kucomms Reference Manual

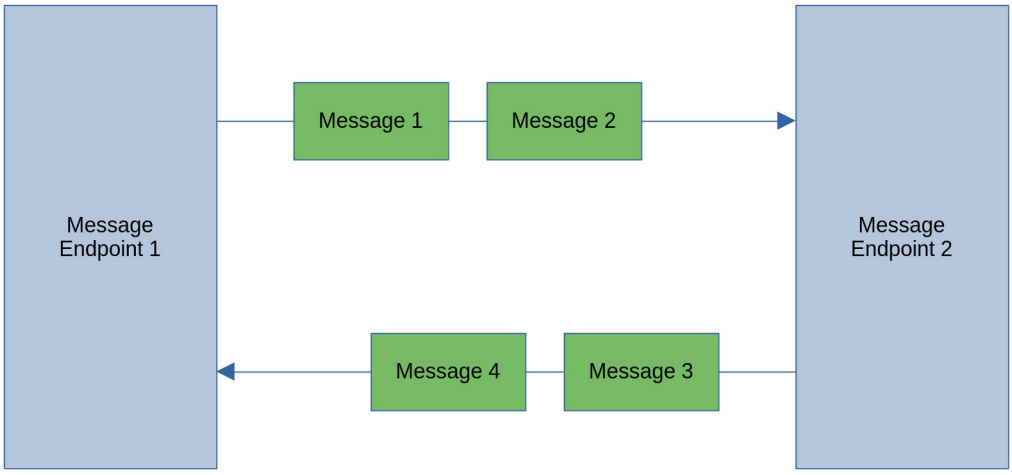
Table of Contents

Overview

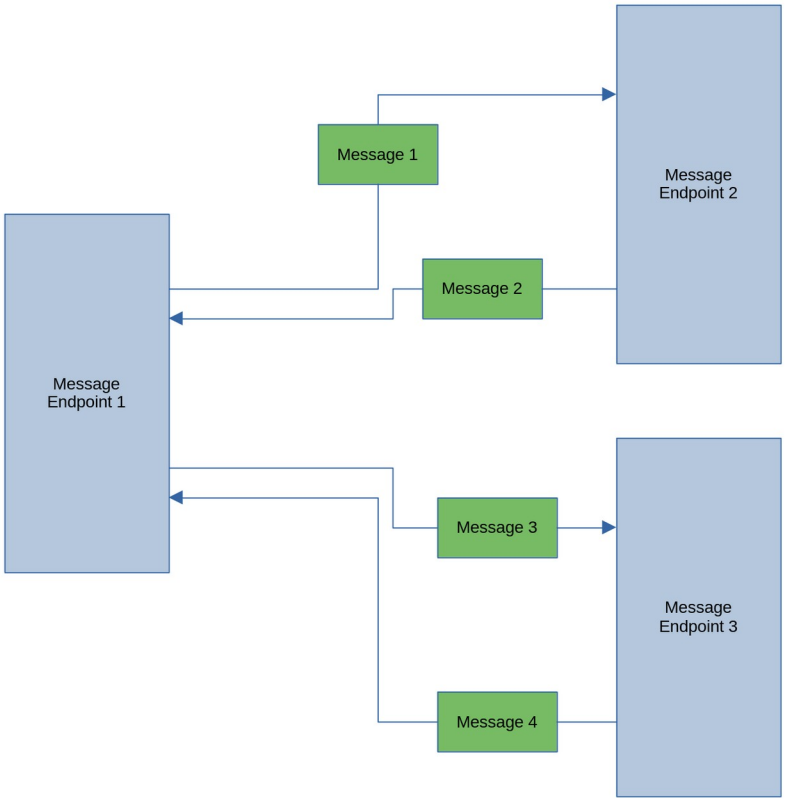
Asynchronous message passing architecture

The kucomms project is built on top of a asynchronous message passing architecture.

A message endpoint is an entity that can send messages and receive messages. Two message endpoints can communicate with each other by sending messages as shown in the diagram below.

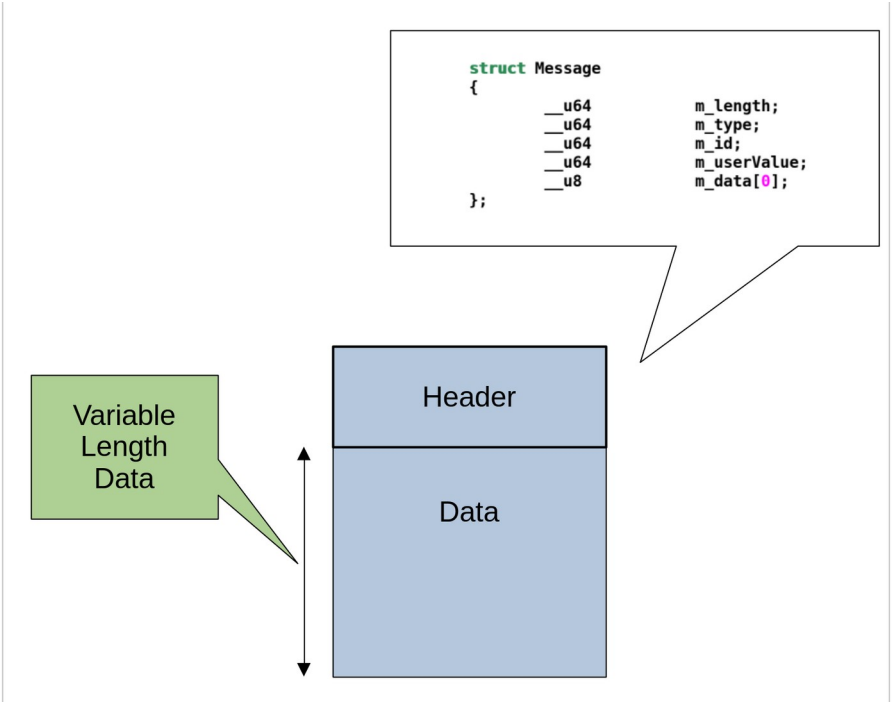


The kucomms project uses the one to one model shown in the diagram above but the messaging framework supports one endpoint connected to multiple endpoints as shown in the diagram below.



Definition of the message

The messages that are sent between endpoints are variable length datagrams defined by the user. The messages have a header and a data section. The layout of a message is shown below.



The fields in the header can be used the by the user in any way as there is no reserved values for the header fields. The user may use the variable length data section in any way. The generic nature of the message allows the two endpoints to define their own protocol for communication which includes a command/response protocol or a notification protocol or any combination of those protocols.

The message endpoint

Each message endpoint provides two important functions. One of the functions is that it provides a C and C++ API that is used to send a message. The other function is that a message endpoint has an internal thread which is responsible for receiving messages and then calling a user callback when a message arrives. The user of a message endpoint has to register a callback which will be called when a message arrives. When a user callback function is called, the message received will be provided to the callback function.

Message endpoints communicate using shared memory

Message endpoints in userspace and kernel

The message endpoint scheduler

The user callbacks

The scheduler

The programmers model for using a message endpoint