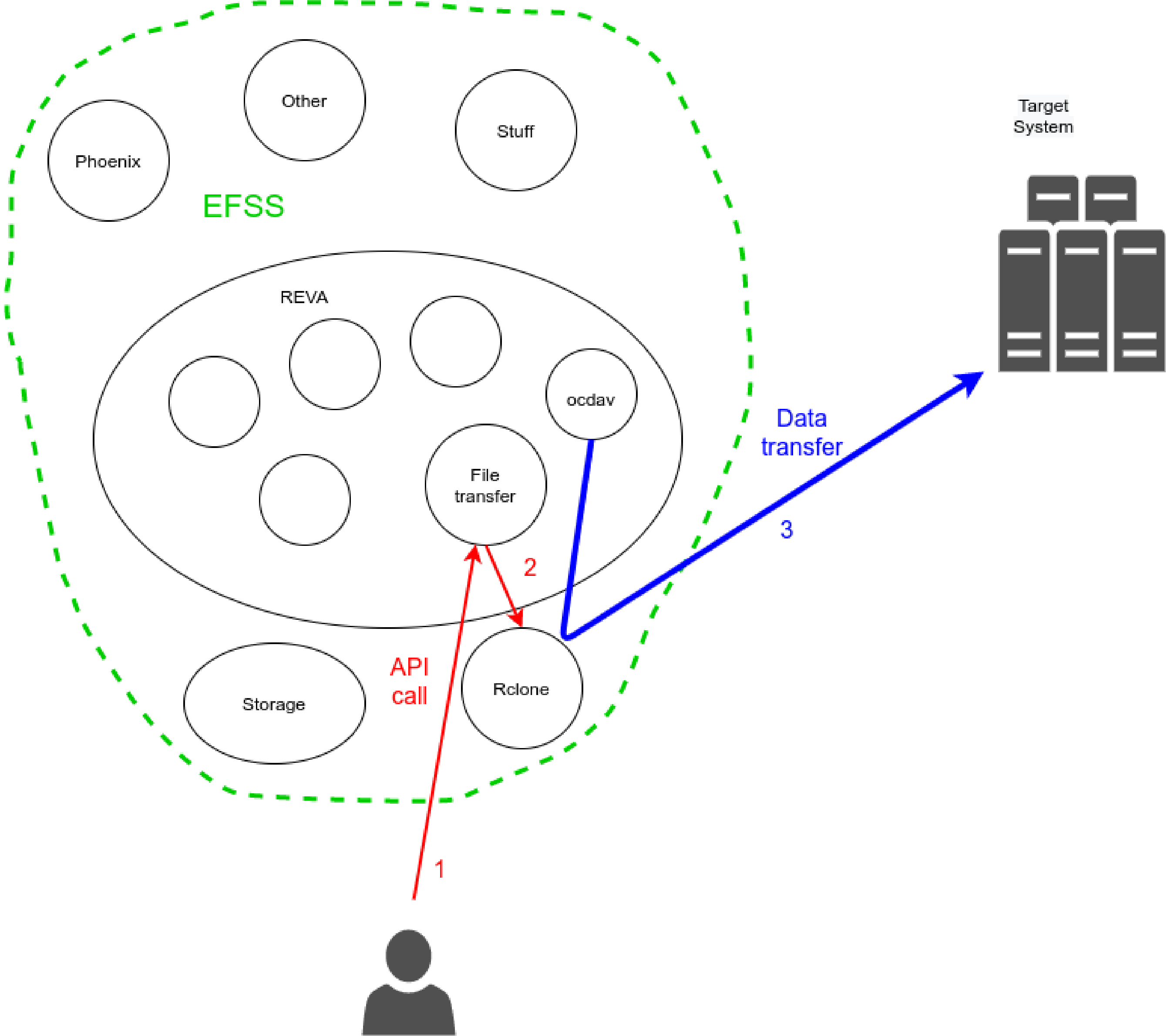
Software Design Rclone Service

Remarks

- Still work in progress
- This is just to get us going
- Lay the ground work for later development
  - Start with rclone, later on FileSender, FTS, Rucio


Other
Stuff
Phoenix
EFSS
Target System
REVA
ocdav
Data transfer
File transfer
3
2
API call
Storage
Rclone
1

Rclone

- Client that can copy data to and from just about any cloud storage known to man amongst many other things
- Can run as a daemon “rclone rcd” accepting http POST requests in JSON
- Needs a config file to define “remotes”
- Can’t do third-party copies (yet), i.e. data traverse through the rclone process

Context

- Data locality
- Service can be part of REVA, but also be standalone→ multivendor
- Start with rclone.
  - Nothing on the critical path
  - Pathfinder for the FileSender/FTS/Rucio work
- REVA is about data/application sharing.
  - This application should run at every EFSS system so it won't be shared

Goals

- HTTP service that can perform “rclone copy” commands
  - Design REST API
    - Make sure it can be re-used for FileSender/FTS/Rucio
  - Front-end to accept http requests
  - Backend to do the data transfer heavy lifting

Non-Goals

- Implementing any of the other rclone commands
- Service accessible over WAN
- Integration with web I/F of EFSS solutions
- High Availability (retry mechanisms?)
- Data transfer management
- Scalability (need to look at this later on, though)

Existing Solutions

- Do your own scripting to copy 200TB of data to a EFSS (yuck)

Proposed Solution

- EFSS systems in general not very well suited for handling the larger data sets
- HTTP-based service to do this dirty work

Alternative Solutions

- Nope

Testing, Monitoring Alerting

- TODO

Cross-Team Impact

- None
- This is very much a stand-alone thingie which has very little impact on the rest of the CS3MESH project

Open Questions

- Security
- Scalability
  - For filesender/FTS/Rucio this may be less of an issue but since data passes through rclone this should be taken into account for this case
- High-Availability
- Multi-user issues

Nitty Gritty Details

Basic workflow

1) Request comes in
2) Process is forked to handle new incoming requests
3) Parent process validates the input
    1) Security
    2) Validity
4) Rclone daemon is fired up “rclone rcd”
5) Issue “POST config” command to it to create config file in memory to create the necessary “remotes”
6) Issue “POST copy” command to do the transfer
7) After the transfers have ended, the stuff that rclone returns (exit code, errors, whatever) is returned to the client issuing the request in step 1
8) Cleanup: Delete rclone config, rclone daemon is killed, process is killed

Timeline

- Coming soon in theatres near you
- Start the development work
- Determine what other partners can/want to contribute
- Look ahead to FileSender/FTS/Rucio