PAWN: Producer – Archive Workflow Network in Support of Digital Preservation[1]

Mike Smorul, Joseph JaJa, Yang Wang, and Fritz McCall
Institute for Advanced Computer Studies
University of Maryland, College Park

Abstract

We describe the design and the implementation of the PAWN (Producer – Archive Workflow Network) environment to enable secure and distributed ingestion of digital objects into a persistent archive. PAWN was developed to capture the core elements required for long term preservation of digital objects as identified by previous research in the digital library and archiving communities. In fact, PAWN can be viewed as an implementation of the Ingest Process as defined by the Open Archival Information System (OAIS) Reference Model, and is currently being used to ingest significant collections into a pilot persistent archive developed through a collaboration between the San Diego Supercomputer Center, the University of Maryland, and the National Archives and Records Administration. We make use of METS (Metadata Encoding and Transmission Standards) to encapsulate content, structural, descriptive, and preservation metadata. The basic software components are based on open standards and web technologies, and hence are platform independent.

1. Introduction

We are pursuing a broad research program to address major components required to build a computing infrastructure for enabling long term digital archiving and preservation of digital assets. It is well-known that digital preservation is substantially more challenging than the traditional problem of the archival and preservation of physical objects. The added complexity is primarily due to: (i) the ease with which digital information can be created and disseminated; (ii) the fast pace of technology evolution; (ii) the fragility of the digital information and computing environments; and (iv) the security threats on the digital information that is interconnected to the internet. We outline in [1] our basic approach and provide a brief description of a three-tiered software architecture that can be used to anchor the major components for digital preservation while easily adapting to the wide variety of needs of different communities. In this paper, we describe the architecture, components, and implementation of our system PAWN (Producer – Archive Workflow Network), which provides secure and distributed ingestion of digital objects and associated preservation metadata into a persistent archive using open standards and web technologies. PAWN captures the essential features of the producer-archive interface methodology articulated in [2], which covers the first stage of the Ingest Process as defined by the Open Archival Information System (OAIS) reference model [3]. We make

[1] Sponsored by the National Archives and Records Administration and the National Science Foundation under the PACI Program.

use of METS (Metadata Encoding and Transmission Standard) schema [4] to encapsulate content, structural, descriptive, and preservation metadata, leading to the specification of the Submission Information Package (SIP) as defined by the OAIS model.

We are in the process of using PAWN to ingest substantial collections into the prototype persistent archive developed through a collaboration with the San Diego Supercomputer Center (SDSC) and the National Archives and Records Administration (NARA). The current pilot archive, built on top of the Storage Request Broker (SRB) data grid middleware, consists of three node servers at SDSC, University of Maryland, and NARA, and manages several terabytes of significant NARA selected collections. PAWN, developed by our group at the University of Maryland, enables secure and distributed ingestion into the pilot persistent archive, taking into consideration upfront preservation issues related to technology evolution, risk management, and authenticity checks.

The PAWN ingestion flow can be viewed as consisting of two major phases. The first phase involves the staging and assembly of data and related information to create the necessary pieces of a SIP, as agreed upon by the producer and the archive. The second phase consists of verification of metadata, bitstreams and preservation information at the archive, followed by a storage into a persistent archive. While developed independently, our ingestion workflow turns out to be in essence similar to the detailed plan described in [2]. The two phases can be carried out through the following steps.

1. Negotiation of the details of the information to be preserved between the producer and the archive. The details should result in a clear understanding of the elements necessary to assemble the SIP. This phase is supported by tools that will lead to the specification of an XML configuration document (consisting of a METS document and a constraints/rules document).
2. Initialization of the ingestion process at the producer site, which includes the arrangement and verification of the information, and the assembly of appropriate SIPs.
3. Transfer data to the archive after establishing secure communication between the producer and the archive.
4. Validation phase consisting of verifying bitstream integrity and validation of metadata as specified by the XML schema at the archive.
5. Organization of data into collections and transfer into a persistent archive.

Before describing the software components of PAWN, we start in the next section with a very brief overview of the OAIS reference model, followed by an overview of the PAWN architecture in Section 3. We elaborate on the above steps in Section 4, while the software components of PAWN are described in Section 5.

2. Brief Overview of the OAIS Reference Model

We briefly introduce the overall framework of the Open Archival Information Systems (OAIS) that defines concepts needed for long term preservation of digital information. We will focus on the first component, Producer – Archive interface, as we have borrowed terminology from there to develop PAWN.

The overall model is shown in Figure 1, where producers prepare and transfer the information to be preserved to an archive responsible for managing the digital information for long term preservation and for providing an interface to the consumers for accessing the information as needed. For each stage, OAIS provides a detailed model of the information, called respectively the Submission Information Package (SIP), the Archival Information Package (AIP), and the Dissemination Information Package (DIP).

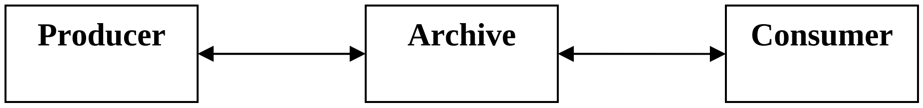

Figure 1

Most relevant to PAWN is the SIP that consists of the following components:

- First is the *Content Information* (CI), which is divided into two parts.
    - *Content Data Object*, consisting of the actual bitstream to be preserved.
    - *Representation Information*, which includes file format, endian issues, and encoding format. Consider for example the case of image files. The corresponding information would be given about expected header formats, location of internal checksums and what utilities can be used to verify the file can be loaded. Enough information should be given to ensure that the archive would be able to process the bitstream.
- Second is *Preservation Description Information* (PDI) that contains four parts.
    - *Chain of custody*.
    - *Context* in relation to other Information Packages
    - *Reference information* unique to the bitstream (eg, ISBN, global identifier, etc)
    - *Fixity information* required to ensure bitstream integrity (eg, hashes, or checksums).
- *Packaging Information* describes the relationship between CI and PDI. This describes the physical location of the Content Information and corresponding PDI.
- *Descriptive Information* used for data discovery. This user-defined metadata will be supplied during the ingestion of the bitstream at the producer. This includes descriptions of the bitstream, authorship and other elements (e.g. Dublin Core). This type of information may also be automatically generated by tools written for a special collection (such as E-mail header harvesting).

We have used METS to represent the various elements of SIP, as will be illustrated later in this paper.

3. Overview of the PAWN Architecture

PAWN consists of three major components: management server at the producer; client at the producer; and receiving server at the archive. The overall architecture is shown in Figure 2.

We assume that, in general, a number of people at the producer will be engaged in preparing and transferring data to the archive. The management server will act as a central point for the initial organization of the data, and for tracking bitstreams and metadata functionality. More specifically, this server performs the following functions:

- It provides the necessary security infrastructure to allow secure transfer of bitstreams between the producer and the archive (using Certificate Authority as explained later).
- It assigns a unique identifier for each bitstream to be archived, which is unique within a collection, but not globally unique.
- It provides an interface for bitstream organization and metadata editing.
- It accepts checksums/digital signature, system metadata and other client supplied descriptive metadata.
- It tracks which bitstreams have been transferred to the archive.

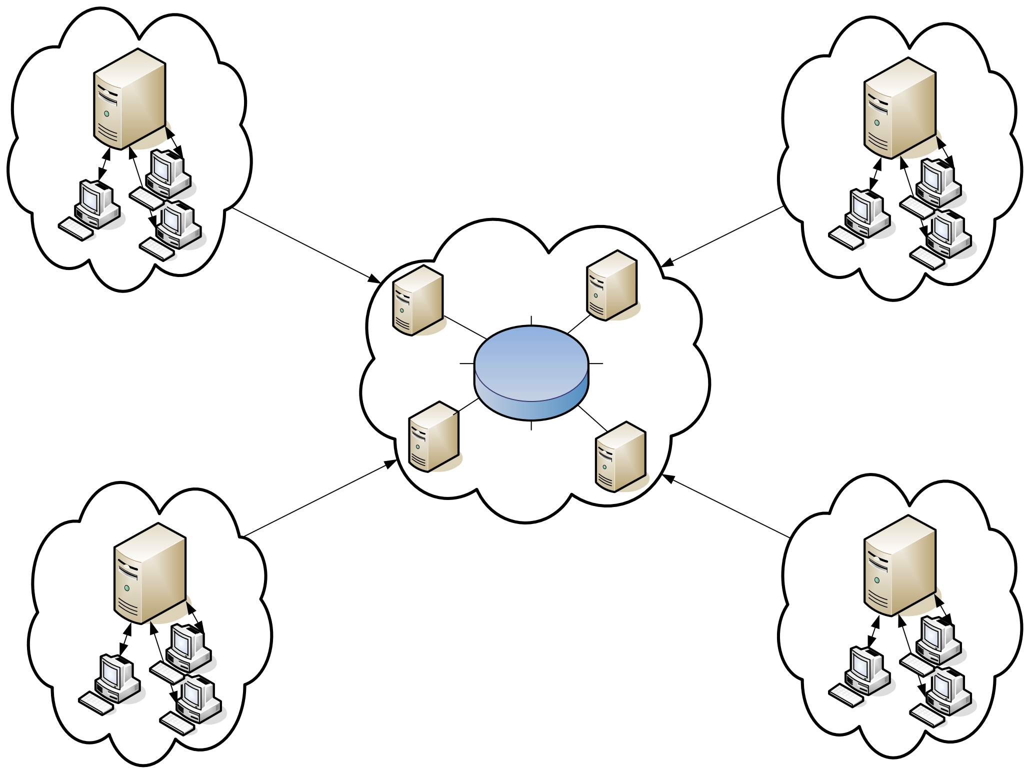

Figure 2

A client will run on each machine to automatically register preservation information and transfer the corresponding SIPs to the archive. The client will be responsible for:

- Bulk registration of bitstreams, checksums and system metadata;
- Assembly of a valid SIP;
- Transmission of SIP to the archive either directly or through a third party proxy server; and
- Automatic harvesting of descriptive metadata (e-mail headers, etc) as necessary.

The archive will have a server setup to receive data transferred from the producer. This server will accept data and initiate verification/validation processes on the bitstream. Some security key negotiation between all three areas may be necessary for the producer to securely transfer documents to the archive. The receiving server will need to do the following:

- Securely accept SIPs from clients at a producer site;
- Process SIPs and initiate verification/validation processes;
- Coordinate authentication with the management server at the producer site;
- Verify with the management server that all SIPs have arrived intact; and
- Provide enough temporary storage for incoming SIPs until they can be replicated into a digital archive and validated.

Description of the software components behind the management server, client, and receiving server will be given in Section 5.

4. Description of the Proposed Ingestion Workflow

We provide some details of each of the steps of the ingestion workflow of PAWN.

Step 1. Specification of Information Objects to be Preserved

The first stage of the ingestion workflow consists of iterative negotiations between the producer and the archive leading to a specification of an XML configuration document.

Before any ingestion can occur, the producer and the archive must work together to develop precise expectations of all the components of a SIP – content information, representation information, preservation information, descriptive metadata, access rights and privileges, and intellectual property rights. We are developing tools to automatically generate METS and constraints documents, which in particular will capture the precise details of the agreed-upon SIP.

Note that the customization of each SIP component will vary. Descriptive information, context, and representation information will likely need to be customized for individual collections. Packaging information, chain of custody, and fixity information will be provided by PAWN and will require almost no customization.

Step 2. Initialization of PAWN

There are two main steps involved in the initialization of PAWN: security initialization and registration of bitstreams, both of which are described next.

Security Initialization. Using the XML schema developed in Step1, we can initialize the three components of PAWN once various trusts are established between the producer and the archive to ensure transparent secure data transfer. In our model, we allow for separate Certificate Authorities at the producer and the archive. This will enable both parties to use pre-existing security configurations at their sites. The producer will need to initialize systems necessary for issuing and managing clients. The security components of PAWN are described in Section 5.2.

Registration of Bitstreams. After the appropriate interfaces have been initialized, the producer can start registering items to be archived. Registering data at the producer covers two separate areas. A client will gather information necessary to track the file on a client such as location and checksum. It will then attach metadata including basic hierarchical arrangement, system and descriptive metadata. A client may also analyze the file and extract descriptive metadata specific to a particular format. This information is used to create a METS document that is merged at the producer management server with other client submissions.

After the bitstream and the corresponding metadata have been registered, the producer will be able to arrange the bitstreams and metadata into appropriate collections within the context of the producer. This arrangement is similar to the community arrangement for DSpace [6]. Allowing the producer a chance to arrange the bitstreams to provide context may be necessary since bitstreams may be ingested out of order, through differing mechanisms, or a client may not be able to supply an overall environment to the data. The result of this arrangement will be incorporated into context information about the object.

Step 3. Data Transfer to the Archive

At some point, a client will need to transfer registered bitstreams to an archive. The client will retrieve a METS document from the producer containing all bitstream location and metadata information about items it has registered. Depending on the client, the bitstream may be verified against available checksums to see if any changes have been made since it was registered. Any changes should be noted in the audit trail and additional action may be taken based on the collection specification. After verification, a SIP is created comprising the bitstream metadata stored in XML format, and the bitstream to be transferred. This packet will contain all the information required to validate the bitstream and ingest it into an archive.

The transfer of SIPs from the producer can occur over a number of mechanisms. In the event of a firewall, it may be necessary to proxy the connections through a host capable of communicating with the archive. The transfer will need to take into consideration the security and size requirements of the collection. Possible mechanisms include:

- Physical transfer of disk drives, CDs, or tapes.
- Clear-text ftp or encrypted sftp transfers;
- Gridftp high throughput parallel transfer.

After the SIP has been transferred, the receiving server at the archive will communicate with the producer's management server and notify it that the transfer has been successful.

The Packaging Information in the IP will track the temporary location of files withing the SIP. Any other staging locations of the file, including original location will be recorded, which will provide a fairly detailed record tracking bitstreams within a SIP. Information regarding the chain of custody should be updated as the IP is replicated and moved from the producer to the archive.

Step 4. Validation and Verification at the Archive

The requirements to maintain a valid copy and to perform verification techniques will have a different set of hardware and software requirements. For verification of the bitstream and metadata, the data will likely have to be processed on a variety of systems. These verification systems may be specialized and may not efficiently integrate into an archive.

When a SIP arrives at the receiving service, it is immediately unpacked and tracking information regarding bitstreams is updated. There are two types of validation performed on an incoming SIP. First is metadata validation of the METS document contained within the SIP, and second is a per bitstream validation.

Metadata Validation. Since the format of the metadata in the SIP is specified by an XML schema, it is possible to automatically verify the format of the supplied metadata. The receiving server can either locally validate metadata as it prepares the bitstream for archiving or can offload the task to another processor. After an entire collection has been transferred, the producer will have to contact the archive to verify the entire collection has been received.

Should metadata validation fail, an entire SIP will be rejected as the metadata included items necessary for further processing of the SIP.

Bitstream Validation. The Representation Information (RI) specifies various mechanisms to process and display the data. Extending this concept, we have also assumed that RI will specify information to ensure readability of bitstreams. A set of seperate validation mechanism can be grouped to form a validation pipeline unique to each type of bitstream. Copies of the received bitstreams will be processed through the appropriate validation pipeline.

Bitstreams that fail validation are removed from the IP and notification of the failure is returned to the producer management server. As the validation processes are operating on copies of the original bitstream, there is no worry about corruption of the original bitstream.

Step 5. Create Collections and Store in a Persistent Archive

After all files in a SIP have passed validation, the entire SIP is moved into a persistent digital archive. This archive should provide functionality for generating a unique identifier to track bitstreams regardless of arrangement, and one or more arrangement mechanisms.

Using the arrangement cababilities of the archive, bitstreams can be organized into an archive specific view while allowing alternate access by referencing the SIP metadata. As the metadata within the SIP provides its own arrangement, it will be pushed into the archive and file reference will occur through the unique identifier supplied by an archive.

The archive will enforce any preservation requirements specified during negotiation. This will consist of how many replicas are to be maintained, expiration dates on parts of the collection, and possible media requirements for preservation. Depending on the overall architecture of the archive, additional information about replica integrity checking, migration policy and geographical dispersal may be required as well.

5. Software Components of PAWN

PAWN was implemented using open standards and protocols and web technologies. Its software components include the XML schema and editing tools, metadata storage and arrangement at the producer, client uploads, and a server at the archive to receive data, and bitstream verification services.

5.1 XML Schema and METS Representation

The METS schema from the Library of Congress [4] was chosen as a base schema to use to define the necessary OAIS components. Expanding on the METS schema, we have developed additional components that can be used within METS to define OAIS components and to restrict METS to the needs of the collection. Since METS does not try to specify a form of metadata, but rather allows arbitrary metadata to be referenced or included and arranged in arbitrary hierarchies, it is a flexible enough base to build PAWN on.

METS Usage. Internal elements within METS satisfy many of the OAIS requirements for defining a SIP. Where possible we have tried to use the tools supplied by METS to accomplish various archival tasks.

Direct mappings between METS and OAIS occur in several areas. The File section within METS allows for Fixity information in the form of checksums, Reference and chain of custody through Flocat tags, and the structural section provides context to bitstreams. Descriptive information can be associated with bitstreams at various points.

We have also chosen to use the structure of METS to represent an entire submission of the producer. This means that each client will create a METS document describing what it will supply. These documents are then merged at the producer management server into a larger virtual METS document. This process is described later.

Constraining METS. The flexibility that METS offers (by not restricting what can be attached or wrapped) may create a problem when trying to express what should be transferred to an archive. To constrain METS and allow it to be molded to fit a particular producer-archive agreement, we create a *constraints document*. This document allows a machine readable document to be created which describes the expected metadata and bitstream arrangements.

The METS profile [7] was created to aid programmers and authors guidance as they create METS documents. The format, while XML, contains guidelines that are expected to be interpreted by a human. This was insufficient for our needs as we required machine actionable enforcement of various constraints.

Example constraint document:

```
<divrule DIVID="ID1" RESTRICTDIV="true" RESTRICTFTPR="true"
RESTRICTMPTR="true"/>
<divrule DIVID="ID1.1" RESTRICTDIV="true" RESTRICTFTPR="true"
RESTRICTMPTR="true"/>
<divrule DIVID="ID1.1.1" RESTRICTDIV="false" RESTRICTFTPR="false"
RESTRICTMPTR="true"/>
```

The above shows the basic properties of a constraints document. The constraints, in combination with a skeleton METS document limit the form any derived METS documents can assume. It should be noted that the rules do not apply to the skeleton document, just modifications to it. There are other attributes, not shown above, which will allow rules on valid metadata and file types that may be included within a METS document

The above rule set would be combined with a skeleton METS document similar to the following:

```
<div ID="ID1" LABEL="Research & Development Records" DMDID="DM1">
  <div ID="ID1.1" LABEL="Research & Development Project Records"
DMDID="DM1.1">
    <div ID="ID1.1.1" LABEL="R&D Project Case Files"
DMDID="DM1.1.1"></div>
  </div
</div
```

The rules above would restrict all new div's, file pointers, and METS pointers except for the '*R&D Project Case Files*' section which allows file pointers to be inserted. We have used similar rules and METS documents to model NARA record schedules and enforce allowed locations for file and directory uploading.

Validation Document. We have developed a validation schema that can be used to specify a pipeline of necessary steps to guarantee a bitstream is able to be processed by an archive. In combination with any additional information specified in the behavior section of a METS document, it will supply a complete set of Representation Information describing a bitstream.

A validation document defines groupings of required/optional tests that should be applied to a given bitstream. It has the dual role of not only specifying a set of tests to perform on the SIP, but also enabling the capture of the results of those tests for long term preservation.

Example validation document:

```
<valgrp label="Tiff" required="true" ID="ID000034">
  <valtest name="generictiff" required="false" />
  <valgrp label="Word Parsing" required="true" ID="ID000035">
     <valtest name="geotiff" required="false" />
     <valtest name="richtiff"  required="false" />
  <valtest name="kodaktiff"  required="false" >
       <valtestresult RESULT="true" DATE="2001-12-17T09:30:47-
05:00"/>

     </valtest>
  </valgrp>
</valgrp>
```

The above example demonstrates several characteristics that the validation document can have. First, using valgrp, several tests can be grouped into a collective requirement. The result for the kodaktiff test is also specified in the document. The test name refers to an abstract test rather than a concrete class, function, web service or other validation service. It is expected that an archive will maintain records of test definitions.

Multiple validation documents are designed to be embedded into the techMD portion of a METS document sent with each SIP.

Descriptive Metadata. While the constraints document specifies the valid metadata constraints, we have not developed any new descriptive metadata standards as they are often domain specific. The METS community has worked with embedding several metadata schemas within METS.Given our extensive experience in dealing with such geospatial data through the Global Land Cover Facility (GLCF) at the University of Maryland, we are in the process of adapting PAWN to handle such collections.

5.2 The Management server

As stated earlier, the management server at the producer site serves as a central point for organizing the data and tracking bitstreams and metadata. It plays two roles, one to store and manage submissions from various clients, and the other to manage security issues between clients, itself, and the archive.

All communication between clients, an archive, and management is performed through a common web service layer. This allows for a variety of clients from various platforms and languages to connect.

Security Management in PAWN

Our overall security architecture is based on open standards (PKI, X.509, and GSI) and distributed trust management. It enables mutual authentication, confidential communication, and requires no or minimum user intervention. Since we assume minimal operational trust between an archive and producer, we allow for each party to manage security locally.

The producer will set up a Certificate Authority (CA), or use a pre-existing one that is trusted by the receiving servers at the archive. The producer’s CA will track current valid certificates and assign a unique certificate for each client that will supply data. At the archive, the receiving server will verify connecting clients by checking their certificates against the producer CA certificate in the same way a web browser verifies a bank's certificate against a commercial CA such as Verisign.

Aside from setting up trust, the validity of connecting clients needs to be determined at both the producer and archive. The producer site may revoke certificates or clients at any point due to security concerns or other reasons. A SOAP call to the security component of the management server will determine whether the presented certificate is on a revocation list, or if it is valid.

We have developed a Web-based CA for sites that do not have a pre-existing CA infrastructure. This CA will generate and manage standard X.509 certificates. Such a CA draws on the advantages of Java and some J2EE features. The software requires minimum knowledge and instructions to set up and configure, and can be repackaged and redeployed at any time for easier back-ups or system migration, and is completely self-contained to ensure minimum external dependencies.

Certificate management within PAWN uses the BouncyCastle API and java keystore technology. While certificates are stored in java keystore format for internal usage, they can be exported as grid compatible certificates.

Bitstream and Metadata Storage

The producer management server allows for METS submissions from multiple clients to be merged into one METS view. As each METS document submitted will be completely defined, that is, all XML ID's referenced in a METS document are defined in the same document. This will lead to redundant information being submitted from clients. For example, each METS document will have definitions of all referenced validation services. The METS server merges the XML ID's from clients and creates a unified METS view from all submissions.

As this document would be too unwieldy to process in memory, it is represented in a SQL database. A modified version of the Harvard University Library METS Java

Toolkit[8] is used to parse client submissions and register it into a database. As needed, complete sub documents of this view are created for clients when they create a SIP. As a client only needs bitstreams it registered, any supporting part of the producer hierarchy and referenced metadata, these sub documents keep the size of the METS SIP document within reason.

Bitstream and Metadata Management

The management interface performs two functions, editing metadata and allowing organization of registered bitstreams.

While clients are best suited to supply per bitstream metadata, descriptions for particular places in the hierarchy may be necessary. This involves merging domain specific metadata schema into the METS structural mapping. The METS group has several examples of Dublin Core, MODS and others available. These specifications generally work at the object level. For metadata descriptions that contain their own structural information, this merging will be more difficult.

We are investigating ways to allow plugin drivers for various metadata types that can be used at the client and producer level. Currently we only support simple textural item based metadata attachment to structural items at the producer level.

An arrangement interface enables the producer to organize and provide context to the bitstreams. This component allows for different individuals within the producer organization to have different roles in arranging and editing attributes associated with a bitstream. In our current version, we provide the ability to hierarchically arrange bitstreams.

As our model is inherently distributed with multiple producers, the roles of individuals vary significantly from other systems that assume ingestion through a centralized pipeline. As the content ingested from one producer is anticiapted to be under one administrative domain, the roles should be considerable simplified. Currently we have only implemented one global or superuser level of access to arrangement.

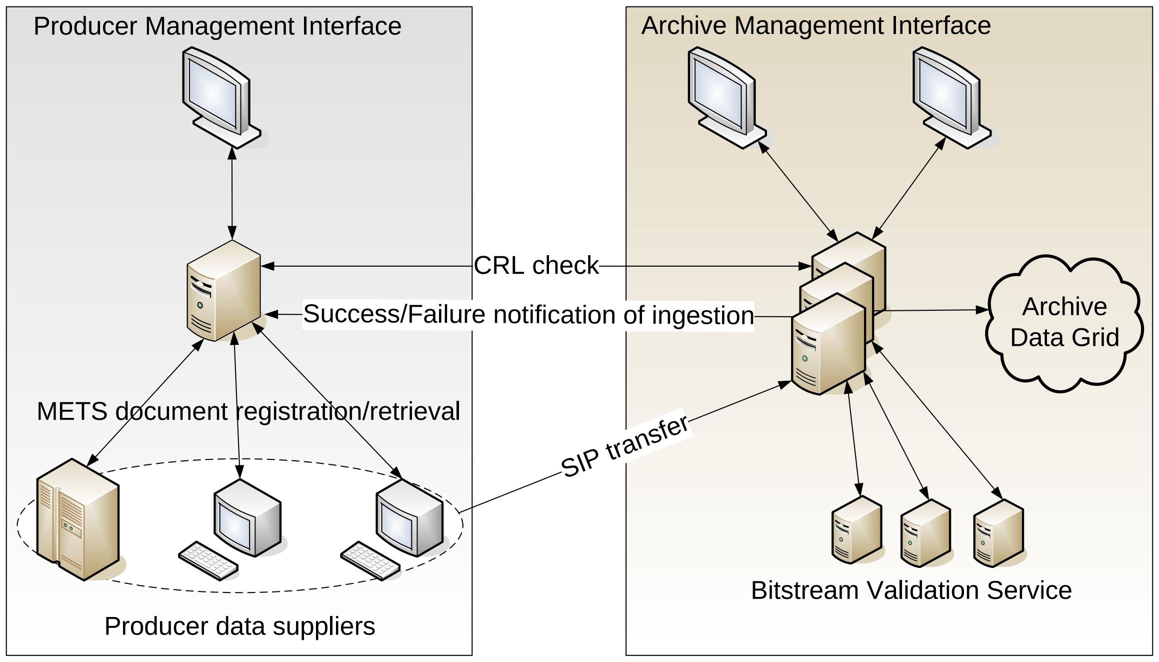
Producer Management Interface
Archive Management Interface
CRL check
Success/Failure notification of ingestion
Archive Data Grid
METS document registration/retrieval
SIP transfer
Producer data suppliers
Bitstream Validation Service

Figure 3

5.3 The Client at the Producer

A stand-alone client was developed to run on each data source at the producer. This client generates and receives METS documents and communicates with the management server using X.509 certificates. The current version was tested on Linux and Windows. It offers file browsing/selection options, perform registration of a bitstream and system metadata to the management server, and transfer data to the archive. The client is also capable of harvesting preservation information such as local file attributes (checksum, size, type, etc.) and original host and file location, and can arrange files into an abstract hierarchy.

To implement the client, we chose to create a standalone Java application run from CDROM to be executed on each workstation. The client does not store any data locally on the machine. We anticipate that in many cases a user will not have appropriate permissions to install software on a machine. Running a standalone application from a CDROM will allow someone to archive their own files without compromising their systems integrity.

For our prototype, each client will have a custom CD created that contains the configuration for a given client and any necessary security certificates. The client will be trivially locked to a given workstation by the workstation's IP address. Future clients should be able to receive its configuration and security information from the management server in addition to having stricter node locking ability.

Creating Submission Information Packages is accomplished by using the tar format as implemented by GNU tar. This is a common format that can be easily decoded on a variety of platforms. Since all metadata and bitstreams will be wrapped into one package, the transfer can be optimized for bulk transfer. While there are minor issues that prevent tar from being an ideal archive container, it provides the necessary functionality to act as a transfer agent.

While the main client used for demonstration purposes will be a highly interactive Java GUI, non-interactive clients will be developed as well. Since the web service interface is platform-independent, it is possible to use a variety of languages to register data and transfer SIPS. We have tested simple PERL scripts against the PAWN producer server that could be used to automatically harvest descriptive metadata from bitstreams during registration.

5.4 The Archive Receiving Server

The receiving server at the archive was briefly described in Section2. It coordinates with the management server of the producer to authenticate a connecting client, provides temporary storage for incoming bitstreams (SIPs), and triggers various validation services. The receiving server is designed to be a standalone server with enough storage for processing and verifying SIPS only. It is a lightweight service that can be easily duplicated behind load balancing technologies to achieve high throughput.

Data is transferred to the receiving server by various clients. When a client requests a transmission, the receiving server will first check with the producer's revocation list to ensure a client is valid. Information regarding which producing server supplied data is contained within the METS header.

Once data arrives at the server, the SIP is dissected and sent to various validation services as soon as possible. In our current implementation, this is done non-interactively to allow for the quickest possible transfer into a more permanent storage facility.

The receiving server is currently using the Storage Resource Broker (SRB) to provide the digital archive functionality. Using the SRB will also allow hierarchical arrangement information regarding the collection to be easily stored, in addition to natively storing audit trails and replicating bitstreams.

Metadata Validation

This component at the archive is used to validate metadata once it arrives. More specifically, it will verify metadata integrity and check collection contents against the manifest. When a SIP arrives, it is immediately unpacked and sent through metadata validation. The validation consists of parsing the received METS document and verifying METS syntax. Further validation is also done to ensure that metadata and structural components are compliant with the negotiated agreement.

Bitstream Verification

Each SIP contains preservation descriptive information as part of its metadata specification. The specification should provide information regarding which verification method(s) to use, and in what order to apply methods. As in the metadata validation stage, many of the bitstream verification steps may be performed in parallel achieving a high throughput. We have developed an XML schema that can be included within the METS agreement and further expanded to record the results of individual tests.

We have created a Web Services Definition Language (WSDL) interface file that will specify the communication between the receiving server and various validation services. Using an http-based communication path allows us to scale the validation service using traditional web load balancing techniques.

We have developed a Web service framework that is accessible via SOAP to provide the validation service. The framework should be extensible through add-on modules to validate bitstreams in different file formats. A few sample plug-in modules for validating major file formats, such as JPEG, TIFF, and PDF have already been demonstrated in our framework.

6. Conclusion

Using current web technologies it is possible to construct an open archival workflow that is largely platform independent. We have described in this paper the design and software components of the PAWN system that enables secure and distributed ingestion of digital information into a persistent archive. The pieces encapsulated within each digital object capture the concepts advocated by the OAIS reference model and will hold the essential information needed for the long term preservation of each object. The system is currently being used to ingest significant collections into a pilot persistent archive prototype developed through a collaboration between SDSC, University of Maryland, and NARA.

7. References

1. The ADAPT Project: An Approach to Digital Archiving and Preservation Technology, J. JaJa, M. Smorul, F. McCall, G. Jackson, and Y. Wang, in preparation.

2. Producer-Archive Interface Methodology: Abstract Standard, Consultative Committee for Space Data Systems, CCSDS- 651.0-R-1, Red Book, December 2002.

3. Reference Model for an Open Archival Information System (OAIS), CCSDS 650.0-B-1, Blue Book, Issue 1, January 2002 [Equivalent to ISO 14721:2002].

4. METS – Metadata Encoding and Transmission Standard. http://www.loc.gov/standards/mets/

5. NARA Persistent Archives: NPACI Collaboration Project, R. Moore et al., SDSC Technical Report, 2003.

6. DSpace - http://libraries.mit.edu/dspace-mit/index.html

7. METS Profile 1.0 Requirements, J. McDonough, http://www.loc.gov/standards/mets/profile_docs/METS.profile.requirements.rtf, June 2003.

8. METS Java Toolkit, S. Abrams, http://hul.harvard.edu/mets/, May 2003.