Nathan Cauwet
DU ID: 873271826
October 4th 2022

## Problem 1 – UDP Demultiplexing

*Suppose a process in host C has a UDP socket with port number 787. Suppose host A and host B each send a UDP segment to host C with destination port number 787. Will both of these segments be directed to the same socket at host C? If so, how will the process at host C know that these segments originated from two different hosts?*

UDP sockets use the same port for incoming connections, therefore both segments will be directed to the same socket as host C (port 787). Host C differentiates the packets from A and B by reading a datagram from each A and B containing the address (source and dest.) and port.

## Problem 2 – UDP and TCP Checksum

a) *Suppose you have the following two bytes: 00110101 and 01101001. What is the resulting checkum?*

```
  00110101
 +01101001
 = 10011110
```

Then take one's complement of 10011110 is = **01100001**

b) *Suppose you have the following two bytes: 11110101 and 01101001. What is the resulting checksum?*

```
   1111 0101
  +0110 1001
 = 1 0101 1110
```

One's comp of 101011110 = **010100001**

c) *For the bytes in part a), give an example where one bit is flipped in each of the two bytes and yet the 1s complement doesn't change.*

00110101 and 01101001 -> flip the 3rd bit from the right:
00110101 becomes 00110001 and 01101001 becomes 01101101.

```
  00110001
 +01101101
 = 10011110
```

Then one's complement of 10011110 is = **01100001, same as part A.**

## Problem 3 – Designing a minimalist data transfer protocol

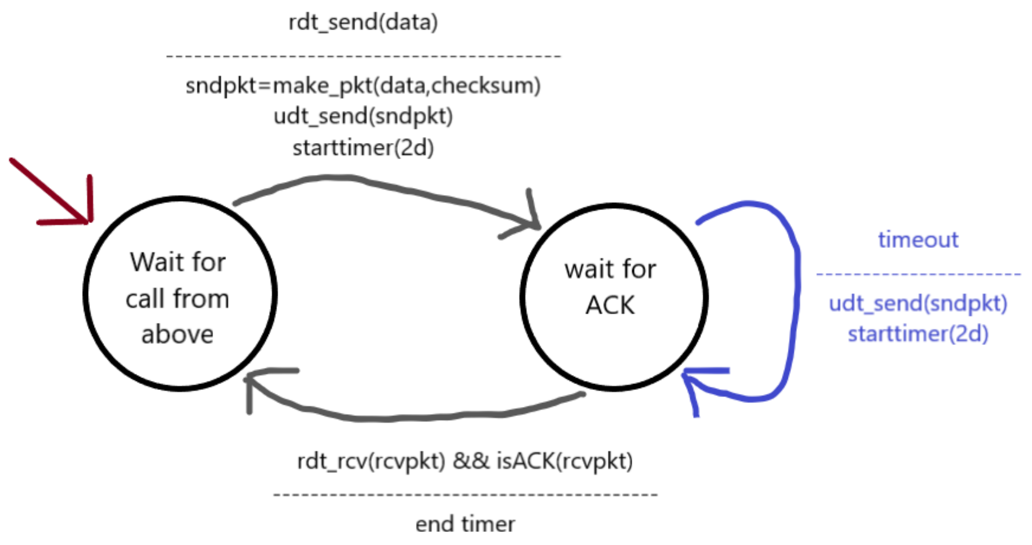
*Chapter 3 shows a number of mechanisms used to provide for reliable data transfer:*

- *Checksum*
- *ACKs*
- *Timers*
- *Sequence numbering*

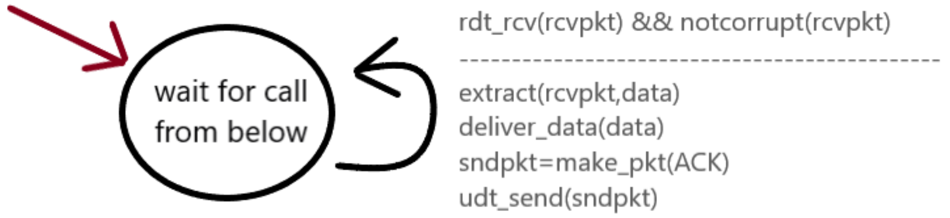
*Consider a sender and receiver that are connected by a sender-to-receiver channel that can corrupt and lose packets. The receiver-to-sender channel is perfect (that is, it will not lose or corrupt packets). The delay on each channel is known to always be less than some maximum value, d. Neither channel will reorder packets. (Note: re-read the channel properties just described and make sure you understand them!) Design a reliable data transfer protocol for this scenario using only those mechanisms (among the four listed above) that are absolutely required. Your protocol should be as simple as possible but have the functionality to deliver data reliably under the stated assumptions. Your solution does not need to be efficient; it must work correctly.*

a) *Draw the sender and receiver FSMs*

**Sender:**



**Receiver:**



b) *For each of the mechanisms (from among the four listed above) that you use in your protocol, explain the role/purpose of the mechanism and why you cannot get by without it. (Note: this* *does not imply that your protocol will use all four mechanisms above—maybe it does; maybe it does not. However, you must explain why you need the mechanisms that you have chosen to include.)*

Since the sender-receiver protocol can corrupt and lose packets, **ACKs** must be used in that protocol to confirm that the whole package was transmitted. In addition, a **timer** must be used to refresh the state of the machine (to check the ACK status). The validity/completeness of the packet (which controls the value of the ACK) is computed using a **checksum**.
In the receiver-sender protocol, only the ACK is needed.

## Problem 4 – Round-trip time estimation

*Let $\alpha = 0.2$. Suppose for a given TCP connection three acknowledgments have been returned with RTTs: RTT for first ACK = 80 msec; RTT for second ACK = 60 msec; and RTT for third ACK = 100 msec. Determine the value of EstimatedRTT after each of the three acknowledgments.*

$\text{EstimatedRTT}_{(N+1)\text{th}} = (1 - \alpha) * \text{EstimatedRTT}_{\text{Nth}} + \alpha * \text{SampleRTT}$

$\text{EstimatedRTT}_{\text{1st}} = \text{SampleRTT}$

First:

SampleRTT = 80ms

$\text{EsimatedtRTT}_{\text{1st}} = \text{SampleRTT} =$ **80ms**

Second:

$\text{EstimatedRTT}_{\text{2nd}} = (1 - 0.2) * \text{EsimatedtRTT}_{\text{1st}} + (0.2*60\text{ms}) = (0.8*80\text{ms}) + (12\text{ms}) =$ **76ms**

Third:

$\text{EstimatedRTT}_{\text{3rd}} = 0.8 * \text{EstimatedRTT}_{\text{2nd}} + (0.2*100\text{ms}) = (0.8*76\text{ms}) + (0.2*100\text{ms}) =$ **71.2ms**