Internet Technology

07r. Assignment 6 review

Paul Krzyzanowski

Rutgers University

Spring 2016

Question 1a

Suppose Host A sends two TCP segments back to back to Host B over a TCP connection. The first segment has sequence number 90; the second has sequence number 110.

1a. How much data is in the first segment?

TCP sequence numbers count bytes from a random starting number.

If segment #2 has sequence # 110 and segment #1 had sequence #90:

- Segment #1: contains bytes #90 … 109 ⇒ 20 bytes
- Segment #2: contains bytes #110... ?

Question 1b

Suppose Host A sends two TCP segments back to back to Host B over a TCP connection. The first segment has sequence number 90; the second has sequence number 110.

1b. Suppose that the first segment is lost but the second segment arrives at B. In the acknowledgment that Host B sends to Host A, what will be the acknowledgment number?

TCP acknowledgements are always for the first missing byte number.

If segment #2 is received (bytes 110…?) but segment #1 is missing, that means the receiver expects to receive bytes starting at 90.

It will send an ACK # 90.

Question 2a

UDP and TCP use 1s complement for their checksums. Suppose you have the following three 8-bit bytes:

01010011

01100110

01110100

2a. What is the 1s complement of the sum of these 8-bit bytes?

(Note that although UDP and TCP use 16-bit words in computing the checksum, for this problem you are being asked to compute 8-bit sums.)

(a) Add digits

(b) Whenever there is an overflow, add 1

(c) Invert (complement) the result

It.

```
  0 1 0 1 0 0 1 1
+ 0 1 1 0 0 1 1 0
-----------------
  1 0 1 1 1 0 0 1

  1 0 1 1 1 0 0 1
+ 0 1 1 1 0 1 0 0
-----------------
  0 0 1 0 1 1 0 1
+               1  (carry)
-----------------
  0 0 1 0 1 1 1 0

~ 0 0 1 0 1 1 1 0
-----------------
  1 1 0 1 0 0 0 1
```


Question 2b

Why is it that UDP takes the 1s complement of the sum; that is, why not just use the sum?

With the 1s complement scheme, how does the receiver detect errors?

To detect errors, the receiver adds all the 16-bit words of the segment, including the checksum.

The result should be all bits 1.

If any bit of the result contains a zero, the receiver knows there is an error in the segment.

Question 2c

(2c) Is it possible that a 1-bit error will go undetected?

No – any 1-bit error will be detected.

It will cause a 1 to become a 0 or a 0 to become a 1 in the sum.

Question 2d

(2d) How about a 2-bit error?

Yes, in some cases two-bit errors can be undetected.

For example, if the last digit of the first word is converted to a 0 and the last digit of the second word is converted to a 1.

```
  1 0 0 1          1 0 1 1
+ 1 0 1 1        + 1 0 0 1     Two bit errors
---------        ---------
  0 1 0 0          0 1 0 0
+       1        +       1
---------        ---------
  0 1 1 0          0 1 1 0
  1 0 0 1          1 0 0 1  <-- Same checksum
```


Question 3

Consider the cross-country example shown in Figure 3.17. How big would the window size have to be for the channel utilization to be greater than 98%? Suppose that the size of a packet is 1,500 bytes, including both header fields and data.

See the text on the bottom of page 215 and top of 217

RTT across the country ≈ 30 ms

Transmission rate, $R = 10^9$ bits/s

(From the text)

Packet size, L = 1500 bytes = 1500×8 = 12,000 bits

Transmission time, $d = L/R = 12000 / 10^9 = 12/10^6 = 12\ \mu s = 0.012$ ms

a. A stop-and-wait protocol in operation

b. A pipelined protocol in operation

Question 3 (continued)

Consider the cross-country example shown in Figure 3.17. How big would the window size have to be for the channel utilization to be greater than 98%? Suppose that the size of a packet is 1,500 bytes, including both header fields and data.

See the text on the bottom of page 215 and top of 217

RTT across the country ≈ 30 ms

Transmission rate, $R = 10^9$ bits/s

(From the text)

Packet size, L = 1500 bytes = 1500×8 = 12,000 bits

Transmission time, $d = L/R = 12000 / 10^9 = 12/10^6 = 12\ \mu s = 0.012$ ms

Utilization for a stop-and-wait protocol (one packet followed by an ack) =
= (L/R) / (RTT + L/R) = 0.012 / (30 + 0.012)

Utilization for a pipeline of N packets, $U = (NL/R)$ / (RTT + L/R) =
= 0.012N / (30 + 0.012)

Solve for N with U = 0.98 (98%)

(0.98 × 30.012)/0.012 = 2450.92 = 2451 packets

Question 4

What is the relationship between the variable `SendBase` in Section 3.5.4 and the variable `LastByteRcvd` in Section 3.5.5?

`SendBase`: the smallest sequence # of a transmitted but unacknowledged byte

`LastByteRcvd`: the number of the last byte in the data stream that has arrived from the network and has been placed in the receive buffer.

At any given time t, `SendBase` – 1 is the sequence number of the last byte that the sender knows has been received correctly, and in order, at the receiver.

The actual last byte received (correctly and in order) at the receiver at time t may be greater if there are acknowledgements in the pipe. Thus

`SendBase` - 1 ≤ `LastByteRcvd`

Question 5

What is the relationship between the variable LastByteRcvd in Section 3.5.5 and the variable `y` in Section 3.5.4?

`y`: value in the ACK field of a received segment

- When, at time t, the sender receives an acknowledgement with value `y`, the sender knows for sure that the receiver has received everything up through `y-1`.
- The actual last byte received (correctly and in order) at the receiver at time t may be greater if `y` ≤ `SendBase` or if there are other acknowledgements in the pipe. Thus:

`y-1` ≤ `LastByteRvcd`

A little more about IPv6

IPv6 Key Features

- Huge address space
  - 128 vs 32 bits - four times as many bits as IPv4
  - No need for NAT because there are enough addresses so each device can have a unique address
- Simpler header
  - Fields that are not always necessary are moved to an optional section at the end
  - Makes the router's job easier
- IPv6 is a separate network-layer protocol from IPv4
  - Routers & hosts need to be aware of it to use it
  - Transport-layer protocols, TCP & UDP, remain exactly the same

IPv6 Address Notation

- IPv4: 32 bits = 4 bytes
  - We used "dot decimal" notation: 4 decimal numbers separated by dots
  - Example: 128.6.4.2
- IPv6: 128 bits = 16 bytes
  - If we did the same thing:
    42.3.40.128.255.254.0.12.250.206.176.12.0.0.0.53

We don't do this!

- Break up an IPv6 address into 8 16-bit blocks
- Each block is converted to hexadecimal and delimited with colons

`2a03:2880:fffe:000c:face:b00c:0000:0035`

16-bit block

Shortening the address

- Remove leading zeros
  - Instead of `2a03:2880:fffe:000c:face:b00c:0000:0035`
  - We can write `2a03:2880:fffe:c:face:b00c:0:35`
  - Every 16-bit block must have at least one digit
- Compress zeros
  - Some IPv6 addresses contain long sequences of 0s
  - A single contiguous sequence of 16-bit blocks that contain all 0s can be abbreviated as `::` – a double-colon
  - Example
    - Instead of `fe80:0:0:0:2aa:ff:fe9a:4ca2`
    - Write `fe80::2aa:ff:fe9a:4ca2`
    - Instead of `ff02:0:0:0:0:0:0:2`
    - Write `ff02::2`
  - Cannot include zeros from parts of 16-bit blocks
  - Can only be used once for an address (so you can figure out # of blocks)

IPv4 CIDR Notation

- IP address contains two parts: network and host
- Classless Inter-Domain Routing (CIDR) notation identifies # of bits that identify the network part of the address
  - Routers look only at those bits when looking up an address to find a route
- Examples
  - 15.240.238.61/8 - top 8 bytes identify the network
  - 128.6.13.255/16 - ls.cs.rutgers.edu: top 16 bits identify the network
    That's what routers outside of Rutgers care about – it is used to send datagrams to Rutgers
  - 128.6.13.255/25 - ls.cs.rutgers.edu: top 25 bits identify the subnet within Rutgers
    This is only within Rutgers – Rutgers has many subnets and internal routers route datagrams between them based on the network prefix

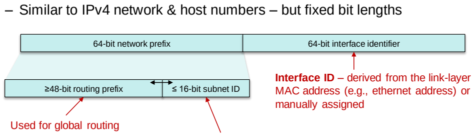
CIDR & IPv6 prefixes

Same CIDR notation for IPv6

`21DA:D3::/48`
`21DA:00D3:0000:0000:0000:0000:0000:0000`

Top 48 bits: route prefix (= network ID)

- IPv6 logically breaks up a 128-bit address into two parts
  - 64-bit network prefix – used for routing
  - 64-bit interface identifier – identifies a host in a network
  - Similar to IPv4 network & host numbers – but fixed bit lengths



The end