4.4 MCAPI Packet Channels

MCAPI packet channels transfer data packets between a pair of connected endpoints. A connection between two endpoints is established via a two-phase process. First, some node in the system calls `mcapi_pktchan_connect_i()` to define a connection between two endpoints. This function returns immediately. In the second phase, both sender and receiver open their end of the channel by invoking `mcapi_pktchan_send_open_i()` and `mcapi_pktchan_recv_open_i()`, respectively. The connection is synchronized when both the sender and receiver open functions have completed. In order to avoid deadlock situations, the open functions are non-blocking.

This two-phased binding approach has several important benefits. The “connect” call can be made by any node in the system, which allows the programmer to define the entire channel topology in a single piece of code. This code could even be auto-generated by some static connection tool. This makes it easy to change the channel topology without having to modify multiple source files. This approach also allows the sender and receiver to do their work without any knowledge of what remote nodes they are connected to. This allows for better modularity and application scaling.

Packet channels provide a “system-specified buffer” interface. The programmer specifies the address of a buffer of data to be sent on the send side, but the receiver's "recv" function returns a buffer of data at an address chosen by the system. This is different from the “user-specified buffer” interface use by MCAPI messaging – with messages the programmer chooses the buffer in which data is received, and with packet channels the system chooses the buffer.

It is not allowed to send messages to connected endpoints.