libgcp Programmer's Guide

Jonathan Lamothe

May 19, 2012

1 Introduction

GCP (the Generic Communications Protocol) is intended to send messages of an arbitrary number of octets over a network in a simple, open manner. While it can be used on may different types of networks, it was specifically designed for serial networks, such as RS232, RS485, or other networks designed to send a stream of octets. The protocol provides its own error detection, making such a service unnecessary at a lower layer.

This protocol does not implicitly provide addressing, message acknowledgement, or streaming services, although, such services can be implemented on top of it. The protocol also places no restrictions on the content or format of its messages' payloads.

libgcp does not actually *send* or *receive* data. It merely assembles and processes frames. It is up to the programmer to send and receive the frames to and from what ever data stream they're using for transport.

Note: This document is intended to give an overview of how to use libgcp. For a more detailed descripiton on the various functions and structures, please refer to the reference manual.

2 The `GCPConn` Structure

Before any data can be processed, a `GCPConn` object needs to be created and initialized. The object is initialized by passing a pointer to it to the `gcp_init()` function, such as in the following example:

```
#include <gcp.h>

int main()
{
    GCPConn conn;
    gcp_init(&conn);

    /* do stuff here */
```

```
    return 0;
}
```

2.1 Allocating Buffers

After the connection has been initialized, send and receive buffers need to be allocated. These buffers contain the payload data for an outgoing or incomming message. If a connection is going to be one way (i.e: send or receive only) only one buffer needs to be allocated. **It is up to the programmer to free these buffers when they are no longer required.**

A buffer is simply an array of type `uint8_t`. These buffers are then pointed to by the `recv_buf` and `send_buf` fields of the `GCPConn` object. Also, the size of the receive buffer needs to be set in the `recv_size` field. The `send_size` value is set to the size of the message to be sent, rather than the size of the entire buffer. This is typically done just before sending. An example follows:

```
#include <gcp.h>

#define SIZE 1024

GCPConn conn;
uint8_t send[SIZE], recv[SIZE];

int main()
{
    gcp_init(&conn);
    conn.send_buf = send;
    conn.recv_buf = recv;
    conn.recv_size = SIZE;

    /* do stuff here */

    return 0;
}
```

2.2 The `send_lock` and `recv_lock` Flags