XRootD/XTNetFile a robust and fault tolerant extension of RootD/TNetFile

Alvise Dorigo, Fabrizio Furano
Venezia & Padova University – INFN Padova

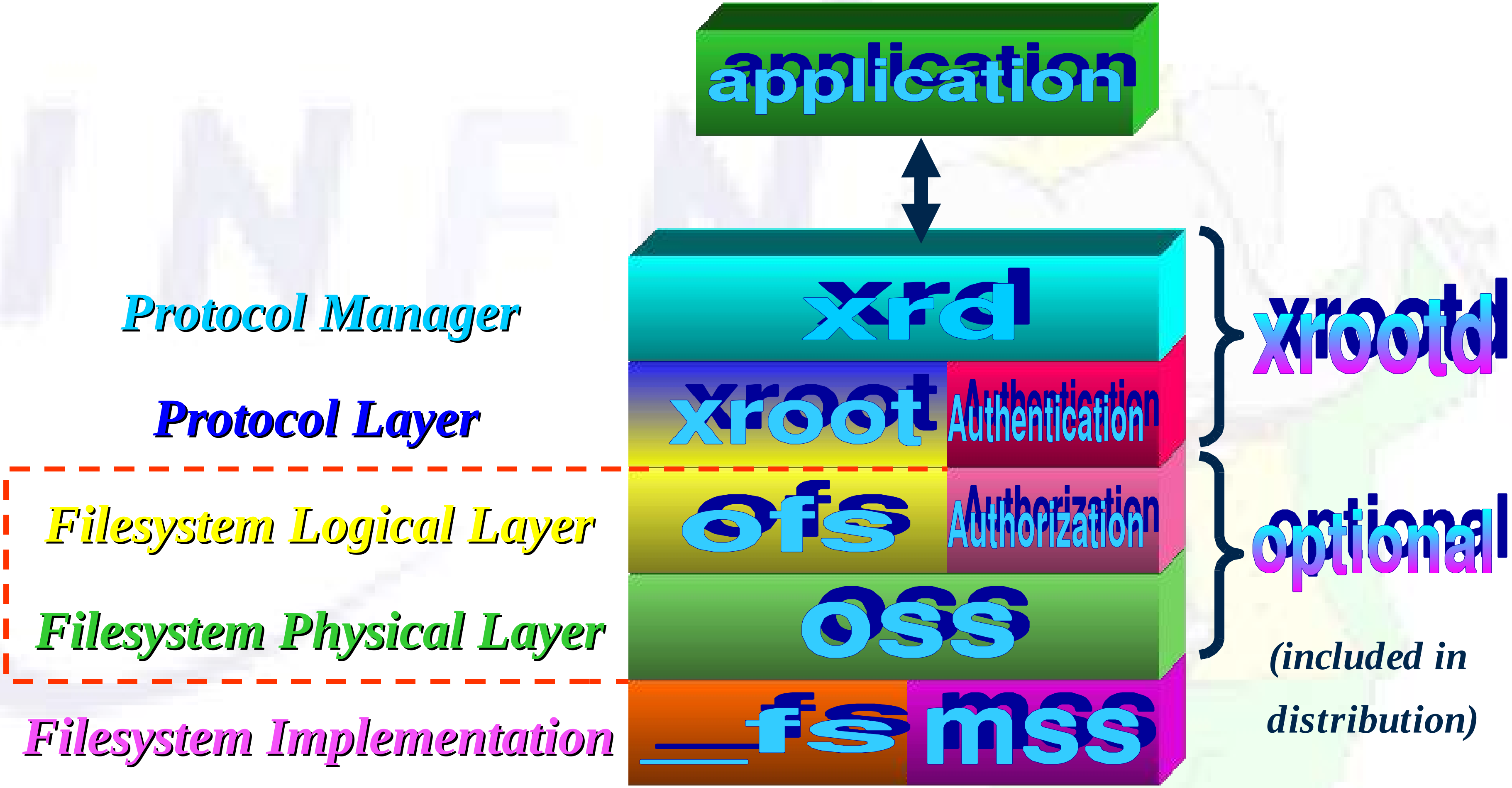
Main goals

- An extension of the TnetFile/RootD system
  - Fault tolerance
  - High performance
  - No resource waste
    - Possibility of effective use of the available computing power
      - In small to large scale sites
  - Nearly linear scaling
  - Possibility to distribute huge data repositories
      - Over many load balanced servers

xrootd

- A high tech scalable replacement for rootd
    - Plugin architecture – can revert to rootd protocol
    - Security – virtually any security protocol
    - Multithreaded code – sticky sockets
    - Connection multiplexing
    - Load balancing, based on
        - redirection mechanism
        - communication between servers and load balancers
    - Extendable protocol
    - Mass storage integration
    - I/O segmenting, caching, server side read-ahead

XrootD internals



Protocol Manager

Protocol Layer

Filesystem Logical Layer

Filesystem Physical Layer

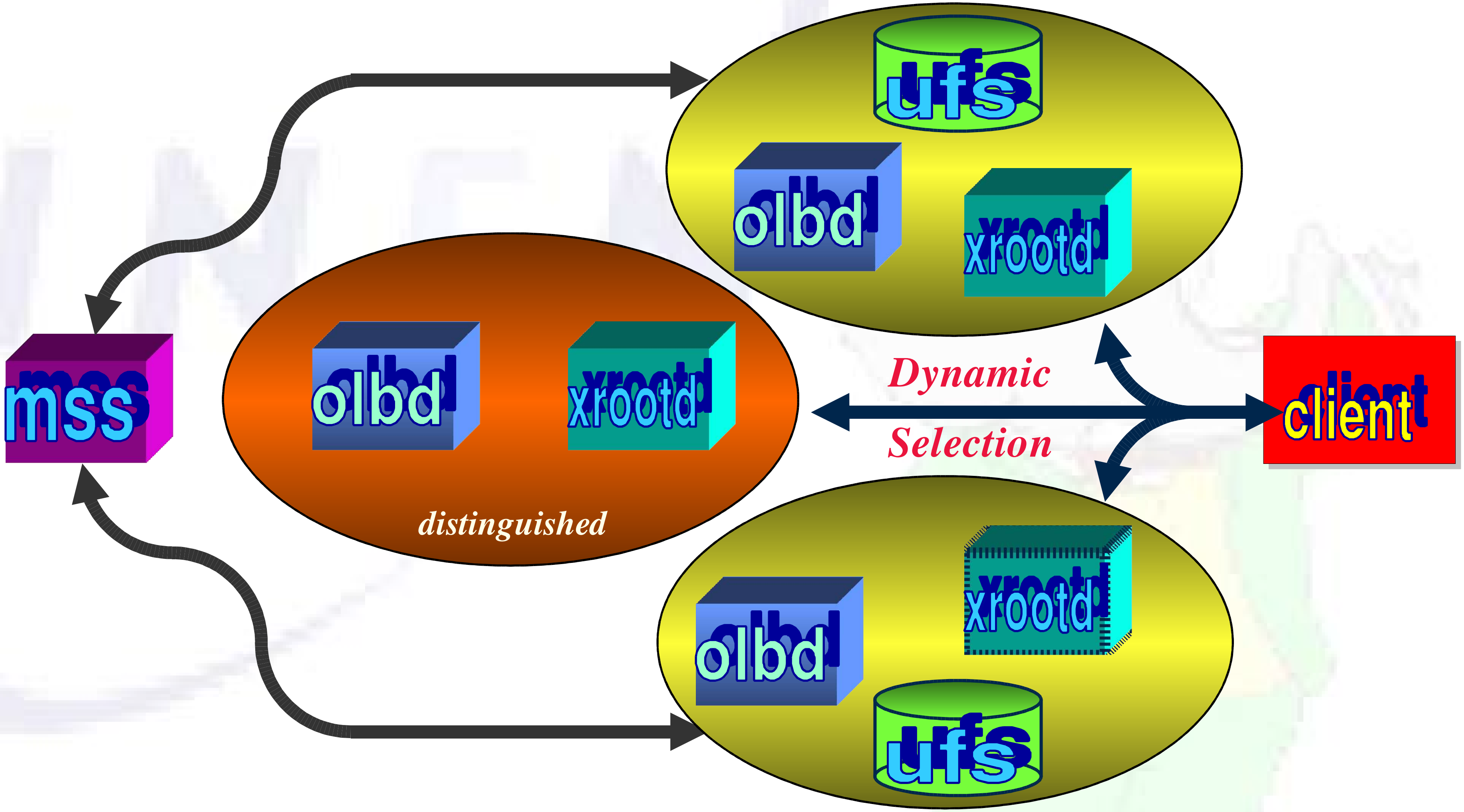
Filesystem Implementation

xrootd

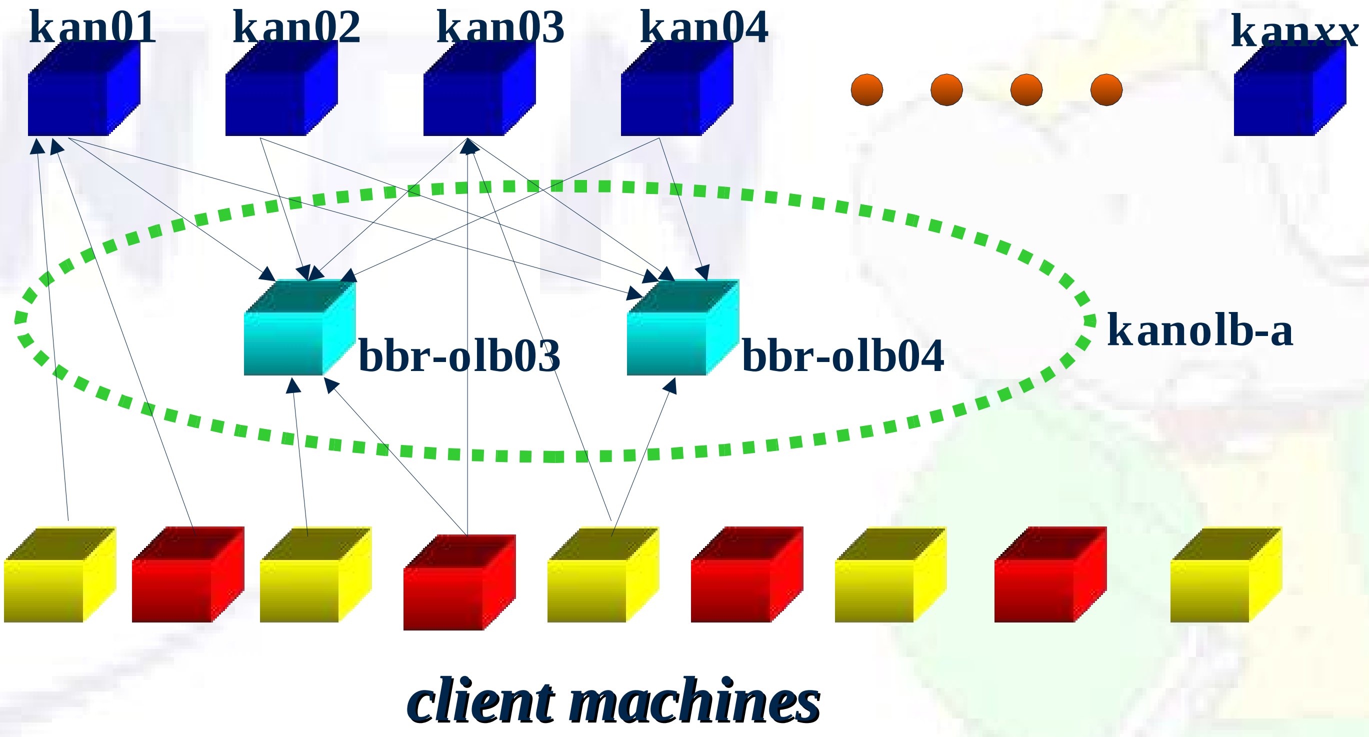
optional

(included in distribution)

XrootD load balancing



Example: SLAC configuration



A client for Xrootd

- Many xrootd's features must have a counterpart in its client

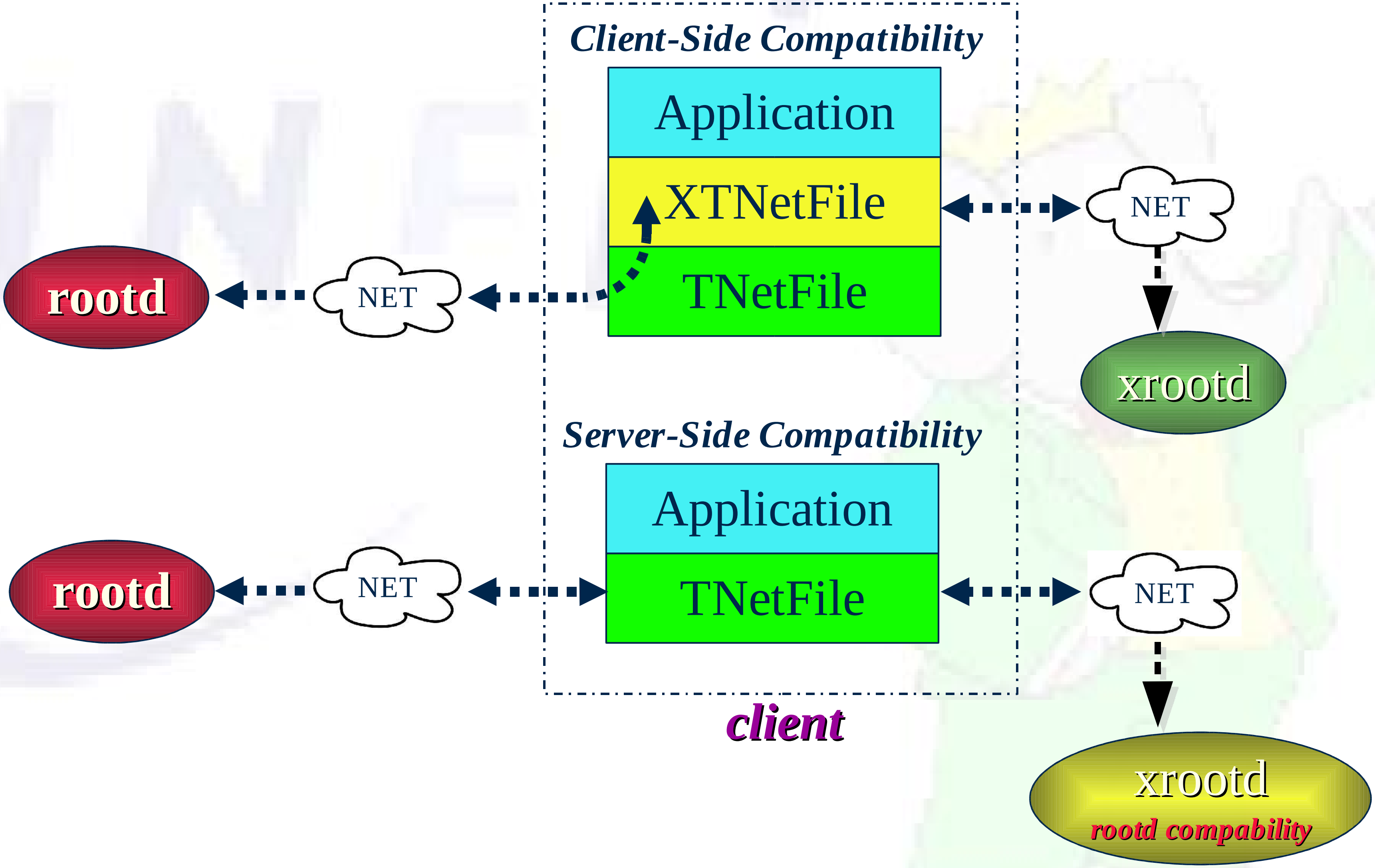
XTNetFile I

- ROOT class used to communicate with a rootd or xrootd server
  - Supports xrootd protocol, with connection multiplexing (several clients mapped on a single physical conn per server) and security
  - Can detect the server kind (rootd, xrootd) and revert back to rootd protocol if needed. Transparently.
  - Supports the client side of the load balancing mechanism (redirections)
  - Error recovery, as defined in the Xrootd protocol documentation
- Supports redirections in data tranferring too

XTNetFile II

- XTNetFile extends TNetFile, the original rootd client
  - Server handshake: transparently detects the server type
  - Redirection mechanism, used for both load balancing and error recovery
  - Main purposes
    - shrink near to 0 the number of jobs/processes unable to proceed due to transient communication troubles
    - Allow many clients to share resources through load balanced servers

Backward compatibility of client and server



XTNetFile / XTNetAdmin

XTNetFile

A single file abstraction

ReadBuffer*
WriteBuffer*
Open*
Close*
ReOpen*
GetRemoteFile

*=overrides of TNetFile methods

XTNetAdmin

Utilities to administrate files and directories

ExistFile
ExistDir
IsFileOnline
PrepareFile (TBF)
Mv
Mkdir
Chmod
Rm
Rmdir

Startup Behaviour

- On startup, XTNetFile can be given a multiple URL in the constructor:

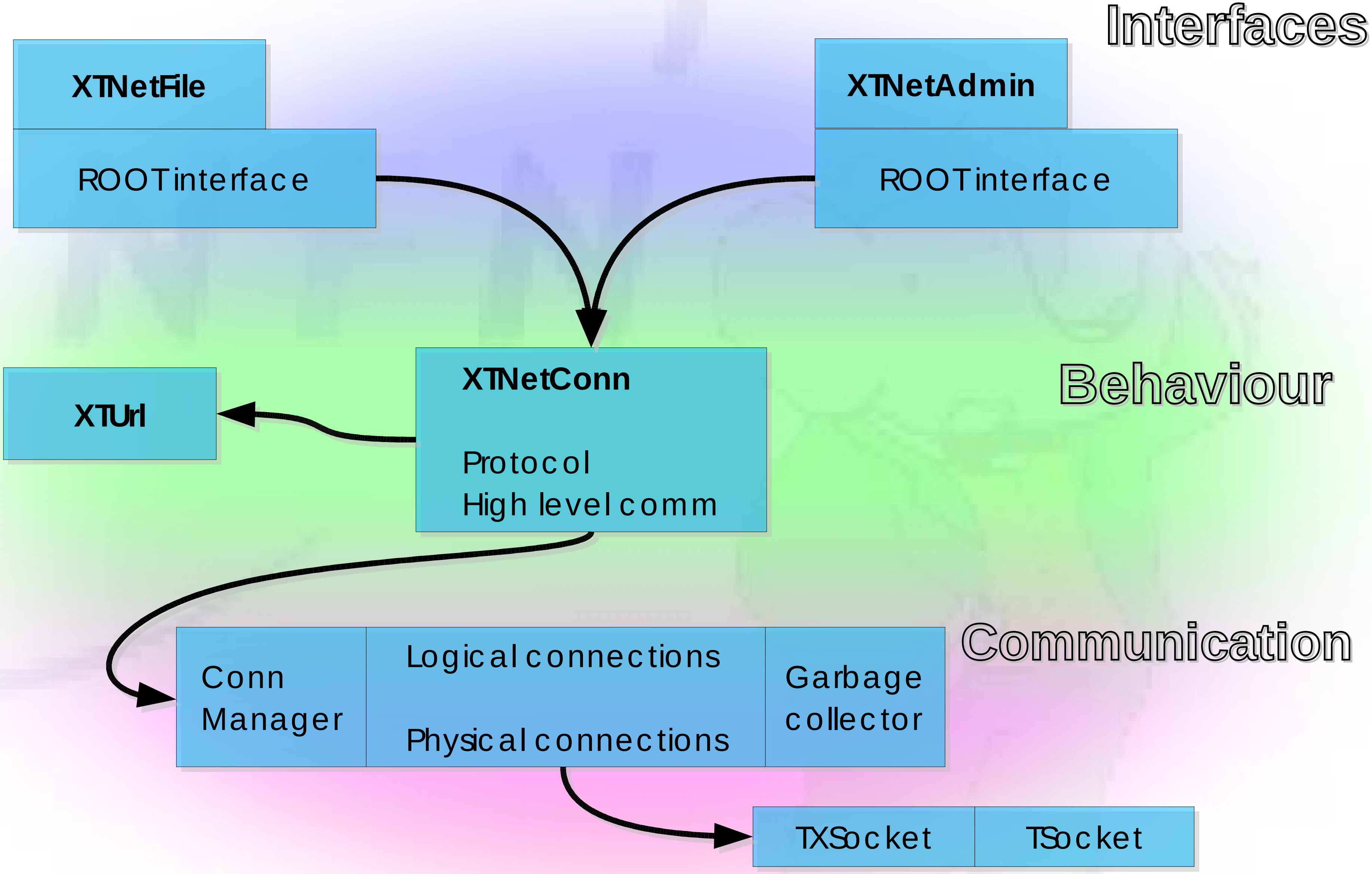
```
root://host1[:port1][,host2[:port2]]...[,hostN[:portN]]/path/file
```

  - DNS aliases are supported. A hostname can throw to a number of servers to choose (random w/o reinsertion); default TCP port 1094 (like rootd)
  - Then it tries to connect for a number of times (default is 120 with 10 secs delay and 30 secs timeout) to one of the resulting servers (random w/o reinsertion)
- After connection, a handshake tells what to do:
  - xrootd protocol or forward calls to TNetFile
- Strict timeout rules are applied to each attempt. Clients have never to lockup.

Error recovery

- Strict timeout rules are applied to all read/write
  - Everything is parametrizable via .rootrc (reading values for all parameters via the ROOT gEnv static object)
- A read/write error is treated as a redirection
  - To the first encountered load balancer (if any)
  - To the same server (rebouncing) if no load balancer
  - Each reconnection attempt is counted as a redirection
- XTNetFile gives up when a max redirection count is reached (default is 256 per hour)
  - A failing command is retried a number of times (default is 10 with 10 secs delay)
  - Can refresh the load balancer file mapping if a data server does not finds a file it was supposed to have
  - Survives to redirections to offline servers or several load balancers

XTNetFile Architecture



Sync vs Async

- Some enhancements in the xrootd protocol ask for an asynchronous structure
  - Unsolicited responses
    - e.g. xrootd admin interface, safe server shutdown
    - A closing server can redirect all the clients (even idle) to another one
    - A server can ask for the clients to wait...
  - Can be useful for multithreaded client applications (performance)

Dev status

- XTNetFile is in production for BaBar
  - Large scale processing needs close to perfection communication primitives
    - Robustness is satisfying, users can kill servers, restart, etc. Without negative consequences
    - If needed, a complete integration inside ROOT is possible (some POSIX stuff, details on socket polls)
- Multithreaded async architecture is there
  - Parametric choice (a gEnv param can turn ON/OFF the async behaviour), interface and primitives are the same
  - Will be production tested in parallel with the server's new features
- Security will be deployed shortly
  - In parallel with the server plugins (and the actual needs)

Work in progress

- Client side Read-ahead
  - ROOT analysis jobs (in BaBar) tend to request very small packets
    - Disks and comm latency limit max performances
    - Reading ahead can boost performances and lower server side load, but only if it's convenient
- Client side Caching
  - The typical analysis jobs (in BaBar) do not process data in a **strictly** sequential way
    - Some kind of caching (of read-ahead blocks) can help even more
- Measurements and developing are in progress
  - Goal: an auto-configuring read-ahead & caching mech.