

Transport

Table of contents

1 Introduction.....	2
2 Addresses.....	2
3 End Points.....	3
4 Connection.....	3
4.1 Buffered Connections.....	3
4.2 Stream Connections.....	3
4.2.1 Acceptor.....	4
4.2.2 Connector.....	4
4.3 Compound Connections.....	4
4.3.1 Composition.....	4
5 Supported Transports.....	4

Note:

Tuesday 2006-03-28: What follows describes not the current implementation of the software but the software as it shortly will be. The description is also partial. It describes blocking operations whereas the existing software also supports non-blocking I/O.

1. Introduction

When an XLattice node needs to communicate with (pass data to or receive data from) another node, it uses a transport. There are many types of transports. The two most used by XLattice applications will be [UDP](#) and [TCP](#) and subclasses of these.

UDP is an unreliable transport which carries **datagrams**. A datagram is a block of data with a fixed length. A datagram is transmitted as a whole. We shall use **send** and **receive** as names for the basic I/O operations for this type of transport.

TCP is a reliable transport based on **streams**. We shall be using **read** and **write** to name I/O operations for streams-based transports.

TCP and other transports derived from it use **connections**. For our purposes, a connection is a triplet:

- a local address
- a remote address
- a transport

UDP also uses connections but with somewhat different semantics. By default a UDP port is promiscuous: it can receive data from anywhere. If it is connected and has a remote address specified, then incoming UDP packets (**datagrams**) from other addresses to the local address are discarded.

For outgoing datagrams, binding the remote address makes it unnecessary to look up the address on each transmission, which can save a great deal of time.

2. Addresses

An XLattice address is a number or string which is meaningful for the purposes of a transport. If the transport is UDP or TCP running over IPv4, a suitable address is a 32-bit host address plus a 16-bit port number, each in **big-endian** network byte order (meaning that the most significant byte is first). A valid domain name, one which can be resolved into a host address, plus a port number can be used for the same purpose. In either case the XLattice address is a six-byte number.

However, it's important to understand that there are other types of addresses used by other transports. For example, the transport might be SMTP, the email protocol, in which case "node07@abc.example.com" would be a valid address if the node could use it to receive data.

TLS (Transport Layer Security) is the IETF's version of SSL, the transport underlying HTTPS, the secure Web protocol. In the XLattice implementation of TLS, the address includes the location of the key store containing the keys and the passphrase necessary to use it.

Addresses need not be unique. Generally, if an address is not unique, then if a message is sent to the address a copy should be delivered to all nodes sharing the address (which would then have been construed as a **broadcast** address). However in some cases (as in round-robin DNS), it might be possible to arrange that a number of nodes share the same address, but when a connection is made only one of the nodes participates.

3. End Points

An XLattice node will have a number of local addresses. In order to be useful for communications, an address must be associated with a transport. An **EndPoint** abstracts that association: it is an address plus the transport.

4. Connection

As we use the term, a **Connection** is a pairing between a local EndPoint (an address plus a protocol) and a remote EndPoint. We shall assume that there are two basic types: **buffered connections** and **stream connections**.

4.1. Buffered Connections

A BufferedConnection carries fixed length messages which are either received or sent in their entirety or not at all. The abstract class has an input buffer and an output buffer and supports **send** and **receive** as I/O operations.

For our purposes, a **UDP connection** is a subclass of the BufferedConnection. It is unreliable in that a UDP datagram may be delivered any number of times: zero, one, or many.

An EndPoint using UDP or another datagram protocol does not need to be connected to communicate, to send or receive messages. However, it may be more efficient to **connect**, to bind the remote end, if more than one message is to be sent to it.

If the transport is UDP or a subclass, the connection is a **DatagramConnection**.

4.2. Stream Connections

An EndPoint using TCP or another stream protocol must bind the far end in order to read or write messages. TCP and its subclasses uses a **StreamConnection**.

4.2.1. Acceptor

An **Acceptor** is an EndPoint which accepts connections. That is, it is a generalization of a server socket.

4.2.2. Connector

A **Connector** is an EndPoint which can make a connection to another EndPoint which is willing to accept such a connection. In other words, it abstracts a socket.

4.3. Compound Connections

A **simple** connection is one in which the transport is the same at both ends and no intermediate nodes act as relays.

Nodes can agree to cooperate in constructing **compound connections** in which intermediate nodes relay messages on behalf of other nodes. In an XLattice compound connection the same transport is used on all constituent segments.

Intermediate nodes can also act as **translators**, relaying messages received over one transport out over another. A **complex connection** is a compound connection in which one or more of the intermediate nodes translates as well as relays messages.

4.3.1. Composition

Composition is the operation which creates one connection from two or more. One of XLattice's fundamental objectives is to make composition straightforward, so that reliable compound connections can be built very easily, and preferably automatically.

5. Supported Transports

At the time of writing (2006-03-28):

- both blocking and non-blocking TCP are supported
- blocking TLS is partially implemented
- development of XLattice blocking UDP has just begun