* Overview: Broker / Channel

A channel is a communication channel, a point-to-point stream of bytes.
Full-duplex, each end point can be used to read or write.
A connected channel is FIFO and lossless, see Section "Disconnecting"
for details about disconnection.

** Connecting

A channel is established, in a fully connected state, when a connect
matches an accept. When connecting, the given name is the one of the
remote broker, the given port is the one of an accept on the remote
broker.

There is no precedence between connect and accept, they let listener and
they are called once the connection is established.

When connecting, we must distinguish between two cases:
(i) there is no accept yet and (ii) there is not such broker.
When the named broker does not exist, the connect returns false, else he
wait to be called with his listener.

** Writing

When writing, the given byte array contains the bytes to write from the
given offset and the number of bytes to write is the given length.
The method "write" returns true and will called the listener of the
reader.

Nota Bene: a channel is a stream, so although the write operation does
take a range of bytes to write from an array of bytes, the semantics is
one that writes one byte at a time in the stream.

The method "write" return true and will called the listener of the reader.
If he can't write,it will add the message to a queue and it will be
written when the channel will be able to write.

If the method has been waiting the calling task and the channel becomes
disconnected, the listener will be call and it will throw an exception
(ClosedException).

** Reading

When reading, the given byte array will contain the bytes read,
starting at the given offset, the number of bytes to read is the
given length.

The method "read" returns true and will called the listener of the writer.
If he can't read,it will add the message to a queue and it will be read
when the channel will be able to read.

** Disconnecting

A channel can be disconnected at any time, from either side. So this requires an asynchronous protocol to disconnect a channel.

Note: since we are not considering only one task per end point (no multi-tasking), if there is a task blocked on an operation, the disconnect may only happen from the other side. We will talk about the local side versus remote side.

When the remote side disconnects a channel, there may be still bytes in transit.
By in transit, we mean bytes that were written by that remote side, before it disconnected the channel, and that have not been read on a local side. Therefore, the local side should not be considered disconnected until these bytes have been read.

But if a local channel appears as not disconnected while its far side has been disconnected, then, we send the length of the bytes in transit.