ECEN424 Fundamentals of Networking MP 4 - Due Date: See lecture01

Reliable transfer using UDP Sockets

Created by: Dr. Srinivas Shakkottai *Scribe: Self*

1 Introduction

The purpose of this machine problem is to familiarize you reliable data transfer. You will obtain, compile and run a stop-and-wait protocol using UDP sockets.

2 Problem statement

In this assignment, implement a Talker and a Listener using Java sockets. The Talker and Listener must use a reliable UDP connection to communicate with each other.

Talker accepts a string consisting of up to 50 characters that should be entered from command line. The entered string is to be broken up into 5 messages of 10 characters each. Talker must frame these messages with unique sequence numbers $(1, 2, 3, ...)$. Additionally, Talker must create message 0 that contains the number of messages to be sent.

Talker must wait for a Listener to request a UDP connection, and accept the connection. Talker should then request a UDP connection to Listener in order to receive ACK messages. Note that in order for this to happen, Listener should listen for a Talker as soon as it issues the connect request to the Talker.

- You must allow the ports for Talker and Listener to be entered from command line.
- You must allow Talker's IP address to be entered from command line into Listener.
- Talker must extract the Listener's IP address and use it to request a UDP connection for the ACKs.

Talker should send each message (starting with message 0) using UDP to Listener. Listener must extract the number of messages to be sent by Talker from message 0. Listener must ACK each message to Talker, with the ACK containing the sequence number of the next message to be sent. Talker may send a new message to listener only if Listener has ACKd the previous message. If Listener does not ACK a message for 2 seconds, Talker should timeout and resend the message. All the message IDs must be shown on the console to see the sequence numbers. After receiving all messages, Listener must close the connection, concatenate the messages, and display the string.

Next, change Listener as follows. Using a random number generator, randomly do not ACK packets with a probability 0.5. Thus, the Listener will not ACK half the packets on average. This will be a way of testing whether your code for timeout and retransmission is working correctly.

3 Evaluation Guidelines

1. Your Talker and Listener will be evaluated with each other when every message is ACKed.
2. Your Talker and Listener will be evaluated with each other when the Listener only ACKs half the packets.
3. Enter arbitrary strings as input to the talker/listener to test for error handling.

References

[1] http://docs.oracle.com/javase/tutorial/networking/sockets/index.html