Abstract
This deliverable introduces the common InterMAC language as well as its semantics. The goal of this InterMAC language is to provide a means to map performance parameters or technology dependent MAC layers on each other as well as means to evaluate whether or not a certain technology dependent MAC layer is in its current usage scenario capable of supporting a certain OMEGA QoS class. As part of the deliverable we analyse state of the art approaches for defining semantics, and explains why the common InterMAC language was defined using XML. In addition to that metrics for QoS are shortly discussed in order to show which challenges need to be faced when decisions for admitting a certain flow need to be taken. Those decisions depend on the running application, QoS class and the combination of individual hops. A mapping of OMEGA service classes onto technology dependent MAC parameters is provided to illustrate that a direct one-to-one mapping is impossible but that there exists an opportunity to provide a proper mapping. This document closes with the introduction of the InterMAC language which is illustrated using OMEGA QoS classes and technology dependent MAC layers. In addition this section provides an example for the construction of mapping rules and how these are included into the MAC description.

Keyword list
Quality of service; XML; common semantics, ontologies, Inter MAC
Executive Summary

This deliverable introduces the common InterMAC language as well as its semantics. The goal of this InterMAC language is to provide a means to map performance parameters or technology dependent MAC layers on each other as well as means to evaluate whether or not a certain technology dependent MAC layer is in its current usage scenario capable of supporting a certain OMEGA QoS class.

Some metrics for QoS are shortly discussed in order to show which challenges need to be faced when decisions for admitting a certain flow need to be taken. Those decisions depend on the running application, QoS class and the combination of individual hops. A mapping of OMEGA service classes onto technology dependent MAC parameters is provided to illustrate that a direct one-to-one mapping is impossible but that there exists an opportunity to provide a proper mapping.

As part of the deliverable we analysed state of the art approaches for defining semantics. The result of our investigation is that due to the requirements of the InterMAC with respect to processing speed, real time constraints etc. Complex ontologies will not lead to a satisfying solution. This is also influenced by the fact that the relationships between QoS requirements -even when higher layers are taken into account – and technology dependent parameters are straight forward. Most of these relationships can be expressed by mathematical equations, i.e. those equations can be used to define the semantics. In our language proposal we follow exactly this line. In order to define the language constructs we use XML which allows defining tree based documents and has the expressiveness of a BNF.

This document closes with the introduction of the InterMAC language which is illustrated using OMEGA QoS classes and technology dependent MAC layers. In addition the last section provides an example for the construction of mapping rules and how these are included into the MAC description.
# List of Authors

<table>
<thead>
<tr>
<th>First name</th>
<th>Last name</th>
<th>Beneficiary</th>
<th>Email address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Steffen</td>
<td>Hiebel</td>
<td>IHP</td>
<td><a href="mailto:hiebel@ihp-microelectronics.com">hiebel@ihp-microelectronics.com</a></td>
</tr>
<tr>
<td>Peter</td>
<td>Langendörfer</td>
<td>IHP</td>
<td><a href="mailto:langendoerfer@ihp-microelectronics.com">langendoerfer@ihp-microelectronics.com</a></td>
</tr>
<tr>
<td>Stefan</td>
<td>Nowak</td>
<td>UDO</td>
<td><a href="mailto:stefan.nowak@uni-dortmund.de">stefan.nowak@uni-dortmund.de</a></td>
</tr>
<tr>
<td>Djamal-Eddine</td>
<td>Meddour</td>
<td>Orange Labs</td>
<td><a href="mailto:djamal.meddour@orange-ftgroup.com">djamal.meddour@orange-ftgroup.com</a></td>
</tr>
<tr>
<td>Michael</td>
<td>Maaser</td>
<td>IHP</td>
<td><a href="mailto:maser@ihp-microelectronics.com">maser@ihp-microelectronics.com</a></td>
</tr>
<tr>
<td>Torsten</td>
<td>Meyer</td>
<td>Siemens</td>
<td><a href="mailto:torstenmeyer@siemens.com">torstenmeyer@siemens.com</a></td>
</tr>
</tbody>
</table>
# List of Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>BER</td>
<td>Bit Error Rate</td>
</tr>
<tr>
<td>BNF</td>
<td>Backus Naur Form</td>
</tr>
<tr>
<td>CAP</td>
<td>Controlled Access Phases</td>
</tr>
<tr>
<td>CBR</td>
<td>Constant Bit Rate</td>
</tr>
<tr>
<td>CFP</td>
<td>Contention Free Period</td>
</tr>
<tr>
<td>CP</td>
<td>Contention Period</td>
</tr>
<tr>
<td>DRP</td>
<td>Distributed reservation Protocol</td>
</tr>
<tr>
<td>DTD</td>
<td>Document Type Definition</td>
</tr>
<tr>
<td>ECMA</td>
<td>European Computer Manufacturers Association</td>
</tr>
<tr>
<td>HCCA</td>
<td>HCF Controlled Channel Access</td>
</tr>
<tr>
<td>HCF</td>
<td>Hybrid Coordinator Function</td>
</tr>
<tr>
<td>MAC</td>
<td>Medium Access Control</td>
</tr>
<tr>
<td>MPEG</td>
<td>Moving Picture Experts Group</td>
</tr>
<tr>
<td>OMEGA</td>
<td>Home Gigabit Access</td>
</tr>
<tr>
<td>OWL</td>
<td>Web Ontology Language</td>
</tr>
<tr>
<td>PCA</td>
<td>Prioritized Contention Access</td>
</tr>
<tr>
<td>PER</td>
<td>Packet Error Rate</td>
</tr>
<tr>
<td>PHY</td>
<td>Physical Layer</td>
</tr>
<tr>
<td>PLC</td>
<td>Power Line Communication</td>
</tr>
<tr>
<td>QoE</td>
<td>Quality of Experience</td>
</tr>
<tr>
<td>QoS</td>
<td>Quality of Service</td>
</tr>
<tr>
<td>RDF</td>
<td>Resource Description Framework</td>
</tr>
<tr>
<td>RDFS</td>
<td>Resource Description Framework Scheme</td>
</tr>
<tr>
<td>TXOP</td>
<td>Transmission Opportunity</td>
</tr>
<tr>
<td>URI</td>
<td>Unique Resource Identifier</td>
</tr>
<tr>
<td>UWB</td>
<td>Ultra Wide Band</td>
</tr>
<tr>
<td>VBR</td>
<td>Variable Bit Rate</td>
</tr>
<tr>
<td>VLC</td>
<td>Visible Light Communication</td>
</tr>
<tr>
<td>XML</td>
<td>eXtensible Markup Language</td>
</tr>
</tbody>
</table>
Table of contents

1 Introduction......................................................................................................... 9

2 Basic functionality of inter-MAC ................................................................. 11

3 Knowledge modelling....................................................................................... 13
  3.1 Introduction .................................................................................................. 13
  3.2 Frame Logic (FLogic) ................................................................................. 14
  3.3 The Resource Description Framework (RDF) ............................................ 14
  3.4 The RDF Vocabulary Description Language 1.0: RDF Schema (RDFS) .... 16
  3.5 The Web Ontology Language (OWL) ........................................................ 17
  3.6 Comparison between RDF/RDFS and OWL ............................................. 17
  3.7 eXtensible Markup Language (XML) ......................................................... 18

4 Description of Service Parameters ............................................................... 20
  4.1 Layered QoS model for better QoS resource management ..................... 20
  4.2 Classification of QoS metrics ................................................................... 20
  4.3 Metrics used to specify network QoS requirements ................................... 21
  4.4 QoS requirements of multimedia applications ......................................... 21
    4.4.1 Interframe vs. Intraframe relations ....................................................... 21
    4.4.2 Dynamicity of Streaming applications ................................................. 22
  4.5 Résumé ........................................................................................................ 23

5 Classification of parameters of existing MAC layers ..................................... 24
  5.1 Overview ..................................................................................................... 24
  5.2 WLAN (IEEE 802.11n)............................................................................. 26
    5.2.1 WLAN QoS MAC parameters ............................................................. 26
    5.2.2 WLAN access categories .................................................................... 26
  5.3 WPAN UWB (WiMedia, ECMA-368) ......................................................... 27
    5.3.1 WPAN UWB QoS MAC parameters .................................................. 27
    5.3.2 WPAN UWB access categories .......................................................... 28
  5.4 WPAN 60 GHz .......................................................................................... 28
    5.4.1 IEEE 802.15.3 access categories ....................................................... 29
  5.5 User priorities and traffic classes based on IEEE 802.1D ......................... 29

6 The OMEGA inter-MAC description language ............................................. 32
  6.1 Language syntax, semantics and extensibility .......................................... 32
  6.2 QoS- Class description ............................................................................. 32
  6.3 MAC protocol and mapping description ................................................. 36
  6.4 Potential Language Extension .................................................................. 42

7 Appendix ........................................................................................................ 44
  7.1 OMEGA XML Scheme for service classes ............................................. 44
  7.2 Description of OMEGA Service Classes ................................................. 47
    7.2.1 Best effort ............................................................................................ 47
    7.2.2 Conferencing ....................................................................................... 49
    7.2.3 Emergency classes ............................................................................... 52

Deliverable D5.1 - Omega Common Semantic

Page 5 (62)
7.2.4 Entertainment........................................................................................................................................ 53
7.2.5 Streaming class ..................................................................................................................................... 54
7.2.6 Other .................................................................................................................................................... 58

8 References .................................................................................................................................................. 61
List of Tables

Table 1 Relation between OMEGA service classes and QoS requirements 25
Table 2 IEEE 802.11 QoS statistics parameter 26
Table 3 User priority to access category mapping for WLAN 27
Table 4 WPAN UWB QoS statistics parameter 28
Table 5 WPAN 60 GHz QoS statistics parameter 29
Table 6 IEEE 802.1D traffic type acronym and user priority mapping 30
Table 7 Defining traffic type for different number of output queues 31
List of Figures

Figure 1: Omega Device Architecture ................................................................. 9
Figure 2: InterMAC Architecture and Information flow ..................................... 11
Figure 3: An RDF graph describing Eric Miller [22] ........................................... 15
Figure 4: The layered QoS model [17] ................................................................. 20
Figure 5: The complete trace of the MPEG-1 video “The Simpsons” consisting of 40000 frames at 25 frames/s [17] ................................................................. 23
1 Introduction

The goal of the OMEGA project is to provide basic means for easy-of-use home networking with extremely good Quality of experience, i.e. user perceived quality of service. In order to achieve this goal it is essential that the network heterogeneity as it is expected for near and mid term future can be handled by the OMEGA system. This heterogeneity stems form the use of different transmission media such as radio, optics and powerlines. In addition to this diversity a extremely broad range of applications needs to be supported, ranging from simple file download to live video conferences in HDTV quality.

The heart of this approach is the Inter MAC (InterMAC) layer, which has the following tasks

1. to mediate between technology dependent MAC approaches in order to ensure proper and automated bridging between different technologies and
2. to select paths within the OMEGA network in such a way that application dependent quality of service constraints are met.

In order to fulfill those tasks the InterMAC needs to provide a common understanding for some parameters at different layers i.e. application and MAC/PHY as well as between different MACs. This common understanding is denotes as common semantics. In addition to the physical parameters, special mechanisms and the like additional language elements for intra InterMAC communication within the control and data plane are required.

Figure 1 highlights the basic idea of the InterMAC and its integration into the OMEGA protocol stack.

This deliverable will focus on language aspects for MAC-to-MAC bridging and assessing and mapping of application demands with respect to QoS onto MAC support. Additional language features will be added as soon as the functionality of control and management plane as well as the Intra InterMAC communication are fixed in detail.

The language which is providing a common semantics or may be more concrete a technology independent understanding of QoS parameters, MAC parameters and signalling issues is defined as an XML dialect. Its syntax is fixed via an appropriate XML scheme, its semantics is defined by mapping rules which define how those language constructs are relate4d to each other. An ontology based solution using ontology approaches as they are defined and used in the Web was ruled out after thorough analysis of these approaches. The major reason was the complexity of ontologies which are designed for describing apriori unknown relations. In case of OMEGA systems all...
parameters are known in advance and need to be included by designers whenever implementing new OMEGA devices.

Structure of the deliverable is as follows. In the following section this deliverable provides a brief overview of the core components of the InterMAC to enable the reader to grasp the basic idea behind the common semantics and where it is used. Section 3 discusses related work with respect to the common semantics aspect. Here relevant approaches for defining a common language which needs to be interpreted and understood by electronic devices are introduced. In section 4 QoS parameters are introduced. Here different aspects are investigated such as layering of QoS requirements or the varying bandwidth requirements of multimedia streams. Then the Omega service classes are introduced and compared with QoS classes and related parameters of technology dependent MAC solutions such as 802.11. In section 6 our common InterMAC language is introduced. Here the syntax of the language is explained using the corresponding XML schema. The appendix provides the XML schema and some examples.
2 Basic functionality of inter-MAC

The core task of the InterMAC is guaranteeing connectivity within an OMEGA network while respecting QoS constraints. In order to fulfill this task it needs to mediate in a vertical direction from application requirements to what technology dependent MAC/PHY layers can provide. It also needs to be capable of interconnecting the different technologies which messages/flows need to traverse inside the OMEGA network and to determine a correct path form any flow entry point to its final destination.

From the above description we can derive the following functional entities inside the InterMAC:

- **QoS mapper**: this entity collects QoS requirements coming from a signaling protocol or an application and calculates what MAC/PHY parameters correspond to those QoS requirements.
- **Path selection**: a connection from a source to a sink is calculated. This can be done in advance with a proactive protocol or on demand with a reactive protocol. Path recalculations are usually triggered when the link parameters change significantly.
- **Monitoring engine**: the supervision of recent and actual link parameters is done by the monitoring engine. This information is used during path set up as well as used to indicate if paths need to be recalculated and reestablished.
- **Handover engine**: dynamic setup and teardown of links in support of dynamic recalculation of paths
- **Admission control**: decides based on actual path parameters provided by the path selection engine whether or not a certain flow may be accepted.
- **Forwarding engine** is responsible for receiving and transmitting packets between technology dependent MAC layers.

Figure 2 depicts the resulting architecture.

![Figure 2: InterMAC Architecture and Information flow](image-url)
In addition to the functional modules the InterMAC needs “communication channels” for intra InterMAC communication but also for information exchange with other parts of the OMEGA system. These communication channels are constituted by the following planes:

- The data plane is responsible of the transmissions of data packets. It consists basically, about the reception of data packets from the in-coming queue, and transfers them to the out-coming queue according to their priority and the filtering rules.

- The control plane is responsible of the establishment and the execution of the path selection process among all network elements according to the predefined QoS policies. As well as the connections configuration (setup, building, shutdown a connection)

- The management plane provides functionalities such as topology discovery and maintenance, exchange of QoS capabilities –no recently monitored measurements-, etc.
3 Knowledge modelling

In this chapter we shortly present different approaches for modelling or representing a common knowledge. In our case this knowledge needs to be shared between the technology dependent MAC layers, applications and the components of the InterMAC e.g. the admission control and path selection.

3.1 Introduction

Semantic models associate every information element with a defined meaning. That way, information is transformed into knowledge. This is an essential condition for the automatic information processing by machines. Semantic models should feature the following three characteristics to achieve this target:

A. Shared conceptualization
Every thing in a particular application area is associated with a clearly defined symbol. The conformances between things from other application areas have to be explicitly presented to merge different application areas.

B. Relationships between things
Knowledge arises from the understanding of the relationships between the things (symbols). Modelling primitives are used to describe the respective role of an individual symbol. Typically, these modelling primitives follow the object oriented paradigms. Therefore a symbol is regarded as an abstract thing (object), which describes the characteristics of other things (objects).

C. Mathematically funded semantics
Semantic models support the possibility to freely combine all modelling primitives. Therefore, it is necessary to define the semantics of these primitives clearly and mathematically correct.

Models are referred to as ontologies, if these fulfill the three characteristics above. A lot of modeling paradigms feature only the first two characteristics. Ontologies are formal models of a particular application area and support the communication between humans and/or machines. A standard definition reads as follows (Gruber, 1993): “An ontology is an explicit specification of a shared conceptualization.”

An ontology defines a catalogue of concepts as well as their relations among themselves. The vocabulary of such ontology serves for shared understanding between various system components. Therefore it is necessary that all system components interpret common vocabularies in the semantically identical meaning. Although various points of view exist within a certain system, ambiguities can be avoided. Additionally, the usage of an ontology allows tool-based checking of consistency. This as well as the automatic reasoning is done by the so called reasoner. Reasoning serves to acquire new knowledge from available information by utilization of rules (axioms). Strictly speaking, the new knowledge already exists implicitly within the ontology. The reasoner makes this information explicitly available. The way of interpretation of defined rules (e.g. by an artificial neural network) is dependent on the underlying logic and affects the quality of the used reasoner.

The modelling of knowledge by ontologies takes place on two levels: the level of concepts (comparable with the classes of an object-oriented programming language) as well as the level of instances (also referred to as facts or artefacts). The reasoning is possible on both levels, which mostly belong to a respective instanceOf relation. A particularity of ontologies is the so called concept-instance-dualism. This means that the same object can be handled as a concept within one statement and as an instance within another statement.

However, ontological concepts are not object-oriented but property-centric. Therefore, an ontological concept is not defined by attributes and methods but by its argument frame (consisting of property domain and property range). This is especially useful if the knowledge on the instance level is incomplete. A reasoner is able to reason from the fact “it has four tires” that the underlying concept can be a vehicle or a large swing, for example. Ontological concepts can inherit the properties from several super concepts at the same time.

The success of an ontology depends on the quantity of its usage. Completeness is necessary for a wide usage (and a high authority), so that all decisions within a relevant scenario can be met. Here it is useful that ontologies model only sections of the real world. Often there are several authors, which describe the same area at the same time but within separate documents. At best, these documents
complement each other. In case of redundant descriptions, the ontology has to guarantee consistency. The reasoner is responsible for the joining of such various documents. Before joining the reasoner checks the consistency of all these documents. The largest advantages of ontologies are their extensibility as well as the possibility of integration of new and even remote documents at runtime. Therefore, ontologies are best suitable for semantic modelling in distributed systems.

As described before, the performance of a reasoner depends on the underlying logic respectively on the used logic elements. The following list enumerates a selection of logics with different expressiveness, in ascending order:

a. Propositional logic
b. Description logic
c. Frame logic
d. Predicate logic I, II

The expressiveness of the used logic and the costs of the decidability of an expression are inversely proportional.

Exemplary languages for the description of ontologies are RDF, RDFS, DAML+OIL, FLogic or OWL for example. RDF, RDFS and OWL are languages of the Semantic Web and have got the state “W3C Recommendation” from the World Wide Web Consortium (W3C). OWL is the successor of DAML+OIL. In the following we will present notation forms for ontologies which cover a certain amount of spectrum in terms of reasoning ability.

### 3.2 Frame Logic (FLogic)

FLogic is an ontology modeling language with roots in object oriented databases. The language supports traditional constructs (classes, instances, relations) of object oriented models for the data structuring. A mathematically clear model theory is given for these object oriented constructs. It is possible to regard classes as instances of other classes, because all classes and instances are placed within a partial order.

An important characteristic of FLogic is the common description of instance plan and concept (class) plan. So, we can describe concepts and instances uniformly in the same language. The relations of the concept plane (“=" for single-valued, “=>” for multi-valued) are well distinguishable from the relations of the instance plan (“->” for single-valued, ”->>” for multi-valued). The FLogic syntax also supports to specify composite expressions such as

```plaintext
mary:woman[name -> "Mary"; age -> 40; friends ->> {bob, sally}];
```

Here, “mary” is declared as an instance of the concept "woman". At the same time the attribute values are defined. FLogic also supports to formulate requests about the model. FLogic uses an own query syntax instead of a common query language (like SQL). Rules without a rule head are interpreted as a request in this case. The following query delivers all sons of “mary” (result: X = {peter, thomas}):

```plaintext
FORALL X <- mary[son->>X]
```

### 3.3 The Resource Description Framework (RDF)

The Resource Description Framework is a XML-based standard for descriptions of internet and intranet resources. RDF extends the well-known XML and URI (Uniform Resource Identifier) technologies. URIs serve to identify the resources as well as to announce statements about this resources. RDF statements describe the properties of a resource with the help of certain attributes. Often, these attributes and their values are also referred to as “triples”. RDF triples consists of a subject (resource), a predicate (certain resource property) and an object (the value of this property). For human reading, the RDF triples can be displayed graphically. But for processing by machines these triples are coded in XML. The following example is taken from the RDF Model and Syntax Specification [22]:

```xml
FORALL X <- mary[son->>X]
```
Figure 6 illustrates that RDF uses URIs to identify:

- **individuals**, e.g. Eric Miller, identified by http://www.w3.org/People/EM/contact#me
- **kinds of things**, e.g. Person, identified by http://www.w3.org/2000/10/swap/pim/contact#Person
- **properties of those things**, e.g. mailbox, identified by http://www.w3.org/2000/10/swap/pim/contact#mailbox
- **values of those properties**, e.g. mailto:em@w3.org as the value of the mailbox property.

RDF also uses character strings, such as "Eric Miller", and values of other data types, such as integers and dates, as the values of properties.

RDF also provides an XML-based syntax (called RDF/XML) for recording and exchanging of those graphs. The following example is a small chunk of RDF in RDF/XML corresponding to the graph in Figure 6:

```xml
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:contact="http://www.w3.org/2000/10/swap/pim/contact#">
    <contact:Person rdf:about="http://www.w3.org/People/EM/contact#me">
        <contact:fullName>Eric Miller</contact:fullName>
        <contact:mailbox rdf:resource="mailto:em@w3.org"/>
        <contact:personalTitle>Dr.</contact:personalTitle>
    </contact:Person>
</rdf:RDF>
```

Note that RDF/XML contains URIs, as well as properties like `mailbox` and `fullName` (in an abbreviated form) together with their respective values `mailto:em@w3.org` and `Eric Miller`. RDF/XML is machine readable and, using URIs, can link pieces of information across the Web. However, unlike conventional hypertext, RDF URIs can refer to any identifiable thing, including things that may not be directly retrievable from the Web (such as the person Eric Miller). The result is, that in addition to describing things like Web pages, RDF can also describe cars, businesses, people, news events etc. Further, RDF properties themselves have URIs, to precisely identify the relationships that exist between the linked items [22].
The RDF language supplies a model as well as syntax only for resource descriptions. But RDF cannot tell us something about the meaning of such a resource. Here we need other technologies, for example RDFS or OWL.

### 3.4 The RDF Vocabulary Description Language 1.0: RDF Schema (RDFS)

RDFS is an extensible knowledge representation language, providing basic elements for the description of ontologies, intended to structure RDF resources. Its first version was published by W3C in April 1998 and the final W3C recommendation was released in February 2004. Main RDFS components are included in the more expressive language OWL.

The RDFS language extends RDF by a collection of RDF resources that can be used to describe properties of other RDF resources (including properties) in application-specific RDF vocabularies. The core vocabulary is defined in a namespace informally called “rdfs”. That namespace is identified by the URI-Reference http://www.w3.org/2000/01/rdf-schema# and is associated with the prefix “rdfs”. RDFS also uses the prefix “rdf” to refer to the RDF namespace http://www.w3.org/1999/02/22-rdf-syntax-ns# [23].

**Classes**

Resources may be divided into groups called classes. The members of a class are known as instances of the class. Classes are themselves resources. They are often identified by RDF URI References and described using RDF properties. The rdf:type property can be used to state that a resource is an instance of a class. RDF distinguishes between a class and the set of its instances. Associated with each class is a set, called its class extension, which is the set of the instances of that class. Two classes may have the same set of instances but be different classes. A class may be a member of its own class extension and hence be an instance of itself [23].

The group of resources that are RDF Schema classes is itself a class called rdfs:Class. If a class C is a subclass of a class C', then all instances of C will also be instances of C'. The rdfs:subClassOf property may be used to state that one class is a subclass of another. The term super-class is used as the inverse of subclass. If a class C' is a super-class of a class C, then all instances of C are also instances of C'[23].

All things described by RDF are called resources and are instances of the class rdfs:Resource. This is the class of everything. All other classes are subclasses of this class. rdfs:Resource is an instance of rdfs:Class. The class rdfs:Class is an instance of rdfs:Class. rdf:Property is the class of RDF properties and an instance of rdfs:Class, too. rdfs:Datatype is the class of data types. All instances of rdfs:Datatype correspond to the RDF model of a data type. Finally, the class rdfs:Literal is the class of literal values such as strings and integers [23].

**Properties**

An RDF property is described as a relation between subject resources and object resources. The rdfs:subPropertyOf property may be used to state that one property is a sub-property of another. If a property P is a sub-property of property P', then all pairs of resources, which are related by P are also related by P'. The term super-property is often used as the inverse of sub-property. If a property P is a super-property of a property P', then all pairs of resources, which are related by P are also related by P'. This specification does not define a top property that is the super-property of all properties [23].

rdfs:range is an instance of rdf:Property that is used to state that the values of a property are instances of one or more classes. The triple

\[
P \text{ rdfs:range } C
\]

states that P is an instance of the class rdf:Property, that C is an instance of the class rdfs:Class and that the resources denoted by the objects of triples whose predicate is P are instances of the class C [23].

rdfs:domain is also an instance of rdf:Property that is used to state that any resource that has a given property is an instance of one or more classes. A triple of the form

\[
P \text{ rdfs:domain } C
\]
states that $P$ is an instance of the class $\text{rdf:Property}$, that $C$ is a instance of the class $\text{rdfs:Class}$ and that the resources denoted by the subjects of triples whose predicate is $P$ are instances of the class $C$ [23].

$rdfs:\text{label}$ is an instance of $\text{rdf:Property}$ that may be used to provide a human-readable version of a resource's name. A triple of the form

$$r\text{dfs:label } L$$

states that $L$ is a human readable label for $R$ [23].

Other properties are $\text{rdf:type}$, $\text{rdfs:subClassOf}$, $\text{rdfs:subPropertyOf}$ and $\text{rdfs:comment}$. For an overview of the vocabulary of RDF, drawing together vocabulary originally defined in the RDF Model and Syntax specification with classes and properties that originate from RDF Schema refer to table “RDF Schema summary” presented in [23].

Using the Domain and Range Vocabulary

RDFS introduces an RDF vocabulary for describing the meaningful use of properties and classes in RDF data. For example, an RDF vocabulary might describe limitations on the types of values that are appropriate for some property or limitations on classes for which it makes sense to ascribe such properties. The RDF Vocabulary Description Language provides a mechanism for describing this information, but does not say whether or how an application should use it. For example, while an RDF vocabulary can assert that an author property is used to indicate resources that are instances of the class Person, it does not say whether or how an application should act in processing that range information. Different applications will use this information in different ways. Data checking tools might use this to help discovering errors in some data set, an interactive editor might suggest appropriate values and a reasoning application might use it to infer additional information from instance data [23].

### 3.5 The Web Ontology Language (OWL)

The OWL Web Ontology Language was specified by the World Wide Web Consortium (W3C) and extends RDFS with a richer vocabulary for the definition of a Semantic Web Ontologies. The RDF-based language can be used to describe the classes and relations between them that are inherent in Web documents and applications. OWL serves to design Web ontologies that are to be distributed. Such Web ontologies can be imported and augmented, creating derived ontologies. The OWL Language facilitates greater machine interpretability of Web content than that supported by XML, RDF and RDF Schema (RDFS) by providing a richer vocabulary along with formal semantics. OWL has three increasingly expressive sublanguages: OWL Lite, OWL DL and OWL Full. Which of them is to be selected depends on the complexity of the semantic model. If RDF resource descriptions are linked with an OWL ontology (everywhere in the Internet or intranet) machines are able to process this semantic information.

### 3.6 Comparison between RDF/RDFS and OWL

The main task of RDF/RDFS is the structuring of vocabulary with regard to hierarchy and class as well as property membership used for ontology description. Furthermore, it can define domains and ranges for properties. But RDFS is too weak to describe resources in sufficient detail [24]:

- RDFS doesn’t have localized range and domain constraints. So, for example, it is impossible to say that the range of $\text{hasChild}$ is $\text{person}$ when applied to persons and elephant when applied to elephants.
- RDFS doesn’t have existence or cardinality constraints. So, for example, it is impossible to assert that all instances of person have a mother that is also a person or that persons have exactly two parents.
- RDFS doesn’t have transitive, inverse or symmetrical properties. So, for example, it is impossible to state that $\text{isPartOf}$ is a transitive property that $\text{hasPart}$ is the inverse of $\text{isPartOf}$ or that $\text{touch}$ is symmetric.

The OWL language extends RDF/RDFS [24]:
- OWL extends RDF/RDFS by some very useful property attributes to better describe such a property. On the basis of the property attributes owl:Restriction, owl:allValuesFrom and owl:someValuesFrom, OWL allows a more precise description of a property.
- RDF/RDFS allows to define subclasses such as carnivores and herbivores, but does not allow to express that these classes should be disjoint. OWL holds the necessary syntax.
- In OWL we can also work with set operators, for example to build a group. RDF/RDFS does not support this feature.
- RDF/RDFS and OWL Lite support only cardinalities “0” or “1”. Therefore, in RDF/RDFS or OWL Lite it cannot be expressed that one person has two parents. OWL DL and OWL Full provide support for any cardinality.

3.7 eXtensible Markup Language (XML)

XML is a language, which is often used in Web applications and for defining a device independent structure of a certain type of documents. In the Web area it is generally used to separate the structure of a document from its design. XML documents define a tree structure. XML is known to be “verbose”. This means that the elements used to structure XML documents are in the most cases human readable words called TAGs, which provide some hint on the meaning of this element. The structure of XML documents can be defined within the document itself using a Document Type Definition (DTD). The second alternative are XML Schemas, which can be defined at any place in the web. Whether a DTD or an XML Schema is used is specified in the header of the XML document. The XML Schema allows for relatively easy extensibility of allowed XML documents. New structure can be defined at a certain location i.e. at a device provider’s site and access is provided by a URI, which is included in the document. The expressiveness of DTDs and XML Schemas is equivalent to Backus Naur Forms (BNF). So, regular expressions can be defined. Whether or not a certain document is correct with respect to a DTD or XML Schema can be validated by a specially designed parser. For development of such parsers significant tool support, e.g. from Apache, is available.

Verifying the correctness of an XML document can be done very efficiently, despite the XML tags are requiring more memory than binary TAGs for example. In [34] the authors have shown that XML based Web services can be run at mobile devices of the PDA class. This clearly indicates that XML can be used even on small mobile devices. In the same publication the authors also discuss measurements, which show that the security means supported by XML such as digital signatures can be used on mobile devices as well.

The tree structure introduced by the XML document allows to include mathematical formulas into XML documents. By that, mapping rules between QoS parameters of OMEGA service classes and MAC parameters can be included into the XML descriptions. The former and the fact that in a certain way XML documents are self explaining to human beings, easy processable even by devices with scarce resources, provide means for supporting security and allow for easy extensibility make it a good candidate for defining the InterMAC common language.

3.8 Résumé

The 802.21 working group addresses issues similar to those the OMEGA project is dealing with in its InterMAC approach. The OMEGA approach goes a step beyond what is discussed in 802.21 since it addresses also the issues of mapping higher layers into technology dependent MAC layers. It incorporates also non-IEEE MAC protocols such as the PLC approach.

The InterMAC common language shall provide a means to mediate between various technology dependent MAC solutions as well as the QoS requirements of higher layers and applications. Essential for fulfilling such task is that different definitions of service requirements can be mapped onto parameters, which can be interpreted by all involved parties. Basically such relationship between parameters can be described by ontologies. Ontologies are however designed to allow for checking of the semantics of previously unknown entities. This provides ontologies with a supreme expressiveness. On the other hand the explanatory power of ontologies makes it difficult to use them on mobile devices. I.e. the processing overhead will exhaust the batteries of mobile devices and response time will be extremely long. The major issue here is that the real meaning of items needs to be verified without involving a human being. In the InterMAC common language merely straight
forward relationships, such as for example the one between the current bit error rate and bandwidth to
the current packet error rate are considered. These relationships do not need the expressiveness of
an ontology, but can be defined by rather simple algebraic expressions. XML allows the integration of
such expressions into the definition of OMEGA service classes and of technology dependent MAC
layers. In addition, extending the OMEGA architecture by a new technology dependent MAC layer
requires that a human being defines parameters of the MAC layer as well as their relationships. Since
these mapping rules can be included in an XML Schema and since XML Schemas can easily be
extended, we see XML as the prime choice to realize the InterMAC common language.
4 Description of Service Parameters

In this section we will discuss factors which influence the Quality of Experience i.e. varying bandwidth requirements of multimedia streams, types of QoS metrics which need to be considered when realizing the path selection algorithms and the relationships between application layer QoS requirements and MAC/PHY parameters.

4.1 Layered QoS model for better QoS resource management

While the network QoS requirements are only bandwidth, maximum transfer delay, maximum loss rate and maximum jitter, applications have got their own requirements or QoS dimensions. For example, the QoS dimensions of a video transmission are picture format, colour depth, frames per second and end-to-end delay or sampling rate, bits per sample and channels for an audio transmission. For the user of this application, the only requirement is that video or audio transmission looks or sounds good. This is called the perceptual quality or user satisfaction. These different requirements lead to a layered QoS model (see figure 4). A mapping is required between the dimensions of each layer [17].

User QoS: user satisfaction (video or audio transmission looks or sounds good)

Application QoS:

Video (frame rate, picture size, colours)

Audio (sampling rate, bits, channels)

Network QoS: min. bandwidth, max. delay, max. loss rate, max. jitter

Figure 4: The layered QoS model [17].

The InterMAC common semantics approach will take this layered QoS model into account. By this we mean it will define relations between the different QoS classes define within the OMEGA project and ensure that these QoS requirements can be mapped on parameters, provided by technology dependent MAC layers. So we provide a means to decide which flow may be admitted and which not depending on their QoS requirements.

4.2 Classification of QoS metrics

QoS metrics can be classified as additive, concave or multiplicative metrics, based on their mathematical properties. Additive metrics are defined as \( \sum_{i=1}^{n} L_i(m) \) over path P of length n, where \( L_i(m) \) is the value of the metric m over link \( L_i \) and \( L_i \in P \). The value of a concave metric is defined as the minimum value of that metric over a path i.e. \( \min(L_i(m)) \). Finally, a multiplicative metric is calculated by taking the product of the values along a path i.e. \( \prod_{i=1}^{n} L_i(m) \). Thus, end-to-end delay for example, is an additive metric. Available channel capacity is a concave metric; here we are only interested in the bottleneck (the minimum value on the path). Finally, path reliability is a multiplicative metric, since the reliabilities of each link in the path must be multiplied together to compute the chance of delivering the packet.

Understanding the type of a certain QoS metric and especially to which category a certain parameter belongs helps on the one hand to formulate correct mapping rules for the InterMAC and provides on
the other hand a proper means to describe how the common semantics can be extended whenever needed.

### 4.3 Metrics used to specify network QoS requirements

For multimedia streams, network guarantees for the following four QoS (quality of service) requirements are necessary:

**A. Minimum required bandwidth (bps)**

Multimedia streams have got at least a minimum bandwidth requirement, e.g. this is the lowest possible resolution and frame rate of a video. Bandwidth reservation is necessary therefore.

**B. Maximum tolerable end-to-end (source to destination) data packet transfer delay (s)**

In phone calls for example, transfer delays of several seconds are unacceptable. Therefore, a given maximum delay should be ensured.

**C. Maximum tolerable delay jitter (s)**

For example in audio or video conferences having a very low delay requirement, buffering of the incoming packets is not possible. In this case, a jitter limit is required. Various definitions of the jitter are possible. In the RTP specification (RFC 1889), the jitter is defined as follows: Let \( R_n \) be the RTP time stamp and \( S_n \) the arrival time stamp of packet. Then, the inter arrival jitter is [17]:

\[
\text{Jitter}_{\text{new}} = \text{Jitter}_{\text{old}} + \frac{1}{16} \left( D_{i-1,i} - \text{Jitter}_{\text{old}} \right); \text{ where } D_{i,j} = (R_j - R_i) - (S_j - S_i).
\]

Of course, both timestamps have to be given in the same units, e.g. microseconds.

**D. Maximum tolerable packet loss ratio (%)**

In a video transmission, a loss of e.g. 20% of the transmitted packets may be acceptable due to interpolation of the missing picture parts. The same loss for an audio transmission may e.g. cause unacceptable noise. An upper limit for the maximum loss rate is therefore necessary.

These QoS parameters can at least to a certain extend be measured at the physical layer. By that they are the basis for determining whether or not an application can be served with the QoS it requires.

### 4.4 QoS requirements of multimedia applications

#### 4.4.1 Interframe vs. Intraframe relations

The application QoS of a multimedia stream consists of description of both the qualities of individual media within the stream (media qualities), and the way in which these media are combined in a multimedia stream (media relations). The media quality component consists of an interframe specification and an intraframe specification. The media relations specify relations among the single media streams of a multimedia stream.

The interframe specification gives the characteristics of a homogeneous media stream (e.g., sample rate, sample size, loss tolerance). The intraframe specification of a video transmission can be defined by frame rate, frame height, frame width and colour depth for example.

For the user (user QoS) of an application, the only requirement is that video or audio transmission looks or sounds good (user satisfaction).

From the InterMAC perspective the interframe requirements are of higher significance since those are the only requirements the InterMAC can handle and obey individually. Intraframe requirements merely translate into bandwidth which then also handled by the InterMAC.
4.4.2 Dynamicity of Streaming applications

The dynamicity of streams has a significant influence on the guarantees for QoS. If the maximal load is used to reserve resources the network will be underutilized leading to reduced number of admitted applications. If those peaks in the load are not considered there is a certain probability that the QoS cannot be satisfied if the peak load occurs. The following paragraphs are discussing issues of variable loads.

Constant bitrate (CBR) vs. variable bitrate (VBR)

For constant bitrate (CBR) traffic, it is only necessary to reserve bandwidth once - the requirements do not change. But for variable bitrate (VBR) traffic, it is not acceptable to reserve the peak rate, since e.g. in an MPEG video it is often 10 or more times larger than the average rate. Therefore, a regular update of the bandwidth mappings is necessary. For already completely stored medias like video and audio files, it is obvious to do an a priori analysis of their transport properties. This information can be used to do the remapping as efficient as possible. Especially, this includes the usage of buffering to reduce bandwidth requirements and therefore the transport cost. Buffering can save a lot of bandwidth allowing more customers to use a link simultaneously, especially if there is a large variance in the frame sizes. The tightest traffic constraint function is the empirical envelope, in other words the maximum number of bytes sent in any interval of length \( t \) (see [19]).

Traffic shaping

Most traffic is sent in bursts, e.g. a video at a frame rate of 10 frames/s will send a sequence of usually many packets every \( 1/10^{th} \) second. Since the router’s queues are limited and especially, they can be overflow by such a burst of packets. Therefore, it is necessary for the sender to smooth the packet rate by buffering the packets: For example, instead of sending 10 packets every \( 1/10 \)s second, send 2 packets every \( 1/50^{th} \) second [17].

Burstiness

A measurement value for the variance of frame sizes is the so called burstiness of a stream:

\[
\text{Burstiness} = \frac{\text{Peak Rate}}{\text{Average Rate}}
\]

But it is important to denote, that buffering is only realistic for delays of up to a few seconds. Short-term burstiness in the scope of some frames can easily be smoothed by buffering. Higher delays are usually not acceptable for the users and also require large buffers at the clients. Therefore, an additional concept is necessary to handle long-term burstiness in the scope of minutes or hours because smoothing by buffering would result in unacceptable delays of many seconds here.

Maximum buffer delay

Multimedia streams have constraints for the maximum transfer delay. Therefore, it is necessary to police the stream’s traffic. That is, checking whether the delay introduced by the buffering is below a given maximum buffer delay. The delay constraint is violated, if

\[
\text{BufferSize} > \text{MaxBufferDelay} \times \text{OutputRate}
\]

This means, that if there are more bytes in the buffer than can be sent during the maximum allowed time, the delay constraint is violated. In this case, a buffer flush is required, that is emptying the whole buffer by dropping all contents inside [17].

Bandwidth remapping

Using one of the traffic models described in [20], it is possible to give a traffic constraint for a stream. As mentioned, short-term burstiness can be smoothed by buffering. Now, it is necessary to examine long-term burstiness. Having a look at the figure 5, there is a large peak of about 240,000 bytes at about frame 37,000, marked by the arrow. Calculating a traffic description using the D-BIND traffic model as described in [20], it is necessary to allocate a bandwidth near the peak rate for very low buffer delays. But since the other peaks are all below 150,000 bytes, a lot of bandwidth would be wasted. This is called over-provisioning. The same effect also can also be found for the higher and lower areas of about 500 to 1000 frames. These are scenes of lower or higher bandwidth requirement causing the long-term burstiness. In this case, even a high buffer delay of up to a few seconds would not be able to improve this over-provisioning [17].

To cope with this problem, the trace can be partitioned into several intervals. For each interval, an own traffic description is calculated and used. This results in parts having lower bandwidth requirements.
allocating less bandwidth and vice versa. Now, a bandwidth remapping is necessary at the interval borders. Therefore, these intervals are called remapping intervals. Now, an algorithm to calculate the interval sizes is required which generates intervals of the lowest cost. Such an algorithm has been developed in [21].

![Graph](image)

**Figure 5:** The complete trace of the MPEG-1 video “The Simpsons” consisting of 40000 frames at 25 frames/s [17].

### 4.5 Résumé

The issues discussed in this section have some impact on the design and realization of the InterMAC especially for components such as admission control. They also will influence the design of the InterMAC data repository. The InterMAC design will also follow the idea of the layered QoS model which also had some influence on the design of the InterMAC common language. Challenges which arise from varying requirements within a single media stream have not yet really influenced the design of the InterMAC common language. This is on the one hand due to the early stage of the OMEGA project, but on the other hand some issues such as burtsiness of a certain flow is not reflected in the OMEGA service classes and by that cannot be handled by the InterMAC. From a research point of view it would be interesting to investigate whether or not such phenomena can be taken into account. But it might well be beyond the scope of this project since an online verification of those parameters would require media analysis tools, which are not available. Reacting on those variations would in addition require extremely good realtime behaviour of the InterMAC which is currently not considered.
5 Classification of parameters of existing MAC layers

5.1 Overview

Within the OMEGA project several state-of-the-art transmission technologies are considered to be deployed in a Home Area Network (HAN). Moreover, it is possible that a single consumer device operating in the HAN has integrated a certain number of these technologies. In this case the appropriate transmission technology has to be chosen to the benefit of the Quality of Experience (QoE) for the user. The QoE is strongly connected to certain Quality of Service (QoS) constraints which have to be met at the technology level. Regarding QoS the different technologies have to be used differently in order to meet these constraints.

The technologies which are considered so far as OMEGA technologies are:

- WLAN (IEEE 802.11n)
- WPAN UWB (WiMedia, ECMA-368)
- WPAN 60 GHz
- Power Line Communication (PLC)
- Hybrid Wireless Optics: Infrared (IR), Visible Light Communication (VLC)

All technologies are characterized by a different grade of maturity and provide a MAC layer that depends on the corresponding technology. The Inter-MAC layer will therefore provide the convergence mechanisms in order to support technology independent services at higher layers. Within OMEGA several usage scenarios have been investigated and documented [42].

In total, six OMEGA service classes have been elaborated which cover a wide range of home applications. Each service class is associated with four basic QoS requirements:

- Throughput
- Delay
- Jitter
- Packet loss

The relation between these parameters and the OMEGA service classes is shown in Table 5.1.

<table>
<thead>
<tr>
<th>Service Class</th>
<th>Average throughput; Peak rate</th>
<th>Delay</th>
<th>Jitter</th>
<th>Packet loss</th>
</tr>
</thead>
<tbody>
<tr>
<td>Emergency</td>
<td>Some kbps (max)</td>
<td>Should be real time</td>
<td>Critical criterion</td>
<td></td>
</tr>
<tr>
<td>Conferencing</td>
<td>2 CBR flows, some 10 kbps</td>
<td>Real time constraint &lt; 10 to 100 ms</td>
<td>&lt;20 ms</td>
<td>&lt;10^-3</td>
</tr>
<tr>
<td></td>
<td>2 flows from 128 kbps to 4 Mbps</td>
<td>20 to 30 ms</td>
<td>&lt;10 ms</td>
<td>&lt;10^-5</td>
</tr>
<tr>
<td>Streaming</td>
<td>CBR flow, some kbps for audio applications, from 2 to 50 Mbps or 600 Mbps in the future for video applications</td>
<td>Video-audio sync. constraints &lt;400 ms</td>
<td>&lt;10^-5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt; 50Kbps</td>
<td>Audio-3D sync. &lt;300ms</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Less Sensitive Traffic</td>
<td>Flow &lt; 100 kbps</td>
<td>&lt;50 ms</td>
<td>&lt;1 ms</td>
<td>&lt;10^-3</td>
</tr>
<tr>
<td>Best Efforts</td>
<td>Variable from some kbps to 1.5 Mbps</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>------------------------------</td>
<td>-------------------------------------</td>
<td>---</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>VBR flow up to the Gbps</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Other</td>
<td>Mono-directional flow 1Gbps or more</td>
<td>&lt;400 ms</td>
<td>&lt;1 ms</td>
<td></td>
</tr>
</tbody>
</table>

Table 1 Relation between OMEGA service classes and QoS requirements

Each transmission technology has different capabilities to cope with these QoS constraints. For example, we will see that WLAN solely provides only four access categories when considering IEEE 802.11e EDCA. The other technologies show similar restrictions. This emphasizes the mapping problem the Inter-MAC has to solve which is visualized in Figure 5.1. It shows the six OMEGA service classes which are associated with certain QoS constraints. The traffic of these classes converges at the Inter-MAC level. The Inter-MAC in turn now has the task to choose an appropriate transmission technology. Each technology is represented by a different MAC layer which is characterized by a variable number of access categories as well as by inherent QoS MAC parameters. These QoS MAC parameters of course differ from one technology to another.

Therefore, a simple mapping between OMEGA service classes and scheduling classes/access categories on MAC level is not sufficient. Instead, an intelligent mapping has to be introduced that directly takes into account the QoS parameter on MAC level. Since every technology is specified by its own standard, the following sections present the available standards and describe the accessible parameters which help to map the service classes to the OMEGA technologies considering the QoS constraints.
5.2 WLAN (IEEE 802.11n)

5.2.1 WLAN QoS MAC parameters

Table 5.2 provides a mapping of the OMEGA QoS parameters to the technology dependent MAC parameters. The table lists measurements defined in IEEE 802.11 that may be used for computing the QoS metrics needed for the Inter-MAC. The set of QoS parameters can be based on an individual station measurement, since this is a medium using a distributed access method.

<table>
<thead>
<tr>
<th>OMEGA QoS parameter</th>
<th>Related IEEE 802.11 parameter</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average/peak bit rate (throughput)</td>
<td>not supported</td>
<td></td>
</tr>
<tr>
<td>Packet loss (PER)</td>
<td>TransmittedFragmentCount MulticastTransmittedFrameCount FailedCount ReceivedFragmentCount MulticastReceivedFrameCount FCSErrorCount TransmittedFrameCount</td>
<td>STA statistics report</td>
</tr>
<tr>
<td></td>
<td>Retry Count MultipleRetryCount FrameDuplicateCount RTSSuccessCount RTSFailureCount</td>
<td></td>
</tr>
<tr>
<td>Delay</td>
<td>Transmit Delay Histogram Average Transmit Delay</td>
<td>Transmit Stream/Category Measurement Report (only specific to STA)</td>
</tr>
<tr>
<td>Jitter</td>
<td>Transmit Delay Histogram Average Transmit Delay</td>
<td></td>
</tr>
</tbody>
</table>

Table 2 IEEE 802.11 QoS statistics parameter

5.2.2 WLAN access categories

WLAN based on IEEE 802.11 provides mechanisms which support QoS. The standard amendment IEEE 802.11e introduces the Hybrid Coordination Function (HCF) which offers two channel access methods: HCF Controlled Channel Access (HCCA) and Enhanced Distributed Channel Access (EDCA). Both define Traffic Classes (TC) representing different medium access priorities.

EDCA offers four access categories:
- Voice (highest priority)
- Video
- Background
- Best Effort (lowest priority)

Traffic with a higher priority has a higher chance of being sent, whereas traffic with a lower priority is more unlikely to be sent. An additional feature that can be used to extend the channel access is the Transmit Opportunity (TXOP). A TXOP is a limited time interval during which a station may send as many frames as possible. On the one hand, time consuming contention for the medium can be reduced, especially when low rate stations gain a large amount of channel time regarding DCF. On the other hand channel access time for each station can be optimized in order to meet the real needs for a certain transmission.

Besides a Contention Period (CP) and a Contention Free Period (CFP) between two beacon frames in the Point Coordination Function, the HCF adds Controlled Access Phases (CAP) which are CFPs that can be initiated at almost anytime during a CP. HCCA offers eight traffic classes:
- 2 x Voice
With HCCA, QoS can be configured with good precision, but it is not mandatory and therefore very often not implemented in current WLAN products. Further features which can enhance the QoS are Block Acknowledgement, QoS Acknowledgement policies and Direct Link Setup.

For the mapping of incoming traffic to the access categories, the user priority is evaluated which represents an integer value between 0 and 7. The mapping is based on IEEE 802.1D priorities.

<table>
<thead>
<tr>
<th>Priority</th>
<th>User priority (Same as 802.1D user priority)</th>
<th>802.1D designation</th>
<th>Access Category</th>
<th>Designation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lowest</td>
<td>1</td>
<td>BK</td>
<td>AC_BK</td>
<td>Background</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>---</td>
<td>AC_BK</td>
<td>Background</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>BE</td>
<td>AC_BE</td>
<td>Best Effort</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>EE</td>
<td>AC_BE</td>
<td>Best Effort</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>CL</td>
<td>AC_VI</td>
<td>Video</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>VI</td>
<td>AC_VI</td>
<td>Video</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>VO</td>
<td>AC_VO</td>
<td>Voice</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>NC</td>
<td>AC_VO</td>
<td>Voice</td>
</tr>
</tbody>
</table>

Table 3 User priority to access category mapping for WLAN

5.3 WPAN UWB (WiMedia, ECMA-368)

5.3.1 WPAN UWB QoS MAC parameters

The WiMedia MAC is defined in ECMA-368 and provides two channel access mechanisms: Distributed Reservation Protocol (DRP) and Prioritized Contention Access (PCA). The latter one is based on the EDCA in WLAN systems. In contrast to the contention based approach in PCA, the DRP offers the possibility for stations to reserve a certain time on the medium. Consequently this enables isochronous data transfers with constant data rates. Nevertheless, one has to consider the wireless channel which always introduces an uncertainty for a successful transmission.

The following table provides the mappings of the OMEGA QoS parameters to the WiMedia MAC parameters which are defined in ECMA-368. Note that one has to differ between PCA and DRP.

<table>
<thead>
<tr>
<th>OMEGA QoS parameter</th>
<th>Related WiMedia parameters</th>
<th>Note</th>
</tr>
</thead>
</table>

Deliverable D5.1 - Omega Common Semantic
Average/peak bit rate (throughput) | DRP: MinBW DesiredBW AvailableBW | The MinBW and DesiredBW parameters define the throughput requirements for the reservation. The AvailableBW parameter indicates the possible throughput after a successful reservation.

Packet loss (PER) | RetryCount ReceiveErrorInfo | The RetryCount parameter remembers the number of retransmission for frame. ReceiveErrorInfo provides additional information for a receive error.

Delay | TransmitTimeout | Specifies the amount of time in milliseconds in which the MSDU needs to be successfully sent (requirement).

Jitter | not specified | not specified

Table 4 WPAN UWB QoS statistics parameter

5.3.2 WPAN UWB access categories

The PCA mechanism provides differentiated, distributed contention access to the medium for four access which is based on IEEE 802.11e. A device employs a prioritized contention procedure for each access category to obtain a transmission opportunity (TXOP). The mapping of user priorities to access categories can also be obtained from Table 5.2, since it is based on IEEE 802.1D priorities as well.

In contrast to PCA, the DRP may provide the chance for constant data rates. Whenever resources are requested by a station apart from the desired throughput a QoS goal can be specified. ECMA-368 defines three QoS goals for a connection which should be served by a DRP reservation:

- Premium
- High
- Best Effort

These parameters can be mapped to target PER and margins (SNR, reservation time) for a connection. Therefore, it can help to distinguish connections with different QoS constraints.

5.4 WPAN 60 GHz

There are several standardization activities that concentrate on WPAN 60 GHz communication:

- IEEE 802.15.3c: WPAN Millimetre Wave Alternative PHY
- ECMA TC48: High Rate 60 GHz PHY, MAC and HDMI PAL
- IEEE 802.11 VHT60: Very High Throughput
- Wireless HD

Each of these activities considers a different MAC protocol.

Wireless HD is made up by a consortium with members from the consumer electronics industry that has developed a specification which has not been adopted by a standardization body yet. Consequently Wireless HD is a proprietary solution and will therefore not be considered any further in this document. Besides, the specification concentrates on peer-to-peer connections between CE devices which are not in the focus of OMEGA.

IEEE 802.11 VHT will make use of the IEEE 802.11 standard MAC protocol, which has already been addressed in section 5.2. Simultaneously, ECMA TC48 is working on a 60 GHz PHY and MAC standard. The MAC protocol will be based on ECMA-368 with some slight modifications, e.g. MAC
support for directive antennas. The accessible QoS MAC parameters have been addressed in section 5.3.

Consequently, we have now to concentrate on IEEE 802.15.3c. The MAC layer will be based on the IEEE 802.15.3 MAC layer. The following table summarizes the accessible MAC parameters for the IEEE 802.15.3 MAC.

<table>
<thead>
<tr>
<th>OMEGA QoS parameter</th>
<th>Related IEEE 802.11 parameter</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average/peak bit rate (throughput)</td>
<td>Not currently supported. RXFrameCount CTARateType CTARateFactor MinNumTUs</td>
<td>Throughput can be calculated by means of RXFrameCount and measurement duration.</td>
</tr>
<tr>
<td>Packet loss (PER)</td>
<td>RXFrameCount RXFrameCountError RXLossCount</td>
<td>The parameters can be requested by a Channel status request.</td>
</tr>
<tr>
<td>Delay</td>
<td>TransmissionTimeout</td>
<td>For setting the MinimumDelay requirement, the IEEE 802.1D priorities can be used: 5: &lt; 100 ms delay and jitter (Video, isochronous) 6: &lt; 10 ms delay and jitter (Voice, isochronous)</td>
</tr>
<tr>
<td>Jitter</td>
<td>Not currently supported.</td>
<td>For setting the MinimumJitter requirement, the IEEE 802.1D priorities can be used: 5: &lt; 100 ms delay and jitter (Video, isochronous) 6: &lt; 10 ms delay and jitter (Voice, isochronous)</td>
</tr>
</tbody>
</table>

Table 5 WPAN 60 GHz QoS statistics parameter

5.4.1 IEEE 802.15.3 access categories

There are currently no traffic classes or access categories defined in IEEE 802.15.3 as it was done for WLAN or ECMA-368. Instead, beside Best Effort traffic, which means that all packets are handled the same without any guarantees for QoS, it is recommended to make use of the hierarchical IEEE 802.1D priority scheme in order to differentiate packets or streams. The IEEE 802.15.3 MAC supports isochronous as well as asynchronous data transfers.

5.5 User priorities and traffic classes based on IEEE 802.1D

Because of the fact that most communication standards for home networking have directly been elaborated by the IEEE or are at least strongly related to IEEE standards (see ECMA-368) the user priority scheme based on IEEE 802.1D is often used to differentiate data packets or streams. Therefore, this section provides a closer on the user priorities, the traffic classes which are defined and the mapping between them.

IEEE 802.1D is a standard for MAC bridges. In order to support integrated services a reasonable and low complexity mapping between user priorities and traffic classes has been introduced. The basic idea behind user priorities is to use a set of output queues which are filled with data corresponding to its priority. These queues in turn have different priorities to access the medium. If data coming from different applications is sorted according to their “importance” or “urgency”, the output queuing will handle the data according to their priority.

In total seven traffic types have been elaborated:

1. Network Control (NC)
2. Voice (VO)
3. Video (VI)
4. Controlled Load (CL)
5. Excellent Effort (EE)
6. Best Effort (BE)
7. Background (BK)

The value for a user priority therefore is represented by one out of seven numbers. Of course, the QoS mapping problem is too complex to be represented by such a simple number, but it serves as a pragmatic means.

The IEEE 802.1D priority scheme is supposed to manage

- latency
- throughput

in order to meet application or traffic specific constraints. Considering only a few traffic classes the focus is on meeting the latency requirement, whereas for a larger number of traffic classes the focus shifts to manage throughput.

<table>
<thead>
<tr>
<th>User Priority</th>
<th>Acronym</th>
<th>Traffic Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>BK</td>
<td>Background</td>
</tr>
<tr>
<td>2</td>
<td>---</td>
<td>Spare</td>
</tr>
<tr>
<td>0 (Default)</td>
<td>BE</td>
<td>Best Effort</td>
</tr>
<tr>
<td>3</td>
<td>EE</td>
<td>Excellent Effort</td>
</tr>
<tr>
<td>4</td>
<td>CL</td>
<td>Controlled Load</td>
</tr>
<tr>
<td>5</td>
<td>VI</td>
<td>Video &lt; 100 ms latency and jitter</td>
</tr>
<tr>
<td>6</td>
<td>VO</td>
<td>Voice &lt; 10 ms latency and jitter</td>
</tr>
<tr>
<td>7</td>
<td>NC</td>
<td>Network Control</td>
</tr>
</tbody>
</table>

Table 6 IEEE 802.1D traffic type acronym and user priority mapping

Because of historical reasons the number 0 represents the most common traffic type today which is of course Best Effort. Therefore, it is used as the default value. Unfortunately, the priorities 1 (Background) and 2 (spare) indicate a lower priority than 0 which should not lead to any confusion.

Depending on the number of queues that are available for data classification or equivalently the mechanisms for distinction of traffic at all, the mapping in Table 5.6 is recommended. It addresses latency for a small number of queues. For an increasing number of queues a distinction regarding throughput is realized.
<table>
<thead>
<tr>
<th>Number Of Queues</th>
<th>Defining Traffic Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>BE</td>
</tr>
<tr>
<td>2</td>
<td>BE</td>
</tr>
<tr>
<td>3</td>
<td>BE</td>
</tr>
<tr>
<td>4</td>
<td>BK</td>
</tr>
<tr>
<td>5</td>
<td>BK</td>
</tr>
<tr>
<td>6</td>
<td>BK</td>
</tr>
<tr>
<td>7</td>
<td>BK</td>
</tr>
<tr>
<td>8</td>
<td>BK</td>
</tr>
</tbody>
</table>

Table 7 Defining traffic type for different number of output queues
6 The OMEGA inter-MAC description language

In this section the OMEGA InterMAC description language is introduced. Our InterMAC description language provides means to describe service classes, MAC description and algorithms.

6.1 Language syntax, semantics and extensibility

The language syntax is defined by using an XML scheme. This allows for easy parsing to check correct description of classes. The format checking of XML documents can be easily done by existing XML parsers and a given DTD or schema. This is especially true for enforcing the definition of all relevant parameters of certain service class. At least all parameters need to be included. Whether or not meaningful data is provided or not, is out of the scope of the syntax. With having one XML document for each service class OMEGA InterMAC is enabled to introduce further service classes later on if required. Since the values for a certain service class or the number of classes might change the XML documents might not be fixed over runtime.

The definition of the syntax is done using XML schema since it is more expressive than DTD.

The language semantics is defined by the mapping rules which define the relationships between service parameters as defined in service classes and the QoS relevant parameters as they are provided by the technology dependent MAC layers. For example the packet error rate at the application layer can be calculated from the bit error rate as measured at the MAC and the length of application layer packets. These mapping rules are integral part of the MAC descriptions. By that extending the InterMAC functionality to an additional MAC layer requires nothing more that providing an appropriate MAC description and making it available to the I.MAC instances.

6.2 QoS- Class description

The service classes determined and enumerated for use in OMEGA have to be described in a machine readable format to enable automatic decision of the InterMAC whether a particular MAC protocol underneath is capable of fulfilling the requirements of a given service class. The description of service classes consists of a service class denominator and a list of attributes which characterize the QoS requirements of the respective class. The values given in those attributes denote the concrete requirements of a given application belonging to that class. To ensure the documents being well-formed for processing in OMEGA InterMAC an XML schema (Listing 1) is created for the service classes.

Listing 1 - Schema for description of OMEGA service classes

```xml
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:simpleType name="ServiceClassNameT">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Emergency"/>
      <xs:enumeration value="Conferencing"/>
      <xs:enumeration value="Streaming"/>
      <xs:enumeration value="Entertainment"/>
      <xs:enumeration value="BestEffort"/>
      <xs:enumeration value="Other"/>
    </xs:restriction>
  </xs:simpleType>

  <!-- the root element of service class description pattern -->
  <xs:complexType name="OMEGA_ServiceClassDescriptionPattern">
    <xs:sequence>
      <!-- the unique name of an OMEGA service class -->
      <xs:element name="ServiceClassName" type="ServiceClassNameT"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```
  <xs:element name="Comment" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
</xs:complexType>
</xs:element>
</xs:complexType>

<xs:complexType name="StreamsDescriptorT">
  <xs:sequence>
    <xs:element name="StreamsDescriptor" type="StreamsDescriptorT" minOccurs="1" maxOccurs="5"/>
  </xs:sequence>
</xs:complexType>

<xs:simpleType name="DegreeOfSymmetryChoiceT">
  <xs:restriction base="xs:string">
    <xs:enumeration value="OneWay"/>
    <xs:enumeration value="TwoWay"/>
    <xs:enumeration value="TwoWayAsymmetric"/>
    <xs:enumeration value="null"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="StreamTypeT">
  <xs:restriction base="xs:string">
    <xs:enumeration value="CBR"/>
    <xs:enumeration value="VBR"/>
    <xs:enumeration value="null"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="StreamNumberT">
  <xs:restriction base="xs:integer">
    <xs:minInclusive value="1"/>
    <xs:maxInclusive value="5"/>
  </xs:restriction>
</xs:simpleType>

<xs:complexType name="QoSParamT">
  <xs:sequence>
    <xs:choice>
      <xs:sequence>
        <xs:element name="MinNrVal" type="xs:double" minOccurs="1" maxOccurs="1"/>
        <xs:element name="MaxNrVal" type="xs:double" minOccurs="0" maxOccurs="1"/>
      </xs:sequence>
      <xs:sequence>
        <xs:element name="MinNrVal" type="xs:double" minOccurs="1" maxOccurs="1"/>
        <xs:element name="MaxNrVal" type="xs:double" minOccurs="0" maxOccurs="1"/>
      </xs:sequence>
      <xs:sequence>
        <xs:element name="MinNrVal" type="xs:double" minOccurs="1" maxOccurs="1"/>
        <xs:element name="MaxNrVal" type="xs:double" minOccurs="0" maxOccurs="1"/>
      </xs:sequence>
      <xs:sequence>
        <xs:element name="MinNrVal" type="xs:double" minOccurs="1" maxOccurs="1"/>
        <xs:element name="MaxNrVal" type="xs:double" minOccurs="0" maxOccurs="1"/>
      </xs:sequence>
      <xs:sequence>
        <xs:element name="MinNrVal" type="xs:double" minOccurs="1" maxOccurs="1"/>
        <xs:element name="MaxNrVal" type="xs:double" minOccurs="0" maxOccurs="1"/>
      </xs:sequence>
    </xs:choice>
  </xs:sequence>
</xs:complexType>

<xs:element name="ApplicationT" type="ApplicationT" minOccurs="0" maxOccurs="1"/>
</xs:complexType>

<xs:element name="Comment" type="xs:string" minOccurs="0" maxOccurs="1"/>
</xs:complexType>

<xs:complexType name="StreamsDescriptorT">
  <xs:sequence>
    <xs:element name="DegreeOfSymmetry" type="DegreeOfSymmetryChoiceT" minOccurs="1" maxOccurs="1"/>
    <xs:element name="StreamType" type="StreamTypeT" minOccurs="1" maxOccurs="1"/>
    <xs:element name="StreamNumber" type="StreamNumberT" minOccurs="1" maxOccurs="1"/>
    <xs:element name="Comment" type="xs:string" minOccurs="0" maxOccurs="1"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="QoSParamT">
  <xs:sequence>
    <xs:choice>
      <xs:sequence>
        <xs:element name="MinNrVal" type="xs:double" minOccurs="1" maxOccurs="1"/>
        <xs:element name="MaxNrVal" type="xs:double" minOccurs="0" maxOccurs="1"/>
      </xs:sequence>
      <xs:sequence>
        <xs:element name="MinNrVal" type="xs:double" minOccurs="1" maxOccurs="1"/>
        <xs:element name="MaxNrVal" type="xs:double" minOccurs="0" maxOccurs="1"/>
      </xs:sequence>
      <xs:sequence>
        <xs:element name="MinNrVal" type="xs:double" minOccurs="1" maxOccurs="1"/>
        <xs:element name="MaxNrVal" type="xs:double" minOccurs="0" maxOccurs="1"/>
      </xs:sequence>
      <xs:sequence>
        <xs:element name="MinNrVal" type="xs:double" minOccurs="1" maxOccurs="1"/>
        <xs:element name="MaxNrVal" type="xs:double" minOccurs="0" maxOccurs="1"/>
      </xs:sequence>
      <xs:sequence>
        <xs:element name="MinNrVal" type="xs:double" minOccurs="1" maxOccurs="1"/>
        <xs:element name="MaxNrVal" type="xs:double" minOccurs="0" maxOccurs="1"/>
      </xs:sequence>
    </xs:choice>
  </xs:sequence>
</xs:complexType>

<xs:element name="Comment" type="xs:string" minOccurs="0" maxOccurs="1"/>
</xs:complexType>

<xs:complexType name="ApplicationT">
  <xs:sequence>
    <xs:element name="ApplicationName" type="xs:string" minOccurs="1" maxOccurs="1"/>
    <xs:element name="Comment" type="xs:string" minOccurs="0" maxOccurs="1"/>
  </xs:sequence>
</xs:complexType>

<xs:element name="StreamsDescriptor" type="StreamsDescriptorT" minOccurs="1" maxOccurs="1"/>
  <!-- check how to make the params unique -->
</xs:element>
</xs:complexType>

<xs:complexType name="QoSParamT">
  <xs:sequence>
    <xs:choice>
      <xs:sequence>
        <xs:element name="MinNrVal" type="xs:double" minOccurs="1" maxOccurs="1"/>
        <xs:element name="MaxNrVal" type="xs:double" minOccurs="0" maxOccurs="1"/>
      </xs:sequence>
      <xs:sequence>
        <xs:element name="MinNrVal" type="xs:double" minOccurs="1" maxOccurs="1"/>
        <xs:element name="MaxNrVal" type="xs:double" minOccurs="0" maxOccurs="1"/>
      </xs:sequence>
      <xs:sequence>
        <xs:element name="MinNrVal" type="xs:double" minOccurs="1" maxOccurs="1"/>
        <xs:element name="MaxNrVal" type="xs:double" minOccurs="0" maxOccurs="1"/>
      </xs:sequence>
      <xs:sequence>
        <xs:element name="MinNrVal" type="xs:double" minOccurs="1" maxOccurs="1"/>
        <xs:element name="MaxNrVal" type="xs:double" minOccurs="0" maxOccurs="1"/>
      </xs:sequence>
      <xs:sequence>
        <xs:element name="MinNrVal" type="xs:double" minOccurs="1" maxOccurs="1"/>
        <xs:element name="MaxNrVal" type="xs:double" minOccurs="0" maxOccurs="1"/>
      </xs:sequence>
    </xs:choice>
  </xs:sequence>
</xs:complexType>

<xs:element name="Comment" type="xs:string" minOccurs="0" maxOccurs="1"/>
</xs:complexType>
<xs:complexType name="QoSParamNameT">
  <xs:restriction base="xs:string">
    <xs:enumeration value="PeakBitRate"/>
    <xs:enumeration value="AvgBitRate"/>
    <xs:enumeration value="Jitter"/>
    <xs:enumeration value="PacketLossRate"/>
  </xs:restriction>
</xs:complexType>

<xs:complexType name="LargerScaleT">
  <xs:restriction base="xs:string">
    <xs:enumeration value="kilo"/>
    <xs:enumeration value="Mega"/>
    <xs:enumeration value="Giga"/>
    <xs:enumeration value="Tera"/>
    <xs:enumeration value="Peta"/>
  </xs:restriction>
</xs:complexType>

<xs:complexType name="SmallerScaleT">
  <xs:restriction base="xs:string">
    <xs:enumeration value="milli"/>
    <xs:enumeration value="micro"/>
    <xs:enumeration value="nano"/>
  </xs:restriction>
</xs:complexType>

<xs:complexType name="ParamUnitT">
  <xs:choice maxOccurs="1">  
    <xs:element minOccurs="1" maxOccurs="1" name="Time" type="TimeT"/>
    <xs:element minOccurs="1" maxOccurs="1" name="Symbols" type="SymbolT"/>
    <xs:element minOccurs="1" maxOccurs="1" name="per"/>
  </xs:choice>  
</xs:complexType>

<xs:complexType name="TimeT" abstract="true"/>

<xs:complexType name="ScalableTime">
  <xs:extension base="TimeT">
    <xs:attribute name="scalar" use="optional" type="SmallerScaleT"/>
    <xs:attribute name="unit" use="required">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="seconds"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
  </xs:extension>
</xs:complexType>

<xs:complexType name="HexagesimalTime">
  <xs:extension base="TimeT">
    <xs:attribute name="scalar" use="prohibited"/>
    <xs:attribute name="unit" use="required">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="minutes"/>
          <xs:enumeration value="hours"/>
          <xs:enumeration value="days"/>
          <xs:enumeration value="years"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
  </xs:extension>
</xs:complexType>
This schema enumerates at first the allowed service classes (lines 4-13 `<xs:simpleType name="ServiceClassNameT">`), which might be extended in the future. One of these values must be used in the beginning of a service class description document (line 20 `<xs:element name="ServiceClassName" type="ServiceClassNameT" minOccurs="1" maxOccurs="1"/>`). It has then to be followed by at least one application description that falls into this service class. Such application description starts with its name (line 32 `<xs:element name="ApplicationName" type="xs:string" minOccurs="1" maxOccurs="1"/>`), must have a descriptor of the required streams and may further define the parameters required by such application.

The stream(s) are described by their degree of symmetry (line 45), which has to be one of those enumerated in lines 56-63 (`<xs:simpleType name="DegreeOfSymmetryChoiceT">`). Further a stream type has to specified (line 47), which may be constant or variable bitrate or ‘not applicable’ (lines 65-71 `<xs:simpleType name="StreamTypeT">`). Finally the number of required streams is specified (line 49), which may be one or two (lines 73-78 `<xs:simpleType name="StreamNumberT">`).

A bit more complex is the definition of the QoS parameters, but this actually results in a quite compact, yet readable description in the actual XML documents. Each required QoS parameter gets an own XML element (line 38 `<xs:element name="QoSParam" type="QoSParamT" minOccurs="0" maxOccurs="5"/>`) that is attributed by the unique name of the parameter (line 101). The allowed values for them are defined in lines 104-112 (`<xs:simpleType name="QoSParamNameT">`). For each parameter an interval, regardless whether it is an open or closed one, or an average can be specified. Even though the specification of text values is currently supported, this is actually deprecated, since it can hardly compared to actual read or measured values (lines 83-94).

Most values carry units. Hence, the schema allows to describe those. To allow a maximum in flexibility and easy capability for the InterMAC to convert between units the schema defines basic units such as ‘second’ or ‘bit’ that easily be scaled into ‘minutes’ or ‘Bytes’ respectively (lines 151-206). Some of those basic units may further be scaled by SI prefixes such as ‘milli’ or ‘Mega’. The allowed values are enumerated in lines 115-131. Reasonably are only seconds to be prefixed to describe divisions and bit or Byte may be prefixed to create larger chunks. Therefore the prefixes are grouped into different types and the respective types of units allow only for one or the other prefix type (lines 156 and 187). ‘Minutes’, ‘hours’ and so on are usually not prefixed (line 171).

Even one combination of units into a more complex unit is possible. These are ‘bits per second’ and its derivations. Lines 133-149 define just these three possible dimensions:

- a time in seconds or scalars of it
- a symbol in bit or scalars of if
- or a symbol per time in bit per second or their respective scalars.

The quotient of two basic dimensions is further marked with a dummy element `<per/>` (line 139). Note, most elements in the description document may further be enriched by a comment that does not require actual processing, but may increase human readability.
The above Listing shows an exemplary description of the service class ‘Entertainment’ by an XML document adhering to the schema defined above.

6.3 MAC protocol and mapping description

A number of existing MAC Protocols that are to be supported by the OMEGA InterMAC have been enumerated and described. All of them do not directly support services classes as stated in OMEGA and mostly not even the QoS parameters declared for these service classes. Each of the MAC protocols provide a number of parameters on their own. Those parameters might be read, set or measured over time in a certain context. Many of those are even optional. That is the standard allows and suggests to implement them but does not require this. Some of the MAC protocols optionally support Quality of Service extensions like IEEE 802.11e. This extends the number of available parameters for that particular MAC protocol but still does not support the OMEGA QoS parameters. The tables in chapter 5 introduce a loose mapping between the OMEGA QoS Parameters, i.e. Average Rate, Packet Loss Rate a.s.o.

To allow an appropriate mapping of any MAC protocol into the OMEGA InterMAC we propose MAC protocol descriptions based on XML to describe the conversion into OMEGA Parameters in detail. Such description should be provided by any MAC protocol that demands to participate in the InterMAC. With having the full conversion description in an XML file that can be provided by the protocol implementation the InterMAC is enabled to hold any available protocol even future developments. Depending on the number of possible parameter subsets from the standards that are implemented a similar number of conversion strategies or formulae may have to be described. It is not feasible to implement every possible conversion formula into the InterMAC. Besides, it was a wasteful performance drawback if the InterMAC would have to determine the available protocol parameters and...
based on them find the matching formula to calculate OMEGA QoS parameters. A sufficient
description in XML provided by each participating MAC protocol allows to develop a universal wrapper
with the XML description being its recipe.

The protocol description has to abide by the format defined by the OMEGA InterMAC. The schema
that defines MAC protocol descriptions is drafted as in Listing 3Listing 3 - Schema for each MAC
protocol description

```xml
  <xs:include schemaLocation="OMEGAServiceClasses.xsd"/>
  <xs:simpleType name="ProtocolParameterEnum">
    <xs:restriction base="xsd:string">
      <xs:enumeration value="TransmittedFragmentCount"/>
      <xs:enumeration value="MulticastTransmittedFrameCount"/>
      <xs:enumeration value="FailedCount"/>
      <xs:enumeration value="ReceivedFragmentCount"/>
      <xs:enumeration value="MulticastReceivedFrameCount"/>
      <xs:enumeration value="FCSErrorCount"/>
      <xs:enumeration value="TransmittedFrameCount"/>
      <xs:enumeration value="RetryCount"/>
      <xs:enumeration value="MultipleRetryCount"/>
      <xs:enumeration value="FrameDuplicateCount"/>
      <xs:enumeration value="RTSSuccessCount"/>
      <xs:enumeration value="RTSFailureCount"/>
      <xs:enumeration value="ACKFailureCount"/>
      <xs:enumeration value="TransmitDelayHistogram"/>
      <xs:enumeration value="AverageTransmitDelay"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:element name="OMEGA_QoS_Parameter">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="ProtocolName" type="xsd:string"/>
        <xs:element name="ProtocolParameter" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="value" type="xsd:double" id=""/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="QoSSupport" minOccurs="0" maxOccurs="1">
          <xs:complexType>
            <xs:sequence maxOccurs="unbounded">
              <xs:element name="Queue">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="UserPriority" type="xsd:unsignedByte"/>
                    <xs:element name="Designation">
                      <xs:complexType>
                        <xs:simpleContent>
                          <xs:extension base="xsd:string">
                            <xs:attribute name="DestIP" type="xsd:string" use="required"/>
                            <xs:attribute name="AC" type="xsd:string" use="required"/>
                          </xs:extension>
                        </xs:simpleContent>
                      </xs:complexType>
                    </xs:element>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="QoSRetryCount"/>
        <xs:element name="QoSMultipleRetryCount"/>
        <xs:element name="QoSFrameDuplicateCount"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```
<xs:element name="PeakRate" type="Dependency" minOccurs="0" maxOccurs="1"/>
<xs:element name="PacketLossRate" type="Dependency" minOccurs="0" maxOccurs="1"/>
<xs:element name="Delay" type="Dependency" minOccurs="0" maxOccurs="1"/>
<xs:element name="Jitter" type="Dependency" minOccurs="0" maxOccurs="1"/>
</xs:sequence>
</xs:complexType>
<xs:complexType name="Dependency">
<xs:sequence>
<xs:element name="Unit" type="ParamUnitT" minOccurs="0" maxOccurs="1"/>
<xs:element name="Comment" type="xs:string" minOccurs="0" maxOccurs="1"/>
<xs:choice minOccurs="1" maxOccurs="1">  
  <xs:element ref="depends"/>
  <xs:element ref="Constant"/>  
</xs:choice>
</xs:sequence>
</xs:complexType>
<xs:element name="depends">
<xs:complexType>
<xs:attribute name="on" use="required"/>
</xs:complexType>
<xs:keyref name="dummy" refer="ProtocolParamKey"/>
<xs:field xpath="@on"/>
</xs:keyref>
</xs:element>
<xs:element name="Constant" type="xs:double"/>
<xs:element name="ConstantBool" type="xs:boolean"/>
<xs:element name="ADD" type="multinaryDbl"/>
<xs:element name="SUB" type="unaryOrBinaryDbl"/>
<xs:element name="MUL" type="multinaryDbl"/>
<xs:element name="DIV" type="binaryDbl"/>
<xs:element name="SQRT" type="unaryDbl"/>
<xs:element name="POWER" type="binaryDbl"/>
<xs:element name="NOT" type="unaryBool"/>
<xs:element name="AND" type="multinaryBool"/>
<xs:element name="OR" type="multinaryBool"/>
<xs:element name="EQUAL" type="binaryBool"/>
<xs:element name="GREATER" type="binaryDbl"/>
<xs:element name="LESS" type="binaryDbl"/>
<xs:element name="GREATEROREQUAL" type="binaryDbl"/>
<xs:element name="LESSOREQUAL" type="binaryDbl"/>
</xs:choice>
</xs:complexType>
<xs:complexType name="ParamUnitT">
<xs:sequence>
  <xs:element name="Delay"/>
  <xs:element name="Jitter"/>
</xs:sequence>
</xs:complexType>
<xs:complexType name="unaryOrBinaryDbl">
  <xs:choice minOccurs="1" maxOccurs="1">  
    <xs:element ref="depends"/>
    <xs:element ref="Constant"/>  
  </xs:choice>
</xs:complexType>
<xs:complexType name="multinaryDbl">
  <xs:choice minOccurs="1" maxOccurs="1">  
    <xs:element ref="depends"/>
    <xs:element ref="Constant"/>  
  </xs:choice>
</xs:complexType>
<xs:complexType name="binaryDbl">
  <xs:choice minOccurs="1" maxOccurs="1">  
    <xs:element ref="depends"/>
    <xs:element ref="Constant"/>  
  </xs:choice>
</xs:complexType>
<xs:complexType name="binaryBool">
  <xs:choice minOccurs="1" maxOccurs="1">  
    <xs:element ref="depends"/>
    <xs:element ref="Constant"/>  
  </xs:choice>
</xs:complexType>
<xs:complexType name="unaryBool">
  <xs:choice minOccurs="1" maxOccurs="1">  
    <xs:element ref="depends"/>
    <xs:element ref="Constant"/>  
  </xs:choice>
</xs:complexType>
<xs:complexType name="unaryBool">
  <xs:choice minOccurs="1" maxOccurs="1">
    <xs:element ref="depends"/>
    <xs:element ref="ConstantBool"/>
  </xs:choice>
  <xs:element ref="NOT"/>
  <xs:element ref="AND"/>
  <xs:element ref="OR"/>
  <xs:element ref="EQUAL"/>
  <xs:element ref="GREATER"/>
  <xs:element ref="LESS"/>
  <xs:element ref="GREATEROREQUAL"/>
  <xs:element ref="LESSOREQUAL"/>
</xs:complexType>

<xs:complexType name="binaryDbl">
  <xs:choice minOccurs="2" maxOccurs="2">
    <xs:element ref="depends"/>
    <xs:element ref="Constant"/>
  </xs:choice>
  <xs:element ref="ADD"/>
  <xs:element ref="SUB"/>
  <xs:element ref="MUL"/>
  <xs:element ref="DIV"/>
  <xs:element ref="SQRT"/>
  <xs:element ref="POWER"/>
</xs:complexType>

<xs:complexType name="unaryDbl">
  <xs:choice minOccurs="1" maxOccurs="2">
    <xs:element ref="depends"/>
    <xs:element ref="Constant"/>
  </xs:choice>
  <xs:element ref="ADD"/>
  <xs:element ref="SUB"/>
  <xs:element ref="MUL"/>
  <xs:element ref="DIV"/>
  <xs:element ref="SQRT"/>
  <xs:element ref="POWER"/>
</xs:complexType>

<xs:complexType name="multinaryDbl">
  <xs:choice minOccurs="2" maxOccurs="unbounded">
    <xs:element ref="depends"/>
    <xs:element ref="Constant"/>
  </xs:choice>
  <xs:element ref="ADD"/>
  <xs:element ref="SUB"/>
  <xs:element ref="MUL"/>
  <xs:element ref="DIV"/>
  <xs:element ref="SQRT"/>
  <xs:element ref="POWER"/>
</xs:complexType>

<xs:complexType name="multinaryBool">
  <xs:choice minOccurs="2" maxOccurs="unbounded">
    <xs:element ref="depends"/>
  </xs:choice>
</xs:complexType>
The schema defines first an enumeration of possible parameter names (lines 4-21). Only parameters of those names may be used in the description later on. This prevents accidental typos. On the other hand it introduces a difficulty to add new MAC protocol parameters in future protocols.

Next it defines the basic structure of the XML document. That is all existing parameters are declared (lines 27-34). Each declared parameter is attributed to distinguish between parameters that are to be different priorities. Those queues can be enumerated and equipped with their own protocol (line 61). To ensure that in this definition no parameter is used that was not declared beforehand, the schema defines an indexing key over the declared parameters (lines 64-67). The conversion definition is dependent on a number of declared parameters. This dependency is expressed by a mathematical term having constants and references to the declared parameters as terminals. The schema defines an indexing key over the declared parameters (lines 64-67). The conversion definition may include up to five different QoS parameters as stated in the OMEGA service classes (lines 71-75). Each of them is dependent on a number of declared parameters. This dependency is expressed by a mathematical term having constants and references to the declared parameters as terminals. The math term is described in the form of its syntax tree with the operators being nodes and the operands being the sub-nodes. Since XML already describes trees it well fits the purpose for describing math syntax trees. Commutative operations like addition, multiplication or conjunction can even have more than two operands. For this purpose the schema defines a number of XML schema types which hold operands of unary, binary and multi-nary operations. They further allow a basic type safety in the operations such that conjunction or disjunction may only be processed on Boolean operands, addition
or division only on real numbers and result in a Boolean type or real type respectively and finally comparisons that always result in a Boolean type but may accept either Boolean or real type operands. Please refer to lines 138-272 for the definitions of the operation schema types.

For the terminals we defined typed constants (lines 118-119) and the variables that refer to the declared parameters (lines 107-116). This element used as a variable makes use of the previously defined key index. This makes sure that only parameters are used as a variable, which are declared above and hence are available from the protocol.

The conversion dependency also holds an element describing the dimension unit (line 80). The XML type definition is inherited from the OMEGA Service classes' schema. This allows to ensure that only those units are used that are known to the OMEGA InterMAC and thus can be reasonably processed.

The following Listing shows an example of such XML protocol description for the IEEE 802.11n MAC protocol.

Listing 4 - Description document for the IEEE 802.11n MAC protocol

```xml
<?xml version="1.0" encoding="utf-8"?>
  <ProtocolName>IEEE 802.11n</ProtocolName>
  <ProtocolParameter id="TransmittedFragmentCount">
    <value>0</value>
  </ProtocolParameter>
  <ProtocolParameter id="MulticastTransmittedFrameCount">
    <value>0</value>
  </ProtocolParameter>
  <ProtocolParameter id="FailedCount">
    <value>0</value>
  </ProtocolParameter>
  <ProtocolParameter id="ReceivedFragmentCount">
    <value>0</value>
  </ProtocolParameter>
  <ProtocolParameter id="MulticastReceivedFrameCount">
    <value>0</value>
  </ProtocolParameter>
  <ProtocolParameter id="FCSErrorCount">
    <value>0</value>
  </ProtocolParameter>
  <ProtocolParameter id="TransmittedFrameCount">
    <value>0</value>
  </ProtocolParameter>
  <ProtocolParameter id="RetryCount">
    <value>0</value>
  </ProtocolParameter>
  <ProtocolParameter id="MultipleRetryCount">
    <value>0</value>
  </ProtocolParameter>
  <ProtocolParameter id="FrameDuplicateCount">
    <value>0</value>
  </ProtocolParameter>
  <ProtocolParameter id="RTSSuccessCount">
    <value>0</value>
  </ProtocolParameter>
  <ProtocolParameter id="RTSFailureCount">
    <value>0</value>
  </ProtocolParameter>
  <ProtocolParameter id="ACKFailureCount">
    <value>0</value>
  </ProtocolParameter>
  <ProtocolParameter id="TransmitDelayHistogram">
    <value>0</value>
  </ProtocolParameter>
  <ProtocolParameter id="AverageTransmitDelay">
    <value>0</value>
  </ProtocolParameter>
</OMEGA_QoS_Parameter>
```
6.4 Potential Language Extension

In addition to the rules that describe those relations between MAC and QoS class parameters, decision rules are needed to refine the behaviour of the InterMAC admission control component.
Those decision rules are needed whenever a direct mapping of QoS service class definitions or more correctly requirements does not match exactly what the technology dependent MAC layer can provide. In such cases the maximum allowed variation or deviation from the requested parameters need to be defined. By that we avoid extremely strict and probably to restrictive handling of QoS requirements. This will have a significant effect for determining whether a certain path is going to support the required parameters. For this purpose we use the types of metrics as introduced already in section 4.3.
7 Appendix

7.1 OMEGA XML Scheme for service classes

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <!-- the root element of service class description pattern -->
  <xs:element name="OMEGA_ServiceClassDescriptionPattern">
    <!-- the unique name of an OMEGA service class -->
    <xs:element name="ServiceClassName" type="ServiceClassNameT" minOccurs="1" maxOccurs="1" />
    <xs:element name="ServiceClassApplication" type="ApplicationT" minOccurs="1" maxOccurs="unbounded" />
    <xs:element name="Comment" type="xs:string" minOccurs="0" maxOccurs="1" />
  </xs:sequence>
</xs:complexType>
<xs:complexType name="ServiceClassNameT">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Emergency" />
    <xs:enumeration value="Conferencing" />
    <xs:enumeration value="Streaming" />
    <xs:enumeration value="Entertainment" />
    <xs:enumeration value="BestEffort" />
    <xs:enumeration value="Other" />
  </xs:restriction>
</xs:complexType>
<xs:complexType name="ApplicationT">
  <xs:sequence>
    <xs:element name="ApplicationName" type="xs:string" minOccurs="1" maxOccurs="1" />
    <xs:element name="Comment" type="xs:string" minOccurs="0" maxOccurs="1" />
    <xs:element name="StreamsDescriptor" type="StreamsDescriptorT" minOccurs="1" maxOccurs="unbounded" />
    <xs:element name="QoSParam" type="QoSParamT" minOccurs="4" maxOccurs="unbounded" />
  </xs:sequence>
</xs:complexType>
<xs:complexType name="StreamsDescriptorT">
  <xs:sequence>
    <xs:element name="DegreeOfSymmetry" type="DegreeOfSymmetryChoiceT" minOccurs="1" maxOccurs="1" />
    <xs:element name="StreamType" type="StreamTypeT" minOccurs="1" maxOccurs="1" />
    <xs:element name="StreamNumber" type="StreamNumberT" minOccurs="1" maxOccurs="1" />
    <xs:element name="Comment" type="xs:string" minOccurs="0" maxOccurs="1" />
  </xs:sequence>
</xs:complexType>
</xs:schema>
```
<xs:complexType name="DegreeOfSymmetryChoiceT">
  <xs:restriction base="xs:string">
    <xs:enumeration value="OneWay"/>
    <xs:enumeration value="TwoWay"/> 
    <xs:enumeration value="TwoWayAsymmetric"/>
    <xs:enumeration value="null"/>
  </xs:restriction>
</xs:simpleType>

- <xs:simpleType name="StreamTypeT">
  <xs:restriction base="xs:string">
    <xs:enumeration value="CBR"/>
    <xs:enumeration value="VBR"/>
    <xs:enumeration value="null"/>
  </xs:restriction>
</xs:simpleType>

- <xs:simpleType name="StreamNumberT">
  <xs:restriction base="xs:integer">
    <xs:enumeration value="1"/>
    <xs:enumeration value="2"/>
  </xs:restriction>
</xs:simpleType>

- <xs:complexType name="QoSParamT">
  <xs:sequence>
    <xs:element name="QoSParamName" type="QoSParamNameT" minOccurs="1" maxOccurs="1" />
    <xs:element name="QoSParamValue" type="ParamValueT" minOccurs="1" maxOccurs="1" />
    <xs:element name="QoSParamScaleUnit" type="ParamScaleUnitT" minOccurs="1" maxOccurs="1" />
    <xs:element name="Comment" type="xs:string" minOccurs="0" maxOccurs="1"/>
  </xs:sequence>
</xs:complexType>

- <xs:simpleType name="QoSParamNameT">
  <xs:restriction base="xs:string">
    <xs:enumeration value="AvgBitRate"/>
    <xs:enumeration value="Delay"/>
    <xs:enumeration value="Jitter"/>
    <xs:enumeration value="PacketLossRate"/> 
  </xs:restriction>
</xs:simpleType>

- <xs:complexType name="ParamValueT">
  <xs:sequence>
    <xs:choice>
      <xs:element name="NrRange" type="RangeT" minOccurs="1" maxOccurs="1"/>
      <xs:element name="MinNrVal" type="xs:double" minOccurs="1" maxOccurs="1"/>
      <xs:element name="MaxNrVal" type="xs:double" minOccurs="1" maxOccurs="1"/>
      <xs:element name="AvgNrVal" type="xs:double" minOccurs="1" maxOccurs="1"/>
      <xs:element name="TxtVal" type="xs:string" minOccurs="1" maxOccurs="1"/>
    </xs:choice>
    <xs:element name="Comment" type="xs:string" minOccurs="0" maxOccurs="1"/>
  </xs:sequence>
</xs:complexType>

- <xs:complexType name="ParamScaleUnitT"/>
- <xs:sequence>
  <xs:element name="BaseUnit" type="xs:string" minOccurs="1" maxOccurs="1" />
  <xs:element name="DerivedUnit" type="DerivedUnitT" minOccurs="0" maxOccurs="unbounded" />
  <xs:element name="Comment" type="xs:string" minOccurs="0" maxOccurs="1" />
</xs:sequence>
</xs:complexType>
- <xs:complexType name="RangeT">
  <xs:sequence>
    <xs:element name="MinVal" type="xs:double" minOccurs="1" maxOccurs="1" />
    <xs:element name="MaxVal" type="xs:double" minOccurs="1" maxOccurs="1" />
    <xs:element name="Comment" type="xs:string" minOccurs="0" maxOccurs="1" />
  </xs:sequence>
</xs:complexType>
- <xs:complexType name="DerivedUnitT">
  <xs:sequence>
    <xs:element name="Unit" type="xs:string" minOccurs="1" maxOccurs="1" />
    <xs:element name="FactorToBaseUnit" type="xs:double" minOccurs="1" maxOccurs="1" />
    <xs:element name="Comment" type="xs:string" minOccurs="0" maxOccurs="1" />
  </xs:sequence>
</xs:complexType>
7.2 Description of OMEGA Service Classes

7.2.1 Best effort

<?xml version="1.0" encoding="UTF-8" ?>

<OMEGA_ServiceClassDescriptionPattern
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="OMEGAServiceClasses.xsd">
  <ServiceClassName>BestEffort</ServiceClassName>
  <ApplicationName>Web navigation</ApplicationName>
  <StreamsDescriptor>
    <DegreeOfSymmetry>TwoWayAsymmetric</DegreeOfSymmetry>
    <StreamType>null</StreamType>
    <StreamNumber>2</StreamNumber>
    <Comment>two or more asymmetric variable streams (no stream type was defined here)</Comment>
  </StreamsDescriptor>
  <QoSParam>
    <QoSParamName>AvgBitRate</QoSParamName>
    <QoSParamValue>
      <MaxNrVal>1.5</MaxNrVal>
      <Comment>some kbps to 1.5 Mbps</Comment>
    </QoSParamValue>
  </QoSParam>
  <BaseUnit>Mbps</BaseUnit>
  <DerivedUnit>
    <Unit>kbps</Unit>
    <FactorToBaseUnit>1000</FactorToBaseUnit>
  </DerivedUnit>
  <DerivedUnit>
    <Unit>bps</Unit>
    <FactorToBaseUnit>1000000</FactorToBaseUnit>
  </DerivedUnit>
  <QoSParam>
    <QoSParamName>Delay</QoSParamName>
    <QoSParamValue>
      <TxtVal>null</TxtVal>
      <Comment>no delay value was defined here</Comment>
    </QoSParamValue>
  </QoSParam>
  <QoSParam>
    <QoSParamName>Jitter</QoSParamName>
    <QoSParamValue>
      <TxtVal>null</TxtVal>
    </QoSParamValue>
  </QoSParam>
</OMEGA_ServiceClassDescriptionPattern>
<Comment>no jitter value was defined here</Comment>

- <QoSParam>
  - <QoSParamScaleUnit>
    - <BaseUnit>ms</BaseUnit>
  - <QoSParam>
    - <QoSParamName>PacketLossRate</QoSParamName>
    - <QoSParamValue>
      - <TxtVal>null</TxtVal>
      - <Comment>no packet loss rate value was defined here</Comment>
    - <QoSParamScaleUnit>
      - <BaseUnit>percent</BaseUnit>
    - <QoSParam>
      - <ServiceClassApplication>
        - <ApplicationName>Downloading</ApplicationName>
        - <Comment>downloading of files</Comment>
      - <StreamsDescriptor>
        - <DegreeOfSymmetry>OneWay</DegreeOfSymmetry>
        - <StreamType>VBR</StreamType>
        - <StreamNumber>1</StreamNumber>
      - </StreamsDescriptor>
    - <QoSParamName>AvgBitRate</QoSParamName>
    - <QoSParamValue>
      - <TxtVal>some</TxtVal>
      - <Comment>VBR flow up to the Gbps</Comment>
    - <QoSParamScaleUnit>
      - <BaseUnit>Gbps</BaseUnit>
      - <DerivedUnit>
        - <Unit>Mbps</Unit>
        - <FactorToBaseUnit>1000</FactorToBaseUnit>
      - </DerivedUnit>
      - <DerivedUnit>
        - <Unit>kbps</Unit>
        - <FactorToBaseUnit>1000000</FactorToBaseUnit>
      - </DerivedUnit>
      - <DerivedUnit>
        - <Unit>bps</Unit>
        - <FactorToBaseUnit>1000000000</FactorToBaseUnit>
      - </DerivedUnit>
    - <QoSParamName>Delay</QoSParamName>
    - <QoSParamValue>
      - <TxtVal>null</TxtVal>
      - <Comment>no delay value was defined here</Comment>
    - <QoSParamScaleUnit>
      - <BaseUnit>ms</BaseUnit>
    - </QoSParam>
  - </QoSParam>
- </QoSParam>
- </ApplicationName>
- </Comment>
- </StreamsDescriptor>
- </ServiceClassApplication>
<?xml version="1.0" encoding="UTF-8" ?>

<OMEGA_ServiceClassDescriptionPattern xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="OMEGAServiceClasses.xsd">
  <ServiceClassName>Conferencing</ServiceClassName>
  <ServiceClassApplication>
    <ApplicationName>Voice over IP</ApplicationName>
    <StreamsDescriptor>
      <DegreeOfSymmetry>TwoWay</DegreeOfSymmetry>
      <StreamType>CBR</StreamType>
      <StreamNumber>2</StreamNumber>
      <Comment>two symmetric CBR flows</Comment>
    </StreamsDescriptor>
    <QoSParamName>AvgBitRate</QoSParamName>
    <QoSParamValue>
      <NrRange>
        <MinVal>10</MinVal>
        <MaxVal>100</MaxVal>
      </NrRange>
      <Comment>some 10 kbps</Comment>
    </QoSParamValue>
    <QoSParamScaleUnit>
      <BaseUnit>kbps</BaseUnit>
    </QoSParamScaleUnit>
  </ServiceClassApplication>
</OMEGA_ServiceClassDescriptionPattern>

7.2.2 Conferencing
- <DerivedUnit>
  <Unit>bps</Unit>
  <FactorToBaseUnit>1000</FactorToBaseUnit>
</DerivedUnit>

- <DerivedUnit>
  <Unit>Mbps</Unit>
  <FactorToBaseUnit>0.001</FactorToBaseUnit>
    <DerivedUnit>
      </QoSParamScaleUnit>
      </QoSParam>
    </QoSParamValue>
- <QoSParamName>Delay</QoSParamName>
  - <QoSParamValue>
    - <NrRange>
      <MinVal>10</MinVal>
      <MaxVal>100</MaxVal>
    </NrRange>
    <Comment>real time constraint: delay time between 10 and 100 ms</Comment>
    </QoSParamValue>
  - <QoSParamScaleUnit>
    <BaseUnit>ms</BaseUnit>
  - <DerivedUnit>
    <Unit>s</Unit>
    <FactorToBaseUnit>0.001</FactorToBaseUnit>
      <DerivedUnit>
        </QoSParamScaleUnit>
        </QoSParam>
      </QoSParamValue>
  - <QoSParamName>Jitter</QoSParamName>
  - <QoSParamValue>
    <MaxNrVal>20</MaxNrVal>
    <Comment>smaller than 20 ms</Comment>
    </QoSParamValue>
  - <QoSParamScaleUnit>
    <BaseUnit>ms</BaseUnit>
  - <DerivedUnit>
    <Unit>s</Unit>
    <FactorToBaseUnit>0.001</FactorToBaseUnit>
      <DerivedUnit>
        </QoSParamScaleUnit>
        </QoSParam>
      </QoSParamValue>
  - <QoSParamName>PacketLossRate</QoSParamName>
  - <QoSParamValue>
    <MaxNrVal>0.1</MaxNrVal>
    <Comment>smaller than 0.1 percent</Comment>
    </QoSParamValue>
  - <QoSParamScaleUnit>
    <BaseUnit>percent</BaseUnit>
      </QoSParamScaleUnit>
      </QoSParam>
    </QoSParam>
  - </ServiceClassApplication>
- <ServiceClassApplication>
  <ApplicationName>Video conference</ApplicationName>
  - <StreamsDescriptor>
<DegreeOfSymmetry>TwoWay</DegreeOfSymmetry>
<StreamType>null</StreamType>
<StreamNumber>2</StreamNumber>
<Comment>two symmetric flows (no stream type was defined here)</Comment>
</StreamsDescriptor>
- <QoSParam>
  <QoSParamName>AvgBitRate</QoSParamName>
  <QoSParamValue>
    <NrRange>
      <MinVal>128</MinVal>
      <MaxVal>4000</MaxVal>
    </NrRange>
    <Comment>between 128 kbps and 4 Mbps</Comment>
  </QoSParamValue>
</QoSParam>
- <QoSParamScaleUnit>
  <BaseUnit>kbps</BaseUnit>
  <DerivedUnit>
    <Unit>bps</Unit>
    <FactorToBaseUnit>1000</FactorToBaseUnit>
  </DerivedUnit>
  <DerivedUnit>
    <Unit>Mbps</Unit>
    <FactorToBaseUnit>0.001</FactorToBaseUnit>
  </DerivedUnit>
</QoSParamScaleUnit>
- <QoSParam>
  <QoSParamName>Delay</QoSParamName>
  <QoSParamValue>
    <NrRange>
      <MinVal>10</MinVal>
      <MaxVal>100</MaxVal>
    </NrRange>
    <Comment>real time constraint: delay time between 10 and 100 ms</Comment>
  </QoSParamValue>
</QoSParam>
- <QoSParamScaleUnit>
  <BaseUnit>ms</BaseUnit>
  <DerivedUnit>
    <Unit>s</Unit>
    <FactorToBaseUnit>0.001</FactorToBaseUnit>
  </DerivedUnit>
</QoSParamScaleUnit>
- <QoSParam>
  <QoSParamName>Jitter</QoSParamName>
  <QoSParamValue>
    <MaxNrVal>10</MaxNrVal>
    <Comment>smaller than 10 ms</Comment>
  </QoSParamValue>
</QoSParam>
- <QoSParamScaleUnit>
  <BaseUnit>ms</BaseUnit>
  <DerivedUnit>
    <Unit>s</Unit>
    <FactorToBaseUnit>0.001</FactorToBaseUnit>
  </DerivedUnit>
</QoSParamScaleUnit>
7.2.3 Emergency classes

<?xml version="1.0" encoding="UTF-8"?>
<OMEGA_ServiceClassDescriptionPattern
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="OMEGAServiceClasses.xsd">
<ServiceClassName>Emergency</ServiceClassName>
<ApplicationName>Sensor flows</ApplicationName>
<Comment>for example alarms</Comment>
<StreamsDescriptor>
<DegreeOfSymmetry>OneWay</DegreeOfSymmetry>
<StreamType>null</StreamType>
<StreamNumber>1</StreamNumber>
<Comment>no stream type was defined here</Comment>
</StreamsDescriptor>
<QoSParam>
<QoSParamName>AvgBitRate</QoSParamName>
<QoSParamValue>
<TxtVal>some</TxtVal>
<Comment>some kbps (max.)</Comment>
</QoSParamValue>
</QoSParam>
<BaseUnit>kbps</BaseUnit>
</QoSParamScaleUnit>
</QoSParam>
<TxtVal>real time</TxtVal>
<Comment>should be real time</Comment>

<QoSParamScaleUnit>
  <BaseUnit>ms</BaseUnit>
</QoSParamScaleUnit>

<QoSParam>
  <QoSParamName>Jitter</QoSParamName>
  <QoSParamValue>
    <TxtVal>null</TxtVal>
  </QoSParamValue>
  <Comment>no jitter value was defined here</Comment>
</QoSParam>

<QoSParamScaleUnit>
  <BaseUnit>ms</BaseUnit>
</QoSParamScaleUnit>

<QoSParam>
  <QoSParamName>PacketLossRate</QoSParamName>
  <QoSParamValue>
    <TxtVal>null</TxtVal>
  </QoSParamValue>
  <Comment>the PLR parameter was marked as critical criterion, but no PLR value was defined here</Comment>
</QoSParam>

<QoSParamScaleUnit>
  <BaseUnit>percent</BaseUnit>
</QoSParamScaleUnit>

<ServiceClassApplication>
  <ServiceClassName>Entertainment</ServiceClassName>
  <ApplicationName>Gaming</ApplicationName>
  <StreamsDescriptor>
    <DegreeOfSymmetry>TwoWayAsymmetric</DegreeOfSymmetry>
    <StreamType>null</StreamType>
    <StreamNumber>2</StreamNumber>
    <Comment>two or more asymmetric streams (no stream type was defined here)</Comment>
  </StreamsDescriptor>
</ServiceClassApplication>

<QoSParam>
  <QoSParamName>AvgBitRate</QoSParamName>
  <QoSParamValue>
    <MaxNrVal>100</MaxNrVal>
  </QoSParamValue>
  <Comment>smaller than 100 kbps</Comment>
</QoSParam>

7.2.4 Entertainment

<?xml version="1.0" encoding="UTF-8" ?>

<OMEGA_ServiceClassDescriptionPattern
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="OMEGAServiceClasses.xsd">
  <ServiceClassName>Entertainment</ServiceClassName>
  <ApplicationName>Gaming</ApplicationName>
  <StreamsDescriptor>
    <DegreeOfSymmetry>TwoWayAsymmetric</DegreeOfSymmetry>
    <StreamType>null</StreamType>
    <StreamNumber>2</StreamNumber>
    <Comment>two or more asymmetric streams (no stream type was defined here)</Comment>
  </StreamsDescriptor>
</OMEGA_ServiceClassDescriptionPattern>
<BaseUnit>kbps</BaseUnit>
- <DerivedUnit>
  <Unit>bps</Unit>
  <FactorToBaseUnit>1000</FactorToBaseUnit>
  </DerivedUnit>
- <DerivedUnit>
  <Unit>Mbps</Unit>
  <FactorToBaseUnit>0.001</FactorToBaseUnit>
  </DerivedUnit>
</QoSParam>

- <QoSParamName>Delay</QoSParamName>
- <QoSParamValue>
  <MaxNrVal>50</MaxNrVal>
  <Comment>smaller than 50 ms</Comment>
</QoSParamValue>
- <QoSParamScaleUnit>
  <BaseUnit>ms</BaseUnit>
  <FactorToBaseUnit>1</FactorToBaseUnit>
  </QoSParamScaleUnit>
</QoSParam>

- <QoSParamName>Jitter</QoSParamName>
- <QoSParamValue>
  <MaxNrVal>1</MaxNrVal>
  <Comment>smaller than 1 ms</Comment>
</QoSParamValue>
- <QoSParamScaleUnit>
  <BaseUnit>ms</BaseUnit>
  <FactorToBaseUnit>1</FactorToBaseUnit>
  </QoSParamScaleUnit>
</QoSParam>

- <QoSParamName>PacketLossRate</QoSParamName>
- <QoSParamValue>
  <MaxNrVal>0.1</MaxNrVal>
  <Comment>smaller than 0.1 percent</Comment>
</QoSParamValue>
- <QoSParamScaleUnit>
  <BaseUnit>percent</BaseUnit>
  </QoSParamScaleUnit>
</QoSParam>

7.2.5 Streaming class
<?xml version="1.0" encoding="UTF-8" ?>
<OMEGA_ServiceClassDescriptionPattern
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="OMEGAServiceClasses.xsd">
  <ServiceClassName>Streaming</ServiceClassName>
  <ServiceClassApplication>
    <ApplicationName>Audio diffusion</ApplicationName>
    <Comment>MPEG-2 layer 3 (MP3) for example</Comment>
  </ServiceClassApplication>
  <DegreeOfSymmetry>OneWay</DegreeOfSymmetry>
  <StreamType>CBR</StreamType>
  <StreamNumber>1</StreamNumber>
  <Comment>CBR flow with peaks for admission control</Comment>
  <StreamsDescriptor>
    <StreamNumber>1</StreamNumber>
    <Comment>CBR flow with peaks for admission control</Comment>
  </StreamsDescriptor>
  <QoSParam>
    <QoSParamName>AvgBitRate</QoSParamName>
    <QoSParamValue>
      <TxtVal>some</TxtVal>
      <Comment>some kbps (avg.)</Comment>
    </QoSParamValue>
    <BaseUnit>kbps</BaseUnit>
    <DerivedUnit>
      <Unit>bps</Unit>
      <FactorToBaseUnit>1000</FactorToBaseUnit>
    </DerivedUnit>
    <DerivedUnit>
      <Unit>Mbps</Unit>
      <FactorToBaseUnit>0.001</FactorToBaseUnit>
    </DerivedUnit>
  </QoSParam>
  <QoSParamName>Delay</QoSParamName>
  <QoSParamValue>
    <MaxNrVal>400</MaxNrVal>
    <Comment>Audio synchronisation constraint: delay time smaller than 400 ms</Comment>
  </QoSParamValue>
  <BaseUnit>ms</BaseUnit>
  <DerivedUnit>
    <Unit>s</Unit>
    <FactorToBaseUnit>0.001</FactorToBaseUnit>
  </DerivedUnit>
</OMEGA_ServiceClassDescriptionPattern>
<BaseUnit>/ms</BaseUnit>
</QoSParamScaleUnit>
</QoSParam>

- <QoSParam>
  <QoSParamName>PacketLossRate</QoSParamName>
  <QoSParamValue>
    <MaxNrVal>0.001</MaxNrVal>
    <Comment>smaller than 0.001 percent</Comment>
  </QoSParamValue>
</QoSParamScaleUnit>

- <BaseUnit>/percent</BaseUnit>
  <QoSParamScaleUnit>
    <QoSParam>
      </ServiceClassApplication>
      <ApplicationName>Video diffusion</ApplicationName>
      <Comment>MPEG-2 base or enhancement layer for example</Comment>
      </StreamsDescriptor>
  
  <QoSParam>
    <QoSParamName>AvgBitRate</QoSParamName>
    <QoSParamValue>
      <NrRange>
        <MinVal>2</MinVal>
        <MaxVal>50</MaxVal>
      </NrRange>
      <Comment>between 2 Mbps and 50 Mbps</Comment>
    </QoSParamValue>
    <BaseUnit>/Mbps</BaseUnit>
    <DerivedUnit>
      <Unit>/kbps</Unit>
      <FactorToBaseUnit>1000</FactorToBaseUnit>
    </DerivedUnit>
    <DerivedUnit>
      <Unit>/bps</Unit>
      <FactorToBaseUnit>1000000</FactorToBaseUnit>
    </DerivedUnit>

  </QoSParam>

- <QoSParam>
  <QoSParamName>Delay</QoSParamName>
  <QoSParamValue>
    <MaxNrVal>400</MaxNrVal>
    <Comment>Video synchronisation constraint: delay time smaller than 400 ms</Comment>
  </QoSParamValue>
</QoSParamScaleUnit>

- <BaseUnit>/ms</BaseUnit>
  <DerivedUnit>
    <Unit>/s</Unit>
  </DerivedUnit>
<FactorToBaseUnit>0.001</FactorToBaseUnit>
</QoSParamScaleUnit>
</QoSParam>

- <QoSParam>
  <QoSParamName>Jitter</QoSParamName>
  <QoSParamValue>
    <TxtVal>null</TxtVal>
    <Comment>no jitter value was defined here</Comment>
  </QoSParamValue>
  <QoSParamScaleUnit>
    <BaseUnit>ms</BaseUnit>
    </QoSParamScaleUnit>
    </QoSParam>

- <QoSParam>
  <QoSParamName>PacketLossRate</QoSParamName>
  <QoSParamValue>
    <MaxNrVal>0.001</MaxNrVal>
    <Comment>smaller than 0.001 percent</Comment>
  </QoSParamValue>
  <QoSParamScaleUnit>
    <BaseUnit>percent</BaseUnit>
    </QoSParamScaleUnit>
    </QoSParam>

- <ServiceClassApplication>
  <ApplicationName>3D Facial Animation</ApplicationName>
  </ServiceClassApplication>

- <StreamsDescriptor>
  <DegreeOfSymmetry>OneWay</DegreeOfSymmetry>
  <StreamType>null</StreamType>
  <StreamNumber>1</StreamNumber>
  <Comment>no stream type was defined here</Comment>
  </StreamsDescriptor>

- <QoSParam>
  <QoSParamName>AvgBitRate</QoSParamName>
  <QoSParamValue>
    <MaxNrVal>50</MaxNrVal>
    <Comment>smaller than 50 kbps</Comment>
  </QoSParamValue>
  <QoSParamScaleUnit>
    <BaseUnit>kbps</BaseUnit>
    </QoSParamScaleUnit>
    </QoSParam>

- <QoSParam>
  <QoSParamName>Delay</QoSParamName>
  <QoSParamValue>
    <DerivedUnit>
      <Unit>bps</Unit>
      <FactorToBaseUnit>1000</FactorToBaseUnit>
      </DerivedUnit>
      <DerivedUnit>
        <Unit>Mbps</Unit>
        <FactorToBaseUnit>0.001</FactorToBaseUnit>
        </DerivedUnit>
        </QoSParamValue>
<MaxNrVal>300</MaxNrVal>
<Comment>Audio 3D synchronisation constraint: delay time smaller than 300 ms</Comment>
</QoSParamValue>
- <QoSParamScaleUnit>
  <BaseUnit>ms</BaseUnit>
</QoSParamScaleUnit>
- <DerivedUnit>
  <Unit>s</Unit>
  <FactorToBaseUnit>0.001</FactorToBaseUnit>
  </DerivedUnit>
</QoSParam>
- <QoSParamName>Jitter</QoSParamName>
- <QoSParamValue>
  <TxtVal>null</TxtVal>
  <Comment>no jitter value was defined here</Comment>
</QoSParamValue>
- <QoSParamScaleUnit>
  <BaseUnit>ms</BaseUnit>
</QoSParamScaleUnit>
- <QoSParam>
  <QoSParamName>PacketLossRate</QoSParamName>
  <QoSParamValue>
    <TxtVal>null</TxtVal>
    <Comment>no packet loss rate value was defined here</Comment>
  </QoSParamValue>
  <DerivedUnit>
    <BaseUnit>percent</BaseUnit>
    <FactorToBaseUnit>0.01</FactorToBaseUnit>
  </DerivedUnit>
</QoSParam>
</ServiceClassApplication>
</OMEGA_ServiceClassDescriptionPattern>

7.2.6 Other

<?xml version="1.0" encoding="UTF-8" ?>
- <OMEGA_ServiceClassDescriptionPattern
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="OMEGAServiceClasses.xsd">
  <ServiceClassName>Other</ServiceClassName>
  <ServiceClassApplication>
    <ApplicationName>Replacement of cable</ApplicationName>
    <StreamsDescriptor>
      <DegreeOfSymmetry>OneWay</DegreeOfSymmetry>
      <StreamType>null</StreamType>
      <StreamNumber>1</StreamNumber>
      <Comment>mono-directional flow (no stream type was defined here)</Comment>
    </StreamsDescriptor>
    - <QoSParam>
      <QoSParamName>AvgBitRate</QoSParamName>
      <QoSParamValue>
      </QoSParamValue>
      <QoSParamScaleUnit>
      </QoSParamScaleUnit>
      <QoSParam>
      </QoSParam>
</ServiceClassApplication>
</OMEGA_ServiceClassDescriptionPattern>
<MinNrVal>1</MinNrVal>
<Comment>1 Gbps or more</Comment>
</QoSParamValue>
- <QoSParamScaleUnit>
  <BaseUnit>Gbps</BaseUnit>
- <DerivedUnit>
  <Unit>kbps</Unit>
  <FactorToBaseUnit>1000</FactorToBaseUnit>
  </DerivedUnit>
- <DerivedUnit>
  <Unit>bps</Unit>
  <FactorToBaseUnit>1000000</FactorToBaseUnit>
  </DerivedUnit>
- <QoSParamName>Delay</QoSParamName>
- <QoSParamValue>
  <MaxNrVal>400</MaxNrVal>
  <Comment>smaller than 400 ms</Comment>
  </QoSParamValue>
- <QoSParamScaleUnit>
  <BaseUnit>ms</BaseUnit>
- <DerivedUnit>
  <Unit>s</Unit>
  <FactorToBaseUnit>0.001</FactorToBaseUnit>
  </DerivedUnit>
  </QoSParamScaleUnit>
  </QoSParam>
- <QoSParamName>Jitter</QoSParamName>
- <QoSParamValue>
  <MaxNrVal>1</MaxNrVal>
  <Comment>smaller than 1 ms</Comment>
  </QoSParamValue>
- <QoSParamScaleUnit>
  <BaseUnit>ms</BaseUnit>
- <DerivedUnit>
  <Unit>s</Unit>
  <FactorToBaseUnit>0.001</FactorToBaseUnit>
  </DerivedUnit>
  </QoSParamScaleUnit>
  </QoSParam>
- <QoSParamName>PacketLossRate</QoSParamName>
- <QoSParamValue>
  <TxtVal>null</TxtVal>
  <Comment>no packet loss rate value was defined here</Comment>
  </QoSParamValue>
- <QoSParamScaleUnit>
  <BaseUnit>percent</BaseUnit>
  </QoSParamScaleUnit>
  </QoSParam>
  </ServiceClassApplication>
</OMEGA_ServiceClassDescriptionPattern>
8 References


[20] Dallas E. Wrege, Jörg Liebeherr, Video Traffic Characterisation for Multimedia Networks with a Deterministic Service, Department of Computer Sience, University of Virginia, Charlottesville, VA 22903, 1996

[22] “RDF Model and Syntax Specification”, available: http://www.w3.org/TR/REC-rdf-syntax/, last access: 03.2008

[23] “RDF Vocabulary Description Language 1.0: RDF Schema”, available: http://www.w3.org/TR/rdf-schema/, last access: 03.2008


[27] „How to write F – Logic – Programs“, Tutorial Series, ontoprise GmbH, April 2007


[32] Pieter De Leenheer, Prof. Dr. Robert A Meersman, “XML / RDF / RDFS / OWL / SOAP”, in the courses “Web-informatiesystemen (HasseltUniversity, Belgium)” and „Informatiesystemen (VrijeUniversiteitBrussel, Belgium)“, VUB STAR.Lab


[37] MPEG-1 Traces Archive, University of Würzburg, Dept. of Computer Science Würzburg/Germany, http://www-info3.informatik.uni-wuerzburg.de/MPEG

[38] Zeng/Liu, 1996, Rate shaping by block dropping for transmission of MPEG precoded video over channels of dynamic bandwidth, 4th ACM Multimedia Conference Boston/U.S.A., 1996


[41] International Telecommunication Union (ITU), ITU-T Recommendation H.263