Computer Networks

Instructor: Shashi Prabh

Lab 8: Reliable data transfer over UDP

Due: March 25

1 Objective

In the previous lab, you saw that file transfers over UDP can result in loss of data. In this lab, you will learn a way to achieve reliability when the lower layers do not provide it. Recall that sliding window ARQ provides reliability (you should revise the relevnt section of the textbook before starting the work). In this lab, you will extend your code from the previous lab in two ways, namely, the client and server use sliding window protocol, and your reliable UDP file transfer application conforms to specified message structure which is described below. You can use the sliding window code (Protocol 5 or 6) in Chapter 3 of the textbook or section 2.5 of Peterson and Davie's book.

2 Client-server messages

2.1 Client to server messages

```
File_request:
uint8_t type = 0;
uint8_t filename_size;
char    filename[ filename_size ];

ACK:
uint8_t  type = 1;
uint8_t  num_sequences;
uint16_t sequence_no[ num_sequences ];
```

2.2 Server to client messages

```
File_info_and_data:
uint8_t  type = 2;
uint16_t sequence_number;
uint8_t  filename_size;
char     filename[ filename_size ];
uint32_t file_size;
uint16_t block_size;
char     data[block_size];

Data:
uint8_t  type = 3;
uint16_t sequence_number;
uint16_t block_size;
char     data[block_size];
```

```
File_not_found:
uint8_t type = 4;
uint8_t filename_size;
char    filename[ filename_size ];
```

After creating socket, the client sends `File_request` message. If the file is found, the server sends `File_info_and_data` message which confirms the filename, informs the file size and sends the first block of data. Subsequent blocks are sent using `Data` messages. If file is not found, it sends `File_not_found` message. The client sends `ACK`s for each `File_info_and_data` and `Data` messages. If you are implementing positive acknowledgments, `num_sequences` will always be 1. For Selective Acknowledgment, `num_sequences` must be set appropriately. Make sure to convert integers larger than 1 byte to network byte order (`htons()` and `htonl()`) before transmitting and back to host byte order on the other side (`ntohs()` and `ntohl()`).

3 Performance

1. Evaluate the correctness of your implementation: send a few large files from a server to a client preferably connected over a wireless link. Compare the size of sent and received files.

2. Vary `SWS` (keeping `RWS = SWS`) and plot the measured data rate against `SWS`.

4 Extra credit

Implement Selective Acknowledgment.

5 Partial credit

Teams that could not finish the implementation of sliding window can submit an implementation of stop-and-wait along-with performance evaluation for 50% partial credit.

6 Submission

Submit your client and server codes, a makefile, and a report on your performance experiments by midnight on the due date. Code must be adequately explained in comments. Both the server and the client codes must print debug messages on terminal related to all the sent, received and retransmitted (upon time-out) messages. The report must list all team members' names.