The new XRootD client

Łukasz Janyst

CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

IT-DSS Group Meeting
Geneva, 02.03.2012

Outline

- High orbit flight over the XRootD protocol
- Key points of the implementation
- Issues
- Status
- What's next?

Message types

- Stateless vs. Stateful
  - Stateless: request-response (mkdir, rm, query...)
  - Stateful: open-do stuff-close (read, write, readv...)
  - The main difference in the error recovery:
    - retry or just go and ask elsewhere
    - vs. reestablish the state, possibly on another machine, and retry (not always possible)
- Solicited vs. Unsolicited
  - Responses to requests (ping-pong, classical client-server system)
  - Messages that have not been requested but affect operation (go away!, don't bother me until further notice, from now on start asking this other guy)

Responses

- The server may respond to a request or just talk unasked:
  - **kXR_attn** - unsolicited server info
  - **kXR_ok** - the query succeeded and this is the answer
  - **kXR_error** - failure
  - **kXR_oksofar** - kind of okay, some part of the response arrived, but problems still may occur
  - **kXR_redirect** - go and ask this other guy (if the request was stateful, the state needs to be re-established)
  - **kXR_wait** - busy, bother me again in n seconds
  - **kXR_waitresp** - the answer is not available now, will be sent in around n seconds as an unsolicited message

Message Handling

- Many messages may be sent over one socket without waiting for a response and responses may arrive in any order
  - each request has a stream identifier field assigned by the client which is echoed in the response, so that the responses may be associated with the requests
- Responses to some messages may be sent over other sockets associated with the same session
  - data-heavy messages: reads and writes
  - to optimize TCP congestion avoidance/window/slow-restart issues

New implementation-features

- Fully asynchronous
  - stateless requests may be handled asynchronously, not only reads and writes
    - listing of huge directories an order of magnitude faster
    - and, probably, heavier for the server to handle :(
  - callback model instead of request-and-poll-the-cache model
  - no need to have a cache to handle async communication
  - synchronous requests implemented in terms of asynchronous (with a semaphore)
- Thread safe
  - the user API classes hold very little or no mutable state at all

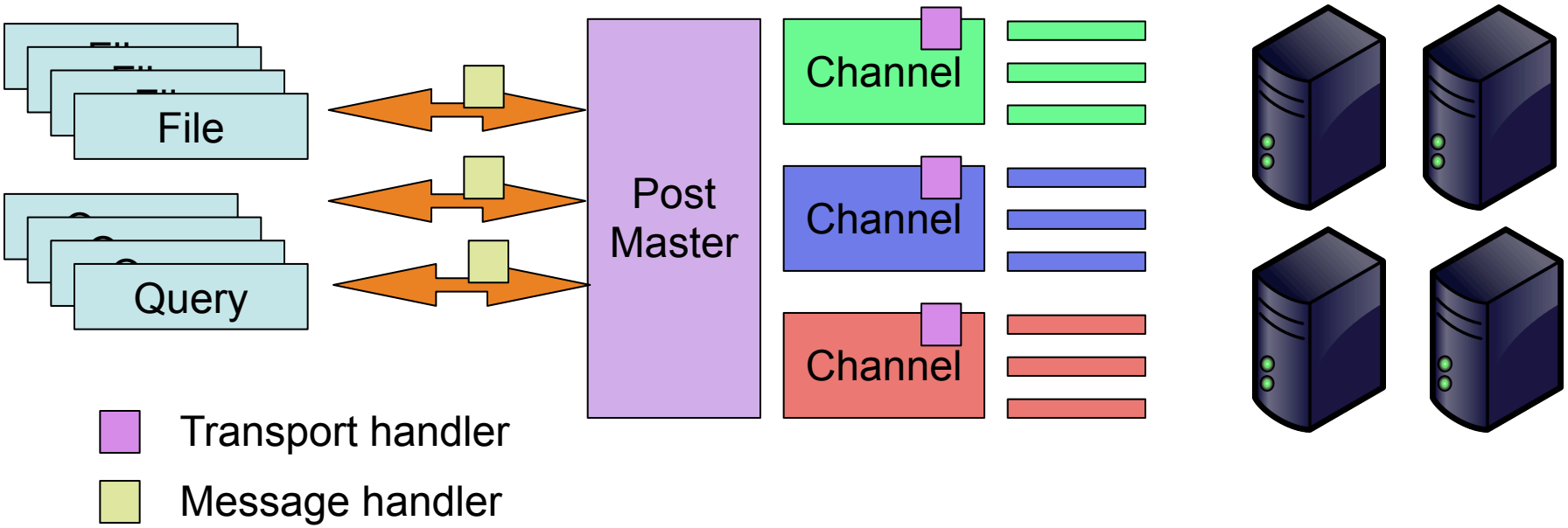
New implementation-features

- Lighter
  - one extra thread to handle socket events
  - one extra thread to handle time events
  - no need to spawn extra thread for every new connection
  - uses host system optimized polling (through libevent) instead of block+timeout model, which should reduce number of syscalls

User classes

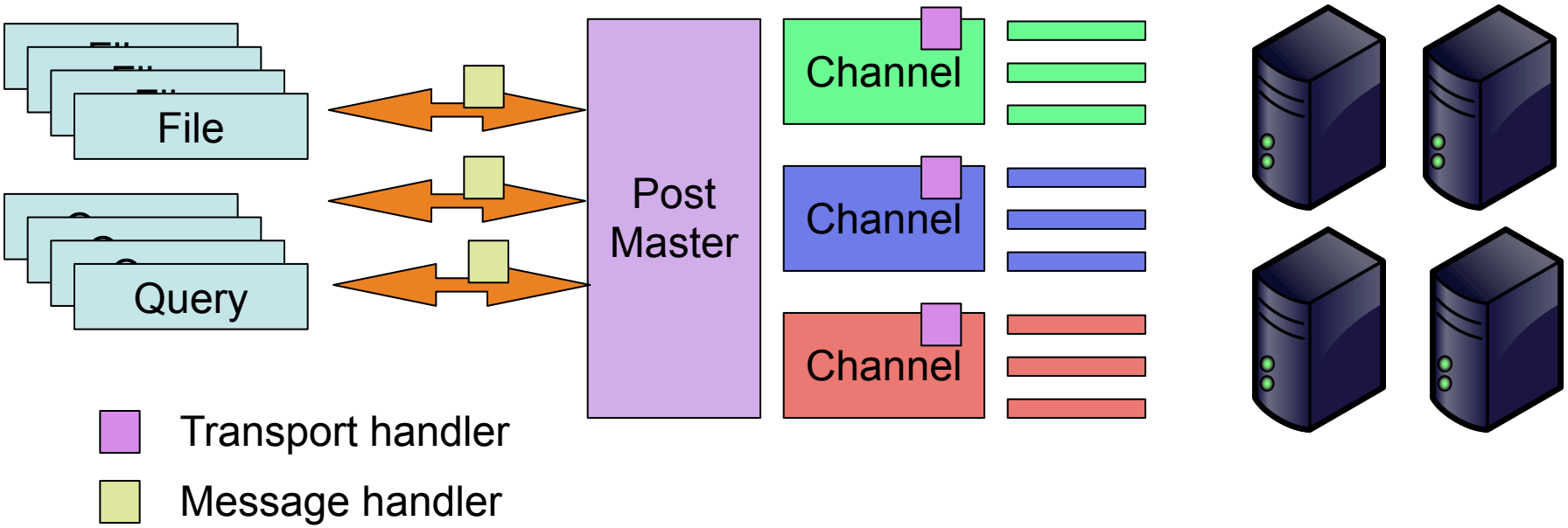
- **XrdClient::Query** for stateless requests
  - mkdir, rmdir, query, locate, move truncate, chmod, ping, stat...
- **XrdClient::File** for (stateful) file operations
  - read, write, readv...

Architecture



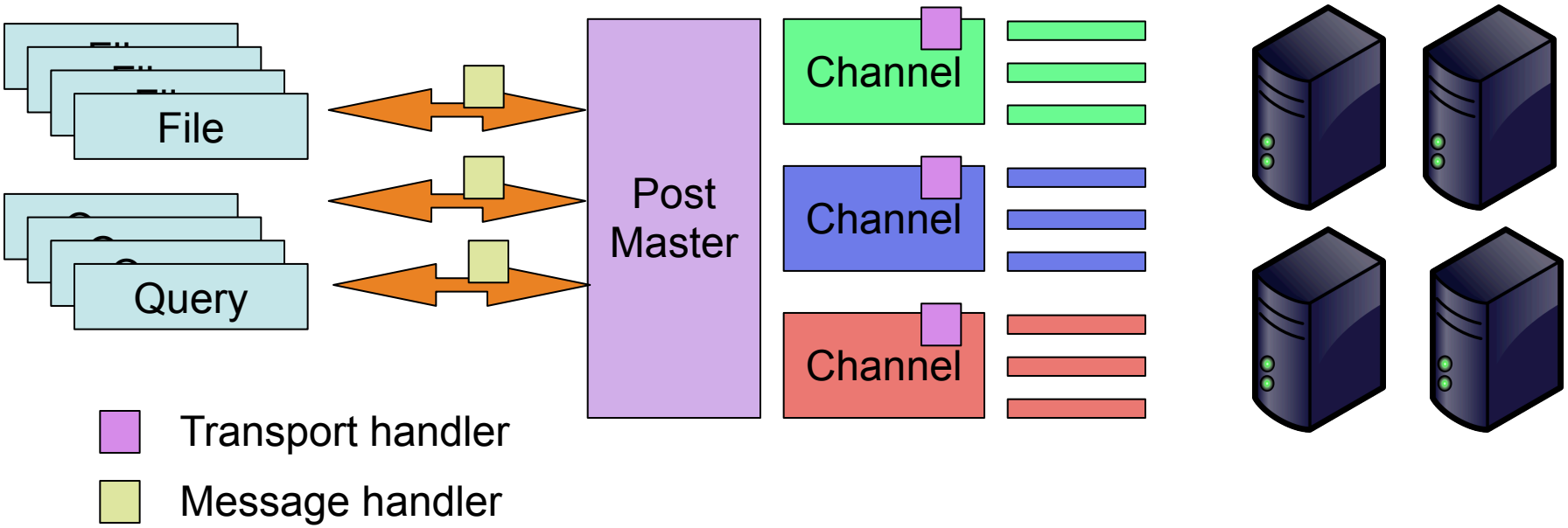
- **File** and **Query** objects
  - the user API entry points, they build and send the requests using a **PostMaster** object
  - hold as little mutable state as possible
  - register message handlers to interpret the incoming messages

Architecture



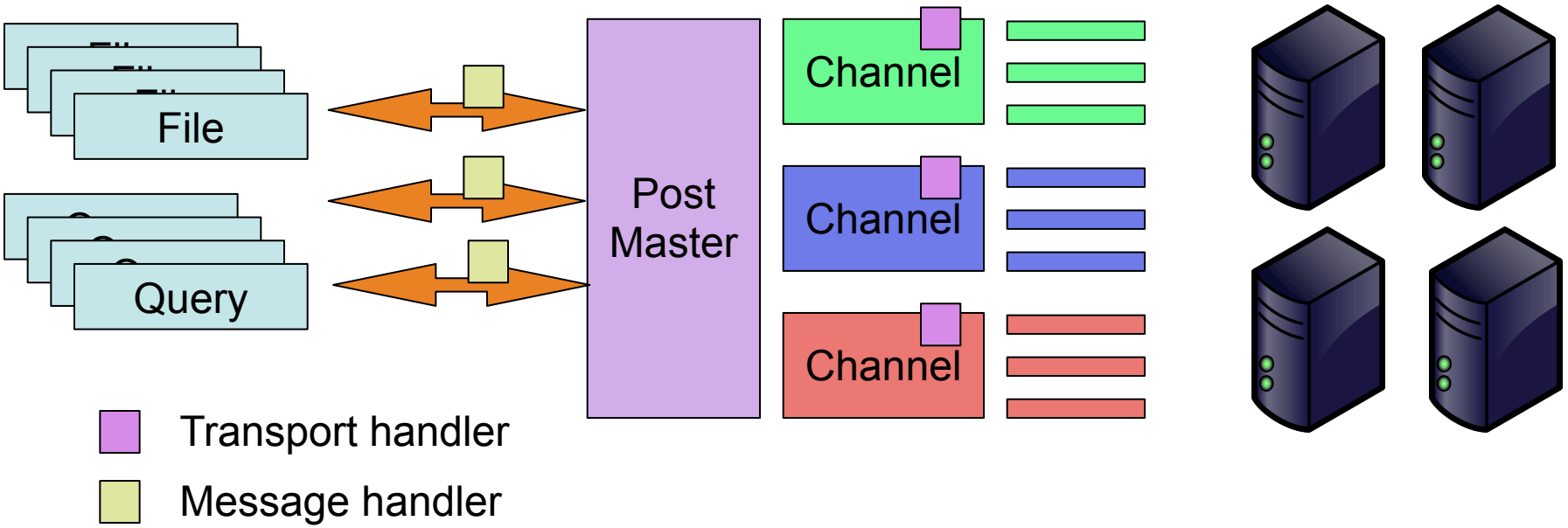
- **Message handlers**
  - receive the notification of incoming messages from the **PostMaster**
  - handle waiting, redirection, glueing partial responses, building response objects, notifying users about final results

Architecture



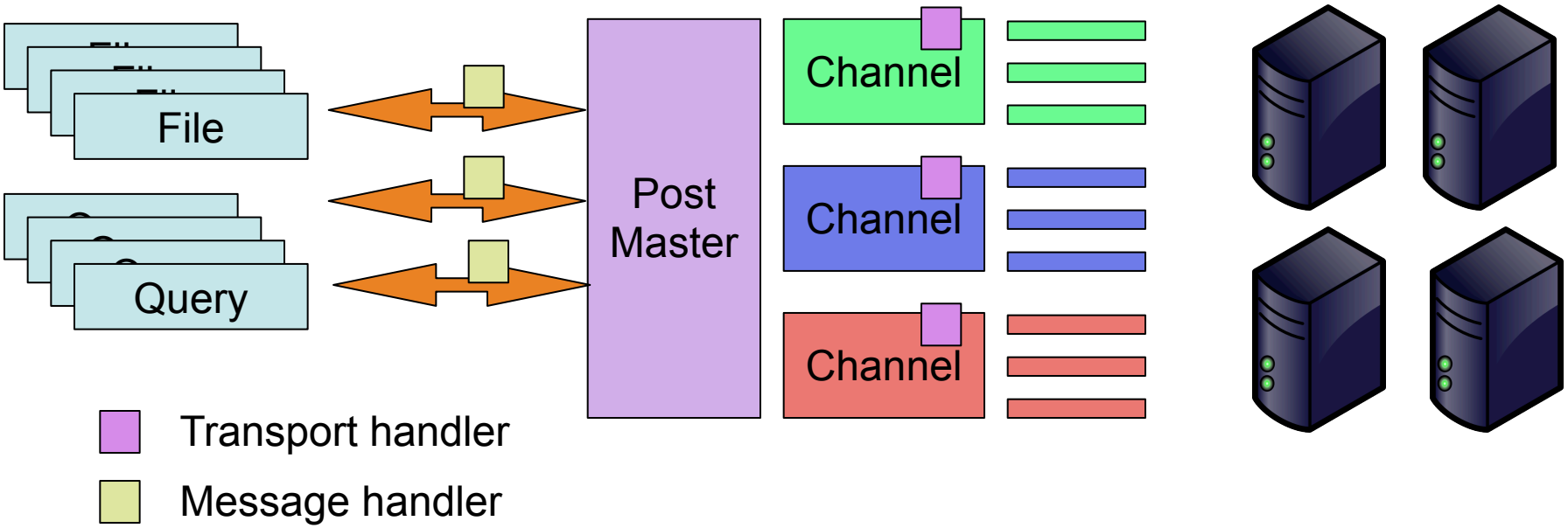
- **Post Master**
  - protocol independent (may be used for any message-based protocol over a TCP)
  - manages physical channels and streams to the servers
  - sends messages to given URLs

Architecture



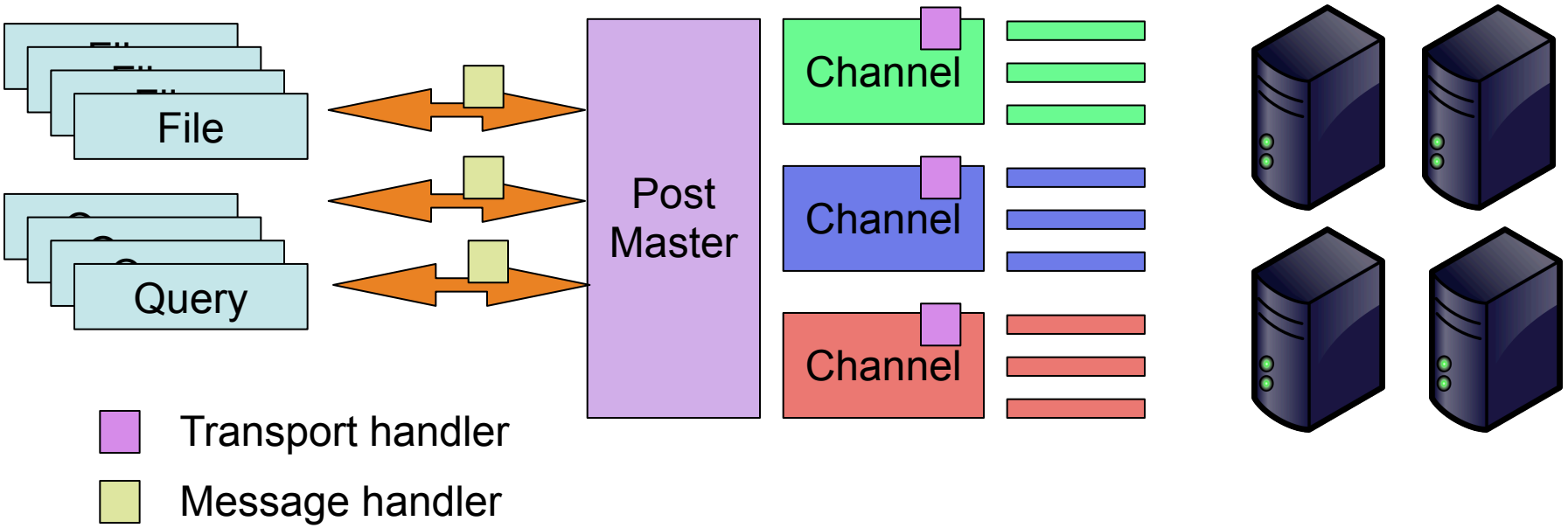
- **Post Master**
  - Manages message handler subscriptions to channels
  - Notifies the handler of incoming messages

Architecture



- **Transport handler**
  - entity within the post master responsible for handling specific protocol
  - assigned to a channel after interpreting the protocol part of the URL (root://, xroot://, whatever)

Architecture



- **Transport handler**
  - Handles logging-in and authentications
  - Stream and message multiplexing
  - Extracting messges from the stream TCP socket
  - Message marshalling and un-marshalling

PostMaster vs. SCTP

- Stream Control Transmission Protocol
  - a transport layer protocol
  - message oriented (like UDP)
  - reliable & congestion-aware (like TCP)
  - supports multi-streaming
  - implemented in Linux and Solaris > 10
- PostMaster could be dramatically simplified if we used SCTP instead of TCP
- To be investigated

Issues

- Sometimes not really clear how to deal with timeouts
  - What if the link is fine but servers tells us to wait longer than we want to?
- Recovery scenarios for complex (federated) setups.
  - Which manager to go to after encountering certain kinds of errors? Which errors should be revered in the local cluster and which on the federation level?
- How to handle write errors & redirections?
  - Start from the middle of a file on another server?

Issues

- Minor protocol issues discovered (easily fixable)
- Using libevent may be a deployment issue
  - Linux distros commonly provide 1.x but 2.x is required by the client.
- Code licensing, CERN's preference is (L)GPL, US Department of Energy prefers BSD.
- All of these to be discussed with the rest of the collaboration.

Status

- Stateful and stateless user API for most of the requests defined by the protocol: **DONE**
- xrdquery utility (replacing xrd): **DONE**
- xrdcopy utility (replacing xrdcp): **IN PROGRESS**
- Unit tests: **DONE**
- Doxygen API documentation: **DONE**
- Total line count: 17274 and growing (including comments and tests)

Next steps

- Caching & prefetching support
  - probably use one of existing solutions
- ROOT plugin (replacing current TXNetFile)
- IPv6 - the protocol is IPv6 aware, the implementation not necessarily
- Test, test, test, test, test & iron-out the rough edges
  - use internally in EOS
  - planned to use in CASTOR when migrating to pure XRootD internal data transfers

Questions

Thanks for your attention!

Questions? Comments?