Storage Resource Managers: Recent International Experience on Requirements and Multiple Co-Operating Implementations

Authors: Large International Collaboration

Presenter: Timur Perelmutov, Fermilab

http://sdm.lbl.gov/srm-wg

Part I

Background

International Collaboration Participants (Co-Authors)

CERN, Swizerland: Lana Abadie, Paolo Badino, Jean-Philippe Baud, Flavia Donno, Ákos Frohner, Birger Koblitz, Sophie Lemaitre, Maarten Litmaath, Giuseppe Lo Presti, Rémi Mollon, David Smith, Paolo Tedesco

DESY, Germany: Patrick Fuhrmann, Tigran Mkrtchan

ICTP/EGRID, Italy: Ezio Corso, Massimo Sponza

INFN/CNAF, Italy: Alberto Forti, Luca Magnoni, Riccardo Zappi

IN2P3, France: Gilbert Grosdidier

Fermilab, USA: Matt Crawford, Dmitry Litvinsev, Alexander Moibenko, Gene Oleynik, Timur Perelmutov, Don Petravick

LBNL, USA: Junmin Gu, Vijaya Natarajan, Arie Shoshani, Alex Sim

Rutherford Lab, England: Shaun De Witt, Jens Jensen

History (1)

- 7 year of Storage Resource Management (SRM) activity
- Experience with system implementations v.1.1 (basic SRM) – 2001
  - MSS
    - HPSS (LBNL, ORNL, BNL)
    - dCache/Enstore (Fermi, DESY)
    - JasMINE (Jlab)
    - Castor(CERN)
    - MSS (NCAR)
  - Disk systems:
    - DPM (CERN)
    - DRM (LBNL)
    - dCache(DESY, Fermi)
- SRM v2.0 (enhanced SRM) spec was finalized – 2003

History (2)

- SRM V2.2 – enhancements introduced after WLCG (the World-wide LHC Computing Grid) adopted SRM standard
  - Several implementations of v2.2 completed or in-progress
  - extensive compatibility testing
  - MSS: HPSS (LBNL), dCache/{Enstore,TSM,OSM,HPSS} (Fermi,DESY), JasMINE (Jlab), CASTOR (CERN, RAL)
  - Disk systems: dCache(Fermi,DESY), DPM (CERN), StoRM (Italy), BeStMan (LBNL)
- Open Grid Forum (OGF)
  - Started Grid Storage Management (GSM-WG): GGF8 – June 2003
  - Last SRM collaboration meeting – Sept. 2006
  - SRM v2.2 spec (for OGF) submitted – August 2007

What are SRMs

<u>Definition</u>

SRMs are middleware components

whose function is to provide dynamic

space allocation
file management in spaces

for shared storage components
on the Grid

Motivation & Requirements

- **Grid architecture needs to include reservation & scheduling of:**
  - Compute resources
  - Storage resources
  - Network resources
- **Storage Resource Managers (SRMs) role in the data grid architecture**
  - Shared storage resource allocation & scheduling
  - Specially important for data intensive applications
  - Often files are archived on a mass storage system (MSS)
  - Wide area networks – need to minimize transfers by file sharing
  - Scaling: large collaborations (100’s of nodes, 1000’s of clients) – opportunities for file sharing
  - File replication and caching may be used
  - Need to support non-blocking (asynchronous) requests

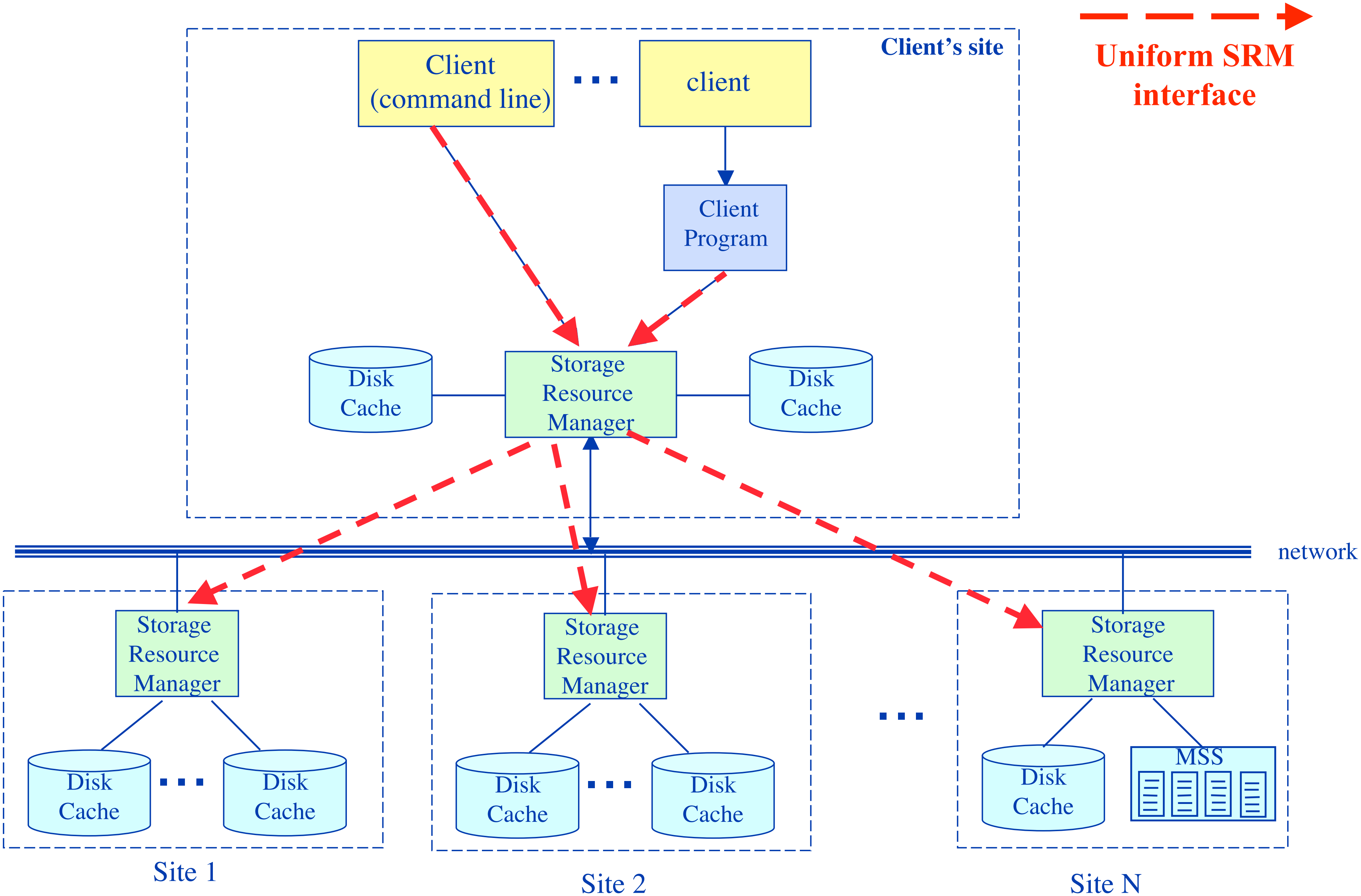
Motivation Justification (1)

- Suppose you want to run a job on your local machine
  - Need to allocate space
  - Need to bring all input files
  - Need to ensure correctness of files transferred
  - Need to monitor and recover from errors
  - What if files don't fit space? Need to manage file streaming
  - Need to remove files to make space for more files
- Now, suppose that the machine and storage space is a shared resource
  - Need to do the above for many users
  - Need to enforce quotas
  - Need to ensure fairness of space allocation and scheduling

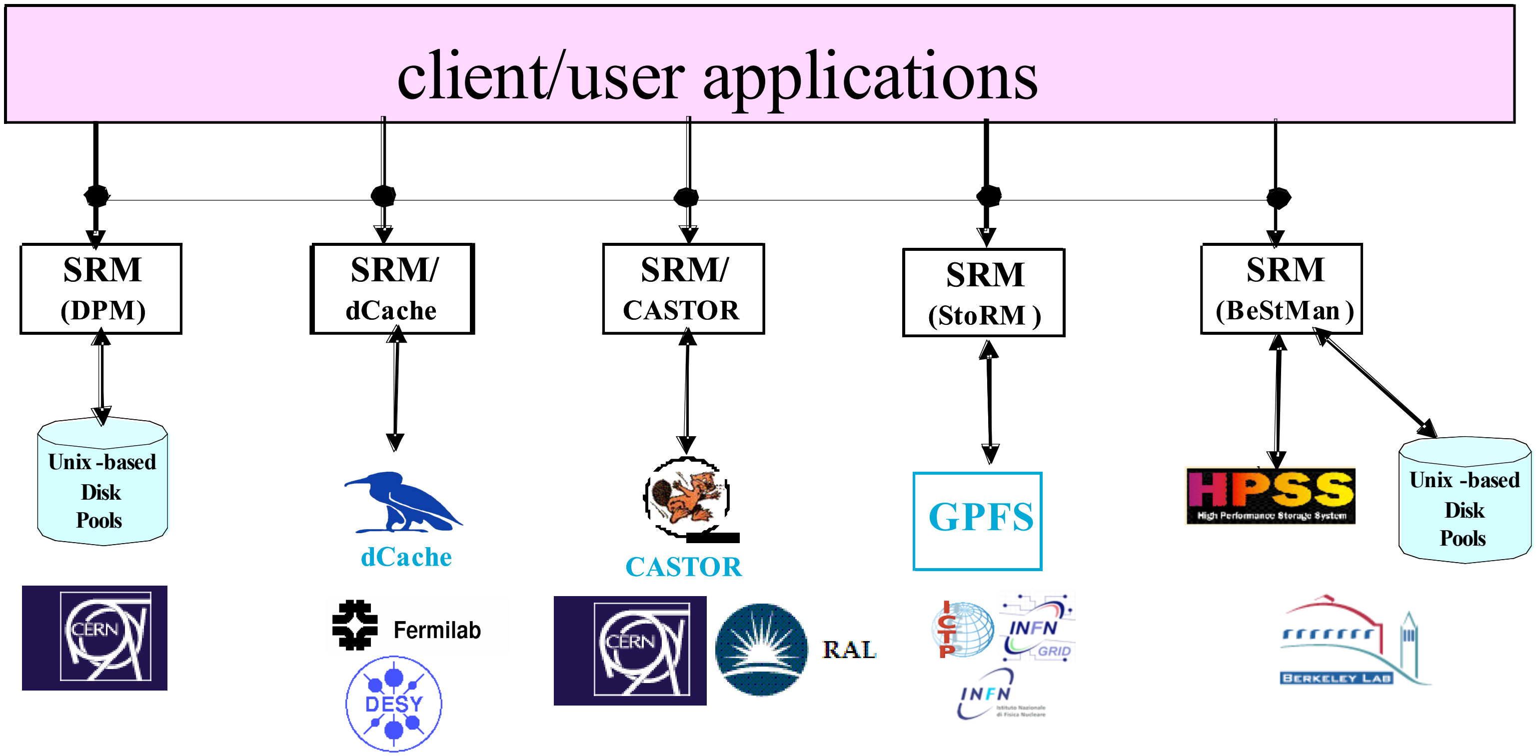
Motivation Justification (2)

- Now, suppose you want to do that on a Grid
  - Need to access a variety of storage systems
  - mostly remote systems, need to have access permission
  - Need to have special software to access mass storage systems
- Now, suppose you want to run distributed jobs on the Grid
  - Need to allocate remote spaces
  - Need to move (stream) files to remote sites
  - Need to manage file outputs and their movement to destination site(s)

Design Goal: Client and Peer-to-Peer Uniform Interface



Uniformity of Interface → Compatibility of SRMs



Multiple inter-operating SRM implementations. Clients can access different mass storage and file systems through a uniform SRM interface

Part II

Concepts

Storage Resource Managers: Main concepts

- Non-interference with local policies
- Advance space reservations
- Dynamic space management
- Pinning file in spaces
- Support abstract concept of a file name: site URL
- Temporary assignment of file names for transfer: Transfer URL
- Directory Management and ACLs
- Transfer protocol negotiation
- Peer to peer request support
- Support for asynchronous multi-file requests
- Support abort, suspend, and resume operations

Concepts: Site URL and Transfer URL

- Provide: Site URL (SURL)
  - URL known externally – e.g. in Replica Catalogs
  - e.g. srm://ibm.cnaf.infn.it:8444/dteam/test.10193
- Get back: transfer URL (TURL)
  - Path can be different than SURL – SRM internal mapping
  - Protocol chosen by SRM based on request protocol preference
  - e.g. gsiftp://ibm139.cnaf.infn.it:2811//gpfs/dteam/test.10193
- One SURL can have many TURL
  - Files can be replicated in multiple storage components
  - Files may be in near-line and/or on-line storage
- In light-weight SRM (a single file system on disk)
  - SURL can be the same as TURL except protocol
- File sharing is possible
  - Same physical file, but many requests
  - Needs to be managed by SRM

Concepts: Types of Storage and spaces

- Access latency
  - On-line
    - Storage where files are moved to before their use
  - Near-line
    - Requires latency before files can be accessed
- Retention quality
  - Custodial (High quality)
  - Output (Middle quality)
  - Replica (Low Quality)
- Spaces can be reserved in these storage components
  - Spaces can be reserved for a lifetime
  - No limit on number of spaces
  - Space reference handle is returned to client
  - Total space of each type are subject to SRM and/or VO policies
- Assignment of files to spaces
  - Files can be assigned to any space, provided that their lifetime expiration is shorter than the lifetime expiration of the space

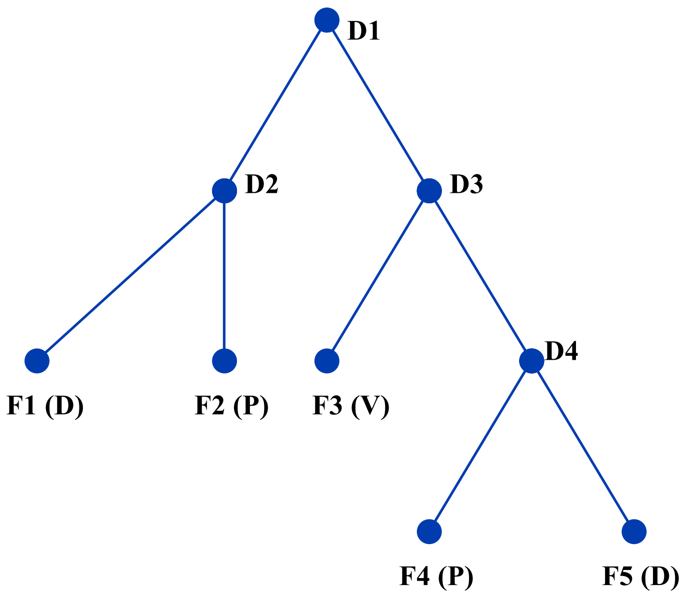
Concepts: managing spaces

- Default spaces
  - Files can be put into an SRM without explicit reservation
  - Defaults are not visible to client
- Files already in the SRM can be moved to other spaces by
  - srmChangeSpaceForFiles
- Files already in the SRM can be pinned in spaces
  - By requesting specific files (srmPrepareToGet)
  - By pre-loading them into online space (srmBringOnline)
- Updating space
  - Resize to request more space or to Release all unused space
  - Extend or Shorten the lifetime of a space
- Releasing files from space by a user
  - Release all files that user brought into the space whose lifetime has not expired
  - Move permanent and durable files to near-line storage
  - Release space that was used by user

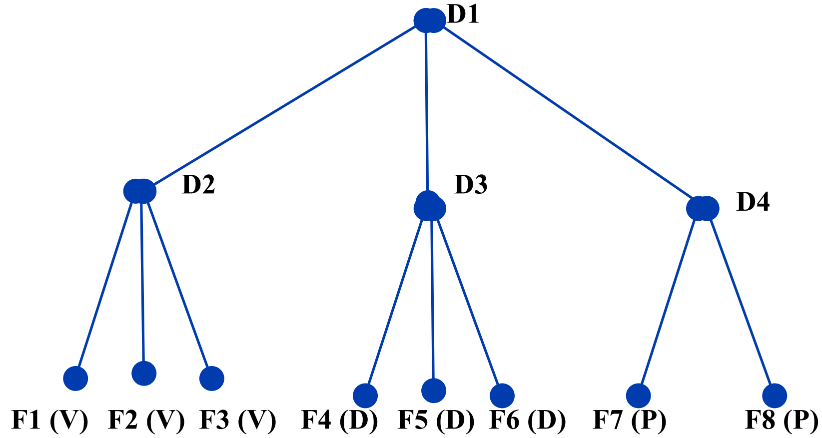
Concepts: Directory Management

- Usual unix semantics
  - srmLs, srmMkdir, srmMv, srmRm, srmRmdir
- A single directory for all spaces
  - No directories for each file type
  - File assignment to spaces is virtual
- Access control services
  - Support owner/group/world permission
    - ACLs supported – can have one owner, but multiple user and group access permissions
    - Can only be assigned by owner
    - When file requested from a remote site, SRM should check permission with source site

Examples of Directory Structures (user defined)



(1) Mixed expiration modes



(2) By expiration mode

- **Advantage**: no need to move a file when file expiration mode is changed, provided that file lifetime is not longer than space lifetime

Concepts: Space Reservations

- **Negotiation**
  - Client asks for space: C-guaranteed, MaxDesired
  - SRM return: S-guaranteed <= C-guaranteed,
    best effort <= MaxDesired
- **Types of spaces**
  - Specified during srmReserveSpace
  - Access Latency (Online, Nearline)
  - Retention Policy (Replica, Output, Custodial)
  - Subject to limits per client (SRM or VO policies)
  - Default: implementation and configuration specific
- **Lifetime**
  - Negotiated: C-lifetime requested
  - SRM return: S-lifetime <= C-lifetime
- **Reference handle**
  - SRM returns space reference handle
  - Client can assign Description
  - User can use srmGetSpaceTokens to recover handles on basis of ownership

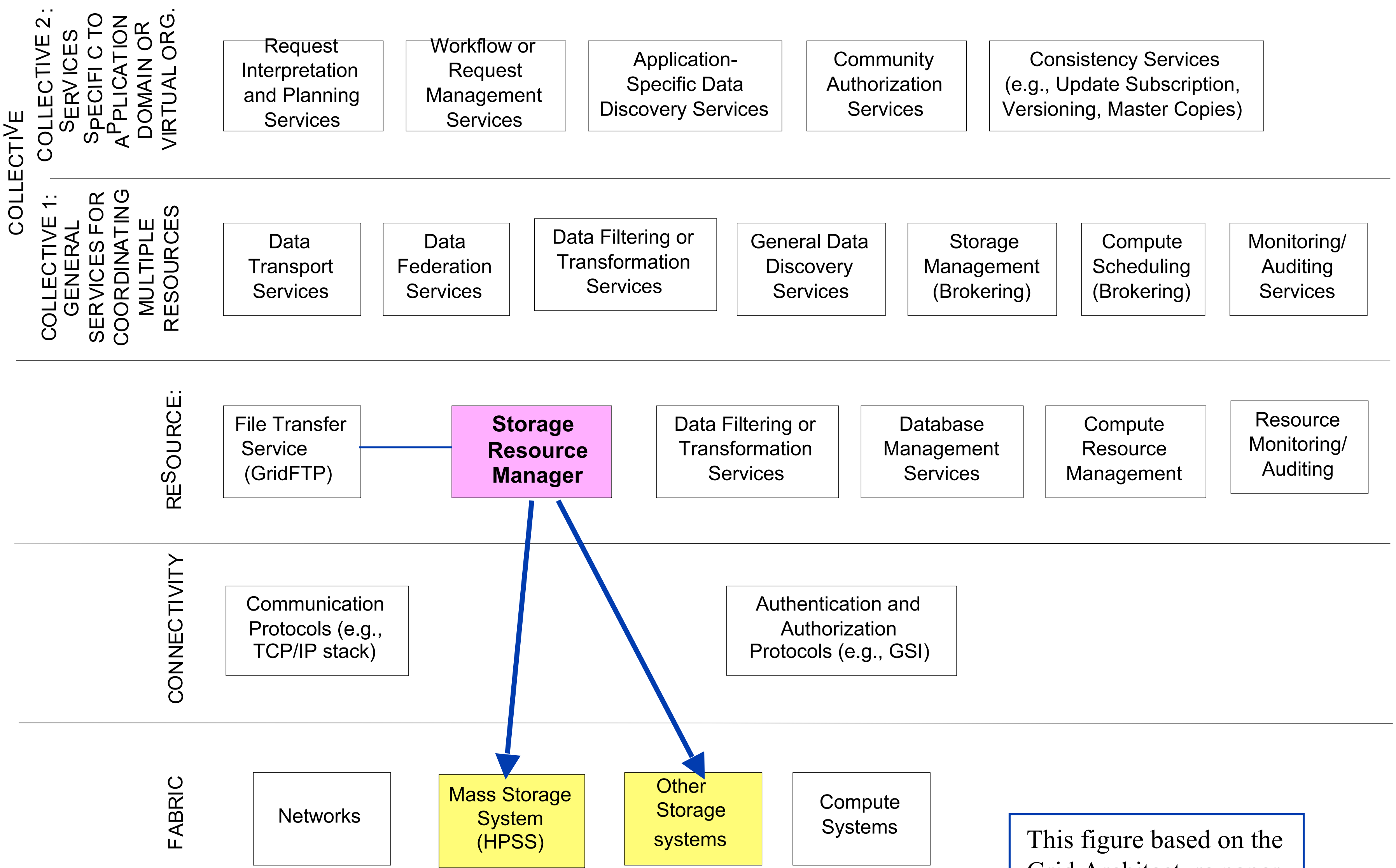
Concepts: Transfer Protocol Negotiation

- Negotiation
  - Client provides an ordered list of protocols
  - SRM return: first protocol from the list it supports
- Example
  - Protocols list: bbftp, gridftp, ftp
  - SRM returns: gridftp
- Advantages
  - Easy to introduce new protocols
  - User controls which protocol to use
- How it is returned?
  - The protocol of the Transfer URL (TURL)
  - Example: bbftp://dm.slac.edu/temp/run11/File678.txt

New Concepts for version 2.2

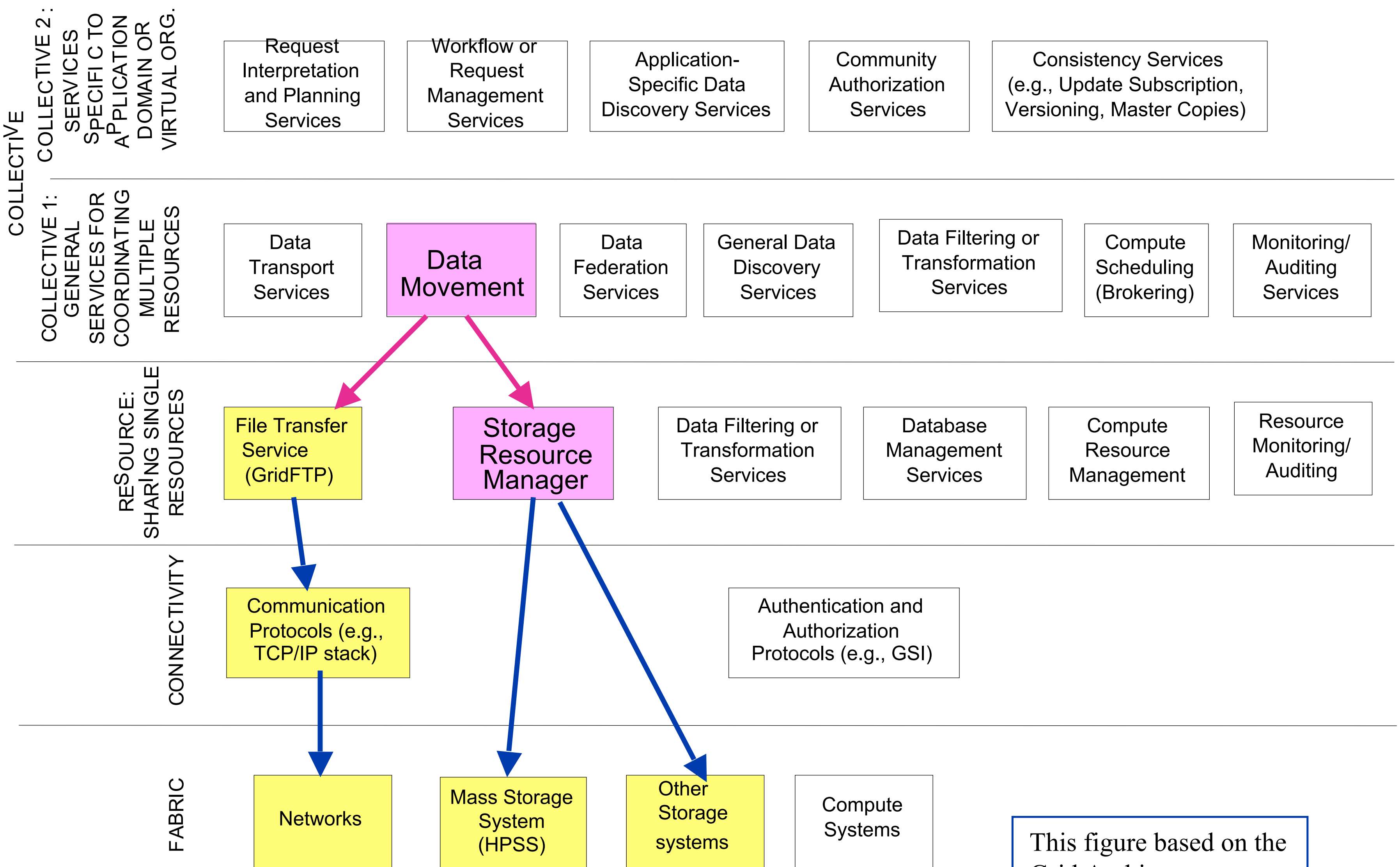
- **Composite Storage Element**
  - **Made of multiple Storage Components**
    - **e.g. typical: component 1: online-replica**
      component 2: nearline-custodial (with online disk cache)
    - **Another e.g.:component1: online-custodial**
      component 2: nearline-custodial (with online disk cache)
  - **srmBringOnline can be used to temporarily bring data to the online component for fast access**
  - **When a file is put into a composite space, the SRM may have (temporary) copies on any of the components**
- **Primary Replica**
  - **When a file is first put into an SRM, that copy is considered the primary replica**
  - **A primary replica can be assigned a lifetime**
  - **The SURL lifetime is the lifetime of the primary replica**
  - **When other replicas are made, their lifetime cannot exceed the primary replica lifetime**
  - **Lifetime of a primary replica can only be extended by an SURL owner.**

Where do SRMs belong in the Grid architecture?



This figure based on the Grid Architecture paper by Globus Team

SRMs supports data movement between storage systems



This figure based on the Grid Architecture paper by Globus Team

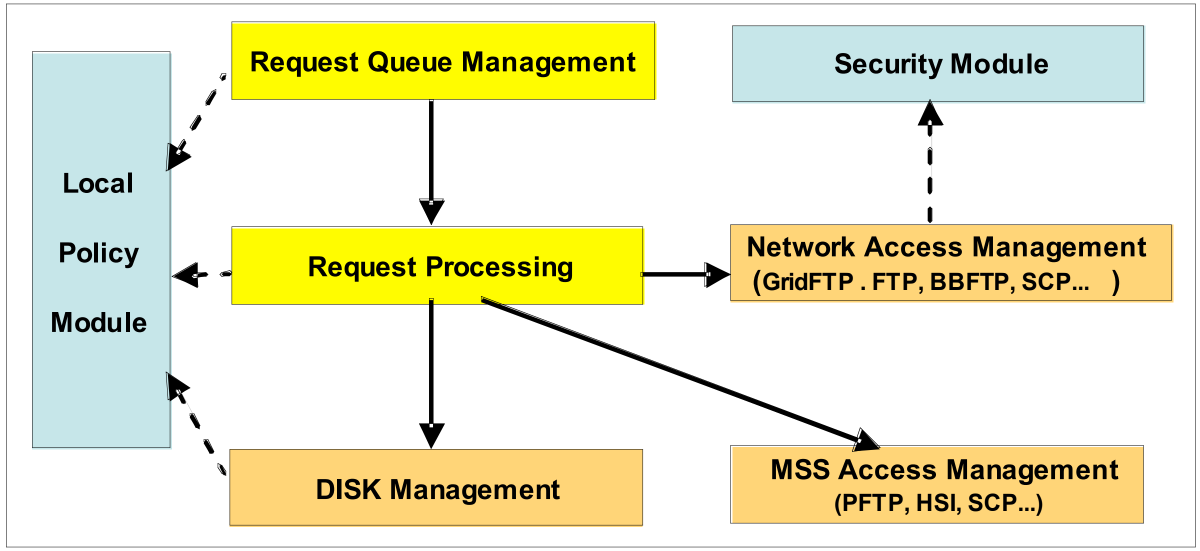
Part III

Implementations

(Alphabetic Order)

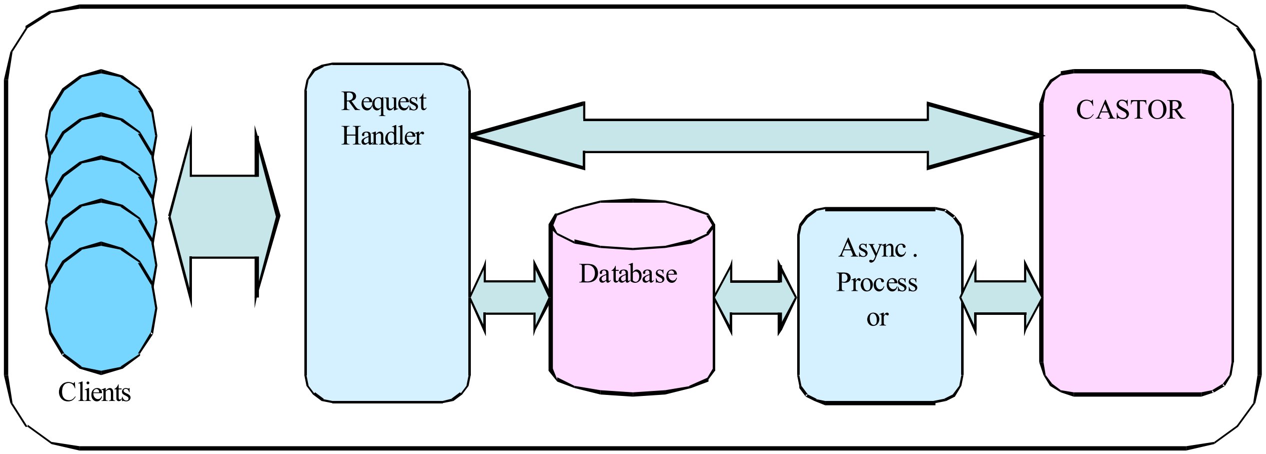
Berkeley Storage Manager (BeStMan)
LBNL

- Java implementation
- Designed to work with disk systems
- As well as MSS to stage/archive from/to its own disk (currently HPSS)
- Uses in-memory database (BerkeleyDB)



- Multiple transfer protocols
- Space reservation
- Directory management (no ACLs)
- Can copy files from/to remote SRMs
- Can copy entire directory robustly
  - Large scale data movement of thousands of files
  - Recovers from transient failures (e.g. MSS maintenance, network down)
- Local Policy
  - Fair request processing
  - File replacement in disk
  - Garbage collection

CASTOR-SRM
CERN and Rutherford Appleton Laboratory

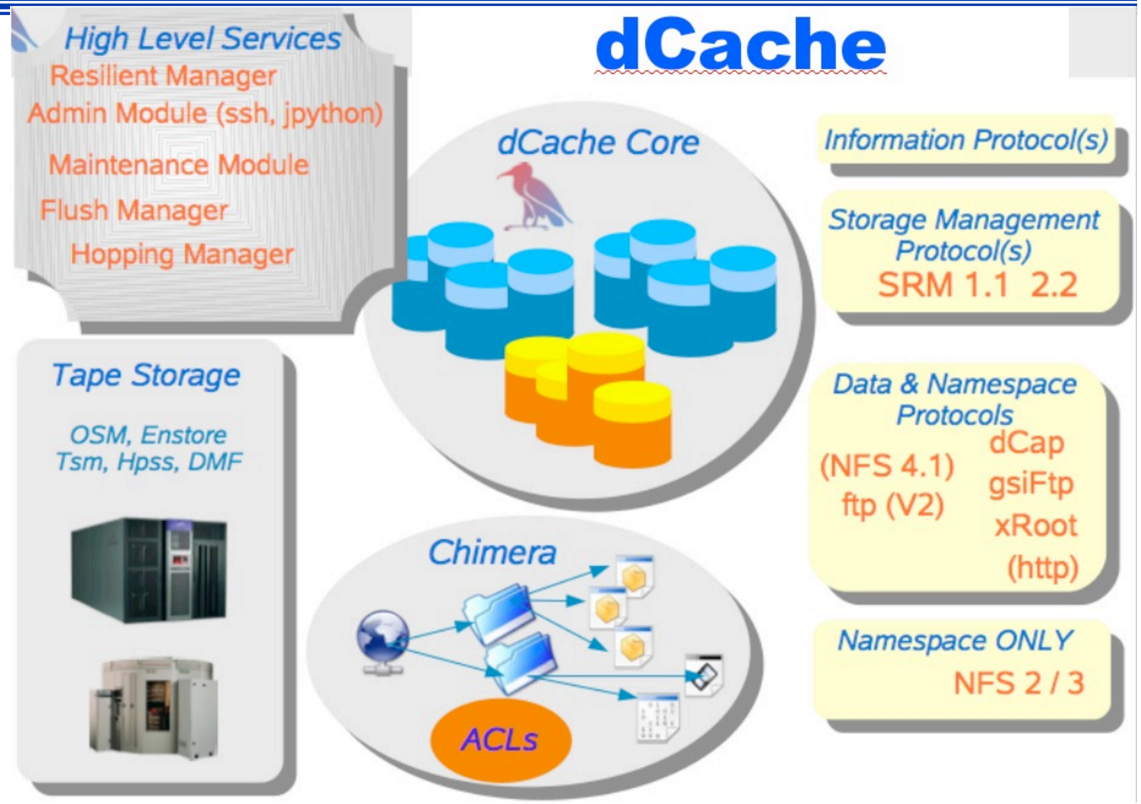


- **CASTOR is the HSM in production at CERN**
- **Support for multiple tape robots**
  - Support for Disk-only storage recently added
- **Designed to meet Large Hadron Collider Computing requirements**
  - Maximize throughput from clients to tape (e.g. LHC experiments data taking)
  - ***More on: G. Lo Presti et al., CASTOR, MSST Poster Session***

- **C++ Implementation**
- **Reuse of CASTOR software infrastructure**
  - Derived SRM specific classes
- **Configurable number of thread pools for both front- and back-ends**
- **ORACLE centric**
- **Front and back ends can be distributed on multiple hosts**

dCache-SRM
Fermilab and DESY

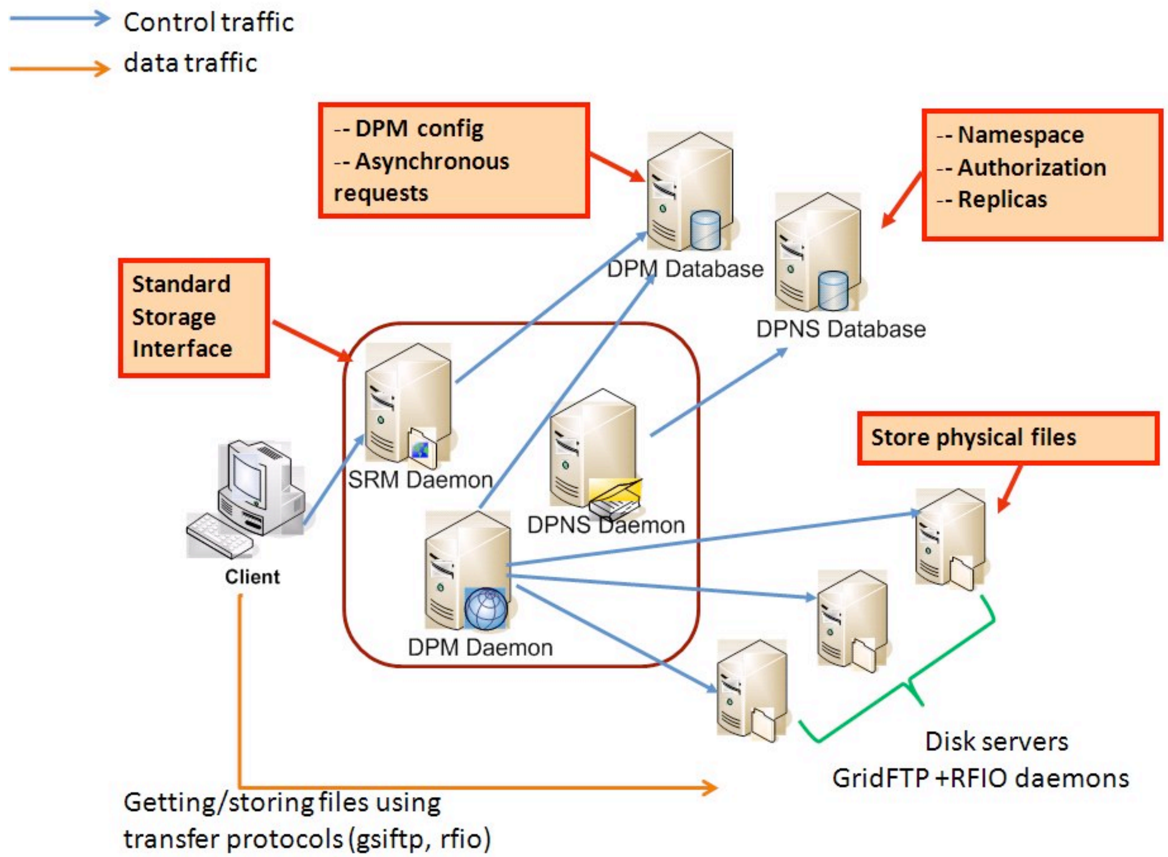
- Strict name space and data storage separation
- Automatic file replication on based on access patterns
- HSM Connectivity (Enstore, OSM, TSM, HPSS, DMF)
- Automated HSM migration and restore
- Scales to Peta-byte range on 1000's of disks
- Supported protocols:
- (gsi/krb)FTP, (gsi/krb)dCap, xRoot, NFS 2/3
- Separate IO queues per protocol
- Resilient dataset management
- Command line and graphical admin interface
- Variety of Authorization mechanisms including VOMS
- Deployed in a large number of institutions worldwide



- SRM 1.1 and SRM 2.2
- Dynamic Space Management
- Request queuing and scheduling
- Load balancing
- Robust replication using SrmCopy functionality via SRM, (gsi)FTP and http protocols

Disk Pool Manager (DPM)
CERN

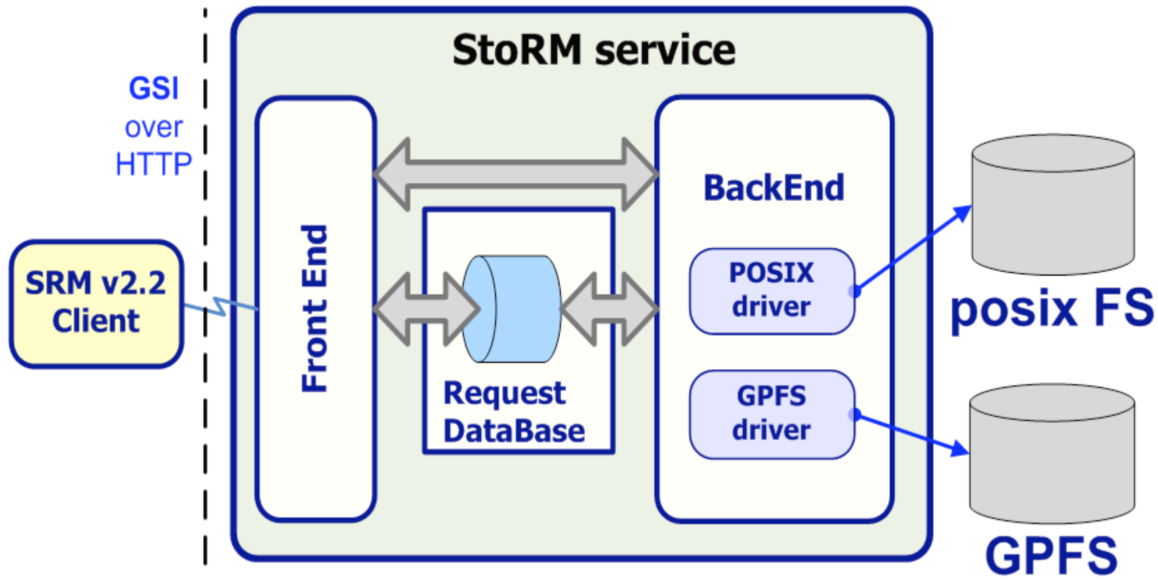
- **Manages storage on disks only**
- **Security**
  - GSI for authentication
  - VOMS for authorization
  - Standard POSIX permissions + ACLs based on user's DN and VOMS roles
- **Virtual ids**
  - Accounts created on the fly
- **Full SRMv2.2 implementation (srmCopy being done)**
- **Standard disk pool manager capabilities**
  - Garbage collector
  - Replication of hot files
- **Transfer protocols**
  - GridFTP
  - Secure RFIO
- **Easy to install and administer**



- **Supported database backends**
  - MySQL/Oracle
- **High availability**
  - All servers can be load balanced (except the DPM one)
  - Resiliency: all states are kept in the DB at all times

Storage Resource Manager (StoRM)
INFN/CNAF - ICTP/EGRID

- It's designed to leverage the advantages of high performing parallel file systems in Grid.
- Different file systems supported through a driver mechanism:
  - generic POSIX FS
  - GPFS
  - Lustre
  - XFS
- It provides the capability to perform local and secure access to storage resources (*file://* access protocol + ACLs on data).



StoRM architecture:

- Frontends: C/C++ based, expose the SRM interface
- Backends: Java based, execute SRM requests.
- DB: based on MySQL DBMS, stores requests data and StoRM metadata.
- Each component can be replicated and instantiated on a dedicated machine.

Part IV

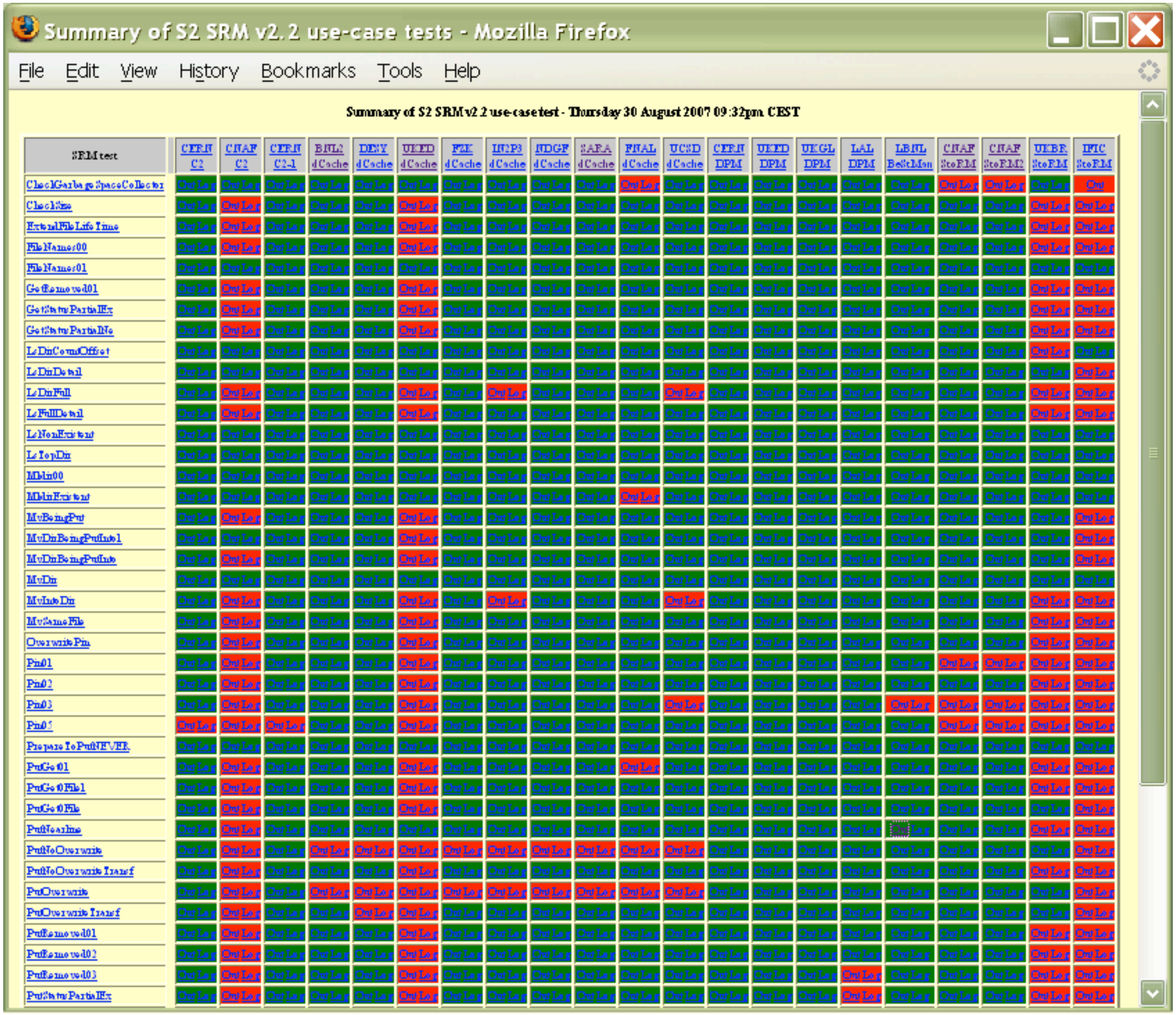
SRM Testing

- Adherence to standard
- Inter-operation

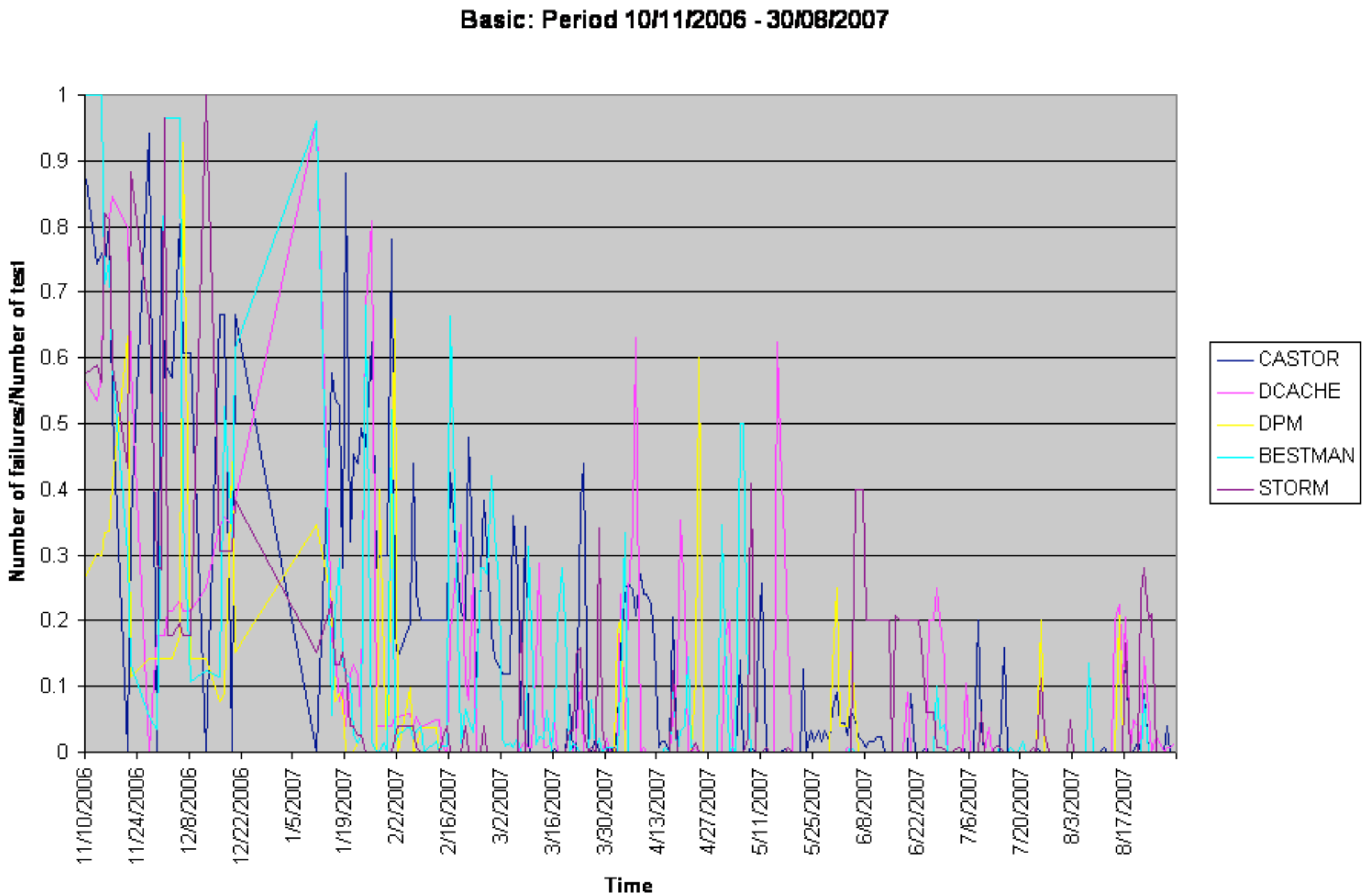
Five Types of Tests

- **Availability:** the srmPing function and a full put cycle for a file is exercised (srmPrepareToPut, srmStatusOfPutRequest, file transfer, srmPutDone). This family is used to verify availability and very basic functionality of an SRM endpoint.
- **Basic:** the equivalence partitioning and boundary condition analysis is applied to verify that an implementation satisfies the specification when it has a single SRM call active at any given time.
- **Use cases:** cause-effect graphing, exceptions, functional interference, and use cases extracted from the middleware and user applications are exercised.
- **Interoperability:** remote operations (servers acting as clients for some basic SRM functions) and cross copy operations among several implementations are executed.
- **Stress:** the error guessing technique and typical stress situations are applied to verify resilience to load.

The S2 result web-pages: Use-case tests

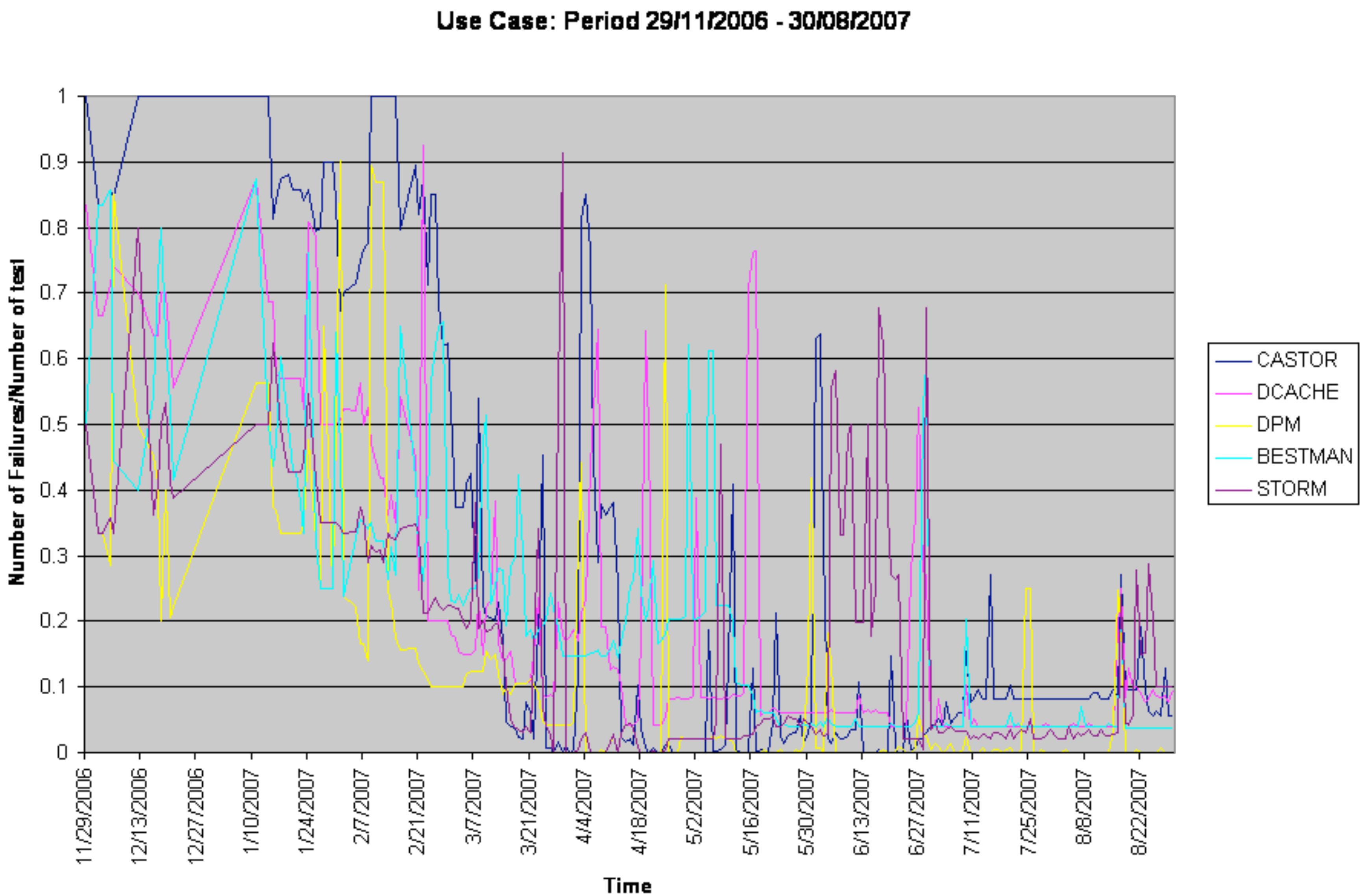


Testing Brought Improvements over time (1)



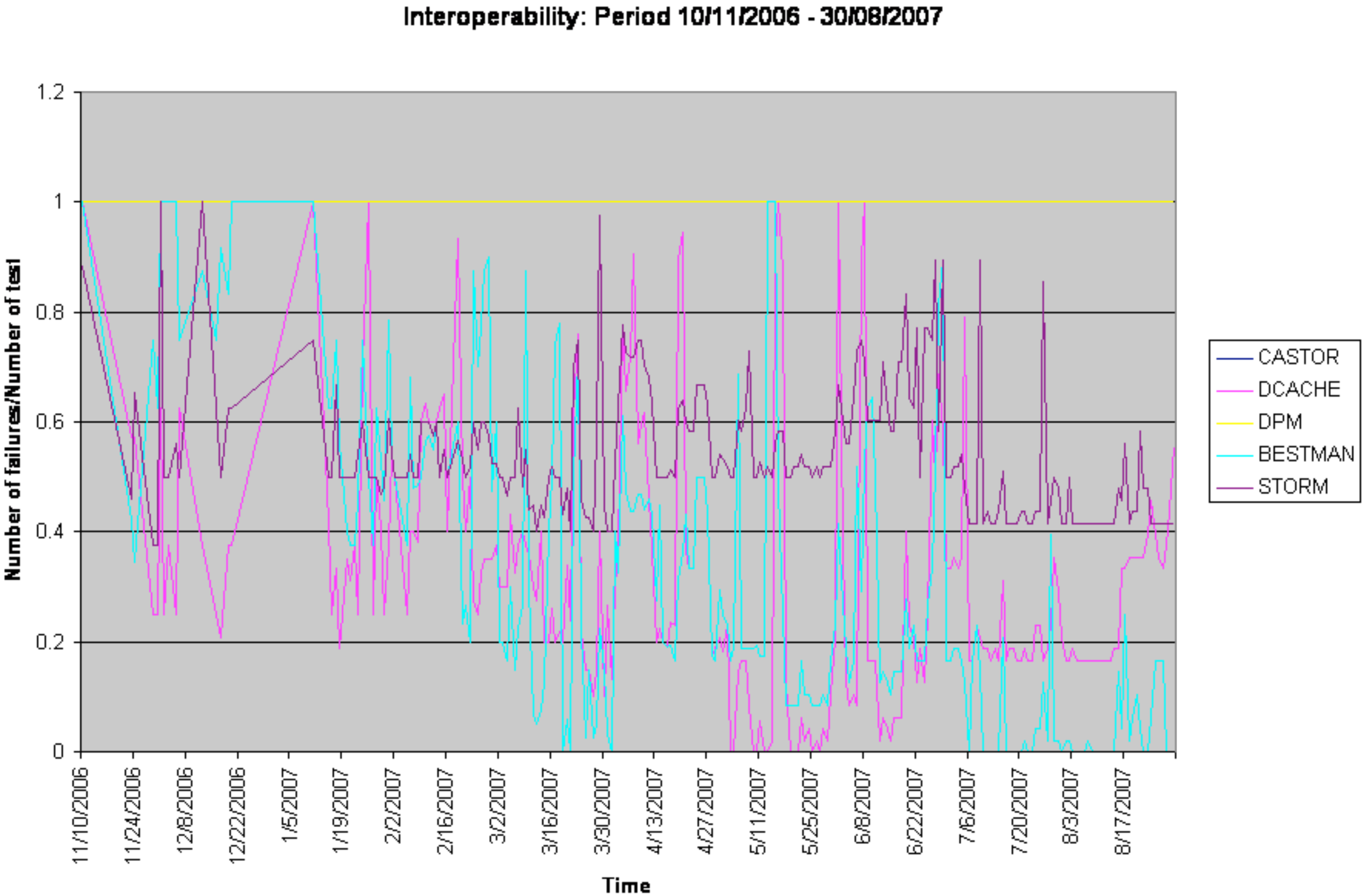
Basic test family: Number of failures/Number of tests over time

Testing Brought Improvements over time (2)



Use-case test family: Number of failures/Number of tests over time

Testing Brought Improvements over time (3)



Interoperability test family: Number of failures/Number of tests over time

Summary

- Storage Resource Management – essential for Grid
- SRM is a functional definition
  - Adaptable to different frameworks (WS, WSRF, …)
- Multiple implementations interoperate
  - Permit special purpose implementations for unique products
  - Permits interchanging one SRM product by another
- Multiple SRM implementations exist and are in production use
  - Data Grids for Particle Physics and other sciences
    - EGEE, OSG, …
  - Earth System Grid
  - Medicine
  - More coming …
- Cumulative experience in OGF GSM-WG
  - Specifications SRM v2.2 submitted

Extra Slides

SRM Methods

File Movement
srmPrepareToGet
srmPrepareToPut
srmRemoteCopy
srmBringOnline

Lifetime management
srmReleaseFiles
srmPutDone
srmExtendFileLifeTime

Terminate/resume
srmAbortRequest
srmAbortFile
srmSuspendRequest
srmResumeRequest

Space management
srmReserveSpace
srmReleaseSpace
srmUpdateSpace
srmChangeSpaceForFiles

Status/metadata
srmGetRequestStatus
srmGetFileStatus
srmGetRequestSummary
srmGetRequestID
srmGetFilesMetaData
srmGetSpaceMetaData

e.g. Space Reservation Functional Spec

```
srmReserveSpace
In:  String                    authorizationID,
     String                    userSpaceTokenDescription,
     TRetentionPolicyInfo      retentionPolicyInfo,
     unsigned long             desiredSizeOfTotalSpace,
     unsigned long             desiredSizeOfGuaranteedSpace,
     int                       desiredLifetimeOfReservedSpace,
     unsigned long []          arrayOfExpectedFileSizes,
     TExtraInfo[]              storageSystemInfo,
     TTransferParameters       transferParameters
Out: TReturnStatus             returnStatus,
     string                    requestToken,
     int                       estimatedProcessingTime,
     TRetentionPolicyInfo      retentionPolicyInfo,
     unsigned long             sizeOfTotalReservedSpace,       // best effort
     unsigned long             sizeOfGuaranteedReservedSpace,
     int                       lifetimeOfReservedSpace,
     string                    spaceToken
```

e.g. Prepare-to-Get Files Functional Spec

srmPrepareToGet

In: string	authorizationID,
TGetFileRequest[]	arrayOfFileRequests,
string	userRequestDescription,
TExtraInfo[]	storageSystemInfo,
TFileStorageType	desiredFileStorageType
int	desiredTotalRequestTime
int	desiredPinLifetime,
string	targetSpaceToken
TRetentionPolicyInfo	targetFileRetentionPolicyInfo
TTransferParameters	transferParameters
Out: TReturnStatus	returnStatus
string	requestToken,
TGetRequestFileStatus[]	arrayOfFileStatuses
int	remainingTotalRequestTime

e.g. “TGetFileRequest” typedef

```
typedef  struct {
      anyURI   sourceSURL,
      TDirOption  dirOption,
  } TGetFileRequest
```