# **Project 2 - RDT 3.0**

# Jacob Leiner

# COP 5518 Computing Essentials

# Date of Project Submission: 8/2/2020

# **Background**

The purpose of this project is to create a sender (client) and receiver (server) program that send a chunked message through a simulated network program. Both the sender and receiver programs use RDT 3.0 to ensure the integrity of the data being transmitted. The message being sent consists of several parts. A string is created that contains 16 bytes for the source IP, 6 bytes for the source port, 16 bytes for the dest IP address, and 6 bytes for the dest port. The message also contains 10 bytes at the end to represent a simulated transport layer. In the 10 byte transport layer, 1 byte is reserved for an error flag, another is reserved for the seq flag, and a third is reserved for an ACK flag. The network program receives a packet from either the sender or the receiver, parses the message, performs some manipulations to the packet based on the command line arguments it received upon executing, and forwards the packet to the user.

The RDT 3.0 protocol consists of two states, a sender and a receiver. The RDT 3.0 protocol is a stop and wait protocol, meaning that the sender will wait for am ACK response or timeout to occur before retransmitting a packet or transmitting the next packet to be sent. This is accomplished by checking three flags: the error flag, the seq flag, and the ACK flag. If the receiver receives a packet with an error flag or a sequence number that equals the sequence number of the last packet that it sent an ACK for, the packet is dropped. The sender will have a timeout occur and will resend the packet it just sent. The receiver will not send the next packet until it receives a packet with an error flag set to 0, a seq number equal to the packet that it just sent, and an ACK flag in the form of 1. These flags alternate their values between 0 and 1, which causes this protocol to be know as the Alternating-Bit Protocol sometimes.

# **Experiment**

I was unable to get my program to work correctly in time. However, the experiment would have been to select an fixed message resulting in a consistent message length and then test different error, delayed, and lost percentages for the network programs arguments, and to run the program 10 times. The experiment would test the effectiveness of this protocol in ensuring the integrity of data being transmitted using the UDP protocol when faced with errors occurring in a network.

# **Results**

As I did not complete the project in time, I unfortunately did not achieve and log results of this experiment. Using a delay of 1,000 milliseconds within the network program, I was able to achieve and average delay of roughly 1,000 milliseconds when the delay, error, and loss parameters of the network program were set to 0. When the delay and lost percent parameters were increased to 0, the program eventually became stuck on a particular sequence number.

# **Conclusions**

RDT 3.0 is a flawed protocol because it is a stop-and-wait protocol. This causes delays in performance to occur. Delays in packets being sent may result in duplicate packets being received if a receiver program’s last received sequence number is different than the delayed packets sequence number. Lost or delayed packets can result in sequence numbers getting switched up at either the Sender or the receiver programs. This can cause problems and potentially result in garbled data being transmitted from sender to receiver. This can be solved by improving the protocol with additional enhancements such as pipelined data, selective repeat, and go-back-n protocols.