xrootd
Authentication & Authorization

Andrew Hanushevsky

Stanford Linear Accelerator Center

6-June-06

Goals

- Flexible security architecture
  - Multiple protocols
    - Easily expandable
  - Simultaneous heterogeneous protocols
    - Allow multiple administrative domains
- Simple administration
  - Minimal server configuration
  - **No** client configuration needed

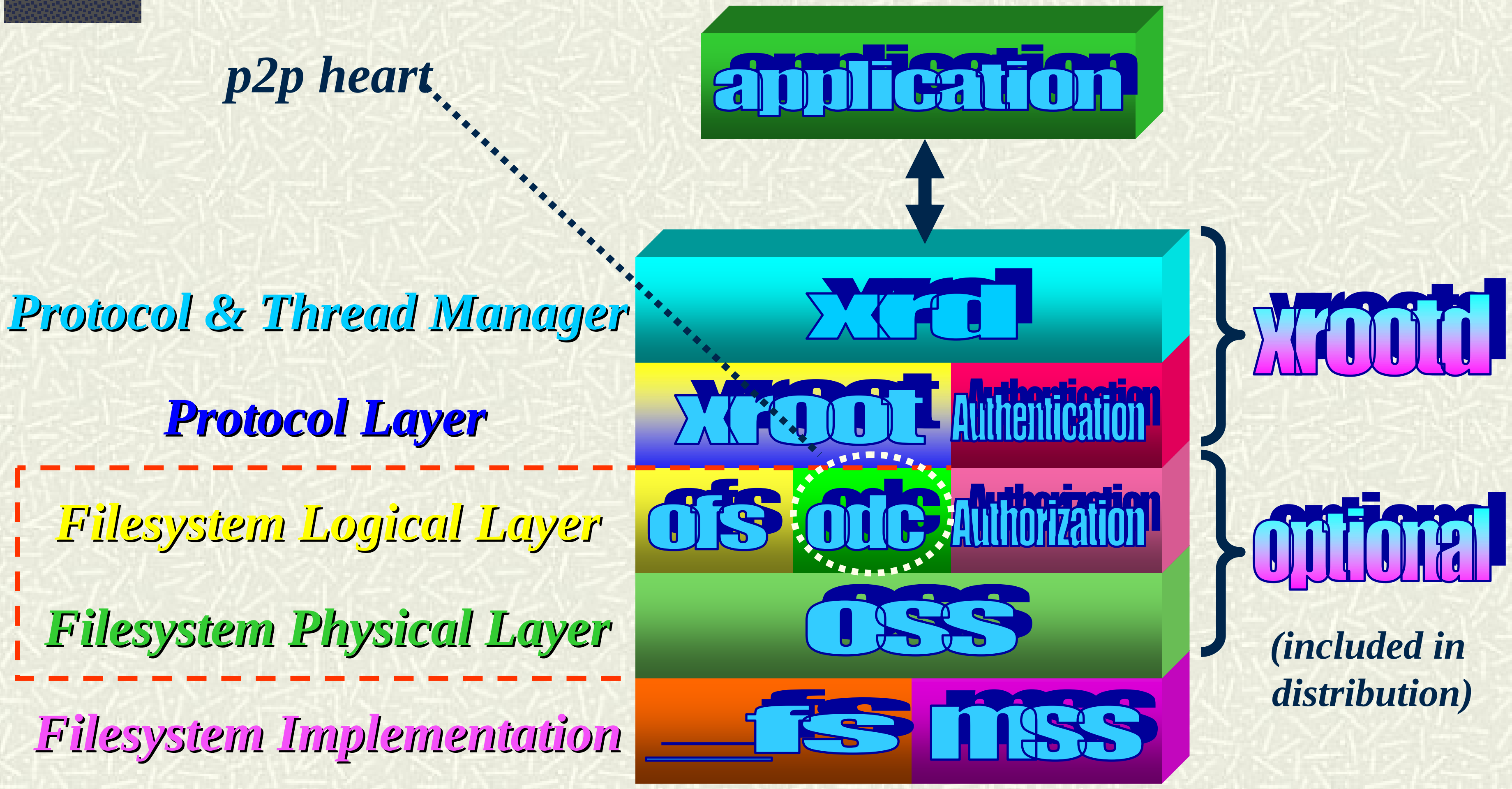
Stanford
Linear
Accelerator
Center

Authentication & Authorization

- Developed as runtime plug-in components
  - Easy to substitute
  - Trivial to extend
- Client/Server architecture plugin aware
  - Designed for flexibility from the start
- Application layer architecture
  - Portable to other application architectures

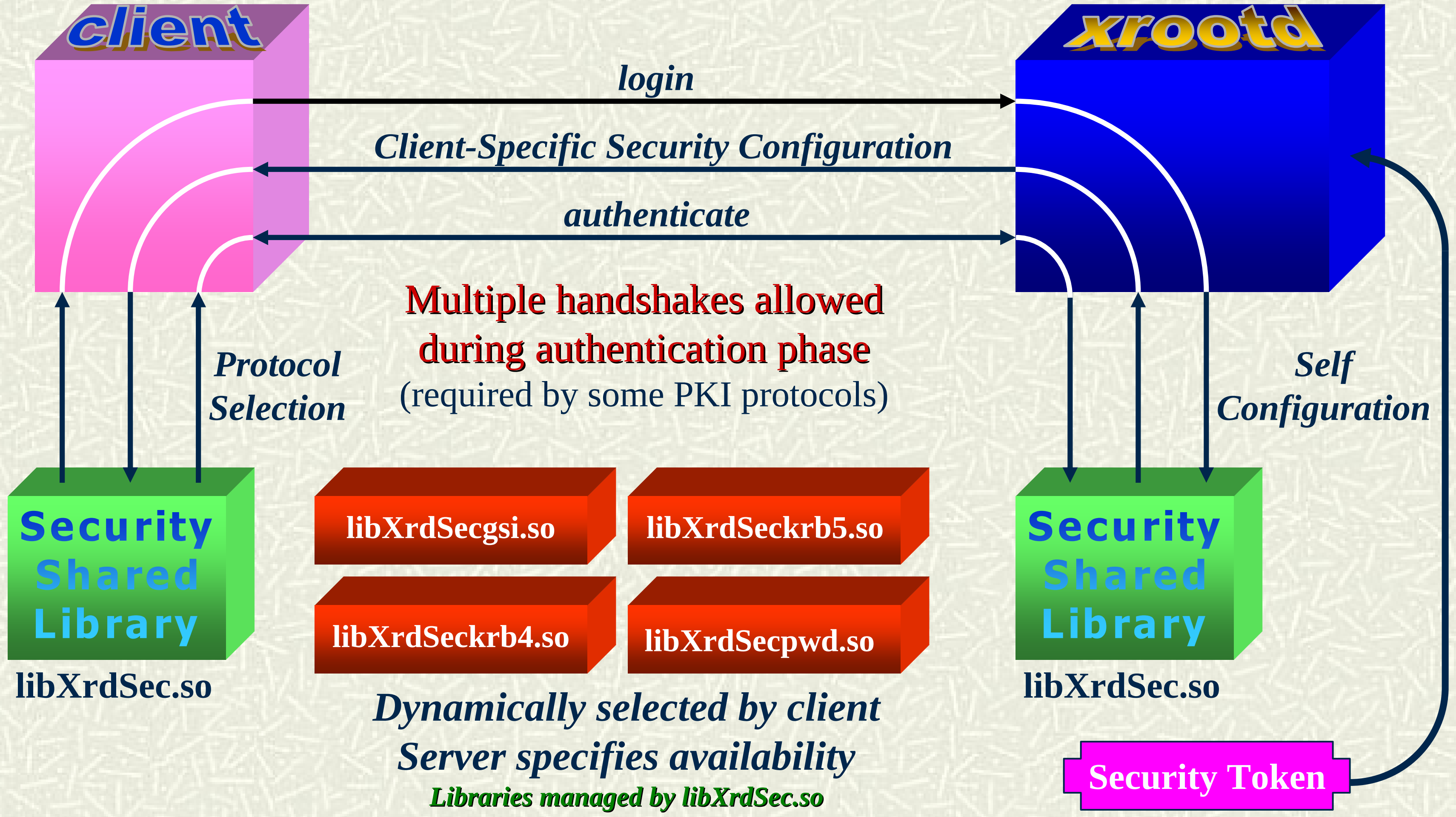
Stanford Linear Accelerator Center

xrootd Server Architecture



Stanford Linear Accelerator Center

Security Architecture



Dynamically selected by client
Server specifies availability
Libraries managed by libXrdSec.so

Stanford Linear Accelerator Center

Authentication I

- Specified by config file directives
  - xrootd.seclib *so_path*
    - xrootd.seclib /opt/rooted/lib/libXrdSec,so
  - sec.protocol [*libpath*] *protid*
    - sec.protocol gsi
    - sec.protocol krb5
  - sec.protbind *hostpat* { **none** | [**only**] *protocols* }
    - sec.protbind * only gsi
    - sec.protbind *stanford.edu krb5 gsi
    - sec.protbind *slac.stanford.edu none

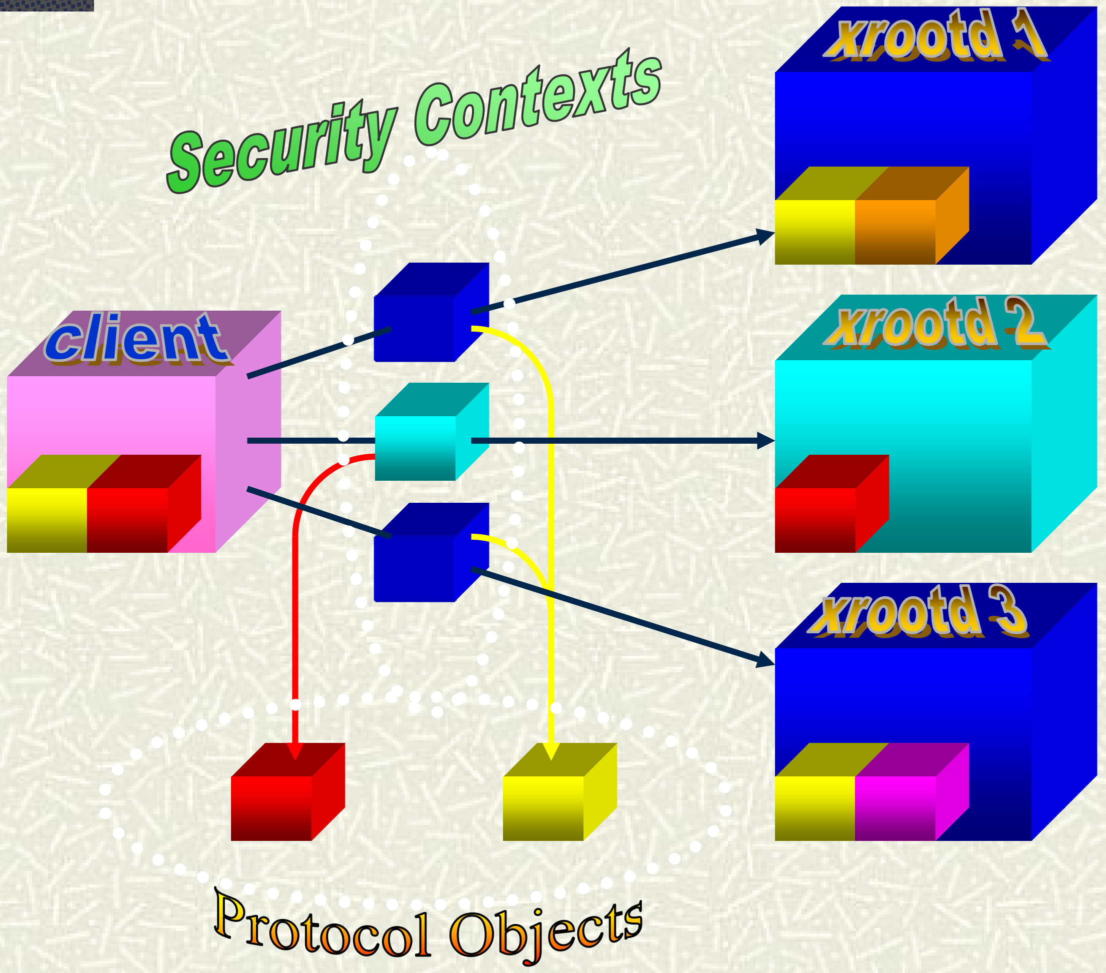
Stanford Linear Accelerator Center

Authentication II

- Server constructs configuration for clients
  - Client specific
  - Information contained in security “token”
- Client needs are simple
  - Protocol manager library + protocol libraries
    - No libraries needed for host authentication
- No fuss, no mess, no bother
  - Server configures the client at run-time

Stanford Linear Accelerator Center

Heterogeneous Security Support



- Servers have one or more protocol objects
- Server protocol objects created at server initialization time
- Client selects which protocol to use when security context created
- Protocol object created based on configuration returned by xrootd
- One security context object per physical xrootd connection
- Protocol objects may be shared by one or more contexts
- Each "pass" through a security context object may generate credentials to be passed to xrootd

Stanford Linear Accelerator Center

Authentication Information

```
char   prot[XrdSecPROTOIDSIZE];  // Protocol used
char   *name;                    // Entity's name
char   *host;                    // Entity's host name
char   *vorg;                    // Entity's virtual organization
char   *role;                    // Entity's role
char   *endorsements;  // Protocol specific endorsements
char   *tident;                  // Trace identifier (do not touch)
```

Passed to file system layer to be used for authorization

Stanford
Linear
Accelerator
Center

Authorization Challenge

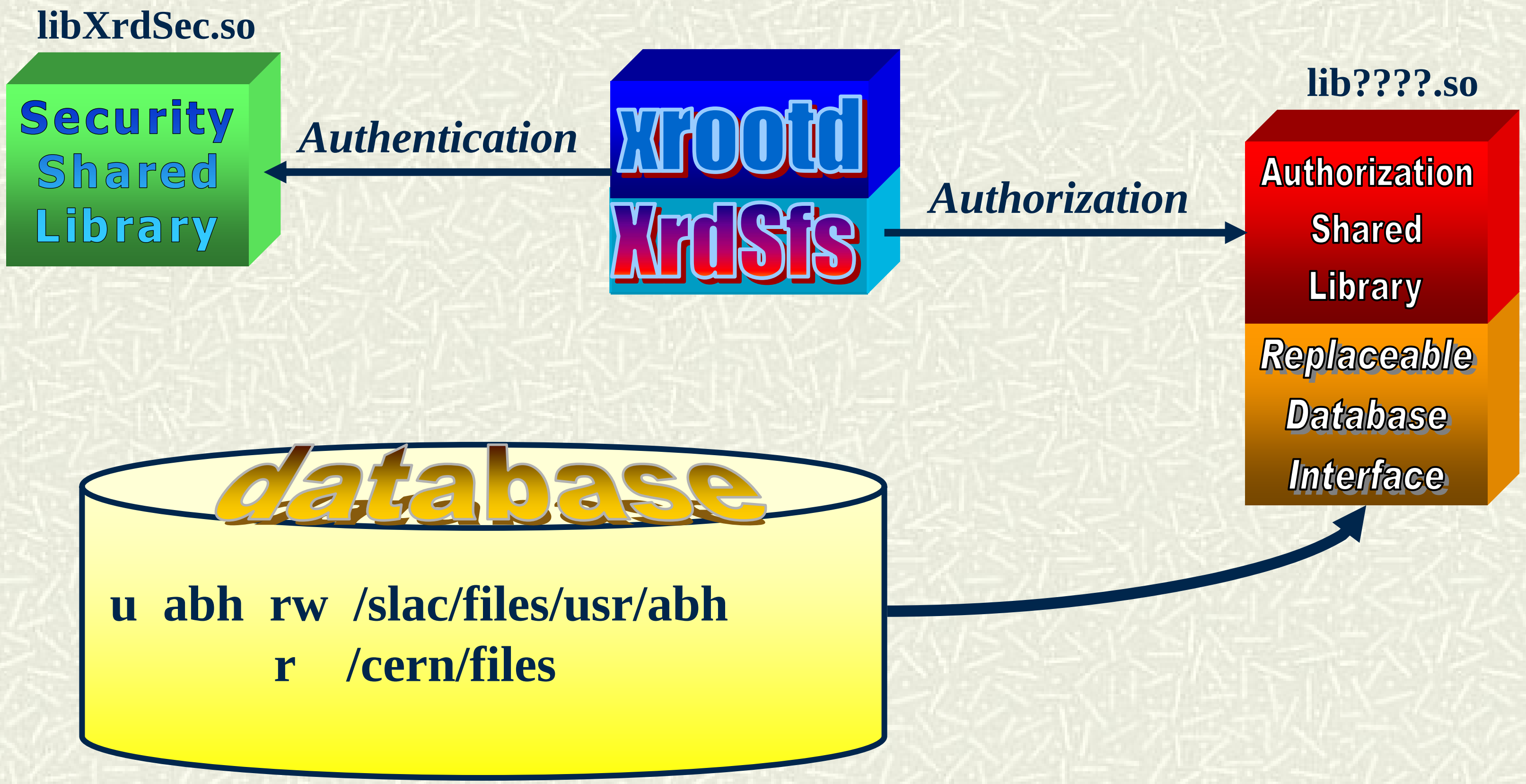
- Number of files
  - Billions and billions of files
    - Amount of data is a moot point
- Access control list model unmanageable
  - Too many files to protect
    - Don’t want to record usernames in many places
- Capability model is manageable
  - Few users relative to number of files
    - Usernames recorded only once
    - Each user given access to arbitrary file space regions

Stanford Linear Accelerator Center

Authorization Approach

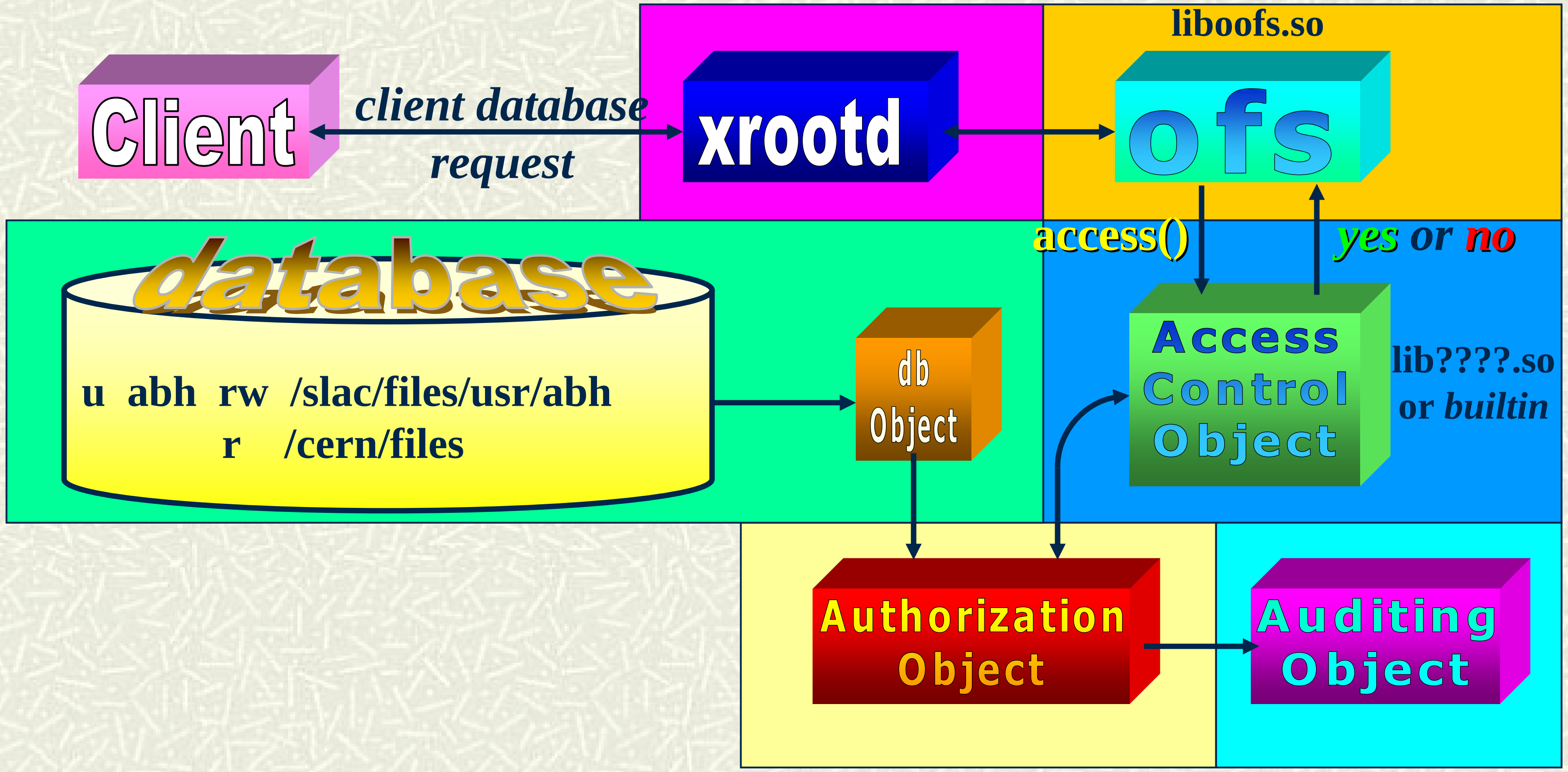
- Works as a plug-in
  - ofs.authlib *path* [*parms*]
  - ofs.authorize
- Default authorization is built-in
  - Basic NT-like access control

Authorization Architecture



```
u  abh  rw  /slac/files/usr/abh
        r   /cern/files
```

Builtin Architecture Detail



Builtin Authorization Model

- Capability based model
  - Each entity has a list of capabilities
  - A capability is a path prefix-privilege pair
    - Any number of such pairs may be specified
    - More scalable when number of objects greatly exceeds number of entities
  - Can mimic an access control model

Entities can be:
Hosts
NIS Netgroups
Unix Groups
Users

```
u  abh  rw  /slac/files/usr/abh
         r   /cern/files
```

Stanford Linear Accelerator Center

Builtin Authorization Entities

- *idtype id* { *path privs* | *tempid* } [• • •] [\ ]
  - **g** - Unix group name
    - Applied when user is a member of the group
  - **h** - Host name
    - Applied when request originates from this host
      - Always fully qualify the host name and specify in lower case
  - **n** - NIS netgroup name
    - Applied when the triplet (hostname, username, domainname) is a member of the specified netgroup
  - **t** - template name
    - Specification substituted in future authorization records for *tempid*
  - **u** - user's name (can be DN)
    - Applied for specific user, as identified by authentication protocol

Stanford Linear Accelerator Center

Special Entities

- **u** = { *path privs* | *tempid* } [• • •] [\ ]
  - User's name replaces the first occurrence of **@=** in *path*
  - ***Fungible*** Allows specializing privileges by user's name without listing all users
    - Only one such entry may exist
  - Example:
    - **u = /usr/@=/files a**
      - User abh has all privileges for /usr/abh/files
- **u *** { *path privs* | *tempid* } [• • •] [\ ]
  - The entry applies to all users regardless of the originating host
  - ***Default*** Essentially default privileges
    - Only one such entry may exist
  - Example
    - **u * /files rws**

Stanford
Linear
Accelerator
Center

Builtin Authorization Privileges

- *idtype id* { *path privs* | *tempid* } [• • •] [\ ]

a - all privileges	**i** - insert (create)	**l** - lookup	**r** - read
d - delete	**k** - lock (unused)	**n** - rename	**w** - write

  - Positive and negative privileges allowed
    - Negative privileges always override positive privileges
  - Examples
    - **u aaa /foo rw**
      - User aaa has read/write privileges in /foo
    - **u abh /foo a-n**
      - User abh has all privileges except rename in /foo
    - **u xyz /foo –wind**
      - User xyz is denied write/insert/rename/delete privileges in /foo

Stanford
Linear
Accelerator
Center

Principal of Least Privilege

- For the first applicable path, if any, in each of
  - Default entry
  - Fungible user entry
  - Specific user entry
  - Entry for originating host
  - All Unix groups in which user is a member
  - All netgroups to which (*hostname*, *username*, *domainname*) applies
- Logically add together positive privileges
  - pos_privs |= new_pos_privs
- Logically add together negative privileges
  - neg_privs |= new_neg_privs
- Final privileges are positive less negative privileges
  - final_privs = pos_privs & ~neg_privs

Stanford Linear Accelerator Center

Entities and Certs

- Some entities equivalent
  - User Name and DN
  - Host
- Others are not clear
  - Group, Netgroup
- Additions are needed
  - Vorg and Role (the definition?)
- Endorsement handling is totally unclear
- *Future Project*

Summary

- xrootd security
  - Fully configurable, extendable, even replaceable
- Standards-based authentication
  - GSI
  - Kerberos (version 4 or 5)
  - Host based
  - Password
- Builtin Capability-based authorization
  - Extensive privilege support
  - Auditing
- Good model for application level security