Connection Class

A **Connection** object represents a connection between a local **Port** object and a remote computer, specified by an IP address/port pair. The **P2PTransport** class manages a number of connections, one between each remote address and each local address, and chooses the best one for sending data. **Connection** objects are responsible for monitoring the health of their writability, and reporting this, as well as sending out pings to keep a connection open, if necessary, though these tasks are handled automatically. Connections send the **Connection::SignalReadPacket** signal when new data arrives over the network; **P2PTransportChannel** calls **Send** to send data out to the remote computer.

This is a pure virtual class. libjingle includes two subclasses of **Connection**: **ProxyConnection**, which defers most of the work to the **Port** object, and **TCPConnection**, which connects directly with the socket itself. This page describes both subclasses; **TCPConnection** differs only in the addition of a constructor, a destructor, and the **Socket** function. A TCPConnection object works with **TCPPort**. A ProxyConnection works with **UDPPort** , **StunPort**, and **RelayPort**.

Connection defines the following timeout values:

Const Name	Description	Value
CONNECTION_READ_TIMEOUT	The length of time we wait before timing out readability on a connection.	30 seconds
CONNECTION_WRITE_TIMEOUT	The length of time we wait before timing out writability on a connection.	15 seconds
CONNECTION_WRITE_CONNECT_TIMEOUT	The length of time we wait before we become unwritable.	5 seconds
CONNECTION_WRITE_CONNECT_FAILURES	The number of pings that must fail to respond before we become unwritable.	5 pings
CONNECTION_RESPONSE_TIMEOUT	This is the length of time that we wait for a ping response to come back.	5 seconds.

Syntax

```
class Connection : public MessageHandler,
                   public sigslot::has_slots<>
```

The following table describes the protected and public methods of **Connection** and the inherited classes. The methods shown are not thread-safe, and can be called on any thread, but you should call them from the worker thread.

Methods

Connection defines the following public or protected members.

Name	Description
void **CheckTimeout**()	Checks whether this connection is unreadable/writable and should be destroyed. If so, it calls **Destroy**.

~Connection	Destructor.
Connection(Port *_port_, size_t _index_, const Candidate &_candidate_)	Constructor. Call it with the local **Port** object, the index of this connection in the caller's vector of connections, and the address of a remote candidate.
void **Destroy**()	Causes this connection to be deleted.
virtual int **GetError**() = 0	Retrieves the error if **Send** returns a value less than zero.
uint32 **last_ping_sent**()	Returns the time that the last ping was sent (absolute time value).
Candidate& **local_candidate**()	Retrieves the local part of this connection. Contrast this with **remote_candidate**.
void **OnConnectionRequestErrorResponse**(StunMessage *_response_, uint32 _rtt_)	Called when a connection request fails.
void **OnConnectionRequestResponse**(StunMessage *_response_, uint32 _rtt_)	Called when a connection request succeeds.
void **OnMessage**(Message *_pmsg_)	Messaging loop message receiver method.
void **OnReadPacket**(const char *_data_, size_t _size_)	Called when the port reads a packet from the other computer.
void **OnSendStunPacket**(const void *_data_, size_t _size_)	Called when a STUN packet is ready to send.
void **Ping**(uint32 _now_)	Sends a ping across a connection. _now_ should be the current time.
Port* **port**()	Returns a pointer to the **Port** object associated with this connection.
const Port* **port**() const	A const version of **port**.
void **Prune**()	Clears pending messages in this connection. Called when a connection is determined no longer useful to this application. It is not destroyed in case the other side wants to use it, but we can safely stop pinging it, and we can allow it to time out if the other side stops using it as well.
bool **pruned**()	Returns a boolean value indicating whether or not this connection has been pruned.
ReadState **read_state**() const	Returns a **ReadState** enumerated value indicating the current read state. **ReadState** values are listed below. See also **write_state**.
void **ReceivedPing**()	Called whenever a valid ping is received on this connection. This is public because the connection intercepts the first ping.
size_t **recv_bytes_second**()	Read speed of this connection, in bytes per second.
size_t **recv_total_bytes**()	Total number of bytes received over this connection.
const Candidate& **remote_candidate**() const	The remote connection. Contrast this with **local_candidate**.
uint32 **rtt**() const	Estimate of the round-trip time over this connection, in milliseconds.
virtual int **Send**(const void *data, size_t size)=0	Sends a packet of data asynchronously.

size_t **sent_bytes_second**()	The write speed of this connection, in bytes per second.
size_t **sent_total_bytes**()	Total number of bytes sent over this connection.
void **set_connected**(bool *value*)	Sets a value indicating whether or not the connection is writable. If False and the read state is STATE_READ_TIMEOUT, the connection will destroy itself.
void **set_read_state**(ReadState **value**)	Sets the read state of the connection. If the connection is both unwritable and unreadable, it will trigger the destruction of the connection.
set_write_state(WriteState value)	Sets the write state of this connection. f False and the read state is STATE_READ_TIMEOUT, the connection will destroy itself.
void **UpdateState**(uint32 *now*)	Checks the read and write states of this connection. If both are bad, it will trigger destruction of the connection. *now* is the current time, which is compared against various timeouts.
WriteState **write_state**()	Returns the current write state. Possible values are listed below. See also read_state.
TCPConnection(TCPPort* *port*, const Candidate& *candidate*, AsyncTCPSocket* *socket*)	Constructor for **TCPConnection** objects. This is called by **TCPPort**, which passes in itself, a remote candidate address, and a socket that handles the connection.
TCPPort* **TCPPort**()	Returns the **Port** object for a **TCPConnection**.

Enumerations

Name	Description
ReadState	Indicates the readability of this connection. The following values are defined: • STATE_READABLE The application has received pings recently. • STATE_READ_TIMEOUT The application hasn't received pings recently.
WriteState	Indicates the writeability of this connection. The following values are defined: • STATE_WRITABLE The application has received ping responses recently. • STATE_WRITE_CONNECT The application has had a few ping failures. • STATE_WRITE_TIMEOUT The application has had many ping failures.

Signals

SignalStateChange<Connection *>
Sent when the state of the connection has changed.

SignalDestroyed<Connection *>
Sent when the connection has been destroyed.

SignalReadPacket<Connection *, const char *, size_t >
Sent when an incoming packet of data has been received from a **Port** object.

> **Attributes:** public

> **Declaration file:** talk/p2p/base/port.h

Last updated March 23, 2012.