第3章

### R6

Yes. The application developer can put reliable data transfer into the application layer protocol. This would require a significant amount of work and debugging, however.

### R8

For each persistent connection, the Web server creates a separate “connection socket”. Each connection socket is identified with a four-tuple: (source IP address, source port number, destination IP address, destination port number). When host C receives and IP datagram, it examines these four fields in the datagram/segment to determine to which socket it should pass the payload of the TCP segment. Thus, the requests from A and B pass through different sockets. The identifier for both of these sockets has 80 for the destination port; however, the identifiers for these sockets have different values for source IP addresses. Unlike UDP, when the transport layer passes a TCP segment’s payload to the application process, it does not specify the source IP address, as this is implicitly specified by the socket identifier.

### R11

A timer would still be necessary in the protocol rdt 3.0. If the round trip time is known then the only advantage will be that, the sender knows for sure that either the packet or the ACK (or NACK) for the packet has been lost, as compared to the real scenario, where the ACK (or NACK) might still be on the way to the sender, after the timer expires. However, to detect the loss, for each packet, a timer of constant duration will still be necessary at the sender.

### R15

a) 20 bytes b) ack number = 90

### P5

No, the receiver cannot be absolutely certain that no bit errors have occurred. This is because of the manner in which the checksum for the packet is calculated. If the corresponding bits (that would be added together) of two 16-bit words in the packet were 0 and 1 then even if these get flipped to 1 and 0 respectively, the sum still remains the same. Hence, the 1s complement the receiver calculates will also be the same. This means the checksum will verify even if there was transmission error.

### P6

Suppose the sender is in state “Wait for call 1 from above” and the receiver (the receiver shown in the homework problem) is in state “Wait for 1 from below.” The sender sends a packet with sequence number 1, and transitions to “Wait for ACK or NAK 1,” waiting for an ACK or NAK. Suppose now the receiver receives the packet with sequence number 1 correctly, sends an ACK, and transitions to state “Wait for 0 from below,” waiting for a data packet with sequence number 0. However, the ACK is corrupted. When the rdt2.1 sender gets the corrupted ACK, it resends the packet with sequence number 1. However, the receiver is waiting for a packet with sequence number 0 and (as shown in the home work problem) always sends a NAK when it doesn't get a packet with sequence number 0. Hence the sender will always be sending a packet with sequence number 1, and the receiver will always be NAKing that packet. Neither will progress forward from that state.

### P8

The sender side of protocol rdt3.0 differs from the sender side of protocol 2.2 in that timeouts have been added. We have seen that the introduction of timeouts adds the possibility of duplicate packets into the sender-to-receiver data stream. However, the receiver in protocol rdt.2.2 can already handle duplicate packets. (Receiver-side duplicates in rdt 2.2 would arise if the receiver sent an ACK that was lost, and the sender then retransmitted the old data). Hence the receiver in protocol rdt2.2 will also work as the receiver in protocol rdt 3.0.

### P22

1. Here we have a window size of N=3. Suppose the receiver has received packet k-1, and has ACKed that and all other preceding packets. If all of these ACK's have been received by sender, then sender's window is [k, k+N-1]. Suppose next that none of the ACKs have been received at the sender. In this second case, the sender's window contains k-1 and the N packets up to and including k-1. The sender's window is thus [k-N,k-1]. By these arguments, the senders window is of size 3 and begins somewhere in the range [k-N,k].
2. If the receiver is waiting for packet k, then it has received (and ACKed) packet k-1 and the N-1 packets before that. If none of those N ACKs have been yet received by the sender, then ACK messages with values of [k-N,k-1] may still be propagating back.Because the sender has sent packets [k-N, k-1], it must be the case that the sender has already received an ACK for k-N-1. Once the receiver has sent an ACK for k-N-1 it will never send an ACK that is less that k-N-1. Thus the range of in-flight ACK values can range from k-N-1 to k-1.

### P24

1. True. Suppose the sender has a window size of 3 and sends packets 1, 2, 3 at . At  the receiver ACKS 1, 2, 3. At   the sender times out and resends 1, 2, 3. At  the receiver receives the duplicates and re-acknowledges 1, 2, 3. At  the sender receives the ACKs that the receiver sent at  and advances its window to 4, 5, 6. At  the sender receives the ACKs 1, 2, 3 the receiver sent at . These ACKs are outside its window.
2. True. By essentially the same scenario as in (a).
3. True.
4. True. Note that with a window size of 1, SR, GBN, and the alternating bit protocol are functionally equivalent. The window size of 1 precludes the possibility of out-of-order packets (within the window). A cumulative ACK is just an ordinary ACK in this situation, since it can only refer to the single packet within the window.

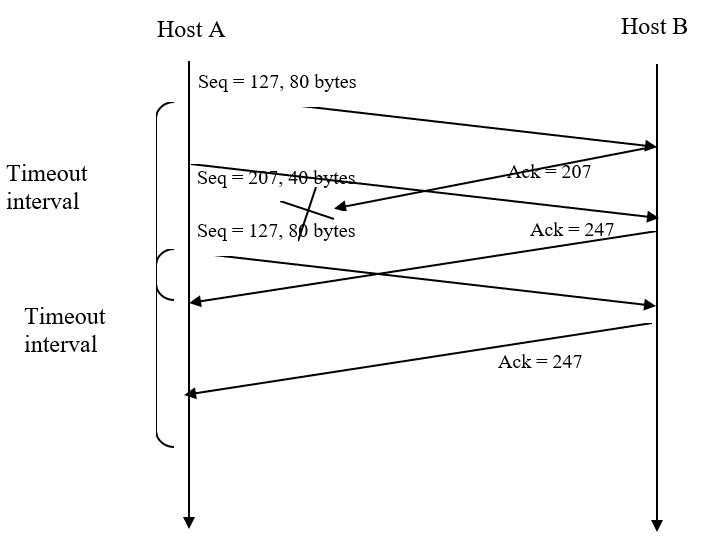
### P26

There are  possible sequence numbers.

1. The sequence number does not increment by one with each segment. Rather, it increments by the number of bytes of data sent. So the size of the MSS is irrelevant -- the maximum size file that can be sent from A to B is simply the number of bytes representable by .
2. The number of segments is. 66 bytes of header get added to each segment giving a total of 528,857,934 bytes of header. The total number of bytes transmitted is  bytes.

Thus it would take 249 seconds to transmit the file over a 155~Mbps link.

### P27

1. In the second segment from Host A to B, the sequence number is 207, source port number is 302 and destination port number is 80.
2. If the first segment arrives before the second, in the acknowledgement of the first arriving segment, the acknowledgement number is 207, the source port number is 80 and the destination port number is 302.
3. If the second segment arrives before the first segment, in the acknowledgement of the first arriving segment, the acknowledgement number is 127, indicating that it is still waiting for bytes 127 and onwards.
4. 

### P33

Let’s look at what could wrong if TCP measures SampleRTT for a retransmitted segment. Suppose the source sends packet P1, the timer for P1 expires, and the source then sends P2, a new copy of the same packet. Further suppose the source measures SampleRTT for P2 (the retransmitted packet). Finally suppose that shortly after transmitting P2 an acknowledgment for P1 arrives. The source will mistakenly take this acknowledgment as an acknowledgment for P2 and calculate an incorrect value of SampleRTT.

### P40

略

### P43

In this problem, there is no danger in overflowing the receiver since the receiver’s receive buffer can hold the entire file. Also, because there is no loss and acknowledgements are returned before timers expire, TCP congestion control does not throttle the sender. However, the process in host A will not continuously pass data to the socket because the send buffer will quickly fill up. Once the send buffer becomes full, the process will pass data at an average rate or R << S.

### P46

1. Let W denote the max window size measured in segments. Then, W\*MSS/RTT = 10Mbps, as packets will be dropped if the maximum sending rate exceeds link capacity. Thus, we have W\*1500\*8/0.15=10\*10^6, then W is about 125 segments.
2. As congestion window size varies from W/2 to W, then the average window size is 0.75W=94 (ceiling of 93.75) segments. Average throughput is 94\*1500\*8/0.15 =7.52Mbps.
3. When there is a packet loss, W becomes W/2, i.e., 125/2=62.  
     (125 - 62) \*0.15 = 9.45 seconds, as the number of RTTs (that this TCP connections needs in order to increase its window size from 62 to 125) is 63. Recall the window size increases by one in each RTT.

第4章

### R1

A network-layer packet is a datagram. A router forwards a packet based on the packet’s IP (layer 3) address. A link-layer switch forwards a packet based on the packet’s MAC (layer 2) address.

### R17

8-bit protocol field in the IP datagram contains information about which transport layer protocol the destination host should pass the segment to.

### R18

Time-to-live.

### R19

No. IP header checksum only computes the checksum of an IP packet’s IP header fields, which share no common bytes with the IP datagram’s transport-layer segment part.

### R25

50% overhead.

P5

a)

Prefix Match Link Interface

11100000 00 0

11100000 01000000 1

1110000 2

11100001 1 3

otherwise 3

b) Prefix match for first address is 5th entry: link interface 3

Prefix match for second address is 3nd entry: link interface 2

Prefix match for third address is 4th entry: link interface 3

### P8

223.1.17.0/26

223.1.17.128/25

223.1.17.192/28

### P11

Any IP address in range 128.119.40.128 to 128.119.40.191

Four equal size subnets: 128.119.40.64/28, 128.119.40.80/28, 128.119.40.96/28, 128.119.40.112/28

### P12

From 214.97.254/23, possible assignments are

a) Subnet A: 214.97.255/24 (256 addresses)

Subnet B: 214.97.254.0/25 - 214.97.254.0/29 (128-8 = 120 addresses)

Subnet C: 214.97.254.128/25 (128 addresses)

Subnet D: 214.97.254.0/31 (2 addresses)

Subnet E: 214.97.254.2/31 (2 addresses)

Subnet F: 214.97.254.4/30 (4 addresses)

b) To simplify the solution, assume that no datagrams have router interfaces as ultimate destinations. Also, label D, E, F for the upper-right, bottom, and upper-left interior subnets, respectively.

**Router 1**

**Longest Prefix Match Outgoing Interface**

11010110 01100001 11111111 Subnet A

11010110 01100001 11111110 0000000 Subnet D

11010110 01100001 11111110 000001 Subnet F

**Router 2**

**Longest Prefix Match Outgoing Interface**

11010110 01100001 11111111 0000000 Subnet D

11010110 01100001 11111110 0 Subnet B

11010110 01100001 11111110 0000001 Subnet E

**Router 3**

**Longest Prefix Match Outgoing Interface**

11010110 01100001 11111111 000001 Subnet F

11010110 01100001 11111110 0000001 Subnet E

11010110 01100001 11111110 1 Subnet C

### P14

The maximum size of data field in each fragment = 680 (because there are 20 bytes IP header). Thus the number of required fragments 

Each fragment will have Identification number 422. Each fragment except the last one will be of size 700 bytes (including IP header). The last datagram will be of size 360 bytes (including IP header). The offsets of the 4 fragments will be 0, 85, 170, 255. Each of the first 3 fragments will have flag=1; the last fragment will have flag=0.

第5章

### R4

Link state algorithms: Computes the least-cost path between source and destination using complete, global knowledge about the network. Distance-vector routing: The calculation of the least-cost path is carried out in an iterative, distributed manner. A node only knows the neighbor to which it should forward a packet in order to reach given destination along the least-cost path, and the cost of that path from itself to the destination.

### R5

The count-to-infinity problem refers to a problem of distance vector routing. The problem means that it takes a long time for a distance vector routing algorithm to converge when there is a link cost increase. For example, consider a network of three nodes x, y, and z. Suppose initially the link costs are c(x,y)=4, c(x,z)=50, and c(y,z)=1. The result of distance-vector routing algorithm says that z’s path to x is z🡪y🡪 x and the cost is 5(=4+1). When the cost of link (x,y) increases from 4 to 60, it will take 44 iterations of running the distance-vector routing algorithm for node z to realize that its new least-cost path to x is via its direct link to x, and hence y will also realize its least-cost path to x is via z.

### P3

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *Step* | *N’* | *D(t),p(t)* | *D(u),p(u)* | *D(v),p(v)* | *D(w),p(w)* | *D(y),p(y)* | *D(z),p(z)* |
| 0 | x | ∞ | ∞ | 3,x | 6,x | 6,x | 8,x |
| 1 | xv | 7,v | 6,v | 3,x | 6,x | 6,x | 8,x |
| 2 | xvu | 7,v | 6,v | 3,x | 6,x | 6,x | 8,x |
| 3 | xvuw | 7,v | 6,v | 3,x | 6,x | 6,x | 8,x |
| 4 | xvuwy | 7,v | 6,v | 3,x | 6,x | 6,x | 8,x |
| 5 | xvuwyt | 7,v | 6,v | 3,x | 6,x | 6,x | 8,x |
| 6 | xvuwytz | 7,v | 6,v | 3,x | 6,x | 6,x | 8,x |

### P5

Cost to

u v x y z

v ∞ ∞ ∞ ∞ ∞

From x ∞ ∞ ∞ ∞ ∞

z ∞ 6 2 ∞ 0

Cost to

u v x y z

v 1 0 3 ∞ 6

From x ∞ 3 0 3 2

z 7 5 2 5 0

Cost to

u v x y z

v 1 0 3 3 5

From x 4 3 0 3 2

z 6 5 2 5 0

Cost to

u v x y z

v 1 0 3 3 5

From x 4 3 0 3 2

z 6 5 2 5 0

第6章

### R2

Although each link guarantees that an IP datagram sent over the link will be received at the other end of the link without errors, it is not guaranteed that IP datagrams will arrive at the ultimate destination in the proper order. With IP, datagrams in the same TCP connection can take different routes in the network, and therefore arrive out of order. TCP is still needed to provide the receiving end of the application the byte stream in the correct order. Also, IP can lose packets due to routing loops or equipment failures.

### R6

After the 5th collision, the adapter chooses from {0, 1, 2,…, 31}. The probability that it chooses 4 is 1/32. It waits 204.8 microseconds.

### R10

C’s adapter will process the frames, but the adapter will not pass the datagrams up the protocol stack. If the LAN broadcast address is used, then C’s adapter will both process the frames and pass the datagrams up the protocol stack.

### P2

Suppose we begin with the initial two-dimensional parity matrix:

0 0 0 0

1 1 1 1

0 1 0 1

1 0 1 0

With a bit error in row 2, column 3, the parity of row 2 and column 3 is now wrong in the matrix below:

0 0 0 0

1 1 0 1

0 1 0 1

1 0 1 0

Now suppose there is a bit error in row 2, column 2 and column 3. The parity of row 2 is now correct! The parity of columns 2 and 3 is wrong, but we can't detect in which rows the error occurred!

0 0 0 0

1 0 0 1

0 1 0 1

1 0 1 0

The above example shows that a double bit error can be detected (if not corrected).

### P8











Thus



### P9













### P15

1. No. E can check the subnet prefix of Host F’s IP address, and then learn that F is on the same LAN. Thus, E will not send the packet to the default router R1.

Ethernet frame from E to F:

Source IP = E’s IP address

Destination IP = F’s IP address

Source MAC = E’s MAC address

Destination MAC = F’s MAC address

1. No, because they are not on the same LAN. E can find this out by checking B’s IP address.

Ethernet frame from E to R1:

Source IP = E’s IP address

Destination IP = B’s IP address

Source MAC = E’s MAC address

Destination MAC = The MAC address of R1’s interface connecting to Subnet 3.

1. Switch S1 will broadcast the Ethernet frame via both its interfaces as the received ARP frame’s destination address is a broadcast address. And it learns that A resides on Subnet 1 which is connected to S1 at the interface connecting to Subnet 1. And, S1 will update its forwarding table to include an entry for Host A.

Yes, router R1 also receives this ARP request message, but R1 won’t forward the message to Subnet 3.

B won’t send ARP query message asking for A’s MAC address, as this address can be obtained from A’s query message.

Once switch S1 receives B’s response message, it will add an entry for host B in its forwarding table, and then drop the received frame as destination host A is on the same interface as host B (i.e., A and B are on the same LAN segment).

### P19

|  |  |
| --- | --- |
| Time, | Event |
| 0 | and  begin transmission |
| 245 | and  detect collision |
| 293 | and  finish transmitting jam signal |
| 293+245 = 538  538+96=634 | 's last bit arrives at ; detects an idle channel  A starts transmitting |
| 293+512 = 805  634+245=879 | B returns to Step2  B must sense idle channel for 96 bit times before it transmits A’s transmission reaches B |
|  |  |

Because 's retransmission reaches  before 's scheduled retransmission time (805+96),  refrains from transmitting while  retransmits. Thus  and  do not collide. Thus the factor 512 appearing in the exponential backoff algorithm is sufficiently large.

P31

(The following description is short, but contains all major key steps and key protocols involved.)

Your computer first uses DHCP to obtain an IP address. You computer first creates a special IP datagram destined to 255.255.255.255 in the DHCP server discovery step, and puts it in a Ethernet frame and broadcast it in the Ethernet. Then following the steps in the DHCP protocol, you computer is able to get an IP address with a given lease time.

A DHCP server on the Ethernet also gives your computer a list of IP addresses of first-hop routers, the subnet mask of the subnet where your computer resides, and the addresses of local DNS servers (if they exist).

Since your computer’s ARP cache is initially empty, your computer will use ARP protocol to get the MAC addresses of the first-hop router and the local DNS server.

Your computer first will get the IP address of the Web page you would like to download. If the local DNS server does not have the IP address, then your computer will use DNS protocol to find the IP address of the Web page.

Once your computer has the IP address of the Web page, then it will send out the HTTP request via the first-hop router if the Web page does not reside in a local Web server. The HTTP request message will be segmented and encapsulated into TCP packets, and then further encapsulated into IP packets, and finally encapsulated into Ethernet frames. Your computer sends the Ethernet frames destined to the first-hop router. Once the router receives the frames, it passes them up into IP layer, checks its routing table, and then sends the packets to the right interface out of all of its interfaces.

Then your IP packets will be routed through the Internet until they reach the Web server.

The server hosting the Web page will send back the Web page to your computer via HTTP response messages. Those messages will be encapsulated into TCP packets and then further into IP packets. Those IP packets follow IP routes and finally reach your first-hop router, and then the router will forward those IP packets to your computer by encapsulating them into Ethernet frames.