Redes de computadoras, 6 de diciembre de 2024

Transferencia confiable, TCP y control de congestión

Martínez Buenrostro Jorge Rafael

correo, *molap*96@*gmail.com*
Universidad Autónoma Metropolitana
Unidad Iztapalapa, México

Knowledge Checks

Connectionless Transport: UDP

UDP header files

Which fields are in a UDP segment header? Source port, destination port, length (of UDP header plus payload), internet checksum

UDP segment length field

Why is the UDP header length field needed? Because the payload section can be of variable length, and this lets UDP know where the segment ends

Internet Checksum and UDP

Over what set of bytes is the checksum field in the UDP header computed over? The entire UDP segment, except the checksum field itself, and the IP senders and receive address fields

- On the sending side, the UDP sender will take each application-layer chunk of data written into a UDP socket and send it in a distinct UDP datagram. And then on the receiving side, UDP will deliver a segment's payload into the appropiate socket, preserving the application-defined message boundary.

What is checksum?

The next statements are true about a checksum:

- A checksum is computed at a sender by considering each byte within a packet as a number, and then adding these numbers (each number representing a bytes) together to compute a sum (which is known as a checksum)
- The sender-computed checksum value is often included in a checksum field within a packet header

- The receiver of a packet with a checksum field will add up the received bytes, just as the sender did, and compare this locally-computed checksum with the checksum value in the packet header. If these two values are **different** then the receiver **knows** that one of the bits in the received packet has been changed during transmission from sender to receiver.

Computing the internet Checksum (1)

Compute the Internet checksum for these two 16-bit words: 11110101 11010011 and 10110011 01000100 To compute the Internet checksum of a set of 16-bit words, we compute the one's complement sum of the two words. That is, we add the two numbers together, making sure that any carry into the 17th bit of this initial sum is added back into the 1's place of the resulting sum); we then take the one's complement of the result. This means that the Internet checksum is **01010110 11100111**

Computing the internet Checksum (2)

Compute the Internet checksum for these two 16-bit words: 01000001 11000100 and 00100000 00101011 To compute the Internet checksum of a set of 16-bit words, we compute the one's complement sum of the two words. That is, we add the two numbers together, making sure that any carry into the 17th bit of this initial sum is added back into the 1's place of the resulting sum); we then take the one's complement of the result. This means that the Internet checksum is **10011110 00010000**

∗https://traductordebinario.com/calculadora-de-sumas-binario/
∗https://www.allmath.com/es/calculadora-del-complemento-a-uno.php

UDP Checksum: how good is it?

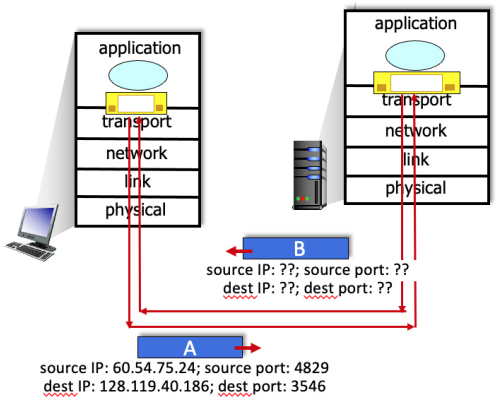
When computing the Internet checksum for two numbers, a single flipped bit (i.e., in just one of the two numbers) will always result in a changed checksum

UDP Checksum: how good is it?

When computing the Internet checksum for two numbers, a single flipped bit (i.e., in just one of the two numbers) will NOT result in a changed checksum.

IP addresses and port numbers in a UDP segment sent in reply

Now consider the UDP datagram (and the IP datagram that will encapsulate it) sent in reply by the application on host 128.119.40.186 to the original sender host, labeled B in the figure above.



- The source port number of the UDP segment (B) sent in reply is: **3546**
- The source IP address of the IP datagram containing the UDP segment (B) sent in reply is: **128.119.40.186**
- The destination port number of the UDP segment (B) sent in reply is: **4829**
- The destination IP address of the IP datagram containing the UDP segment (B) sent in reply is: **60.54.75.24**

Principles of Reliable Data Transfer

Reliable data transfer protocol mechanisms

Consider the purpose/goals/use of different reliable data transfer protocol mechanism. For the given purpose/goal/use this is the RDT mechanism that is used to implement the given purpose/goal/use

NAK	Lets the sender know that a packet was NOT received correctly at the receiver
Checksum	Used by the sender or receiver to detect bits flipped during a packet's transmission
Sequence numbers	Allows for duplicate detection at receiver
ACK	Lets the sender know that a packet was received correctly at the receiver
Retransmission	Allows the receiver to eventually receive a packet that was corrupted or lost in an earlier transmission

Cumulative ACK

What is meant by cumulative acknowledgment, ACK(n)? A cumulative ACK(n) acks all packets with a sequence number up to and including n as being received

Stop-and-wait: channel utilization

Suppose a packet is 10k bits long, the channel transmission rate connecting a sender and receiver is 10 Mbps, and the round-trip propagation delay is 10 ms. What is the maximum channel utilization of a stop-and-wait protocol for this channel?

$$U_{sender} = \frac{L/R}{RTT + L/R} = \frac{10,000/10,000,000}{0{,}01 + 10,000/10,000,000} = 0{,}09 \approx 0{,}1$$

where

L	packet size
R	transmission rate
RTT	Round-trip propagation

Channel utilization with pipelining

Suppose a packet is 10K bits long, the channel transmission rate connecting a sender and receiver is 10 Mbps, and the round-trip propagation delay is 10 ms. What is the channel utilization if a pipelined protocol with an arbitrarily high level of pipelining for this channel?

$$U_{sender} = \frac{N * L/R}{RTT + L/R}$$

where

L	packet size
R	transmission rate
RTT	Round-trip propagation
N	packet pipelining increased utilization

Channel utilization with pipelining (more)

Suppose a packet is 10K bits long, the channel transmission rate connecting a sender and receiver is 10 Mbps, and the round-trip propagation delay is 10 ms. How many packets can the sender transmit before it starts receiving acknowledgment back? The answer is 10 packets

Pipelining

The next statements are true about pipelining:

- A pipelined sender can have trasmitted multiple packets for which the sender has yet to receive an ACK from the receiver
- With a pipelined sender, there may be transmitted packet ïn flight propagating through the channel - packets that the sender has sent but that the receiver has not yet received

Packet buffering in Go-Back-N

What are some reasons for discarding received-but-out-of-sequence packets at the receiver in GBN?

- The sender will resedn that packet in any case
- The implementation at the receiver is simpler

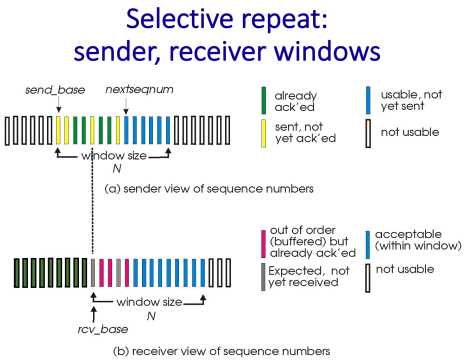
Packet buffering in Go-Back-N (more)

What are some reasons for *not* discarding received-but-out-of-sequence packets at the receiver in GBN?

- Even though that packet will be retransmitted, its next retransmission could be corrupted, so don't discard a perfectly well-received packet

Receiver operation in Selective Repeat

In the SR receiver window (see diagram below), why haven't the red packets been delivered yet?



There is a packet with a lower sequence number than any of the red packets that has yet to be received, so in-order dalivery of data in the red packets up to the application layer is not possible

Receiver operation in Selective Repeat (more)

In SR, why does the receiver have to acknowledge packets with sequence numbers that are less than those in its window, which starts at *rcv-base* Because the sender may not have received an ACK for that packet yet