TXNetFile

A fault tolerant extension of ROOT TNetFile

Alvise Dorigo – Peter Elmer - Fabrizio Furano – Andrew Hanushevsky – SLAC - BaBar – INFN Padova - 2004

Purpose

When dealing with the concurrent access from a multitude of clients to petabyte-scale data repositories, high performance, robustness, and scalability are very important issues.

To address these challenges, xrootd and TXNetFile have been designed. Such a technologiy can also be used in any other field in which a reliable and scalable data access is an important issue.

The current deployment allows thousands of batch jobs and interactive data analysis sessions to effectively access the data repositories of the BaBar experiment.

Hence, the built distributed data access systems are highly robust and tolerant to communication troubles, load balanced and scalable to an extent which allows “no jobs to fail”, as long as the clients are able to find their path over servers which may be overloaded, restarted, become unavailable or prone to communication errors.

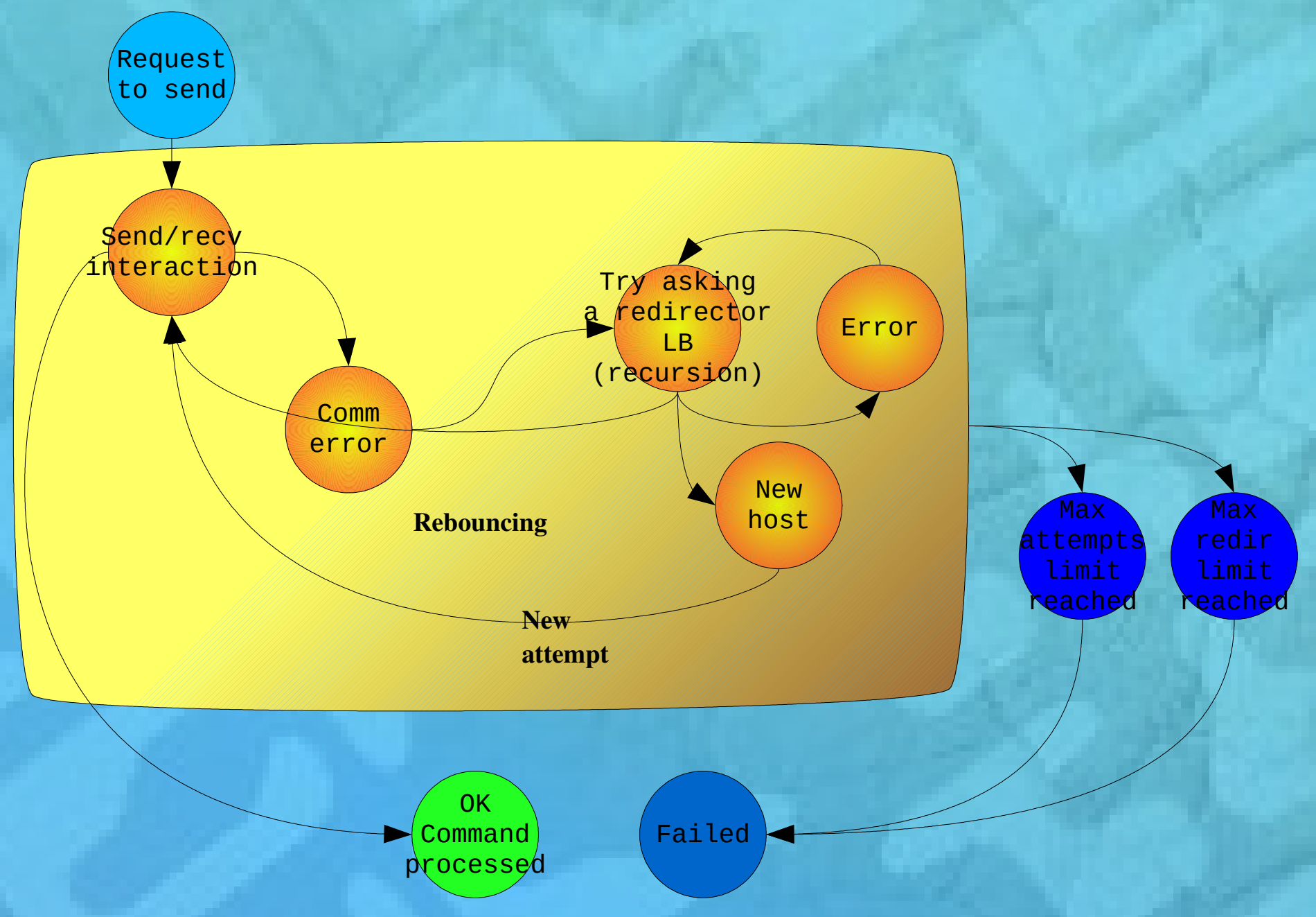
Fault tolerance

Multiple servers, plus DNS aliases expansion make the client able to find its way through a dynamical set of servers. The client tries to connect randomly to the hosts coming from the processing of an URI until one accepts it

Communication robustness

Every connection or communication attempt has applied a specified timeout. When it elapses, an error condition is generated and handled internally.

Every low level communication error is handled internally.



Scalability

The implemented communication protocol makes servers able to redirect clients to other ones. Hence, the client can deal with a network of servers spanned through multiple hosts, each partecipating in giving the whole data access.

Clients are redirected where the files are, while the load of multiple servers keeping redundant data gets balanced.

Multiplexed persistent connections and multithread structure make this feasible through the lowering of the needed system resources at the client and server side.

Security

The secure authentication client/server handshake guarantees no use of the resources by unauthorized users.

Allowed/denied domains for connection and redirection make the system administrators able to limit the freedom of the clients to connect everywhere, thus avoiding potential DoS attacks to external sites and unpermitted accesses to outer network domains.

ROOT integration

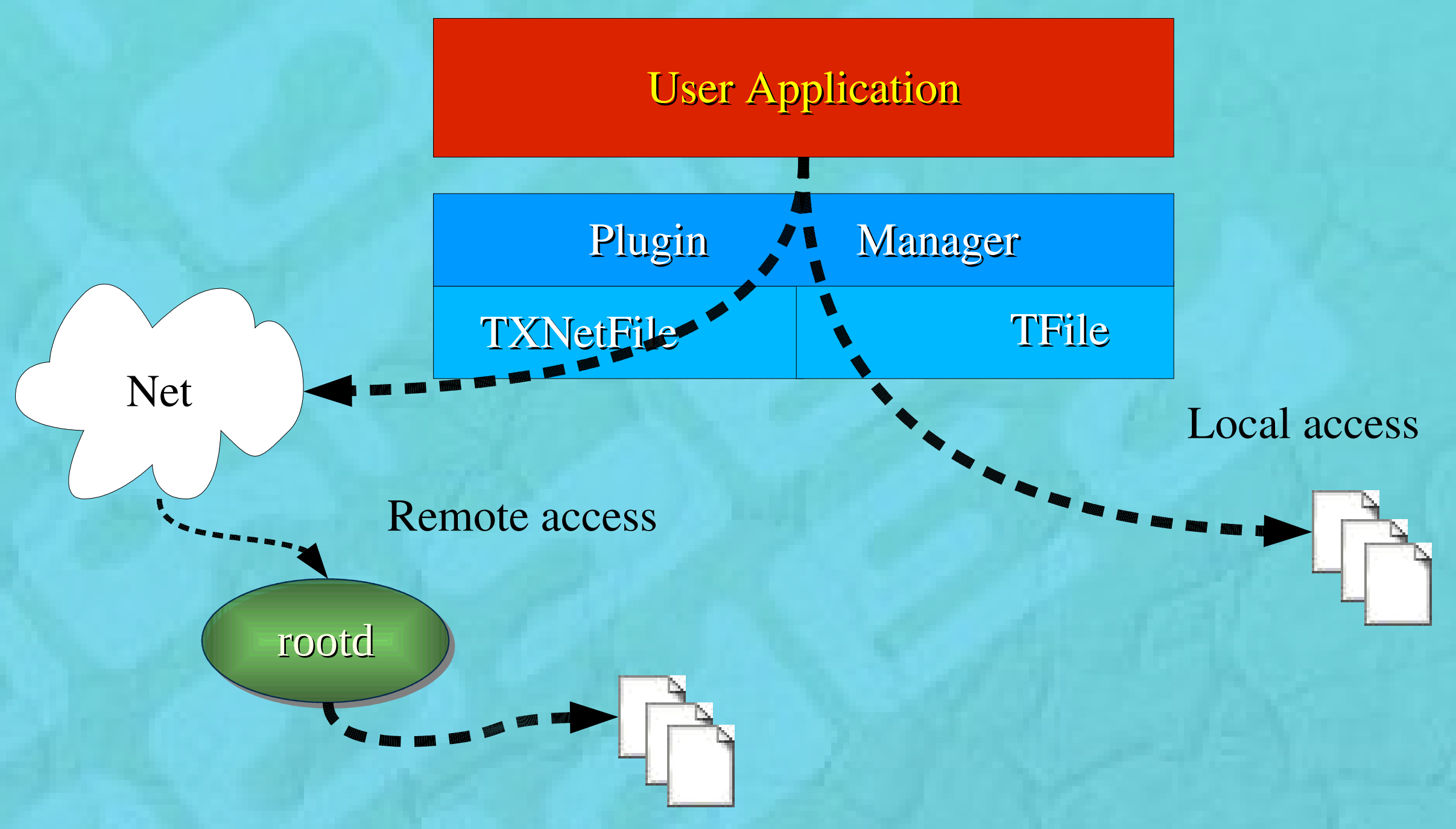
Entirely coded in C++, the client side was born as a ROOT extension, officially available with the newest releases of ROOT 4. Also, a general UNIX reference implementation exists and a Java one is in development also for mobile devices.

TXNetFile is an extension of TNetFile, hence it's able to communicate also with rootd servers in a fully transparent back compatibility mode. Existing applications can switch to the new features by using it instead of TNetFile.

Transparent for applications

What makes even easier for an application to use the features of TXNetFile in conjunction with xrootd is the plugin architecture of ROOT.

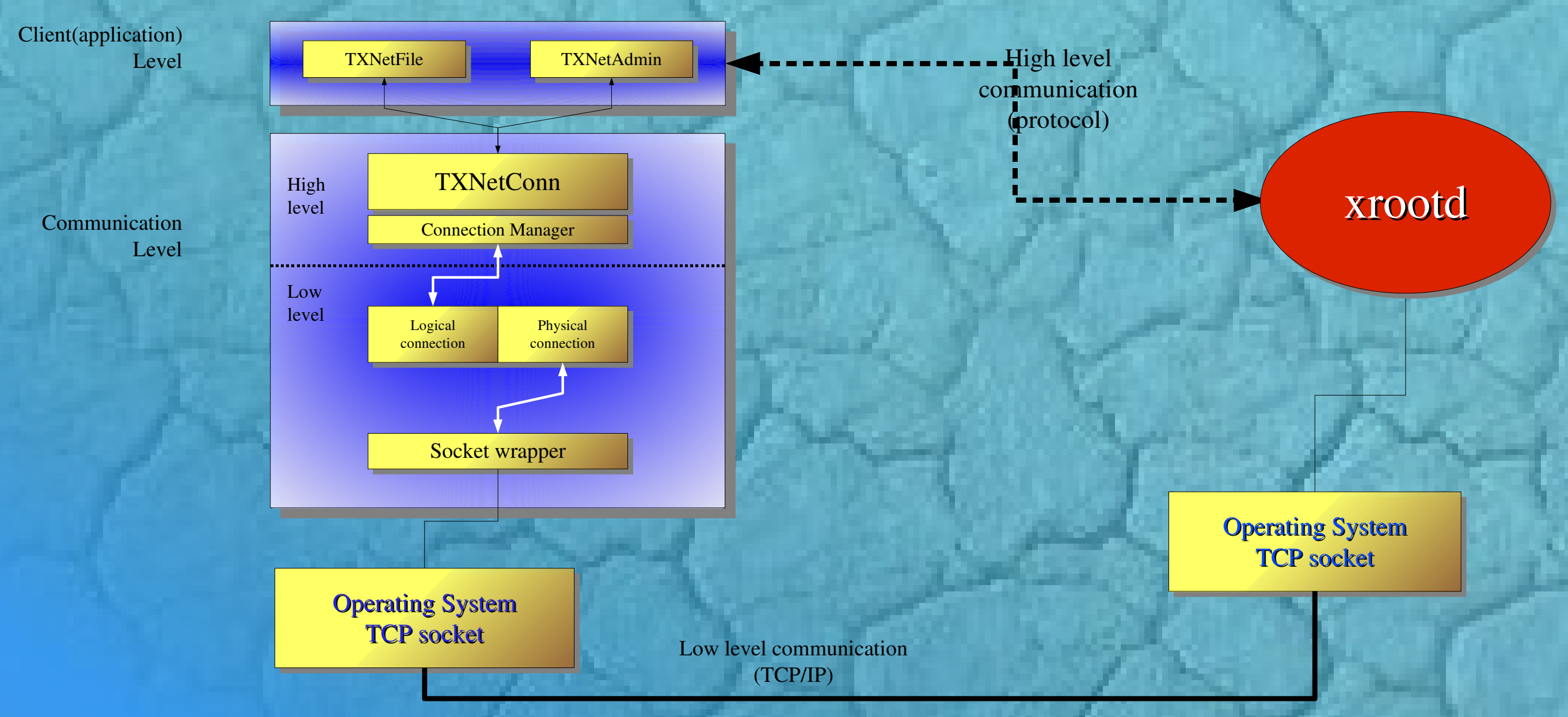
Given an URL specifying a data access protocol, the Plugin Manager provides at runtime the correct object to be used by the application, which remains unmodified and unaware of the communication paths it's using.

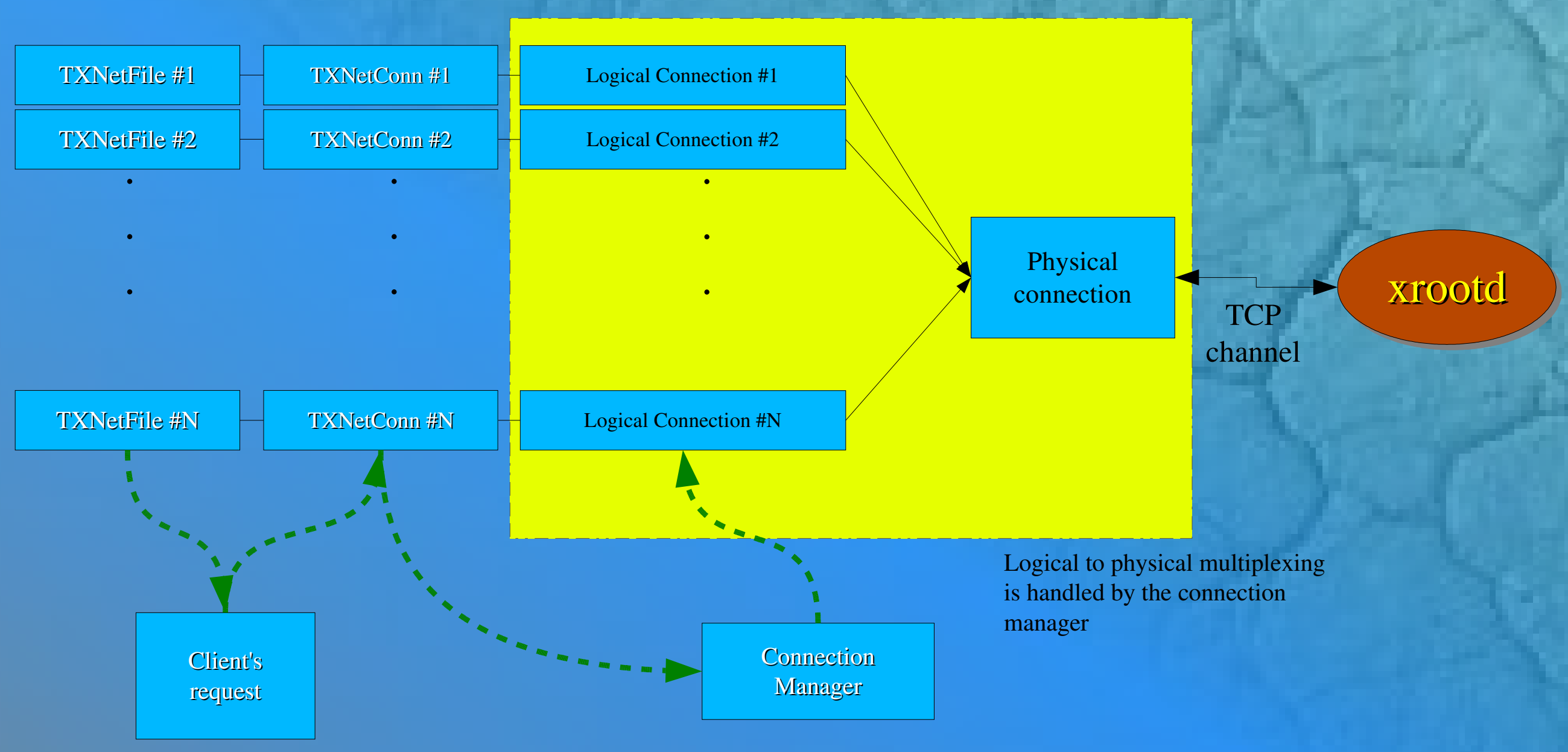


Architecture

TXNetFile is composed by three layers:

- the interface layer: here the XTNetFile class reimplements all the virtual methods inherited from ROOT's TNetFile; another class TXNetAdmin has been built at this layer in order to perform administration operations like file copy/moving/stating/ deletion, file retrieve from tape systems, file system space check, etc.
- the high level communication layer: here the protocol directives are implemented, as well as the load balancing and the policies related to the fault tolerance;
- the low level communication layer: here the protocol packet structure is known, together with the physical connection information, in order to give the functionalities of: connection multiplexing, socket polling, raw data sync/async reading, raw data writing, error handling.


Client(application) Level
TXNetFile
TXNetAdmin
High level communication (protocol)
xrootd
Communication Level
High level
TXNetConn
Connection Manager
Low level
Logical connection
Physical connection
Socket wrapper
Operating System TCP socket
Operating System TCP socket
Low level communication (TCP/IP)


TXNetFile #1
TXNetConn #1
Logical Connection #1
TXNetFile #2
TXNetConn #2
Logical Connection #2
Physical connection
TCP channel
xrootd
TXNetFile #N
TXNetConn #N
Logical Connection #N
Logical to physical multiplexing is handled by the connection manager
Client's request
Connection Manager