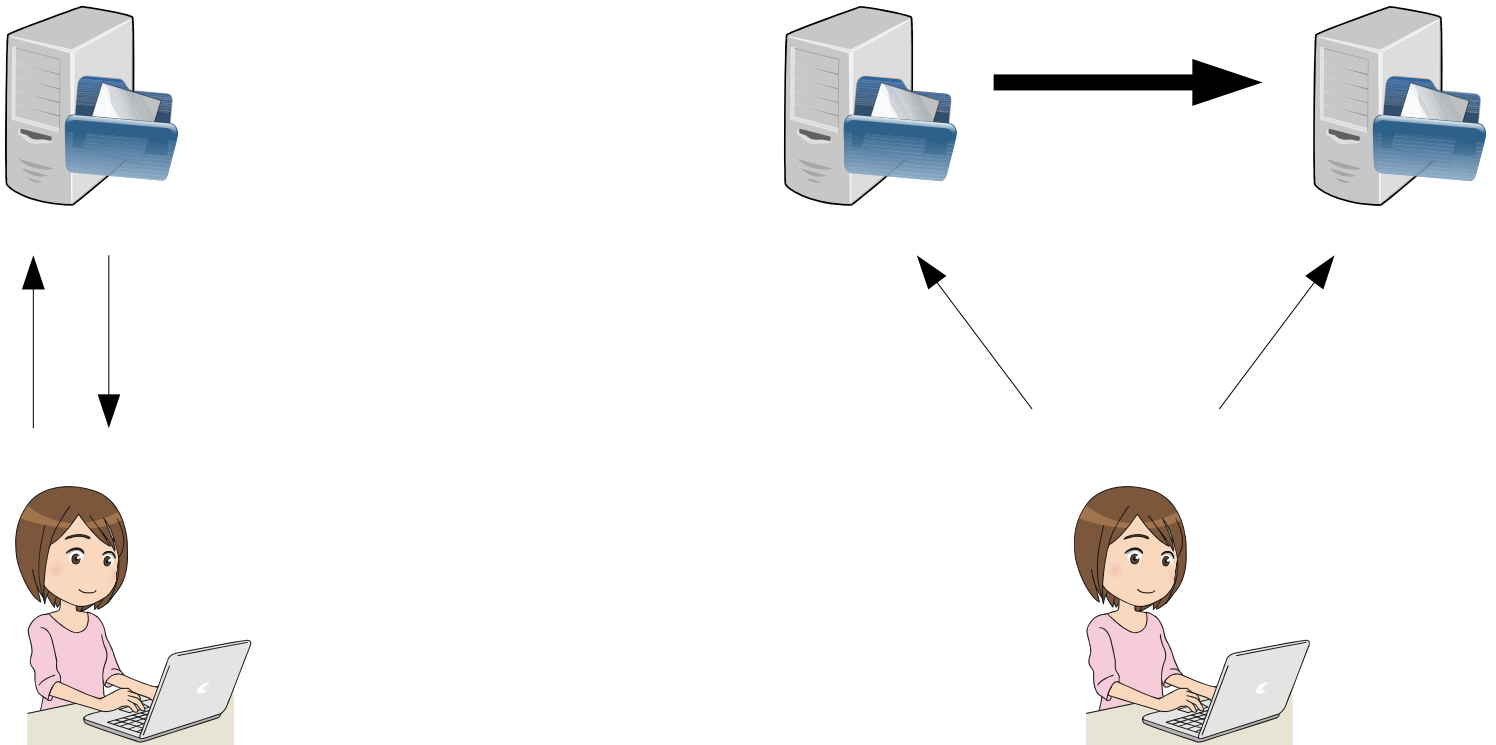
Third-Party Copy with HTTP

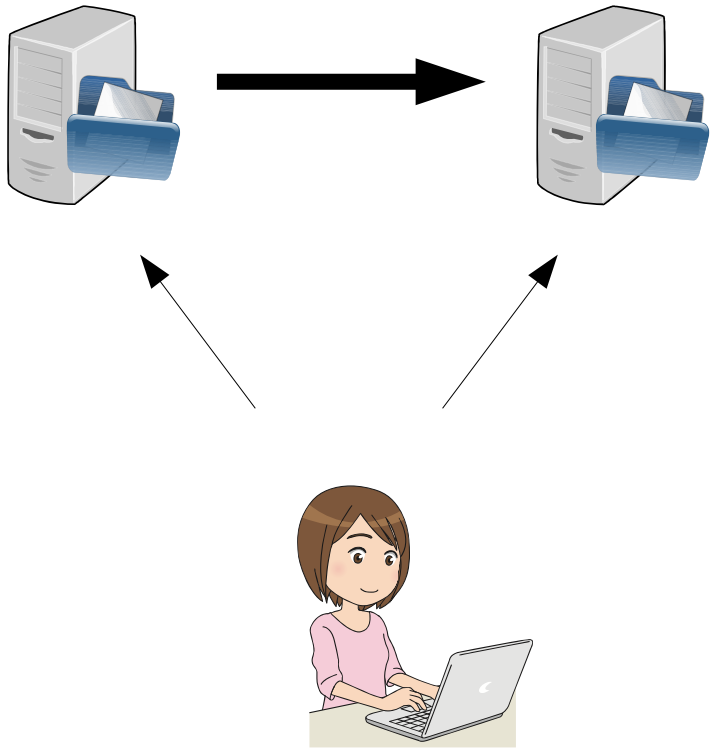
Paul Millar
(on behalf of the dCache team)

dCache Workshop 2019 at Madrid, Spain; 2019-05-21
https://indico.desy.de/indico/event/22170

Third-party copy: what is it



Third-party copy: what is it



- Client connects to source and destination servers
- Tells destination to be passive and accept data; client is told where to connect
- Tells source to be active and send data, connecting to destination
- Source connects to destination and sends data

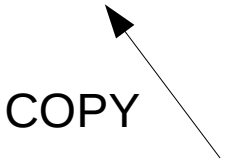
Why are we looking at alternatives?

- Many storage implementations (but NOT dCache) use the **Globus Toolkit** (GT) to support GridFTP.
- Support from Globus for the GT **ended January 2018**.
- GT is renamed **Grid Community Toolkit** (GCT) and maintained by Grid Community Forum (GridCF)
- The concern about whether GridCF's "voluntary" effort is **sustainable**, particularly if there is a large change (e.g., OpenSSL API change)
- Alternatives?

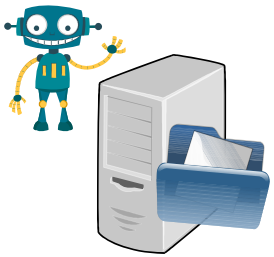
The problem with HTTP TPC – we need an extension

- How about HTTP?
- FTP works because we have separate control and data channels
- HTTP doesn't have this: the data comes over the same connection used to make requests

The HTTP-TPC extension

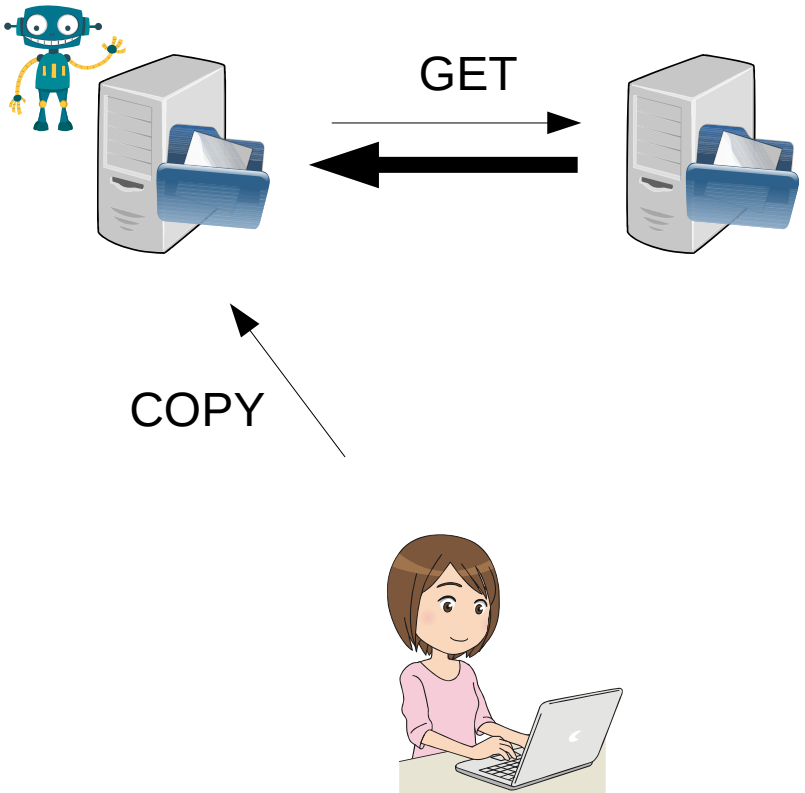


The HTTP-TPC extension

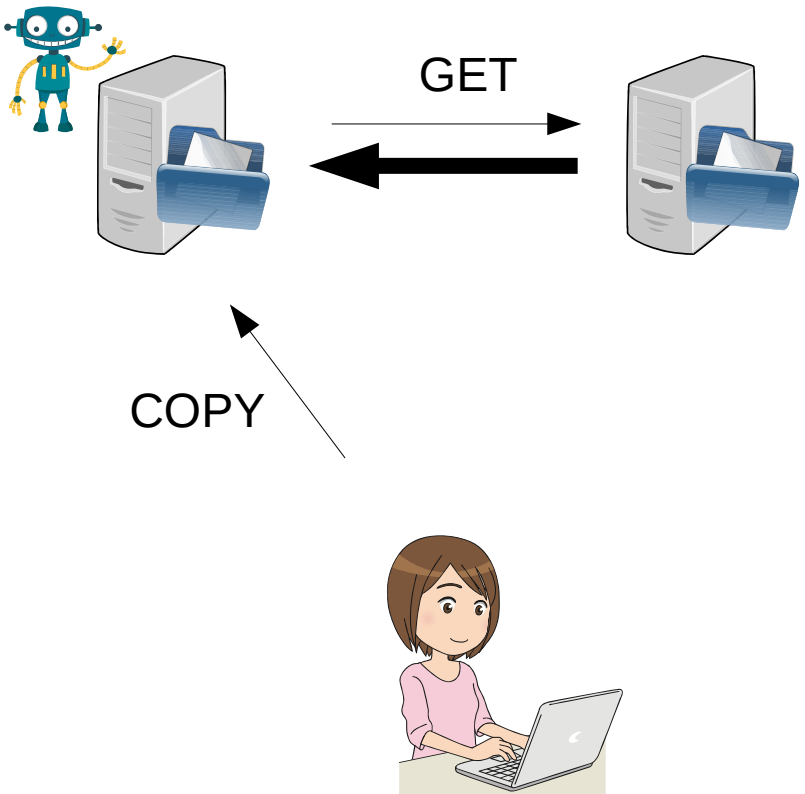


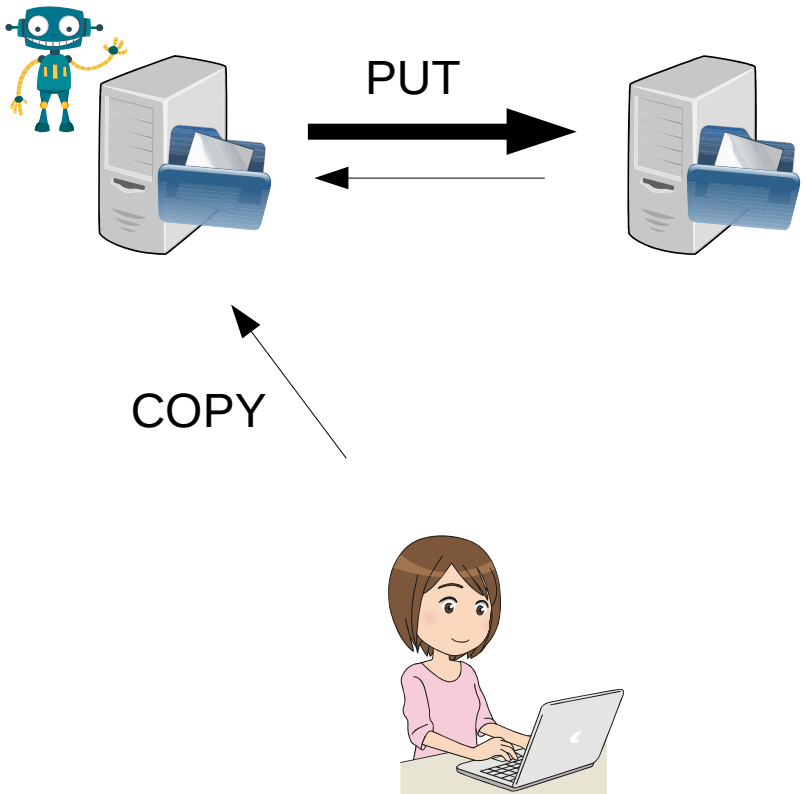
COPY

The HTTP-TPC extension



The HTTP-TPC extension





How it works in dCache

- Client makes COPY request to WebDAV door.
- WebDAV door requests RemoteTransferService organise the transfer; periodically checking for progress.
- RemoteTransferService does pool selection and requests pool does transfer.
- Mover makes HTTP GET or PUT request, to initiate transfer.
  - If PUT, makes subsequent HEAD to fetch file size and checksums.
  - In either case, ensure checksums match, if requested.

How to configure it in dCache

- WebDAV: mostly works out-of-the-box.

  {dcache, webdav}.enable.macaroons = true (default)

  webdav.authn.require-client-cert = false (default)

- Pool needs to trust remote server

  Install IGTF trust store on the pool nodes.

- DOMA-TPC documentation:

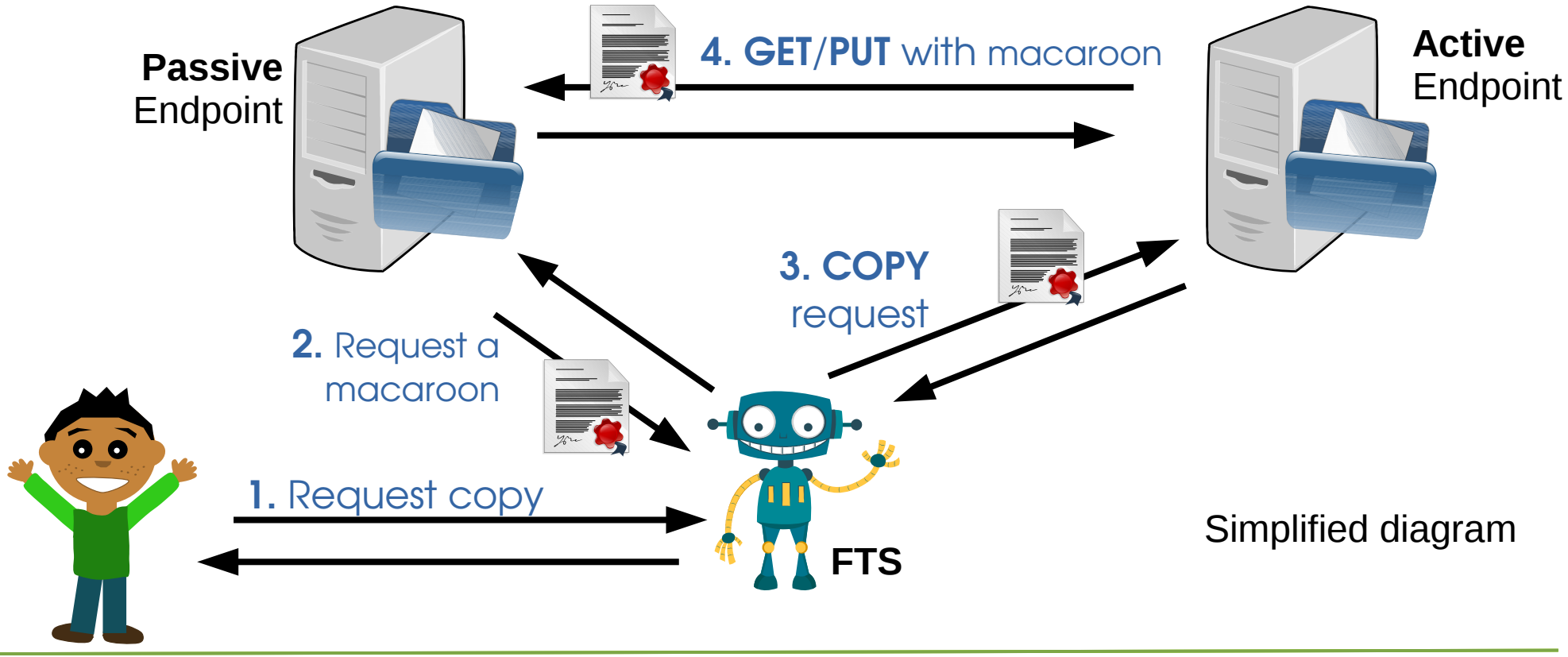
  https://twiki.cern.ch/twiki/bin/view/LCG/DCacheConfig

How delegation works: with x.509

- How it works:
  - Client makes a **COPY request**, indicating that an X.509 credential should be used for the data-bearing request.
  - webdav door **checks** if that user has a delegated credential.
  - If not, the webdav door **redirects** the client to itself, indicating the endpoint of the GridSite delegation server in **response headers**.

    Currently, this is on port 8445 of the srm endpoint.
  - Client **delegates** a credential,
  - Client repeats the **COPY request**.
- This works with FTS,
  - By default, this is a fall-back if passive endpoint does not support generating a bearer token.
  - It is not considered the final solution in DOMA-TPC group.

How delegation works: with macaroons



How to diagnose problems

- COPY requests are logged in **access log file**.
- The **`failed-copy.sh`** script (from http-tpc-utils) lists only failed HTTP-TPC transfers, providing an easier-to-read summary of what went wrong.
- The **`smoke-test.sh`** script (also from http-tpc-utils) performs a series of tests to verify that everything is working.
  - Currently requires dteam VO.
  - HTTP-TPC test transfers hard-coded to use prometheus.

`https://github.com/paulmillar/http-tpc-utils`

Working with storage outside grid

- The passive endpoint is just a regular HTTP server.

  This means it can be anything that supports HTTP, which is basically anything.

- Caveat: requests (typically) must be authorised

  We have a couple options:

  - Put something in the standard Authorization header
  - Use a special, pre-authorized URL.

- S3 servers (typically) offer pre-authz URLs.

Thanks for listening!