determined by the application. This specification does not define a communication byte order because MCAPI is an API specification. Since MCAPI does not currently define a wire protocol (there may be a future Multicore Association specification which does specify that) then endian-ness and alignment issues etc are not relevant to this specification.

Connectionless communication allows an endpoint to send or receive messages with one or more endpoints elsewhere in the communication topology. A given message can be sent to a single endpoint (unicast). Multicast and broadcast messaging modes are not supported by MCAPI, but can be built on top of it.

Connection-oriented communication allows an endpoint to establish a socket-like streaming connection to a peer endpoint elsewhere in the communication topology, and then send data to that peer. A connection is established using an explicit handshake mechanism prior to sending or receiving any application data. Once a connection has been established, it remains active for highly efficient data transfers until it is terminated by one of the sides, or until the communication path between the endpoints is severed (for example, by the failure of the node or link which one of the endpoints is utilizing).

Connection-oriented communication over MCAPI channels is designed to be reliable, in that an application can send data over a connection and assume that the data will be delivered to the specified destination as long as that destination is reachable.

THINGS TO REMEMBER:
- In an MCAPI communication topology where different nodes may be running on different CPU types and/or operating systems, applications must ensure that the internal structure of a message is well-defined, and account for any differences in message content endian-ness, field size and field alignment.

3.5 Data Delivery

On the surface, whether the communications use messages, packet channels or scalar channels, the data delivery in MCAPI is a simple series of steps: a sender creates and sends the data (either a buffer for messages and packet channels, or a scalar value for scalar channels), the MCAPI implementation carries the data to the specified destination, and the receiver receives and then consumes the data. In practice, this is exactly what happens most of the time. However, there are a number of places along the way where things can get complicated, and in these cases it is important for application designers to understand exactly what MCAPI will do.

The sections that follow describe the various steps performed by MCAPI during the transfer of a data packet.

3.5.1 Data Sending

The first step in sending data is to create it. The user application then sends the data using one of several mechanisms. The application supplies the data in a user buffer for messages and packet channels. The data is supplied directly as a scalar variable for scalar channels.

The most common reason MCAPI is unable to send a piece of data is because the sender passes in one or more invalid arguments to the send routine. The term "invalid" refers both to values that are never acceptable under any circumstances (such as specifying a data buffer size exceeding the maximum size allowed for a specific implementation) and to values that are not acceptable for the current sender (such as an invalid endpoint handle, or an unconnected channel).

In all of these cases the send operation will return a failure code indicating that the intended data was not sent. If the data is sent successfully, the send operation returns a success indication. Success means that the entire buffer has been sent.

THINGS TO REMEMBER: