UNICORE

JÜLICH
FORSCHUNGSZENTRUM

UFTP
High-performance data transfer for UNICORE

Jason Milad Daivandy
Federated Systems and Data division
Jülich Supercomputer Centre
Forschungszentrum Jülich GmbH

Novembre 27th 2012
EGI/EUDAT/PRACE Workshop on Data Management, Amsterdam

Outline

- UFTP
  - Features
  - Principles
  - Deployment
  - Performance
- Potential Employment in Use Cases
  - DHIRM, EPOS, Mapper, Verce, ScalaLife, Molecular... , … ?

UFTP Features

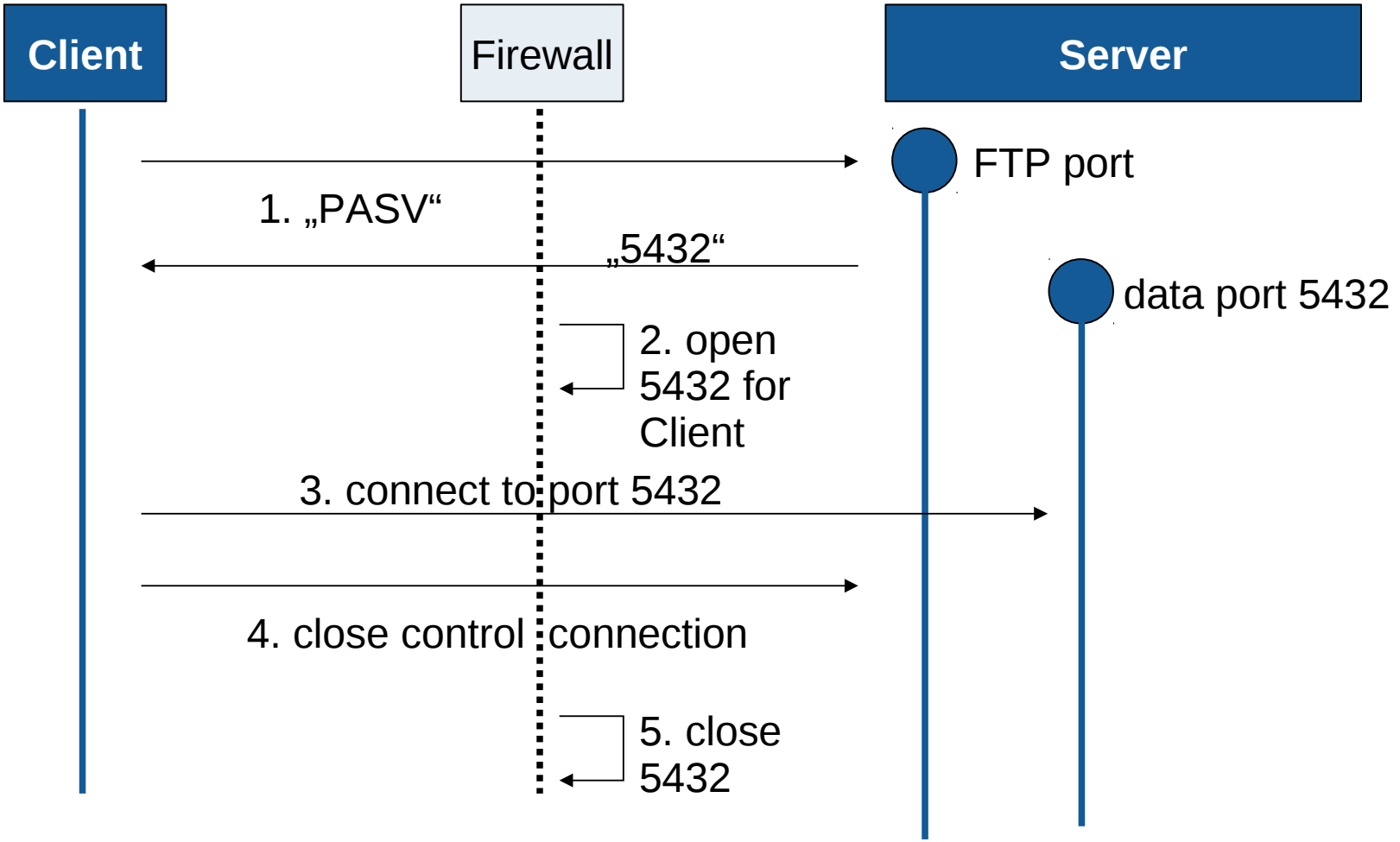
- Performant Data Transfers
- Firewall-Friendly
- Stream-based (TCP)
- Parallel Streams
- Client-Server, Server-Server
- Transfer Resumption *(in the works)*
- Quality of Service via identity-based bandwidth control
- Secure (SSL, optional Stream Encryption)
- Different Deployment Modalities
- Platform-independent (Java)

UFTP Principles I

- Dynamic Port Opening

(instead of statically setup open port ranges → security risk)

UFTP Principles I
use passive FTP to open ports



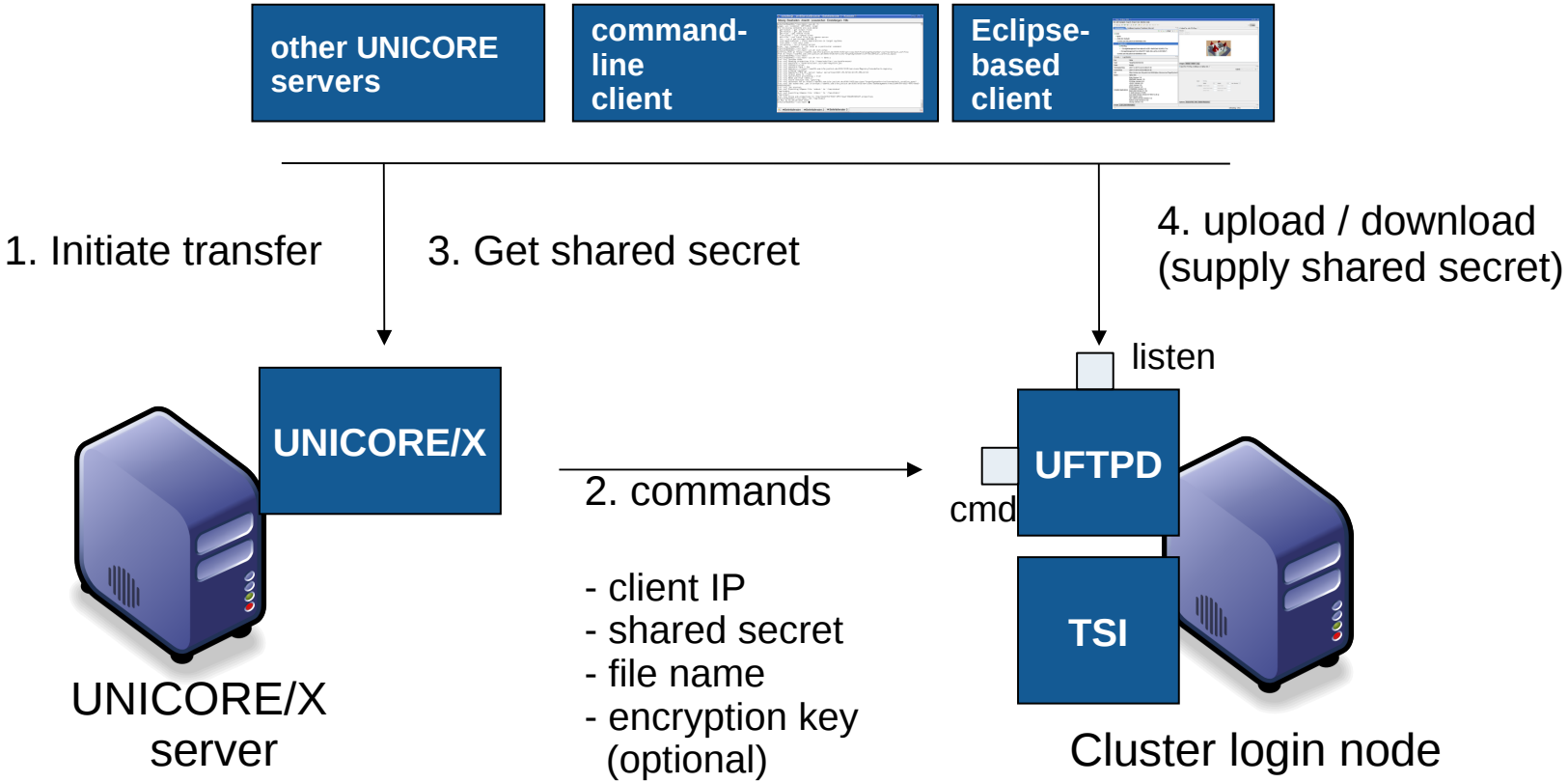
UFTP Principles II
Combining passive FTP and UNICORE

- FTP by itself is insecure:
  - Users log in using username/password
- UNICORE provides a highly secure channel from client to server which is used for additional security measures:
  - File transfers are always initiated via UNICORE
  - Client must authenticate using a „secret“ that is exchanged via UNICORE
- Requires a secure „command port“ in addition to the FTP port
  - Ideally not accessible from outside / to clients

UFTP Principles III
Security challengers and their resolution

- uftpd server runs with root privileges (because it needs to access files from all users)
  - Switch effective user/group ID before file access
- Sending commands via the Command channel allows local users to read/write files under any user ID
  - Command port not accessible outside the firewall
  - Protect it using client authenticated SSL and ACL file
- Attacker might connect to the newly opened sockets on the uftpd server
  - Client IP is checked, and a secret key is required for authentication
- Data channels might be sniffed
  - Optional symmetric encryption (64 bit key, blowfish algorithm)

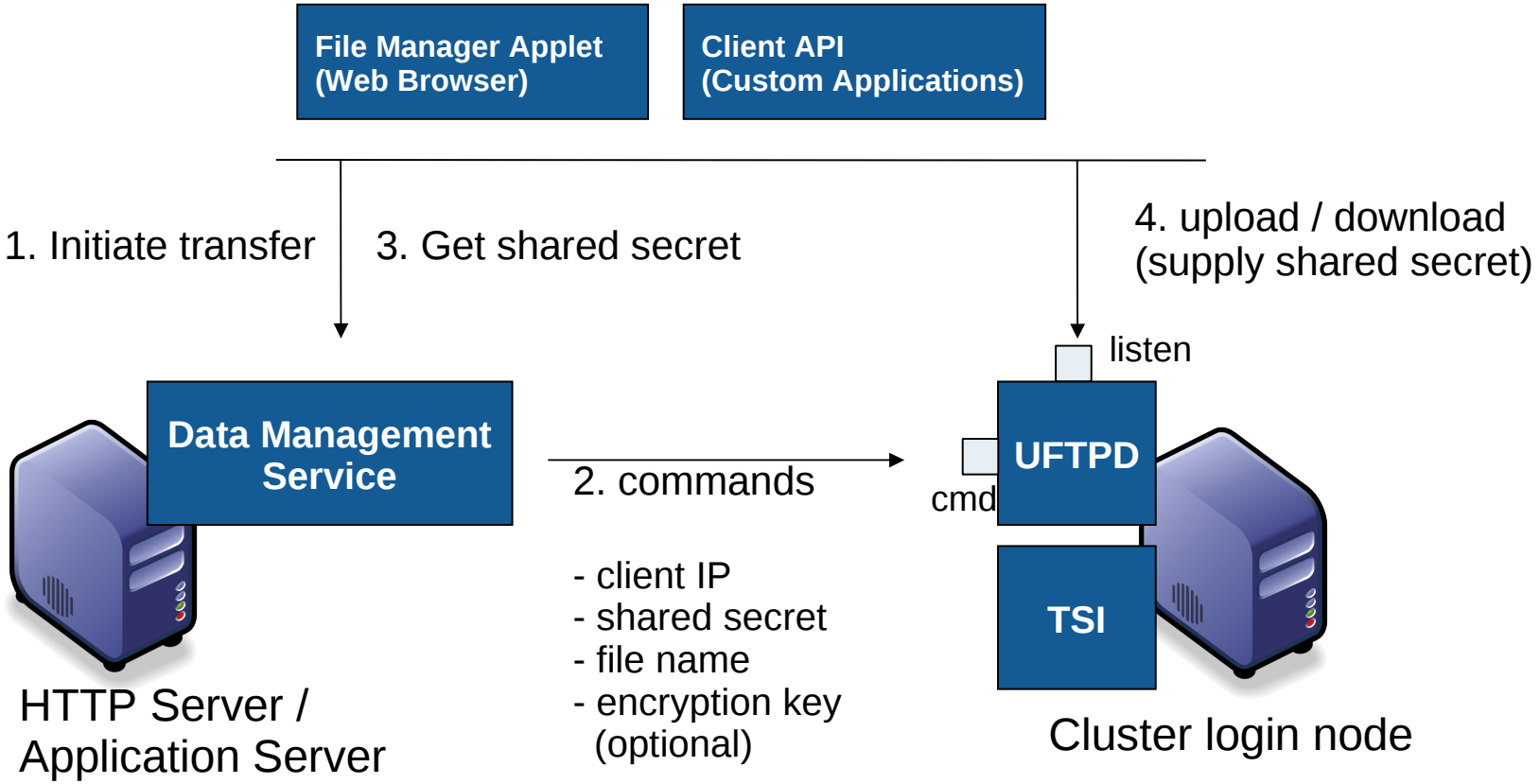
UFTP Deployment: via UNICORE



Data Management Service (in development)

- Transfer Protocol – agnostic RESTful Service
- Easily integrated into Web Applications
  - via adapter to IdM System in place
  - acts as a RESTful Security Token Service
- Delivers a File Transfer Applet to authenticated user's Web Browser
  - zero Installation File Browser / Transfer GUI Client
  - direct file transfers between user machine and storage(s)
  - manage Storage Credentials (symmetric encryption possible)
- Currently supports UFTP, WebDAV, SFTP
- Easily extensible with additional file transfer protocols (e. g. S3)

UFTP Deployment: via Data Management Service



UFTP Performance: tests in PRACE – results

- Results depend on filesystems and the network interface(s) used!
- 98% bandwidth utilisation can be achieved

- JSC – CINECA (1 Gbit/s link, 1GB file size)
  - JSC tmp → CINECA tmp : 70 MB/s
  - JUGENE tmp → CINECA dev/null: 128 MB/s

- JSC – SARA (10 Gbit/s link, 10 GB file size)
  - SARA Home → JSC Home: 72 MB/s
  - SARA Scratch → JSC tmp: 126 MB/s

- For comparison (internal 10 Gbit/s link):
  JSC JUROPA Home → JUGENE Home : 258 MB/s

Use Cases

DRIHM (Bert Jagers)

- Req's Use Case 1:
  - Stage In: 250 – 350 GB from remote [x]
  - Stage Out: 50 – 100 GB to local / remote [x]
  - Storage as a Service: could be setup with UFTP [x]

Use Cases

DRIHM (Bert Jagers)

- Req's Use Case 2:
  - Stage In: 100 – 200 MB form remote [x]
  - Stage Out: 10 – 100 MB to local / remote [x]
  - Storage as a Service: could be setup with UFTP [x]

Use Cases

EPOS (Luca Trani)

- Req's:
  - User's Perspective: Data Staging [x]
  - Potential Reuse of Stage Out Data [x]

Use Cases

Mapper (Ilya Saverchenko)

- Req's:
  - Data Stage In [x]
  - Data Stage Out [x]
  - Encrypted File Transfer [x]

Use Cases

Verce (Iraklis Klampanos)

- Req's:
  - Data Stage In from local: GBs [x]
  - Data Stage Out to remote: GBs – TBs [x]
  - Streams [x]

Use Cases

Molecular Knowledge Engine ... (Antonio Laganà)

- Req's:
  - Data Stage In from remote: rather small [x]
  - Data Stage Out to remote: GBs [x]

Use Cases

ScalaLife (Rossen Apostolov)

- Req's Use Case 1:
  - Temporary Storage [x]