Storage Resource Management: a uniform interface to Grid storage systems

Arie Shoshani

LBNL

(on behalf of the SRM collaboration)

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

SRM Collaboration Goal

Develop the functional specification of:

Storage Resource Managers (SRMs)

Definition

SRMs are middleware components

whose function is to provide dynamic

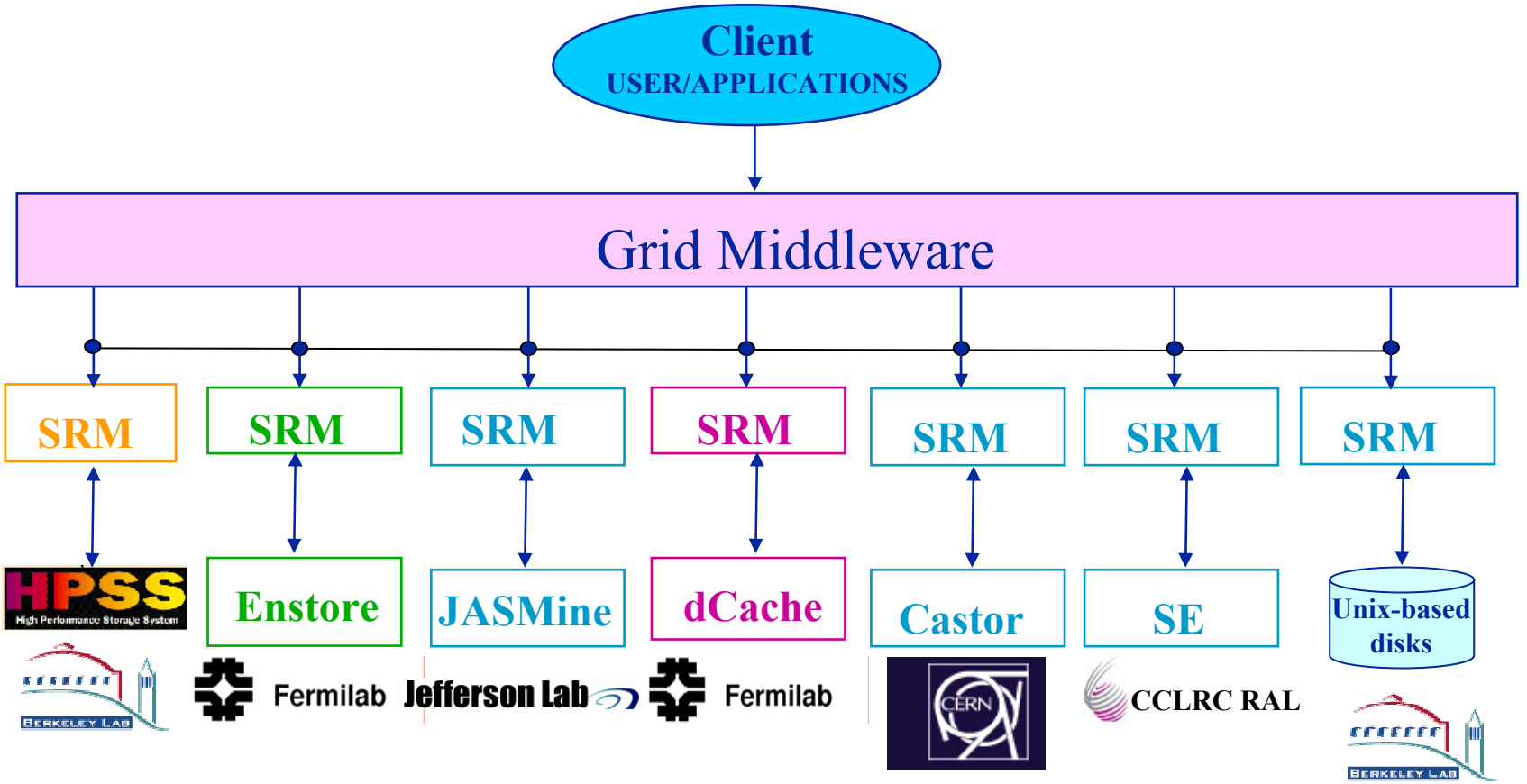
space allocation
file management

of shared storage components on the Grid

History

- 4 year of Storage Resource (SRM) Management activity
- Experience with system implementations v.1.x - 2001
  - MSS: HPSS (LBNL, ORNL, BNL), Enstore (Fermi), JasMINE (Jlab), Castor (CERN), MSS (NCAR), SE (RAL) …
  - Disk systems: DRM(LBNL), dCache(Fermi), jSRM (Jlab), …
- SRM v2.x spec was finalized - 2003
- Several implementations of v2.x completed or in-progress
  - Jlab, Fermi, CERN, LBNL
- Started GSM: GGF-BOF at GGF8 (June 2003)
- Last SRM collaboration meeting – Sept. 2004
- SRM v3.x spec (for GGF) being finalized - 2005

Uniformity of Interface → Compatibility of SRMs



Current Storage Resource Management Active Working Group

CERN: Olof Barring, Jean-Philippe Baud, James Casey, Peter Kunszt

Rutherford lab: Jens Jensen, Owen Synge

Jefferson Lab: Bryan Hess, Andy Kowalski, Chip Watson

Fermilab: Don Petravick, Timur Perelmutov

LBNL: Junmin Gu , Arie Shoshani, Alex Sim, Kurt Stockinger

Univa: Rich Wellner

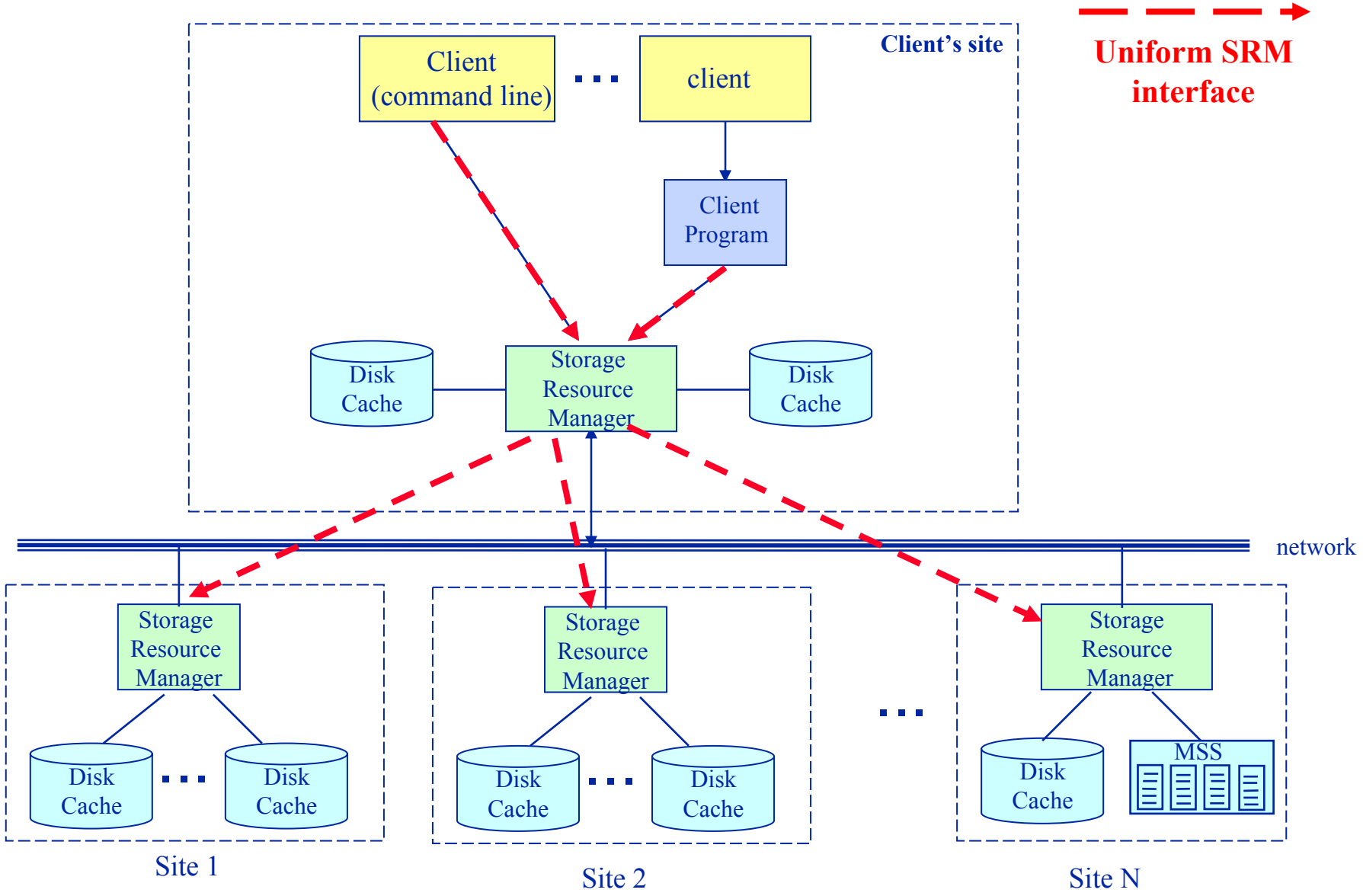
Basic Issues

- 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 to the above for many users
  - Need to enforce quotas
  - Need to ensure fairness of space allocation and scheduling

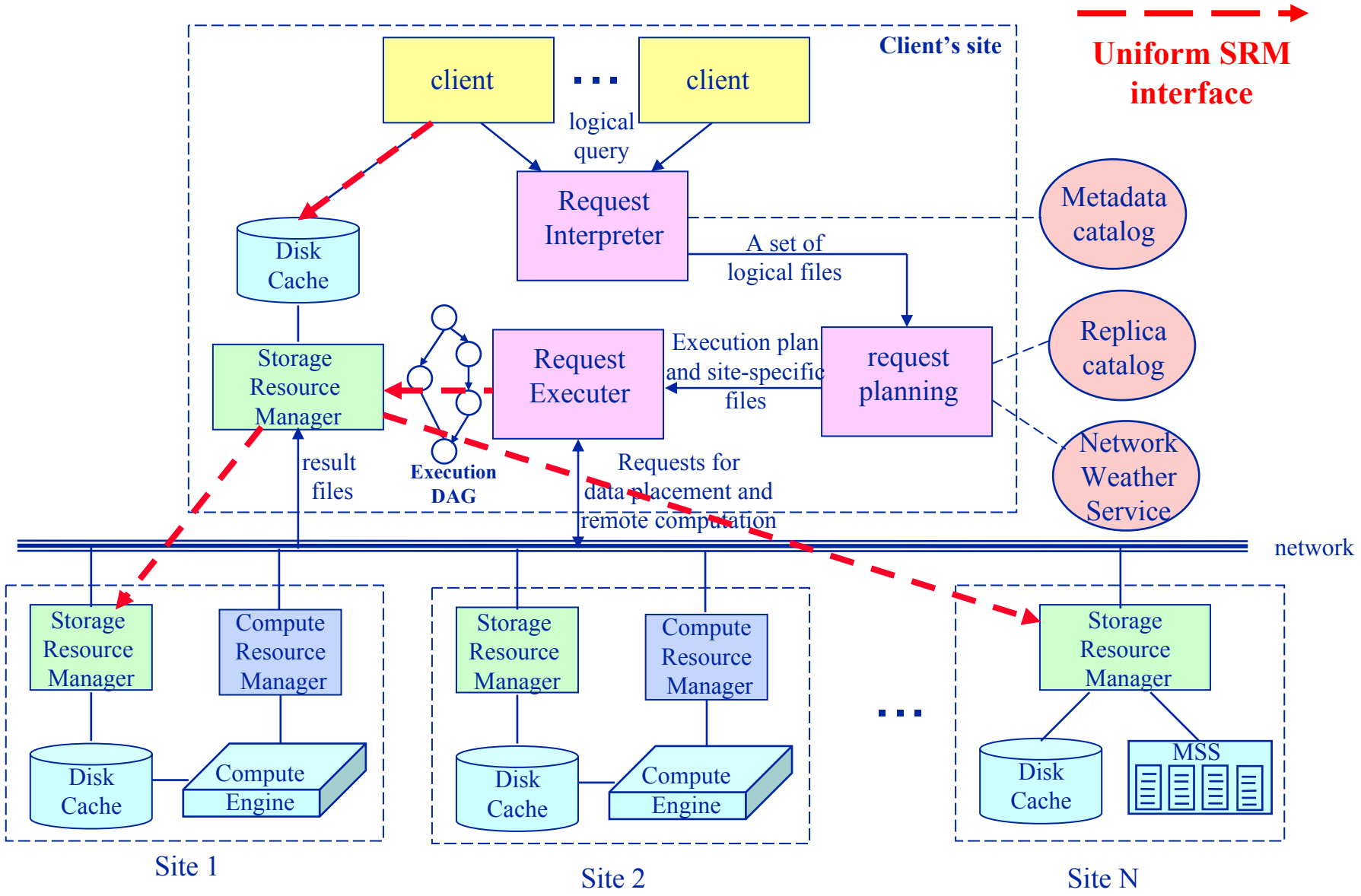
Basic Issues

- Now, suppose you want to do that on a Grid
  - Need to access a variety of storage systems
  - mostly remote systems, need at 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)

Peer-to-Peer Uniform Interface



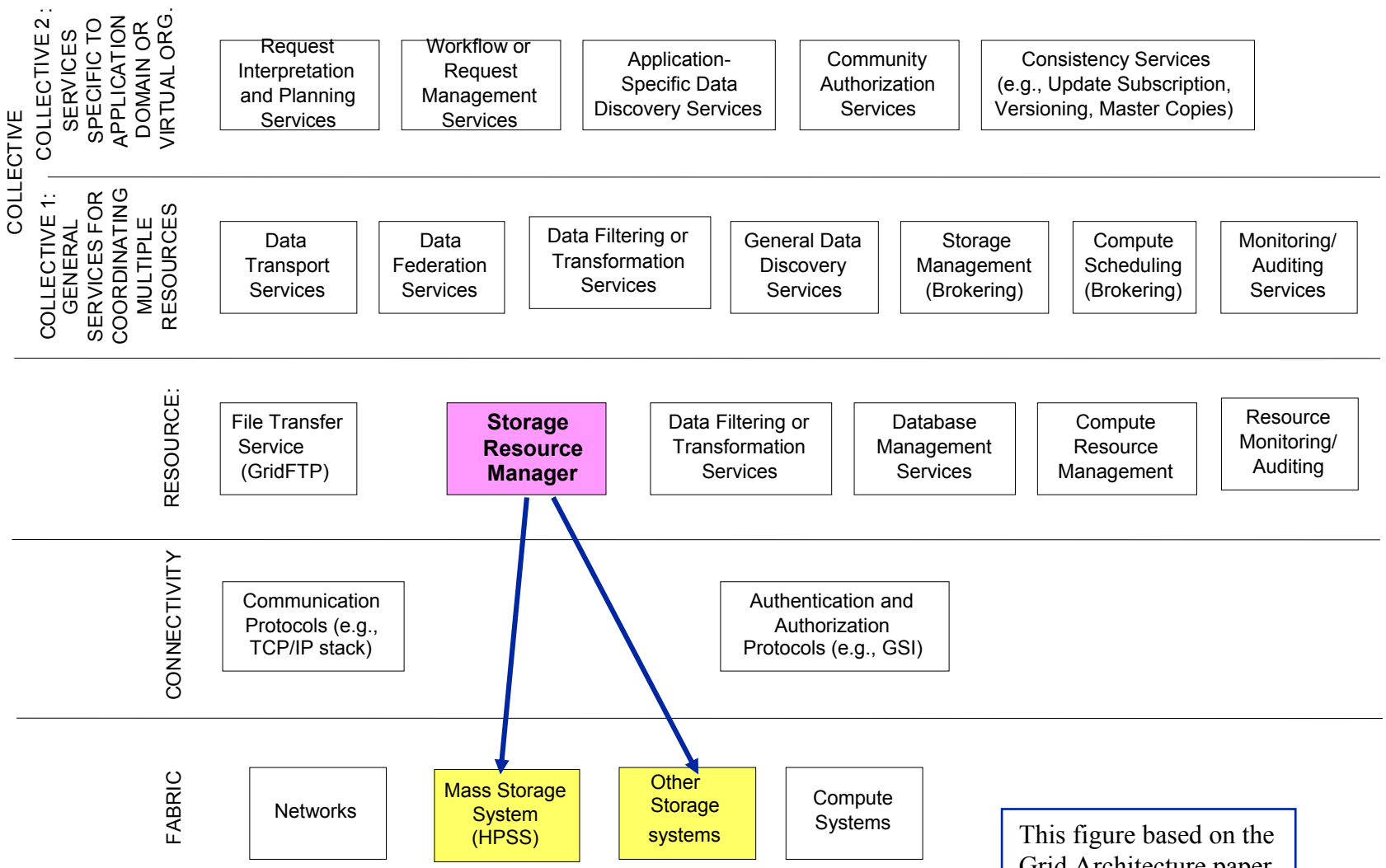
General Analysis Scenario



Standards for Grid Storage Management

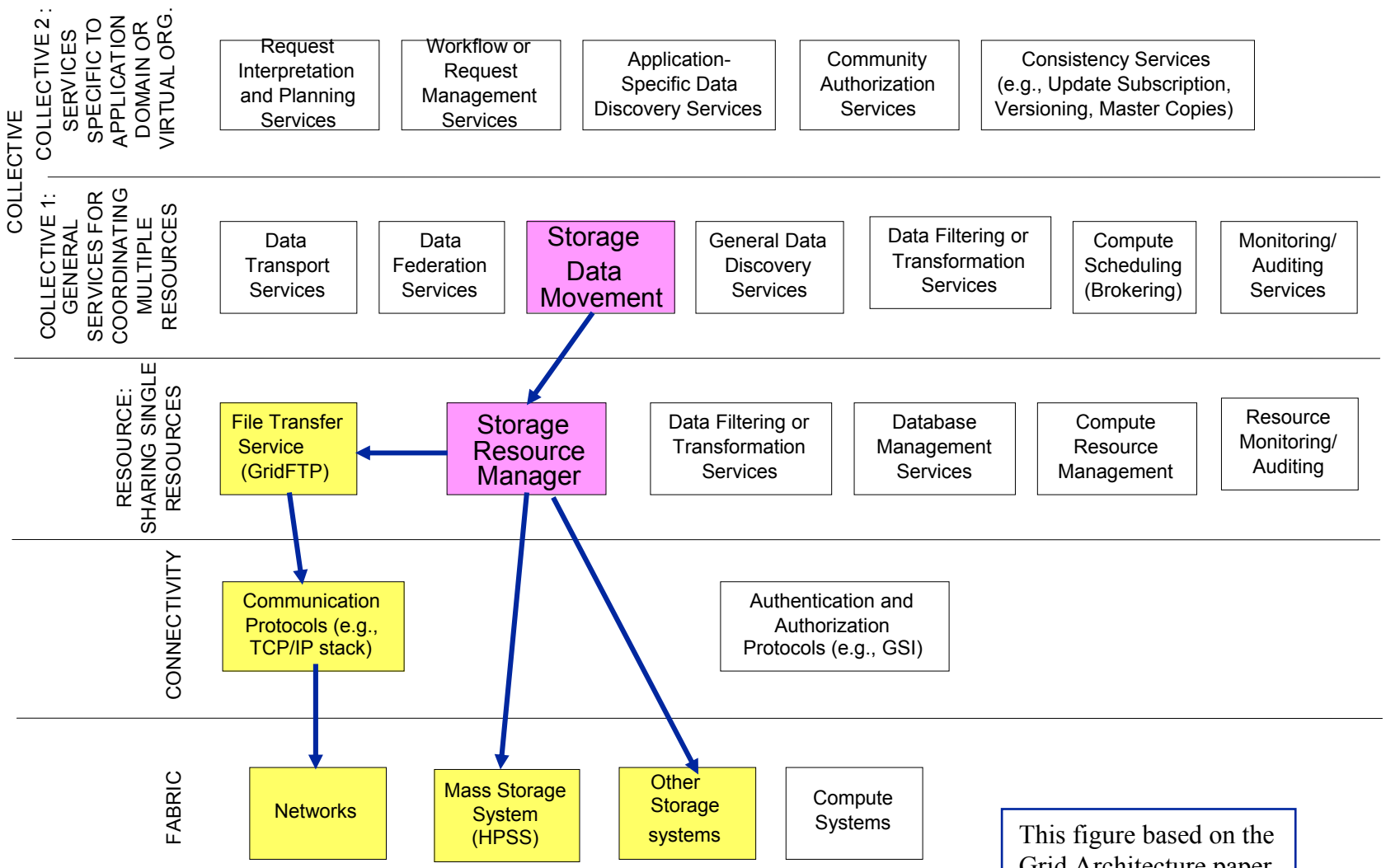
- **Main concepts**
  - Allocate spaces
  - Get/put files from/into spaces
  - Pin files for a lifetime
  - Release files and spaces
  - Get files into spaces from remote sites
  - Manage directory structures in spaces
  - SRMs communicate as peer-to-peer
  - Negotiate transfer protocols
  - No logical name space management (rely of GGF- GFS)

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

SRM Functional Concepts

- Manage Spaces dynamically
  - Reservation, lifetime
  - Negotiation
- Manage files in spaces
  - Request to put files in spaces
  - Request to get files from spaces
  - Lifetime, pining of files, release of files
  - No logical name space management (done by replica location services)
- Access remote sites for files
  - Bring files from other sites and SRMs as requested
  - Use existing transport services (GridFTP, https, …)
  - Transfer protocol negotiation
- Manage multi-file requests
  - Manage request queues
  - Manage caches
  - Manage garbage collection
- Directory Management
  - Uxix semantics: srmLs, srmMkdir, srmMv, srmRm, srmRmdir

Concepts: Types of Files

- **Volatile: temporary files with a lifetime guarantee**
  - Files are “pinned” and “released”
  - Files can be removed by SRM when released or when lifetime expires
- **Permanent**
  - No lifetime
  - Files can only be removed by creator (owner)
- **Durable: files with a lifetime that CANNOT be removed by SRM**
  - Files are “pinned” and “released”
  - Files can only be removed by creator (owner)
  - If lifetime expires – invoke administrative action (e.g. notify owner, archive and release)

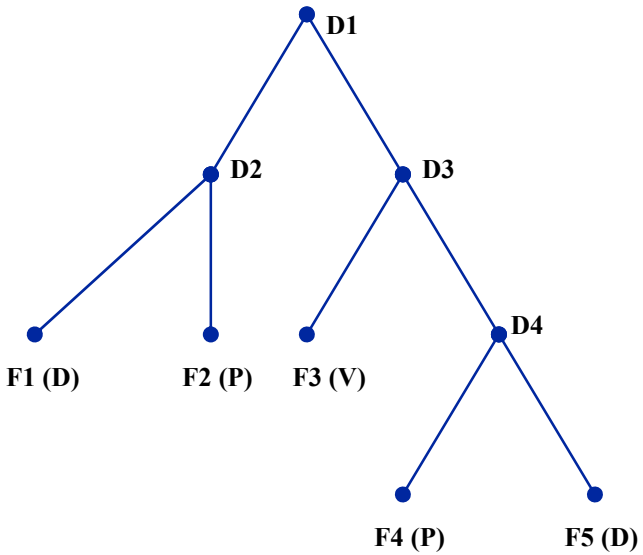
Concepts: Types of Spaces

- Types
  - Volatile
    - Space can be reclaimed by SRM when lifetime expires
  - durable
    - Space can be reclaimed by SRM only if it does NOT contain files
    - Can choose to archive files and release space
  - Permanent
    - Space can only be released by owner or administrator
- Assignment of files to spaces
  - Files can only be assigned to spaces of the same type
- Spaces can be reserved
  - 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
- Default spaces
  - Files can be put into SRM spaces without explicit reservation
  - Defaults are not visible to client
- Compacting space
  - Release all unused space – space that has no files or files whose lifetime expired

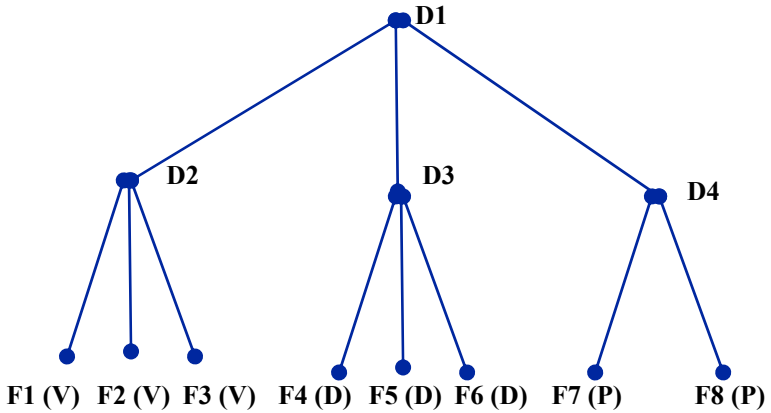
Concepts: Directory Management

- **Usual unix semantics**
  - srmLs, srmMkdir, srmMv, srmRm, srmRmdir
- **A single directory for all file type**
  - No directories for each type
  - File assignment to types is virtual
  - File can be placed in SRM-managed directories by maitaining mapping to client's directory
- **Access control services**
  - Support owner/group/world permission
    - Can only be assigned by owner
    - When file requested by user, SRM should check permission with source site

Examples of Directory Structures (user defined)



(1) Mixed file types



(2) By file type

- Supported function: ChangeFileType
- Advantage of (1): no need to move files when file types are changed

Concepts: Space Reservations

- **Negotiation**
  - Client asks for space: C-guaranteed, MaxDesired
  - SRM return: S-guaranteed <= C-guaranteed,
    best effort <= MaxDesired
- **Type of space**
  - Can be specified
  - Subject to limits per client (SRM or VO policies)
  - Default: volatile
- **Lifetime**
  - Negotiated: C-lifetime requested
  - SRM return: S-lifetime <= C-lifetime
- **Reference handle**
  - SRM returns space reference handle
  - User can provide: srmSpaceTokenDescription to recover handles

Concepts: Transfer Protocol Negotiation

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

Concepts: Multi-file requests

- Can srmRequestToGet multiple files
  - Required: Files URLs
  - Optional: space file type, space handle, Protocol list
  - Optional: total retry time
- Provide: Site URL (SURL)
  - URL known externally – e.g. in Rep Catalogs
  - e.g. srm://sleepy.lbl.gov:4000/tmp/foo-123
- Get back: transfer URL (TURL)
  - Path can be different that in SURL – SRM internal mapping
  - Protocol chosen by SRM
  - e.g. gridftp://dm.lbl.gov:4000/home /level1/foo-123
- Managing request queue
  - Allocate space according to policy, system load, etc.
  - Bring in as many files as possible
  - Provide information on each file brought in or pinned
  - Bring additional files as soon as files are released
  - Support file streaming

SRM Methods

File Movement
srmPrepareToGet
srmPrepareToPut
srmCopy

Lifetime management
srmReleaseFiles
srmPutDone
srmExtendFileLifeTime

Terminate/resume
srmAbortRequest
srmAbortFile
srmSuspendRequest
srmResumeRequest

Space management
srmReserveSpace
srmReleaseSpace
srmUpdateSpace
srmCompactSpace

FileType management
srmChangeFileType

Status/metadata
srmGetRequestStatus
srmGetFileStatus
srmGetRequestSummary
srmGetRequestID
srmGetFilesMetaData
srmGetSpaceMetaData

SRM v3.x: Basic vs. Advanced Features

	BASIC	ADVANCED
• File movement		
• PrepareToGet	yes	yes
• PrepareToPut	yes	yes
• Copy	no	yes
• Request capabilities		
• Multi-file Streaming	yes	yes
• Trans. Prot. Negotiation	yes	yes
• File lifetime negotiation	no	yes
• File types		
• Volatile	yes	yes
• Permanent	yes (for MSS)	yes
• durable	no	yes

Features in Basic vs. Advanced SRM

	BASIC	ADVANCED
• Space reservations		
• Space-time negotiation	no	yes
• Space types	no	yes
• Remote access		
• gridFTP	no	yes
• Other SRMs	no	yes
• User-specified Directory		
• Volatile	no	yes
• Permanent	yes	yes
• Durable	no	yes
• Terminate/suspend		
• Abort file	yes	yes
• Abort request	yes	yes
• Suspend/resume request	no	yes

Use Case

Use of SRMs
for
Robust directory-to-directory file replication

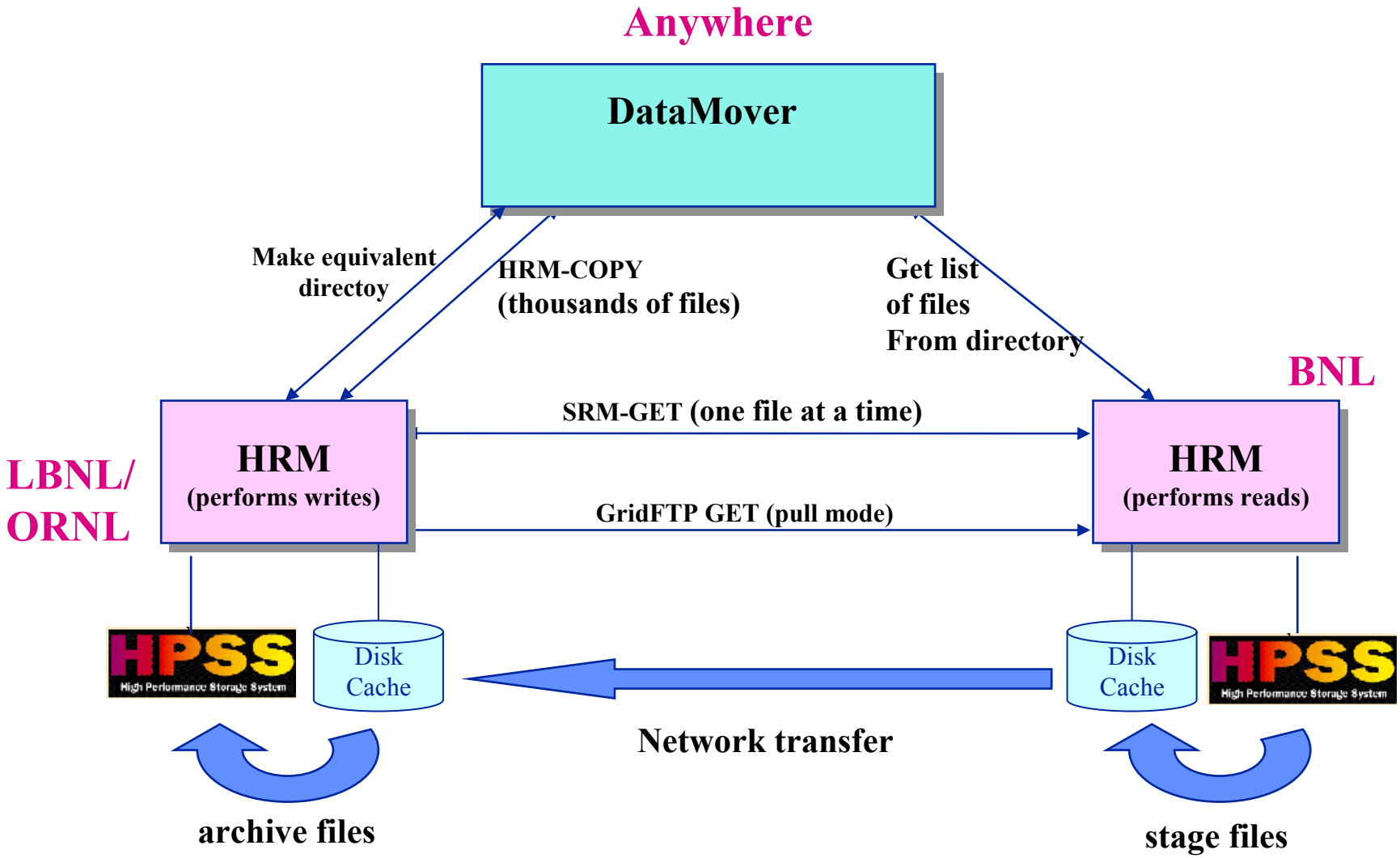
Massive Robust File Replication

- **Multi-File Replication – why is it a problem?**
  - Tedious task – many files, repetitious
  - Lengthy task – long time, can take hours, even days
  - Error prone – need to monitor transfers
  - Error recovery – need to restart file transfers
  - Stage and archive from MSS – limited concurrency, down time, transient failures
  - Use of FTP – no large windows / multiple streams
  - Security – both for local MSS and the network
  - Firewalls – transfer from/to MSS must be internal to the site
  - Specialized MSS – HPSS at NERSC, ORNL, …,
  - Legacy MSS – MSS at NCAR

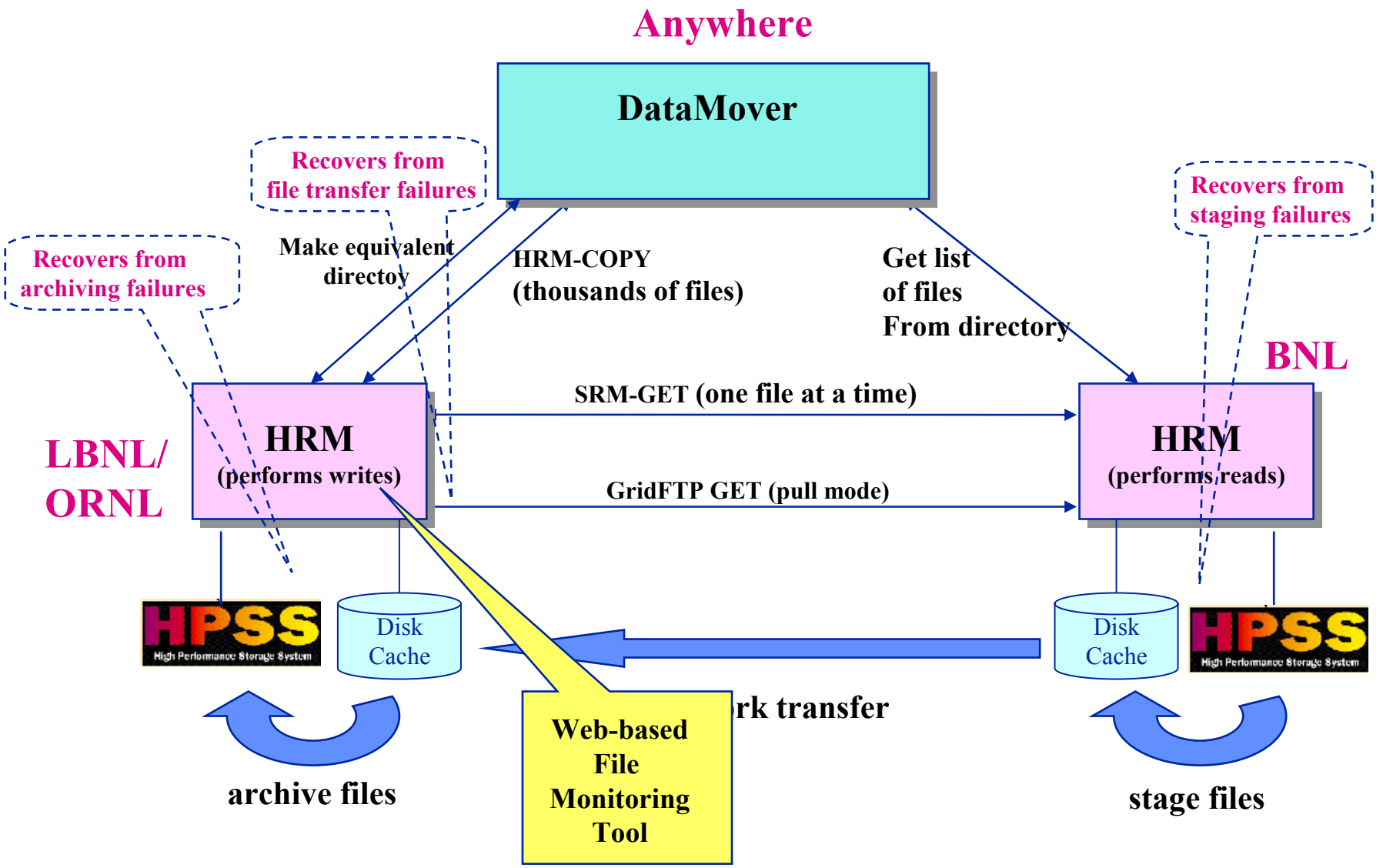
Main Idea

- **Leverage off Storage Resource Managers (SRMs) Technology**
  - Supported by SRM middleware project
  - Leverage from experience with other SciDAC projects – PPDG
- **What do you get?**
  - SRMs queue multi-file requests
  - SRMs allocate space and release space automatically
  - SRMs request files from remote SRMs
  - Recover from network failures
  - SRMs invoke GridFTP – use large windows & parallel streams

DataMover: HRMs use in ESG for Robust Muti-file replication



DataMover: HRMs use in ESG for Robust Muti-file replication



Web-Based File Monitoring Tool

Shows:

-Files already transferred

- Files during transfer

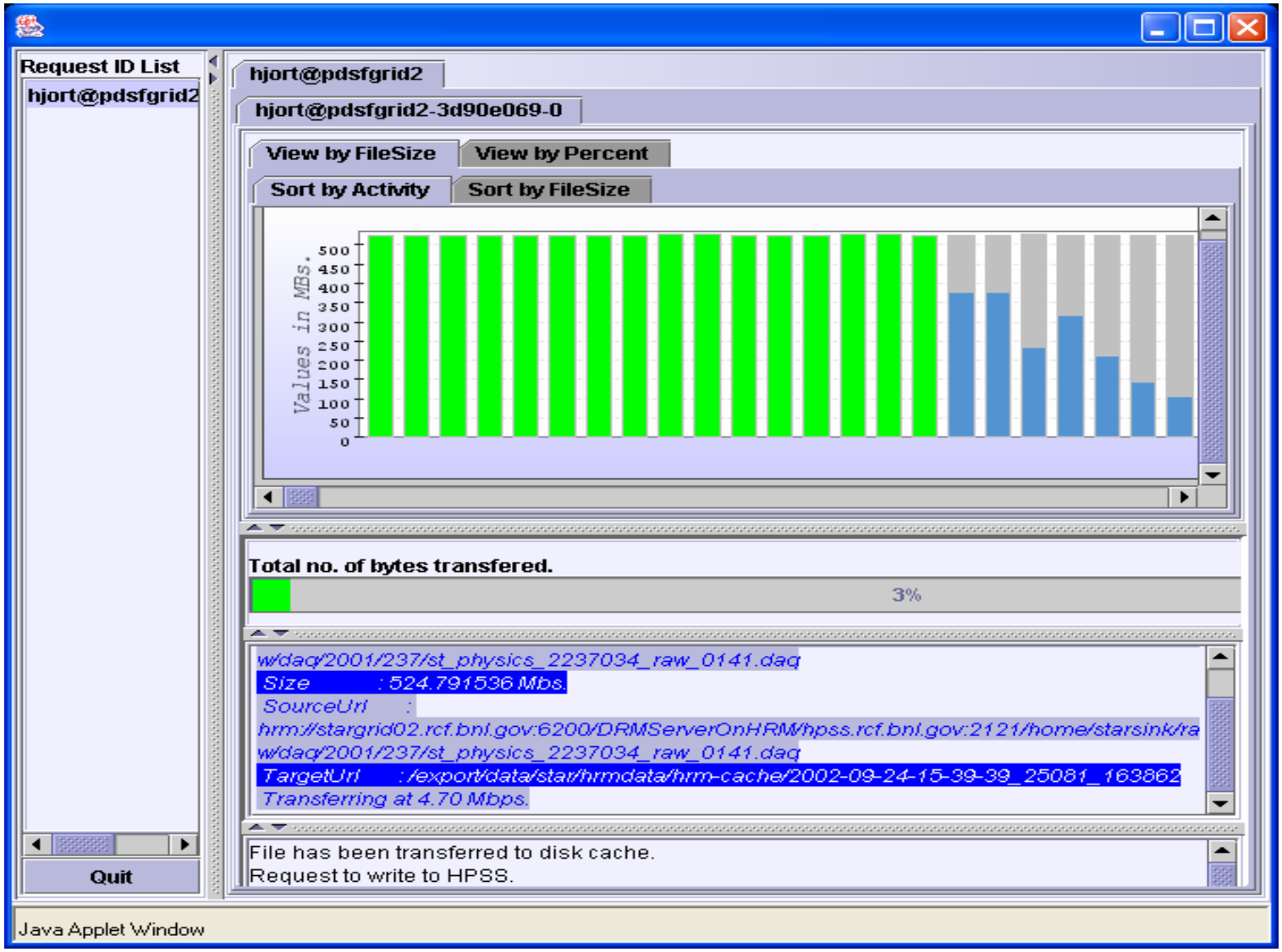
- Files to be transferred

Also shows for each file:

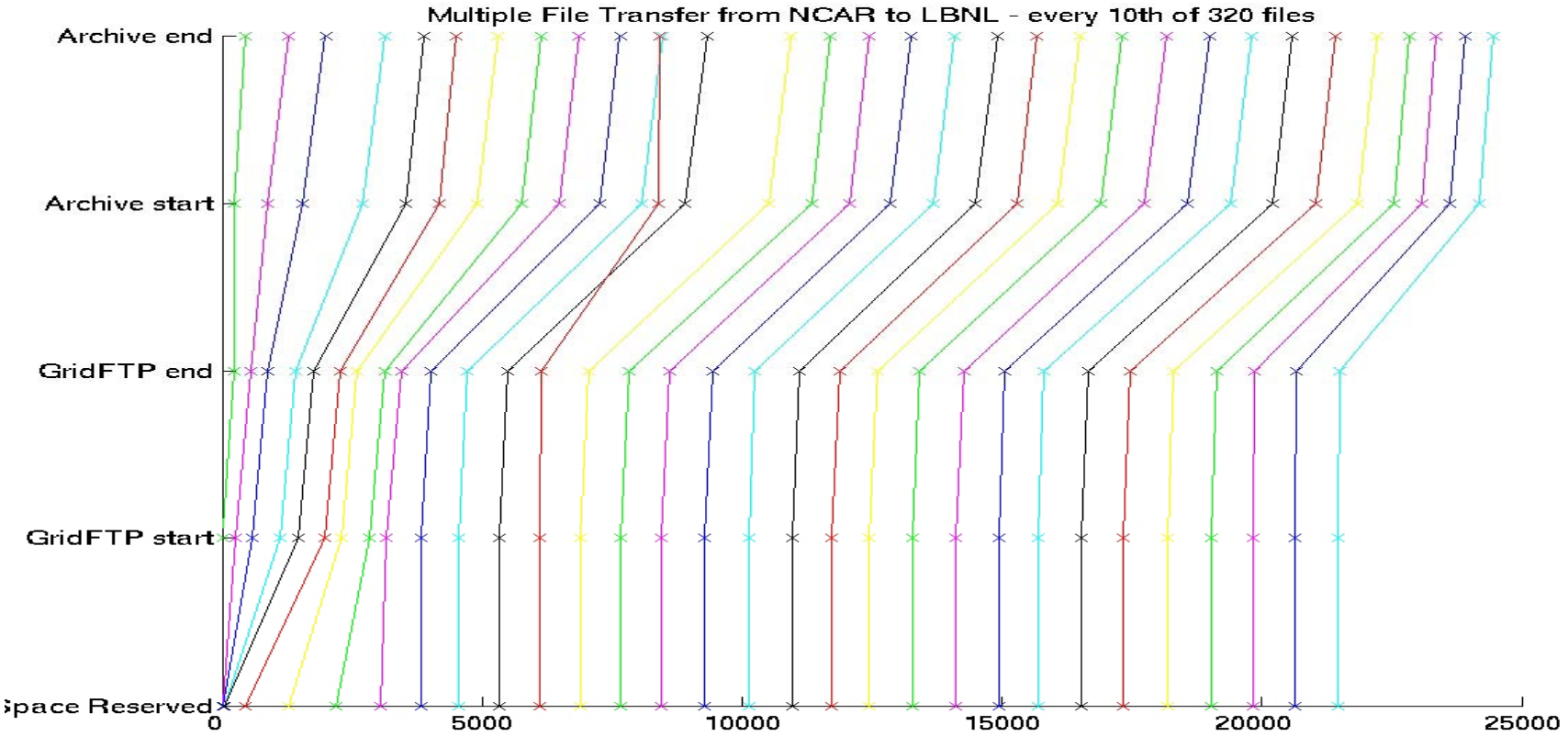
-Source URL

-Target URL

-Transfer rate

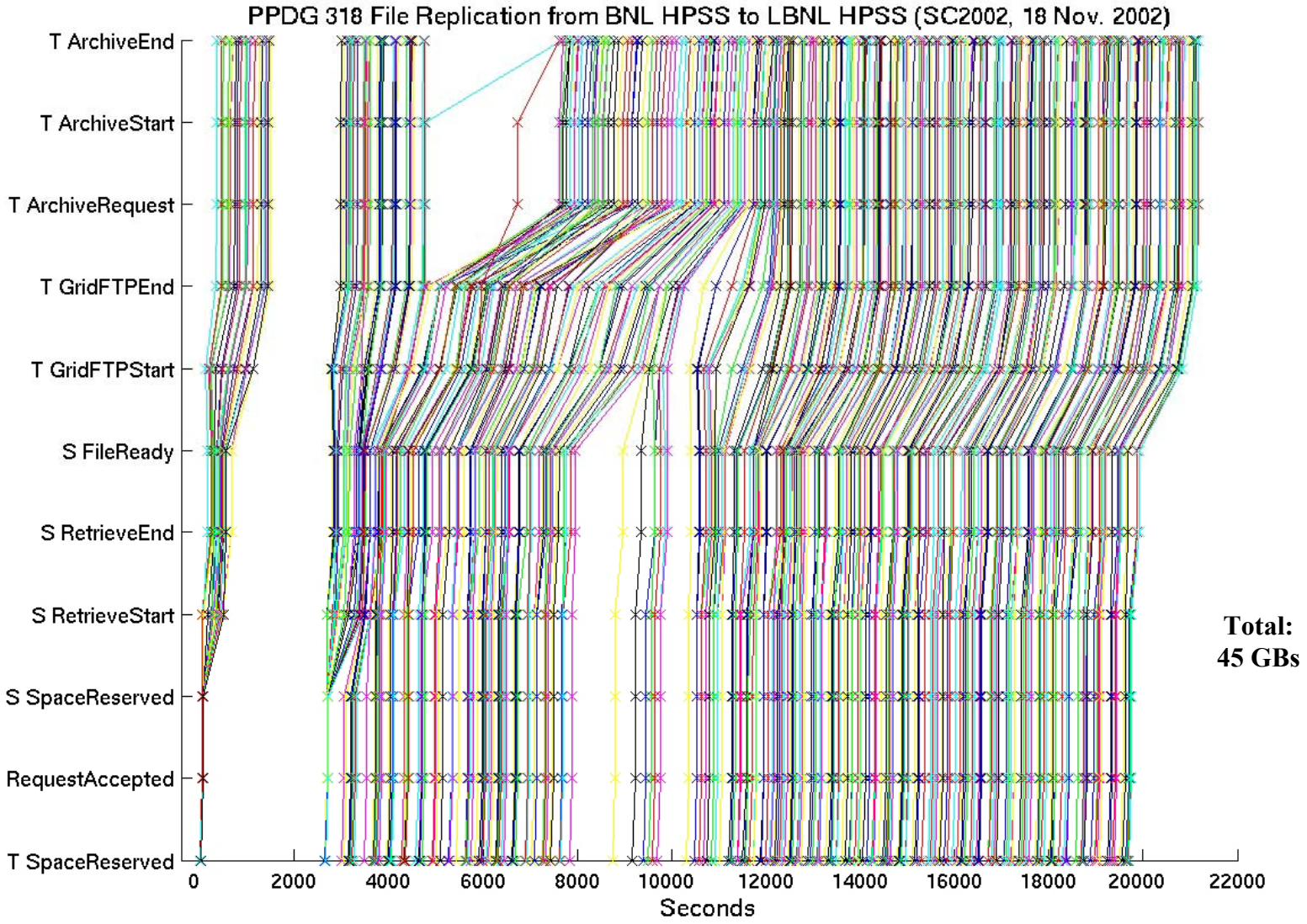


File tracking helps to identify bottlenecks



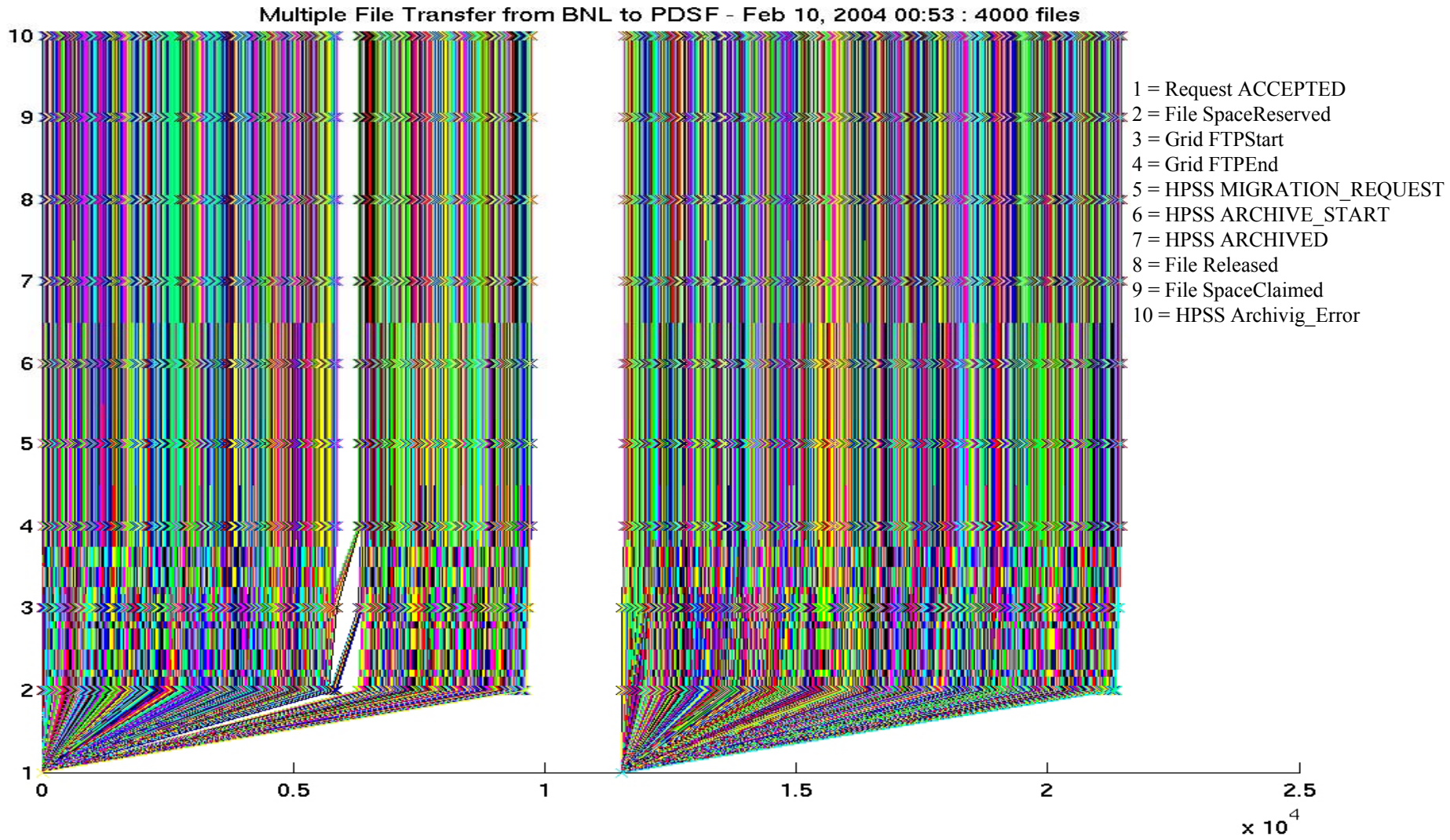
Shows that archiving is the bottleneck

File tracking shows recovery from transient failures



Total: 45 GBs

Multi-file Transfer plot from BNL to LBNL (10/02/04)



Summary

- Storage Resource Management – essential for Grid
- SRM is a functional definition
  - Adaptable to different frameworks (WS, OGSA, WSRF, …)
- Multiple implementations interoperate
  - Permit special purpose implementations for unique products
  - Permits interchanging one SRM product by another
- SRM implementations exist and some in production use
  - Particle Physics Data Grid
  - Earth System Grid
  - More coming …
- Cumulative experience in GGF-WG
  - Specifications SRM v3.0 complete

Extra Slides

Space Reservation Functional Spec

srmReserveSpace

In:	TUserID	userID,
	TSpaceType	typeOfSpace,
	String	userSpaceTokenDescription,
	TSizeInBytes	sizeOfTotalSpaceDesired,
	TSizeInBytes	sizeOfGuaranteedSpaceDesired,
	TLifeTimeInSeconds	lifetimeOfSpaceToReserve,
	TStorageSystemInfo	storageSystemInfo
Out:	TSpaceType	typeOfReservedSpace,
	TSizeInBytes	sizeOfTotalReservedSpace,
	TSizeInBytes	sizeOfGuaranteedReservedSpace,
	TLifeTimeInSeconds	lifetimeOfReservedSpace,
	TSpaceToken,	referenceHandleOfReservedSpace,
	TReturnStatus	returnStatus

"Request-to-Get" Files Functional Spec

srmPrepareToGet

In:	TUserID	userID,
	TGetFileRequest[ ]	arrayOfFileRequest,
	string[]	arrayOfTransferProtocols,
	string	userRequestDescription,
	TStorageSystemInfo	storageSystemInfo,
	TLifeTimeInSeconds	TotalRetryTime
Out:	TRequestToken	requestToken,
	TReturnStatus	returnStatus,
	TGetRequestFileStatus[ ]	arrayOfFileStatus

"TGetFileRequest" typedef Functional Spec

typedef	struct {TSURLInfo	fromSURLInfo,
	TLifeTimeInSeconds	lifetime, // pin time
	TFileStorageType	fileStorageType,
	TSpaceToken	spaceToken,
	TDirOption	dirOption
	} TGetFileRequest	

Detailed sequence of actions For each file being replicated