liber::rsync

Roles:

* Requester: A node that wishes to update its local copy of a file.
* Authority: A node that is in possession of copy of a file that is considered authoritative based on certain criteria. For the sake of flexibility, liber::rsync is not responsible for determining whether a file is authoritative. A node becomes an authority based on the direction of initiated transaction. For example, if a node A sends a PULL request on a file from node B, then node B is definitively the authority for the file. However, liber::rsync does provide opportunities for nodes to reject the Authoritative role if application-defined criteria is not meet. For example, if node A initiates a PULL request to node B, B’s PULL-request hook could check file status to determine whether its local file actually meets authority criteria. If the file does not meet the criteria, B replies to A with a role rejection notification.

Operations:

There is one high-level operation: the Requester-initiated PULL. This is because every node can function in the Requester and the Authority role.

General Flow:

1. Given a file handle or name, the Requester queries the Authority to find out if it has the file.
2. The authority determines whether it has the file and whether it meets authoritative criteria. If it does not, the Authority sends a PULL rejection response. Otherwise, the Authority creates a file authority session, sends an “ok to proceed” response to the Requester. The response also contains segmentation parameters.
3. If the Requester receives a rejection response, the PULL operation for the file is complete. If the requester receives an “ok to proceed” response, it segments its copy of the file, using the Authority-supplied parameters, and streams the segments to the Authority. Since the Requester cannot keep the entire file in memory (well, it could for small files, so maybe a size threshold?), it computes a checksum for the file and closes it. The checksum will be used later, when the rebuild instructions are received, to determine whether the file has been modified in the mean time.
4. The Authority receives segments as they become available begins compiling rebuild instructions composed of segment IDs and raw data blocks (can this be streamed?). When all Requester segments have been received and the build instructions are complete, the Authority streams the build instructions back to the Requester.
5. The Requester opens the local file again for reading (re-computes and validates the checksum) and a stage file for writing. Finally, Requester rebuilds the file in the stage and then swaps the original and stage. At this point, the PULL operation is complete.

Having the requester initiate a second file query while the first file is segmenting can pipeline the flow. The second file would be segmented while the first file is streaming, etc.

PacketSubscriber

RsyncNode

RsyncJob

RsyncRequestJob

RsyncAuthorityJob