# EuroStore
# Initial Design and first Results

Martin Gasthuber (Martin.Gasthuber@desy.de)
Patrick Fuhrmann (Patrick.Fuhrmann@desy.de)
Deutsches Elektronen Synchrotron – DESY Hamburg/Germany
Duncan Roweth (duncan@quadrics.com)
Quadrics Limited – Bristol/Great Britain

**Abstract**

A European consortium formed by science and industrial partners have started the EuroStore project[1] to develop and market a Hierarchical Storage Management System (HSM) together with a high performance parallel file system (PFS). The EuroStore project aims to design and develop a high performance file store. It will combine the features of a Hierarchical Storage Manager (HSM) with the performance of a parallel file system to provide a system capable of meeting the requirements of the most demanding applications in industry, commerce, and science. The few existing HSM and parallel file system products do not fully address all the needs of the user community, and, in some cases, represent a major investment that users are not willing to undertake. This paper will show the user requirements and outline the design and implementation work as of today.

[1] Funded by the European Commission ESPRIT PROJECT 26317

## 1 Introduction

The complete EuroStore project is divided into two major components – the HSM at the bottom and the parallel file system as its client on top. Figure 1 shows the basic overview of the complete system. This simple separation into two major building blocks leads to the separation of development tasks where both, the parallel file system and the HSM will be developed by two different teams with minimal dependencies to each other.

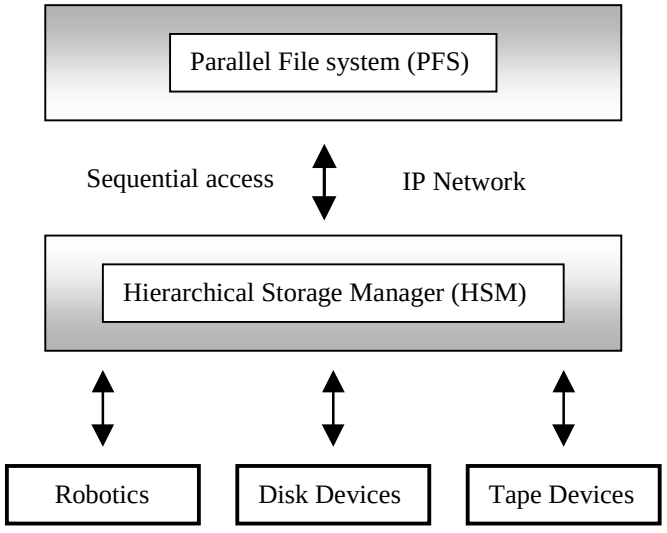


Figure 1: Basic component overview

The complete EuroStore collaboration consists of the following companies and science institutes.

- QSW (Supercomputer - UK)
- CERN (Science - France)
- DESY (Science - Germany)
- HCSA (Space Techn. - Greece)
- Tera Foundation (Medical - Italy)
- HNMS (Met-Service - Greece)
- AMC (Medical - Greece)

DESY and QSW will be the development sites for the HSM and parallel file system respectively. The proposed project was approved by the European Commission and started at March 1st, 1998 for the duration of 2 years in the first stage.

# 2 Requirements

This section will describe the initial requirements being the main and initial motivation to start the EuroStore project, and the user requirements created in the context of the project by CERN, Tera, HNMS and AMC.

## 2.1 Initial Requirements

Both science partners (CERN and DESY) are working in the area of High Energy Physics, where large storage capacity and high aggregate throughput are historically demanding requirements. For the typical physics analysis step the basic requirement is very simple – we just need a very big disk. This is of course contrary to the economic situation of HEP[2] labs and on the other side not easy to realize from the technical standpoint. Today's storage capacity requirements are going from several 100 TB up into the PB region (see Shiers [1]). More demanding are the bandwidth requirements of hundreds of megabytes per second for a complete, random access to datasets over the complete repository. Other demanding requirements come from the online data logging process, which will dump new physics raw-data from the running experiment to the repository. In this case, the storage system must have minimal latencies (real time behavior) and some sort of guaranteed aggregate bandwidth, because the spooling capacities on disks are limited and/or the online data rate is very high.

QSW and HCSA coming from the industrial world, have detected an increasing demand for storage management systems in the mid and high end ranges. Some sites are using backup systems in order to manage their data, with some success. Other today available systems (commercial or semi commercial) are inadequate with respect to their storage management features and capacity or are simply too expensive in the initial and running costs. In addition, the manageability of such systems is too complex and thus too expensive.

## 2.2 User Requirements

During the execution of the project the end-user partners have to create a user requirements catalogue. This catalogue has been revised by the development parties and is now formally agreed. In the following I will briefly go through these requirements highlighting the special and extraordinary needs.

### 2.2.1 Scalability

Scalability needs to be divided into performance scalability and administration scalability (effort to keep the system up and running)

- Storage Capacity – several hundred GB up to several PB
- Transfer Rate – hundreds of concurrent streams with tens of megabyte per second, resulting in a total aggregate bandwidth of several GB/sec. This actually depends on the installed hardware environment, but the achievable data rate should be no less than 85% of the slowest component involved in the data transfer.
- Number of simultaneous clients – sites like CERN, require that the system can handle at least 1000 simultaneous clients accessing the system. Other end-users have lower limits.

### 2.2.2 Architecture and Efficiency

The majority of requirements result in a network centric architecture, allowing a very efficient and flexible configuration on one or more heterogeneous computing and storage systems. Storing and retrieving a dataset should have transactional behavior and the process of storing and retrieving should be accompanied by a set of parameters characterizing the demand for this operations. Some end-users require an optimized handling of small datasets (a few KB) whereas large dataset handling in an optimized way is not considered a major problem. Much of the end-user interface architecture is covered by the definition of a POSIX compliant file system.

[2] High Energy Physics

### 2.2.3 Data and Storage Management

Identified as a mandatory feature is the ability to organize the storage media into levels and groups, in order to:

- define the set of media where related datasets will be stored – building the relation between set of volumes and users and/or dataset type.
- organize the internal HSM migration, where datasets are moved between different set of volumes based on administrator defined policies.

### 2.2.4 Real-Time Operations

This requirement comes from CERN and the meteorological (HNMS) group in order to allow online data dumping applications to write new data with a contiguous high data rate. This relates to features EuroStore implements in the PVL (Physical Volume Library) component of the HSM.

### 2.2.5 System and User Administration

In order to configure and share the (often) rare physical resources – the physical storage environment – between different user communities with different needs and priorities, the HSM needs to support mechanism to partition these resources logically. In addition the HSM shall implement priorities and some sort of pre-selected storage devices to user groups and/or applications.

Another required feature is quotas for the primary and secondary storage on a per user basis.

The primary administration tool should use the WWW (HTTP) system for GUI based administration tasks. In addition, some end-users require the ability to use command line tools in order to run shell script based administration tasks.

### 2.2.6 Reliability and Security

All end-users require a stable, robust and continuously running system with no need to shutdown the system in order to perform various administration and configuration tasks. Due to the fact that many separate processes (probably on different heterogeneous computer systems) making up EuroStore need to talk to each other, the network communication needs to be secure (encrypted) and safe. The various parts of persistent meta data shall be managed in a way that allows easy recovery and backup operations.

### 2.2.7 User Support and Statistics

To help the administrative tasks, the system has to provide a versatile range of statistical data covering all possible internal and external operations. As usual, the end-users require a complete and sufficient set of documentation in order to operate the system. It should be noted here that some end-users explicitly asked not to try to extend or circumvent features and limitations imposed by the underlying platform (i.e. removing limitations of the underlying file system).

### 2.2.8 Import and Export Mechanism

Most end-users with an already existing Mass Storage environment require data import and export mechanisms to allow bulk data movement without copying the real data. Most of this requirement relies on an open standard for meta data which could be transferred between different types and instances of HSMs.

### 2.2.9 Hardware and Software Support

Simply said: EuroStore should run on almost all hardware and software systems available today.

In order to run EuroStore with a wide range of storage device types, the system shall define the proper interfaces in order to adapt to new or different storage devices. This is also valid for the robotics (libraries).

The media format shall be an open standard which makes offsite access easier and keeps the administrators happy in the case of disaster recovery activities. Disaster recovery generates the requirement to produce self describing storage media.

### 2.2.10 Operational Environment

The system shall support various existing technologies and methods to access the data. Network file systems shall run without modifications. All types of database systems should run with no difference compared to other classical environments.

In the following design description you will see which of the above end-user requirements can be met by the first stage of the EuroStore project. The remaining requirements have to be targeted in a future stage of the project or by the partner doing the commercial exploitation.

## 3 Basic Abstract Design Goals

Due to the limited resources (in time and number of developers) we had to focus our activities on key components and functionality. The components on which EuroStore is based on are limited to:

- the parallel file system (PFS) running on CS2 systems today

- current experience and know-how in building layered and parallel file systems.
- current experience and know-how of HSM development and usage

In order to get feedback from the end-users the prototype is expected to be installed at CERN in March 1999.

The basic structure of EuroStore is classical and divided into two major building blocks[3] with minimal interference. The first building block is the parallel file system PFS, being the interface to the end-user applications as a POSIX file system. The second building block is the HSM managing and controlling the secondary storage environment. The interface between both blocks is simple and implemented as a C based API library for the HSM used by PFS. The PFS development site is QSW in Bristol/Great Britain and the HSM will be developed at DESY in Hamburg/Germany. This scheme allows a very efficient and frictionless cooperation between the two development partners.

The project scope and resources limits the results, so that the first stage of the EuroStore project will concentrate on basic functions of the PFS plus HSM. The primary implementation platform will be Sparc/Solaris, whereas the PFS will be ported to Digital UNIX within the scope of the project.

### 3.1 PFS

PFS will be extended in numerous directions to cope with the requirements of EuroStore. The following objectives summarize the goals:

- interface to the HSM by adding migration features, space management and user tools to control explicit migration
- add RAS capabilities
- enhance management and administration capabilities
- enhance portability – test with the port to Digital UNIX.
- evaluate and test the implementation of software based RAID5 features
- performance enhancements

The last item will be based on the development of a new global locking mechanism. The portability issue will introduce some re-working to be platform independent as much as possible.

[3] as shown in figure 1

### 3.2 HSM

The HSM goals are described by the terms simple, smart and efficient. The HSM needs to meet the performance requirements but doesn't try to support more single stream performance than the underlying hardware can give. This will definitely exclude support for parallel tape access or other related techniques. The focus is the support for hundreds of concurrent data streams with a moderate to high data rate (depending on network and storage hardware). The term 'simple' focuses on the manageability of the HSM and the scaling of administration efforts with the size of the installation. From the software implementation point of view, we try to avoid dependencies on third party products as much as possible. As of today, the only third party product required by the HSM is the object oriented database system for storing and managing the HSM internal meta data. Due to the scaling requirement (which is the most important one) the basic design of the HSM is network centric and in line with the IEEE reference model.

## 4 PFS Structure

PFS is a System V layered file system. It can basically run on any platform supporting layered file systems and thereby manage the data distribution over multiple nodes and disks. PFS is built out of two major components, first, the so called map file system and second, the underlying data file system. The map file system stores structural information, distribution information and file attributes. In addition the map file system will also manage the migration state of the file and the necessary migration meta data in order to recall the file if requested. The data file system can be a local (i.e. UFS) file system or a remote NFS mounted file system. Each regular PFS file is represented by one entry in the map file system and one or more (could be all) entries in the data file systems, depending on the stripe configuration. The directory structure of the PFS is directly represented within the map file system, but the regular files there only contain information on how to access the stripes of the PFS files stored in the data file systems. The directory structure of the data file systems bears no relationship to that of the PFS; it is only there to speed lookup on the data file systems.

### 4.1 Proposed changes

During the EuroStore project the directory information on the map file system will be redistributed on the data file systems on a per directory base. In

addition PFS will be enhanced by the following features within the EuroStore project:

- support for MPI File I/O
- mmap support
- concurrency control with global lock manager
- backup and restore capabilities
- RMS integration

Figure 2 shows the new proposed structure of the PFS. It eliminates the separate map file system by using

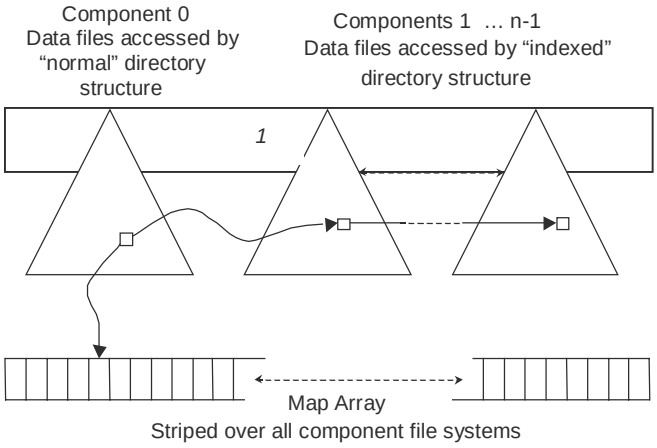


Figure 2: PFS basic overview

the first component file system for the directory structure and storing mapping information in a separate array striped over all the component file systems. The first component file system looks rather like the map file system of old PFS. However a regular file found here is empty, unless the corresponding PFS file has data striped on this component file system. The *inode* number of the file is used to index into the map array. The mapping information found there describes how the PFS file is striped over the component file systems, and the same *inode* number is used find the remaining data files in the other component file systems.

With this scheme, a PFS file requires a maximum of one *inode* on each component file system. The space overhead is the fixed at approximately 128 bytes per PFS file; the entry in the map array and I/O to read and update the mapping information is distributed over all component file systems.

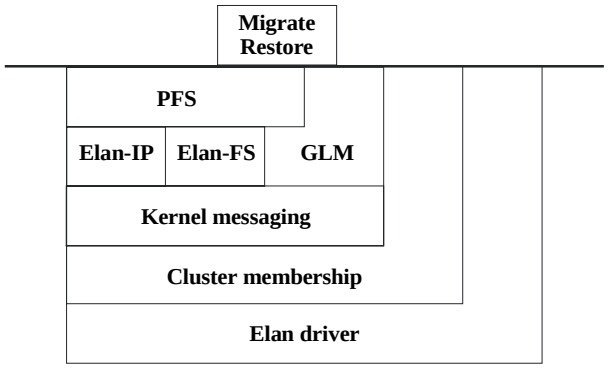


Figure 3: PFS layers

Figure 3 shows the layers of and around PFS. Elan is the name of the high speed low latency network which can be used to built MPP architectures and the parallel file system PFS.

### 4.2 Administration

The administration tasks on the PFS will be performed using a JAVA/Applet based environment, allowing all JAVA enabled WEB-browsers to be used for this purpose.

### 4.3 HSM Integration

The HSM will provide a simple C based API library in order to create, read and remove files in the HSM. PFS will be added by an external process using this API, the so called migration daemon. This process receives messages from the kernel level PFS code and starts and controls the data transfer from/to the HSM. The initial data transport vehicle will be a TCP stream, whereas the API will also allow other transport mechanism in the future. In the first stage of the EuroStore project the HSM API will only support sequential access of complete files. Again – Future stages of the EuroStore project will allow partial reads (specifying offset and length of data to read).

## 5 HSM

The HSM will have a network centric design following the IEEE reference model. The complete HSM service is built out of sub-services implemented as separate processes or threads running on the same or different heterogeneous platforms. The communication

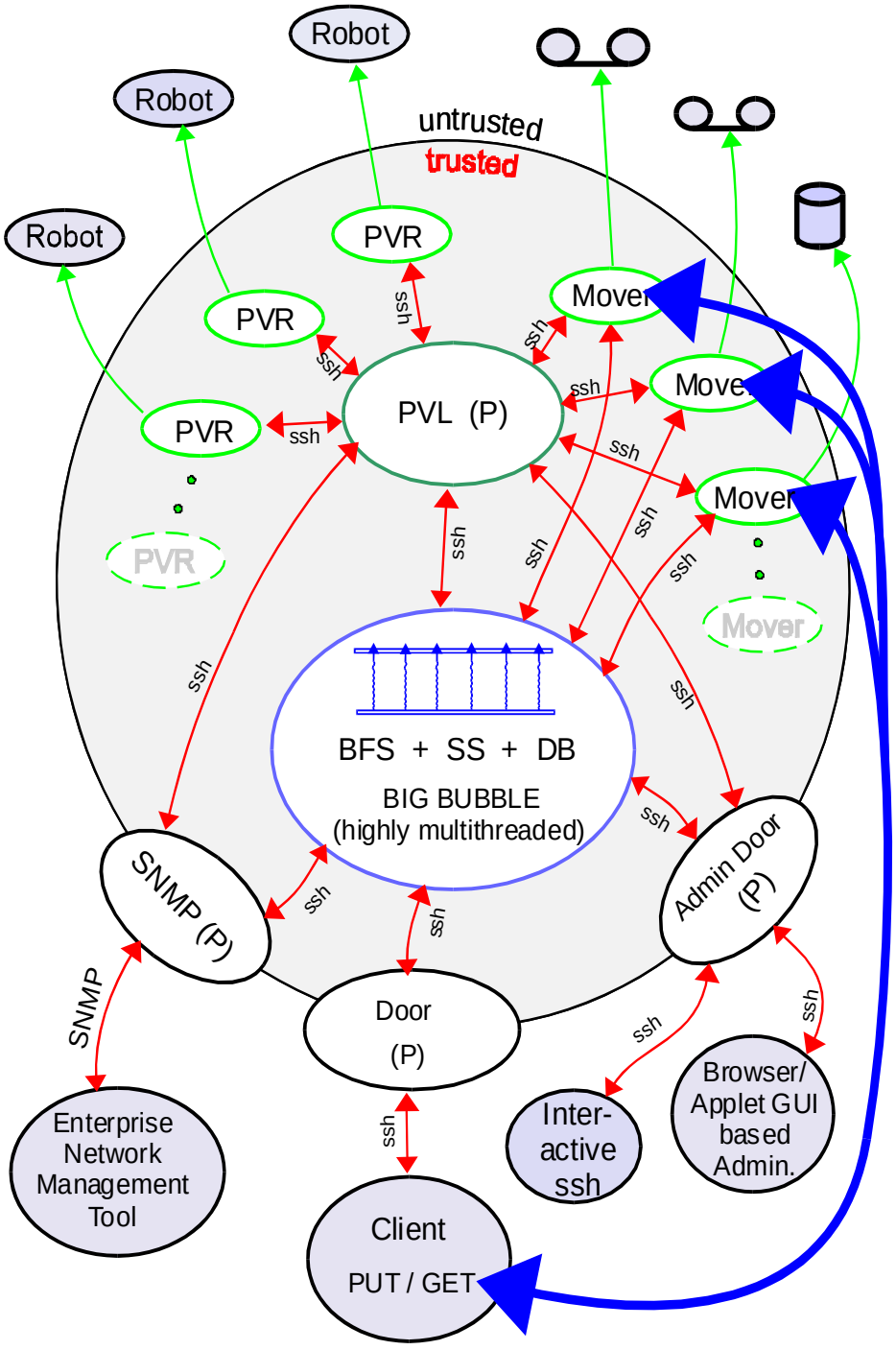


Figure 4: HSM Structure

mechanism between all these sub-services is done with a secure message passing environment (called Cell-Communication). In order to support heterogeneous platforms and to implement the HSM in an object oriented fashion we select JAVA as the implementation language. Only the PVR, which will be highly library vendor specific, and parts of the Mover code will use C code. The Cell-Communication environment allows the secure communication of pure C code processes with JAVA virtual machines. Of course, this combination can not exchange JAVA objects and uses only ASCII strings to avoid problems using heterogeneous platforms. Figure 4 shows the overall design of the HSM. Not shown are the possible configuration options in order to achieve the appropriate scaling effects.

### 5.1 Internal Object Separation

The HSM supports the notion of Storage Groups to allow a single Store to be divided into several sub-domains containing specific user groups and/or dataset types. The Store represents the root of the internal HSM object structure, whereas the Store is built out of Storage Groups. The Storage Group is further subdivided into Volume Sets which act as the source and destination for the HSM internal migration of datasets. The Volume Set is itself built out of Volume Containers defining the set of physical volumes belonging to a single physical library. To describe and control the internal HSM migration there exists an object, called Migration Path, which encloses the migration condition and the source/destination Volume Set. Each dataset stored in the HSM has a link to an existing Migration Path describing the dataset migration characteristics.

### 5.2 Interfaces

The HSM provides a simple service to the PFS (or other clients), namely storing and retrieving complete datasets (or files in the PFS nomenclature) sequentially. A future version of the EuroStore HSM might support read operations on parts of datasets (partial reads). This simplicity is mirrored in the data access API in that it contains only 3 functions: create/write a dataset, read an existing dataset and remove an existing dataset. In addition, the API will support simple query operations (ask for all files on a given volume, etc.) for its clients (like PFS). The data access API is implemented as a C based thread safe library.

### 5.3 Structure of HSM components

Figure 4 shows the EuroStore HSM components and how they interact. The big gray shaded area enclosing the central components identifies the trusted or authenticated area. All messages inside are pre-authenticated by the components located on the border line. These components, namely the Door, Admin Door and the SNMP authenticate incoming messages and forward them to the final destination component. All the ssh labeled (red) arrows indicate a secure (ssh based) network communication path. In the following I will briefly describe the functions implemented by the HSM components in Figure 4.

#### 5.3.1 Door – Admin Door – SNMP

These components are agents and the only direct network reachable components in the system. All messages from the outside world will be passed via these components (processes). Their primary function is to authenticate users/hosts and set up a secure channel between both worlds. In addition the SNMP components will translate SNMP messages into internal messages. The SNMP component allows any enterprise network management tool to query the HSM. It also supports SNMP traps. The other two Door components are doing minimal translation of messages between the internal and the external world. The separation between the main Door and the Admin Door is made in order to avoid problems reaching the system in the case of heavy load on the main Door. The main Door will have a separate TCP connection for each client contacting the HSM, so is a possible candidate for running into TCP connection limits induced by the underlying operating system. Beside this, both Doors implement similar functions.

#### 5.3.2 BB – Big Bubble

The BB is the central component representing the instance of a Store. It implements the functions of a bitfile server (BFS) and storage server (SS). To a very high degree, both server components are mainly database operations. For every new request entering the system, BB will start a new thread which performs all necessary tasks before the request is being forwarded to the PVL and the thread dies (stops).

#### 5.3.3 PVL/PVR

The PVL/PVR components will implement the classical functions outlined in the IEEE reference model. In order to cope with the user requirements the PVL contains numerous features not available in many other implementations.

- priorities – specified by the client application. The priorities age over time and aging stops at a configurable limit. This allows the administrator to configure the HSM in a way to handle more important requests with a minimal latency.
- Configurable number of write operations on a given Volume Set allowing the administrator to choose between two excluding characteristics:
  - datasets will be in strict chronological order on the storage media, but the number of concurrent write operations to that Volume set is limited to one.
  - the PVL will pick any suitable physical media from the Volume Set and allow write operations. This PVL selection is based on the current situation (mounted/allocated drives and media) of its managed libraries and drives. This option allows concurrent writes but datasets won't be in chronological order on the media.
- Regular expressions assigned to a storage device (drive). The PVL will manage a defined set of variables (most of them are request dependent) which can be used to build the regular expression. The expression will be evaluated by the PVL each time a new drive scheduling sequence starts. This feature allows a very flexible way to pre-select fixed drives for a given class of requests, i.e., allow only write requests from a given hostname and user on the assigned drive.
- Virtual library partitioning allows partitioning a single resource (the library with the attached drives) among all served clients (Stores). This is in contrast to the static behavior of the regular expression option, because no explicit drive will be partitioned among clients versus saying "30% of the total resources are guaranteed for the client".

#### 5.3.4 Mover

The Mover's task is to contact the client (through a dedicated connection, which must not be a TCP connection) and send or receive the data. The Mover will also accept commands from the PVL to execute load/unload and other similar operations.

### 5.4 Administration

There are two ways to achieve administration task connections. The first one allows a standard interactive ssh application to log into the Admin Door. The Admin Door implements a shell like interface to execute simple ASCII command line operations. The second option use the WEB (HTTPS) interface to send secure (signed) JAVA/Applet code to the client JAVA enabled browser. The Applet code connects to the Admin Door using the ssh protocol.

### 5.5 Scaling Options – Configuration

The HSM allows a wide range of configurations in order to support the wide range of end-user site requirements. The minimum configuration consists of a Door – BB – PVL – PVR – Mover and Admin Door whereas the resource consumption of all these HSM components should be low enough to allow them to run on the same system. For larger end-user environments the HSM can be scaled up in the following ways:

- A single Door can serve one or more BB's (representing the Store instance).
- A single PVL can serve one or more BB's.
- A single BB can use one or more PVL's.
- A single PVL can manage one or more different libraries served by the appropriate PVR.
- Movers can be installed appropriate to the number of available storage devices.
- A single Mover can be used by one or more BB's.
- A single Admin-Door and SNMP process can serve the complete domain (all BB's, PVL's, PVR's, etc. in the complete domain of the end-user (i.e. a laboratory)

These configuration possibilities shall allow the administrators to configure the HSM according to their storage requirements and available hardware.

### 5.6 Secure Message Passing environment

Developed as a pre EuroStore component, the secure message passing environment, called Cell-Communication, is the basis for all interprocess message communication. The system implements secure network channels using the open SSH protocol allowing other SSH based applications to connect to the HSM components. The Cell Communication encloses the following features:

- asynchronous message passing (JAVA objects)
- encryption through network paths

- key exchange and connection initialization based on the SSH protocol
- ciphers available: DES/IDEA/BLOWFISH
- 100% JAVA code
- source routing
- routing scheme similar to IP on UNIX systems
- global routing management module based on well known Cell addresses (broker)
- addressing by name
- sender buffering to overcome small network glitches
- modules for
  - ssh login (interactive ssh application)
  - telnet login
  - SNMP

In addition, the C based client code allows standard C based applications to connect to the JAVA world to send ASCII messages (not JAVA objects in the case of a JAVA to JAVA message).

## 6 Conclusions

We believe that our approach, combining a parallel file system, extended by migration features, with a lightweight but highly scalable HSM, fulfills the requirements of scientific and commercial applications of today and tomorrow. The short project lifetime is possible because the parallel file system is well understood, and the existence of the object oriented JAVA environment allows fast, reliable new implementation schemes.

---

1 Building a Database for the LHC – the Exabyte Challenge, Jamie Shiers CERN – Proceedings 6. NASA and 15 IEEE Mass Storage conference March 1998