FP7-ICT-2007-3-231161

Deliverable D2.3.1

Service-Oriented Models for Audiovisual Content Storage

Stephen C Phillips (University of Southampton IT Innovation Centre)

2010-04-14

Document administrative table

Document Identifier	PP_WP2_D2.3.1_SOAforAV_R1 Release 1
Filename	PP_WP2_D2.3.1_SOAforAV_R1_v1.01.doc
Workpackage and Task(s)	WP2: Models and Environments for Long-Term Audiovisual Content Preservation WP2T3: Storage Strategies and Rule Sets for Preservation
Authors (company)	Stephen C Phillips (ITInnov)
Contributors (company)	Richard Wright (BBC), Walter Allasia, Elisa Todarello, Francesco Gallo (Eurix), Jean-Hugues Chenot (INA), Matthew Addis, Fu Cheng Chen, Mark McArdle, Ken Meacham, Nena Schädler (ITInnov), Christoph Bauer (ORF), Roberto Borgotallo (RAI), Günter Mühlberger (UIBK)
Internal Reviewers (company)	Werner Bailer (JRS), Laurent Boch (RAI), Ann Gledson, Adil Hasan (UniLiv)
Date	2010-04-14
Status	Final
Type	Deliverable
Deliverable Nature	R
Dissemination Level	PU
Planned Deliv. Date	2009-12-31
Actual Deliv. Date	2010-02-12

Abstract

What are the important topics to understand if involved with storage services to hold digital audiovisual content? This report takes a look at how content is created and moves into and out of storage; the storage service value networks and architectures found now and expected in the future; what sort of data transfer is expected to and from an audiovisual archive; what transfer protocols to use; and a summary of security and interface issues.

DOCUMENT HISTORY

Version	Date	Reason of change	Status	Distribution
0.01	2009-06-02	First Draft	Outline	Confidential
0.02	2009-07-22	Structure changes following WP2/3 meeting	Outline	Confidential
0.03	2009-11-27	Update to chapters 2, 3, 6 7 and 9	Draft	Confidential
0.04	2009-12-18	Major content update	Draft	Confidential
0.05	2009-12-21	Added security section	Draft	Confidential
0.06	2009-12-24	Major update	Final draft	Confidential
0.07	2010-01-05	Corrections following review	Final draft	Confidential
0.08	2010-01-08	Corrections following review	Final draft	Confidential
0.09	2010-02-10	Corrections following review	Final	Confidential
1.00	2010-02-12	Finalised and delivered	Delivered	Public
1.01	2010-04-14	Minor corrections	Delivered	Public

Table of contents

Scope

The European Commission supported PrestoPRIME project is researching and developing practical solutions for the long-term preservation of digital media objects, programmes and collections, and finding ways to increase access by integrating the media archives with European on-line digital libraries in a digital preservation framework. This result will be a range of tools and services, delivered through a networked Competence Centre.

To preserve digital audiovisual data, large storage systems are required. This report looks into a variety of issues important when considering using service-oriented storage solutions. The report covers, by way of introduction:

- What is the difference between “storage” and “preservation”?
- How are the lifecycles of data, services and service level agreements intertwined? How and where is content produced? What is archived, is it ever discarded and when?

To understand the current positioning of storage services:

- What storage services and software exist today?
- What are the relevant value networks for storage systems and services?
- What storage architectures are found today and expected in the future?

On a more technical level, to get data to and from a storage service and protect it both in transit and whilst it is there we must know:

- What types and quantities of data could be expected to flow in and out of a storage service?
- What are the different data transfer protocols that can be used to move data across a network and which is fastest?
- What security issues are there with storing data in a service and how can they be addressed?
- What would a software interface for a storage service look like?

In combination with the related PrestoPRIME documents ID3.2.1[1] on threats to mass storage and ID3.4.1[2] on service level agreements this completes a round up of information on all the topics relevant to choosing and designing a storage service.

Executive summary

The European Commission supported PrestoPRIME project is researching and developing practical solutions for the long-term preservation of digital media objects, programmes and collections, and finding ways to increase access by integrating the media archives with European on-line digital libraries in a digital preservation framework. This result will be a range of tools and services, delivered through a networked Competence Centre.

This document provides a wealth of information for people who want to understand more about how online storage services may be used to host digital audiovisual content. The document covers the subject at three levels:

1. What is the difference between storage and preservation and how are the lifecycles of the data and the services related?
2. What is the state of the art in storage services today, what are the value networks and how are existing systems architected?
3. What are the characteristics of the data that can be expected to enter and leave a storage service, what are the options for transporting it on a network and what security and interface issues should be considered?

It is recommended that this document be read in conjunction with the related PrestoPRIME documents ID3.2.1[3] on threats to mass storage and ID3.4.1[2] on service level agreements for preservation services to provide a broad round up of information on all the topics relevant to choosing and designing a storage service.

This document is predominantly about storage services but PrestoPRIME is about preservation. “Storage” is about keeping a defined sequence of bytes whereas “preservation” is about being able to understand the information held in those bytes at a later stage. A successful preservation system requires an advanced, robust storage system underneath it but cannot treat the storage as a black box that will maintain data permanently. It must include management systems (directly or through service level agreements) to monitor the storage and be proactive in maintaining the data.

Looking at the lifecycle of digital audiovisual data we draw on the experience of a national broadcaster. It is clear that we are currently in a time of great change with policies developed for archiving, reviewing and even discarding content developed in an age of physical formats, being updated and rethought for the digital world where production systems and the archive are ever more closely linked. For instance, it is becoming technically feasible to have a “keep everything” approach to archiving including programme rushes and regional output all kept centrally whereas previously this material may have been discarded. If an archive cannot keep everything however, then the problem of when to review content arises. Previously, content was reviewed when its physical format was at risk but this changes with digital systems and as a consequence new policies will be needed.

The most obvious policy requirement arising from the study into the lifecycles of content, rights-holding organisations, service providers and contracts is that, given the aim of preserving the content for ever, any contract with a storage service provider must also include an exit strategy to ensure the data can be moved to another provider.

To determine the requirements for a storage service, it pays to know what systems already exist. The review of the state of the art finds many interesting technologies, looking across the board from related projects such as CASPAR and SHAMEN to commercial online

systems such as Amazon S3 and Nirvanix. Several interesting features can be found in the systems reviewed such as the iRODS rule engine, erasure coding used instead of RAID by several services and federation across multiple data stores: all worthy of further consideration in the specification of a storage service.

Much can also be learned from looking at how existing archives across Europe have architected their systems. Varying degrees of outsourcing can be found with the most recently specified system using multiple sites, multiple mirrors and links and a single service provider. With fast dedicated network connections now commonly available and the increasing maturity of the market for large storage we can see that outsourcing the storage of online data will become more common and that organisations will move towards using multiple providers for increased security. Each archive must recognize that it is not working alone and that value can be added to its content by cooperation and federation with other similar archives in a variety of ways, as described in OAIS.

Finally this document delves into some technical detail, considering firstly the data entering and leaving a storage service. Data entering an archive's storage system will typically be anything from production quality data at 50 Mb/s to master quality at around 200 Mb/s for standard definition with high definition coming in at 100 Mb/s to almost 1 Gb/s, giving file sizes of up to 500 GB per hour of video. The data leaving the storage will normally not be the master quality. Instead it is browse quality of maybe 0.5 Mb/s (streaming) or the production quality material. The number of files being ingested and accessed is only going to increase with the tighter integration of archives with production and with the opening up of archives to commerce and the public. To give one example, in October 2009, the BBC iPlayer dealt with 79.3 million requests for TV and radio programmes, transferring 7 PB of data and peaking at 12.5 GB/s. In contrast, the BBC's archive delivers approximately 10000 items per week to programme makers: a much smaller number but still significant.

To get all this data in and out of a storage service via a network connection efficiently requires an understanding of the network file transfer protocols and the underlying internet protocol technologies. A summary of approximately thirty protocols is presented along with a comparison of their features. Although the venerable FTP protocol is still widely used, its foundation on TCP causes it to use today's long high-bandwidth networks inefficiently. Several commercial offerings claim much superior performance by using proprietary protocols based on UDP instead. GRIDFTP is an open source alternative that can use UDP and its performance is tested against FTP in a variety of simulated network conditions. It is found that for the transfer of a single file at one time, GRIDFTP's performance is much better than FTP for networks with latency over 20ms.

Many documents try to treat security as somebody else's problem but we argue that security must be considered from the start when commissioning or developing a storage service for audiovisual data. The interests of rights-holders must be protected and access to the data must be controlled both in the storage service and in the transmission channel. In an archive system where the aim is to preserve data, the most obvious control is that of making sure very few people have the right to delete anything.

Service security is part of the risk management process. Proportional security measures should be used to mitigate perceived threats, both malicious and accidental, to a system's assets. This document provides up to date information on methods of user and service identification, secure connections and access control systems, policies and standards. The document is then completed by an overview of what the interface to a storage and preservation service would look like, following OAIS recommendations.

1. Introduction

The focus of this deliverable is service-oriented storage in all forms including federated storage. This means storage as (a) in-house services developed by an organisation's IT department, (b) managed services provided by a third-party but still using equipment installed locally at an organisation, (c) services delivered by an organisation using the facilities of a data centre as an off-site hosting environment, (d) storage services provided in their entirety by a third-party, e.g. in the way that Amazon provide S3, or (e) a combination of any or all of these, e.g. federating storage between multiple archives, or combining local storage and remote outsourced storage to meet the needs of access and safety for a community of content producers and consumers.

It's worth defining at this point what we mean by 'service oriented' and 'federated'. OASIS uses the following definition: a service is a mechanism to enable access to one or more capabilities, where the access is provided using a prescribed interface and is exercised consistent with constraints and policies as specified by the service description[4]. A service description includes: what the service does (interface); how it should be used (protocol); how delivery can be measured (metrics); and how the service is governable by a Service Level Agreement (SLA). Federated storage describes multiple storage services coordinated as one large system, but where each service remains under control of a separate host.

For PrestoPRIME, the important thing is that the service 'interface' separates how the service is implemented (details of the ever changing IT used in storage systems) from how the service is used in preservation systems (ingest and access to content, preservation actions performed on content etc.). This allows one to change without affecting the other, which can be a very powerful approach. However, the question is then: how can an archive be maintained, and preserved for the future, when its physical manifestation, its digital storage, is no longer under the direct control of the archive? Most approaches make an implicit assumption that a repository is 'one thing': unified management, including unified management of storage.

A federation of cooperating parties needs a set of rules so that collectively they can also form a 'trusted repository' that can meet requirements such as those specified by OAIS and TRAC and related documents. These rules include how such a federation operates from day to day. There must then also be an overall strategy to control the evolution of the federation, to ensure content is preserved. At the top-level, we need a formal framework of operating rules for the use of storage as a service so that a federated approach can also meet trusted digital repository standards.

The concept of SLAs is the key to this, especially where services are provided/used across administrative domains (either between business units or across organisations). An SLA describes the function performed by a service, obligations on the provider and consumer of the service, the agreed bounds of performance (QoS) for the service, and how deviations are handled (exceptions). Without SLAs that can be automated and enforced, online storage services are unmanageable, unpredictable, not guaranteed and as a result their use undermines the whole concept of trusted repositories.

This document first describes the key differences between preservation services and the underlying storage services. It then looks at content lifecycles and the relationships between the lifecycles of content and those of the organisations and contracts involved in

storing the content. The state of the art in storage services is then explored followed by a discussion of the variety of value networks and architectures for storing content found today and expected in the future. What will be stored and how is then investigated in chapters on the data, the network transfer protocols and a preliminary investigation into the speed of different transfer methods. Finally, aspects of security for storage services are discussed followed by a preliminary look at the interface requirements. The related concepts of trusted repositories and SLAs are explored separately in PrestoPRIME ID3.4.1[2]

2. Preservation Models versus Storage Models

PrestoPRIME research and development focus is on tackling the problem of long-term preservation of digital media objects. This deliverable focuses more on storage systems which necessarily underpin preservation systems. Before discussing aspects of storage though, we should understand the difference between storage and preservation.

2.1. What is the essential difference between storage and preservation?

Storage is really a generic denomination inside of which fall several greatly different cases, from the highly valuable information stored in the banks for each customer account to the far less critical web pages stored in a free web internet publication server. Some retained data is more important than others but all depends on the perspective of the data owner or the data beneficiary. In any case it is certain that *the reliability of the storage is one of the major concerns.*

Generally we refer to a computer file as the entity which is the object of the storage process. The computer file is an arbitrary block of data (sequence of bytes) that turn out to be information (with a meaning in a context) when considered according to the format of the file itself. The format in fact implies the type of processing that is required for getting usable data/information from the raw file: that is “The nature of data affects the access and the processing”.

Storage, as a process, includes:

- writing/recording the data,
- retaining the data for an undetermined length of time,
- reading and accessing the data.

More advanced storage may include:

- integrity checking of data through checksums and disc scrubbing,
- replication of files through techniques such as RAID, erasure coding, or the simple generation of multiple copies on different media or at different sites (geo-replication),
- optimisation of data placement, e.g. in hierarchical storage management (HSM) systems.

Preservation is a process that includes the storage, but is not limited to it.

The Preservation process provides all the means and the sub-processes necessary to guarantee the access (i.e. correct fruition) of the information contained in a file. This implies that is necessary to preserve not only the pure audiovisual data but also the complete set of properties and metadata that are indispensable for this purpose, whatever the technology changes occurred from the first ingestion of the files.

Thus, in addition to storage, the preservation process includes:

- ingest and access methods appropriate to the *designated community* (e.g. AV users),
- preservation of the environment for the tools used to access the data,
- the capture and preservation of descriptive metadata,
- the identification and preservation of appropriate *representation information* to interpret the bits held in storage as information,
- migration processes for format conversion, used for keeping at a reasonable cost the ability to access and process the content,
- management of strategies for redundancy/data integrity and recovering,
- tracking and reporting on preservation process activities.

While the storage service is liable for the persistency of data (bytes) in their integrity, a preservation service is responsible, in addition, for usability and reliability of the information contained in the data.

For these reasons actual pure storage systems must be far improved to at least support basic requirements listed above. We could assume to start from a state of the art storage system with a high trustworthiness (reached by mean of replication or other technologies) and a good level of management (the blue block in Figure 1). On top of this, there should be support for the specific requirements of multimedia: big files, access in the typical terms of the AV domain (component or tracks and time intervals or Edit Units), validation against well known formats of coding and wrapping, the possibility to provide content even in the case of partial corruption (though perhaps with reduced quality).

Finally higher level software should be provided to deal with preservation administration and strategy in order to guarantee the safeguarding of the capability to access and exploit the information contained in the files. In practise this means the ability to keep and preserve the *representation information*, the tools for playback and their environment and eventually to schedule format migrations.

2.2. What are the specific needs of AV preservation?

To answer this question we focus on the nature of the AV document.

In an IT environment an AV document is typically recorded in the form of a computer file, which we will call AV-media-file. It is important to understand the main exceptions to this statement: it is possible to organise the single AV document as a set of separate components, each of which can be saved as AV-media-file or computer file. Sometimes this set can be further grouped either as a file system (e.g. DVD) or as an archive file (e.g. tar) or as a another type AV-media-file according to a wrapper format or even as a logical-only grouping with a handle component pointing to dispersed resources.

Specifically AV data can be either uncompressed, in which case they are easily accessible for use, or compressed (lossless or lossy), in which case they need a specific decoding process before their generic use (some functionalities are sometimes possible also in the compressed domain, but this is not the general case).

The use of an AV-media-file requires the ability to correctly dewrap it (extract components) and decode it (give access to uncompressed domain).

The preservation process is responsible for the usability of the archived item. However what is important from the *consumer* (in the OAIS model) perspective is the usability of the delivered item (*DIP*). This means that the delivered format should be either uncompressed or compressed in an agreed format providing that the *consumer* is able to interpret the files. The wrapping format must be agreed as well (agreed with the *consumer* according to SLAs).

As AV-media-files are, for the master quality level, very large datasets and consumers often require only a segment of the whole document, the preservation system must be able to efficiently provide delivery items obtained through the extraction from a larger archived item.

With respect to a generic storage system, a basic AV preservation system should at least supplement the basic operations as expressed in Figure 1.

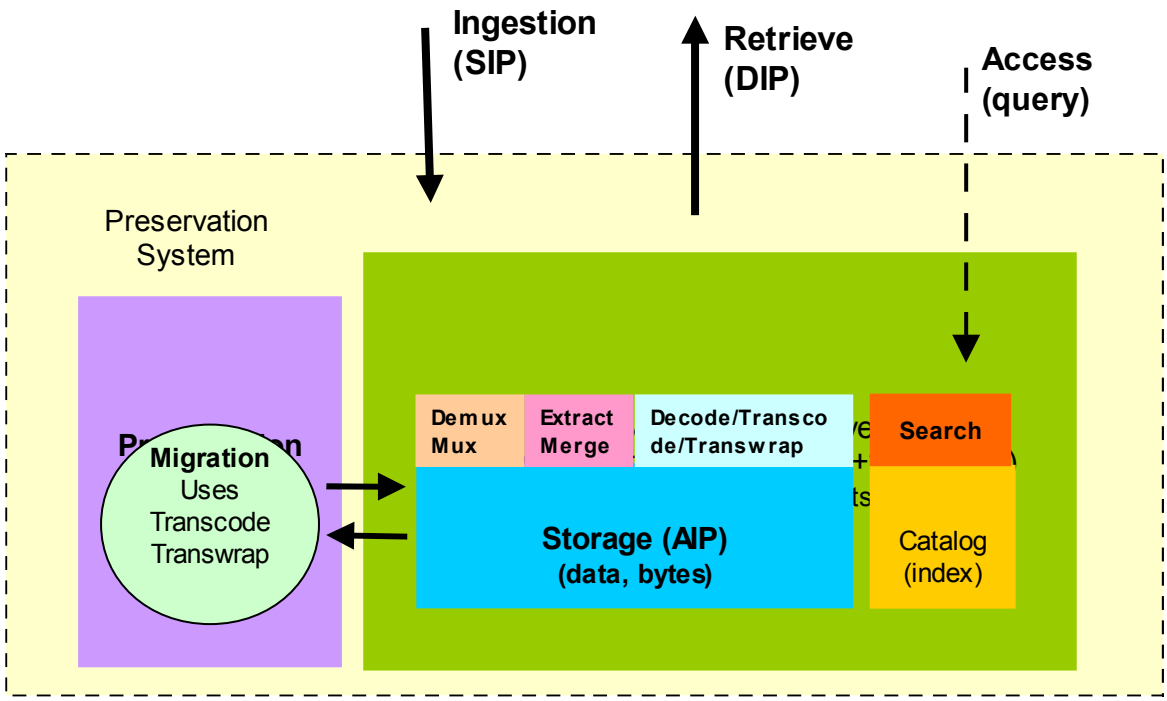


Figure 1 Example of how a storage system is embedded in a preservation system providing additional services.

INPUT (Ingest)

- Support for validation against a minimum set of widespread and standard wrappers and encoding of audio and video. Because the preservation is focused on AV, the ingestion phase should be aware of the nature of these files and reject any unknown file type.
- Support for big files. (Note that if considered convenient, they could be broken up internally by the storage system but with respect to input and output operations they are considered single items.)

OUTPUT (Retrieve)

- Support partial retrieval (the nature of multimedia is timeline based).
- Tolerate slight errors, giving the possibility to recover even at a lower quality a certain piece of multimedia. The reality of really big files and high bit rates means that read errors are not so unlikely.

There are several levels of tolerating errors: (a) correction mechanisms build into the storage/file system, (b) correction mechanisms provided by the decoder, and (c) errors that have to be dealt with on higher level (e.g. if a frame cannot be decoded any more, repeat the previous one or interpolate it from previous and next). Category (a) would need to be dealt with by the storage component of a preservation system. Categories (b) and (c) may be covered by a preservation system depending on the scope of its facilities.

Another important and basic aspect is the usability of outputs, i.e. formats coming out have to be up to date in order to permit their reproduction, editing and so on without compatibility problems with respect to the latest tools and devices. This means that an AV preservation system has to include a migration engine or rely in some way on a multivalent platform.

Beside basic requirements there is a large set of enhanced features that could be highly desirable for specific customers such as broadcasters:

User and material management features:

- Effective control over users and their rights
- Rights management of editorial objects
- Effective versioning system e.g. to track different copies and extracts
- Easy reports on who has retrieved or ingested what and when
- Reports on movement of volumes in terms of editorial objects, files, sizes (gigabytes, GB)

Performance features:

- Fast in and out operations with simultaneous accesses
- Low probability of corruption of files

Advanced annotation features:

- Ability to manually annotate editorial objects and materials e.g. adding descriptions, titles, credits, relations with other products (useful for later search and retrieve)

Advanced retrieve features:

- Ability to search for specific editorial object and materials based on previous annotations or technical characteristics.
- Ability to search by similarity (query by a sample image, movie or audio).

2.3. What must storage services deliver?

A preservation service should deliver exactly the same content that has been ingested (this should be always a requirement) or an equivalent version if the content has been migrated to different formats because of obsolescence or specific necessities. The quality should not be affected at all by the long term storage. The storage services should be tied to the generic concept of a computer file and provide access to:

- AV-media-files
- Every single track e.g. the first audio-track of an audiovisual file
- A portion of an AV-media-file. (e.g. first 10 minutes of 3 GB from a certain byte-offset)
- The set of information (metadata) about the administration of storage service on an AV-media file (size, access/modification time, integrity status, and other)

The preservation system should also be able to deliver derived information, like proxy versions and any kind of automatically extracted metadata or previously manually annotated information.

2.4. What management is required in a preservation system?

In general the management of a preservation system should be focused on the compliance to the SLA terms. For instance the various storage services should be managed in order to respect the agreed values on access failure and data loss.

The basic management should be able to accommodate multiple users with respective rights to ingest and extract material, management of the space with convenient alert and management mechanisms when resource limits are approaching, protection against overload in order not to interfere with basic fundamental batch processes like format migration. Eventually the preservation system management should be capable of assessing the risks related to the use of each storage service. (e.g. multiple failures could trigger storage migration).

If we think about the preservation system as closer to a content management system we have another category of management to take care of: rights of editorial objects, aggregation of editorial objects into products (e.g. a serial), relation between editorial objects and materials and between materials and materials (e.g. subparts, different quality etc.).

2.5. Conclusions

“Storage” is about keeping a defined sequence of bytes. “Preservation” is about being able to understand the information held in those bytes at a later stage.

A successful preservation system requires an advanced, robust storage system underneath it.

By understanding the information, preservation systems can cope with some data loss.

A preservation system cannot treat the storage as a black box that will maintain data permanently. It must include management systems (directly or through SLAs) to monitor the storage and be proactive in maintaining the data.

3. Lifecycles

When considering the long-term storage of audio-visual data in a preservation service it is important to understand the variety of lifecycles that must be dealt with. How and where is content produced? What is archived? When is it discarded? What about the services, contracts and organisations?

3.1. Content

Content is created. Some is kept and some discarded. Some content is kept for a long time and some for a short time. Taking just one broadcaster's experience, the life cycle of content can be summarised as follows:

First, for broadcast archiving, various distinctions should be made:

1) content completely external to the broadcaster (e.g. commercially-recorded music; cinema films),

2) local versus national programming,

3) final programmes versus rushes (material from which the final programme was made).

4) live vs. pre-recorded output

5) content produced in-house vs. content independently produced under contract to the broadcaster

For a broadcaster moving to file-based working covering the whole production chain, the majority of files in use at any one time would be rushes. The ratio of rushes to final product is the 'shoot-to-show' ratio of a production, and can be as high as 20:1 for high-end factual productions such as the BBC's "Horizon" programme, or major natural history features (e.g. "Blue Planet").

The following table summarises the life cycle of 'hold it in your hand' content – on physical formats, for various categories of content:

Type of Content	**Life Cycle**
1. National, final programme	Keep in central archive, with reviews at times of physical format migrations (if made regionally, material moved to central archive)
Implication:	Permanent holding (or as long as the broadcaster lasts)
2. Regional, final programme	Hold in regional archive; include the content in the central archive's physical format reviews.
Implication:	Lasts for life of the carrier; could get included in a preservation migration, depending upon outcome of review. If included, moves to central archive and treated as national programme

Type of Content	Life Cycle
	programme
3. External material – cinema films Implication:	Keep just for weeks or at most months, to cover broadcast and one or two repeats (whatever rights were purchased) Short life material; complications arise for material that has been subtitled, edited, or digitised at the broadcaster's expense. Such material ends up as type 2: keep it while it lasts.
4. External material – recorded music (45, LP, CD)	Keep it while the physical copy lasts. Some broadcasters have no programme for digitisation of commercial music holdings, but NRK (Norway) is actively digitising and marketing the resultant files to other broadcasters (legally) – which is very useful for music that has not been re-issued and is 'orphaned'.
Implication:	Content will be usable for decades; much of the 45 and LP content was re-issued on CDs, and so was 'renewed'; and much of the CD content is now available electronically, or will be.
5. Live Programming	Some is selected for high-quality recording, in which case it becomes type 1 or 2 (depending on whether it is national or regional)
Implication:	Archiving of live output, especially from radio, has been highly selective. Digital production changes this situation: capture of national radio output (uncompressed) is now standard; automated full capture of TV output remains highly compressed (transmission quality).
6. Rushes material – national	Mainly held by programme-making units themselves. Central archive only holds material specified by programme-making units (with 'shelf costs' paid by them). Durations vary enormously; some programmes end and everything is dispersed/lost. Some programmes continue for decades and hold material 'while it lasts'. Rushes material held centrally is included in the reviews, but only for either handing back or destroying – not for migration.
Implication:	Lasts as long as the physical media – or less if actively destroyed / erased / re-used.
7. Rushes material – regional	Basically never archived, never catalogued, never migrated.
Implication:	As for 6, except likely to be stored in an office environment rather than in conditions of controlled temperature and humidity.

Table 1 The life cycle of 'hold it in your hand' content – on physical formats.

Constructing this table highlights how much of *all* current policy is based on physical formats, and the requirement to periodically 'move on or die'. The ecology for file-based media is entirely different:

- migration could be automated and very much cheaper
- migration from one file-type to another would only be needed for file format obsolescence; master file formats – especially if uncompressed – could last for decades (WAV was developed in 1992)
- migration from one storage medium to another may continue as generations i.e. from LTO-3 to LTO-4 data-tape, but such 'media refreshing and replacing' could become an invisible, background data centre activity
- all broadcast output can be captured, in high (archive) quality, if a central repository is connected to wherever the programmes are made
- rushes files will reside, at least for a few days or weeks, in the production systems and could also be captured by a central repository. The keep everything approach then is technical feasible.
- the triggers – the key events that initiate reviews and life-cycle decisions – are largely eliminated. The main trigger has been obsolescence of physical formats. For files, there is also format obsolescence, but this could become largely an event associated with access format.

The following figure highlights just how integrated a "media asset management" (MAM) system can be in a modern production workflow.

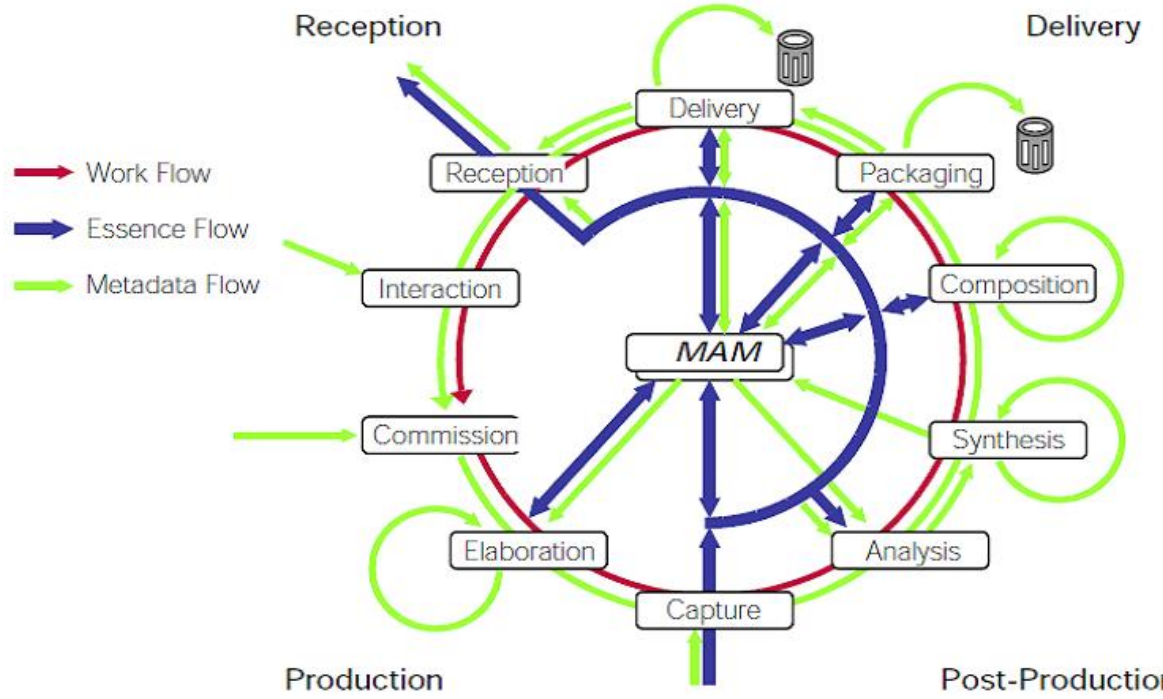


Figure 2 The content lifecycle is increasingly integrated directly into a media asset management (MAM) system. This picture was created by Blue Order.

Starting at the "commission" label the workflow and metadata go right round the process loop. Essence comes in at the "capture" stage and travels through the analysis, synthesis,

composition, packaging and delivery processes to be finally received, e.g. by the viewer. The viewer may interact with the programme (for instance with the red button on the remote control or more simply by choosing whether or not to watch the programme) and this metadata is fed back into the commissioning process.

In a file-based workflow, the MAM system and the archive can potentially become one and the same with essence and metadata being ingested and delivered at the points shown in Figure 2.

The figure gives an impression of the complexity of the content lifecycle as related to a preservation service. For new content it will not just be the case of creating a programme and then place it into the archive. For a single programme, different parts of essence and metadata are ingested and accessed throughout the production cycle. In addition, programmes are not generally commissioned singly: rather a series will be commissioned and ingested into the archive over a long period with each programme becoming one part of a greater whole.

When to Review

Rules for reviewing content can possibly be devised that are based on things that the repository can measure – like usage. Possibly content can be tagged for its preservation priority. Possibly the repository can deduce final content from rushes from the provenance, or audit trail, of final programmes. Alternatively, anything not overtly marked, by the programme makers or an archivist, as a final programme could automatically be deemed to be rushes material.

Certainly anything in the way of rule-based decisions will also need to ‘know’ about quality – which formats represent uncompressed or high-end production quality, which are transmission quality and which are web-browse quality – plus a mezzanine level somewhere between uncompressed and production/transmission/access may also be needed.

At present, all broadcast archive content is being reviewed on a format by format basis, usually within 15 to 30 years of date of creation. File-based content in a repository creates a problem: if the format doesn’t become obsolete, there is no trigger and hence no review. The format-based review cycles were the times when any rushes material that had entered the main archive would be removed from the archive: sent to the originators, or destroyed. A relatively short review cycle was important because the rushes material was largely undocumented (no analytical description) and so it had to be reviewed when there were people still around who knew something about the material.

The archiving of file-based final programmes can proceed pretty much as for tape- or film-based programmes: kept permanently. But for rushes material the current policies and rules need to be completely replaced – because much more could easily be kept, some of it could even have descriptive metadata, and there will be no obvious trigger point for review of rushes holdings.

A similar issue arises with external material (cinema productions, commercial music recordings). If such files do get into the central repository, there would be no obvious trigger point for getting them out. Again, the ingest of such content should include compulsory tagging so that the material is identified as not belonging to the broadcaster.

All such material should also be marked for inclusion in the appropriate retention schedule (a compulsory review date after a specified period).

3.2. Organisations, Contracts and Services

Aside from the lifecycle of content just discussed, there are many other entities that come and go during the scenario of producing and storing content. These include:

- content rights (e.g. copyright)
- organisations (rights-holders and service providers)
- contracts
- SLAs
- services

If we were to try and describe the relationships in a single sentence it would be:

"*Content* has associated rights; right-holder *organisations* have *contracts* with *service providers* to store the content referring to *SLAs* which may refer to specific *services*."

The relationships are further illustrated in Figure 3.

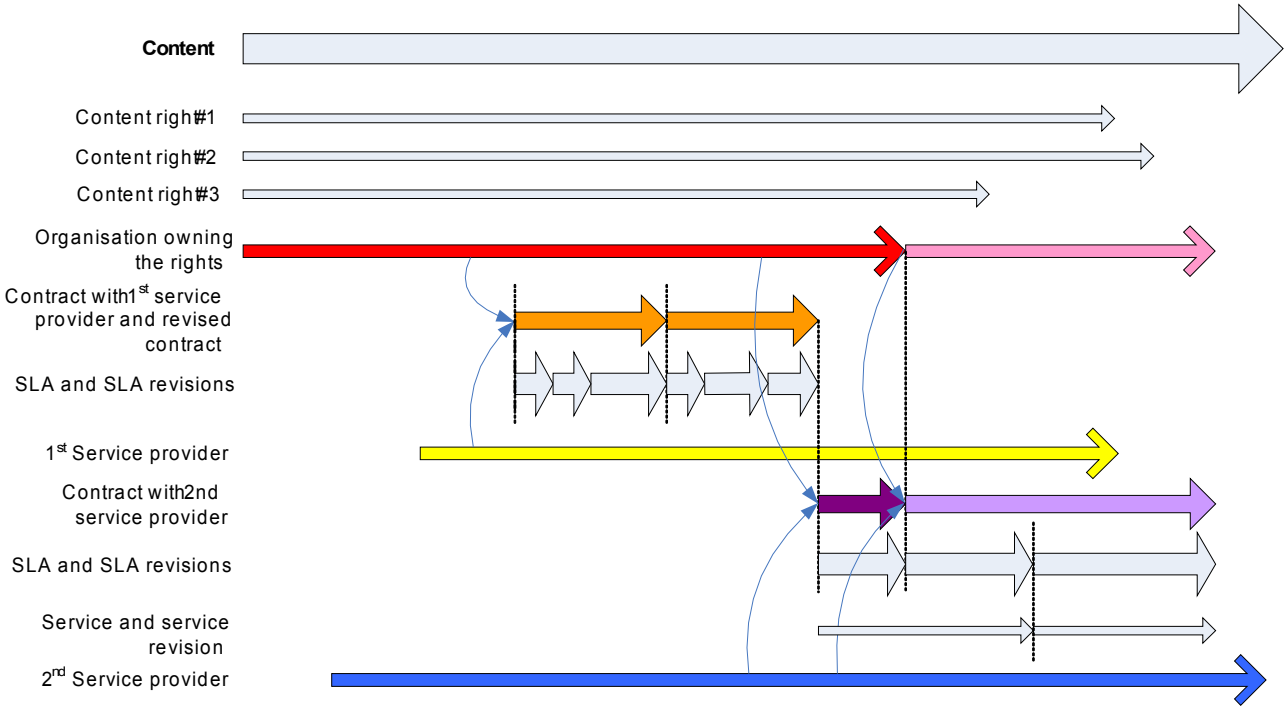


Figure 3 Illustration of how different entities come and go during the life of content.

The content can outlive all, and continues even after various economic exploitation rights on the content have expired. An organisation can own certain rights on content and some of these rights can be transferred between organisations (red and pink in the figure above).

An organisation may choose to enter into a contract (orange) with a service provider (yellow) if they wish to store the content at a service. The contract may have a fixed

duration (e.g. 10 years) and may then be renegotiated. A contract will have an associated service level agreement (SLA) which may itself be renegotiated during the lifetime of the contract.

The organisation owning the data may wish to change to a different service provider (or the service provider could be bought out by another company). In either case, a contract with a new company is required (purple contract with blue service provider) and an associated new SLA. In the example illustrated, the second service provider has referred explicitly to the service in the SLA and so the service itself has been shown on the diagram.

In our example the some rights on the content are transferred to a second organisation (pink) and so a new contract and SLA with the second service provider are negotiated but the same service is continues to be used. The second service provider then switches to a new service and the SLA is renegotiated.

Service Lifecycle

Of particular importance is the lifecycle of a service. A service goes through phases: specification, negotiation, provisioning, use and decommissioning. These phases are represented in Figure 4.

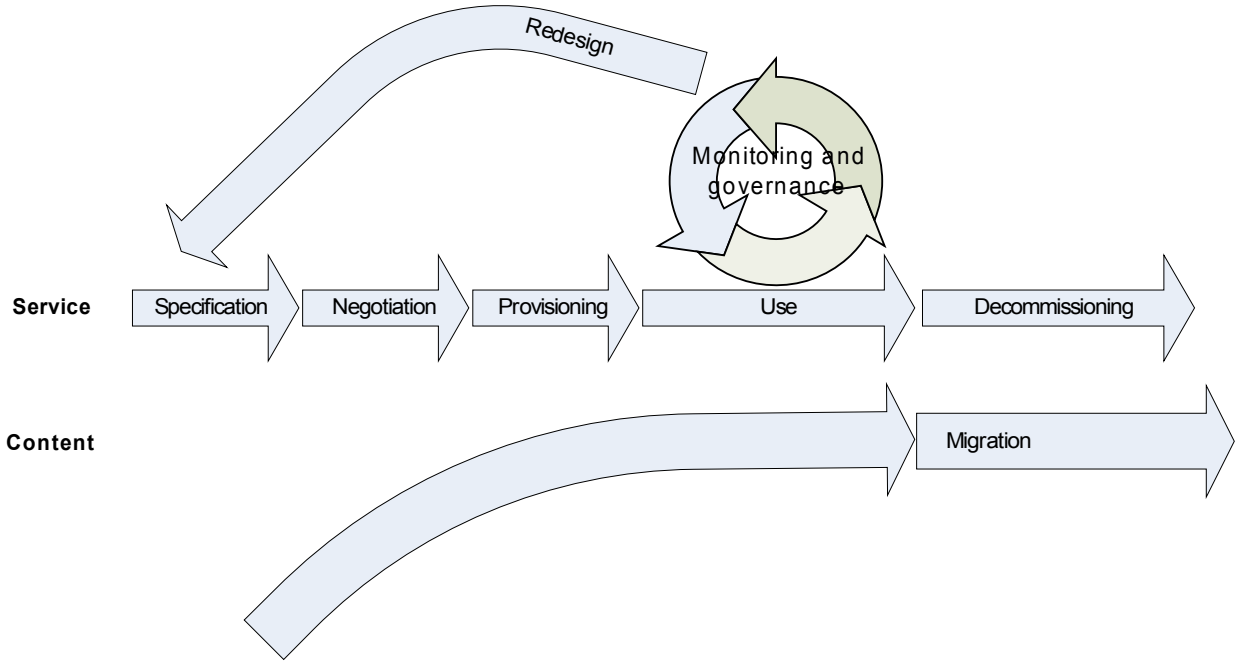


Figure 4 Service lifecycle.

The CCSDS Producer-Archive Interface Methodology Abstract Standard[5] discusses in detail the stages represented simply as “specification” and “negotiation” in the diagram above. It is a companion document to the OAIS document and defines the actions required from the initial time of contact between the producer and archive (service provider) until the objects are received and validated by the archive. This covers the OAIS “Negotiate Submission Agreement”, “Receive Submission” and “Quality Assurance” aspects. More detail on these processes is also available in the related PrestoPRIME document D2.2.1.[6]

Once a service is being used and content is ingested, a continual monitoring process should be in operation. Monitoring data should be available to a service governance system (either automatic or manual). Experience of the service behaviour and user

requirements will generally feed into an improvement or redesign process and eventually the service will be decommissioned and the data held in the service will be migrated to a new service (either at the same provider or elsewhere).

The final point, that data may need to be migrated elsewhere should be emphasised. Moving huge quantities of data from one physical location to another is very expensive (in time and money) and any contract with a service provider must include an exit strategy to cover this eventuality.

3.3. Conclusions

The lifecycles of physical assets are different to file-based assets but, as for physical assets, content in a central repository will also need to be reviewed (unless the “keep everything” strategy is seen as reasonable and affordable).

With file-based assets the potential for keeping much more arises: regional output, rushes, all broadcast output in high quality.

As content is longer-lived than the contracts that deal with the storage of said content, such a contract must have a exit strategy defined in it so that the content can be retrieved and placed elsewhere.

4. State of the Art in Storage Services

In this chapter we look at two areas of interest: how other preservation systems and projects deal with storage and what can be learnt from online storage services.

4.1. Related Projects

Currently there are several projects (some of them funded by EU in the 6th or 7th Framework Program) which aim to develop preservation solutions and are therefore strongly related to PrestoPRIME.

In this section, we investigate technologies used for storage services (i.e. Archival Storage functional entity) in the following projects:

1. CASPAR
2. Sun Microsystems' ZFS
3. SHAMAN
4. PRESERV 2 / EPrints
5. PLANETS

Projects 1, 3, 4 and 5 are OAIS compliant. It's worth noticing that none of the reviewed systems' main focus was audiovisual digital content.

CASPAR

The CASPAR project looked at the preservation of cultural, artistic and scientific knowledge. This paragraph is a summary of the relevant aspects of the CASPAR Preservation Datastore Interface report[7] and the IBM white paper on the same subject[8].

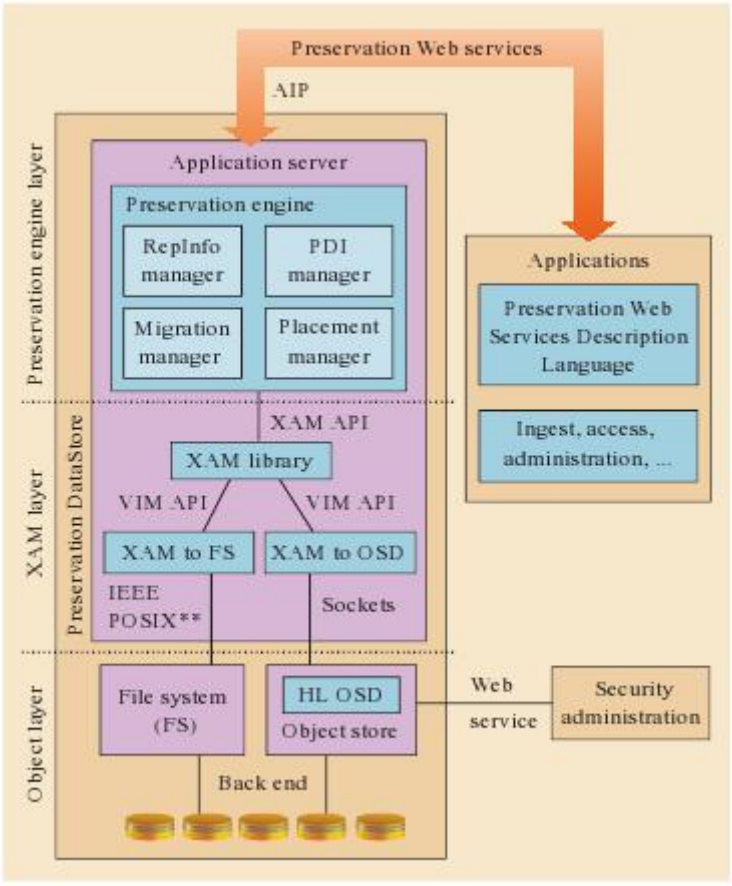


Figure 5 Preservation Data Store architecture. (Taken from Ref. 8.)

The OAIS Archival Storage is mapped to the CASPAR PreservationDataStores (PDS) module. The idea behind PDS is to transform the logical information object, i.e. the AIP, into physical storage objects. To accomplish this, a three-layered architecture was adopted, as illustrated in Figure 5.

At the top, the preservation engine layer, which is based on OAIS, provides an external interface to PDS and implements preservation functions. PDS exposes several interfaces. Implementation supports both a direct API as well as a web services API[9]. Among the interfaces there's the PDSManager. Its interface contains the main methods users may call in order to preserve AIPs. Its methods allow the user to perform several actions which can be grouped in:

- ingest AIP - implements different ways to ingest an AIP or an AIC to PDS
- access AIP - implements access an AIP or to its content data and content data RepInfo sections
- handle AIP - enables manipulating AIPs that were previously ingested
- query AIP - performs queries on the AIPs in the system
- load validation - loads content validation modules into PDS to be executed later
- handle policies - enables manipulating PDS policies

- informative entry points - provide information about the PDS system

The top layer of PDS also provides a mapping between the AIP and the XAM structures. XAM[10] serves as the storage mid-layer. The top software module in the XAM architecture is the XAM library, which exposes the XAM API. This module interacts with the application and provides a view of the underlying storage through the entities of the XAM world. Under the XAM library resides the Vendor Interface Module (VIM), which acts as a bridge between the XAM standard APIs and the vendor storage systems.

XAM Specification defines three primary objects: XSet, XSystem and XAM Library. The XSet, which is the fundamental artefact in XAM, is the basic unit of data for application to commit to persistent storage. It is a data structure that is a package of multiple pieces of data and metadata, bundled together for access under a common globally unique external name, called an XUID. The XSystem, a logical container for one or more XSet records, serves as a virtual storage to XAM applications. The XAM Library enables applications to discover and communicate with XSystems. Data can be attached to each XSet, in a form of XAM field. Each field has a unique name in the scope of its primary object. There are two types of XAM fields:

1. Properties: fields that usually include metadata and thus will be indexed and used in queries. Their type is a “simple” type and is one of Boolean, Int64, Float64, String256, DateTime, XUID.

2. XStreams: fields that include unbounded byte streams. Their type is a valid MIME-type.

Each XAM field has a fixed set of attributes for its content and behaviour:

1. Value: The actual value (content) of the field.

2. Type: The type of the value of the field; namely simple type for Properties, MIME-type for XStreams.

3. Readonly - A Boolean value indicating whether the field can be modified by an application.

4. Length: The actual size of the field value in bytes.

5. Binding: A Boolean value indicating whether the field is immutable within the XSet. If true, then field modification causes the automatic creation of a new XSet with a different XUID. This attribute is relevant only to XSets.

An AIP is mapped to an XSet. The AIP contains pieces of data and metadata bundled together under a unique ID. An AIP is constructed of Information Objects. Within an Information Object, the content data (an opaque byte stream) is naturally mapped to an XStream. The RepInfo reference is mapped to a property of type XSet reference (XUID), since each RepInfo is represented by a separate AIP and therefore mapped to a separate XSet[114,114].

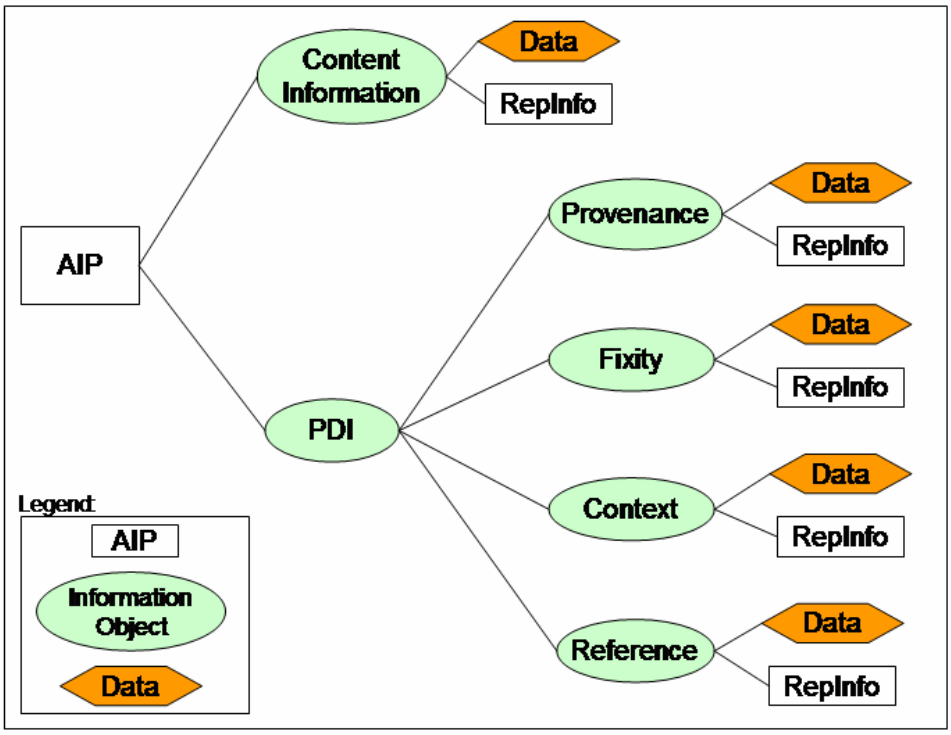


Figure 6 AIP general OAIS compliant structure (Taken from Ref. 7.)

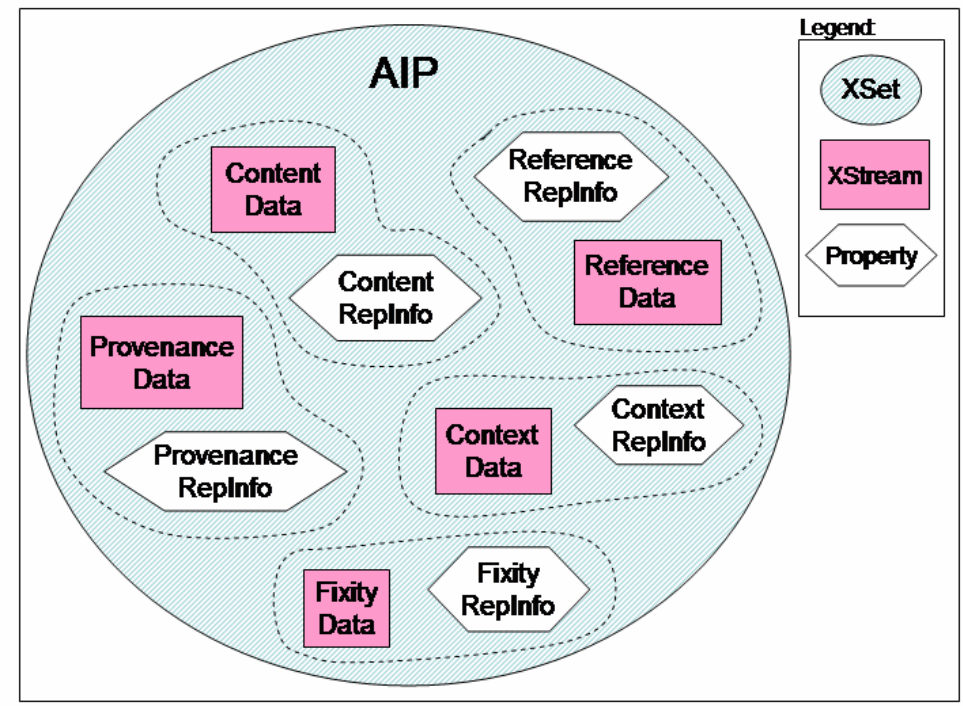


Figure 7 Mapping of AIP to XSet (Taken from Ref. 7.)

The bottom layer of the PDS architecture suggests two alternative back-end storage systems: a standard file system or an Object Store Device (OSD)[a,11]. A higher-level application programming interface (API), labelled HL OSD, on top of the OSD provides abstraction and simplification to the object-based storage device (or object store) interface,

[a] SNIA - Storage Networking Industry Association. OSD: Object Based Storage Devices Technical Work Group

which resembles a SCSI. The OSD interface is standardized as ANSI T10 SCSI OSD V1[12].

An OSD is analogous to a logical unit. Unlike a traditional block-oriented device providing access to data organized as an array of unrelated blocks, an object store allows access to data by means of storage objects. A storage object is a virtual entity that groups data together that has been determined by the user to be logically related. Space for a storage object is allocated internally by the OSD itself instead of by a host-based file system. OSDs manage all necessary low-level storage, space management, and security functions. Because there is no host-based metadata for an object (such as inode information), the only way for an application to retrieve an object is by using its object identifier (OID). The collection of objects in an OSD forms a flat space of unique OIDs. Virtual file hierarchies can be emulated by rearranging pointers to objects.

ZFS

ZFS[13] is an advanced file system developed by Sun Microsystems for Solaris OS designed to provide high storage capacities with continuous integrity checking and automatic repair.

Some of ZFS features are:

Pooled Storage Model

ZFS uses the concept of storage pools to manage physical storage. The storage pool describes the physical characteristics of the storage (device layout, data redundancy, and so on,) and acts as an arbitrary data store from which file systems can be created. File systems are not constrained to individual devices, but share space with all file systems in the pool. File systems grow automatically within the space allocated to the storage pool. When new storage is added, all file systems within the pool can immediately use the additional space.[14]

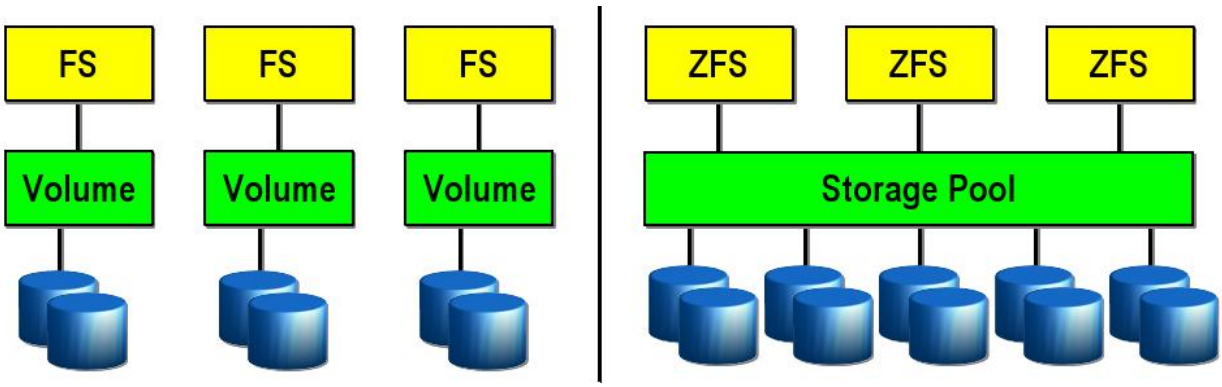


Figure 8 Traditional Volumes vs. ZFS Pooled Storage. (Taken from Ref 14.)

Transactional object system

ZFS is a transactional file system. Most file system modifications are bundled into transaction groups and committed to disk asynchronously. Until these modifications are committed to disk, they are termed pending changes. Data is managed using copy-on-write semantics. Data is never overwritten, and any sequence of operations is either entirely committed or entirely ignored. This mechanism means that the file system can never be corrupted through accidental loss of power or a system crash. While the most recently written pieces of data might be lost, the file system itself will always be consistent.

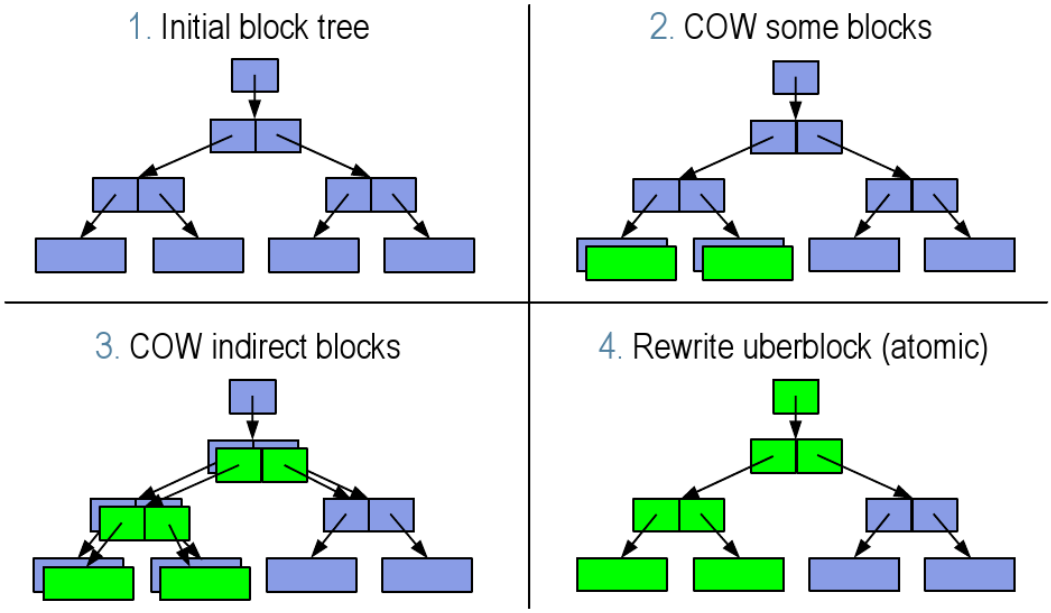


Figure 9 Copy-On-Write Transactions. (Taken from Ref. 14.)

The Data Management Unit (DMU) is a general-purpose transactional object store. It consumes blocks and groups them into logical units called objects. Objects can be further grouped by the DMU into object sets. Everything in ZFS is an object.

The ZPL, ZFS POSIX Layer, makes DMU objects look like a POSIX filesystem. POSIX is a standard defining the set of services a filesystem must provide. ZFS filesystems provide all of these required services.

Besides file systems, ZFS Storage Pool can supply volumes to applications that need a raw-device semantics. For instance, ZFS volumes can be used ad swap devices or can be exported through the network by means of iSCSI.

Protection from data corruption

All block pointers within the filesystem contain a 256-bit checksum (currently a choice between Fletcher-2, Fletcher-4 or SHA-256) of the target block which is verified when the block is read. Traditional file systems that do provide checksumming have performed it on a per-block basis, out of necessity due to the volume management layer and traditional file system design. The traditional design means that certain failure modes, such as writing a complete block to an incorrect location, can result in properly checksummed data that is actually incorrect.. ZFS validates the entire I/O path. In addition, ZFS provides for self-healing data. ZFS supports storage pools with varying levels of data redundancy. When a bad data block is detected, ZFS fetches the correct data from another redundant copy, and repairs the bad data, replacing it with the good copy.

Snapshots and clones

A snapshot is a read-only copy of a file system or volume. An advantage of copy-on-write is that when ZFS writes new data, the blocks containing the old data can be retained, allowing a snapshot version of the file system to be maintained. ZFS snapshots are created very quickly, since all the data composing the snapshot is already stored; they are also space efficient, since any unchanged data is shared among the file system and its snapshots.

Writeable snapshots ("clones") can also be created, resulting in two independent file systems that share a set of blocks. As changes are made to any of the clone file systems, new data blocks are created to reflect those changes, but any unchanged blocks continue to be shared, no matter how many clones exist.

Capacity

ZFS is a 128-bit file system, allowing for 256 quadrillion zettabytes of storage (1 zetabyte = 1 ZB = 1 billion terabytes). Directories can have up to 2^{48} entries, and no limit exists on the number of file systems or number of files that can be contained within a file system. In fact, it is claimed that to fully populate a 128-bit storage pool would require more energy than it takes to boil all the world's oceans[15].

ZFS is part of Sun's Solaris operating system and is thus available on both SPARC and x86-based systems. Since the code for ZFS is open source, a port to other operating systems and platforms can be produced without Sun's involvement.

OpenSolaris 2008.05 and 2009.06 use ZFS as their default filesystem[16].

ZFS has been part of FreeBSD since version 7.0.

ZFS port to NetBSD was started as a part of the 2007 and in August 2009 the code has made it into NetBSD's source tree.

Complete ZFS support was once advertised by Apple as a feature of Snow Leopard Server (Mac OS X Server 10.6). In October 2009, Apple announced a shutdown of the ZFS project on Mac OS Forge. No explanation was given.

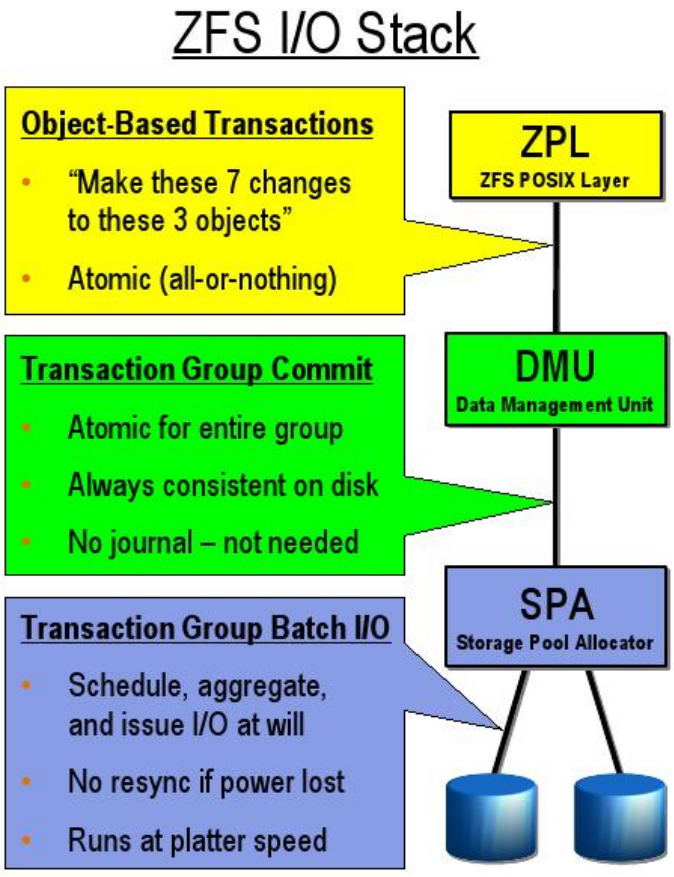


Figure 10 ZFS I/O stack. (Taken from Ref. 14.)

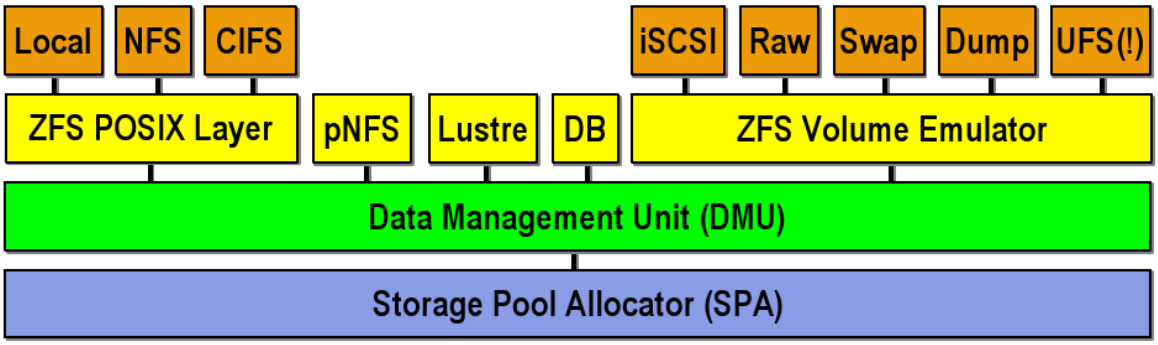


Figure 11 Universal storage. (Taken from Ref. 14.)

SHAMAN

The EU funded project SHAMAN[17] (Sustaining Heritage Access through Multivalent ArchiviNg) uses Data Grids as a storage technology. In particular, making use of iRODS[18] (integrated rule-oriented data system), the open source successor to SRB.[19] The overall architecture of the framework is shown in the illustration below.

SHAMAN is a policy based system. It uses the concept of multiple independent policies /procedures / workflows for managing preservation. Examples include replication of data across multiple storage systems, association of descriptive and provenance metadata with each record, and uniform creation of AIPs for all records independently of where they are stored. Additional policies govern the federated environment. These policies control the replication of data between the independent data grids, the synchronisation of records between a deep archive and an access environment, and the identification of the authentic source. In the SHAMAN framework, policies include the use of Multivalent to automate the viewing of individual frames every three months; the use of Multivalent to identify file formats; the capture of audit trails, etc.

Policies are implemented using the iRODS data management system. In fact, the main new feature of iRODS with respect to SRB is the rule engine that enforces the policies.

Rules follow the event-condition-action paradigm and run on the iRODS servers as micro services. Micro services can be implemented and integrated via a plugin feature in iRODS, so there are no limitations on functionality and extensibility. Examples for such micro services are to create a copy of an ingested object or check an object for integrity based on checksums. Furthermore micro services can then be connected to more complex rules, which can again follow events and conditions.

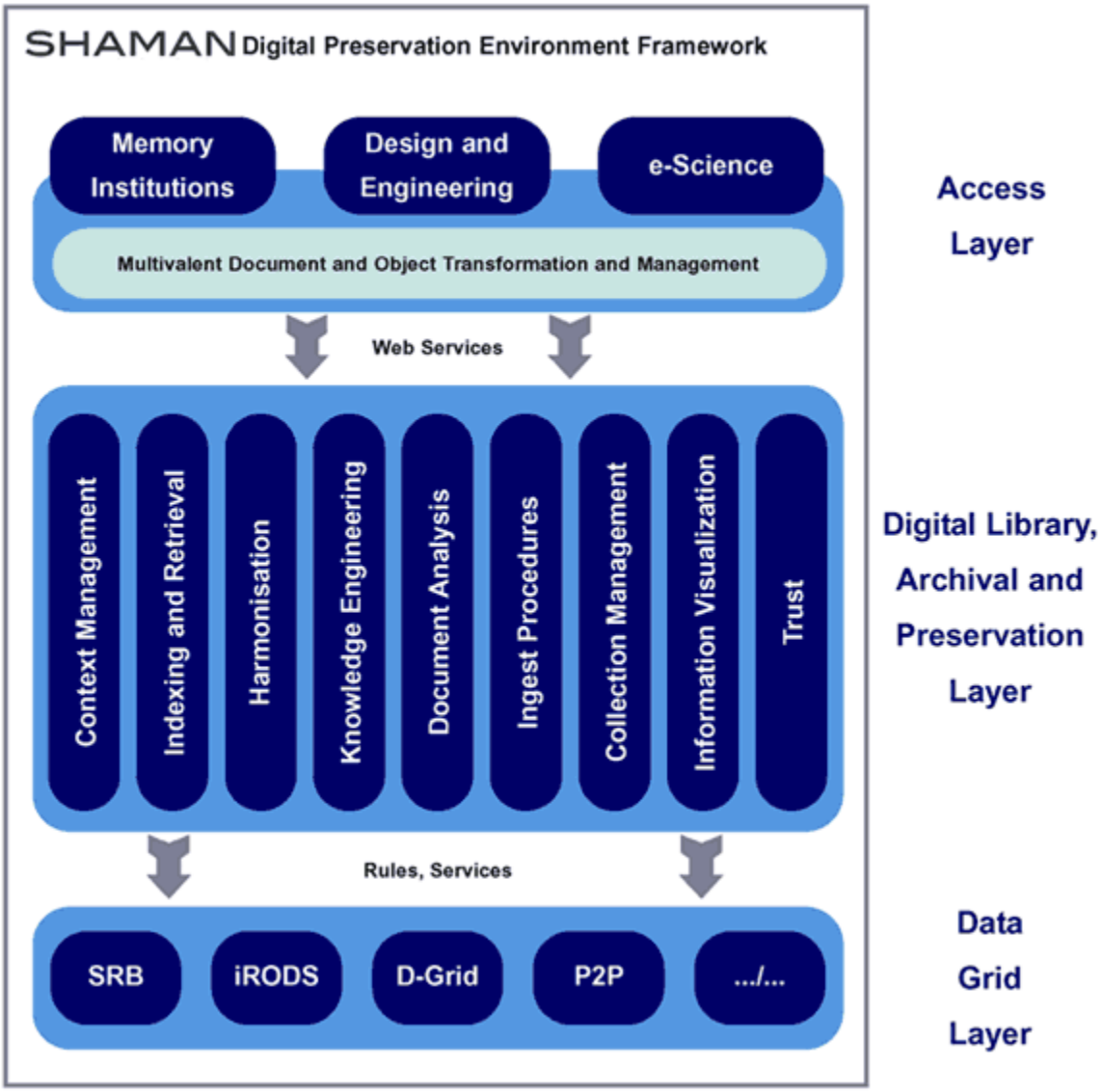


Figure 12 Overall SHAMAN Architecture[20] (Taken from Ref. 20)

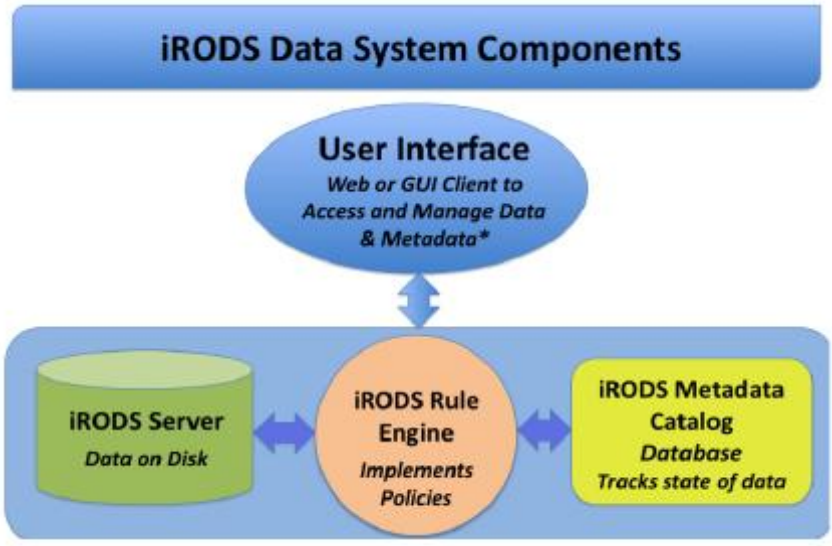


Figure 13 An overview of the iRODS system. (Taken from Ref. 18)

iRODS Server software and Rule Engine run on each data server. The iRODS iCAT Metadata Catalog uses a database to track metadata describing data and everything that happens to it. User and Community-defined Policies are mapped to computer-actionable Rules applied at each storage system. Procedures are mapped to workflows composed by chaining Micro-services (C code encapsulating a desired function). Workflows are executed at storage locations, under Rule control. State information in iCAT Catalog tracks outcome of each Micro-service, can be queried to validate assessment criteria such as data authenticity. Access controls and audit trails can be analysed to verify policy enforcement, enabling Trustworthy Repositories.

iRODS interfaces let users search for, access and view, add/extract metadata, annotate, analyse and process, manage, replicate, copy, share, re-purpose, control and track access, subscribe, and more.

IRODS clients for managing data are[20]:

- iRODS Web-based Browser
- iRODS Explorer for Windows
- iRODS Web Client (Python)
- iRODS i-Commands (Command line interface utilities, the most complete command set)
- DAVIS (WebDAV interface)

There are currently four client APIs:

- icommands (C)[20]
- pyRODS (Python)[20]
- JARGON (Java)[21]
- PRODS (PHP)[22]

SHAMAN integrates the Cheshire3[23] analysis system and the Multivalent Browser technology with the iRODS data grid. Cheshire is a full-text information retrieval system based on an fast XML search engine. On the basis of indexes it gives access to the essential search and browse functionality of digital libraries. Cheshire's development started 10 years ago at these UC Berkley and currently is at version 3 developed by the University of Liverpool. It supports several protocols like Z39.50, SRW/SRU or OAI-PMH for access of metadata. A key part of the integration was the implementation of micro-services that are capable of executing Python scripts. This enabled the iRODS rule engine to control workflows that included Cheshire3 functions. The Multivalent Browser technology (media engine) provided a means to parse legacy data formats using portable parser technology written in Java.

PRESERV2 / EPrints

Founded by JISC[24], Preserv2[25] was created with the aim of implementing and testing web services for combining bitstream storage and 'active' preservation. Preserv2 developed the

first repository storage controller, which will be a feature of EPrints version 3.2 software (due 2009)[26].

The EPrints Storage Controller enables the repository manager to use multiple storage platforms including local, institutional and cloud based.

The storage controller decides where to put a file. It uses rule based policy defined by a configuration file (XML). For example, large binary files of scientific data (raw machine result data) can be stored in a large disk (slower access) system and sent to a tape company for long term storage; processed results can be stored locally and in the cloud ready for rapid delivery to end points.

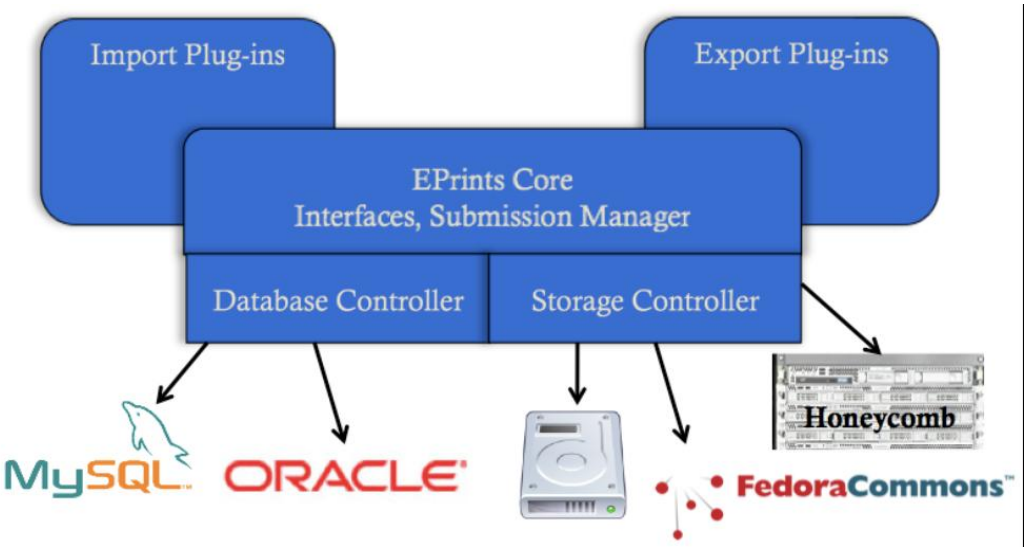


Figure 14 EPrints architecture. (Taken from Ref. 25.)

The following plug-in applications that use the controller have been written:

- Local Plug-ins (Local Disk, Local Compressed)
- Local Archival Plug-ins (Honeycomb, NAS, SAN)
- Cloud Plug-ins (Amazon S3, SunCSS, Flickr)

More plug-ins can be created for any storage service with an application interface, and will work with the storage controller in v3.2.

Preserv2 uses OAI-ORE[27] (Open Archives Initiative Object Reuse and Exchange) to move data between two repositories with quite distinct data models, from an EPrints repository to a Fedora repository and then back again[28].

OAI-ORE defines standards for the description and exchange of aggregations of Web resources. These aggregations, sometimes called compound digital objects, may combine distributed resources with multiple media types including text, images, data, and video. Binding objects in this manner would allow the construction of a layered repository where the core is the storage and binding and all other software and services sit on top of this layer. In this scenario, if a repository wanted to change its software, instead of migrating the objects from one software to another, we could simply swap the software.

OAI-ORE specifies import and export interfaces to enable the re-use and exchange of digital objects. From a digital preservation perspective, this enables future migration of objects to a newer platform while preserving the functionality expected from the digital repository.

PLANETS

Planets[29] Interoperability Framework (IF) is an OAIS compliant platform which enables the user to perform preservation actions.

It consists of a pre-configured JBoss Application Server, a set of IF core components:

- Administration Interface
- Monitoring and Logging
- Service Interface
- Service Registry
- Data Registry
- Workflow Design Tool

IF uses Enterprise Java Bean architecture. The release report can be found at ref. [30]. Planets IF is downloadable at ref. [31] where the source code and Javadoc are available as well.

Relevant to our discussion is the Data Registry module, which provides storage of and access to files and metadata. The Data Registry uses the Apache Jackrabbit[32] implementation of the standard Java Content Repository API[33] (as specified by Java Specification Request 170). Binary data is stored by reference only to avoid the overhead of submitting and accessing it through the XML-oriented Jackrabbit.

The data registry is implemented using Jackrabbit exposed through JNDI (Java Naming and Directory Interface) on JBoss. The full API is available as Java methods callable by IF components; a portion of the API is also exposed as a web service interface.

The data registry controls the Planets shared storage area, i.e. common network storage accessible by any IF component. Each registered user has their own storage area in which they can:

- Add content file references and associated metadata
- Save references to files created by workflow tasks
- Save metadata generated by workflow tasks
- Search content and create standard IF fileset messages using Xquery

A low-level graphical interface to the underlying Java Content Repository the Planets Data Registry is built on is also available. The GUI is web-based and implemented on the JSF1.2 specification using the Apache MyFaces implementation. Functions offered are:

- Browse content/data (by navigating through the tree representation)
- Create, Delete, and Modify content (e.g. by adding new content, modifying properties, etc.)
- Search content (by both XPath and SQL queries)

4.2. Online Storage Services

Many online storage services currently exist, perhaps the best known example being Amazon S3. None are “preservation-grade” systems but by reviewing them we can tease out what features exist and what is important to PrestoPRIME partners.

The facilities these vendors provide vary greatly and the range of information available depends on the information publicly available for each service by the provider, and information taken from sites like Wikipedia. Some vendors provide little information on the technology behind the service (e.g. Box.net) and others promote the technology underlying their as a selling point (e.g. Allmydata and Tahoe). All providers give *no guarantee* of the safety of data despite presenting their storage as a reliable backup for users, so we get conflicting statements like

- Front Page Splash: “Allmydata was created for all of us who have lost important information due to disk crashes, viruses or even a mere computer replacement”
- Terms and Conditions: “Allmydata shall not be responsible or liable for the deletion, correction, destruction, damage, loss or failure to store any Data”

The policy on the ownership of data varies greatly, for instance:

- “You agree that Box.net or its licensor's retain all proprietary right, title and interest, including copyright and all other intellectual property rights, in and to Box.net service and content, including, without limitation, text, images, and other multimedia data.”

We need to look at the following factors in relation to storage services:

1. Security – How do we authenticate?
2. Management – Can we manage the usage of the service, does it have any SLAs?
3. Federation – Can we federate two heterogeneous services together; does the service federate internally itself?
4. Trusted Repository – Does the service conform to TRAC and OAIS requirements?
5. Large Data Objects – How large can each object in the storage be and how much storage can be used overall
6. Rules Engine – Can we apply rules and policies to data, how do we specify these rules?
7. Interfaces – How do we access the service?

8. Integrity – How does the system cope with failing hardware?

There are also various research efforts on the next generation of tools and services, for example A-Stor[34] developed by IT Innovation as part of the UK TSB funded AVATAR-m project[35] federates across multiple back-end storage systems (file-systems, FTP, etc) to provide a unified view of storage resources and uses a rule-engine to define preservation policies. We haven't reviewed all of these and instead in this report focus on existing tools and services that are commercially available or open-source.

Summary

It would not be useful to try to list all the available storage services. Instead we pull out some interesting details of selected services.

Remote Storage Service

Providing On-line Storage as a remote service:

1. Amazon S3: highest consumer profile, cloud, open API and tools, 5Gb per file.
2. Allmydata: uses erasure coding, open source back-end (Tahoe), 8Gb per file.
3. Nirvanix: enterprise service, cloud, custom API, 256Gb per file

Also considered: Digital Bucket, Box.net, Tarmin, Evault, Iomega IStorage, Strong Drive, Windows Live, BT on-line backup.

Software Storage Solutions

For installation at the client site on client hardware:

4. Tahoe: erasure coding, open source, unlimited storage.
5. Eucalyptus: S3 interface, cloud, open source, tools, commercial side to help maintain code.
6. Parascale: private cloud.

Hardware/Software Solutions

Installation of hardware and software at the client site:

7. Permabit: uses erasure coding.

Details

Amazon S3

Amazon S3 (Simple Storage Service) is the best known online storage service. It provides “unlimited” storage through a simple web services interface. S3 was launched in the US in March 2006 and in Europe in November 2007. Since its inception, Amazon has charged end users $0.15 per gigabyte-month, with additional charges for bandwidth used in sending and receiving data, and a per-request (get or put) charge. As of November 1, 2008, pricing moved to tiers where end users storing more than 50 terabytes receive

discounted pricing. Amazon S3 is reported to store more than 52 billion objects as of March 2009. S3 comes with no guarantee that customer data will not be lost.

Features:

- Provides REST and SOAP APIs to access data. Default access is via HTTP but also provides a Bit Torrent interface.
- File sizes up to 5GB.
- Simple security model: public or private objects and associated authentication with backend encryption.
- A simple SLA stating that Amazon commits itself to a "monthly uptime percentage" of 99.9% and that if this is not met then users will receive a refund of a proportion of what they pay.
- Security through username/password, X.509 certificates and public URLs with expiry times.
- Provides data bulk data ingest and access by shipping hard disc drives using courier services.

Pricing:

- Storage: $0.150 per GB-month of storage used (for first 50 TB of data, $0.055 per GB-month for data over 5000 TB).
- Data Transfer: data transfer in is currently free, data transfer out ranges from $0.170 per GB for the first 10 TB per month to $0.100 per GB when going over 150 TB per month.
- Requests: $0.01 per 1,000 PUT, COPY, POST or LIST requests and $0.01 per 10,000 GET and all other requests (delete free).

Of the services investigated, Amazon may have the infrastructure scale required for AV archiving but the 5GB limit on file sizes lets it down. Whilst the SLA is about as basic as it can get, the service is used and relied upon by many commercial enterprises. For instance, the photo sharing website SmugMug stores its users' photos in S3, saving $500000 on hardware costs in 2006 and adding 10 TB of data per month to the system.

Allmydata

Allmydata was created as a consumer back up service. It has a unique selling point of using erasure coding[36] to improve its robustness to disc errors. Erasure coding is implemented at Allmydata using the Tahoe distributed file system. Upon ingest, the system splits the file into many pieces and stores the pieces on many discs. When the file is accessed, the original file can be reconstructed from a small subset of the pieces (typically 3 out of 10).

Features:

- A variety of access methods are supported including mounting the file-system as a local drive.
- File sizes up to 8GB (though 8TB is planned).
- Security through username/password, encrypted transfer and storage.
- Uses erasure coding for added safety.

Pricing:

- Remarkably, the charge is a flat rate of $9.99 per month. However, if your usage greatly exceeds the average usage, they say they will contact you to revise the pricing plan.

Nirvanix

The Nirvanix Storage Delivery Network a managed, secure "cloud storage service" designed for enterprise rather than personal use. Files are stored on one to three RAID 6 file-systems depending on the customer's choice (and purse).

Features:

- Access is through CIFS, NFS, FTP, SOAP or REST. In addition there is a custom API providing an "Internet File System" which provides translation from POSIX style file-system commands and RESTful HTTP calls to the cloud. In addition to standard file system operations, media services for transcoding video etc are provided at the storage locations.
- File sizes up to 256GB.
- The SLA is very similar to Amazon S3's and just commits Nirvanix to 99.9% availability for a singly-replicated file through to 100% availability for a three-times replicated file with service credits in the case of failure.
- Security is provided through optional SSL transmission and AES 256 encryption in the file-system. Extensive physical data-centre security is also provided.
- In addition to the SLA, Nirvanix state they have a "Statement on Auditing Standards No. 70 (SAS 70) Type II certification that verifies that our control processes are documented and have followed industry standard requirements for more than a year".
- Provides data bulk data ingest and access by shipping hard disc drives using courier services.

Pricing (for basic no-contract up to 2 TB service with single copy stored):

- Storage: $0.25 per GB-month
- Data transfer: $0.18/GB

Tahoe

The Tahoe project[37] provides the distributed, encrypted file-system that is used by the aforementioned Allmydata commercial offering. The software is actively developed and is available under the GNU General Public License v2 or Transitive Grace Period Public Licence.

Eucalyptus

Elastic Utility Computing Architecture for Linking Your Programs To Useful Systems (Eucalyptus[38]) is an open-source software infrastructure for implementing "cloud computing" on clusters. The current interface to Eucalyptus is deliberately compatible with Amazon's EC2, S3, and EBS interfaces, but the infrastructure is designed to support multiple client-side interfaces. Eucalyptus is implemented using commonly available Linux tools and basic Web-service technologies making it easy to install and maintain.

The software has been included in Ubuntu since version 9.04 and is available under the GNU General Public License v3.

Parascale

Parascale[39] markets its software both to storage service providers and directly to enterprises. The software provides a “private cloud” facility with automatic load balancing across many physical nodes. Data safety is managed by storing a set number of file replicas on different nodes in the system and in the case of disc failure the system will keep on running while additional replicas are automatically created to replace a lost disc.

Permabit

Permabit provides a hardware and software solution to enterprises. The interesting aspect of their solution is that rather than using RAID they use erasure coding to ensure data safety. This is the same class of technique as that employed by Allmydata and Tahoe mentioned above. Permabit claim that their implementation of erasure coding is 250 times more robust than RAID 6 and uses “50% less capacity… compared to mirroring technology”.

Permabit’s solution also tackles issues such as data retention for regulatory compliance and de-duplication.

4.3. Conclusions

Similar preservation projects provide some useful insight into how to adapt the OAIS model into software. The iRODS system also has an interesting rule engine that must be considered in more detail.

There are many online storage services available. We are not suggesting archives should use them store their most valuable data, not least because of the difficulty of ingesting data over the internet. The online services do demonstrate some interesting features nonetheless such as the use of erasure coding instead of RAID in the case of Allmydata and Permabit and the phenomenal success of Amazon S3 even with a remarkably simple SLA.

Tahoe, Eucalyptus and AStor all provide interesting features for storage: an encrypted, erasure coding file-system, a clone of Amazon S3 and EC2 and a system for storage providing both federation and a rules engine.

5. Value Networks and Architectures

There is a wide range of concepts available for storing large amounts of data for the long term today. These concepts combine different technologies in hardware and software and may involve distributed locations. Examples are data tape, storage area networks (SANs), hierarchical storage management (HSMs), Grid and even Cloud storage.

Furthermore, the concepts can involve service providers and consequently certain parts and responsibilities of a storage system can be outsourced. The following schema shows the different options for outsourcing that affect the architecture of a storage system:

1. Storage system hardware:
    a. can be purchased as a ready-made storage system
    (e.g. a tape library including tape drives, cartridge slots, barcode reader and a tape robot)
    b. or can be constructed internally
    (e.g. the IT department could create their own storage area network with several storage technologies at hand; another more likely example are tapes on shelves).
2. Storage system management software:
    a. can be purchased from a company,
    b. can be purchased from a company who customises the software according to the specific requirements of the organisation
    (e.g. the hardware requirements)
    c. or can be developed internally.
3. Storage system management/administration:
    a. can be performed by the organisation itself
    b. or can be performed by a service provider.
4. Location of the storage system:
    a. can be a location managed by the organisation itself
    b. or can be a location managed by the service provider.

Any combination with the different values of the criteria is possible though they might be more or less likely.

But not only outsourcing can have an impact on the storage architecture. In addition there are different classes of data to consider. For audiovisual archiving there are broadly three commonly recognised classes of data:

- Browse quality (suitable for quick viewing, perhaps through a web browser)
- Exploitation quality (suitable for reusing in new programmes)
- Master copy (the original recorded data)

The different types of data may be more or less appropriate to store in different locations using different technologies.

In this chapter we examine and classify example architectures found today and consider what architectures we may see in years to come. The focus lies on concepts involving service providers in different stages and distributed locations. The first subchapter illustrates therefore the value networks that are applied by organisations in order to provide storage facilities. The second subchapter then presents different storage architectures on an abstract level, however with examples from the real world.

5.1. Value Networks

An archive organisation is part of many inter-related value networks, dealing for instance with the retrieval and usage of content, how to sustain an archive by generating income and the expenses of an archive including staff, electricity, buildings and services.

In this document we focus on the services that an archive organisation might use to provide storage facilities. The related document D2.1.1 discusses value chains and business models of archives for obtaining revenue and preserving file based content. The value networks are presented as value network diagrams, using ovals to depict actors, solid arrows for tangible values and dashed arrows for intangible values.

A common pattern found today is for an archive organisation to manage its own storage systems. For this they still need to obtain hardware and software which will normally be bought in from suppliers along with support contracts. This value network is shown in Figure 15. The archive organisation states the requirements to the suppliers and receives not only the actual product, software or hardware, but also updates and support at a later point in time from the suppliers. These receive in turn a payment for their efforts.

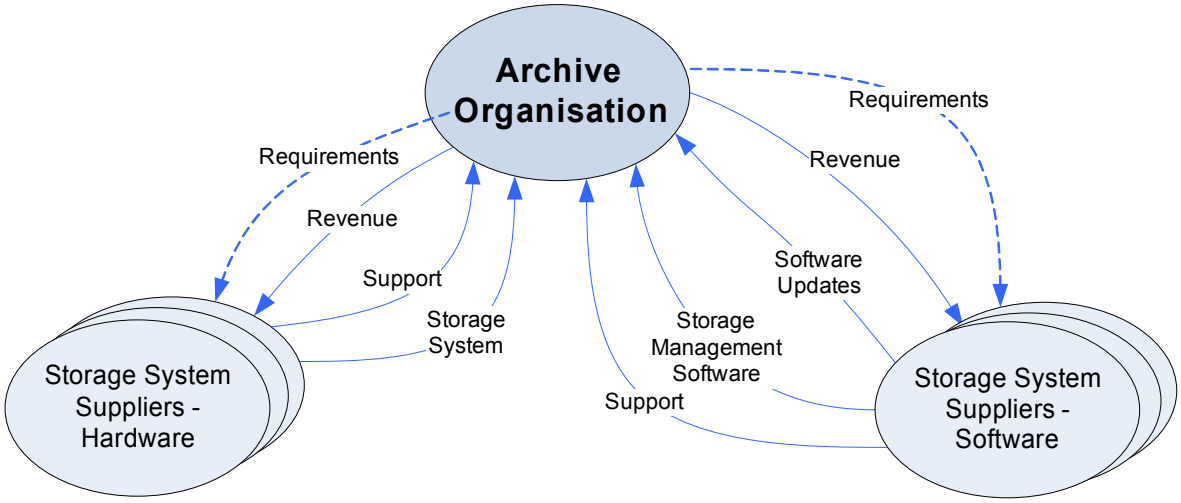


Figure 15 Value network when storage facilities are provided in-house.

Another common model is to put the responsibility for the storage system in the hands of a service provider. The corresponding value network can be seen in Figure 16. Here it is the service provider who interacts with storage system suppliers for hardware and software. The archive organisation thus only receives the storage system service and support and provides the requirements and revenue.

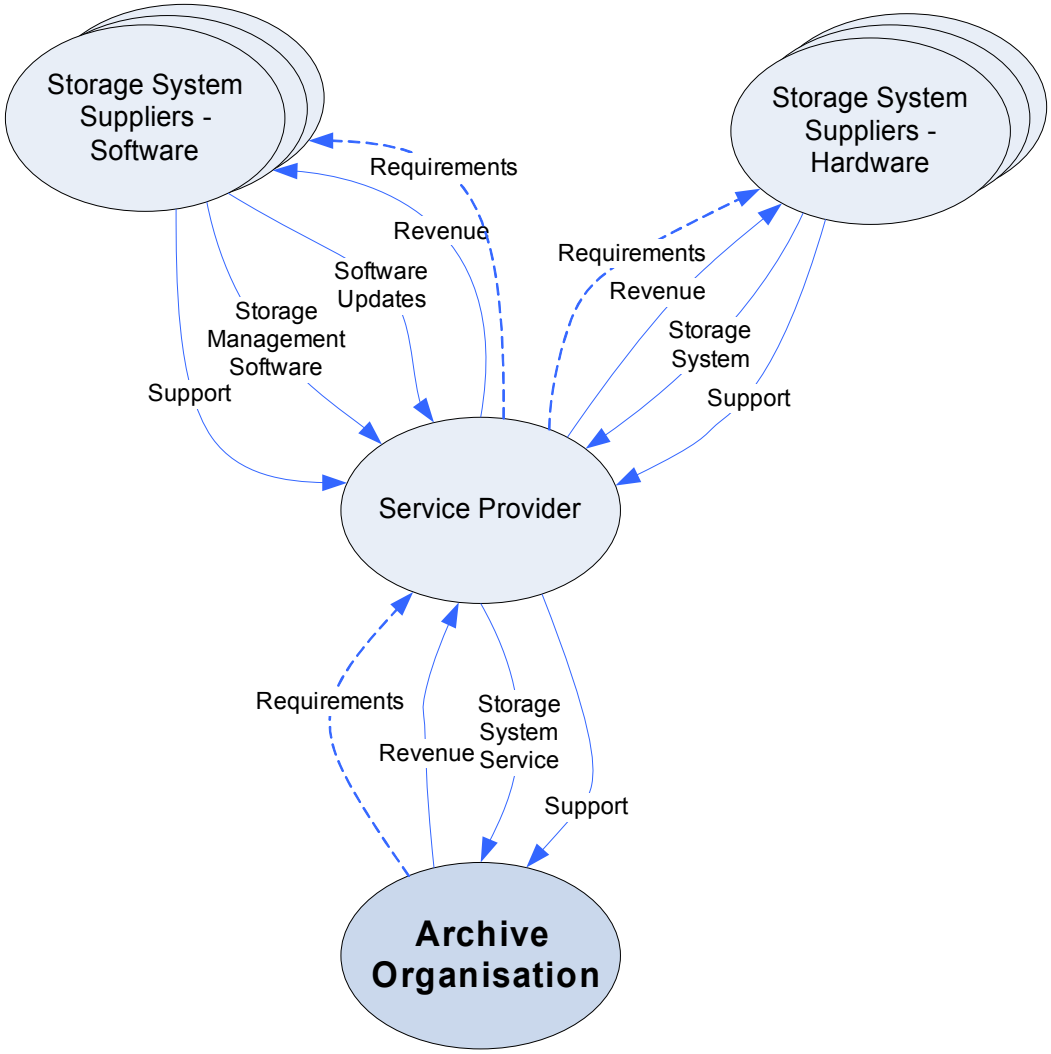


Figure 16 Value network when using a service provider to provide storage systems.

In the future we may see archives using multiple storage providers. Figure 17 shows this option with two service providers, but the diagram could be extended by additional service providers anytime. The advantage to be gained with this is the potential increase in the safety of the data: For example two different service providers could mirror the data, each in a different location. They might even use different technologies. Consequently the data would be twice as safe though of course there are increased management problems.

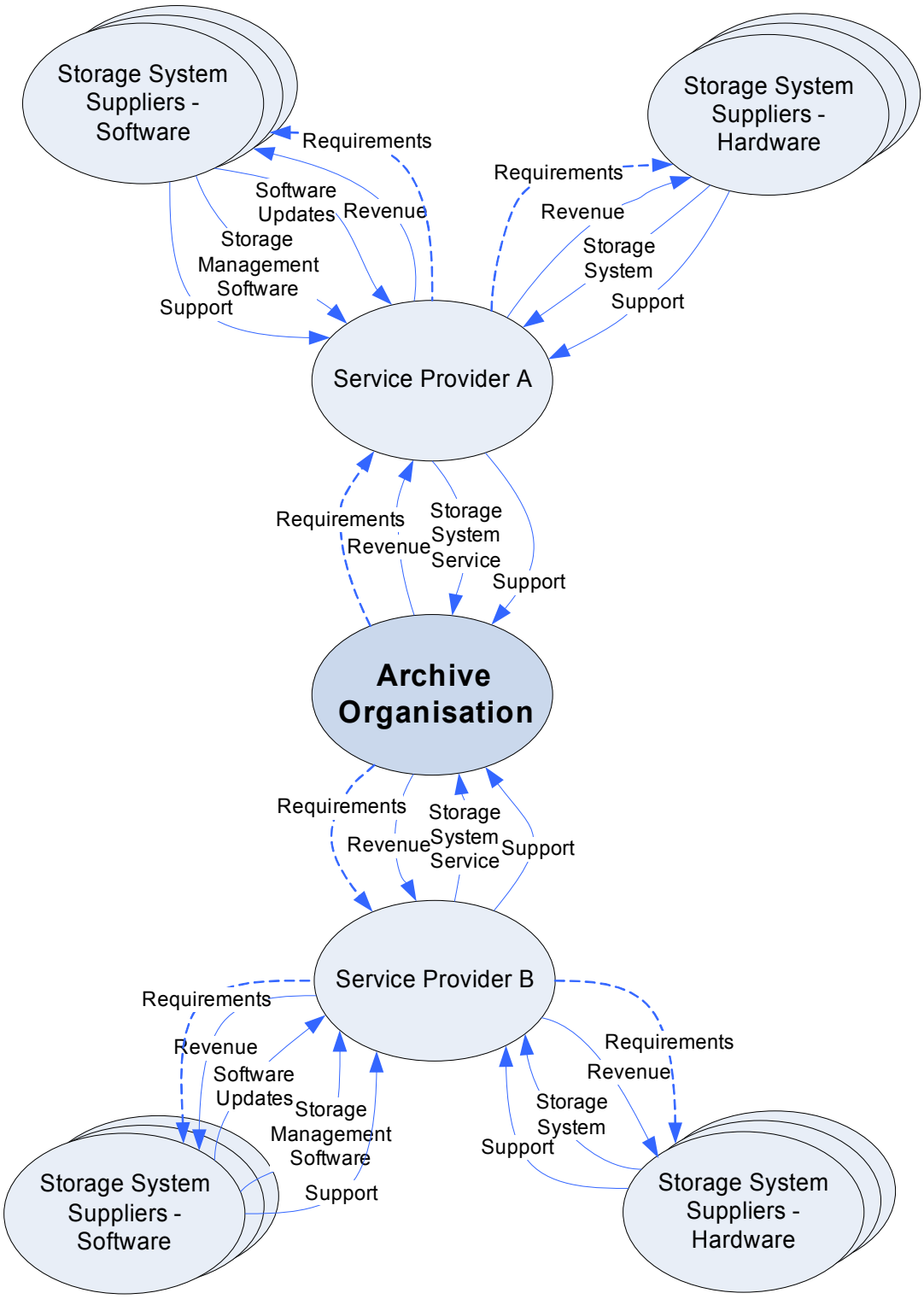


Figure 17 Value network when using multiple service providers to provide storage systems.

Now that we have introduced possible value networks to provide storage infrastructure for an archive organisation, the next step is to look at the architectures that apply these value networks.

5.2. Architectures

In this chapter we present storage architectures found today or likely to be applied in the future. The different models are illustrated in an abstract way on the one hand and, if existing, in a detailed diagram with an example from a case study that implements the model on the other hand.

The abstract diagrams show locations and storage systems as rectangles. Different colours are used to indicate that storage systems and locations are managed by different actors:

- Blue rectangles imply management by the archive organisation or broadcaster,
- pink rectangles imply management by a service provider
- and yellow rectangles imply management by a second service provider.

There are three broad types of content shown as being stored, these are:

- Browsing: used for examination of archive content during a search activity
- Exploitation: a format the quality of which is typically lower than the master but which permits commercial exploitation and, technically, simple production activity, although fine post-production processes might be compromised.
- Master: the highest quality original.

Solid arrows may represent the ingestion or access of content and dashed arrows indicate the direction of backup or mirroring processes.

The detailed diagrams have a more complex notation. A legend for them can be found in the appendix.

Archives with a Single Controller

First we consider a variety of architectures where one company is in control of the entire system. By that, we do not exclude outsourcing of functions to other organisations but the important distinction is that the functions and performance of the outsourced systems are specified and controlled by the main location. This is distinct to architectures where two separate archives cooperate on various levels discussed later.

Here we discuss different architectures, all found in systems today, ranging from no outsourcing to the complete outsourcing of the storage of some forms of data in multiple locations.

The simplest architecture of course is that where all systems are on one site and are managed by the archive owner. Simplicity has many advantages but for many archives this solution is not appropriate as off-site backups are required and out-sourcing of various functions is necessary to reduce costs. An example of a large university data centre architecture not specifically designed for archiving but representative of its type is shown in Figure 18.

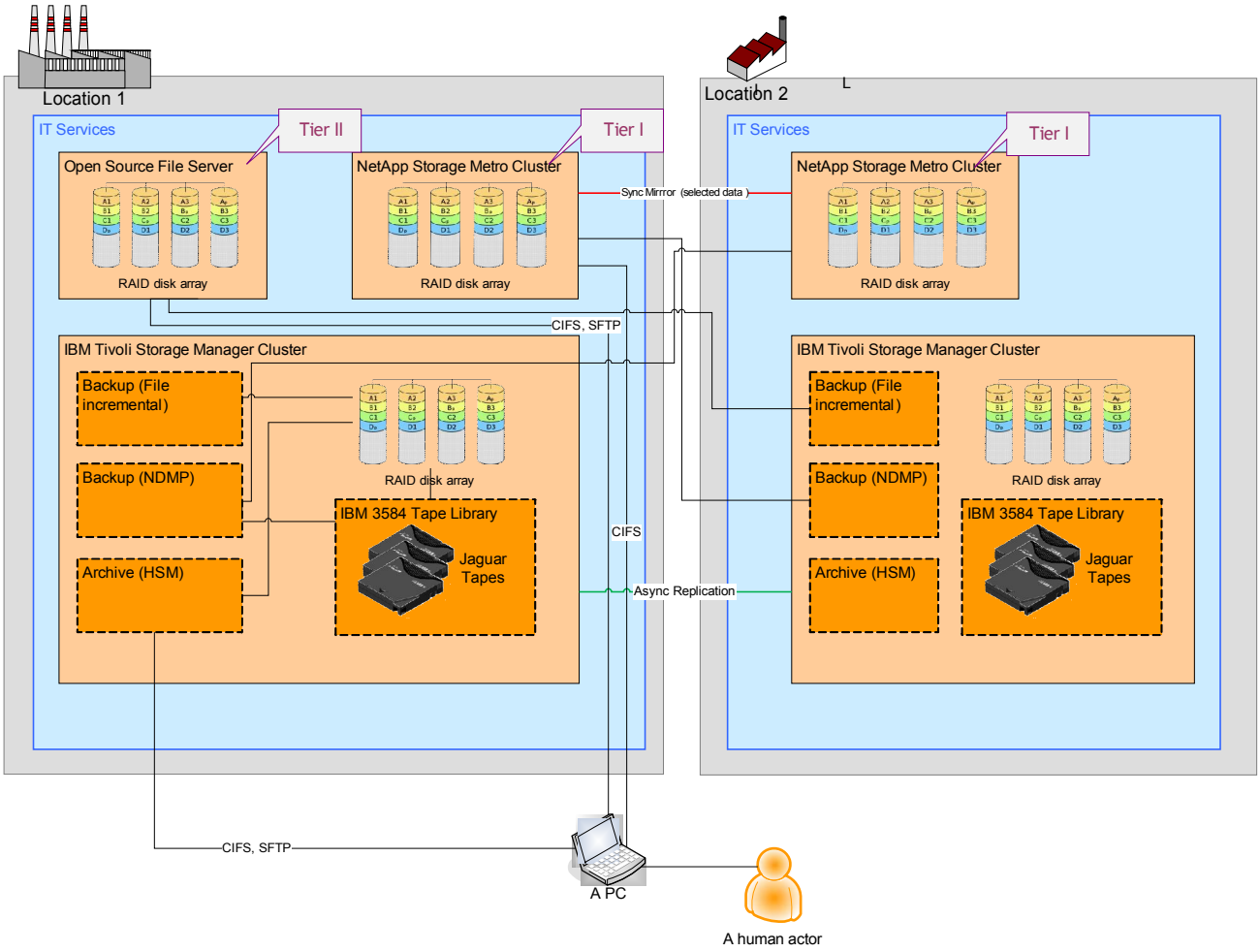


Figure 18 The architecture of a university data centre.

Figure 19 shows simply the option of managing all storage systems within the archive organisation but with off-site facilities. As already suggested different content types are held in separate storage systems and an offline storage system is used to backup all data in a separate location. The valuable master quality data may also be held in a single but separate location such as a specialised data vault.

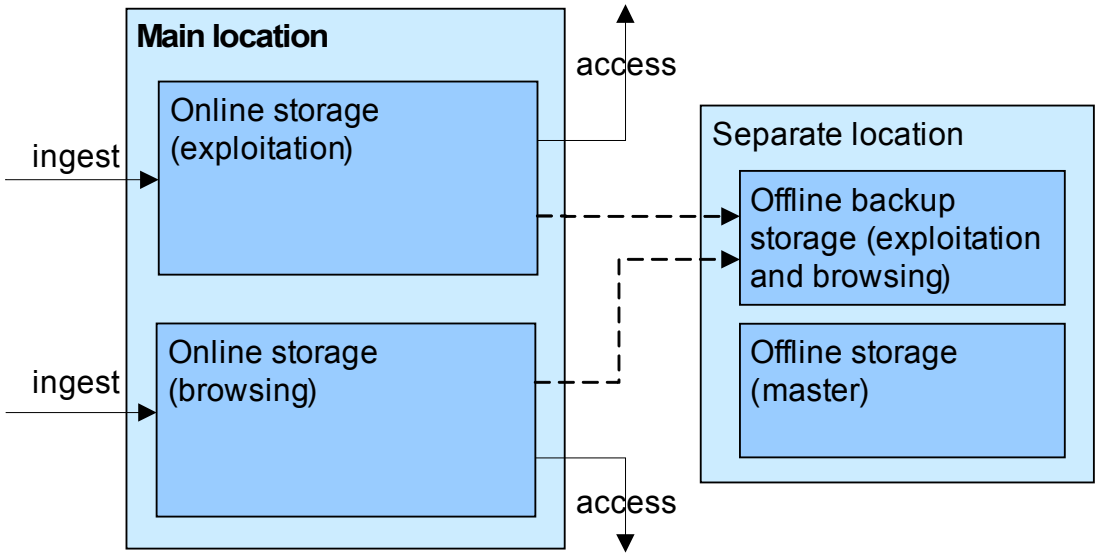


Figure 19 A simplified example of managing the storage facilities within one organisation.

The option of using off-site facilities is for example implemented in a national TV archive which is shown in Figure 20. It can be clearly seen how different storage technologies and storage systems are used to store and mirror the different types of content. Also several locations have been set up for secure mirroring and backup.

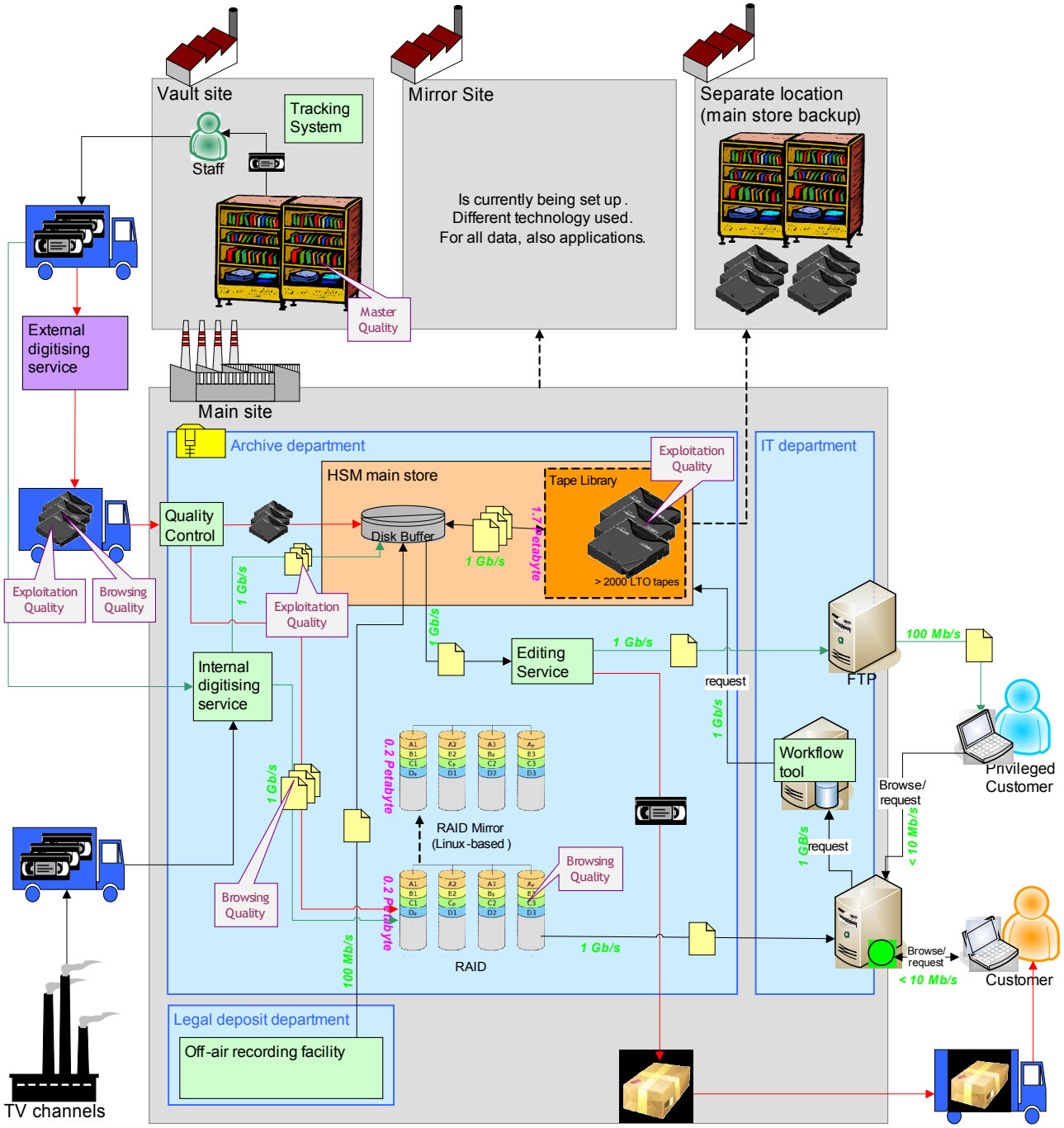


Figure 20 The storage system architecture of a national TV archive.

Since many existing audiovisual storage systems have evolved rather than been designed from scratch, a relatively simple step is to take an existing internal system and have it managed instead by a service provider. An example of this scenario is shown in Figure 21. The organisation has outsourced the provision and management of the online systems but they remain on the organisation's site.

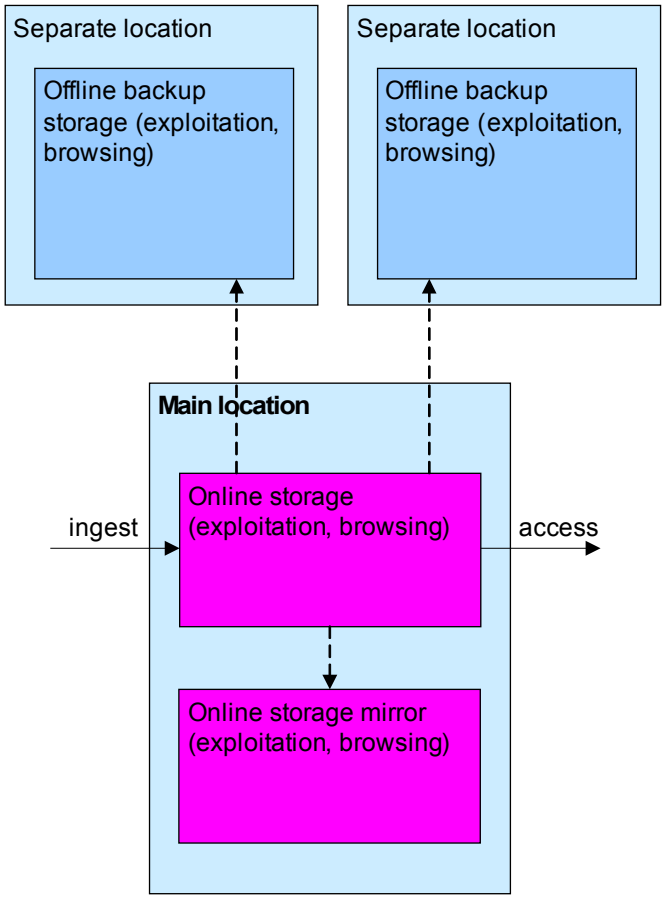


Figure 21 A simple example of outsourcing the management of on-site storage facilities.

An example implementation of the scenario can be seen in Figure 22 where a major news broadcaster has outsourced its IT infrastructure. The advantage of this approach is that the crucial storage systems are on site where the data is used rather than in another location where the connectivity might be influenced by network problems like potential latency and low bandwidth.

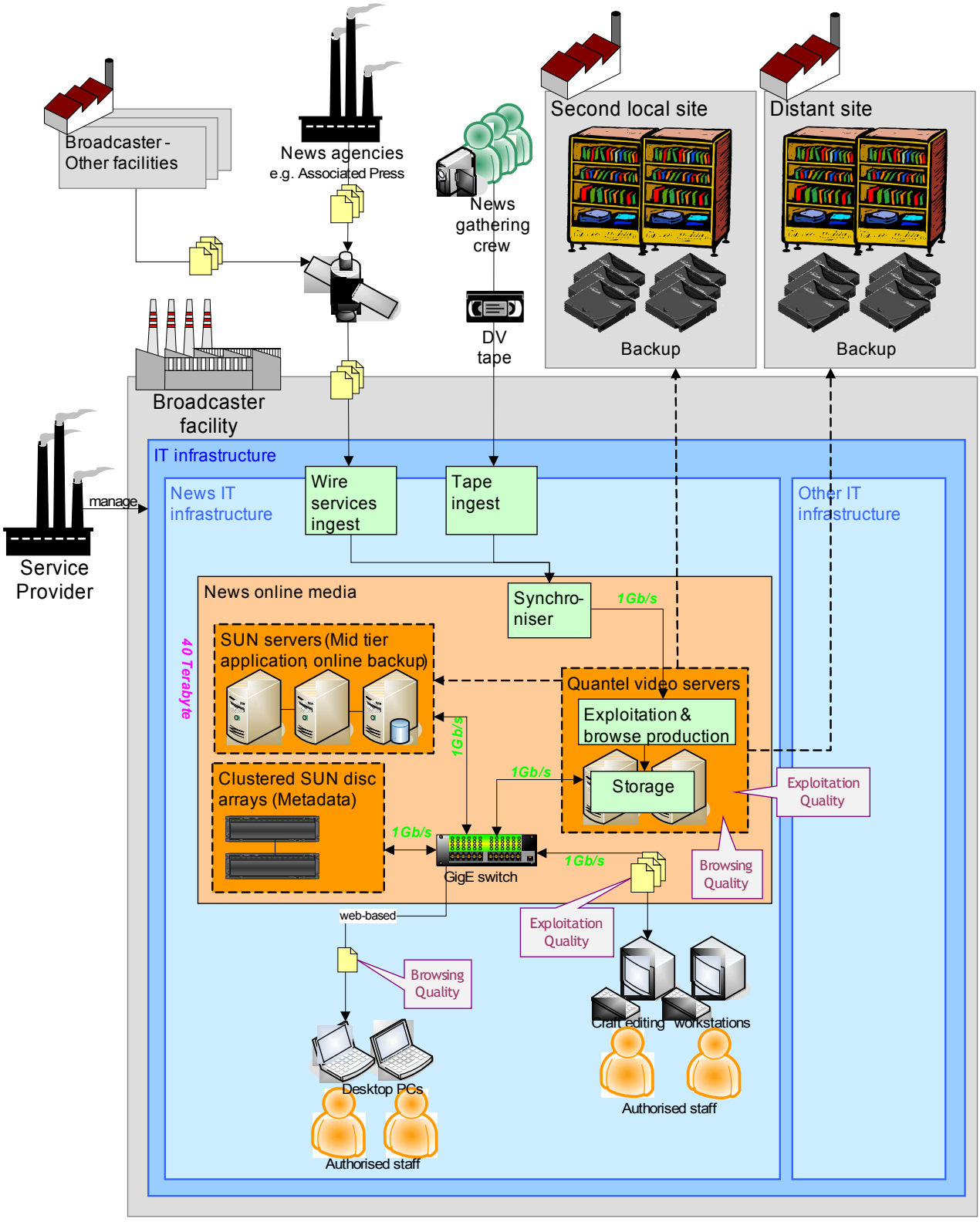


Figure 22 The storage system architecture of a large broadcast newsroom. No "master quality" material is generated here as news production works in lower quality than other departments using DVCAM tape and DV files. This is still higher quality than transmission quality.

Another approach is to outsource both the management of the online storages and their location which can be viewed in Figure 23. An offline backup is retained at the main site. Access to the content still passes through a web or software interface managed by the organisation.

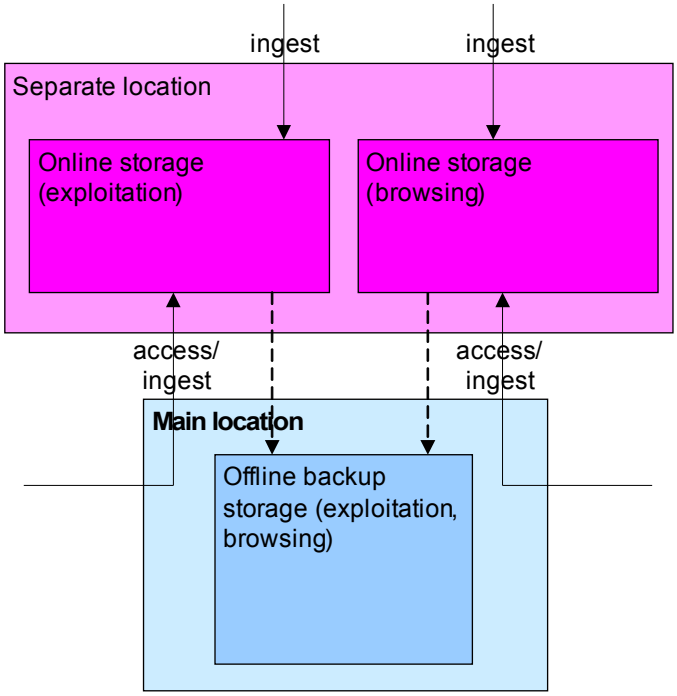


Figure 23 An example of outsourcing management and location of the online storage systems.

An example for this approach can be found in the storage system architecture of another European archive facility where only the offline backup storage is managed in-house (Figure 24). As an archive institution they use the advantage of not having to provide facilities for space, technologies and personnel in order to run the storage system and focus rather on the provision of a service interface for their clients and the public.

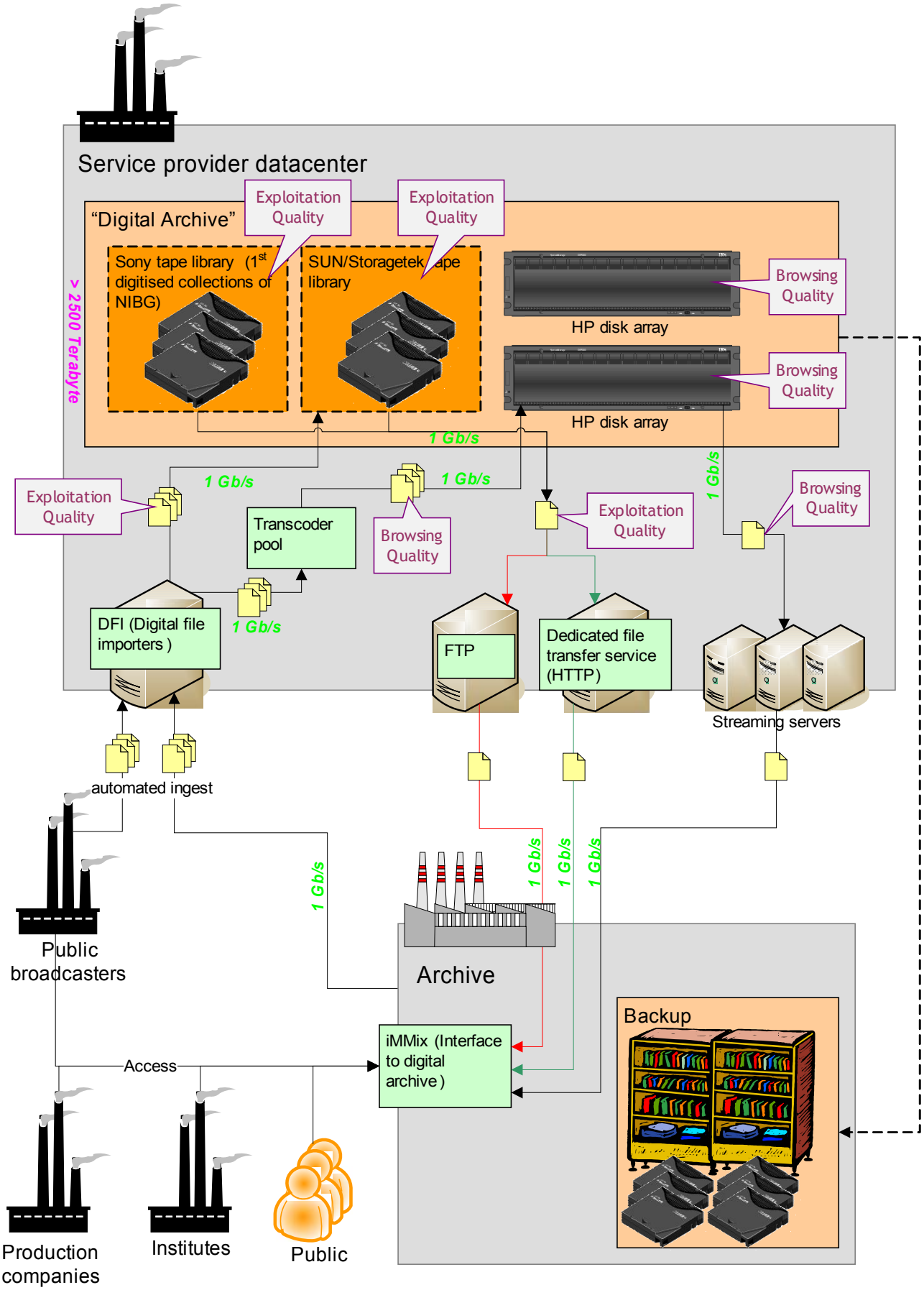


Figure 24 The storage system of a European archive. In this case, the "exploitation" and "master quality" material are the same.

A similar but even more sophisticated concept is the outsourcing of online storages into separate locations as presented in Figure 25. Additional security against loss is provided as in one location the storage of the other location is mirrored and vice versa. The content in both storage systems is accessible via a central cache and access is routed through the main organisation.

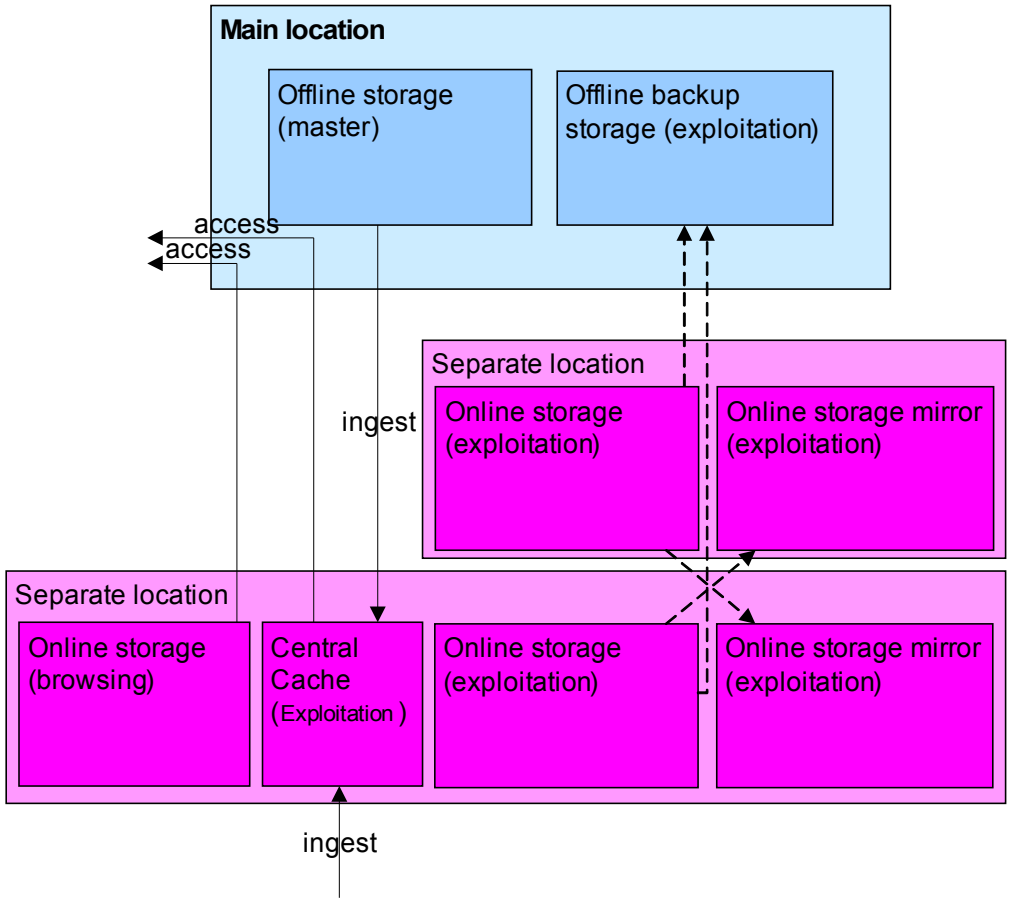


Figure 25 An example of outsourcing the online storages into separate locations.

An implementation of this concept can be seen in the up-to-date storage architecture shown in Figure 26.

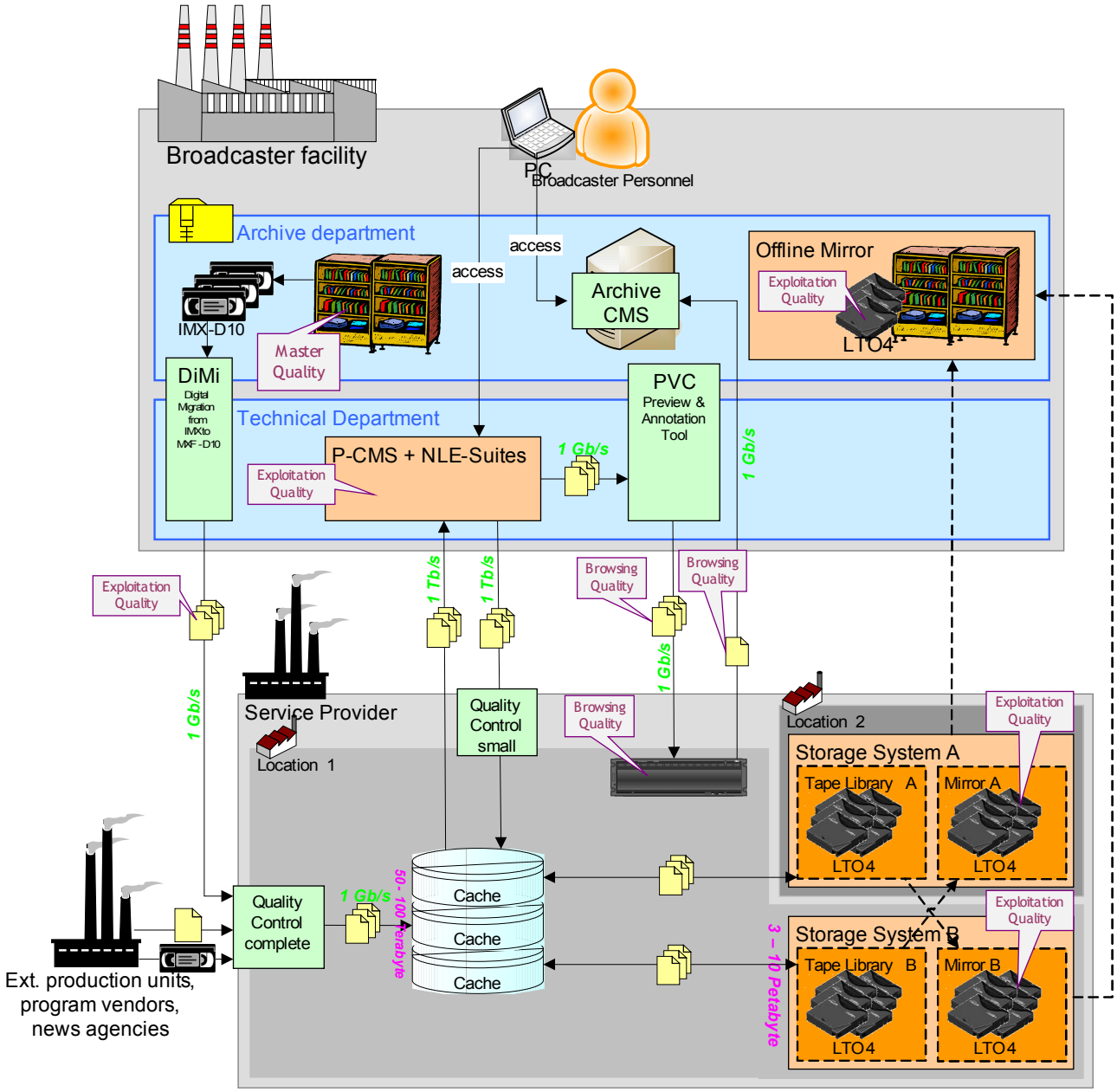


Figure 26 The future storage architecture a mid-European broadcaster.

The concept shown in Figure 25 is likely to be extended in the future in order to achieve an even greater security by diversifying not only the locations, but also the service providers managing the storage facilities. Figure 27 illustrates what such a scenario could look like.

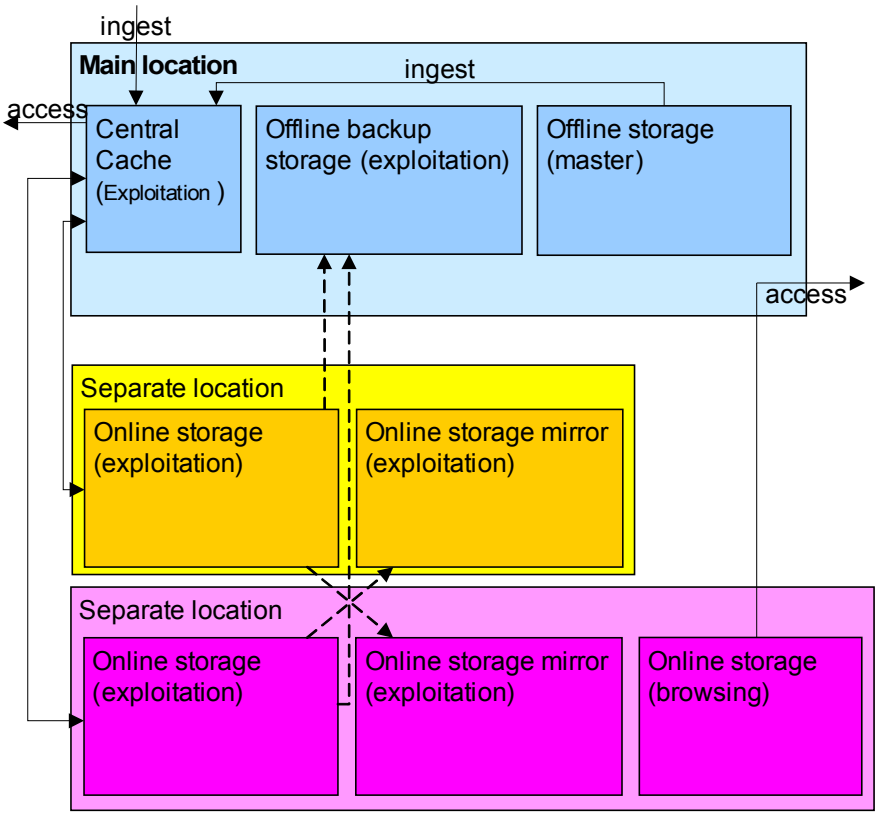


Figure 27 An example of outsourcing the online storages into separate locations managed by different service providers.

In this scenario, where two service providers are commissioned to store data for the main location one must pay attention to the issues of both data and identity federation. The concept of “federation” in this context refers to separately managed systems conforming to some agreed common practices so that they can interoperate closely and even appear indistinguishable to the end user. Identity federation therefore is about permitting end users to use the same identity in multiple locations, for instance being able to log into two separately managed data stores with the same username and password. The “Security” section below discusses this in more detail. Data federation in this case is achieved by having all access for exploitation quality material pass through a central cache in the main location. The cache can be intelligent and know where to fetch the data from without the consumer even knowing that there are multiple sources.

The scenario is not necessarily limited to using two service providers and locations as it is shown in the diagram. Any amount is imaginable when thinking of grid or cloud technologies. CERN for example uses grid technology to store the research data of the LHC experiments. This involves the redundant distribution and storage of the data in various data centres across the world[40]. However it is questionable if such concepts are already applicable for audiovisual content as the technologies used are still under research and therefore not mature enough.

Cooperating Archives

Cooperating archives are discussed in the OAIS Blue Book[41] in section 6.1.2. The concept refers to two separately managed archives cooperating loosely by agreeing on at least one common *SIP* and *DIP* format. There are two variants:

1. Cooperating archives with a mutual exchange agreement where each can request content from the other.

2. Cooperating archives with standard ingest and access methods where producers and consumers can use both archives in the same way.

These two scenarios are shown in the following two figures.

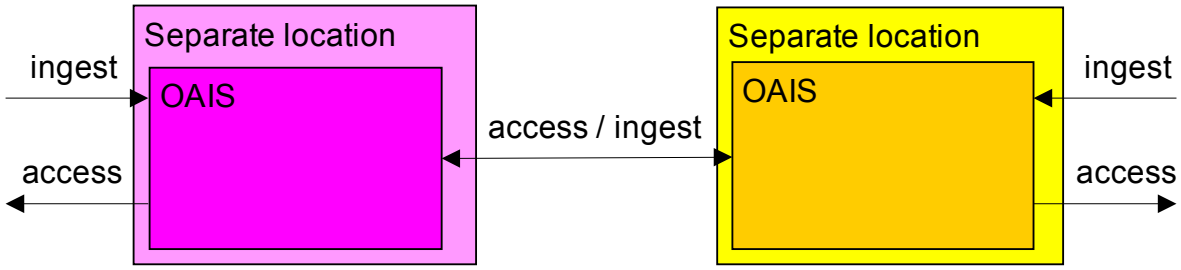


Figure 28 Simple cooperation between OAIS archives

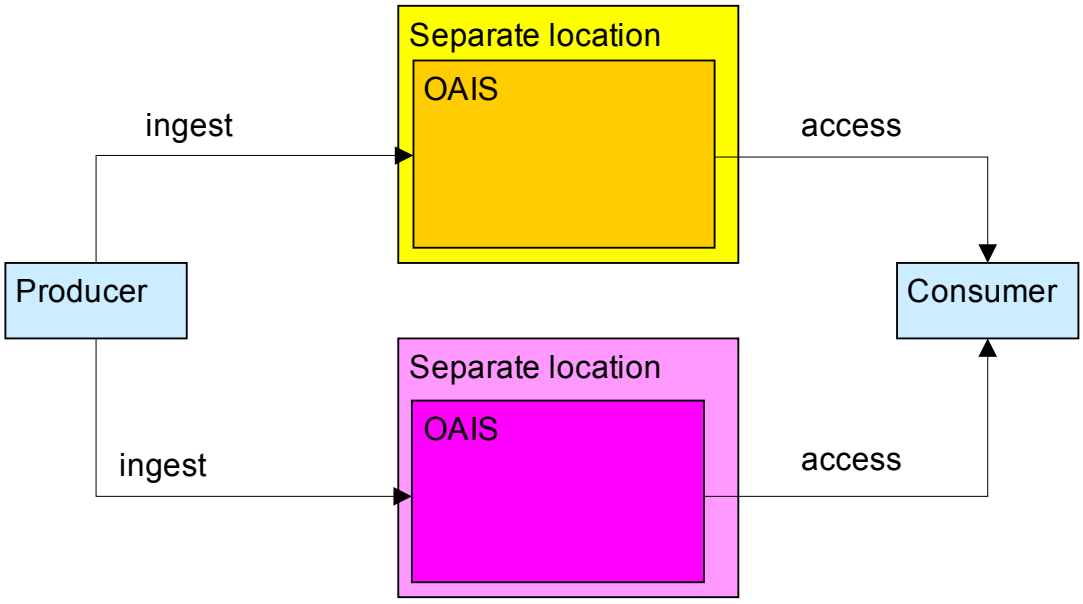


Figure 29 Indirect cooperation between archives by standardised SIP and DIP.

Federated Archives

Federated archives are discussed in section 6.1.3 of the OAIS Pink Book. If two separate archives have sufficiently similar *designated communities* then they may wish to cooperate more closely than just agreeing on common data formats.

OAIS distinguishes between three levels of functionality:

1. *Central site:* Here a central site keeps a catalogue of the holdings of the federated sites and presents this to the consumer who can search the catalogue. To access an object of interest the user goes directly to the site that holds the object of interest.

2. *Distributed finding aid:* In this case the central site can distribute a query to the federated sites, potentially translating the format of the query as necessary. As with the *central site* federation, the consumer goes directly to the site holding the object to access it.

3. *Distributed access aid:* The third level of functionality adds a standard ordering and dissemination mechanism so that the consumer does not need to directly visit the federated site to access the object of interest.

An example of the "central site" system is OAIster project hosted by OCLC[42]. OAIster harvests data from Open Archives Initiative (OAI)-compliant digital libraries, institutional repositories, and online journals using the Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH) protocol and contains more than 23 million records. The advantage of this architecture Is that the search can be very fast but may not be quite up to date. This architecture is shown in Figure 30.

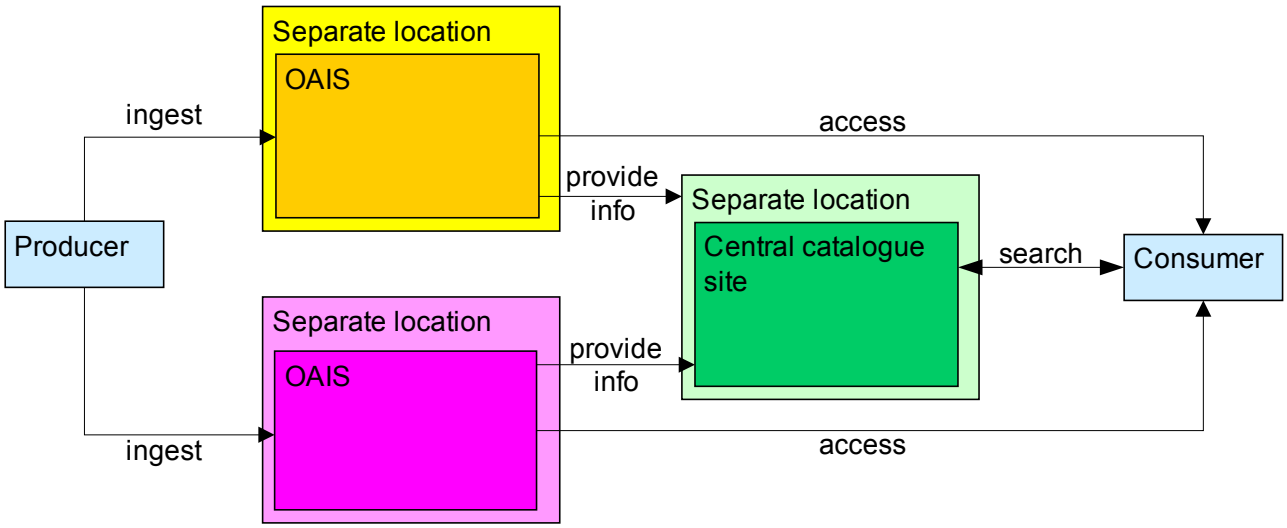


Figure 30 The OAIS "central site" federated architecture. Searching is performed in a central consolidated catalogue and access is direct to the federated sites.

The "distributed finding aid" architecture, shown in Figure 31, is often used in university library systems operating MetaLib (ExLibris) or WebFeat (Serials Solutions). The search can be quite slow as the query is distributed to many locations via a variety of APIs including Z39-50, SRU/SRW[43] or even screen scraping.

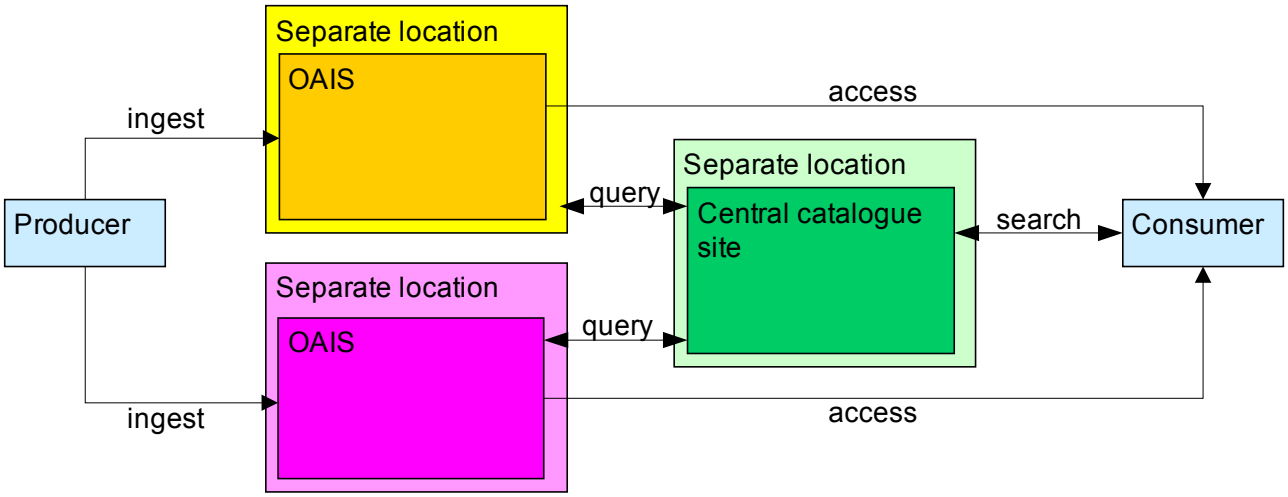


Figure 31 The OAIS "distributed finding aid" federation architecture. Queries are passed on to the federated site by the central site.

Finally, the "distributed access aid" architecture is shown in ies are passed on to the federated site by the central site..

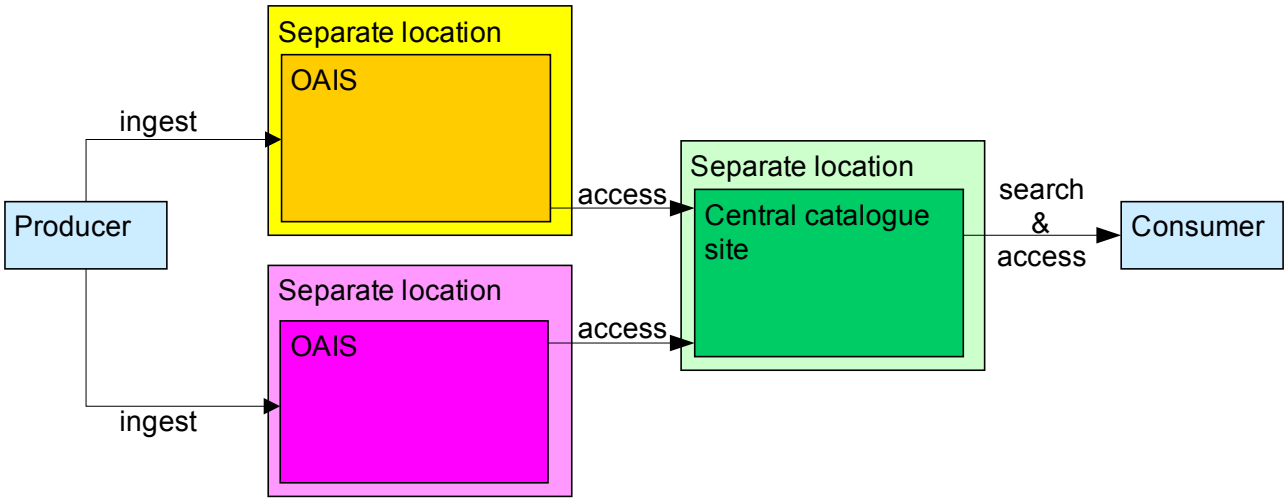


Figure 32 The OAIS "distributed access aid" federation architecture where access goes through the central site.

The Europeana initiative is a prime example of federation between archives. Europeana aims to be a common multilingual access point to Europe's distributed digital cultural heritage. It provides a central site for consumers to search across European archives. This is achieved by having a centrally maintained catalogue ("central site" above) which has the advantage of facilitating a fast complex semantic search. In addition, Europeana provides "surrogate" objects (e.g. thumbnails) and directs the consumer to the original binary objects via a link to the content provider site.

5.3. Conclusions

Many storage systems, especially online storage systems, have ready-made, purchased hardware and software. This is a reflection of the maturity of the market for large storage systems and the difficulty of building such a system from scratch. Also we have found that both the storage system and location are more likely to be managed internally than by a contracted service provider.

Offline and master storage facilities are most likely to following the pattern of tapes on shelves with a purchased tracking system. Both the storage system and the location are commonly managed by the organisation rather than outsourced. After all, the master quality data is the most valuable and also the least accessed so this is a sensible solution.

With fast dedicated network connections now commonly available and the increasing maturity of the market for large storage we can see that outsourcing the storage of online data will become more common and that organisations may move towards using multiple providers for increased security.

Finally, in today's networked world, each archive must recognize that it is not working alone and that value can be added to its content by cooperation and federation with other similar archives in a variety of ways, as described in OAIS.

6. Data to be Stored

Whilst it might be argued that a storage service should be ambivalent to what type of data is stored and accessed, the type of data does have an influence on the design of the service. PrestoPRIME is focussed on systems for audio-visual data where the most obvious characteristic is “many large files”. This is quantified in more detail below, but another aspect to note is the high data rate both in to the archive through new content being ingested and migration programmes converting and ingesting old content, and the quantity of data coming out of an archive for use in new productions. Finally, this chapter touches on what an OAIS preservation system should do with ingested data.

6.1. Audio going in

Not all files are equal. In general, audiovisual files are large. Uncompressed stereo audio at CD quality (1.4 Mbits/sec) uses 650 Mbytes per hour. CD quality is 44.1 K samples per second. Broadcasting usually uses 48k, for slightly larger files. Highest quality digitisation, with 24-bit data and 96K sampling would be three times larger -- roughly 3 GB per hour. While popular music is often only a few minutes in duration, broadcast archives tend to have content of, on average, 20 to 30 minutes duration. Many formats support recording times of up to two hours (from analogue audio cassettes to Digibeta videotape). The Broadcast Wave Format variant of the wave file is the widely-used standard for audio digital preservation[44], in broadcasting and across the whole range of professional audio.

In the 1990’s, and continuing to the present day to a limited degree, a standard production format for audio in broadcasting was “MPEG2” (i.e. MPEG-1 layer 2). Files in this coding format were produced in substantial amounts before use of uncompressed audio became standard. The data rates were typically 192 to 384 Kbits/sec. Indeed, the format was so important that the Broadcast Wave Format was actually written to support two codings: uncompressed, and MPEG2. Large amounts of audio in broadcasting was also carried by ISDN, at 64 or 128 Kb/s, though this was a streaming format, not a file format, so most ISDN content would have been transcoded (to MPEG2 or uncompressed) before being saved in a file.

A digital library setting up operation today, and encountering a number of MPEG2 files, would have to decide whether to save them as is, or decode them and save them as uncompressed audio.

6.2. Video going in

Formats

Video files can be much larger than audio files. Very compressed video, as used on the Internet, is roughly the same size as uncompressed audio: about 1 Mb/s used to be the expected rate for video that was roughly as good as video on VHS videotape (example: MPEG-1). With advances in coding, the AVC coders (advanced video coding, also called H.264; a version of MPEG-4[45]) can produce that same quality at about 500 Kbits/sec. A one hour video would then require a file of about 250 Mbytes. Such files can now be easily physically transported, by removable media such as memory sticks and CD-ROMs – but they remain a challenge to computer networks. The spread of broadband Internet (with speeds of 5 Mbits/sec or more) has made it possible to attempt download of files with sizes in the hundreds of megabytes, but problems abound. Downloading 500 MB at 5 Mbits/sec takes 800 seconds (nearly 15 minutes) – sufficiently long for many things to go wrong! Special technology has grown up (bit-torrent, peer-to-peer networks) to improve the likelihood of successful electronic distribution of files measuring hundreds of megabytes.

A consideration in broadcasting is that production quality video has a data rate 100 times higher than the 500 Kbits/sec needed for 'Internet quality' MPEG-4. Typical production formats are DV at 25 Mbits/sec, DVC-PRO 50 at 50 Mbits/sec, and MPEG-2 main level main profile at 50 Mbits/sec. In general, professional broadcast production formats (for standard definition or "SD" video) are in the DV or MPEG-2 family, and have data rates of between 25 and 50 Mbits/sec. This in turn means that a half-hour of production-quality video content would be 6 to 12 GB in size. Sending a 0.5GB file around standard office IP networks can be difficult, and so these production quality files would have greater difficulty, unless special high-bandwidth (or low traffic) networks are available.

"Preservation quality" is anything from production quality upward, but limits at uncompressed video which for standard definition is 200 Mbits/sec, roughly 100 GB per hour.

For high definition (HD), data rates for production quality are in the region of 100 to 150 Mbits/sec, equating to files (for 60-minute durations) of 50 to 75 GB, which would be as awkward to move about as are uncompressed SD files. Uncompressed HD has a data-rate roughly five times higher than SD, meaning 500 GB per hour – and 3D files (depending upon the type of 3D) could double the size or more. At that point, a digital production system or digital library or repository would be working with terabyte file sizes.

Typical file types for video (or video and audio) are MOV (Quicktime; Apple), AVI (Microsoft) and MXF (SMPTE open standard). These are wrappers, and a wide variety of encodings (codecs) can be accommodated in each of these wrappers. MPEG encodings can occur on both MOV and AVI files, but also may exist as .mpg files.

Video files on DVDs are VOB files, a variant of MPEG-2. Overall there are somewhere between 10 and 20 file types that could be called common or standard. One way to estimate what file types matter, is to look at the types supported by standard software. The widely-used consumer editing package AVS Video Editor[46] supports the following file types:

> Read: HD Video (inc. Blu-ray video, AVCHD, MPEG-2 HD and WMV HD), AVI (DivX, Xvid, etc.), DV AVI, MP4 (inc. Sony PSP, Apple iPod and Archos), WMV, 3GP, 3G2, QuickTime (MOV, QT), DVD, VOB, VRO, MPEG-1, 2, 4, TOD, MOD, MPG, DAT, VCD, SVCD, Real Video (RM, RMVB), ASF,

ASX, MJPEG, H.263, H.264, DVR-MS, MKV, OGM, FLV, AMV, MTV, TS, M2TS, M2T, MTS, DPG, NSV, FLI, FLC, CDG.

Write: HD Video (inc. Blu-ray video, AVCHD, MPEG-2 HD and WMV HD), AVI (DivX, Xvid, etc.), MP4 (inc. Sony PSP, Apple iPod and Archos), WMV, 3GP, 3G2, QuickTime (MOV, QT), SWF, FLV, DVD, MPEG-1, 2, 4, MPG, MJPEG, H.263, H.264, Real Video (RM, RMVB).

Typical video formats and data-rates taken from the PrestoSpace wiki[47]:

Compression Type	Datarate, Mb/s	Quality	Comment
No compression	270	Master	Rec. 601, standard def TV
Lossless JPEG2000	Approx 90	Master	Rec. 601, standard def TV
MPEG-1	1.2	VHS	Wide internet use
MPEG-2	5	DVD	Used on DVD and digital TV broadcasting (DVB)
MPEG-4	0.5	VHS	Will replace earlier MPEGs
MPEG-4 AVC	8	HDTV	Will be used on HD DVDs, and possibly on HD TV
DVX	0.5	Near VHS	Wide internet use
Digibeta	80	Near Master	Nearly full quality
DV, DVCAM	25	"Pro-sumer"	Pictures near digibeta quality, quality suffers on repeated decode-encode
DVC-PRO 50	50	Near Master	Pictures near digibeta quality, quality suffers on repeated decode-encode

Table 2 Data Rates and Quality Levels for Digital Video (Standard Definition).

Quantities

Compared to the quantity (terms of bytes) of audio, the quantity of video ingested into a preservation system would be overwhelming. There are two distinct sources of data:

1. data from current production,
2. data from format migration programmes.

Both sources produce considerable quantities of data. For instance, the BBC transmits approximately 1000 hours per week of television (not counting commercial or non-UK channels). For standard definition material this equates to 100 TB/week but for high definition it will be 400 TB/week. The broadcast material is only a small fraction of the total however. Figure 2 which depicts the content lifecycle reminds us that a transmitted programme does not just arrive fully-formed. The quantity of “rushes” or raw material is ten times the final output. If this is stored then the requirement is actually for 1 to 4 PB/week.

A second example is that of INA which holds France’s legal deposit archive for audiovisual content. INA captures and compresses 24 hours of programming per day from both public and private stations which comes to 900000 hours per year. This is of the same scale as the BBC example above.

Similar quantities of data may also be ingested through migration programmes where an archive holding data on at-risk formats moves the data into a modern storage system. The BBC preserves approximately 800 hours per week of video, equating to 80 TB/week which will not increase with the move to HD.

RAI have a large archive stored on Betacam tapes. The data is being migrated to the MXF/D10 format which requires approximately 30 GB/hour. The quantity of video to be migrated is between 750000 and one million hours, or approximately 30 TB.

6.3. Audio coming out

Again, the Broadcast Wave Format (BWF) is standard as both the storage and delivery format. The other major output category consists of 'browse' formats, needed for electronic (networked) archive access. The particular need is for compact formats for Internet access, or for download (as in podcasts) to devices such as MP3. The most common browse or access format is now is MP3, though a great deal of Real Audio remains on websites. The repository can either have browse audio created at time of ingest, and so stored as a 'related version' or proxy or some such – or the browse version can be created from the BWF on demand.

6.4. Video coming out

Formats

The major issue in video is providing browse formats at the required quality, and in the required file format. Because so many formats are in use, many content holders have a need to 'satisfy the customer' in a variety of formats – which in turn means a need to efficiently code access formats on demand (because keeping 10 to 20 different access version online is hardly an attractive proposition).

Uncompressed video can be used to make any needed browse version, but there are more efficient approaches. The EDCine project[48], developing the standards for digital cinema, advocates using a high data-rate version of lossy JPEG2000 – because that format is very efficient at generating lower datarate 'proxies'.

The approach of keeping a version just for efficient computation of access copies is referred to as having a mezzanine format. For EDCine, lossless JPEG2000 or uncompressed video would be the preservation format, but lossy JPEG2000 at a high-data rate is the mezzanine: a slightly lower data-rate than for preservation, but higher than the needed access copies – and selected for maximum efficiency in producing access copies on demand.

Quantity

If a storage service was to support public browsing as well as preservation facilities then recent statistics on the BBC's iPlayer[49] give us an indication of the quantities of data that might need to be served. In October 2009 the iPlayer dealt with 79.3 million requests for TV and radio programmes, transferring 7 PB of data and peaking at 12.5 GB/s. In contrast, the BBC's archive delivers approximately 10000 items per week to programme makers: a much smaller number but still significant.

A third access type is that of providing data to professional consumers. INA runs a public web site (www.ina.fr) which provides 25000 hours of online archive content but also operates a commercial rights licensing arm, Inamédiapro. This commercial operation uses a base of 500000 hours of digitised content and processed over 8000 orders in 2008.

6.5. Storage format and metadata

A workflow for a digital library needs to know what files it has, so it needs a catalogue. For any practical library the catalogue has to be made as automatically as possible, implying stripping identification information out of the input files.

For an OAIS compliant digital preservation repository the requirements are stronger: the input files are supposed to be combined with any required supporting information into a Submission Information Package – and then an ingest process updates the catalogue and creates an Archive Information Package.

The essential steps in a digital preservation workflow are:

1. strip identifying metadata from the input file
2. strip any other metadata from the input file, if it is relevant to other metadata operations of the repository (e.g. converting metadata to a standard format)
3. align all stripped-out metadata elements with some standard naming process
4. pack (write) the results in a standard form

As an example, a digital repository might seek to handle a range of input formats, take out the descriptive, administrative, technical and preservation metadata from all the supported formats, re-label (map) all such metadata using one or more standards (such as MODS[50] for descriptive metadata or PREMIS[51] for preservation metadata) – and finally put the original files plus the new metadata files into a METS[52] wrapper as an AIP.

There is standard software for identifying file types (JHOVE[53] , DROID[54]), and also for stripping metadata (National Library of New Zealand Metadata Extraction Tool[55]). A basic motivation for PrestoPRIME is to extend these concepts – and tools – to audiovisual files. The New Zealand tool supports WAV and MP3 audio files, and four kinds of image file (BMP, GIF, JPEG and TIFF), but has no support for any video formats.

Another issue for the design of a storage service is how to handle the relationship between SIPs, AIPs and DIPs. This issue will not be solved here, but will be resolved by the project later. Does it make more sense for

1. a large number of small SIPs (e.g. shots, rushes etc.) to be aggregated into bigger AIPs (e.g. all the material for a particular TV episode or series),
2. big SIPs broken down into a larger number of small AIPs (e.g. submission is in big 'one off' steps and then the content is taken apart and its pieces stored in separate AIPs to allow easier management or reuse, or
3. neither because the move to 360 degree commissioning removes all notion of programmes/series/broadcasts etc. and what will get archived is a complex set of interrelated objects with no specific purpose?

7. Data Transfer Protocols

7.1. Introduction

Delivery of broadcast content between media organisations using file transfer is becoming an increasingly attractive alternative to the physical movement of tapes, films or disks. However, widespread adoption of file-based delivery, especially over public networks such as the Internet, requires the adoption of fast, secure, reliable and interoperable data transfer protocols. This chapter reviews and summarizes current solutions.

7.2. Summary

The following sections summarise the protocols outlined in this document with respect to various criteria, grouped into their associated layers. The table headings are as follows:

Protocol: protocol name

Layer: which layer the protocol is in (in the protocol stack)

Transport Protocol Used: for application protocols

Security: is the protocol secure? (confidentiality)

Integrity: can the file's integrity be ensured? (what is received is what was sent)

Reliable: is the protocol reliable? (e.g. can it resume?)

Fast: is the protocol fast (not too slow)?

Large Files: can the protocol handle large files?

Manageable: can the protocol (potentially) be manageable? (e.g. bandwidth controls)

Comments: any other comments

Some of the judgements (particularly with respect to “fast”) are more hunches than scientifically based. A preliminary investigation into quantitative measures of transfer protocol speed follows in Chapter 8.

Transport Layer Protocols

Protocol	Security	Integrity	Reliable	Fast	Large Files	Manageable	Comments
TCP	✗	?	✓	✗	-		
UDP	✗	-	✗	✓	-		
DCCP	✗	-	✗	✓?	-		
SCTP	✓	-	✓	?	-		

Protocol	Security	Integrity	Reliable	Fast	Large Files	Manageable	Comments
HighSpeed TCP	✗	-	✓	✓	-		
Scalable TCP	✗	-	✓	✓	-		
FAST TCP	✗	-	✓	✓	-		
XCP	✗	-	✓	✓	-		

Table 3 Transport layer protocol summary.

Application Layer Protocols

Protocol	Transport Protocol Used	Security	Integrity	Reliable	Fast	Large Files	Manageable	Comments
BitTorrent	TCP	✗	✓?	✓	✓	✓		
FTAM	TCP							
FTP	TCP	✗	✓	✓	✗	✓		
FTPS	TCP	✓	✓	✓	✗	✓		
SFTP	TCP	✓	✓	✓	✗	✓	✓	
HFTP	TCP	✗	✓	✓	✗	✓		
HTTP	TCP	(✓)	✗	✗	(✓)	✓?		Basic security, med. fast
HTTPS	TCP	✓	✗	✗	(✓)	✓?		Med. fast
GridFTP	TCP/UDP	✓	✓	✓	✓	✓	✓	
WebDAV	TCP	(✓)	✗?	✗?	(✓)	✓?		Basic security, med. fast
rcp	TCP	✗	✓	✓	✗	✓		
SCP	TCP	✓	✓	✓	✗	✓		
rsync	TCP	✗	✓	✓	(✓)	✓		Med. fast
TFTP	UDP	✗	✓	✓	✗	✗		
FSP	UDP	✗	???	✓?	✓?	✓		
UDT	UDP	✗	✓	✓	✓	✓		
UFTP	UDP	✗	✓	✓	✓	✓		Multicast

Protocol	Transport Protocol Used	Security	Integrity	Reliable	Fast	Large Files	Manage able	Comments
RBUDP	UDP	✘	✓	✓	✓	✓		
SABUL	UDP	✘	✓	✓	✓	✓		
Tsunami	UDP	(✓)	✓	✓	✓	✓		Basic auth
FASP	UDP	✓	✓	✓	✓	✓	✓	
C2	UDP	???	✓	✓	✓	✓	?	
BlazeBand	UDP	?	?	?	✓	✓		
Signiant	UDP	?	?	?	✓	✓		
FileCatalyst	UDP	?	✓?	✓?	✓	✓		

Table 4 Application layer protocol summary.

Management Protocols

Protocol	Data Transfer Protocol Used	Security	Integrity	Reliable	Fast	Large Files	Manage able	Comments
RFT	GridFTP	✓?	✓	✓	✓	✓	✓	
MDP	Various	✓?	✓	✓	✓	✓	✓	

Table 5 Management protocol summary.

7.3. The Internet Protocol Suite

In order to evaluate data transfer protocols, with their pros and cons, it is first necessary to understand how they fit into the stack of protocols known as the “Internet Protocol Suite”. The stack is divided into the following layers (from top to bottom):

Application Layer: Most data transfer protocols reside in this layer, which generally includes all process-to-process protocols via an Internet Protocol (IP) network, on top of Transport Layer protocols to establish underlying host-to-host connections. Examples include FTP, HTTP, SSH.

Transport Layer: These protocols are responsible for encapsulating data blocks into data units (e.g. datagrams or segments) suitable for transfer across the network infrastructure to a destination host. Examples include TCP, UDP, DCCP, SCTP.

Internet Layer: These protocols are used to transport data units (e.g. datagrams) from the originating host across network boundaries, if necessary, to the destination host specified

by a network address (IP address) which is defined for this purpose by the Internet Protocol (IP). Examples include IPv4, IPv6, ICMP.

Link Layer: The link is the physical and logical network components used to interconnect hosts or nodes in the network and a link protocol is a suite of methods and standards that operate only between adjacent network nodes of a Local Area Network (LAN) segment or a Wide Area Network (WAN) connection. Examples include ARP, Ethernet, DSL, ISDN.

Clearly the Application Layer protocols are most relevant to this chapter, but as these are built upon different Transport Layer protocols, the latter will be discussed in the following section.

7.4. Transport Layer Protocols

The Transport Layer is responsible for delivering data to the appropriate application process on the host computers. This involves statistical multiplexing of data from different application processes, i.e. forming data packets, and adding source and destination port numbers in the header of each Transport Layer data packet. The major protocols are summarised below.

TCP

Transmission Control Protocol (TCP) provides reliable, ordered delivery of a stream of bytes from a program on one computer to another program on another computer, and is commonly used for web, email and file transfer applications. TCP can control message size, the rate at which messages are exchanged, and network traffic congestion. It is optimised for accurate delivery rather than timely delivery, and therefore, TCP sometimes incurs relatively long delays (in the order of seconds) while waiting for out-of-order messages or retransmissions of lost messages.

For data transfer, there are a few key features that set TCP apart from UDP.

- Ordered data transfer - the destination host rearranges according to sequence number
- Retransmission of lost packets - any cumulative stream not acknowledged will be retransmitted
- Discarding duplicate packets
- Error-free data transfer
- Flow control - limits the rate a sender transfers data to guarantee reliable delivery. When the receiving host's buffer fills, then next acknowledgement contains a 0 in the window size, to stop transfer and allow the data in the buffer to be processed.
- Congestion control - sliding window

One big problem is that an application cannot get at the packets coming after a lost packet until the retransmitted copy of the lost packet is received.

UDP

User Datagram Protocol (UDP) is the main alternative to TCP. It uses a simple transmission model without implicit hand-shaking dialogues for guaranteeing reliability, ordering, or data integrity. Thus, UDP provides an “unreliable” service and datagrams may

arrive out of order, appear duplicated, or go missing without notice. This makes it generally more suitable for streaming applications, VoIP, etc. However, certain data transfer protocols do manage to use this protocol whilst guaranteeing reliability and data integrity (e.g. UDT).

TCP vs. UDP

TCP is a connection-oriented protocol, which means that upon communication it requires handshaking to set up end-to-end connection. A connection can be made from client to server, and from then on any data can be sent along that connection.

- **Reliable** – TCP manages message acknowledgement, retransmission and timeout. Many attempts to reliably deliver the message are made. If it gets lost along the way, the server will re-request the lost part. In TCP, there's either no missing data, or, in case of multiple timeouts, the connection is dropped.
- **Ordered** – if two messages are sent along a connection, one after the other, the first message will reach the receiving application first. When data packets arrive in the wrong order, the TCP layer holds the later data until the earlier data can be rearranged and delivered to the application.
- **Heavyweight** – TCP requires three packets just to set up a socket, before any actual data can be sent. It handles connections, reliability and congestion control. It is a large transport protocol designed on top of IP.
- **Streaming** – Data is read as a "stream," with nothing distinguishing where one packet ends and another begins. Packets may be split or merged into bigger or smaller data streams arbitrarily.

UDP is a simpler message-based connectionless protocol. In connectionless protocols, there is no effort made to set up a dedicated end-to-end connection. Communication is achieved by transmitting information in one direction, from source to destination without checking to see if the destination is still there, or if it is prepared to receive the information.

- **Unreliable** – When a message is sent, it cannot be known if it will reach its destination; it could get lost along the way. There is no concept of acknowledgement, retransmission and timeout.
- **Not ordered** – If two messages are sent to the same recipient, the order in which they arrive cannot be predicted.
- **Lightweight** – There is no ordering of messages, no tracking connections, etc. It is a small transport layer designed on top of IP.
- **Datagrams** – Packets are sent individually and are checked for integrity only if they arrive. Packets have definite boundaries which are honoured upon receipt, meaning a read operation at the receiver socket will yield an entire message as it was originally sent.

DCCP

Datagram Congestion Control Protocol (DCCP) is a message-oriented transport layer protocol which implements reliable connection setup, teardown, explicit congestion notification (ECN)[56], congestion control, and feature negotiation. DCCP is a fairly new protocol, and is mainly targeted at applications such as streaming media, Internet telephony, etc, where getting new messages is preferred to resending lost messages.

SCTP

SCTP is another transport layer protocol that provides reliable stream oriented services not so dissimilar from TCP. It is newer and considerably more complex than TCP so has not yet seen widespread deployment. However, it is especially designed to be used in situations where reliability and near-real-time considerations are important. SCTP has multihoming support, in which one (or both) endpoints of a connection can consist of more than one IP address, enabling transparent fail-over between redundant network paths (and thus better reliability). It has improved security over TCP, including a 4-way handshake.

TCP variants

TCP variants intend to take the place of the current standard TCP. They generally have the disadvantage that, in order to install them, the O/S kernel needs to be patched/rebuilt. Several of the latest TCP variants are listed in the following sections.

HighSpeed TCP

HighSpeed TCP (HSTCP) is a new congestion control algorithm protocol defined in RFC 3649 for TCP. Standard TCP performs poorly in networks with a large bandwidth delay product (BDP). It is unable to fully utilize available bandwidth. HSTCP makes minor modifications to standard TCP's congestion control mechanism to overcome this limitation. Results show a speed-up of 20 – 50% over standard TCP[57]. HSTCP has no authentication mechanisms.

Scalable TCP

The main goal of Scalable TCP is to improve the loss recovery time of the standard TCP. The idea is built on the idea of HighSpeed TCP.

Packet loss recovery times for a traditional TCP connection (as well as HighSpeed TCP connection) are proportional to the connection's window size and Round-Trip Time (RTT) whereas a Scalable TCP connection's packet loss recovery times are proportional to connection's RTT only. The developers claim that the Scalable TCP is "TCP friendly", i.e. it does not significantly affect other standard TCP flows. Results show an improvement in throughput of 80% over standard TCP116. Scalable TCP has no authentication mechanisms.

FAST TCP

FAST TCP aims to adjust a source's sending rate so that a link resource is shared fairly by all TCP connections and congestion is avoided with maximum link utilisation. Fast TCP totally discards fundamental mechanisms in TCP such as slow start, AIMD and congestion avoidance. Instead, its objective is achieved by implementing two control mechanisms. One is implemented at the source to adjust the send rate dynamically, based on a complex equation[b,116] and another one is to obtain a congestion measure based on the aggregate flow rate on a link. FAST TCP is similar to TCP in that 1) FAST TCP uses the same acknowledgement mechanism for reliable delivery; 2) FAST TCP uses a windowing[c,58] mechanism to control the send rate at the source.

[b] The equation for adjusting send rate is obtained by proper parameter assignment and pole-zero placement using Nyquist stability analysis.

[c] TCP uses a sliding window flow control protocol. In each TCP segment, the receiver specifies in the receive window field the amount of additional received data (in bytes) that it is willing to buffer for the connection. The sending host can send only up to that amount of data before it must wait for an acknowledgement and window update from the receiving host.

Results using a gigabit Ethernet card demonstrate throughput of 925 Mbps (95% utilisation) compared to 266 Mbps (27% utilisation) for standard TCP116. FAST TCP has no authentication mechanisms.

XCP

The recently-developed Explicit Control Protocol (XCP) represents a major advance in Internet congestion control[59]. XCP claims to deliver the highest possible application performance over a broad range of network infrastructure, including extremely high speed and very high delay links that are not well served by TCP. In so doing, it achieves maximum link utilisations and wastes no bandwidth due to packet loss. XCP is novel in separating the efficiency and fairness policies of congestion control, enabling routers to quickly make use of available bandwidth while conservatively managing the allocation of bandwidth to flows. XCP is built upon a new principle: carrying per-flow congestion state in packets. XCP packets carry a congestion header through which the sender requests a desired throughput. Routers make a fair per-flow bandwidth allocation without maintaining any per-flow state. Thus, the sender learns of the bottleneck router's allocation in a single round trip.

Unfortunately, no useful implementations of XCP seem to exist, and XCP currently exists as a draft IETF standard.

7.5. Data Transfer (Application Layer) Protocols

The following subsections collect together the most important / current data transfer protocols, based on their underlying transport protocol. At this point, only protocols based on TCP/IP and UDP have been identified (i.e. none based on DCCP, SCTP, etc). Certain protocols have been left out, as they are considered too old or too platform dependent (e.g. Apple Filing Protocol).

TCP/IP based protocols

BitTorrent

BitTorrent is a peer-to-peer[60] (P2P) file sharing[61] protocol[62], which is very widely used for sharing large data files. An initial file provider ("seed") allows other hosts ("peers") to download the file (or parts of the file). These hosts then publish the file themselves, becoming additional seeds. As more seeds are added, possibilities for clients to download data fragments increase exponentially. An advantage of this is that there is greater redundancy in the network against system problems BitTorrent uses many small data requests over TCP (c.f. a single HTTP request).

BitTorrent is built from the ground up to support partial file transfers and has a lot of resilience to transfer errors that ensure that the data is received error-free. However, it is intended for use in an anonymous P2P environment (almost anonymous - IP addresses are known) in which every user who can access the tracker server can access the file. For our purposes, it is likely that a) users will not be using the same file at the same time, so will not be able to benefit from the P2P aspect and b) users will not be accessing the data using the same credentials, thus reducing the number of possible peers for the data. Whilst it may be possible to use BitTorrent in a 1-to-1 scenario, this is not its natural usage and performance under these circumstances is uncertain.

FTAM

File Transfer Access and Management (FTAM) attempted to combine into a single protocol both file transfer, similar in concept to FTP, as well as remote access to open files, similar to NFS. It has not seen wide adoption on the Internet, so is unlikely to be of much interest[63].

FTP

FTP is arguably the best known of all file transfer protocols, and has been a stable specification since 1985. It uses two TCP connections; a control channel and a data channel. The control channel (set up by the client connecting to port 21 on the server) is used to send instructions between the client and server, and for the server to send responses back to the client. The data channel is used for the transfer of data. Depending on the transfer mode, the process of setting up the data stream is different.

In **active mode**, the FTP client starts listening on a dynamic, unprivileged port (>1023), sends this port number to the FTP server via the control stream and waits for a connection from the FTP server. When the FTP server initiates the data connection to the FTP client it binds the source port to port 20 on the FTP server.

In **passive mode**, the FTP server starts listening on a dynamic, unprivileged port (>1023), sends the FTP client the server's IP address and port to connect to (via the control stream) and waits for a connection from the FTP client. In this case, the FTP client binds the source port of the connection (on the server) to another dynamic port on the client.

Active FTP is beneficial to the FTP server admin, but detrimental to the client side admin. The FTP server attempts to make connections to random high ports on the client, which would almost certainly be blocked by a firewall on the client side. Passive FTP is beneficial to the client, but detrimental to the FTP server admin. The client will make both connections to the server, but one of them will be to a random high port, which would almost certainly be blocked by a firewall on the server side.

Luckily, there is somewhat of a compromise. Since administrators running FTP servers will need to make their servers accessible to the greatest number of clients, they will almost certainly need to support passive FTP. The exposure of high level ports on the server can be minimised by specifying a limited port range for the FTP server to use. Thus, everything except for this range of ports can be firewalled on the server side. While this doesn't eliminate all risk to the server, it decreases it tremendously.

FTP is often chosen for file transfer because of its resume functionality, which allows a failed transfer to be restarted from the point at which it failed. However, other protocols include resume mechanisms, including HTTP.

These days, FTP is often considered inadequate for large transfers for various reasons of reliability, security and controllability, not to mention its relatively slow speed. It is a complex protocol, with a number of features that are hardly used, providing malicious exploitation opportunities. Now that Apache and other web servers support large files (greater than 2GB) HTTP is considered a better alternative, especially as HTTPS is widely supported (whereas FTPS is not). See also SFTP.

FTPS

FTPS (also known as FTP Secure and FTP-SSL) is an extension to FTP that adds support for the Transport Layer Security (TLS) and the Secure Sockets Layer (SSL) cryptographic protocols. Client security may be invoked either explicitly or implicitly. In explicit mode, an FTPS client must "explicitly request" security from an FTPS server (e.g. “AUTH TLS” or “AUTH SSH”) and then step-up to a mutually agreed encryption method. If a client does not request security, the FTPS server can either allow the client to continue insecure or refuse/limit the connection. In implicit mode, negotiation is not allowed, and servers expect client challenges with a TLS/SSL ClientHello message.

Because FTP uses a dynamic secondary port (for data channels), many firewalls were designed to snoop FTP protocol control messages in order to determine what secondary data connections they need to allow. However, if the FTP control connection is encrypted using TLS/SSL, the firewall cannot determine the TCP port number of a data connection negotiated between the client and FTP server. Therefore, in many firewalled networks, an FTPS deployment will fail when an unencrypted FTP deployment will work, but this problem can be solved with the use of a limited range of ports for data and configuring the firewall to open these ports.

FTPS is not widely supported, but is included in the FileZilla client.

SFTP (SSH File Transfer Protocol)

SSH (or Secure) File Transfer Protocol (SFTP) provides file transfer and manipulation functionality over any reliable data stream. Compared to the older SCP protocol, which allows only file transfers, the SFTP protocol allows for a range of operations on remote files, e.g. resuming interrupted transfers, directory listings, and remote file removal. It is more platform-independent than SCP, being commonly available on most platforms. SFTP is not FTP run over SSH, but rather a new protocol designed from the ground up by the IETF SECSH working group. As SFTP does not provide authentication and security itself, it can be used on top of various security protocols, e.g. SSH-1, SSH-2.

The protocol is not yet an Internet standard, however there are various implementations, including OpenSSH and Microsoft.

HFTP

HFTP is a protocol for accessing FTP resources via an HTTP proxy. It uses the `ftp` URL scheme in HTTP requests to a proxy, and is mainly used by the UNIX based Lftp program.

HTTP(S)

Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems. It is the over-abused cure-all protocol of the Internet, guaranteed to work through firewalls and used by the vast majority of desktop applications. It's a cleartext protocol, but it can be secured using HTTPS (more properly HTTP/TLS).

There are some nice features in the HTTP protocol. First of all, support for partial GET requests mean that the client can request a byte range of a particular resource, rather than having to download the entire resource. This has great potential for transfer from an archive/storage system, as it means that clients can download only the data they want,

and, potentially, jump to any point in a file quickly. Secondly, HTTP includes some rudimentary support for security, in the form of the `WWW-Authenticate` and `Authorization` headers.

As standard, HTTP clients and servers include support for two authorisation schemes, BASIC and DIGEST. BASIC is the usual username/password challenge that can be applied to a particular 'realm' on a Web server, and has been standard since HTTP/1.0. DIGEST is essentially the same as BASIC, with the added feature that the digest (hash) of the password is sent, rather than the plaintext. In theory any digest algorithm can be used, but in practice MD5 is commonplace.

Support for HTTP is very good. There are many server and client implementations out there, including API libraries such as those in Java. What's more, server-side scripting makes it very easy to modify and customise the server side behaviour, far more so than with FTP or BitTorrent.

GridFTP

GridFTP is a high-performance, secure, reliable data transfer protocol optimised for high-bandwidth wide-area networks, and is distributed as part of the Globus Toolkit. It is based upon the FTP protocol, and implements extensions for high-performance operation that were either already specified in the FTP specification but not commonly implemented or that were proposed as extensions by the Globus team. The current GridFTP protocol specification is now a "proposed recommendation" document in the Global Grid Forum (GFD-R-P.020).

GridFTP uses Globus Grid Security Infrastructure (GSI) on both control (command) and data channels. Other features include multiple data channels for parallel transfers, partial file transfers, third-party (direct server-to-server) transfers, reusable data channels, and command pipelining[d]. One interesting development is that GridFTP now provides support for UDT (rather than TCP), and preliminary results show that GridFTP over UDT significantly outperforms GridFTP over TCP on networks where throughput is the bottleneck[64].

GridFTP has been implemented in the BBC's PRISM project[65].

WebDAV

Web-based Distributed Authoring and Versioning (WebDAV), is a set of extensions to the Hypertext Transfer Protocol (HTTP) that allows users to edit and manage files collaboratively on remote WWW servers. Most of the work was put into developing the WebDAV specifications and recommendations in the late 1990s and since that time many other approaches to solving the same and similar problems have developed. WebDAV is an approach to what would now be called 'content management'.

rcp

rcp is the Unix remote copy command, used to copy one or more files from one computer to another, via TCP/IP. However it is insecure for network use, so only its more secure version, scp, is of interest here.

[d] Command pipelining in GridFTP provides efficient sending of multiple outstanding transfer requests, i.e. new requests can be sent before a current one has finished.

SCP

The Secure Copy (SCP) protocol runs on port 22, and is a secure version of the rcp protocol. Similar to SFTP, SCP does not provide authentication and security itself, but relies on the underlying protocol, SSH, to provide these features. SCP has been largely superseded by the more comprehensive SFTP protocol; for example, WinSCP actually defaults to the SFTP protocol.

rsync

rsync is a software application for Unix systems which synchronises files and directories from one location to another while minimizing data transfer using delta encoding when appropriate. It has been designed as a replacement for rcp and scp, and is commonly used for mirroring of data between hosts, e.g. for backup purposes. With a scheduling utility such as cron, one can even schedule automated encrypted rsync-based mirroring between multiple host computers and a central server.

NFS

Network File System (NFS) is a network file system protocol originally developed by Sun Microsystems in 1984, allowing a user on a client computer to access files over a network in a manner similar to how local storage is accessed. NFS, like many other protocols, builds on the Open Network Computing Remote Procedure Call (ONC RPC) system. The Network File System is an open standard defined in RFCs, allowing anyone to implement the protocol. Originally, NFS operated entirely over UDP, however from v3 onwards, TCP support was added. NFS has also been implemented as a stateless protocol in the past, however NFSv4 introduces a stateful protocol, along with performance improvements, and stronger security. NFSv4.1 adds the Parallel NFS pNFS capability, which enables data access parallelism. NFS performance is closely related to RPC performance. Since RPC is a request-reply protocol, it exhibits very poor performance over wide area networks. NFS performs best on fast LANs.

pNFS

pNFS is an extension to version 4.1 of the NFS protocol, which enables data access parallelism. The NFSv4.1 protocol defines a method of separating the filesystem metadata from the location of the file data; it goes beyond the simple name/data separation by striping the data amongst a set of data servers. This is different from the traditional NFS server which holds the names of files and their data under the single umbrella of the server. There exist products which are multi-node NFS servers, but the participation of the client in separation of metadata and data is limited. The NFSv4.1 client can be enabled to be a direct participant in the exact location of file data and avoid solitary interaction with the single NFS server when moving data.

CIFS / SMB

The Common Internet File System or CIFS was first implemented by Microsoft in 1996 as part of Windows NT 4.0. It was an evolution of the earlier Server Message Block or SMB protocol first used by Microsoft in 1987 and is often still known as “SMB”. It is the standard network file system protocol used by Microsoft operating systems but also has a free implementation for Linux (and others) called Samba as well as other commercial and free implementations. An important limitation of CIFS is that is it a very “chatty” protocol. That is, many messages are sent to and fro to accomplish even common tasks. This has the

effect of it being a very poor protocol on high latency networks. The wide availability of clients and expertise however makes it an appealing protocol nonetheless.

With Windows Vista in 2006, Microsoft introduced a new version of the protocol, “SMB2”, which reduced the number of commands from over a hundred to just nineteen. It also has added command pipelining and compound actions within single requests, all of which improves its performance on high latency networks. SMB2 is also supported by the SAMBA implementation.

FCoE

Short for Fibre Channel over Ethernet, FCoE is a standard for using the Fibre Channel (FC) protocol over Ethernet networks. FCoE enables SAN traffic to be natively transported over Ethernet networks, while protecting and extending the investment enterprises have made in storage networks. FCoE uses Ethernet cards, cables and switches to route Fibre Channel traffic at the link layer, and uses Ethernet to transmit the FC protocol. FCoE basically would enable organisations and enterprises to continue to run Fibre Channel over the same wires as their data networks. The goal of FCoE is to reduce management complexity, reduce time to deployment, lower capital and operating costs and lower power utilisation.

iSCSI

iSCSI is an IP-based storage networking standard for linking data storage facilities. By carrying SCSI commands over IP networks, iSCSI is used to facilitate data transfers over intranets and to manage storage over long distances. In essence, iSCSI simply allows two hosts to negotiate and then exchange SCSI commands using IP networks. By doing this iSCSI takes a popular high-performance local storage bus and emulates it over wide-area networks, creating a storage area network (SAN). Unlike some SAN protocols, iSCSI requires no dedicated cabling; it can be run over existing switching and IP infrastructure. However, the performance of an iSCSI SAN deployment can be severely degraded if not operated on a dedicated network or subnet (LAN or VLAN). As a result, iSCSI is often seen as a low-cost alternative to Fibre Channel, which requires dedicated infrastructure.

UDP based protocols

TFTP

Trivial File Transfer Protocol (TFTP) has the functionality of a very basic form of File Transfer Protocol (FTP), and uses UDP on port 69 unlike FTP which uses TCP on port 21. Due to its simple design, TFTP can be implemented in a very small amount of memory, and has therefore been used for booting devices such as routers. Data transfers are performed in “lock-step”, with only one packet (either a block of data, or an 'acknowledgement') ever in flight on the network at any time. Due to this lack of windowing, TFTP provides low throughput over high latency links. Furthermore, it is dangerous to use across the Internet, as it does not provide any authentication or encryption mechanisms.

FSP

File Service Protocol (FSP) is a UDP-based replacement for the File Transfer Protocol, designed for anonymous access with lower hardware and network requirements than FTP. In particular, because it uses UDP, it avoids the problems that many FTP servers have had with requiring a separate process for each client, and because it is built to use an unreliable protocol, it can more easily handle resuming a transfer after a network failure. However, the FSP protocol is not officially recognized by IANA, and therefore has no official port number. Due to lack of support for FSP in web browsers, and poor security, it has largely been superseded by HTTP(S).

UDT

UDP-based Data Transfer (UDT) is a reliable UDP-based application level data transport protocol for distributed data intensive applications over wide area high-speed networks[66], and is a successor to the SABUL protocol. UDT uses UDP to transfer bulk data with its own reliability control and congestion control mechanisms. The new protocol can transfer data at a much higher speed than TCP does. UDT is also a highly configurable framework that can accommodate various congestion control algorithms. Some key features are listed below:

Fast. UDT is designed for extremely high speed networks and it has been used to support global data transfer of terabyte sized data sets.

Fair and Friendly. Concurrent UDT flows can share the available bandwidth fairly, while UDT also leaves enough bandwidth for TCP.

Easy to Use. UDT resides completely at the application level. Users can simply download the software and start to use it. No kernel reconfiguration is needed. In addition, UDT's API is very similar to the traditional socket API so that existing applications can be easily modified.

Highly Configurable. UDT supports user-defined congestion control algorithms with a simple configuration. Users may also modify UDT to suit various situations. This feature can also be used by students and researchers to investigate new control algorithms.

Firewall Friendly. UDT is completely based on UDP, which makes it easier to traverse the firewall. In addition, multiple UDT flows can share a single UDP port, thus a firewall can open only one UDP port for all UDT connections. UDT also supports rendezvous connection setup.

UDT won the bandwidth challenge award at Supercomputing 2008, where it was demonstrated supporting several Cloud applications[67]. It is already being used within several commercial software products[68], for example Movie2Me, a tapeless solution for worldwide movie distribution[69].

UFTP

UFTP[70] is a multicast file transfer program, using a protocol based on Starburst MFTP. It is designed to reliably and efficiently transfer files to multiple receivers simultaneously, where either the intended receivers can be specified beforehand, or receivers can join the transfer when it is initiated. This is useful for distributing large files to a large number of receivers, and is especially useful for data distribution over a satellite link (with two way

communication), where the inherent delay makes any TCP based communication terribly inefficient.

The protocol works in 3 phases:

1. Announce/Register phase: server broadcasts Announcement about file to be sent over a public multicast address (to specific hosts, if required). Clients return a Registration over a private address

2. Transfer/NAK phase: data packets sent in passes. Packets are numbered (as UDP does not guarantee order). After each pass, the client sends back the list of NAKs (negative acknowledgements) for each packet that was not received in that particular section. The server will continue with subsequent passes of the data until all clients have either received the file or have timed out while the server was waiting for a Status message.

3. Completion/Confirmation phase: when a client has received the entire file, it sends a special flag in the Status message for the last section alerting the server that it has finished. Server sends a confirmation to the client.

One study has shown UFTP to outperform FTP by 20 times[71], and client software is available for Windows and UNIX.

Reliable Blast UDP (RBUDP)

RBUDP is a very aggressive protocol designed for dedicated or QoS-enabled high bandwidth networks[72]. It eliminates TCP's slow-start and congestion control mechanisms, and aggregates acknowledgements so that the full bandwidth of a link is used for pure data delivery. The protocol works in a similar way to UDT, using UDP for bulk data transfer and TCP for communication messages. A C++ API is available, but RBUDP does not provide any authentication. A bulk transfer rate of 680Mbps over a 1Gbps has been reported116.

SABUL

SABUL (Simple Available Bandwidth Utilization Library) is an application level data transfer protocol for data intensive applications over high bandwidth-delay product networks. SABUL is designed for reliability, high performance, fairness, and stability. This unidirectional protocol uses UDP to transfer data and TCP to send back control messages. A rate-based congestion control that tunes the inter-packet transmission time helps achieve both efficiency and fairness. To remove the fairness bias between flows with different network delays, SABUL adjusts its rate control at uniform intervals, instead of at intervals determined by round trip time. The protocol has demonstrated its efficiency and fairness features in both experiments and practical applications.

SABUL has been superseded by UDT[73]; whilst SABUL uses UDP to transfer data and TCP to transfer control information, UDT uses UDP only for both data and control information.

Tsunami UDP

Tsunami[74] is a high performance, user-space file transfer protocol, designed to transfer files faster in high-speed networks than that appears possible with standard implementations of TCP. Tsunami uses UDP for data transfer and TCP for transferring

control information. The UDP datagram size is negotiated during the connection setup, along with the length of the file to be transferred.

A single thread handles both network and disk activity at the sender side, whereas the receiver employs separate threads for disk and network activities. The receiver periodically makes retransmission requests, and retransmissions have higher priority than normal sends. The receiver periodically updates the error rate to the sender and the sender adjusts its inter-packet delay, based on this value. The receiver sends a complete signal after receiving all the datagrams. Tsunami allows the user to configure parameters such as size of the datagram, the threshold error rate (used to adjust sending rate), size of retransmission queue and acknowledgement interval.

A simple authentication mechanism is used. Upon connection establishment, the server sends a small block of random data to the client. The client xor's this random data with a shared secret, calculates an MD5 checksum, and transmits the result to server. The server performs the same operation and verifies that the results are identical.

Transfer rates of 600 – 850 Mb/s have been reported116.

FASP

FASP is a proprietary transfer technology from Aspera[75], being the basis of several of their file transfer solutions. FASP eliminates the fundamental bottlenecks of conventional file transfer technologies such as FTP, and dramatically speeds transfers over public and private IP networks. Aspera claims that the approach achieves "perfect" throughput efficiency, independent of path latency, and is robust to packet losses. In addition, users have a high level of control over individual transfer rates and bandwidth sharing.

Regardless of the distance or the dynamic conditions of the network, file transfer times can be guaranteed, even through the most unreliable media such as satellite, wireless, and international links. Complete security is built-in, including secure endpoint authentication, on-the-fly data encryption, and integrity verification. FASP also has built-in transfer reporting, for monitoring and billing purposes.

FASP uses standard UDP and achieves reliability in the application layer through a theoretically optimal approach that retransmits precisely the real packet loss on the channel. Transfer speeds are very impressive. For example, on a gigabit Ethernet link with 200ms latency and 2% packet loss, FASP achieved 505 Mb/s, compared to only 551 Kb/s for standard FTP (i.e. 938 times speed-up)[76].

Digital Rapids C2

The Digital Rapids C2[77] media delivery framework streamlines the transfer of media files between multiple distribution and collaboration points. Working seamlessly with other Digital Rapids solutions as part of complete media distribution workflows, C2 combines exceptionally fast transfer speeds and reliability with network mesh topologies, parallel transfers, simultaneous send/receive and receipt verification for outstanding efficiency when distributing large media and files to multiple recipients. C2 is the successor to Copper.

The exact protocol used by C2 is uncertain, though it is likely to be using UDP at the application layer, as it claims "significant speed advantage over TCP/IP-based transfer methods such as FTP".

BlazeBand

Blazeband[78] is a new acceleration technology from KenCast[79] designed to integrate into the company's broad portfolio of content delivery solutions including both the EdgeSpan (hardware) and Fazzt (software) families.

Blazeband utilizes UDP for bulk packet delivery. Blazeband protocols provide bandwidth and congestion control through UDP messages, using intelligent algorithms which distinguish congestion losses from other types of packet losses. These algorithms prevent Blazeband bandwidth from dropping off in the presence of occasional packet loss.

Signiant

The Signiant Media Exchange[80] software supports a variety of protocols, including UDP, TCP, HTTP and FTP, and claims accelerated file transfers.

FileCatalyst

FileCatalyst[81] is a suite of applications for accelerated file transfers. In particular, FileCatalyst Direct features its own patent-pending UDP-based protocol, aimed at transferring large data sets when there is high latency or packet loss over the network. Where UDP is not possible, FileCatalyst can also use Multiple TCP Streaming. Other features that optimise bandwidth utilisation include on-the-fly compression and delta transfers (c.f. rsync).

Tests over a T3, 45Mbps link (with 250ms RTT and 1.5% packet loss) show a 138-fold speed-up compared to standard FTP[82].

7.6. Security Protocols

Whilst certain data transfer protocols (e.g. HTTPS) include security mechanisms, others are inherently insecure, e.g. UDT. However, it is possible to provide security via a separate security layer, which can work in conjunction with the underlying transport protocol. These can operate at various different layers of the protocol stack, from the Internet to the Application Layer. Some of these are discussed in the following sections.

SSL

Secure Sockets Layer (SSL), is a cryptographic protocol that provides security for communications over networks such as the Internet. SSL encrypts the segments of network connections at the Transport Layer end-to-end. The SSL protocol was originally developed by Netscape. Version 1.0 was never publicly released; version 2.0 was released in February 1995 but "contained a number of security flaws which ultimately led to the design of SSL version 3.0", which was released in 1996. SSL has now been superseded by TLS (see below).

TLS

The TLS protocol allows client/server applications to communicate across a network in a way designed to prevent eavesdropping, tampering, and message forgery. TLS provides endpoint authentication and communications confidentiality over the Internet using cryptography. TLS provides RSA security with 1024 and 2048 bit strengths. A well-known use of TLS is to secure HTTP traffic, forming HTTPS. TLS provides several enhancements

over SSL, including protection against Cipher block chaining (CBC) attacks, more flexible hash and signature algorithms, and support for Advanced Encryption Standard (AES).

DTLS

The Datagram Transport Layer Security (DTLS) protocol provides communications privacy for datagram protocols, in an analogous way to TLS (generally running over TCP), and provides similar security guarantees. The datagram semantics of the underlying transport are preserved by the DTLS protocol - the application will not suffer from the delays associated with stream protocols, but will have to deal with packet reordering, loss of datagram and data larger than a datagram packet size. DTLS runs at the application layer which therefore makes it simple to install and use (see comparison with IPsec).

IPsec

IPsec was designed as a generic security mechanism for Internet protocols. Unlike TLS, IPsec is a peer-to-peer protocol. For many years IPsec was expected to be a suitable security protocol for datagram traffic generated by client-server applications. In practice, however, there are a number of problems with using IPsec for securing such traffic. These problems stem directly from IPsec residing at the network layer rather than the session or application layer.

7.7. Data Management Protocols

This section collects together protocols which are not low-level data transfer protocols themselves, but more like data management protocols, for example working in conjunction with other data transfer protocols.

RFT

The Reliable Transfer Service (RFT) is a WSRF based service that provides interfaces for controlling and monitoring third party file transfers using GridFTP servers.

MDP (Media Dispatch Protocol)

Media Dispatch Protocol (MDP) has been developed by the Pro-MPEG forum[83], which is actively working to develop codes of practice for file-based delivery of content. MDP is not a file transfer protocol as such, but facilitates agents to negotiate details of a file transfer, including which protocol to use, along with other requirements such as start time for transfers, file sizes and priorities, and which security mechanisms to use.

MDP agents communicate with each other via HTTP messages (requests and responses), with requirements being encapsulated in a manifest document. Once file transfers have started (using the chosen data transfer protocol), the agents monitor progress and report status via MDP messages.

MDP provides functionality for initiating, supervising, auditing and retrying file transfers. As this is done at a higher abstraction than the underlying data transfer protocol, MDP has the potential to be a useful tool within the PrestoPRIME architecture. It is already being championed by the BBC, via such projects as the Production Gateway[84], and a white paper on standardising media delivery is available[85]. As part of the evaluation of MDP, various data transfer protocols were compared, in terms of their transfer rate versus network latency. In order of slowest to fastest, these were SMB/CIFS < FTP < UFTP <

UDT-4. This strongly suggests UDT as fast, useable data transfer protocol, used in conjunction with MDP.

8. Data Transfer Benchmarking

In PrestoPRIME, we have designed an automated test framework to understand and study the behaviour and performance of various data transfer protocols in terms of their suitability for transferring large media files across different types of networks. This chapter describes the hardware and software setup for the test environment, and the automation framework, which has been created using Python script. The data transfer protocols we have analysed using the framework to date are FTP and GRIDFTP. These two protocols were chosen for the initial analysis as FTP is commonly used to transfer media files today and GRIDFTP was expected to outperform FTP in a simulation of the sort of “long fat pipe” network of importance to archive users and thereby provide an interesting comparison.

First of all, we discuss the test environment, test file, network emulation and automation script in this section. We then present the test results and summarise our observation in section 2. Finally, we finish by describing what we would like to do in the future in Section 8.8.

8.1. Terminology

This chapter unavoidable has to use some terminology that may be unfamiliar to the average reader. The more specialised terms are explained below:

Network latency:

> the time between sending and receiving a packet of data, measured in milliseconds. Also known as “ping time”.

Round trip time (RTT):

> the “there and back” time – twice the latency.

Bandwidth delay product (BDP):

> refers to the product of a data link's capacity (in bits per second) and its end-to-end delay (in seconds). The result, an amount of data measured in bits (or bytes), is equivalent to the maximum amount of data on the network circuit at any given time, i.e. data that has been transmitted but not yet received.

TCP receive window (RWIN):

> the amount of data that a computer can accept without acknowledging the sender. TCP transmits data up to the receiving window size before waiting for acknowledgement of the receipt of the first packet. If at this point the sender has not received acknowledgement for the first packet it sent then it stops and waits and even retransmits the data if the wait is too long.

TCP congestion window

> determines the number of bytes that can be unacknowledged at the sender side at any one time. It would not normally exceed RWIN.

8.2. Test Environment

The test environment is setup in a private testing network with a Netgear Gigabit switch. Two identical testing machines are used as server and client. Following diagram outlines the hardware and network configuration of the test environment.

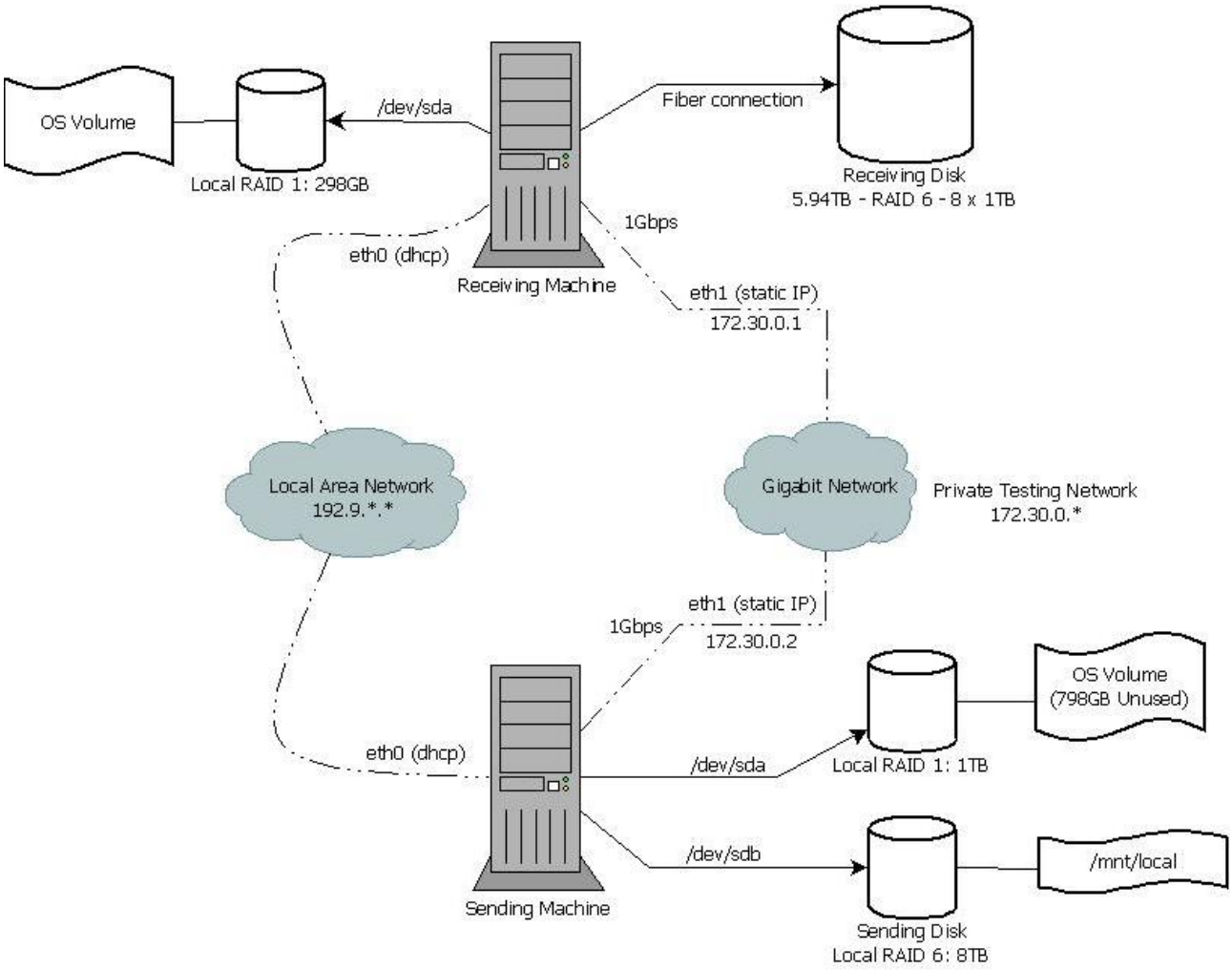


Figure 33 Test Environment Configuration Diagram.

Following table describes the system specification of the testing machines.

Specification	Description
OS	Ubuntu 9.04 Linux 2.6.28
CPU	4 Intel Xeon E5410 2.33GHz 2000MHz
NIC	2 Intel 80003ES2LAN Gigabit Ethernet Controller
PCI Bus	PCI Express Bus
RAM	Psychical: 8059504KB Swap: 7815612KB
HDD	RAID 6 Local: 2.39TB Sumo: 5.94TB
Storage Controller	Intel 631xESB/632xESB/3100 Chipset SATA IDE Controller QLogic ISP2432-based 4Gb Fibre Channel to PCI Express HBA

Table 6 Testing machine specification.

Those two machines are connected to the private network using standard CAT-5 cable. The Gigabit Ethernet chips are connected to the system through PCI Express Bus, so the network cards do not share the PCI bus bandwidth with other system components and can achieve its full operation speed.

Recent versions of Linux (version 2.6.17 and later) have included TCP auto-tuning functionality, which enables the system to automatically scale its TCP receiving and sending windows up to 4MB by default[86]. Thus, the receiver TCP window size can be dynamically updated for each TCP connection to optimise the network throughput and achieve the theoretical maximum transfer rate for a gigabit network of 1000Mbps.

To ensure the disk performance of the test machines does not hold back the speed of a file transfer on the Gigabit network, Bonnie++ (Version 1.03c) benchmark was used. The result shows the sending machine is able to read data at 336545KB/s (2692360Kbps) rate, and the receiving machine is able to write data at 174198KB/s (1393584Kbps) rate. Therefore, they are able to saturate the Gigabit network, and would not hold back the speed of a file transfer on the network.

8.3. Test Files

In PrestoPRIME, the system is expected to handle various media data. In order to determine which data transfer protocol is suitable for the system, a range of test files are selected. These files consist of proper media data in different formats, and random generated data using "/dev/random". The following table shows the detailed information for the selected media file.

Size[e,87]	File Format	Video Format	Audio Format	Bit Rate
12GB	Base Media/Version2 MPEG-4	Advanced Video Codec: Baseline@L5.0	Advanced Audio Codec: Version 4 LC	172 Mbps
18GB	QuickTime MPEG-4	YUV[f]: 2vuy	PCM Audio: sowt	831 Mbps
42GB	QuickTime MPEG-4	Digital Video: ProRes 422 HQ, apch[g]	PCM Audio: sowt	184 Mbps
189GB	QuickTime Original Apple specifications	YUV: 2vuy	PCM Audio: sowt	831 Mbps
411GB	QuickTime Original Apple specifications	YUV: 2vuy	PCM Audio: sowt	856 Mbps

Table 7 Media Files for Testing.

We are most interested in measuring the transfer performance of media files but they do not necessarily come in neat sizes for efficient sampling. We wanted to use files of random data with sizes that are evenly spaced on a logarithmic scale but needed to be certain that the transfer performance of these generated files was the same as the media files. Below is the list of random data files we generated between a data range of 10GB to 500GB. In order to check whether a transfer protocol would have behaved differently with a random

[e] All sizes are reported in powers of 1024 bytes. Media information is extracted using MediaInfo Version 0.7.24, 2009-10-30

[f] A colour space that is not used in digital media, but in analogue PAL-based stuff as analogue TV transmission or analogue video tapes.

[g] 'apch' (hcpa in little-endian) denotes "ProRes 422 High-Quality".

data than a proper media data, 5 pieces of random data were generated with the same size as those media files listed above.

Size/GB	Filename	Size/GB	Filename
10	r_10GB_file	12	r_12GB_file
21.5	r_21.5GB_file	18	r_18GB_file
46.4	r_46.4GB_file	42	r_42GB_file
100	r_100GB_file	189	r_189GB_file
215.4	r_215.4GB_file	411	r_411GB_file
500	r_500GB_file		

Table 8 Random Files for Testing.

Using GRIDFTP (single stream), we have taken samples of transfer rate for each media file and the random data files of the same sizes. By using t-test we have statistically analysed the samples, and we are confident that random data can be used as the test data instead of the proper media files. Therefore the remaining tests were carried out using the random data files with logarithmically spaced sizes shown in the left column of Table 8.

8.4. Test Script and Automation

For running multiple tests, we have automated the test script as much as possible using Python script. The basic test steps are described below:

- Run “Iperf” to measure the native TCP/UDP performance.
- Run “ping” to measure and record the round trip time (RTT).
- On one testing machine, use Linux commands to initialise the data transfer.
- During the data transfer, the transfer rates, total data transferred amount, CPU usage and memory usage are monitored and recorded.
- After the completion of transfer, data integrity is checked by MD5 or SHA-1 checksum.
- Delete all transferred data and repeat the test with the next data transfer protocol.

To automate the above test steps three python scripts are created. The following diagram outlines the function of the scripts. All data transfer protocols under test are configured in auto_test.py. The check_usage.py executes all shell command that is sent by auto_test.py and monitors the data transfer progress. As do_checksum.py checks the integrity of the file after data is transferred.

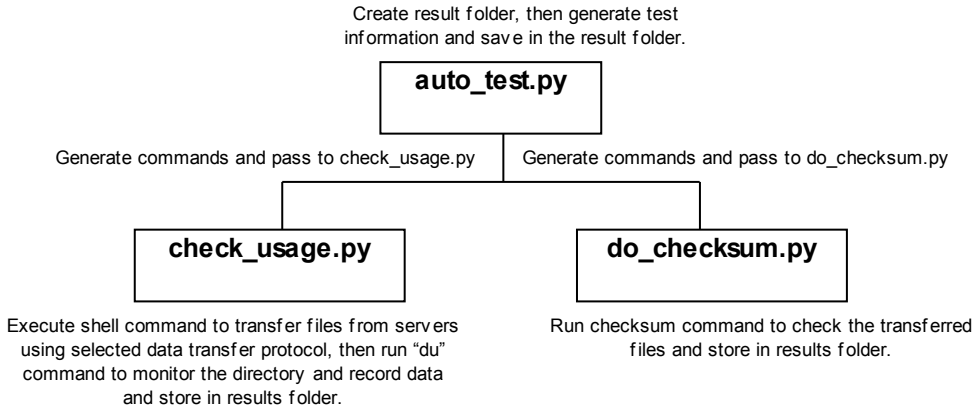


Figure 34 Automation Script Diagram

8.5. Data Transfer Protocols

The survey of data transfer protocols in the previous chapter is extensive. For this preliminary investigation we have decided to concentrate on FTP (which is commonly used today in existing AV archive systems) and GRIDFTP which ticks all the boxes of security, reliability, speed etc in the summary Table 4.

Iperf

Iperf is a network performance measuring tool. It is used to measure the native performance of TCP and UDP. The version of the software we are using is 2.0.4 released on 7 April 2008. Before each test, the native performance of the transfer protocol is recorded.

GRIDFTP

GRIDFTP is part of Globus Toolkit, which is an open software toolkit used for building grids. The protocol itself is based on FTP optimised for high-bandwidth WAN. By applying Globus Grid Security Infrastructure (GSI) on control and data channels, GRIDFTP is able to supply a high-performance, secure and reliable data transfer service. With the latest development, UDT can be used instead of TCP during file transfer, as well as multi-threading the TCP transfers. Therefore, the test framework includes tests for single-thread GRIDFTP, multi-thread GRIDFTP and GRIDFTP with UDT though only multi-threaded GRIDFTP using TCP is reported on here. The version of Globus Toolkit used in these tests is 4.2.1.

Other commonly used protocols

In order to analysis the data gathered by the test framework, some of the standard protocols are included in the test framework, such as FTP and HTTP. Linux command "wget" is used to transfer file from server using those protocols. Both of them are based on TCP therefore, they are expected to perform well on a high-bandwidth and low-latency network.

8.6. Network Emulation

In order to test each protocol under different network conditions, WANem[88] and Netem codes are used. To emulate high performance private or shared WANs, also known as "Long Fat Networks" (LFNs), following Netem code is applied on both test machines.

tc qdisc add dev eth0 root handle 1: netem delay <latency>ms

While those high performance networks offer high bandwidth, they also have a large bandwidth and delay product (BDP). The performance of TCP protocol could be affected by the delay on the network. By tuning the TCP window better performance can be achieved on a high latency network. The maximum TCP window size of our test machines is 4MB, so theoretically they can maintain maximum TCP throughput close to 1Gbps with RTT up to 30ms. Therefore, FTP and GRIDFTP (multi-threaded) are tested under following network conditions:

- 1 Gigabit network without any modification to network settings.
- 1 Gigabit network with added network latency of 5ms, 10ms, 12.5ms, 15ms, 17.5ms, 20ms, 25ms, 30ms, 40ms, 50ms and 60ms (RTT of 10 to 120ms).

Latency depends on the network hardware but for larger distances it is predominantly a function of the speed of light and therefore increases with distance. A metropolitan area network (up to 75km) might have a latency of 10ms and an intercontinental link could be 150ms. At the extreme, communication via geostationary satellite incurs a RTT almost 500ms just taking into account the limit imposed by the speed of light, and is likely to be twice that in practice.

Within the private network a WANem machine can also be setup to emulate sophisticated network conditions such as packet loss, reordering, duplication, corruption and random disconnect. During testing, data transferred between test machines can be routed via this machine. Since it is used to mimic bad network conditions over the internet, it does not require high-quality hardware.

8.7. Results

Data Transfer Protocols behave differently on various network conditions. For example, TCP-based protocols can be very efficient over a local, low latency network. However, over the LFNs, while the latency increases, due to TCP's slow-start property, these protocols could perform progressively badly. Because TCP is a reliable and in-order transfer protocol, it requires every packet it sent over the network to be acknowledged. Slow-start is one of TCP congestion control algorithms. It is used to by TCP to judge the appropriate size for the congestion window. The size of the congestion window starts low and is increased until a data loss is detected. For some data loss events the congestion window will be reduced in size again (perhaps by half) cutting throughput. The more packets sent that are unacknowledged, the more TCP will cut throughput and stop sending further packets over the network. The TCP protocol will slowly try to increase the bandwidth again as acknowledgements are received and perceived network congestion is reduced. Increasing RTT delays the packet acknowledgement, and can cause TCP to believe there is congestion on the network.

The basic TCP window size (RWIN) is limited to 64kB[89]. TCP window size is the amount of packets the sending machine can accept without acknowledgement. The maximum theoretical through put for TCP in bits per second is RWIN (bits) / RTT (s). With an RTT of 1ms a 64kB window size could reduce the network throughput to 500Mbps on a Gigabit network. So it is insufficient for today's LFNs in which the RTT could be well over 10ms[90].

Thus, one method of improving throughput in high latency network is to increase TCP window size to achieve desired throughput. Another method is to use multi-threading and transfer data with multiple TCP streams, this could help reduce the TCP bandwidth recovery time and maintain high TCP throughput[91].

By default, our test machines are able to auto-tune their sending and receiving TCP window size up to 4MB. So by calculation RTT (s) = TCP window size (bits) / throughput (bps), to maintain 1000Mbps of throughput with 4MB window size, the maximum RTT they could cope should be $4194304*8/10^9 = 0.034$ s.

Furthermore, GRIDFTP is able to split a large file into multiple parts and send them simultaneously using multiple TCP streams. The number of parallel connections GRIDFTP could use to transfer data depends on the environment. But, for our benchmarking exercise we decided to use four parallel streams as suggested117. Since we are using multiple TCP streams, the bandwidth is divided across them, but each stream could still be auto-tuned to have 4MB for window size. Thus, by calculation $4*(4194304*8)/10^9 = 0.134$ s, GRIDFTP is theoretically able to maintain throughput close to 1Gbps with 134ms RTT.

Figure 35 shows the actual network throughput of FTP and GRIDFTP on our test environment for different network condition described previously. Each point in Figure 35 is taken from 10 test samples of transferring the 10GB random data file.

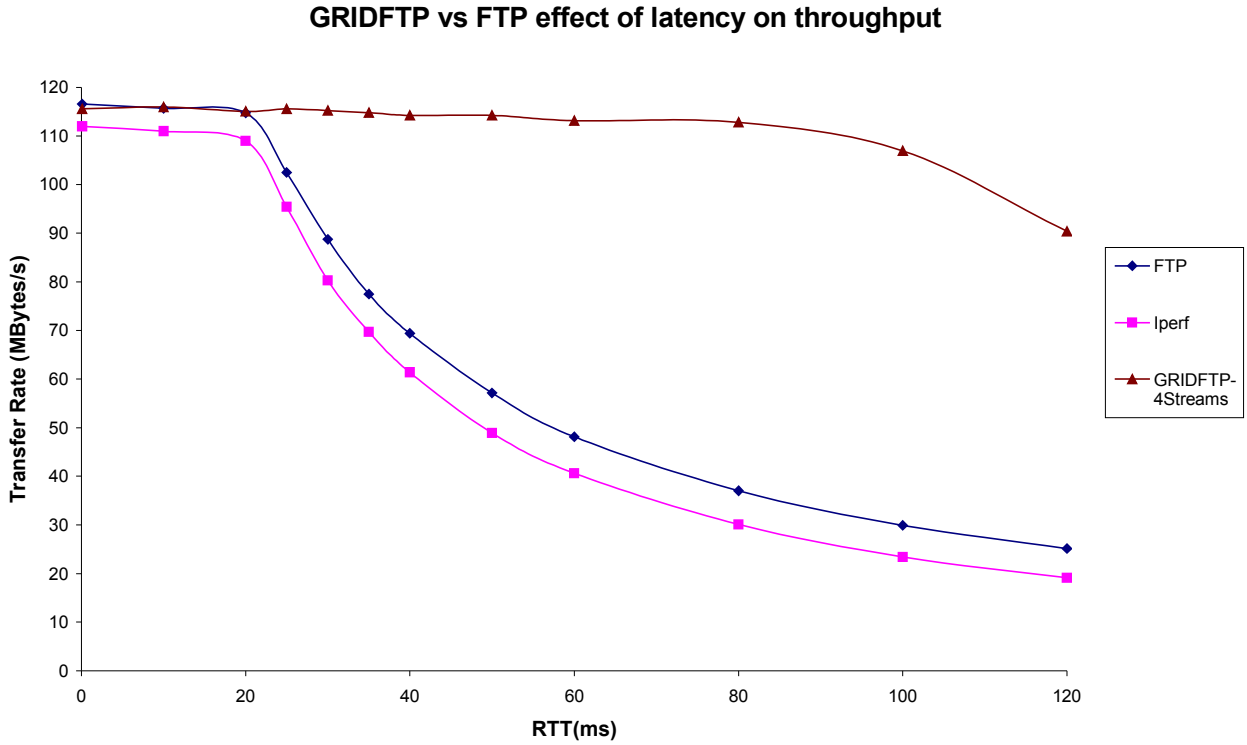


Figure 35 Effect of latency on throughput for FTP and GRIDFTP.

The graph shows that the actual FTP performance was not greatly affected by the increasing of RTT on the network until RTT reaches 20ms, after which the performance tails off rapidly as RTT increases. This is a typical behaviour of FTP on a LFN, as the window size could not hold all unacknowledged packets to fully utilise the available bandwidth117. As expected, GRIDFTP performed much better than FTP, as RTT increased to 80ms, it starts to show some effect of RTT on its performance.

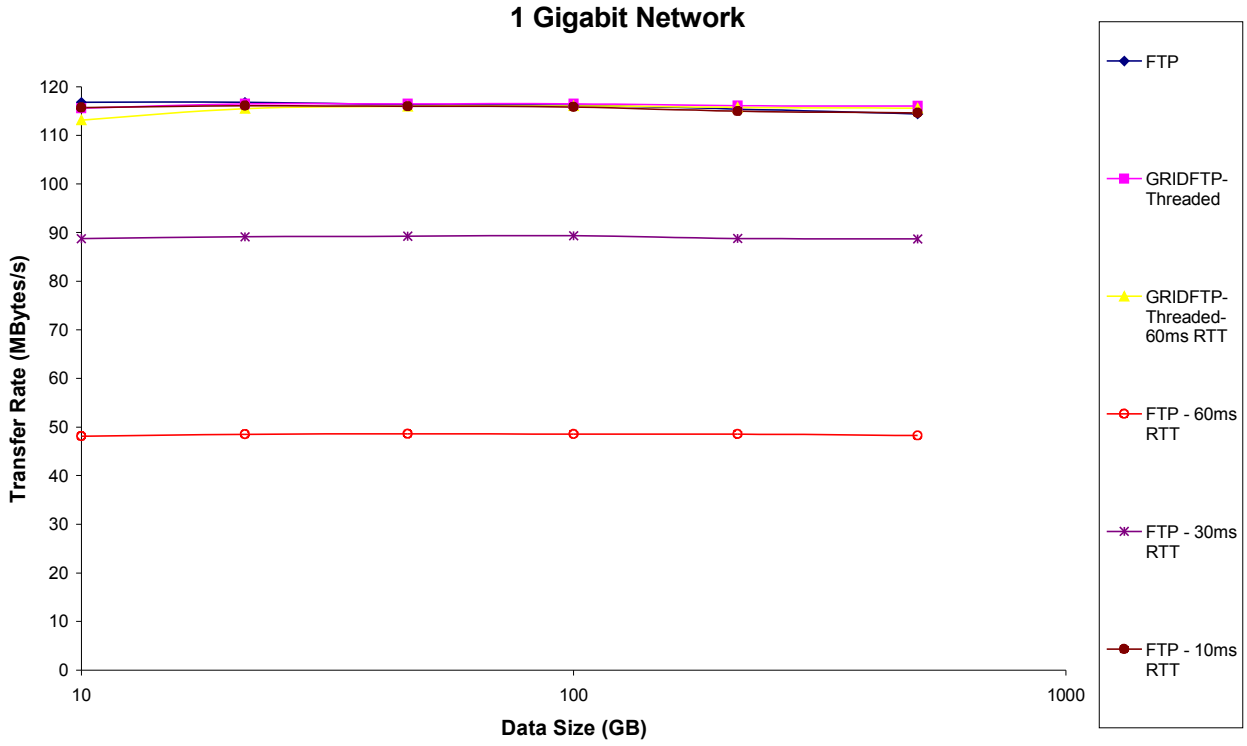


Figure 36 Various file size transferred on the network using FTP and GRIDFTP.

Figure 36 shows the results of various random data files ranged from 10GB to 500GB, which are transferred over the test environment using FTP and GRIDFTP with no added RTT, 10ms, 30ms and 60ms RTT for FTP and 60ms for GRIDFTP. The results clearly show that for sufficiently large data, the transfer speed is unaffected by the data size and that GRIDFTP greatly outperformed FTP on a network that has a high bandwidth-delay product. Therefore, for transferring a single file, GRIDFTP is a much more suitable data transfer protocol than FTP on a high performance private or shared WANs. For networks handling with many simultaneous file transfers we would expect GRIDFTP's advantage to remain, but be reduced as multiple FTP connections utilise the bandwidth more effectively than a single one.

8.8. Future Work

In this preliminary study, we have investigated aspects of FTP and GRIDFTP, and discussed their behaviours on LFNs with a range of latencies. In PrestoPRIME it is necessary to study other protocols such as UDT, HTTP, GRIDFTP-UDT and perhaps commercial offerings such as those from Signiant or Aspera to determine their suitability for the project. UDT removes the TCP window constriction by using UDP to transfer bulk data instead of TCP but adding its own reliability and congestion control mechanisms. It can transfer data at a much higher speed than TCP (un-tuned) does[92,93]. As it is completed based on UDP, UDT is firewall friendly. In addition, UDT won the bandwidth challenge award at Supercomputing 2008, where it was demonstrated supporting several Cloud

application[94]. Apart from testing it against GRIDFTP, we hope to test the protocol without any configuration. It is expected to perform better on a high-bandwidth and high-latency network.

Furthermore we would like to test these protocols under the more sophisticated network conditions that WANem offers such as: bandwidth limitation, packet loss, reordering, random disconnect, data duplication and corruption. As well as this we would like to analyse the memory and CPU usages of the test machines for those protocols during data transfer. Then we will compare and study the results to help us to reveal the most efficient data transfer protocol for PrestoPRIME.

9. Security

In this chapter we explore the different approaches to user identification, authentication and authorisation as well as a standard architecture for web service security. This chapter does not discuss the use of digital rights management (DRM) for file security which is looked at elsewhere in the PrestoPRIME project.

Security focuses on risks to the assets of a business. The object of security is to reduce the risk of specific undesirable outcomes to assets as a result of an attack. The word "attack" does not just mean the sort of actions a hacker would take; the archive's assets must be secured against human error (or incompetence) and against deliberate sabotage.

Information security management standards (e.g. ISO/IEC 27001:2005[95]) specify how to establish a corporate governance framework for security management. They require a continuous quality cycle in which security is enacted, measured, revised and updated. The standardisation process currently underway for the TRAC trusted digital repository scheme aims to include ISO/IEC 27001:2005 where appropriate (see PrestoPRIME D3.1).

A storage service for preservation purposes will need to restrict access to both actions in general and actions on data. For instance, to give some examples:

- Only certain people should be able to initiate the ingestion of new data: we are not building a system like YouTube where anyone can upload content.
- The action of deleting archived data should be available either to no-one or only via a very restricted process to certain trusted people.
- Modifying existing metadata may be a more restricted action than adding new metadata.
- Access to archived content (certainly the exploitation or master quality content) will be restricted to certain people.
- Access to management reports on the system's performance should be restricted.

This simple list already demonstrates that in security systems the policy decision to allow or deny depends on various pieces of *context.* The first and last items on the list should be available to people who have a certain *role* ("archivist" and "system manager" perhaps) and the middle three items also have the context of a certain piece of data (e.g. "this file can be accessed by all but this other file is restricted"). There is much more information on this topic in the related PrestoPRIME deliverable D3.2.1114 .

9.1. Identifying the User

A simple access control system works by first identifying the user and then looking up in a policy whether they are permitted to perform the action requested. We will see later that actually explicitly identifying the user is not always necessary but it is a useful place to start. Formally, a user (or *entity*) has one or more *identities* each of which consists of one or more *attributes* or *identifiers* some of which are shared and some of which are unique. For instance, you might have two online accounts for different banks: these are two identities. You may then have a different username and password for each account

(different identifiers) but both systems could also request your mother's maiden name (a shared attribute).

Identifying the user is commonly called "authentication" (or "AuthN") and in computer systems there are many ways to do this such as:

1. username and password
2. X.509 certificate (a public key infrastructure standard)
3. biometrics
4. tokens

Many online systems just require a username and password, but if the likelihood of someone breaking into another person's account and the resulting threat to the assets in the system is judged to be high enough then two-factor (or multi-factor) authentication methods are used. A two-factor authentication method tests two of three classes of information:

1. something the user *knows* (e.g. a password, PIN, mother's maiden name)
2. something the user *has* (e.g. an ATM card or smart card)
3. something the user *is* (e.g. a fingerprint or iris scan)

The most common form of two-factor authentication is invoked when withdrawing money from an ATM: to do this you need to *know* the PIN and *have* the cash card. For online banking systems, cards displaying a frequently updated pseudo-random number that must be entered into the web-page are becoming more common.

In a preservation system, it may be that two-factor authentication would be required if a potentially damaging (high risk) action such as a system configuration change or deletion of data was being attempted.

A potential problem with security systems is that if the system is made too hard for a user to access then the user will try to subvert the security in some way and overall the security of the system is reduced. For instance, if the security policy forces a user to change their password once a week and ensures that the password is a strong one (i.e. not a dictionary word, one never used before, including numeric and punctuation characters) then so called "password fatigue" can result and the chances are that the user will write the password down and attach it to their monitor, completely defeating the high security objective.

To ease the difficulty of both users and system managers in dealing with multiple passwords, identity federation systems have been developed. These are of two distinct forms: single sign on (SSO) systems and systems allowing a single set of credentials to be used for multiple services. SSO systems are important in corporate environments where for instance you log in to your Windows PC and you are automatically authenticated by the mail server and file server. This is commonly achieved using Kerberos for Linux systems or Windows Active Directory (which uses Kerberos under the hood). In decentralised scenarios involving more than one organisation, systems are needed to permit the system the user is logging into (the "relying party") to trust assertions made by a separate identity

provider (such as the user's home system authentication service) so that the relying party can effectively say "well, I can't authenticate this user myself but I trust a certain relying party to authenticate them properly". Note that these federation systems are independent of the authentication system.

For web services the WS-Federation specification was developed for this purpose. WS-Federation is used in several grid middleware solutions including GRIA[96] and an adaptation of it (WS-Federation Passive Requestor Profile[97]) is used to federate Microsoft systems such as SharePoint. In the academic world, identity federation of this sort is done using Shibboleth[98] which has been developed as part of the "Internet2" initiative. Modern web applications increasingly use another alternative called OpenID[99]. Microsoft have a similar identity selector technology called CardSpace and IBM and Novell have an alternative called Higgins.

9.2. Identifying the Service

As well as the service provider needing to identify its users, the user often wants to identify the service provider in a secure way. The user should be careful to identify the service provider to avoid "phishing" attacks where a rogue site/service masquerades as a genuine one in order to harvest the users' login credentials.

Secure web sites and web services provide server authentication using transport layer security[100] (TLS) which is also commonly used for securing the communications channel against eavesdropping by way of encryption. TLS typically uses public key infrastructure (PKI) and X.509 certificates to enable the client to verify the server's identity. During the initialisation of the client/server connection (the "handshake") message authentication codes (HMAC) are exchanged. The HMACs are generated using the service's private key and the client can verify the authenticity of the codes because it has either previously had the (independently verified) server's certificate installed into its local trusted list or knows to trust the certificate authority that issued the server's certificate.

In online banking, the login page for a bank account will appear in a web browser with a padlock symbol to show that the web browser trusts the web site. The bank will have purchased (from Verisign for instance) a certificate stating who they are the web browser has a long list of trusted authorities (including Verisign) to check against.

TLS can be used to identify the client as well as the service. For this, the client must have a person X.509 certificate issued and installed in their browser (or other client) and the service must trust the client's certificate authority. There are many more users than service providers, and whilst users can manage with a small list of trusted authorities used by major service providers it is harder in general for a service to maintain a list of trusted users in this way.

9.3. Securing the Channel

If data being placed into or retrieved from a storage system is not in the public domain then there is value in keeping that data private. For this reason it is often prudent (if not legally mandated) that the communications channel over which the data is sent is secure from eavesdropping.

It has already been mentioned that TLS can be used to secure the communication channel between a client and service. TLS is not the only way to do this and other methods have been discussed above in Chapter 7 and particularly in Section 7.6 on security protocols.

9.4. Access Control Lists

Once the user is authenticated, the server trusted and the communications channel (encrypted or not) established, the user proceeds to request an operation such as ingesting a file, accessing a file or searching. The system must then make an authorisation (or AuthZ) decision. The following three figures show progressively more advanced ways of dealing with access control. For these examples we are ignoring the additional contexts of e.g. exactly what file is being accessed or what collection is being searched.

The most basic approach to access control lists is shown in Figure 37. Here the entire policy is stored at the service and stated purely in terms of the user identities (e.g. “Alice” and “Bob”). It is immediately apparent that (1) the example is rather contrived as Alice the archivist is not permitted to access the content and (2) if there were many operations and many users then the policy would become unwieldy.

Figure 37 A basic server-side access control system.

A more advanced system is depicted in Figure 38. In this case there are two policies, the first one defining user groups and the second stating what members of the groups can do. The two groups are called “Archivists” (containing just Alice) and the “Users” containing the Archivists plus Bob. So, now Alice can access data and also these user groups can be used many times to control access to many operations and objects in the system. This pattern of user and group authorisations is found in many file systems.

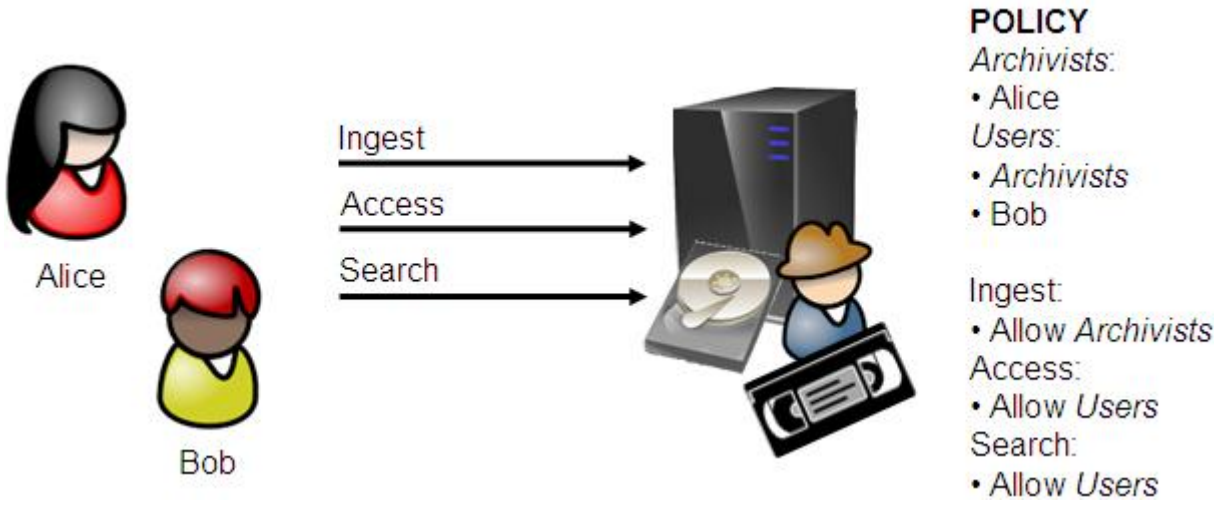


Figure 38 A server-side access control system making use of user groups or roles.

Finally Figure 39 shows a system where the parts of the policy have been distributed. The service provider maintains the policy stating what role a user must have in order to invoke an operation but the decision on who has what role has been delegated to the client organisation. This makes practical sense for two reasons: firstly it is the client organisation that knows who should have what role (not the service provider) and secondly the role allocations can then potentially be used at multiple service providers whilst only defining them in a single place.

Figure 39 An advanced access control system where the service delegates.

Technically, this sort of assertion about a user is often made using the security assertion mark-up language[101] (SAML) and the protocol for sharing the assertion is commonly WS-Federation. When the user from the client organisation accesses the service, a SAML token would automatically be retrieved from a local system and attached to the request. The service can then validate the authenticity of the assertion and determine the user's role. In this situation, the service does not actually need to know the user's identity (i.e. whether it is "Alice" or "Bob"), only what role they have. However, it is good practice to also authenticate the user's identity for auditing reasons if nothing else.

A very important operation class of operation that certain members of the client organisation should have access to is the ability to update the access policies at the service. Of course, this cannot be permitted in an uncontrolled fashion or it would become a free-for-all with anyone able to grant themselves any authority. Rather, it must be done

by way of *delegation*. For instance, someone at the client organisation with the "administrator" role on a file may be permitted to grant someone else read access to it but would not be permitted to make someone else an administrator.

9.5. Complex Policies

In addition to simple access control lists such as those discussed above, there are several more advanced access control policy languages that use logic to process policy statements and arrive at an authorisation decision. For instance, SecPAL[102] (a proprietary Microsoft technology) can express statements such as:

- FileStore says JuniorEmployee can read /docs/
- FileStore says SeniorEmployee can act as JuniorEmployee
- FileStore says Bob can act as SeniorEmployee

and understand that to mean that Bob can read /docs/. A system called Delegation Logic[103] is very similar and non-proprietary and implemented using the Prolog language.

Finally, and most well known, the eXtensible Access Control Markup Language[104] (XACML) defines:

- An architecture for a security system, with a PEP (Policy Enforcement Point), PDP (Policy Decision Point), etc.
- An XML language for expressing security policies.
- An XML language for expressing access requests and decisions.

XACML version 2.0 was ratified by OASIS in 2005 and version 3.0 (which adds delegation) is a work in progress but already implemented (in working draft) in commercial software such as Axiomatics.Policy Server[105].

The complexity of the policy definition language required for a PrestoPRIME preservation system will depend on an analysis of the user requirements.

9.6. Web-Service Standards

Web services commonly use the architecture defined by XACML. This is shown in Figure 40 below.

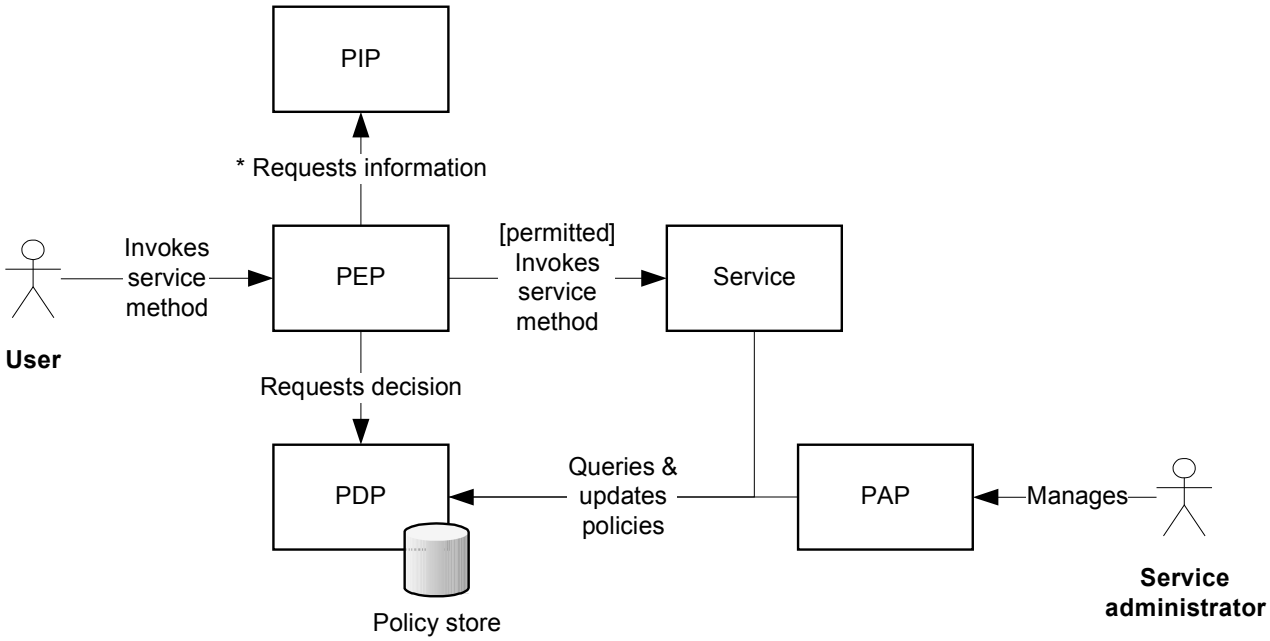


Figure 40 The XACML server-side architecture.

Briefly, the user invokes a method on the service which is intercepted by the policy enforcement point (PEP) often implemented as part of the web service handler chain. The PEP extracts the context from the request (e.g. user attributes, operation, resource) and can retrieve additional information from the policy information point (PIP). All the context is then passed to the policy decision point (PDP) which makes the decision according to the appropriate policy in its policy store (which can be in any policy representation). If the PDP's decision is positive then the PEP passes the request on to the service, otherwise an error is returned to the user. Figure 40 also shows that the service can potentially query and update the policies and the service administrator (at the service provider's site) can manage the policies by way of a policy administration point (PAP).

At the Web Service level, the main trust and security standards published by OASIS and WS-I concentrate on the message level. WS-Security[106] is the base standard, focusing on the integrity and confidentiality of SOAP messages. More recently, the WS Basic Security Profile 1.0[107] was published describing how WS-Security can be used to send secure messages in heterogeneous environments. Declarative configuration of WS-Security settings can be made using WS-SecurityPolicy[108]. Recent work on token exchange has led to WS-Trust[109] and WS-Federation[110] to enable dynamic and federated trust. Encrypted messages based on WS-Security means that each message requires a new session key for encrypting the message. WS-SecureConversation[111] removes this problem, building on WS-Security and WS-Trust to enable secure contexts and key exchange between services. WS-Secure Exchange (WS-SX)[112] is a more recent initiative to further refine WS-SecureConversation, WS-Trust and WS-SecurityPolicy. Recent work, e.g. NextGRID[113], has concentrated on interoperation across heterogeneous security environments, including X.509, SAML and Kerberos token exchange. Notably, edutain@grid[114] has taken a keen interest in the use of WS-Trust and the SAML Token Profile[115] for a light-weight security infrastructure in Real-Time On-Line Interactive Applications (ROIA).

9.7. Conclusions

Service security is part of the risk management process. Proportional security measures should be used to mitigate perceived threats, both malicious and accidental, to a system's assets.

A security system must be designed with the use cases in mind. Aspects to consider include:

- Two-factor authentication for high risk operations.
- Identity federation systems to reduce password fatigue and improve user management.
- Authentication of the service.
- Encrypted connections.
- Having the client organisation maintain role allocations locally.
- The importance of the ability to delegate certain authorities.
- Advanced policies such as those provided by XACML.

10. Interfaces

In this section we analyse the storage and preservation interfaces to be exposed from the PrestoPRIME Preservation Platform.

First, the state of the art in current storage and preservation systems is described, taking into account adopted standards. Then, we compare the available technologies and propose a candidate solution for PrestoPRIME.

Our challenge is to deal with user generated contents as well as digital contents belonging to digital libraries and to broadcasters. The latter have specific requirements due to the huge file size of digitised material and complex metadata structures to be handled.

PrestoPRIME preservation platform will be compliant to the CCSDS OAIS115 standard. Therefore, in the following we adopt the naming conventions and terms provided by OAIS which are suitable for the PrestoPRIME interfaces, such as Submission Information Package (SIP), Dissemination Information Package (DIP) for ingestion and retrieval, Archival Information Package (AIP) for storage and internal management.

10.1. Comparison of Available Technologies and Solutions

In the following table we summarise the aspects of projects analysed in the state of the art chapter (Section 4.1) which are relevant to a possible reuse of software.

We try to answer, where possible, the following questions:

- Which standards and third party technologies are used?
- Under which license is the software released?
- What are the start and end dates of the project?
- Is the software maintained?
- What language is the software written in?
- Is the software well documented?

Project	Standards and technologies	Software License	Start/End	Maintained	Language	Documentation
CASPAR	XAM, OSD	GPL, PDS interfaces and client are distributed with Common Public License or BSD license, PDS server will eventually be placed on IBM	Apr 2006/ Oct 2009	?	Java	Yes, distributed with the Creative Commons Attribution Non-Commercial Share Alike license

Project	Standards and technologies	Software License	Start/End	Maintained	Language	Documentation
		alphaworks				
PLANETS	JCR API, Jackrabbit,	LGPL	Jun 2006/ Jun 2010	Yes	Java	Yes
PRESERV 2 / EPrints	OAI-ORE, EPrints Storage Controller	GPL	Jul 2007/ Mar 2009	Yes, KeepIt	Java	Yes
SHAMAN	Multivalent iRODS, SRB		Dec 2007/ Nov 2011	Yes	?	No
ZFS		Common Development and Distribution *License* (CDDL)		Yes	-	Yes

Table 9 Projects summary.

10.2. Candidate Solution

In this section we focus our attention to the requirements and definition of storage and preservation services we aim to provide within the PrestoPRIME preservation platform.

Software design and implementation of these services will be provided within the Work Package 5 Task 2, according to the scenarios identified in task 1. SIP and DIP format adopted in the PrestoPRIME preservation platform will be defined in Work Package 4 and 5. Here we describe the high level functionalities for preservation and storage for exchanging generic Information Packages made up of contents and metadata.

The requirements of the candidate solution could be distinguished into common and specific as follows:

Common requirements

The PrestoPRIME preservation platform should support different types of software languages and technologies as well as different hardware environments.

- Hence the interfaces exposed should be general enough to be implemented by different programming languages.
- Moreover the selected protocols of the interfaces should be "standard" and possibly supported by some active community.
- Concerning the communication layer (client <-> server), this should not be limited to specific configurations as specific ports/sockets or transmission protocols.
- Each operation performed by the user should be managed as much as possible as ACID (atomicity, consistency, isolation, durability) in order to enable the management of transactions

- The exposed interfaces should enable the user accountability by means of authentication and authorisation, possibly creating different profiles.

Specific requirements

Concerning the storage we can consider the following requirements:

- these interfaces should enable the CRUD operations (Create Retrieve Update and Delete) that are usually the basis of common persistence layers.

Concerning the preservation we can consider the following requirements:

- these interfaces should wrap basic storage functionalities to ensure the content preservation over a indefinitely long period of time, which means that content should always be available and usable (as well as undamaged). To accomplish this, we will comply to the OAIS reference model, which introduces functionalities such as ingestion, search and retrieval and stressed the need of referring the digital content to be stored to the metadata which allow its preservation.

Interface definitions

The choice of a standard solution for the preservation and storage interfaces is strongly supported by PrestoPRIME aims and is also motivated by the possibility to overcome the fragmentation of different solutions delivered by the several projects we listed above. We firmly believe that a long term Preservation Platform must deal with standards and must be open. The drawback of proprietary (non-standard) solutions is that they cannot guarantee the support of format changes and will become obsolete within the lifetime of the digital content they store.

Storage

The high level storage functions will enable the CRUD operations (Create Retrieve Update and Delete). Storage interfaces are published internally to the OAIS Archival Storage functional entity.

Internally the OAIS system works with the AIP and there are no specific recommendations on how to manage/store internally the AIPs in the OAIS model, leaving it to the implementation details of the archive.

We can figure out an abstract layer providing the physical storage functionalities to the archive, in charge of store and retrieve AIPs. The AIP itself could be split into several physical parts within the OAIS brought together by logical links maintained by the archive. According to the OAIS model, the software components implemented in the Archive interact with the storage component by exchanging the Archival Information Package (AIP), the complete bundle containing the contents and all the related information such as the preservation plans. A simple view is depicted below.

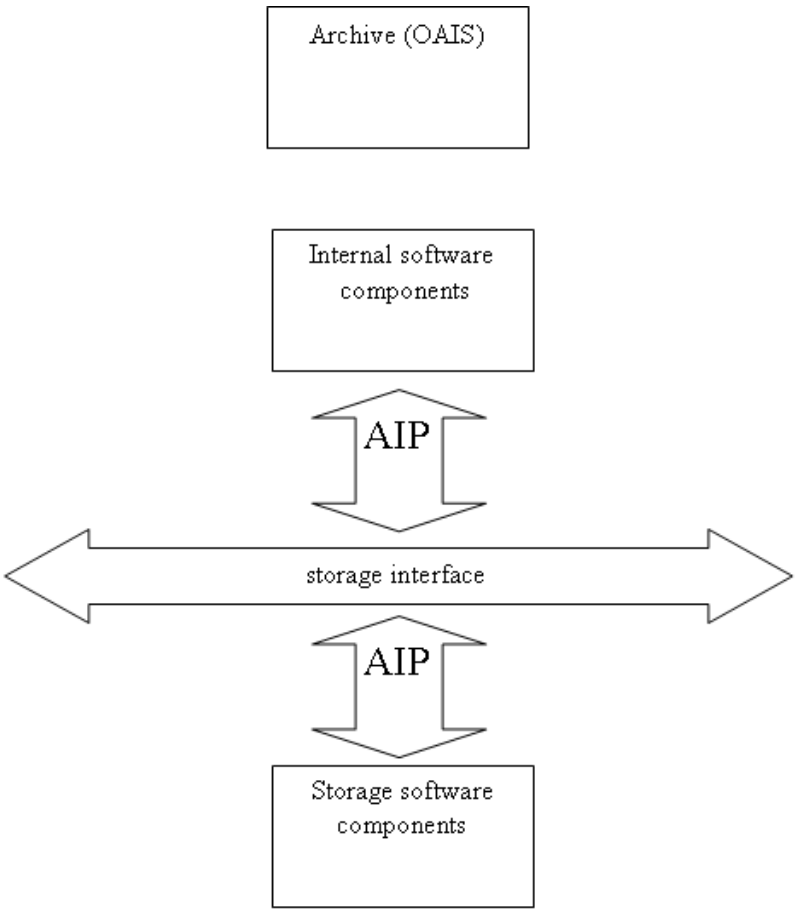


Figure 41 Abstract view of the separation layer of the storage interface within the OAIS archive.

According to the figure above, we can imagine the OAIS archive made up of the internal software component and also of the storage software components or we can separate it logically at the storage interface level. In the first case we have a complete system with its internal storage, without the actual need of a storage interface separation layer for it could be managed internally without the use of complex protocols and the introduction of useless overheads (as an example it could make use of shared network file systems instead of GRIDftp protocols for exchanging the resources). The latter case, we have a more open and flexible architecture but we are moving the storage components elsewhere from the archive. In this case the exchanging of AIPs should be managed and controlled by the OAIS archive and the storage interface should guarantee the AIP integrity and prevent any access to the stored AIP not managed by the OAIS. According to the UML definition of "composition" we can state that the storage software components must be bounded to the archive by means of the "composed by" relation (black diamond). The composition forbids any access to the composed element from outside except by the composer. Clients must ask to the OAIS the access to the resource.

Preservation Interfaces

Preservation interfaces are the OAIS compliant functionalities PrestoPRIME Preservation Platform will expose to final user.

The final user can assume the role of Producer or Consumer, as defined in OAIS standard:

"Consumer: The role played by those persons, or client systems, who interact with OAIS services to find preserved information of interest and to access that information in detail. This can include other OAISs, as well as internal OAIS persons or systems."

"Producer: The role played by those persons, or client systems, who provide the information to be preserved. This can include other OAISs or internal OAIS persons or systems."

We can divide the interfaces to final users into three main categories:

- Interfaces for dealing with digital contents (ingestion, retrieval, update/removal):these interfaces enable the user to insert a new content into the system, to get a content from the system and ask the system to delete it.
- Interfaces for dealing with metadata (search, access and browsing information):these interfaces provide functionality for both accessing and searching metadata enabling the user to access and browse the content by mean of its properties, to navigate, add or delete some properties/attributes/(generally metadata) related to a specific digital content, to get the information with regard to the digital rights associated to a specific content.
- Interfaces for management (management and audit): these interfaces are for managing and administering the digital contents that a system has to preserve and store. These interfaces have to expose some of the internal workflow (in order to understand if a job has been completed or not) and quality control.

It will be necessary to make use of widely adopted standard for processes and workflows definition, such as BPEL and BPMN.

In the following table, we list the high level interfaces a complete OAIS compliant preservation system exposes. We also state in which of the previously defined categories each interface falls, to which OAIS functional entity it is mapped, to what kind of user it is addressed – i.e. Producer or Consumer – and we give a brief description of its functionalities.

Name	Interface for dealing with	OAIS Functional Entity Mapping	Interface to	Description	Remarks
Ingest	Digital contents and Metadata	Ingest / Administration	Producer	Enables the user to submit a SIP to the Archive – previously created by the Producer according to the negotiated Submission Agreement. Provides to the Producer a confirmation of receipt.	In the most general case, the SIP will be constituted of a Content Information part and a metadata part. According to this division, this interface may be split into two parts, i.e. an interface for AV content ingestion (e.g. MXF file) and an interface for metadata ingestion (e.g. METS file)
Access	Digital contents	Access	Consumer	Accepts an order request from the Consumer and provides the requested DIPs by means of a Deliver Response.	An order may be an Ad hoc Order – executed only once – or an Event Based Order that will be maintained by the Activate Requests function in Administration
Search	Metadata	Access	Consumer	Enables the Consumer to perform a query to the Archive, sending a query request and to get a result set. This interface also handles report requests and delivers report results to the Consumer	
Navigation	Metadata	Access	Consumer	Enables the Consumer to navigate the Descriptive Information of a specific AIP and makes use of the functionality provided by the search, retrieval and access interfaces	
Removal/ Update	Digital content/ Metadata	Ingest	Producer / Consumer	This interface is in charge of the update of the Archive and enables the user to send a Descriptive Information of a specific AIP as well as a complete new SIP (only for update)	

Name	Interface for dealing with	OAIS Functional Entity Mapping	Interface to	Description	Remarks
Audit Submission	Management	Administration / Ingest	Producer	This interface is responsible of providing the Producer Audit Reports. If the SIP results inappropriate a list of liens and a resubmission request are sent to the Producer, who can decide to resubmit the SIP, in which case he's redirected to Ingest interface, or appeal the decision. After successful ingestion, a final ingest report is provided to the Producer.	After ingestion, the SIP undergoes a validation process.
Access Control	Manage-ment	Ingest / Access	Producer / Consumer	Responsible for access control. Enables the user to authenticate, log-in and profiling (forcing the use of the proper managed services)	
Manage-ment and Auditing	Manage-ment	Administration	Manage-ment	Administers the Archive, enabling the Management to submit Policies and Standards. It also lets the user manage system configuration and allows users to audit and verify of the fulfilment of the preservation plans	Every-day administration in an internal function in the OAIS reference model. In practice, a user interface will be needed.

Table 10 High level interfaces of a complete OAIS compliant preservation system.

Conclusion

This document has discussed many of the important aspects for the provision of storage services in PrestoPRIME. It draws some conclusions and acts as a reference for consumers looking at outsourcing storage systems.

A storage service cannot be considered in isolation from its use. “Storage” is about keeping a defined sequence of bytes but “preservation” is about being able to understand the information held in those bytes at a later stage and a storage service must support the preservation system in keeping hold of the information content as well as the bits and bytes.

In the case of broadcast production systems, more data is being kept all the time such as rushes, regional programming, HD programming, etc. This may have a dramatic impact on the physical requirements of a storage service with many petabytes of data potentially ingested per year. Consumer desire for access to video-on-demand services for both new and archived footage places a further demand on the underlying storage.

Ingesting all that data into the service of a single provider may be a popular option now, but looking to the future we may see greater use of federated architectures and novel software to protect the data with rules, coding techniques and encryption. Wherever the data is put we must remember that the data will live longer than the contract or even the organisation that holds the rights and so defining a realistic exit strategy is paramount.

Many protocols exist to transfer data from one site to another and these have been enumerated along with an initial investigation into the performance of FTP and GRIDFTP with GRIDFTP showing much higher performance for a “long fat pipe” that might typically connect an archive to a storage supplier.

Finally, aspects of service security have been discussed and a short preliminary investigation into the necessary interface has been made.

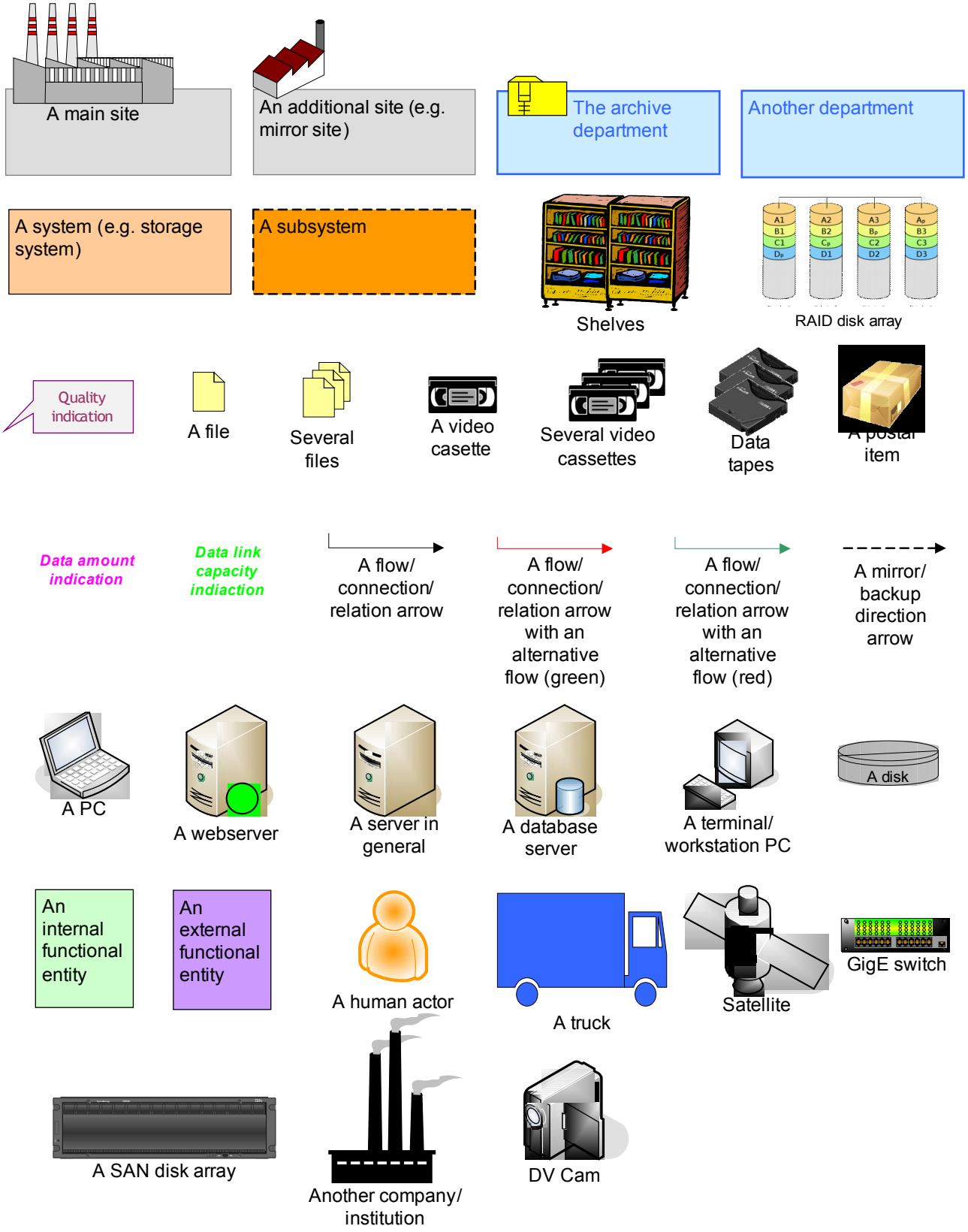
Glossary

Term	Definition
ECN	Explicit Congestion Notification: an extension to the Internet Protocol. ECN allows end-to-end notification of network congestion without dropping packets. It is an optional feature, and is only used when both endpoints signal that they want to use it.
BDP	Bandwidth delay product: refers to the product of a data link's capacity (in bits per second) and its end-to-end delay (in seconds). The result, an amount of data measured in bits (or bytes), is equivalent to the maximum amount of data on the network circuit at any given time, i.e. data that has been transmitted but not yet received. Sometimes it is calculated as the data link's capacity multiplied by its round trip time (RTT).
RTT	Round trip time: the “there and back” time – twice the latency
SAN	Storage area network: an architecture to attach remote computer storage devices (such as disk arrays, tape libraries, and optical jukeboxes) to servers in such a way that the devices appear as locally attached to the operating system.
Network latency	The time between sending and receiving a packet of data, measured in milliseconds. Also known as “ping time”.
LFN	Long fat network: a high latency, high bandwidth network.
OAIS	Open Archival Information System: a reference model developed by CCSDS.
TCP	Transmission control protocol: a transport layer protocol, part of the internet protocol suite.
UDP	User datagram protocol: a transport layer protocol, part of the internet protocol suite. No acknowledgement is required from the recipient for transmitted data.
SLA	Service Level Agreement. A formal and negotiated agreement between a service provider and service consumer. In the context of software services, SLAs are part of policy-based service governance, i.e. all terms of the service are described in the SLA and the service provider manages the service so it conforms to the SLA.
QoS	Quality of Service: forms part of a Service Level Agreement. Quantitative definition of the service to be delivered that can be measured using a set of metrics. For example, QoS of a media streaming service might be defined in terms of acceptable bandwidth, jitter, data loss etc.
DIP	Dissemination information package (from OAIS). The object used to disseminate data out of the archive.
Rushes	The raw unedited footage.
Media asset management	Management tasks and decisions surrounding the ingestion, annotation, cataloguing, storage, retrieval and distribution of digital media assets.

Term	Definition
AIP	Archive information package (from OAIS). The object that stores the data in the archive.
Z39.50	A client-server protocol for searching and retrieving information from remote computer databases.
SRW/SRU	Search/Retrieve web service and search/retrieval via URL: a more modern replacement for Z39.50 using HTTP and XML.
OAI-PMH	Open archives initiative protocol for metadata harvesting
OAI-ORE	Open archives initiative object exchange and reuse
XML	Extensible mark-up language: a set of rules for encoding documents electronically.
XPath	A language for addressing parts of an XML document.
SQL	Structured query language: a database computer language designed for managing data in relational database management systems.
Erasure coding	A way of encoding and duplicating data, more advanced than simple mirroring, which enables the reconstruction of the original file using a subset of the recorded pieces.
RAID	Redundant array of inexpensive disks: common method for increasing the reliability of hard disc drive based storage in a single machine by using mirroring and other more advanced techniques.
POSIX	Portable Operating System Interface (for Unix): a family of related standards defining the interfaces to the Unix operating system.
REST	Representational state transfer: a style of software architecture used in the world wide web.
SIP	Submission information package (from OAIS). The object that is used to send data to the archive.
PIP	Policy information point: part of the XACML standard where additional information can be obtained by the PEP.
PEP	Policy enforcement point: part of the XACML standard which protects the service from the user.
PDP	Policy decision point: part of the XACML standard which determines access according to the policies and information.
PAP	Policy administration point: part of the XACML standard where the policy can be set and/or updated.

Annexes

The notation used in the detailed architecture pictures is shown below.



List of Tables

List of Figures

List of References

[1] PrestoPRIME ID3.2.1, “Threats to data integrity from use of large-scale data management environments,” February 2010, http://www.prestoprime.eu

[2] PrestoPRIME ID3.4.1, “Service Level Agreements for Preservation Services”, February 2010, http://www.prestoprime.eu

[3] PrestoPRIME ID3.2.1, “Threats to data integrity from use of large-scale data management environments,” February 2010, http://www.prestoprime.eu

[4] OASIS: http://www.oasis-open.org/committees/download.php/19679/soa-rm-cs.pdf

[5] CCSDS Blue Book: http://public.ccsds.org/publications/archive/651x0b1.pdf

[6] PrestoPRIME D2.2.1, “Preservation Process Modelling”, February 2010, http://www.prestoprime.eu

[7] CASPAR Preservation DataStore Interface report: http://www.casparpreserves.eu/Members/cclrc/Deliverables/preservation-datastore-interface/at_download/file

[8] IBM white paper “*Preservation DataStores: New storage paradigm for preservation environments*”: http://www.haifa.ibm.com/projects/storage/datastores/papers/rabinovici.pdf

[9] CASPAR PDS API: http://developers.casparpreserves.eu:8080/hudson/job/CASPAR-PDS/

[10] XAM: http://www.snia.org/tech_activities/standards/curr_standards/xam/

[11] Object Storage Device: http://www.haifa.il.ibm.com/projects/storage/objectstore/index.html

[12] ANSI T10 SCSI OSD V1: http://www.techstreet.com/cgi-bin/detail?product_id=1204555

[13] ZFS: http://hub.opensolaris.org/bin/view/Community+Group+zfs/

[14] “*ZFS The Last Word In File Systems*”: http://hub.opensolaris.org/bin/download/Community+Group+zfs/docs/zfslast.pdf

[15] Jeff Bonwick’s Blog: http://blogs.sun.com/bonwick/entry/128_bit_storage_are_you

[16] ZFS Distribution List: http://hub.opensolaris.org/bin/view/Community+Group+distribution/links

[17] Project SHAMAN: http://shaman-ip.eu/

[18] Project iRods: http://irods.org/pubs/iRODS_Fact_Sheet-0907c.pdf

[19] SRB, Storage Resource Broker: http://www.sdsc.edu/srb/index.php/Main_Page

[20] iRODS Clients for Managing Data: https://www.irods.org/index.php/Documentation#iRODS_Clients_for_Managing_Data

[21] JARGON: https://www.irods.org/index.php/Jargon

[22] PRODS: https://www.irods.org/prods_doc/

[23] Cheshire3:

http://www.cheshire3.org/

[24] JISC:
http://www.jisc.ac.uk/

[25] Preserv 2:
http://preserv.eprints.org/, http://ie-repository.jisc.ac.uk/381/2/preserv2-finalreport.pdf
and http://wiki.preserv.org.uk/index.php/Main_Page

[26] EPrints:
http://wiki.eprints.org/w/New_Features_in_EPrints_3.2
and http://wiki.eprints.org/w/StorageController

[27] OAI-ORE:
http://www.openarchives.org/ore/

[28] Article "*Using OAI-ORE to Transform Digital Repositories into Interoperable Storage and Services Applications*":
http://journal.code4lib.org/articles/1062

[29] PLANETS:
http://www.planets-project.eu/

[30] PLANETS IF release report:
http://www.planets-project.eu/software/Planets_IF2345-D3_ReleaseReport-Final_Website.pdf

[31] PLANETS IF download page:
http://gforge.planets-project.eu/gf/project/if_sp/?action=index

[32] Apache Jackrabbit:
http://jackrabbit.apache.org/

[33] Java Content Repository API:
http://www.ibm.com/developerworks/java/library/j-jcr/
and http://www.day.com/maven/jsr170/javadocs/jcr-1.0/javax/jcr/package-summary.html

[34] A-Stor, white paper "Reliable Audiovisual Archiving Using Unreliable Storage Technology And Services":
http://www.avatar-m.org.uk/wp-content/uploads/2009/10/addisibc2009paperid379_2.pdf

[35] Avatar-m:
http://www.avatar-m.org.uk/

[36] Erasure coding:
http://oceanstore.cs.berkeley.edu/publications/papers/pdf/erasure_iptps.pdf

[37] Tahoe project:
http://allmydata.org

[38] Eucalyptus:
http://www.eucalyptus.com/

[39] Parascale:
http://www.parascale.com

[40] CERN LHC Grid:
http://lcg.web.cern.ch/LCG/public/overview.htm

[41] OAIS Blue Book:
http://public.ccsds.org/publications/archive/650x0b1.pdf

[42] OAIster project:
http://www.oclc.org/oaister/

[43] SRU:
http://www.loc.gov/standards/sru/

[44] The standard guide to audiovisual preservation is IAS TC-04, which stipulates use of Broadcast Wave Format: http://www.iasa-web.org/special_publications.asp

[45] H.264/MPEG4 Advanced Video Coding:
http://iphome.hhi.de/wiegand/assets/pdfs/h264-AVC-Standard.pdf

[46] AVS Video Editor:
http://www.avs4you.com/AVS-Video-Editor.aspx

[47] PrestoSPACE wiki:
http://wiki.prestospace.org/pmwiki.php?n=Main.TechRef

[48] EDCine project:
http://www.edcine.org/intro/

[49] BBC iStats:
http://www.bbc.co.uk/blogs/bbcinternet/iplayer/iplayerpublicitypackoct09.ppt

[50] MODS:
http://www.loc.gov/standards/mods/

[51] PREMIS:
http://www.loc.gov/standards/premis/

[52] METS:
http://www.loc.gov/standards/mets/

[53] JHOVE:
http://hul.harvard.edu/jhove/

[54] DROID:
http://freshmeat.net/projects/droid

[55] National Library of New Zealand Metadata Extraction Tool:
http://meta-extractor.sourceforge.net/

[56] ECN:
http://tools.ietf.org/html/rfc3168

[57] Survey of Protocols and Mechanisms for Enhanced Transport over LONG FAT PIPES:
http://www.globus.org/alliance/publications/papers/Survey.pdf

[58] Sliding Window Protocol:
http://en.wikipedia.org/wiki/Sliding_Window_Protocol

[59] XCP:
http://www.isi.edu/isi-xcp/

[60] Peer-to-Peer:
http://en.wikipedia.org/wiki/Peer-to-peer

[61] File Sharing:
http://en.wikipedia.org/wiki/File_sharing

[62] Communications Protocol:
http://en.wikipedia.org/wiki/Communications_protocol

[63] FTAM:
http://tools.ietf.org/html/rfc1415

[64] GridFTP over UDT:
http://www.globus.org/alliance/publications/papers/gridnets07.pdf

[65] PRISM project:
http://www.bbc.co.uk/rd/projects/prism/index.shtml

[66] UDT:
http://udt.sourceforge.net/index.html

[67] UDT wins SC08 bandwidth challenge award:
http://sourceforge.net/forum/forum.php?forum_id=891763

[68] List of Projects/Software/Companies that use UDT:
http://udt.sourceforge.net/software.html

[69] Movie2Me:
http://www.bce.lu/BCE_1D948DC84C0A4405B5FF8D523400B9BF.htm

[70] UFTP homepage:
http://www.tcnj.edu/~bush/uftp.html

[71] UFTP paper:
http://www.phatpackets.com/papers/jzhang-UFTPpapersubdspac.pdf

[72] RBUDP:
http://www.evl.uic.edu/cavern/papers/cluster2002.pdf

[73] Comparison of SABUL/UDT:
http://www.rgrossman.com/sabul.htm

[74] Tsunami UDP:
http://tsunami-udp.sourceforge.net/

[75] FASP:
http://www.asperasoft.com/technology/aspera-technology.pdf

[76] FASP results:
http://www.asperasoft.com/technology/comparisons/gigabit.html

[77] Digital Rapids C2:
http://www.digital-rapids.com/C2

[78] Blazeband:
http://www.kencast.com/index.php?option=com_content&view=article&id=37:blazeband&catid=11:KenCast%20Labs&Itemid=90

[79] KenCast:
http://www.kencast.com/

[80] Signiant Media Exchange:
http://www.signiant.com/media-exchange/

[81] FileCatalyst Homepage:
http://www.filecatalyst.com/

[82] FileCatalyst:
http://www.filecatalyst.com/products/accel.html

[83] Pro-MPEG forum:
http://www.pro-mpeg.org/

[84] BBC Production Gateway:
http://www.bbc.co.uk/rd/projects/index.shtml

[85] BBC white paper - "*Standardising media delivery in a file-based world*":
http://downloads.bbc.co.uk/rd/pubs/whp/whp-pdf-files/WHP158.pdf

[86] Matt Mathis, Raghu Reddy, "*Enabling High Performance Data Transfers*":
http://www.psc.edu/networking/projects/tcptune/

[87] Media Info:
http://mediainfo.sourceforge.net/en

[88] WANem
http://wanem.sourceforge.net/
free software created for Wide Area Network emulation using Linux "tc" commands under Netem package:
http://www.linuxfoundation.org/en/Net:Netem

[89] IETF RFC793: Transmission Control Protocol:
http://www.ietf.org/rfc/rfc793.txt

[90] Peter Brightwell, "*High-performance File Transfer over IP networks*":
http://tech.ebu.ch/docs/techreview/trev_2009-Q4_IP-Networks_Brightwell.pdf

[91] William Allcock, John Bresnaham, "*Maximizing Your Globus Toolkit GridFTP Server*":
http://www.globus.org/alliance/publications/clusterworld/0904GridFinal.pdf

[92] Peter Brightwell, "*Standardising media delivery in a file-based world*":

http://www.bbc.co.uk/rd/pubs/whp/whp-pdf-files/WHP158.pdf

[93] Richard Huang, Andrew Chien, "*Benchmarking High Bandwidth-Delay Product Protocols*": www.bbc.co.uk/rd/pubs/whp/whp-pdf-files/WHP158.pdf

[94] UDT wins SC08 bandwidth challenge award: http://sourceforge.net/forum/forum.php?forum_id=891763

[95] Information technology - Security Techniques - Information security management systems - Requirements, International Organisation for Standards (ISO), ISO/IEC 27001:2005.

[96] GRIA: http://www.gria.org/

[97] WS-Federation Passive Requestor Profile: http://en.wikipedia.org/wiki/WS-Federation_Passive_Requestor_Profile

[98] Shibboleth: http://shibboleth.internet2.edu/

[99] OpenID: http://www.openid.co.uk/node/3

[100] Transport Layer Security: http://tools.ietf.org/html/rfc5246

[101] SAML: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security#overview

[102] SecPAL: http://research.microsoft.com/en-us/projects/secpal/

[103] Delegation Logic: http://www.cs.yale.edu/homes/jf/LGF.pdf

[104] XACML: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml

[105] Axiomatics.Policy: http://www.axiomatics.com

[106] WS-Security: http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf

[107] WS Basic Security Profile 1.0: http://www.ws-i.org/Profiles/BasicSecurityProfile-1.0.html

[108] WS-SecurityPolicy: http://www.oasis-open.org/committees/download.php/15979/oasis-wssx-ws-securitypolicy-1.0.pdf

[109] WS-Trust: http://docs.oasis-open.org/ws-sx/ws-trust/200512/ws-trust-1.3-os.html

[110] WS-Federation: http://www-128.ibm.com/developerworks/library/specification/ws-fed/

[111] WS-SecureConversation: http://www.oasis-open.org/committees/download.php/15978/oasis-wssx-ws-secureconversation-1.0.pdf

[112] WS-Secure Exchange: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=ws-sx

[113] NextGRID: http://www.nextgrid.org/

[114] edutain@grid: http://www.edutaingrid.eu/

[115] SAML Token Profile: http://www.ws-i.org/Profiles/SAMLTokenProfile-1.0.html