CS 571 Programming Project

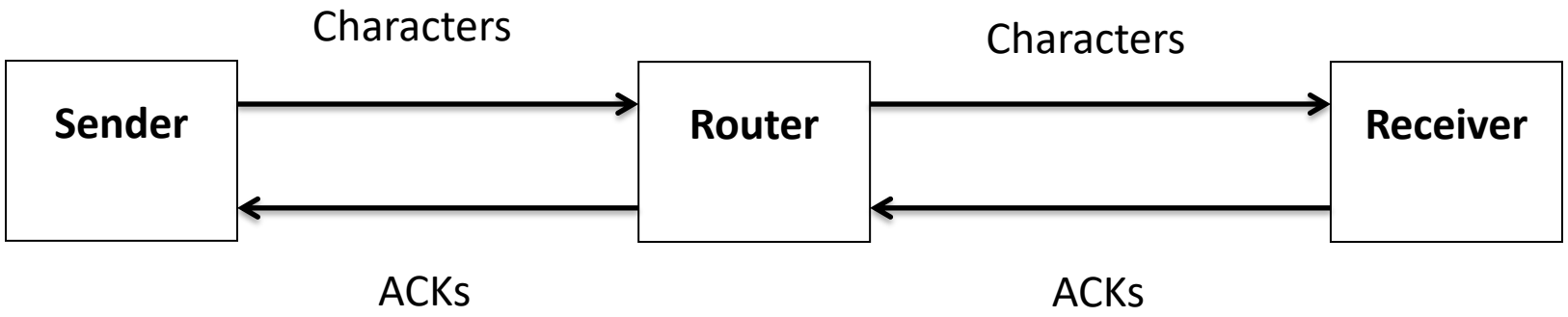
"Due" Wednesday December 13

CS 571 Programming Project

In this assignment you will implement a simple file transfer protocol using UDP and a stop-and-wait protocol.

You will write three programs: the sender, the router and the receiver. The sender expects 5 command line arguments: the sender port, the router IP, the router port, the name of an input text file and a timeout value in milliseconds. The router expects 6 command line arguments: the router port, the sender IP, the sender port, the receiver IP, the receiver port, and a probability that a packet is discarded. The router is simulating an unreliable network. The receiver expects 4 command line arguments: the receiver port, the router IP, router port and the name of the output file.

CS 571 Programming Project



CS 571 Programming Project/Sender

The sender program sends the input file contents to the receiver indirectly by sending it through the router. The sender sends chunks of 100 characters (the last chunk could be less than 100 characters) and waits for an acknowledgement. If an acknowledge does not arrive in time, the sender resends the characters otherwise the sender sends the next 100 characters. This process is continued until all the characters in the file have been sent and acknowledged.

CS 571 Programming Project/Router

The router program is simulating an unreliable network. It receives messages from the sender and receiver and either discards each message or forwards it. The probability of discarding a message is given as a command line argument. The value in the command line argument will be an integer between 0 and 1000. Each unit represents a 0.1% probability of message loss. So a value 1 means 0.1 % of the messages are discarded. A value of 100 means 10% of the messages are discarded, etc. The router knows the IP and port of both the sender and the receiver so a message received from the sender should be forwarded to the receiver and a message received for the receiver should be forwarded to the sender (this would be an ACK). The router program does not examine the content of the messages.

CS 571 Programming Project/Receiver

The receiver receives chunks of 100 bytes and if the bytes are the expected bytes it writes the bytes to the output file and sends an acknowledgement. If the bytes are not the expected bytes they are discarded. Since this is a stop-and-wait protocol the only unexpected bytes would be duplicates so the receiver should resend the last acknowledgment when unexpected bytes are received.

CS 571 Programming Project Sequence Numbers, Timeouts and Program Termination

Your implementation will have to implement a 1 bit sequence number and a message format for the file contents and the acknowledgements. The sequence number is not included in the 100 bytes of data so a single UDP sendto will use more than 100 bytes.

Your application should also decide how the sender tells the receiver that the file transfer is complete and what the receiver should do after the last ACK is sent

Since you are using UDP you will use sendto and recvfrom to send and receive messages. On the sender side when the program executes a recvfrom to receive the acknowledgement it should only block up to timeout amount of seconds. This can be done by setting a socket option.

CS 571 Programming Project/Timeout

```
struct timeval x;
 x.tv_sec = 0;
 x.tv_usec = 200000;
 setsockopt(sockfd, SOL_SOCKET, SO_RCVTIMEO,&x,sizeof(struct timeval));
 len = recvfrom(sockfd, msg, 100, 0, (struct sockaddr *) &client, &clen);
 if (len == -1 && errno == EWOULDBLOCK) {
    printf("timeout occurred\n");
    exit(1);
 }
```

UDP Echo Client With Timeout

```
clen = sizeof(server);
len = strlen(argv[1]);
sendto(sockfd, argv[1], len, 0, (struct sockaddr *) &server, clen);
struct timeval x;
x.tv_sec = 0;
x.tv_usec = 200000;
setsockopt(sockfd, SOL_SOCKET, SO_RCVTIMEO,&x,sizeof(struct timeval));
ret = recvfrom(sockfd, msg, 100, 0, NULL, NULL);
 while (ret == -1 && errno == EWOULDBLOCK) {
      printf("timeout occurred\n");
      sendto(sockfd, argv[1], len, 0, (struct sockaddr *) &server, clen);
      ret = recvfrom(sockfd, msg, 100, 0, NULL, NULL);
  }
```

Submission

You will demonstrate your program to me. After the demo you should email me one zip file that contains you source code files.