Bootstrapping a (New?) LHC Data Transfer Ecosystem

Brian Paul Bockelman, Andy Hanushevsky, Oliver Keeble, Mario Lassnig, Paul Millar, Derek Weitzel, Wei Yang
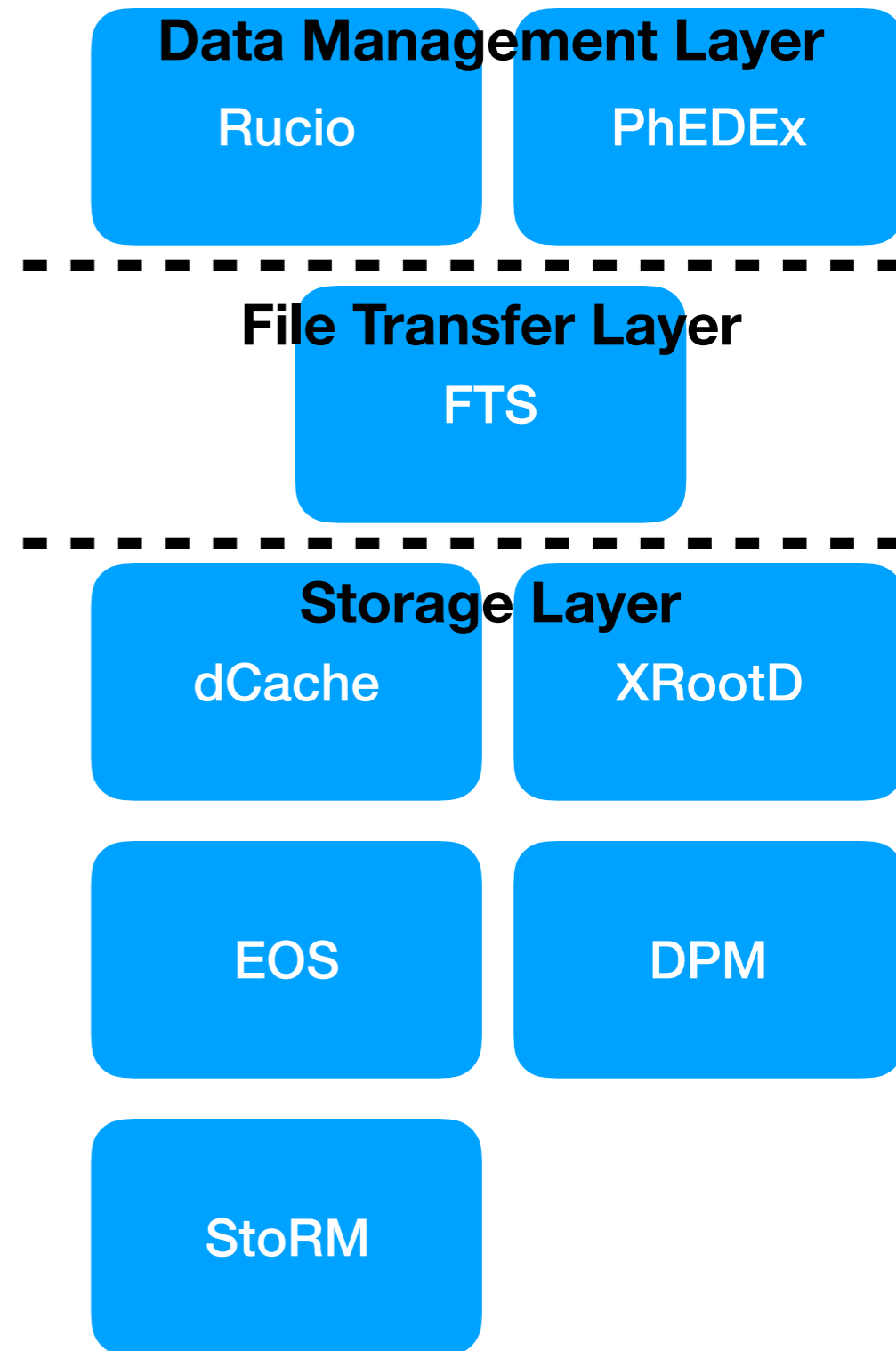
Why am I here?

- The announcement in mid-2017 that Globus Toolkit support would end set off a flurry of activity.

    - Some of it was toward shorter-term collaborations around community support of this software. See https://gridcf.org

- This reinvigorated existing work around replacing various Globus Toolkit components; the most pressing are:

    - **Grid Security Infrastructure (GSI)**: An authentication and authorization infrastructure based around concepts of identity and X509 proxies.

    - **GridFTP**: A FTP-like transfer protocol that build on top of GSI, supports third-party-transfers, and multi-TCP-stream transfers.

- Luckily, there's a huge amount of prior effort to draw on, some dating back several years.

    - **Hence it's not really "new."** Thanks to the work of many people, what's shown here is just some clever re-arrangements!
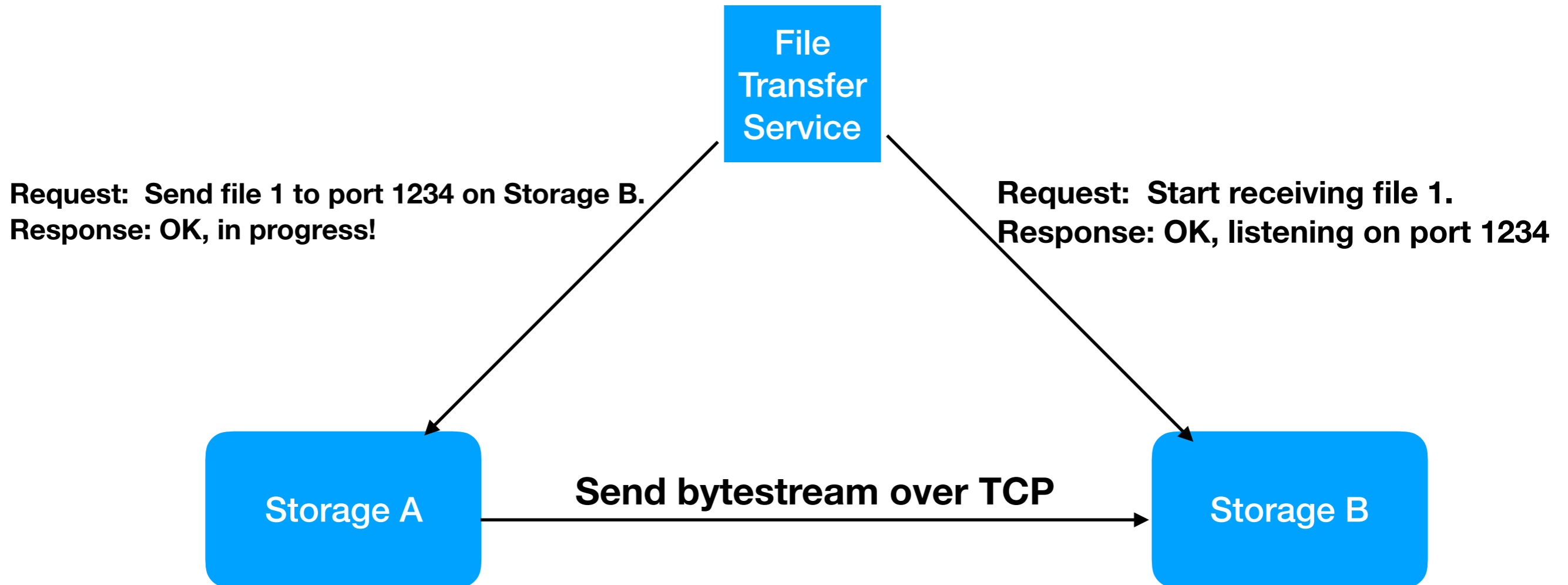
WLCG Transfer Ecosystem Demonstrator

- There's a need to organize the entire vertical stack to have a cohesive solution approach.

- We benefit little if multiple storage elements take mutually-incompatible approaches.

  - Same applies for moving across the data management / file transfer / storage layers.

- Put together a Google group to coordinate this activity and start to scale:
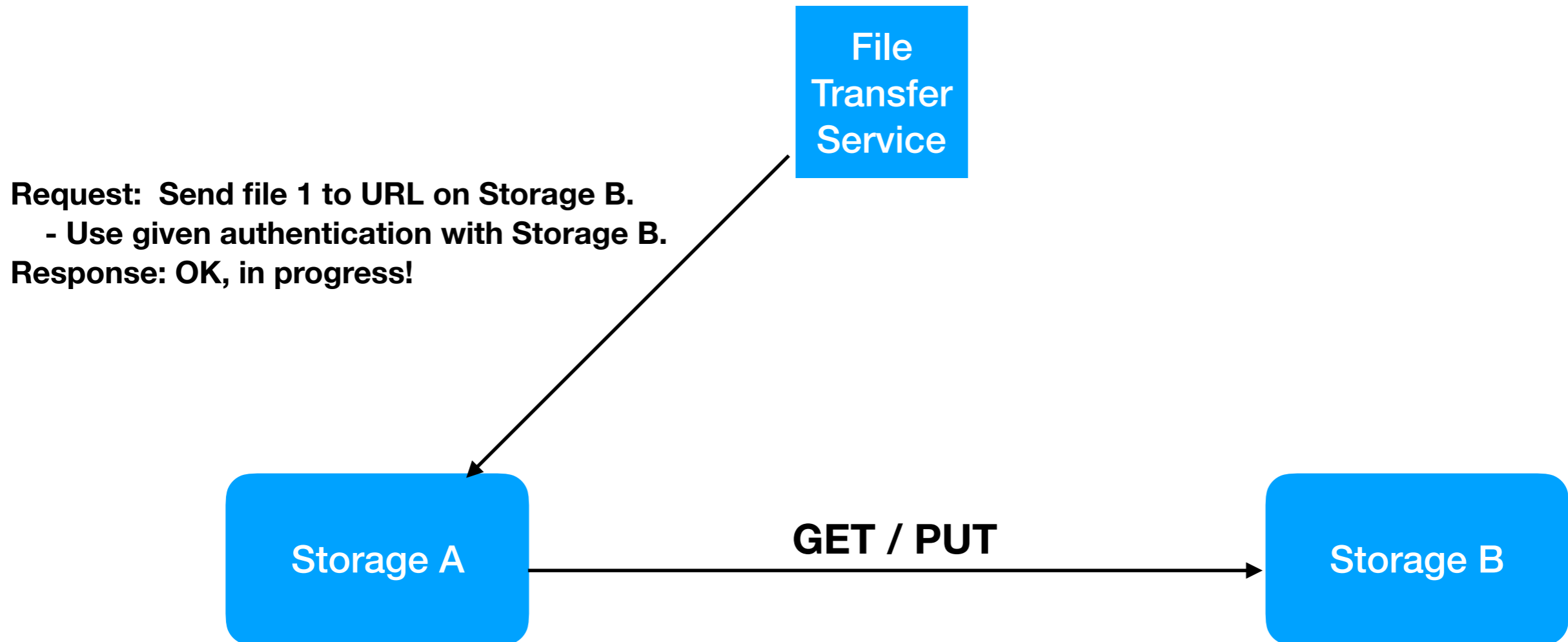
  - Feel free to join!

  - https://groups.google.com/forum/#!forum/wlcg-http-transfer

Data Management Layer

Rucio

PhEDEx

File Transfer Layer

FTS

Storage Layer

dCache

XRootD

EOS

DPM

StoRM

3

Transfers Under GridFTP - Where we are today!

File Transfer Service

Request: Send file 1 to port 1234 on Storage B.
Response: OK, in progress!

Request: Start receiving file 1.
Response: OK, listening on port 1234

Storage A — Send bytestream over TCP → **Storage B**

- **FTS must be authorized to talk to both endpoints.**
- **Endpoints support the same protocol (GridFTP).**
- **Queueing (in implementation) is in FTS layer.**

Alternate TPC Model - Where we might go!

File Transfer Service

Request: Send file 1 to URL on Storage B.
**   - Use given authentication with Storage B.**
Response: OK, in progress!

Storage A → **GET / PUT** → **Storage B**

- **FTS only communicates with the active storage (A).**
  - **FTS provides URL for B and authz token.**
- **Transfer from A->B may occur on any mutual protocol.**
- **FTS relies on storage A for heavy lifting.**

HTTPS / WebDAV

- WebDAV is a set of HTTP extensions that provide a more standardized, file-like API with minimal HTTP changes.

  - Example: "`MKCOL`" (make collection) is mostly equivalent to a POSIX `mkdir()`.

- Another WebDAV extension is `COPY`, which instructs the WebDAV server to copy to/from a given URL.

  - Precisely what is needed for the alternate TPC model!

  - The URL is given in the `Source` header; not necessarily HTTPS!
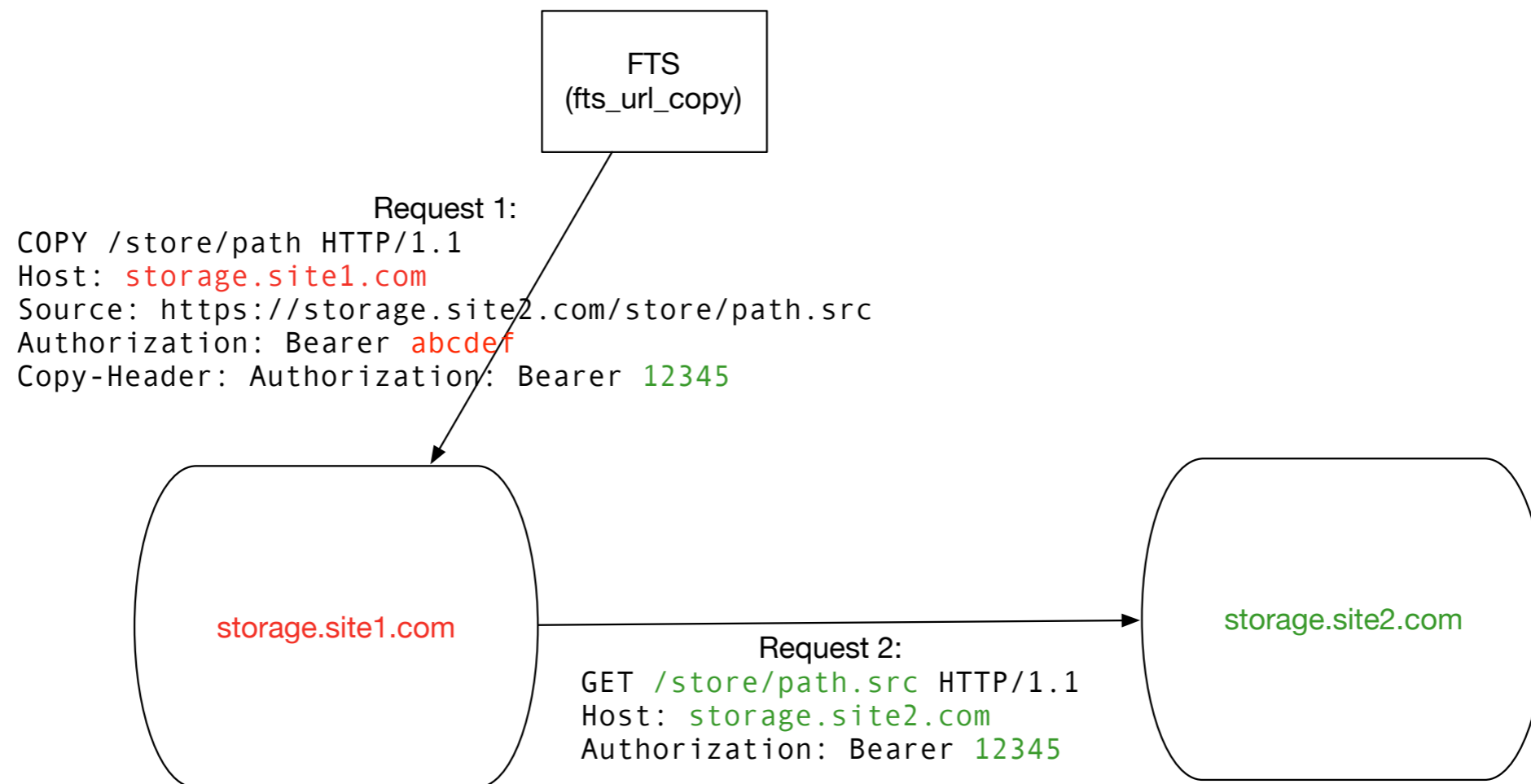
```
COPY /store/path HTTP/1.1
Host: storage.site1.com
Source: https://storage.site2.com/store/path.src
```

HTTPS / WebDAV - Authorization Step

- It's clear FTS can use its favorite existing mechanism when communicating with the "active" SE (Storage A).

  - How does it transfer a credential to the active SE for use with Storage B?

- In X509-land, we have the concept of delegating a credential for this movement.

  - Unfortunately, the delegation procedure is only "standardized" at the transport layer (TCP).

  - The WLCG community has a somewhat ad-hoc* standard for this based on SOAP, as defined by GridSite.

- We advocate a token-based method for authorizing the transfer instead.

* https://egee-jra1-data.web.cern.ch/egee-jra1-data/GridSiteDelegation/HEAD/doc/glite-security-delegation-interface/DelegationInterface.html

Reminder:
Here's our picture

FTS
(fts_url_copy)

Request 1:
```
COPY /store/path HTTP/1.1
Host: storage.site1.com
Source: https://storage.site2.com/store/path.src
Authorization: Bearer abcdef
Copy-Header: Authorization: Bearer 12345
```

storage.site1.com

storage.site2.com

Request 2:
```
GET /store/path.src HTTP/1.1
Host: storage.site2.com
Authorization: Bearer 12345
```

Here, I illustrate the case where the actual copy also goes over HTTP

HTTP Request

HTTP verb

Resource at active SE (destination)

- Example request from FTS to "active" SE:
  COPY /store/path HTTP/1.1
  Host: storage.site1.com
  Source: https://storage.site2.com/store/path.src ← Source URL
  Authorization: Bearer abcdef ← Token for active SE
  Copy-Header: Authorization: Bearer 12345 ← Token for inactive SE

  ↑ Indication to copy header to GET request

-

"Real" Request

Here's an example request from yesterday:

```
COPY /user/uscms01/pnfs/..truncated../LoadTestDownload/LoadTest07_UCSD_B0_nI3SsXhd0iT5RF6W_286 HTTP/1.1
User-Agent: fts_url_copy/3.7.7 gfal2/2.15.0 neon/0.0.29
TE: trailers
Host: red-gridftp12.unl.edu:1094
Source: https://gftp-1.t2.ucsd.edu:1094/cms/..truncated../LoadTest07_UCSD_B0
```
X-Number-Of-Streams: 3
Secure-Redirection: 1
```
Authorization: Bearer eyJhbGciOi..truncated..NYU5gx6yrZhKpdCt2SedVocIhZsuqKNUNZcRhXj6tBjxozA
ClientInfo: job-id=dc417124-30d7-11e8-bd67-5254000b9cba;file-id=1080;retry=0
TransferHeaderAuthorization: Bearer eyJhbGci..truncated..5_-Z7XQw
```
RequireChecksumVerification: false

Get Your Tokens!

- In the latest FTS release, at the start of a transfer, FTS will:

  - Generate a SciToken* if a token issuer is specified (see next presentation!).

  - If that fails, use the X509 proxy to generate a macaroon (see Paul's presentation on macaroons).

  - If that fails, fall back to gridsite-based delegation.

- The multiple fallbacks are designed to provide a smooth transition off X509!

*currently authenticates with token server using X509 proxy, but hopefully not in future

Working up the Stack

- We have an initial prototype functioning as XRootD plugins.

  - Stable enough to put at production servers at three different sites.

  - Hopefully this can benefit EOS and DPM as well!

- dCache has test code for SciTokens and macaroons are in production.

- GFAL2, DAVIX, and FTS have patches in release (or testing) supporting the end-to-end.

- PhEDEx changes available as patch and Rucio changes are in a testing branch.

Working the vertical: patches across about a dozen software packages.

Timeline to Success?

- Given the bleeding edge version of software (and appropriate configurations), we have shown HTTPS can technically replace GridFTP use for all storage used in the WLCG.

- Work for the remainder of 2018:

  - Demonstrate the full compatibility matrix of SEs in production.

  - Design and perform scale tests.

- More importantly, what's next? Need to work with a few experiments (LHC or otherwise) to determine schedule and future intent.

  - Should we set a bold goal like "**All WLCG sites supporting CMS must support non-GridFTP third party copy in 2019**"?

  - Otherwise, there's danger this effort will fizzle out!