Aarhus university

Computer-Science

Distributed Systems and Security

Handin 0

Author	*Student number*
Søren M. Damsgaard	**202309814**
Alexander	**202408929**
Thor behrmann	**202410271**

February 12, 2026

Messages with Framing and Serialization

As explained in the assignment, TCP is a byte stream that needs encoding and a helping hand.
The encoding we used is the one recommended, the standard ENCODING/JSON library, which was used to marshal the Message struct into JSON byte slices. When recieved it is then unmarshaled back into a struct. We chose JSON because it was recommended in the assignment and because of our previous experience with Java and Javascript.

For the framing of the messages we used a lenght prefix solution, because TCP might mess op the boundary/size of the message sent, or the amount of bytes read. The way it works is that we prefix every message with a 4-byte header holding the lenght of the message in the form of a basic 32-bit usigned integer. We specify and ue 32-bit for consistency and since any more would be 'overkill'.

When a Peer receives a messages it gets the prefix first by reading the 4 bytes to ascertain the size of the 'payload' and with the help of 'io.ReadFull' to read exactly the size of the payload based on the prefix/header..

How to Run the Code

Move into the *./Handin_1* directory, should be the one sent through brightspace, with all the files in the same folder, main, test and the module.
To run the code you need to 'build' and then 'run' the code/file. Use the following commands in the terminal while you're in the mentionen directory.

```
go build ./handin0_main.go
go run ./handin0_main.go
```

This should run the main function in the code, which currently creates 3 Peers and connects them all, while sending a pair of test Messages.

How to Run Tests

To run the automatic test-cases the only command needed is:

```
go test -v
```

This should run the test file and all its cases and announce wether any tests failed.

Appendix

Clearly a misguided appendixed left redundant after its creator realized that the code was not that long and could be easily included in the main text without a kerfuffle.