CSE 109 – Systems Software

Programming Lab 4

Due: 5/3/2019

Networking Lab

Background

Networking is when two or more computers communicate over a wire, either physical or wireless. This communication takes place as an exchange of bit streams. Bits from one computer are sent to another, which causes the receiver to take some action. In order to interpret the streams of bits, a protocol has to be established that governs the exchange of information. For example, consider the protocol of a human conversation:

1. The intent to start a conversation is established with a greeting.
2. A greeting is said in response.
3. The purpose and context of the conversation is established.
4. The conversation commences as an exchange of statements.
5. Further information can be obtained from one of the parties of the conversation by asking a question.
6. A response to a question can be to ask for clarification, to give an answer, or to express uncertainty e.g. “I don’t know the answer to the question.”
7. The end of the conversation is indicated by a farewell salutation.

We are all familiar with how a conversation between humans proceeds, but now we need to become familiar with how a conversation between computers works. In this project, we will create a chat program between two computers. The program will communicate via messages of the form:

```
struct Message {
        int timestamp;
        string username;
        string message;
}
```

In order to send this message over a wire, it first needs to be serialized, the process of turning a structured object into a form suitable for transmission. To serialize this object, we will need to implement several functions:

```
vector<unsigned char> serialize_int(int data);
vector<unsigned char> serialize_string(string data);
vector<unsigned char> serialize_int_field(string name, int data);
vector<unsigned char> serialize_string_field(string name, string data);
vector<unsigned char> serialize_message(Message msg);
```

The process of serialization will be to call serialize_message(), which will call serialzed_string_field() and serialize_int_field(), which will in turn call serialize_int() and serialize_string(). This process will turn a Message type into a byte stream.

We also need to convert a byte stream back into a Message. We do this by implementing the following functions:

```
int deserialize_int(vector<unsigned char> stream);
string deserialize_string(vector<unsigned char> stream);
Message deserialize_message(vector<unsigned char> stream);
```

Deliverables

You will write a conversation over a chat system between two people that follows the above 7-step protocol. Note that the conversation isn't necessarily 7 statements long, as several of the stages can be multiple messages. This conversa should be at least 10 messages long.

Encode your conversation in a program and demonstrate that each message can be serialized and deserialized.

Your program should be in the standard project structure we have been using:

- include – your serialize and deserialize functions
- src – where your main.cpp is. This is where you will write your conversation and serialize/deserialize it.
- Makefile – builds your program
- readme.md – your write up

In your readme, answer the following questions

- What is the length of the int message in bytes
- What is the length of the average serialized message in your conversation, in bytes
- Choose a single message in your conversation and compress it (by hand) using recursive byte pair encoding. Indicate the compression ratio (compressed bytes / original bytes)
- With the required decompression lookup table, is the compressed version a net gain?