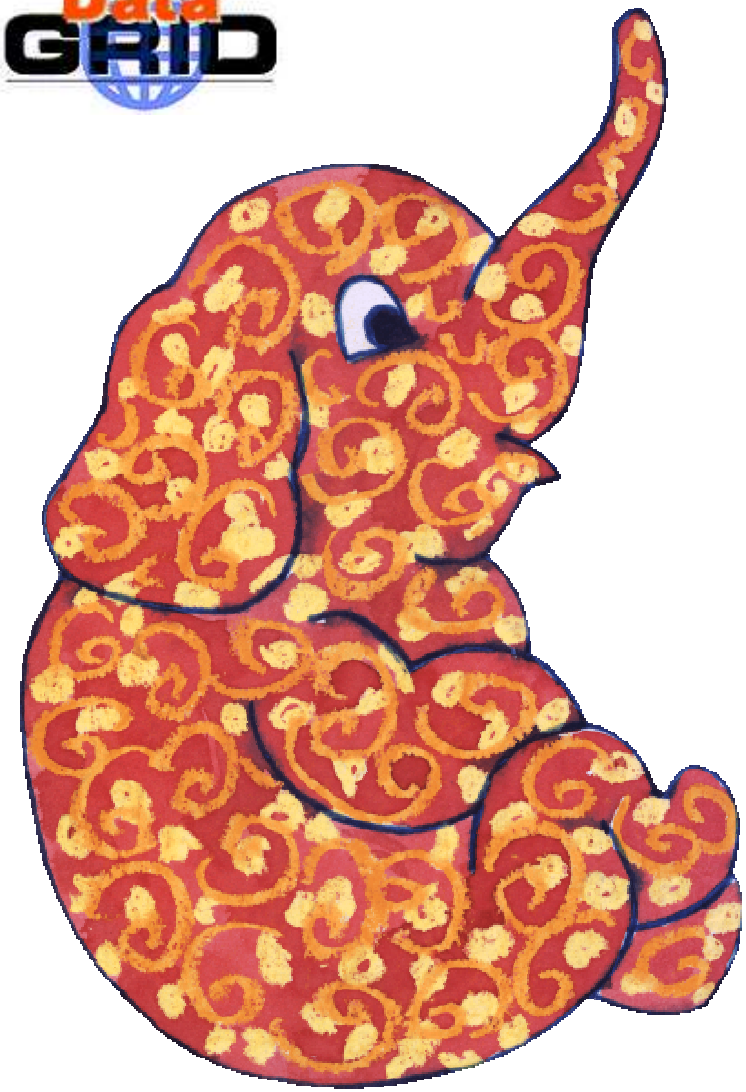


Open Issues

WP5 Storage Elephant

Tim EVES, Jens G JENSEN, Owen SYNGE, Regina TAM

j.jensen@rl.ac.uk

Architecture and Security
ATF, April 2002

This presentation will be updated to reflect changes:
Last updated 08 April 2002

Part 1: Architecture

- Dependencies
- Vagueness
- Redundancies
- Holes

Dependencies

	Currently	Additionally, in future
Short term SE	• Globus 2 (gsiwuftpd) • GDMP	• Slashgrid? • RFIO
Long term SE	• Globus 2 • OpenLDAP	• Replica Manager (WP2) • XML libraries (SAX) • Slashgrid • RFIO • Crypto libraries (which?)

Vagueness

- See Open Issues, at the end of this presentation

Redundancies

- Authorisation implementation (not written yet, discussed later in this presentation)
- Logging (not written yet)
  - Auditing
- Configuration (not written yet)
  - Must have a easy-to-use interface for administrators

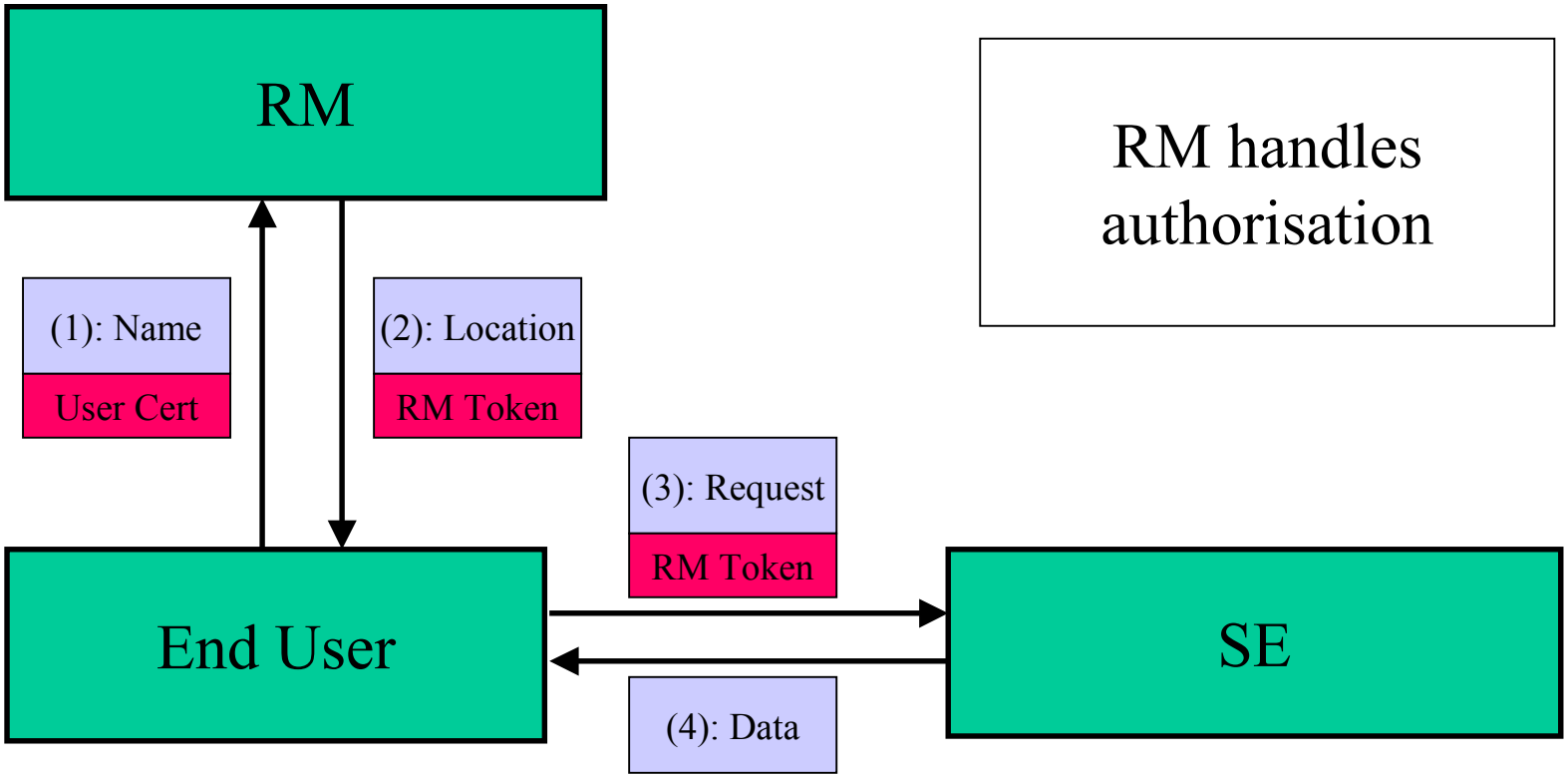
Holes

- See Open Issues, at the end of this presentation

Part 2: Security

- File access model - authorisation
- Authorisation of SE to underlying MSS

Access to Replicated Files

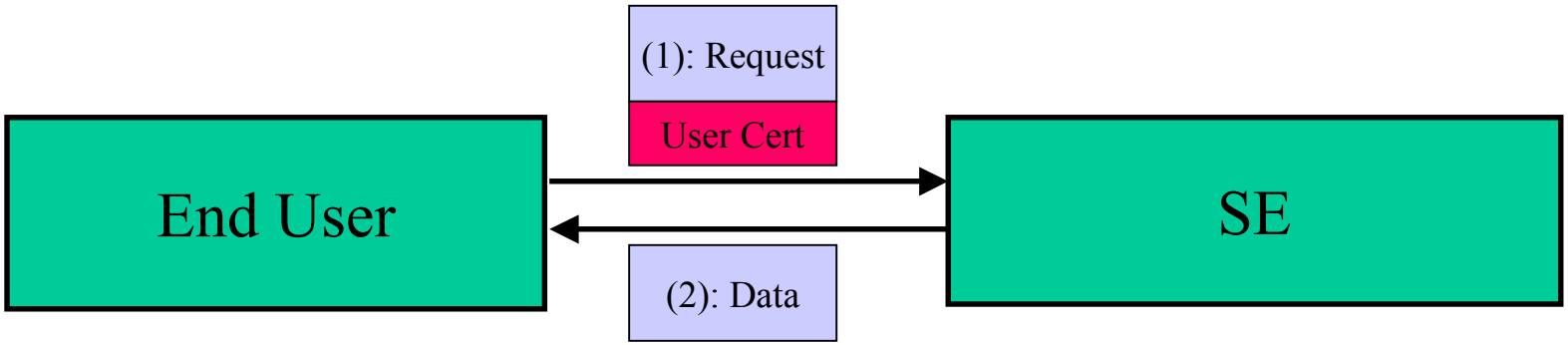


- (1) The Client asks a catalogue to provide the location of a file
- (2) The catalogue responds with the **name** of an SE plus a **token** allowing the client to fetch the file from the SE
- (3) The client asks the SE for the file using this token

Access to “Private” Files

(No RM !)

SE handles authorisation



• (1) The Client asks the SE to provide a file (client knows TFN)
• (2) SE sends data to the client

File Access Models

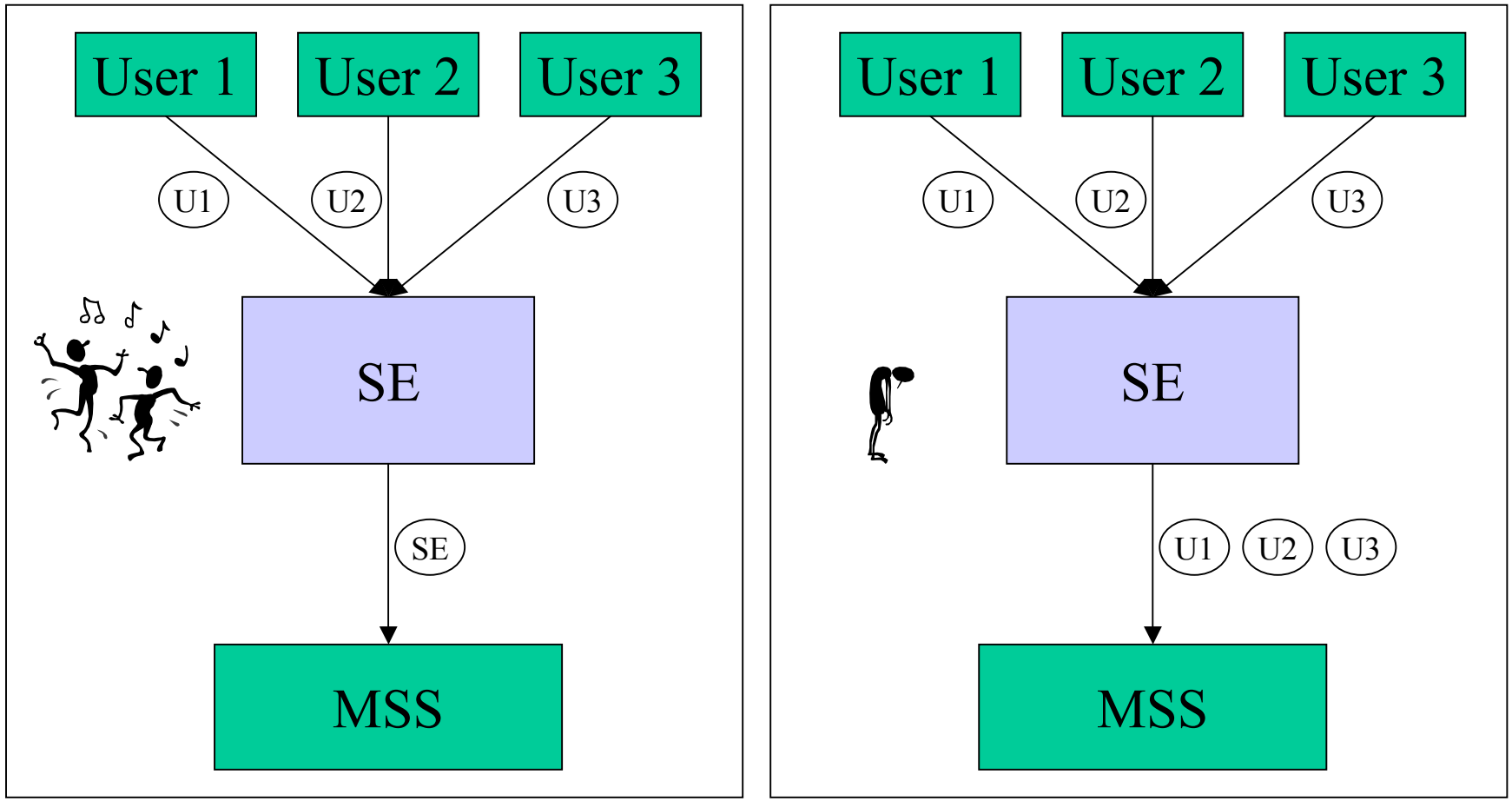
Agreed with WP2 and A. Frohner (WP7, security):

- RM performs user authorisation
  - RM is allowed *all file rights* in a particular VO and *delegates* these to end users.
  - Used for "Grid"-files, i.e. files that (may) get replicated
  - When the user accesses a file, the RM should check with the SE that stores the master copy whether permissions have changed
- SE performs user authorisation
  - Used for "local" files: files created in the SE but not managed by a RM. E.g.:
    - temporary files
    - "private" files

Since all clients are treated the same way, both models can be supported easily with the proposed SE architecture

MSS Authorisation

How SE appears as a client to the underlying MSS: 2-3 different models



MSS Authorisation, Model 1

- The SE handles *all* authorisation to files managed by the SE
- The MSS only sees one user, the SE. The SE can access *all* the files it manages
- Advantages
  - Simple for the MSS manager - only need one user
  - Simple for the SE: no need to pretend to be someone else
- Disadvantages
  - Non-SE access to files stored in the MSS: need additional MSS ids to enforce access control?
  - MSS admin must have a way of asking “who owns this file in my MSS?”

MSS Authorisation, Model 2

- The SE passes on a user id to the MSS
- Advantages:
  - Some authorisation handled by the MSS - no need to store file access permissions (doesn't really simplify the SE, though; the SE must be able to manage permissions anyway)
  - Allows same users to access files outside the MSS
  - MSS manager doesn't need to trust the SE
- Disadvantages:
  - MSS management complexity: A (potentially large) number of users must be able to access the MSS
  - SE complexity: e.g., if the MSS user id is a UNIX user id, the SE must be able to switch user, i.e. run as root

Local UNIX user ids

Allocated from a *pool*

- Need fewer ids
- Persistency: next time the user accesses the file, the uid will be different?

(As an example, we currently need 160 accounts on CASTOR and HPSS owned by the Grid =sysadmin unhappiness)

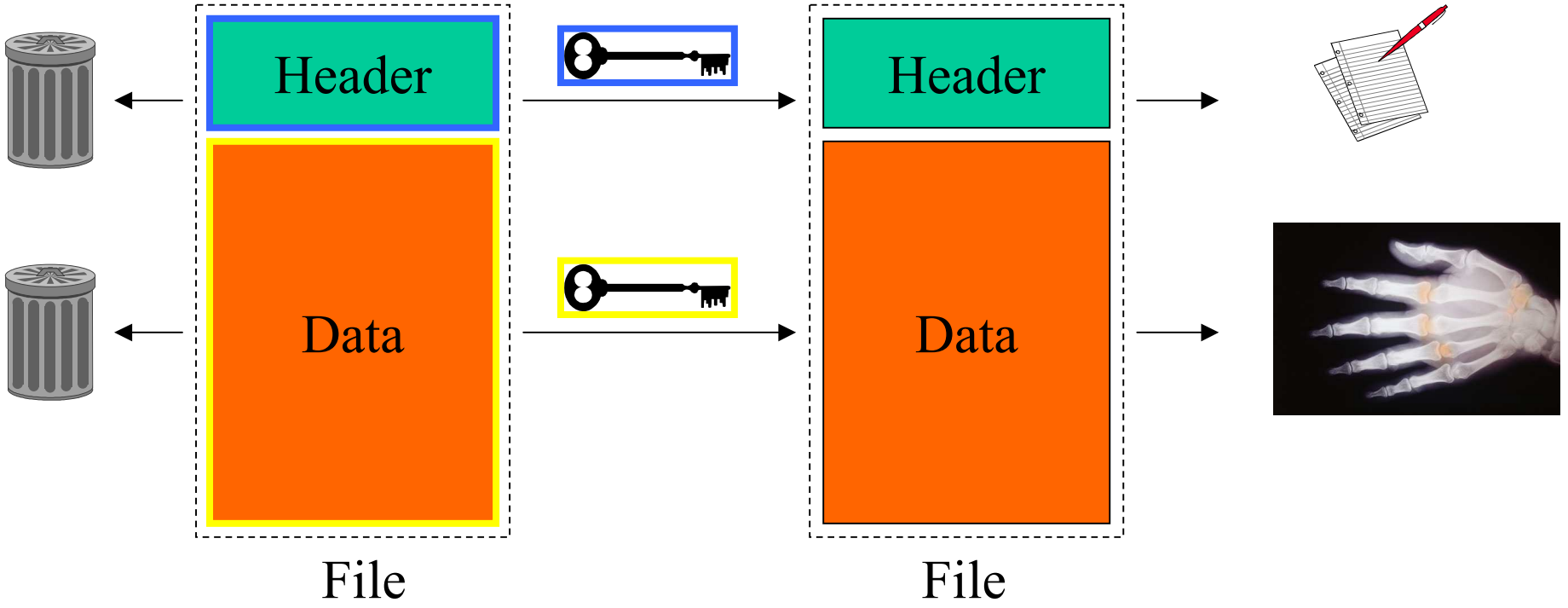
Each user has a unique id

- Need many ids (we have 32 bit uids in Linux 2.4)
- Difficult to administrate? (Large gridmap file/database...?)

Part 3: Encryption

File encryption

- Different slices of file (« tranches ») enciphered with different keys
- This will be implemented on top of *partial file transfers*
- The SE stores the whole file but doesn't know about the slices: the client must know where the slices are and which slice is encrypted with which key



Part 4: Open Issues

- Current practical issues
- Some thoughts on *robustness*
- Major issues
- Minor issues

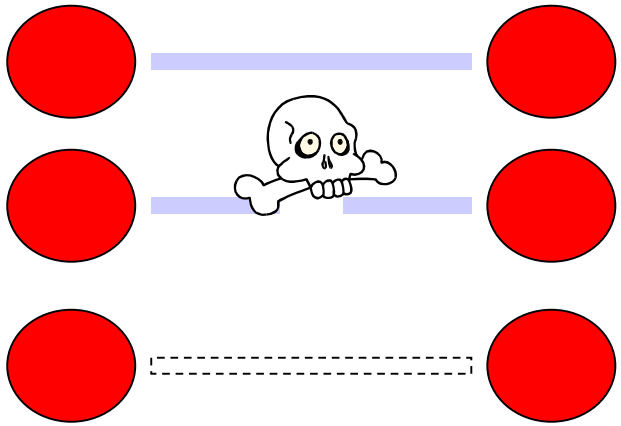
Current practical issues

- We need to scan the *log files* to see who owns a file in HPSS and CASTOR
  - Need to scan several files to match files, uid, and DN with the Globus gatekeeper to get find the files owned by a person, or find out which person owns a particular file
  - Completely different log file for gsiwuftpd
- SE information (LDAP) servers aren't stable (as a general rule they're down when we try to demonstrate the query program to someone) - WP3?
- We spend too much time getting "practical" things like RPM, CVS, and configure to work
- We need 160 Grid accounts on CASTOR / HPSS

Robustness

Connection protocols:

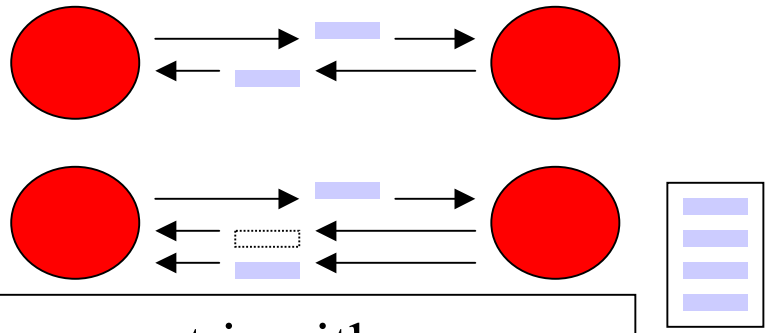
1. FTP connection is opened
2. Network or host dies
3. Network/host comes back but connection is broken - client must open new connection and resume transfer



Connectionless protocols:

E.g.: HTTP packets

Request queue: packets are queued and acknowledged



- Need higher level connection management in either case
- Easier with connectionless protocols?
- Need to see how well GridFTP can cope with this

Open Issues, 1

- OGSA (http://www.globus.org/ogsa)
  - What does it mean for the SE? (More an *architecture* question than a *security* question)
  - OGSA *may* influence design of internal protocol
- Service based authentication (A. Frohner, WP7)
  - Mutual authentication between host/service and client - should be easy to support with the proposed SE architecture
- NFS or /Grid?
  - Can we use Andrew McNab's slash-grid file system (http://www.gridpp.ac.uk/slashgrid) instead of NFS?

Open Issues, 2

- Modifying file metadata (discuss with WP2)
  - Can replicas be modified or only the master copy? If replicas can be modified, what happens if two copies are modified simultaneously?
- Delegations:
  - Can group permissions be delegated? [group perms probably implemented as user perms, so delegation follows from that]
  - Should there be restrictions on who can delegate permissions to whom? (E.g., users can only delegate permissions to users in the same VO.)
    - E.g. user can delegate to all /O=Grid/OU=UKHEP/*
    - E.g. user can delegate for files /grid/ukhep/foo/*

Open Issues, 3

- CE disk cache
  - If the CE needs a “Grid aware” local cache, then this cache should be an SE
- Uids: if we need a 32 bit uid space, need to move to Linux 2.4 (e.g. RedHat 7.2)
- IBP (“Internet Backplane Protocol”) (http://icl.cs.utk.edu/ibp/)

Open Issues, 4

- Efficiency of storage systems:
  - MSS not normally 100% full
    - sometimes there is a performance hit if storage system 100% full
    - some MSS are assumed to be always partially empty
  - SE will be filled with data
    - Should there be a published *soft* upper limit?

Minor Issues, 1

- ACL
  - Should the SE support POSIX Access Control Lists?
- JDL (WP1) doesn't specify maximal size of file in sandbox
- File locking issues - remote locking is difficult
  - Co-operative or mandatory locking?
  - Pin is a type of lock so something must be provided
  - WP5: not sure what we can realistically implement
- Use SRM API for control interface?
  - SRM is more than control
  - Few or no other suggestions
  - Can always cook our own…

Minor Issues, 2

- RPMs: not ideal package system - difficult to build and extract without being root.
  - .deb can be built and extracted with standard UNIX tools, gzip, tar, ar.
  - Nice higher level tools, like apt take care of dependencies, etc
  - With .rpm, one package can satisfy several dependencies (confusing), or the dependency doesn't correspond to package name (confusing)
- Support directories on "flat" file systems?
  - Moving directories is the only major difficulty
- Many simultaneous queries from RM?
  - Does the RM have a "handle"?
  - Or does it just bundle all the queries?
- RC verification
  - Is there a verification program?