3.12 Timeout/Cancellation Philosophy

The MCAPI API provides timeout functionality for its non-blocking calls through the timeout parameters of the `mcapi_wait()` and `mcapi_wait_any()` functions. For blocking functions implementations can optionally provide timeout capability through the use of endpoint attributes.

3.13 Backpressure mechanism

In many systems, the sender and receiver must coordinate to ensure that the messages are handled without overwhelming the receiver. Without such coordination, the sender must implement elaborate recovery mechanisms in case a message is dropped by the receiver. The overhead to handle these error cases is higher than pausing for some time and continuing later. The sender handles such scenarios by inquiring the receiver status before sending a message. The receiver returns the status based on the number of available buffers or buffer size.

The sender throttles the messages using the `mcapi_endpoint_attribute_get`() API and the `MCAPI_ATTR_NUM_RECV_BUFFERS_AVAILABLE attribute`. If required, the sender and receiver can reserve some buffers for higher priority messages. Further, a more sophisticated mechanism can be build on top of this API. For example, zones (green, yellow and red) can be defined using the number of available messages. The sender can use the zones to throttle the messages.

3.14 MCAPI Implementation Concerns

3.14.1 Link Management

The process for link configuration and link management are not specified within the MCAPI API spec. However, it is important to note that MCAPI does not define APIs or services (such as membership services) related to dynamic link state detection, node or service discovery and node or service state management. This is because MCAPI is designed to address multicore and multiprocessor systems where the number of nodes and their connectivity topology is known when the MCAPI system is configured. Since the MCAPI API specification does not specify any APIs that deal with network interfaces, an MCAPI implementation is free to manage links underneath the interface as it sees fit. In particular, an MCAPI implementation is able to restrict the topologies supported or the number of links between nodes in an MCAPI communication topology.

3.14.2 Thread Safe Implementations

MCAPI implementations are assumed to be reentrant (thread safe). Essentially, if an MCAPI implementation is available in a threaded environment, then it needs to be thread safe. MCAPI implementations can also be available in non-threaded environments, but the provider of such implementations will need to clearly indicate that the implementation is not thread safe.

3.15 MCAPI Potential Future Extensions

With the goals of implementing MCAPI efficiently, the APIs are kept simple with potential for adding more functionality on top of MCAPI later. Some specific areas include additional zero copy functionality, multicast, and informational functions for debugging, statistics (optimization) and status. These areas are strong candidates for future extensions and they are briefly described in the following subsections.