ELSEVIER

Signal Processing: *Image Communication* 15 (2000) 347–363

SIGNAL PROCESSING: IMAGE COMMUNICATION

www.elsevier.nl/locate/image

The delivery layer in MPEG-4

G. Franceschini

CSELT, Centro Studi e Laboratori, Telecomunicazioni SpA, Via G. Reiss Romoli 274, 10148 Torino, Italy

Abstract

The MPEG-4 specifications have provided substantial progress in many areas of multimedia technology. Following MPEG tradition, MPEG-4 focuses on media coding. However, a couple of innovative aspects other than media coding characterise MPEG-4 with respect to its predecessors: the ability to code an audio-visual scene, and the ability to abstract from the delivery technology. This paper focuses its attention on this later aspect, which is covered by part 6 of the MPEG-4 specification: DMIF (Delivery Multimedia Integration Framework). The paper explains the motivations that have driven the delivery technology abstraction, analyses the details of the DMIF architecture, and highlights the practical impact on the "bits-on-the-wire" and on conformance issues. It is important not to forget, throughout this paper, that the whole focus of this work is on real-time delivery of multimedia content.

Keywords: Delivery; Synchronization; Interface; DMIF; DAI; QoS

1. Introduction

The MPEG-4 specifications have provided substantial progress in many areas of multimedia technology. Following MPEG tradition, MPEG-4 focuses on media coding. However, a couple of innovative aspects other than media coding characterise MPEG-4 with respect to its predecessors: the ability to code an audio-visual scene, and the ability to abstract from the delivery technology.

This paper focuses its attention on this later aspect, which is covered by part 6 of the MPEG-4 specification: DMIF (Delivery Multimedia Integration Framework) [8]. The main objective that motivated the DMIF effort has been the definition of a separate delivery layer, so that the MPEG-4 Systems specification does not enter the details of the various delivery technologies (differently from MPEG-1 and MPEG-2). Moreover, DMIF aimed at enabling the simultaneous access, presentation and synchronisation of MPEG-4 content carried through different delivery technologies (here, and in the rest of the paper, the term "delivery technology" is used in a broad sense, which includes storage technologies). These requirements motivated the definition of the DMIF reference architecture, including the definition of a uniform interface (the DAI – DMIF/Application Interface) such that an application making use of it is kept unaware of the delivery technology details. DMIF has also defined a generic signalling protocol able to fulfil the requirements for multimedia streaming, and has specified how to integrate it with other existing signalling protocols. Last but not least, DMIF has been designed as a tool that fully supports the

E-mail address: guido.franceschini@cselt.it (G. Franceschini)

0923-5965/00/$ - see front matter
PII: S0923-5965(99)00053-3

MPEG-4 Systems [4] features, but that can also be used in non MPEG-4 Systems-based applications. Valid MPEG-4 Systems-based implementations may exploit the DMIF reference architecture, if designed to access, possibly concurrently, multiple delivery technologies. Or develop their own, if targeted to specific environments. The adoption of the DMIF reference architecture, and in particular of the DAI, is a local terminal decision. Interoperability between terminals is only affected by the usage of common specifications for the particular delivery technology used, i.e. the only mandatory conformance point is defined at the interface to the outside world.

This paper explains the motivations that have driven the delivery technology abstraction (Section 2), analyses the details of the DMIF architecture (Section 3), shows how it works (Section 4) and highlights the practical impact on the "bits-on-the-wire" and on conformance issues (Section 5). It also briefly describes the DMIF related software developed in the IMplementation 1 (IM1) group of MPEG (Section 6). It is important not to forget, throughout this paper, that the whole focus of this work is on real-time delivery of multimedia content.

2. The delivery layer abstraction

2.1. Background

In MPEG-1 and MPEG-2 the transport of the encoded content represented an integral part of the specifications. MPEG-1 defined how to store data on a file, since local retrieval was the business case. Similarly, MPEG-2 developed solutions for the two promising business cases that it was facing: local retrieval (again) and TV broadcast over dedicated networks. Two different "Systems" of MPEG-2 were specified, Program Stream for the local retrieval, Transport Stream for the broadcast scenario. Each solution was optimised and specific for the addressed delivery technology. MPEG-2 TS (transport stream) [12] has been very successful, and is now the transport protocol used to convey the digital television signal over dedicated networks. However, it was designed and optimised for "raw" transport mediums. MPEG-2 TS thus includes Data Link layer features (such as sync byte and fixed packet length) as well as synchronisation features (such as Clock References and Time Stamps) at the same level, making these functionalities hardly separable. As a consequence, it is difficult and/or inefficient to transport MPEG-2 content over networks other than TV networks, since Data Link layer features already exist in computer networks. The carriage of MPEG-2 TS over IP (or ATM) thus requires either the removal of the replicated functionality (which is difficult) or the acceptance of a duplication of them (which is inefficient). In the IP case multiple solutions, each with its pros and cons, have been defined by IETF [16,3]; in the ATM case a single solution was finally approved by the ATM Forum [1], consisting in a quite inefficient mapping of a default of 2 MPEG-2 Transport Packets into eight ATM cells with AAL5.

As a conclusion, in MPEG-1 and MPEG-2 there was a quite clear initial focus on particular delivery technologies that resulted in a concentration of the efforts and in the generation of optimised but monolithic solutions. The later interest into other delivery technologies lead to somehow sub-optimal solutions.

2.2. The MPEG-4 approach to the delivery issue

Differently from its predecessor, MPEG-4 has been targeted since the beginning to adapt to multiple operating scenarios (local retrieval, remote interaction, broadcast or multicast) and delivery technologies. Instead of defining a number of optimised monolithic variations, the design choice was to abstract the functionality that the delivery layer has to provide, and focus the MPEG-4 Systems activity on the common features. The goal is still to produce effective solutions, however a demarcation has been drawn between the aspects that can be managed uniformly and independently from the delivery technology (included in MPEG-4 Systems), and those that instead are impacted by the delivery technology and by the operating scenario (included in MPEG-4 DMIF). This demarcation line is named DMIF-Application Interface (DAI).

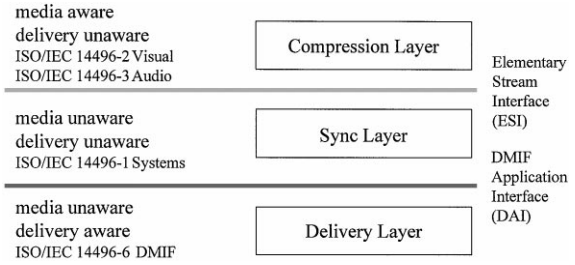

Fig. 1. MPEG-4 layered model.

This idea is depicted in Fig. 1 along with the generic MPEG-4 layered model, which comprises the Compression Layer, the Sync Layer (that is part of Systems) and the Delivery Layer (DMIF).

The Compression Layer performs media encoding and decoding of Elementary Streams and is specified in [5] and [6]; the Sync Layer manages Elementary Streams and their synchronisation and hierarchical relations and is specified in [4]; the Delivery Layer ensures transparent access to content irrespective of delivery technology and is specified in [8].

The separation of common and specific delivery tasks has allowed to describe complementary walkthroughs dealing, respectively, with the application aspects (Systems part) and with the delivery specific issues (DMIF part). In DMIF there are separate walkthroughs for each operational scenario, all showing the same behaviour at the DAI.

2.3. The advantages of the Systems/DMIF approach

Since MPEG-4 targets multiple delivery technologies, the definition of a single, common synchronisation layer syntax has been considered as a useful unifying factor. This way all the details related to the delivery technology are kept in the delivery layer, below the synchronisation layer. This makes it possible, by adopting the DMIF architecture, to access, present and synchronise MPEG-4 content transmitted over different delivery technologies, such as MPEG-2 TS and IP multicast. Also simultaneously. Latest activities related to the carriage of MPEG-4 signals in MPEG-2 TS [14] or over RTP (Real Time Protocol, [18]) [2] have reduced the synchronisation layer commonality to its semantic, in order to adopt (and adapt to) the syntax already used in those contexts. This change in philosophy could create additional complexity in gateways, but does not prevent the ability to simultaneously access, present and synchronise MPEG-4 content transmitted over different delivery technologies.

2.4. The requirements at the base of the DMIF reference architecture

The DMIF reference architecture is based on the requirement of hiding to the application the delivery technology details as well as the operational scenario. DMIF thus considers uniformly three major families of technologies that are based on local retrieval, remote interactive or broadcast scenarios (see Fig. 2). The definition of an architecture that allows fulfilling the above requirements represents an important step to favour the development of truly multimedia applications. The identification of a common interface (from an application's point of view) to today's and tomorrow's networks, encompassing both the QoS enabled networks, and the best effort model, is the first DMIF requirement (that has been partially reached in Version 1, since the best effort model is not yet fully supported). This is particularly useful in relation to Quality of Service (QoS) issues. As of today the most widely used network is the Internet, which is currently based on best effort techniques. However, developments for adding QoS management on IP-based networks already date several years (RSVP: ReSerVation Protocol), while other solutions, with quite different definitions in terms of QoS parameters, have been produced in the ATM context. As a result multiple QoS models are defined and more or less used (and maybe others will be defined in future). At least intranets could be soon able to provide QoS on selected transport technologies. Before DMIF, no unified interface was defined to manage these different models, making the development of QoS demanding application strictly related to the technique used, thus delivery technology aware.

The second, less obvious requirement of the DMIF reference architecture is to also hide the

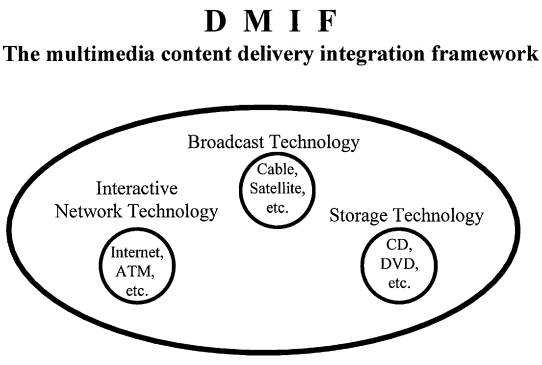

Fig. 2. DMIF addresses the delivery integration of three major technologies.

operational scenario details to the application. It means managing the access to locally or remotely retrieved streams, as well as broadcast/multicast streams, through a common interface to the delivery system. The reasons for this requirement are not that evident, since broadcast content is certainly designed with different criteria than content meant to be interactively retrieved. By implementing this requirement the obvious differences between operational scenarios would however have no impact on the interface, nor on the way the application manages the streamed content, but just on the authoring process. For example, MPEG-4 scenes meant to be used in a broadcast environment will not activate features like reverse playing that would be possible instead when playing content designed to work in a local or remote retrieval environment. The advantage of this approach is that it simplifies the design of applications using both broadcast/multicast and retrieved content. As an example, multimedia conferencing applications could make use of a mixture of retrieved and multicast content without any need to differently manage the streamed multimedia information received. IP multicast could be used for audio and video streams representing the conferee, or for slide shows; IP unicast for "private" conversations; IP unicast or multicast to access content external to the conference itself. Even pre-downloaded content could be locally played. All this could be integrated in a single, harmonised presentation. Moreover, the same application could perform quite differently in an intranet able to manage QoS, or in the best effort Internet. This would have not been possible if a uniform interface were not defined.

3. The key elements of DMIF

There are a few fundamental elements that characterise DMIF: the reference architecture, the DMIF-Application Interface and the DMIF Signalling Protocol. These elements are presented in the following subsections.

3.1. The reference architecture

The DMIF reference architecture is represented in Fig. 3. The picture shows how the different operational scenarios are uniformly modelled, through the identification of four basic blocks: originating application, originating DMIF, target DMIF, and target application. The elements in the upper part are meant to be part of the originating terminal, while the elements in the bottom are part of the target terminal (in the case of a remote interactive scenario).

The originating DMIF module is meant to work in co-operation with the target DMIF module to provide a session-level service. The distinction between originating and target DMIF modules in the local retrieval and broadcast scenario is a bit artificial, but has been left for uniformity with the remote interactive scenario.

The originating application is the actual application in the terminal, e.g. an MPEG-4 browser, or a multimedia conferencing application. It is assumed that it has in all cases a counter part, the target application. The originating application interacts with the target application through DMIF. In the case of remote interactive operational scenarios, the two applications typically reside on separate hosts, and the communication between the two is regulated by some signalling protocol, not known by the applications themselves. DMIF has elaborated a generic protocol to carry these signals – the DMIF Signalling Protocol – , which is further explained in Section 3.3. In the case of local storage and multicast/broadcast scenarios, the target application resides in the same terminal. In such

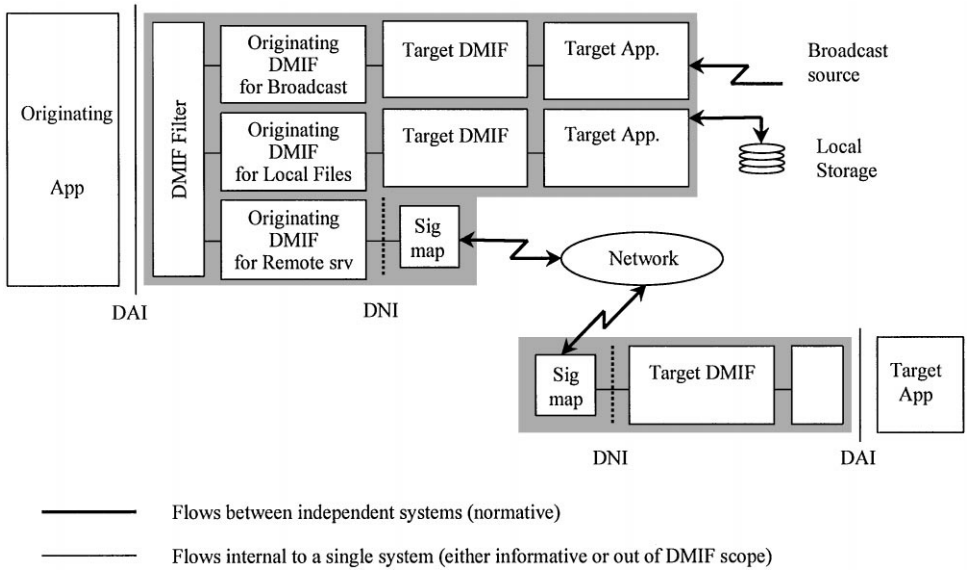

Fig. 3. DMIF communication architecture.

scenarios, the target application will likely not correspond to a "real" application, i.e. to a process, but will still keep its conceptual role.

There are elements, at the DMIF-Application Interface, that are considered opaque to DMIF, and only understood by the target application. This is true regardless of the operational scenario and of the actual form that the target application assumes. For instance MPEG-4 Elementary Stream ID are not explicitly communicated at the DAI, but passed as opaque data; only the target application has knowledge that such data actually carries ES_IDs. Another example relates to user commands, that are carried transparently by DMIF, and only understood by the target application. This approach allows to keep DMIF independent from evolutions in the applications (MPEG-4 ES_IDs may not be the universal method to identify streams; moreover not all scenarios need to be aware of the full list of user commands, and this list could be quite application dependent). Thus, DMIF is not limited to MPEG-4 Systems-based applications.

Fig. 3 also shows further elements, such as the DMIF Filter. This module is identified in the DMIF specification to highlight the potential benefit of the architecture. It represents a sort of container for the various DMIF instances available in a terminal, and its role is to select the appropriate DMIF instance to provide a certain service, based on the DMIF URL requested by the application. The DMIF Filter (if implemented "smart") is the architectural element that enables the plug & play of DMIF instances without any need to re-compile nor re-link applications. An example of such a "smart" implementation is given in Section 6. The existence of a DMIF Filter does not impact the DMIF-Application Interface as defined in DMIF, but only the ability to support multiple, not initially known DMIF instances. This architectural feature is important for systems willing to support not yet known protocols/network environments, and allows DMIF based applications to automatically make use of new delivery technologies.

Another module shown in Fig. 3 is the Signalling module (Sig Map). This element applies only to the DMIF instances for remote interactive scenarios, and is kept separate from the other DMIF elements in order to highlight the role of the DMIF-Network Interface (DNI). Particularly for the case of remote interactive systems, DMIF factorises features that do not depend from the actual underlying

transport technology (generic message flows and protocol state machines) from features that are instead specific of a certain network (such as signalling protocols). The DNI represents the border between the generic and specific tasks of a DMIF instance for remote interactive scenarios.

3.2. The DMIF-application interface

The DMIF-Application Interface (DAI) is a semantic API that directly derives from the requirement of hiding the delivery technology and operational scenario details to the application: thus local and remote retrieval are not different, at this API, from multicast or broadcast. An MPEG-4 browser using this API would be therefore able to access and present multimedia content uniformly and independently of the operational scenario. In the MPEG-4 context the DAI formalises the demarcation line between part 1 (Systems) and part 6 (DMIF), and separates the elements and tools of MPEG-4 that are conceptually network unaware (defined in Systems) from those that instead relate with the delivery technology (covered by DMIF); as already mentioned however, the DMIF specifications and the DAI in particular are applicable as well in other contexts. A significant part of the design effort has been devoted to the identification of the parameters that should be exposed at this interface, focusing the study on the semantic value that such parameters should carry. The scope rules represent, maybe, the most significant value of this interface, which is otherwise quite simple and straightforward. These rules define which parameter have to be dealt with by which module in the DMIF architecture: some parameters thus appear as opaque data at the DAI.

The DMIF Application Interface comprised the following classes of primitives:

- Service primitives, which deal with the Control Plane, and allow the management of service sessions (DA_ServiceAttach( ) and DA_ServiceDetach( ));
- Channel primitives, which deal with the Control Plane, and allow the management of channels (DA_ChannelAdd( ) and DA_ChannelDelete( ));
- Data primitives, which deal with the User Plane, and serve the purpose of transferring data through channels (DA_Data( ) and DA_UserCommand( ), respectively, for real data and application control data).

In addition to these primitives, additional functions to initialise, reset, query the status, etc. have to be implemented by a "real" interface. Moreover, a "real" interface will be characterised by a specific programming language as well as by the detailed definition of its syntax. DMIF Version 1 specification deliberately chose to not specify this kind of details which have no impact on the concept and on the model of DMIF, and limited itself to the definition of just the primitives listed above and of the exact semantic meaning of the relevant parameters.[1]

This simple set of primitives (and associated parameters) is sufficient to fully describe the behaviour of a DMIF instance. Moreover it implies, from an MPEG-4 Systems perspective, the exact definition of the normative behaviour of an MPEG-4 receiver (MPEG-4 Systems defines, in terms of DAI primitives, the operations that an MPEG-4 receiver should perform to get access to and then control streamed content).

Differently from the rest of the MPEG-4 specification, the DMIF part addresses both receiver and sender roles. This has been done in order to extend the delivery unawareness philosophy from consumer only applications (which only act as receivers) to conversational applications as well (which play both the receiver and sender role). This has also allowed to maintain uniformity when describing the behaviour through complete walkthroughs. However, many compression techniques combine source and channel coding, and this is not yet supported by the DAI defined in DMIF Version 1. It is the author's belief that the DMIF sender side requires additions to actually support these techniques, while the DMIF receiver side is less troublesome. Nevertheless, it should be clear that the adoption of the DMIF architecture is only

[1] There are currently demands for defining in DMIF Version 2 a precise syntax for C++ and Java.

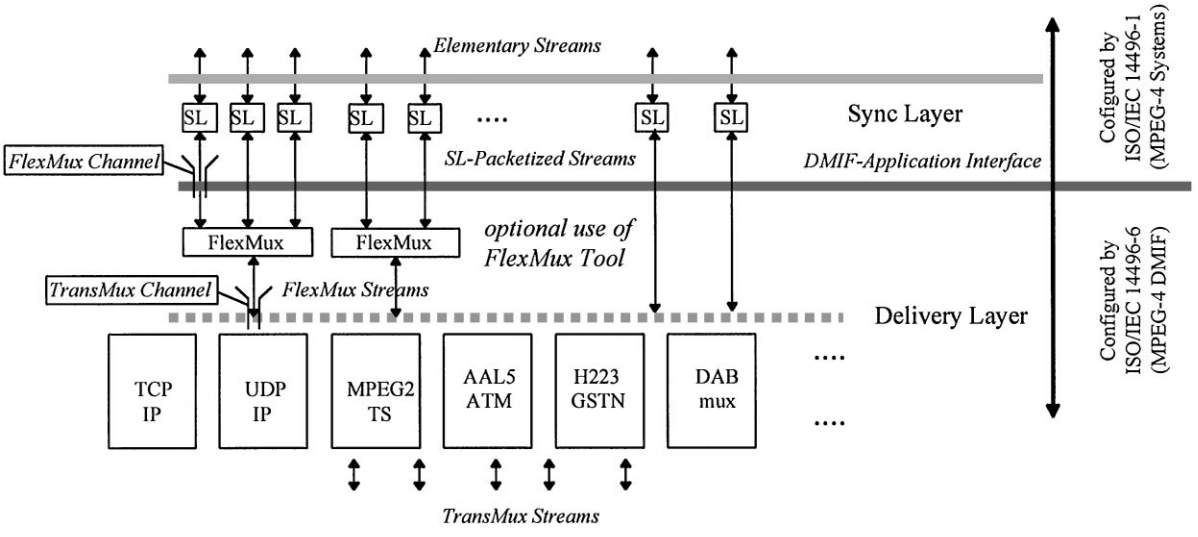

Fig. 4. The User Plane in an MPEG-4 terminal.

a local matter, which does not affect interoperability between equipment.

Fig. 4 illustrates the User Plane in an MPEG-4 terminal, and specifically in the Delivery Layer: Elementary Streams cross the DAI in individual channels and are possibly multiplexed/demultiplexed in the Delivery Layer, thus generating the so-called FlexMux Streams that are carried into the so-called TransMux channels. TransMux channels are in turn multiplexed by means that are characteristics of the protocol stack used (e.g. IP port numbers, ATM VCs, MPEG-2 PIDs ...).

The Delivery layer is responsible for the configuration of the transport protocol stacks. Each DMIF instance is in charge of configuring the exact protocol stack for each channel, and of keeping track of the associations of channels and transport resources. Fig. 4 provides a sample of the (hypothetical) choice of native transport protocol stacks. It outlines the fact that either an Elementary Stream or a group of streams multiplexed together (e.g., with the MPEG-4 Systems FlexMux tool) can be carried over a native transport.

3.3. The DMIF Signalling Protocol

The DMIF Signalling Protocol is a generic session level protocol designed to fulfil the requirements for multimedia data streaming. The protocol has been conceived with a look into the possible future evolutions in networking technologies, and supports features that are not readily available with current techniques. Such features are QoS and resource management and operation over a variety of networks, including heterogeneous networks. It also allows logging of the resources consumed in a session. The DMIF Signalling Protocol is used to configure the protocol stacks at the peer ends. The protocol stack may also include the FlexMux tool specified in MPEG-4 Systems, and this possibility is meant to be exploited by DMIF implementations to optimise the network resource consumption (such as the number of sockets used and QoS requirements for each network connection). The DMIF specification does not specify how to perform such optimisation, but provides the means that make it possible.[2]

The DMIF Signalling Protocol mostly derives from the MPEG-2 DSM-CC User-to-Network Protocol, which was specified in ISO/IEC 13818-6 [13] and meant to provide a signalling mechanism

[2] The latest trend in the joint IETF-MPEG activity for the carriage of MPEG-4 signals over the Internet is to not exploit the MPEG-4 Systems Flexmux tool, but an RTP multiplex (yet to be completely defined).

for heterogeneous networks. It differs from DSM-CC U-N in that it does not discriminate between a client and a server role, thus allowing for not just one-way communication. On the other hand, DMIF Version 1 is only focused on homogeneous networks, while future DMIF versions are expected to cover the heterogeneous case as well. In other words, the DSM-CC U-N protocol was designed to solve a quite specific problem: session and connection set-up for video-on-demand services over heterogeneous networks (access network technology different from core network technology). The DMIF Signalling Protocol is designed instead to cover a much more generic case and focuses first on homogeneous technologies (such as Internet).

The DMIF Signalling Protocol is a session-level protocol. As such, it is somehow equivalent to the FTP. In both cases the first step consists in opening a session with a peer entity. This process may include authorisation and other security checks, which are however out of the scope of the DMIF Signalling Protocol, that only supports the carriage of user data. Once the session has been established, a number of streams are selected and requested, just like files are selected and requested in an FTP session. However, in FTP files are immediately downloaded, sequentially, on a specific socket. The DMIF Signalling Protocol instead creates one channel for each requested stream, possibly multiplexing multiple channels into a single socket by means of the FlexMux tool. It does not download the stream, not even start the streaming; it simply sets up the channels and configures the protocol stack. It is then up to the application to control the streaming.

This protocol makes use of resource descriptors. Resource descriptors are a generic, easily extendible mechanism that makes any kind of resource equally manageable by the protocol. Thus IP port numbers and addresses are not different than ATM VCs or VPs, or MPEG-2 TS resources. It is also envisioned to use resource descriptors to require transcoders or mixers, although there is currently no fully specified walkthrough for such scenario. The generic protocol is then made specific for each specific network technology by integrating it with the relevant signalling tools that characterise that particular network. In cases where the native signalling tools are able to provide functionality defined also by the DMIF Signalling Protocol, these functionality are mapped instead of being duplicated, thus optimising the total number of exchanged messages. Specific mappings of the DMIF Signalling Protocol have been defined in DMIF Version 1 for IP, IP with RSVP and ATM with Q.2931, while in DMIF Version 2 [10] an additional mapping is being defined on top of H.245 signalling (for H.324 -mobile- terminals).

The DMIF Signalling Protocol supports the exchange of generic QoS information, and can be therefore exploited over QoS enabled networks. However other existing tools can be used instead. In that case features which are unique to the DMIF Signalling Protocol, such as the ability to configure the MPEG-4 FlexMux, would be sacrificed. A potential competitor to the DMIF Signalling Protocol in the Internet environment is the Real Time Streaming Protocol (RTSP, [19]) defined by IETF. The specification of a DMIF instance supporting RTSP is envisioned as one of the possible extensions to the DMIF Version 1 specification. The RTSP however mixes the roles of session and connection set-up with the role of Stream Control, that in MPEG-4 Systems and especially in DMIF have been kept separate. This makes the integration of RTSP in the DMIF framework a bit more complex.

4. Delivery of MPEG-4 content

4.1. The MPEG-4 tools

Three tools are used to fully describe an MPEG-4 scene and locate its components:

- a scene description stream (BIFS), that describes how a scene is composed, and refers to its components through pointers to Object Descriptors;
- an Object Descriptor (OD) Stream, that describes each individual Elementary Stream in a scene, including its unique identification (a 16 bit integer – ES_ID – , which is also used for internal cross-references, or optionally also an unconstrained identifier to locate the stream by means other than a 16 bit number – the ESURL – , which is however not used for cross-references);

- a Stream Map Table that maps each ES_ID (or ES URL) to its actual physical location.

Obviously, in addition to the above information, the actual audio-visual content is streamed as well.

A consequence of this organisation is that the scene description does not depend on the underlying delivery technology, which allows to author the BIFS stream just once, and use it in several different contexts. The Object Descriptor Stream represents a fully specified and almost completely delivery-independent mechanism to carry the Object Descriptor Protocol. It is believed that the OD Stream will be used in conjunction with the BIFS Stream, but in principle, for restricted environments not requiring the full set of MPEG-4 features (e.g. QoS or IPMP) it is conceivable that mechanisms other than an OD Stream may carry the reduced OD Protocol features. Anyway, there is no such legacy system identified so far. The third tool mentioned above is instead deeply dependent on the delivery mechanism.

4.2. The DMIF instance responsibilities

For each delivery technology a DMIF instance needs to be specified which addresses:

- the control plane signalling tools,
- the protocol stack for the transport of MPEG-4 elementary streams,
- the mechanism to carry the Object Descriptor Protocol (usually: an OD stream),
- the mechanism to carry the Stream Map Table information.

All these aspects are essential for interoperability purposes. The actual specification process for each of the above issues need not be restricted to DMIF or MPEG. Anyway, from an MPEG-4 Systems perspective, a single, uniform walkthrough can be depicted independently of how the above issues are solved in the various cases.

4.3. The generic walkthrough

The DMIF reference architecture states that the originating application accesses the multimedia content through the DMIF-Application Interface. The application requests are processed by a DMIF Filter, which determines the actual DMIF instance that should serve the request. The application will not have any direct knowledge of the DMIF instance actually activated: this is completely transparent. An application can also concurrently use more than one DMIF instance.

The generic walkthrough for an application based on MPEG-4 Systems starts with the selection of the service to activate. The service is identified by a DMIF URL, and may further incorporate other such URLs, that from a DMIF point of view will refer to new "services". The originating application thus requests the activation of a service, and uses the DA_ServiceAttach primitive of the DAI to create a service session. The DMIF Filter examines the DMIF URL passed by the application, and determines the originating DMIF instance in charge of providing the service. This module contacts the target DMIF instance (trivially, in the case of local storage and broadcast scenarios; using the DMIF Signalling Protocol – or an equivalent mechanism – in the case of remote interactive scenarios). The target DMIF instance in turn identifies and contacts the target application; it also establishes a service session with it. In the case of remote interactive scenarios a network session is established as well, with network-wide significance, and locally mapped by each DMIF peer to a locally meaningful service session. The target application finally locates the service and, if existing, returns a positive answer that is forwarded back to the originating application. In the case of applications based on MPEG-4 Systems, the answer will also include the (Initial) Object Descriptor for the requested service. The application peers then use the service session to create connections that are used to transport streamed data.

For an application based on MPEG-4 Systems the next action would consist in the parsing of the Initial OD, and in the request to open channels for the streams described there (typically, the BIFS and OD streams). When the originating application requests streams, it uses the DA_ChannelAdd primitive of the DAI, indicating the Service they belong to. It also indicates the requested streams in the user data parameter, which is opaque to the DMIF instance and transparently delivered to the

target application. The target application, by means that depend on the specific delivery technology (in the case of local storage and broadcast scenarios) or on the server implementation (in the case of remote interactive scenarios), locates the desired streams. In the case of remote interactive scenarios the peer DMIF instances set-up the requested channels using the appropriate protocol (such as the DMIF Signalling Protocol for that specific network). The QoS parameters that have been exposed at the DAI in association with each requested stream may influence the set-up of network connections, depending on the criteria that the sender side adopts to aggregate multiple elementary streams into a single socket (by means of the MPEG-4 FlexMux or any other – to be specified – multiplex technique).

The sending application peer then uses the channel to deliver the streamed data (DA_Data primitive of the DAI), according to the stream control commands delivered by the receiving application with some other mechanism (possibly the DA_UserCommand primitive at the DAI).

An application based on MPEG-4 Systems would, at this point, receive the BIFS and OD streams, parse them, and request additional elementary streams, again through the DA_ChannelAdd primitive of the DAI.

When a stream is no more requested, the originating application uses the DA_ChannelDelete primitive of the DAI to free the channel. When a service is no more requested, the originating application uses the DA_ServiceDetach primitive of the DAI to free the service.

Fig. 5 provides a high-level view of a service activation and of the beginning of data exchange in the case of interactive scenarios; the high level walkthrough consists of the following steps:

1. The originating application requests the activation of a service to its local DMIF instance: a communication path between the originating application and the originating DMIF instance is established in the control plane (1), and associated to a locally meaningful service session.
2. The originating DMIF instance contacts the target DMIF instance: a communication path between the originating DMIF instance and the target DMIF instance is established in the control plane (2), and associated to a network unique network session.
3. The target DMIF instance identifies the target application and forwards the service activation request: a communication path between the target DMIF instance and the target application is established in the control plane (3), and associated to a locally meaningful service session.
4. The peer application creates channels (requests flowing through communication paths 1, 2 and 3). The resulting channels in the user plane (4) will carry the actual data exchanged by the applications.

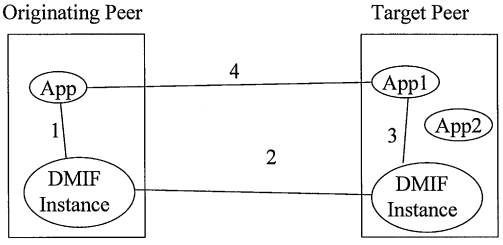

Fig. 5. DMIF computational model.

4.4. Broadcast scenario

When accessing content in a broadcast or multicast scenario the service is represented by a bundle of streams that are delivered over a set of channels. When the application requests a particular service (identified through a specific DMIF URL), the target application module is supposed to parse the URL and determine the service requested, retrieve the Initial OD for that service (to be returned to the originating application) and the associated Stream Map Table (to be maintained internally). Once the target application module has retrieved this piece of information, it is able to satisfy further originating application requests for adding channels: by comparing the requested ES_ID (or ES URL) with the appropriate elements in the table, it will locate the actual physical channel carrying the requested stream. The Stream Map Table could be conveyed in a number of ways. Current standardisation activity is focusing in MPEG-2 TS broadcast, in which case the Stream Map Table is conveyed by extending the MPEG-2 Tables [14]. In the case of

IP Multicast there are a few hypothesis currently under consideration: extend and use the Session Description Protocol (SDP, [17]) in combination with the Session Initiation Protocol (SIP, [11]) or the Session Announcement Protocol (SAP, [15]); or define a new multicast protocol to fulfil the peculiar MPEG-4 requirements: there is an effort in this direction in DMIF Version 2 [10]. Some more ideas are given in Section 6.4.2. The main issue here is that the Stream Map Table is, in general, dynamic, since channels can be added or removed during a session. While static Stream Map Table would be easily managed with a few extensions of existing tools, dynamic tables are not. The number of receivers and the possibility to accommodate late tune-in are additional elements that impact the performances of a solution. It is even conceivable that a mechanism for retrieving static tables is adopted as a base, and algorithms for dynamic updates are applied for just the dynamic portion of them. In any case, no stable solution is available at the moment.

4.5. Local retrieval scenario

When accessing content in a local retrieval scenario the service is represented by a set of files that co-operatively provide the required content. When the originating application requests a particular service (identified through a specific DMIF URL), the target application module is supposed to parse the URL and determine the service requested, retrieve the Initial OD for that service (to be returned to the requesting application) and the associated Stream Map Table (to be maintained internally). The MPEG-4 file format (being specified in MPEG-4 Systems Version 2 [9]) defines how to describe these elements. Once the target application module has retrieved this piece of information, it is able to satisfy further originating application requests for adding channels: by comparing the requested ES_ID (or ES URL) with the appropriate elements in the table, it will locate the actual content providing the requested stream.

4.6. Remote retrieval scenario

When accessing content in a remote retrieval scenario the service is organised by the server. When the originating application requests a particular service (identified through a specific DMIF URL), the originating DMIF instance is supposed to parse the initial part of the URL and determine the server address; the target DMIF instance in turn will identify the application executive running the service requested, and a logical connection between originating and target applications will be established. The server will provide the Initial OD for that service, that will be returned to the originating application, but not the Stream Map Table. The Stream Map Table will be instead generated incrementally on demand, as a result of the application requesting a stream: this is due to the fact that the Stream Map Table provides a map between Elementary Streams and physical resources, and that such resources are made available, in such operational scenario, only on demand.

4.7. Other scenarios

Heterogeneous scenarios are already partly supported by DMIF Version 1 specifications. Heterogeneity is supported at a terminal, meaning that several different delivery technology (and operational scenarios) may co-exist and even run concurrently and co-operatively. Heterogeneity is not yet supported instead on a single connection: in other words, it is not currently possible to deliver MPEG-4 content over a sequence of different delivery technologies (e.g. MPEG-2 TS broadcast followed by IP multicast). This is the kind of heterogeneity provided by the MPEG-2 DSM-CC U-N protocol from which the DMIF Signalling Protocol derives, and in future versions of the DMIF specification it is planned to cover again this functionality (consistently with the added DMIF features).

5. DMIF in the MPEG-4 context

5.1. The real extent of DMIF specifications (*and conformance issues*)

DMIF defines architecture, interfaces, protocols, URLs, etc. Apparently, DMIF provides a whole

new world that just ignores the existing components. At a closer look however, it is exactly the contrary. DMIF deliberately chose to specify the very minimum needed to pave the ground to the future exploitation of MPEG-4: it defines a reference architecture to consistently select and make use of the existing tools, but does not force the usage of any specific technology. The reference architecture specifies what a DMIF instance should provide, in terms of parameters and behaviour at the interface with the application. It provides a semantic specification of the DMIF-Application Interface, to clearly define the scope of a DMIF instance, and defines the parameters exposed at the DAI based only on the semantic they carry, without binding the interface to any specific current technology: for example it supports a generic template for the carriage of QoS metrics, to open the door to future exploitation of QoS-enabled technologies. The DMIF reference architecture finally specifies the behaviour (walkthrough) of a DMIF instance for each of the envisioned scenarios. The reference architecture (and the DAI) are just a local matter for a terminal and do not impact on interoperability between equipments. Interoperability is affected instead by the particular DMIF instances specifications, which define for both control and data plane the actual bits-on-the-wire. Current and future envisioned specifications relate to MPEG-4 File Format, IP, IP multicast, MPEG-2 Transport and Program Streams, DAB, ATM and H.324 terminals. As a consequence conformance for DMIF primarily addresses the bits on the wire, in particular the DMIF Signalling Protocol, and only optionally the DAI.[3] It has to be taken into account, however, that the DMIF instances specifications are defined in conjunction with other standardisation bodies, which creates issues of "jurisdiction" for conformance testing.

5.2. The impact on Systems specification

DMIF has marginally impacted on the MPEG-4 Systems specification, in a way however which is very important to guarantee future evolutions. MPEG-4 Systems has, with the DAI, a clear, delivery and operational scenario-independent mechanism to describe the behaviour of an MPEG-4 Systems receiver. Systems walkthroughs can thus be maintained unique and general. Also, in DMIF, all walkthroughs start at the DAI, so that all delivery technologies are analysed consistently. One major advantage of the DMIF effort consists in the careful definition of the parameters to be exposed at the DAI, in their semantic meaning and scope rules. This study, as well as the equivalent study of the parameters to be included in the DMIF signalling protocol, makes sure that the interface is not tied to the specificity of a particular technology (such as the MPEG-2 TS PIDs and Association Tags), and that could be used also in heterogeneous environments. The process of abstracting the semantic requirements from the technology and operational scenario details has not been trivial, and this is proven by the effort which is required to consistently define the details of the DMIF instances specifications. The DMIF reference architecture and the DAI in particular thus make sure that different delivery technologies and operational scenarios can be used without affecting the Systems features. As far as the bits-on-the-wire are concerned, DMIF has allowed formalising how cross-references between services should be resolved. In order to uniformly manage cross-references it has been decided to make use of a common syntax: the URL. Thus cross-references in ObjectDescriptors, which actually point to another ObjectDescriptor, have to be expressed in the form of URLs. Such URLs are part of the MPEG-4 content, thus DMIF has had a real impact here. Moreover, such URLs have a precise meaning: they do not point to individual content, but to a "service", that hides the content. The "service" is actually identified in MPEG-4 by an (usually Initial)Object Descriptor.

As a conclusion, DMIF only minimally impacts on MPEG-4 Systems, but by making sure that the various tools are used coherently, it is extremely helpful also as a formal instrument to verify the consistency of specifications such as the MPEG-4 File Format, or for the integration in the MPEG-4 world of existing tools such as RTSP.

[3] A conformance test suite for the DAI will be added if DMIF Version 2 will specify a precise syntax for the DAI.

6. An implementation: IM1

The DMIF reference architecture has been already validated by some implementation. One such implementation has been developed by the IM1 group (IMplementation 1), in the MPEG community. This group was activated in April '97 to verify the functionality of the MPEG-4 Systems specification, and the software produced has progressed with time, including also the DMIF architecture. The IM1 software is freeware and will be included in the MPEG-4 reference software [7]; additional modules exist that have been integrated in IM1 but not donated. As far as the DMIF part is concerned, the key aspects of the reference architecture constitute an integral part of the IM1 freeware code. They include the DMIF Filter and the MPEG-4 FlexDemux, as well as a DMIF instance for accessing streaming content from local files (in both a so-called TRIvial File format – TRIF – , defined internally to IM1, and MPEG-4 File Format – MP4 –). The author has participated in the definition of the interfaces and lead the development of various other DMIF instances (for delivering MPEG-4 content over MPEG-2 Transport Stream, over IP multicast and over IP unicast). The various elements of these implementations are described in the following sections.

6.1. The DMIF architecture

The essential elements in the IM1 software for the DMIF architecture are represented in Fig. 6, highlighted in grey. Note that the picture references the IM1-2D Player, which represents the application against which the DMIF part has been extensively tested. The DMIF software architecture in IM1 only considers the "client" side that acts as a receiver only. It is basically made of the DMIF Filter and of the DMIF instances. Moreover, the implementation effort obviously includes the integration of the DMIF software in the "core" of the MPEG-4 Systems code. The DMIF part, as all the IM1 software, is implemented in C++, for the WindowsNT(4.0) operating system.

6.2. The interfaces

The IM1 implementation defines the DMIF-Application Interface (DAI) as a set of methods of two classes, one implemented in an object in the DMIF Filter, the other one implemented as a sink object

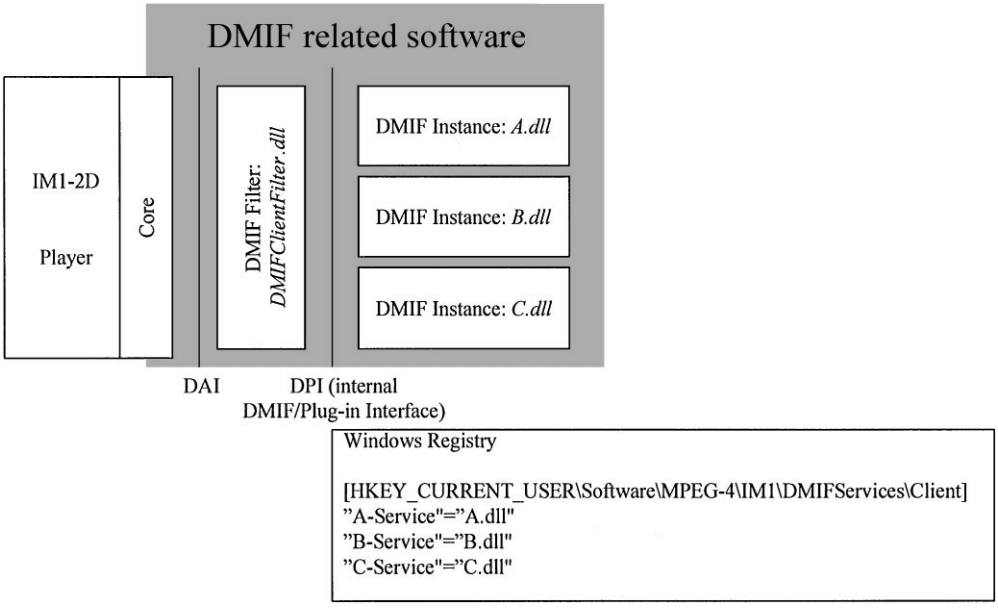

Fig. 6. Software elements in the IM1 implementation for DMIF.

in the application, for callbacks. The interface between the DMIF Filter and the various DMIF instances (DPI – DMIF/Plug-in Interface) is realised as well by means of a couple of classes definitions, one for each direction of the flow. In particular the interface defines a base class from which all DMIF instances shall inherit their own specific derived class. The derived class characterises the DMIF instance, but the DMIF Filter only invokes the methods defined in the base class. This allows the DMIF Filter to control DMIF instances that are not known in advance, and thus ensures one key feature of the DMIF architecture, that is to allow fully independence of an application from the delivery technology used.

6.3. The DMIF Filter

The DMIF Filter implementation is the key element to support the DMIF reference architecture, since it represents the entity that is able to identify and load the appropriate DMIF instance, and make sure that new DMIF instances can be used without any need to re-compile or re-link any application code. The originating application issues a DA_ServiceAttach( ) call providing the URL identifying the required service. This function is implemented by the DMIF Filter, which examines the URL to identify the correct DMIF instance to activate. This step is performed, in the IM1 implementation, with the help of the Windows registry: all available DMIF instances are in fact registered in a well-known branch of the registry (DMIF instances are implemented as DLLs in IM1). The DMIF Filter, based on the registry content, loads, tests and possibly unloads the registered DLLs, one at a time, until the correct one is found. Each DLL represents a DMIF instance implementation, and provides a particular method to test whether it is capable to access a service as identified by the URL or not. With this approach, a new DMIF instance can be added at run time in a system, or just updated, and be activated seamlessly through appropriate URLs. Once a DMIF instance (DLL) has been activated, the DA_ServiceAttach( ) call is forwarded to it for further processing; further flows through the DAI are then directly processed by that DMIF instance.

6.4. DMIF instances in IM1

Five DMIF instances have been integrated so far in the IM1 platform, with the goal of validating the key concept of the DMIF reference architecture, that is the ability to hide to the application the delivery technologies as well as the operational scenarios. Thus, at least an instance for each scenario (local retrieval, broadcast, multicast and remote retrieval) has been developed and successfully integrated.

6.4.1. The local retrieval scenario

Part of the free software developed in IM1 is represented by the DMIF instances for local retrieval. There are two such instances: one for reading files formatted according to a "Trivial File Format" internally defined and developed in IM1; another one for reading files formatted according to the "MPEG-4 File Format" being specified in MPEG-4 Systems Version 2. The first one has been widely used for tests and demos in the past, while the second one has been recently released, and has a value as reference software.

6.4.2. The multicast scenario

A DMIF instance has been integrated in IM1 to work with IP multicast. This instance, whose implementation has been lead by the author, has not been freely released, and does not represent any "official" use of MPEG-4 over IP multicast: the correct mechanism to use MPEG-4 in combination with the IP multicast technology is instead still under study. This implementation simply aimed at demonstrating the conceptual correctness of the DMIF reference architecture and of the DAI, in particular its ability to uniformly handle the multicast scenario. This instance is implemented as a single DLL and has been used in various demos.

The DMIF instance implemented simply adopted the UDP/IP protocol stack for carrying the MPEG-4 (flexmultiplexed) elementary streams, while the current joint IETF-MPEG activities are working on a Real Time Protocol based solution [2]. The Object Descriptor Protocol is carried, in this implementation, as a usual OD stream. While conceptually a different approach is possible

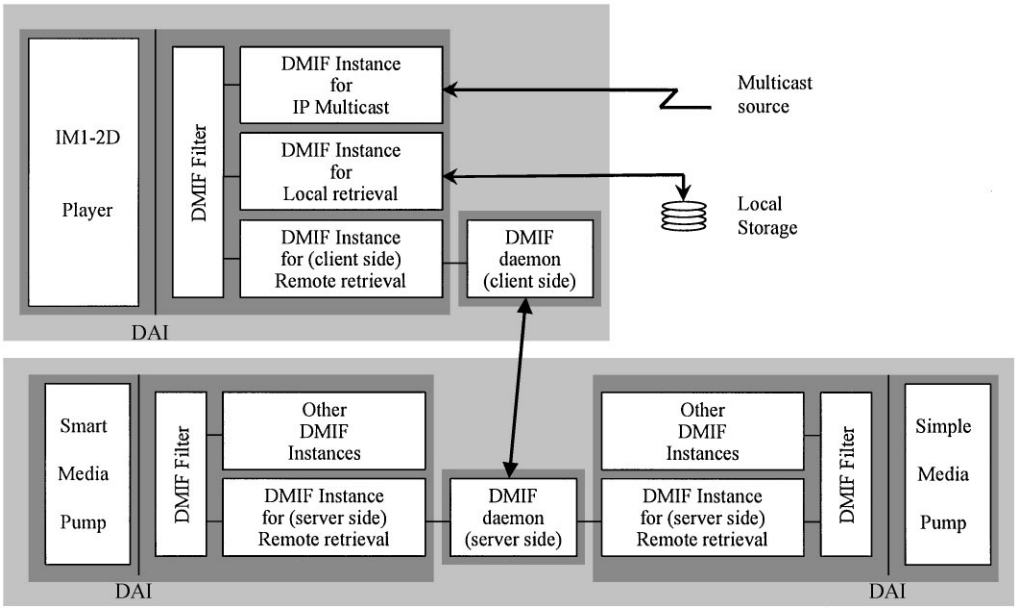

Fig. 7. Software modules in the DMIF implementation integrated in IM1.

(e.g. appending OD information as part of a Session Description Protocol [17] description), the selected solution was the simplest to implement (and is probably also the best solution).

6.4.3. The remote retrieval scenario

The retrieval scenario implementation is by far the most complex, since it includes also a server part and a connection management entity. The implementation integrated in IM1, that has been lead by the author and has not been freely released, is made of the following modules:

- DMIF instance (client side) for remote retrieval (DLL loaded in the IM1 player process space).
- DMIF daemon (client side) for connection management.
- DMIF daemon (server side) for connection management.
- DMIF instance (server side) for remote retrieval (DLL loaded in the target application process space).
- DMIF Filter for the server side.
- target applications (a simple media pump, for basic play/stop commands, and a smart media pump, for additional stream controls – e.g., pause, fast forward, … -).

A comprehensive scheme of the modules integrated in the IM1 platform is shown in Fig. 7, where the light grey area defines the borders of a terminal, the dark grey area defines the borders of a process, and each box corresponds to a separate software module (library or application).

This implementation makes use of the DMIF Signalling Protocol for IP networks for establishing the communication channels. This implementation supports Quality of Service management only in terms of interfaces and of management of unconstrained QoS parameters; however no "real" QoS is supported yet. A number of activities are in progress, in contexts other than IM1, to implement the monitoring of the objective QoS delivered (e.g. RTP/RTCP for the Internet), and strategies for aggregating Elementary Streams into FlexMux Streams.

Activities are also in progress to unify the client and server interfaces, as well as the client and server DLLs and daemons, thus allowing the exploitation of this software for applications based on MPEG-4

but other than an MPEG-4 browser (e.g. conferencing applications, where a terminal acts as both a receiver and a transmitter).

7. Conclusions

The DMIF specification defines an architecture that is open to future evolutions in the delivery technology, and that is able, if actually implemented in the terminals, to protect investments in the development of multimedia applications. DMIF also provides support, with the DMIF Signalling Protocol, to log resource consumption in a session, which represents the minimum requirement to enable per-consumption charging models. It therefore represents an interesting piece of technology for both application developers and telecom operators. The correctness of the DMIF reference architecture concepts and specifications (the DAI) has been validated and demonstrated by the IM1 implementation, and the key software modules are part of the reference code for MPEG-4.

As far as interoperability is concerned, it only relates to the specification of the particular DMIF instance to be used in the addressed delivery technology, not to the adoption of the DMIF reference architecture in a terminal.

Acknowledgements

This paper reflects the hard work of various individuals, but a special acknowledgement is due to Vahe Balabanian, chairman of the MPEG Delivery group during the development of DMIF Version 1, to Zvi Lifshitz, soul of the MPEG IM1 group, and to Mauro Rossi and Gianluca Di Cagno, who dedicated most of their time in developing, integrating and exercising the various DMIF software releases.

References

[1] ATM Forum, Audiovisual Multimedia Services: Video on Demand Specification 1.0, 1995.

[2] M. Civanlar, V. Balabanian, A. Basso, S. Casner, C. Herpel, C. Perkins, RTP payload format for MPEG-4 Streams, Internet-Draft draft-ietf-avt-rtp-mpeg4-01.txt (work in progress).

[3] M. Civanlar, G. Cash, B. Haskell, RTP payload format for bundled MPEG, RFC2343, 1998.

[4] Coding Of Audio-Visual Objects: Systems, ISO/IEC 14496-1 Final Draft International Standard, ISO/IEC JTC1/SC29/WG11 N2501, November 1998.

[5] Coding Of Audio-Visual Objects: Visual, ISO/IEC 14496-2 Final Draft International Standard, ISO/IEC JTC1/SC29/WG11 N2502, November 1998.

[6] Coding Of Audio-Visual Objects: Audio, ISO/IEC 14496-3 Final Draft International Standard, ISO/IEC JTC1/SC29/WG11 N2503, November 1998.

[7] Coding Of Audio-Visual Objects: Reference Software, ISO/IEC 14496-5 Draft Text of Final Draft International Standard, ISO/IEC JTC1/SC29/WG11 N2505, November 1998.

[8] Coding Of Audio-Visual Objects: Delivery Multimedia Integration Framework, ISO/IEC 14496-6 Final Draft International Standard, ISO/IEC JTC1/SC29/WG11 N2506, November 1998.

[9] Coding Of Audio-Visual Objects: Systems, ISO/IEC 13818-1/Proposed Draft Amendment 1, ISO/IEC JTC1/SC29/WG11 N2739, March 1999.

[10] Coding Of Audio-Visual Objects: Delivery Multimedia Integration Framework, ISO/IEC -6/Proposed Draft Amendment 1, ISO/IEC JTC1/SC29/WG11 N2720, March 1999.

[11] E. Schooler, H. Schulzrinne, M. Handley, SIP: session initiation protocol, Internet-Draft draft-ietf-mmusic-sip-11.txt (work in progress).

[12] Generic Coding Of Moving Pictures and Associated Audio: Systems, ISO/IEC 13818-1, 1994.

[13] Generic Coding Of Moving Pictures and Associated Audio: Digital Storage Media Command & Control, ISO/IEC 13818-6, 1996.

[14] Generic Coding Of Moving Pictures and Associated Audio: Systems, ISO/IEC 13818-1/Final Proposed Draft Amendment 7, ISO/IEC JTC1/SC29/WG11 N2664, March 1999.

[15] M. Handley, V. Jacobson, SAP: session announcement protocol, Internet-Draft draft-ietf-mmusic-sap-00.txt (work in progress).

[16] D. Hoffman. G. Fernando, V. Goyal, M. Civanlar, RTP payload format for MPEG1/MPEG2 video, RFC2250, 1998.

[17] V. Jacobson, M. Handley, SDP: session description protocol, RFC 2327, 1998.

[18] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, RTP: a transport protocol for real-time applications, RFC 1889, 1996.

[19] H. Schulzrinne, A. Rao, R. Lanphier, Real time streaming protocol (RTSP), RFC 2326, 1998.

Guido Franceschini received the Electronic Engineering Degree from the Politecnico of Torino in October 1989. In 1990 he joined CSELT, where he was originally involved in the design and implementation of early ATM terminals. He has then been involved in the implementation of a DAVIC compliant Video on Demand system. He is currently working at an MPEG-4 prototype, with a focus on delivery, synchronization and server aspects. Since 1993 he has been participating to the ISO/IEC Moving Picture Expert Group, to work on the MPEG-2 Systems and then on the MPEG-2 DSMCC specifications. He has been heavily involved in the specification of MPEG-4 Systems and MPEG-4 DMIF. He acted as editor for MPEG-4 DMIF Version 1, and chaired in 1999 the MPEG Delivery subgroup. From 1994 to 1996 he has been also participating to the ATM Forum, in the Service Aspects and Applications subgroup.