Berkeley-SRM v2.1.1

Alex Sim
Junmin Gu
Arie Shoshani

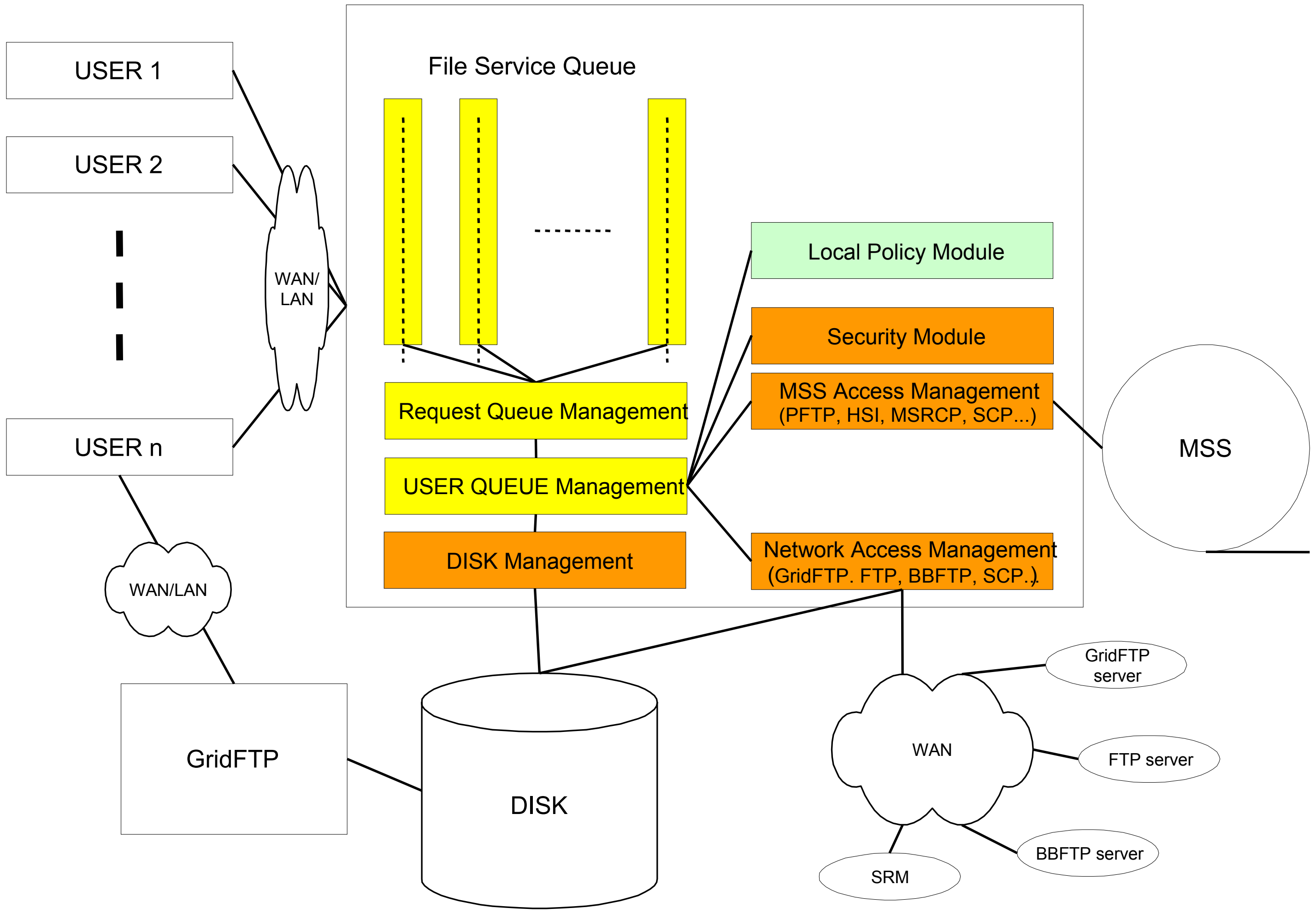
LCG workshop
April 6, 2005

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

Berkeley SRM v.2.1.1 Implementation Status

- This version will combine DRM and HRM-HPSS into a single code base called Berkeley-SRM
  - Berkeley-SRMforNCAR A specialized version for NCAR MSS
- Prototype for DRM part is ready for testing
- All interfaces except permission related functions (for ACL authorization) are implemented
- Based on JDK 1.2.4_07, Ant 1.6.1, and Globus 3.2 (including Java-CoG)

Berkeley-SRM Architecture



Changes from v2.1 to V2.1.1

- **Changed all userID argument name to authorizationID.**
    - **from the SASL RFC 2222**
- **Added an additional, optional argument to srmChangeFileStorageType()**

In:	TUserID	authorizationID,
	TSURLInfo[]	arrayOfPath,
	TFileStorageType	desiredStorageType,
	TSpaceToken	spaceToken

Berkeley SRM v.2.1.1 Features Implemented

- Data Transfer Functions
  - srmPrepareToGet
  - srmPrepareToPut
  - srmCopy
  - srmRemoveFiles
  - srmReleaseFiles
  - srmPutDone
  - srmExtendFileLifeTime
- Status functions
  - srmStatusOfGetRequest
  - srmStatusOfPutRequest
  - srmStatusOfCopyRequest
  - srmGetRequestSummary
  - srmGetRequestID
- Abort/resume
  - srmAbortRequest
  - srmAbortFiles
  - srmSuspendRequest
  - srmResumeRequest
- Space Management Functions
  - srmReserveSpace
  - srmReleaseSpace
  - srmUpdateSpace
  - srmCompactSpace
  - srmGetSpaceMetaData
  - srmChangeFileStorageType
  - srmGetSpaceToken
- Directory Functions
  - srmMkdir
  - srmRmdir
  - srmRm
  - srmLs
  - srmMv
- Not implemented
  - Permission Functions
  - srmSetPermission
  - srmReassignToUser
  - srmCheckPermission

Changes required in WSDL for inter-operability

- Due to the Apache Axis handling of an array (a bug), SRM-WSDL file had to be changed as in the next slide
  - http://issues.apache.org/bugzilla/show_bug.cgi?id=22213
  - We have tested the modified SRM-WSDL with gSoap 2.7, and the behavior is okay.

  - Modified WSDL file URL:
    - http://sdm.lbl.gov/srm-wg/srm.v3.modified.wsdl

An example of replacing an “array” with a “sequence”

Previous: soapenv:array

```
<complexType name="ArrayOfTSpaceToken">
        <complexContent>
              <restriction base="soapenc:Array">
                     <attribute ref="soapenc:arrayType"
                     wsdl:arrayType="impl:TSpaceToken[]"/>
              </restriction>
        </complexContent>
</complexType>
```

New: xsd:sequence

```
<complexType name="ArrayOfTSpaceToken">
        <sequence>
              <element name="tokenArray" maxOccurs="unbounded"
                     type="impl:TSpaceToken"/>
        </sequence>
</complexType>
```

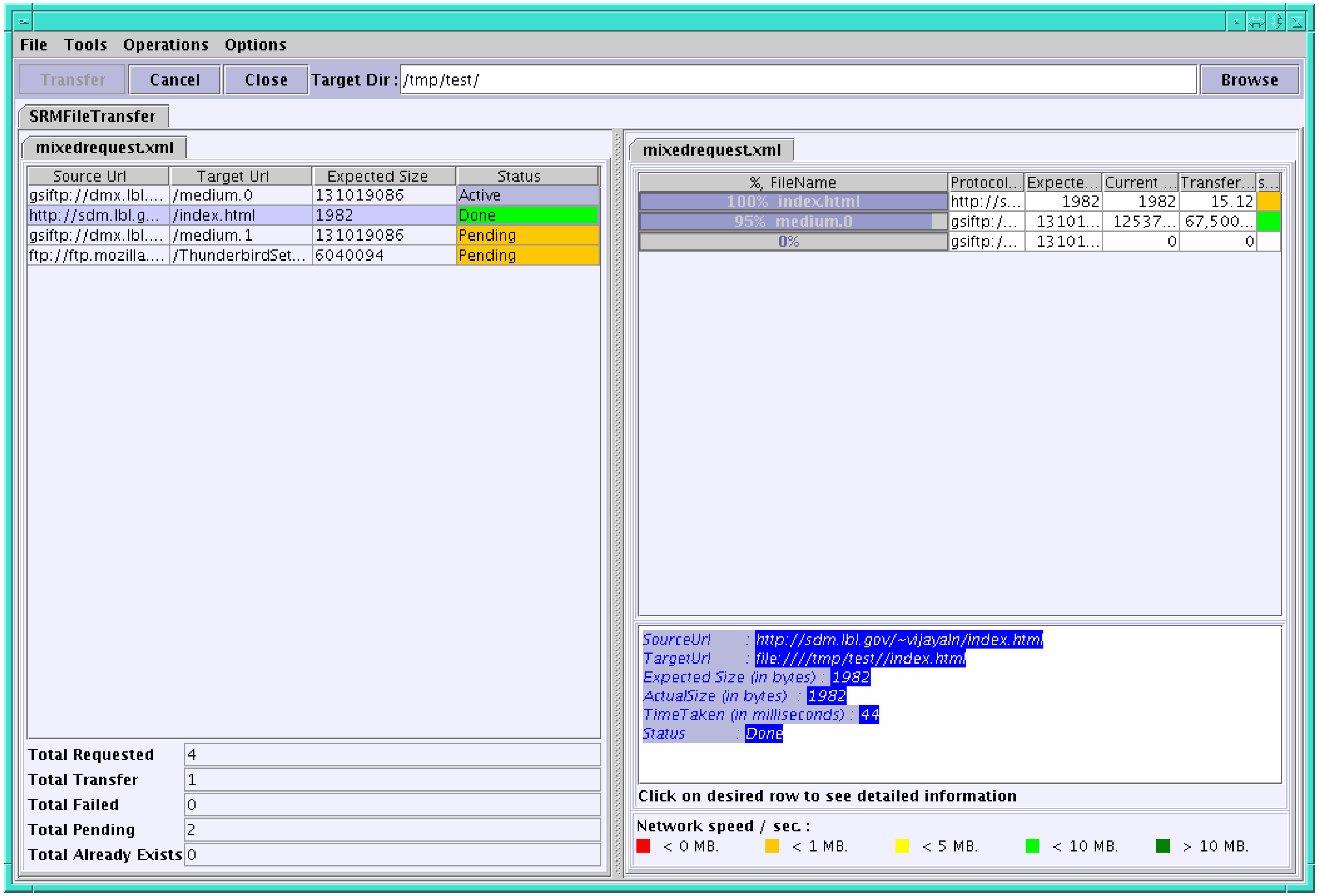
Where:

```
<complexType name="TSpaceToken">
        <sequence>
               <element name="value" minOccurs="1" maxOccurs="1" nillable="false"
                     type="xsd:string"/>
        </sequence>
</complexType>
```

Berkeley SRM v.2.1.1 Plans

- Testing/scalability of current v.2.1.1 prototype
- Replace current v.1.1 based Berkeley SRMs (DRMs and HRMs), but…
  - continue to support v1.1 clients
    - Requires internal translation of v.1.1 based client calls to v2.1
  - Continue to support v1.1 servers
    - Requires discovery of remote SRM version
    - Requires translation of v2.1 functions to v1.1 call when possible (e.g. srmGet, srmCopy)
- Develop a general SRM Client tool
  - GUI and command-line tools, also Java API

SRMClient GUI - file transfer



Target Dir : /tmp/test/

mixedrequest.xml

Source Url	Target Url	Expected Size	Status
gsiftp://dmx.lbl....	/medium.0	131019086	Active
http://sdm.lbl.g...	/index.html	1982	Done
gsiftp://dmx.lbl....	/medium.1	131019086	Pending
ftp://ftp.mozilla....	/ThunderbirdSet...	6040094	Pending

Total Requested	4
Total Transfer	1
Total Failed	0
Total Pending	2
Total Already Exists	0

mixedrequest.xml

%, FileName	Protocol...	Expecte...	Current ...	Transfer...	s...
100% index.html	http://s...	1982	1982	15.12	🟧
95% medium.0	gsiftp:/...	13101...	12537...	67,500...	🟩
0%	gsiftp:/...	13101...	0	0	

SourceUrl : http://sdm.lbl.gov/~vijayaln/index.html
TargetUrl : file:///tmp/test//index.html
Expected Size (in bytes) : 1982
ActualSize (in bytes) : 1982
TimeTaken (in milliseconds) : 44
Status : Done

Click on desired row to see detailed information

Network speed / sec. :

🟥 < 0 MB. 🟧 < 1 MB. 🟨 < 5 MB. 🟩 < 10 MB. 🟩 > 10 MB.

SRM Collaboration

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

GGF Grid Storage Management WG
GSM-WG

Goal: Deveolp and get a standard approved for

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

Current Storage Resource Management Active Working Group

<u>CERN</u>: Peter Kunszt, Erwin Laure, Heinz Stockinger, Jean-Philippe Baud, Olof Barring

<u>Rutherford lab</u>: Jens Jensen, Owen Synge

<u>Jefferson Lab</u>: Bryan Hess, Andy Kowalski, Chip Watson

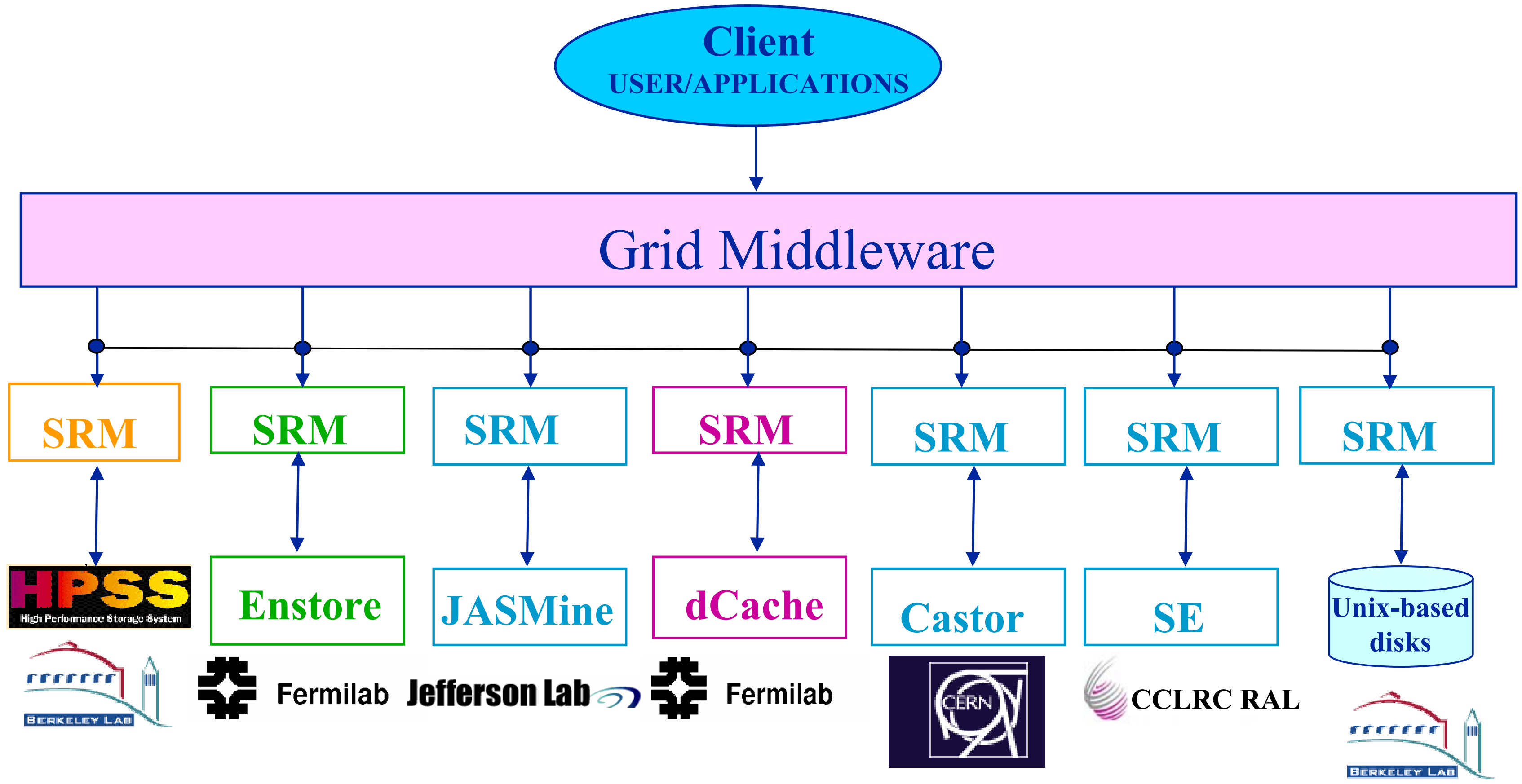
<u>Fermilab</u>: Don Petravick, Timur Perelmutov, Rich Wellner

<u>LBNL</u>: Junmin Gu , Arie Shoshani, Alex Sim, Kurt Stockinger

Collaboration 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

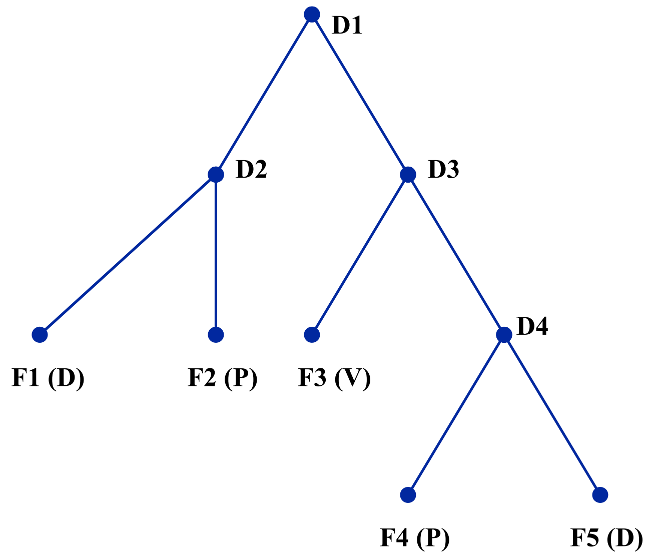
Uniformity of Interface → Compatibility of SRMs



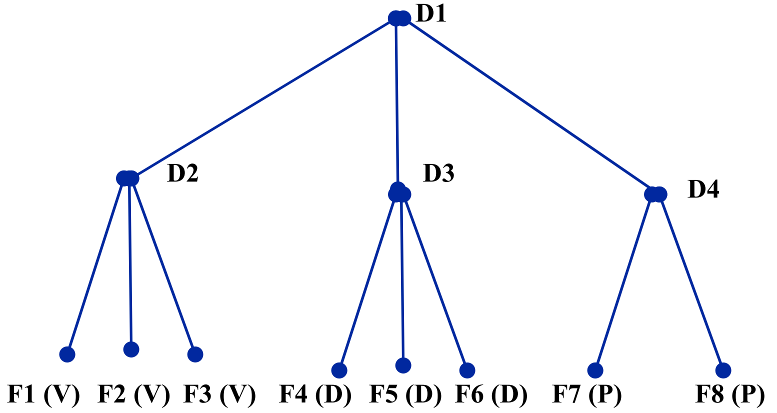
SRM Main Functional Concepts

- Manage Spaces dynamically
  - Reservation, lifetime
  - Negotiation
  - Guaranteed, best-effort
  - Space types: volatile, durable, permanent
- Manage files in spaces
  - Request to put files in spaces
  - Request to get files from spaces
  - Lifetime, pining of files, release of files
  - File types: volatile, durable, permanent
- 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

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

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

SRM v3.0
Core vs. Advanced Features

Core Functions

1. srmAbortRequest
2. srmChangeFileStorageType
3. srmExtendFileLifetime
4. srmGetFeatures
5. srmGetRequestSummary
6. srmGetRequestToken
7. srmGetSRMStorageInfo
8. srmGetSURLMetaData
9. srmGetTransferProtocols
10. srmPrepareToGet
11. srmPrepareToPut
12. srmPutDone
13. srmReleaseRequestedFiles
14. srmStatusOfGetRequest
15. srmStatusOfPutRequest

Advanced Features

1. Remote Copy Functions
2. Space Management Functions
3. Directory Management Functions
4. Authorization Functions
5. Request Administration Functions

SRM v3.0 advanced features

Remote Copy Functions

1. srmCopy
2. srmCopyAndGet
3. srmPutAndCopy
4. srmStatusOfCopyRequest
5. srmStatusOfCopyAndGetRequest
6. srmStatusOfPutAndCopyRequest

Space Management Functions

1. srmCompactSpace
2. srmGetSpaceMetaData
3. srmGetSpaceToken
4. srmRepeaseFilesFromSpace
5. srmReleaseSpace
6. srmReserveSpace
7. srmUpdateSpace

Directory Management Functions

1. srmLs
2. srmMkdir
3. srmMv
4. srmRm
5. srmRmdir

Authorization Functions

1. srmCheckPermission
2. srmGetStatusOfReassignment
3. srmReassignToUser
4. srmSetPermission

Request Administration Functions

1. srmAbortRequestedFiles
2. srmRemoveRequestedFiles
3. srmResumeRequest

Next SRM collaboration meeting

September 14-15

(Wed.-Thurs.)

Jlab

Extra Slides

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 over multiple spaces**
  - **SRMs communicate as peer-to-peer**
  - **Negotiate transfer protocols**

SRM functionality

- Space reservation
  - Negotiate and assign space to users
  - Manage “lifetime” of spaces
  - Release and compact space
- File management
  - Assign space for putting files into SRM
  - Pin files in storage when requested till they are released
  - Manage “lifetime” of files
  - Manage action when pins expire (depends on file types)
- Get files from remote locations when necessary
  - Purpose: to simplify client’s task
  - srmCopy: in “pull” and “push” modes

SRM functionality (Cont'd)

- Space management policies and file sharing
  - Policies on what should reside on a storage resource at any one time
  - Policies on what to evict when space is needed
  - Share files to avoid getting them from remote locations
- Manage multi-file requests
  - Queues file requests, pre-stage when possible
- Status functions
  - Files: lifetime remaining, what's available locally
  - Requests: what files are available (needed in lieu of callbacks)
  - Request summary: for progress report
  - Space metadata: space in use, space available, lifetime
- Provide grid access to/from mass storage systems
  - HPSS (LBNL, ORNL, BNL), Enstore (Fermi), JasMINE (Jlab), Castor (CERN), MSS (NCAR), SE (RAL) …

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

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

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
```

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