

Assignment 1

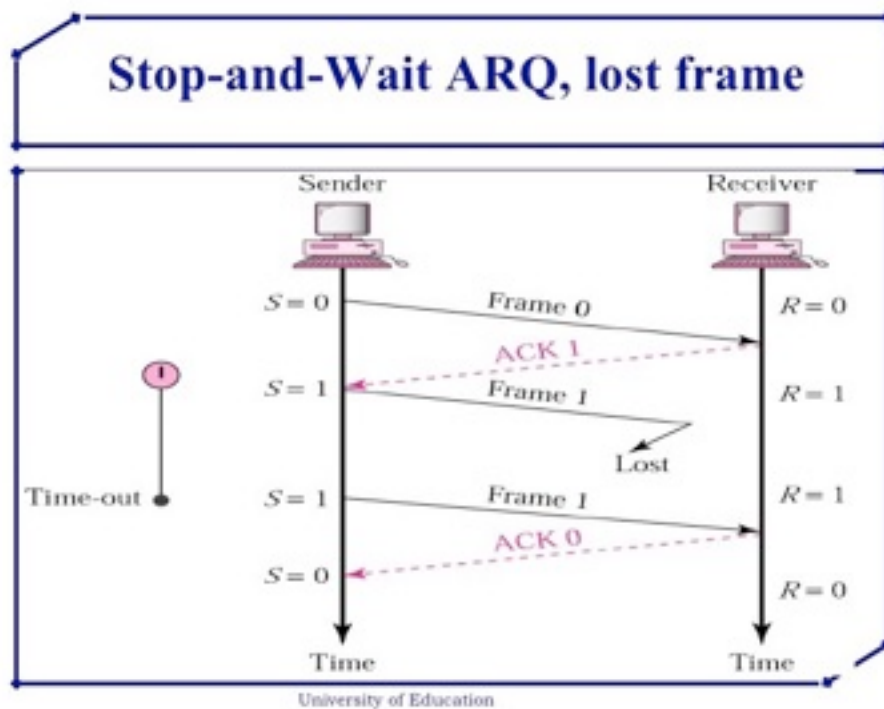
17 November 2015

1 Introduction

This report illustrates the approach behind the two programs which have been implemented for this assignment. The report consist of implementation details for: (1) Stop& Wait protocol and (2) Selective repeat protocol. This report will show the correct implantation of actual protocols in telecommunications, and how were they implemented in the program. It also shows some error states that might occur. At the end, program testing is been provided.

2 Stop & Wait protocol

At this section, the implementation of protocol is described. The following diagram illustrates the protocol approach:



At the sender side, program receives a string to send from terminal. The sender will send one character at the time associated with frame number. For example, string entered is "Hello". The program will send H0, e1, l0, l1 and o0. At the start of sending procedure, sender will send a character at time(packet), and will stop and wait till acknowledgement is been sent from the receiver confirms receiving a certain packet, and indicating what is it expecting next. For example, sender sends H0, and receiver will send ACK1, which indicates it received H0, and expecting the next frame sequence to be 1. At any time of the program when a packed is dropped or went missing, receiver will not receive any packet. Therefore, it will not send any acknowledgement. To over come this error state, sender will start a timer when sends any packets. If the timer times out before receiving any

acknowledgement back from receiver, then sender will re-send a certain packet that is not been acknowledged by receiver.

The actual implementation of this protocol works in the same way. For simplicity and testing the code, the receiver side will receive the odd number character of the total characters to be send, and will drop the even. For example, if sending "hello", receiver will receive h, drop e, receive l, and so on The following pseudo code illustrates the implementation:

```
int sequenceNumber = 0;
String str = getString();
for(int i=0; i< str.length; i++){
    send(str.charAt(i) + sequenceNumber);
    startTimer();
    if(receiveAck == true){
        // make sequence number to be the opposite
        sequenceNumber = sequenceNumber+1%2;
        stopTimer();
    }else if(TimerTimesOut == true){
        send(str.charAt(i) + sequenceNumber);
        startTimer()
    }
}
```

Sender side

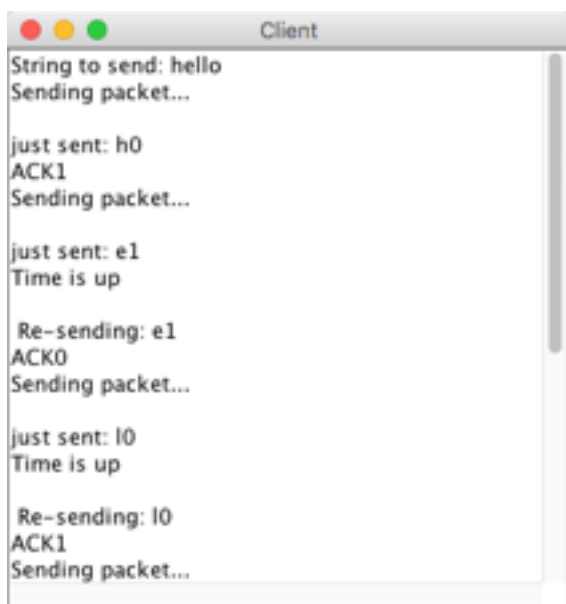
```
while(receive()){
    // extract sequence number, and
    // convert it the opposite
    // because its one expected next

    int sequenceNumber = str.charAt(1);
    sequenceNumber =
    sequenceNumber+1%2;
    send(ACK + sequenceNumber);
}
```

Receiver side

2.1 Testing Result

Due to some unexpected encoding problems (because the program had been implemented on MAC platform) the execution on the code provided might be slightly different on Windows platform, so here is a screenshot of the output of the program:



```
Client
String to send: hello
Sending packet...

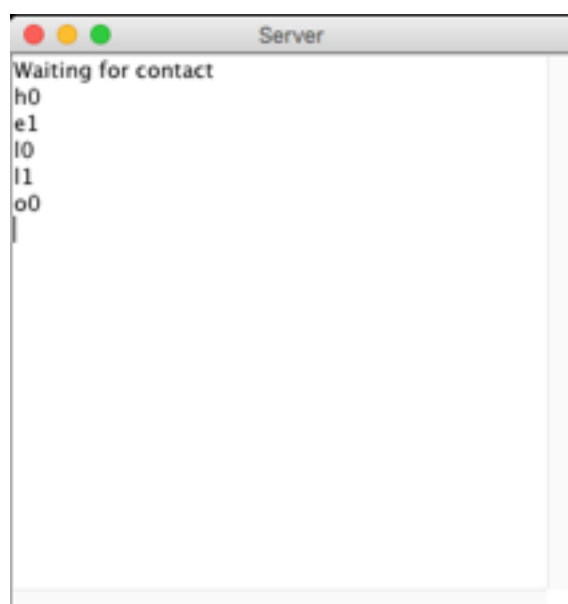
just sent: h0
ACK1
Sending packet...

just sent: e1
Time is up

Re-sending: e1
ACK0
Sending packet...

just sent: l0
Time is up

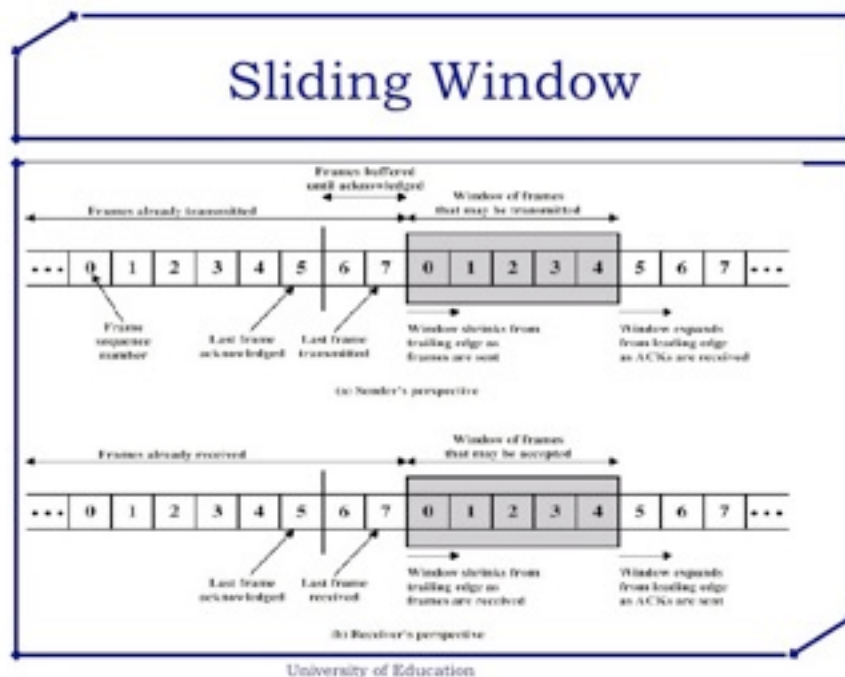
Re-sending: l0
ACK1
Sending packet...
```



```
Server
Waiting for contact
h0
e1
l0
l1
o0
|
```

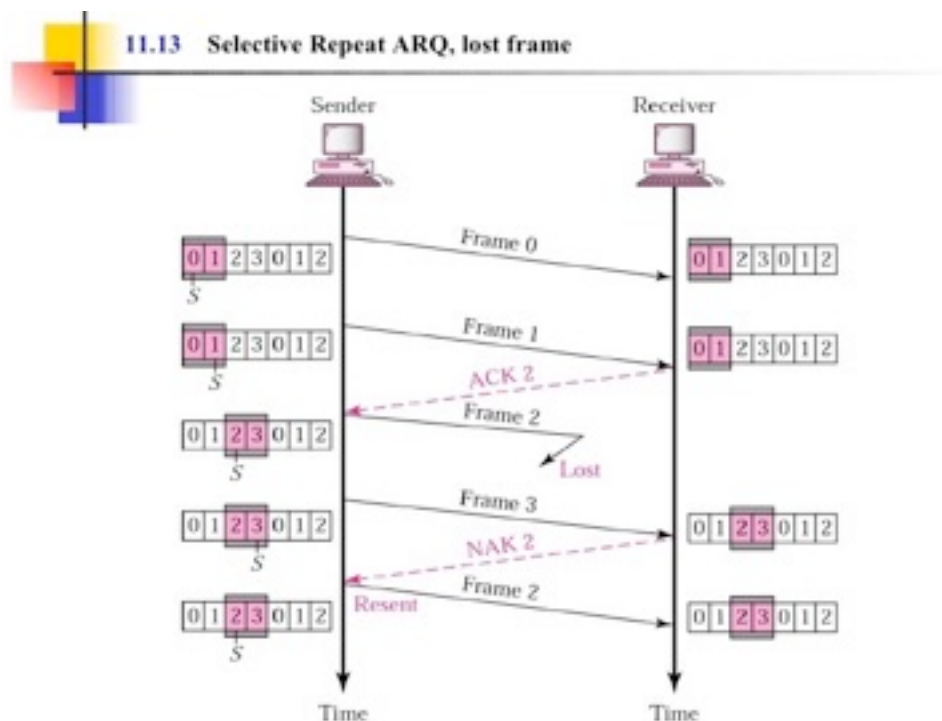
3 Selective repeat protocol

At this section, the implementation of selective repeat protocol is described. This protocol is using a sliding window approach.



The sliding window allows to program to take a string of n number of characters, then it sends a part of the string. For example, if sending "hello", and sliding window is 2 then program will send h & e one after the other without waiting for acknowledgements or negative acknowledgements. This methodology will allow a multiple packets to be transmit.

The following diagram illustrates the protocol approach:



Selective repeat protocol sends a sliding window of size n , and then waits for either acknowledgements or negative acknowledgements of what it is expecting to be sent. If a packet goes missing, the receiver will send negative acknowledgements of what it is still expecting to be received. On the other hand, if an acknowledgement has been dropped, the timer at the senders side will expire, and re-send the certain packet that has not been acknowledged.

The actual implementation of the protocol in the program provided does the same, but it has some errors, and works differently than how the protocol works in real world. First error state, the receiver sends acknowledgments for every packets received within the sliding window. Meanwhile, in real world, receiver will receive few packets and sent only one acknowledgment back to the sender. Secondly, the receiver sends acknowledgments or negative acknowledgments of what it has received, in real world, sender sends acknowledgments of what it is expecting next. Finally, the programs only deals with the error case where packets went missing, but it does not deal with the case if acknowledgments went missing. To sum up, the program does not simulate the protocol in real world, but it goes close to the actual protocol. The following pseudo code illustrates the implementation:

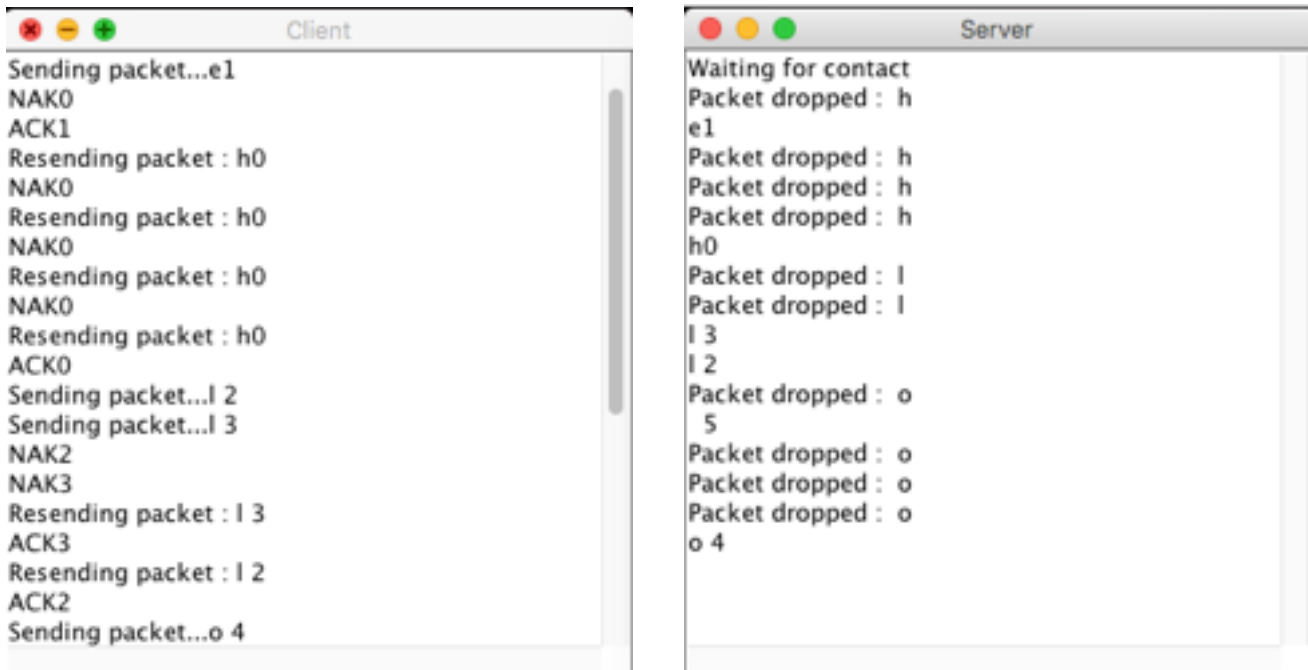
```
int sequenceNumber =0;
String str = getString();
for(int i=0; i< str.length; i++){
    for(int j=0; j<2; j++){
        send(str.charAt(i) +
sequenceNumber);
        if(receiveAck == true){
            if(receive == NAK){
                resend();
            }
        }
    }
}
```

```
while(receive()){
    // extract sequence number, and convert it
    send(ACK + sequenceNumber);
}

// if not receiving, then send NAK
send(NAK + sequenceNumber);
```

3.1 Testing Result

Due to some unexpected encoding problems (because the program had been implemented on MAC platform) the execution on the code provided might be slightly different on Windows platform, so here is a screenshot of the output of the program:



```
Client
Sending packet...e1
NAK0
ACK1
Resending packet : h0
NAK0
Resending packet : h0
NAK0
Resending packet : h0
NAK0
Resending packet : h0
ACK0
Sending packet...l 2
Sending packet...l 3
NAK2
NAK3
Resending packet : l 3
ACK3
Resending packet : l 2
ACK2
Sending packet...o 4

Server
Waiting for contact
Packet dropped : h
e1
Packet dropped : h
Packet dropped : h
Packet dropped : h
h0
Packet dropped : l
Packet dropped : l
l 3
l 2
Packet dropped : o
5
Packet dropped : o
Packet dropped : o
Packet dropped : o
o 4
```

4 Conclusion

To conclude, both protocols have been used at some stage in telecommunications. It is important to know old protocols and how they work. If requested to improve the program in the future, making protocols work as in the real world would be a good implementation, especially selective repeat protocol, and also would implement header & payload approach.