ACTS Project AC085

Wireless ATM Network

Demonstrator

The Magic WAND

Deliverable 3D5
Wireless ATM MAC – Final Report

DELIVERABLES

Part I

Project Number:	AC085
Project Title:	The Magic WAND
Deliverable Type (P/R/L/I):	P

CEC Deliverable Number:	AC085/INT/ACT/DS/P/035/b1
Contractual date if delivery to the CEC:	31.08.1998
Actual date of delivery to the CEC:	31.08.1998
Title of deliverable:	Wireless ATM MAC – Final Report
Workpackage contributing to the deliverable:	WP3
Nature of the deliverable (P/R/S/T/O):	R
Author(s):	Ioannis Dravopoulos, Nikos Pronios, Anastasia Andritsou, Ioannis Piveropoulos, Nikos Passas, Dimitris Skyrianoglou, Geert Awater, Jan Kruys, Neda Nikaein, Alain Enout, Stephane Decrauzat, Thomas Kaltenschnee, Thorsten Schumann, Jürg Meierhofer, Stefan Thömel, Jouni Mikkonen

Abstract:	This deliverable describes the final Specification of the Medium Access Control (MAC) layer of the Magic WAND project. The deliverable presents the reason for a special MAC needed for Wireless ATM, along with the requirements it should fulfil. The WAND MAC is based on a TDMA / TDD channel sharing among the Access Points and the Mobile Terminals. Thus, based on the above, the MAC protocol employed in WAND is named as ***M***obile ***A***ccess ***S***cheme based on ***C***ontention ***A***nd ***R***eservation for ***A***TM, or ***MASCARA.*** The addressing scheme and the Time Frame detailed structure is provided in this deliverable. Then, this deliverable describes the MAC functional entities and the message primitives exchanged among them. The functional entities include those of responsible for ATM traffic buffering, error control, allocation of ATM cells into the air interface, and the overall control of MASCARA system. The interfaces with the higher layers and the Physical layers are also provided. At the end of the deliverable, there are appendices on the Scheduler algorithm of the WAND MAC protocol, on an improved Error Control and memory management of ATM cells, on the Signals exchanged within the MAC layer, on the Basic MASCARA description using SDL and on some Message Sequence Charts that show the more important parts of the MASCARA behaviour.
Keyword list:	Wireless ATM, Medium Access Control, Data Link Control, Radio Resource, Handover, Scheduler, DCA, Wireless Traffic Management, SDL

Part II

(Executive Summary)

The Mobile Access Scheme based on Contention And Reservation for ATM (MASCARA), contains a MAC protocol with several features for ATM traffic transportation over the air. Therefore it can support resource allocation according to the traffic contract of each ATM connection, error control as well as mobility related services.

This deliverable describes the Specification of the MASCARA protocol used within the target WAND system. In this deliverable, the general requirements and functionality of the MASCARA layer are described.

The MASCARA addressing scheme over the air is presented. Afterwards, a detailed analysis on the MASCARA Time Slot and Time Frame length and structure is given. The format of the MASCARA Packet Data Units (MPDUs) is presented, along with all the parameters that are used for the MASCARA peer-to-peer protocol.

Then the description and specification of the functional entities comprising MASCARA follows. The functional entities are divided in five main high level functional entities, according to the function they are performing. These main functional entities (FE) are Inter Control Communication, MASCARA Control, Control Segmentation and Reassembly, Wireless Data Link Control, and MAC Data Pump.

- *Inter Control Communication* (ICC) FE is responsible for exchanging signals with the ATM layer and the upper layers.
- *MASCARA Control* (MCL) FE is responsible for internal MASCARA management and peer to peer communication between a Mobile Terminal and the Access Point. Issues like handover, beacon announcement, Mobile Terminal association, etc. are addressed within this FE.
- *Control Segmentation and Reassembly* (CSR) FE puts the MCL peer-to-peer messages into ATM cell format to be transmitted over the wireless link, and vice-versa. Thus, MASCARA control signalling can be served as any other ATM connection.
- *Wireless Data Link Control* (WDLC) FE has twofold purpose. Primarily, it is responsible for collecting ATM cells and sending ATM cells that contain user information to/from the ATM layer, as well as cells exchanged with CSR. Its other purpose, is to provide error control to some ATM connections. The error control scheme adopted in WAND Demo is Go-Back-N ARQ. According to the ATM connection traffic contract and type, error control will apply or not apply to ATM connections.
- *MAC Data Pump* FE is responsible for several issues. It controls the interfaces with the physical layer as well as the physical modem. It forms the ATM cells into longer packets, called MPDUs; it also contains the Scheduler, which allocates ATM cells to time frame slots, according to their traffic contract.

The interfaces of MASCARA with the higher layers and the physical layer are provided. These interfaces are divided in the user plane information flow interfaces, that is responsible for allocating and passing ATM cells over the wireless link, and in the control plane information flow, that is responsible for controlling the physical layer and exchanging useful information with the higher layers.

Finally, there are 5 appendices dealing with:

- The scheduling algorithm of the WAND MAC
- An improved WDLC-Scheduler interaction
- The signals exchanged to, from and inside MASCARA
- The general MASCARA architecture given in SDL
- The basic MASCARA scenarios presented in Message Sequence Charts

Part III

Table of Contents

Table of Figures

1. Introduction

1.1. Purpose

The purpose of this deliverable is to provide the architecture and functionality of the Medium Access Control layer used within Magic WAND (Wireless ATM Network Demonstrator) AC085 ACTS Project.

1.2. Scope

This deliverable defines the functionality of the Magic WAND Medium Access Control Layer. The MAC layer has to provide transparent services to the ATM layer, along with coping with the Radio Physical layer and its impairments (e.g. not stable channel, high error rates).

Some of the issues addressed in this deliverable are the Multiple Access technique used by MAC, the association of a Mobile Terminal with the Wireless network, ways to decide if handover is needed and cope with it and ways to deal with ATM traffic requirements.

1.3. Definitions, acronyms and abbreviations

AAA	AP Association Agent
AAL	ATM Adaptation Layer
ABA	AP Beacon Agent
ABR	Available Bit Rate
ABT	ATM Block Transfer
ACK	Acknowledgement
ACR	Allowed Cell Rate
ACTS	Advanced Communications Technologies and Services
ADC	AP Dynamic Control
ADG	AP Dynamic Generic control
AIA	AP I_am_alive Agent
ALS	AP Link Status recorder
AMA	AP MVC Agent
AP	Access Point
API	Application Programming Interface
ARQ	Automatic Repeat Request
ASG	AP Steady State Gen Ctrl

ASN.1	Abstract Syntax Notation 1
ASS	AP Steady State Control
ATC	ATM Transfer Capability
ATI	AP Tip Agent
ATM	Asynchronous Transfer Mode
BCPN	Business Customer Premises Network
BER	Bit Error Rate
B-ISDN	Broadband ISDN (see ISDN)
BRAN	Broadband Radio Access Networks
BT	Burst Tolerance
CAC	Connection Admission Control
CBR	Constant Bit Rate
CC	Call Control
CDT	Cell Delay Tolerance
CDV	Cell Delay Variation
CDVT	Cell Delay Variation Tolerance
CLP	Cell Loss Priority
CLR	Cell Loss Rate
CMR	Cell Misinsertion Rate
COFDM	Coded Orthogonal Frequency Division Multiplex
con	confirmation
CPN	Customer Premises Network
CRC	Cyclic Redundancy Check
CRT	Cathod Ray Tube
CS	Control Station
CTD	Cell Transfer Delay
CTV	Cell Tolerance Variation
DA	Destination Address
DB	Data Base
DMA	Direct Memory Access
EIRP	Equivalent Isotropic Radiated Power
EPC	Emitted Power Control

EPD	Early Packet Discard
ETSI	European Telecommunications Standards Institute
FDD	Frequency Division Duplex
FDMA	Frequency Division Multiple Access
FE	Functional Entity
FEC	Forward Error Control
FH	Frame Header
FIFO	First In First Out
FP	Frame Processor
GBN	Go-Back-N
GMC	Generic Mascara Control
GSM	Global System for Mobile communications
HH	Handover Handler
HI	Handover Indicator
HIPERLAN	HIgh PErfomance Radio Local Area Network
HO	HandOver
ind	indication
IP	Internet Protocol
ISDN	Integrated Services Digital Network
ITU	International Telecommunications Union
ITU-R	ITU Radiocommunication Standardization Sector
ITU-T	ITU Telecommunication Standardization Sector
JVTOS	Joint Viewing and Tele-Operation Service
LAN	Local Area Network
LCP	Layer Control Protocol
LLC	Logical Link Control
LOS	Line-Of-Sight (path)
MAA	MT Association Agent
MAC	Medium Access Control
MASCARA	Mobile Access Scheme based on Contention And Reservation for ATM
MBA	MT Beacon Agent
MBS	Maximum Burst Size

MCL	MASCARA ControL
MCPN	Mobile Customer Premises Network
MCR	Minimum Cell Rate
MDC	MT Dynamic Control
MDG	MT Dynamic Generic control
MEF	Measurement Functions
MEU	Message Encapsulation Unit
MHI	MT Handover Indicator
MIA	MT I_am_alive Agent
MMA	MT MVC Agent
MMC	Mobility Management Control
MME	Modem Management Enable
MMI	Modem Management Interface
MMI_SIO	Modem Management Interface - Serial Input and Output
MPDU	MAC-PDU
MPR	MPDU Handler Receive
MPS	MT Power saving
MPX	MPDU Handler Transmit
MSG	MT Steady state Generic control
MSS	MT Steady State control
MT	Mobile Terminal
MTC	MT Target Cell
MTI	MT TIP agent
MVC	MAC Virtual Circuit
NMX	Network Management
OFDM	Orthogonal Frequency Division Multiplexing
OSI	Open Systems Interconnection
PBX	Private Branch eXchange
PC	Personal Computer
PCM	Pulse Code Modulation
PCMCIA	Personal Computer Memory Card International Association
PCR	Peak Cell Rate

PDA	Personal Digital Assistant
PDU	Protocol Data Unit
PHY	Physical Layer
PN	Public Network
PNNI	Private Network-Network Interface
PSTN	Public Switched Telephone Network
PT	Payload Type
PVC	Permanent Virtual Circuit
QoS	Quality of Service
RACE	Research and Development in Advanced Communications for Europe
RCL	Radio ControL
RCM	Radio Control Manager
RECV	Receive
req	request
res	response
RF	Radio Frequency
RLQ	Radio Link Quality
RN	Request Number
RRM	Radio Resource Manager
RSSI	Received Signal Strength Indication
RX	Receive
RXRN	Received Request Number
S&W	Stop & Wait
SA	Source Address
SAR	Segmentation And Reassembly
SBR	Statistical Bit Rate
SCR	Sustainable Cell Rate
SDL	Specification & Description Language
SDU	Service Data Unit
SECBR	Severely Errored Cell Block Rate
SN	Sequence Number
SNMP	Simple Network Management Protocol

SVC	Switched Virtual Circuit
SW	SWitch
TA	Terminal Adapter
TCP	Transmission Control Protocol
TCP/IP	Transmission Control Protocol / Internet Protocol
TDD	Time Division Duplex
TDM	Time Division Multiplexing
TDMA	Time Division Multiple Access
TE	Terminal Equipment
TED	Timing Element Descriptor
TF	Time Frame
TIP	Temporarily Incomunicado Protocol
TM	Traffic Management
TS	Time Slot
TX	Transmit
TXRN	Request Number to be transmitted
UBR	Unspecified Bit Rate
UBR	Unspecified Bit Rate
UMTS	Universal Mobile Telecommunications System
UNI	User Network Interface
UPC	Usage Parameter Control
VBR	Variable Bit Rate
VBR-nrt	Variable Bit Rate - Non Real Time
VBR-rt	Variable Bit Rate - Real Time
VC	Virtual Connection
VCI	Virtual Circuit Identifier
VPI	Virtual Path Identifier
WAND	Wireless ATM Network Demonstrator
WATM	Wireless ATM
WCAC	Wireless Connection Admission Control
WDLC	Wireless Data Link Control
WLAN	Wireless Local Area Network

WWW	World Wide Web

1.4. Overview

Wireless ATM is a technology that will extend both ATM/B-ISDN signalling and ATM virtual channels / virtual paths to the mobile users. The studies on how to accomplish this have been active since the early 1990s. Today, ATM as has evolved from research to a commercial technology and first standards are ready. Therefore, it is time to take the next step and build a complete wireless ATM system demonstrator.

The European Union supports these pre-competitive activities in the Fourth Framework programme called Advanced Communications Technologies and Services (ACTS). Within ACTS, the Magic WAND project is focusing on the customer premises wireless ATM access systems. The main objectives of the project are as follows:

- To specify a wireless, customer premises, access system for ATM networks that maintains the service characteristics and benefits of ATM networks;
- To demonstrate and carry out user trials with the selected user group and test the feasibility of a radio based ATM access system;
- To promote the standardisation, notably in European Telecommunications Standards Institute (ETSI), of wireless ATM access as developed in this project.

This deliverable describes the special Medium Access Control (MAC) layer needed to bridge ATM technology with the Wireless Physical Medium. This layer is quite important, since it has to cope with the characteristics of the wireless link, as well as the traffic considerations of the ATM layer. Its goal is to keep the Radio Physical layer transparent to the ATM layer as much as possible. This bridging requires several mechanisms in the MAC layer.

The deliverable's structure is presented below:

- Chapter 2 gives a general presentation on the current work in the area of Wireless ATM, as well as the concept of the Magic WAND project.
- Chapter 3 presents the general architecture of the MAC layer, the network environments it is supposed to operate in, the types of networks it can serve and the reference model of the MAC layer.
- Chapter 4 provides information on the addressing scheme used in WAND for the Wireless Connections and the addressing of the equipment used in the wireless network.
- In chapter 5, the Time Frame and Time Slot of the WAND MAC are presented. Moreover, a detailed report on the structure of the packets exchanged between MAC entities.
- Chapter 6 deals with the peer-to-peer control communication of the MAC entities. It includes mechanisms for MT association to the Wireless ATM Network,

mechanisms to handover, mechanisms to share the Radio Channel when needed, etc.

- In Chapter 7 we present how the ATM connections' cells are handled in the MAC layer. Ways to deal with the radio channel errors are presented, along with a scheduling mechanism for ATM cell allocation within the MAC time frames.
- Chapter 8 provides the interfaces of the MAC layer with the Physical layer and the Upper layers. Both interfaces for User information plane and Control information plane are described.
- Chapter 9 provides the papers that are referenced in the deliverable.
- There are five appendices in this deliverable:
  - Appendix I presents the scheduling algorithm used in Magic WAND.
  - Appendix II provides an improved WDLC architecture.
  - Appendix III provides the signals exchanged among the MAC entities and the signals exchanged with the higher and physical layers.
  - Appendix IIV gives the WAND MAC general SDL description.
  - Appendix V provides Message Sequence Charts (MSCs) that indicate the WAND MAC behaviour in the most important scenarios.

1.

2. Wireless ATM Concept

2.1. Background

The state-of-the-art in private and business wireless access networks is rapidly changing from voice/video orientation to a data orientation. The early '90s saw the emergence of the first wireless LAN with data rates in the range of 1-2 Mbit/s. The projected demand for wireless data facilities is growing beyond the simple data sharing function to support real-time multimedia; this requires significantly higher bit rates, up to 25 Mbit/s. Today these systems enjoy a rapidly growing market, in applications requiring the transfer or sharing of data by roaming users [LUPP97], [NIKR96], [RFSB97], [AHKM96], [AYEK96], [RAYC96], [ACAM96], [PETR96], [KUWU96]. Circuit switching is being succeeded by packet switching; the main reason is the better efficiency in handling high variable-capacity demands. Future packet switching is driven by the parallel development of ATM and IP technology. The combination of these two will be able to meet the needs of network operators and users in the near future.

The wide adoption of mobile telephony by both business and private users within their daily routines, necessitates means for unified IP/ATM end-to-end networking, supporting high bit-rates for wireline and mobile terminals. This need is only partially fulfilled by technology today since there are no systems that support mobile broadband services.

The aim of ETSI's Broadband Radio Access Networks (BRAN) Project [BRAN] is to define the standards for service independent broadband radio access networks and systems having a peak rate of at least 25Mbit/s at the user network interface. BRAN networks shall be capable of efficiently carrying existing services, as well as providing the transport mechanism(s) for future services. Both circuit-oriented and packet-oriented transport protocols including ATM shall be supported. The BRAN Project has two top-level objectives:

- To produce specifications for high quality fixed radio access networks as described in the Radio Equipment and Systems (RES); Radio Local Loop (RLL) Co-ordination Group; Survey of ETSI activities and Recommendations for ETSI Work Program
- To produce specifications for high quality business, residential and public access radio access networks as described in Radio Equipment and Systems (RES); HIgh PErformance Radio Local Area Networks (HIPERLAN); HIPERLAN Requirements and architectures for Asynchronous Transfer Mode (ATM) (TR 101 031).

In Europe ETSI Project BRAN is developing a family of standards for wireless broadband communications serving variety of applications, including wireless LANs, mobile wireless ATM access networks and wireless ATM infrastructure networks. These standards operate both in licensed and unlicensed frequency bands. The family standards BRAN is working on consists of four different systems:

1. *HIPERLAN Type 1 - the Wireless LAN* standard refers to a radio communications sub-system which is primarily intended to provide high bit-rate, short distance radio links between computer systems. Typically type 1 will be used for local, on-premises networking. Operation is based on Carrier Sense Multiple Access (CSMA) peer-to-peer (Ethernet based) communication with decentralised architecture. The functional standard is already completed and the approval of Type Approval specifications is in its final stages.
2. *HIPERLAN Type 2 - Wireless ATM Access* standardisation work has started recently and it is active work item at BRAN at the moment. The work aims to standardise wireless access to the fixed ATM/IP networks offering a peak bit rate of 24 Mb/s. The architecture will be a micro-cell like. Type 2 networks will operate in the unlicensed 5 GHz frequency band.
3. *HIPERACCESS - Wireless Access (Local Loop (LL))* is an outdoor system intended for . delivering point-to-multipoint multi-megabit ATM/IP services over distances of up to 5 km. These systems operate both on licensed and unlicensed frequency bands.
4. *HIPERLINK - Wireless ATM Interconnect (ICN)* is intended for short range, high bit-rate interconnection applications, e.g., point-to-point interconnection of ATM switches and wireless access points at data rates up to 155 Mb/s. HIPERLINK is intended for operation in the 17 GHz band. Highly directional antennas will allow high frequency re-use as well as wideband, high bit-rate operation.

The ATM Forum has also envisaged mobility in the ATM world [ATMF]. The ATM Forum has identified the changes needed in ATM and ATM signalling in order to support mobility. It has produced several documents that propose modifications in the ATM specifications. ATM Forum has set requirements on the Medium Access Control used for Wireless ATM. ATM Forum works together with the ETSI BRAN project to achieve common understanding on the MAC/PHY issues needed to support ATM over the air.

The standard developed in the IEEE 802.11 committee for W-LAN, dictates the Medium Access Control (MAC) and the Physical (PHY) layer components of wireless devices. It can be used for ad-hoc and infrastructure networks. It includes specifications for spread spectrum and infrared devices operating in the 2.4 GHz band and at rates of 1-2 Mbps. It should enable interoperability among the embedded components or adapters in mobile devices. It supports ad-hoc as well as infrastructure networks.

- IEEE 802.11 PHY defines physical layers including diffused infra-red (DFIR), direct sequence spread spectrum (DSSS) and slow frequency hopped spread spectrum (FHSS). The spread spectrum techniques are used in the 2.4 GHz band for high throughput, availability in many countries and for relatively lower hardware costs in comparison to the higher microwave frequencies.
- IEEE 802.11 MAC defines mechanisms to provide contention and contention-free access control on a variety of PHY layers (infra-red, RF). The functions within the

MAC are independent of data rates or physical characteristics. However, problems with the IEEE 802.11 exist that mainly stem from the multiple PHYs.

2.2. Magic WAND Concept

The main objective of the project is to develop, and evaluate in realistic user environments, a wireless ATM transmission facility that expands the reach of ATM technology to the ambulant user of premises communications networks [MIKR96], [AWKR96], [MIKK96], [ALAW97], [MAAL98], [PAME96]. This project consists of three major phases. In the first phase, system and component design were undertaken. In the second phase an implementation of the design is produced. The implementation, using mobiles based on portable computers in combination with Access Points serviced by an ATM switch located at a National Host, will be as close as possible to the targeted functionality defined in the design phase. In the final phase of the project, wireless ATM Access will be tested out in realistic hospital and university environments.

Project Objectives:

- To specify a wireless, customer premises, access system for ATM networks that maintains the service characteristics and benefits of ATM networks;
- To demonstrate and carry out user trials with the selected user group and test the feasibility of a radio based ATM access system;
- To promote the standardisation, notably in ETSI, of wireless ATM access as developed in this project.

Wireless ATM transmission will be subject to the vagaries of the radio medium and therefore special radio design measures will be required to offer users an adequate level of service. These measures constitute some of the major technical challenges of this project. Another challenge will be the retrofitting of special call set-up and re-routing features into the Control and Signalling functions developed for the wired broadband world. The output of the project will include a Wireless ATM Access Network Demonstration system, wireless technology that can be commercialised, and the necessary (ETSI, ATM Forum) standards for the wireless subsystem and its support functions within the ATM infrastructure. The UMTS architecture will provide an initial framework for the modelling of the hybrid wired/wireless ATM system. However, as UMTS radio technology and radio standards will not be suitable for wireless ATM, new standards will have to be developed. The wireless ATM radio design and its medium access control functions as well as wireless ATM specific Control and Signalling functions will be contributed to ETSI so that the results of this project can become European standards for wireless ATM systems.

The main benefit of wireless ATM is that it enhances the working effectiveness of people in many occupations and businesses by providing them with location independent and high capacity [20 Mbit/s] access to broadband infrastructure networks. Wireless ATM will allow users to transmit and receive data at realistic data rates and controlled service levels that match those of the wired ATM world. This assurance of service is imperative as long distance ATM services are relatively expensive and users can not be expected to accept service degradation on the final (radio) link.

The main components of a Magic WAND system, as shown in Figure 1, are:

- ***Mobile Terminals*** (MTs), the end user equipment, which are basically ATM terminals with a radio adapter card for the air interface,
- ***Access Points*** (APs), the base stations of the cellular environment, which the MTs access to connect to the rest of the network,
- an ***ATM Switch*** (SW), to support interconnection with the rest of the ATM network, and
- a ***Control Station*** (CS), attached to the ATM switch, containing mobility specific software, to support mobility related operations, such as location update and handover, which are not supported by ATM.

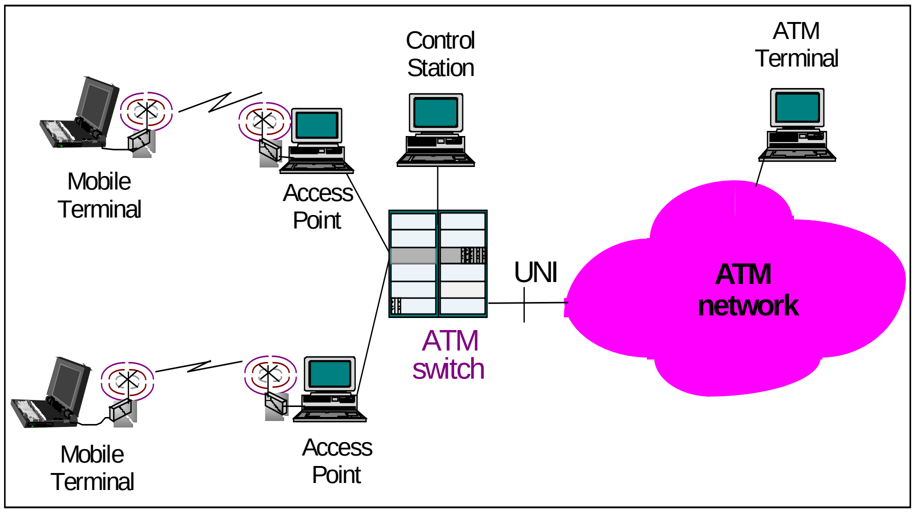


Figure 1 A WAND Network System

1.

3. Medium Access Control Layer in Magic WAND

3.1. Medium Access Control Services

3.1.1. Types of Networks

The WAND system is designed mainly for indoor networks, where we have nomadic users, i.e. their mobility range is quite small. Though, the WAND extensions also consider outdoor environments, where the mobility range in larger.

3.1.1.1. Indoor Networks

Indoor networks, such as customer premises networks, dictate an antenna range of about 50 meters for each AP. This range makes the round trip delay for AP-MT communications almost zero, thus the MAC protocol does not take round trip delay under consideration. Moreover, 50 meters of antenna range and indoor environment, adds some requirements on the MAC layer for efficient handover policies, both for forward or backward handover cases.

3.1.1.2. Outdoor Networks

To achieve higher range, it is better to use directional antennas. Higher range means that the AP and the MT/User might not be very near to each other. Thus, the round trip delay is not negligible. This has an impact on the MAC protocol design, since it has to calculate this extra delay for the AP-MT communication. The MAC protocol should have ways to efficiently allocate the channel to MTs, notwithstanding the delays occurred due to the AP-MT distance.

3.1.2. Types of Applications

The WAND MAC layer should provide services to the ATM layer and ATM applications. Moreover, it should deal with other network applications such as IP applications.

3.1.2.1. ATM Network

The MAC layer has to cope with the ATM connections and the traffic contracts they set with the ATM network during connection setup. Thus, it has several mechanisms to cope with this issue. Some of these mechanisms apply error control, admission control and scheduling of the ATM traffic over the air.

To make this scheduling, the traffic parameters of the connections are passed to the scheduling function. A priority is introduced for each connection, based on its service class:

Priority number	Service class
5	CBR
4	rt-VBR
3	nrt-VBR
2	ABR
1	UBR

The greater the priority number, the greater the priority of a connection. Additionally, a token pool, located at the AP, is introduced for each connection. Tokens are generated at a fixed rate equal to the mean cell rate, and the size of the pool is equal to the “burst size” of the connection. The burst size depends on the characteristics of each connection, and is the maximum number of cells that can be transmitted with rate greater than the declared mean. For every slot allocated to a connection a token is removed from the corresponding pool. In this way, at any instance of time, the state of each token pool gives an indication of the declared bandwidth that the corresponding connection has consumed.

ATM addressing is replaced by another addressing scheme used in MASCARA, which uses reduced headers for terminal and connection identification. This is done because the wireless part of the Wireless ATM network will have a limited number of devices that do not need the large addresses the ATM layer uses.

3.1.2.2. IP Network

Within the WAND project, there has been extensive research on a wireless ATM network exclusively intended for IP traffic [MAKU98], [ALNS98], [HPFM98]. RSVP, IFMP, traffic classifiers and differentiated services were considered as the solution for QoS provision over the IP network. AAL5 over ATM will be used to pass IP application over the unmodified MAC layer. Thus, the RSVP service classes are mapped to ATM classes (that MAC understands and handles) as the following table indicates:

IP Service Class	ATM/MASCARA Service Class
Guaranteed Service	CBR or *rt-VBR*
Controlled Load Service	CBR, *nrt-VBR* or ABR
Best-Effort Service	*UBR*, any ATM class.

3.2. General Functionality of the WAND Medium Access Control (MAC)

A special Medium Access Control (MAC) should be employed, in order to bridge the ATM with the Wireless world. This MAC should be able to provide ATM services transparently, over a wireless link [BMMP96], [PMSB98], [PRDA98]. Thus, the MAC must confront the error prone environment, support bandwidth management over a channel with scarce resources, achieve statistical multiplexing of the sources, an still ensure the Quality of Service (QoS) for every ATM connection. The MAC concept is particularly applicable for the Multi-access channel of the Wireless Link in the scope of this document.

In the WAND system, the multiple access scheme is Time Division Multiple Access (TDMA), using Time Division Duplex (TDD). A Time Frame will consist of several periods. There will be periods for downlink transmission (AP to MTs) and periods for uplink transmission (MT to AP). The TDMA scheme will be dynamic, i.e. the time frame length will vary and there will not be fixed uplink and downlink periods per each time frame. The structure and the content of each time frame will depend on the ATM connections that are active in the area of the Access Point. The connections will use reserved slots for as long as they are active. The number of reserved slots may vary from time frame to time frame, depending on the type of ATM connection. In addition, there will be a contention period in each time frame for new Mobile Terminals that wish to enter the area of the Access Point.

Thus, based on the above, the MAC protocol employed in WAND is named as ***M***obile ***A***ccess ***S***cheme based on ***C***ontention ***A***nd ***R***eservation for ***A***TM, or ***MASCARA***.

The MASCARA protocol must support the following functions: *traffic allocation* according to the traffic contracts of the ATM connections, *error control* as well as *mobility features*.

3.2.1. Traffic Allocation

Each accepted ATM connection has a certain traffic contract, described in terms of both Traffic and QoS Parameters [I.371]. When the ATM cells for this connection are transmitted over the wireless link, the traffic contract should not be violated. The main entity responsible for “microscopic” QoS requirements provision in WAND is the Scheduler. This entity, taking under consideration the various traffic contracts, must build a time frame, allocating slots according to the connection type. Special issues are taken into account such as the TDD scheme, the Cell Transfer Delay of each connection, etc. The Scheduler interacts both with the Radio Resource Manager (RRM) entity and the WDLC entity performing error Control. We should note that RRM belongs to the Layer Management Plane, is responsible for “macroscopic” QoS requirements provision performing the Wireless Call Admission Control (WCAC), and for the WAND Demo will not be part of the MASCARA layer.

3.2.2. Error Control

The wireless link typically provides a high BER; thus, during a transmission of ATM cells, many of them can be corrupted, so that the agreed Cell Loss Rate of an ATM connection may be exceeded [PRDA98], [MEIE98]. An effective and efficient error control scheme must be designed and implemented to handle properly such undesired situations. Since different ATM connections do not have the same requirements, criteria like Cell Loss Rate (CLR) and Cell Transfer Delay (CTD) do not have the same importance for any type of ATM connections. Telnet services are more sensitive to the former (CLR), while Real Time services care more about the latter (CTD).

Therefore, different error control schemes must be applied for different ATM connection types. In WAND Error Control is performed at:

- the MAC Packet Data Unit (or MPDU) level; this takes place at the boundaries of the MASCARA/PHY interface, the MASCARA Data Pump.
- the ATM-cell level; the Wireless Data Link Control (WDLC) entity is responsible for the protocol execution, while the MASCARA Data Pump performs the CRC calculations
- the PHY Layer; this is beyond the scope of this document.

3.2.3. Mobility Features

Since the MAC will support ATM connection to/from Mobile Terminals, Mobility support is one of the main attributes of the MAC to be employed in WAND. There will be situations of a MT passing from the area cell of an AP to another area cell, leading to the need for a handover [KAHA96], [MERA96], [MIHI96], [HAHI96], KADM98]. The MAC layer is responsible for detecting the need for a handover [DRPR98]. By gathering information such as Received Signal Strength Indication (RSSI) and residual errors of the data received, MASCARA checks if the MT should handover or not. If handover is decided, the MASCARA will manage the scanning of neighbouring APs and will initiate the handover procedure, indicating the desired AP for handover. Moreover, there might occur cases when the wireless resources are not adequate to serve the accepted traffic. Then, the AP will force some MTs to handover to other Access Points.

3.3. MAC Architecture Components

The main components the Magic WAND MAC layer are (see Figure 2):

- The **Inter-Control Communication (ICC)**, which allows the MASCARA FE's to interface, for control purposes, with the upper layers. It thus allows cross-layer communications to exchange information between the MASCARA layer and the entities located within upper layers such as at the Q.2931m level for the MT, or the MRR for the AP. Furthermore, this FE is functionally identical in the AP and in the MT.
- The **MASCARA Control FE** which is responsible for the ”non stationary”, control functions of the MASCARA protocol, related or not to the control of the radio modem. For instance this FE is in charge of the association and re-association procedures. It is also involved in the hand-over procedure by receiving (through ICC) some requests coming from upper layer components. This FE differs between the AP (master side) and the MT (slave side). Moreover, this functional entity might contain functions of the Radio Resource Management, such as Wireless Connection Admission Control [ZAND97]. Finally, functions to control the channel and means to share it when interference occurs (use of Dynamic Channel Allocation among different APs) are also part of MASCARA Control.

- The **Control Segmenting Reassembly FE** segments the control messages received from Mascara_Control into cells, and reassembles cells received into a control message delivered to Mascara_Control. This FE is functionally identical in the AP and in the MT.
- The **W-DLC**, which is split into the transmit part **(W-DLC Xmit)** and the receive part **(W-DLC Rcv)**. This FE is responsible for recovering (even partly) from the bad quality of the transmission channel thanks to techniques like ARQ, ACK or CRC. The first part is in charge of building the cell train which will later be sent over the air as a MPDU payload, whereas the second part is in charge of sending to the MAC-ATM interface the cells which were previously received within MPDU payloads. Furthermore, this FE is functionally identical in the AP or in the MT.
- The **MAC Data Pump** manages the slot map according to the connection profile (AAL) and the Quality of Service (QoS) parameters, and sends and receives MPDU to/from the physical layer. This FE contains:

— the MASCARA scheduler and differs thus between the AP (master side) and the MT (slave side).Scheduler [DRMP96]

— Frame Processor; the interface to the physical layer, performs MPDU error control and ATM cell CRC calculations.

— MPDU transmit and receive modules

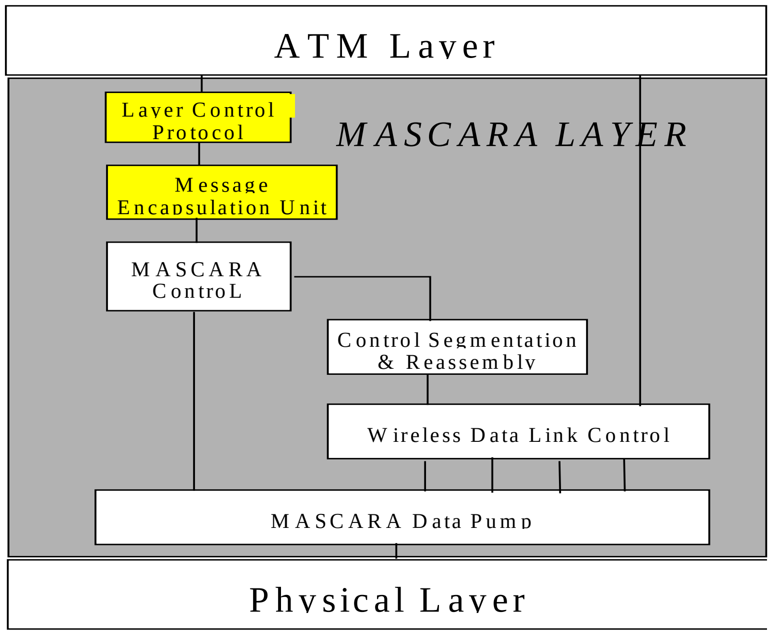


Figure 2 MASCARA Functional Entities

These components communicate with each other to exchange useful information for the DLC operation. Moreover, some DLC components will communicate with some higher layer functions to acquire critical information or to inform them about some incidents (e.g. handover).

The previous MASCARA FE's exchange signals with the external world thanks to the two following interfaces:

- The **MAC-ATM interface**, through which the MASCARA FE's interact with the ATM layers for the reception or transmission of ATM cells. Such ATM cells may either carry ATM bearer services traffic or control traffic (such as some control information exchanged during hand-over procedure). Thus this interface belongs both to the data plane and the control plane. Furthermore, this interface is functionally identical in the AP or in the MT.
- The **MAC-PHY interface**, through which the MASCARA FE's interact with the radio transceiver, either for the transfer (in/out) of MPDU, or for the exchange of control information (such as channel selection, RSSI, transmit power setting or dormant state setting). This interface can thus be seen as belonging both to the data and control planes. Furthermore, this interface is almost identical in the AP or in the MT, as it only differs in the way the time slot clock is handled.

A more detailed description of the MAC components, as well as of the inter-component communication will be given in the following sections.

The MAC components proposed can be matched with the common reference model that the ATM Forum and ETSI BRAN use for wireless ATM (see Figure 3). Some of the MAC Control functions are in the PAC and TAC entities (Access Control entities at the MT and the AP respectively). The W-DLC functionality can be included in the ATM Connection function (ATMC). The RRM and Scheduling functions are implemented within the Radio Resource Control (RRC) function. The Cell Handling functions are mapped to the Radio Transmission and Reception functions (RTR).

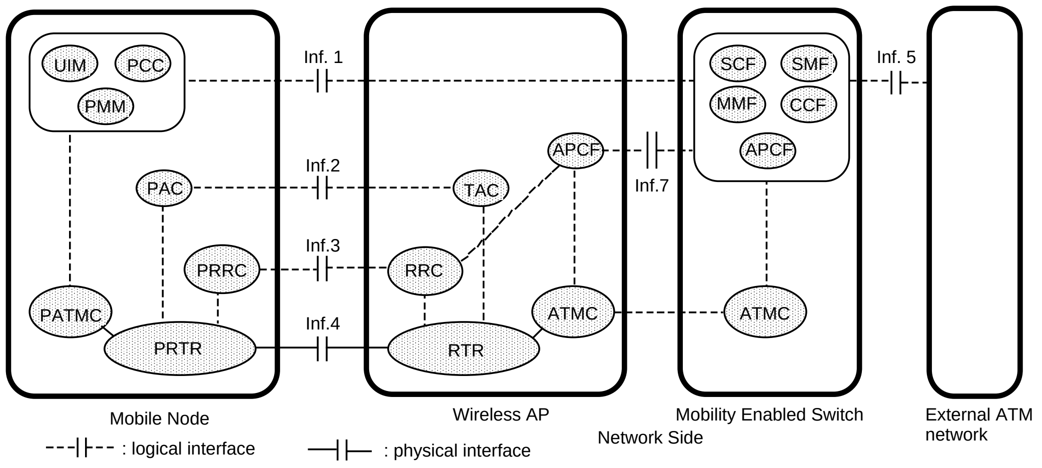


Figure 3 ATM Forum WATM – BRAN common reference model

APCF (AP Control Function)

ATMC (ATM Connection function)

CCF (Call control and Connection control Function)

MMF (Mobility Management Function)

PAC (Portable terminal Access Control function)

PATMC (Portable ATM Connection function)

PCC (Portable Call control and Connection control function)

PMM (Portable Mobility Management function)

PRRC (Portable Radio Resource Control function)

PRTR (Portable Radio Transmission and Reception)

RRC (Radio Resource Control function)

RTR (Radio Transmission and Reception function)

SCF (Service control Function)

SMF (Security Management Function)

TAC (Terminal Access Control function)

UIM (User Identification Management function)

1.

4. WAND MAC Addressing

Addressing in MASCARA can be divided into end-system addressing, for identification of both the MTs and the AP, and connection addressing, for identification of the individual connections (uplink and downlink). The following table shortly presents the identifiers relationship in MASCARA and ATM layers. The relationship is described below.

Identifier	ATM Layer	MASCARA Layer
Terminal Identification	ATM Address	MASCARA Address
Connection Identification	ConnectionIdentifier (VPI/VCI)	MVCI

In the ATM world, end systems are uniquely identified by a 20-byte ATM address that is composed of the network suffix (i.e. the identifier of the directly connected ATM switch; the Control Station in WAND), and the user part (i.e. the identifier of the end system; the MT in WAND). In MASCARA, the information is transferred through MPDUs. Broadcasting in the downlink, and multiple access in the uplink, require the address of both the sender and the receiver in every MPDU header. Trying to transfer the whole ATM address in every MPDU header, in order to identify the sender and the receiver, would significantly decrease the protocol's performance. This performance decrement would occur due to the large number of bytes used to identify a Mobile Terminal and the long mapping tables needed for addressing. Accordingly, a shorter MT MASCARA address is given for naming the MTs at the MASCARA layer. This address is given by the AP to every MT, during the association phase, and is unique in the range of a single AP. Each MASCARA address is 16 bits long, permitting up to $2^{16} = 65536$ MT's to co-exist in the area of an AP. Similarly, all APs, within the same Wireless ATM network, own a unique, 2-byte long AP MASCARA address. In this way, the required space for sender and receiver address in each MPDU header is reduced from 20 bytes to 2 bytes. This exceeds the protocol's performance, without limiting the system's availability, since more than 65536 MTs are almost impossible to co-exist in the range of a single AP. The only additional operation required, is a mapping at the AP, between MASCARA addresses and real ATM addresses.

Moreover, to gain space in the ATM header (to be used for instance for flow information), the VPI/VCI field is replaced, only from the MASCARA layer level of the sender to the MASCARA layer level of the receiver, by a shorter, 8-bit long identifier, called the MASCARA Virtual Channel Identifier (MVCI). The MVCI is unique in the range of a MT, and is granted to each connection, during connection setup. Accordingly, up to 256 connections coming from the same MT can be supported by MASCARA. At the AP, a mapping is done such as the pair (MT MASCARA address, MVCI) would correspond to a unique VPI/VCI in the AP-CS link. A possible mapping could be for the MT MASCARA address to correspond to the ATM VPI, and the MVCI to the ATM VCI. In the downlink direction at the AP, the VPI/VCI fields of each ATM cell, arriving from the ATM layer, is replaced by the corresponding MVCI, and forwarded to the MT with the corresponding MASCARA address. In the uplink, the MVCI of each ATM cell, arriving in the MASCARA layer from a particular MT, is replaced with the corresponding VPI/VCI, before forwarded to the ATM layer.

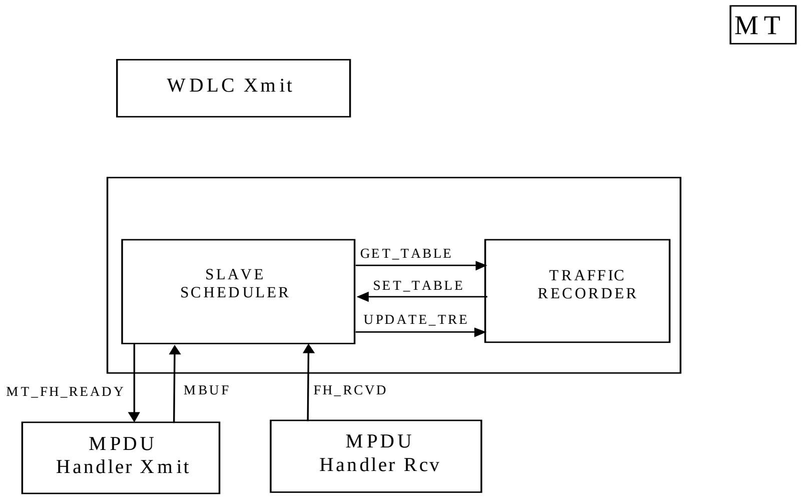


Figure 4 Mapping of (MASCARA_addr, MVCI) to VPI/VCI

Figure 4 depicts how different MTs can have the same MVC values for their connections, without introducing any problem to the ATM layer addressing.

The identifier of a wireless ATM network is the hashed version of its name (the probability for two networks to have the same name or the same identifier is very low).

1.

5. WAND MAC Frame and MPDU Formats

5.1. The Time Slot

The MAC protocol in WAND is based on Time Division Multiple Access scheme, using Time Division Duplex for the communication among the Access Point and the Mobile Terminals. The time in MASCARA is slotted in fixed time slots. Each time slot can accommodate one ATM cell payload, along with a modified ATM header and some Error Control information. The size of each time slot is 54 bytes. The information found in a Wireless ATM cell with length of 54 bytes comprises:

- 4 bytes for the modified ATM header. The modification includes conversion of VPI/VCI to MVC. The modified ATM header structure will be fully analysed in the next sections.
- 48 bytes for the ATM cell payload.
- 2 bytes for Error Control.

Another reason for deciding the size of the time slot is the modulation technique used by the WAND physical layer. WAND uses Orthogonal Frequency Division Multiplexing (OFDM) with 16 sub-carriers as its physical layer modulation technique. The slot size is an integer multiple of an OFDM symbol.

5.2. The Time Frame

The MASCARA protocol is built around the concept of MASCARA Time Frame (TF). A Time Frame is a variable length timing structure during which ATM data traffic and / or MASCARA control information flows are exchanged through the wireless link. The WAND architecture has adopted a centralised approach, where the Access Point decides which ATM connections will be served within the time frame, and the Mobile Terminals accept this allocation. The AP can allocate bandwidth over the wireless link dynamically. Dynamically means that when an ATM connection gets a number of slots in one time frame, it is not sure that it will get the same number of slots in the next time frame. In fact it might not be served in the next time frames. An ATM connection is allocated slots according to the traffic requirements it has at every specific moment. The efficient performance of this latter requires the real-time needs of all (uplink and downlink) connections. As downlink traffic we define the traffic that flows from the ATM network via the AP to the MTs, while uplink traffic is defined as the traffic that flows from the MTs to the ATM network via the AP. Downlink needs are immediately derived from the arriving downlink ATM cells. Uplink needs are either expressed through reservation requests that are piggybacked in the data MPDUs or derived from ATM service and QoS parameters. This technique offers the advantage of avoiding collision, when accessing the wireless communication medium. To deal with the part of traffic that cannot be anticipated by the AP, a contention-based method must be nevertheless considered. In WAND the contention period is based on the slotted-Aloha technique. As contention period suffers from limited channel efficiency (due to collision resolution schemes), the MASCARA protocol limits the use of the contention-based method to the strict minimum. Regarding the downlink traffic, contention can be avoided as the scheduling responsibility is located to the AP; for the uplink traffic, contention is only used to issue reservation requests in order to receive (in a subsequent Time Frame) some reserved bandwidth, or to transmit control information.

The name given to the MASCARA protocol developed in WAND reflects the combination of both reservation-based and contention-based access to the wireless transmission medium: Mobile Access Scheme based on Contention And Reservation for ATM (MASCARA). Each Time Frame is divided into a downlink broadcast period, a downlink traffic period (reservation-based), an uplink traffic period (also reservation-based) and an uplink contention-based period as outlined in the following Figure 5.

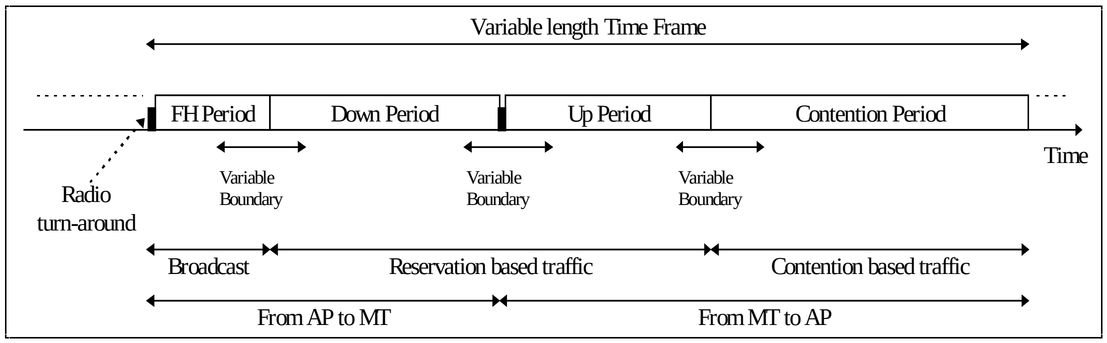


Figure 5 MASCARA Time Frame General Structure

Each of the three periods has a variable length, depending on the instantaneous traffic to be carried on the wireless channel. For the two periods operating in reservation mode, uplink and downlink, it is possible that they collapse to empty periods when no traffic is present. For the contention period, a minimum size is kept to allow any new MT to signal its presence by sending a dedicated control packet. The Time Frame is always beginning with the Frame Header (FH) period, which is used to broadcast, from the AP to the MTs, a descriptor of the current Time Frame. As the size of the Time Frame can change, it is necessary that each MT can learn when each period begins and how long it lasts. Each Time Frame period comprises a variable number of Time Slots.

The physical layer overload of the wireless link is considerably larger than that of wired media. Hence, efficient data transmission can only be achieved if the length of transmitted data packets is not too small. On the other hand, the high BER, characterising the wireless media ask for not-too-large data packets to keep the packet error rate below maximum acceptable values. In the ATM world, the information granularity corresponds to an ATM cell, which is 53 bytes long. This piece of data is considered short compared, for instance, with conventional LAN MASCARA frames (such as IEEE 802.3 or 802.5), so that it would be inefficient to send each individual ATM cell on the air as a single MPDU. Therefore the MASCARA protocol defines the concept of ‘*cell train*’ which is a sequence of ATM cells sent as the payload of a MPDU. More precisely, each MASCARA Protocol Data Unit (MPDU) comprises a MPDU header, followed by a MPDU payload containing ATM cells generated by or destined to the same MT. In terms of duration, the time required by the physical layer to initiate a MPDU transmission (referred to as physical header) plus the time needed to send the MPDU header is equal to the transmission time of a single ATM cell, i.e. to one slot. The details of the MPDU structure will be given in a subsequent section.

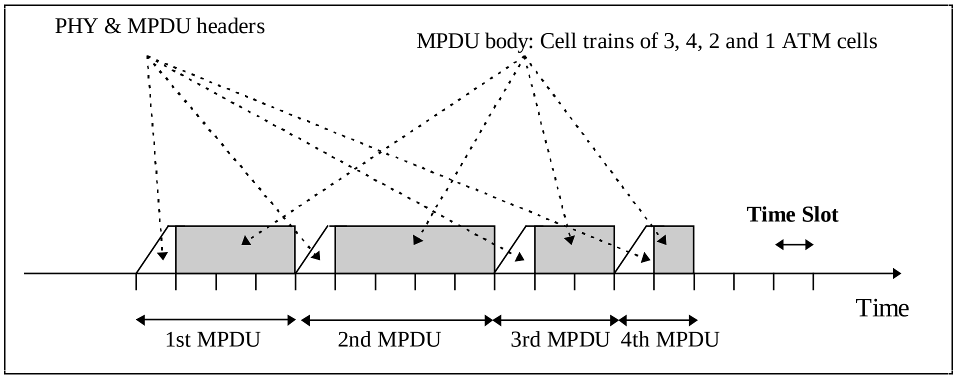


Figure 6 Packing of "cell train" within MPDU

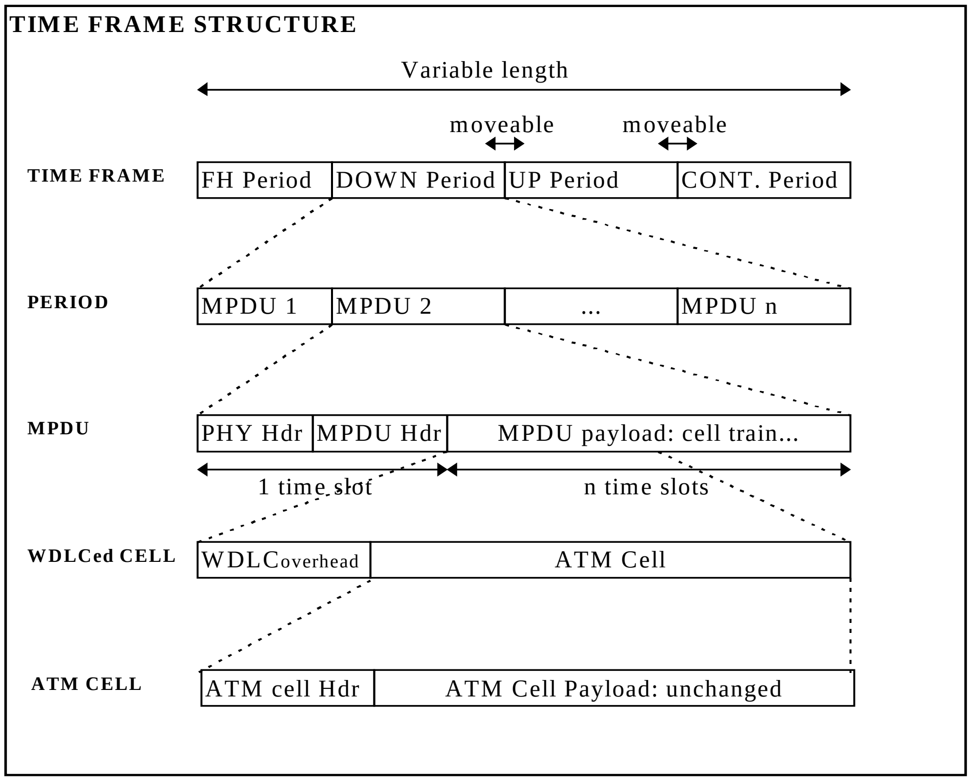


Figure 7 MASCARA TDMA Structure Summary

The time frame of the end-system WAND MASCARA might be variable or fixed. Its default scheme is variable, allowing the MASCARA to set it to fixed length if needed.

Concerning the ATM performance, it appears that variable frame length could be better than fixed frame length. A scheduler using a variable time frame could utilise frame length freedom to accommodate better the ATM traffic on the wireless link, increasing or decreasing the frame length according to the ATM traffic contract needs.

A fixed time frame could restrict the scheduling algorithm to accommodate ATM cells in a ‘sub-optimal’ way.

Variable time frames make the scheduling process more complex, albeit potentially more efficient, since the scheduler has an extra degree of freedom for allocating slots in the next time frame, the time frame length. The frame length is found after the slot map is ready. On the other hand, when the time frame is fixed, the scheduler will do its best to accommodate the traffic within a certain amount of time slots, optimising the allocation for the specific time frame.

As far as system performance is concerned, a fixed time frame would be easier to work with than a variable one. When a MT wants to handover, it should listen to the other APs; if the MT knows when the new AP will initiate its next time frame, the handover procedure could be faster.

In case of Dynamic Channel Allocation (DCA), channel sharing among several APs would be easier using fixed time frames. When a central point controls the channel sharing, it need not take into account the different lengths of the time frames. When the APs need to content for acquiring a channel, the DCA performance would be better if each AP knew the length of the time frames of the other APs. Then the APs would have an idea on when the channel would be available for acquisition.

When a MT looses, for some reason, the frame header, it should search for the next frame header to re-establish its connections with the rest of the network. When the time frame is fixed, the MT knows with more accuracy when to search again for the beginning of the next time frame. Meanwhile it can enter a micro-sleep mode to save some power. When the time frame length is variable, the MT has no indication on when the next time frame will start. In that case, it should scan continuously the wireless link to retrieve the new frame header.

Concluding the issue on variable vs. fixed time frame length, variable time frame approach appears to be better for ATM traffic performance when the AP area cell is at equilibrium (no handovers are performed, DCA is not needed). Several DLC implementation issues and system complexity though, make the fixed time frame length approach also attractive.

Time Frame	*Fixed*	*Variable*
ATM Performance		✓
MAC protocol Complexity	✓	
Handover	✓	
DCA	✓	
AP-MT Communication	✓	

5.3. MAC Protocol Data Unit Structure

The MPDU is the vehicle used to exchange information between peer MASCARA entities across the air interface. It is therefore the piece of information transferred from the sending MASCARA entity to its underlying physical layer for transmission, as well as the piece of information transferred from the receiving physical layer to its upper peer MASCARA entity.

The MPDU comprises a header part (the "MPDU Header") and a payload part (the "MPDU body" or "MPDU payload"). The MPDU Header part contains information mainly relevant to the source and destination stations (such as addressing fields), whereas the MPDU payload carries a cell train made by the concatenation of several WDLC-ed cells. It is possible that a MPDU only contains a MPDU header (the payload being empty), for some situations where only control information carried in the MPDU header must be exchanged (such as reservation requests).

MPDU		
offset(byte)	0 26	27 n.54+80
field	MPDU Header	MPDU Body
length(byte)	27	n.54

It must be noted that the MPDU can carry either MASCARA control traffic or ATM bearer services traffic. Therefore it is the generic vehicle used to exchange any type of information between AP and MTs.

5.3.1. MPDU Header structure

This section describes the format of the MPDU Header. The latter is related to the frame processor interface definition, since it controls the operation of a frame processor. The structure of the MPDU Header depends on the type of the MPDU.

For the header we define the following layout (the numbers above the header are the fields lengths in bytes, the numbers below are the relative position if the bytes in the header). The structure of MPDU header is given in below (whatever MPDU it is).

MPDU HEADER							
offset(byte)	0 1	2 3	4 5	6 20	21	22	23 26
field	Net-Id	SA	DA	Typespecific MPDU info	Type	Length	CRC
length(byte)	2	2	2	15	1	1	4

• Net_Id + SA + DA	6 bytes (2/2/2): 0 to 5
• Type specific MPDU info	15 bytes: 6 to 20

Aforementioned 15 bytes can be divided according to type of MPDU as follows:

  - MVC_info(2)+MVC_info(2)+ reserved(11) for data upstream type
  - frame_length(2) + map_length (2) + iaa_mac_ad (2) + time_to_next_beacon (2) + reserved(7) for FH type
  - MVC_info(14) + reserved(1) for Uplink reservation requests MPDU type
  - WDLC_FB(14)+reserved(1) for WDLC feedback MPDU type
  - Reserved(15) for data downstream type to be determined for other MPDU types
- Type+Length+CRC 6 bytes (1/1/4): 21 to 26

We note that in an FH MPDU, the destination address (DA) is always set to the broadcast value.

5.3.1.1. Frame fields

Network identification (Net-id)

A 2-byte field, which identifies the network that the originator of the packet belongs to.

Source Address (SA)

A 2-byte field, which carries the MAC address of the originator, or source of the packet.

Destination Address (DA)

A 2-byte field, which carries the MAC address of the destination(s) of the packet**.**

Type

A 1-byte field discriminates different types of packets. All the fields are independent of the packet type, except from the 'type' field itself. There are at least three different types of fields, to which we refer in the next paragraphs. They correspond to' type' field values:

MPDU Type	Value
User MPDU up	00000000
User MPDU down	00000001
FH	00000011

The least significant bit is interpreted as direction (0 is UP, 1 is Down). In principle an Uplink FH could be specified. There is no need for this type of packets, for obvious reasons.

5.3.1.1.1. Type specific MPDU info

We must discern between an uplink MPDU and a downlink MPDU, since they require different use of the reserved' field. This is discussed below.

5.3.1.1.1.1. Type specific MPDU info (for FH)

For type = FH, we use structure illustrated below

Type specific MPDU info (for FH)

offset (byte)	6 7	8 9	10 11	12 13	14 20
field	frame_length	map_length	iaa_mac_ad	time_to_next_beacon	reserved
length (byte)	2	2	2	2	7

These fields are required for downlink FHs only. But since there are no uplink FHs, no new type needs to be introduced.

Frame length

A 2 - byte field is an (16-bit) unsigned integer, which counts the number of slots that were allocated within the current frame.

Map length

A 2-byte field is an (16-bit) unsigned integer, which counts the number of slot map entries within a slot map.

Iaa_mac_ad

A 2-byte field is an (16-bit) unsigned integer, which determines the MAC address of the MT that should response to the I Am Alive invitation.

Time to next beacon

A 2-byte field is an (16-bit) unsigned integer, which determines time to next beacon MPDU to be sent.

Reserved

Possible use for the reserved 7 bytes could be for acknowledgements of contention mode MPDU's received by the AP in the previous frame.

5.3.1.1.1.2. Type specific MPDU info (for data up)

The following drawing depicts the layout of the specific MPDU info field for the uplink transmission (data up).

Type specific MPDU info (Data up)

offset (byte)	6 7	8 9	10 20
field	MVC_info	MVC_info	reserved
length (byte)	2	2	11

Strictly speaking, these fields are required for uplink MPDUs only, which motivate the introduction of separate up and down link frames. Besides this allows the MAC to detect error conditions (when a terminals receives another terminal, thinking it is a base station). The MVC field is a 16bit field, which is subdivided in an 8bit MVC field and an 8 bit unsigned integer that quantifies an allocation request.

MAC Virtual Circuit info

The 'MVC' fields are used by MT's to state an allocation request for 0,1 or 2 different MVCs. The MVC field is a 16-bit field, which is subdivided in an 8bit MVC field and an 8 bit unsigned integer that quantifies an allocation request. " MVCI" is placed in Bit 0-7 and the "request" in Bit 8-15 of the MVC info field.

Reserved

An 11-byte field, possible future use of the reserved could for Error Control functions.

5.3.1.1.1.3. Type specific MPDU info (for Uplink reservation requests MPDU):

The following drawing depicts the layout of the specific MPDU info field for the Uplink reservation requests MPDU.

Type specific MPDU info (uplink reservation requests)

offset(byte)	6 7		18 19	20
field	MVC_Info(1)	…	MVC_Info(7)	Reserved
length(byte)	2		2	1

This MPDU type is used to carry uplink reservation request, in case that no uplink traffic exists and thus normal uplink MPDU cannot be used.

WDLC_FB

Seven 2-byte fields, that specifies the uplink reservation requests.

Reserved

1-byte field, possible use not specified.

5.3.1.1.1.4. Type specific MPDU info (for WDLC feedback MPDU):

The following drawing depicts the layout of the specific MPDU info field for the WDLC feedback MPDU.

Type specific MPDU info (uplink reservation requests)

offset(byte)	6 7		18 19	20
field	WDLC_FB(1)	…	WDLC_FB(7)	Reserved
length(byte)	2		2	1

This MPDU is used to carry WDLC feedback info.

MVC_Info

Seven 2-byte fields, that specifies the WDLC feedback.

Reserved

1-byte field, possible use not specified.

5.3.1.1.1.5. Type specific MPDU info (for data down):

The following drawing depicts the layout of the specific MPDU information field for the downlink transmission (data down)

Type specific MPDU info (data down)

offset(byte)	6	20
field	Reserved	
length(byte)	15	

Reserved

To be determined

5.3.1.1.1.6. Length

A 1-byte field, which specifies the number of cells following the header.

5.3.1.1.1.7. Cyclic Redundancy Check (CRC)

A 4 -byte field, which detects, using a code, any errors occurred during the transmission of data. It contains 16-bit CRC, calculated over the Header. It has not been decided which particular CRC generator and initial value is used.

5.3.2. MPDU Body structure

This section describes briefly a proposal of the format of the body, or payload. The payload carries a cell train that is consisted of several WDLC-ed cells. Each of these cells has a length of 54 bytes. For FH, each individual 54 byte long “equivalent cell” is made of 50 bytes of information followed by a 4 byte CRC (with the same polynomial than for the MPDU header). Consensus to have each map slot entry specifying both slot offset and slot length. The last “equivalent cell” of the FH MPDU body is padded with zeros to reach the boundary of a 54 byte long equivalent cell (the assumption being that only the slot map is passed in the payload of the FH MPDU).

The structure of the MPDU body is illustrated in the following drawing:

MPDU Body

offset (byte)	0 … 53	54 … 107	... …	n.54 … n.54+53
field	equivalent cell	equivalent cell	... …	equivalent cell
length (byte)	54	54	... …	54

The MPDU body is consisted of several WDLC-ed cells of 54 bytes long each. There are two types of MPDU bodies.

One for the FH as the following drawing depicts:

Equivalent cell(forFH)

offset (byte)	0 4	5 9	10 44	45 49	50 53
field	slot map entry	slot map entry	...	slot map entry	CRC
length (byte)	5	5	...	5	4

And one not for the FH as the following drawing depicts

Equivalent cell (Not for FH)

offset (byte)	0 3	4 51	52 53
field	Cell Header	original ATM cell or MASCARAcontrol cell	CRC
length (byte)	4	48	2

By observing the drawings above we can conclude to the following statements:

All MPDU headers are CRC4 protected.

MPDU bodies are CRC4 (FH) or CRC2 (not FH) protected for each of their cells

5.3.2.1. Frame fields

5.3.2.1.1. Equivalent cell format (for FH)

For FH, each individual 54 byte long “equivalent cell” is made of 50 bytes of information followed by a 4 byte CRC (with the same polynomial than for the MPDU header). Consensus to have each map slot entry specifying both slot offset and slot length. The last “equivalent cell” of the FH MPDU body is padded with zeros to reach the boundary of a 54 byte long equivalent cell (the assumption being that only the slot map is passed in the payload of the FH MPDU).

The following drawing depicts the format of the equivalent cells for the frame header.

Equivalent cell (for FH):

Offset (byte)	0 4	5 9	10 44	45 49	50 53
field	slot map entry	slot map entry	...	slot map entry	CRC
length (byte)	5	5	...	5	4

Note: for the last equivalent cell, potential padding with zeroes to reach the offset 53.

5.3.2.1.2. Slot map entry format

The slot map entry describes the transmission of a number of MPDU. The following drawing depicts the format of the slot map entry

Slot map entry

offset (bit)	0 15	16 23	24 29	30	29	32 39
field	MAC Addr	MVC	length	more	direction	Offset
legnth (bit)	16	8	6	1	1	8

Each slot map entry is:

Parameter	Size
MAC_Addr	16 bits
MVC	8 bits
length	6 bits
more	1 bit - this field is present, but ignored for the WAND demo
direction	1 bit
Offset	8 bits

MAC Addr

A 16-bit field, which specifies the source MT in case of an UP slot map entry, or the destination MT in case of a DOWN entry. The following special rules apply. There is one special MAC Address, which is interpreted as the broadcast address. The following entries have a special interpretation.

An entry, with MAC_Addr equal to that of the AP is interpreted as void (or NOP). The implementation is trivial, as none of the MT map parsers will never filter out a slot map entry with the AP's MAC_Address. This function is used typically to allow time to turn radio link direction around.

An UP entry, with MAC_Addr equal to a broadcast value is interpreted as CONTENTION. All MTs can engage in a contention mode protocol in the specified slot. The AP always receives in every contention slot.

A DOWN entry, with MAC_Addr equal to a broadcast value is interpreted as a broadcast packet. It is receivable by all MTs. No special purpose for broadcast packet have been defined until now, except from the FH MPDU, which carries the slot map, but is not referred to in the map itself.

***MVC (*MAC Virtual Circuit)**

An 8-bit field, which specifies the virtual connection that the slot payloads pertains to.

Length

A 6-bit field, which specifies the number of payloads slots to be transmitted. In case all slots are transmitted in a single packet, the packet size is one larger than **'length'** multiple.

More

A 1-bit field, which allows more than one *slot map entries* to refer to the same packet. If it has the value 'more', then it specifies that the slot payloads specified in the next entry should be appended to the current packet. In principle the more bit can specify the concatenation of slot payloads with different MAC source or destination address, even though this serves no practical purpose: a packet cannot be sent to / received from different destinations at once. If the **'more'** bit is set, then the **'MAC_Addr'** and **'direction'** fields are ignored

more bit	value
no more	0
More	1

Direction

A 1-bit field, that determines the direction of the slots. It specifies whether an the payload cells will be sent from AP to MT (DOWN), or from MT to AP (UP).

Direction bit	value
UP	0
DOWN	1

If the bit has 0 as a value then it is referred in the uplink transmission otherwise is referred in the downlink transmission (for value equal to 1)

Offset

An 8-bit field, which specifies the slot in which the MSDU transmission will start. If the indicated entry describes a "**more**d" payload, then the payload can be sent immediately, other wise an MPDU header and a PHY header () have to be transmitted in this slot, followed by the payload in the next slot.

5.3.2.1.3. Equivalent cell format (not for FH)

For other MPDU (either carrying ATM data or MASCARA control information), each cell constituting the cell train is protected by a 2 byte CRC field, the body is 48 bytes long and the header is 4 byte long as follows:

- MVC_info (16 bits)
- WDLC info (8 bits)
- Reserved (4 bits)

- PT (3 bits)
- CLP (1 bit)

The following drawing depicts the format of the equivalent cells not for the frame header.

Equivalent cell (Not for FH)

offset (byte)	0 3	4 51	52 53
field	Cell Header	original ATM cell or MASCARAcontrol cell	CRC
length (byte)	4	48	2

Original ATM cell (or MASCARA control cell)

A 48-byte field, which contains the ATM cell or the MASCARA control cell.

CRC (Cyclic Redundancy Check)

A 4 -byte field, which detects, using a code, any errors occurred during the transmission of data.

Cell Header

The cell header is subdivided in two parts. One is the WDLC-ed coverhead and the other is the ATM cell header. The following drawing depicts the format of the cell header:

Cell Header

offset (bit)	0 15	16 23	24 27	28 30	31
field	WDLC_Info	MVC_info	Reserved	PT	CLP
length (bit)	8	16	4	3	1

Wireless Data link Control (WDLC) Info

An 8-bit field, which controls information for WDLC cell's numbering and acknowledges received cells.

WDLC Info

Offset	16 19	20 23
field	SN	RN
length	4	4

MAC Virtual Circuit (MVC) Info

A 2-byte (16-bit) field, which is subdivided in a 1-byte MVC field and in a 1-byte unsigned integer that quantifies an allocation request. It determines the virtual channel from MAC to MAC communications.

MVC info		
Offset(bit)	0 7	8 15
field	MVC id	Request
length(bit)	8	8

Reserved

To be determined.

Payload type(PT)

A 3-bit field which specifies the payload type of a regular ATM cell header.

CLP(Cell loss priority)

A 1 bit field, which specifies the ATM cell loss priority.

1.

6. Peer-to-Peer MASCARA Protocol

This section deals with the MAC control functions for the peer AP and MT MASCARA layers. These functions perform tasks that are highly dependent on the mobility of the ATM terminals, as well as tasks needed for synchronisation.

6.1. MCL

Due to the unbalanced functionality of MASCARA Control between the AP side and the MT side, the following description is split into two separate parts, each of them corresponding to a given station.

6.1.1. AP-MASCARA Control (MCL)

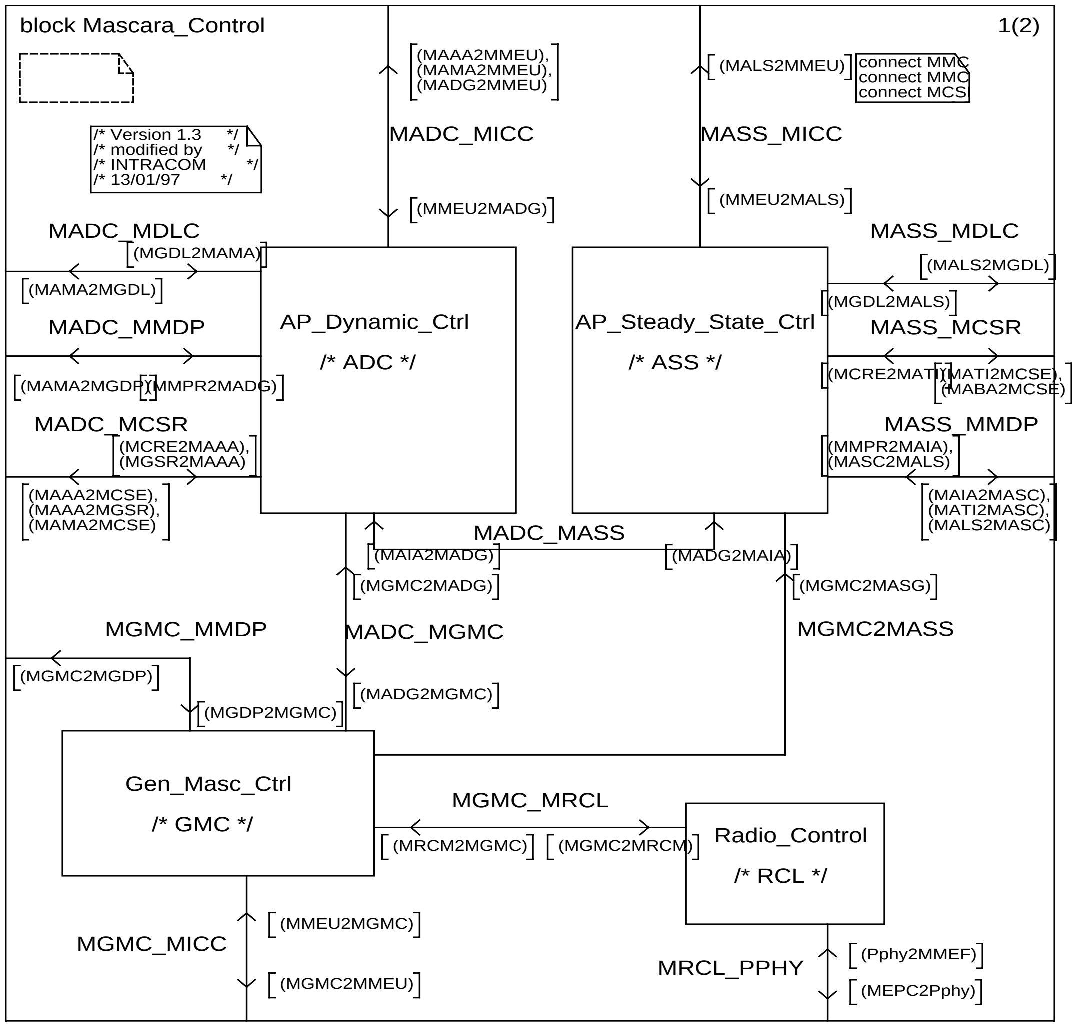


Figure 8 AP-MASCARA Control.

6.1.1.1. The AP Generic Mascara Control (AMGMC)

The AMGMC is the entity responsible for the AP to start in time and synchronously other MASCARA entities at startup and closing them at the end.

6.1.1.1.1. Startup

Upon the reception of the signal **OPEN_AP_MAC_req** coming from the Control, the AMGMC process sends the following signals to initialise and start MASCARA:

INIT_RCL starts the Radio Control, which is the very first entity to be initialised.

INIT_MDP starts the Mac Data Pump.

INSTANCIATE_CSR instanciates processes in the block responsible for Segmenting and Reassembly packets for the Broadcast channel.

SET_TRAFFIC_PARM requires the MDP to allocate the necessary resources for the Broadcast channel.

Notice here that the AMGMC includes the behaviour of the Broadcast MVC Agent. This is necessary since the Broadcast channel is to be used before the end of all MASCARA entities initialisations.

INIT_ADC initialises the block responsible for the management of associations and handling of connections.

INIT_ASS starts the block containing most permanent actions to take place once the setup is terminated.

At the end of startup, the GMC sends **OPEN_AP_MAC_cnf**, a confirmation signal, to the Control.

6.1.1.1.2. Shutdown

Upon the reception of the signal **CLOSE_MAC_req** coming from the Control, the AMGMC process sends signals to close the opened entities:

AP_DEASSOCIATE deassociates all MTs and closes each connection.

CLOSE_ASS closes all permanent entities which are no more useful.

TRAFFIC_RELEASED asks the MDP to release the allocated resources bound to Broadcast channel.

DESTROY the segmenting-reassembly instances.

CLOSE_MDP stops all MDP activity.

CLOSE_RCL the radio can stop, Mascara has closed.

At the end of shutdown part, the GMC sends CLOSE_MAC_cnf, a confirmation signal, to the Control.

6.1.1.2. The AP Dynamic Control (AMADC)

The various functional entities defined in this set are dynamically created according to services provided by the AP MASCARA layer [DBBM98]. Such entities are thus instanciated. The AMADC block is composed of three different kinds of SDL processes organised in a tree-oriented manner, as depicted in the following Figure 9.

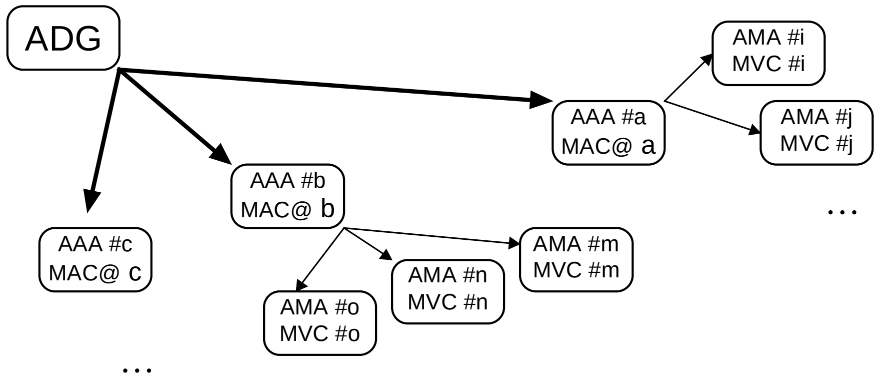


Figure 9 Creation of AP Dynamic Control Functional Entities.

6.1.1.2.1. The AP Dynamic Generic Agent (AMADG)

This entity is in charge of creating / killing the AAA instances. It also dispatches the messages to the right AAA instance. Moreover, it dispatches also the initialisation requests coming from the AMGMC.

6.1.1.2.2. The AP Association Agent (AMAAA)

This entity is in charge of creating / killing the AMA instances. It also dispatches the messages to the right AMA instance. It performs the association (see section on the Association). Per MAC address, there is one couple of AMAAA - MMMAA.

6.1.1.2.3. The AP MVC Agent (AMAMA)

This entity controls a MVC. More precisely, it is created / killed when a new connection is to be set up. During its lifetime, it will make sure that all necessary resources for the connection are available and correctly set.

There are three predefined AMAs that are automatically created and activated when a MT is just associated: the MASCARA control AMA, the Q.2931 AMA, and the Mobility AMA.

6.1.1.2.4. AP Assisted HandOver entity (APAHO)

Handover within the Magic WAND was originally designed to be mobile initiated. Handover is executed due to bad channel quality and deterioration of the wireless data link. However, there are some cases where the AP may decide that handover is needed. For example, when there is no bandwidth to accommodate an already accepted ATM connection, a MT cannot initiate a handover since it is not aware of this situation. Then the network resources are congested, i.e. the input traffic is more than the network can accommodate [KOMM95], [TAHP94], [KADM98], [DRPR98]. In such a case, the AP could force one or more MTs to handover to neighbouring APs. This mechanism is called AP Assisted HandOver (APAHO).

The main reason for an AP not to be capable of accommodating already accepted connections, is that some bandwidth is reserved for the ATM connections' information, and for overheads of the MAC layer. The overheads that we are concerned with, are those that can be 'translated' in extra bandwidth requirements (or time consumption). These overheads are due to *DLC Information(Headers, DLC Signalling)*, *physical layer headers, error control information* and ATM cells *retransmissions*.

These overheads are traceable within the DLC layer of the AP that can trace, measure and make the appropriate actions needed to overcome such problems. The functional entity that will implement this mechanism will interact with some other entities in the AP-DLC layer, to obtain useful information for the decision mechanism. These entities are the AP Scheduler and the AP Link Status Recorder, which is a database within the DLC layer. The traffic contract of each connection should be mapped to some bandwidth requirement values for the connection (mean, least, maximum bit rate). This can be done in conjunction with the Wireless CAC entity (APCP). For each MT, the APAHO entity will keep some information about its ATM connections. The information kept for each connection is: *Traffic contract parameters* (including mean and max bit rate requirements), *overall bandwidth reserved*, *bandwidth used for retransmissions*, *bandwidth used for the 'Net' connection requirements, retransmission over 'Net' bandwidth percentage*, *number of discarded cells, number of cells received erroneous*. Note that the term 'Net' bandwidth refers to the bandwidth allocated to an ATM connection from the fixed CAC mechanism.

Moreover, some other parameters will also be used to calculate the remaining bandwidth within a cell area. These will be: *Number of slots allocated for frame headers*, *number of slots allocated for the contention period, number of MPDUs per ATM connection*, *number of erroneous received MPDUs per ATM connection*.

The gathering of the aforementioned parameters is done periodically. These parameters can be used to calculate the available bandwidth of an AP cell area. When the parameters are available, the parameters kept for the connections are also calculated for the individual MTs, according to the connections they serve. After this step, the bandwidth consumed by the frame headers and contention slots are calculated. Then the bandwidth wasted by erroneous MPDUs is calculated.

Each *bandwidth value* is subtracted from the *available bandwidth* the AP has to serve the ATM connections. The initial value of the available bandwidth is set to 25 Mbps for every AP. When all the bandwidth values are subtracted, the final available bandwidth value for that specific time is derived. If this value is below a threshold, then a congestion alarm is initiated and the APAHO mechanism must find which is (are) the MT(s) that should be forced to handover to neighbouring cells. Then, appropriate signals should be sent to these MTs, to make them initiate their handover procedures.

The selection of the appropriate MT to be forced to handover is the most important part of the APAHO mechanism. There are several parameters that have to be taken into account for this selection. The APAHO mechanism calculates how much bandwidth each MT is using, in the following parameters: overall bandwidth reserved, bandwidth used for retransmissions, bandwidth used for the network connection requirements, retransmission over network bandwidth percentage, number of discarded cells and number of cells received erroneously.

The MTs are then sorted according to the bandwidth they consume. The most important parameters that indicate the bandwidth dedicated to each MT are the overall bandwidth, the network bandwidth, and the retransmission over overall bandwidth percentage. The first two parameters indicate roughly how much bandwidth is reserved for a specific MT. The third parameter implicitly indicates the MT-AP channel quality. If the channel quality is bad, then many ATM cells may need to be retransmitted, using extra bandwidth.

When congestion occurs, the AP calculates the bandwidth values used by the MTs. Then the APAHO entity selects the MT with the highest overhead consumption, to force it to handover. Before sending the appropriate messages to that MT, APAHO checks if the remaining MTs also cause congestion in the cell area. If this is the case, then the MT with the second highest priority will also be forced to handover. This procedure continues until no congestion is caused within the AP. Special DLC signalling is used to send a HO_Force signal from the APAHO entity, located at the AP, to the peer Handover entity, located at the MT.

6.1.1.2.5. Wireless Connection Admission Control (WCAC)

In general CAC is responsible for deciding whether the network can serve a new connection, while the traffic contracts of the already established connections are preserved [GAPU96], [SAIT92], [RSKJ91], [EMW95], [PEEL96], [KEWC96], [ELMI93], ABSO94], [CCYC97]. In Wireless ATM Networks, two Connection Admission Controls exist. The first one is located in the switch that the APs are connected to and it performs the functions that any fixed network CAC would do to check if the new connection could be admitted in the network. The second one is located in the RRM entity of each AP and it takes into account the special issues that the MAC layer introduces in the wireless link of the ATM communication path [HYLP93]. Some of them are the TDMA/TDD scheme that our DLC will use, the overheads that are introduced by the DLC protocol, etc. Thus, when a new connection requests set-up, WCAC will decide whether to accept it or not based on the connection’s pure ATM characteristics and the characteristics of the MAC protocol.

6.1.1.2.6. AP Dynamic Channel Allocation entity (ADCA)

Systems in unlicensed environments without frequency planning, frequently suffer high interference from adjacent Access Points using the same frequency [MSAD98].

The application of a Dynamic Channel Allocation in the WAND system will result in efficient capacity allocation and, at the same time, solve interference problems.

DCA Schemes

- Centralised

In the Centralised DCA approach, the frequency channel allocation is performed by a Central Point (CP) which can be the Switch, an AP, or, depending on the physical implementation, the Central MASCARA Unit.

One of the advantages of the Centralized allocation policy is that it is easy to be changed when needed, as the change will only occur in one node of the network. Also, note that the APs need not be synchronized. However, knowledge of the overall network requirements is needed and thus there is increased processing load, added overhead for frequency allocation and signaling information.

APs have to send information frequently to the CP which will coordinate the time-based frequency allocation thus allowing APs to use the frequency channel for their time frames. The CP should estimate the maximum time frame duration for all APs based on their loads.

The CP should have the option to communicate with other WAND systems for frequency sharing.

- Distributed

In this case the DCA mechanism is not performed in a single location, but it is executed independently within each AP. Each AP will decide on which frequency channel to use and for how long. In a distributed DCA scheme the complexity of the implemented mechanism is expected to be reduced, compared to the centralized approach, since the mechanism is executed only in each AP, and only for a limited section of the wireless network.

Coordinated and *uncoordinated* approaches exist for distributed DCA. The former allows information to be exchanged between APs in order to decide on channel allocation. In the latter case no such inter-AP communication exists.

6.1.1.3. AP Steady State Control (ASS)

The various functional entities defined in this set correspond to static (as opposed to dynamic) control functions running during regular steady state operations of the AP Mascara Control.

6.1.1.3.1. AP Steady State Gen Ctrl (ASG)

This FE is in charge of initialising the other FE’s beneath the ASS, by providing them with operational parameters.

6.1.1.3.2. The AP I Am Alive Agent (AMAIA)

This Functional Entity running in the AP is responsible for detecting any MT whose association has been lost without explicit notification.

6.1.1.3.2.1. AMAIA Functionality

The AMAIA is initialised by the INIT_AIA signal sent by the Steady State Gen Control FE. It then waits for at least one MT to be associated in order to proceed. The association/deassociation of the MTs are taken into account thanks to the MT_ASSOCIATED/MT_DEASSOCIATED signals sent from the ADG.

Upon the reception of any valid MPDU by the MPR, a MT_ALIVE signal will be received by the AMAIA from the MPR. The AIA then concludes that the corresponding MT is still alive.

The signal FRAME_START, forwarded by the Master Scheduler, informs the AIA of the beginning of a new time frame. The AIA then starts to check all of its associated MTs. If there is a MT for which the AIA has not recently received a MT_ALIVE signal, a SEND_IAA_INVITATION signal will be sent to the master scheduler carrying the MAC address of the MT. This signal is used to force the MT to respond by sending a MPDU. If after several transmissions of the SEND_IAA_INVITATION signal, there is still no response from the MT, the IAA considers the MT as lost. It thus sends a MT_LOST signal to the ADG. There is at most, one IAA invitation sent per time frame.

The two signals MT_SLEEP / MT_WAKE_UP are sent from the TIP (Temporarly Incommunicado Protocol) Agent. These signals are used to notify the AIA when a MT goes incommunicado and when it comes back. This implies that the AIA must not check the state of the MTs which are in the sleep mode.

The signals from/to AMAIA are shown in the following Figure 10:

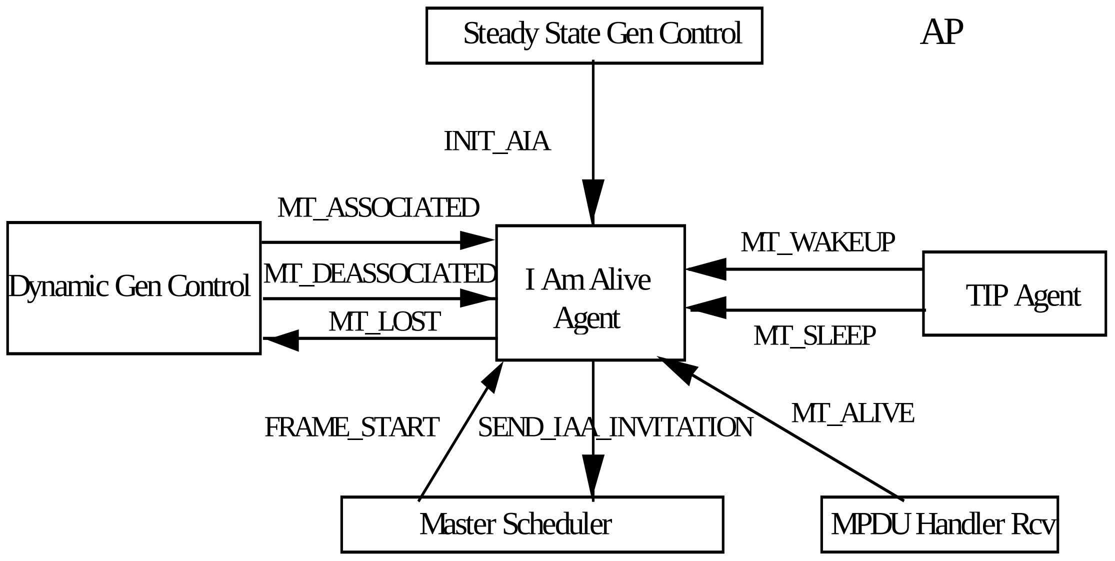


Figure 10 I Am Alive Functional Entity Signal Exchange.

6.1.1.3.3. AP Beacon Agent (ABA)

This FE running in the AP is responsible for broadcasting at regular intervals the Beacon control MPDU used to assist the radio environment procedure of a new MT, as well as the hand-over procedure of an associated MT. Moreover this FE receives from the CON layer some information related to neighbouring AP, which is transmitted within the payload of the Beacon MPDU. This FE operates with its peer entity in the MT (MT BEACON AGENT).

This functional entity collects information about the neighbour Access Points, such as Radio Frequency operation, the MAC address of the neighbour AP. This beacon MPDU is sent to the MTs, at regular time intervals, The AP scheduler knows when this MPDU is to be transmitted, and it announces the time to next beacon to the MTs, via the frame header.

6.1.1.3.4. AP Tip Agent (ATI)

This FE running in the AP is responsible for running the TIP protocol, that is identifying when any MT goes incommunicado and when it comes back. For these reasons it informs accordingly the MAC DATA PUMP.

When a MT requires to get incommunicado, the AP TIP Agent informs the Master Scheduler for this incident. The Master Scheduler checks if the data flow for this MT can be interrupted and if yes, it orders the AP TIP agent to send a control MPDU to the specific MT, indicating that TIP procedure may start. At that time, the Master Scheduler stops allocating slots for the specific MT. If the Master Scheduler decides that TIP cannot be performed, the AP TIP Agent issues an MPDU to the specific MT, indicating that TIP may not be started.

6.1.1.3.5. AP Link Status Recorder (ALS)

This FE is in charge of compiling and reporting on request status information regarding the current state of the transmission channel. It handles the RR_STATUS_req/cnf and GET_INFO_req/cnf signals. As this FE is a MASCARA focal point for various status information, it may report, if justified, alarm information to upper layers thanks to the ALARM_ind signal).

6.1.1.4. Radio Control (RCL)

The **Radio Control** is responsible for the “non stationary” control functions of the MASCARA protocol related to the radio modem. For instance this FE is in charge of receiving RSSI information, or of putting the RF modem in dormant state.

6.1.1.4.1. Emitted Power Control (EPC)

This FE is responsible for driving the RF modem emitted power, according to policies not yet defined.

6.1.1.4.2. Measurements Functions (MEF)

This FE is responsible for recording the RSSI information provided by the RF modem upon reception of any valid MPDU. Such measurements will be used for instance to control the emitted power.

6.1.1.4.3. Radio Control Manager (RCM)

This FE is responsible for driving the RF modem control functions, such as the selection of the current frequency channel. Further design is needed to identify all the functions covered by this entity.

6.1.2. MT-MASCARA Control (MCL)

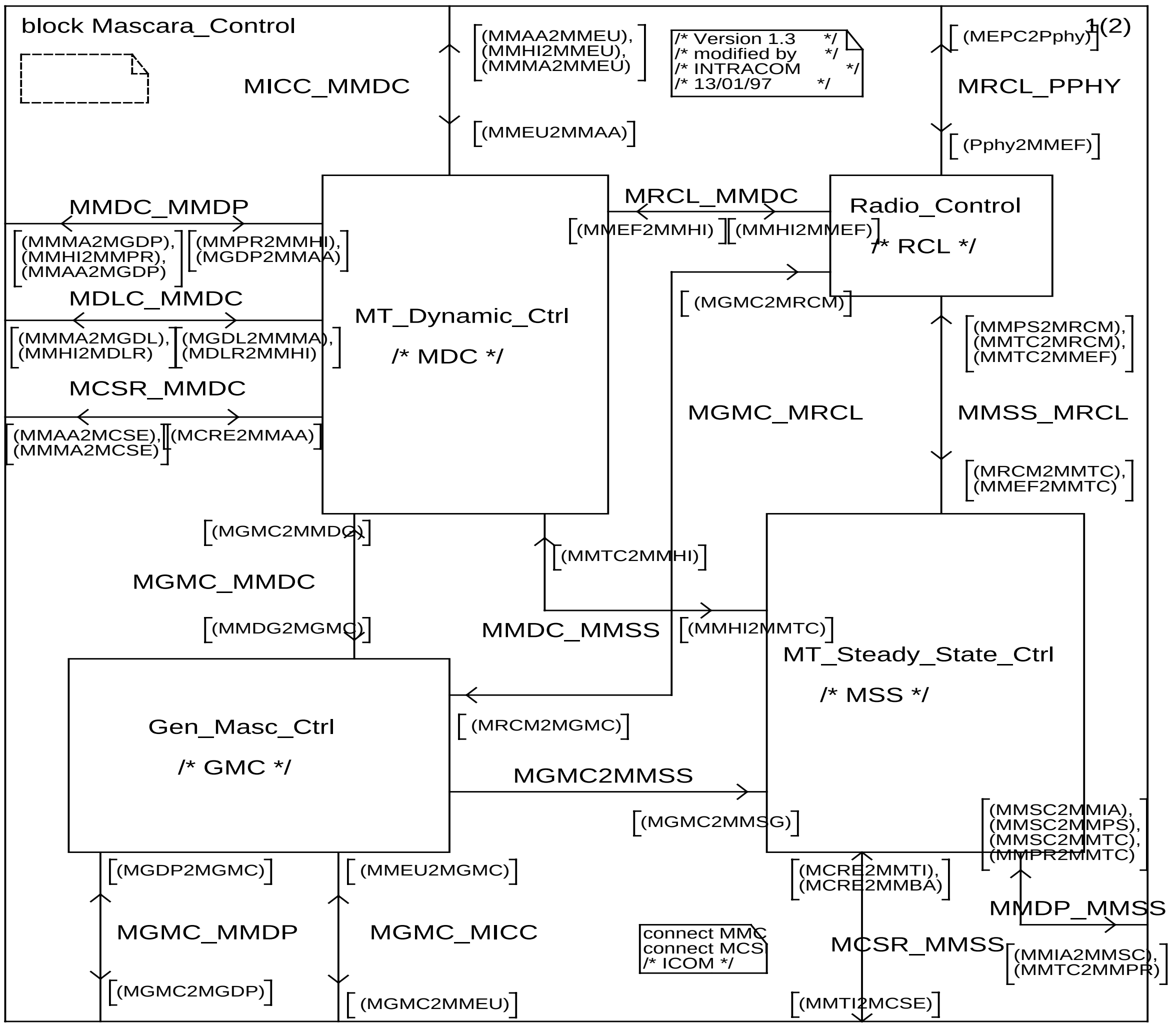


Figure 11 MT-MASCARA Control.

6.1.2.1. The MT Generic Mascara Control (MMGMC)

The MMGMC is the entity responsible for the MT to start in time and synchronously other MASCARA entities at startup and closing them at the end.

6.1.2.1.1. Startup

Upon the reception of the signal **OPEN_MT_MAC_req** coming from the Control, the MMGMC process sends the following signals to initialise and start MASCARA:

- **INIT_MDP** starts the Mac Data Pump.
- **INIT_RCL** starts the Radio Control, which is needed for nearly every action.
- **INIT_MDC** enables the management of associations and the handling of connections.

- **INIT_MSS** initialises processes in the block containing most permanent action to take place once the setup is terminated.

Notice that we don't have to INSTANCIATE a CSR, because we only have one instance of CRE/CSE at the MT side. Moreover, there is no SET_TRAFFIC_PARM because the broadcast channel initialisations are done directly by the CRE.

At the end of startup part, the GMC sends OPEN_MT_MAC_cnf, a confirmation signal, to the Control.

6.1.2.1.2. Shutdown

Upon the reception of the signal **CLOSE_MAC_req** coming from the Control, the MMGMC process sends signals to close the opened entities:

- **MT_DEASSOCIATE** we must deassociate all MTs and close each connection.
- **CLOSE_MSS** all permanent entities are no more useful.
- **CLOSE_RCL** the radio can stop, Mascara will be closed.
- **CLOSE_MDP** stops all Mac Data Pump activity.

At the end of shutdown part, the GMC sends CLOSE_MAC_cnf, a confirmation signal, to the Control.

6.1.2.2. The MT Dynamic Control (MMMDC)

The various functional entities defined in this set are dynamically created according to services provided by the MT MASCARA layer. Such entities are thus instanciated.

The MMMDC block is composed of three different kinds of SDL processes organised in a tree-oriented manner, as depicted in the following Figure 12.

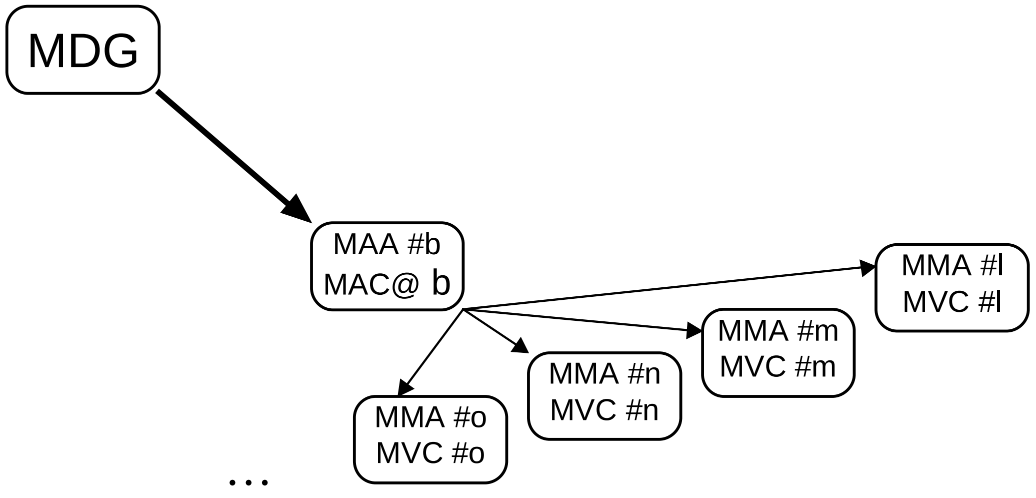


Figure 12 Creation of MT Dynamic Control Functional Entities.

6.1.2.2.1. The MT Dynamic Generic Agent (MMMDG)

As there is only one unique instance of MAA, the MDG only dispatches the initialisation requests coming from the MMGMC.

6.1.2.2.2. The MT Association Agent (MMMAA)

This entity is in charge of creating / killing the MMA instances. It also dispatches the messages to the right MMA instance. It performs the association (see section on the Association). Per MAC address, there is one couple of AMAAA - MMMAA.

6.1.2.2.3. The MT MVC Agent (MMMMA)

This entity controls a MVC. More precisely, it is created / killed when a new connection is to be set up. During its lifetime, it will make sure that all necessary resources for the connection are available and correctly set. There are three predefined MMAs that are automatically created and activated when a MT is just associated : the MASCARA control MMA, the Q.2931 MMA, and the Mobility MMA.

6.1.2.2.4. The Association

In WAND, the association is the succession of actions necessary for a MT to set up the different control channels (MASCARA, Q.2931 and M) between itself and a hosting AP. This is to be differentiated from network registration (conveyed directly to the switch and CS through the Q.2931 and M channels) which is the next step to allow the MT to have data connections.

Association, which has the form of a MT-initiated classical four-way handshake protocol, takes place after a MT power-on, or a backward / forward handover. No security control is performed during the association phase. This is due to the fact that on one hand, security procedures are a mandatory step in the network registration and on the other hand, security in the association would not prevent a denied MT to act as a radio jammer, i.e. to illegally use AP-cell radio resources. The following Figure 13 represents two associated MTs with data connections and an associating MT.

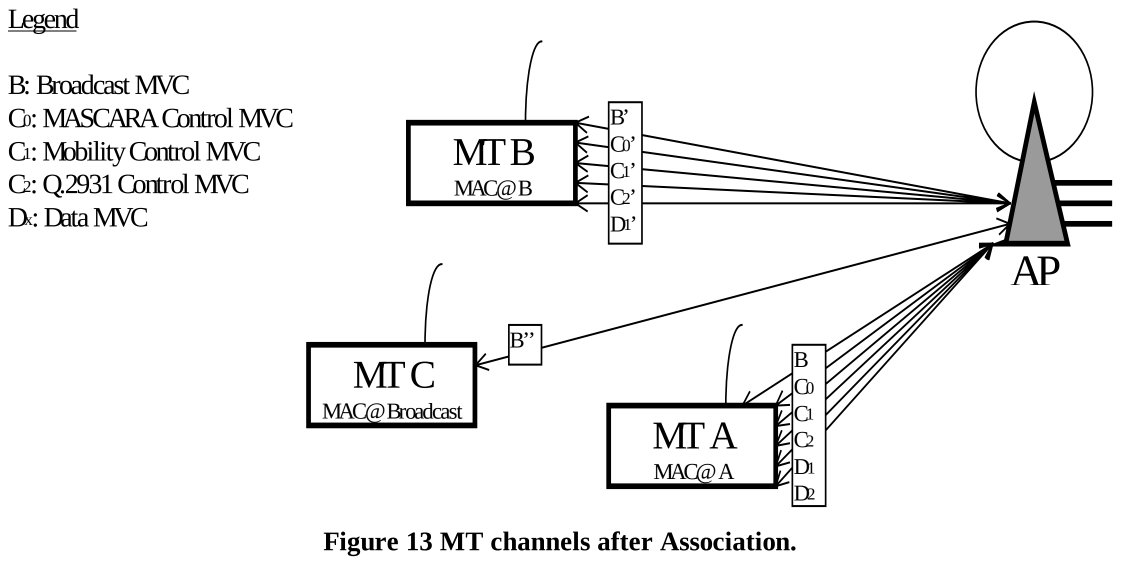


Figure 13 MT channels after Association.

6.1.2.2.4.1. Power-on Association

The following scenario sums up the first association to be initiated by a MT :

The MT is powered-on and MASCARA is initialised.

The unique MMMAA has default MAC@ = BroadcastMAC@.

The three control MMMMAs are created and the MT can then now listen to a potential AP (the Broadcast channel can be used).

The MT scans different frequencies so as to select an AP.

The MMMAA sends an association demand to the AP.

The AMADG receives the demand and creates an AMAAA with a corresponding MAC@i.

The newly created AMAAA creates the three control AMAMAs and answers the MT in giving it MAC@i.

The MT receives it and changes its MAC@ to MAC@i.

The MT acknowledges the AP message using its new MAC@i.

The AP finally replies in giving some last information.

The MT is now associated, network registration can thus be initiated then data connections can be set up.

6.1.2.2.4.2. Handover Association or Re-Association

The following scenario describes the re-association to be initiated by a MT due to a forward handover (FHO) or a backward handover (BHO):

The MT experiences link outage with its old AP (FHO) or de-associates with the old AP (BHO).

The old AP releases all resources previously associated with the MT.

The unique MMMAA sets its MAC@ to BroadcastMAC@.

The MT scans different frequencies so as to select a new AP (FHO) or knows which new AP it has to re-associate with (BHO).

The MMMAA sends a re-association demand to the new AP.

The AMADG receives the demand and creates an AMAAA with a corresponding MAC@j.

The newly created AMAAA creates the three control AMAMAs and answers the MT in giving it MAC@j.

The MT receives it and changes its MAC@ to MAC@j.

The MT acknowledges the AP message using its new MAC@j.

The AP finally replies in giving some last information.

The MT is now re-associated, network location update can thus be initiated then data connections can be reactivated.

6.1.2.2.5. MT Handover Indication (MHI)

This functional entity is responsible for checking the MT environment and for deciding whether a handover is needed or not. To take this decision, several parameters are being monitored [DRPR98].

When an MT moves in the area cell of an AP, the Handover Indicator (HI) process checks three parameter values: the current Radio Signal Strength Indicator (RSSI) value, the MPDU errors occurred and the Cell errors occurred during a certain period. This checking is done over this certain period basis. A timer is set, and when it expires the HI asks for RSSI value from the Measurement Functions (MEF) process, for MPDU Errors from the MPDU Receiver (MPR) process, and for the Cell Errors from the WDLC Receiver (DLR) process.

Note that the RSSI value represents the most recent packet received from the AP. The MPDU Errors and the Cell Errors are integer numbers that correspond to the MPDUs and Cells that have been received erroneous within the period of the timer life. All these values are gathered, and a procedure will decide whether a handover is needed. This procedure should have several threshold values for each parameter as well as a certain algorithm for deciding whether a handover is needed or not.

Handover could be backward or forward. If the old AP is unreachable, then forward handover should be made. If the RSSI is bad but the old AP is still accessible, backward handover will occur.

6.1.2.2.5.1. Handover Indication Procedure

HI has first of all to be initialised when the Mobile Terminal is switched on, with INIT_MHI. Then, the process waits the MT to be associated with an Access Point. After the association phase is completed, MHI receives ASSOCIATION_COMPLETED and the handover indication mechanism is set. Note that HI assumes that the handover type is backward by default.

Firstly, a timer is set, corresponding to the period the HI process checks the radio environment of the AP it is associated with. Then three events might happen. The first two belong to the same category:

1. The HI process must stop working some times. This happens when the MT switches to other AP frequencies in order to scan their radio environments. When this procedure starts, the HI has to be suspended with SUSPEND_MHI and to be resumed with RESUME_MHI when the MT turns back to the AP it is associated with.
2. When the timer expires, the HI scans the environment of its Access Point. HI collects information about the RSSI, the number MPDUs and the number of cells that were received erroneous during the timer period, using signals MEASURE_RLQ_req, MPDU_ERRORS_req, CELL_ERRORS_req. A decision will be made whether handover is needed based on the collected data. If handover is not needed, then the timer is set again, to scan the AP environment later. If handover is needed, then the HI will ask the MT Target Cell process (MTC) to find a new AP to handover to, using GET_TARGET_AP. When the MTC responds with a new AP MAC address, using TARGET_AP_FOUND, the HI forwards to the higher layers the handover indication with the AP address that the MT would want to handover to with HO_ind. Then, the HI will wait until the MT associates

with the new AP to set its timer again and start checking the new AP radio environment.

3. If the communication with the old AP is suddenly lost, MHI receives AP_LOST, then handover is forward, the HI, irrespective of phase it is, will ask the MTC to search for a new AP. The MTC will respond with the MAC address of the new AP and MHI will again forward the handover indication with the AP address. Then, the HI will send to the higher layers MT_ASSOCIATION_LOST_ind, will wait until the MT associates with the new AP to set its timer again and start checking the new AP radio environment.

6.1.2.2.6. MT Dynamic Channel Allocation entity (MDCA)

The DCA entity in the MT is responsible for detecting interference from other devices, and informing the peer DCA entity of the AP that interference occurs. Then the AP might initiate its DCA mechanism.

6.1.2.3. MT Steady State Control (MSS)

The various functional entities defined in this set correspond to static (as opposed to dynamic) control functions running during regular steady state operations of the MT Mascara Control.

6.1.2.3.1. MT Steady State Gen Control (MSG)

This FE is responsible for initialising the other FEs of the MT Steady state control set, by providing them with operational parameters.

6.1.2.3.2. MT Beacon Agent (MBA)

This FE running in the MT is responsible for receiving the Beacon control MPDU sent by the AP for assisting the MT during the radio environment procedure (for initial association or for hand-over). This FE operates with its peer FE in the AP (AP BEACON AGENT).

Each MT Beacon Agent receives the beacon information transmitted by the Access Point. Then it sends the information about the neighbour APs to the MT Target Cell functional entity, whose duty is to check the environment for other valid A Ps.

6.1.2.3.3. MT Tip Agent (MTI)

This FE running in the MT is responsible for running the TIP protocol, that is signalling to its associated AP when it goes incommunicado and when it comes back. For these reasons it informs accordingly the MAC DATA PUMP.

TIP protocol is needed when the MT has to check its environment about the situation of the neighbour APs in respect to itself; i.e. what is the Radio Link Quality of the MT-neighbour AP wireless link, what is the load of each neighbour AP. The scanning is performed by the MT Target Cell functional entity. The MT TIP Agent though, is responsible for sending a control MPDU to the current AP, requesting to get ‘out of communication ‘ (incommunicado), while this searching is performed. This is done to notify the Master Scheduler that the MT will not be able to transmit or receive any information until further notification. If the AP replies affirmatively to the TIP request, MT TIP Agent informs MT Target Cell entity to start looking for other APs. When this procedure is finished, i.e. when all neighbour APs are scanned, MTC informs MT TIP agent that TIP procedure is completed and the MT should return to the old AP. Then MT TIP Agent issues a control MPDU to inform the old AP that it is about to return. Meanwhile, MTC has ordered the Radio Control Manager entity to get back to the old AP's radio frequency. Since the MT has no uplink slots to transmit the control MPDU, it sends it via contention slots.

6.1.2.3.4. MT Target Cell (MTC)

The MT Target Cell (MTC) process, located in each MT, has several purposes. Its first purpose is to scan for the neighbouring APs and collect data that will be used to select a candidate AP for handover. MTC controls the TIP procedure in order to inform the scheduler that the MT is going to move out of the current area cell. Then, using the RCL process to communicate with the physical layer, it will hop to the neighbouring frequencies in order to collect data for the APs.

Moreover, the MTC process will search the best candidate AP when a handover is needed. When the HI requires a target AP to handover to, the MTC checks the neighbouring APs. The search is done based on the information gathered while the MT was in the area cell of the old AP. The neighbouring AP that was found to be the best candidate, will be examined first. If this is not 'good' at the handover time, the next one will be examined, until we find an appropriate AP to handover to.

6.1.2.3.4.1. MT Target Cell Procedure

The MTC waits to be initialised when the MT powers on with INIT_MTC from Static MASCARA Control. Then it waits till the MT gets associated. A timer is set for TIP; then two cases may occur:

1. Beacon is received from the AP and the list of neighbour APs is being received by the Beacon Agent process, through SEND_NEIGH_AP_INFO. The neighbour AP information is being stored in an array, used for scanning for TIP or handover reasons. When the TIP timer expires, MTC checks if the Beacon has been received. If the beacon is received, then MTC asks TIP to inform the AP that the MT will be out of communication, with INIT_HO_SEEK_INFO. The AP responds and TIP forwards INIT_HO_SEEK_INFO_OK/KO to MTC. If the response is negative, meaning that the AP allows the MT to go incommunicado, then the TIP timer is set again. If the AP response is positive, MTC suspends the HI and scheduler processes. This is performed because the HI and scheduler entities should not perform some of their functions when the MT does not communicate with its old AP. MTC switches to all the neighbour APs and collects information about their load and their RSSI. When all the neighbour APs are scanned, then MTC returns to the old AP frequency and asks TIP to inform the AP that the MT is back, using STOP_HO_SEEK_INFO. It also resumes HI and scheduler processes to continue working properly.

2. HI asks for a candidate AP to handover to with GET_TARGET_AP. This might be either for forward or backward handover. Each of the two cases will be described separately :

    - Forward Handover. The connection with the old AP is lost, thus there is no need to suspend HI and scheduler processes. The array of neighbour APs is used to check one by one the candidate new APs. When an AP is being accessed, the RSSI value is checked. If this value is above a threshold, then this AP is selected as the proposed one. Otherwise, the next AP in the array is checked, until an appropriate AP is found. MTC sends to HI the new AP MAC address with TARGET_AP_FOUND and waits until the MT gets associated with the new AP.
    - Backward Handover. In this situation, the connection with the old AP is not lost, and all the handover signalling exchange will be made via the old AP. Thus, the scheduler process has to be suspended while MTC searches for a new AP. The array of neighbour APs is used to check one by one the candidate new APs. When an AP is being accessed, the RSSI value is checked. If this value is above a threshold, then this AP is selected as the proposed one. Otherwise, the next AP in the array is checked, until an appropriate AP is found. MTC sends to HI the new AP MAC address. Then it resumes the scheduler and returns back to the old AP.

6.1.2.3.5. MT Power Saving (MPS)

This FE runs in a MT, and is responsible for entering or leaving low consumption modes during which the electrical power of the station is saved. The definition of the underlying procedures is currently not yet done. *(This FE is not used in the Demo)*

6.1.2.4. Radio Control (RCL)

The **Radio Control** which is responsible for the “non stationary”, control functions of the MASCARA protocol related to the radio modem. For instance this FE is in charge of receiving RSSI information, or of putting the RF modem in dormant state.

6.1.2.4.1. Emitted Power Control (EPC)

This FE is responsible for driving the RF modem emitted power, according to policies not yet defined.

6.1.2.4.2. Measurements Functions (MEF)

This FE is responsible for recording the RSSI information provided by the RF modem upon reception of any valid MPDU. The MTC or the MHI triggers either such measurement. Their use is to keep the handover algorithm in MHI well informed about the link quality of the air medium while to inform MTC during the TIP process about the link quality of adjacent APs.

Signals that are exchanged between MHI/MTC and MEF are MEASURE_RLQ_req and MEASURE_RLQ_cnf, respectively, and between the MEF and PHY are PHY_MEASURE_RSSI and PHY_MEASURED_RSSI, respectively.

6.1.2.4.3. Radio Control Manager (RCM)

The functionality of the RCM FE is a repetitive pattern that is triggered for the first time when the MT powers on and each time that the MT decides to go TIP or enters the handover phase (backward or forward). It involves the following sequence of steps; a) it drives the RF modem (PHY) to tune in a specific channel and then b) it notifies the MDP (MPX) to start hunting for a FH. Upon reception of the MPX response (either successful or unsuccessful reception of the FH) RCL, in turn, notifies the FE that triggered the RCM's functionality in the first place.

The signals exchanged in each case are as follows:

1) Power on

GMC sends INIT_RCL signal to RCL, which, consequently, sends PHY_SELECT_CHANNEL signal to the RF modem to switch to a specific channel. After a period of duration 1 ms the RCL safely assumes that the switching has occurred. It then notifies MPX via signal FH_HUNT_req to start hunting for the FH. The result of the hunt procedure is carried in the FH_HUNT_cnf and, if successful, together with the MAC address of the AP that corresponds to this channel. Finally, if hunting has been successful, RCL responds to the GMC's initial request via signal INIT_RCL_OK carrier of the channel number and the AP's MAC address . In case of unsuccessful hunting RCL requests from PHY to select another channel and this process goes on until an AP is found or all channels are exhausted. In the latter case RCL reports this result to GMC via signal INIT_RCL_KO.

2) TIP and Handover

Since both cases are eventually handled by the Target Cell (MTC), RCL receives signal ACQUIRE_NEW_AP carrier of the channel number to switch to. RCL again enters step a) and b) and if step b) was successful it responds to MTC with signal ACQUIRE_NEW_AP_OK, otherwise with signal ACQUIRE_NEW_AP_KO

6.2. CSR (Control Segmentation & Reassembly)

6.2.1. General Aspects

The functional entities of block CSR are the processes Control Segmentation (CSE) and Control Reassembly (CRE). In the MT there exists one fixed pair of those entities whereas in the AP they are created dynamically for each new attached Mobile

Terminal. Creation and Destruction is the task of process Generic Segmentation and Reassembly (GSR).

6.2.2. Signalling via control channel

Control information has to be exchanged between MASCARA Control (MCL) entities of AP and MT through special peer-to-peer interfaces. The underlying layers have to be transparent to these interfaces. When a new MT has associated to an Access Point, this AP has to create a special control connection (MVC_Id = CONTROL–_MVC_ID = 0) for the peer-to-peer signalling.

The main purpose of the CSR block is to segment the control signals and their parameters into ATM cells and give those cells to the WDLC for transmitting. Like the cells of any other connection these cells are sent over the wireless channel. After reception of those ATM cells by the remote WDLC the corresponding CSR entity has to reassemble the ATM cells and to retrieve the control information. Then the signals (and parameters) will be delivered to appropriate MCL entities.

6.2.3. Signalling without existing control channel

Note, that there exist some special signals which cannot be transmitted via such a dedicated control connection.

6.2.3.1. Association

The first is the MPDU_MT_ASSOCIATION_req, sent from a not yet associated MT to the desired AP during the contention period. Before successful association has finished, no control channel exists (e.g. there are no instances of CSR/WDLC in the AP). Even after creation of these instances the control channel cannot be used until the MT knows its MT_MAC_Addr.

The new associated MT has to learn its MT_MAC_Addr (chosen by the destination AP) from the signal (MPDU_AP_MT_MAC_ADDR_ALLOCATE), which has to be sent broadcast to each MT (or at least to each MT in association phase). Each of them has to reassemble it and to take care if its ATM-Address (unique, 20 bytes) is contained as a further parameter. When the Mobile Terminal has received its MT_MAC_Addr, the control channel is established and can be used for further control signalling.

6.2.3.2. Beacon

Another signal that has to be sent broadcast, now from AP to any MT, is the MPDU_BEACON. The beacon is sent periodically for assisting the MTs during the radio environment procedure (for initial association or for handover). Beacons include notes about contents, frequency of appearance of beacon announcements etc. of the sending AP and the APs in the neighbourhood.

6.2.4. Broadcast CSR/WDLC Entities

To cope with the need of broadcast signalling some extra effort in both, AP and MT has to be done, concerning the instanciation of CSR/WDLC entities for broadcast traffic. This is reflected in the implementation of AP's process Generic Segmentation & Reassembly (GSR), which is responsible for creating of CSR instances. Since in the Mobile Terminal those processes are static, no GSR exists; instead of GSR the CRE process is responsible for the instanciation during the initialisation phase.

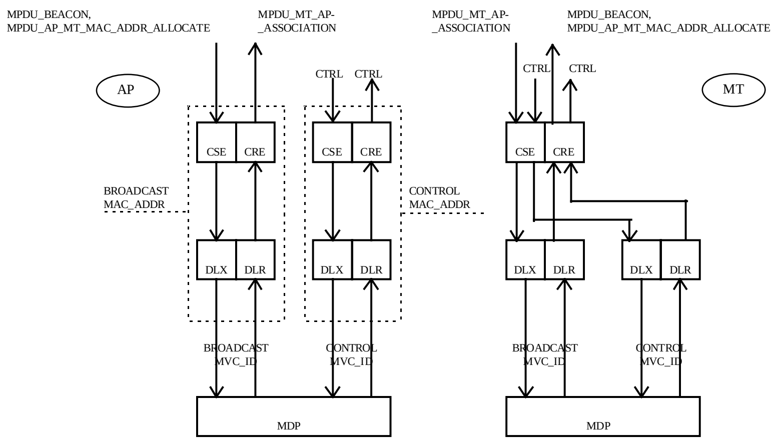


Figure 14 CSR and DLC entities.

6.2.4.1. Access Point

The AP creates one instance of CSR/WDLC for each MT associated to it.

Beside these 'normal' MTs maintained by the AP, there is a special virtual Mobile Terminal introduced, the 'broadcast MT' with MT_MAC_Addr = BROAD-CAST_MT_MAC_ADDR. All broadcast signalling (send MPDU_BEACON & MPDU_MT_MAC_ADDR_ALLOCATED and receive signal MPDU_MT_AP_ALLO-CATE_req) is done through the CSR/WDLC of that virtual connection. The control channel for this broadcast MT is created at system start-up, before the first MT can associate to this AP. The broadcast traffic is sent by or delivered to this broadcast CSR/WDLC using MVC_Id = BROADCAST_MVC_ID.

6.2.4.2. Mobile Terminal

In the MT there will be two instances of WDLC connected to one (the) CSR. The first is the normal control WDLC (MVC_Id = CONTROL_MVC_ID = 0) and the second one is the broadcast WDLC (MVC_Id = BROADCAST_MVC_ID). As there is no GSR in the MT, the instanciation of both WDLCs is done by the CRE. The sending process CSE has to maintain the Pids of both DLX entities and deliver the constructed ATM cells either to the 'normal' or to the 'broadcast' DLX. That is possible, because the CSE knows which signals have to be sent broadcast or not.

6.2.5. Functionality - Dynamic Aspects

6.2.5.1. Generic Segmentation and Reassembly (GSR)

As mentioned before, this process exists only in the Access Point and is responsible for the creation and destruction of the CSR instances CRE and CSE.

6.2.5.1.1. Creation

Creation of new CSR instances is triggered by the signal INSTANCIATE-_CSR(MT_MAC_Addr, MVC_Id, Pid). The signal is sent either by the Association Agent (AAA) in order to establish the control connection (MVC_Id = CONTROL_MVC_ID = 0) for a new MT after successful association request or by the Generic Mascara Control (GMC) in order to create the broadcast connection (MT_MAC_Addr = BROADCAST_MAC_ADDR, MVC_Id = BROADCAST_MVC_ID). GSR then creates a new process instance of CRE and CSE. The Pids of these child processes are stored in a table with entries for each Mobile Terminal. Now the creation of corresponding WDLC instances for the new connection is requested via INSTANCIATE_DLC(MT_MAC_Addr, MVC_Id, QoS, WAND_ATM_Parameters, ProcessType = CRE, CRE_Pid, ConnectionIdentifier) sent to Generic WDLC (GDL). The parameters QoS and WAND_ATM_Parameters describe the traffic parameters of the connection. For Control Connections proper values have to be chosen by the GSR (for ATM traffic they are defined in higher layers). In case of Control Connections the WDLC strategy (indicated by QoS parameter) should be ARQ, in case of Broadcast Connections the WDLC strategy has to be non-ARQ, as there is no broadcast feedback.

Now some Process Identifiers (Pids) have to be exchanged using the signal PID_REGISTERED(MT_MAC_Addr, MVC_Id, ProcessType, Pid).

The instanciation request is answered from GDL by sending PID_RE-GISTERED(MT_MAC_Addr, MVC_Id, ProcessType = DLX, DLX_Pid) after creation of WDLC instances DLR and DLX. GSR retrieves the DLX_Pid from the signal and forwards it to the CSE.

If INSTANCIATE_CSR was sent from AAA (normal case) the AAA_Pid is given to CRE and the CSE_Pid is given to the AAA instance. In the special case of creating the broadcast connection, the GMC_Pid is sent to CRE and the CSE_Pid is given to GMC.

6.2.5.1.2. Destruction

If the Mobile Terminal deassociates from the AP, the Control Connection has to be destroyed. The Association Agent (AAA) indicates this by sending DESTROY-(MT_MAC_Addr, MVC_Id). The GDL requests the destruction of the corresponding WDLC instances by forwarding the DESTROY signal to GDL. Then it looks up in its table to retrieve the CRE_Pid and the CSE_Pid for the connection described by MT_MAC_Addr and MVC_Id. It sends the KILL signal to both processes (results in termination of their lifetime) and deletes the Pid entries from the table.

6.2.5.2. Instanciation parts of CRE/CSE

Since there is no GSR in the MT the process Control Reassembly (CRE) is responsible for the instanciation task in its initial phase. In this section only these aspects of CRE and CSE are mentioned, the general functionality is described separately as it is valid for both, the AP and MT.

As there exists only one static CSR in the MT, there is no need for signal INSTANCIATE_CSR. CRE starts with requesting the instanciation of the Broadcast WDLC by sending INSTANCIATE_DLC with MVC_Id = BROADCAST_MVC_ID. After confirmation it forwards the Broadcast_DLX_Pid to the CSE process. Then it requests the instanciation of the second WDLC for normal control purposes by sending INSTANCIATE_DLC with MVC_Id = CONTROL_MVC_ID = 0. The returned Control_MVC_Id is also forwarded to the CSE. As mentioned before the traffic parameters have to be chosen carefully (non-ARQ for broadcast, ARQ for control connections).

In the AP the only dynamic related task for CSE is to learn the destination DLX_Pid and for CRE it is to learn the AAA_Pid (resp. GMC_Pid in case of Broadcast).

6.2.6. Functionality - Static Aspects

6.2.6.1. Overview

In this section the main functionality, the segmentation and reassembly of control signals, is discussed. The scenario is as the following. An entity of Mascara Control (MCL) wants to send a signal to its peer entity, located in the remote device. Instead of addressing the signal to the peer entity, it sends the signal to the proper CSE instance. In AP this is the CSE of the dedicated control connection or the broadcast CSE, in MT exists only one CSE. The CSE builds one or more ATM like cells from the signal and its parameters and delivers the ATM cell(s) to the proper DLX entity. In MT this is the Control_DLX or the Broadcast_DLX, in AP there exists only one DLX per CSE (Remember that in the AP a special virtual MT was introduced for broadcast purposes). It’s the task of lower layers to transport the ATM cells to the DLR located in the remote device (not discussed here).

The receiver’s DLR is in charge of retrieving the ATM cells containing the control information and passing them to the CRE. CRE will reassemble the control message and send the signal and its parameters to the Mascara Control entity. The CSR block provides the peer-to-peer signalling interface as described above. Signals that will be segmented in the MT have to be reassembled in the AP and vice versa.

6.2.6.1.1. Control Signals to be segmented/reassembled

The control signals from AP to MT currently supported are:

- MPDU_AP_MT_ASSOCIATION_ACK
- MPDU_AP_MT_MAC_ADDR_ALLOCATE
- MPDU_AP_MT_DEASSOCIATION

- MPDU_AP_MT_CONN_ACTIVE
- MPDU_AP_MT_CONN_REFUSED
- MPDU_AP_MT_CONN_PENDING
- MPDU_TIP_ACK
- MPDU_BEACON

The control signals from MT to AP currently supported are:

- MPDU_MT_AP_MAC_ADDR_RECEIVED
- MPDU_MT_AP_DEASSOCIATION
- MPDU_MT_AP_CONN_ACTIVATE
- MPDU_TIP
- MPDU_MT_AP_ASSOCIATION

6.2.6.1.2. Procedures implemented

The signals mentioned above carry several parameters of different types. A basic functionality needed is to segment a variable of a certain datatype into some bytes and on the other hand to reassemble the bytes into a variable of that type. So there are some pairs of procedures implemented which are in charge to segment or reassemble standard datatypes and datatypes build from standard types.

The following procedures are currently defined:

Segment_Int(IntVal Integer, BYTES_OF_INT Integer)

Segments the variable IntVal of type Integer. If BYTES_OF_INT is positive, IntVal is represented as an unsigned Integer with BYTES_OF_INT bytes, if BYTES_OF_INT is negative, IntVal is represented as a signed Integer with -BYTES_OF_INT bytes. The flexible length of Integers is introduced to avoid wasting of payload bytes if the range of IntVal is limited.

Segment_String (String Charstring)

Segments a charstring String (length up to 255 chars) into bytes.

Segment_ATM_Addr(ATM_Addr ATM_AddrType)

Segments variable ATM_Addr of type ATM_AddrType into 20 bytes.

Segment_CON_Id(ConnectionIdentifier ConnectionIdentifierType)

Segments variable ConnectionIdentifier of ConnectionIdentifierType into bytes.

Segment_Neigh_AP_List(List NeighAP_ListType)

Segments variable List of NeighAP_ListType into bytes. The amount of bytes used is variable and depends on the number of list entries.

Each of the procedures mentioned above has a counterpart to reassemble the variable from the payload bytes. The reassembling procedures are *Reassemble_Int, Reassemble_String, Reassemble_ATM_Addr, Reassemble_CON_Id, Reassemble-_Neigh_AP_List.*

6.2.6.2. Process Control Segmenting (CSE)

The structure of CSE in AP and MT is the same, but obviously the signals are different. After initialisation the process remains in state IDLE, until one of the signals to segment is received. (Unexpected signals are reported to global procedure Error_Log).

For each of the control signals exists an unique Identifier (Message_Id) and a dedicated segmenting procedure which is called after reception of the signal.

The goal is to build from the control signal one or more ATM-Payloads (Array of 48 bytes), represented as array of payloads. There exists a pointer to the current byte position in the payload under construction (L_PayloadBytePos). The first thing each segmenting procedure performs is to write the Message_Id in byte #0 of the first payload used. The byte #1 should contain the number of payloads needed to contain the message, but as there are signals with variable length, this will be determined after writing the last byte of the signal. The L_PayloadBytePos is set to byte #1 to reserve one byte for this information. Now for each parameter of the signal the appropriate segmenting procedure (e.g. Segment_Int) is called. The segmentation procedures are responsible to increment the pointer L_PayloadBytePos by one, **before** writing a byte into the payload. This is done by calling the procedure IncPayloadBytePos_S. This procedure assures that the first byte of the next payload (from the array of payloads) is used, when the last byte of the current payload was written before.

When the last parameter of the control signal is segmented, the number of payloads used is known and can be written into byte #1 of the first payload.

The following steps are done for each payload used:

- request memory for one ATM cell from the ATM-layer by ALLOCATEAAL0_req
- get pointer to the allocated memory (L_ATM_CellPtr), if L_ATM_CellPtr = NULL (no memory available) report to Error_Log and try again
- manipulate ATM cell header using operators (ADTs) PUT_HEADER_FIELD
- put payload into ATM cell (ADT operator: PUT_PAYLOAD)
- deliver ATM cell containing payload to DLX (signal XMIT_CTRL_CELL)

After sending of the last ATM cell, the process returns to state IDLE, waiting for the next signal to segment.

6.2.6.3. Process Control Reassembly (CRE)

The structure of the process CRE is the same in AP and MT, but the signals are different. After initialisation the process remains in the state IDLE, awaiting the signal CTRL_CELL_RCVD, where a pointer to an ATM Cell (located in the receive buffer) can be retrieved from (L_ATM_CellPtr). The ATM cell pointed to is copied into a local ATM cell (L_ATM_Cell) and the payload is extracted from it (L_Payload) and stored in an array of payloads. The first byte contains the signal identifier (L_Message_Id) and the second byte the number of payloads the message exists of. Now in the same manner as for the first payload, all ATM_Cells for that message will be received (CTRL_CELL_RCVD) and the payloads will be appended to the payload array.

For each signal that can be contained in the payloads exists an appropriate reassembling procedure that will be used according to the L_Message_Id. This procedure will call the proper reassembling procedures for each parameter (e.g. Reassemble_Int). The reassembling procedures are responsible for incrementing the L_PayloadBytePos pointer thanks to procedure IncPayloadBytePos_R.

After retrieving all parameters, the control signal is delivered to the proper MCL process (for each possible signal the receiver is known). After sending the control signal the process returns to state IDLE, waiting for the next ATM cell to arrive.

1.

7. MASCARA Information Flow

7.1. Introduction

This section describes how the information of ATM cells or MASCARA signalling is handled by MASCARA and is passed / received to /from the physical layer.

7.2. WDLC Architecture and Function

7.2.1. The Wireless Data Link Control (WDLC) Sublayer

Within MASCARA the Wireless Data Link Control (WDLC) is responsible for error control over the radio link. Different WDLC techniques are used depending on the Quality of Service (QoS) parameter and thereby on the traffic class of each ATM connection.

In general, the WDLC entity may employ **Forward Error Control (FEC)**, **Automatic Repeat reQuest (ARQ)** or a hybrid form of those (**hybrid ARQ**). Moreover the WDLC may control the **PHY rates (fallback rates)** according to the measured quality of the channel in a way that the resulting quality of the discrete channel (includes PHY level) allows an efficient application of the error control techniques in the WDLC (high-rate FEC + ARQ). At the physical layer, there is an inner code as an integral part of the COFDM (coded OFDM) system: it is designed to control the peak-to-average power ratio and thus provides only ‘limited’ error-correction capabilities. The different PHY rates can be obtained (preferably) by changing the size of the modulation signal set or by modifying the inner code: it is expected that reduced PHY rates are the most efficient mean to recover from (very) low channel qualities.

WDLC error control techniques are applied on cell level and on a per MT/per MVC-basis. The MASCARA control (frame) structure is assumed to be visible to the WDLC FE.

The code rate of the error-correction code (**FEC, hybrid FEC**) is selected according to the discrete channel quality (including the PHY rates), the required QoS and the number of retransmissions (hybrid ARQ). With an appropriately selected PHY rate, the required code rate will be relatively high which results in a lower decoding complexity and a smaller processing delay compared to low-rate codes. A similar consequence is implied by the application of non-binary (shortened, ’punctured’) codes for error-correction/ -detection purposes, especially the maximum-distance-separable Reed-Solomon code, which results in a processing on symbol-level. As additional features, the level of the code redundancy can be chosen cell-selectively dependent on the time-to-live or the number of retransmissions and some ‘minimal’ interleaving consistent with the MPDU structure (real-time constraint) can be provided.

ARQ strategies are well suited for bursty channels of ‘good’ quality to guarantee a high data integrity at the expense of some delay variation. For delay-insensitive but highly loss-sensitive services (such as TCP/IP over ATM) ARQ is a good choice for ‘good’ link qualities. Delay-sensitive services (such as voice) may suffer from too much delay when ARQ is applied.

For loss sensitive services, an Automatic Repeat Request (ARQ) protocol is used to guarantee error-free transmission, performing retransmission of erroneously delivered cells. On the other hand ARQ is prohibited in WAND in case of delay sensitive traffic. For WAND there will be mainly two different traffic classes within MASCARA, i.e. real-time traffic (CBR and VBR) and non-real-time traffic (UBR). Retransmission is applied only for the non-real-time traffic. For that purpose WDLC uses a Go-Back-N (GBN) protocol with a sliding window size of M=15.

The sliding window size M is a critical parameter, as it should be small with respect to transmission overhead and required buffer sizes but also large enough to avoid transmission pauses caused by delayed feedback.

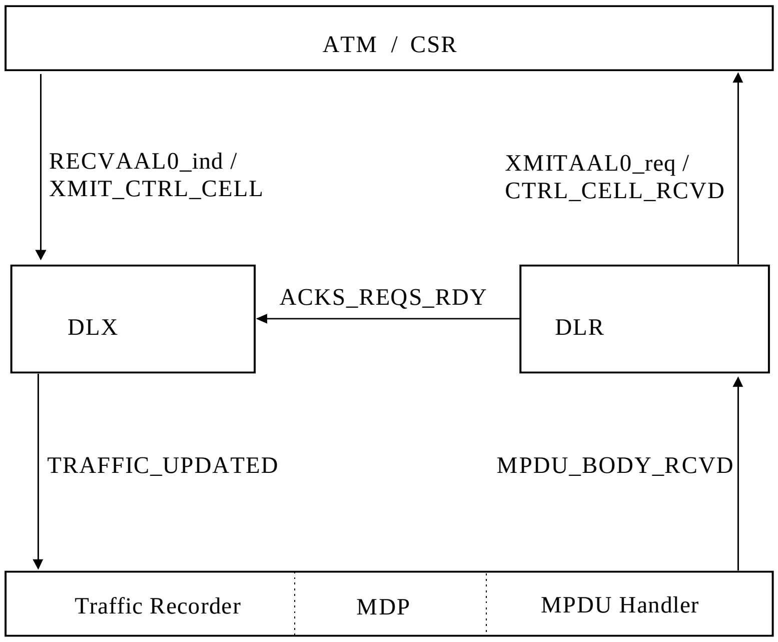


Figure 15 Block Diagram of WDLC and the interfacing blocks

In the implementation of the protocol stack WDLC is located between the ATM layer and the MAC Data Pump (MDP). WDLC itself is split into two main blocks, DLX and DLR, for the transmit and the receive direction, respectively. Any single ATM or control connection is treated by an own pair of DLX/DLR instances within SDT. Such instances are created dynamically whenever a connection is established, and deleted again after the connection is closed. The instanciation process is triggered by appropriate signal generated by the MASCARA control entity.

The DLX can receive traffic, either from the ATM layer or from the Control Segmentation and Reassembly (CSR) block. As CSR segments signalling in standard ATM cells a DLX entity does not have to distinguish between cells from ATM or CSR. For every ATM cell received the DLX replaces the standard ATM cell header by a WDLC specific header of the same length. This WDLC header includes for example the Sequence Number (SN) of the current cell and a Request Number (RN) for the reverse connection. It also includes a connection identifier, the MAC Virtual Circuit Identifier (MVC_ID), which allows recovering the original ATM cell header at the reception side.

In order to transport the RN for the WDLC feedback this information generally is piggybacked in the WDLC headers of cell traffic in the reverse direction.

In the most basic version of WDLC cells for non-real-time connections are retransmitted as often as required until they get through to the receiver. It is assumed that higher layers (such as TCP) have to control the traffic flow in case of reduced throughput. Nevertheless, to avoid endless transmissions optionally a cell-discarding feature can be included that limits the maximal number of transmission attempts.

7.2.2. General Functions of WDLC

7.2.2.1. WDLC Functional Description for Real Time Traffic

For RT-Traffic the use of ARQ is prohibited, as the applications based on it are in general delay sensitive. In this case the WDLC-Transmitter (DLX) and the WDLC-Receiver (DLR) have a reduced functionality.

7.2.2.1.1. Basic Functionality of the WDLC Transmitter (DLX)

The DLX can receive traffic, or equivalently cell pointers, either from the ATM layer or from Control Segmenting and Reassembly (CSR). As CSR segments signalling in standard ATM cells a DLX entity does not have to distinguish between cell pointers from ATM or CSR. DLX maintains a traffic table (TT) to store the cell pointers. This TT is implemented as a ring-buffer for continuous operation. Cell pointers received from ATM or CSR are inserted at the end of the filled part of the ring buffer. There are two different traffic classes, i.e. RT-traffic (CBR and VBR) and NRT-traffic (UBR). Retransmission is applied only for the NRT-traffic.

For every cell pointer received by DLX from ATM or CSR the corresponding cell header is replaced by the modified header, whose fields have been presented in section 5.

In the simplest case the signal TRAFFIC_UPDATED (giving the newest cell pointer(s) to the scheduler) is issued every time a cell pointer has been received from ATM or CSR. It may turn out that this results in too many signal exchanges between WDLC and MDP (MAC Data Pump). In order to avoid this the signal TRAFFIC_UPDATED could be issued when a certain number of cell pointers (e.g., K) have been received or if a time-out for collecting has occurred. This reduces the number of signal exchanges by a factor up to K. The time-out should be in the order of a mean frame duration (e.g. 10-20 cell slot duration), so that there is at least one TRAFFIC_UPDATED within this time.

7.2.2.1.2. Basic Functionality of WDLC Receiver (DLR)

When DLR receives cell pointers from MDP through the signal MPDU_BODY_RCVD it replaces the WDLC cell headers by ATM cell headers. It then gives the cell pointers immediately up to ATM or CSR, without taking care of any CRC field.

7.2.2.1.3. Memory Release for RT-Traffic

As WDLC does not perform any retransmission for RT-traffic it also gets no acknowledges for received cells from the remote entity. DLX immediately forwards all cell pointers from ATM or CSR to the MDP and only MDP knows when the corresponding cells are finally sent. Therefore MDP is in charge of releasing the buffer of those cells.

7.2.2.2. WDLC Functional Description for Non Real Time Traffic

For NRT-traffic the WDLC has the task to perform retransmissions of erroneously transmitted cells. This Automatic Repeat Request (ARQ) protocol results in some additional functionality within DLX and DLR.

7.2.2.2.1. Additional Functionality of DLX

DLX has a transmission window of length N (Go-back N), which is a parameter that remains to be determined by simulations (however N<=15 since the sequence number ranges only from 0 to 15).

Sequence numbers (SNs) are assigned to all cells in the Traffic Table (TT). The SN is taken modulo 16 (4 bits) and must be written into the cell header. After start-up, the first SN is zero.

There is a current value of a request number for the reverse connection (TXRN, as explained below). This value is updated through ACKS_REQS_RDY (issued by DLR). After start-up this value is zero.

Every cell pointer received from ATM or CSR is inserted in the TT. The ATM cell header is replaced by a WDLC header (including the dedicated SN and the current value of TXRN) and TRAFFIC_UPDATED is issued giving a list of the new cell pointers in the TT to the scheduler. In order to reduce the number of signal exchanges a signal TRAFFIC_UPDATED is issued after the reception of a certain number K (actually K=5) of cell pointers from ATM/CSR. Also a time-out for reception is set, i.e. if there are no more cells received the DLX will issue the TRAFFIC_UPDATED despite there are not K cells in the TT.

A cell pointer that has been given to the scheduler must still be kept in the TT until it is acknowledged or deleted (discarding feature).

After giving the maximum of N cell pointers to the scheduler without receiving an acknowledgement (i.e., the GBN window is full) DLX must store new cell pointers from ATM or CSR in the TT and it has to stop issuing any further TRAFFIC_UPDATED.

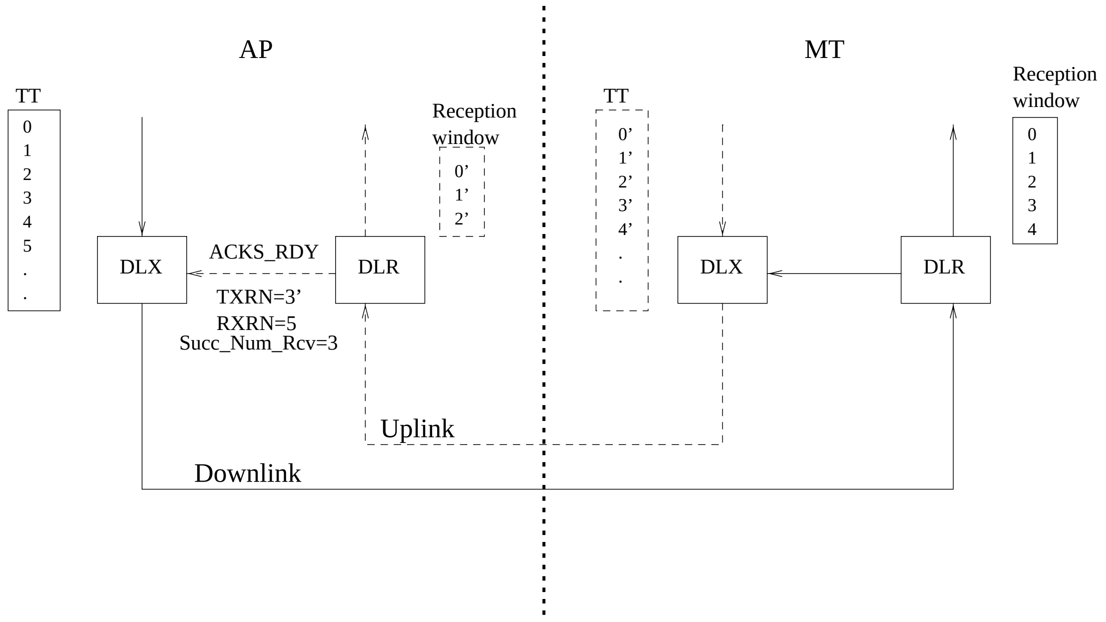


Figure 16 Example for ARQ traffic flow.

An example of the traffic flow is given in Figure 16. The signal ACKS_REQS_RDY carries the parameters RXRN, TXRN, SuccNumRcv and NumOfRequests (in the AP only). The request number RXRN represents the received feedback for the transmitted cells, e.g., for downlink connections the AP sends ATM cells 0-4 to an MT and the MT returns RXRN=5 to the AP in order to request the next cell. On the other hand, TXRN denotes feedback for received cells, e.g. for the uplink connection the AP receives ATM cells 0’-2’ from the MT and returns TXRN=3’ to the MT. The parameter SuccNumRcv stands for the number of successfully received uplink cells in the AP. It will be 3 in our example. It is given to MDP (master scheduler) as a parameter of the signal TRAFFIC_UPDATED and is required by the master scheduler for the token handling in the leaky bucket algorithm.

The parameter NumOfRequests carries the number of requested slots for the corresponding up-link connection. The DLR extracts this parameter from the incoming up-link cells and delivers it via ACKS_REQS_RDY to the DLX. Since the number has to pass to TRE, DLX includes it to the TRAFFIC_UPDATED signal.

When DLX receives ACKS_REQS_RDY from DLR it must go through the TT from the start and delete the cell pointers of all acknowledged cells (cells with SN 0-4 according to RXRN). In the case of no further traffic, i.e. there are no more cells in the TT to transmit, the last confirmed cell remains in the TT (see also 0). Then it must update the value of TXRN and write it in the cell headers of all cells that remain in the TT. After this the signal TRAFFIC_UPDATED is issued with a list of the next cells to be transmitted.

Avoiding Superfluous Re-Transmissions due to Delayed Feedback

With TRAFFIC_UPDATED a number of cell pointers, say m, is given to the MDP.

The MDP may send all m cells within the next frame but it can also happen, that two or even more frames are required to transmit all m cells. Then at least another frame is required before the WDLC will receive some feedback for the transmitted cells, the request number RXRN.

An example of this is given in Figure 17. Assume that DLX always feeds the scheduler with more cells than can be sent within the next frame. Although the WDLC will get an indication RXRN=4 after frame I+1 it should not start with an immediate retransmission of this cell. This is because it does not know whether this request was caused by the limited capacity for cells in the last frame or by a real transmission error. Only when receiving multiple times the same RXRN the WDLC can assume with a increased reliability, that a transmission error has occurred.

→ A point concerning the scheduler:

To obtain homogenous transmission flow, the mean scheduled cells per frame and MVC_ID should not be larger than half the window size (i.e., not more than 7), as the overall transmission delay is at least 2 frames.

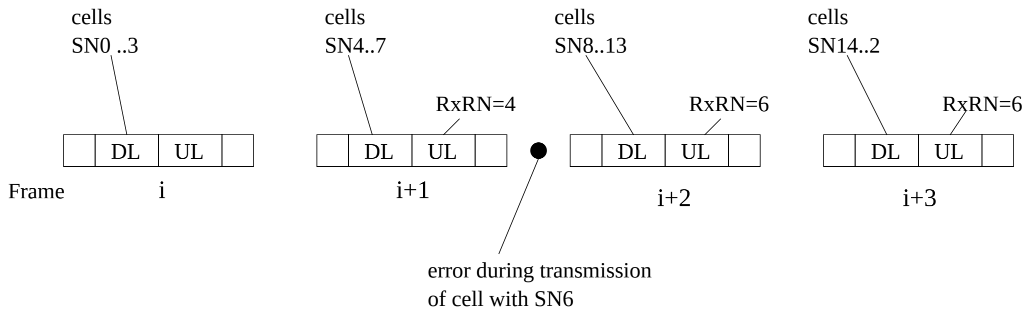


Figure 17 Example for delayed feedback.

The variable delay of feedback must be considered in the ARQ scheme as it definitely has large impact on the overall performance with respect to throughput.

Retransmission is started if RXRN indicates twice the same request for a certain cell number within two succeeding MPDU transmissions.

The full range of sequence numbers (0-15) is used for the transmission window. It is shifted only upon the reception of feedback. But the next cell pointer given to MDP is not defined by the feedback but rather by the latest cell pointer given to MDP.

If twice the same feedback has been received, it is assumed that a transmission error has likely been occurred or that there are no more cells under way. Therefore the transmission must be continued with the cell indicated by the feedback. In this case, cells that are eventually still buffered in MDP can be discarded.

The MDP is responsible in comparing new cell pointers received by TRAFFIC_UPDATE with such that are still in its buffer list.

7.2.2.2.2. Additional Functionality of DLR

There is a current value of the expected sequence number for the receive direction (TXRN). There is also a current value of the request number for the transmit direction (RXRN). Modifications of these values are indicated to the DLX entity of the same MVC_Id through ACKS_REQS_RDY. After start-up they are set to zero.

DLR receives cell pointers from MDP through MPDU_BODY_RCVD. It checks the 2 byte CRC error syndrome at the end of the cell. If this field is unequal to zero the cell is erroneous and must be discarded (along with the pointer). Else the SN and the request number (RXRN) are read from the cell header and the current value of RXRN is updated. If the SN is unequal to the current value of TXRN the cell and its pointer are discarded. Else the cell can be considered o.k. and TXRN as well as SuccNumRcv are incremented by one (modulo the SN range). When the reception of an MPDU is finished DLR issues ACKS_REQS_RDY to DLX carrying TXRN, RXRN, SuccNumRcv and NumOfRequests.

7.2.2.2.3. Memory Release for NRT-Traffic

For acknowledged cell transmission only DLX knows when a cell is successfully transmitted. Therefore this instance is also responsible for issuing the RELEASEAAL0_req signal to mascenv.c to initiate the memory release for the corresponding cells.

7.2.2.3. Transport of WDLC Feedback without Traffic in Reverse Direction

The case may occur that DLX receives a new value of TXRN through ACKS_REQS_RDY but there is no traffic to piggyback TXRN on. Therefore the last successfully transmitted cell remains in the TT until further traffic arrives. If a new TXRN is received by the DLX it is piggybacked on this cell. The opposite DLR receives this cell, reads the WDLC-header and updates TXRN. Thereafter the DLR discards the already successfully received cell, since the expected SN does not match with the SN of the received cell. Actually DLR checks if the last TXRN is equal to the SN of the received cell. Then the DLR delivers TXRN to the DLX of the same WDLC by sending ACKS_REQS_RDY.

A dummy cell is created at the DLX-start-up with SN=15. This dummy cell is used for piggybacking TXRN on if no traffic was sent by this DLX since initialisation. Since DLR expects SN=0 after start-up, this dummy cell will be discarded, after extracting the TXRN from the WDLC-header.

7.2.2.4. Differences of WDLC in AP and MT

Concerning the implementation of the pure non-ARQ or ARQ algorithms within DLX and DLR, the realisation in the MT is simply a one to one copy of the AP processes. Nevertheless there are some differences in how to set up DLX and DLR blocks for a new MVC_ID within the Generic Data Link module (GDL). The instanciation process in the AP has an extension in address handling because it has to manage more than one

MT. Furthermore MT and AP have to support Mascara control in a different way to obtain some statistics on the traffic.

Generic WDLC (GDL)

In the MT there exists only one single CSR instance[1] which is responsible to handle both, broadcast and dedicated control signalling. During the first initialisation of the system this CSR instance is linked to two WDLC instances, one for broadcast and one for dedicated signalling. This link is fixed for the lifetime of the MT. So during operation of the MT a new WDLC instance can only be created as a result of opening a new ATM connection via the MT MVC Agent.

In the AP there are dynamic attachment and detachment of MT’s during operation. So new WDLC instances are not only requested for ATM connections by the AP MVC Agent, but also by CSR instances of newly attached MT’s.

7.3. MDP (MAC Data Pump)

This block is responsible for receiving ATM cells form the upper layer or from MASCARA entities, allocating time slots to these cells, packing them into larger packets called MASCARA Protocol Data Units (MPDUs), preparing the physical layer for their transmission and providing the MPDUs to the physical for transmission. It is also responsible for receiving MPDUs from the physical layer, checking if they are correct, extracting the ATM cells and forwarding the cells to WDLC.

7.3.1. MPDU/ATM Cell Transmission

The MPDU Handler Transmit FE (or MPX for short) is mainly responsible to build a sequence of TEDs (Traffic Element Descriptor) based on the slot map of the current MAC time frame. A TED is later read by the Slot Sequencer (SSE) to follow the time-dependent directives (radio reception/transmission) coming from the scheduler.

7.3.1.1. The AP MPDU Handler Transmit (AMMPX) Functionality

AMMPX is responsible for the MPDU transmission in the AP. It is initialised by the Generic MAC Data Pump thanks to the signal INIT_MPX, which carries relevant operational parameters.

The input signal that triggers the operations is the FH_PAYLOAD signal coming from the AP Scheduler. The MPX finds in the input signal the full slot map from which it can learn, among other things, which MPDU must be transmitted and when. Then the MPX builds the FH MPDU and the corresponding downlink MPDUs, as well as the sequence of TEDs (including TEDs for reception of uplink MPDUs) which will be consumed, one after the other, by the slot sequencer in order to ‘run’ the slot map. The signals from/to AMMPX are shown in the following Figure 18:

[1] An instance is regarded to be a pair of reception and transmission process

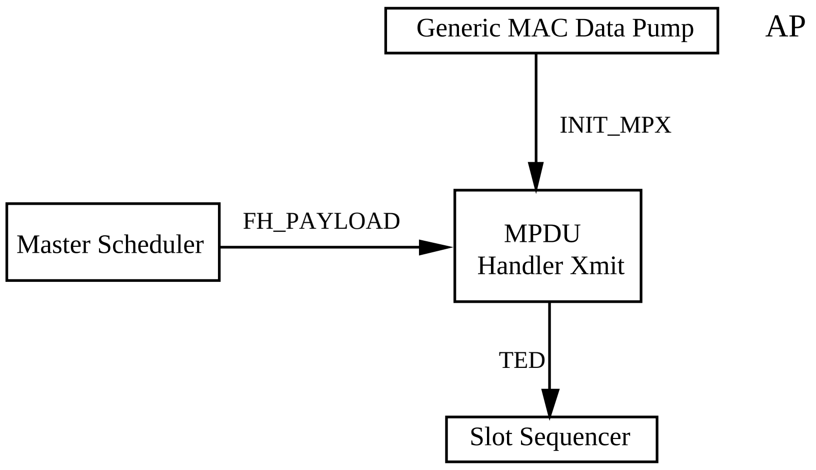


Figure 18 AMMPX interfaces.

7.3.1.2. The MT MPDU Handler Transmit (MMMPX) Functionality

MMMPX is responsible for the MPDU transmission in the MT. It is initialised by the Generic MAC Data Pump thanks to the signal INIT_MPX which carries relevant operational parameters. Immediately after the initialisation, MPX allocates memory space for the mbuffer. The mbuffer is the buffer containing the whole time frame; thus all sent and received MPDUs. It then sends the mbuffer's pointer to the MT scheduler through the signal MBUF.

Upon the reception of FH_HUNT_req signal, the MPX enters the hunt mode where it waits for the reception of a FH. If the FH is received correctly by the MPR, it will receive the FH_CAUGHT signal from the MPR. The MPX in turn sends a FH_HUNT_cnf signal to the Radio Control Manager to inform it of a successful reception of the FH. If no FH has been received for a certain amount of time, a FH_HUNT_cnf is sent to the Radio Control Manager to inform it of a hunt failure.

The input signal which triggers the operations is the MT_FH_READY signal coming from the MT scheduler. The MPX finds in this input signal the current slot map from which it can learn, among other things, which MPDU must be transmitted and when. It can also find the reservation requests for the Master Scheduler and the control MPDUs (MPDUs to be sent in contention mode) in this input signal (if there are too few MPDU headers to host all the reservation requests of connections that do not have allocations in the current time frame, special control MPDUs must be used). Then the MPX builds the corresponding control MPDUs and uplink MPDUs. It then sends a sequence of TEDs (including TEDs for reception of downlink MPDUs of next time frame) which will be consumed, one after the other, by the slot sequencer in order to 'run' the slot map. The signals from/to MMMPX are shown in the following Figure 19:

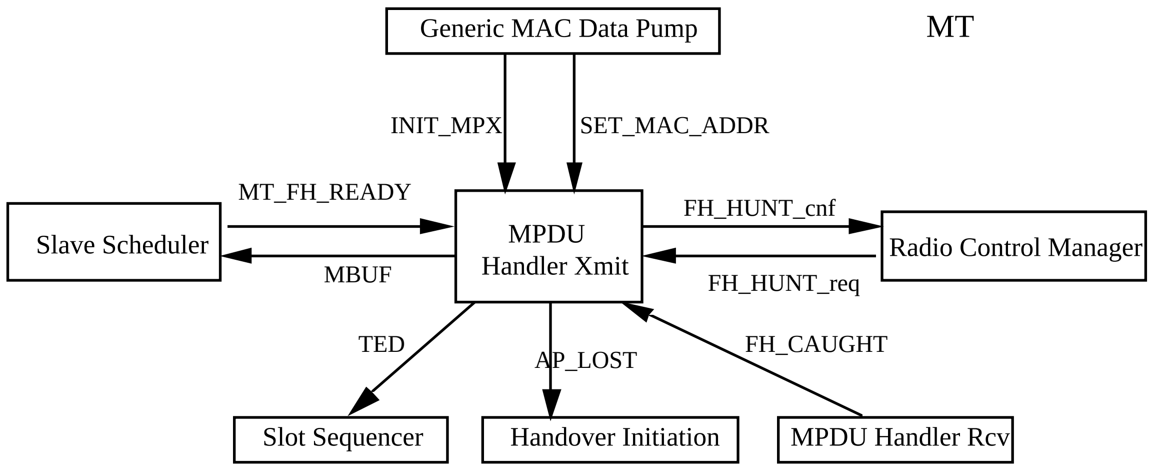


Figure 19 MMMPX interfaces.

7.3.2. MPDU/ATM Cell Reception

The MPDU Handler Receive FE (or MPR for short) is mainly responsible to pass to the relevant MASCARA entities the MPDU received by the Slot Sequencer entity (Frame Processor Block). It also checks the integrity and of the MPDU and if this latter is found to be erroneous, then it is simply discarded.

7.3.2.1. The AP MPDU Handler Receive (AMMPR) Functionality

AMMPR is responsible for the MPDU reception at the AP side. It is initialised by the Generic MAC Data Pump thanks to the signal INIT_MPR which carries relevant operational parameters.

The input signal which triggers the reception operations is the MPDU_RCVD signal coming from the Slot Sequencer. It is received each time a MPDU has been received by the Slot Sequencer. If the received MPDU carries piggybacked reservation requests, the signal RES_REQ_RCVD is sent to the Traffic Recorder FE.

If the received MPDU originates from an associated station, then the MPDU_BODY_RCVD signal is sent to the DLR instance bound to the corresponding MVC/MAC address. This binding is set up / released thanks to the PID_REGISTERED / PID_DEREGISTERED signals sent by the GDL.

If the received MPDU originates from a MT which is not associated (i.e. not bound to any DLR instance yet), then this MPDU is forwarded to the special broadcast DLR instance. This situation occurs when receiving an association request MPDU, so that the signal MPDU_MT_AP_ASSOCIATION is sent to the ADG entity. This special behaviour of MASCARA internal interfaces is needed as the relevant DLR instance associated to the new MT are not created yet.

Finally, a MT_ALIVE signal is sent to AIA (AP I_Am_Alive entity) each time any valid MPDU is received. This signal is used to keep trace of active MT, without having to send them any explicit polling request. The signals from/to AMMPR are shown in the following Figure 20:

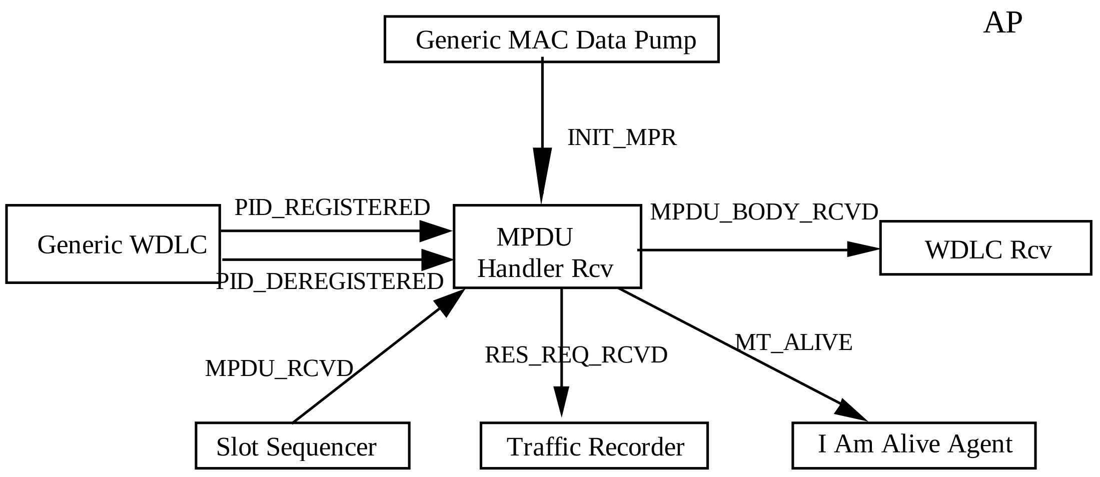


Figure 20 AMMPR interfaces.

7.3.2.2. The MT MPDU Handler Receive (MMMPR) Functionality

MMMPR is responsible for the MPDU reception at the MT side. It is initialised by the Generic MAC Data Pump thanks to the signal INIT_MPR, which carries relevant operational parameters. The signal SET_MAC_ADDR is used to update the current MAC address of the MT. The broadcast MAC address is nevertheless still accepted.

The input signal that triggers the reception operations is the MPDU_RCVD signal coming from the Slot Sequencer. It is received each time a MPDU has been received by the Slot Sequencer.

If the received MPDU corresponds to the control FH MPDU, then the signal FH_RCVD is sent to the Slave Scheduler FE and the signal FH_CAUGHT is sent to the MT MPDU Handler Transmit FE. Otherwise, if the received MPDU originates from an associated station, then the MPDU_BODY_RCVD signal is sent to the DLR instance bound to the corresponding MVC/MAC address. This binding is set up / released thanks to the PID_REGISTERED / PID_DEREGISTERED signals sent by the GDL.

Whenever an erroneous MPDU is received, a counter is incremented. The Handover Initiation block monitors the number of errors by sending a MPDU_ERRORS_req signal to the MPR. The MPR in turn sends the number of errors by the MPDU_ERRORS_cnf signal. It then resets its counter. The signals from/to MMMPR are shown in the following Figure 21:

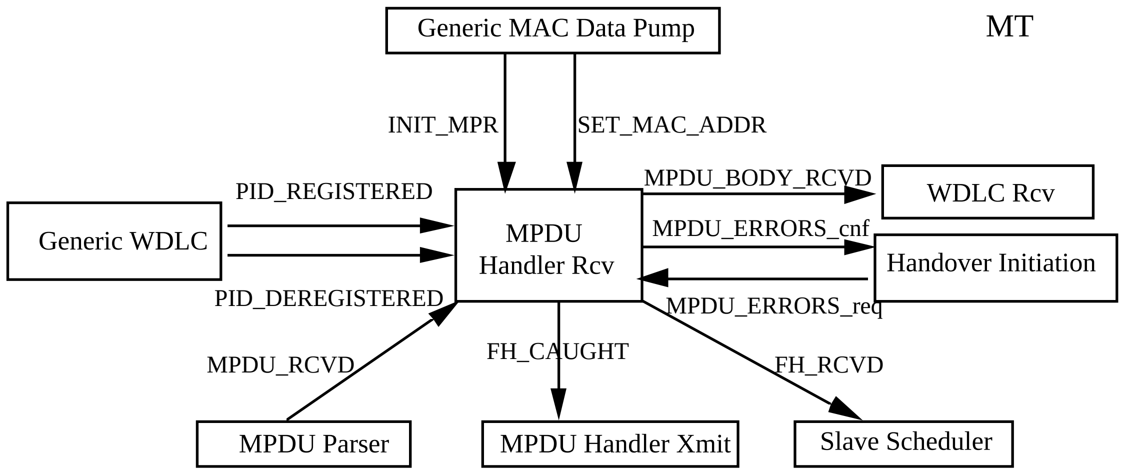


Figure 21 MMMPR interfaces.

7.3.3. Scheduler Description

In general, the Scheduler is responsible for scheduling the traffic transmitted through the wireless medium [PPVM97], PAMS97], [WFKM97]. In other words, it is the component that decides on the time an ATM cell will be transmitted. In MASCARA, we introduce two kinds of Schedulers: the Master Scheduler, referred to as ASC, which runs in the APs, and the Slave Scheduler, referred to as MSC, which runs in the MTs [DRMP96].

The task of ASC is to determine how the slots of each time frame are allocated to its associated MTs and to downlink transmissions. In its current form, ASC allocates slots in a per connection basis. The information on the assignment of all time slots together with the relative size of the three periods of the time frame, form the slot map, which is included in the Frame Header, broadcast at the beginning of each frame. ASC should also interact with the WCAC, in order to provide the best possible traffic contract preservation over the wireless link [HYLP93].

MSC is responsible for prioritising its own transmissions within the slots allocated to its connections by ASC. It can either be “passive” or “active”. In the first case, MSC blindly follows the allocation performed by ASC. In the second case, it can modify this scheduling within the constraint of allocated slots, and transmit the part of its traffic that must be serviced first, taking advantage of the up-to-date information it has about the status of pending ATM cells. This leads to improved protocol efficiency. For the demo, the “passive’ approach is chosen.

Both entities interact mainly with the Traffic Recorder (TRE) entity. TRE maintains the Traffic Table, where all the up-to-date information about active connections (such as, number of requests, arrival time of ATM cells, traffic parameters, etc.) are stored.

7.3.3.1. ASC Functionality

The basic operation of ASC is to decide on both uplink and downlink traffic to be sent, form the Frame Header and transmit it to the MPDU Transmit (MPX) entity. Upon receiving FRAME_START signal from the Slot Sequencer, ASC starts the construction of the next frame. It first sends the GET_TABLE signal to TRE, asking for the current Traffic Table. TRE responds with the SET_TABLE signal, sending to ASC the whole Traffic Table, containing cell pointers, reservation requests, etc. ASC runs the scheduling algorithm presented in [3uan021d], and decides on the number and particular positions of the slots allocated for each uplink and downlink connection. It then forms the Frame Header, and transmits it to MPX, together with the cell pointer lists of the downlink ATM cells that are going to be transmitted in this frame, through the FH_PAYLOAD signal.

One of the operations during this phase is the calculation of the contention period. Several existing algorithms can be adopted or adjusted, considering the peculiarities of the system, or even new algorithms can be designed.

Finally, ASC should inform TRE about the number of ATM cells of each connection that were transmitted. This information is required by TRE in order to update the Traffic Table (for ARQed connections) or discard the cell pointers (for non-ARQed connections). This is done through the UPDATE_TRE signal.

ASC interacts also with two entities of the Mascara Control (MCL) block: The I_Am_Alive_Agent (AIAA) and the AP Link Status Recorder (ALS). AIAA asks ASC, through the signal SEND_IAA_INVITATION, to send an I_Am_Alive invitation to a specified MT. The vehicle to carry this invitation is not yet decided, but it will be probably the Frame Header. The interaction with ALS involves two signals. ALS sends signal ASC_COLLECT_DATA_req to ASC, in order to gather the required information about the status of the radio link. ASC responds with ASC_COLLECT_DATA_cnf, which contains the requested information [3inn048a]. The signals from/to ASC are summarised in the following Figure 22.

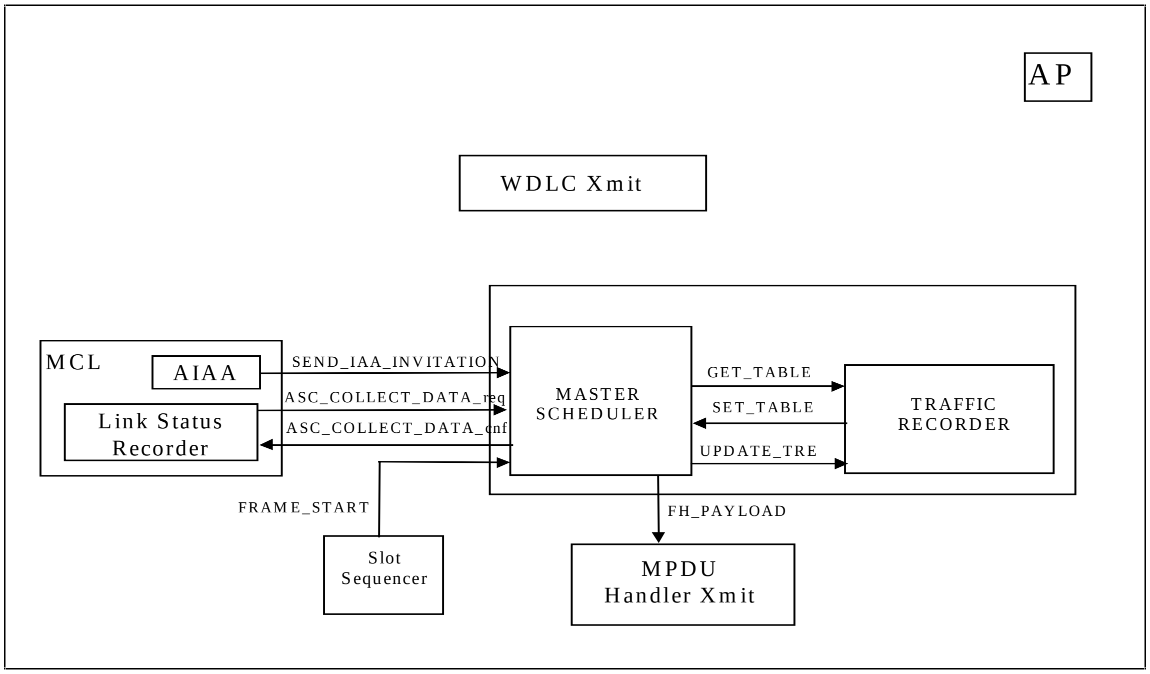


Figure 22 Scheduler interfaces.

7.3.3.2. MSC Functionality

The basic operation of MSC is to read the Frame Header, sent by ASC, and send the uplink ATM cells that ASC choose for transmission ("passive" operation), together with the new requests. In an extended system, MSC can also perform re-allocation of the slots allocated to the specific mobile ("active" operation).

In each frame, the operation of MSC starts with the receipt of the signal FH_RCVD, from MPDU Receive (MPR), used to inform MSC about the receipt of the Frame Header. MSC knows the memory position where the Frame Header is stored, through the MBUF signal received during starting of MPX. It then interprets the Frame Header to find out how many and which slots were allocated to the specific MT. It then requests the Traffic Table through the GET_TABLE signal to TRE. TRE responds with the SET_TABLE signal, sending to MSC the whole Traffic Table, containing cell pointers, and the number of successfully transmitted ATM cells for ARQed connections, as known by WDLC. Upon receipt of the SET_TABLE signal, MSC calculates the reservation requests for every connection by subtracting from the length of each cell list the number of allocated slots for the specific connection in the current time frame. MSC then forms a new signal called MT_FH_READY, destined to MPX.

Before sending MT_FH_READY, MSC has to piggyback in each MPDU, reservation requests of the connections that own them. Since MSC has the pointers to the memory positions where the ATM cells to be transmitted are stored, it can use existing ADT operators to fill the reservations field of all the ATM cells of each cell train to be transmitted, with the number of requested slots for the corresponding connections. Filling only the requests filed of the first ATM cell of each cell train would be sufficient, but in this case the system would not be robust to transmission errors.

Finally, MSC should inform TRE about the number of ATM cells of each connection that were finally transmitted. As already mentioned, this information is required by TRE in order to update the Traffic Table. This is done through the UPDATE_TRE signal, which is the same as in the ASC case.

When the MPDU headers are not enough to host all the reservation requests of connections that do not have allocations in the current frame, special control MPDUs have to be generated. This is a job performed by MSC. MSC allocates memory positions for control MPDUs (one memory position for every MPDU) and fills the fields, using existing operators. It then has to pass the list of pointers to these positions to MPX, along with information on the slot positions in the contention period that these control MPDUs will be transmitted. This is performed with the addition of an extra parameter on the MT_FH_READY signal, called Control_MPDU_List, which is a list of structures, where each structure consists of a) a pointer to the memory position where a control MPDU is stored, and b) an integer indicating the slot that this control MPDU will be transmitted. The signals from/to MSC are summarised in the following Figure 23.

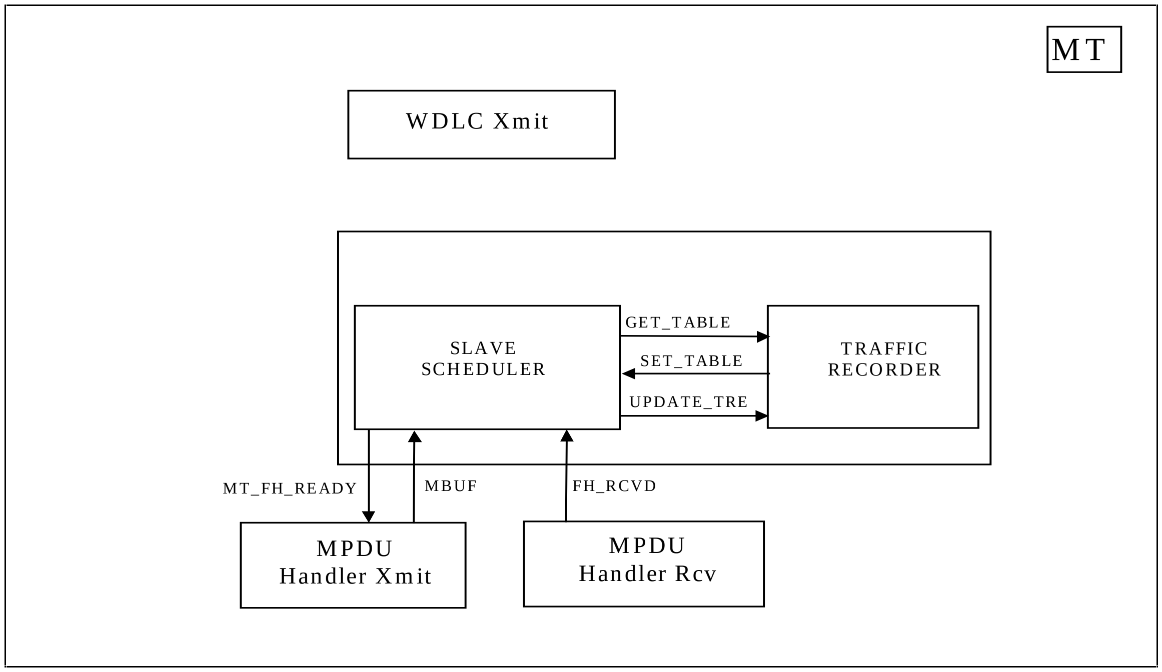


Figure 23 The signals from/to MSC .

7.3.3.3. Structure of Traffic Table

A visual representation of TRE's Traffic Table structure is given. This table is managed by TRE and is read by the Scheduler in order to get informed for the pending traffic needs of each active connection in order to build the slot map.

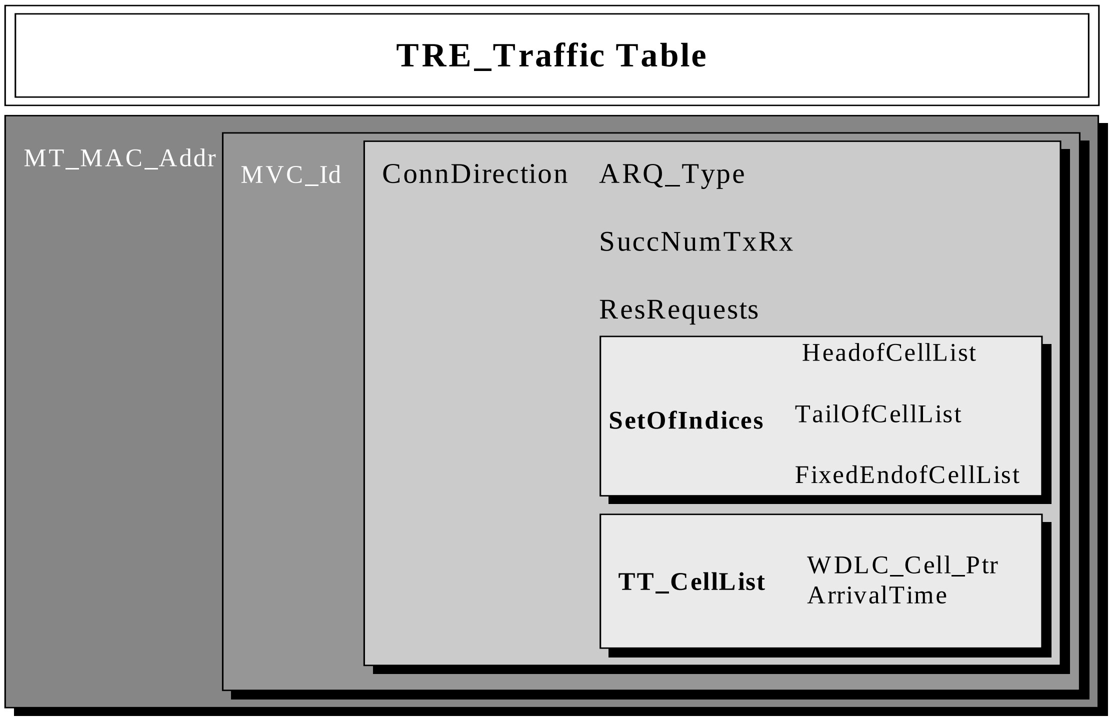


Figure 24 Traffic Table Information.

7.3.3.3.1. Connection parameters

Each connection (Traffic or Control) will have its stack of cell pointers indicating how many cells are pending transmission, or acknowledgement.

7.3.3.3.1.1. MT_MAC_Addr & MVC_Id parameters

The key to each connection is the pair of *MT_MAC_Addr*, *MVC_Id* parameters. This pair constitutes a unique key to every connection in the area of the AP. Note also that each connection is considered to be duplex, i.e. there is an uplink and a downlink connection for each MT_MAC_Addr and MVC_Id.

7.3.3.3.1.2. ConnDirection parameter

Because of the duplex connection, a third parameter need to be introduced, which is the connection direction. This parameter sets the right direction that the MAC_Addr and MVC_Id are referred to.

7.3.3.3.2. ARQ_Type parameter.

This parameter has a form of a flag. It will be sent at the beginning of the connection stack creation within the Traffic Table. This stack creation coincides with the creation of a new connection during the connection setup.

7.3.3.3.3. SuccNumTxRx parameter

This parameter is assigned the values of SuccNumRcv or SuccNumTx parameters if the ConnDirection assumes the values of UPLINK or DOWNLINK, respectively.

7.3.3.3.4. TT_CellList Parameter

TRE will keep for every connection (and direction) a stack, named *TT_CellList*. Parameter TT_CellList is utilised by the interface between WDLC and TRE for exchanging information via a relevant signal. Depending on the type of connection (ARQ & non-ARQ) and of direction (UPLINK & DOWNLINK), TT_CellList is used differently within the scope of TRE.

More specifically:

For non-ARQ, DOWNLINK connections, TT_CellList holds the pointers to the cells, plus their arrival time information, which is used by the ASC algorithm.

For non-ARQ, UPLINK connections, TT_CellList holds only an estimation of the arrival time of cells which are physically located at the MT. These cells wait for allocation of slots by the ASC which have been requested via the reservation request mechanism.

For ARQ, DOWNLINK connections, TT_CellList holds only the pointers to the cells since the arrival time information is irrelevant for this type of connection.

For ARQ, UPLINK connections, TT_CellList is not applicable and remains empty.

7.3.3.3.5. ResRequests parameters

The *ResRequests* field is exclusively used by ARQ, UPLINK connections and holds the number of requests received by the AP from the MT.

7.3.3.3.6. SetOfIndices parameters

The *SetOfIndices* parameters are used for the efficient management of the Traffic Table pertaining to insertions, deletions, and shifting of cells.

7.3.3.4. TRE Functionality in AP.

The main functionality of TRE is to keep an up-to-date status of the traffic to be transmitted by the ASC. To this end, it has to constantly receive information of incoming traffic from the WDLC block. This is achieved via the frequent reception of the TRAFFIC_UPDATED signal originating from every WDLC instance representing an active connection.

The structure of the TRAFFIC_UPDATED signal is the following:

TRAFFIC_UPDATED	MT_MAC_Addr
	MVC_Id
	TT_CellList
	NumOfRequests
	SuccNumXmit
	SuccNumRcv
	TrafficStatus

The use of the parameters is the following:

- MT_MAC_Addr corresponds to the MT owning both the downlink and the uplink connection (all connections are duplex).
- MVC_Id is the identifier of the pair of uplink and downlink addresses. The pair (MT_MAC_addr, MVC_Id) identify a pair of one uplink and one downlink connection.
- The TT_CellList is a list of pointers to the memory positions hosting the ATM cells that are going to be transmitted, together with their arrival times. The latter is applicable only to non-ARQed connections. These are either new ATM cells or in case of ARQ connections cells that were not received correctly and must be retransmitted.

- The parameter NumOfRequests carries the number of requested slots for the corresponding uplink connection. DLR has to interpret the incoming uplink ATM cells by reading the cell header to get the piggybacked number of requests. It then has to include this number to the ACKS_REQS_RDY signal destined to DLX. Since this number has to pass to TRE, DLX includes it to the TRAFFIC_UPDATED signal.
- SuccNumXmit is the number of correctly received by the MT ATM cells (i.e. downlink connection).
- SuccNumRcv is the number of correctly received by the AP ATM cells (i.e. uplink connection). Both parameters (SuccNumTx, SuccNumRcv) are not used in the MT, since their use is to assist the Master Scheduler in the orderly implementation of the Leaky Bucket. This task is solely handled by ASC.
- The parameter TrafficStatus, is boolean. Its value can be either APPEND (true) or CLEAR (false) and its use will be explained later in the document.

On the other hand TRE has to delete all those entries in its traffic table that have been serviced in the last Time Frame. The information carried by the UPDATE_TRE signal, is required by TRE in order to update the Traffic Table (for ARQed connections) or discard the cell pointers (for non-ARQed connections).

7.3.3.5. TRE's Traffic Table Management in AP.

At the centre of TRE's functionality lies the management of the Traffic Table. The management mainly consists of a) inserting and holding the pointers of cells awaiting DOWNLINK transmission, and the reservation requests pertaining to cells of UPLINK connection and received via the RES_REQ_RCVD signal or piggybacked on the ATM cell header, and b) the deletion of cells and requests which have been serviced by the Master Scheduler. The logic of performing the aforementioned tasks depends on the type of connection under consideration as well as on the direction of it.

Cells of ***DOWNLINK*** connections are given to TRE by the WDLC block, via the TRAFFIC_UPDATED signal triggered by each DLX instance representing the connection. This signal carries the pointers to the cells of the downlink connection as well as the arrival time of each cell. The latter is only applicable to non-ARQ connection.

In particular, cell pointers passed via the TRAFFIC_UPDATED signal and belong to non-ARQ connections are always appended at the end of the specific connection's cell list kept at the Traffic Table. Hence, TrafficStatus always assumes the value APPEND. In contrast, cell pointers of ARQ connections are appended at the end of the specific connection's cell list only in two cases: i) when the signal was issued by the arrival of M new ATM cells, or, ii) when the signal was issued by the arrival of acknowledgements, but all the previously sent cells encapsulated in an MPDU have been correctly acknowledged by the MT, thus TrafficStatus equals APPEND. If, however, a negative acknowledgment is received by the AP, DLX will send a new cell list which includes retransmissions and new arrived cells not yet send to TRE. The indication of TrafficStatus will be CLEAR.

The stack information for the ***UPLINK*** direction of each connection will be acquired both from the TRAFFIC_UPDATED signal from DLX and RES_REQ_RCVD signal from the MPR.

In case of ARQ connections there is no need for TRE to make use of a cell stack since only the number of requests submitted by the MT is useful information for the Scheduler’s algorithm; the current value of the number of requests is kept in the ResRequests parameter of the Traffic Table.

On the other hand, for non-ARQ connections a cell stack similar to the one kept for downlink connections is utilised by the traffic table. However, no explicit reference of the exact pointer value where the cell can be found is possible since the exact placement of the uplink cells is physically located at the MT’s Host Memory and not at the AP. Each uplink stack though, will be as the stack of downlink connections, but will have null pointer values at the cell pointer parameter. The logic will remain the same, as the number of uplink ATM cells will inform the scheduler on the uplink connection needs. Concerning the arrival times of uplink ATM cells, TRE has to estimate it, since this information is not transmitted along with the requests. The estimation algorithm should take into account the fact that requests may arrive at the AP via different paths (MPR or DLX), the possibility of loosing the reservation requests and/or of loosing the FH carrier of allocated slots requested previously. The estimation algorithm is described below.

While TRE accepts new or retransmitted cells and reservation requests in its Traffic Table so as always to inform Scheduler about pending traffic, ASC requests the current status of Traffic Table via the GET_TABLE signal. Getting the table ASC allocates slots based on the information received and on its internal logic. It, then, has to send feedback to TRE via the UPDATE_TRE signal in order TRE to update the Traffic Table according to the decisions taken by ASC. TRE examines the values of parameters carried via the UPDATE_TRE and updates its Traffic Table executing different tasks according to the direction and the type of connection:

a) DOWNLINK, non-ARQ

TRE deletes all those entries for which slots have been allocated in the current Time Frame. It also releases ATM memory since it is not needed any more. Note that ASC may return a value for the number of the allocated slots that is higher than the actual number of the allocated slots. That is because ASC returns in the same parameter the actual number of allocated slots plus the number of all those cells that have been expired and thus no allocation of slots is necessary. From TRE’s point of view the returned number is taken as if it corresponds to transmitted cells since it uses it to release ATM memory.

b) DOWNLINK, ARQ

TRE also deletes the entries corresponding to cells being allocated slots and thus transmitted in the current Time Frame. No release of ATM memory is exerted as this task is performed by WDLC when it receives the acks.

c) UPLINK, non-ARQ

As already mentioned uplink connections are serviced via reservation requests sent from the MT towards the AP. The requests are kept in the AP TRE Traffic Table and managed as follows.

MT sends for the first time its requests for a specific connection either via piggybacking in the MPDU header of another connection's MPDU or via a special Control MPDU in the absence of any uplink traffic. These requests are delivered in TRE via the RES_REQ_RCVD signal originated from MPR. Then TRE makes an equivalent cell entry in the Traffic Table for the connection under consideration for each reservation request and estimates the arrival time at the MT of each cell entry. TRE then gives the traffic table to ASC which in turn informs TRE on how many slots have been allocated to the specific connection in the current Time Frame. TRE then deletes these entries and awaits for the next arrival of requests. This arrival is realised via the WDLC since now the requests are piggybacked on the cell headers arrived as cell train (MPDU) in the slots previously allocated.

Note that the MT always sends its requests in absolute value, i.e. it requests allocation of slots for every cell entry in its traffic table no matter if some of these cells have been requested in a previous Time Frame but have not been serviced yet.

Subsequent requests are delivered via the TRAFFIC_UPDATED signal and TRE appends to its Traffic Table cell entries which amounts to the difference between the value of the number of requests and the current number of cell entries. Moreover, it estimates the arrival time of each new cell entry (see subsection 3.2.5). Using this scheme, TRE always preserves the estimated arrival time value of each cell of each connection between successive Time Frames as well as the Traffic Tables in AP and MT have identical entries in corresponding connections.

d) UPLINK, ARQ

This is simpler than the previous case. Since MT always sends reservation requests in absolute value, each time TRE receives the number of requests overwrites the previous value of the ResRequests parameter with the new one. The only complication arises when a cell train carrier of new reservation requests arrives at AP (DLR) and one or more of these cells is incorrect. Then DLR ignores the demand of the MT in terms of requests and assigns the value of zero (0) to the NumOfRequests parameter. The reason behind this is, that even though AP reports back to MT the error via the RN value in the immediate Time Frame the retransmissions due to the errored cell(s) will appear at least in the next Time Frame. Therefore, it is a waste of bandwidth if AP allocates slots in the immediate Time Frame which will be used by cells that are not going to get accepted by DLR as the SN is not the same with the RN.

TRE should be able to perform the management of its Traffic Table as efficient as possible. Among the tasks frequently encountered during management are inserting, deleting, shifting, etc. Therefore, a dynamic approach is needed in order TRE not to be the bottleneck of the scheduler block. Towards this direction a set of special indices has been introduced in the structure of Traffic Table encapsulated in the parameter, named SetOfIndices.

The indices have been chosen such as each Cell List entry in the Traffic Table to get managed as a circular queue. Since the insertions and the deletions occur at the bottom or the top of the physical array, we have used a pair of indices which corresponds to these actions.

More specifically, TailOfCellList assumes the value of the array index that holds the last cell pointer arrived via the TRAFFIC_UPDATED signal. On the other hand, HeadOfCellList assumes the value of the array index holding the first cell pointer to be deleted due to successful transmission.

From the discussion above it is obvious that after some time in operation we would reach the end of the physical array and thus we need to know the exact array-index value in order not to attempt to perform an invalid insertion. In this case, HeadOfCellList assumes the value of the first array index i.e. zero. The same action is taken for TailOfCellList index.

7.3.3.6. Arrival Time Estimation

As the philosophy of the scheduling algorithm is to perform a worst case estimation, the estimated arrival time that will be included in the Traffic Table is the beginning of the previous frame, since this is the time that MSC got informed on the needs of the uplink connections. TRE considers the beginning of each frame by the receipt of a signal.

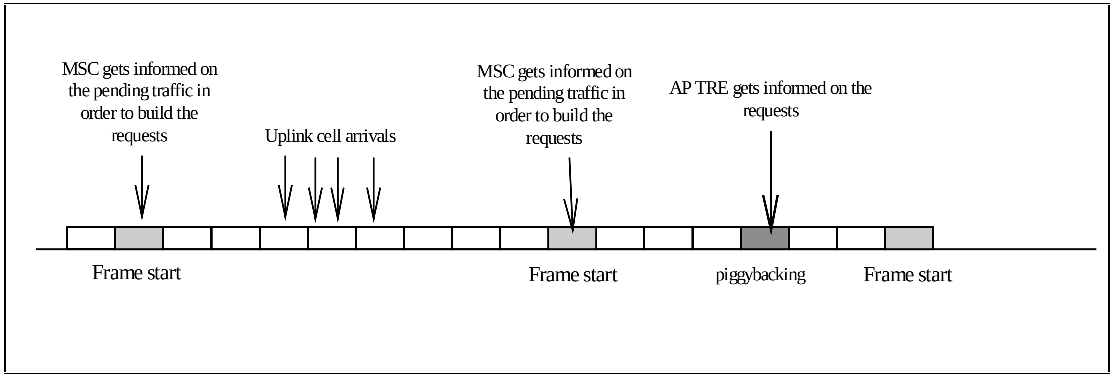

MSC gets informed on the pending traffic in order to build the requests
Uplink cell arrivals
MSC gets informed on the pending traffic in order to build the requests
AP TRE gets informed on the requests
Frame start
Frame start
piggybacking
Frame start

Figure 25 Worst case estimation.

Each time AP TRE receives the reservation requests via two relevant signals, it updates its traffic table by inserting a number of “fake” entries for the corresponding non-ARQ connection, equal to the number of requests. By “fake” we mean entries where the ATM cell pointer is null. TRE puts in the “ArrivalTime” field of each fake entry the time at which the previous frame header was transmitted. Therefore new arrivals at the MT during the previous time frame will show up as requests at the AP during the current Time Frame and will be allocated slots during the next Time Frame.

However, this situation may be reversed when a FH is lost carrying the allocation of slots. Since AP TRE is not aware of the missing FH, it will delete the entries for which slots have been allocated and new slots will be allocated for the remaining cell entries in the next Time Frame. These new slots will be interpreted by the MT as allocation of slots for cells that should have been serviced during the lost Time Frame. When MT sends cells during the allocated slots period along with new requests, a number of these requests includes cells that an entry in AP TRE traffic table no longer exists due to the lost FH and they will appear in TRE as new entries with a false (overoptimistic) arrival value estimation. These cells most likely would never get transmitted since they would expire in MT and other cells would take their place. It is evident that the effect of a lost FH persists for some time and will result in lost cells due to expiration. The two Traffic Tables are expected to get aligned some time later.

7.3.3.7. TRE Functionality in MT.

TRE functionality in MT is analogous with the TRE functionality in AP with the exception that parameters NumOfRequests, Succ_Num_Xmit, Succ_Num_Rcv are not applicable in the MT.

7.3.3.8. TRE's Traffic Table Management in MT.

The management of the MT TRE Traffic Table applicable only to UPLINK connections is analogous with the management of the AP TRE Traffic Table for downlink connections

1.

8. MASCARA Interfaces

8.1. User Plane Interfaces

8.1.1. ATM Layer Interface

The ATM Layer user-plane interface utilises the ATM Adaptation Layer 0 to transmit and receive ATM cells. When a new connection is set-up, the Generic Data Link block of the DLC layer interacts with the ATM layer. This interaction includes the procedure of forwarding the ATM cells of the specific connection to the appropriate DLX entity created for this reason. Thus, when a new ATM cell of a certain connection needs to be forwarded to the MAC layer, a signal is sent to the appropriate DLX entity, indicating the pointer of the ATM cell in the memory.

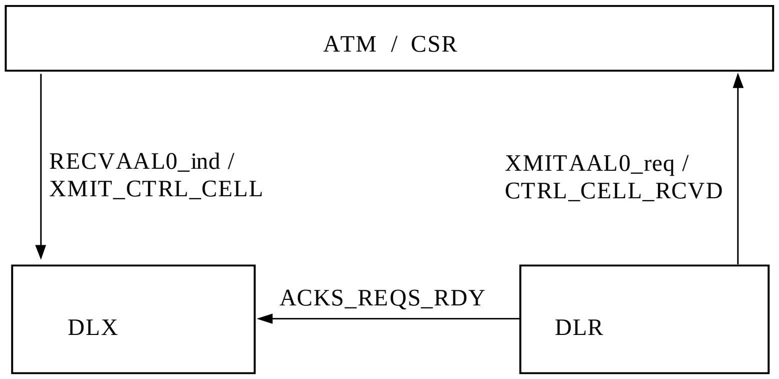


Figure 26 WDLC interface with the ATM layer

8.1.2. Physical Layer Interface

8.1.2.1. Frame Processor

The frame processor makes up the lowest layer of MASCARA, which is implemented as hardware. Basically, it is a Direct Memory Access (DMA) processor that executes a batch of I/O commands from a queue, at user-specified times.

The prime function of the frame processor is to control the transmission and reception of packets (MASCARA PDUs, or MPDUs) in real-time. Thereby it relieves higher, software-based layers of MASCARA (upper MASCARA, for short) of *timing critical* tasks. These tasks must be fulfilled at precise time instants. This does not free upper MASCARA of *time critical* tasks. These tasks must also be completed before a precisely defined deadline. However, the frame processor can take some work out of upper MASCARA’s hands. In particular, time consuming tasks require processing of data units smaller than an ATM cell, notably CRC generation, checking (which requires byte-by-byte processing) and slot map parsing.

The frame processor is controlled by so called Timing Element Descriptors (TEDs) or descriptors for short, that are sent to it by upper MASCARA. The descriptors specify when packets must be transmitted or received by the PHY. Descriptors also specify the amount of data to get transferred as well as the memory location of the upper MASCARA where the data can be read from or written to.

The MT frame processor parses the Frame Header, the first packet of every frame. This function is not required in the AP. The FH contains the slot map that designates every slot in a frame as:

Uplink transmission (from MT to AP)

Either by a single, specified MVC of a specified MT (reservation mode),

or by any MT (from any, or from a specified MVC) (contention mode).

Downlink transmission (from AP to MT)

Either to a single, specified MVC of a specified MT (unicast),

or to all or a group of MTs to a specified queue (broadcast or multicast).

Void which is used to insert a slot of radio silence, to allow time to switch AP and MT radios from transmitting to receiving mode.

Parsing involves extraction of the relevant slots map entries from the slot map in the frame header. This function is not timing critical and could therefore also be performed in the upper MASCARA software. Hardware parsing relieves the MAC from the processing the slot map, which involves operations on data units smaller, then a cell.

Another service that the frame processor provides is CRC generation and checking. This is a computationally intensive (bit-level) task that is difficult to implement efficiently in software.

Finally, although this is not a frame-related function, the frame processor interfaces between the MAC software and the PHY's modem management interface.

8.1.2.1.1. Functional parts

The hardware frame processor consists of five functional parts:

8.1.2.1.1.1. Sequencer

The sequencer processes descriptors, which reads from a FIFO, (the sequencer queue). The sequencer queue is written into by the MAC. A descriptor is a command, with a timing tag that specifies the instant at which it must be executed. The time base is the internal slot count of the frame processor that is driven by the slot clock that the PHY provides. The sequencer controls a DMA processor that constitutes the actual data path between the memory of the MAC engine and the PHY.

8.1.2.1.1.2. DMA processor

The DMA processor performs the actual data transfers, under the sequencer's supervision. This remains invisible for the frame processor user, hence the MAC.

8.1.2.1.1.3. Parser

The parser monitors incoming packets. If the packet is a frame header packet, the parser extracts only those entries from the time slot map that address the current MAC (the MAC that uses the frame processor service). To that end the MAC must have related its address to the frame processor when it initialised it. The extracted entries are written into a FIFO queue (the parser queue), that is read from by the MAC. The sequencer notifies the parser when the incoming packet is an FH, so that it will not attempt to extract slot map entries from non-FH MPDUs.

8.1.2.1.1.4. CRC processor

The CRC processor calculates and checks either a 16 bit (or 32 bit) checksum over pre-defined regions of both incoming and outgoing packets. Different regions can have different length checksums. See section 6 for the specification of these regions.The last 16(or 32) bits of each region is replaced by the result of the checksum calculation. In a correctly received packet the result of the CRC calculation will be 0. In other words CRC checking and generation have the same implementation.

Over the MPDU headers a CRC is applied, over slot payloads either a 4-byte CRC (For FH MPDUs) or a 2-byte CRC(For other MPDUs) is applied.

To send a packet, the MAC prepares a 27-byte header, followed directly by a number of 52 byte cells in memory. At the position of the CRCs in the header and the cells, zero bits must be supplied. The MAC then sends a descriptor to the frame processor. The ‘ address’ field points to the first byte of the header, the ‘ length ‘ field specifies the number of cells follow the header. The command field is set to ‘ Tx ‘, and ‘ slot ‘ is assigned a slot number at which the transmission should take place.

When the slot count equals the specified slot in the *descriptor*, the frame processor resets its CRC accumulator (the reset value depends on the particular CRC chosen and has not been specified yet), and starts calculating the CRC (dividing by the polynomial) over the 27 byte header, on-the-fly, (i.e. as it is being forwarded to the PHY). The resultant 4 byte value (the remainder of the division of the message polynomial by the generator polynomial) is written over the all-zero CRC field, at the 24th to 27th byte position in the transmit byte stream.

When the frame processor starts the reception of an incoming packet, it resets its CRC accumulator (the reset value has not been specified yet), and starts calculating the CRC over the 27 byte header. The resultant 4-byte value (the remainder of the division of the message polynomial by the generator polynomial) replaces the last four bytes of the incoming header that contains the transmitted CRC.

The CRC accumulator is then reset again, and the frame processor starts calculating the CRC over the 54-byte cell. The resultant 2 (or 4) byte value replaces the 53rd (or 51st) through 54th byte (containing the transmitted CRC) of the incoming slot payload in the received byte stream. This is repeated until all cells of the MPDU have been received. The frame processor can differentiate between FH MPDUs and other MPDUs.

The procedures described above allow the MAC to assert if, and where a CRC error occurred by checking if the CRC fields in the MPDU contain 16 zero bits. If not, then the corresponding header or cell contains errors and the MAC can decide what to do with it.

If a CRC error was detected in the FH, be it in the header or in any of the cells that hold the *slot map*, then all *descriptors* that have been derived from this slot map and have already been sent to the sequencer queue must be revoked. The obvious way to prevent this is to have the MAC software check the CRC of each part of the slot before any slot map entry contained in this part is actually used.

8.1.2.1.1.5. Modem management

This part interfaces between the physical modem management interface (MMI) on the PHY (see section 9 for a definition), and the LPC entity in the MAC.

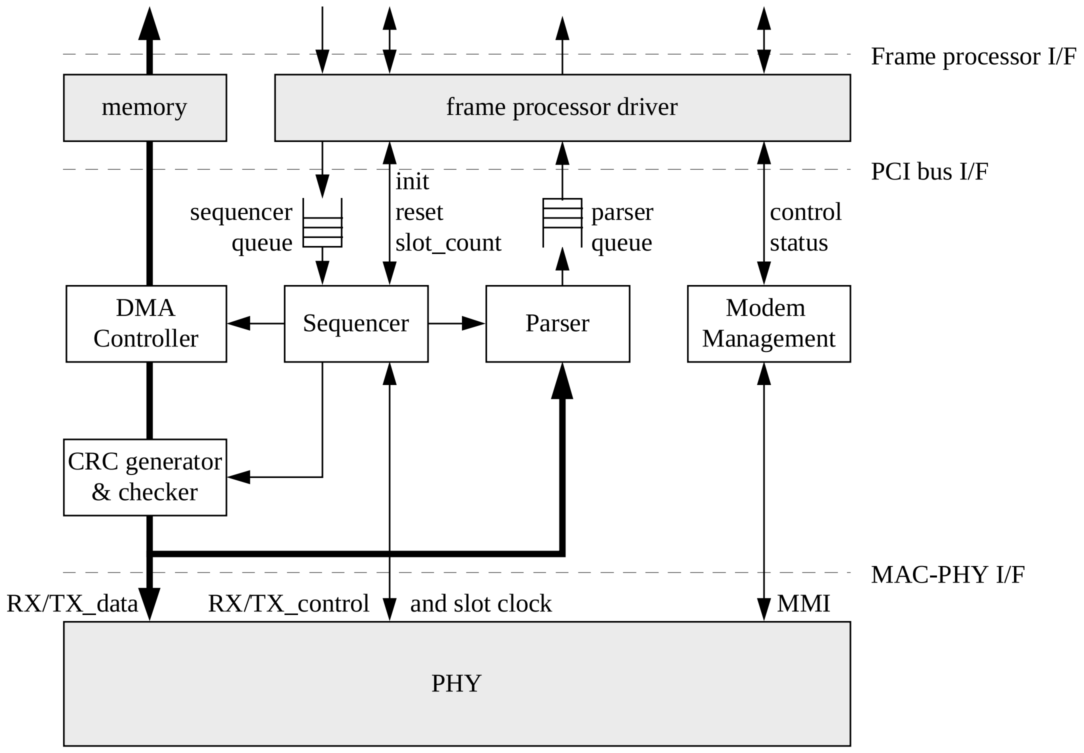


Figure 27 MAC-PHY Interface.

The white boxes are the functional blocks of the frame processor. The sequencer controls all blocks and the PHY. The modem management is independent.

The frame processor is described in Deliverable 3D3 and specified in internal design notes. Not being a hardware description language, the MASCARA SDL specification attempts to describe the Frame Processor’s behaviour as accurately as possible. It’s main purpose is to enable SDL system simulation, not to serve as a design specification of the Frame Processor.

The FPE block in the SDL system is controlled by the upper layers of MASCARA by the following signals.

Dir	**Signal**	**Parameters**	**Function**
In	TED	Command Slot Address Length	Put a descriptor in the descriptor queue.
Out	FRAME_START	none	Indicates to the AP scheduler that a new frame has started.
	MPDU_RCVD	SlotCount, Address	Indicates reception of an MPDU, received at slot *SlotCount*, stored at *Address* specified. This signal is generated in response to the arrival of an indicator from the frame processor.
In	MMI_MEASURE_RSSI	none	Request latest RSSI measurement
	MMI_SELECT_CHANNEL	ChannelNumber	Switch the PHY to a different channel.
Out	MMI_MEASURED_RSSI	MeasuredRSSI	Returns last measured RSSI

The TED signals sends a *command* for execution to the frame processor. The commands are:

Command	Interpretation
SOF_TX	start of frame at the AP: next transmission has long training.
SOF_RX	start of frame at the MT: next reception has long training
TX	transmit MPDU of given length stored at specified memory location
RX	receive MPDU of given length and store it at specified memory location
RX_OFF	resets PHY, to break out of Frame Header Hunt

The *length* and *address* arguments are used in conjunction with the TX and RX commands only. The *length* specifies the MPDU length, measured in payload slots. If the length is n, the MPDU length in bytes is found as 27 + n × 54. The address is a 32 bit host memory address. The 32 bits (or dword, for double word) correspond to an addressing range of 4Gbyte, but, as host memory is addressed dword-wise via the PCI bus, the address needs to be divisible by 4. In fact the two least significant bits are be ignored and zeroed by the Frame Processor.

The *slot* argument identifies the slot in which the command must be executed. A TX command with slot *x* tells the frame processor to transmit a packet in slot *x*. Since the FP may be required to send signal to the PHY before the packet is actually sent, this means that the actual dequeueing and processing of those descriptors, (and, for simplicity, for any descriptor), starts Δ slots early. If the length of the packet is m slots, then the packet will occupy slots x, … x+m, where slot x is used for transmission of the PHY header and the 27 byte MASCARA header, while slots x+1,...,x+m are used to transmit the m payload slots. Execution commences in slot x - Δ. If the packet is a frame header, it has a long PHY header, which occupies an additional slot before the taken by the PHY and MAC header, so the packet will occupy slots x,…,x+m+2.

Descriptors are queued, which implies that they must be ordered according to their slot value. Only one descriptor can be executed per slot, which means that the slot values of successive descriptors must differ by at least 1. A descriptor’s execution time can fall in the execution interval of the preceding descriptor, i.e. its slot argument can be less than or equal to x+m+1. In that case the descriptor preempts the execution of the preceding descriptor. Also, a descriptor can be too late. It is always overdue when the value of its slot is less than the current slot. If that is the case, the descriptor is discarded from the head of the queue.

A slot count of value 0 has a special interpretation. The descriptor is executed in the first slot after the moment it has become head of the descriptor queue. Thus, when it enters an empty queue, it is executed in the beginning of the slot that is following the current one. When it enters a full queue, it begins execution one slot after that in which the preceding descriptor started execution.

The FRAME_START signal is indicates the instant when the AP must start scheduling. MASCARA issues a SOF_TX for execution at some later time. When the descriptor is processed, the FP reminds MASCARA that the new frame has started, and the AP can assemble the frame header, which, when finished it can transmit by sending a TX descriptor. Thus, MASCARA does not have to keep track of time for this purpose.

The MPDU_RCVD signal notifies MASCARA that an MPDU has arrived. Again, even though MASCARA has scheduled the reception earlier, though the use of RX descriptors, the MPDU_RCVD signal serves as a reminder. Again, this relieves MASCARA from the obligation to remember when MPDUs are expected.

The FRP block controls the PHY by means of SDL representations of the signals, which are an approximation of the real PHY signals. SDL signals are sent instantly, while real signals have set-up times and delays.

The FRP consists of two blocks the SSD (Slot Sequencer, Descriptor queue) and SSE (Slot Sequencer Entity).

The SSD block queues descriptors submitted to the frame processor, and forwards them to the SSE blocks when they are due. When they are overdue, the SSD block discards them. The SSE blockexecutes the desriptors, depending on their type.

The SOF_TX descriptor causes the SSE to:

- switch off the RX modem,
- switch on the TX modem,
- stop the SSE's slot counter and set it to invalid

The SOF_RX descriptor causes the frame processor to:

- switch off the TX modem,
- switch on the RX modem,
- stop the SSE's slot counter and set it to invalid

The TX descriptor has different interpretation depending on the preceding descriptor. If the previous descriptor was an SOF_TX, then the TX descriptor results in the transmission of a Frame Header MPDU, otherwise the SSE requests transmission of a regular MPDU.

For the transmission of an FH MPDU, the frame processor's function is to:

- switch off the RX modem, if necessary,
- reset the slot counter to 0,
- retrieve MPDU data from memory and forward it to the PHY for transmission in a packet with a long preamble,
- switch off the TX modem at the end of the packet, if necessary.

For the transmission of a regular MPDU, the frame processor's function is to:

- switch off the RX modem, if necessary,
- switch on the TX modem, if necessary,
- retrieve MPDU data from memory and forward it to the PHY for transmission in a packet with a short preamble,
- switch off the TX modem at the end of the packet, if necessary.

The RX descriptor has different interpretation depending on the preceding descriptor. If the previous descriptor was an SOF_RX, then the RX descriptor causes the SSE to receive an FH MPDU. Otherwise, the SSE anticipates a regular MPDU.

For the reception of an FH MPDU, the frame processor's function is to:

- switch off the TX modem, if necessary,
- instruct the PHY to hunt for a FH MPDU,
- when the FH is received, reset the slot counter to 0 + T. Here, T is the transit time through the modem chain, comprising of modulation delay, propagation delay and demodulation delay. T is a multiple of the slot duration (currently 2).

- receive a packet with a long PHY header and subsequent packets with a short PHY preamble, and store the MPDU data in host memory.
- switch off the RX modem at the end of the packet, if necessary.

For the reception of a regular MPDU, the frame processor's function is to:

- switch off the TX modem, if necessary,
- receive all MPDUs within the reception window (defined by the descriptor's slot and length parameters), and store the MPDU data in host memory.
- switch off the RX modem at the end of the packet, if necessary.

The RX_OFF descriptor switches off the Receive modem. This descriptor is used when an MT hunted for a Frame Header without success. MASCARA then switches off the radio, switches it to a different channel and starts hunting for an FH on the new channel.

8.2. Control Plane Interfaces

8.2.1. Inter-Control Communication(ICC)

ICC is the FE that allows the other MASCARA FEs to interface, for control purposes, with the upper layers such as Q.2931m level at the MT or the MRR at the AP. The ICC FE is functionally identical in both AP and MT and is an implementation specific entity whose main role is to provide a transparent medium for the exchange of the control messages between MASCARA and the upper control layers. The main function of ICC is the encapsulation of MASCARA specific messages into AAL5 SDUs, so that the messages can be transferred over ATM, and the de-encapsulation of the messages received from ATM layer, in the form of AAL5 SDUs, into MASCARA messages as well as the delivery of these messages to the appropriate MASCARA entity. ICC performs no processing on the content of the incoming messages but, in some special cases, keeps track of the originator of a message so that it would be able to deliver the reply to that message to the correct entity.

ICC consists of two main entities. These are the Layer Control Protocol (LCP) and the Message Encapsulation Unit (MEU).

8.2.1.1. Layer Control Protocol (LCP)

LCP is a stop-and-wait like protocol controlling the exchange of the messages over AAL5. Its main process is the LCP Controller (LCN) which is the entity responsible for sending and receiving the AAL5 SDUs over ATM. It is also responsible for initialising AAL5 service.

For each LCP instance in MASCARA layer there is a corresponding LCP instance in the upper layer. Each LCP entity has 4 different ports. In order to achieve communication between two entities in two different layers, both entities should be bound to the same port of LCP, each one to the peer LCP instance residing in its layer.

Once the LCP is instanciated, the AAL5 is initialised and the communicating entities are bound to the appropriate LCP Port, LCP can send and receive messages to and from ATM layer using the AAL5 primitives.

When a message from MASCARA arrives at LCP, LCP sends the message to ATM by invoking the appropriate primitive and then waits until it receives an acknowledgement. Unless an acknowledgement from the peer LCP instance has been received, LCP does not send any more messages through the same port. If no acknowledgement has been received for a certain period of time the protocol times out and resend the last message. For each LCP Port there is a process called LCP Transmitter (LTX) which performs the above described control procedure.

In the opposite direction, when a message arrives from ATM to LCP, the LCP checks the sequence number of the message in order to ensure that it is the expected one. If the sequence number is correct, the message is passed to MEU for de-encapsulation, else it is rejected. However, in any case, all the messages that are correctly received by LCP are acknowledged. This control is performed for each port by a process called LCP Receiver (LRX).

The control messages are passed to AAL5 in the form of AAL5 SDUs. The structure of and AAL5 SDU as it is constructed in LCP is shown in the next figure:

1bit	28bits	1bit	2bits	2 Kbytes
Seq.	Reserved	Type	Port	LCP Data

The Sequence field indicates the sequence number of the SDU (it can either be 0 or 1), the Type filed indicates the type of the SDU (an SDU carrying data or just an acknowledgement) and the port field indicates the LCP port for which the SDU is destined for. There are also 28 reserved bits for future use. In case of a Data SDU the LCP Data are stored in the Data field which can be up to 2 Kbytes long. The Acknowledgement SDUs carry no data.

8.2.1.2. Message Encapsulation Unit (MEU)

This entity is responsible for the encapsulation of the MASCARA messages into an LCP_Data structure, that will be eventually transmitted over ATM, and vice versa. It is also responsible for instanciating LCP and delivering the messages, coming from the higher layers, to the correct entity in MASCARA.

When initialised, MEU should first of all instanciate LCP. When LCP is instanciated, since, in both AP and MT, MEU has to communicate with two different entities (RRM and NMX in AP, MMC and CCS in MT) it binds itself to the appropriate ports of LCP and instanciates two MEU Transmitter (MTX) processes for controlling the transmission of messages through these ports.

When a message from MASCARA arrives at MEU, MEU encapsulates it in and LCP_DataType structure Which should be passed to LCP. Prior to passing the

LCP_Data to LCP, the MTX checks whether LCP is ready to send i.e. does not wait for the acknowledgement of a previously transmitted message. If the LCP is ready to send the LCP_Data structure is passed to the appropriate LCP port for transmission. On the other hand, when a message from LCP arrives at MEU (in the form of an LCP_Data structure), it is de-encaspulated and the corresponding real message is constructed and sent to the appropriate MASCARA entity. This encapsulating and de-encapsulating procedure is performed by process MEU Controller (MCN).

In the table below the signal interface between LCP and MEU is described:

Service	**signal**	**parameters**	**description**
LCP - MEU Interface	LCP_INSTANTIATE_req	LCP_ConnectionIdentifierType	Instantiates the LCP
	LCP_INSTANTIATE_cnf	LCP_ConnectionIdentifierType, RC_Type	Confirms Instanciation of LCP
	LCP_BIND_req	HandleType, LCP_PortType	Requests binding to an LCP Port
	LCP_BIND_cnf	HandleType, LCP_PortType,RC_Type	Confirms binding to an LCP Port
	LCP_SEND_req	HandleType, LCP_DataType, LengthType, LCP_PortType	Requests transmission of a message through LCP
	LCP_SEND_cnf	HandleType, LCP_PortType, RC_Type	Informs sender entity (MEU) that the message was transmitted
	LCP_RECEIVE_cnf	HandleType, LCP_DataType, LengthType, LCP_PortType	Passes the received data to the appropriate entity (MEU)

Notes:

– The LCP_ConnectionIdentifier parameter specifies the ID of LCP instance .

– The Handle parameter specifies the LCP instance that an entity is bound to.

– The LCP_Port parameter specifies the LCP Port that an entity is bound or wishes to be bound to

– The LCP_Data parameter is a special structure that is used for storing the transmitted or received messages. The fields of this structure correspond to all the possible parameters that a message can carry.

– The RC parameter is used just for error handling and is used to indicate success or failure of an operation.

– The Length parameter specifies the length of the LCP_Data.

8.2.2. Radio Control (RCL)

The **Radio Control** which is responsible for the "non stationary", control functions of the MASCARA protocol related to the radio modem. For instance this FE is in charge of receiving RSSI information, or of putting the RF modem in dormant state.

8.2.2.1. Emitted Power Control (EPC)

This FE is responsible for driving the RF modem emitted power, according to policies not yet defined.

8.2.2.2. Measurements Functions (MEF)

This FE is responsible for recording the RSSI information provided by the RF modem upon reception of any valid MPDU. The MTC or the MHI triggers either such measurement. Their use is to keep the handover algorithm in MHI well informed about the link quality of the air medium while to inform MTC during the TIP process about the link quality of adjacent APs.

Signals that are exchanged between MHI/MTC and MEF are MEASURE_RLQ_req and MEASURE_RLQ_cnf, respectively, and between the MEF and PHY are PHY_MEASURE_RSSI and PHY_MEASURED_RSSI, respectively.

8.2.2.3. Radio Control Manager (RCM)

The functionality of the RCM FE is a repetitive pattern that is triggered for the first time when the MT powers on and each time that the MT decides to go TIP or enters the handover phase (backward or forward). It involves the following sequence of steps; a) it drives the RF modem (PHY) to tune in a specific channel and then b) it notifies the MDP (MPX) to start hunting for a FH. Upon reception of the MPX response (either successful or unsuccessful reception of the FH) RCL, in turn, notifies the FE that triggered the RCM's functionality in the first place.

The signals exchanged in each case are as follows:

1) Power on

GMC sends INIT_RCL signal to RCL, which, consequently, sends PHY_SELECT_CHANNEL signal to the RF modem to switch to a specific channel. After a period of duration 1 ms the RCL safely assumes that the switching has occurred. It then notifies MPX via signal FH_HUNT_req to start hunting for the FH. The result of the hunt procedure is carried in the

FH_HUNT_cnf and, if successful, together with the MAC address of the AP that corresponds to this channel. Finally, if hunting has been successful, RCL responds to the GMC's initial request via signal INIT_RCL_OK carrier of the channel number and the AP's MAC address . In case of unsuccessful hunting RCL requests from PHY to select another channel and this process goes on until an AP is found or all channels are exhausted. In the latter case RCL reports this result to GMC via signal INIT_RCL_KO.

2) TIP and Handover

Since both cases are eventually handled by the Target Cell (MTC), RCL receives signal ACQUIRE_NEW_AP carrier of the channel number to switch to. RCL again enters step a) and b) and if step b) was successful it responds to MTC with signal ACQUIRE_NEW_AP_OK, otherwise with signal ACQUIRE_NEW_AP_KO

9. References

[1D1] ACTS Project AC085 Magic WAND Deliverable 1D1 'Draft Prototype System Specification'.

[ABSO94] S. Abe, T. Soumiya, 'A Traffic Control Method for Service Quality Assurance in an ATM Network', IEEE JSAC, vol. 12, no. 2, February 1994, pp. 322-332.

[ACAM96] A. Acampora, Wireless ATM: A Perspective on Issues and Prospects, IEEE Communications Magasine, August 1996, pp. 8-17

[AHKM96] P. Agrawal, E. Hyden, P. Krzyzanowski, P. Mishra, SWAN: A Mobile Multimedia Wireless Network, IEEE Communications Magasine, April 1996, pp. 18-35

[ALAV96] J. Aldis, Althoff, R. Van Nee Physical Layer Architecture and Performance in the WAND User Trial System ACTS Mobile Summit '96 Proceedings, pp. 196-203 Granada, Spain Nov 96

[ALNS98] J. Ala-Laurila, S. Lorraine, N. Nikaein, J. Seppala Designing a Wireless Broadband IP System with QoS Guarantees ACTS summit 98 Proceedings Rhodes, Greece June 8-11, 98

[ATMF] The ATM Forum, Wireless ATM Group

[AWKR96] G. Awater, J. Kruys Wireless ATM - an overview Journal on Mobile Networks and Applications - Oct 96

[AYEK96] E. Ayanoglou, Y. Eng, M. Karol, Wireless ATM: Limits Challenges and Proposals, IEEE Communications Magasine, August 1996, pp. 18-34

[BMMP96] F. Bauchot, G. Marmigere, L. Merakos, N. Passas, S. Decrauzat MASCARA, A MAC protocol for Wireless ATM ACTS Mobile Summit '96 Proceedings, pp. 647-651 Granada Spain Nov 96

[BRAN] ETSI Broadband Radio Access Networks Standardisation Project

[CCYC97] C. Chang, K. Chen, M. You, F. Chang, Guaranteed QoS Wireless Access to ATM Networks, IEEE JSAC, January 1997, pp.106-118

[DBBM98] S. Decrauzat, C. Bonnet, F. Bauchot, G. Marmigere Wireless ATM MAC Dynamic Control within WAND wmATM 98 Proceedings Hangzhou, China April 98

[DRMP96] I. Dravopoulos, I. Marias, N. Pronios Distributed Aspects in Wireless ATM Networks International Workshop on Mobile Communications pp. 154-158 Thessaloniki, Greece Sep 96

[DRPR98] I. Dravopoulos, N. Pronios, 'A Comprehensive Handover Mechanism for Wireless ATM Networks', Wireless Mobile ATM, Hanghzou, China, 6-10 April 1998.

[ELMI93] A. I. Elwalid, D. Mitra, “Effective Bandwidth of General Markovian Traffic Sources and Admission Control of High Speed Networks”, IEEE/ACM Transactions on Networking, vol. 1, no. 3, June 1993, pp. 329 - 343.

[EMW95] A. Edwalid, D. Mitra, R. Wentworth, ‘A New Approach for Allocating Buffers and Bandwidth to Heterogeneous Regulated Traffic in an ATM node’, IEEE JSAC, vol. 13, 1995, pp. 1115-1127.

[GAPU96] D. Gaiti, G. Pujolle, ‘Performance Management Issues in ATM Networks: Traffic and Congestion Control’, IEEE/ACM Transactions on Networking, vol. 4, no. 2, April 1996, pp. 249-257.

[HAHI96] H. Hansen, S. Hadjiefthymiades, J. Immonen, A. Kaloxylos handover algorithm for the Wireless ATM Network (WAND) ACTS Mobile Summit '96 Proceedings, pp. 543-548 Granada, Spain Nov 96

[HPFM98] S. Hadjiefthymiades, S. Paskalis, G. Fankhauser, L. Merakos Mobility Management in an IP-based Wireless ATM Network ACTS Summit 98 Proceedings Rhodes, Greece June 8-11, 98

[HYLP93] J. M. Hyman, A. A. Lazar, G. Pacifici, “A Separation Principle Between Scheduling and Admission Control for Broadband Switching”, IEEE journal on Selected Areas in Communications, vol. 11, no. 4, May 1993, pp. 605 - 616.

[I.371] ITU-T Recommendation I.371 ‘Traffic Control and Congestion Control in B-ISDN’.

[KADM98] A. Kaloxylos, G. Dravopoulos, H. Hansen Mobility Management Extensions in the Wireless ATM Network Demonstrator ACTS Summit 98 Proceedings Rhodes, Greece June 8-11, 98

[KAHA96] A. Kaloxylos, S. Hadjiefthymiades Mobility Management in WAND International Workshop on Mobile Communications - Thessaloniki, Greece 19-20 Sep 96

[KAHM98] A. Kaloxylos, S. Hadjiefthymiades, Prof. L. Merakos Mobility Management & Control Protocol for Wireless ATM Networks IEEE Network Special Issue on PCS Network Management

[KEWC93] G. Kesidis, J. Warland, C. Chang, “Effective Bandwidths for Multiclass Markov Fluids and Other ATM Sources”, IEEE/ACM Transactions on Networking, vol. 1, no. 4, August 1993, pp. 424 - 428.

[KOMM95] K. Kawahara, Y. Oie, M. Murata, H. Miyamara, ‘Performance Analysis of Reactive Congestion Control for ATM Networks’, IEEE JSAC, vol. 13, no. 4, 1995, pp. 651-661.

[KUWU96] T. Kuehnel, Y. S. Wu Seamless Interworking of Wireless and ATM Networks IEEE ATM '96 Workshop TA1 San Francisco, U.S Aug 96

[LUPP97] A. Lupper Evaluating the Market for Mobile Broadband Applications Mobile Broadband Conference Proceedings London Mar 97

[MAAL98] J. Mikkonen, J. Aldis, G. Awater, A. Lunn, D. Hutchinson The Magic WAND - Functional Overview IEEE JSAC Magazine Journal Issue Accepted for publication

[MAKU98] S. Martignoni,. Kuehnel Extension of ClassicalIP over ATM to Support QoS at the Application Level ICATM 98 Proceedings Colmar, France Jun 98

[MEIE98] J. Meierhofer Data Link Control for Indoor Wireless Networks Wireless 98 Proceedings July, 98

[MERA96] L. Merakos Magic WAND Project: Mobility Aspects Wireless ATM Workshop Proceedings Espoo, Finland 2-3 Sep 96

[MIHI96] H. Mitts, H. Hansın, J. Immonen Lossless handover for wireless ATM Mobicom'96 Proceedings, pp. 85-96 New York, U.S. Nov 96

[MIKK96] J. Mikkonen The Magic WAND project overview Wireless ATM Workshop Proceeding Espoo, Finland Sep 96

[MIKR96] J. Mikkonen , J. Kruys The Magic WAND: a wireless ATM access system ACTS Mobile Summit '96 Proceedings, pp. 535-542 Granada Spain Nov 96

[MSAD98] G. F. Marias, D. Skyrianoglou, A. Andritson, G. Dravopoulos Dynamic Channel Allocation in WAND ACTS summit 98 Proceedings Rhodes, Greece June 8-11, 98

[NIKR96] M. Niemi, J.Kruys Wireless ATM Standardisation ACTS Mobile Summit '96 Proceedings, pp. 250-255 Granada, Spain Nov 96

[PAME96] N.Passas, L. Merakos A Medium Access Control Framework for Wireless ATM International Workshop on Mobile Communications - Thessaloniki, Greece19-20 Sep 96

[PAMS97] N.Passas, L.Merakos, D.Skyrianoglou Traffic Scheduling in Wireless ATM Networks IEEE ATM'97 Workshop Proceedings Lisbon, Portugal May 97

[PEEL96] H. G. Perros, K. M. Elsayed, “Call Admission Control Schemes : A Review”, IEEE Communications Magazine, November 1996, pp. 82 - 91.

[PETR96] D. Petras, A. Hettich, A. Kramling ‘Design Principles for a MAC Protocol of an ATM Air Interface’ In PIMRC'96, Granada, Spain, Nov. 1996.

[PMSB98] N. Passas, L. Merakos, D. Skyrianoglou, F. Bauchot, S. Decrauzat MAC Protocol and Traffic Scheduling for Wireless ATM Networks ACM Mobile Networks and Applications Journal Journal Issue Accepted for publication

[PPVM97] N. Passas, S. Paskalis, D. Vali, L. Merakos, 'QoS Oriented, Medium Access Control for Wireless ATM Networks', IEEE Communications Magazine, November 1997.

[PRDA98] N. Pronios, I. Dravopoulos, A. Andritsou, A DLC Layer Framework for Wireless ATM Networks, accepted in PIMRC 98, Boston, U.S.

[RAYC96] D. Raychadhuri, Wireless ATM Networks: Architecture, System Design and Prototyping, IEEE Communications Magasine, August 1996, pp. 42-49

[RFSB97] D. Raychadhuri, L. French, R. Siracusa, S. Biswas, WATMnet: A Prototype Wireless System for Multimedia Personal Communication, IEEE JSAC, January 1997, pp. 83-95

[RSKJ91] C. Rasmussen, J. Sorensen, K. Kvols, S. Jacobsen, 'Source-Independent Call Acceptance Procedures in ATM Networks', IEEE JSAC, vol. 9, 1991, pp. 351-358.

[SAIT92] H. Saito, 'Call Admission Control in an ATM Network using UpperBound of Cell Loss Probability', IEEE Transactions on Communications, vol. 40, 1992, pp. 1512-1521.

[TAHP94] L. Tassiulas, Y. Hung, S. Panwar, 'Optimal Buffer Control during Congestion in an ATM Network Node, IEEE/ACM Transactions on Networking, vol. 2, no. 4, August 1994, pp. 374-386.

[WFKM97] R. Wood, G. Fankhauser, R. Kreula, E. Monni QoS Mapping for Multi-Media Applications in A Wireless ATM ACTS Mobile Summit '97 Proceedings Aalborg, Denmark Oct 97

[ZAND97] J. Zander, Radio Resource Management in the Future Wireless Networks: Requirements and Limitations, IEEE Communications Magasine, August 1997, pp. 30-36

10. Appendix I. Scheduler Algorithm

The scheduling mechanism is critical for the performance of a reservation-based protocol, such as MASCARA. An arbitrary order of slot allocation from the AP, in accordance with some properties of the MASCARA protocol, such as UP/DOWN period separation and cell train construction, can alter the traffic pattern of a connection. This may result in violation of the contractual values of QoS and traffic characteristics, such as peak cell rate (PCR), cell delay tolerance (CDT), and cell delay variation tolerance (CDVT), and cause discarding of ATM cells deeper in the network, or late arrival at the receiver. The maintenance of contractual values for PCR and CDVT for uplink connections can be controlled with the use of a shaper at the fixed network port in each AP, while for downlink connections maintaining PCR and CDVT values in the radio part is less important since this is the last hop of the connection. CDT values for both uplink and downlink can only be controlled by a traffic scheduler, located at the AP, that takes into account the delay constraints of individual connections in the allocation of bandwidth.

In this section, a brief description of the scheduling algorithm is given for the Master Scheduler of MASCARA. It is called **P**rioritized **R**egulated **A**llocation **D**elay **O**riented **S**cheduling (**PRADOS**), and has two main objectives:

i) traffic regulation based on traffic characteristics, and

ii) maintenance of the delay constraints of the connections in the radio interface.

We refer to the Master Scheduler as simply “the Scheduler”. At the beginning of each frame, the Scheduler has a number of pending “requests” for slot allocation to service, which are either downlink ATM cells waiting to be transmitted, or uplink reservation requests, piggybacked in the data MPDUs. The algorithm can be separated in two independent actions, which are performed in parallel:

1. specification of how many requests for slot allocation from each active connection will be serviced in the current frame, and
2. determination of the exact location in the frame of the time slot allocated to each serviced request.

For the first action, the algorithm combines priorities with a leaky bucket traffic regulator [ECK89]. It sorts connections based on their service class [ATM89], and assigns priorities to them (the larger the priority number the higher the priority).

Additionally, a token pool is introduced for each connection. Tokens are generated at a fixed rate equal to the mean cell rate, and the size of the pool is equal to the “burst size” of the connection [ATM89], as declared and agreed upon at the time of connection setup. For every slot allocated to a connection, a token is removed from the corresponding pool. In this way, at any instance of time, the state of each token pool gives an indication of the declared bandwidth that the corresponding connection has consumed. A token pool is implemented as a token variable, which is increased by one every time a token is generated, and decreased by one every time a slot is allocated to the corresponding connection. The token variable of a connection is allowed to take negative values when slots are allocated to the connection while its token pool is empty.

The first action is divided in two steps. In the first step, the Scheduler services “conforming” requests, defined as requests that belong to connections whose token pool is non-empty (i.e., positive token variable). Starting from priority class 5 (CBR), and going down to priority class 2 (ABR), the Scheduler services requests from connections, as long as tokens and slots are available. UBR connections have no guaranteed bandwidth, thus no token pool is maintained for them, and they are not serviced during this step. At every priority class, it is very probable to have more than one connections having ATM cells to transmit. In that case, PRADOS gradually allocates one slot at a time to the connection (or connections) that possesses the most tokens (i.e., highest token variable), removing one token from the corresponding pool. The rationale is that the connection with the most tokens has consumed proportionally the least bandwidth compared to its declared one, and thus has higher priority for getting slots allocated. At this state, one or both of the following statements hold:

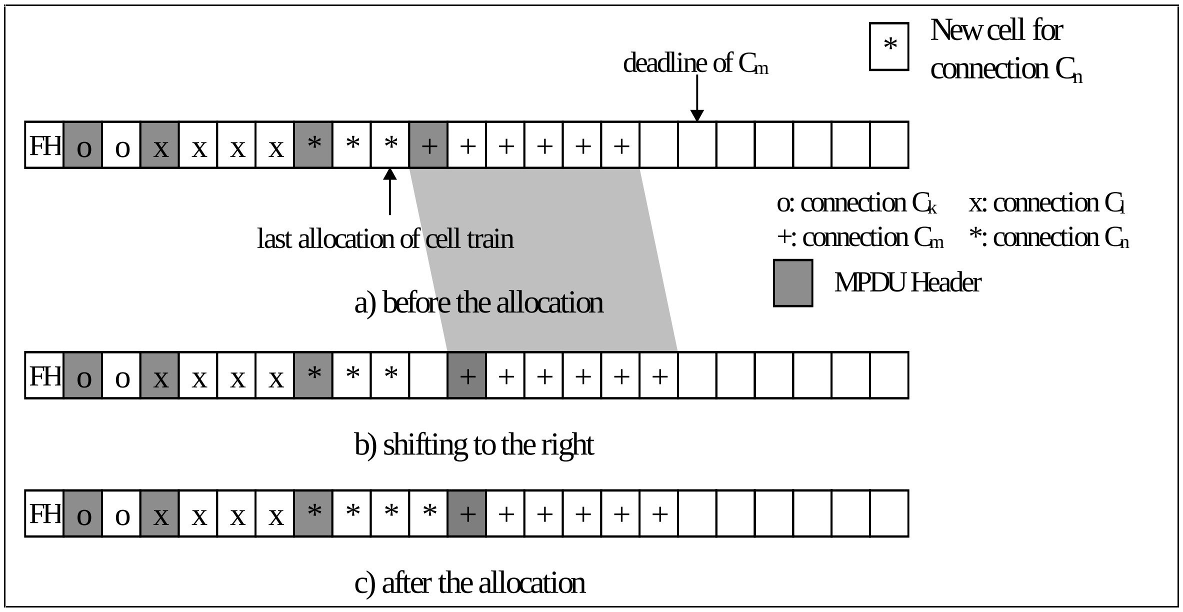
i) all token pools are empty (i.e., token variables are less than or equal to zero),

ii) all requests have been satisfied.

If only statement (i) holds, the Scheduler proceeds with the second step, which involves service of “non-conforming” requests, i.e., requests from connections with non-positive token variable. It starts again allocating slots beginning from priority class 5 (CBR), and down to priority class 1 (UBR), following the same procedure as described above.

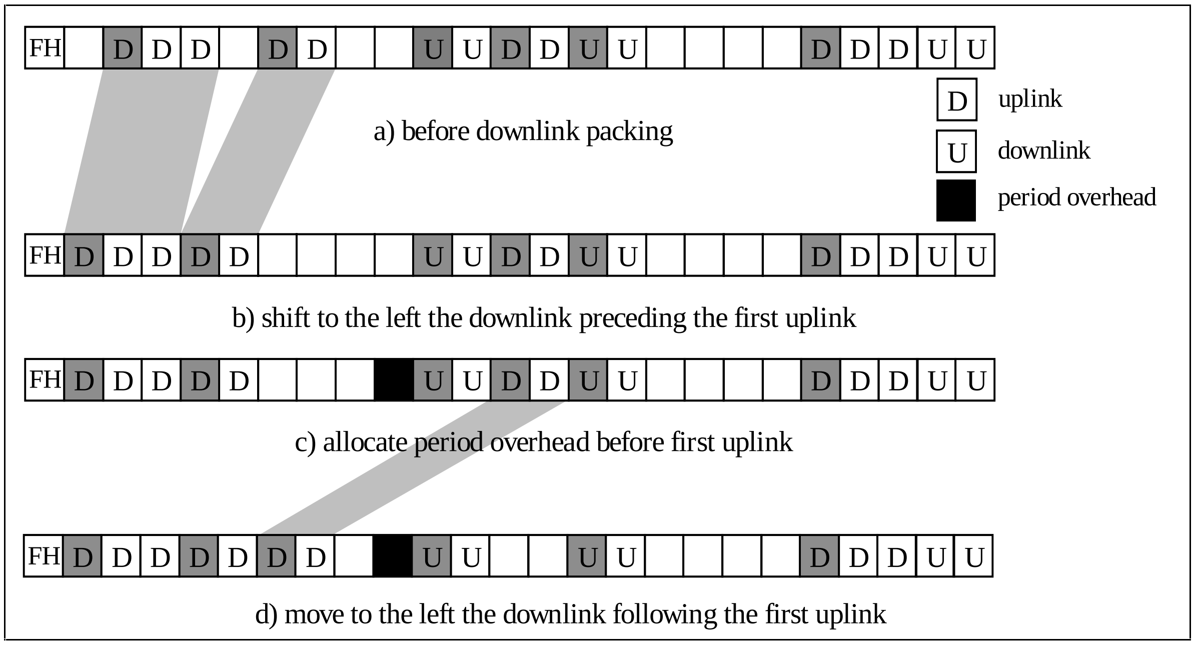
For the second action (determination of the location in the frame of the slot allocated to each serviced request), PRADOS is based on the intuitive idea that, in order to maximize the fraction of ATM cells that are transmitted before their deadlines, each ATM cell is initially scheduled for transmission as close to its deadline as possible [LIN96]. To attain high utilization of the radio channel, the algorithm is “work-conserving”, meaning that *“the channel never stays idle as long as there are ATM cells requesting transmission”* [ZHA91]. Consequently, the final transmission time of an ATM cell will be the earliest possible given the ATM cell’s initial ordering. The Sheduler allocates slots gradually and constructs the time frame in such a way, so that to satisfy the wireless hop CDT of each connection. The wireless hop CDT can be evaluated by decomposing end-to-end CDT into CDT for each hop of the ATM connection path. If an allocation causes violation of the deadlines of existing allocations, then this allocation is not performed. Below we briefly describe the operation of the algorithm for this second action.

The operation of the second action can be divided in three steps. The purpose of the first step is to make the initial transmission ordering, based on the deadlines. When a request corresponding to an ATM cell is selected for service, the Scheduler attempts to allocate one slot for its transmission. If the request is the first of the corresponding connection, then the algorithm attempts to make the allocation before and as close to its deadline as possible. If shifting of the existing allocations is required, the algorithm ensures that none of them exceeds its deadline. If this is not possible, the allocation is not performed. If the request is not the first for the corresponding connection, the algorithm tries to make the allocation at the end of the connection's cell train, provided again that, if shifting is required, no allocation exceeds its deadline. An example illustrating this procedure is shown in AI.1. When all pending requests have been processed, the Scheduler proceeds to the second step.



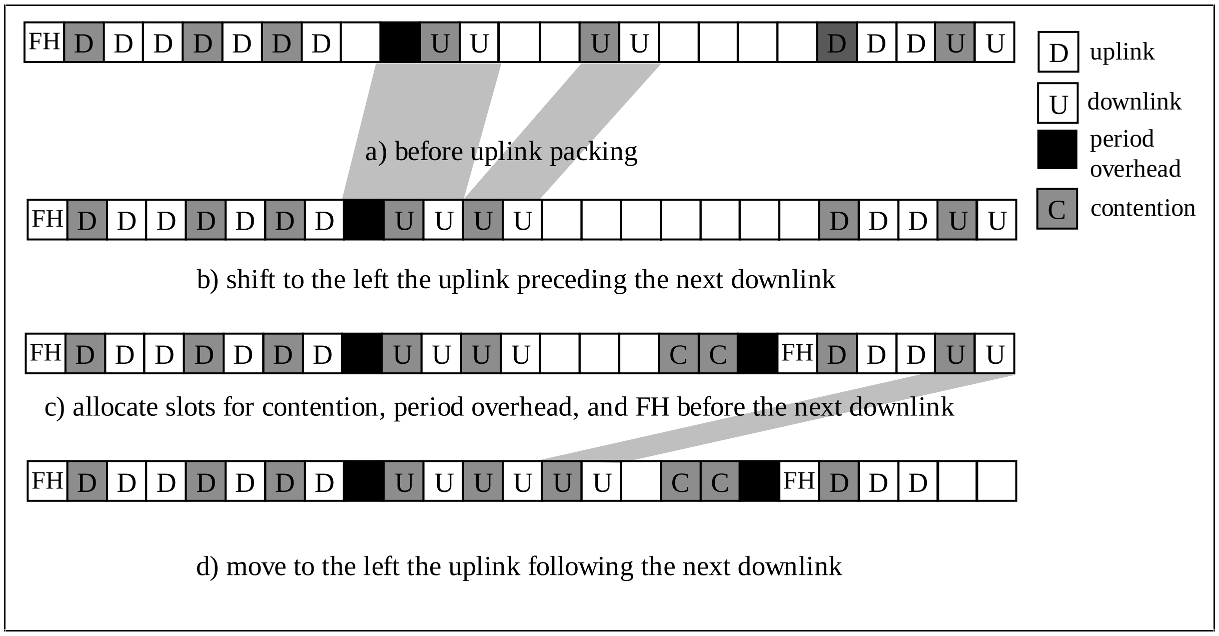
A I. 1 Example of allocation when there are no free slots before the deadline

In the second step, the DOWN period of the frame is built. The Scheduler packs, as close to the beginning of the frame as possible, all allocations between the beginning of the frame and the first slot allocated to an uplink connection (clearly all these allocations correspond to downlink connections). In the space left empty between the last packed downlink allocation and the first uplink allocation, the algorithm adds the period overhead, and tries to pack as many downlink allocations as possible by moving them to the left (AI.2)


FH
D
U
a) before downlink packing
D uplink
U downlink
period overhead
b) shift to the left the downlink preceding the first uplink
c) allocate period overhead before first uplink
d) move to the left the downlink following the first uplink

A I. 2 Packing of downlink allocations

The purpose of the third step is to build the UP period. The operation is analogous to the second step, but now packing is performed between the end of the DOWN period, as produced from the second step and the first unpacked downlink allocation. In the space left empty between the last packed uplink allocation and the first unpacked downlink allocation, the algorithm adds the CONTENTION period, the required period overhead and the frame header, and tries to pack as many uplink allocations as possible by moving them to the left.


FH
D
U
C
a) before uplink packing
D uplink
U downlink
period overhead
C contention
b) shift to the left the uplink preceding the next downlink
c) allocate slots for contention, period overhead, and FH before the next downlink
d) move to the left the uplink following the next downlink

A I. 3 Packing of uplink allocations

The length of the CONTENTION period depends on the expected traffic, and the specific accessing method used. As already mentioned the CONTENTION period is used for control messages from the MTs to the AP. These control messages are single slot MPDUs, consisting of only a MPDU header including the control information. Two are the most important types of messages that use this period:

- association requests from MTs performing “power on” or handover, and
- reservation requests from uplink connections having traffic to transmit but no allocated slots to piggyback their requests in (e.g., after connection setup or after an idle period).

Quick transmission of these messages is essential for the performance not only of MASCARA, but of the WAND system in general (e.g., handover delay). On the other hand, since MASCARA control traffic is unexpected, the CONTENTION period should be minimized, as much as possible, to avoid waste of bandwidth. The random access algorithm used for the CONTENTION period is part of the traffic scheduling algorithm and depends on the type of feedback information provided to the contending MTs, which in turn depends on the kind of detection that can be provided by the physical layer. If a collision cannot be reliably detected (i.e., the physical layer cannot differentiate a collided slot from an empty slot), then the only available information to the Scheduler is from the number of successfully received messages. This limit the design choices of the random access algorithms that can be used to the class of ALOHA-type algorithms. In any case, the Scheduler should allocate a CONTENTION period length large enough to attain an acceptably low successful transmission delay for the control messages. Additionally, the MTs can contribute to the collision resolution process by appropriately reducing the probability of transmitting in the next frame in case of repeated collisions (backoff algorithms). On the other hand, if collisions can be detected, more efficient algorithms, based on collision resolution controlled by the AP, can be used.

1.

11. Appendix II. Improved WDLC Proposal

11.1. Introduction

This appendix discusses improvements and enhancements of the WDLC scheme developed for the Magic WAND demonstrator. The WDLC functionality of the demonstrator was kept relatively simple in order to allow the implementation within the given time frame. Basically, the demonstrator WDLC scheme applies a simple Go-Back-N (GBN) Automatic Repeat Request (ARQ) scheme. A detailed description of the demonstrator WDLC and its performance evaluation can be found in [6.1]. Moreover, the block structure of MASCARA / WDLC has some drawbacks which were imposed by SDT constraints at a very early design stage.

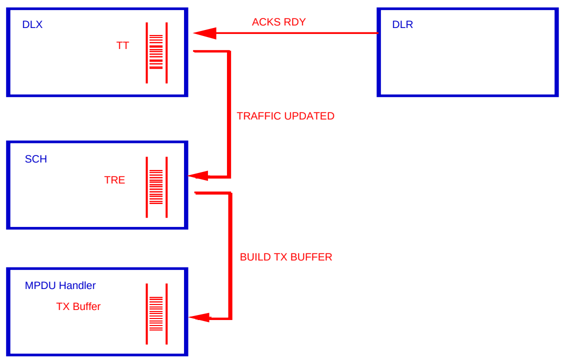
The first part of this section describes an improved block structure. Simulation results show how these modifications yield better cell transfer delay figures. Moreover, it turns out that (for normal traffic situations) less signal exchanges are required with the modified block structure. This is an essential improvement since this means that less processing power is required for the implementation of the WDLC algorithm on a given hardware platform.

The second part of this report describes an improved WDLC scheme that can be built on top of the existing scheme and the improved block structure, i.e., no fundamental changes are required to the existing scheme. By the introduction of selective cell discarding the cell transfer delay can be strictly bounded at the expense of an increased cell loss rate. This feature allows for a gradual differentiation between different traffic classes.

11.2. Limitations of the Demonstrator WDLC

11.2.1. Functional Description of the Demonstrator WDLC Scheme

In the demonstrator, there are basically three queues in series in the WDLC / MASCARA layer (see figure 6.1).



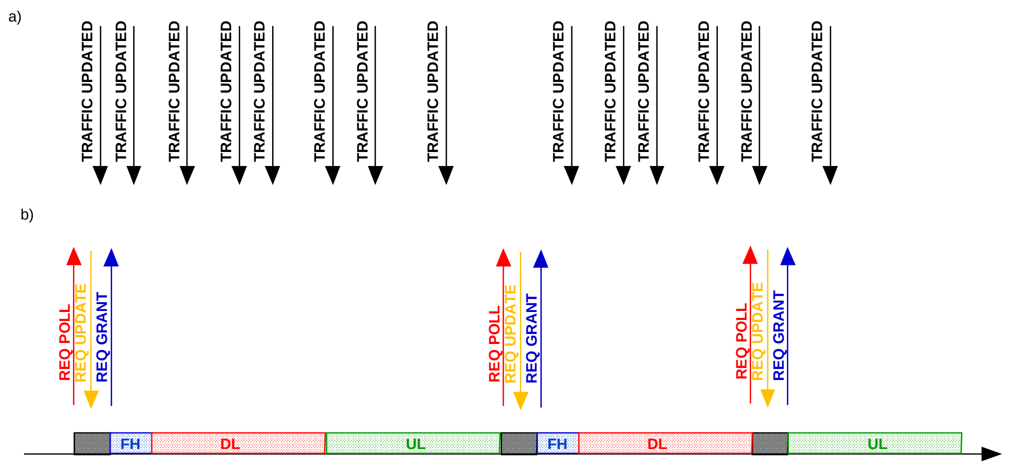
A II. 1 WDLC / MASCARA Queues

The WDLC transmit part (DLX) takes pointers of cells to transmit from the traffic table (TT) and gives them to the scheduler (SCH) through the signal TRAFFIC UPDATED. In SCH the pointers are stored in the traffic recorder (TRE). Before the start of a new frame, the MPDU (MAC protocol data unit) handler copies the cells corresponding to the slots allocated by the scheduling algorithm into the TX buffer (signal BUILD TX BUFFER), where they are at the disposal of the physical layer for transmission.

11.2.2. Drawbacks

11.2.2.1. Superfluous Signal Exchanges

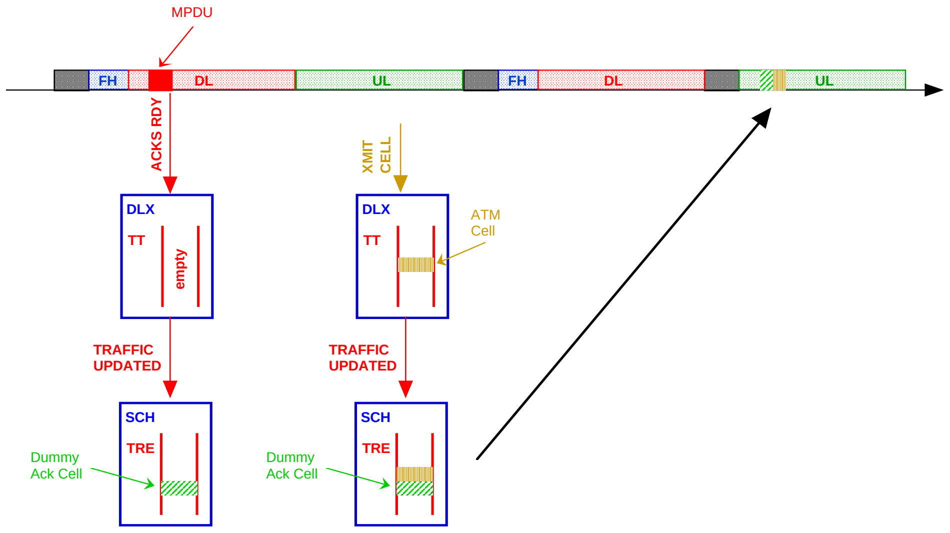
This block structure has a couple of drawbacks. First, SCH needs to update the number of requests only when a new scheduling phase starts, i.e., before the start of a new frame. However, since the frame length is variable, DLX has no information about the time when a new frame starts. In order to keep the SCH informed about the number of cells waiting for transmission at any point of time, DLX has to issue TRAFFIC UPDATED whenever it gets a new cell from the ATM layer (which happens very often) or an acknowledgement from the WDLC receive part (DLR) (ACKS RDY, which happens less often) (see figure 6.2). Therefore, the signal TRAFFIC UPDATED has to be exchanged very often, which would not be the case if SCH could inspect the TT exactly then when the current number of requests is required, i.e., at the start of a new frame. It will be shown later that by combining TT and TRE the number of signal exchanges in the WDLC/MASCARA layer can be reduced, which means that less processing power will be required by the demo platform to handle the protocol.



A II. 2 Signal exchanges between WDLC and SCH a) for the demonstrator architecture and b) for the improved architecture where TT and TRE are combined to one buffer

11.2.2.2. Superfluous Acknowledgement Cells

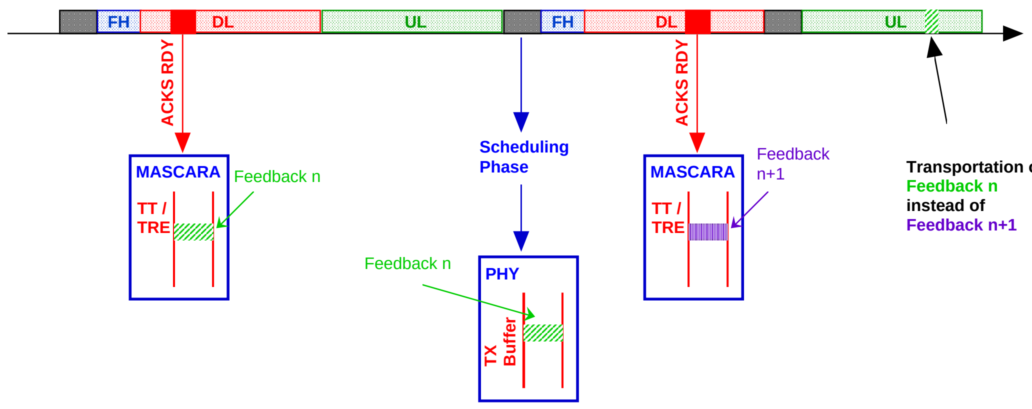
Another problem that arises out of the demonstrator block structure is the handling of special acknowledgement cells when no traffic is available in TT to piggy-back the. Whenever DLX receives the signal ACKS RDY from DLR carrying a new feedback value, it tries to piggy-back this feedback on the cells waiting in its TT. When the TT is empty at this point of time, DLX generates a dummy ack cell, piggy-backs the feedback on this cell and issues TRAFFIC UPDATED carrying a pointer to this cell. The problem now is that the signal ACKS RDY may occur at any instance of time somewhere inside the time frame (ACKS RDY is issued by DLR when the last cell of an MPDU has been received). Thus, a long interval may elapse between the issuing of ACKS RDY and the start of the next frame. The TT may well be empty when ACKS RDY is issued but there may be new traffic in TT by the time the next frame starts. However, since DLX has no information about the start of the next frame, it must generate a dummy ack cell as soon as it gets the signal ACKS RDY even though it may not be necessary. Figure 6.3 shows an example where a dummy ack cell is transmitted although there exists an ordinary user cell to carry the feedback. Hence, some of the dummy ack cells are superfluous and generate additional traffic in the system which deteriorates the overall system performance. It will be shown later that by combining TT and TRE it is possible to generate dummy ack cells only when they are really required, which can improve the system throughput and the cell transfer delay figures.



A II. 3 Transmission of superfluous acknowledgement cells

11.2.2.3. Outdated Feedback

The TX Buffer is built at the start of the time frame, i.e., the cells are copied from the host memory to the TX Buffer before the downlink phase starts. If an MT receives traffic during the downlink phase, there is a new feedback value that should be piggy-backed on the uplink cells. However, since the TX buffer has already been built and is not accessible to DLX, the new feedback cannot be transported up to the AP before the uplink phase of the next time frame, i.e., there is an unnecessary delay of one frame length. Figure 6.4 depicts an example where this situation occurs. The feedback (n) for the traffic received in the downlink phase of the first frame can be transmitted only in the uplink phase of the next frame. However, in the next frame, there is already the feedback value (n+1) ready for transmission. This problem can be circumvented by building the TX Buffer for uplink traffic shortly before the start of the uplink phase instead of at the start of the frame.

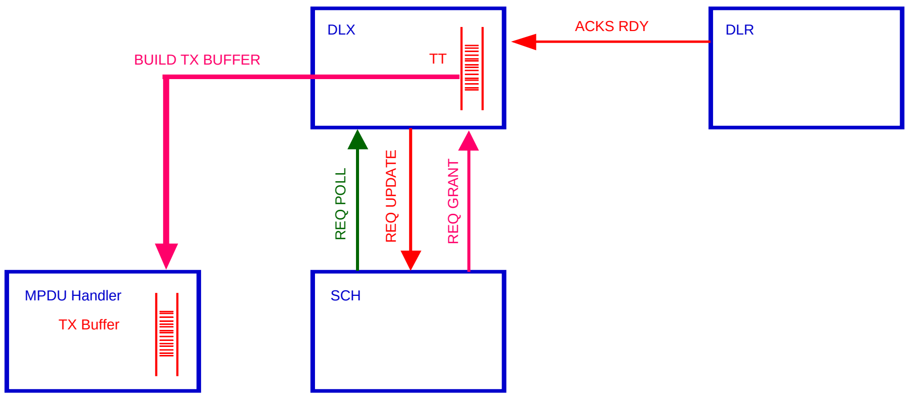


A II. 4 Superfluous delay for feedback in the uplink direction

11.3. Improved WDLC Block Structure

11.3.1. Functional Description

The TRE is required for the scheduler to know the number of cells waiting for transmission and to store the corresponding pointers. Keeping in mind that all this information is also contained in TT it is clear that the TRE is superfluous. The following architecture is proposed (see figure 6.5):



A II. 5 Modified WDLC / MASCARA Block Structure

There is no queue any more in SCH. Before the start of a new frame (i.e., when a new scheduling phase starts), the scheduler needs to know the number of requests for every connection. Therefore, SCH issues the signal REQ POLL to every instance of DLX, which then evaluates the number of cells that can still be transmitted according to the GBN ARQ policy. This number, which is equal to the number of requests, is given to SCH through the signal REQ UPDATE. The scheduler then runs the scheduling algorithm (leaky bucket), whose output is the slot map. Based on the slot map SCH counts the number of allocated slots for every connection and issues the signal REQ

GRANT carrying this number and the position of the cells in the frame to every DLX instance that has more than zero slots allocated. Upon this, DLX copies the corresponding number of cells into the TX buffer.

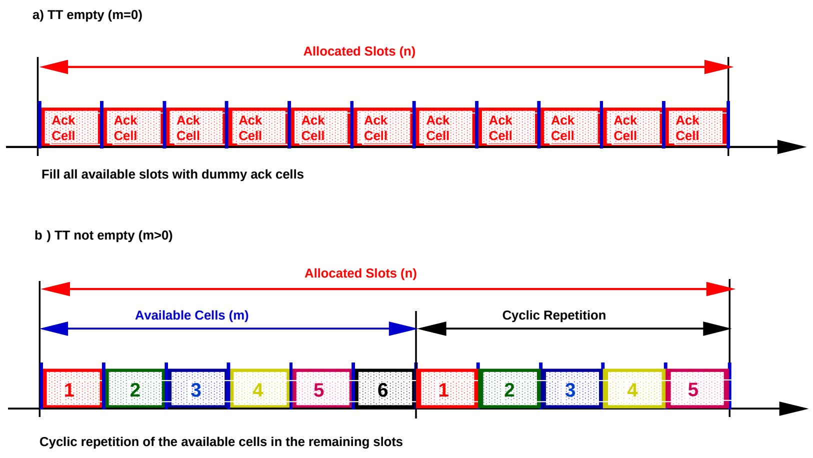
In contrast to what is done in the demonstrator, there is no signal exchange between DLX and SCH any more upon the reception of the signal ACKS RDY. The signal ACKS RDY simply effects that DLX deletes the acknowledged cells and moves the transmission window.

The sequence of signal flows REQ POLL, REQ UPDATE, and REQ GRANT may appear complicated at a first glance. However, it must be kept in mind that is happens only once per time frame. In contrast, with the structure of the demonstrator, the signal TRAFFIC UPDATED is issued for every ATM cell received from ATM and every time the signal ACKS RDY is received from DLR. For very low-rate traffic, the demonstrator scheme may cause less signal exchanges than the scheme presented here. But for connections that transmit more than 3 cells per frame the proposed architecture outperforms the demonstrator scheme (see also figure 6.2). Moreover, the new scheme has the advantage that the number of signal exchanges does not depend on the source rate, which means that the computation effort required to handle the protocol becomes more predictable.

11.3.2. Special Features

11.3.2.1. More Allocated Slots than Available Cells

For reasons that will become evident later, the situation may occur that DLX indicates that there is a certain number of cells n available for transmission through REQ UPDATE, but, by the time it gets the grants from SCH (signal REQ GRANT), there are only m<n cells available. It is anticipated here that this may be the case when selective cell discarding is applied or when the TX buffer for uplink connections is not built at the frame start but rather shortly before the start of the uplink phase. In such a case, there are slots allocated to a connection that are not needed. Since it is too late at this point of time to return unused slots to other users, there is only one reasonable reaction. If the number of available cells is m=0, all n allocated slots are filled with dummy ack cells, which may be useful in case the feedback has been lost before (see figure 6.6). In other words, DLX generates n dummy ack cells and copies them into the TX buffer. If, however, m>0 (but still m<n), the remaining (n-m) slots are filled with a cyclic repetition of the m available cells. In other words, some cells are sent several times. This redundancy may be useful since in case of a cell loss on the radio link there is still a chance that a following version of the same cell arrives correctly.



A II. 6 Use of superfluously allocated slots

11.3.2.2. Handling of Dummy Ack Cells

DLX keeps a flag to indicate if the newest feedback value has been sent over the air. We call this flag of type boolean newest_ack_sent (TRUE or FALSE). Whenever DLX receives the signal ACKS RDY, it checks if the feedback value to piggy-back on the cells is different from the feedback value received by the previous issue of ACKS RDY. If this is the case, the flag newest_ack_sent is set to FALSE and the piggy-backed feedback value of all cells available in TT (if any) is updated.

Next time DLX receives the signal REQ POLL from SCH, i.e., before the start of the next frame, DLX checks the number of cells available for transmission in TT (nr_of_available_cells). If nr_of_available_cells>0, i.e., the TT is not empty, the signal REQ UPDATE carries the value nr_of_available_cells (i.e., the number of requests is nr_of_available_cells). If, however, nr_of_available_cells=0, i.e. (the TT is empty), DLX checks if there is a new feedback value to be transmitted (i.e., if newest_ack_sent=FALSE). This case (i.e., the TT is empty and the newest feedback has not yet been sent) means that a dummy ack cell is required. Therefore, the signal REQ UPDATE is issued carrying the value 1 for the number of requests in order to request one slot for the dummy ack cell. If the connection gets a slot allocation for this request and the TT is still empty, we have the situation described above in the section “More Allocated Slots than Available Cells”. This means that DLX generates a dummy ack cell and copies it into the TX buffer.

11.3.3. Performance Evaluation

For the system scenario that has already been assumed for the demonstrator performance evaluation (12 source, 67.5% input load, MT-to-AP distance 23 meters) the improved block diagram yielded considerably better performance. While the mean cell transfer delay was 300ms for the demonstrator WDLC, it was only 150ms for the WDLC based on the improved block structure. The improved transport mechanism for the uplink feedback did not result in a dramatic improvement (mean delay 130ms).

1.

12. Appendix III. Signals and Signallists used in WAND MAC

Note: **.Ref:** means that the signallist contains a reference to another signallist instead of a list of signals.

12.1. PEER TO PEER INTERFACES

Signallist: MMAA2MAAA_INITIAL_MPDU MPDU MT Association Agent to AP Association Agent	
Event	Parameters
MPDU_MT_AP_ASSOCIATION	NetworkName MT_ATM_Addr MT_Name ConfigData HO

Signallist: MMAA2MAAA_MPDU MPDU MT Association Agent to AP Association Agent	
Event	Parameters
MPDU_MT_AP_MAC_ADDR_RECEIVED	MT_ATM_Addr MT_MAC_Addr RC
MPDU_MT_AP_DEASSOCIATION	MT_ATM_Addr MT_MAC_Addr

Signallist: MAAA2MMAA_MPDU MPDU AP Association Agent to MT Association Agent	
Event	Parameters
MPDU_AP_MT_ASSOCIATION_ACK	MT_MAC_Addr RC
MPDU_AP_MT_MAC_ADDR_ALLOCATE	MT_ATM_Addr MT_MAC_Addr AP_MAC_Addr

MPDU_AP_MT_DEASSOCIATION	MT_ATM_Addr MT_MAC_Addr

Signallist: MMMA2MAMA_MPDU MPDU MT MVC Agent to AP MVC Agent	
Event	Parameters
MPDU_MT_AP_CONN_ACTIVATE	MT_MAC_Addr MVC_Id ConnectionIdentifier

Signallist: MAMA2MMMA_MPDU MPDU AP MVC Agent to MT MVC Agent	
Event	Parameters
MPDU_AP_MT_CONN_ACTIVE	MVC_Id ConnectionIdentifier
MPDU_AP_MT_CONN_REFUSED	MVC_Id ConnectionIdentifier RC
MPDU_AP_MT_CONN_PENDING	MVC_Id ConnectionIdentifier

Signallist: MMTI2MATI_MPDU MPDU MT TIP Agent to AP TIP Agent	
Event	Parameters
MPDU_TIP	MT_MAC_Addr RC

Signallist: MATI2MMTI_MPDU MPDU AP TIP Agent to MT TIP Agent	
Event	Parameters
MPDU_TIP_ACK	MT_MAC_Addr

Signallist: MABA2MMBA_MPDU MPDU AP Beacon Agent to MT Beacon Agent	
Event	Parameters

MPDU_BEACON	NetworkName AP_Name AP_MAC_Addr OperationFrequency NeighAP_List

12.2.

AP MASCARA

PEER TO PEER MASCARA CONTROL TO SIGNALLING

Signallist: MALS2Ccon AP Link Status Recorder to Control	
Event	**Parameters**
ALARM_ind	**AlarmId** **AlarmInfo**
GET_INFO_cnf	**RequestType** **RequestInfo** **RC**
RR_STATUS_cnf	**GuaranteedAggregateBwUsed** **RetransmissionBwUsed** **RemainingBufferSpace** **MT_RadioLinkQuality** **MT_ATM_Addr** ***(ignored for the demo)***

Signallist: MAAA2Ccon Mascara AP Association Agent to Control	
Event	**Parameters**
ASSOCIATION_ind	**MT_ATM_Addr** **MT_Name** **RRM_ConfigData** **HO**
AP_DEASSOCIATION_ind	**MT_ATM_Addr**
AP_VC_RELEASE_cnf	**ConnectionIdentifier** **MT_ATM_Addr** **RC**

Signallist: MADG2Ccon Mascara AP Dynamic Gen Ctrl to Control	
Event	**Parameters**
AP_DEASSOCIATION_cnf	**MT_ATM_Addr** **RC**
AP_ASSOCIATION_LOST_ind	**MT_ATM_Addr**

AP_VC_RELEASE_cnf	**ConnectionIdentifier** **MT_ATM_Addr** **RC**

Signallist: MAMA2Ccon Mascara AP MVC Agent to Control	
Event	**Parameters**
VC_RESERVED_cnf	**ConnectionIdentifier** **OriginalConnectionIdentifier** **MT_ATM_Addr** **RC**
VC_SWITCHED_cnf	**ConnectionIdentifier** **OriginalConnectionIdentifier** **MT_ATM_Addr** **RC**

Signallist: MGMC2Ccon General Mascara Control to Control	
Event	**Parameters**
CLOSE_MAC_cnf	**RC**
OPEN_AP_MAC_cnf	**RC**

PEER TO PEER SIGNALLING TO MASCARA CONTROL

Signallist: Ccon2MAAA Control to Mascara AP Association Agent	
Event	**Parameters**
AP_VC_RELEASE_req	**ConnectionIdentifier** **MT_ATM _Addr**

Signallist: Ccon2MADG Control to AP Dynamic Gen Ctrl	
Event	**Parameters**
AP_DEASSOCIATION_req	**MT_ATM_Addr**

Signallist: Ccon2MAMA Control to Mascara AP MVC Agent	
Event	**Parameters**

VC_RESERVED_req	**ConnectionIdentifier** **OriginalConnectionIdentifier** **QoS** **WAND_ATM_Parameters** **MT_ATM _Addr**
VC_SWITCHED_req	**ConnectionIdentifier** **OriginalConnectionIdentifier** **MT_ATM _Addr**

Signallist: Ccon2MALS   Control to Mascara AP Link Status Recorder

Event	**Parameters**
GET_INFO_req	**RequestType**
RR_STATUS_req	**MT_ATM _Addr (ignored for the demo)**

Signallist: Ccon2MGMC   Control to Generic Mascara Control

Event	**Parameters**
CLOSE_MAC_req	***none***
OPEN_AP_MAC_req	**NetworkName** **AP_Name** **AP_MAC_Addr** **OperationFrequency** **Min_MT** **Max_MT** **BeaconPeriod** **MAC_ConfigData**

MASCARA - PHYSICAL INTERFACE (MMAC_PPHY)

Signallist: Mmac2Pphy   Mascara to Physical

.Ref: MEPC2PMMI, MMEF2PMMI, MRCM2PMMI, MSSE2PTXM, MSSE2PRXM

Signallist: Pphy2Mmac   Physical to Mascara

.Ref: PMMI2MMEF, PTXM2MSSE, PRXM2MSSE

CONTROL TO PHYSICAL LAYER (MMCL_PPHY)

Signallist: MEPC2Pphy Mascara Emitted Power Control to Physical Modem Management Interface	
Event	**Parameters**
MEPC2Pphy_signal	***TBD (Ignored for the Demo)***

Signallist: MMEF2PMMI Mascara Measurement Functions to Physical Modem Management Interface	
Event	**Parameters**
PHY_MEASURE_RSSI	***none***

Signallist: MRCM2PMMI Mascara Radio Control Manager to Physical Modem Management Interface	
Event	**Parameters**
PHY_SELECT_CHANNEL	***ChannelNumber***

PHYSICAL LAYER TO CONTROL (MMCL_PPHY)

Signallist: PMMI2MMEF Physical Modem Management Interface to Mascara Measurement Functions	
Event	**Parameters**
PHY_MEASURED_RSSI	**MeasuredRSSI**

MAC DATA PUMP TO PHYSICAL LAYER (MMDP_PPHY)

Signallist: MSSE2PTXM Mascara Slot Sequencer to Physical Transmit Modem	
Event	**Parameters**
TX_SHORT_req	***none***
TX_LONG_req	***none***
TX_STANDBY	**OnOff**
TX_MPDU	**MPDU**
TX_FH	**FH**

Signallist: MSSE2PRXM Mascara Slot Sequencer to Physical Receive Modem	
Event	**Parameters**
RX_STANDBY	**OnOff**
RX_LONG_HUNT	***none***

PHYSICAL LAYER TO MAC DATA PUMP (MMDP_PPHY)

Signallist: PTXM2MSSE Physical Transmit Modem to Mascara Slot Sequencer	
Event	**Parameters**
TX_CLEAR	***none***

Signallist: PRXM2MSSE Physical Receive Modem to Mascara Slot Sequencer	
Event	**Parameters**
RX_FH	**FH**
RX_MPDU	**MPDU**
RX_ind	***none***

MASCARA INTER-CONTROL COMMUNICATION (ICC)

ICC TO ATM (AATM_MICC)

Signallist: Aatm2MLCP ATM to Layer Control Protocol	
Event	**Parameters**
RECVAAL5_ind	**VPI,** **VCI,** **AAL5_Sdu_ptr,** **SduLen**

Signallist: MLCP2Aatm Layer Control Protocol to ATM	
Event	**Parameters**
OPENAAL5_req	**VPI,** **VCI,** **Pid**
CLOSEAAL5_req	**VPI,** **VCI**
XMITAAL5_ req	**VPI,** **VCI,** **AAL5_Sdu_ptr,** **SduLen**
RELEASEAAL5_ req	**AAL5_Sdu_ptr**

ICC TO CONTROL (MICC_MMCL)

Signallist: MMCN2MADG Control to Mascara AP Dynamic Gen Ctrl
.Ref: Ccon2MAAA, Ccon2MAMA, Ccon2MADG

Signallist: MMCN2MALS Control to Mascara AP Link Status Recorder
.Ref: Ccon2MALS

Signallist: MMCN2MGMC Control to Generic Mascara Control
.Ref: Ccon2MGMC

INTERNAL INTERFACES

Signallist: MMCN2MMTX MEU Controller to MEU TX

Event	**Parameters**
MEU_SEND_DATA_req	**LCP_Data,** **Length,** **LCP_Port**
MEU_DATA_SENT_cnf	**LCP_Port,** **RC**

Signallist: MMTX2MMCN MEU TX to MEU Controller

Event	**Parameters**
MEU_CAN_SEND_cnf	**LCP_Data,** **Length,** **LCP_Port**

AP MASCARA Control

CONTROL TO ICC (MICC_MMCL)

Signallist: MALS2MMCN AP Link Status Recorder to Control
.Ref: MALS2Ccon

Signallist: MAAA2MMCN Mascara AP Association Agent to Control
.Ref: MAAA2Ccon, MAMA2Ccon

Signallist: MAMA2MMCN Mascara AP MVC Agent to Control

.Ref: MAMA2Ccon,

Signallist: MADG2MMCN Mascara AP Dynamic Gen Ctrl to Control

.Ref: MADG2Ccon, MAMA2Ccon,

Signallist: MGMC2MMCN Generic Mascara Control to Control

.Ref: MGMC2Ccon

CONTROL TO SEGMENTING REASSEMBLY (MCSR_MMCL)

Signallist: MAAA2MCSE AP Association Agent to Control Segmentation

.Ref: MAAA2MMAA_MPDU

Signallist: MAMA2MCSE AP MVC Agent to Control Segmentation

.Ref: MAMA2MMMA_MPDU

Signallist: MABA2MCSE AP Beacon Agent to Control Segmentation

.Ref: MABA2MMBA_MPDU

Signallist: MATI2MCSE AP TIP Agent to Control Segmentation

.Ref: MATI2MMTI_MPDU

Signallist: MAAA2MGSR AP Association Agent to Generic Segmentation_Reassembly

Event	Parameters
INSTANCIATE_CSR	**MT_MAC_Addr** **MVC_Id** **AMA_Pid**
DESTROY	**MT_MAC_Addr** **MVC_Id**

CONTROL TO WDLC (MDLC_MMCL)

Signallist: MAMA2MGDL AP MVC Agent to Generic DLC	
Event	**Parameters**
INSTANCIATE_DLC	**MT_MAC_Addr** **MVC_Id** **QoS** **WAND_ATM_Parameters** **ProcessType** **DLR_DestPid** **ConnectionIdentifier**
DESTROY	**MT_MAC_Addr** **MVC_Id**

Signallist: MALS2MGDL AP Link Status Recorder to Generic DLC	
Event	**Parameters**
WDLC_COLLECT_DATA_req	**MT_ATM_Addr** **WDLC_InfoType**

CONTROL TO MAC_DATA_PUMP (MMCL_MMDP)

Signallist: MGMC2MGDP Gen_Masc_Ctrl to Gen_Data_Pump	
Event	**Parameters**
INIT_MDP	**Network_Name** **AP_Name** **AP_MAC_Addr** **Beacon_Period** **AP_MAC_Config_Data**
CLOSE_MDP	***none***

SET_TRAFFIC_PARM	**MT_MAC_Addr** **MVC_Id** **QoS** **WAND_ATM_Parameters**
TRAFFIC_RELEASED	**MT_MAC_Addr** **MVC_Id**

Signallist: MAMA2MGDP AP_MVC_Agent to Gen_Data_Pump

Event	**Parameters**
SET_TRAFFIC_PARM	**MT_MAC_Addr** **MVC_Id** **QoS** **WAND_ATM_Parameters**
TRAFFIC_RELEASED	**MT_MAC_Addr** **MVC_Id**

Signallist: MAIA2MASC AP_I_Am_Alive Agent to AP Scheduler

Event	**Parameters**
SEND_IAA_INVITATION	**MT_MAC_Addr**

Signallist: MALS2MASC AP Link Status Recorder to Master Scheduler

Event	**Parameters**
ASC_COLLECT_DATA_req	**MT_MAC_Addr** **ASC_InfoType**

Signallist: MALS2MMPR AP Link Status Recorder to MPDU Receiver

Event	**Parameters**
MPR_COLLECT_DATA_req	**none**

Signallist: MATI2MASC AP TIP Agent to Master Scheduler

Event	**Parameters**
TIP_MT_ind	**MT_MAC_Addr** **RC**

INTERNAL INTERFACES

Signallist: MGMC2MADG Generic Mascara Control to AP_Dynamic_Gen_Ctrl

Event	**Parameters**
AP_DEASSOCIATE	***none***
INIT_ADC	**Pid** **AP_MAC_Addr**

Signallist: MADG2MGMC AP_Dynamic Gen Ctrl to Generic Mascara Control

Event	**Parameters**
AP_DEASSOCIATE_DONE	***none***

Signallist: MGMC2MASG Generic Mascara Control to AP_Steady_State_Gen_Ctrl

Event	**Parameters**
INIT_ASS	**NetworkName** **AP_Name** **AP_MAC_Addr** **OperationFrequency** **Min_MT** **Max_MT** **BeaconPeriod** **MAC_ConfigData** **NeighAP_List**
CLOSE_ASS	**none**

Signallist: MGMC2MRCM Generic Mascara Control to Radio Control Manager

Event	**Parameters**
INIT_RCL	***OperationFrequency***

CLOSE_RCL	**none**

Signallist: MRCM2MGMC Radio Control Manager to Generic Mascara Control	
Event	**Parameters**
INIT_RCL_OK	***none***
INIT_RCL_KO	**RC**
CLOSE_RCL_DONE	**none**

Signallist: MAIA2MADG AP I_Am_Alive Agent to AP Dynamic Gen Ctrl	
Event	**Parameters**
MT_LOST	**MT_ATM_Addr**

Signallist: MADG2MAIA AP Dynamic Gen Ctrl to AP I_Am_Alive Agent	
Event	**Parameters**
MT_ASSOCIATED	**MT_ATM_Addr** **MT_MAC_Addr**
MT_DEASSOCIATED	**MT_MAC_Addr**

AP_Dynamic_Ctrl internal interfaces

Signallist: MADG2MAAA AP_Dynamic_Gen_Ctrl to AP Association Agent	
Event	**Parameters**
KILL	**none**
MT_AP_ASSOCIATION	**NetworkName** **MT_ATM_Addr** **MT_MAC_Addr** **MT_Name** **AP_MAC_Addr** **ConfigData** **HO**
.Ref: Ccon2MAAA	
.Ref: Ccon2MAMA	

Signallist: MAAA2MADG AP Association Agent to AP_Dynamic_gen_Ctrl	
Event	**Parameters**

MT_DEASSOCIATED	**MT_MAC_Addr**

Signallist: MAAA2MAMA AP Association Agent to AP MVC_Agent	
Event	**Parameters**
KILL	**none**
.Ref: MMMA2MAMA_MPDU	
.Ref: Ccon2MAMA	

Signallist: MAMA2MAAA AP_MVC_Agent to AP Association Agent	
Event	**Parameters**
VC_CLEARED	**ConnectionIdentifier**

AP_Steady_State_Ctrl internal interfaces

Signallist: MASG2MAIA AP Steady_State_Gen_Ctrl to AP I_Am_Alive Agent	
Event	**Parameters**
INIT_AIA	**none**
CLOSE_AIA	**none**

Signallist: MASG2MABA AP Steady_State_Gen_Ctrl to AP Beacon Agent	
Event	**Parameters**
INIT_ABA	**NetworkName** **AP_Name** **AP_MAC_Addr** **OperationFrequency** **NeighAP_List** **BeaconPeriod**
CLOSE_ABA	**none**

Signallist: MASG2MATI AP Steady_State_Gen_Ctrl to AP TIP Agent	
Event	**Parameters**
INIT_ATI	**none**
CLOSE_ATI	**none**

Signallist: MASG2MALS AP Steady_State_Gen_Ctrl to AP Link Status Recorder	
Event	**Parameters**
INIT_ALS	**NetworkName** **AP_Name** **AP_MAC_Addr** **OperationFrequency** **Min_MT** **Max_MT** **BeaconPeriod** **MAC_ConfigData**
CLOSE_ALS	**none**

Signallist: MATI2MAIA AP TIP Agent to AP I_Am_Alive Agent	
Event	**Parameters**
MT_WAKE_UP	**MT_MAC_Addr**
MT_SLEEP	**MT_MAC_Addr**

Radio_Control internal interfaces

Signallist: MRCM2MMEF Radio Control Manager to Measurement Functions	
Event	**Parameters**
INIT_MEF	**none**
CLOSE_MEF	**none**

Signallist: MRCM2MEPC Radio Control Manager to Measurement Functions	
Event	**Parameters**
INIT_EPC	**none**
CLOSE_EPC	**none**

MASCARA SEGMENTING REASSEMBLY

SEGMENTING REASSEMBLY TO MASCARA CONTROL (MCSR_MMCL)

Signallist: MCRE2MAAA Control Reassembly to AP Association Agent	
Event	**Parameters**
.Ref: MMAA2MAAA_MPDU	
.Ref: MMMA2MAMA_MPDU	

Signallist: MCRE2MATI Control Reassembly to AP TIP Agent	
Event	**Parameters**
.Ref: MMTI2MATI_MPDU	

Signallist: MGSR2MAAA Generic Segmenting Reassembly to Association Agent	
Event	**Parameters**
PID_REGISTERED	**MT_MAC_Addr** **MVC_Id** **ProcessType** **Pid**

SEGMENTING REASSEMBLY TO WIRELESS DLC (MCSR_MDLC)

Signallist: MGSR2MGDL Generic Segmenting Reassembly to Generic DLC	
Event	**Parameters**
INSTANCIATE_DLC	**MT_MAC_Addr** **MVC_Id** **QoS** **WAND_ATM_Parameters** **ProcessType** **DLR_DestPid** **ConnectionIdentifier**
DESTROY	**MT_ MAC_Addr** **MVC_Id**

Signallist: MCSE2MDLX Ctrl Segmentation to WDLC Xmit	
Event	**Parameters**
XMIT_CTRL_CELL	**ATM_Cell_ptr** **MVC_Id (X'00' for control)** **DA_MAC_Addr (meaningful in AP only, it identifies the target MT)** **PayloadsToSend**

SEGMENTING REASSEMBLY TO ATM LAYER (AATM_MCSR)

Signallist: MCSE2Aatm Ctrl Segmentation to ATM Layer	
Event	**Parameters**
ALLOCATEAAL0_req	**none**

Signallist: Aatm2MCSEATM Layer to Ctrl Segmentation	
Event	**Parameters**
ALLOCATEAAL0_cnf	**ATM_Cell_ptr**

INTERNAL INTERFACE

Signallist: MGSR2MCRE Generic_Seg_Reasm to Control_Reassembly	
Event	**Parameters**
PID_REGISTERED	**MT_MAC_Addr** **MVC_Id** **ProcessType** **Pid**
KILL	***None***

Signallist: MGSR2MCSE Generic_Seg_Reasm to Control_Segmenting	
Event	**Parameters**
PID_REGISTERED	**MT_MAC_Addr** **MVC_Id** **ProcessType** **Pid**
KILL	***None***

MASCARA WDLC

WDLC TO ATM LAYER (AATM_MDLC)

Signallist: Aatm2MDLX ATM Layer to WDLC Xmit	
Event	**Parameters**
ALLOCATEAAL0_cnf	**ATM_Cell_ptr**
RECVAAL0_ind	**ATM_Cell_ptr**

Signallist: MDLX2Aatm WDLC Xmit to ATM Layer	
Event	**Parameters**
ALLOCATEAAL0_req	
RELEASEAAL0_req	**WDLC_Cell_ptr**

Signallist: MDLR2Aatm WDLC Receive to ATM Layer	
Event	**Parameters**
XMITAAL0_req	**ATM_Cell_ptr**

Signallist: MGDL2Aatm Generic_WDLC to ATM Layer	
Event	**Parameters**
OPENAAL0_req	**VPI** **VCI** **Pid**
CLOSEAAL0_req	**VPI** **VCI**

WDLC TO MASCARA DATA PUMP (MDLC_MMDP)

Signallist: MDLX2MTRE WDLC Xmit to Traffic Recorder	
Event	**Parameters**

TRAFFIC_UPDATED	**MT_MAC_Addr (meaningful in AP only: it identifies the target MT)** **MVC_id** **CelPtrList** **NumOfRequests** **SuccNumXmit** **SuccNumRcv** **TrafficStatus**

Signallist: MDLX2MASC WDLC Xmit to AP Scheduler	
Event	**Parameters**
GEN_WDLC_FB	**MT_MAC_Addr** **MVC_Id** **RN_Xmit**

Signallist: MGDL2MMPR Generic_WDLC to MPDU_Handler_Rcv	
Event	**Parameters**
PID_REGISTERED	**MT_MAC_Addr** **MVC_Id** **ProcessType** **Pid**
PID_DEREGISTERED	**MT_MAC_Addr** **MVC_Id** **DLR_Pid**

WDLC TO CONTROL SEGMENTING REASSEMBLY (MCSR_MDLC)

Signallist: MGDL2MGSR Generic DLC to Generic Segmenting Reassembly	
Event	**Parameters**
PID_REGISTERED	**MT_MAC_Addr** **MVC_Id** **ProcessType** **Pid**

Signallist: MDLR2MCRE WDLC Receive to Control reassembly	
Event	**Parameters**

CTRL_CELL_RCVD	**ATM_Cell_ptr** **SA_MAC_Addr (meaningful in AP only, it identifies the target MT)**

WDLC TO MASCARA CONTROL (MDLC_MMCL)

Signallist: MGDL2MAMA Generic DLC to MVC Agent	
Event	**Parameters**
PID_REGISTERED	**MT_MAC_Addr** **MVC_Id** **ProcessType** **Pid**

Signallist: MGDL2MALS Generic DLC to Link Status Recorder	
Event	**Parameters**
WDLC_COLLECT_DATA_cnf	**MT_MAC_Addr** **WDLC_InfoType** **WDLC_InfoData** **TimeElapsed**
WDLC_ALARM_ind	**AlarmId** **AlarmInfo**

INTERNAL INTERFACES

Signallist: MDLR2MDLX WDLC Receive to WDLC Xmit	
Event	**Parameters**
MDLR2MDLX_Signal	***none***
ACKS_REQS_RDY	**Integer** **Integer** **Integer** **Integer** **Integer** **Integer**

Signallist: MGDL2MDLR Generic_DLC to WDLC_Rcv	
Event	**Parameters**

PID_REGISTERED	**MT_MAC_Addr** **MVC_Id** **ProcessType** **Pid**
KILL	**None**
SERVICE_PARAMETER	**QOS** **VPI** **VCI**

Signallist: MGDL2MDLX Generic_DLC to WDLC_Xmit	
Event	**Parameters**
PID_REGISTERED	**MT_MAC_Addr** **MVC_Id** **ProcessType** **Pid**
KILL	**None**
SERVICE_PARAMETER	**QOS** **VPI** **VCI**
SCHED_CONNDATA_DLX	**TT_ServicedEntry**
ALS_DLX_req	**none**

Signallist: MDLX2MGDL WDLC_Xmit to Generic_DLC	
Event	**Parameters**
ALS_DLX_cnf	**DLX_WDLC_Infodata**

MASCARA DATA PUMP

MASCARA DATA PUMP TO MASCARA CONTROL (MMCL_MMDP)

Signallist: MGDP2MGMC Gen_Data_Pump to Gen_Masc_Ctrl	
Event	**Parameters**
INIT_MDP_DONE	***none***
CLOSE_MDP_DONE	***none***

Signallist: MMPR2MAIA Masc MPDU Handler Receive to Masc Ctrl AP I Am Alive Agent	
Event	**Parameters**
MT_ALIVE	**MT_MAC_Addr**

Signallist: MMPR2MADG Masc MPDU Handler Receive to AP_Dynamic_Gen_Ctrl	
Event	**Parameters**
.Ref: MMAA2MAAA_INITIAL_MPDU	

Signallist: MMPR2MALS MPDU Receiver to AP Link Status Recorder	
Event	**Parameters**
MPR_COLLECT_DATA_cnf	**MPR_Infodata** **Time_Elapsed** **Res_Req**
MPR_ALARM_ind	**AlarmId** **AlarmInfo**

Signallist: MASC2MALS Master Sceduler to AP Link Status Recorder	
Event	**Parameters**
ASC_COLLECT_DATA_cnf	**MT_ATM_Addr** **ASC_InfoType** **ASC_InfoData** **TimeElapsed**
ASC_ALARM_ind	**AlarmId** **AlarmInfo**

MASCARA DATA PUMP TO WDLC (MDLC_MMDP)

Signallist: MMPR2MDLR MPDU handler receive to WDLC Receive	
Event	**Parameters**
MPDU_BODY_RCVD	**TimeStamp** **MPDU_PayloadPtr**

Signallist: MTRE2MGDL Traffic Recorder to Generic WDLC

Event	Parameters
SCHED_CONNDATA	LengthIndx TT_Serviced_Entries Number_of_Connections Expired_Cells

INTERNAL INTERFACES

Signallist: MASC2MMPX AP Scheduler to MPDU Handler Xmit	
Event	**Parameters**
FH_PAYLOAD	MPDU_InfoList

Signallist: MSSE2MASC Slot Sequencer to AP Scheduler	
Event	**Parameters**
FRAME_START	*none*

Signallist: MMPR2MTRE MPDU Handler Receive to Traffic Recorder	
Event	**Parameters**
RES_REQ_RCVD	MVC_Info SA_MAC_Addr

Signallist: MGDP2MTRE Gen_MAC_Data_Pump to Traffic Recorder	
Event	**Parameters**
INIT_TRE	none
SET_NEW_CONN	MT_MAC_Addr MVC_Id ARQ_Type
KILL_CONN	MT_MAC_Addr MVC_Id

Signallist: MGDP2MMPR Gen_MAC_Data_Pump to MPDU_Handler_Rcv	
Event	**Parameters**

INIT_MPR	**Net_Id** **MAC_Addr** **Mbuf_ptr**

Signallist: MGDP2MMPX Gen_MAC_Data_Pump to MPDU_Handler_Xmit	
Event	**Parameters**
INIT_MPX	**NetID** **MAC_Addr** **Mbuf_ptr**

Signallist: MGDP2MASC Gen_MAC_Data_Pump to AP_Scheduler	
Event	**Parameters**
INIT_ASC	**NetworkName** **AP_Name** **AP_MACAddr** **BeaconPeriod** **AP_MAC_ConfigData** **Mbuf_ptr**
CLOSE_ASC	***none***
SET_TRAFFIC_PARM	**MT_MAC_Addr** **MVC_Id** **QoS** **WAND_ATM_Parameters**
TRAFFIC_RELEASED	**MT_MAC_Addr** **MVC_Id**

Signallist: MASC2MGDP AP_Scheduler to Gen_MAC_Data_Pump	
Event	**Parameters**
CLOSE_ASC_DONE	***none***

MPDU_Handler internal interfaces

Signallist: MMPX2MSSE MPDU Handler Xmit to Slot Sequencer	
Event	**Parameters**

TED	**TED_Type** **BufferAddr** **BufferLgth** **SlotCount**

Signallist: MSSE2MMPR Slot Sequencer MPDU Receive	
Event	**Parameters**
MPDU_RCVD	**Slot** **MPDU_Ptr**

Scheduler internal interfaces

Signallist: MASC2MTRE AP Scheduler to Traffic Recorder	
Event	**Parameters**
UPDATE_TRE	**LengthIndex** **TT_ServiceEntries** **Integer**
GET_TABLE	***none***

Signallist: MTRE2MASC Traffic Recorder to AP Scheduler	
Event	**Parameters**
SET_TABLE	**TrafficTable**

12.3.

MT MASCARA

PEER TO PEER BETWEEN MASCARA CONTROL AND SIGNALLING

PEER TO PEER MASCARA CONTROL TO SIGNALLING

Signallist: MMAA2Ccon Mascara MT Association Agent to Control	
Event	**Parameters**
ASSOCIATION_cnf	**MT_ATM_Addr** **RC**
MT_DEASSOCIATION_ind	***none***
VC_REOPEN_cnf	**ConnectionIdentifier** **RC**
VC_OPEN_cnf	**ConnectionIdentifier** **RC**
MT_VC_RELEASE_cnf	**ConnectionIdentifier** **MT_ATM_Addr** **RC**
MT_DEASSOCIATION_cnf	**RC**

Signallist: MMHI2Ccon Mascara MT Handover Initiator Agent to Control	
Event	**Parameters**
HO_ind	**AP_MAC_Addr** **MT_RadioLinkQuality**
MT_ASSOCIATION_LOST_ind	***none***

Signallist: MGMC2Ccon General Mascara Control to Control	
Event	**Parameters**
CLOSE_MAC_cnf	**RC**
OPEN_MT_MAC_cnf	**AP_List *(pairs of (AP_Name, AP_MAC_Addr))*** **RC**

Signallist: MMMA2Ccon Mascara MT MVC Agent to Control	
Event	**Parameters**

VC_OPEN_cnf	**ConnectionIdentifier** **RC**

PEER TO PEER SIGNALLING TO MASCARA CONTROL

Signallist: Ccon2MMAA Control to Mascara MT Association Agent	
Event	**Parameters**
VC_REOPEN_ req	**ConnectionIdentifier**
ASSOCIATION_req	**AP_MAC_Addr**
MT_DEASSOCIATION_req	**MT_ATM_Addr** **HO**

Signallist: Ccon2MGMC Control to Generic Mascara Control	
Event	**Parameters**
CLOSE_MAC_req	***none***
OPEN_MT_MAC_req	**NetworkName** **MT_Name** **MT_ATM_Addr** **MAC_ConfigData**

Signallist: Ccon2MMMA Control to Mascara MT MVC Agent	
Event	**Parameters**
MT_VC_RELEASE_req	**ConnectionIdentifier** **MT_ATM_Addr**
VC_OPEN_req	**ConnectionIdentifier** **QoS** **WAND_ATM_Parameters**

MASCARA - PHYSICAL INTERFACE (MMAC_PPHY)

Signallist: Mmac2Pphy Mascara to Physical
.Ref: MEPC2PMMI, MMEF2PMMI, MRCM2PMMI, MSSE2PTXM, MSSE2PRXM

Signallist: Pphy2Mmac Physical to Mascara
.Ref: PMMI2MMEF, PTXM2MSSE, PRXM2MSSE

CONTROL TO PHYSICAL LAYER (MMCL_PPHY)

Signallist: MEPC2Pphy Mascara Emitted Power Control to Physical Modem Management Interface	
Event	**Parameters**
MEPC2Pphy_signal	***TBD (Ignored for the Demo)***

Signallist: MMEF2PMMI Mascara Measurement Functions to Physical Modem Management Interface	
Event	**Parameters**
PHY_MEASURE_RSSI	***none***

Signallist: MRCM2PMMI Mascara Radio Control Manager to Physical Modem Management Interface	
Event	**Parameters**
PHY_SELECT_CHANNEL	***ChannelNumber***

PHYSICAL LAYER TO CONTROL (MMCL_PPHY)

Signallist: PMMI2MMEF Physical Modem Management Interface to Mascara Measurement Functions	
Event	**Parameters**
PHY_MEASURED_RSSI	**MeasuredRSSI**

MAC DATA PUMP TO PHYSICAL LAYER (MMDP_PPHY)

Signallist: MSSE2PTXM Mascara Slot Sequencer to Physical Transmit Modem	
Event	**Parameters**
TX_SHORT_req	***none***
TX_LONG_req	***none***
TX_STANDBY	**OnOff**
TX_MPDU	**MPDU**
TX_FH	**FH**

Signallist: MSSE2PRXM Mascara Slot Sequencer to Physical Receive Modem	
Event	**Parameters**

RX_STANDBY	**OnOff**
RX_LONG_HUNT	***none***

PHYSICAL LAYER TO MAC DATA PUMP (MMDP_PPHY)

Signallist: PTXM2MSSE Physical Transmit Modem to Mascara Slot Sequencer	
Event	**Parameters**
TX_CLEAR	***none***

Signallist: PRXM2MSSE Physical Receive Modem to Mascara Slot Sequencer	
Event	**Parameters**
RX_FH	**FH**
RX_MPDU	**MPDU**
RX_ind	***none***

MASCARA INTER_CONTROL COMMUNICATION

ICC TO ATM (AATM_MICC)

Signallist: Aatm2MLCP ATM to Layer Control Protocol	
Event	**Parameters**
RECVAAL5_ind	**VPI,** **VCI,** **AAL5_Sdu_ptr,** **SduLen**

Signallist: MLCP2Aatm Layer Control Protocol to ATM	
Event	**Parameters**
OPENAAL5_req	**VPI,** **VCI,** **Pid**
CLOSEAAL5_req	**VPI,** **VCI**

XMITAAL5_ req	**VPI,** **VCI,** **AAL5_Sdu_ptr,** **SduLen**
RELEASEAAL5_ req	**AAL5_Sdu_ptr**

ICC TO CONTROL (MICC_MMCL)

Signallist: MMCN2MMAA Control to Mascara MT Association Agent **.Ref: Ccon2MMAA, Ccon2MMMA**

Signallist: MMCN2MGMC Control to Generic Mascara Control **.Ref: Ccon2MGMC**

MT MASCARA CONTROL

CONTROL TO ICC (MICC_MMCL)

Signallist: MMAA2MMCN Mascara MT Association Agent to Control **.Ref: MMAA2Ccon**

Signallist: MMMA2MMCN Mascara MT MVC Agent to Control **.Ref: MMMA2Ccon**

Signallist: MMHI2MMCN Mascara MT Handover Initiator Agent to Control **.Ref: MMHI2Ccon**

Signallist: MGMC2MMCN Generic Mascara Control to Control **.Ref: MGMC2Ccon**

CONTROL TO SEGMENTING REASSEMBLY (MCSR_MMCL)

Signallist: MMAA2MCSE MT Association Agent to Control Segmentation **.Ref: MMAA2MAAA_MPDU, MMAA2MAAA_INITIAL_MPDU**

Signallist: MMMA2MCSE MT MVC Agent to Control Segmentation **.Ref: MMMA2MAMA_MPDU**

Signallist: MMTI2MCSE MT TIP Agent to Control Segmentation
.Ref: MMTI2MATI_MPDU

CONTROL TO WDLC (MDLC_MMCL)

Signallist: MMMA2MGDL MT MVC Agent to Generic DLC	
Event	**Parameters**
INSTANCIATE_DLC	**MT_MAC_Addr** **MVC_Id** **QoS** **WAND_ATM_Parameters** **ProcessType** **DLR_DestPid** **ConnectionIdentifier**
DESTROY	**MT_MAC_Addr** **MVC_Id**

Signallist: MMHI2MGDL MT Handover Initiation to Generic WDLC	
Event	**Parameters**
CELL_ERRORS_req	**none**

CONTROL TO MAC_DATA_PUMP (MMCL_MMDP)

Signallist: MGMC2MGDP Gen_Masc_Ctrl to Gen_Data_Pump	
Event	**Parameters**
INIT_MDP	**NetworkName**
CLOSE_MDP	***none***

Signallist: MMMA2MGDP MT_MVC_Agent to Gen_Data_Pump	
Event	**Parameters**
SET_TRAFFIC_PARM	**MT_MAC_Addr** **MVC_Id** **QoS** **WAND_ATM_Parameters**

TRAFFIC_RELEASED	**MT_MAC_Addr** **MVC_Id**

Signallist: MMAA2MGDP MT_Association_Agent to Gen_Data_Pump	
Event	**Parameters**
SET_MAC_ADDR	**MT_MAC_Addr**

Signallist: MMTC2MMPR MT Target Cell to MPDU Handler Receive	
Event	**Parameters**
NEIGH_AP_LOAD_INFO_req	***none***

Signallist: MMHI2MMPR MT Handover Indication to MPDU Handler Receive	
Event	**Parameters**
MPDU_ERRORS_req	***none***

INTERNAL INTERFACES

Signallist: MGMC2MMDG Generic Mascara Control to MT_Dynamic_Gen_Ctrl	
Event	**Parameters**
MT_DEASSOCIATE	**None**
INIT_MDC	**NetworkName** **MT_Name** **MT_ATM_Addr** **ConfigData**

Signallist: MMDG2MGMC MT_Dynamic Gen Ctrl to Generic Mascara Control	
Event	**Parameters**
MT_DEASSOCIATION_DONE	***none***

Signallist: MGMC2MMSG Generic Mascara Control to MT_Steady_State_Gen_Ctrl	
Event	**Parameters**

INIT_MSS	**ChannelNumber** **AP_List**
CLOSE_MSS	**none**

Signallist: MGMC2MRCM Generic Mascara Control to Radio Control Manager

Event	**Parameters**
INIT_RCL	**none**
CLOSE_RCL	**none**

Signallist: MRCM2MGMC Radio Control Manager to Generic Mascara Control

Event	**Parameters**
INIT_RCL_OK	**AP_List** **ChannelNumber**
INIT_RCL_KO	**RC**
CLOSE_RCL_DONE	**none**

Signallist: MMPS2MRCM MT_Power_Saving to Radio Control Manager

Event	**Parameters**
RADIO_ENTER_PSM	**none**
RADIO_LEAVE_PSM	**none**

Signallist: MMTC2MRCM MT Target Cell to MT Radio Control Manager

Event	**Parameters**
ACQUIRE_NEW_AP	**Channelnumber**

Signallist: MRCM2MMTC MT Radio Control Manager to MT Target Cell

Event	**Parameters**
ACQUIRE_NEW_AP_OK	***None***
ACQUIRE_NEW_AP_KO	***None***

Signallist: MMTC2MMEF MT Target Cell to MT Measurements Functions

Event	**Parameters**

MEASURE_RLQ_req	**Integer**

Signallist: MMEF2MMTC MT Measurements Functions to MT Target Cell	
Event	**Parameters**
MEASURE_RLQ_cnf	***MT_RadioLinkQualityType***

Signallist: MMEF2MMHI MT Measurements Functions to MT Handover Indication	
Event	**Parameters**
MEASURE_RLQ_cnf	***MT_RadioLinkQualityType***

Signallist: MMHI2MMEF MT Handover Indication to MT Measurements Functions	
Event	**Parameters**
MEASURE_RLQ_req	**Integer**

Signallist: MMTC2MMHI MT Target Cell to MT Handover Indicator	
Event	**Parameters**
TARGET_AP_FOUND	***AP_MAC_AddrType,*** ***MT_RadioLinkQualityType,***
AP_LOST	***None***
SUSPEND_MHI	***None***
RESUME_MHI	***None***

Signallist: MMHI2MMTC MT Handover Indicator to MT Target Cell	
Event	**Parameters**
GET_TARGET_AP	**HO**

MT_Dynamic_Ctrl internal interfaces

Signallist: MMAA2MMMA MT Association Agent to MT_MVC_Agent	
Event	**Parameters**
KILL	**none**
.Ref: MAMA2MMMA_MPDU	

.Ref: Ccon2MMMA	

Signallist: MMMA2MMAA MT_MVC_Agent to MT_Association Agent	
Event	**Parameters**
VC_CLEARED	**ConnectionIdentifier**

Signallist: MMDG2MMAA MT Dynamic Gen Ctrl to MT_Association Agent	
Event	**Parameters**
MT_DEASSOCIATION_req	**MT_ATM_Addr**
INIT_MAA	**NetworkName** **MT_Name** **MT_ATM_Addr** **ConfigData**

Signallist: MMAA2MMDG MT_Association Agent to MT Dynamic Gen Ctrl	
Event	**Parameters**
MT_DEASSOCIATION_cnf	**RC**

Signallist: MMDG2MMHI to MT Dynamic Gen Ctrl to MT Handover Initiation	
Event	**Parameters**
INIT_MHI	**NetworkName** **ConfigData**

MT_Steady_State_Ctrl internal interfaces

Signallist: MMTC2MMTI MT Target Cell to MT TIP Agent	
Event	**Parameters**
INIT_HO_SEEK_INFO	**MT_MAC_Addr**
STOP_HO_SEEK_INFO	***none***

Signallist: MMTI2MMTC MT TIP Agent to MT Target Cell	
Event	**Parameters**

INIT_HO_SEEK_INFO_OK	***None***
INIT_HO_SEEK_INFO_KO	***None***

Signallist: MMBA2MMTC MT Beacon Agent to MT Target Cell	
Event	**Parameters**
SEND_NEIGH_AP_INFO	***NeighAP_List***

Signallist: MMSG2MMBA MT_Steady_Generic_Ctrl to MT Beacon Agent	
Event	**Parameters**
INIT_MBA	***none***
CLOSE_MBA	***none***

Signallist: MMSG2MMTC MT_Steady_Generic_Ctrl to MT Target Cell	
Event	**Parameters**
INIT_MTC	***none***
CLOSE_MTC	***none***

Signallist: MMSG2MMTI MT_Steady_Generic_Ctrl to MT TIP Agent	
Event	**Parameters**
INIT_TIP	***none***
CLOSE_TIP	***none***

Signallist: MMSG2MMPS MT_Steady_Generic_Ctrl to MT Power Saving	
Event	**Parameters**
INIT_MPS	***none***
CLOSE_MPS	***none***

Radio_Control internal interfaces

Signallist: MRCM2MMEF Radio Control Manager to Measurement Functions	
Event	**Parameters**
INIT_MEF	**none**
CLOSE_MEF	**none**

Signallist: MRCM2MEPC Radio Control Manager to Measurement Functions	
Event	**Parameters**
INIT_EPC	**none**
CLOSE_EPC	**none**

MASCARA SEGMENTING REASSEMBLY

SEGMENTING REASSEMBLY TO ATM LAYER (AATM_MCSR))

Signallist: MCSE2Aatm Ctrl Segmentation to ATM Layer	
Event	**Parameters**
ALLOCATEAAL0_req	***none***

Signallist: Aatm2MCSE ATM Layer to Ctrl Segmentation	
Event	**Parameters**
ALLOCATEAAL0_cnf	**ATM_Cell_ptr**

SEGMENTING REASSEMBLY TO MASCARA CONTROL (MCSR_MMCL)

Signallist: MCRE2MMAA   Control Reassembly to MT Association Agent
.Ref: MAAA2MMAA_MPDU, MAMA2MMMA_MPDU

Signallist: MCRE2MMTI   Control Reassembly to MT TIP Agent
.Ref: MATI2MMMTI_MPDU,

Signallist: MCRE2MMBA   Control Reassembly to MT Beacon Agent
.Ref: MABA2MMMBA_MPDU,

SEGMENTING REASSEMBLY TO WIRELESS DLC (MCSR_MDLC)

Signallist: MCSE2MDLX Ctrl Segmentation to WDLC Xmit	
Event	**Parameters**

XMIT_CTRL_CELL	**ATM_Cell_ptr** **MVC_Id (X'00' for control)** **DA_MAC_Addr (meaningful in AP only, it identifies the target MT)**

Signallist: MCRE2MGDL Ctrl Reassembly to Generic WDLC	
Event	**Parameters**
INSTANCIATE_DLC	**MT_MAC_Addr** **MVC_Id** **QoS** **WAND_ATM_Parameters** **ProcessType** **DLR_DestPid** **ConnectionIdentifier**

INTERNAL INTERFACES

Signallist: MCRE2MCSE Ctrl Reassembly to Ctrl Segmentation	
Event	**Parameters**
PID_REGISTERED	**MT_MAC_Addr** **MVC_Id** **ProcessType** **Pid**

MASCARA WDLC

WDLC TO ATM LAYER (AATM_MDLC)

Signallist: Aatm2MDLX ATM Layer to WDLC Xmit	
Event	**Parameters**
RECVAAL0_ind	**ATM_Cell_ptr**
ALLOCATEAAL0_cnf	**ATM_Cell_ptr**

Signallist: MDLX2Aatm WDLC Xmit to ATM Layer	
Event	**Parameters**
RELEASEAAL0_req	**WDLC_Cell_ptr**
ALLOCATEAAL0_req	***None***

Signallist: MGDL2Aatm Generic WDLC to ATM Layer	
Event	**Parameters**
OPENAAL0_req	**VPI** **VCI** **Pid**
CLOSEAAL0_req	**VPI** **VCI**

Signallist: MDLR2Aatm WDLC Receive to ATM Layer	
Event	**Parameters**
XMITAAL0_req	**ATM_Cell_ptr**

WDLC TO MASCARA DATA PUMP (MDLC_MMDP)

Signallist: MDLX2MTRE WDLC Xmit to Traffic recorder	
Event	**Parameters**
TRAFFIC_UPDATED	**MT_MAC_Addr** **MVC_Id** **CellPtrList** **SuccNumXmit**

Signallist: MDLX2MMSC WDLC Xmit to MT Scheduler	
Event	**Parameters**
GEN_WDLC_FB	**MT_MAC_Addr** **MVC_Id** **RN_Xmit**

Signallist: MGDL2MMPR Generic_WDLC to MPDU_Handler_Rcv	
Event	**Parameters**
PID_REGISTERED	**MT_MAC_Addr** **MVC_Id** **ProcessType** **Pid**

PID_DEREGISTERED	**MT_MAC_Addr** **MVC_Id** **DLR_Pid**

WDLC TO CONTROL SEGMENTING REASSEMBLY (MCSR_MDLC)

Signallist: MGDL2MCRE Generic WDLC to Ctrl Reassembly	
Event	**Parameters**
PID_REGISTERED	**MT_MAC_Addr** **MVC_Id** **ProcessType** **Pid**

Signallist: MDLR2MCRE WDLC Receive to Control reassembly	
Event	**Parameters**
CTRL_CELL_RCVD	**ATM_Cell_ptr** **SA_MAC_Addr (meaningful in AP only, it identifies the target MT)**

WDLC TO MASCARA CONTROL (MDLC_MMCL)

Signallist: MGDL2MMMA Generic DLC to MVC Agent	
Event	**Parameters**
PID_REGISTERED	**MT_MAC_Addr** **MVC_Id** **ProcessType** **Pid**

Signallist: MMHI 2MGDL MT Handover Indicator to Generic WDLC	
Event	**Parameters**
CELL_ERRORS_req	***None***

Signallist: MGDL2MMHI Generic WDLC to MT Handover Indicator	
Event	**Parameters**
CELL_ERRORS_cnf	**Integer**

INTERNAL INTERFACES

Signallist: MDLR2MDLX WDLC Receive to WDLC Xmit	
Event	**Parameters**
MDLR2MDLX_SIGNAL	***none***
ACKS_REQS_RDY	**Integer** **Integer** **Integer**

Signallist: MGDL2MDLR Generic_DLC to WDLC_Rcv	
Event	**Parameters**
PID_REGISTERED	**MT_MAC_Addr** **MVC_Id** **ProcessType** **Pid**
FB_RCVD	**Integer**
KILL	***none***
SERVICE_PARAMETER	**QoS** **VPI** **VCI**

Signallist: MGDL2MDLX Generic_DLC to WDLC_Xmit	
Event	**Parameters**
PID_REGISTERED	**MT_MAC_Addr** **MVC_Id** **ProcessType** **Pid**
KILL	***none***
SERVICE_PARAMETER	**QoS** **VPI** **VCI**

MASCARA DATA PUMP

MASCARA DATA PUMP TO MASCARA CONTROL (MMCL_MMDP)

Signallist: MGDP2MGMC Gen_Data_Pump to Gen_Masc_Ctrl	
Event	**Parameters**

INIT_MDP_DONE	***none***
CLOSE_MDP_DONE	***none***

Signallist: MGDP2MMAA Gen_Data_Pump to MT_Association_Agent	
Event	**Parameters**
SET_MAC_ADDR_ACK	***none***

Signallist: MMSC2MMPS MT Scheduler to MT Power Saving	
Event	**Parameters**
POWER_OFF	***Duration***

Signallist: MMPR2MMHI MT MPDU_Receive to MT Handover Indicator	
Event	**Parameters**
MPDU_ERRORS_cnf	***Integer***

Signallist: MMPR2MMTC MT MPDU_Receive to MT Target Cell	
Event	**Parameters**
NEIGH_AP_LOAD_INFO_cnf	***GuaranteedAggregateBwUsedType***

MASCARA DATA PUMP TO WDLC (MDLC_MMDP)

Signallist: MMPR2MDLR MPDU handler receive to WDLC Receive	
Event	**Parameters**
MPDU_BODY_RCVD	**TimeStamp** **MPDU_PayloadPtr** **MPDU_PayloadLgth** **SA_MAC_Addr**

INTERNAL INTERFACES

Signallist: MMSC2MMPX MT Scheduler to MPDU Handler Xmit	
Event	**Parameters**

MT_FH_READY	**MBuf_ptr** **MPDU_InfoList** **ResReqInfo**

Signallist: MMPX2MMSC MPDU Handler Xmit to MT Scheduler	
Event	**Parameters**
MBUF	**MBuf**

Signallist: MMPR2MMSC MPDU Handler Receive to MT Scheduler	
Event	**Parameters**
FH_RCVD	**FH_PayloadPtr** **FH_PayloadLgth**

Signallist: MGDP2MTRE Gen_MAC_Data_Pump to Traffic Recorder	
Event	**Parameters**
SET_NEW_CONN	**MT_MAC_Addr** **MVC_Id** **ARQ_Type**
KILL_CONN	**MT_MAC_Addr** **MVC_Id**
INIT_TRE	***TBD***

Signallist: MGDP2MMSC Gen_MAC_Data_Pump to Slave Scheduler	
Event	**Parameters**
INIT_MSC	***TBD***
CLOSE_MSC	***none***
RESUME_MSC	***none***
SUSPEND_MSC	**RC**
SET_MAC_ADDR	**MT_MAC_Addr**

SET_TRAFFIC_PARM	**MT_MAC_Addr** **MVC_Id** **QoS** **WAND_ATM_Parameters**
TRAFFIC_RELEASED	**MT_MAC_Addr** **MVC_Id**

Signallist: MGDP2MMPR Gen_MAC_Data_Pump to MPDU_Handler_Rcv	
Event	**Parameters**
SET_MAC_ADDR	**MT_MAC_Addr**
INIT_MPR	**Net_Id**

Signallist: MMSC2MGDP Slave Scheduler to Gen_MAC_Data_Pump	
Event	**Parameters**
SUSPEND_MSC_OK	***none***
RESUME_MSC_OK	***none***
CLOSE_MSC_DONE	***none***

Signallist: MGDP2MMPX Gen_MAC_Data_Pump to MPDU_Handler_Xmit	
Event	**Parameters**
SET_MAC_ADDR	**MT_MAC_Addr**
INIT_MPX	**NetID** **MAC_Addr**

MPDU_Handler internal interfaces

Signallist: MSSE2MMPR Slot Sequencer to MPDU Handler Receive	
Event	**Parameters**
MPDU_RCVD	**TimeStamp** **MPDU_Type_ptr** **MPDU_Lgth**
FRAME_START	**none**

Signallist: MMPX2MSSE MPDU Handler Xmit to Slot Sequencer

Event	Parameters
TED	**TED_Type** **BufferAddr** **BufferLgth** **SlotCount**

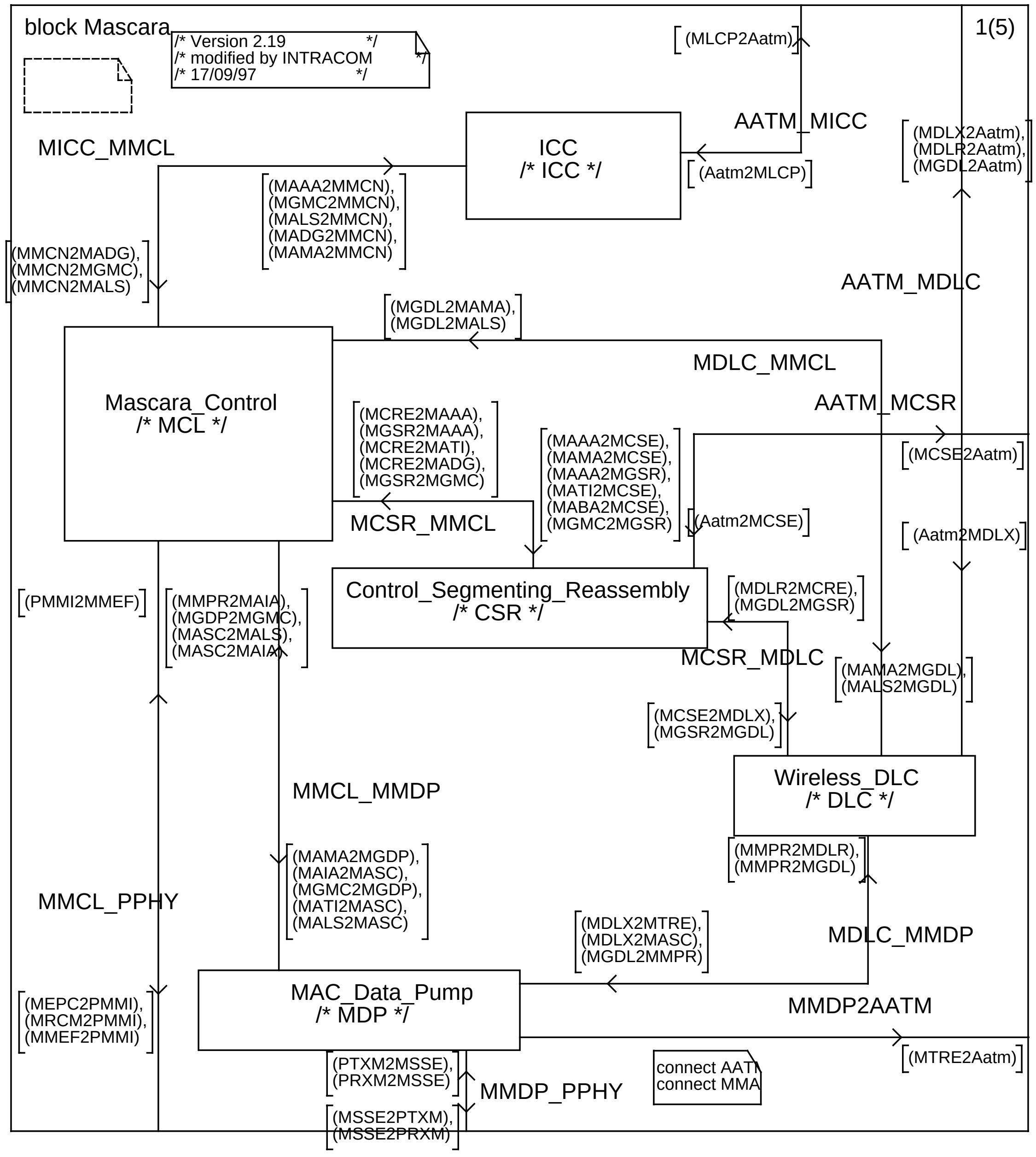
Scheduler internal interfaces

Signallist: MMSC2MTRE MT Scheduler to Traffic Recorder	
Event	**Parameters**
UPDATE_TRE	**LengthIndex** **TT_ServiceEntries**
GET_TABLE	***none***

Signallist: MTRE2MMSC Traffic Recorder to MT Scheduler	
Event	**Parameters**
SET_TABLE	**TrafficTable**

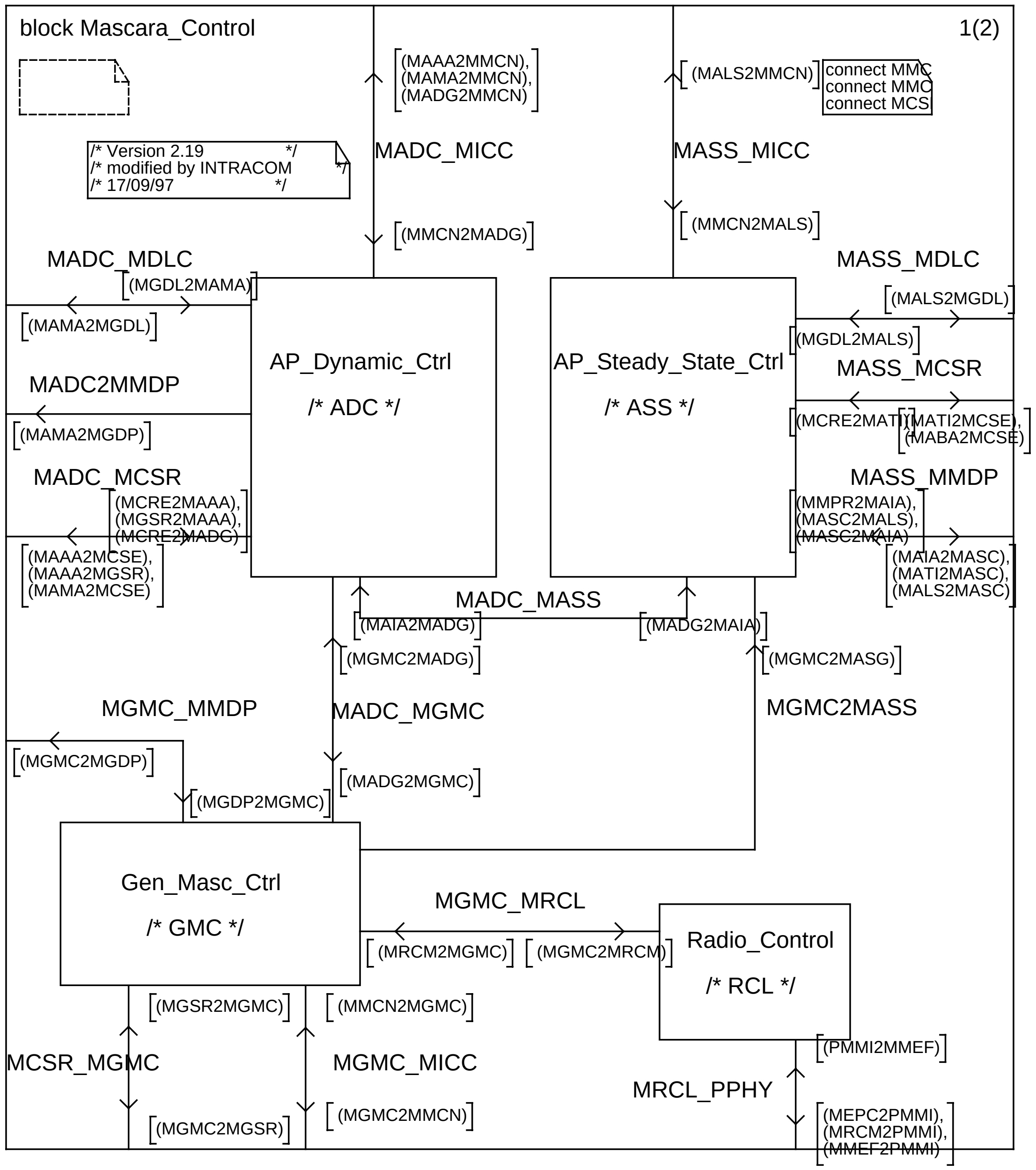
13. Appendix IV. MASCARA SDL Description

13.1. Access Point


block Mascara
/* Version 2.19 */
/* modified by INTRACOM */
/* 17/09/97 */
1(5)
(MLCP2Aatm)
AATM_MICC
ICC
/* ICC */
(Aatm2MLCP)
(MDLX2Aatm),
(MDLR2Aatm),
(MGDL2Aatm)
MICC_MMCL
(MAAA2MMCN),
(MGMC2MMCN),
(MALS2MMCN),
(MADG2MMCN),
(MAMA2MMCN)
(MMCN2MADG),
(MMCN2MGMC),
(MMCN2MALS)
AATM_MDLC
(MGDL2MAMA),
(MGDL2MALS)
MDLC_MMCL
Mascara_Control
/* MCL */
AATM_MCSR
(MCRE2MAAA),
(MGSR2MAAA),
(MCRE2MATI),
(MCRE2MADG),
(MGSR2MGMC)
(MAAA2MCSE),
(MAMA2MCSE),
(MAAA2MGSR),
(MATI2MCSE),
(MABA2MCSE),
(MGMC2MGSR)
(MCSE2Aatm)
(Aatm2MCSE)
(Aatm2MDLX)
MCSR_MMCL
Control_Segmenting_Reassembly
/* CSR */
(PMMI2MMEF)
(MMPR2MAIA),
(MGDP2MGMC),
(MASC2MALS),
(MASC2MAIA)
(MDLR2MCRE),
(MGDL2MGSR)
MCSR_MDLC
(MAMA2MGDL),
(MALS2MGDL)
(MCSE2MDLX),
(MGSR2MGDL)
Wireless_DLC
/* DLC */
MMCL_MMDP
(MMPR2MDLR),
(MMPR2MGDL)
(MAMA2MGDP),
(MAIA2MASC),
(MGMC2MGDP),
(MATI2MASC),
(MALS2MASC)
MMCL_PPHY
(MDLX2MTRE),
(MDLX2MASC),
(MGDL2MMPR)
MDLC_MMDP
MAC_Data_Pump
/* MDP */
MMDP2AATM
(MEPC2PMMI),
(MRCM2PMMI),
(MMEF2PMMI)
(MTRE2Aatm)
(PTXM2MSSE),
(PRXM2MSSE)
MMDP_PPHY
connect AATM
connect MMA
(MSSE2PTXM),
(MSSE2PRXM)

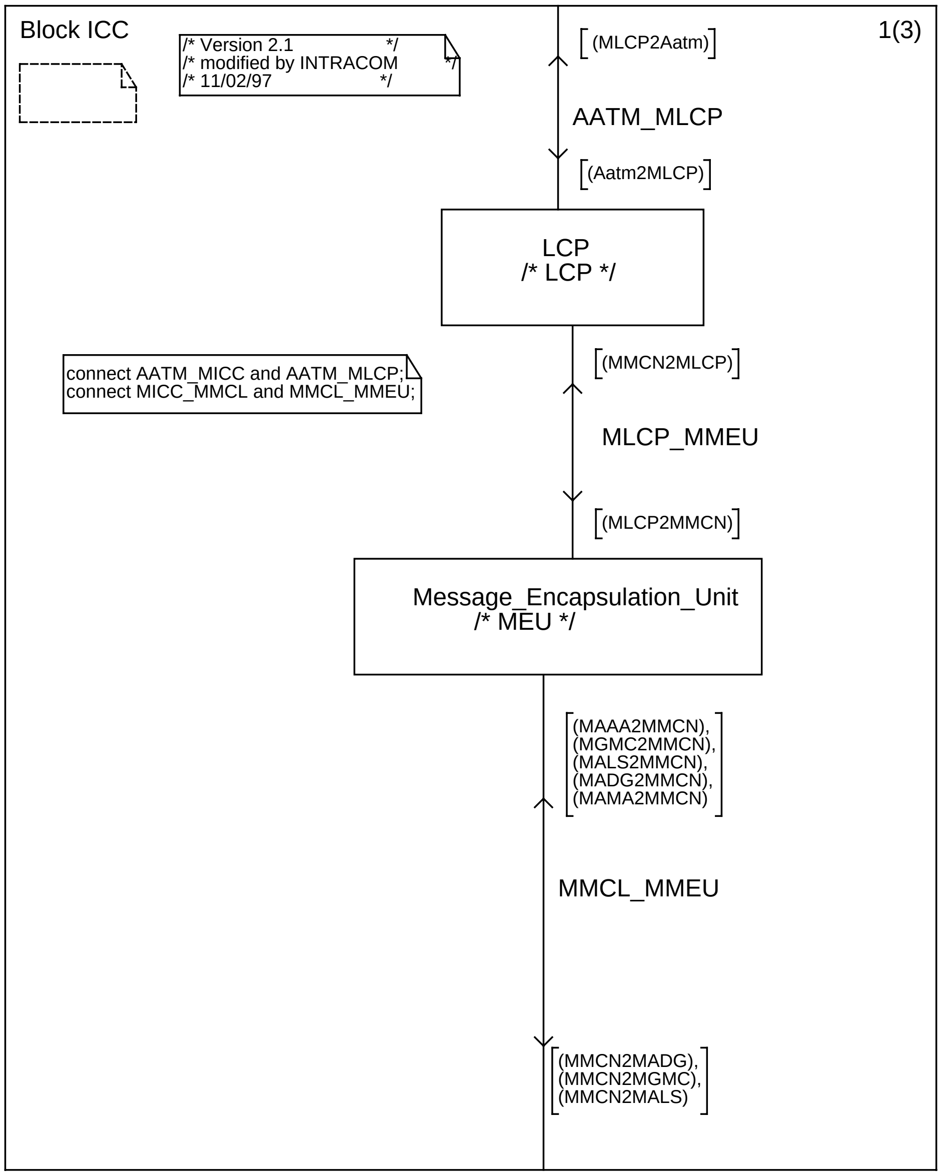
13.1.1.

AP MASCARA Control



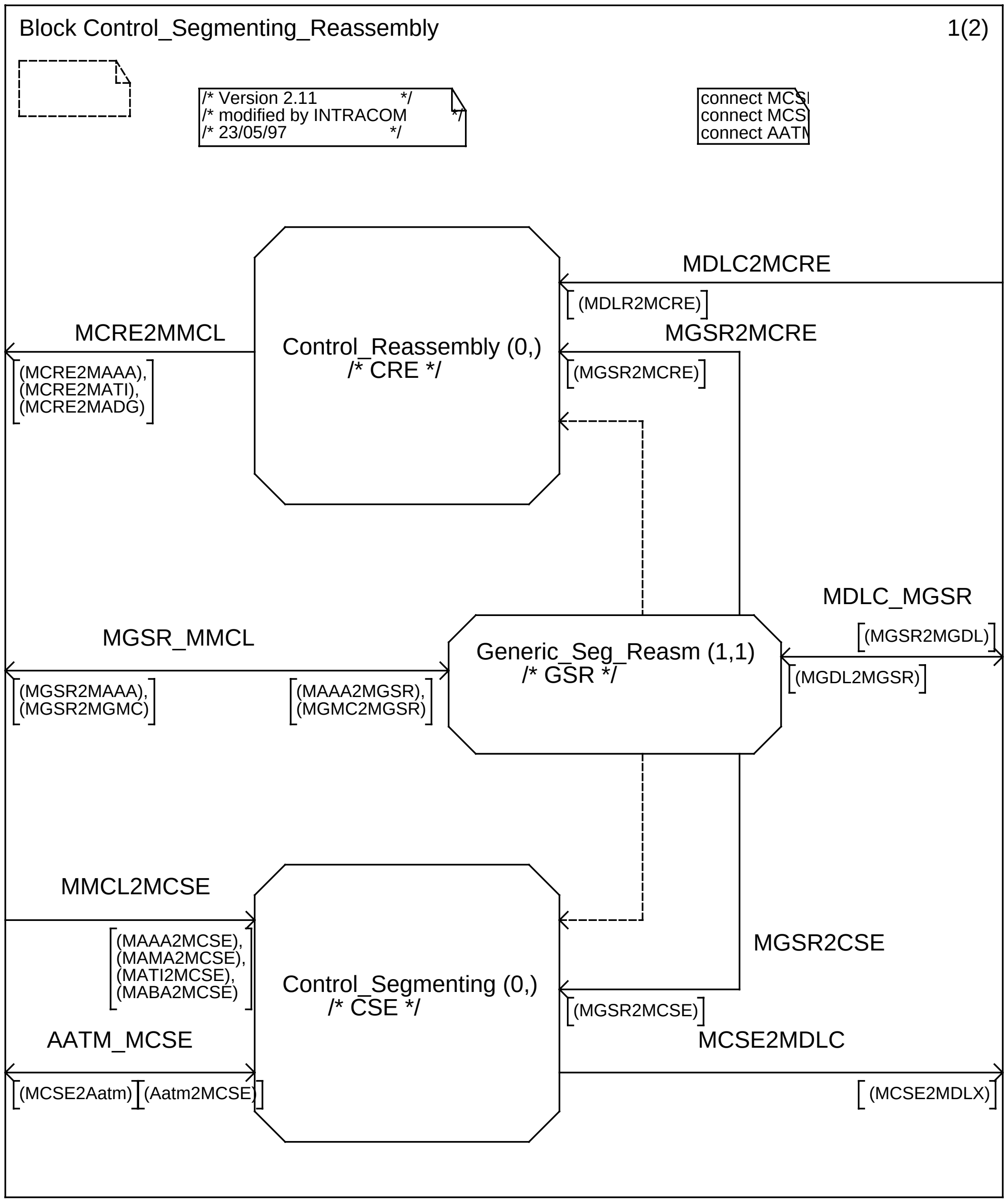
13.1.2.

AP Inter Control Communication


Block ICC
1(3)
/* Version 2.1 */
/* modified by INTRACOM */
/* 11/02/97 */
(MLCP2Aatm)
AATM_MLCP
(Aatm2MLCP)
LCP
/* LCP */
connect AATM_MICC and AATM_MLCP;
connect MICC_MMCL and MMCL_MMEU;
(MMCN2MLCP)
MLCP_MMEU
(MLCP2MMCN)
Message_Encapsulation_Unit
/* MEU */
(MAAA2MMCN), (MGMC2MMCN), (MALS2MMCN), (MADG2MMCN), (MAMA2MMCN)
MMCL_MMEU
(MMCN2MADG), (MMCN2MGMC), (MMCN2MALS)

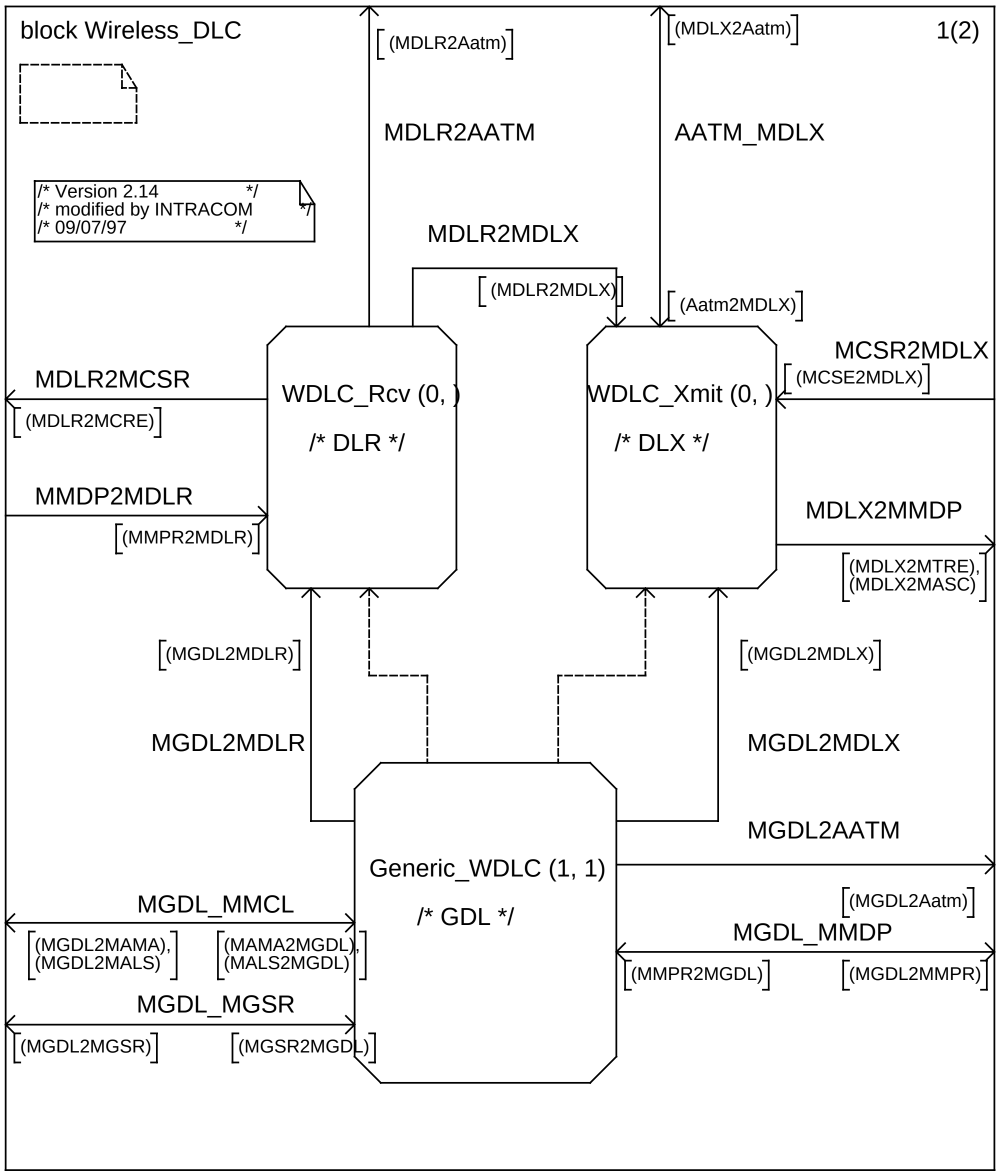
13.1.3.

AP Control Segmentation and Reassembly



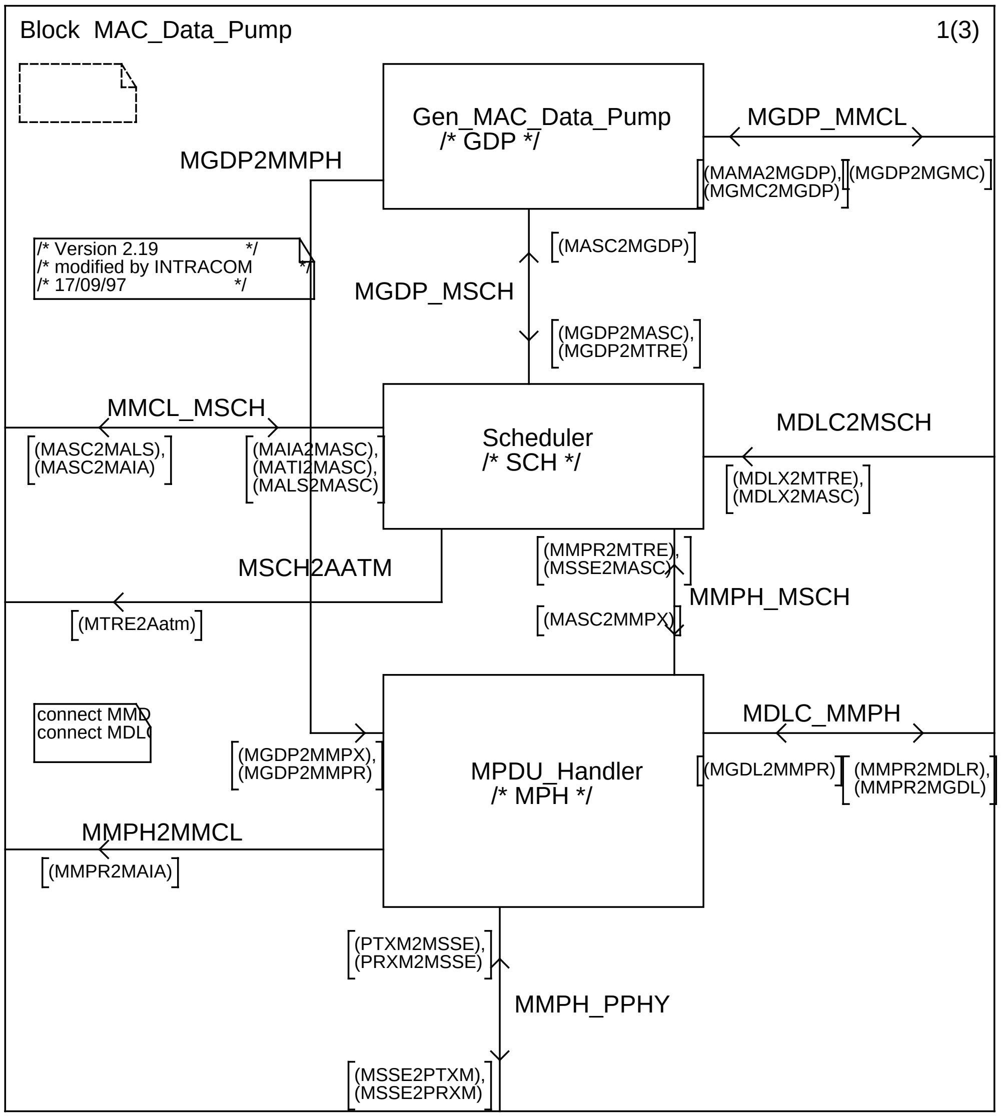
13.1.4.

AP Wireless Data Link Control



13.1.5.

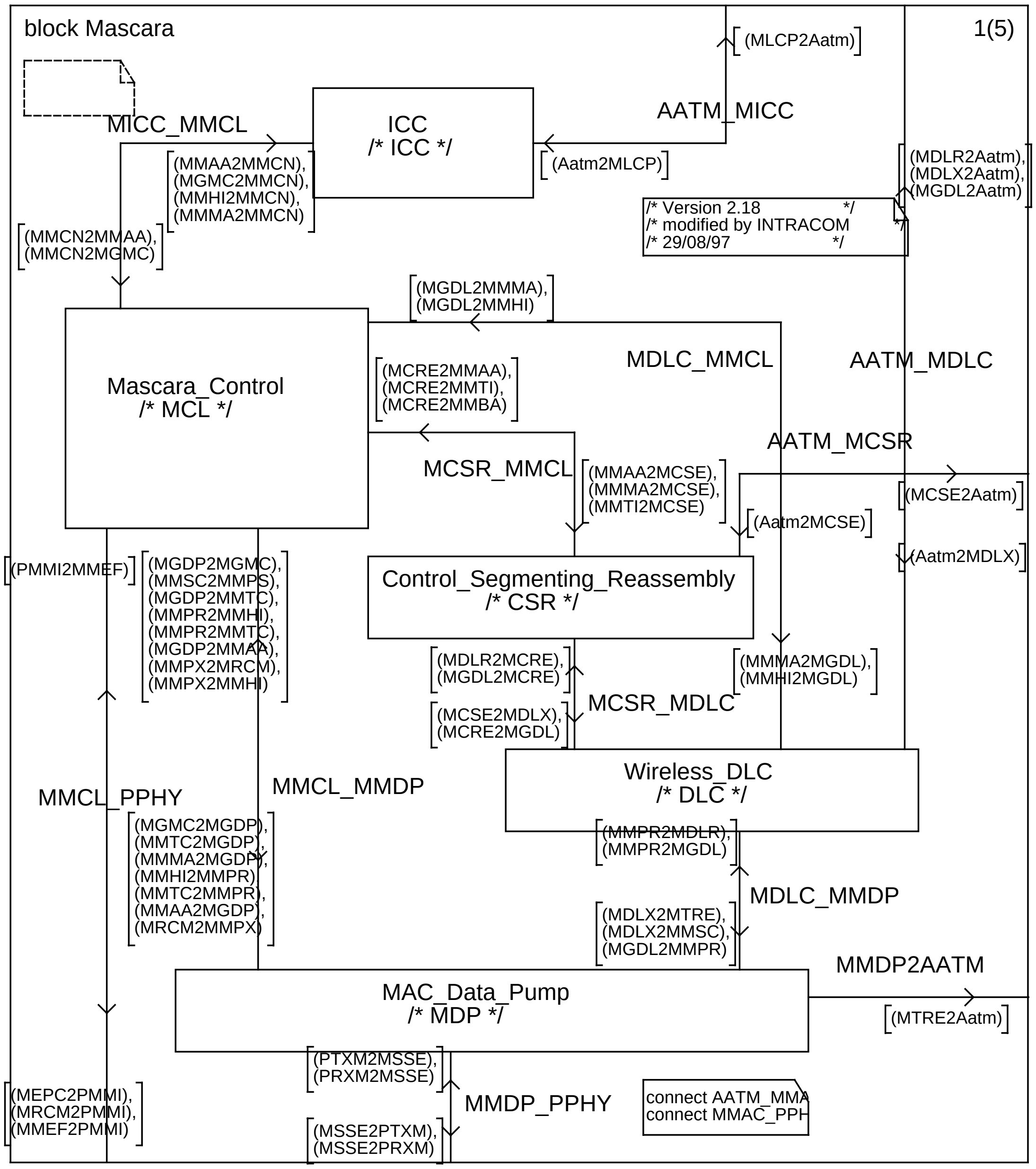
AP MAC Data Pump



13.2.

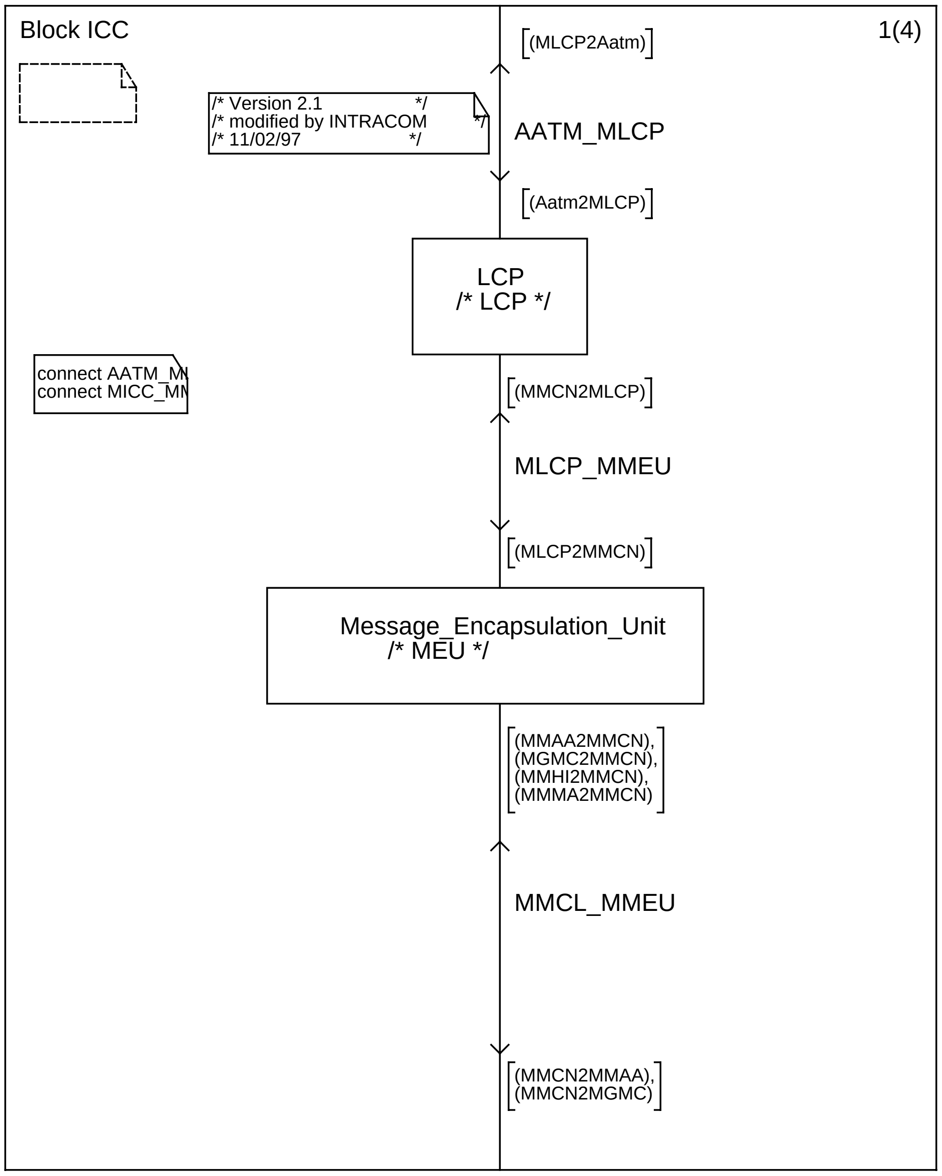
Mobile Terminal

13.2.1. MT MASCARA Control



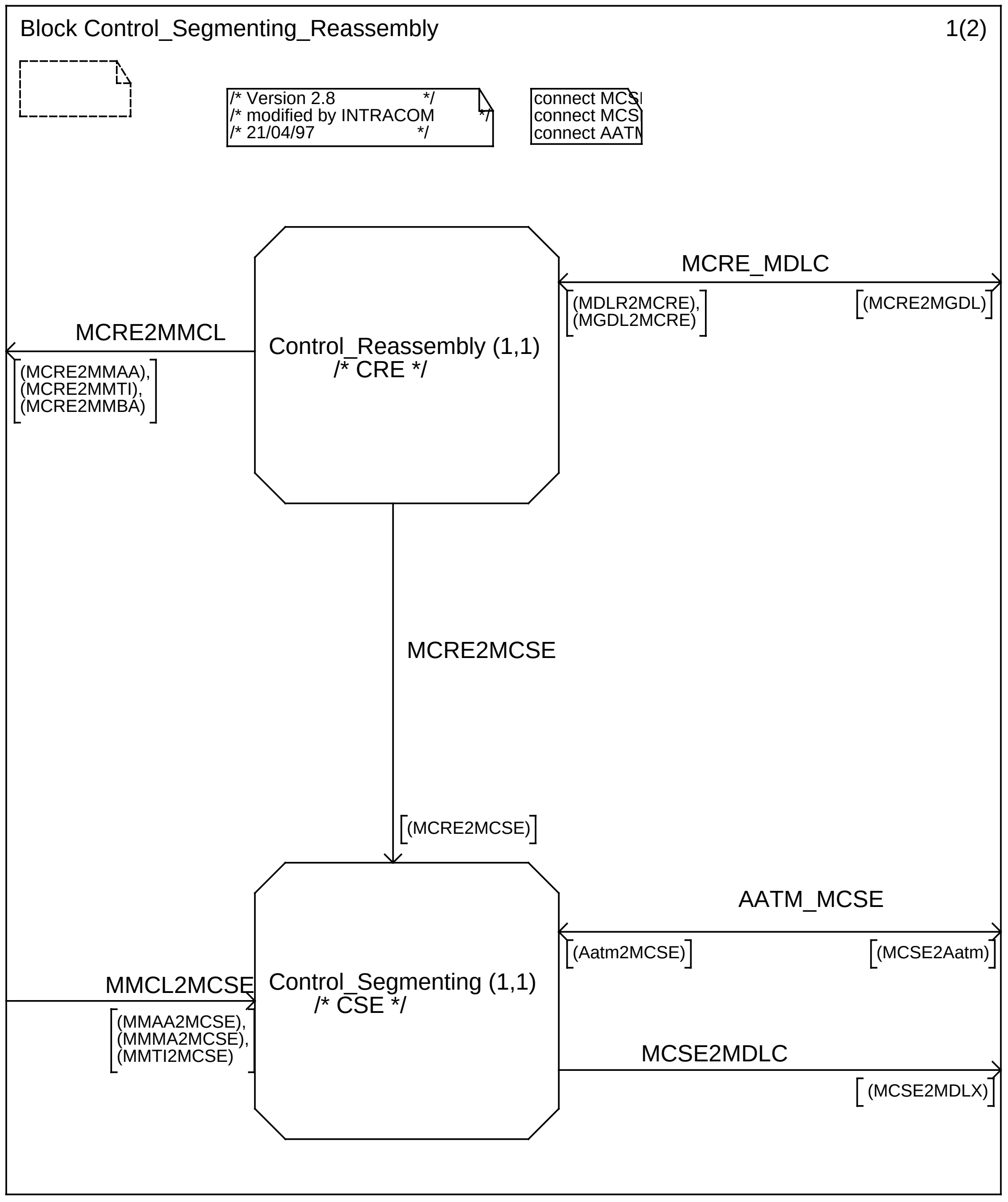
13.2.2.

MT Inter Control Communication



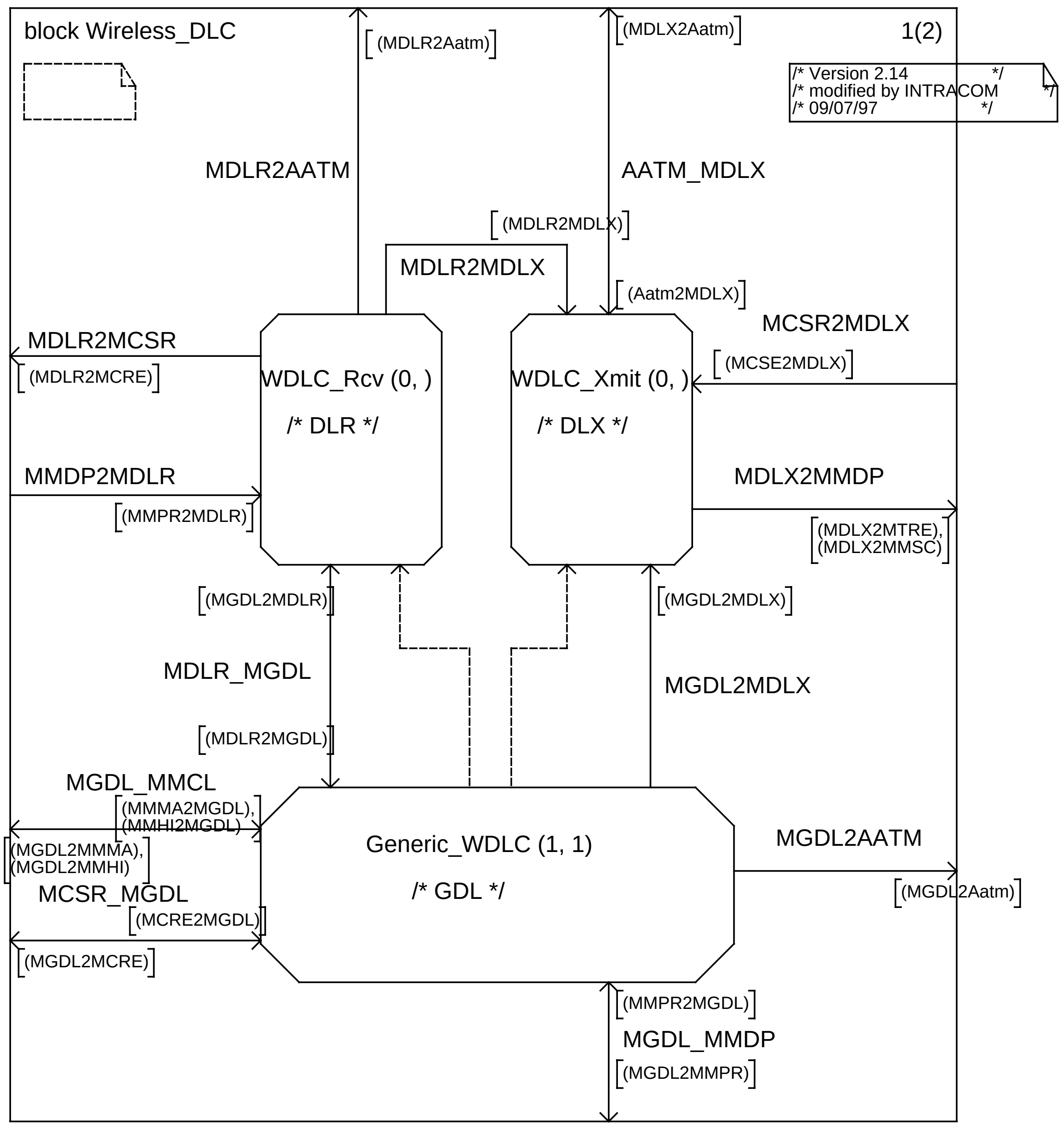
13.2.3.

MT Control Segmentation and Reassembly



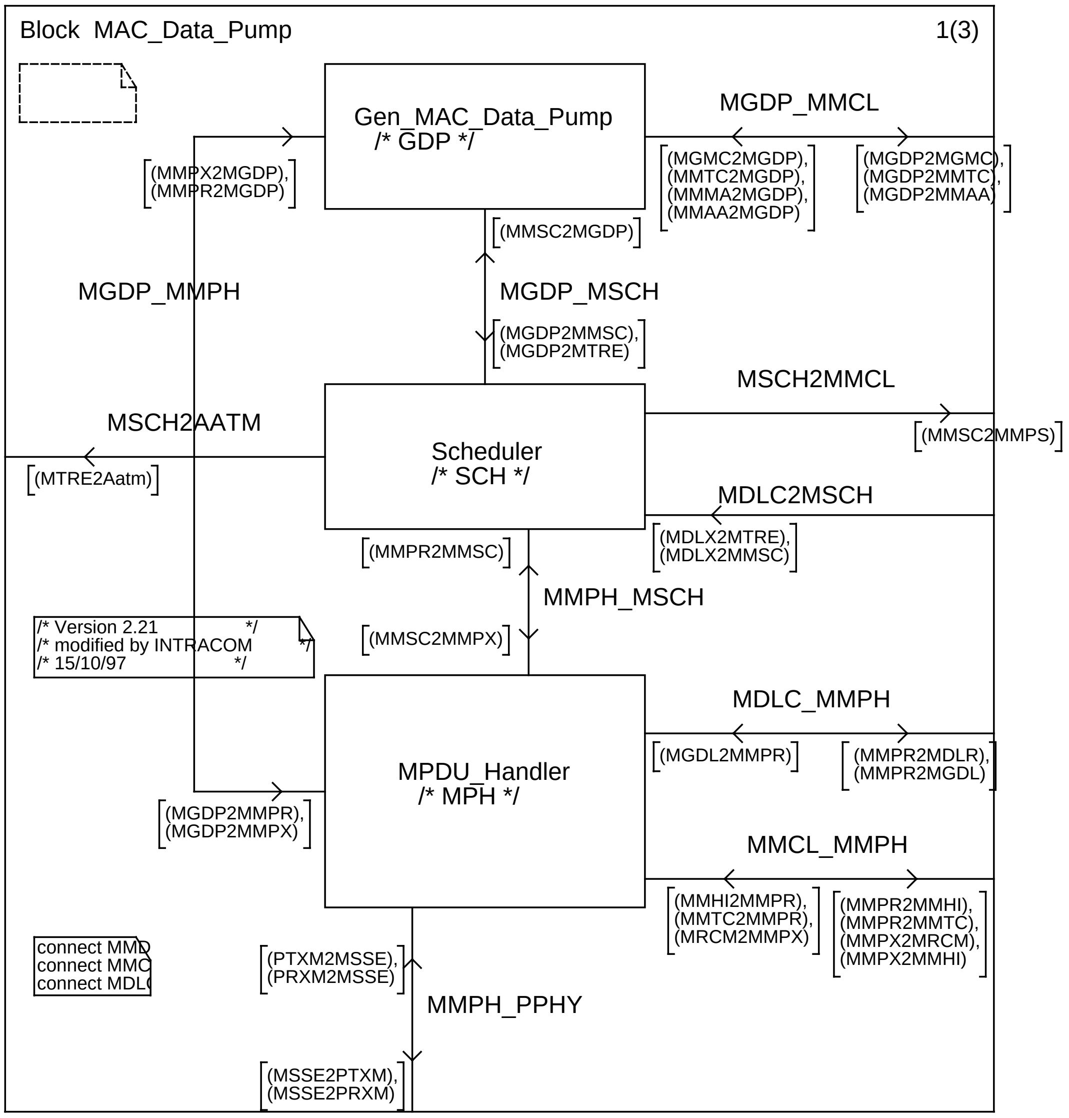
13.2.4.

MT Wireless Data Link Control

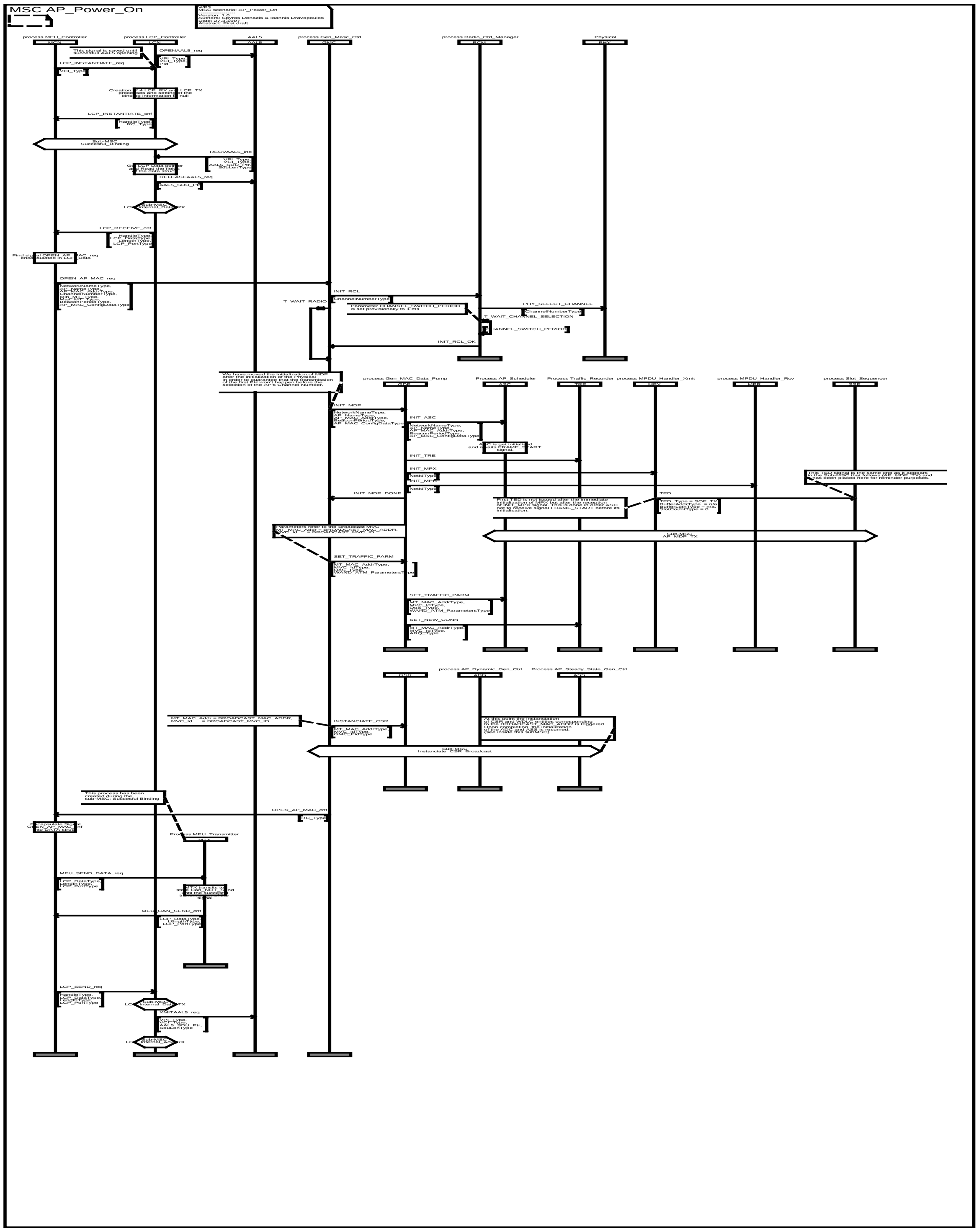


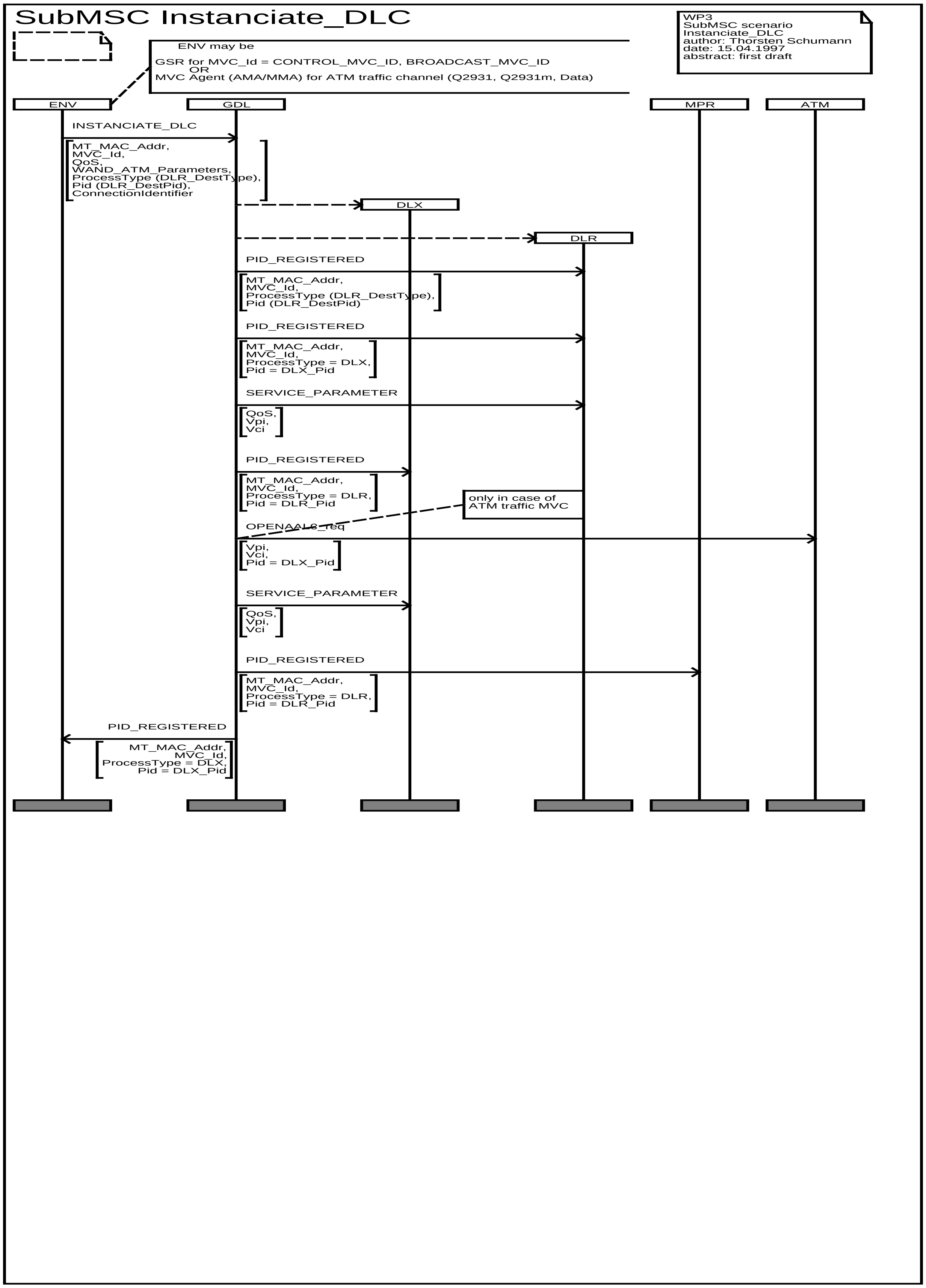
13.2.5.

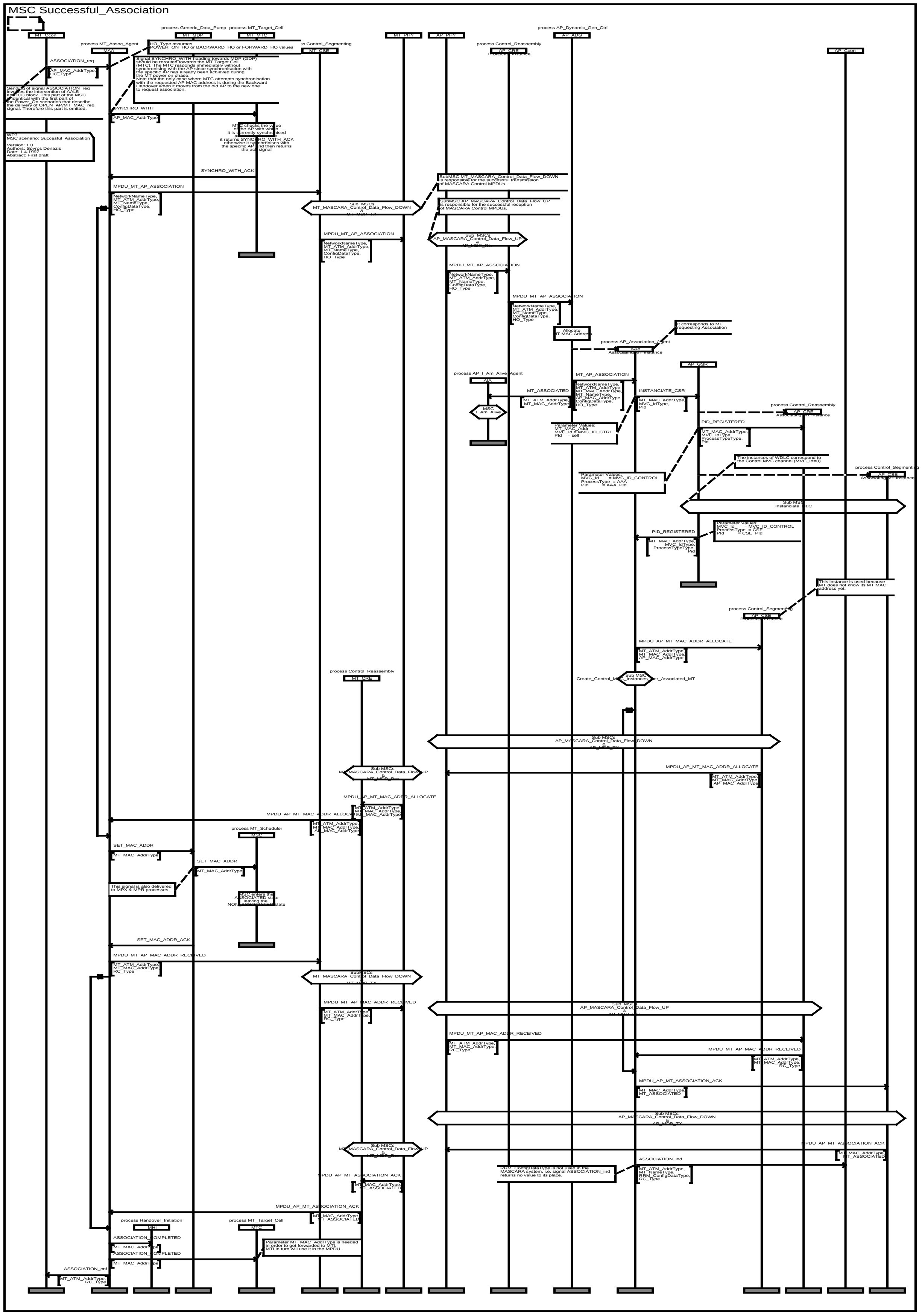
MT MAC Data Pump



14. Appendix V. MASCARA Scenarios Message Sequence Charts


MSC AP_Power_On
WP3
MSC scenario: AP_Power_On
Version: 1.0
Authors: Spyros Denazis & Ioannis Dravopoulos
Date: 27.3.1997
Abstract: First draft
process MEU_Controller
MCN
process LCP_Controller
LCN
AAL5
AAL5
process Gen_Masc_Ctrl
GMC
process Radio_Ctrl_Manager
RCM
Physical
PHY
This signal is saved until succesfull AAL5 opening
OPENAAL5_req
VPI_Type, VCI_Type, Pid
LCP_INSTANTIATE_req
VCI_Type
Creation of 4 LCP_RX and LCP_TX processes and setting of the binding information to null
LCP_INSTANTIATE_cnf
HandleType, RC_Type
Sub-MSC Succesful_Binding
RECVAAL5_ind
VPI_Type, VCI_Type, AAL5_SDU_Ptr, SduLenType
Get LCP Data pointer and Read the fields of the data struct
RELEASEAAL5_req
AAL5_SDU_Ptr
Sub-MSC LCP_Internal_Data_RX
LCP_RECEIVE_cnf
HandleType, LCP_DataType, LengthType, LCP_PortType
Find signal OPEN_AP_MAC_req encapsulated in LCP_Data
OPEN_AP_MAC_req
NetworkNameType, AP_NameType, AP_MAC_AddrType, ChannelNumberType, Min_MT_Type, Max_MT_Type, BeaconPeriodType, AP_MAC_ConfigDataType
INIT_RCL
ChannelNumberType
T_WAIT_RADIO
Parameter CHANNEL_SWITCH_PERIOD is set provisionally to 1 ms
PHY_SELECT_CHANNEL
ChannelNumberType
T_WAIT_CHANNEL_SELECTION
CHANNEL_SWITCH_PERIOD
INIT_RCL_OK
We have moved the initialization of MDP after the initialization of the Physical in order to guarantee that the transmission of the first FH won't happen before the selection of the AP's Channel Number.
process Gen_MAC_Data_Pump
GDP
Process AP_Scheduler
ASC
Process Traffic_Recorder
TRE
process MPDU_Handler_Xmit
MPX
process MPDU_Handler_Rcv
MPR
process Slot_Sequencer
SSE
INIT_MDP
NetworkNameType, AP_NameType, AP_MAC_AddrType, BeaconPeriodType, AP_MAC_ConfigDataType
INIT_ASC
NetworkNameType, AP_NameType, AP_MAC_AddrType, BeaconPeriodType, AP_MAC_ConfigDataType
ASC is get initialised and awaits FRAME_START signal.
INIT_TRE
INIT_MPX
NetIdType
INIT_MPR
NetIdType
INIT_MDP_DONE
This TED signal is the same one as it appears in the Sub-MSC that follows (AP_MDP_TX) and it has been placed here for reminder purposes.
TED
TED_Type = SOF_TX, BufferAddrType = n/a, BufferLgthType = n/a, SlotCountType = 0
First TED is not issued after the immediate initialization of MPX but after the reception of INIT_MPX signal. This is done in order ASC not to receive signal FRAME_START before its initialisation.
Parameters refer to the Broadcast MVC
MT_MAC_Addr = BROADCAST_MAC_ADDR, MVC_Id = BROADCAST_MVC_ID
Sub-MSC AP_MDP_TX
SET_TRAFFIC_PARM
MT_MAC_AddrType, MVC_IdType, QoS_Type, WAND_ATM_ParametersType
SET_TRAFFIC_PARM
MT_MAC_AddrType, MVC_IdType, QoS_Type, WAND_ATM_ParametersType
SET_NEW_CONN
MT_MAC_AddrType, MVC_IdType, ARQ_Type
GSR
process AP_Dynamic_Gen_Ctrl
ADG
Process AP_Steady_State_Gen_Ctrl
ASS
MT_MAC_Addr = BROADCAST_MAC_ADDR, MVC_Id = BROADCAST_MVC_ID
INSTANCIATE_CSR
MT_MAC_AddrType, MVC_IdType, GMC_PidType
At this point the instanciation of CSR and WDLC entities corresponding to the BROADCAST_MAC_ADDR is triggered. Upon completion, the initialization of the ADG and ASS is resumed. (see inside this subMSC)
Sub-MSC Instanciate_CSR_Broadcast
This process has been created during the sub-MSC: Succesful Binding
OPEN_AP_MAC_cnf
RC_Type
Encapsulate Signal OPEN_AP_MAC_cnf into DATA struct
Process MEU_Transmitter
MTX
MEU_SEND_DATA_req
LCP_DataType, LengthType, LCP_PortType
MTX transits to state Can_NOT_Send until the succesful transmission of the signal
MEU_CAN_SEND_cnf
LCP_DataType, LengthType, LCP_PortType
LCP_SEND_req
HandleType, LCP_DataType, LengthType, LCP_PortType
Sub-MSC LCP_Internal_Data_TX
XMITAAL5_req
VPI_Type, VCI_Type, AAL5_SDU_Ptr, SduLenType
Sub-MSC LCP_Internal_Ack_RX


SubMSC Instanciate_DLC
WP3
SubMSC scenario
Instanciate_DLC
author: Thorsten Schumann
date: 15.04.1997
abstract: first draft
ENV may be
GSR for MVC_Id = CONTROL_MVC_ID, BROADCAST_MVC_ID
OR
MVC Agent (AMA/MMA) for ATM traffic channel (Q2931, Q2931m, Data)
ENV
GDL
MPR
ATM
INSTANCIATE_DLC
MT_MAC_Addr, MVC_Id, QoS, WAND_ATM_Parameters, ProcessType (DLR_DestType), Pid (DLR_DestPid), ConnectionIdentifier
DLX
DLR
PID_REGISTERED
MT_MAC_Addr, MVC_Id, ProcessType (DLR_DestType), Pid (DLR_DestPid)
PID_REGISTERED
MT_MAC_Addr, MVC_Id, ProcessType = DLX, Pid = DLX_Pid
SERVICE_PARAMETER
QoS, Vpi, Vci
PID_REGISTERED
MT_MAC_Addr, MVC_Id, ProcessType = DLR, Pid = DLR_Pid
only in case of ATM traffic MVC
OPENAAL6_req
Vpi, Vci, Pid = DLX_Pid
SERVICE_PARAMETER
QoS, Vpi, Vci
PID_REGISTERED
MT_MAC_Addr, MVC_Id, ProcessType = DLR, Pid = DLR_Pid
PID_REGISTERED
MT_MAC_Addr, MVC_Id, ProcessType = DLX, Pid = DLX_Pid



HO_Type assumes POWER_ON_HO or BACKWARD_HO or FORWARD_HO values

Signal SYNCHRO_WITH heading towards MDP (GDP) should be rerouted towards the MT Target Cell (MTC). The MTC responds immediately without synchronising with the AP since synchronisation with the specific AP has already been achieved during the MT power on phase.
Note that the only case where MTC attempts synchronisation with the requested AP MAC address is during the Backward Handover when it moves from the old AP to the new one to request association.

Sending of signal ASSOCIATION_req involves the intervention of AAL5 and ICC block. This part of the MSC is identical with the first part of the Power_On scenarios that describe the delivery of OPEN_AP/MT_MAC_req signal. Therefore this part is omitted.

WP3
MSC scenario: Succesful_Association

Version: 1.0
Authors: Spyros Denazis
Date: 1.4.1997
Abstract: First draft

MTC checks the value of the AP with which it is currently synchronised and if it is the same it returns SYNCHRO_WITH_ACK otherwise it synchronises with the specific AP and then returns the ack signal

SubMSC MT_MASCARA_Control_Data_Flow_DOWN is responsible for the successful transmission of MASCARA Control MPDUs.

SubMSC AP_MASCARA_Control_Data_Flow_UP is responsible for the successful reception of MASCARA Control MPDUs.

It corresponds to MT requesting Association

Parameter Values:
MT_MAC_Addr
MVC_Id = MVC_ID_CTRL
PId = self

Parameter Values:
MVC_Id = MVC_ID_CONTROL
ProcessType = AAA
PId = AAA_PId

The instances of WDLC correspond to the Control MVC channel (MVC_Id=0)

Parameter Values:
MVC_Id = MVC_ID_CONTROL
ProcessType = CSE
PId = CSE_PId

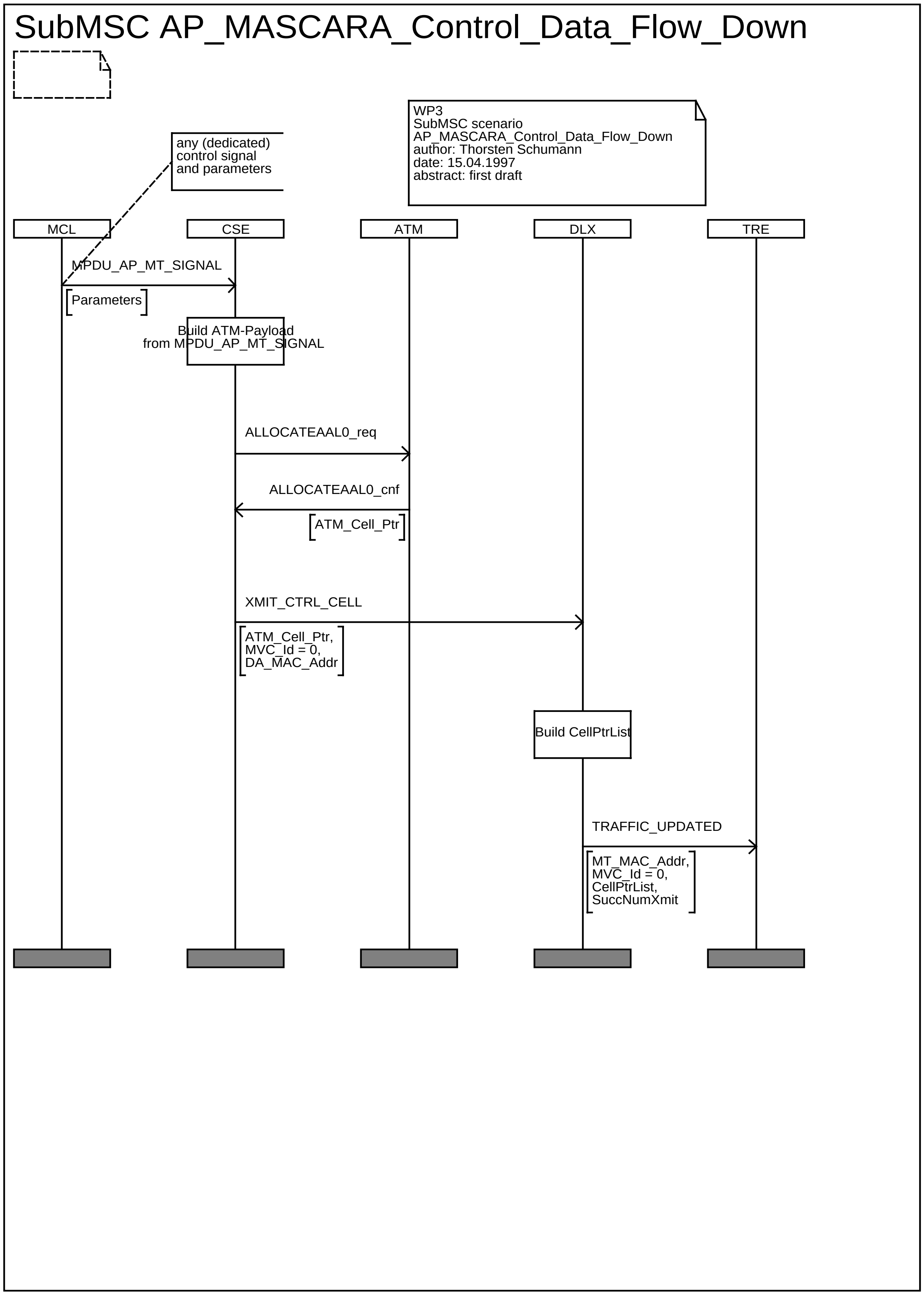
This instance is used because MT does not know its MT MAC address yet.

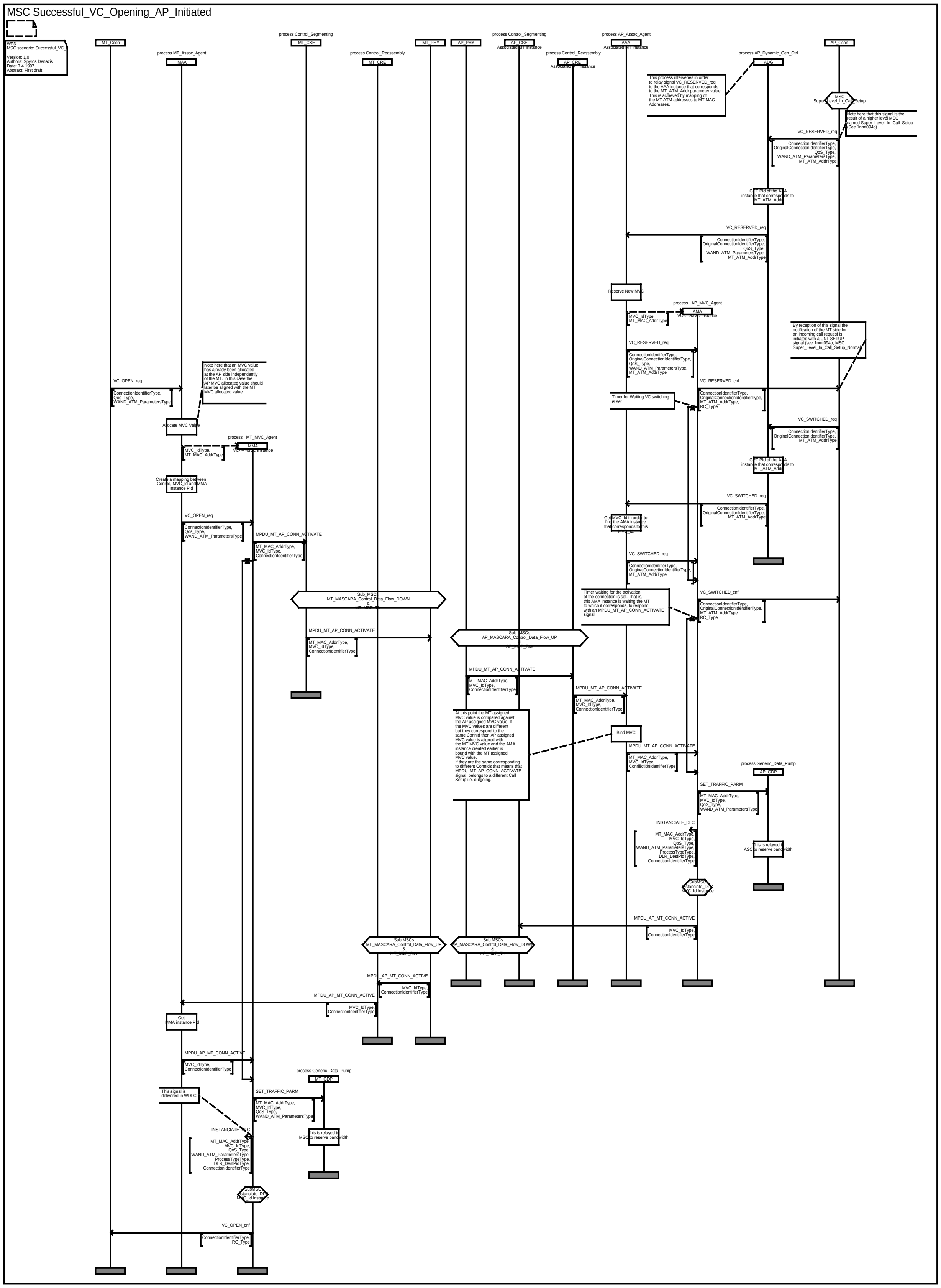
This signal is also delivered to MPX & MPR processes.

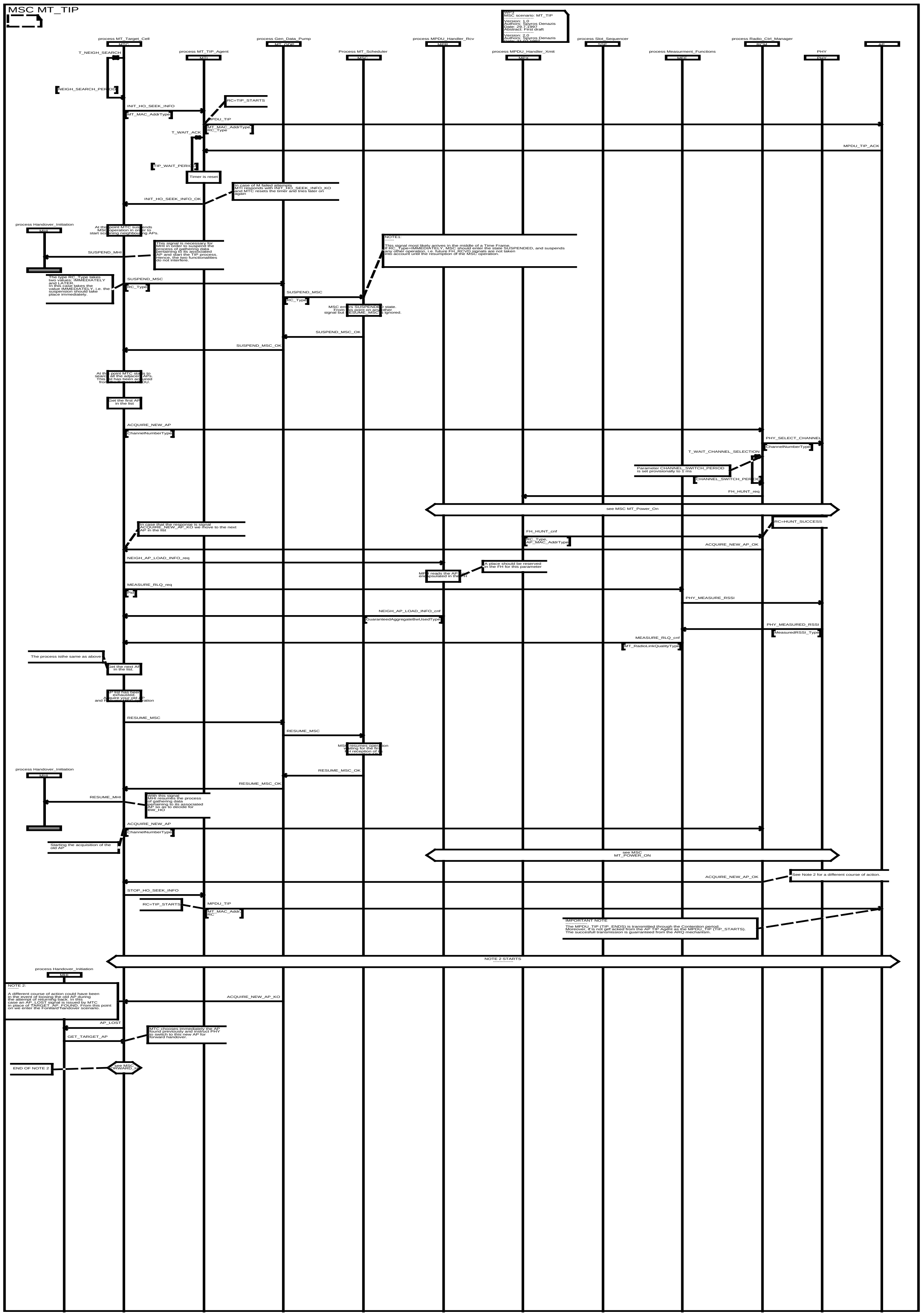
MSC enters the ASSOCIATED state leaving the NON_ASSOCIATED state

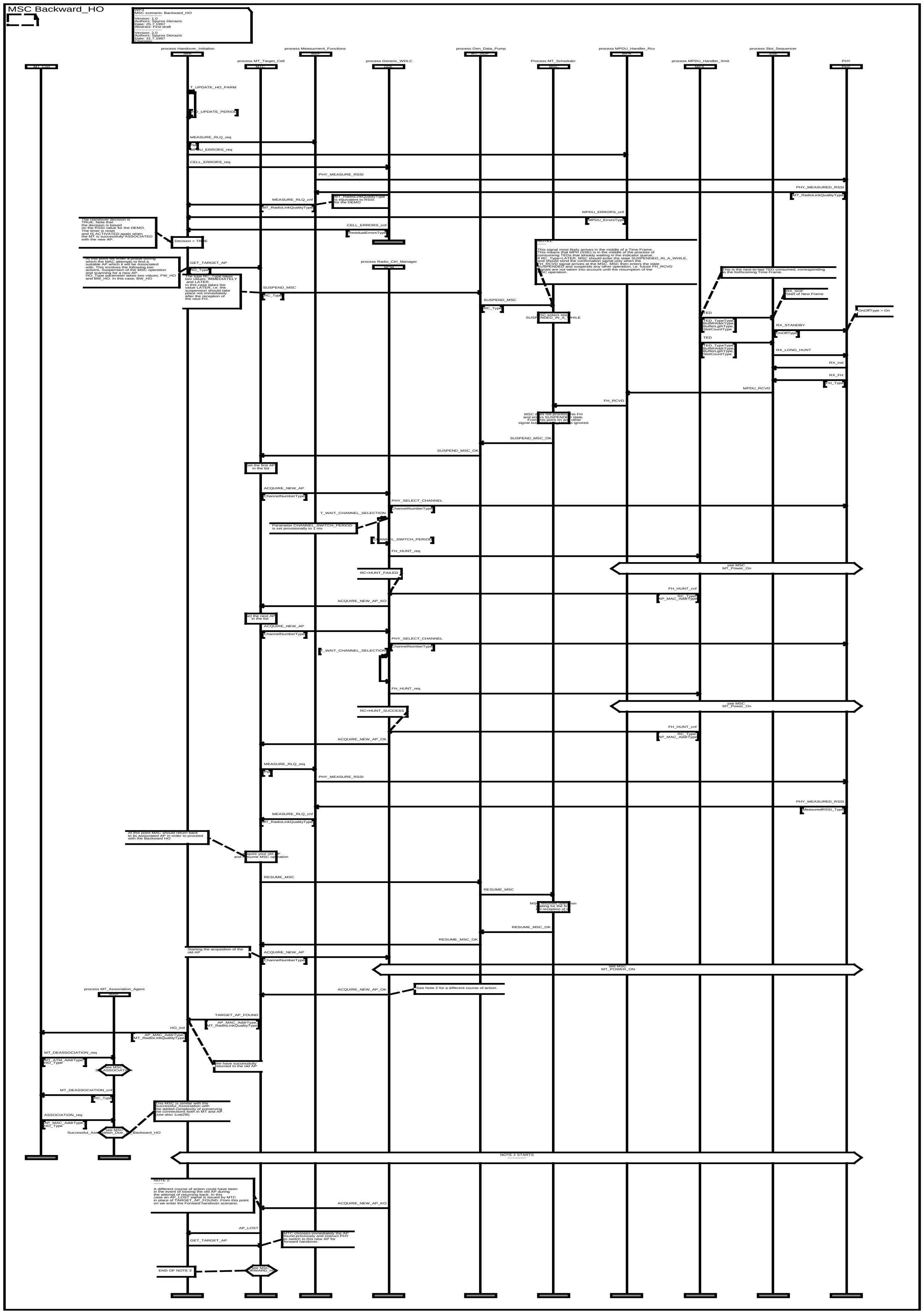
RRM_ConfigDataType is not used in the MASCARA system, i.e. signal ASSOCIATION_ind returns no value to its place.

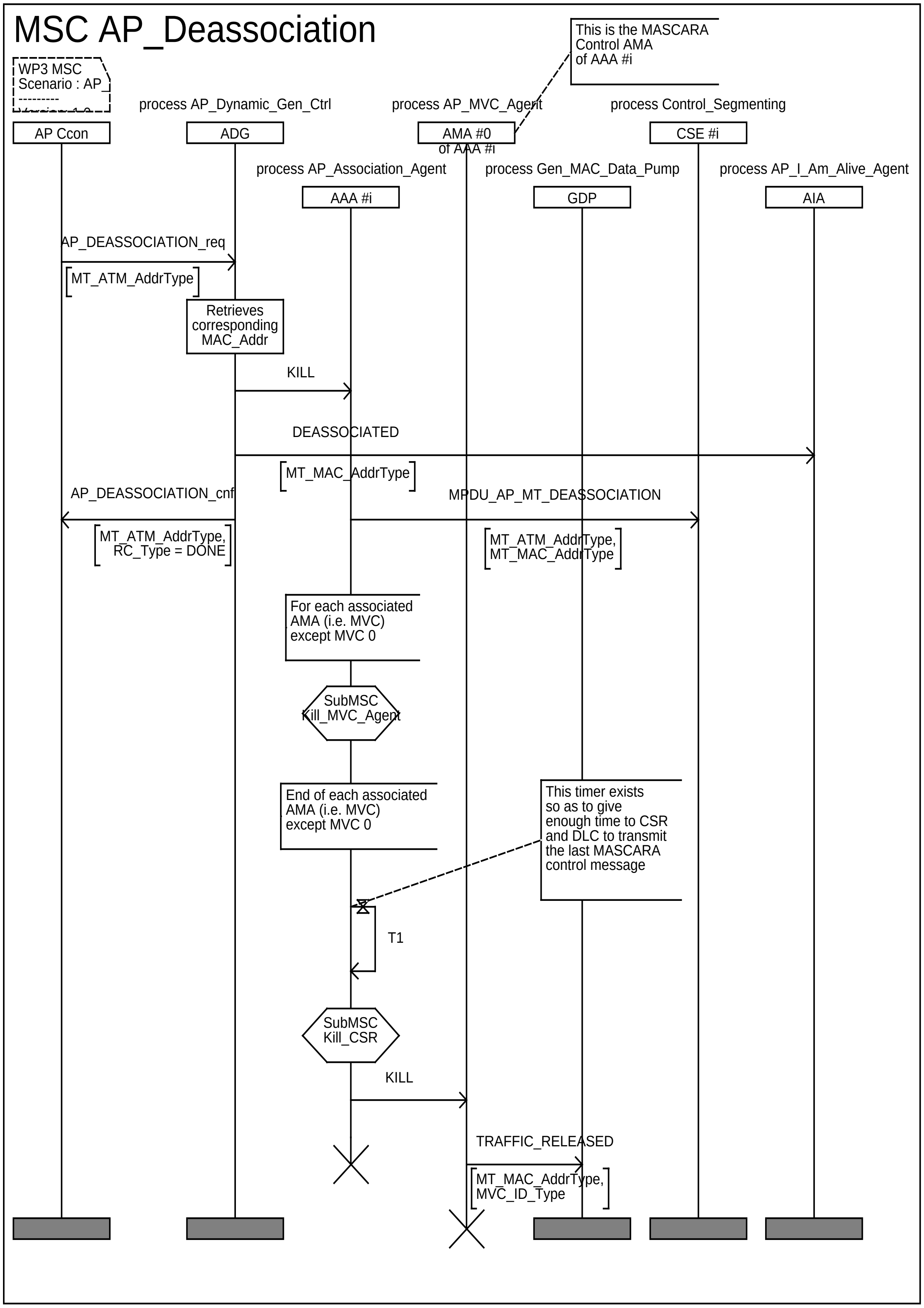
Parameter MT_MAC_AddrType is needed in order to get forwarded to MTI. MTI in turn will use it in the MPDU.

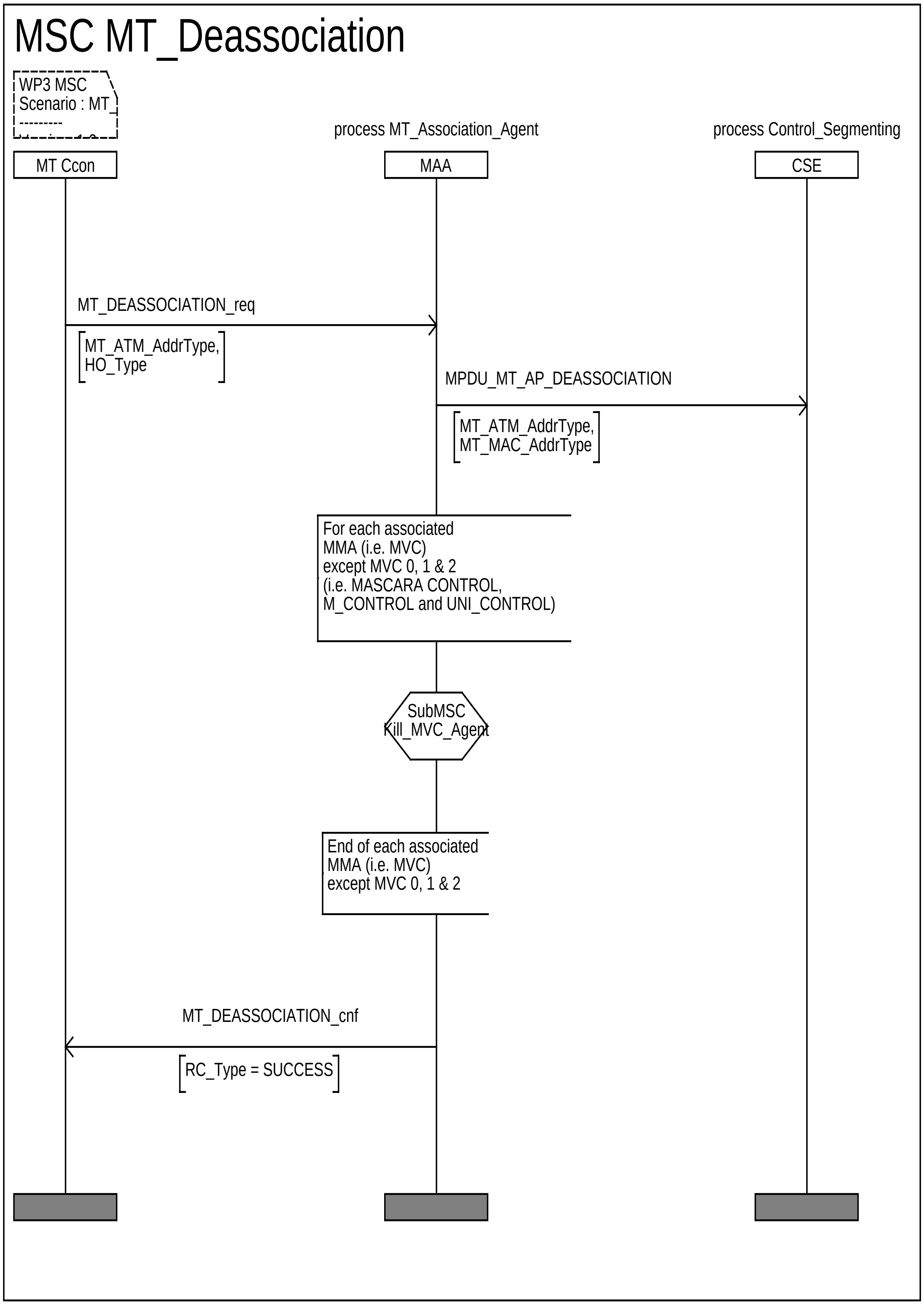

SubMSC AP_MASCARA_Control_Data_Flow_Down
WP3
SubMSC scenario
AP_MASCARA_Control_Data_Flow_Down
author: Thorsten Schumann
date: 15.04.1997
abstract: first draft
any (dedicated)
control signal
and parameters
MCL
CSE
ATM
DLX
TRE
MPDU_AP_MT_SIGNAL
Parameters
Build ATM-Payload
from MPDU_AP_MT_SIGNAL
ALLOCATEAAL0_req
ALLOCATEAAL0_cnf
ATM_Cell_Ptr
XMIT_CTRL_CELL
ATM_Cell_Ptr,
MVC_Id = 0,
DA_MAC_Addr
Build CellPtrList
TRAFFIC_UPDATED
MT_MAC_Addr,
MVC_Id = 0,
CellPtrList,
SuccNumXmit

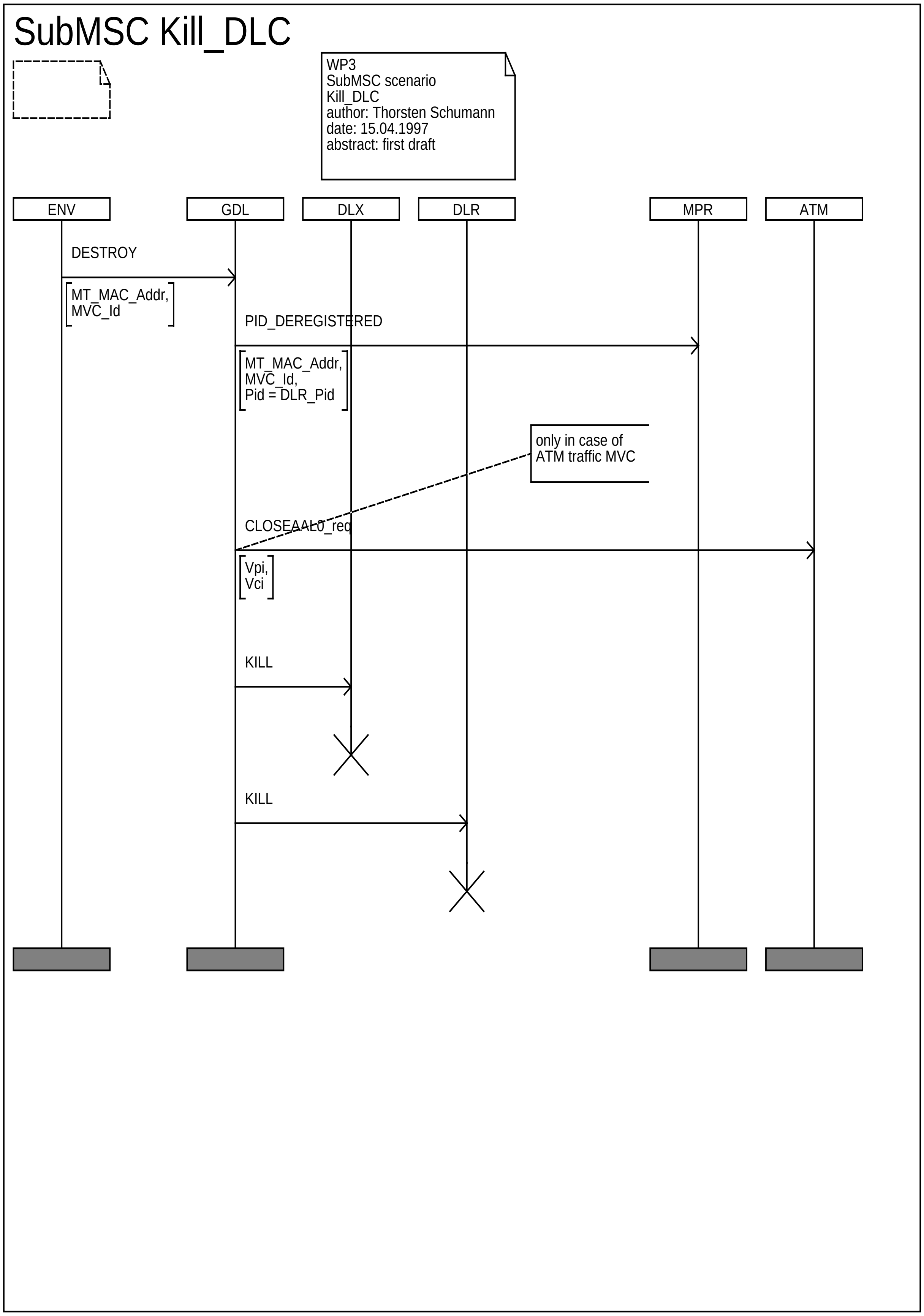

MSC Successful_VC_Opening_AP_Initiated
WP3
MSC scenario: Successful_VC_
Version: 1.0
Authors: Spyros Denazis
Date: 7.4.1997
Abstract: First draft
MT_Ccon
process MT_Assoc_Agent
MAA
process Control_Segmenting
MT_CSE
process Control_Reassembly
MT_CRE
MT_PHY
AP_PHY
process Control_Segmenting
AP_CSE
Associated MT instance
process Control_Reassembly
AP_CRE
Associated MT instance
process AP_Assoc_Agent
AAA
Associated MT instance
process AP_Dynamic_Gen_Ctrl
ADG
AP_Ccon
This process intervenes in order to relay signal VC_RESERVED_req to the AAA instance that corresponds to the MT_ATM_Addr parameter value. This is achieved by mapping of the MT ATM addresses to MT MAC Addresses.
MSC Super_Level_In_Call_Setup
Note here that this signal is the result of a higher level MSC named Super_Level_In_Call_Setup (See 1nmt094b)
VC_RESERVED_req
ConnectionIdentifierType, OriginalConnectionIdentifierType, QoS_Type, WAND_ATM_ParametersType, MT_ATM_AddrType
Get Pid of the AAA instance that corresponds to MT_ATM_Addr
VC_RESERVED_req
ConnectionIdentifierType, OriginalConnectionIdentifierType, QoS_Type, WAND_ATM_ParametersType, MT_ATM_AddrType
Reserve New MVC
process AP_MVC_Agent
AMA
VC<->MVC instance
MVC_IdType, MT_MAC_AddrType
By reception of this signal the notification of the MT side for an incoming call request is initiated with a UNI_SETUP signal (see 1nmt094b, MSC Super_Level_In_Call_Setup_Normal
VC_RESERVED_req
ConnectionIdentifierType, OriginalConnectionIdentifierType, QoS_Type, WAND_ATM_ParametersType, MT_ATM_AddrType
Note here that an MVC value has already been allocated at the AP side independently of the MT. In this case the AP MVC allocated value should later be aligned with the MT MVC allocated value.
VC_OPEN_req
ConnectionIdentifierType, Qos_Type, WAND_ATM_ParametersType
VC_RESERVED_cnf
ConnectionIdentifierType, OriginalConnectionIdentifierType, MT_ATM_AddrType, RC_Type
Timer for Waiting VC switching is set
Allocate MVC Value
process MT_MVC_Agent
MMA
VC<->MVC instance
MVC_IdType, MT_MAC_AddrType
VC_SWITCHED_req
ConnectionIdentifierType, OriginalConnectionIdentifierType, MT_ATM_AddrType
Get Pid of the AAA instance that corresponds to MT_ATM_Addr
Create a mapping between ConnId, MVC_Id and MMA Instance Pid
VC_SWITCHED_req
ConnectionIdentifierType, OriginalConnectionIdentifierType, MT_ATM_AddrType
VC_OPEN_req
ConnectionIdentifierType, Qos_Type, WAND_ATM_ParametersType
MPDU_MT_AP_CONN_ACTIVATE
MT_MAC_AddrType, MVC_IdType, ConnectionIdentifierType
Get MVC_Id in order to find the AMA instance that corresponds to this
VC_SWITCHED_req
ConnectionIdentifierType, OriginalConnectionIdentifierType, MT_ATM_AddrType
Sub_MSCs MT_MASCARA_Control_Data_Flow_DOWN &
Timer waiting for the activation of the connection is set. That is, this AMA instance is waiting the MT to which it corresponds, to respond with an MPDU_MT_AP_CONN_ACTIVATE signal.
VC_SWITCHED_cnf
ConnectionIdentifierType, OriginalConnectionIdentifierType, MT_ATM_AddrType, RC_Type
MPDU_MT_AP_CONN_ACTIVATE
MT_MAC_AddrType, MVC_IdType, ConnectionIdentifierType
Sub MSCs AP_MASCARA_Control_Data_Flow_UP
MPDU_MT_AP_CONN_ACTIVATE
MT_MAC_AddrType, MVC_IdType, ConnectionIdentifierType
MPDU_MT_AP_CONN_ACTIVATE
MT_MAC_AddrType, MVC_IdType, ConnectionIdentifierType
At this point the MT assigned MVC value is compared against the AP assigned MVC value. If the MVC values are different but they correspond to the same ConnId then AP assigned MVC value is aligned with the MT MVC value and the AMA instance created earlier is bound with the MT assigned MVC value. If they are the same corresponding to different ConnIds that means that MPDU_MT_AP_CONN_ACTIVATE signal belongs to a different Call Setup i.e. outgoing.
Bind MVC
MPDU_MT_AP_CONN_ACTIVATE
MT_MAC_AddrType, MVC_IdType, ConnectionIdentifierType
process Generic_Data_Pump
AP_GDP
SET_TRAFFIC_PARM
MT_MAC_AddrType, MVC_IdType, QoS_Type, WAND_ATM_ParametersType
INSTANCIATE_DLC
MT_MAC_AddrType, MVC_IdType, QoS_Type, WAND_ATM_ParametersType, ProcessTypeType, DLR_DestPidType, ConnectionIdentifierType
This is relayed to ASC to reserve bandwidth
SubMSC Instanciate_DLC MVC_Id Instance
MPDU_AP_MT_CONN_ACTIVE
MVC_IdType, ConnectionIdentifierType
Sub MSCs MT_MASCARA_Control_Data_Flow_UP &
Sub MSCs AP_MASCARA_Control_Data_Flow_DOWN &
MPDU_AP_MT_CONN_ACTIVE
MVC_IdType, ConnectionIdentifierType
MPDU_AP_MT_CONN_ACTIVE
MVC_IdType, ConnectionIdentifierType
Get MMA instance Pid
MPDU_AP_MT_CONN_ACTIVE
MVC_IdType, ConnectionIdentifierType
process Generic_Data_Pump
MT_GDP
This signal is delivered in WDLC
SET_TRAFFIC_PARM
MT_MAC_AddrType, MVC_IdType, QoS_Type, WAND_ATM_ParametersType
INSTANCIATE_DLC
MT_MAC_AddrType, MVC_IdType, QoS_Type, WAND_ATM_ParametersType, ProcessTypeType, DLR_DestPidType, ConnectionIdentifierType
This is relayed to MSC to reserve bandwidth
SubMSC Instanciate_DLC MVC_Id Instance
VC_OPEN_cnf
ConnectionIdentifierType, RC_Type


MSC MT_TIP
WP3
MSC scenario: MT_TIP
Version: 1.0
Authors: Spyros Denazis
Date: 29.7.1997
Abstract: First draft
Version: 2.0
Authors: Spyros Denazis
process MT_Target_Cell
MTC
process MT_TIP_Agent
MTI
process Gen_Data_Pump
MT_GDP
Process MT_Scheduler
MSC
process MPDU_Handler_Rcv
MPR
process MPDU_Handler_Xmit
MPX
process Slot_Sequencer
SSE
process Measurment_Functions
MFF
process Radio_Ctrl_Manager
RCM
PHY
AP
T_NEIGH_SEARCH
NEIGH_SEARCH_PERIOD
INIT_HO_SEEK_INFO
MT_MAC_AddrType
RC=TIP_STARTS
MPDU_TIP
MT_MAC_AddrType, RC_Type
T_WAIT_ACK
MPDU_TIP_ACK
TIP_WAIT_PERIOD
Timer is reset
In case of M failed attempts MTI responds with INIT_HO_SEEK_INFO_KO and MTC resets the timer and tries later on again
INIT_HO_SEEK_INFO_OK
process Handover_Initiation
MHI
At this point MTC suspends MSC operation in order to start scanning neighbouring APs.
This signal is necessary for MHI in order to suspend the process of gathering data pertaining to its associated AP and start the TIP process. Hence, the two functionalities do not interfere.
SUSPEND_MHI
NOTE1:
This signal most likely arrives in the middle of a Time Frame. If RC_Type=IMMEDIATELY, MSC should enter the state SUSPENDED, and suspends any other operation, i.e. future FH_RCVD signals are not taken into account until the resumption of the MSC operation.
The type RC_Type takes two values: IMMEDIATELY and LATER. In this case takes the value IMMEDIATELY, i.e. the suspension should take place immediately.
SUSPEND_MSC
RC_Type
SUSPEND_MSC
RC_Type
MSC enters SUSPENDED state. From this point on any other signal but RESUME_MSC is ignored.
SUSPEND_MSC_OK
SUSPEND_MSC_OK
At this point MTC starts to search all the adjacent APs. This list has been acquired from the MPDU.
Get the first AP in the list
ACQUIRE_NEW_AP
ChannelNumberType
PHY_SELECT_CHANNEL
ChannelNumberType
T_WAIT_CHANNEL_SELECTION
Parameter CHANNEL_SWITCH_PERIOD is set provisionally to 1 ms
CHANNEL_SWITCH_PERIOD
FH_HUNT_req
see MSC MT_Power_On
RC=HUNT_SUCCESS
In case that the response is signal ACQUIRE_NEW_AP_KO we move to the next AP in the list
FH_HUNT_cnf
RC_Type, AP_MAC_AddrType
ACQUIRE_NEW_AP_OK
NEIGH_AP_LOAD_INFO_req
A place should be reserved in the FH for this parameter
MPR reads the AP load encapsulated in the FH
MEASURE_RLQ_req
Pid
PHY_MEASURE_RSSI
NEIGH_AP_LOAD_INFO_cnf
GuaranteedAggregateBwUsedType
PHY_MEASURED_RSSI
MeasuredRSSI_Type
MEASURE_RLQ_cnf
MT_RadioLinkQualityType
The process isthe same as above
Get the next AP in the list.
AP list has been exhausted. Acquire your old AP and Resume MSC operation
RESUME_MSC
RESUME_MSC
MSC resumes operation waiting for the first FH reception of its
RESUME_MSC_OK
RESUME_MSC_OK
process Handover_Initiation
MHI
RESUME_MHI
With this signal MHI resumes the process of gathering data pertaining to its associated AP so as to decide for BW_HO
ACQUIRE_NEW_AP
ChannelNumberType
Starting the acquisition of the old AP
see MSC MT_POWER_ON
ACQUIRE_NEW_AP_OK
See Note 2 for a different course of action.
STOP_HO_SEEK_INFO
RC=TIP_STARTS
MPDU_TIP
MT_MAC_Addr, RC
IMPORTANT NOTE
The MPDU_TIP (TIP_ENDS) is transmitted through the Contention period. Moreover, it is not get acked from the AP TIP Agent as the MPDU_TIP (TIP_STARTS). The succesfull transmission is guarranteed from the ARQ mechanism.
NOTE 2 STARTS
process Handover_Initiation
MHI
NOTE 2:
A different course of action could have been in the event of loosing the old AP during the attempt of returning back. In this case an AP_LOST signal is issued by MTC in place of TARGET_AP_FOUND. From this point on we enter the Forward handover scenario.
ACQUIRE_NEW_AP_KO
AP_LOST
GET_TARGET_AP
MTC chooses immediately the AP found previously and instruct PHY to switch to this new AP for forward handover.
END OF NOTE 2
see MSC FORWARD_HO


MSC Backward_HO
process Handover_Initiation
process MT_Target_Cell
process Measurment_Functions
process Generic_WDLC
process Gen_Data_Pump
Process MT_Scheduler
process MPDU_Handler_Rcv
process MPDU_Handler_Xmit
process Slot_Sequencer
PHY
T_UPDATE_HO_PARM
HO_UPDATE_PERIOD
MEASURE_RLQ_req
MPDU_ERRORS_req
CELL_ERRORS_req
PHY_MEASURE_RSSI
PHY_MEASURED_RSSI
MEASURE_RLQ_cnf
MPDU_ERRORS_cnf
CELL_ERRORS_cnf
Decision = TRUE
GET_TARGET_AP
SUSPEND_MSC
SUSPEND_MSC_OK
ACQUIRE_NEW_AP
PHY_SELECT_CHANNEL
T_WAIT_CHANNEL_SELECTION
CHANNEL_SWITCH_PERIOD
FH_HUNT_req
FH_HUNT_cnf
ACQUIRE_NEW_AP_KO
ACQUIRE_NEW_AP_OK
RESUME_MSC
RESUME_MSC_OK
TARGET_AP_FOUND
HO_ind
MT_DEASSOCIATION_req
MT_DEASSOCIATION_cnf
ASSOCIATION_req
NOTE 2 STARTS
AP_LOST
END OF NOTE 2
process Radio_Ctrl_Manager
process MT_Association_Agent
RX_STANDBY
RX_LONG_HUNT
RX_ind
RX_FH
MPDU_RCVD
FH_RCVD
see MSC MT_Power_On
see MSC MT_POWER_ON
See Note 2 for a different course of action.


MSC AP_Deassociation
WP3 MSC Scenario : AP_
This is the MASCARA Control AMA of AAA #i
process AP_Dynamic_Gen_Ctrl
process AP_MVC_Agent
process Control_Segmenting
AP Ccon
ADG
AMA #0 of AAA #i
CSE #i
process AP_Association_Agent
process Gen_MAC_Data_Pump
process AP_I_Am_Alive_Agent
AAA #i
GDP
AIA
AP_DEASSOCIATION_req
MT_ATM_AddrType
Retrieves corresponding MAC_Addr
KILL
DEASSOCIATED
MT_MAC_AddrType
AP_DEASSOCIATION_cnf
MPDU_AP_MT_DEASSOCIATION
MT_ATM_AddrType, RC_Type = DONE
MT_ATM_AddrType, MT_MAC_AddrType
For each associated AMA (i.e. MVC) except MVC 0
SubMSC Kill_MVC_Agent
End of each associated AMA (i.e. MVC) except MVC 0
This timer exists so as to give enough time to CSR and DLC to transmit the last MASCARA control message
T1
SubMSC Kill_CSR
KILL
TRAFFIC_RELEASED
MT_MAC_AddrType, MVC_ID_Type


MSC MT_Deassociation
WP3 MSC
Scenario : MT_
----
process MT_Association_Agent
process Control_Segmenting
MT Ccon
MAA
CSE
MT_DEASSOCIATION_req
MT_ATM_AddrType,
HO_Type
MPDU_MT_AP_DEASSOCIATION
MT_ATM_AddrType,
MT_MAC_AddrType
For each associated
MMA (i.e. MVC)
except MVC 0, 1 & 2
(i.e. MASCARA CONTROL,
M_CONTROL and UNI_CONTROL)
SubMSC
Kill_MVC_Agent
End of each associated
MMA (i.e. MVC)
except MVC 0, 1 & 2
MT_DEASSOCIATION_cnf
RC_Type = SUCCESS


SubMSC Kill_DLC
WP3
SubMSC scenario
Kill_DLC
author: Thorsten Schumann
date: 15.04.1997
abstract: first draft
ENV
GDL
DLX
DLR
MPR
ATM
DESTROY
MT_MAC_Addr,
MVC_Id
PID_DEREGISTERED
MT_MAC_Addr,
MVC_Id,
Pid = DLR_Pid
only in case of
ATM traffic MVC
CLOSEAAL0_req
Vpi,
Vci
KILL
KILL