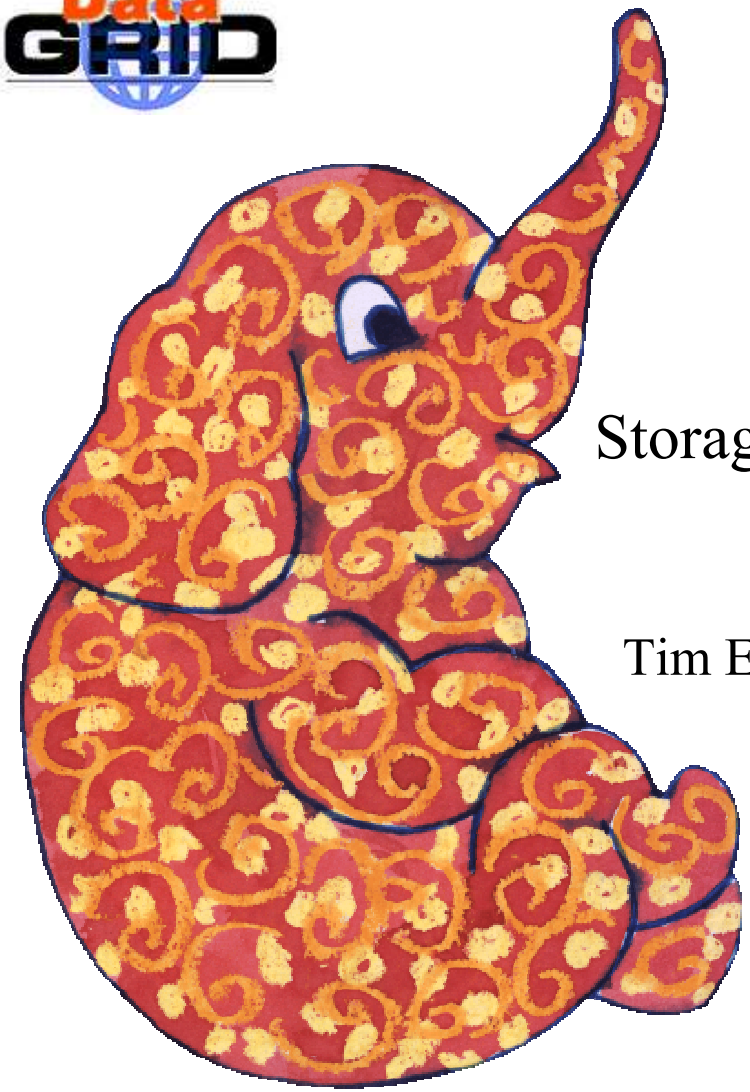


SE: Open Issues

Storage Element Architecture and Security Issues

WP5

Tim Eves, Jens G Jensen, Owen Synge, Regina Tam
j.jensen@rl.ac.uk

ATF, Paris, March 2002

This presentation will be updated to reflect changes
Last updated 12 March 2002

Part 1: Architecture

- Dependencies (internal and external)
- Vagueness
- Redundancies
- Holes

External Dependencies

	Currently	Additionally, in future
Short term SE	• Globus 2 (gsiwuftpd)	• Slashgrid • RFIO
Long term SE	• Globus 2 • OpenLDAP	• XML libraries (SAX?) • Slashgrid • RFIO

Internal Dependencies

	Currently	Additionally, in future
Short term SE	• None	• None?
Long term SE	• None	• Replica Manager (WP2)

Vagueness

- See Open Issues, at the end of this presentation

Redundancies

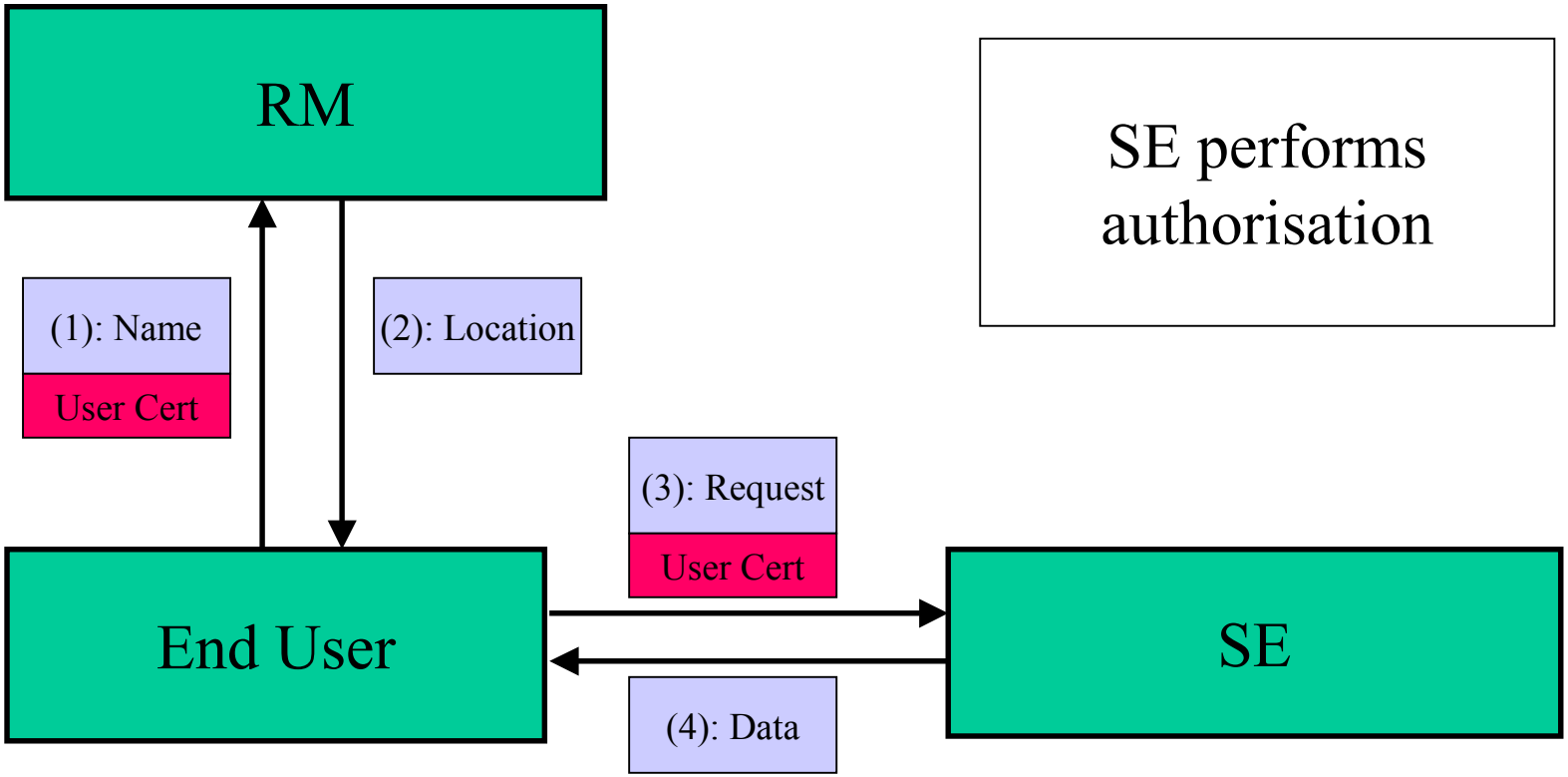
- Authorisation implementation (not written yet, discussed later in this presentation)
- Logging (not written yet)
- Configuration (not written yet)

Holes

- See Open Issues, at the end of this presentation

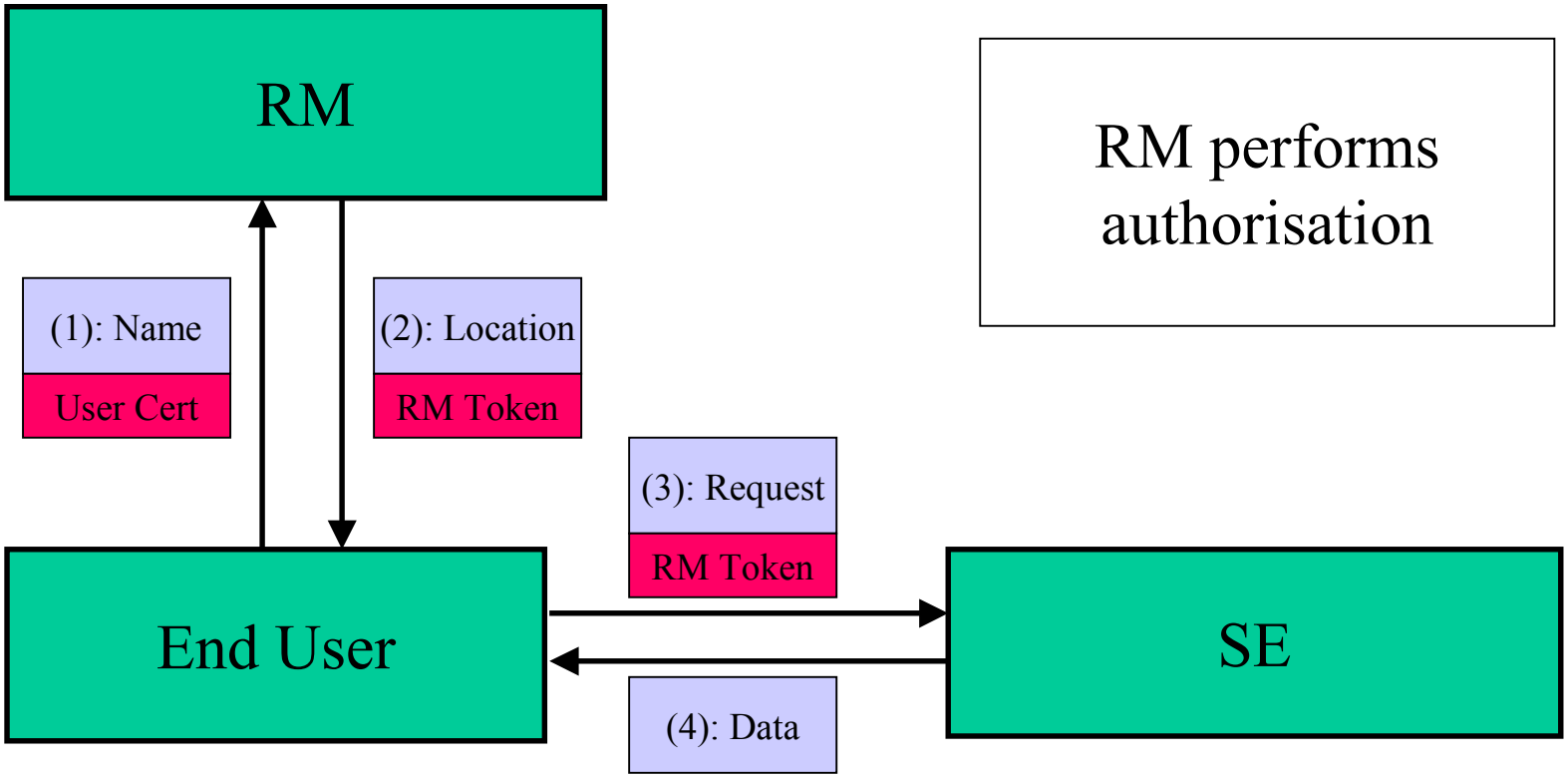
Part 2: Security

File Access, Model 1



- (1) The Client asks a catalogue to provide the location of a file
- (2) The catalogue responds with the **name** of an SE
- (3) The client asks the SE for the file using the its own certificate

File Access, Model 2



- (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

File Access Models

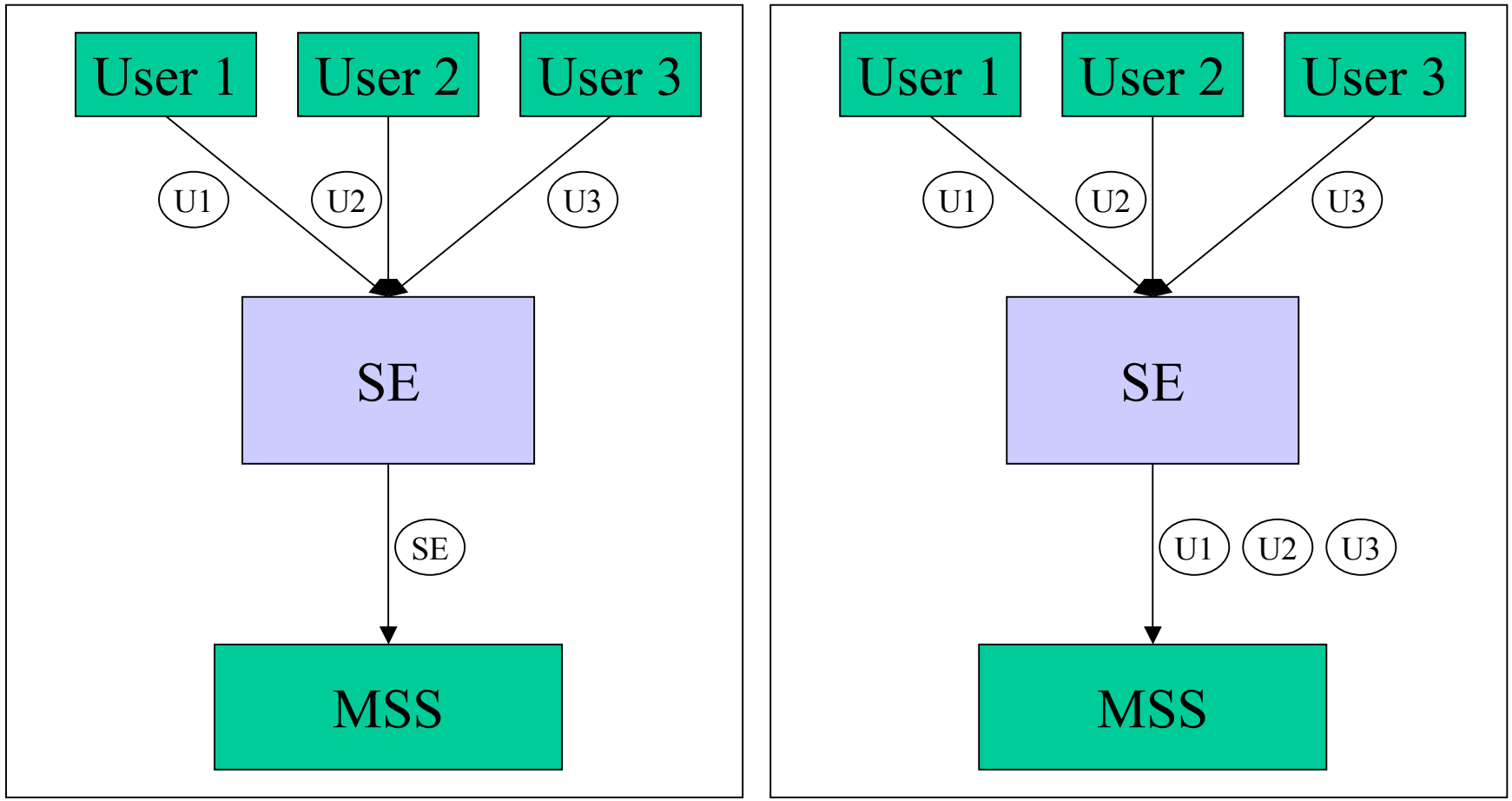
Agreed with WP2 and A. Frohner (WP7, security):

- 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
- RM performs user authorisation
  - RM is allowed *all file rights* 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

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

MSS Authorisation

Two 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
- Disadvantages
  - Non-SE access to files stored in the MSS:

MSS Authorisation, Model 2

- The SE passes on a user id to the MSS
- Advantages:
  - Simpler for the SE: some authorisation handled by the MSS - no need to store file access permissions
  - 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
  - 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?

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...?)

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 will 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
- Users can create files in a SE which are not replicated

  E.g.
  - “private” files
  - temporary files
  - output from programs
- Uids: if we need a 32 bit uid space, need to move to Linux (kernel) 2.4

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…
- Disk encryption (WP10 requirement)
  - Should SE support disk encryption - current WP5 policy is that the client should encrypt the data…?

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?