SRM V2.1: Additional Design Issues

Participants:

JLAB: Bryan Hess, Andy Kowalski

Fermi: Don Petravick, Timur Perelmutov

LBNL: Alex Sim, Arie Shoshani

WP2: Peter Kunszt, Heinz Stockinger, Kurt Stockinger

WP5: Jean-Philippe Baud, Stefano Occhetti, Jens Jensen

meeting held at CERN

December, 2002

SRM V2.1: Additional Design Issues

Current version proposal

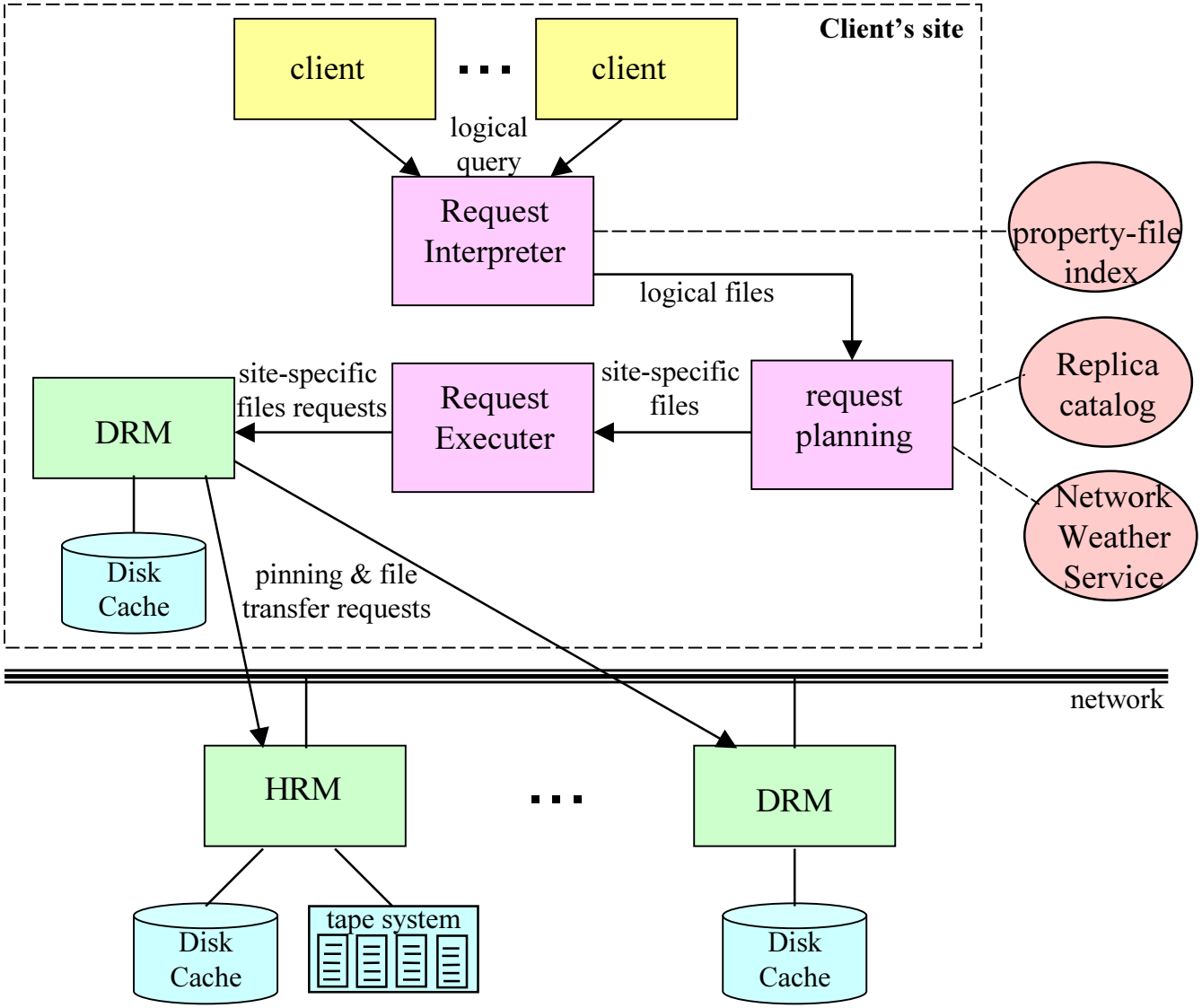
by

Alex Sim, Junmin Gu, Arie Shoshani

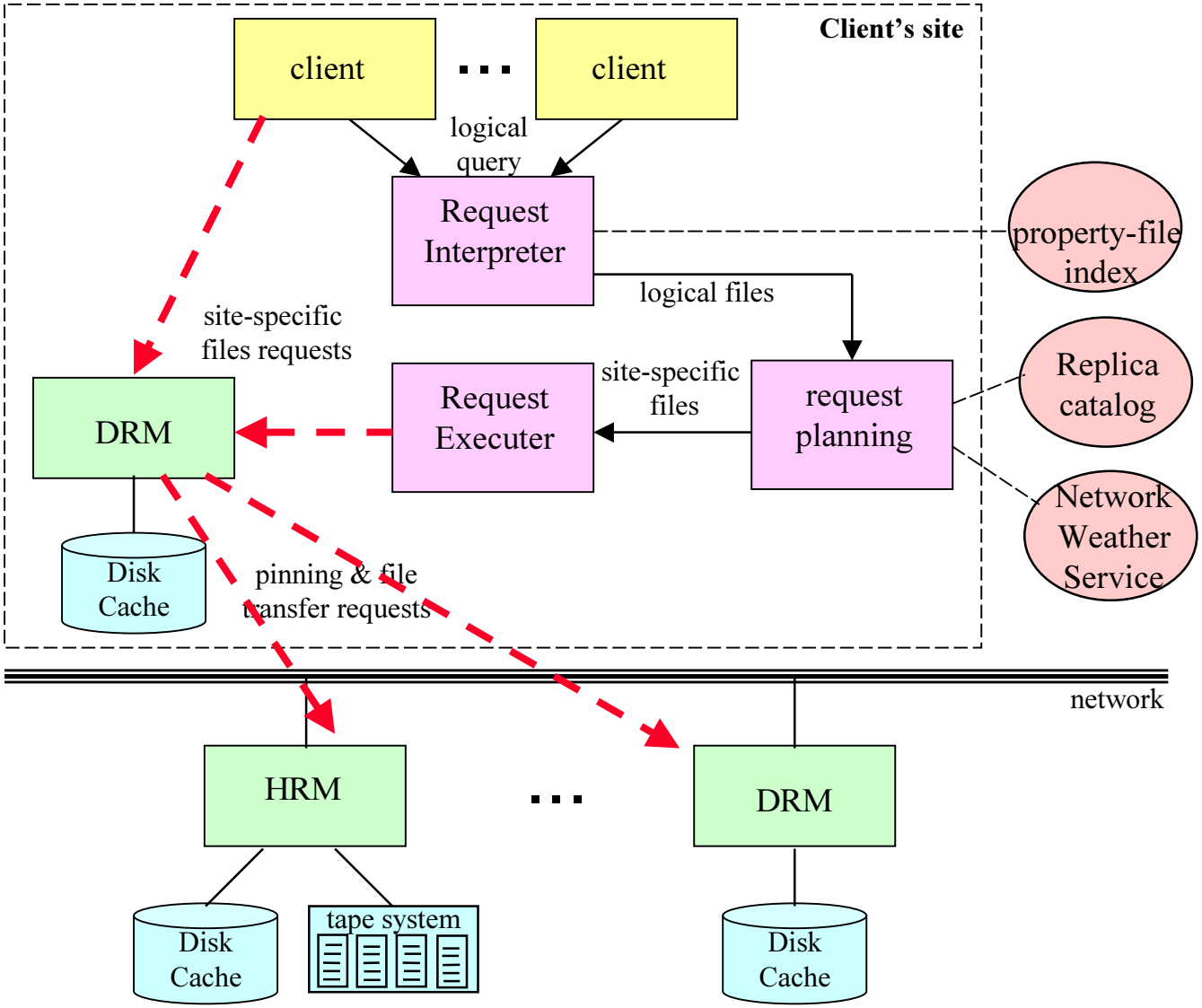
Agenda

- Goal
  - Agree on open SRM issues
  - Main issues
    - Space reservations
    - Directory functions
    - Access control
    - Lifetime negotiations (space and file)
  - Other issues
    - Meaning of lifetime
    - Renew lifetime
    - Revisit srmCopy (push mode)
    - Revisit call-backs with WSDL handles
- Architecture
  - Where do SRM fit?

Request Manager and SRMs



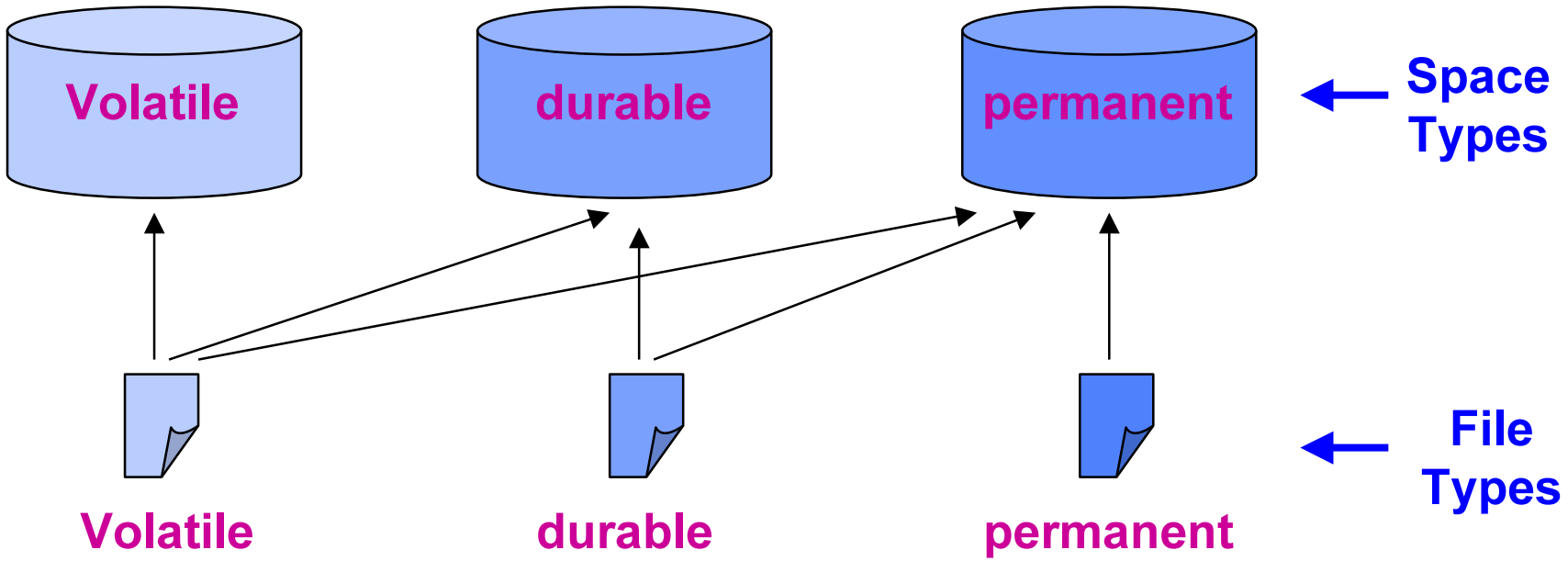
Request Manager and SRMs



Space Reservation

- Space types
  - Option 1: volatile, durable, permanent
  - Option 2: volatile, permanent
- Why support “durable” space?
  - Reminder: durable files are temporary files that can only be removed by owner/administrator
  - For systems that do not intend to have permanent space (e.g. DRMs), but wish to support durable files
- Recommendation
  - Support space reservations for all three types
  - Size of reservation is negotiable
  - Request for additional space possible
  - Lifetime of reservation is negotiable
  - Request for lifetime extension possible

File assignment to spaces



- Implications
  - Need to acquire durable/permanent space to store durable files
  - Need to acquire permanent space to store permanent files
  - Can change file types within a space according to above figure

Semantics of spaces

- File assignment
  - File lifetime cannot exceed space lifetime
- Reservation guarantees
  - Volatile: soft guarantee – space can be reclaimed, minimum is guaranteed
  - Durable, permanent – hard guarantee
- Space removal/release
  - Permanent – by client
    - ReleaseSpace – requires all files in space released
    - ForceReleaseSpace – remove all files as well
  - Durable – by client or administrator if lifetime expires
    - ReleaseSpace – requires all files in space released (files with lifetime expired are not considered “released”)
    - ForceReleaseSpace – remove all files as well
  - Volatile
    - Assign minimum quota if request is still active
    - Otherwise, reclaim space

Parameters of space reservation request / negotiations

Function: srmReserveSpace

- In parameters
  - User_ID
  - Type of space
  - Size (granularity: MBs)
  - Lifetime (granularity: minutes)
- Return parameters
  - Space_size_approved (MBs)
  - Lifetime_approved (minutes)
  - Request_ID
  - Hold lifetime (minutes)

Function: srmConfirmSpace

- In parameters
  - User ID
  - Request ID
- Return parametrs
  - OK, refuse

Space request functions

- srmReserveSpace()
- srmReleaseSpace()
- srmReduceSpace()
- srmAddSpace ()
- srmCompactSpace()
  - Reduce space to size of active files (non-released files)

Space Usage Reporting

- Space used (in MB-Minutes)
- Space reserved (in MB-Minutes)
- Usage bandwidth (in MBs) - optional
  - Total MBs of data transferred on behalf of user associated with space

File Sharing in Spaces

- Files (owned) in any spaces can be shared by requesters
  - Provided that requester has permission to “read”
- In volatile space
  - File size is counted against requester’s quota
- In durable / permanent space
  - Requester does not have to have durable / permanent space
  - File size not counted against requester’s quota
  - File is not moved to requester’s space
  - If owner removes / releases file, requester may get error when reading / transferring file
- srmCopyToSpace()
  - If requester wants to move file to its space
  - Can designate a file type different from original

Directory Semantics

- Client can establish directories for any of their spaces
  - Including “volatile” spaces !
  - File sharing in “volatile spaces” must be supported (implementation choice – e.g. links)
- Only one space type per client !
- Clients refer to spaces as:
  - /volatile, /durable, /permanent (only reserved terms)
  - Top level directory names is local DRM choice
    - E.g. /home/srm/john/durable
- TFNs (in TURLs) returned to clients include 4 levels + user established directories
    - E.g. /home/srm/john/durable/abc/xyz/file.foo (here abc, xyz are directories john setup)

Directory Operations

- ls, mkdir, mv, cd, rm, rmdir, cd, pwd
  - (Prefix with srm?)
  - Relative to space name: /volatile, /durable, /permanent
  - Default is /volatile
  - Allow –r (recursive) in rm, rmdir, ls
  - Allow for ls: max_list, offset (Chip Watson)
- Rm
  - Hard delete in /durable or /permanent space
  - Soft delete (advisory delete) in /volatile
- Access control
  - In future: all access through CAS
  - In current version:
    - Defaults
      - Volatile: read-world
      - Durable / permanent: read-owner
    - Can use chmod
      - Volatile – chmod may be refused by SRM

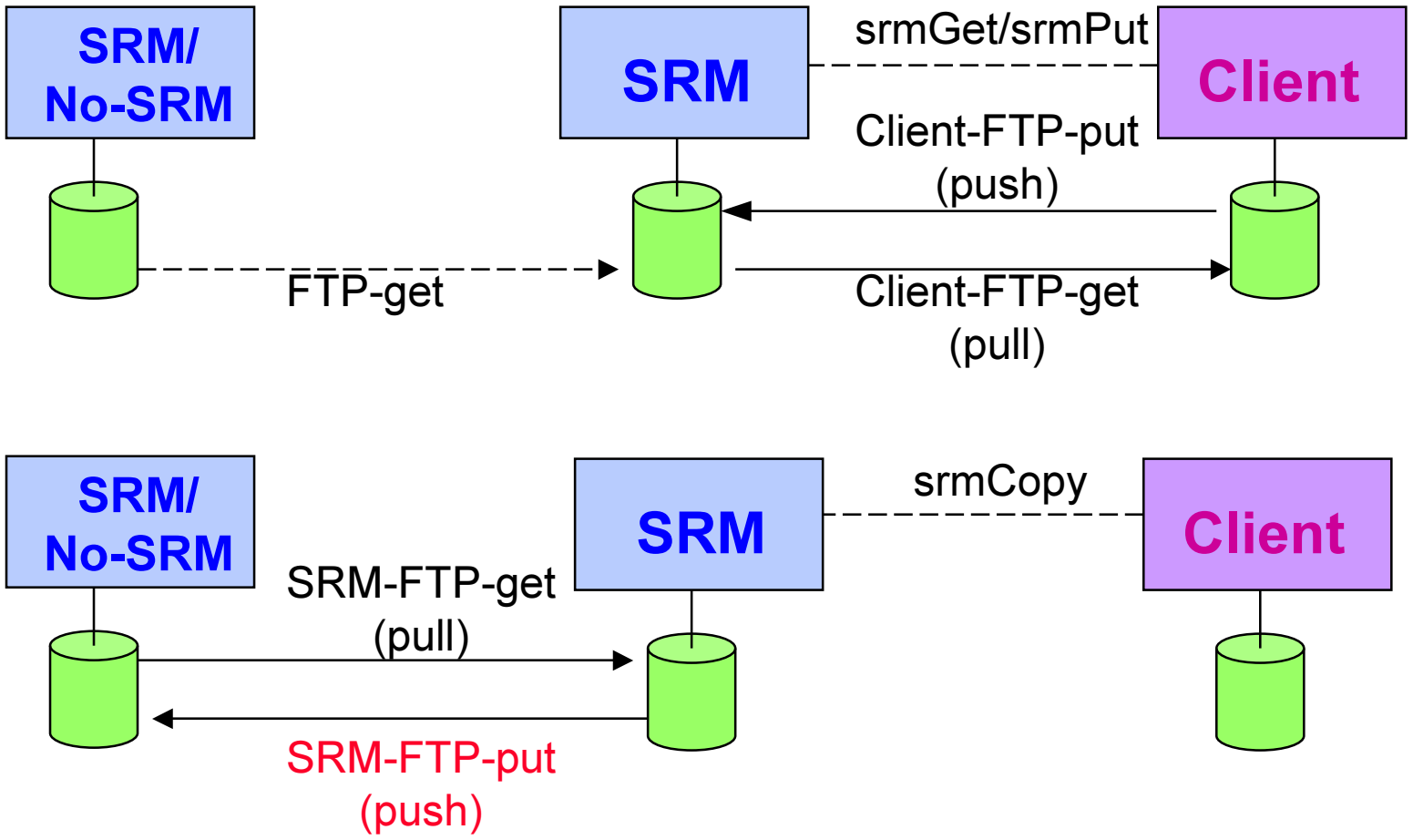
Lifetime Issues

- Option 1: Relative time since response to request is made?
  - e.g. 30 min from now
  - what if communication is slow?
- Option 2: Absolute time
  - how to synchronize clocks
- Lifetime granularity – minutes OK?
- Should a lifetime of a file be negotiable?
  - Per request? Per file? Per client?
  - If so, need to add confirmation
- Alternative: not negotiable
  - Volatile files: srm's choice
  - Durable: client's choice (not exceeding space lifetime)

Security Issues

- How to support multiple security models
  - GSI, Kerberos, https, httpg, GSI, ftp, gsiftp, etc…
- Need a common ground
  - GSI over HTTP (httpg) for communication
  - gsiftp for data transfer
  - E.g. expereince with GSI – kerboros map (at Fermi?)
- How to deal with systems that have their own internal security
  - E.g. HPSS at NERSC
- Who is enforcing file access permissions in a Virtual Organization (VO)
  - If CAS in the future, what mechanism?
  - How do SRMs report usage to CAS?

Revisit Features: srmGet, srmPut, srmCopy



Revisit Features: get, put

- **srmGet, srmPut: relative to client**
  - **srmGet: (Client wants to pull a file from the SRM)**
    - **If file already in SRM space => pin, return location**
    - **If file not in SRM space => allocate space, get file from its source location, pin, call-back when file arrives**
    - **New: Can specify location in a directory, file-type, space-type**
    - **Followed by srmRelease**
  - **srmPut: (Client wants push a file into SRM)**
    - **Allocate space, return location**
    - **New: Can specify location in a directory, file-type, space-type**
    - **Followed by srmPutDone**

Revisit Features: copy

- srmCopy: relative to SRM
  - srmCopyPull: Copy from remote location to SRM
    - Call-back when done
    - Option: notify remote source to release file when done
  - srmCopyPush: Copy from SRM to remote location (add?)
    - Call-back when done
    - Option: Release file when transfer done
      - Example scenario: ask an HRM to push file to a site that have only GridFTP
      - Discussion: Don Petravick

Revisit Features: directory functionality

- Should we allow specification of a directory for srmGet, srmPut, srmCopy?
  - srmGet
    - Now: srmGet {LFN, SURL}
    - Add: srmGet (Dir=path/directory_name)
    - Recursive? No
  - srmPut
    - Now: srmPut {LFN, StURL}
    - Add: srmPut [{LFN}, StDir= path/directory_name]
    - Recursive? No
  - srmCopy
    - Now: srmCopy {source-URL}
    - Add: srmCopy (source-URL, target-StDir)
    - Add: srmCopy (source-StDir, target-StDir)
    - Recursive? - srmCopy (source-StDir, target-StDir, - r)

Revisit Features: LFNs

- LFN can be very long
  - Long path name, long filename
- Any ideas
  - SRMs could assign unique IDs internally
  - Should IDs be visible externally, like RDBMSs?

Revisit Features: others

- "Call_backs" through WSDL – handle
  - Since supported by WSDL, why not include in Basic Version?
- File and system status functionality
  - Add: which files in file-set are currently in SRM cache
  - what should be reported that is useful for planning?
  - what is useful for advertising to RLS and MDS?
  - should status be active or passive?
- Add max-file-length to get/put/copy requests?
  - Expect the system to allocate max-file-length and adjust space after file is transferred
  - Expect system to kill transfer if file size exceeds max-file-length
- Remove "srm" from all commands except get, put, copy?

New feature: include Replica Catalog connectivity

- Replica Catalogs
  - RLS, Jlab-RepCat, …
  - Private catalogs: STAR-catalog, Babar-SRB, …
- Replica management
  - Globus: grid-ftp + RLS
  - Desirable: SRM + some catalog
- File Replication Service
  - Robust copying of large number of files (1000's)
  - Automatic recovery from transient failures
  - Logs, dynamic tracking and monitoring
  - Automatic registration to replica catalogs
- Add: post-hook?
  - After each file gets transferred, call post-hook service
- (Also add: pre-hook?)
  - Call filtering function before file is transferred

SOAP Inter-Operational issue

- SOAP implementations are not compatible in complex data types, specially with string array and struct array.
  - Most array test failures are due to the lack of support for SOAP sparse array and partially transmitted array encodings. Array tests with 'regular' arrays are mostly okay and interoperable
- Need a common ground: OGSA/GT3

SOAP InterOp test

	String Array	Integer Array	Float Array	Struct Array	Struct to Simple	Simple to Struct	2D String Array	Nested Struct	Nested Array
gSOAP	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS
SOAP::Lite	PASS	PASS	PASS	PASS	PASS	PASS	FAULT	PASS	PASS
Apache 2.2	FAULT	PASS	FAULT	FAULT	FAULT	FAULT	FAULT	FAULT	FAULT
Apache Axis	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS
ASP .NET	FAIL	PASS	PASS	FAIL	PASS	PASS	FAULT	PASS	FAIL
easySOAP++	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS
GLUE	FAIL	PASS	PASS	FAIL	PASS	PASS	FAULT	PASS	FAIL
MS SOAP 3.0	FAULT	PASS	FAULT	FAULT	PASS	PASS	FAULT	PASS	FAULT
nuSOAP	FAIL	PASS	PASS	FAIL	PASS	PASS	FAULT	FAULT	FAIL
PEAR SOAP	FAIL	PASS	PASS	FAIL	PASS	PASS	FAULT	PASS	FAIL
SOAP4R	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS
Spheon jSOAP	PASS	PASS	PASS	PASS	PASS	PASS	FAULT	PASS	PASS
SQLData	PASS	PASS	FAULT	FAULT	PASS	PASS	FAIL	PASS	PASS
White Mesa	FAULT	PASS	FAULT	PASS	PASS	PASS	FAULT	PASS	PASS
XMLBus	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS
xSOAP	FAULT	PASS	FAULT	FAULT					

Source: http://www.cs.fsu.edu/~engelen/soap.html

With gSOAP client

SOAP InterOp test

	String Array	Integer Array	Float Array	Struct Array
Apache Axis	Down	Down	Down	Down
Apache SOAP	Down	Down	Down	Down
ASP.Net	OK	OK	OK	OK
EasySoap++	OK	OK	OK	OK
Glue	FAIL	FAIL	FAIL	FAIL
gSOAP	OK	OK	OK	OK
IONA XMLBus	OK	OK	OK	OK
MS STKV3 Typed	OK	OK	OK	OK
NuSOAP	Fault	Fault	Fault	Fault
Pear	OK	OK	OK	OK
Soap::Lite	OK	OK	OK	OK
Spheon JSOAP	Fault	Fault	Fault	Fault
SQLData	OK	OK	OK	OK
Sun JAX RPC	OK	OK	OK	OK
White Mesa	OK	OK	OK	OK

	String Array	Integer Array	Floar Array	Struct Array
Apache Axis	Down	Down	Down	Down
Apache SOAP	Down	Down	Down	Down
ASP.Net	OK	OK	OK	OK
EasySoap++	OK	OK	OK	OK
Glue	FAIL	FAIL	FAIL	FAIL
gSOAP	OK	OK	OK	OK
IONA XMLBus	OK	OK	OK	OK
MS STKV3 Typed	OK	OK	OK	OK
NuSOAP	Fault	Fault	Fault	Fault
Pear	OK	OK	OK	OK
Soap RMI	OK	OK	OK	OK
Soap4R	OK	OK	OK	OK
Soap::Lite	OK	OK	OK	OK
Spheon JSOAP	Fault	Fault	Fault	Fault
SQLData	OK	OK	OK	OK
White Mesa	OK	OK	OK	OK

Source: http://www.apache.org/~rubys/ApacheClientInterop.html
With SOAP 2.3 and Axis beta 3 clients
According to http://www.whitemesa.com/interop/proposal2.html
And http://www.whitemesa.com/interop/proposalB.html

Arie Shoshani – Dec 2002

What else?

- Issues we do not want to discuss now, but may be important in the future
  - Interacting with RLS and other catalogs
  - Pre/post processing hooks
  - Advertising SRM capabilities
  - Providing soft guarantees for planning
  - …