CS3530: Computer Networks **2023**

Lecture 8: 13 September 2023

Instructor: Praveen Tamanna *Scribe: Gautam Singh*

8.1 UDP (Cont'd)

8.1.1 UDP Segment Header

The header contains four fields, each 16-bit wide.

1. Source port
2. Destination port
3. Length of data
4. Checksum

Data bytes follow after these fields.

8.1.2 UDP Checksum

The goal of UDP checksum is to *detect* errors in transmitted segment, and help in reliable transmission of packets.

At the sender:

1. Treat contents of UDP segment as sequence of 16-bit integers.
2. The checksum contains the addition (one's complement sum) of segment content.
3. Checksum value put into UDP checksum field.

At the receiver:

1. Recompute checksum.
2. If not equal to checksum field, then manipulation detected.
3. If equal, does not *guarantee* manipulation.

Note: Add carry bit back to 16-bit checksum and take complement.

However, this checksum is *weak*. For instance, more than one bit flip can result in the same checksum. But this is a fast way of computing a checksum, and easy to implement. Therefore, we need to tradeoff speed and security.

8.2 Principles of Reliable Data Transfer (RDT)

Reliable data transfer is implemented as sender and receiver sides of transport layer protocol sending data over an unreliable channel in the network layer. Complexity of reliable data transfer protocol will depend on characteristics of unreliable channel.

8.2.1 RDT Protocol Interfaces

There are four interfaces in this protocol, which is bidirectional. Finite State Machines (FSMs) will be used to specify sender and receiver.

1. `rdt_send()`
2. `udt_send()`
3. `rdt_rcv()`
4. `deliver_data()`

8.2.2 RDT 1.0: Reliable Transfer Over Reliable Channel

Here, we consider a perfectly reliable channel with no bit errors or loss of packets. We have separate FSMs for sender and receiver, where sender sends data from underlying channel and receiver reads from it.

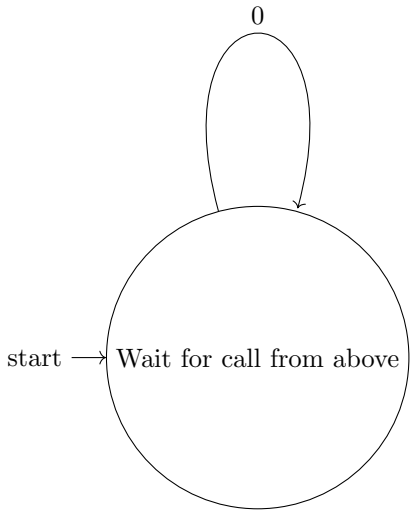

Figure 8.1: RDT 1.0 FSM.

8.2.3 RDT 2.0: Channel With Bit Errors

Here, the underlying channel may flip bits in packet. Error recovery can be done in the following ways:

1. **Acknowledgements (ACKs)**: receiver explicitly tells sender that packet was received OK.
2. **Negative Acknowledgements (NAKs)**: receiver explicitly tells sender that packet had errors, and *retransmits* packet on recipt of NAK.

This is known as *stop and wait*. Note that the state of receiver isn't known to sender unless communicated back to sender.

RDT 2.0 has a fatal flaw. Suppose that the ACK/NAK gets corrupted, then the sender cannot verify whether a NAK or ACK was received. Moreover, the packet cannot be retransmitted, since it may be a possible duplicate. Handling duplicates is done using a *sequence number* added to each packet.

8.2.3.1 RDT 2.1: Handling Garbled ACK/NAK