTIMIX specifications document D3.2.

In Chapter 3, the data link layer multicast is discussed in the context of multimedia transmission. The chapter presents available solutions to address the deficiencies of the multicast support defined in the IEEE 802.11 standard in the case of multimedia delivery. The main problems with the existing standard are that there is no support for acknowledgements in the multicast case and the transport speed is limited to the low base rates.

Moreover, this document discusses two data link level solutions for optimizing multimedia delivery across a wireless link: Chapter 4 discusses a solution for multiuser scheduling to be used in the OPTIMIX system. Chapter 5 presents a solution developed for header compression for DCCP/IPv6 profile being compliant to the Robust Header Compression (RoHC) standard. RoHC can be used for reducing the protocol overhead associated with multimedia transmissions in the case of bandwidth constrained wireless links and thus increasing efficiency.

Chapter 6 outlines measurements conducted in an IEEE 802.11 b/g environment. The measurement results back up the assumptions taken in the project, that is, capturing corrupt frames is worthwhile and that it is possible to estimate the future characteristics of a WLAN channel. However, the channel model is hard to be described with a universal equation and parameters, and should be measured instead. Once obtained, such a model can be used in the development of an error detection/correction code later in the project.

The envisioned DLL simulation models to be implemented into the OMNeT++ simulator are described shortly in Chapter 7. This document gives a general level overview of the DLL procedures that will be supported in the OPTIMIX simulation chain. No simulation results are presented, however, but they are left for the upcoming deliverables D3.3b and D3.3c. Chapter 8 finally concludes the document.

2 Generic Data Link Layer Functionalities

2.1 IEEE 802.11 DLL

The IEEE 802.11 DLL consists of two sublayers: the logical link control (LLC) and the medium access control (MAC). The LLC, which is based on the IEEE 802.2 standard, forms the upper sublayer of the DLL. The purpose of the LLC is to exchange data between end users across a LAN using an 802-based MAC controlled link, i.e. 802.11 in this case. The LLC is independent of the topology, transmission medium, and medium access control technique. During communications, the higher layers (transport and network layers) pass user data (IP packets) down to the LLC. The IP packets are encapsulated into a control header, creating an LLC PDU (protocol data unit). The LLC PDU is then sent to the MAC layer through the MAC SAP (service access point). Figure 1 illustrates the LLC PDU structure.

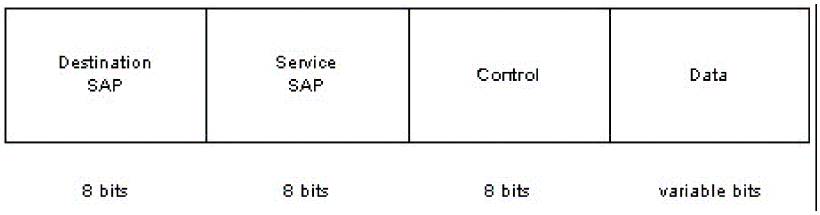


Figure 1 – The LLC PDU structure.

The LLC provides three types of services to network layer protocols:

- Unacknowledged connectionless service
  - Datagram-style
  - No error or flow control mechanisms
  - Supports individual, multicast and broadcast addressing
- Connection-oriented service
  - Establish connection between peer LLCs
  - No support for multicast or broadcast
  - Provides error and flow control (continuous and stop-and-wait ARQ)
- Acknowledged connectionless service
  - A successful delivery of datagrams is acknowledged
  - Flow and error control is provided using stop-and-wait ARQ

The IEEE 802.11 MAC service [13] defines the protocol and compatible interconnection for peer LLC entities to exchange MAC service data units (MSDUs). The MAC uses the underlying physical layer services (defined by IEEE 802.11 PHY standard) to transport the MSDUs to the peer MAC entity which will forward it to the peer LLC entity. MSDU delivery is best effort by default and there is no guarantee that the transmission is successful. IEEE 802.11e standard defines mechanisms to introduce Quality of Service (QoS) to the frame transmission. The MAC provides both multicast and broadcast data services in addition to unicast. The nodes are identified by a 48-bit IEEE MAC address. The IEEE 802.11 standard uses special broadcast and multicast addresses to support these kinds of communication. The MAC supports two modes of operation: under control of an access point (AP) and between independent stations (i.e. ad hoc). The protocol includes authentication, association, and re-association services, an optional encryption/decryption procedure, power management to reduce power consumption in mobile stations, and a point coordination function for time-bounded transfer of data. The MAC services are split into station services (SSs) and distribution system services (DSSs).

Each IEEE 802.11 station (STA), including the APs, provides the following MAC services:

a) Authentication
b) Deauthentication
c) Data confidentiality
d) MSDU delivery
e) Dynamic frequency selection (DFS)
f) Transmit power control (TPC)
g) Higher layer timer synchronization (QoS facility only)
h) QoS traffic scheduling (QoS facility only)

The services that are part of IEEE 802.11 distribution service (DS) are as follows:

a) Association
b) Disassociation
c) Distribution

d) Integration
e) Reassociation
f) QoS traffic scheduling (QoS facility only)

The DS is the architectural component used to interconnect basic service sets (BSSs). A BSS is the basic building block of an IEEE 802.11 based LAN and may include two or more STAs. The IEEE 802.11 standard does not specify in detail the implementation of the DS. It may be created from many different technologies including current IEEE 802 wired LANs. The standard does not constrain the DS to be either data link or network layer based. Moreover, a DS may be either centralized or distributed in nature.

The general MAC frame structure used in IEEE 802.11 WLAN is shown in Figure 2 Each frame consists of a MAC header, a frame body, and a frame check sequence (FCS). The header comprises of frame control, duration, address, and sequence control information. The fields Address 2, Address 3, Sequence Control, Address 4, QoS Control, and Frame Body are present only in certain frame types and subtypes. The Frame Body contains information specific to the frame type and the FCS contains a cyclic redundancy code (CRC). The maximum frame body size is determined by the maximum MSDU size (2304 octets) plus any overhead from security encapsulation.

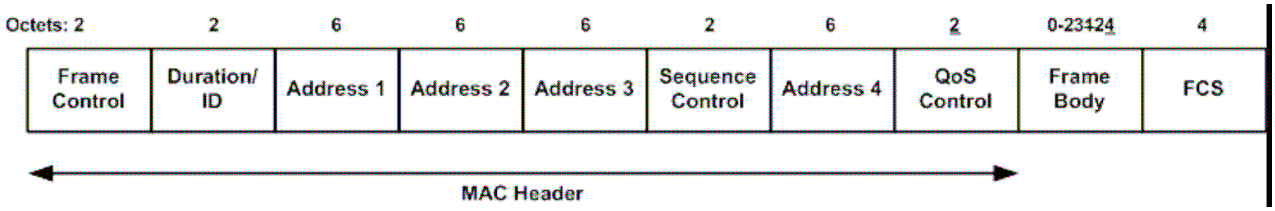


Figure 2 – General MAC frame structure in IEEE 802.11 (Source: [13]).

The IEEE 802.11 uses contention-based channel access for STAs with no QoS support. The multiple access scheme used in IEEE 802.11 WLAN is called Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA). For CSMA/CA two basic coordination functions can be used: the Distributed Coordination Function (DCF) and the Point Co-ordination Function (PCF). CSMA/CA is based on asynchronous frame transmission (connectionless) and delivers a best effort service without any bandwidth and latency guarantees. The main advantages of the method are that it suits well for network protocols such as TCP/IP, is quite adaptable to variable traffic conditions, and is rather robust against interferences. The operation principle of CSMA/CA is the following: When a STA detects that the medium is free it begins to decrement its back-off counter from the DIFS (DCF Inter-Frame Space) period. When the back-off counter reaches zero the STA begins the transmission provided that the medium is still free. If the transmission collides, the STA chooses new random back-off counter value and waits for the next transmission opportunity. PCF allows also for contention-free transmissions through dividing the time after each transmission into a Contention Free Period (CFP) and a Contention Period (CP). During the CFP, the STAs are polled by the point coordinator with round-robin algorithm and granted a priority-based access to the medium. The PCF thus provides contention free services for non-QoS STAs.

The IEEE 802.11 MAC includes also a more advanced coordination function that can be used to achieve QoS support for the 802.11 link, namely the Hybrid Coordination Function (HCF). The HCF is supported by all QoS STAs. The HCF combines functions from the DCF and PCF with enhanced, QoS-specific mechanisms and frame subtypes to allow a uniform set of frame exchange sequences to be used for QoS data transfers during both the CP and CFP. The HCF uses both a contention-based channel access method, called the enhanced distributed channel access (EDCA) mechanism for contention-based transfer and a controlled channel access, referred to as the HCF controlled channel access (HCCA) mechanism, for contention-free transfer. The EDCA mechanism provides differentiated service for STAs using eight different UPs (user priorities). The HCCA employs a QoS-aware centralized coordinator, called a hybrid coordinator (HC), which is collocated with the AP of a BSS. HCCA can be used to achieve limited-duration controlled access phase (CAP) for contention-free transfer of QoS data. Figure 3 summarizes the components of the MAC architecture discussed above.

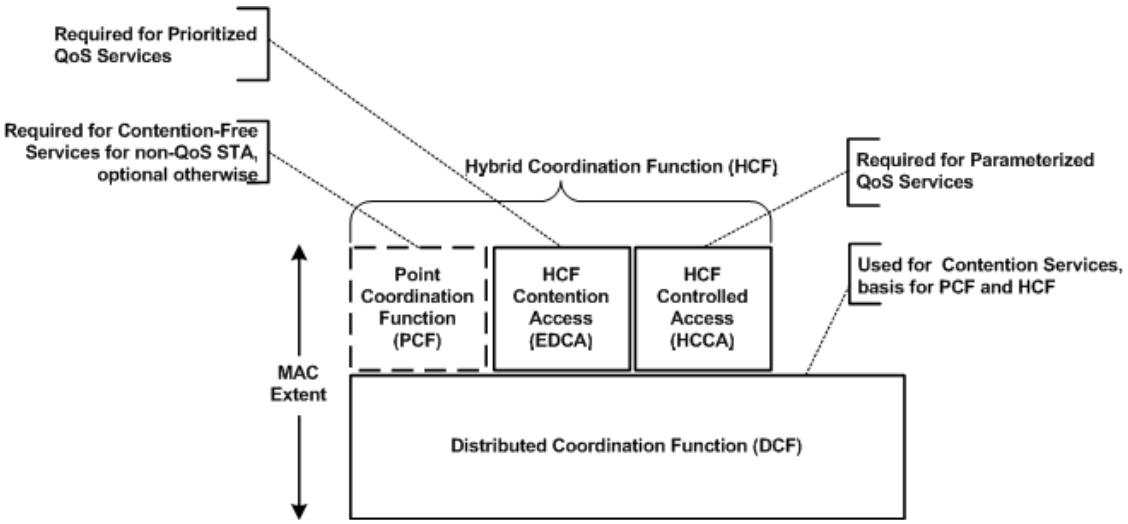


Figure 3 – The IEEE 802.11 MAC architecture (Source: [13]).

To address the hidden node problem in IEEE 802.11 WLANs, a mechanism called RTS/CTS (Request To Send/Clear To Send) can be used. The hidden node problem is caused by the attenuation of the radio signal during transmission, and thus all nodes may not hear each other because the attenuation is too strong between them. Since the transmissions are based on the carrier sense mechanism, this may cause nodes to transmit at the same time. The RTS/CTS introduces a handshaking to the beginning of a data transmission: before sending a frame, the transmitter sends a RTS and waits for a CTS from the receiver. The reception of a CTS indicates that the receiver was able to receive the RTS and that the channel is clear in its area, so the data transmission can begin. Since all the nodes may not hear the actual data transmission, the RTS and CTS messages contain the expected length of the upcoming transmission. This is the collision avoidance feature of the RTS/CTS mechanism (also referred to as virtual carrier sense): all the other nodes restrain themselves from accessing the channel after a CTS even, if they do not sense any transmissions over the medium. The RTS/CTS also results in shorter collisions in time. If two nodes attempt to transmit in the same slot of the contention window, their RTSs collide and they don't receive any CTS. In the normal scenario, the whole packet would be lost. However, because of the overhead of RTS/CTS handshaking, it is typically not used for small packets or lightly loaded networks.

In CSMA/CA, the transmitter cannot detect collisions on the medium. The air is also more prominent of errors than a wire, so there is a higher chance of packets being corrupted in a WLAN than in a LAN link. To introduce reliability to the frame transmission, MAC protocols implement positive acknowledgement (automatic repeat request, ARQ) and MAC level retransmissions. The ARQ mechanism uses acknowledgments (ACKs) and timeouts to detect problems in frame transmission. The receiver sends an ACK to the transmitter to indicate that it has correctly received a data frame. If the sender does not receive an ACK after a predetermined amount of time (defined with a timeout parameter), it re-transmits the frame until it receives an ACK or exceeds a predefined number of re-transmissions allowed. Sequence numbering of data frames is used for determining the correct reception of data packets. The ACKs are "embedded" in the MAC protocol, so they are guaranteed not to collide. According to the standard, broadcast and multicast packets are not acknowledged, so they are more likely to fail. The use of MAC-level ACKs and retransmissions however creates the possibility that a frame may be received more than once, so the receiver MAC also needs to implement detection and filtering of such duplicate frames.

Frame retransmission is not very efficient in the case of long frames (a single error causes the retransmission of the entire frame). Thus, frame fragmentation can be used to increase performance under high error rates. Fragmentation means that big packets are sent in small pieces over the medium. Fragmentation always adds some overhead as the frame headers are duplicated in every fragment. Moreover, each fragment is individually checked and retransmitted, if necessary. The benefits of this method however are that in case of error, the node needs to retransmit only a small fragment, making it faster, and if the medium is very noisy, a small packet has a higher probability of getting through without errors, increasing the chance of successful transfer under bad conditions. According to the standard, only MPDUs with a unicast receiver address are fragmented; multicast and broadcast frames should not be fragmented.

The MAC interacts with the underlying physical layer (PHY) through the PHY-SAP interface [13]. The primitives associated with communication between the IEEE 802.11 MAC and the IEEE 802.11 PHY are divided into service primitives that support MAC peer-to-peer interactions and service primitives that have local significance and support sublayer-to-sublayer interactions. The PHY-SAP peer-to-peer service is realized using PHY-DATA request, indication, and confirmation primitives. The PHY-SAP sublayer-to-sublayer service primitives and their parameters are listed in Table 1 through Table 3. A more detailed description of each primitive is also included below to give a better view of

the interactions between the IEEE 802.11 MAC and the IEEE 802.11 PHY. The reader is advised to study the standard for the PHY related details as they are out of the scope of this document.

Primitive	Request	Indicate	Confirm
PHY-TXSTART	X		X
PHY-TXEND	X		X
PHY-CCARESET	X		X
PHY-CCA		X	
PHY-RXSTART		X	
PHY-RXEND		X	

Table 1 – PHY-SAP sublayer-to-sublayer service primitives.

Parameter	Primitive	Value
DATA	PHY-DATA.request PHY-DATA.indication	Octet value X'00'-X'FF'
TXVECTOR	PHY-TXSTART.request	A set of parameters
STATUS	PHY-CCA.indication	BUSY, IDLE
RXVECTOR	PHY-RXSTART.indication	A set of parameters
RXERROR	PHY-RXEND.indication	NoError, FormatViolation, CarrierLost, UnsupportedRate

Table 2 – PHY-SAP service primitive parameters.

Parameter	Primitive	Value
DATARATE	TXVECTOR, RXVECTOR	PHY dependent. The name of the field used to specify the Tx data rate and report the Rx data rate may vary for different PHYs.
LENGTH	TXVECTOR, RXVECTOR	PHY dependent

Table 3 – Vector descriptions.

PHY-DATA.request

Function and use:
The PHY-DATA.request primitive is generated by the MAC sublayer to transfer an octet of data to the PHY entity and it can only be issued following a transmit initialization response (PHY-TXSTART.confirm) from the PHY. When the PHY entity receives the octet, it will issue a PHY-DATA.confirm to the MAC sublayer.

Semantics:
The primitive includes the following parameters: PHY-DATA.request(DATA). The DATA parameter is an octet of value X'00' through X'FF'.

PHY-DATA.indication

Function and use:
This primitive is generated by the PHY entity to inform the local MAC entity about the transfer of data. The receiving PHY entity issues a PHY-DATA.indication to transfer the received octet of data to the local MAC entity. The effect of receipt of this primitive by the MAC is unspecified in the standard.

Semantics:
The primitive provides the following parameters: PHY-DATA.indication (DATA). The DATA parameter is an octet of value X'00' through X'FF'.

PHY-DATA.confirm

Function and use:
This primitive is sent by the PHY to the local MAC entity to confirm the transfer of data from the MAC entity to the PHY. The PHY will issue this primitive in response to every PHY-DATA.request primitive issued by the MAC sublayer. The receipt of this primitive at the MAC will cause the MAC to start the next MAC entity request.

Semantics:
The semantics of the primitive are as follows: PHY-DATA.confirm. This primitive has no parameters.

PHY-TXSTART.request

Function and use:
The MAC sublayer uses this primitive to request the local PHY entity to start the transmission of an MPDU. This primitive will be issued by the MAC sublayer to the PHY entity when the MAC sublayer needs to begin the transmission of an MPDU. When receiving this primitive, the PHY entity will start the local transmit state machine.

Semantics:
The primitive provides the following parameters: PHY-TXSTART.request (TXVECTOR). The TXVECTOR is a list of parameters that the MAC sublayer provides to the local PHY entity in order to transmit an MPDU.

PHY-TXSTART.confirm

Function and use:
The PHY entity issues this primitive to the local MAC entity to confirm the start of a transmission. The PHY will issue this primitive whenever it receives a PHY-TXSTART.request primitive from the MAC sublayer and is ready to begin accepting outgoing data octets from the MAC. The receipt of this primitive will cause the MAC entity to start the transfer of data octets.

Semantics:
The semantics of the primitive are as follows: PHY-TXSTART.confirm. This primitive has no parameters.

PHY-TXEND.request

Function and use:
The MAC sublayer uses this primitive to request the local PHY entity to complete the current transmission of the MPDU. The MAC sublayer generates this primitive after the reception of the last PHY-DATA.confirm from the local PHY entity for the MPDU currently being transferred. The receipt of this primitive will cause the local PHY entity to stop the transmit state machine.

Semantics:
The semantics of the primitive are as follows: PHY-TXEND.request. This primitive has no parameters.

PHY-TXEND.confirm

Function and use:
The PHY-TXEND.confirm primitive is sent by the PHY to the local MAC entity to confirm the completion of a transmission. This primitive is generated in response to every PHY-TXEND.request primitive received by the PHY immediately after transmitting the end of the last bit of the last data octet indicating that the symbol containing the last data octet has been transferred. The receipt of this primitive by the MAC entity provides the time reference for the contention backoff protocol.

Semantics:
The semantics of the primitive are as follows: PHY-TXEND.confirm. This primitive has no parameters.

PHY-CCARESET.request

Function and use:
The MAC sublayer issues this primitive to the local PHY entity to reset the CCA (clear channel assessment) state machine. This primitive is generated at the end of a NAV timer. This request can be used by some PHY implementations that may synchronize antenna diversity with slot timings.

Semantics:
The semantics of the primitive are as follows: PHY-CCARESET.request. This primitive has no parameters.

PHY-CCARESET.confirm

Function and use:
The PHY entity issues this primitive to the local MAC entity to confirm that the PHY has reset the CCA state machine after receiving a PHY-CCARESET.request. The effect of receipt of this primitive by the MAC is unspecified.

Semantics:
The semantics of the primitive are as follows: PHY-CCARESET.confirm. This primitive has no parameters.

PHY-CCA.indication

Function and use:
This primitive is used by the PHY to inform the local MAC entity about the current state of the medium. This primitive is generated as a response to a change in the status of the channel, that is, from channel idle to channel busy or from channel busy to channel idle. The effect of receipt of this primitive by the MAC is unspecified.

Semantics:
The primitive includes the parameter: PHY-CCA.indication (STATE). The STATE parameter value can be BUSY or IDLE. The parameter value is BUSY, if the channel assessment by the PHY determines that the channel is not available, otherwise the value is IDLE.

PHY-RXSTART.indication

Function and use:
The PHY issues the PHY-RXSTART.indication primitive to the local MAC sublayer when the PHY has successfully received a valid header at the start of a new packet. After generating a PHYRXSTART.indication, the PHY is expected to maintain physical medium status as BUSY during the period required by the PHY to transfer a frame of the indicated LENGTH at the indicated DATARATE. The effect of receipt of this primitive by the MAC is unspecified.

Semantics:
The primitive provides the following parameter: PHY-RXSTART.indication (RXVECTOR). The RXVECTOR represents a list of parameters that the PHY provides for the local MAC entity upon receipt of a valid PLCP (physical layer convergence procedure) header. This vector may contain both MAC and MAC management parameters.

PHY-RXEND.indication

Function and use:
This primitive is used by the PHY to inform the local MAC entity that the MPDU currently being received is complete. The success of the MPDU reception is also indicated by the primitive. If RXERROR value is NoError, the MAC uses the PHY-RXEND.Indication as reference for channel access timing. The effect of receipt of this primitive is for the MAC to begin inter-frame space processing.

Semantics:
The primitive provides the following parameter: PHY-RXEND.indication (RXERROR). The RXERROR parameter can include one or more of the following values: NoError, FormatViolation, CarrierLost, or UnsupportedRate. The parameter values returned for each possible error condition are explained in the following:

- NoError signal indicates that no error occurred during the receive process.
- FormatViolation signal indicates that the format of the received PPDU was erroneous.
- CarrierLost signal refers to a situation where the carrier was lost during the reception of the incoming MPDU and no further processing of the MPDU can be done.
- UnsupportedRate signal indicates that an unsupported date rate was detected during the reception of the incoming PPDU (PLCP protocol data unit).

2.2 IEEE 802.16 WiMAX DLL

WiMAX MAC acts as an adaptation layer between PHY and upper layers and also performs data mapping for the data exchanged between them. Service Data Units (SDUs) received from the upper layers are segmented and concatenated into MAC Protocol Data Units (MPDU). MPDUs from upper layers are passed down to the PHY layer with the selected burst profile and power level for transmission.

IEEE 802.16-2004 [9], also referred to fixed WiMAX (IEEE 802.16d), and IEEE 802.16e-2005 [10], referred to Mobile WiMAX (IEEE 802.16e), divide the WiMAX MAC into three distinct sublayers, as illustrated in Figure 4.

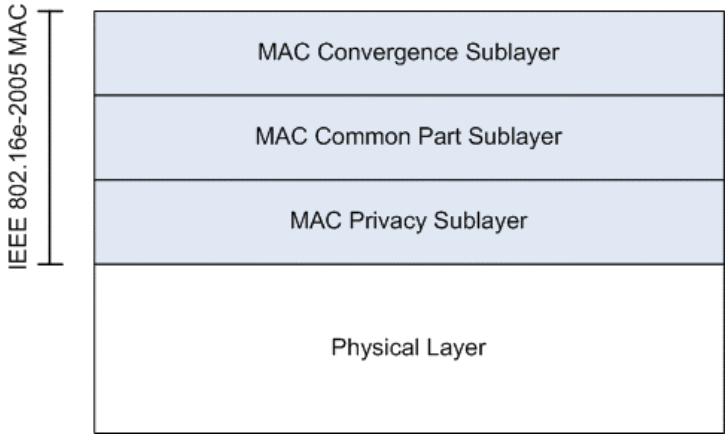


Figure 4 – WiMAX MAC.

The Convergence Sublayer (CS) accepts higher-layer SDUs and performs a classification of them. [10] specifies CSs for the following protocols: ATM, IPv4, IPv6, Ethernet, and Robust Header Compression (RoHC). The WiMAX MAC is connection oriented and uses a unidirectional connection identifier (CID) in order to identify a connection between MN and BS. CS is responsible for the mapping of higher layer addresses (often meaning IP) and the corresponding CID.

In contrast to CS, MAC common part sublayer is independent of the higher layer protocol. Common part sublayer is the basis of WiMAX MAC and handles, for example, packet MPDU encapsulation/decapsulation, packet scheduling, Automatic Repeat-Request (ARQ), bandwidth allocations, modulation, and code rate selection [11]. The supported modulations schemes for data bursts in [10] are QPSK, 16 QAM, and 64 QAM with various error coding rates.

WiMAX MAC layer allows flexible allocation of transmission capacity to different users. Variable size MPDUs can be included into one data burst handed down to the PHY layer. As well, multiple SDUs can be embedded into one MPDU, or an SDU can be fragmented into multiple MPDUs. This way the MAC header overhead can be decreased and the overall performance increased.

WiMAX MAC supports ARQ for improving the reliability. In addition, Hybrid Automatic Repeat-Request (HARQ) scheme is an optional feature in the OFDMA PHY layer of IEEE 802.16e. ARQ brings reliability over the air interface and thus protects against transient signal deteriorations, but WiMAX MAC also provides QoS assurance for MPDUs belonging to the following traffic types:

- Unsolicited Grant Services (UGS) assures bandwidth and maximum delay for delay and jitter sensitive application with fixed size data packets at constant bitrate.
- Real-time Polling Services (rtPS) provides QoS services for real-time applications with variable size data packets, such as video stream.
- Non-real-time Polling Services (nrtPS) is similar to rtPS but does not offer as strict delay variance protection. Thus, nrtPS is reasonable with data transmissions.
- Extended Real-time Variable Rate (ert-VR) guarantees bitrate and delay requirements of applications with variable bitrate.
- Best Effort (BE) provides minimal amount of QoS and data is sent when resources are available.

Each MPDU consists of a 48-bit header followed by a payload and a cyclic redundancy check (CRC) parts. Figure 5 shows DL MAC and UL MAC header generic to each MPDUs containing either management or data.

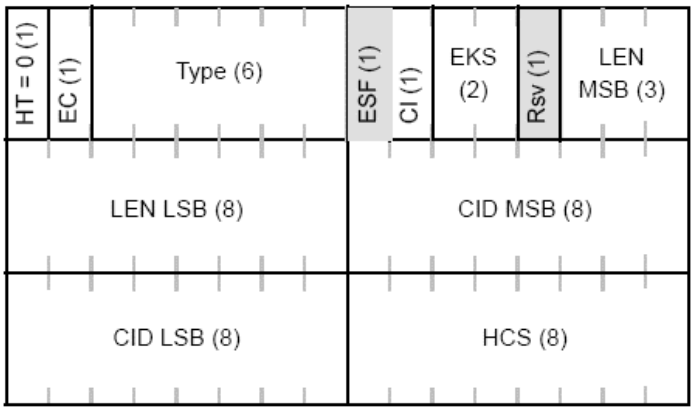


Figure 5 – Generic MAC Header Format (Source: [10]).

Figure 6 illustrates the structure of the type I UL MAC header format for MAC signalling messages sent without payload. This is used by the MN for bandwidth requests, transmission power reports, Carrier-to-Interference-plus-Noise Rate (CINR) reports, Channel-Quality Indicator Channel (CQICH) allocation requests, PHY channel reports, sleep control, and MPDU sequence number report used in ARQ. The BS assigns DL bandwidth to the attached MNs according to the inbound data from the network side. The UL bandwidth is allocated to each MN based on their requests.

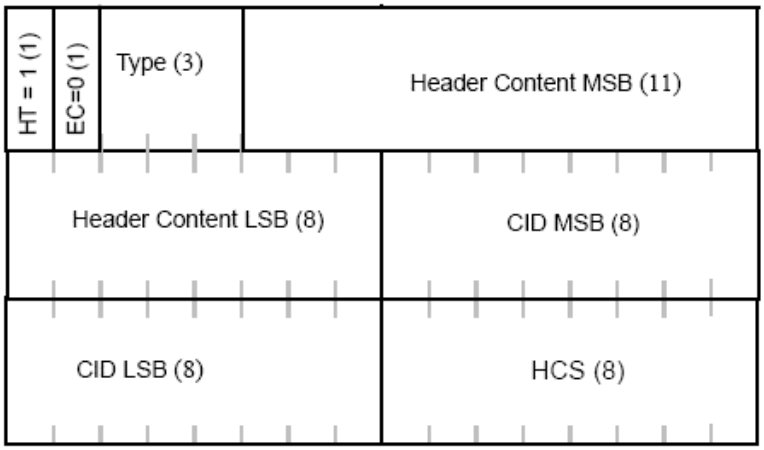


Figure 6 – MAC Signalling Header Type I Format (Source: [10]).

Type II UL MAC header format for MAC signalling is shown in Figure 7. It is used for feedback, for example, on MIMO channel.

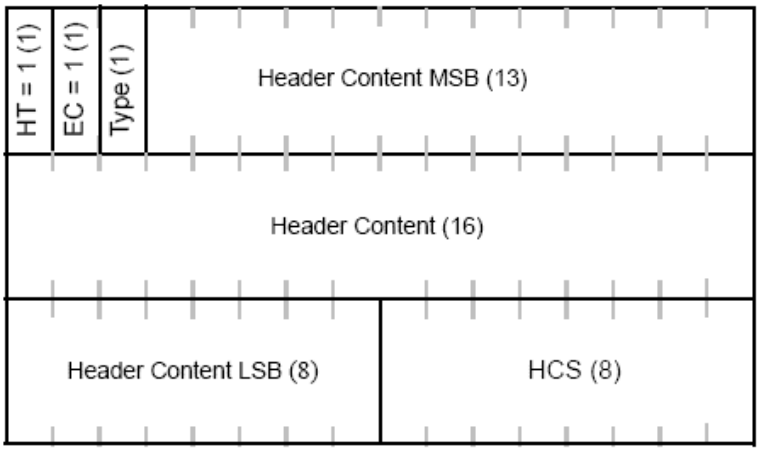


Figure 7 – MAC Signalling Header Type II Format (Source: [10]).

For more detailed information on the header format and for the field values for each message type, refer to [10].

Security sublayer (SS) provides MN privacy, authentication, and confidentiality and network side entities protection against unauthorized access. MPDUs sent over the air interface between MN and BS are encrypted. Security sublayer provides MN and network side key management protocol mechanisms to authenticate each other and to generate and exchange symmetric keys for MPDU encryption. The key management protocol is based on Extensible Authentication Protocol (EAP) or X.509 digital certificates together with RSA.

2.3 Perspectives

The OPTIMIX system requires that the DLL provides the basic mechanisms for L2 data transmission over IEEE 802.11 and IEEE 802.16 links. The implementation of these mechanisms in the simulation environment will not include all the features discussed above, but rather an approximation of them. In addition to the standard-specific features of the WLAN and WiMAX DLLs, additional requirements are introduced to the DLL by the OPTIMIX system. These requirements are discussed in this section. An overview of the DLL simulation model is given in Section 7.1.

In the OPTIMIX system, DLL interfaces with the IEEE 802.21 MIH Function and the Mobile Observer running Triggering Engine functions, as illustrated in Figure 5. The specification of these interfaces (i.e. MIH_LINK_SAP, MIH_NET_SAP, and TRG) is presented in [12], and the DLL implementation is required to support them. These interfaces provide internal, end-to-end event reporting and dynamic link condition monitoring. At the sender side, the data stream travels from the network layer and RoCH entities to the DLL, which then processes the packets and forwards them to the physical layer for transmission over the communications medium. At the receiver the process is reversed.

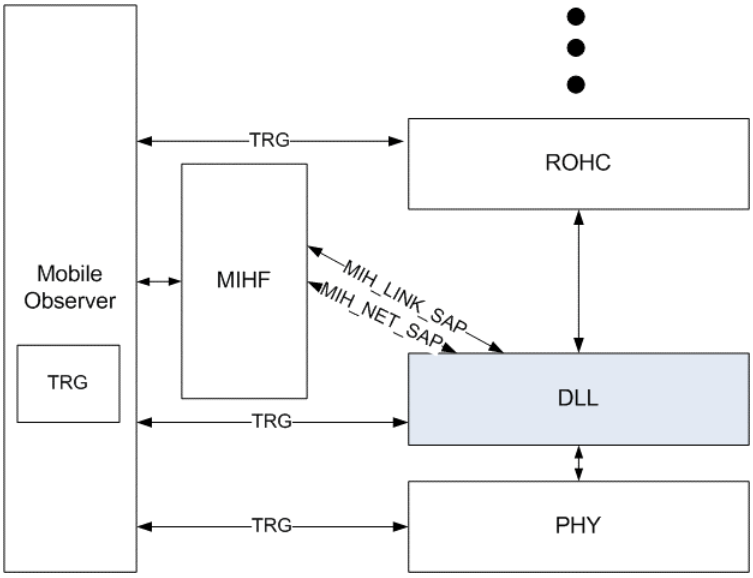


Figure 8 – DLL module placement within the OPTIMIX system.

In OPTIMIX system, the DLL is also required to support passing of corrupted payload data to higher layers for processing, as this is required for multimedia data transmission. The OPTIMIX system requires that the DLL is capable of passing corrupted payload data to higher layers for processing in the receiver. This means that corrupted data packets should not be dropped by the DLL module. Instead, damaged packets are sent to upper layers (network & transport layers) and finally to the application layer. To achieve this operation, partial checksum or no checksum at all needs to be used at the DLL, provided that partial checksum is used at the transport layer of the simulation chain.

3 Data Link Layer Multicast

In the case of the current 802.11 standard the multicast implementation is not multimedia friendly at all. The problems are that there is no acknowledgement on the reception and the transport speed is limited do the base rates (1 or 2 Mbps). Many researchers already investigated this problem and tried to propose a solution for that. Here we summarize all the different approaches that handle the multimedia multicast situation.

3.1 Leader Based Protocol (LBP)

This protocol is proposed by Joy Kuri and Sneha Kumar Kasera in their Reliable Multicast in Multi-access Wireless LANs publication in 1999. This could be the oldest proposal that tries to fix the problem of the missing acknowledgements utilizing other 802.11 features.

The protocol is based on RTS and CTS (Ready to Send and Clear to Send) messages. The 802.11 protocol is modified so that these two messages could be used for multicast purposes as well. Furthermore a *leader* is elected in the multicast group whose role is to send feedbacks to the access point about frame reception. When all the frames have arrived successfully, then the leader sends an ACK (Acknowledgement) message, otherwise it sends a NACK (Negative ACK) message. All the others signal only the missing frames via NACK messages. When the Access Point receives an ACK from the leader, and nothing else, then the transmission was correct. In case of any NACK messages the Access Point repeats the transmission.

First the Access Point sends a CTS message to the multicast group signalling that it would like to send multicast data. This also allocates the channel for the time of the transmission. This procedure is shown in Figure 9.

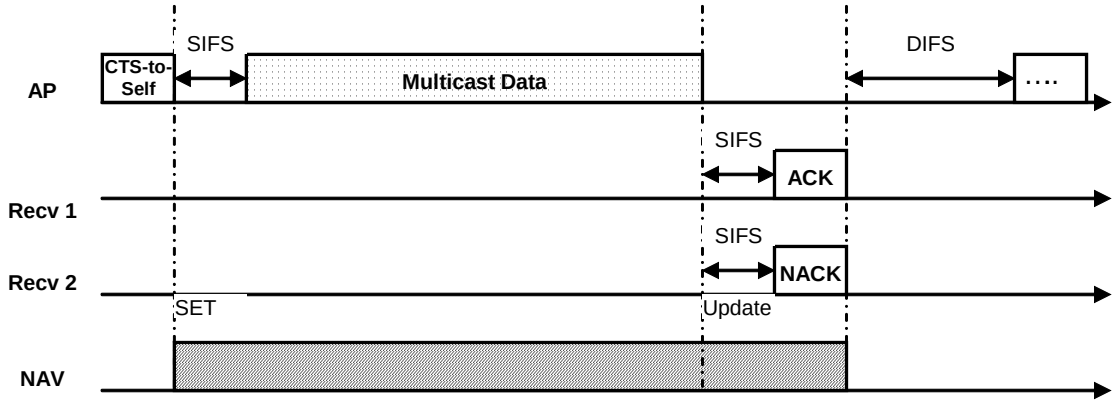


Figure 9 - LBP message timing.

The Access Point stores the addresses of the leaders of each multicast group. When a new receiver would like to join the group then the access point looks up the leader of the group first. If there is no leader yet for a certain group, then the newcomer receiver will be the leader. If, however, a leader exists then the request is acknowledged only. When the leader leaves the group and there are receivers that still would like to receive the multicast flow, then they must rejoin to the same group.

The drawbacks are that this approach does not solve the data rate problem. The leader election is not defined. And nodes that do not receive the RTS and CTS may cause interference.

3.2 Broadcast Medium Window (BMW)

This solution is provided by Ken Tang and Mario Gerla in the publication named MAC Reliable Broadcast in ad-hoc Networks. The main idea of the protocol is that the neighbouring nodes send frames to each other using a Round Robin like scheduling. To achieve this functionality each node has to maintain three lists: the list of the neighbours and the lists of the sent and received frame sequence numbers. The nodes always know who their neighbours are. Receiving a frame makes the sender node to be listed. However if there is no frame from a neighbouring node for a long time, then this node is deleted from the list. For the neighbour discovery nodes periodically send Hello messages.

During transmission the sender first sends an RTS message specifying the sequence numbers that it would like to send. The receiver replies with a CTS message and specifies all those sequence numbers that are not received by itself yet. Upon receipt of the CTS, the sender sends all the frames that are requested by the receiver. If the reception is successful, the receiver sends an ACK message. This procedure is repeated between the neighbours until all the nodes

have all the frames. The procedure is reinitialized when there is a new frame ready to be sent for the group. If the number of unsuccessful sending exceeds a certain threshold then the sender deletes the neighbour from the list.

The algorithm implements a feedback system by which the lost or corrupted frames can be reported. The drawback is that the transmission speed cannot be modified. Multiple CTS replies for a given RTS message can cause problems as well.

3.3 Batch Mode Multicast MAC Protocol (BMMM)

This work is provided by Min-Te Sun, Lifei Huang, Anish Arora and Ten-Hwang Lai in their Reliable MAC Layer Multicast in IEEE 802.11Wireless Networks publication. They further developed the BMW algorithm introducing a RAK (request for ACK) message. This message is for avoiding frame collision and idle channel problems.

Here the sender first waits for the channel to be clean and gets it. At the beginning of the transmission it sends an RTS message specifying all the group members and an interval value. Upon receiving the CTS message, the data transmission begins. At the end of the transmission the RAK message is sent out to each member waiting for an ACK message.

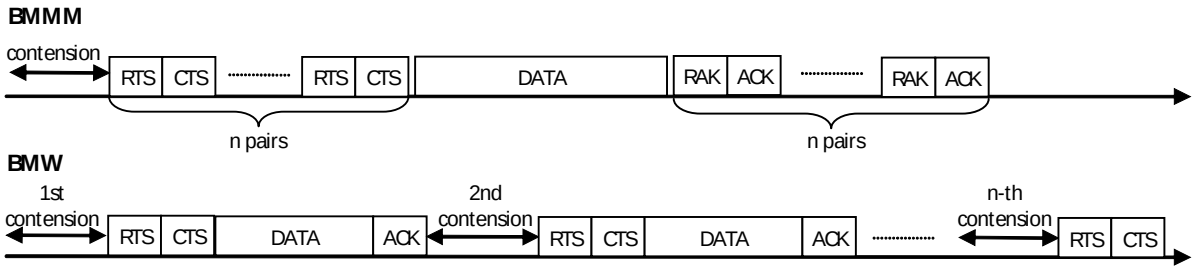


Figure 10 – BMMM protocol.

The main advantage of the BMMM protocol is that it reduces the number of concurrent states which means time sparing. The reduced time consumption remains significant despite of the introduction of a new message (RAK). The method is compatible with other MAC protocols, too.

The drawback is that the transmission speed cannot be modified. Moreover, in heavily congested networks the amount of messages degrades the performance.

3.4 Broadcast Support Multiple Access (BSMA)

This is a new work from the authors Ken Tang and Mario Gerla. They published this protocol in their Random Access MAC for Efficient Broadcast Support in Ad Hoc Networks work. This protocol is supposed to overcome the multicast frame collision problem by using RTS, CTS and ACK messages.

In a DCF (Distributed Coordination Function) system, all nodes can get the channel with equal probability. When the channel is clean then the sender is able to transmit. Here the sender sends an RTS message and sets up a timer to wait for CTS messages. The receivers send CTS messages and set up a timer to receive the data. Receiving the CTSes the sender sets up a new timer to wait for the NAK messages. If no NAK message is received the sender declares the transmission to be successful, and prepares for sending the next piece of data.

The method provides a coordination scheme that prevents collision. Also, it reduces the number of messages in the network. The drawback is that the transmission speed cannot be modified.

3.5 HIMAC

This algorithm is provided by Ai Chen, Dongwook Lee, Gayathri Chandrasekaran and Prasun Sinha in their High Throughput MAC Layer Multicasting in Wireless Networks publication. The protocol focuses on two problems: creating a good feedback system and providing means for the transmission speed modification. Also, it handles the interference problem caused by broadcast messages. There are two mechanisms inside: Unary Channel Feedback (UCF) and Unary Negative Feedback (UNF). These are simple signals that contain the information in their length. The good thing is that a node can receive multiple of such signals, while in the previous cases only one signal could be received at a certain time.

Whenever the sender would like to transmit an information piece to the neighbours, it should reserve the channel first. After successful reservation, it sends an RTS message with the list of the group members. In the case of a receiver, it

should reply with an UCF message. The UCF length determines the best data rate. The shortest signal means the best available data rate, while the longest means the slowest one. In the case where there is no UCF received, then the procedure is repeated. The UNF could further improve the performance of this algorithm. If no UCF is received, then the procedure is repeated. The UNF could further improve the performance of this algorithm. If interference occurs, the receiver may already know the content of the frame, so it may decide that the retransmission of the frame is not necessary. Figure 11 and Figure 12show the UCF and UNF messages.

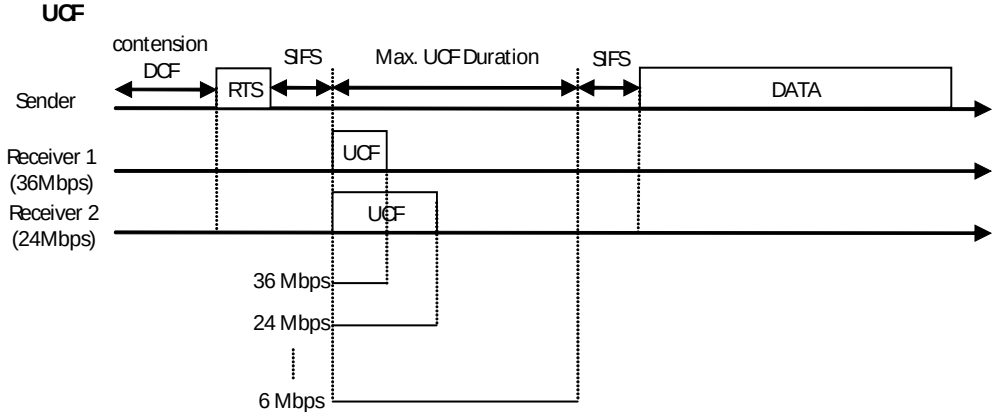


Figure 11 - UCF message.

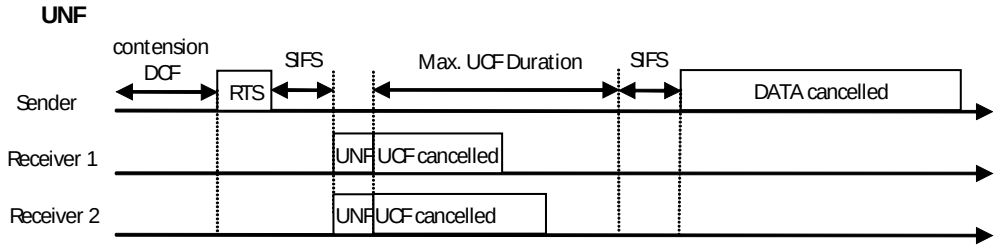


Figure 12 – UNF message.

Through the reduction of collisions and unwanted repetitions this feedback mechanism results in a good performance, and also allows for an increase in transmission speed.

3.6 RMAC

This approach is proposed by Weisheng Si and Chengzhi Li in their A Reliable Multicast MAC Protocol for Wireless Ad Hoc Networks work. This approach introduces a busy/tone signal to overcome the problems of the multicast data transmission. A peculiarity of this approach is that in order not to disturb the data communication an out-of-band busy/tone signal is used. Two messages are used: a Receiver Busy Tone (RBT) and an Acknowledgment Busy Tone (ABT) message. RBT provides solution for the hidden terminal problem and also replaces the Network Allocation Vector (NAV) during the RTS/CTS massages. The ABT message is used to signal successful data transmission. The advantage of the ABT is that there is no processing in the physical layer is needed and there is also no need for message header ABT is as short as possible. However, in order to distinguish the ABT message from one another, there is a new control message called Multicast Request-to-Send (MRTS). The receivers send feedback according to their positions in the MRTS message. Thus only one control message is necessary.

The procedure of the algorithm is the following: the sender sends an MRTS message. When a node receives it, then it looks up itself in the message. If its MAC is listed then it stores the index and sends an RBT message. The sender waits for the RBTs for a certain amount of time then it starts to send the data to the receivers. If the sender receives an ABT, then the transmission was successful. Otherwise an MRTS message is resent.

3.7 802.11MX

This is a work done by S. K. S. Gupta, V. Shankar and S. Lalwani. They published their solution is Reliable Multicast MAC Protocol for Wireless LANs publication. The main idea is analogous with the RMAC protocol. In the abovementioned BMMM and BMW protocols the number of node control message cycles is way too high and it leads to more frequent collisions. This problem is overcome by Busy/Tone signals. The 802.11MX introduces two new messages: NCTS and NAK. NCTS means that the receiver is not ready to receive the data and NAK means that there was an error in the last transmission.

The sender should reserve the channel first and then it should send an RTS message. However, instead of waiting for CTS messages it listens for NCTS signals. If there is no such signal, the sender sends the data. At the end of the transmission the sender waits for NAK messages. If there is no NAK, then the transmission was successful. On the receiving side, when a node gets an RTS message it determines if it is a unicast or a multicast message. In case of unicast transmission the node sends a CTS message. In case of multicast transmission, the node can signal that it is not ready for reception by sending out an NCTS message. If the transmission is erroneous, or there is no transmission at all after a certain time, the node sends the NAK message.

The advantage is the reduced number of collisions and the global handling of unicast and multicast flows. The drawback is the lack of transmission speed modification and the necessity of the new control messages introduction.

3.8 Leader Based Protocol with Auto Rate Fallback

This approach is presented by Sungjoon Choi, Nakjung Choi, Yongho Seok, Taekyoung Kwon and Yanghee Choi in their Leader-based Rate Adaptive Multicasting for Wireless LANs work. The protocol seeks solution for three problems: the already mentioned multicast problems plus the problem of fairness. In certain cases multicast traffic might inhibit unicast traffic. The solution is based on two approaches. A Leader Based Protocol is introduced to handle the multicast issue while the Auto rate Fallback handles the fairness. The LBP is already described above.

The Auto Rate fallback mechanism maintains a timer and a list to record the actual transmission speed and the number of successful and unsuccessful transmissions. The sender checks the list after each transmission and decides to increase, decrease or to keep the speed. After 10 successful transmissions the speed is increased. After two unsuccessful transmissions the speed is decreased. After an increase and an immediate unsuccessful transmission the speed is restored to its previous value immediately.

3.9 Receiver Based Auto Rate (RBAR)

This approach is similar to the Auto Rate Fallback mechanism. The difference is that the speed is determined by the receiver and not the sender. The receiver analyzes the channel and sets the highest data rate that is possible for itself.

3.10 Opportunistic Auto-Rate (OAR)

This solution is published by B. Sadeghi, V. Kanodia, A. Sabharwal, and E. Knightly in their An Opportunistic Autorate Media Access Protocol for Ad Hoc Networks publication. The solution is similar to the RBAR or the ARF algorithm. The difference is that in the OAR protocol, the chances are equal for everybody. Every node get the same time for sending data, regardless of the speed of the transmission. Thus faster receivers may have higher traffic.

3.11 Summary of the Protocols

Some of the protocols deal only with the feedback mechanism of the multicast transmission. These are the LBP, BMW, BMMM, BSMA, RMAC, 802.11MX protocols. The HIMAC is outstanding as it provides both feedback and speed adaptation. Some of the protocols use new control messages and thus they could be introduced quickly into the current systems. However the collision of the control messages is a new source of problems. The great amount of control messages degrades the performance of the network. The RMAC and 802.11MX use busy/tone signals instead of control messages thus reducing the number and the collisions of control messages. The busy/tone signals carry no further information. The problem is that they require a separate channel and therefore cannot be easily introduced to the current system.

The LBP-ARF, RBAR, OAR protocols form the other group as they utilize other protocols to adapt the transmission speed.

4 Queue System and Multiuser Scheduling

The multiuser scheduling algorithm included in the BS Controller (see [14], section 3.2) relies on the knowledge of some specific information about the packets to be transmitted through the air interface, e.g. the total time spent in queue. Its working principles are very general and may be applied to serve transmission buffers differently located within the protocol stack. In the current approach we assume to have a system of buffers within the DLL, containing the video data destined to the different users. This queue system logically follows the SVC adaptation eventually performed at the Application Layer and the multicast processing at Network and DL layers.

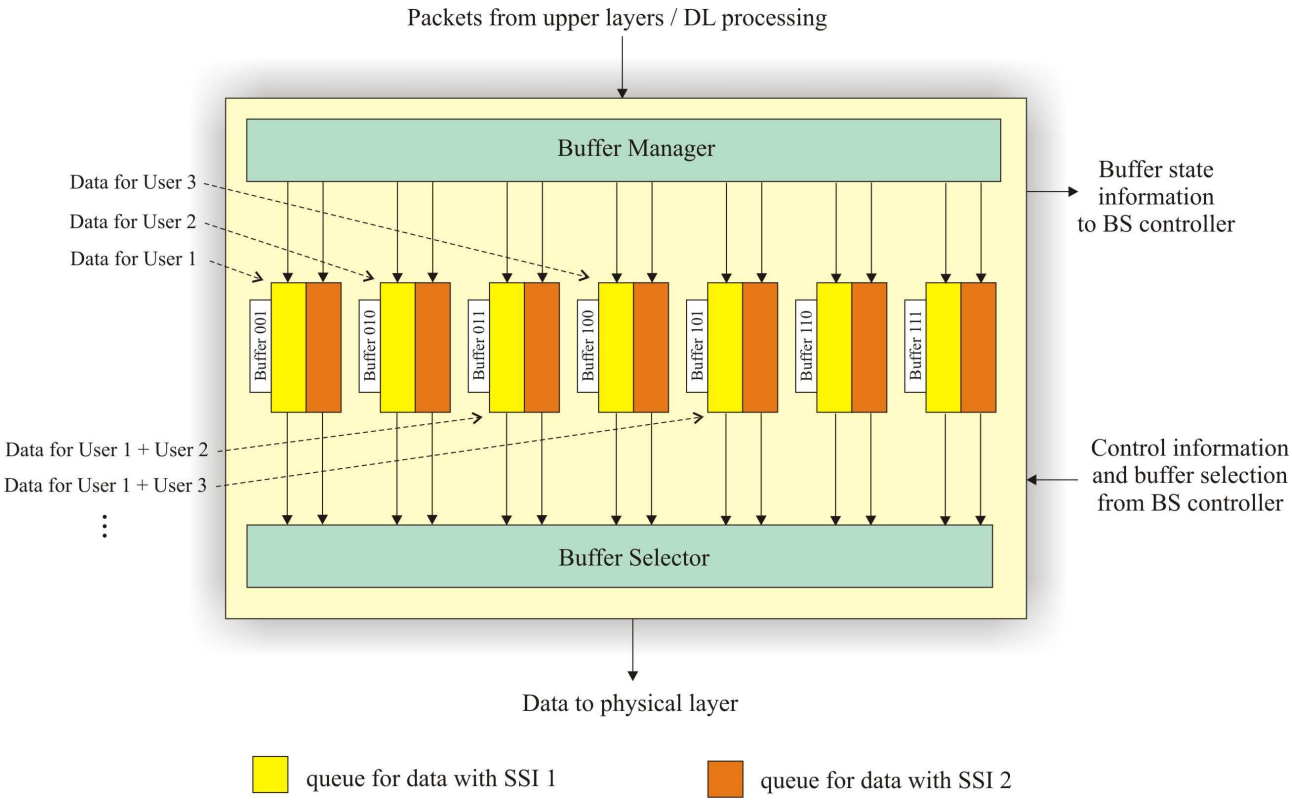


Figure 13 – Example of DLL queue system for 3 users and 2 SSI data class.

As depicted in the Figure 13, data destined to different users are kept logically separated (i.e. in different buffers). Moreover, different queues for each user permit to distinguish packets characterized by different SSI. In this way each single queue is associated to a particular SSI class and contains data for one particular user. Moreover, DL multicast groups can be seen as virtual macro-users, subject to the same treatment of standard single users. In Figure 13 each buffer has been indicated with a binary string allowing to easily identify the receiver/group of receivers to which the corresponding data have to be sent (see the position of the 'ones' in the string). With N_{SSI} different data classes and N_U distinct users, the maximum number of buffers in the DL queue system is given by

$$N_{buf,\max} = N_{SSI} \cdot \left(2^{N_U} - 1\right)$$

Clearly, this number refers to the maximum theorical value, including all possible combinations, while in practical situations a significantly lower number of buffer elements is sufficient and a dynamical allocation of the buffers is advisable.

The multiuser scheduling algorithm selects the queues from which drawing the packets according to a given priority function, based on SSI values, CSI, total time spent in buffer, quality requirements, etc. (see [14], section 3.2.2). Then the selected data are sent to the Physical Layer module, where they are channel encoded, modulated and assembled into the transmission frame. The DLL buffer system state is kept monitored, in order to avoid any packet dropping at this level: thus, proper information is sent to the preceding stream adaptation module and, in case its optimisation process is not enough, to the Master Application Controller, which may be able to significantly change the encoding parameters and, as a consequence, the bitstream characteristics.

Finally, we point out that, in some particular scenarios and according to the adaptation policy implemented by BS controller, some sort of multicast rearrangement may occur also on the data drawn from the buffers.

5 RoHC for DCCP

When considering multimedia streaming applications over IP networks coupled to wireless transmissions, the problem of bandwidth cost is crucial. Indeed, network packetization adds non negligible protocol overhead to the useful video data, therefore introducing a lot of redundancy that is not a concern for wired IP links but is crucial for constrained bandwidth wireless links. An interesting approach to cope with this problem is to process network headers in order to decrease their inner redundancy. The potential gain achieved in this way can be used for a target transmission rate on the application side to decrease the constraint on the video source bit rate, allowing then a better end to end final video quality. In this section we describe a proposal solution of header compression for DCCP/IPv6 profile being compliant to RoHC standard described in the following.

5.1 Interest and overview of network header compression

The issue of IP protocol performances on constrained bandwidth links has started as early as 1984 with Thinwire protocol [4] and in 1990 with Van Jacobson [5] concerning TCP/IPv4 compression mechanisms based on information redundancy. Several proposals have followed including CTCP [6] compression handling several IPv6 streams and CRTP [7] extending the previous principal to compress RTP headers to finally achieve IETF's standardisation for multimedia streams header compression RoHC (Robust Header Compression) defined in RFC 3095 [8].

The general scheme depicted in RoHC standard, as by most other header compression techniques, is illustrated in **Figure 14**. The main characteristics of the protocol are:

- a classification of the network header fields according to their variation profile as i) INFERRED (derived from other parameters), ii) STATIC (expected not to vary during the lifetime of the point to point connection), iii) SEMISTATIC (fields changing according to a known profile of variation) or iv) IRREGULAR (fields changing in an unpredictable manner);
- the storage both at sender and receiver side of a context with the STATIC fields, thus enabling the mechanisms to remove redundancy inside one packet or between several consecutive packets. A key issue for the use of RoHC over error prone channels is the ability to keep those contexts synchronised despite of errors occurring during the transmission;
- compression algorithms used to process semi-static fields as the RTP sequence number or timestamp and improve overall compression efficiency;
- robustness tools allowing the user to statically or dynamically adapt some parameters to the channel conditions allowing the compression objective to remain compatible with transmission errors;
- separation of protocol handling through the use of compression profiles.

The interest in compressing network headers is even higher when considering multimedia application over IPv6 networks for which the protocol overhead can be as large as 60 bytes. With typical encapsulated payloads of 188 bytes (e.g. for MPEG-TS standard), the ratio between network header and video data can be as high as 25%. Typical performance achieved by RoHC protocol shows an average of 5 bytes for compressed headers when using RTP/UDP(-lite)/IPv6 profile in Unidirectional mode.

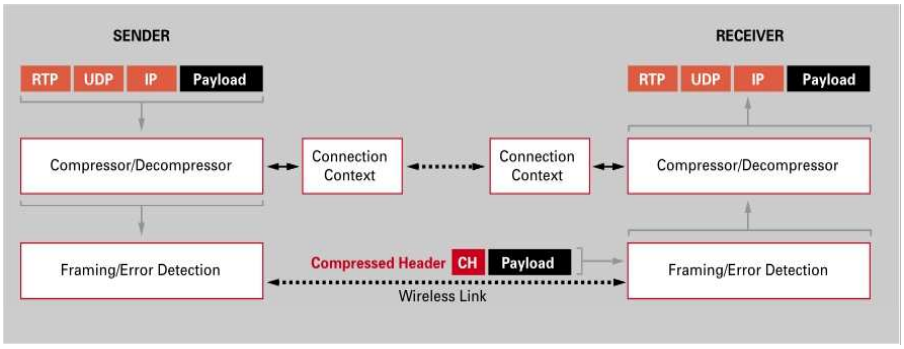


Figure 14 – General header compression scheme.

The case of application of RoHC to video over IP streaming has already been subject of studies also coupled to the use of UDP-lite transport protocol of high interest for error resilient video codecs. Real improvements have been achieved, for example, in IST PHOENIX project using robust header compression associated with video coding standards H.264 and MPEG 4.

5.2 The DCCP protocol

For multimedia applications, the perspectives introduced by new IETF transport protocol DCCP are of real interest. Indeed DCCP helps improving the mechanisms already provided by UDP and UDP-lite for audio and video streaming over IP by adding control congestion mechanisms.

The results achieved both in terms of compression gain and robustness enhancement, due to the use of Robust Header Compression over traditional transport protocols UDP and UDP-lite, obviously open a path to an evolution of RoHC protocol application to new network and transport protocols. Furthermore, as detailed in the following paragraph, DCCP's additional features (including CCID handling) introduce an overhead cost which is even more important than compared to UDP/UDP-lite. Typically, a DCCP header is 12 to 16 bytes long and can increase much more when extensions are used.

In opposition to UDP which has a simple unique packet format including an 8 bytes long header, DCCP provides several types of packets as described in next table:

Type of packet	Role	Minimum Header size (bytes)
DCCP-Request	Sent by the client to initialize the connection	20
DCCP-Response	Sent by the server in response to a DCCP-request packet.	28
DCCP-Data	Used to transmit application data. DCCP uses sequence numbers to deal with reordering but does not involve retransmission mechanisms as it is a non-connected protocol.	12-16
DCCP-Ack	Used to send acknowledgement packets	12-24
DCCP-DataAck	Used to send at the same time acknowledgement and data packets.	12-24
DCCP-CloseReq	Sent by the server as a request towards the client to close the connexion	24
DCCP-Close	Sent by the client or server to close the connection	24
DCCP-Sync **DCCP-SyncAck**	These 2 packets deal with improving the validity and synchronisation of the connection.	24

Table 4 – DCCP packets.

Each DCCP packet header includes at least a generic header whose size varies from 12 to 16 bytes, and extension header are often used thus increasing again the protocol overhead. Compared to UDP header including at most 8 bytes, it seems obvious that header compression mechanisms are even more interesting in the case of DCCP in order to limit the protocol overhead and optimize the bandwidth resources. On the other hand, the use of multiple packet types and bidirectional transport link will complexify RoHC mechanisms originally designed for unidirectional and single packet protocols and IETF RoHC working group has not published yet a new profile concerning DCCP.

5.3 Classification of DCCP header fields

In this paragraph we propose a classification of DCCP header fields which will be used as a base to extend RoHC algorithms to DCCP compression. Each DCCP packet type is built from the same general structure composed by:

- A generic header, common to all packets
- Additional subheaders, specific to each type of packet
- An option sub header with a size multiple of 32 bits
- The application payload (including RTP header if used)

The generic header can be 12 to 16 bytes long as depicted on Figure 15 and Figure 16:

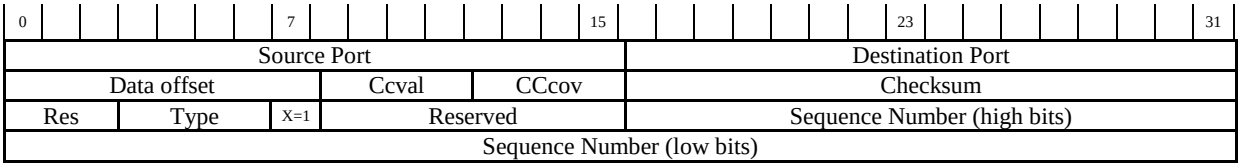


Figure 15- DCCP Generic header (12 bytes)

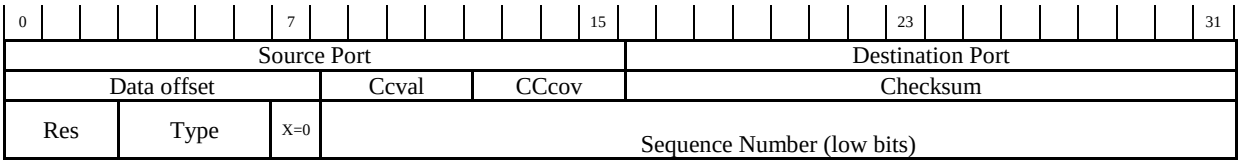


Figure 16 – DCCP Generic header (16 bytes).

The sequence number length depends on the X bit field.

The header includes the following fields:

Source and destination ports:
These fields are similar to UDP and UDP-lite ports and are used to define the point to point connection between two entities. They are constant during the lifetime of the connection and are consequently classified as STATIC for RoHC compression.

Data Offset:
This field defines the length of the DCCP header, it only changes when there is a different type of DCCP packet used or a change in the X bit of the generic header. This field will be classified as DYNAMIC because it can not be inferred from other values .

CCVal:
This field is only used in specific cases of use for control congestion, it is generally set to zero, and so will be classified as INFERRED.

CsCov:
This field defines the portion of the DCCP packet covered by the checksum in a similar way of what is done in UDP-lite transport. This field will follow the compression strategy defines for UDP-lite RoHC profile, it can be either classified as INFERRED when the whole packet is covered, DYNAMIC when only the header is covered or CHANGING when a portion of the packet is covered.

Checksum:
This is the DCCP checksum calculated over the CsCov part of the packet. It has not a predictable way of variation and will be classified as CHANGING.

Res & Reserved:
These fields are set to zero and will consequently not be transmitted.

Type:
This field defines the type of DCCP packet sent among the following:

Type	Paquet
0	DCCP-Request
1	DCCP-Response
2	DCCP-Data
3	DCCP-Ack
4	DCCP-DataAck
5	DCCP-CloseReq
6	DCCP-Close
7	DCCP-Reset
8	DCCP-Sync
9	DCCP-SyncAck

10-15 Reserved

This field can change during a connection and will be classified as DYNAMIC

DX:
This field defines the extension of the generic DCCP header. It will be classified as DYNAMIC for compression of data transfer.

Sequence Number:
Each DCCP packet carries a sequence number, enabling the detection and report of packet losses. Sequence number is incremented after each packet, containing data or not. The DCCP sequence number cannot be directly inferred from RTP sequence number, when those two protocols are used together over the same stream. They don't have the same profile of evolution; indeed DCCP SN is incremented for every packet including acknowledgements whereas RTP sequence number is only used for data. Still, DCCP SN will be encoded with the same algorithm available in RoHC, though modifications are expected to handle the 48 bits format (whereas 16 bits for RTP).

Once the classification of header fields is obtained, we can define the RoHC packets that will be used for DCCP/IPv6 compression. First of all, the static part of the context will be very similar to UDP/IPv6 case, thus including all fields that are not subject of variation during a client-server connection as indicated on Figure 17.

Static part

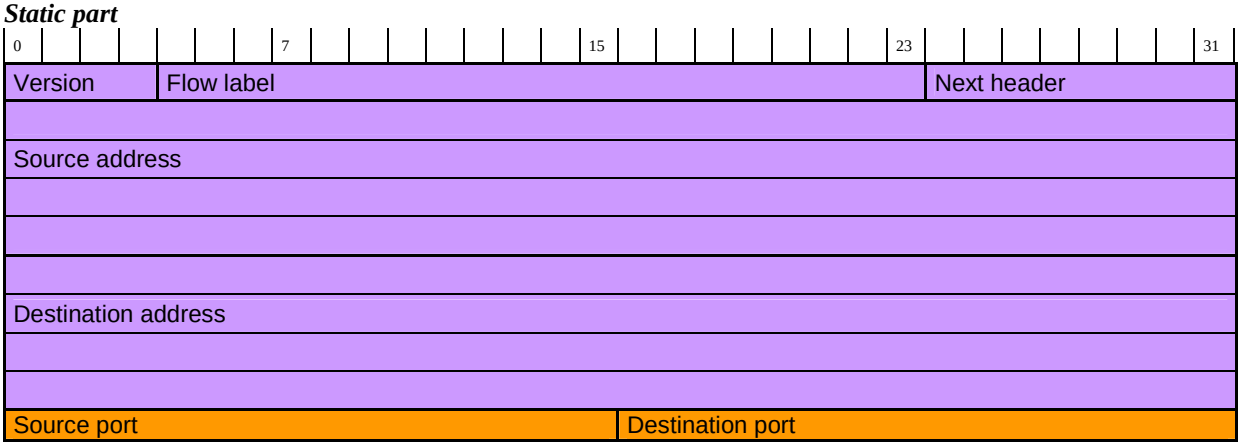


Figure 17 – DCCP/IPv6 static context part (IPv6 fields, DCCP fields).

The dynamic part of the context is depicted on Figure 18. It includes the DCCP dynamic header fields Data Offset, CCVal, CsCov, Type, DX, Sequence number and Checksum together with the IPv6 Traffic Class and Hop Limit fields. The RoHC mode of operation is also included leading to a total of 10 bytes for dynamic part.

Dynamic part

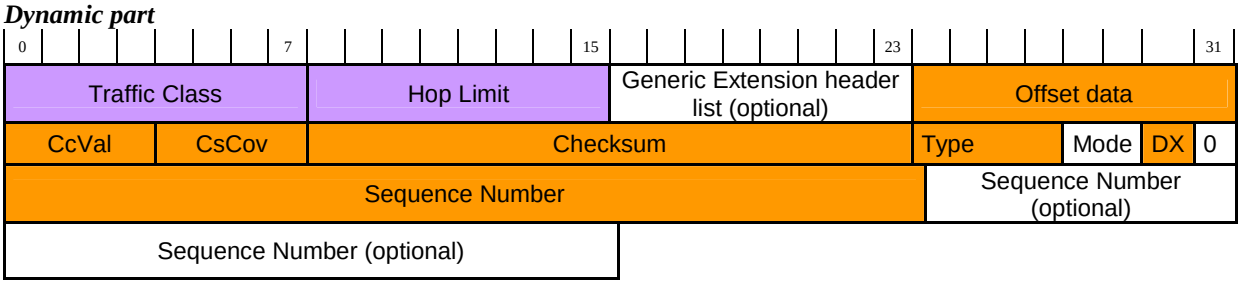


Figure 18 – DCCP/IPv6 dynamic context part (IPv6 fields, DCCP fields).

Once the static and dynamic parts of the context are defined, we can now build the RoHC packets that will be used during the different compression phases and for each mode of operation. The state machine of RoHC Compressor is reported in **Figure 19**.

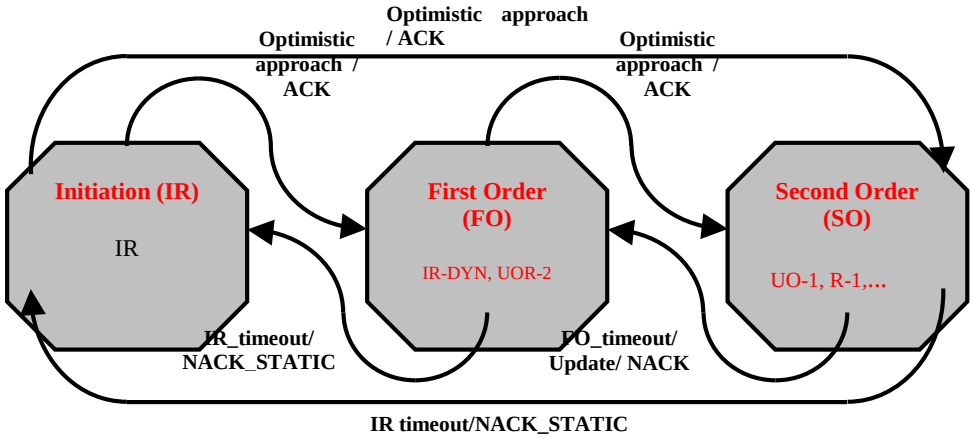


Figure 19 – Compression level for each mode of operation.

The RoHC compressor operates in 3 states: Initialization and Refresh (IR), First Order (FO) and Second Order (SO). The states describe the increasing level of confidence about the correctness of the context at the decompressor side. This confidence is reflected in the increasing compression of packet headers. In case of error conditions, indicated by the decompressor using feedback packets, the compressor can move to a lower compression state. However, the compressor can also periodically move to a lower state of operation: the IR TIMEOUT is used to come back to IR compression level, the FO TIMEOUT is used to come back to FO compression level.

In the IR (Initialization & Refresh) phase, the packets structure will be similar to UDP/IPv6 profile, with the previous modifications of static and dynamic contexts included. This leads to the following IR and IR_DYN packets:

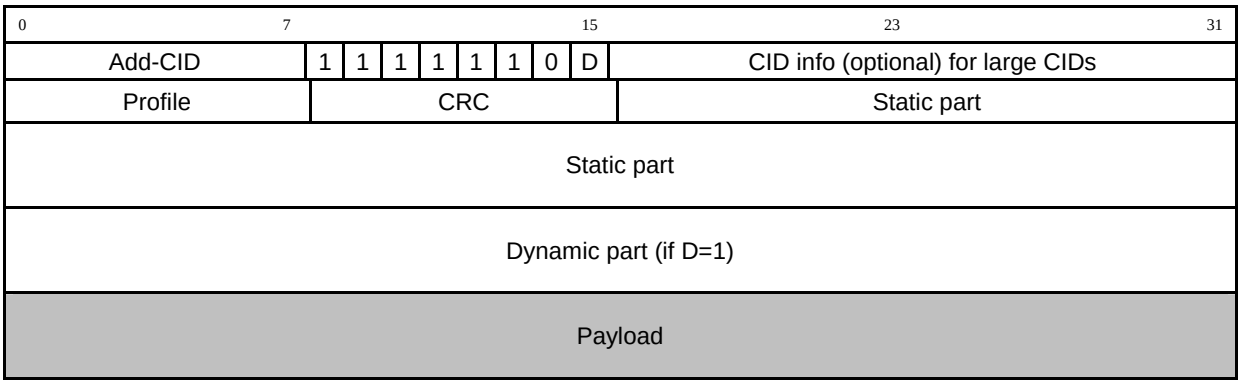


Figure 20 – IR packet.

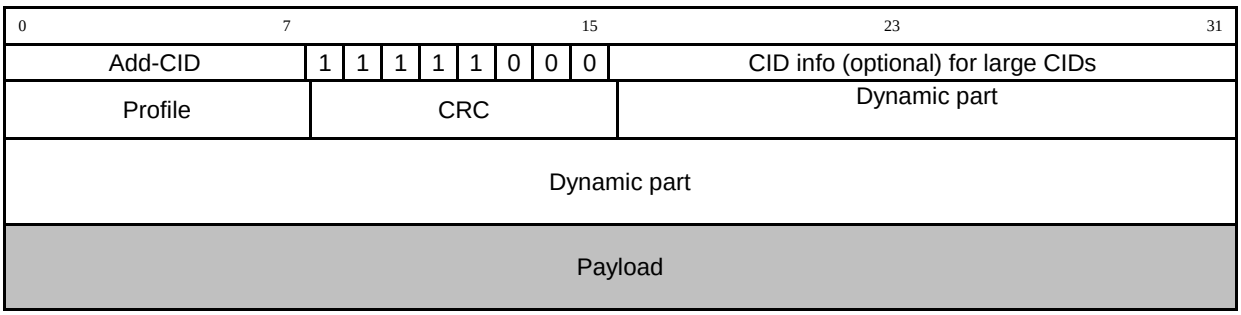
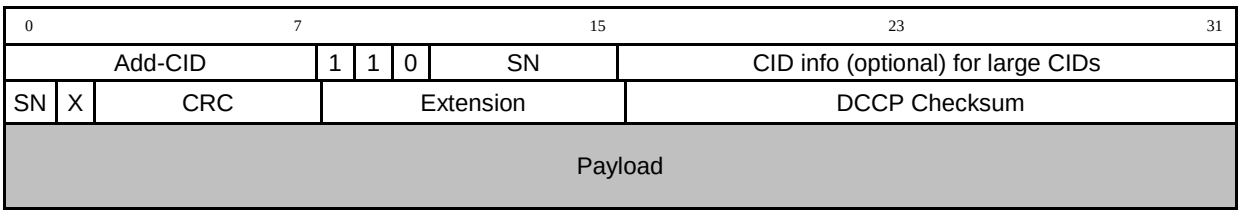


Figure 21 – IR-DYN packet.

The first order (FO) state of the compressor defines an intermediate compression state in which the packet type used is common to all modes of operation, it is called UOR-2 packet. The DCCP sequence number is encoded over 6 bits using W-LSB algorithm which is similar to RTP Sequence Number compression and the CRC covering the packet is reduced to 6 bits (compared to 7 for RTP profile) to remain byte-aligned. The UOR-2 packet extensions provided for RTP profile are used mostly to compress the Timestamp on a larger scale, when DCCP is only used, those extensions become useless, though a one byte extension is still proposed to extend the SN compression to 9 bits as suggested by the standard for RTP profile. The remaining bits left can be used to update the DX and Type fields from DCCP header.



Extension:

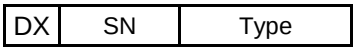


Figure 22 – UOR-2 packet.

The following described packets are used in the Second Order (SO) state which is the highest compression state of the protocol. For each packet, a parallel is done with the RTP/UDP/IPv6 profile described by the standard.

In the RTP/UDP/IPv6 profile description (profile 1), packets UO-1 and UO-0 differed only from the encoded RTP Timestamp length. For DCCP/IPv6 profile, UO-1 and UO-0 will be identical and used only to transmit the Sequence Number encoded over 3 bits and the DCCP Checksum. UO-0 packet header offers a minimum compression size of 4 bytes.

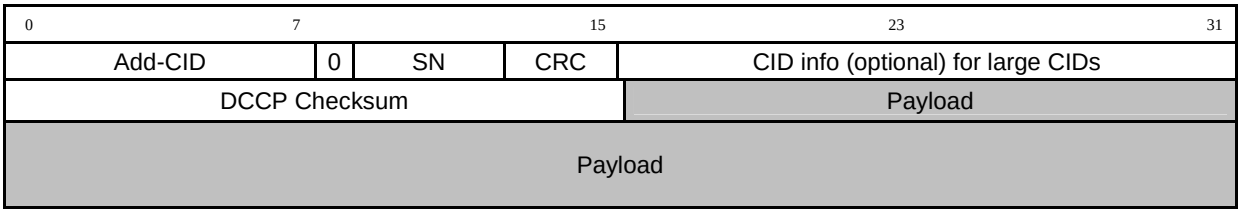


Figure 23 – UO-0 packet.

The R-1 packet structure, used in Bidirectional Reliable mode is quite different from profile 1conception. It does not include extensions that were previously used for RTP timestamp encoding. Furthermore, the sequence number is encoded over 9 bits, and the DX and Type fields are updated. These choices are made to meet both compression efficiency and robustness requirements.

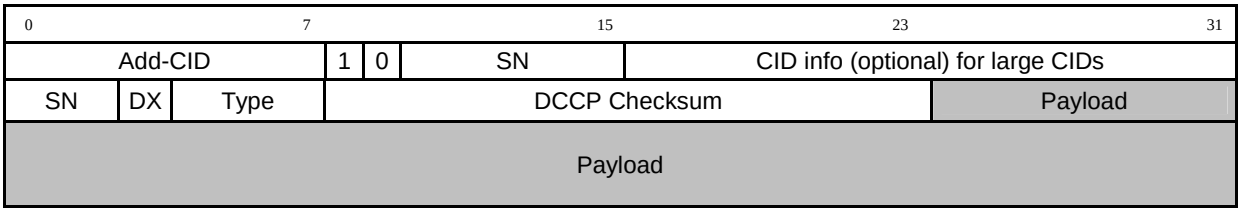


Figure 24 – R-1 packet.

Finally, the R-0 packet and its CRC covered version remain similar to profile 1 as they only transmit the compressed sequence number.

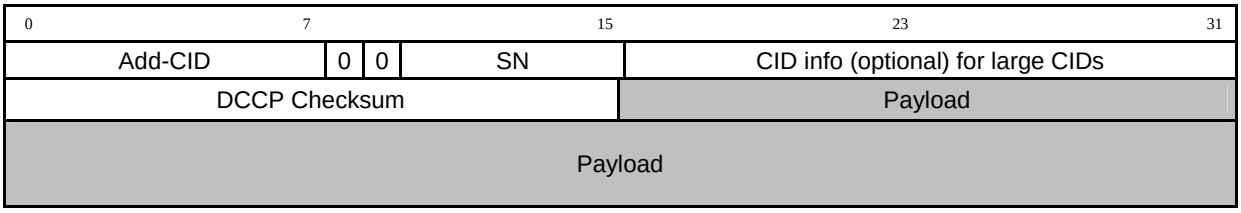


Figure 25 – R-0 packet.

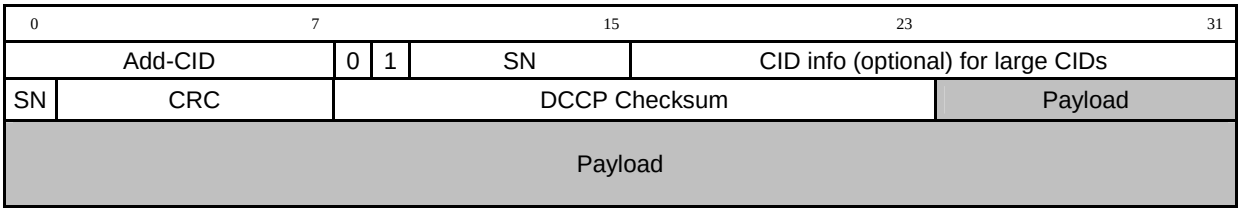


Figure 26 – R-0-CRC packet.

The results obtained with the definition of RoHC compressed packets for DCCP/IPv6 profile are recalled in the next table showing the coherence between packet header sizes and compression level which they belong to. In the highest level of compression , the header size is decreased up to 90% of its original value.

RoHC packet types	Sizes (bytes)
DCCP/IPv6 profile	
IR	48 – 61
IR-DYN	14 – 18
UOR-2	5 – 8
UO-1	4 – 6
UO-0	4 – 6
R-1	5 – 7
R-0	4 – 6
R-0-CRC	5 – 7

Table 5 – RoCH compressed packets for DCCP.

Typical application cases are identified and summed up in the next table and the overall compression gain is indicated (considering both headers and useful data).

Application	RoHC profile	Original packet size	RoHC packet size	Compression gain
H.264 video streaming	DCCP/IPv6	240	194	20%
VoIp audio streaming	DCCP/IPv6	88	42	50%

5.4 Performance evaluation

In order to validate the correct behaviour and test the compression and robustness performance of our model, a simple point to point scenario, depicted in Figure 23 is first considered.

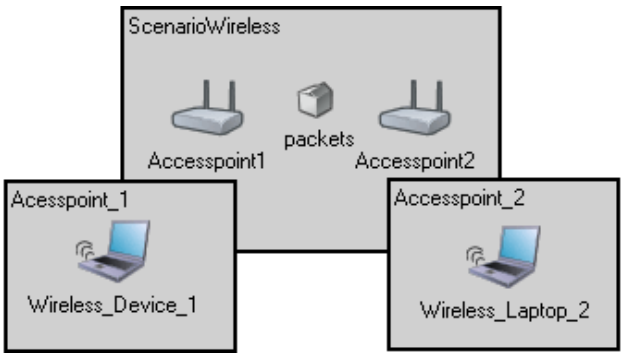


Figure 27 – Point to point scenario

Each wireless device includes the stack described in Sec. 7 and a simulated channel is also included to test the influence of errors behaviour of header compression algorithms.

The parameters characterizing RoHC behavior are as following:

1. L: In U-mode and O-mode the ROHC compressor uses a confidence variable (L) in order to ensure the correct transmission of header information.
2. Timer 1 (IR TIMEOUT): In U-mode, the compressor uses this timer to return to the IR compression level and periodically resends static information.
3. Timer 2 (FO TIMEOUT): The compressor uses this timer in U-mode and to return to FO compression level if the compressor is working in SO compression level.
4. k and n: The decompressor does not assume context damage and stays in the current state until k packets arrive with error in the last n packets. The k1, n1 values are used to assume dynamic context damage and k2, n2 to

assume static context damage.

Unidirectional Mode
Original DCCP/IPv6 header size = 56 bytes
IR_TIMEOUT = 1000
FO_TIMEOUT = 40
L = 3
$k_1 = 1000$
$n_1 = 20$
$k_2 = 900$
$n_2 = 1000$

The evolution of the compressed header size in Unidirectional mode is depicted in Figure 28. We can observe the major aspects of RoHC compression behaviour:

- The compressor stays in each state for a duration of L packets, this mechanism is referred as optimistic approach, the parameter L being dependant of the lower layers and channel impact.
- Periodically, every IR_TIMEOUT, the whole context is updated by transmitting IR packets. The RoHC state machine transits then to its lower compression level.
- Similarly, every FO_TIMEOUT, the dynamic part of the context is updated thanks to IR-DYN packets transmission.
- The highest compression state is achieved after the sequence of packets (IR-DYN → UOR-2 → UO-1).

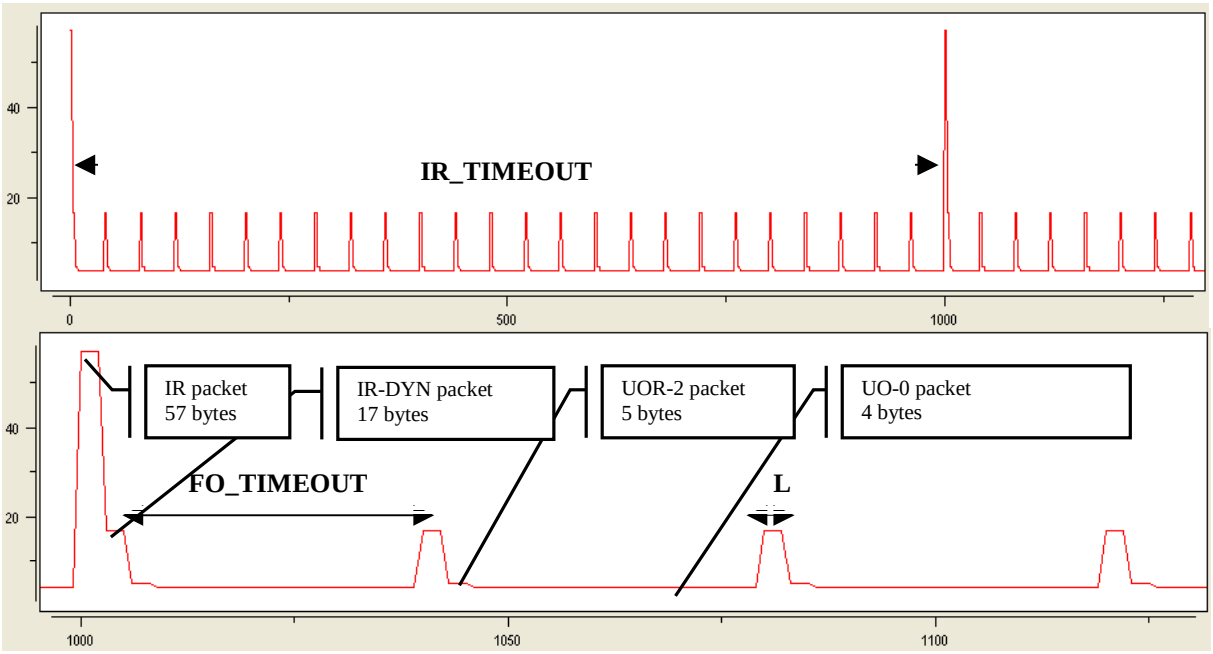


Figure 28 – Compressed header size in U-mode.

Bidirectional Optimistic Mode / Bidirectional Reliable Mode

Original DCCP/IPv6 header size = 56 bytes
IR_TIMEOUT = 1000
FO_TIMEOUT = 40
L = 3
$k_1 = 15$
$n_1 = 20$
$k_2 = 900$
$n_2 = 1000$

In bidirectional optimistic mode, the transition to the highest compression state is quickly achieved and the compressor stays in this state permanently if no NACK feedback is sent by the decompressor to inform that one or several packets were discarded.

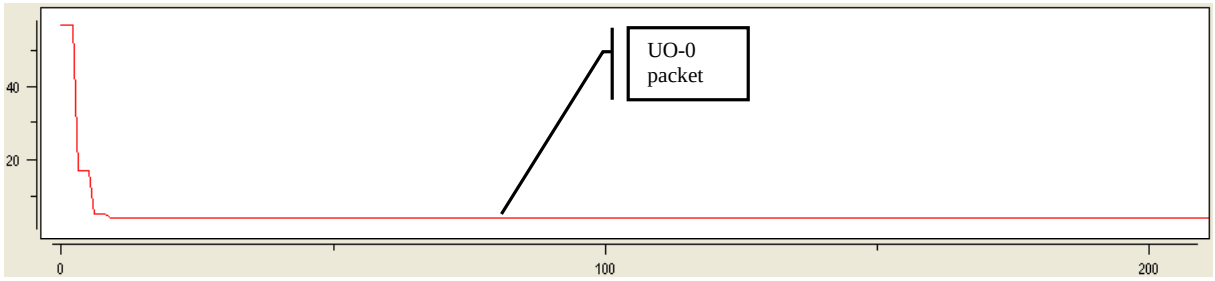


Figure 29 – Compressed header size in O mode.

In Reliable mode, the compression is optimum with a periodic transmission of R-0 and R-0-CRC packets due to the fact not all packets include CRC as feedback is generated for each correctly received packet.

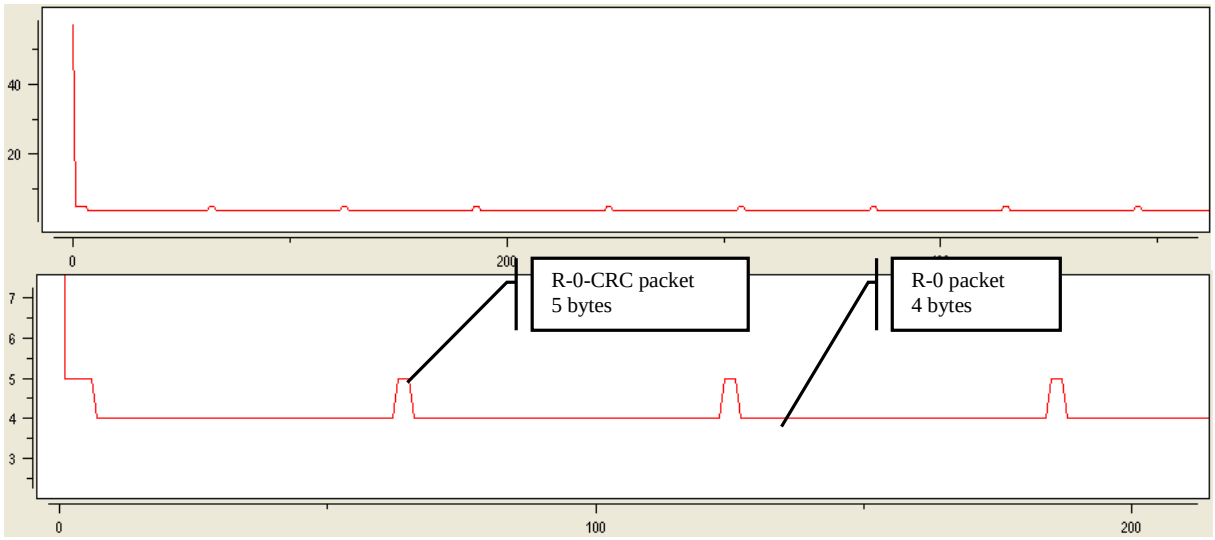


Figure 30 – Compressed header size in R mode.

5.5 Perspectives

We have introduced a new header compression scheme for DCCP/IPv6 profile based on IETF standard RoHC, whose use can be very interesting in the OPTIMIX approach. This work will be extended to include multi-point issues which will have an impact on the RoHC CID handling procedures.

6 IEEE 802.11b/g Data Link Measurement Experiences

We have measured WiFi (more preciously 802.11b/g) traffic and its throughput characteristics in the function of distance and terrain objects. The goal was to establish a model for the throughput characteristics.

6.1 Bit Error Models

In order to describe the model we have considered three options:

- The Gilbert – Elliot model that has two states: good and bad
- The Fritchman model that has more than two states
- And the bipartite model that are special Markov models

6.1.1 The Gilbert – Elliot Model

The Gilbert-Elliot model is the most often used model. The model was constructed in the 60s to describe bit errors. Despite the many years that are already passed it is still applicable to describe current wireless networks. It is a two state Markov chain and allows describing burst errors.

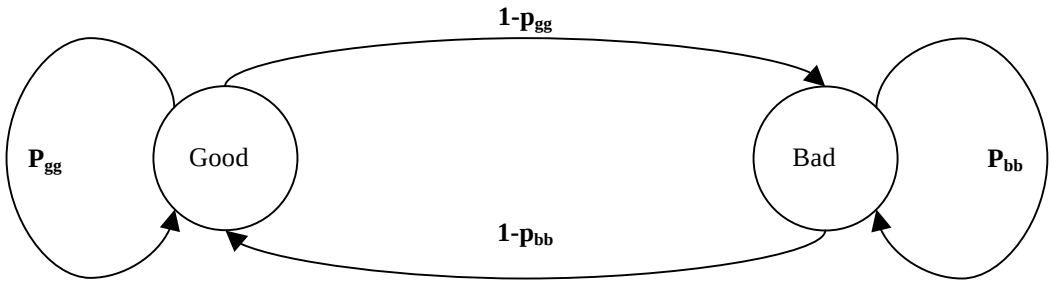


Figure 31 – Gilbert – Elliot model.

6.1.2 The Fritchman Model

The Fritchman model is a generalization of the Gilbert – Elliot model constructed by B. D. Fritchman at the end of the 60s. This is a partitioned, N state Markov chain where the good state of the Gilbert – Elliot model is extended to k number of good states and N-k number of bad states. In this model there are no sate transitions between the good states. EGD is the Error Gap Distribution and used to construct the matrix of the transitions. The values can be determined using measurements.

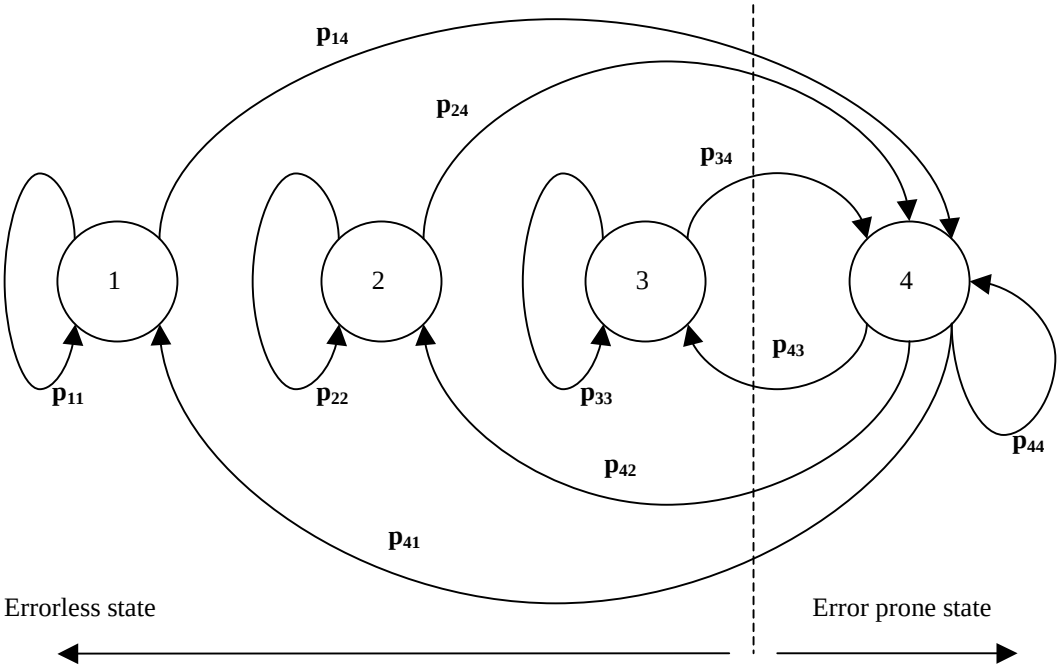


Figure 32 – The Fritchman model.

6.1.3 The Bipartite Model

The bipartite models are further generalized Fritchman models, where there are no transitions between states belonging to the same group (good or bad).

6.2 Measurement Environment

We have modified an access point to provide data for the measurements. In the driver software of the access point we have replaced the original beacon generation procedure and used an alternative one that adjusted the wireless speed of the beacons and created longer beacons also. For this reason we used an ASUS WL-500gP router, but the wireless card of the router was replaced by an Atheros chipset based card. This hardware modification let us to modify the beacons to our needs. The beacons in the tests were longer than the usual one and the speed of the beacons was changed as well. We used 1134 bytes long beacons and the speed was 1, 2, 5.5, 11, 6, 9, 12, 18, 24, 36, 48 and 54 Mbps changing in cycle. The modification of the beacon happened according to the WiFi standard, so the resulted beacons were standard as well. This way the wireless clients can use the access points with no modifications.

To be able to detect corrupt wireless packets we have to use a driver that is able to catch them. The newest Linux wireless driver architecture introduced the mac80211 driver, which is an universal driver for all the wireless cards that support this architecture. These cards only need a thin driver to make a connection between the card and the mac80211 driver and then wireless card is ready to use. Nowadays, more and more cards become available through this architecture. One of the benefits of the architecture is that the driver can be changed at one place (the mac80211 module) and the change affects all the supported cards. Moreover, in order to get wireless packets with bit errors, we did not have to change anything, since the mac80211 driver already supports it. Unfortunately, our experiences showed that even through receiving erroneous wireless packet should be available on many models, only those cards are supported perfectly that are using the Zydas 1211 chipset. For this reason we used an SMC EZ Connect 802.11g Wireless USB 2.0 adapter. Figure 33 shows the devices that we used for the tests.

Figure 33 – Access Point and the wireless card.

To host the wireless card we used Dell D800 notebooks. The OS was Linux using the 2.6.27.4 kernel. We run a packet collector software on the notebook. This software made an SQL database record for all the received packets. As we knew all the byte values of the beacons, we also knew what bytes are right and wrong in the transmission. The database record logged the wrong bytes, if there were any, the signal strength of the transmission, the noise strength and the speed of the wireless packet. Lost frames were detected by sequence numbers provided by the beacons.

6.2.1 Measurement Scenarios

We had 8 different measurement scenarios, 5 indoor and 4 outdoor settings. In the indoor scenarios the router and the wireless laptop were placed in different rooms separated by brick walls. In the first scenario they were in the same room, the distance was about 4 meters between them. In the further scenarios they were placed in different rooms and along the increasing distance the number of wall increased as well. In the outdoor measurements the distance of the AP and the measurement laptop were 40, 80 and 130 meters respectively. During the outdoor measurements the weather was a little rainy and foggy. Between the AP and the notebook there were some bushes and trees, there were no line of sight Figure 34 and Figure 35 show the measurement scenarios.

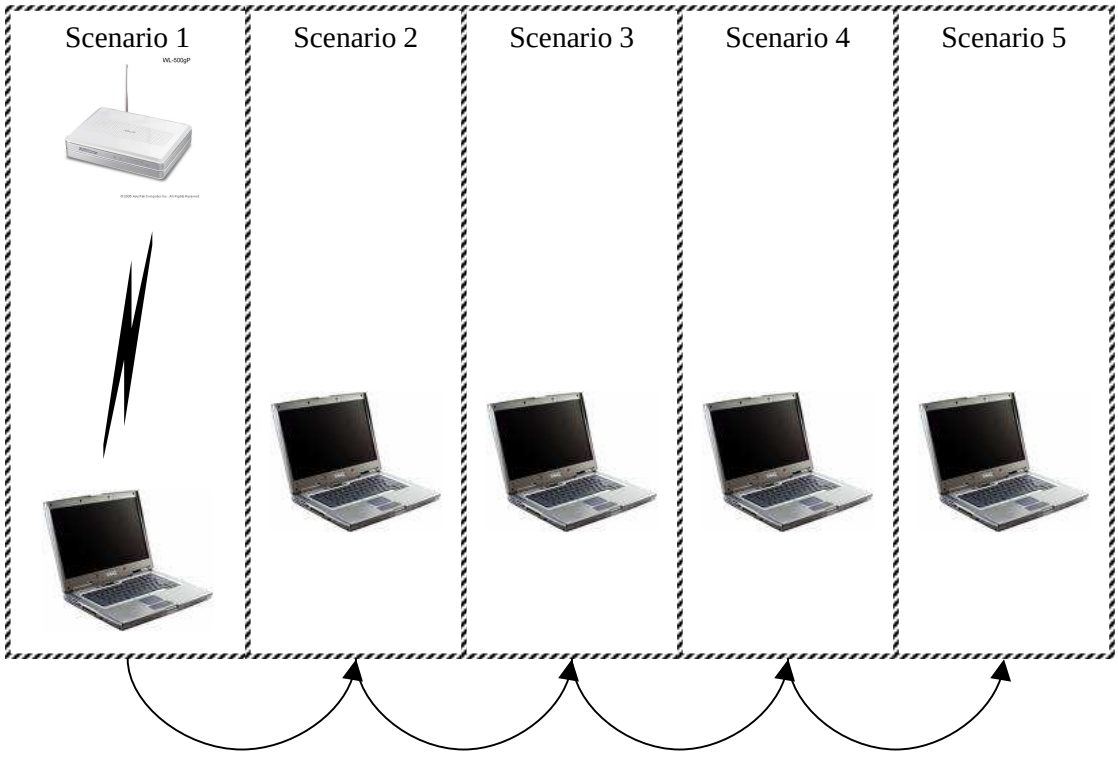


Figure 34 – Indoor measurements

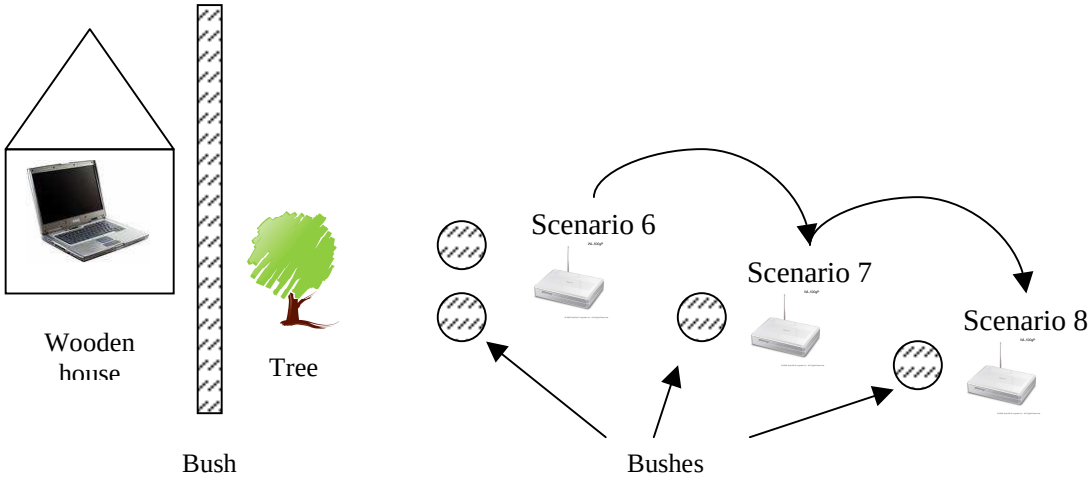


Figure 35 – Outdoor measurements

6.3 Overall Measurement Results

One measurement took 6 hours. During this time we have sent out about 200000-300000 wireless beacons. The number of received frames depended on the measurement scenario. Figure 36 shows the relative number of the frame errors and drops in each scenario. The ratio of corrupt frames expresses the ratio comparing the number of corrupt frames to the received frames.

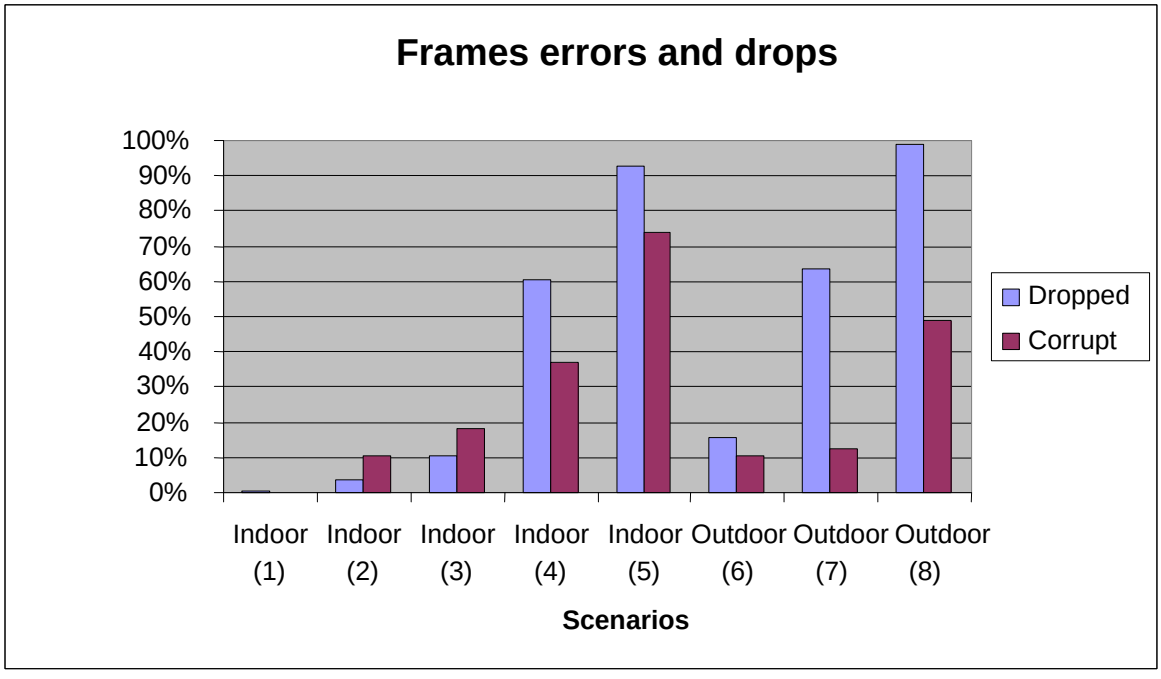


Figure 36 - Frame errors and drops in different scenarios.

As it can be seen on the figure, as the distance grows the number of drops and corrupt frames grow as well. In the case of the first scenario, almost all frames could be received correctly, while in the case of the 5th indoor and 3rd outdoor scenario, a large portion of the beacon frames were lost and among the received frames a large portion was corrupt. This figure shows that if we could use the corrupt packets as well, then this would cover a large number of packets.

On Figure 37 we present the average number of byte errors in the corrupt frames. We made 5 categories, 1-50 byte errors, 51-100, 101-150, 151-200 byte errors and finally a group where the number of byte errors was over 200 bytes.

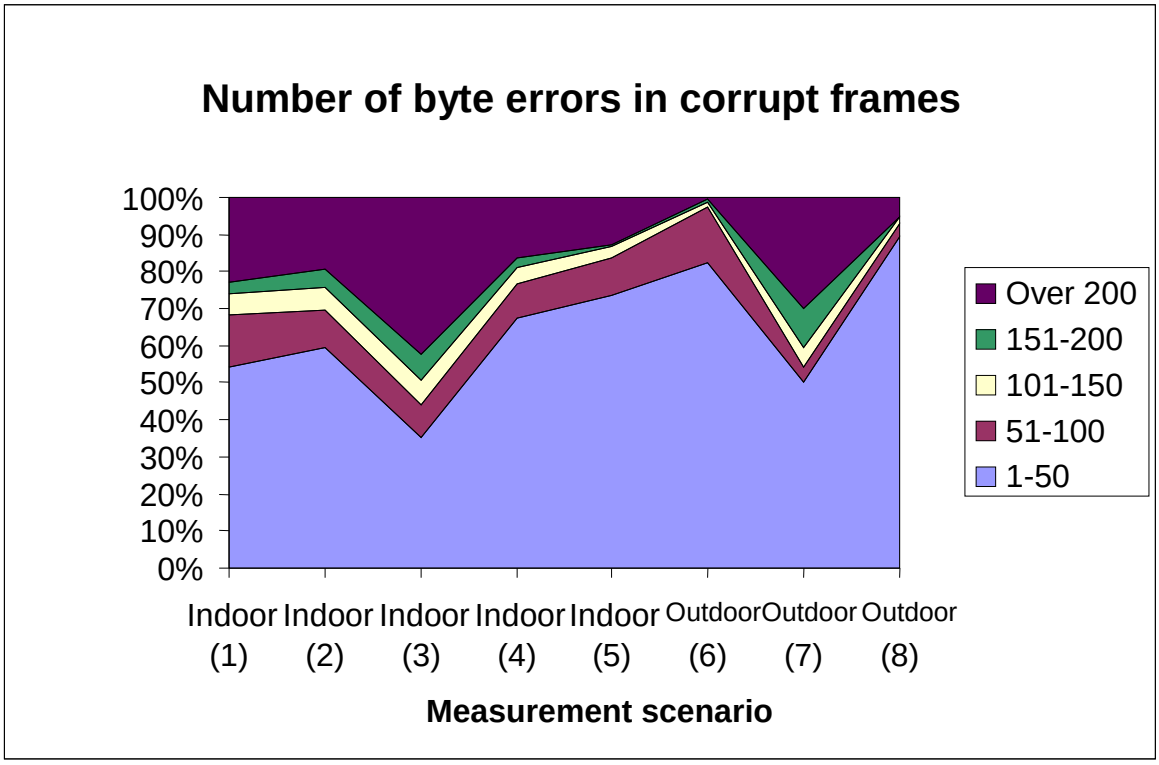


Figure 37 – Number of byte errors in corrupt frames.

On the figure one can see that in most cases the number of byte errors is below 50 byes. The exceptions are indoor 3 and outdoor 7 scenarios, where there are a great portion of frames with over 200 byte errors. In the case of the indoor 4 and 5 and outdoor 8, the situation looks better, however in this case the severely damaged frames were lost, so this is the reason of the better looking performance.

We have analyzed all the results of the different scenarios, here we presents only some of them.

6.3.1 Measurement Results of Scenario 1

In the first scenario, indoor 1, there were 213217 sent frames and we managed to get 282461 (99.73%) frames. Out of the received frames 282113 (99.88%) were correct. We got transmission errors only on the highest rates (24-54 Mbps).

In Table 6 we present the calculated state transition parameters of the Gilbert - Elliot model. We found that the basic model describes well the measurement results, no further generalized model is necessary.

	p_{gg}	$1-p_{gg}$	p_{bb}	$1-p_{bb}$	**Keep state (bytes) p_{gg}**	**Keep state (bytes) p_{bb}**
54 Mbit/s	0,9656	0,0344	0,7483	0,2517	29	4
48 Mbit/s	0,9881	0,0119	0,7584	0,2416	84	4,1
36 Mbit/s	0,9967	0,0033	0,6257	0,3743	303	2,67
24 Mbit/s	0,9986	0,0014	0,8324	0,1676	714	6

Table 6 – GE parameters of scenario 1.

The results also show that the average corrupt byte burst is 2-6 bytes long. As the transmission speed decrease, the length of the errorless bursts is longer

6.3.2 Measurement Results of Scenario 2

In the second indoor scenario we had a 10 cm thick wall between two rooms. The notebook was about 6m away from the AP. Here we sent out 220140 beacons and we captured 211601 frames. Thus there is already a 3.88% loss. We got 189279 correct frames. Here we already had corrupt frames even on the lowest datarate.

In Table 7 we present the calculated state transition parameters of the Gilbert - Elliot model. We found again that the basic model describes well the measurement results, no further generalized model is necessary.

	p_{gg}	1-p_{gg}	p_{bb}	1-p_{bb}	Keep state (bytes) p_{gg}	Keep state (bytes) p_{bb}
54 Mbit/s	0,9710	0,0290	0,7349	0,2651	34,48	3,77
48 Mbit/s	0,9820	0,0180	0,7186	0,2814	55,55	3,55
36 Mbit/s	0,9773	0,0227	0,6697	0,3302	44	3
24 Mbit/s	0,96	0,04	0,7135	0,2865	25	3,5
18 Mbit/s	0,9771	0,0229	0,7131	0,2868	43,66	3,48
12 Mbit/s	0,9906	0,0094	0,6176	0,3823	106,38	2,61
9 Mbit/s	0,9887	0,0113	0,6829	0,3171	88,5	3,15
6 Mbit/s	0,9875	0,0125	0,5606	0,4394	80	2,27
11 Mbit/s	0,9850	0,0150	0,5777	0,4223	66,7	2,36
5,5 Mbit/s	0,9613	0,0387	0,6323	0,3677	25,83	2,71
2 Mbit/s	0,9904	0,0096	0,6707	0,3293	104	3
1 Mbit/s	0,9921	0,0079	0,5839	0,4161	126,5	2,4

Table 7 – E parameters of scenario 2.

We found that the average length of the corrupt byte bursts lasts 2-4 bytes.

6.3.3 Measurement Results of Scenario 4

In this scenario the room where we set up the notebook was in 4 room distance to the Access Point. The walls were 10, 34 and 10 cm thick, respectively. The AP sent out 232906 beacons, however only 92250 of them were captured, which means a 60.39% loss. The average signal strength was 17dB, and this value is almost on the limit of the wireless card. Among the received packets there were 58125 correct ones (63.01 %). In this measurement there was no beacon captured at full speed.

In Table 8 we present the calculated state transition parameters of the Gilbert - Elliot model. We found again that the basic model describes well the measurement results, no further generalized model is necessary.

	p_{gg}	1-p_{gg}	p_{bb}	1-p_{bb}	Keep state (bytes) p_{gg}	Keep state (bytes) p_{bb}
48 Mbit/s	0,5842	0,4158	0,9642	0,0358	2,4	28
36 Mbit/s	0,607	0,393	0,96	0,04	2,5	25
24 Mbit/s	0,7739	0,2261	0,9087	0,0913	4,4	11
18 Mbit/s	0,941	0,059	0,8429	0,1571	17	6,4
12 Mbit/s	0,9871	0,0129	0,5813	0,4187	77,5	2,38
9 Mbit/s	0,984	0,016	0,7285	0,2715	62,5	3,7
6 Mbit/s	0,9942	0,0058	0,4484	0,5516	172,4	1,81
11 Mbit/s	0,9855	0,0145	0,5725	0,4275	69	2,33
5,5 Mbit/s	0,9587	0,0413	0,5995	0,4005	24,2	2,5
2 Mbit/s	0,9913	0,0087	0,5348	0,4652	115	2,14
1 Mbit/s	0,994	0,006	0,5057	0,4943	166,7	2

Table 8 – GE parameters of scenario 4.

Here it can be observed that using high data rates, the average length of the correct byte burst is really short. Over 12 Mbps the frames are so corrupted that they cannot be used for any purpose. The table does not tell but we received more frames using the 802.11b coding than in the case using the 802.11g coding. This also suggests that in such situations (distance and objects) using the 802.11b coding and rates has more benefit than using the 802.11g standard.

6.3.4 Measurement Results of Scenario 7

In scenario 7 the AP and the notebook was 80 meters away. The length of the measurement was 12.5 hour, during this time the AP sent out 439553 beacons. The notebook captured 159279 frames, so the drop was high: 63.76%, however the average signal strength was 27 dB, which is not so low. Among the captured packets 138997 were correct (87.27%). Here again, the 802.11g rates had a bad performance, while 802.11b were significantly better.

In Table 9 we present the calculated state transition parameters of the Gilbert - Elliot model. We found again that the basic model describes well the measurement results, no further generalized model is necessary.

	p_{gg}	1-p_{gg}	p_{bb}	1-p_{bb}	Keep state (bytes) p_{gg}	Keep state (bytes) p_{bb}
54 Mbit/s	0,87	0,13	0,9013	0,0987	7,7	10,1
48 Mbit/s	0,9509	0,0491	0,8409	0,1591	20,4	6,3
36 Mbit/s	0,9933	0,0067	0,6377	0,3623	149,25	2,76
24 Mbit/s	0,9989	0,0011	0,7693	0,2307	909	4,33
18 Mbit/s	0,9932	0,0068	0,791	0,209	147	4,8
9 Mbit/s	0,9959	0,0041	0,7207	0,2793	243,9	3,6
6 Mbit/s	0,4245	0,5755	0,9736	0,0264	1,73	37,9
11 Mbit/s	0,981	0,019	0,5771	0,4229	52,7	2,36
5,5 Mbit/s	0,987	0,013	0,5475	0,4525	76,9	2,2
2 Mbit/s	0,998	0,002	0,5235	0,4765	500	2,09

Table 9 – GE parameters of scenario 7.

The numbers on the no so high data rates are impressive, but we should not forget about that there was a huge portion of frame loss, so many corrupt packet lost already.

6.4 Measurement Conclusion

Analyzing the measurement results we can also point out that using old Gilbert – Elliot model is a good way to model the wireless transmission and its error. However the parameters are varying from scenario to scenario. Therefore, in real situations, when a model is necessary, the best approach is to measure the transmission first then calculate the GE parameters and finally predict the future behaviour based on the resulted model.

We can also show that during worse circumstances using the 802.11b standard is a better solution than sticking into the 802.11g standard.

Finally, the measurements also proved that using an out of box operating system (Linux) and an off the shelf (and cheap) wireless card it is possible to capture corrupt frames. Video and audio decoding may profit from these corrupt frames, the quality of the media stream could be increased with them.

7 Data Link Layer Models for OMNeT++

This section provides a brief overview of the DLL elements to be included in the OPTIMIX OMNeT++ simulation chain. An overview of the mobile node model including two DLL modules, that is, the MAC module and the RoCH module, is shown in Figure 38. The models will be detailed further in the follow-up documents of Task 3.3, namely in the documents D3.3b and D3.3c. Also no simulation results are included in this document but they will be part of these two future documents.

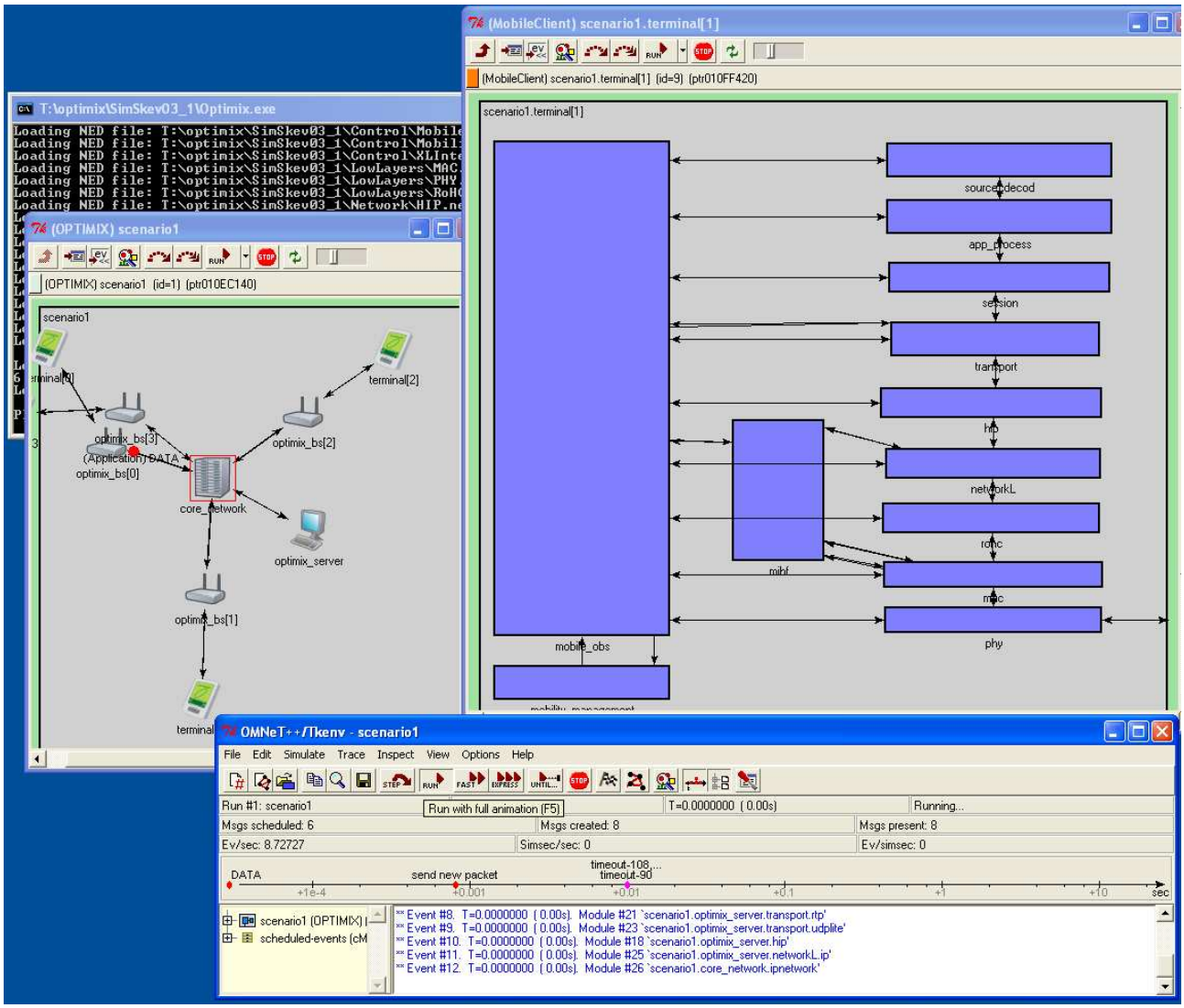


Figure 38 – Snapshot of the OMNeT++ model of the mobile node.

7.1 Basic DLL Functionalities

In the OPTIMIX OMNeT++ simulation chain, the MAC module will be present in nodes communicating over a wireless link (i.e. base stations and mobile nodes). The MAC module will be implemented between the PHY and RoCH modules. Only a selected set of DLL functionalities will be modelled into the MAC module to include a sufficient approximation of the DLL operation into the simulation chain. The first phase implementation will be based on the IEEE 802.11 and will include the essential IEEE 802.11 LLC and MAC functionalities. The OMNeT++ model of the IEEE 802.16 WiMAX functionalities will be follow later in the project. The main interest in this project relates to the common part sublayer of WiMAX MAC and its functions performing QoS, packet segmentation and burst profile selections. The rtPS QoS scheduling service is interesting in the OPTIMIX point of view as it assures bandwidth and maximum delay and jitter for real-time variable bitrate data stream.

The following WLAN specific functionalities will be present in the OMNeT++ MAC model:

- Encapsulation of the packets coming from the network layer and RoCH modules into MAC frames at the sender side prior to their delivery to the physical layer module. At the receiver the process is reversed.

- Basic mechanisms for enabling packet delivery between two STAs over a physical medium, including STA addressing, channel access scheme, etc.
- An automatic retransmission (ARQ) mechanism to implement reliability for the frame transmission
- Simple QoS support through frame prioritization and priority-based queuing
- Enabling corrupted payload data to be passed to higher layers for processing

The MAC operation implemented to the OMNeT++ model follows the state machine shown in Figure 39. The transmitter operation follows four states: in the IDLE state no messages are in transmission queue and there are no received packets to be handled. The DEFER state includes radio resource check and is followed by the WAIT_DIFS state involving the transmitter to wait for DIFS period of silent radio channel and then send message. At the receiver, the IDLE state is followed by the RECEIVE state that is triggered by the reception of a frame. In the RECEIVE state, the received message is handled by the MAC module, and disseminated to the upper layers if it is data message. The ACK mechanism and frame transmissions will be included to achieve reliability to the frame transmission.

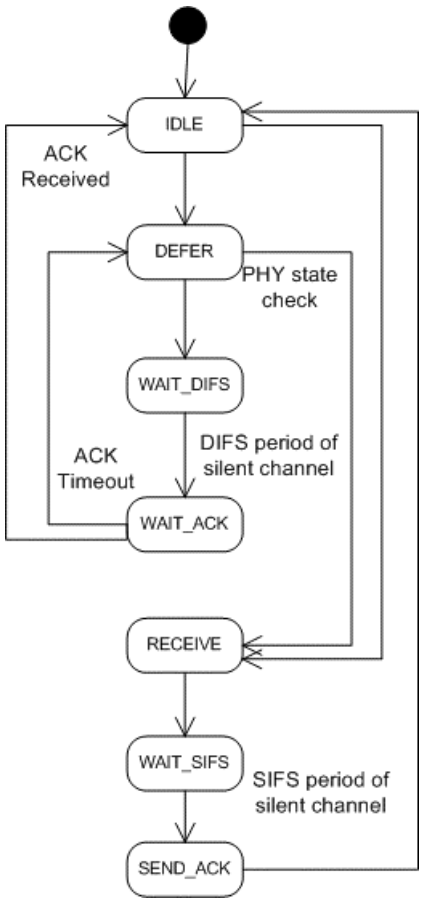


Figure 39 – State diagram of the WLAN MAC operation.

The main functionalities presented above remain the same whether the DLL resides on MN or an access point. However, MAC on the access point requires more sophisticated routing functionalities in the cases where multiple MN has connected to it. This comprises both the unicast and multicast routing.

7.2 Robust Header Compression

The RoHC protocol will be implemented between network and data link layer in Base Stations and Mobile Terminals. The RoHC module receives as input the IP packet and generates as output a packet with a compressed header.

8 Conclusion

This document describes the first specification and design of the data link layer functionalities identified in Task 3.3 of the OPTIMIX project. The document presented an overview of the basic data link layer mechanisms associated with the IEEE 802.11 WLAN and IEEE 802.16 WiMAX technologies. In addition, various enhancements to the standard data link functionalities were considered in the context of multimedia transmission. The document discussed solutions related to header compression techniques, point-to-multipoint transmission, and message scheduling techniques. Modelling of the data link performance was analyzed based on real measurements. The first designs of the DLL modules to be included in the OPTIMIX OMNeT++ simulation chain were presented as well.

The work described in this deliverable will continue under OPTIMIX WP3 and more detailed specifications and simulation results related to the elements discussed in this document will be presented later on in documents D3.3b and D3.3c.

9 Appendix 1: Results from the WiFi Measurement Scenarios

Table 10 – WiFi measurement, scenario 1 (indoor)

	All	**54 Mbit/s**	**48 Mbit/s**	**36 Mbit/s**	**24 Mbit/s**	**18 Mbit/s**	**12 Mbit/s**	**9 Mbit/s**	**6 Mbit/s**	**11 Mbit/s**	**5,5 Mbit/s**	**2 Mbit/s**	**1 Mbit/s**
Sent out	283.217	23.602	23.602	23.602	23601	23.601	23.601	23.601	23.601	23.601	23.601	23.602	23.602
Drop	756 (0,27%)	110 (0,47%)	43 (0,18%)	58 (0,25%)	46 (0,19%)	68 (0,29%)	34 (0,14%)	58 (0,25%)	52 (0,22%)	5 (0,02%)	135 (0,57%)	147 (0,62%)	0
Captured	282.461 (99,73%)	23.492 (99,53%)	23.559 (99,82%)	23.544 (99,75%)	23.555 (99,81%)	23.533 (99,71%)	23.567 (99,86%)	23.543 (99,75%)	23.549 (99,78%)	23.596 (99,98%)	23.466 (99,43%)	23.455 (99,38%)	23.602 (100%)
Correct	282.113 (99,88%)	23.320 (99,27%)	23.435 (99,47%)	23.498 (99,8%)	23.549 (99,97%)	23.533 (100%)	23.567 (100%)	23.543 (100%)	23.549 (100%)	23.596 (100%)	23.466 (100%)	23.455 (100%)	23.602 (100%)
1-50 error	189 (54,31%)	65 (37,79%)	75 (60,48%)	43 (93,48%)	6 (100%)	0	0	0	0	0	0	0	0
51-100 error	49 (14,08%)	12 (6,98%)	36 (29,03%)	1 (2,17%)	0	0	0	0	0	0	0	0	0
101-150 error	19 (5,46%)	11 (6,4%)	6 (4,84%)	2 (4,35%)	0	0	0	0	0	0	0	0	0
151-200 error	12 (3,45%)	10 (5,81%)	2 (1,61%)	0	0	0	0	0	0	0	0	0	0
over 200 error	79 (22,7%)	74 (43,02%)	5 (4,03%)	0	0	0	0	0	0	0	0	0	0

Table 11 – WiFi measurement, scenario 2 (indoor)

	All	54 Mbit/s	48 Mbit/s	36 Mbit/s	24 Mbit/s	18 Mbit/s	12 Mbit/s	9 Mbit/s	6 Mbit/s	11 Mbit/s	5,5 Mbit/s	2 Mbit/s	1 Mbit/s
Sent out	220.140	18.345	18.345	18.345	18.345	18.345	18.345	18.345	18.345	18.345	18.345	18.345	18.345
Drop	8.539 (3,88%)	3.065 (16,71%)	1.981 (10,8%)	1.278 (6,97%)	522 (2,85%)	493 (2,69%)	264 (1,44%)	320 (1,74%)	239 (1,3%)	62 (0,34%)	146 (0,8%)	133 (0,72%)	36 (0,2%)
Captured	211.601 (96,12%)	15.280 (83,29%)	16.364 (89,2%)	17.067 (93,03%)	17.823 (97,15%)	17.852 (97,31%)	18.081 (98,56%)	18.025 (98,26%)	18.106 (98,7%)	18.283 (99,66%)	18.199 (99,2%)	18.212 (99.28%)	18.309 (99,8%)
Correct	189.279 (89,45%)	5.398 (35,33%)	10.083 (61,62%)	13.703 (80,29%)	16.876 (94,69%)	17.022 (95,35%)	17.883 (98,9%)	17.579 (97,53%)	18.035 (99,61%)	18.074 (98,86%)	18.142 (99,69%)	18.187 (99,86%)	18.297 (99,93%)
1-50 error	13.229 (59,26%)	4.677 (47,33%)	4.378 (69,7%)	2.415 (71,79%)	470 (49,63%)	469 (56,51%)	168 (84,85%)	353 (79,15%)	61 (85,92%)	171 (81,82%)	39 (68,42%)	16 (64%)	12 (100%)
51-100 error	2.323 (10,41%)	1.092 (11,15%)	744 (11,85%)	214 (6,36%)	108 (11,4%)	80 (9,64%)	17 (8,59%)	41 (9,19%)	1 (1,41%)	13 (6,22%)	5 (8,77%)	8 (32%)	0
101-150 error	1.361 (6,1%)	809 (8,19%)	248 (3,95%)	134 (3,98%)	72 (7,6%)	61 (7,35%)	5 (2,53%)	16 (3,59%)	3 (4,23%)	11 (5,26%)	2 (3,51%)	0	0
151-200 error	1.086 (4,87%)	724 (7,33%)	143 (2,28%)	74 (2,2%)	68 (7,18%)	51 (6,14%)	2 (1,01%)	19 (4,26%)	1 (1,41%)	3 (1,44%)	1 (1,75%)	0	0
over 200 error	4.323 (19,37%)	2.580 (26,11%)	768 (12,23%)	527 (15,67%)	229 (24,18%)	169 (20,36%)	6 (3,03%)	17 (3,81%)	5 (7,04%)	11 (5,26%)	10 (17,54%)	1 (4%)	0

Table 12 – WiFi measurement, scenario 3 (indoor)

	All	54 Mbit/s	48 Mbit/s	36 Mbit/s	24 Mbit/s	18 Mbit/s	12 Mbit/s	9 Mbit/s	6 Mbit/s	11 Mbit/s	5,5 Mbit/s	2 Mbit/s	1 Mbit/s
Sent out	247.543	20.629	20.628	20.628	20.628	20.628	20.628	20.629	20.629	20.629	20.629	20.629	20.629
Drop	25.790 (10,42%)	8.795 (42,63%)	4.784 (23,19%)	2.748 (13,32%)	2.234 (10,83%)	2.045 (9,91%)	1.373 (6,66%)	1.350 (6,54%)	1.256 (6,09%)	369 (1,79%)	337 (1,63%)	318 (1,54%)	181 (0,88%)
Captured	221.753 (89,58%)	11.834 (57,37%)	15.844 (76,81%)	17.880 (86,68%)	18.394 (89,17%)	18.583 (90,09%)	19.255 (93,46%)	19.279 (93,46%)	19.373 (93,91%)	20.260 (98,21%)	20.292 (98,37%)	20.311 (98,46%)	20.448 (99,12%)
Correct	181.700 (81,94%)	6 (0,05%)	108 (0,68%)	10.567 (59,1%)	17.796 (96,75%)	18.019 (96,96%)	18.410 (95,61%)	18.215 (94,48%)	19.145 (98,82%)	18.829 (92,94%)	19.985 (98,49%)	20.215 (99,53%)	20.405 (99,79%)
1-50 error	14.028 (35,02%)	48 (0,41%)	4.339 (27,57%)	6.314 (86,34%)	260 (43,48%)	109 (19,33%)	632 (74,79%)	610 (57,33%)	181 (79,39%)	1.228 (85,81%)	206 (67,1%)	71 (73,96%)	30 (69,77%)
51-100 error	3.576 (8,93%)	118 (1%)	2.644 (16,8%)	350 (4,79%)	21 (3,51%)	26 (4,61%)	101 (11.95%)	154 (14,47%)	12 (5,26%)	103 (7,2%)	30 (9,77%)	10 (10,42%)	7 (16,28%)
101-150 error	2.654 (6,63%)	349 (2,95%)	1.864 (11,85%)	165 (2,26%)	18 (3,01%)	60 (10,64%)	49 (5,8%)	84 (7,89%)	8 (3,51%)	36 (2,52%)	17 (5,54%)	2 (2,08%)	2 (4,65%)
151-200 error	2.876 (7,18%)	933 (7,89%)	1.601 (10,17%)	133 (1,82%)	14 (2,34%)	60 (10,64%)	29 (3,43%)	74 (6,95%)	2 (0,88%)	19 (1,33%)	6 (1,95%)	4 (4,17%)	1 (2,33%)
over 200 error	16.919 (42,24%)	10.380 (87,76%)	5.288 (33,6%)	351 (4,8%)	285 (47,66%)	309 (54,79%)	34 (4,02%)	142 (13,35%)	25 (10,96%)	45 (3,14%)	48 (15,64%)	9 (9,38%)	3 (6,98%)

Table 13 – WiFi measurement, scenario 4 (indoor)

	All	54 Mbit/s	48 Mbit/s	36 Mbit/s	24 Mbit/s	18 Mbit/s	12 Mbit/s	9 Mbit/s	6 Mbit/s	11 Mbit/s	5,5 Mbit/s	2 Mbit/s	1 Mbit/s
Sent out	232.906	19.409	19.409	19.409	19.409	19.409	19.409	19.409	19.409	19.409	19.409	19.408	19.408
Drop	140.656 (60,39%)	19.409 (100%)	19.405 (99,98%)	19.325 (99,57%)	17.401 (89,65%)	16.507 (85,05%)	11.928 (61,46%)	11.669 (60,12%)	10.901 (56,16%)	5.267 (27,14%)	3.422 (17,63%)	3.033 (15.63%)	2.389 (12,31%)
Captured	92.250 (39,61%)	0	4 (0,02%)	84 (0,43%)	2.008 (10,35%)	2.902 (14,95%)	7.481 (38,54%)	7.740 (39,88%)	8.508 (43,84%)	14.142 (72,86%)	15.987 (82,37%)	16.375 (84,37%)	17.019 (87,69%)
Correct	58.125 (63,01%)	0	0	0	8 (0,4%)	49 (1,69%)	1.573 (21,03%)	912 (11,78%)	7.301 (58,81%)	3.868 (27,35%)	12.825 (80,22%)	14.978 (91,47%)	16.611 (97,6%)
1-50 error	23.025 (67,47%)	0	0	0	39 (1,95%)	208 (7,29%)	4.936 (83,55%)	4.332 (63,44%)	1.139 (94,37%)	8.563 (83,35%)	2.142 (67,74%)	1.274 (91,2%)	392 (96,08%)
51-100 error	3.097 (9,08%)	0	0	0	25 (1,25%)	132 (4,63%)	518 (8,77%)	1.124 (16,46%)	25 (2,07%)	918 (8,94%)	254 (8,03%)	86 (6,16%)	15 (3,68%)
101-150 error	1.471 (4,31%)	0	0	0	34 (1,7%)	204 (7,15%)	199 (3,37%)	536 (7,85%)	17 (1,41%)	311 (3,03%)	144 (4,55%)	25 (1,74%)	1 (0,25%)
151-200 error	965 (2,83%)	0	0	0	33 (1,65%)	284 (9,95%)	98 (1,66%)	299 (4,38%)	5 (0,41%)	154 (1,5%)	86 (2,72%)	6 (0,43%)	0
over 200 error	5.567 (16,31%)	0	4 (100%)	84 (100%)	1.869 (93,45%)	2.025 (70,98%)	157 (2,66%)	537 (7,86%)	21 (1,74%)	328 (3,19%)	536 (16,95%)	6 (0,43%)	0

Table 14 – WiFi measurement, scenario 5 (indoor)

	All	54 Mbit/s	48 Mbit/s	36 Mbit/s	24 Mbit/s	18 Mbit/s	12 Mbit/s	9 Mbit/s	6 Mbit/s	11 Mbit/s	5,5 Mbit/s	2 Mbit/s	1 Mbit/s
Sent out	255.828	21.319	21.319	21.319	21.319	21.319	21.319	21.319	21.319	21.319	21.319	21.319	21.319
Drop	237.217 (92,73%)	21.319 (100%)	21.319 (100%)	21.319 (100%)	21.317 (99,99%)	21.315 (99,98%)	21.306 (99,94%)	21.303 (99,92%)	21.299 (99,91%)	21.108 (99,01%)	19.631 (92,08%)	17.784 (83,42%)	8.197 (38,45%)
Captured	18.611 (7,27%)	0	0	0	2 (0,01%)	4 (0,02%)	13 (0,06%)	16 (0,08%)	20 (0,09%)	211 (0,99%)	1.688 (7,92%)	3.535 (16,58%)	13.122 (61,55%)
Correct	4.839 (26%)	0	0	0	0	0	2 (15,38%)	2 (12,5%)	12 (60%)	7 (3,32%)	46 (2,73%)	116 (3,28%)	4.654 (35,47%)
1-50 error	10.104 (73,37%)	0	0	0	0	0	9 (81,82%)	4 (28,57%)	6 (75%)	34 (16,67%)	63 (3,84%)	1.658 (48,49%)	8.330 (98,37%)
51-100 error	1.411 (10,25%)	0	0	0	0	0	1 (9,09%)	4 (28,57%)	2 (25%)	7 (3,43%)	21 (1,28%)	1.246 (36,44%)	130 (1,54%)
101-150 error	408 (2,96%)	0	0	0	0	0	1 (9,09%)	1 (7,14%)	0	6 (2,94%)	15 (0,91%)	377 (11,03%)	8 (0,09%)
151-200 error	116 (0,84%)	0	0	0	0	0	0	1 (7,14%)	0	10 (4,9%)	11 (0,67%)	94 (2,75%)	0
over 200 error	1.733 (12,58%)	0	0	0	2 (100%)	4 (100%)	0	4 (28,57%)	0	147 (72,06%)	1.532 (93,3%)	44 (1,29%)	0

Table 15 – WiFi measurement, scenario 6 (outdoor)

	All	54 Mbit/s	48 Mbit/s	36 Mbit/s	24 Mbit/s	18 Mbit/s	12 Mbit/s	9 Mbit/s	6 Mbit/s	11 Mbit/s	5,5 Mbit/s	2 Mbit/s	1 Mbit/s
Sent out	246.816	20.568	20.568	20.568	20.568	20.568	20.568	20.568	20.568	20.568	20.568	20.568	20.568
Drop	38.982 (15,79%)	5.274 (25,64%)	4.762 (23,15%)	4.642 (22,57%)	4.115 (20,01%)	3.965 (19,28%)	4.066 (19,77%)	3,867 (18,8%)	4.438 (21,58%)	1.061 (5,16%)	1.071 (5,21%)	1.080 (5,25%)	641 (3,12%)
Captured	207.834 (84,21%)	15.294 (74,36%)	15.806 (76,85%)	15.926 (77,43%)	16.453 (79,99%)	16.603 (80,72%)	16.502 (80,23%)	16.701 (81,2%)	16.130 (78,42%)	19.507 (94,84%)	19.497 (94,79%)	19.488 (94,75%)	19.927 (96,88%)
Correct	186.329 (89,65%)	8 (0,05%)	11.077 (70,08%)	15.831 (99,4%)	15.806 (96,07%)	16.009 (96,42%)	16,475 (99,84%)	16.605 (99,43%)	16.115 (99,91%)	19.491 (99,92%)	19.497 (100%)	19.488 (100%)	19.927 (100%)
1-50 error	17,706 (82,33%)	12.179 (79,67%)	4.719 (99,79%)	15 (15,79%)	57 (8,81%)	586 (98,65%)	27 (100%)	98 (98,96%)	13 (86,67%)	15 (93,75%)	0	0	0
51-100 error	3.236 (15,15%)	3.042 (19,9%)		1 (1,05%)	184 (28,44%)	6 (1,01%)	0	1 (1,04%)	2 (13,33%)	0	0	0	0
101-150 error	314 (1,46%)	64 (0,42%)	0	0	248 (38,33%)	1 (0,17%)	0	0	0	1 (6,25%)	0	0	0
151-200 error	119 (0,55%)	1 (0,01%)	0	1 (1,05%)	117 (18,08%)	0	0	0	0	0	0	0	0
over 200 error	130 (0,6%)	0	10 (0,21%)	78 (82,11%)	41 (6,34%)	1 (0,17%)	0	0	0	0	0	0	0

Table 16 – WiFi measurement, scenario 7 (outdoor)

	All	54 Mbit/s	48 Mbit/s	36 Mbit/s	24 Mbit/s	18 Mbit/s	12 Mbit/s	9 Mbit/s	6 Mbit/s	11 Mbit/s	5,5 Mbit/s	2 Mbit/s	1 Mbit/s
Sent out	439.553	36.630	36.630	36.630	36.630	36.630	36.629	36.629	36.629	36.629	36.629	36.629	36.629
Drop	280.274 (63,76%)	34.051 (92,96%)	30.177 (82,38%)	28.554 (77,95%)	28.084 (76,67%)	28.015 (76,48%)	28.037 (76,54%)	27.834 (75,99%)	27.991 (76,42%)	14.032 (38,31%)	13.529 (36.94%)	13.432 (36,67%)	6.538 (17,85%)
Captured	159.279 (36,24%)	2.579 (7,04%)	6.453 (17,62%)	8.076 (22,05%)	8.546 (23,33%)	8.615 (23,52%)	8.592 (23,46%)	8.795 (24,01%)	8.638 (23,58%)	22.597 (61,69%)	23.100 (63.06%)	23,197 (63,33%)	30.091 (82,15%)
Correct	138.997 (87,27%)	0	0	162 (2,01%)	8.512 (99,6%)	8.613 (99,98%)	8.592 (100%)	8.793 (99,98%)	8.636 (99,98%)	20.823 (92,15%)	21.692 (93,90%)	23.083 (99,51%)	30.091 (100%)
1-50 error	10.162 (50,1%)	0	3 (0,05%)	7.299 (92,23%)	34 (100%)	2 (100%)	0	2 (100%)	0	1.393 (78,52%)	1.315 (93,39%)	114 (100%)	0
51-100 error	851 (4,2%)	0	107 (1,66%)	471 (5,95%)	0	0	0	0	0	247 (13,92%)	26 (1,85%)	0	0
101-150 error	1.060 (5,23%)	0	880 (13,64%)	109 (1,38%)	0	0	0	0	0	64 (3,61%)	7 (0,5%)	0	0
151-200 error	2.089 (10,3%)	0	2.025 (31,38%)	28 (0,35%)	0	0	0	0	0	30 (1,69%)	6 (0,43%)	0	0
over 200 error	6.120 (30,17%)	2.579 (100%)	3.438 (53,28%)	7 (0,09%)	0	0	0	0	2 (100%)	40 (2,25%)	54 (3,84%)	0	0

Table 17 – WiFi measurement, scenario 8 (outdoor)

	All	54 Mbit/s	48 Mbit/s	36 Mbit/s	24 Mbit/s	18 Mbit/s	12 Mbit/s	9 Mbit/s	6 Mbit/s	11 Mbit/s	5,5 Mbit/s	2 Mbit/s	1 Mbit/s
Sent out	259.524	21.627	21.627	21.627	21.627	21.627	21.627	21.627	21.627	21.627	21.627	21.627	21.627
Drop	256.871 (98,98%)	21.627 (100%)	21.627 (100%)	21.627 (100%)	21.627 (100%)	21.627 (100%)	21.627 (100%)	21.627 (100%)	21.627 (100%)	21.620 (99,97%)	21.563 (99,7%)	21.506 (99,44%)	19.166 (88,62%)
Captured	2.653 (1,02%)	0	0	0	0	0	0	0	0	7 (0,03%)	64 (0,3%)	121 (0,56%)	2.461 (11,38%)
Correct	1.355 (51,07%)	0	0	0	0	0	0	0	0	0	0	3 (2,48%)	1.352 (54,94%)
1-50 error	1.161 (89,45%)	0	0	0	0	0	0	0	0	1 (14,29%)	4 (6,25%)	47 (39,83%)	1.109 (100%)
51-100 error	45 (3,47%)	0	0	0	0	0	0	0	0	0	0	45 (38,14%)	0
101-150 error	21 (1,62%)	0	0	0	0	0	0	0	0	0	0	21 (17,8%)	0
151-200 error	5 (0,39%)	0	0	0	0	0	0	0	0	0	0	5 (4,24%)	0
over 200 error	66 (5,08%)	0	0	0	0	0	0	0	0	6 (85,71%)	60 (93,75%)	0	0

10 References and Glossary

10.1 References

[1] OPTIMIX Consortium, "Technical annex for OPTIMIX project: Optimisation of multimedia over wireless IP links via X-layer design," version 2.0.1, October 2007.

[2] Claude E. Shannon, *A Mathematical Theory of Communication*, *Bell System Technical Journal*, vol. 27, pp. 379-423 and 623-656, July and October, 1948.

[3] L.R. Bahl, J. Cocke, F. Jelinek and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," IEEE Trans. on Information Theory, vol. 20, pp. 284—287, March 1974.

[4] Faber, D., et. Al. "A Thinwire Protocol for connecting personal computers to the INTERNET", RFC 914, 1984.

[5] Jacobson, V., "Compressing TCP/IP Headers", RFC 1144, 1990.

[6] Degermark, M., et. Al., "IP Header Compression", IETF RFC 2507, February 1999.

[7] Casner, S., "Compressing IP/UDP/RTP Headers for Low-Speed Serial Links", IETF RFC 2508, February 1999.

[8] Bormann C., et Al., "RObust Header Compression (ROHC): Framework and four profiles: RTP, UDP, ESP and uncompressed", IETF RFC 3095, 2001.

[9] IEEE Standard for Local and Metropolitan Area Networks. Part 16: Air Interface for Fixed Broadband Wireless Access Systems. IEEE Std. 802.16-2004, October 2004.

[10] IEEE Standard for Local and Metropolitan Area Networks. Part 16: Air Interface for Fixed Broadband Wireless Access Systems. Amendment 2: Physical and Medium Access Control Layer for Combined Fixed and Mobile Operation in Licensed Bands. IEEE Std. 802.16e-2005, December 2005.

[11] J. G. Andrew, G. Arunabha, and Rias Muhamed, "Fundamentals of WiMAX: Understanding Broadband Wireless Networking". Prentice Hall, Upper Saddle River, NJ, 2007.

[12] OPTIMIX Consortium, "Deliverable D3.2a: Specification and Preliminary Design of Network Architecture", version 1.0, January 2009.

[13] IEEE Std 802.11-2007 Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications (Revision of IEEE Std 802.11-1999).

[14] OPTIMIX Consortium, "Deliverable D2.3a: Preliminary Scheme of JSCC/D Controller for Point to Multipoint Communication and Optimization Criteria" version 1.0, January 2009.

10.2 Glossary

AP	*Access Point*	
ACK	*Acknowledgement*	
ARF	*Auto Rate Fallback*	
ARQ	*Automatic Repeat-Request*	
BE	*Best Effort*	
BMMM	*Batch Mode Multicast MAC Protocol*	
BMW	*Broadcast Medium Window*	
BS	*Base Station*	
BSMA	*Broadcast Support Multiple Access*	
CID	*Connection Identifier*	
CINR	*Carrier-to-Interference-plus-Noise Ratio*	
CQICH	*Channel-Quality Indicator Channel*	

CSI	*Channel State Information*	
CSMA/CA	*Carrier Sense Multiple Access with Collision Avoidance*	
CTS	*Clear-To-Send*	
DCCP	*Datagram Congestion Control Protocol*	
DCF	*Distributed Coordination Function*	
DIFS	*DCF Inter-Frame Space*	
DL	*Downlink*	
DLL	*Data Link Layer*	
EAP	*Extensible Authentication Protocol*	
EDCA	*Enhanced Distributed Channel Access*	
ert-VR	*Extended Real-time Variable Rate*	
HARQ	*Hybrid Automatic Repeat-Request*	
HCCA	*HCF Controlled Channel Access*	
HCF	*Hybrid Coordination Function*	
JSCC	*Joint Source and Channel Coding*	
LBP	*Leader Based Protocol*	
LLC	*Logical Link Control*	
MAC	*Medium Access Control*	
MAP	*Maximum A Posteriori*	
MIMO	*Multiple-Input Multiple-Output*	
MN	*Mobile Node*	
MPDU	*MAC Protocol Data Units*	
NAK	*Negative Acknowledgement*	
NAV	*Network Allocation Vector*	
nrtPS	*Non-real-time Polling Service*	
OAR	*Opportunistic Auto-Rate*	
OFDMA	*Orthogonal Frequency Division Multiple Access*	
PCF	*Point Coordination Function*	
PDU	*Protocol Data Unit*	
PHY	*Physical Layer*	
QoS	*Quality of Service*	
RBAR	*Receiver Based Auto Rate*	
RMAC	*Reliable Multicast MAC Protocol*	
RoCH	*Robust Header Compression*	
RSA	*Rivest, Shamir, Adleman*	
rtPS	*Real-time Polling Service*	
RTP	*Real-time Transport Protocol*	
RTS	*Ready-To-Send*	
SAP	*Service Access Point*	
SDU	*Service Data Unit*	
SS	*Security Sublayer*	
SSI	*Source Significance Information*	
STA	*Station*	
SVC	*Scalable Video Coding*	
TCP	*Transmission Control Protocol*	
UDP	*User Datagram Protocol*	
UL	*Uplink*	
UGS	*Unsolicited Grant Services*	
WiMAX	*Worldwide Interoperability for Microwave Access*	*WiMAX is a common name for a technology based on IEEE 802.16 air interface standards provided by WiMAX Forum.*
WLAN	*Wireless Local Area Network*	*WiFi is a commercial name for a WLAN realization.*