Efficient Media Asset Transfer in a Unified Framework Managing Broadcasting Systems

Mathurin Body, Bernard Cousin

Efficient Media Asset Transfer in a Unified Framework Managing Broadcasting Systems

Mathurin BODY, Bernard COUSIN
IRISA– Université de Rennes 1
Campus universitaire de Beaulieu
35042 RENNES Cedex - FRANCE
Bernard.Cousin@irisa.fr

Abstract

File transfer acts an increasing role in digital TV studios and especially for their interconnections. Using adequate file formats to exchange data presents several advantages: low transfer cost and efficient handling of both essence and metadata. Many new file formats have been introduced to cope with system and user requirements. Meanwhile new user requirements for file transfer have also evolved, requesting enhanced transfers capabilities, which takes full advantages of the recently defined exchange formats and architectures.

The goal of this work is to specify an optimised transfer mechanism, this mechanism is integrated in a unified framework managing digital TV content. Our transfer mechanism provides an abstraction level for different network infrastructures and protocols in a seamless way, taking advantage of each of these technologies. For example, multicast and QoS features are supported and managed, as far as they are provided by the underlying network interfaces. Moreover, striped and partial Media Asset transfers are handled independently of the network infrastructures, at the Application layer.

1 Introduction

The need for an integrated system in the broadcasting environment is becoming essential to cope with the end user requirements. Currently, the heterogeneity of equipment, applications and interfaces requires complex and costly system integration, maintenance and upgrade. Moreover, the growing volume of content, programme material and metadata has led to the introduction of Media Asset Management systems that is fully operational in an end-to-end and integrated architecture.

In this context, most of the emerging solutions rely on service-based architectures. They indeed ensure, through the modularity of theirs components, seamless interoperability in heterogeneous environment, scalability from LAN to WAN as well as simple integration and maintenance.

Given this particular framework, some usual processes have to be adapted, rebuilt and also improved. The Media Asset transfer is one of them. Currently, most of the broadcasting studios are using FTP [1] as the "de facto" standard or tailored version of it, according to the specific needs of constructors or systems integrators. FTP indeed presents some limitations (restricted to TCP, partial transfers unsupported, no QoS or multicast features, etc.) and it becomes especially not adapted in a service-based architecture. Interconnection capacity of the entities, localisation and identification of the repositories, filenames and most of the elements usually manipulated within FTP are hidden in such integrated framework. Another benefit of our approach is the possibility to abstract the transfer process and therefore to support several network interfaces and protocols simultaneously. The most suited transmission mechanism can thus be deduced for each transfer request, depending on its particular context (QoS constraint, multicast configuration, etc.). This abstraction also improves the upgradability of the system, since new interfaces or protocols would be simply integrated, by developing appropriated adapters.

For all these reasons, it is particularly advantageous to redesign and improve transfer process for the specific context of an integrated framework.

This work is based on the ASSET project [2], targeted at the definition and the development of software architecture and the corresponding technologies necessary for a unified management of digital TV content.

Recently, the research community has brought forward the problem of multimedia transfer in broadcasting systems. The renewal of interest for this topic has been motivated by the standardisation of new formats for the exchange of programme material and the evolution of software architecture in the broadcasting studios. MPEG-21 is one of these efforts [10].

The specification of FTP+ [3] by the EBU/SMPTE task force was another more focused step in this field. After identifying the existing protocols and their limitations, they added new commands to enhance FTP capabilities. They first introduce a lot of concepts that we have adopted in our architecture. The support for different network and transport protocols is one of the most important

requirements, renewing the traditional TCP/IP basis of FTP.

Nonetheless, if FTP+ overcomes the current limitations of the file exchange and also the future ones by allowing the integration of new protocol profiles, it is adapted to a service-based architecture. It is indeed based on the FTP mechanisms and software infrastructure, which is not suited for a middleware approach. Furthermore, the integrated architecture that we consider relies on a Media Asset Manager (MAM) that induces some constraints on the transfer process (e.g. the asset naming).

FTP+ sets up strong basis for new approaches in the content transfer process. Our work can be seen as an adaptation or an enhancement of it for a service-based architecture.

Some works are still under progress in the main standardisation bodies of this field. The SMPTE organisation is directing a committee on "File Management and Networking Technology" which is working especially on the definition of transfer protocols, structures for storage, transmission and physical networks to carry them. Some related works are also performed in the EBU committee NMC (Network Technology Management Committee). Within this committee, the N/FT-AVC project indeed investigates the method of transmission on networks of files conforming to new formats (mainly MXF [4, 5], AAF [6] and GXF [7]), the file transfer protocols on different infrastructures and also distributed storage.

The remaining of this paper is organised as follows. In section 2, we present the overall ASSET architecture and its main components. Section 3 is dedicated to the Media Asset Transfer architecture, components and relationships. Section 4 discusses the solutions addressed by this proposal and gives some implementation aspects. Finally, section 5 draws conclusions based on our experimentations.

2 Service-based ASSET architecture

This section presents the global architecture of the unified framework managing heterogeneous broadcasting systems which has been defined in the ASSET project. The Media Asset Management of this framework is then presented in more details, since it is closely linked with the Media Asset Transfer process introduced in this paper.

2.1 The Asset Architecture

The following diagram (Fig. 1) presents the overall ASSET architecture and its main components. The ASSET Architecture defines numbers of concepts, components and functions that enable the implementation of an ASSET Compliant Framework. The core of the ASSET Framework consists of three main components:

- The *ASSET Public Services* expose the mandatory services of the ASSET framework to the Application & Business Logic Layer through the ASSET Public API. These services provide a minimum set of multimedia functionalities, sufficiently rich and extensible to not limit the system efficiency. They ensure the consistency and integrity of the system by handling the internal logic (e.g. access right, resource allocation, etc.) required for each operation.

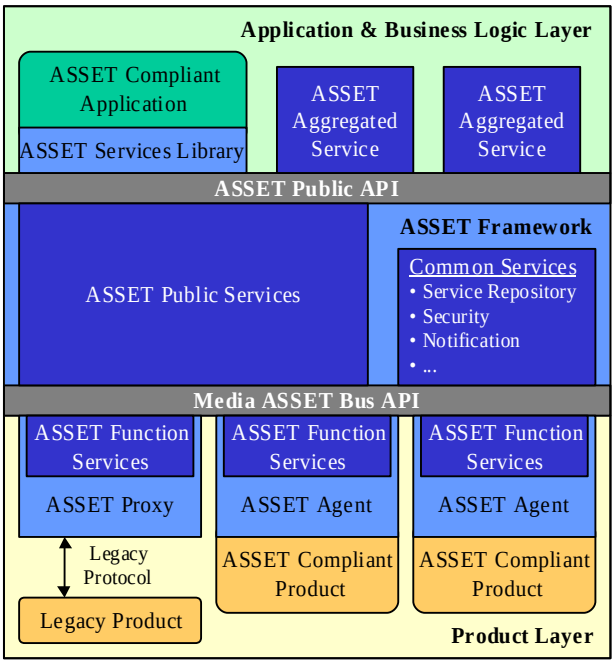

Fig. 1 – The ASSET Architecture

- The *ASSET Common Services* provide implementation of key infrastructure requirements such as security, logging, notification, resource management, etc. This allows a uniform and single implementation of these services throughout the solution.
- The *ASSET Function Services* provide an abstraction of functionalities (encoder, recorder, player, etc.) to the ASSET Public services. They hide the specificities of the different interconnected products (e.g. a VTR output and a Video Server output are considered as two system-wide logical output ports). These services are generally implemented on the products. Sometime, they can be considered as adapters of legacy products.

At the Application and Business Logic layer, three components are introduced:

- The *ASSET Compliant Applications* are the top level ASSET software components. They use the ASSET Public API to access services provided by the ASSET Framework and, optionally, by ASSET aggregated services.
- The *ASSET Services Library* is a software component included in (or linked with) an application, which makes it compliant to the ASSET framework and gives it access to the ASSET services.
- The *ASSET Aggregated Service*s implement additional business logic on top of Public services (or even on top of other aggregated services). They register in the framework as new services available for ASSET compliant applications (e.g. complex automated workflows may be specified as aggregated services

and then, be available for other connected applications or aggregated services).

At the Product layer, we call *product* a manageable hardware or software component that implements one or several ASSET Function services. Two ways exist to make a product compatible with the ASSET framework:

- An *ASSET Compliant Product* is a product that is managed by the framework through a built-in *ASSET Agent*.
- A *Legacy Product* is a product that has not (or cannot have) a built-in ASSET agent. The ASSET framework can nonetheless managed such a product through an external software module called an *ASSET Proxy* (e.g. a VTR cannot include a built-in ASSET agent and has to be connected through an ASSET Proxy).

Products, Public and Common services communicate within the framework using the API provided by the Media ASSET Bus, which is the backbone infrastructure for message interchanging and interconnection of ASSET components.

2.2 Media Asset Management

2.2.1 What is a Media Asset?

In the ASSET framework, we distinguished several entities: the *Media Asset* or *Material instance* is basically a file that contains some media or essence and optionally metadata embedded. It is identified by a UMID. A *Material* is the first level of abstraction. It corresponds to a class of file. Materials can have several Material instances. A *Material Group* gathers several materials into a coherent unit (for example linking sound tracks materials with their corresponding video). A material group is actually a way of representing a particular *content*, which is the higher level of abstraction of a Media asset. The content is completely format independent. It only describes the content itself (title, actors, commentaries, etc.). The different entities considered inside the ASSET framework can finally be summarised by the following diagram (Fig. 2):

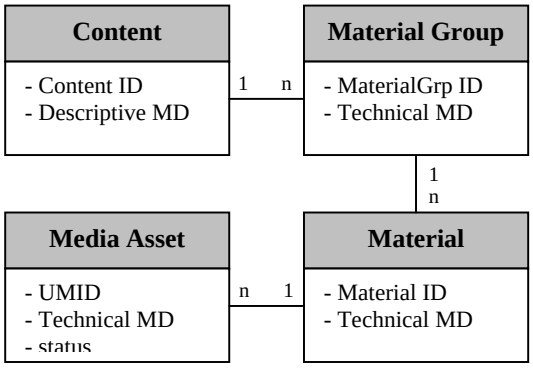

Fig. 2 – Media Asset Structure in the ASSET Framework

2.2.2 Repositories

In the ASSET framework, every product presenting the capability of storing or archiving data exposes a *repository* functionality (repository Function service). The different types of repository (on-line, near-line, off-line) are then abstracted from the user point of view. They are actually only distinguishable by their capabilities (storage capacity, time access, etc.).

A unique identifier (UUID) identifies each repository in the framework. They handle Media Asset deletion, update, notification, etc. They also manage the internal representation of the Media Assets in the local file system: storage techniques and filenames are completely hidden for the system.

The Media Asset Transfer Function service that is introduced in this paper is strongly linked with the repository Function service. Both Function services have to be developed in a common software package in order to ensure their full interoperability.

2.2.3 Media Asset Management Common Service

The Media Asset Management Common service is a central service in the ASSET framework. It handles all the information related to the media assets and the entities previously presented. The interface of this service has been specified according to the ASSET framework requirements but several types of Media Asset Manager (MAM) can be implemented behind this interface (e.g. through a proxy).

This Common service mainly maintains the listing of the Media Assets, their status and their associated links (e.g. same materials, descriptive metadata associated, etc.).

3 Media Asset Transfer

This section first presents the overall transfer mechanism and the main components that we introduce in the Media Asset Transfer process.

3.1 Architecture overview

The two main components in this Media Asset Transfer (MAT) architecture are:

A- The **MAT Function Services**, provided by the products presenting repository functionality. They handle endpoints to establish data channels and may optionally support several network interfaces (TCP, FC, XTP [8], etc.).

B- The **MAT Public service**, integrated to the framework. It ensures three principal tasks: exposing the transfer capabilities to the Application & Business Logic layer, ensuring consistency of each transfer requests and controlling the entities involved in the transfer.

Figure 3 represents the overall Media Asset Transfer Architecture. The relationships between the core components are highlighted through a typical workflow of a transfer request.

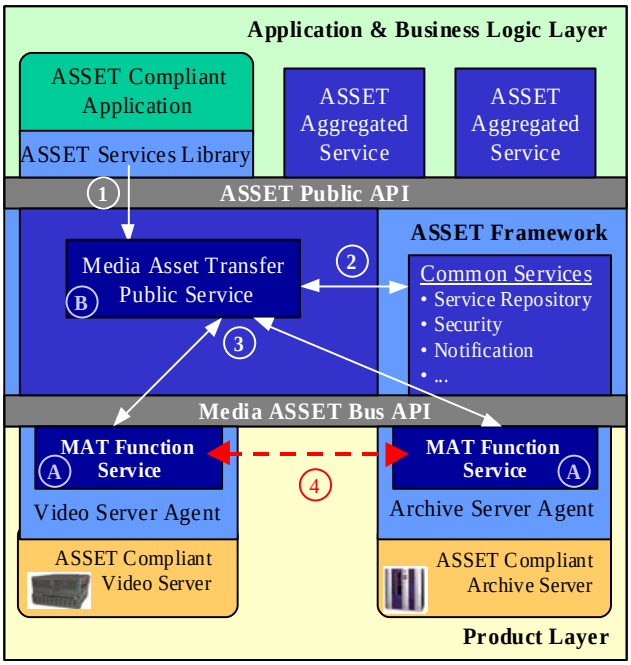

Fig. 3 – The Media Asset Transfer Architecture

The arrows on the diagram represent a typical workflow in this architecture:

1- The application emits a transfer order through the Public API. The consumer of the MAT Public Service could actually be any component of the Application & Business logic layer: either an application (e.g. a journalist wishing to archive an asset) or an aggregated service, handling a business workflow for example.

2- In order to ensure consistency and security (resolving rights, session, job and abstraction issues) messages are exchanged with the Common services of the framework.

3- Once the consistency and optional abstractions resolved, the Media Asset Transfer Public service enters in the phase of establishing and controlling the data channel between the repositories. During this phase, which contains several steps, the MAT Public service uses the commands provided by the MAT Function service of each repository involved in the transfer. This exchanging of messages is quite similar to the control channel in the case of a server-to-server transfer in the FTP.

4- The data channel is established and the media asset is transferred on this channel. Depending on the interconnected repositories and the transfer order requested by the initiator, different types of data channel can be established (multicast, QoS, security, etc.).

3.2 Media Asset Transfer Public Service

The following diagram (Fig. 4) presents the internal structure of the Media Asset Transfer (MAT) Public service.

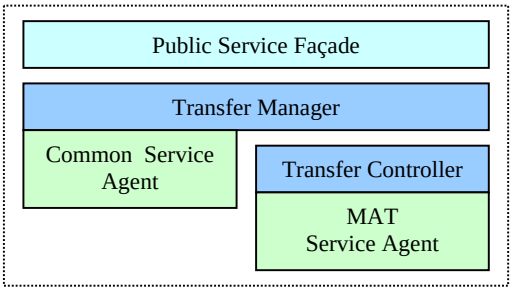

Fig. 4 – The Media Asset Transfer Public service

The MAT Public service is made of three main components described in the next sections.

3.2.1 Public Service Façade

First, the MAT Public service presents and provides services through the *Public Service Façade*, which exposes the public interfaces. It is in charge of receiving, responding and eventually queuing messages from consumer entities.

3.2.2 Transfer Manager

When the Façade receives a transfer order, it simply forwards the request to the *Transfer Manager*. This component takes care of the "high-level" tasks, preparing and ensuring the consistency of transfer requests. For example, rights of the request initiator have to be checked or asset may have to be located or even identified since abstraction of material is providing in this framework. Repositories can be located and also selected according their capabilities (type of storage, capacity, access time, etc.). Transfer order can eventually be scheduled via the dedicated Common service. Since new instances of existing asset (or even new assets in case of partial transfer) are created during the transfer, new UMIDs identifying them have to be generated. This is also one of the tasks of this Business Component. All these tasks are mainly performed by Common services, which are accessed through Service Agent.

3.2.3 Transfer controller

Once the Transfer Manager has performed these "high-level" tasks, it delegates the actual processing of the transfer to the *Transfer controller*. The business component responsible for the transfer control ensures the establishment of the data channel between the repositories. It also supervises the progression and ending of the transfer. This business component interacts with the MAT Function services provided by the repositories through service agents.

The establishment of a data channel consists of five subtasks:

a. Negotiate the protocols, mode of transfer and QoS.

b. Create and put the passive endpoint(s) in listening state.
c. Create the active endpoint(s).
d. Notify the passive endpoint(s) that the connection has been established.
e. Initiate the transfer on each endpoint.

3.3 Media Asset Transfer Function services

The following diagram (Fig. 5) presents the internal structure of a Media Asset Transfer (MAT) Function service.

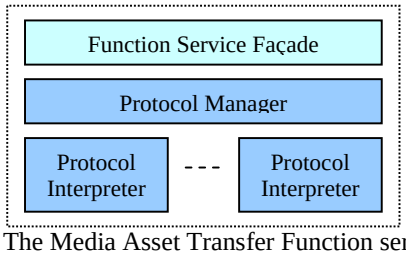

Fig. 5 – The Media Asset Transfer Function service

This Function service presents three types of components, described in the next sections.

3.3.1 Function Service Façade

Similarly to the MAT Public service, the MAT Function service presents its functions through the *Function Service Façade*. It is in charge of the interactions with its service consumer (the MAT Public service).

3.3.2 Protocol Manager

The *Protocol Manager* is the business component responsible for controlling the different network interfaces or profiles supported by the repository, and consequently by this service. It knows its capabilities and interacts with the MAT Public service during the protocol negotiation phases. Once the protocol has been chosen (by the MAT Public service), the protocol manager instantiates a business component of the corresponding *Protocol Interpreter* and forwards every messages related to the creation and management of the data connection to it.

3.3.3 Protocol Interpreter

Each *Protocol Interpreter* presents a common interface (abstracting the establishment of a data connection) but implements each abstracted function for one particular protocol or network interface. In that way, MAT Function services are able to handle several network profiles and to choose the most suited one for each transfer.

4 The Media Asset Transfer solution

In this section, we first introduce the interface of the Media Asset Transfer Public service through the definition of Transfer orders. Based on this definition, we present the enhancement provided by our approach.

4.1 The Transfer Order scheme

The MAT Public service receives Transfer Order (or Groups of Transfer Order) in XML [9] messages respecting the following scheme:

```
<?XML version="1.0"?>
<!ELEMENT TransferOrder
            (TransferType?, StartTime?,
             QoSFeature*,Source?,Asset,Destination)>

<!ELEMENT TransferType ("COPY"|"MOVE"|"STRIPED"...)>
<!ELEMENT StartTime                          (#PCDATA)>
<!ELEMENT QoSFeature  (QoSType,(QoSValue|QoSRange))>
<!ELEMENT QoSType              ("RATE"|"BURST"|...)>
<!ELEMENT QoSRange         (QoSValue, To, QoSValue)>
<!ELEMENT QoSValue                           (#PCDATA)>
<!ELEMENT Source     (RepositoryType|RepositoryID+)>
<!ELEMENT RepositoryType
              ("NEARLINE"|"ONLINE"|"OFFLINE"|...)>
<!ELEMENT RepositoryID                       (#PCDATA)>
<!ELEMENT Asset
             (AssetID, MaterialType?,
              Partition*,ByteOffset?, ByteLength?)>
<!ELEMENT AssetID                  (UMID|MaterialID)>
<!ELEMENT UMID                               (#PCDATA)>
<!ELEMENT MaterialID                         (#PCDATA)>
<!ELEMENT MaterialType     ("DATA"|"VIDEO"|"AUDIO")>
<!ELEMENT Partition
   (PartitionType,(PartitionValue|PartitionRange)+)>
<!ELEMENT PartitionType
     ("CLIP"|"FRAME"|"SCENE"|"SEQUENCE"|"TIME"|...)>
<!ELEMENT PartitionValue                     (#PCDATA)>
<!ELEMENT PartitionRange
           (PartitionValue, To, PartitionValue)>
<!ELEMENT To                                  (empty)>
<!ELEMENT ByteOffset                         (#PCDATA)>
<!ELEMENT ByteLength                         (#PCDATA)>
<!ELEMENT Destination
                  (RepositoryType|RepositoryID+)>
```

This XML scheme illustrates how we implement some of the Media Asset Transfer issues in our approach. We will not exhaustively explain all the elements introduced in this scheme. We indeed prefer to focus on the key issues that limit traditional FTP and that are addressed in our proposal.

4.2 Issues addressed by the MAT architecture

4.2.1 Adaptations to the integrated framework

The XML scheme demonstrates the abstractions that are made possible in such an approach. In a unified and integrated framework, one of the most important advantages is indeed the virtualisation of all interconnected components, storage locations and content materials.

For example, the initiator of a Transfer Order may only specify a material (and not specific files any longer) through the *MaterialID* element and/or a type of repository. The *RepositoryType* element set to "OFFLINE" triggers the transfer of the material to an Archive server, automatically chosen by the framework.

In the same way, source repository will be involved most of the time, since storage location is transparent for the consumer's point of view. The MAT Public service resolves all these abstractions by interacting with Common Services, which are responsible for the system integrity.

4.2.2 Support for different network interfaces

Several types of network interfaces, transport and network protocols are currently available on the market, each of them presenting some specificities, some advantages but also some drawbacks. Actually, none of them can be considered as the "one-fit-all" technology. In our proposal, we abstract the underlying transport protocols and network interfaces, it allows to select the best of each

world: TCP is probably the most suited protocol for exchanging data ("unicastly") over the Internet, but does neither allow any multicast transfer nor guarantee any QoS.

Moreover, besides supporting any protocol or network interface, the proposed architecture provides adaptable heuristics to select among the available transfer mechanisms, the one that will be the most suited one, according to the context of a transfer request.

For example, in a Transfer Order, an initiator may request a certain type of transfer and/or QoS features that will entail one or more transport mechanisms. If several mechanisms fit to this Transfer Order context, the MAT Public service chooses the most suited one, with respect to the rules defined by heuristics.

4.2.3 Partial transfer

The issue of transferring parts of files must be distinguished from partial Asset transfer. The first one is only of use for recovery procedures after failure on the data channel. In this case, parts of file are described with offset and a number of bytes. On the other hand, the partial Asset transfer is related to the transfer of a specific content or portion of a Media asset. Assets may indeed contain several different materials: metadata, one or more essence(s) organised in several segments, etc. It may also append that the user only needs one of these parts (e.g. ten minutes from a video of several hours). It has become a key issue, especially with the growing size of the media asset in the broadcasting environment.

Both issues are address in our scheme. In an *Asset* element, *partitions* can indeed be specified to select specific portions of a Media Asset (Sequence, Scene, Frame, etc.). Parts of file can also be explicitly requested by the *ByteOffset* and *ByteLength* elements. Parts or portions of assets are only extracted at the source repositories by the MAT Function services before (or during) the transfer.

4.2.4 Multicast transfer

The multicast transfers allow the transmission of assets from one repository to many without replications of the data, shortening delays and freeing bandwidth on the network. These types of transfer are still underused in the broadcasting studio, probably because its benefits are still less important than its costs of implementation. Nonetheless, the exponential growing of data transfers in a studio will make the multicast transfer much more advantageous than they are for the moment.

Our architecture fully supports multicast protocols (such as XTP or any reliable multicast protocols based on UDP) and therefore does not restrict its scope to unicast transfers. In the Transfer Order, several destination repositories can be specified. In this case, all the repositories supporting a common reliable multicast protocol are requested to establish a multicast transfer. For others, unicast or a different type of multicast protocol can be chosen by the MAT Public service.

4.2.5 Striped transfer

Recently, a new mechanism for optimising file transfer has been proposed: the striped transfer. First, it supposes that the file to transmit is replicated among several repositories and that the bottleneck in the transfer process is not the sink endpoint. Actually, the striped data transfer optimises the transmission by parallelising it among several sources. Several data connections are established, making a many-to-one channel. Continuous parts of the file (forming a partition) can then be simultaneously transmitted on these connections.

The Media Asset Transfer architecture takes this transfer mode into account and handles it at the application level. Therefore, it allows the combination of this transfer mode with other features already supported. For example, different types of data connections can be established, since they are independent. We also support striped multicast transfer: N multicast connections can be established from N sources to M destinations, forming a N-to-M data transfer.

A striped transfer can be requested by specifying either several source repositories or a "STRIPED" *TransferType* element. In both cases, only a *MaterialID* (and not a *UMID*) can be specified. Several instances of the same material are indeed used in a striped transfer. A destination repository in a striped transfer should be able to receive simultaneously the different parts of the file and to restore it at the end of the transfer.

4.2.6 Quality of Service

The so-called Quality of Service (QoS) denotes the implementation of policies to offer better and guaranteed service in the transport mechanisms and the underling network interfaces. Using traffic and performance management parameters (e.g. priorities, resources allocation, dedicated bandwidth, controlled jitter and latency, etc.), QoS technologies enable the user (or automated system) to qualitatively and quantitatively express and measure the service being provided.

QoS is highly dependent of the underlying network infrastructures and transport mechanisms. Since we choose to abstract these concepts in our architecture, QoS parameters could only be represented in flexible structures. The QoSFeature element of the Transfer Order addresses this need. This XML scheme indeed allows the description of QoS parameters, either by key/value or key/range. For example, users wishing to restrain the transfer rate may specify a maximum rate value and a maximum burst value. Our QoS implementation consists of three steps:

- QoS features supported are identified and retrieved to the users requesting them.
- User-defined QoS parameters are then co-ordinated between the different network components.
- QoS policy is followed up during the transfer and controlled to guarantee the requested service.

4.3 Security in the MAT architecture

Security has become a critical issue with the digitalisation of the broadcasting systems and their

extension on wider area network. Security requirements are indeed much more important in a WAN (wide area network) than in a LAN (local area network).

In case of a LAN, the main security needs focus on data flowing protection, against ill intentioned acts or more usually against mistakes. For WAN interconnections, security must considerably be increased. Exchanging data through Internet may be cost effective but important security issues have to be addressed in consequence (protection against intruders, data integrity, confidentiality, etc.).

We do not specifically address the security issue in our proposal. We indeed prefer to rely on both the underlying network connections (private network, VPN, etc.) and the framework security (access rights, account checking, secured interconnections in the framework).

5 Conclusion

This paper gives an overview of a Media Asset Transfer service integrated in a unified framework managing broadcasting systems. The leading advantage of this solution is the abstraction of the underlying network infrastructure, which allows the support of complementary transport interfaces. It takes also advantage of the integrated framework, providing services for scheduling transfer, managing Media Assets and repositories. Finally, it provides new functionalities to cope with the user requirements in the broadcasting environment (e.g. partial asset transfer).

Our architectural solution has been implemented using simple heuristic. For instance, XTP is used when several destinations have been selected in the Transfer Order and XTP is available on all transfer endpoints. Similary partial transfer requires the availability in the framework of a cutting agent adapted to the format of the material to be exchanged. Nevertheless, based on the parameters of the Transfer Order (parameters which are automatically deduced from the metadata of the asset to be exchanged) the above enhanced services have demonstrated their accuracy and great efficiency. For instance, stripped and partial transfer of large MPEG videos has demonstrated a decreasing in the overall transfer delay up to 90 %.

6 References

[1] J. Postel, J. Reynold, File Transfer Protocol (FTP), IETF RFC 959, October 1985.

[2] ASSET project web site – http://www.ist-asset.com

[3] SMPTE/EBU, "Joint EBU / SMPTE Task Force on User Requirements for the Exchange of Television Programme Material as Bit Streams", http://www.smpte.org

[4] Pro-MPEG Forum, "Material eXchange Format (MXF)", http://www.pro-mpeg.org/mxf.htm, October 16, 2002.

[5] Bruce Devlin, "MXF – The Material eXchange Format", EBU Technical Review, July 2002.

[6] Brad Gilmer, "AAF – the Advanced Authoring Format", EBU Technical Review No. 291, July 2002.

[7] Bob Edge, "GXF – the General eXchange Format", EBU Technical Review No. 291, July 2002.

[8] XTP Forum, "Xpress Transport Protocol", ftp://dancer.ca.sandia.gov

[9] Extensible Markup Language (XML) 1.0 (Second Edition. W3C Recommendation. World Wide Web Consortium, http://www.w3c.org/TR/REC-xml, October 6, 2000.

[10] Multimedia Framework (MPEG 21), ISO/IEC 21000-1 to 3, March 2003.

[11] P. van Beek et al., "Metadata Driven Multimedia Access", IEEE Signal Processing, vol 20, no 2, March 2003.

Acknowledgement

This work has been partially supported by the European Commission through the contract (IST-2001-37379 Architectural Solutions for Services Enhancing digital Television).