

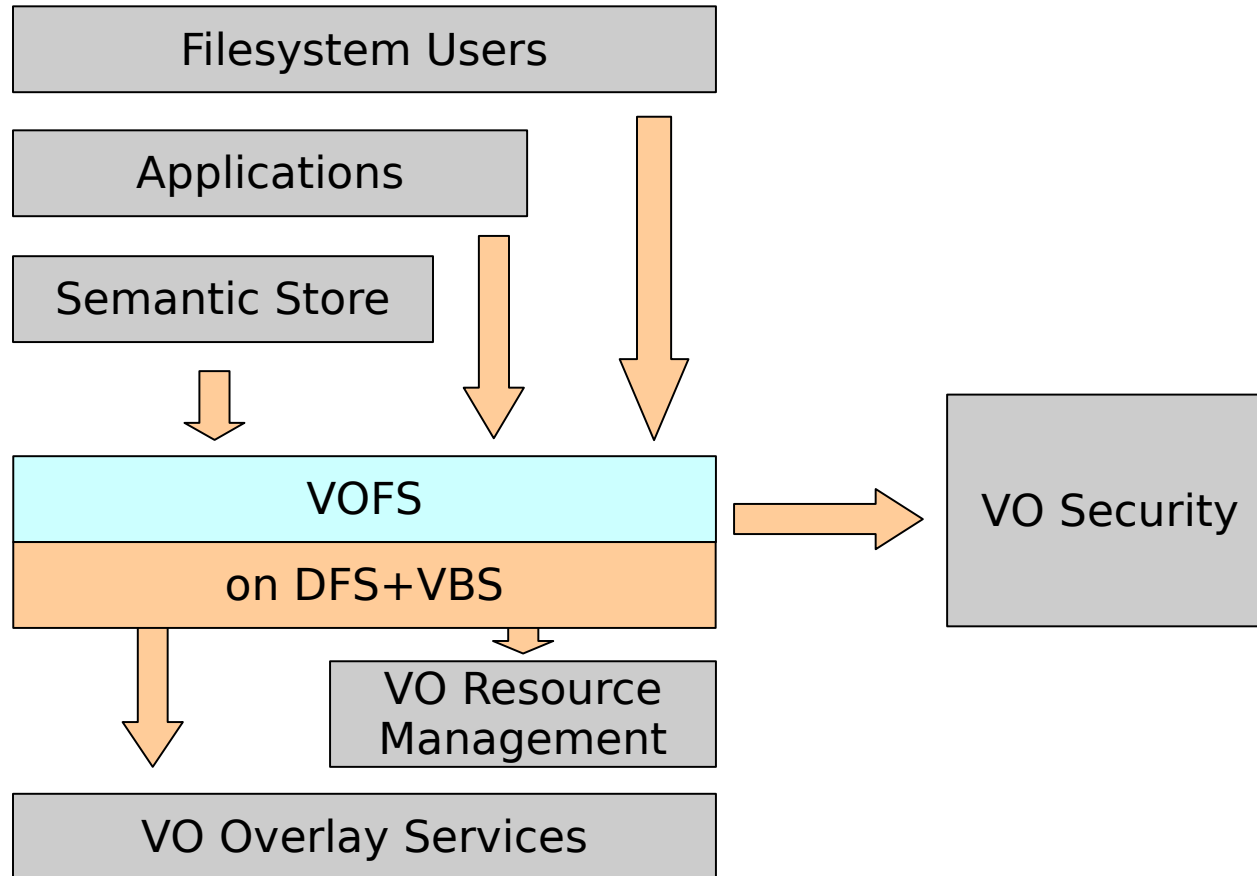
# VOFS: Federative File System for Grid4All

Georgios Tsoukalas, ICCS  
gtsouk@cslab.ece.ntua.gr

*Paris, July 2008*



# VOFS in the Grid4All architecture



# Character of VOFS

- Federation
  - put files and storage together
- User-oriented, personal tool like a web browser
  - not a system one like a web server.
- Pub-Sub notifications, disconnected operation, workspaces, POSIX
- Filesystem as distributed platform for applications
- Best-effort consistency

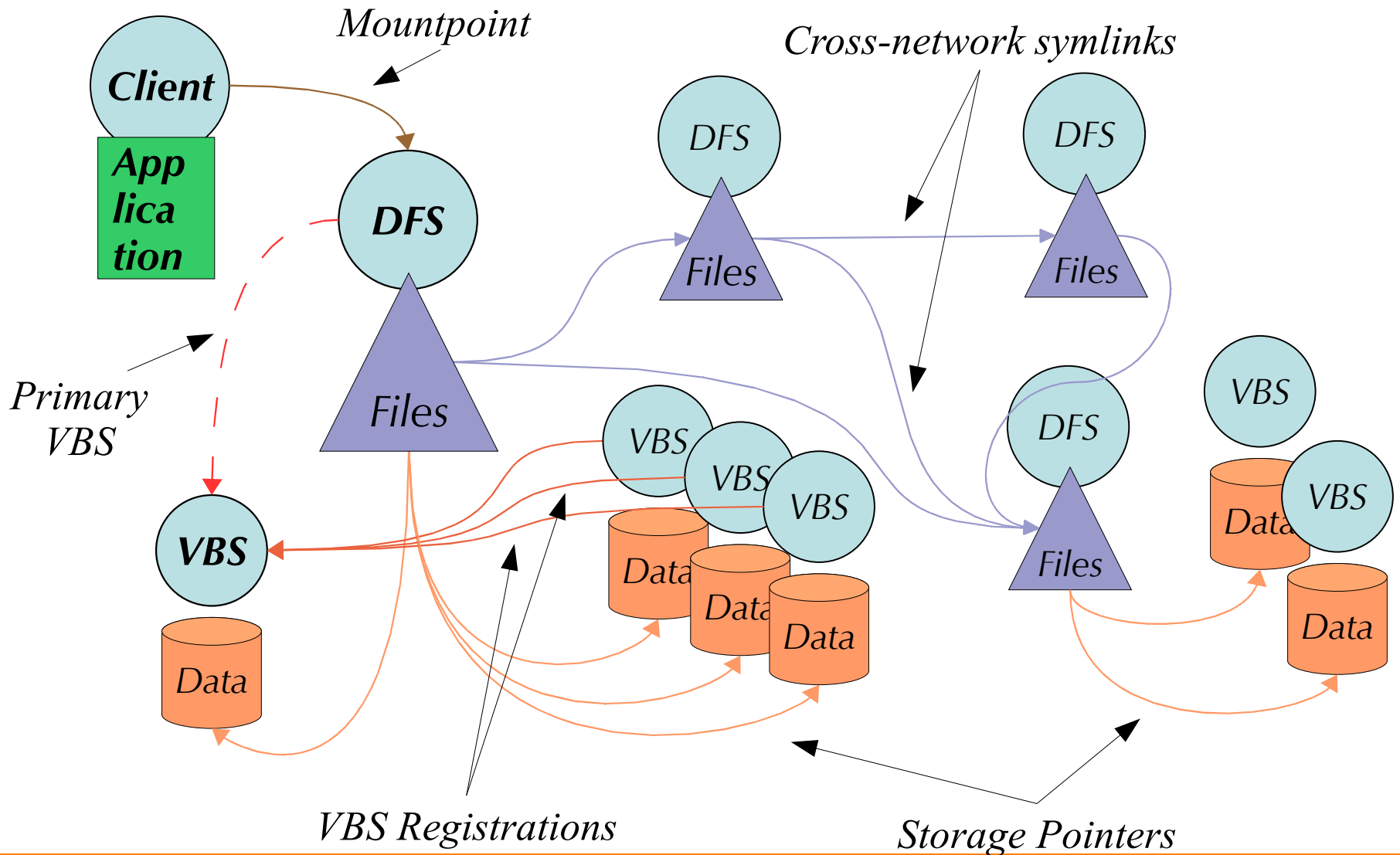
# XtreemOS Contrast

- Not focusing on creating a VO or policies
  - but on exploring tools/mechanisms useful by persons in this context
- Not managing inside the node
- Not focusing on establishment of commercial workflow
  - but rather on personal use as a tool
- Focus on social aspect of self-management
  - Node administrators are VO users

# How does VOFS work?

- Constructs folders from distributed files
  - and files from distributed storage, by linking
- Browses the resulting web of files
- Each VO has its own VOFS presence
  - where members or other providers link in their contributions
  - which arranges the hosting of produced VO files and data
- Peers and resources identified globally (e.g. DNS or DHT)

# VOFS Network



# VOFS Network

- Built upon DFS (file serving) and VBS (storage serving) subsystems
- Each peer has a cryptographic identity
- Each peer serves its own file namespace and storage, as the Authority for them
- Authority means having access to the single master copy of a resource
- Clients make access requests to authorities and cache resources

# URIs

`dfs://<user>:<service>/<path>` ← *URI format*

`dfs://iccs:fs/shared/profile.pdf` ← *File URI (DFS)*

`dfs://iccs:storage/cG9saXRpY2FsbHkgY29ycmVjdAo`

`dfs://iccs:storage/shared/profile.pdf` ← *Storage URI (VBS)*

## INODE Object

```
.uri = dfs://iccs:fs/shared/profile.pdf
```

```
...
```

```
.data = <offset> <uri>
```

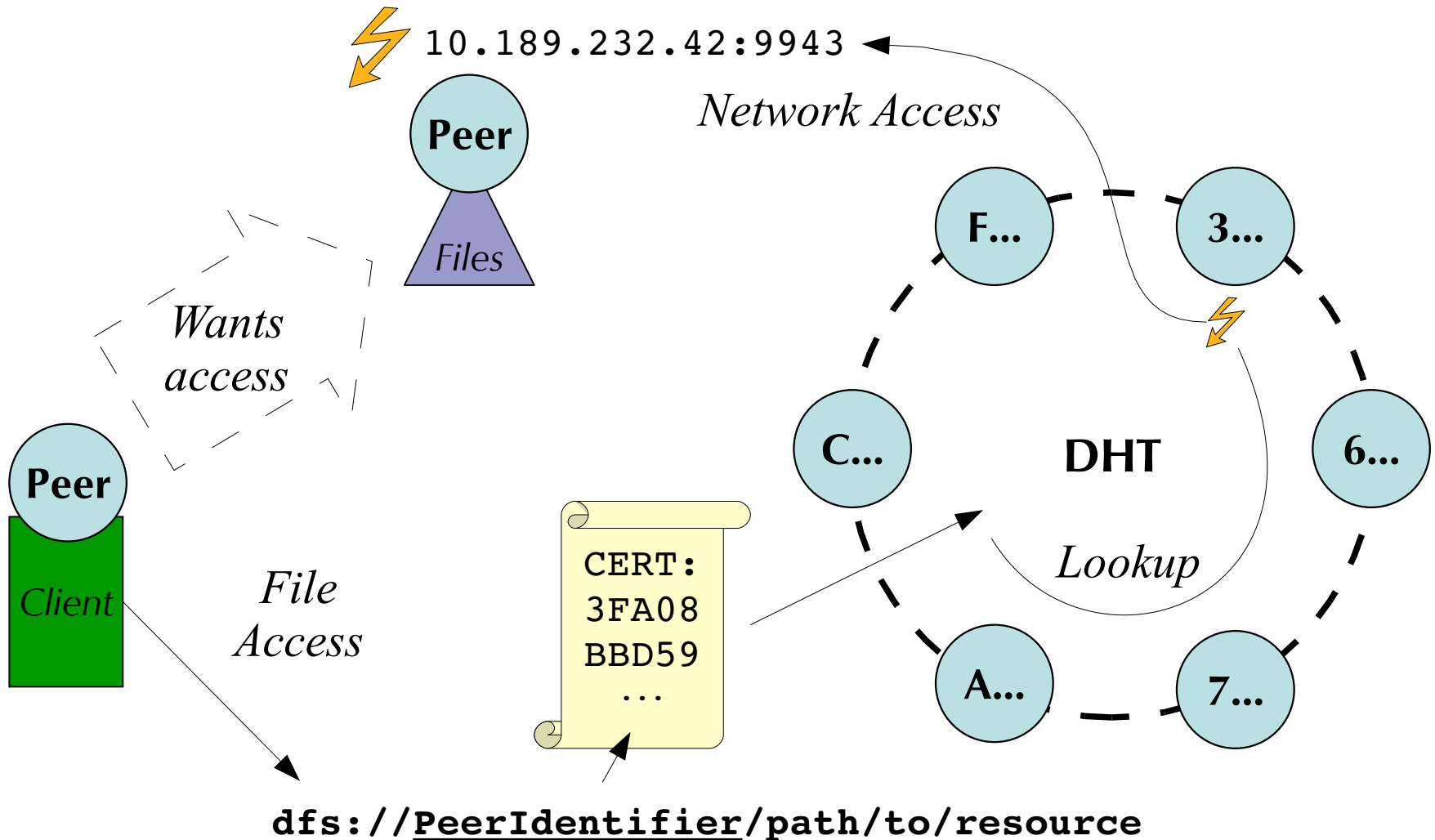
```
00000000 dfs://iccs:storage/cG9saXRpY
```

```
00001000 dfs://ntua:storage/066d66d44
```

```
00008000 --
```

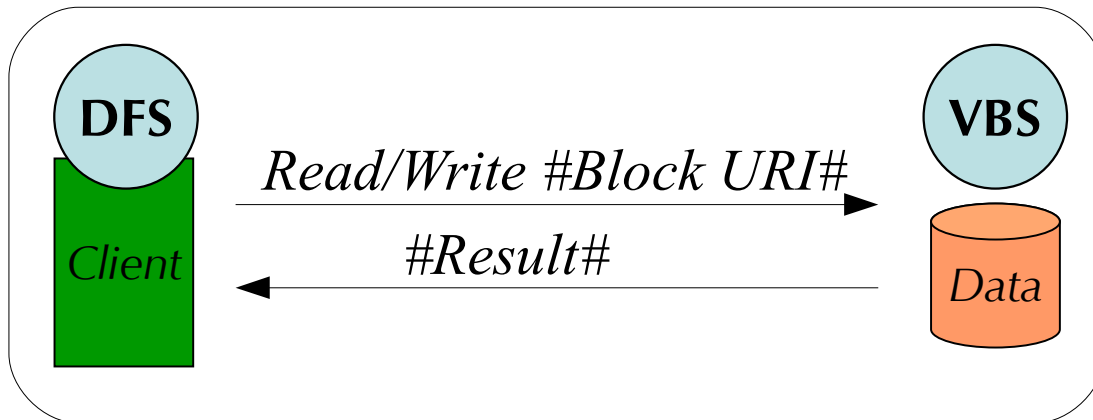
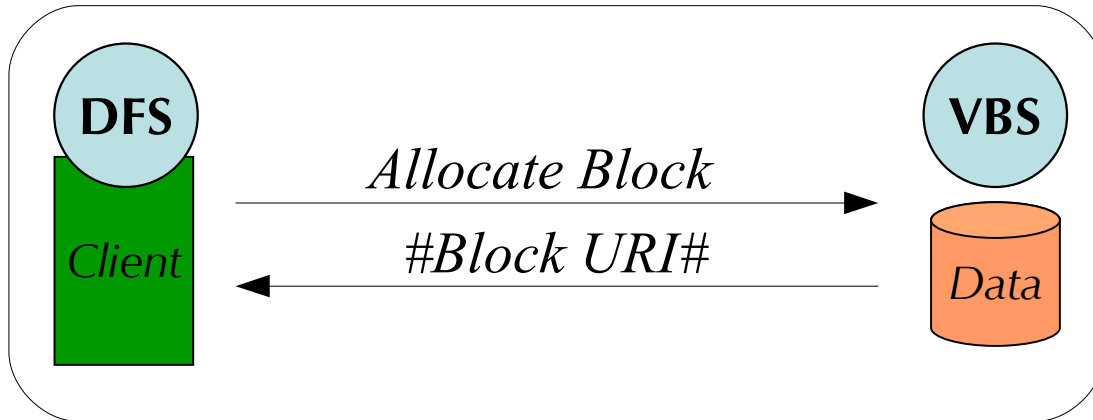
} *Storage Pointers*

# DHT Global Identity Index



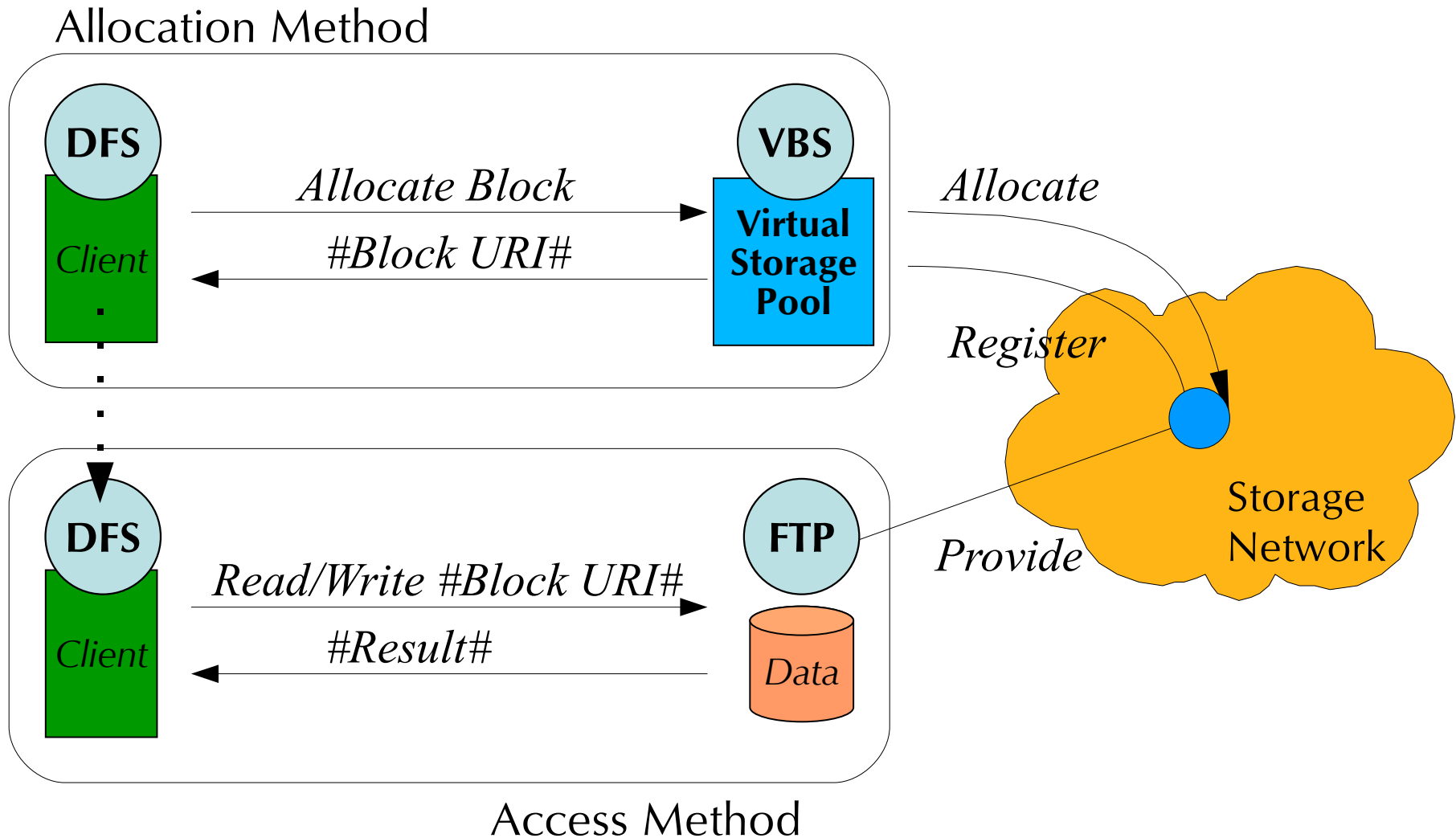
# Allocating Storage

Allocation Method

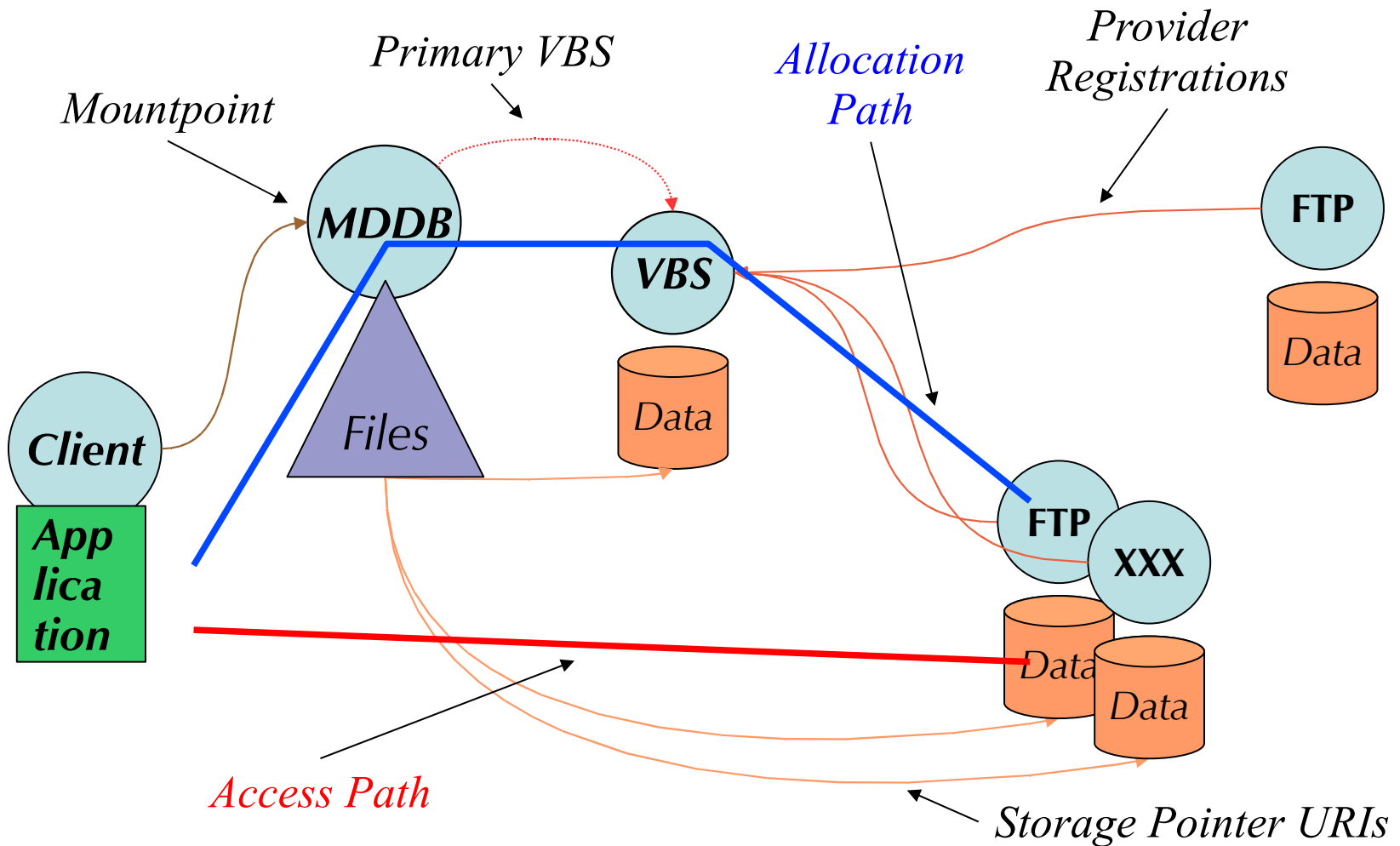


Access Method

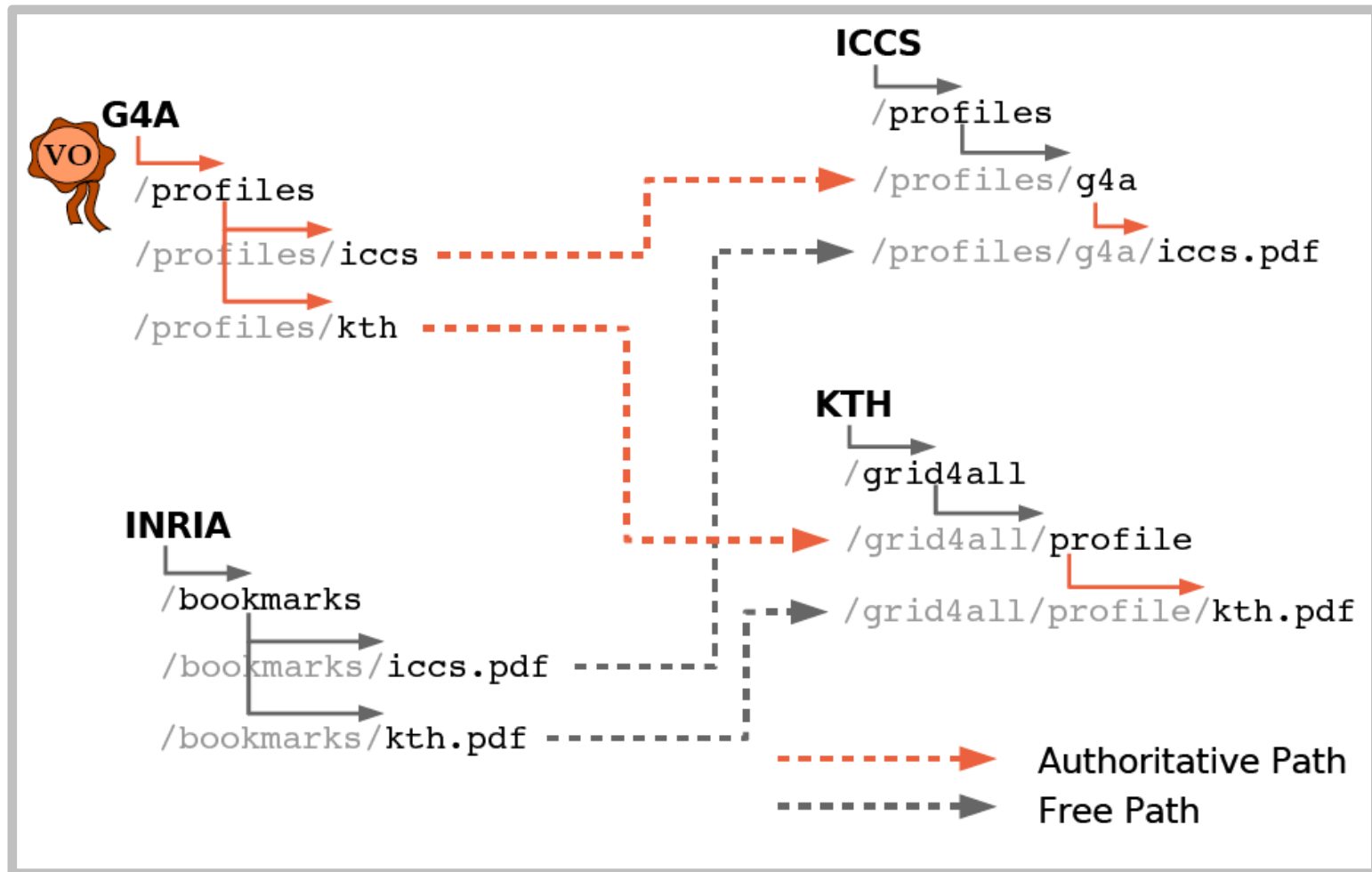
# External Storage Providers



# Network with Storage Providers



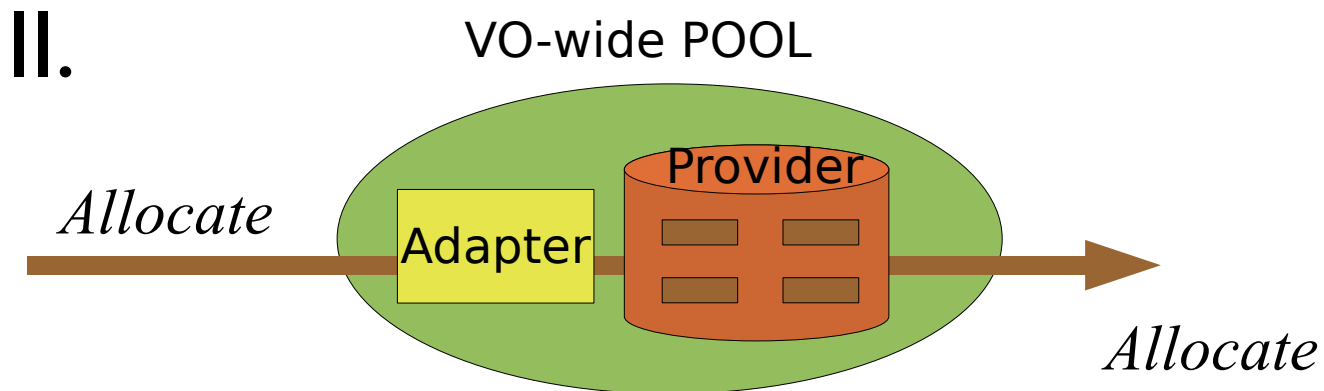
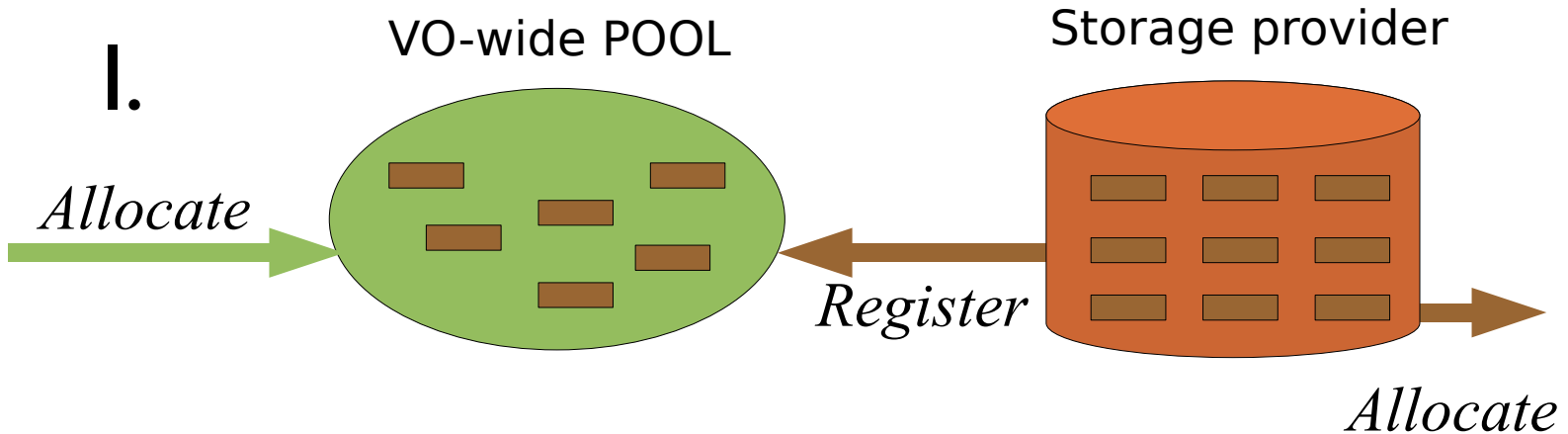
# Federation of Files



# Federation of Files

- Cross-network symbolic links bring distributed files into the same directory
- Only a directory authority can link a file in
- Only a linked file authority can change the change the file
- Clients sent their requests to authorities

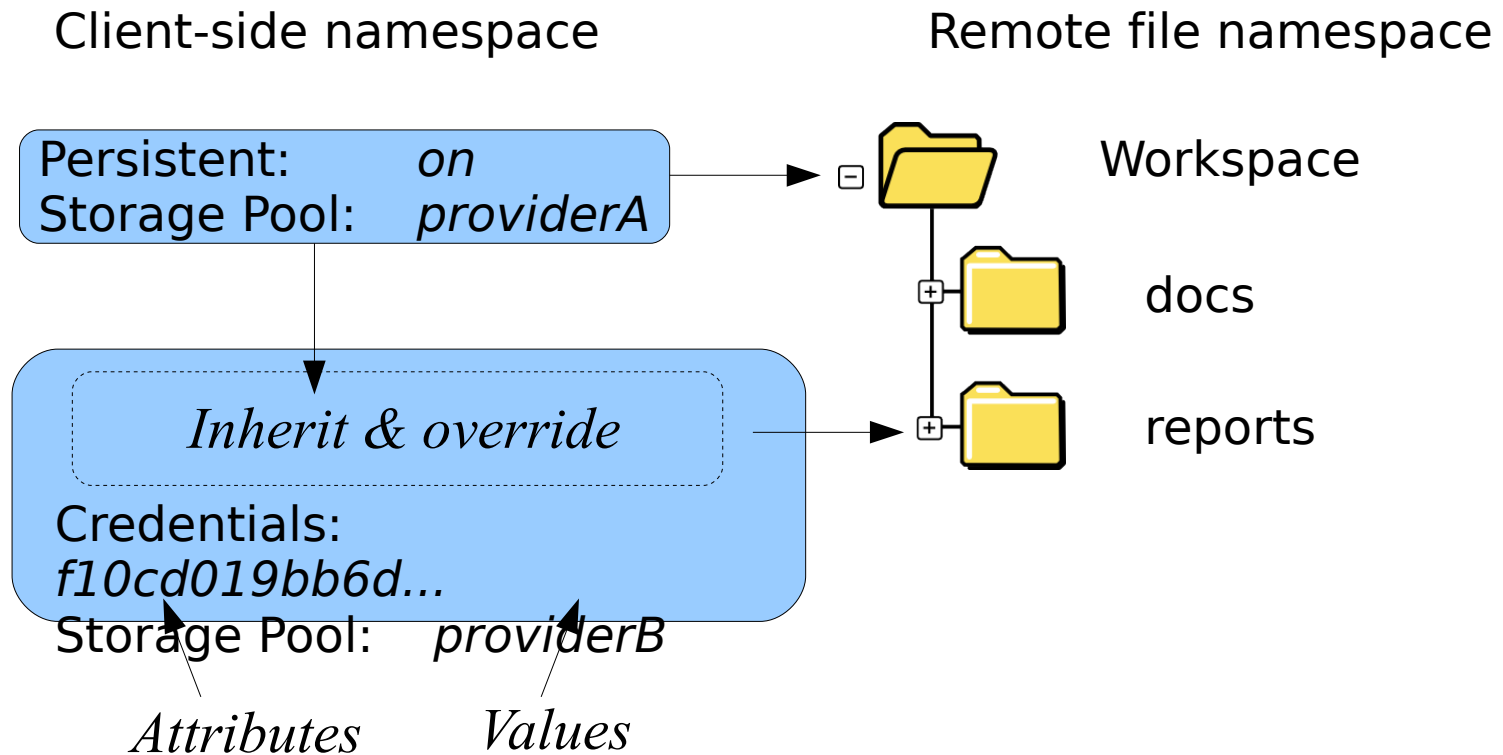
# Storage Pooling



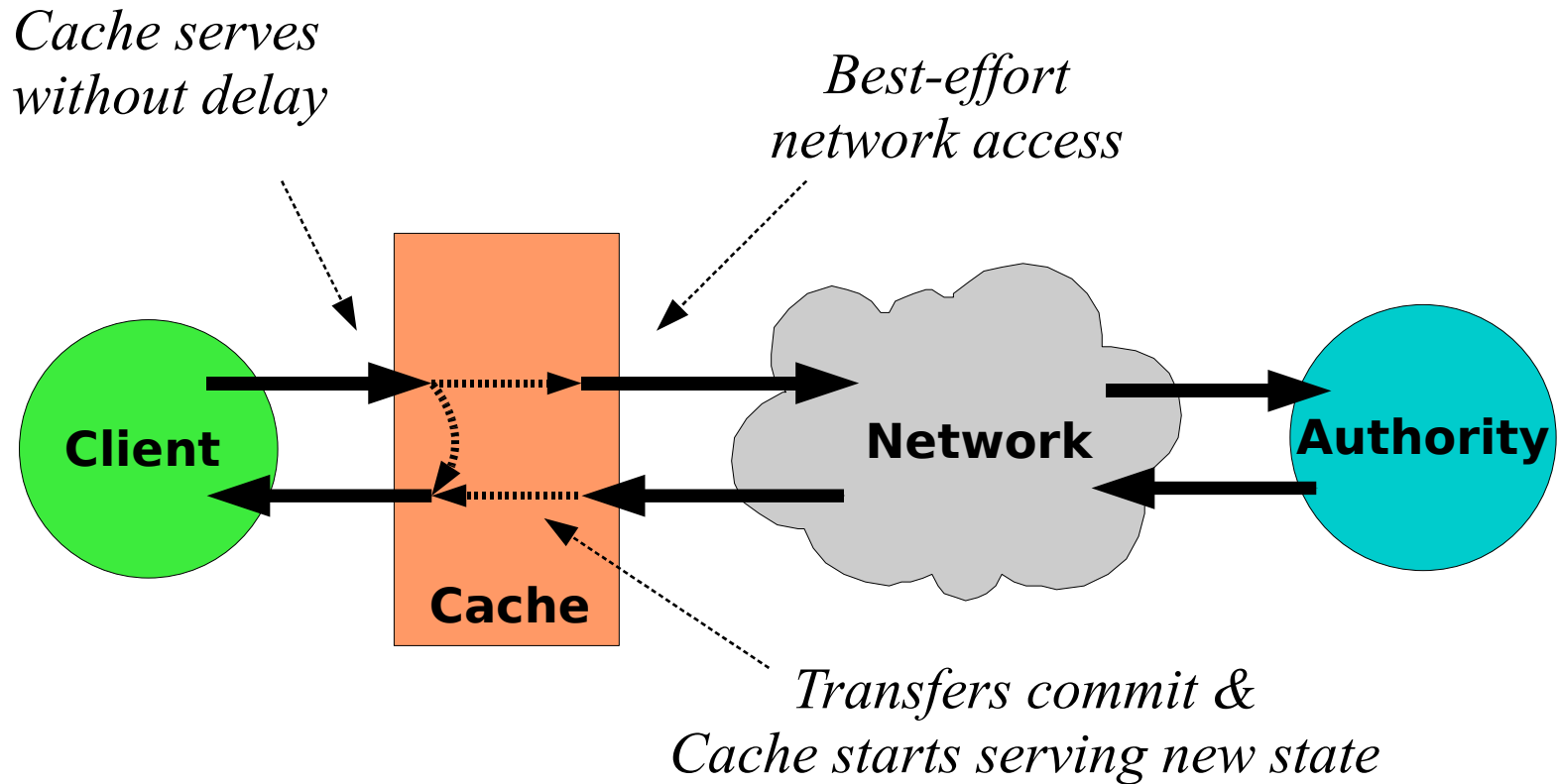
# Storage Pooling

- Files contain lists of URIs to actual storage
- Each peer has a pool of storage for allocations
- The pool may be populated with storage and managed externally
- External storage system may be adapted to implement the storage pool (ftp, p2p, etc)
- One can contribute storage to specific peers or namespaces by registering storage with their pool

# Client-side Namespace



# Disconnected Operation

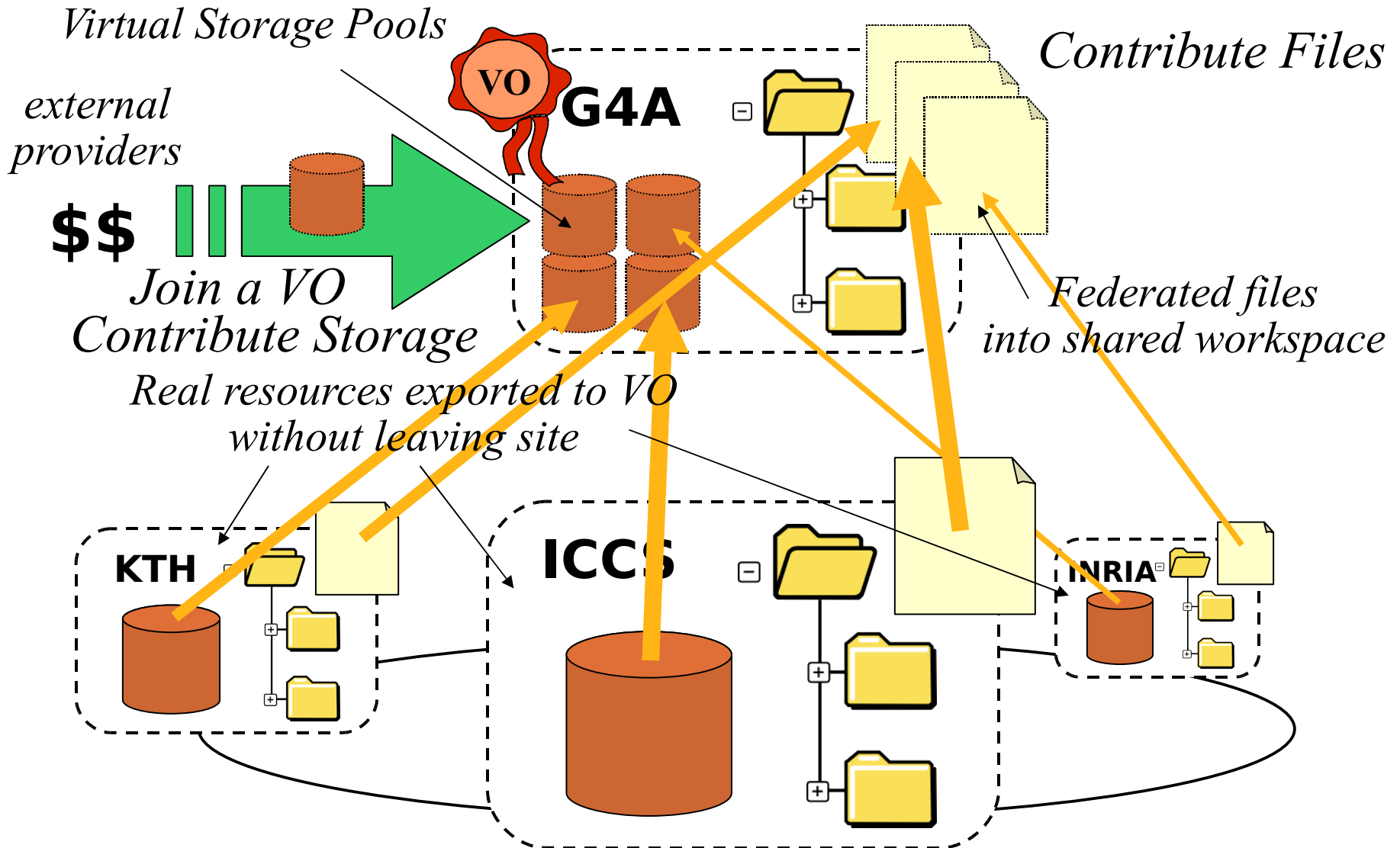




# Policy Enforcement Points

- Users store credentials as extended attributes of their files
  - children inherit and override parents
- VOFS forwards opaque credentials into requests
- PEP filters incoming at file/storage servers
  - If PEP check fails, error is returned
- VOFS regards this as another error,
  - caches negative results as disconnection

# Creating a VO Workspace



# Connecting Applications

- Subscribe for access events on remote files
- Subscribe for application messages to remote files (generic pub-sub)
- Store per-file application data
- Have VOFS transfer your credentials along its access path

# Contrast

- Not focusing on creating a VO or policies
  - but on exploring tools/mechanisms useful in the context
- Not managing inside the node
- Not focusing on commercial workflow
  - but rather on personal use as a tool
- Focus on social self-management
  - Node administrators are VO users

# Scenarios

- Sharing files across VOs
- Accessing VOs with multiple identities/roles
- Mappings between local and VO identities
  - and general indexing
- Dynamic users / workspaces / views
  - e.g. draw content from a content library to compile a temporary collection concerning a subject

End