INGI2141 Project 1: reliable transfer protocol

1 Statement

1.1 Description

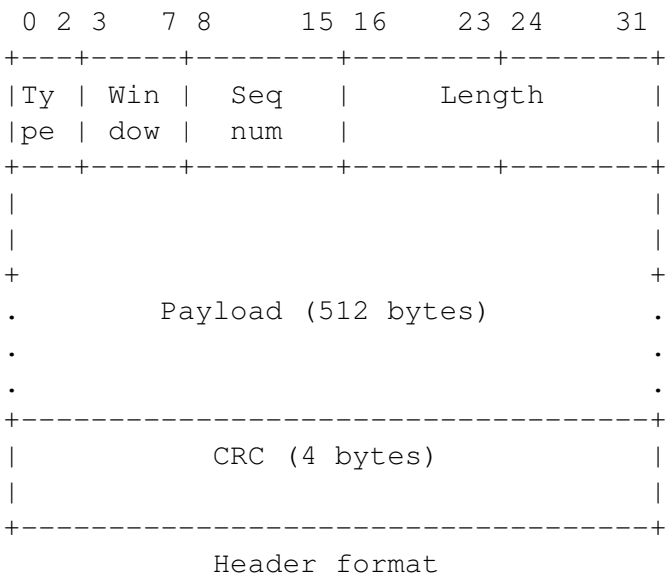
For this first project you have to implement in C a reliable transmission protocol on top of UDP, using IPv6. This protocol is based on **selective repeat**. You have to implement a sender and a receiver to reliably transmit a file of any length (we consider that the transmission is one-way and the receiver will only transmit acknowledgments). The packets have a fixed-length payload of 512 bytes.

1.2 Program usage

The sender program must take the following arguments: `./sender [--file filename] [--sber x] [--splr x] [--delay d] hostname port`. It will send the file `filename` to `hostname` (that can be a name or an IPv6 address) on port `port`. If the file name is omitted, then the program must read the file from the standard input. The option `--sber` takes an integer argument and specifies the byte transmission error ratio in per-thousand. For example, a value of `24` means that, on average, 24 bytes out of 1000 are corrupted. Please note that this corruption must be applied to the entire packet, **after** the computation of the CRC (see below for more details). The option `--splr` takes an integer argument and specifies the packet loss ratio, in percentage. For example, a value of `3` mean that, on average, 3 packets out of 100 are lost. The option `--delay` takes an integer argument and specifies the delay to apply before transmitting each packet, in milliseconds.

The receiver must take the following arguments: `./receiver [--file filename] hostname port`. It listens on the host/IPv6 `hostname` (use `::` to listen on all interfaces) on port `port` and writes the data into file `filename`. If the file name is omitted, then the program must write the data on the standard output.

1.3 Format



Header format

- The `type` is encoded in 3 bits. It can take the following values: `PTYPE_DATA = 1` for a data packet, or `PTYPE_ACK = 2` for an acknowledgment packet. Any packet with a different `type` value MUST be dropped.
- The `window` is encoded in 5 bits, leaving a maximum window size of 31 packets. It states the receiving window of the segment originator inside ACK packets. As the sender is not supposed to receive any DATA packet, all segments (i.e. DATA segments) sent by the sender SHOULD have the `window` set to 0. Of course, the receiving window of the receiver may vary during the transmission.
- The `sequence number` is encoded in 8 bits. For DATA packets, it states the sequence number of the packet. The first DATA packet of a connection has sequence number 0. For ACK packets, this field states the next sequence number that is expected (i.e. all packets whose seq num is lower are acknowledged).
- The `length` is encoded in 16 bits and specifies the useful length of the payload. If the `length` of a packet is higher than 512, then the packet SHOULD be dropped. A value lower than 512 means that the packet is the last packet of the connection and the receiver should not expect to receive further packets.
- The `payload` is a block of 512 bytes. If less than 512 bytes of user-data need to be transmitted, then the payload MUST be padded with nul bytes.
- The field `CRC` is the CRC32 of the header and the payload. If the value does not correspond to the actual CRC32 of the header+payload, then the packet

MUST be dropped. The CRC is applied to the packet that is about to be sent, or that is just received (i.e. in network-byte order). The divisor (polynomial) of the CRC32 is $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$, and the normal (MSB-first) representation of this polynomial is `0x04C11DB7`.

2 Interoperability test

About one week before the submission deadline, you will have to form groups of 3 groups so that you can test your implementation with two other implementations. You will most likely find some issues during this test and you will have to determine whether the issues come from your code or from the code of another group. Use this information to fix your implementation if necessary. You need to describe in the report how useful was the interoperability test and which bugs were discovered during this test.

3 Deliverables

Submission deadline: **31st of October 2014**

- A report in PDF format (4 pages maximum) explaining your implementations choices. You must also state the limitations of your implementation, and notably the main factors limiting the performance (throughput). Do not forget to explain the results of the interoperability test your performed.
- The source code of your implementation. The code **MUST** compile on the Linux hosts in the Intel room of the Réaumur building. You **MUST** include a Makefile whose default target will produce two executables named `sender` and `receiver`. If you do not provide a valid Makefile, your code will not be graded.

Additional information regarding the submission format will be announced soon.