

---

# **XNetFile**

## **a robust and fault tolerant extension of TNetFile**

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

---

# 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)
    - Extendable protocol
    - Mass storage integration
    - I/O segmenting, caching, server side read-ahead
  - Many xrootd's feature must have a counterpart in its client
-

# XNetFile 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 transferring too
-

# XNetFile II

---

- XNetFile 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
-

# XNetFile / XNetAdmin

---

## **XNetFile**

A single file abstraction

ReadBuffer\*

WriteBuffer\*

Open\*

Close\*

ReOpen\*

GetRemoteFile

\*=overrides of TNetFile  
methods

## **XNetAdmin**

Utilities to administrate files  
and directories

ExistFile

ExistDir

IsFileOnline

PrepareFile (TBF)

Mv

Mkdir

Chmod

Rm

Rmdir

# Startup Behaviour

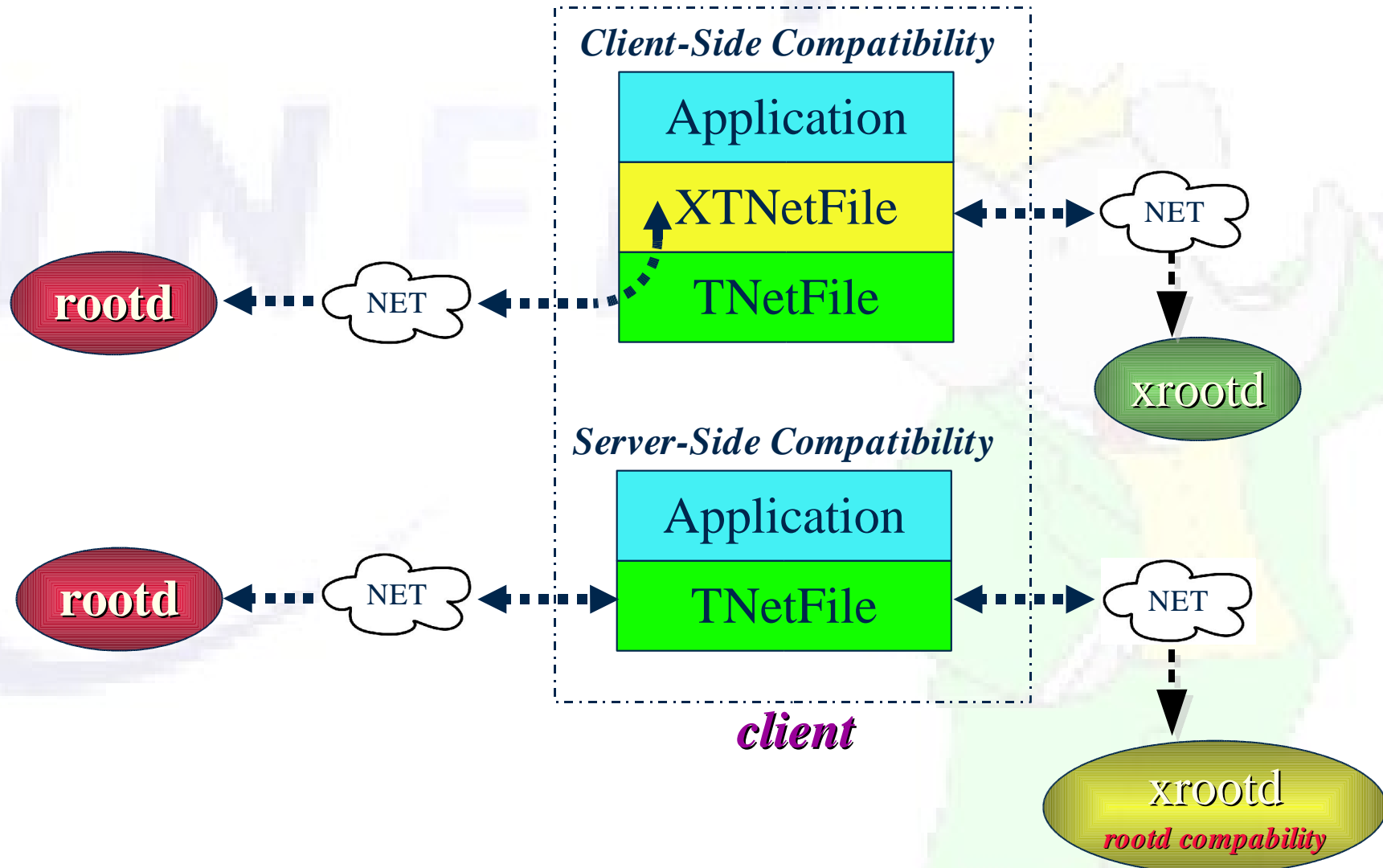
---

- On startup, XNetFile 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
  - 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.
-

# Backward compatibility of client and server

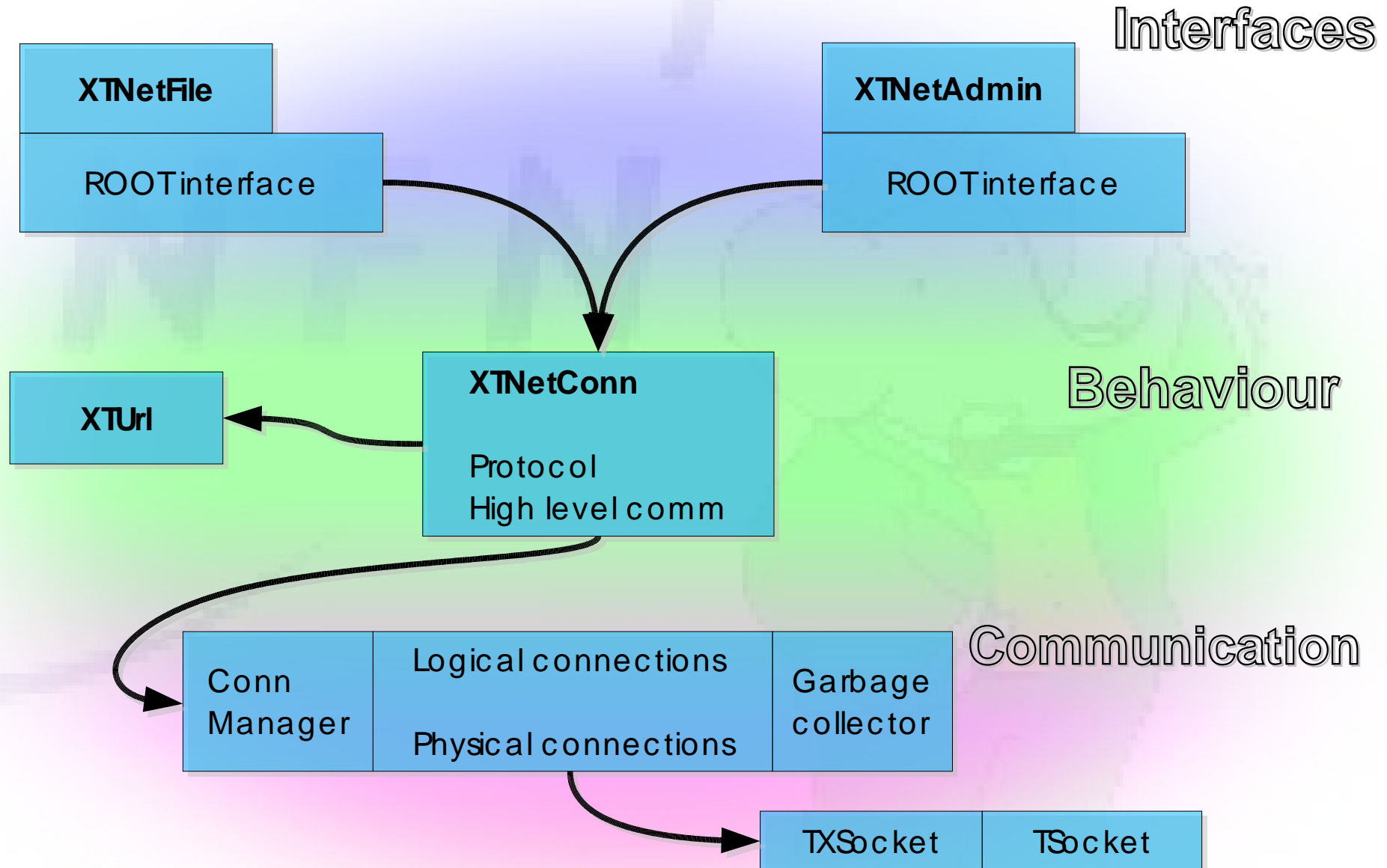


# Error recovery

---

- 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 (rebounding) if no load balancer
    - Each reconnection attempt is counted as a redirection
  - XNetFile 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 find a file it was supposed to have
    - Survives to redirections to offline servers or several load balancers
-

# XNetFile 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 the clients to wait...
  - Suitable also for:
    - Parallel client side tasks (e.g. Client side read-ahead)
    - Data transfer performance enhancing (Client side buffering)
-

# Dev status

---

- XNetFile 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)
  - 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
-