<div align="center">

Computer Networks, Fall 2016

Instructor: Shashi Prabh

</div>

Lab 3: Reliable data transfer over UDP

Due: Sep. 26 (Group A), Sep. 28 (Group B)

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 (revising section 2.5 of the textbook at this time is a good idea). 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 of section 2.5.

2   Client-server messages - 70%

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 - 30%

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 - 10%

Implement Selective Acknowledgment.

5 Submission

Submit your client and server codes, a makefile, and a report on your performance experiments on blackboard by 4 PM on the respective due dates. The report must list all team members' names. Demo within one week from the due date is necessary to get credit.