Department of Computer Science
The Hong Kong University of Science and Technology

Student ID_____

COMP 361 Computer Communications Networks

Spring Semester 2005

Final Examination – Solution key – version of 5/25/2005 10:58 AM

Date: May 21, 2005. Time: 12:30pm – 3:30pm. Venue: Rm 3007

Name:_____Student ID:_____Email:_____

Instructions:

1. This examination paper consists of 20 pages and 10 questions
2. Please write your name, student ID and Email on this page.
3. For each subsequent page, please write your student ID at the top of the page in the space provided.
4. Please answer all the questions within the space provided on the examination paper. You may use the back of the pages for your rough work.
5. Please read each question very carefully and answer the question clearly and to the point. Make sure that your answers are neatly written, readable and legible.
6. Show all the steps you use in deriving your answer, where ever appropriate.
7. For each of the questions assume that the concepts are known to the graders. Concentrate on answering to the point what is asked. Do not define or describe the concepts unless specifically asked to.

Question	Points	Score
1	8	
2	15	
3	12	
4	8	
5	12	
6	10	
7	12	
8	7	
9	10	
10	6	
TOTAL	100	

The notice at the bottom of this page, noting two typos in Q3 and answering one question concerning Q7 was handed out with the exam. The typos noted in Q3 were corrected in this answer sheet. Also, at the time of the exam, another typo in Q3 (the swapping of the labeling of the 802.11 and 803 boxes in Q3 b) was noted and described on the board. This typo has also been fixed in this answer sheet.

Addendum to COMP361 L1 Spring 2005 – Final Exam

Question 3)

- The **binary-backoff** algorithm is another name for the **exponential-backoff** algorithm we learnt in class

- In part (b), in the first line after the 2nd box there is a typographical error.
  Original: *What is the reason that step ii' is different than step ii'.*
  Corrected: *What is the reason that step ii' is different than step ii?*

  That is, the last ii' should be changed into ii (with no prime " **'** ") and a question mark (" **?** ") should be added.

Question 7)

- When answering parts (b) and (c) of the question you should describe what happens in **both** TCP Reno and TCP Tahoe and when it happens (in cases in which there is a difference between TCP Reno and TCP Tahoe then you should make the difference clear in your answer)

1) Answer the following true/false questions by circling either T or F. (8 points)

a) CDMA is a random-access MAC protocol used by many wireless networks T **F**

b) The 802.3 Ethernet protocol is an unslotted protocol **T** F

c) The value in the PPP *control* field signifies the upper level protocol to which the delivered frame belongs T **F**

d) Switch tables learn IP/MAC address mappings in a plug-and-play manner T **F**

e) OSPF is a link-state algorithm **T** F

f) The 802.11 protocol includes an acknowledgement from the receiver to the sender that the frame was received properly **T** F

g) IP datagrams may be fragmented into multiple parts when they are carried over ATM networks **T** F

h) The sender in the TCP protocols keeps an individual timer for each unacknowledged packet T **F**

2) (15 pts) Briefly answer each of the following questions

i) What does it mean for a protocol to be a **CSMA** protocol?

ii) What does it mean for a protocol to be a **CSMA/CD** protocol?
That is, what does the **CD** part do in addition to what the **CSMA** part already implies?

iii) What is the major advantage of a **CSMA/CD** protocol over a pure **CSMA** protocol?

iv) Wireless **CSMA** protocols, e.g., 802.11, **do no**t implement **CSMA/CD**. We said in class that one reason for this is the *Hidden-Terminal problem*. What is the Hidden-Terminal problem and why would it make it difficult to implement **CSMA/CD** in wireless networks?

v) We said in class that there are two reasons that 802.11 is not able to implement the **CD** part of **CSMA/CD.** The Hidden-Terminal problem was one reason. What is the second reason?

i) CSMA = Carrier Sense Multiple Access
Listen before transmitting. If someone else is already transmitting then back off (wait) until channel is free

ii) In CD, the protocol listens for collisions and, if it detects a collision, it aborts sending the frame.

iii) reduces channel wastage; improves efficiency.

iv) The hidden terminal problem is that two nodes A & C might both be transmitting to node B; while node B can detect transmissions from both A and C, nodes A & C can not detect transmissions from each other. This might be because they are hidden from each other by a physical block, e.g., a mountain, or just because they are so far from each other that their signals can not be detected by each other. Since CSMA/CD depends upon detecting collisions this is a problem (since collisions of simultaneous messages from A & C to B would not be detected).

v) Many wireless nodes turn off their receivers while transmitting (or their receivers do not work while transmitting). Thus, collision detection is impossible.

3) (12 pts)

a) Describe the details of the standard Ethernet (802.3) *binary-backoff* algorithm.

When does 802.3 use the binary-backoff algorithm?

b) Consider the following difference between the wired Ethernet (802.3) protocol and the wireless (802.11) protocols in how hosts handle dealing with a new frame to transmit.

802.3: (i) If the adaptor senses channel idle,
it starts to transmit frame
(ii) If the adaptor senses channel busy
it waits until channel idle and then immediately transmits frame

802.11: (i') If the adaptor senses channel idle for DISF seconds
it starts to transmit frame
(ii') If the adaptor senses channel busy during the DISF seconds
it enters the binary-backoff algorithm

Answer the following question: What is the reason that step ii' is different than step ii? That is, 802.3 waits until the channel is idle and then *immediately transmits* while 802.11 will *immediately enter binary-backoff*. Why didn't the designers of 802.11 copy the 802.3 protocol?

a) 802.3 uses the binary-backoff algorithm after it detects a collision.

After a collision, the transmitter aborts and chooses a random value K and then delays K x 512 bit transmission times before attempting to transmit again.
*After waiting the K*512BTT, the adaptor waits until the channel is idle and then immediately transmits.*
K is chosen as follows:
After first collision: choose K from {0, 1}
After second collision: choose K from {0, 1, 2, and 3}…
After ten or more collisions, choose K from {0, 1, 2, 3, 4, and 1023}

b) In 802.3, collisions are detected so we can abort transmission once a collision occurs and collisions don't cause the waste of tremendous bandwidth.

In 802.11, there is no collision detection so once a frame starts transmitting, it is transmitted in its entirety, even if a collision occurs (in which case, the receiver would not ACK the transmission).

'Another' reason is the hidden-terminal problem; since collision detection by transmitters might not be possible in 802.11 due to the hidden-terminal problem, collisions can not be aborted.

For both of the reasons just given, collisions can waste lots of bandwidth. Therefore 802.11 tries to avoid collisions, if possible. (Note that the two answers given above are very related; one can think of the h-t problem as the reason that there is no collision detection in 802.11. Either answer would have been accepted as part of the solution to this problem.)

If 802.11 used ii instead of ii' then it is quite likely that a collision would occur. This would happen if two stations both are given a frame to transmit while a 3rd station (that they both could hear) is currently transmitting. In this case, both stations would start transmitting immediately after the 3rd station stopped, leading to a collision.

4) (8 pts) In this question you must calculate CRC bits.

The input data is the string of 7 bits **D = 1 1 0 1 0 0 1**

Set r=4. The r+1 generator bit pattern used by the algorithm will be **G = 1 0 0 1 1**

Given **D** and **G** as above, find the r CRC bits generated by the CRC algorithm.
For full credit you must show your calculations.

G(x) = x^4 + x + 1

x^rD(x) = x^10 + x^9 + x^7 + x^4

= G(x) (x^6 + x^5 + x + 1) + x^2 +1

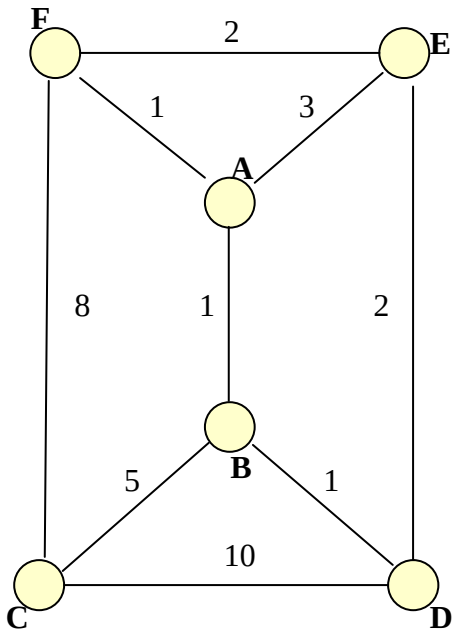
*So R(x) = X^2 +1 = 0*X^3 + 1*x^2 + 0*X + 1*

R = 0101

Student ID_____

5) (12 pts)

a) Consider the network drawn on the right.



Suppose a distance-vector algorithm with *Poison Reverse* has been run and converged to its stable solution. Fill in the entries in the table below **assuming that the table is located at node A** (that is, the values in the table should be those that A receives from its neighbors and/or calculates itself)

Cost to

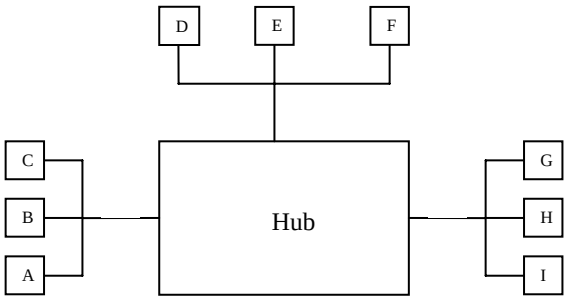
from	A	B	C	D	E	F
A	0	1	6	2	3	1
B	1	0	5	1	3	∞
E	3	3	8	2	0	2
F	1	∞	∞	∞	2	0

b) *Poison-reverse* is used to solve the count-to-infinity problem.

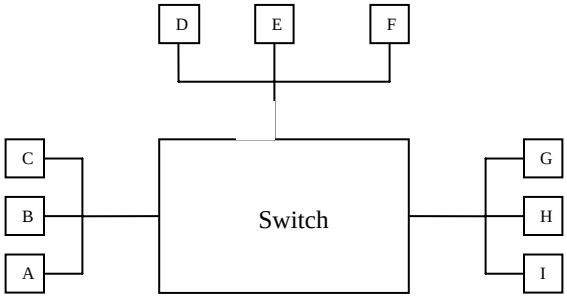
Explain why the RIP protocol uses poison reverse but the OSPF protocol does not use poison reverse.

The count-to-infinity problem only occurs with distance-vector protocols but not with link-state ones. Since RIP is a distance-vector protocol, poison-reverse is used. Since OSPF is a link-state one, poison-reverse is not used. (in fact, poison reverse is only defined for use in a distance-vector algorithm; it makes no sense for a link-state one)

6) (10 pts) Consider the two different CSMA/CD Ethernet LANS sketched below



LAN 1



LAN 2

Now consider the following sequence of frames sent consecutively in the given order. Assume that every host already contains all of the MAC addresses of all of the other hosts in its ARP table. Assume that all other existing device tables are empty before the first frame is sent and that TTL values are large enough so that once information is learnt by any device that information will **not** time out while the remaining frames in the sequence are sent.

Frame #1 is sent from A to G
Frame #2 is sent from E to G
Frame #3 is sent from A to G
Frame #4 is sent from B to A
Frame #5 is sent from B to G
Frame #6 is sent from G to A
Frame #7 is sent from A to G
Frame #8 is sent from F to E

a) If the frames are sent on **LAN 1** in the order given, which of the 8 frames are sensed by D's NIC, i.e., which frames are broadcast in D's local collision domain? Fully explain your answer.

b) If the frames are sent on **LAN 2** in the order given, which of the 8 frames are sensed by D's NIC, i.e., which frames are broadcast in D's local collision domain? Fully explain your answer.

a) D's NIC hears all eight messages. This is because a hub is a physical level repeater and does not isolate collision domains. Thus, all 9 hosts are in the same collision domain and every host (not just d) in Lan 1 hears every frame that every other host transmits.

b) The switch separates Lan2 into three different collision domains: {A,B,C}, {D,E,F} and {G,H,I}. The switch will always forward a message into the c.d. containing the proper destination. In cases in which the ARP address of the destination is not stored in the switch table it will also forward a message into the two non-source c.ds

i	*S_i = Source of frame i*	*D_i = Dest of frame i*	*C_i = content of switch table after frame i, $= C_{i-1} U \{S_i\}$*	*Is $D_i \in C_{i-1}$?*
1	*A*	*G*	*A*	*No*
2	*E*	*G*	*A,E*	*No*
3	*A*	*G*	*A,E*	*No*
4	*B*	*A*	*A,B,E*	*Yes*
5	*B*	*G*	*A,B,E*	*No*
6	*G*	*A*	*A,B,E,G*	*Yes*
7	*A*	*G*	*A,B,E,G*	*Yes*
8	*F*	*E*	*A,B,E,F,G*	*Yes*

D will hear all all frames (i) with sender in its collision domain (2 & 8), (ii) with receiver in its collision domain (8) and (iii) with sender not in the switch table at the time the frame is sent (1,2,3,5). So, D will hear the frames 1,2,3,5,8

7) (12 pts) In this question you will be describing the congestion control algorithm used by TCP. In the TCP congestion control algorithm we maintain the invariant that

LastByteSent-LastByteAcked ≤ Congwin

Recall that the Maximum Segment Size (in bytes) is denoted by **MSS.**

a) What is meant by a *loss-event*, i.e., what types of loss events exist?

b) Describe what is meant by the "*slow-start"* phase of the congestion control algorithm. To answer this question you must describe what *slow-start* does and when it is called.

c) Describe what is meant by the "*additive-increase multiplicative-decrease (AIMD)"* phase of the congestion-control algorithm. To answer this question you must describe what *AIMD* does and when it is called.

a) A loss event is a timeout or receiving 3 duplicate acks for the same packet

Note that TCP maintains a variable THRESHOLD, which starts with some large value.

*When timeout occurs, **Threshold** is set to **CongWin/2** and **CongWin** is set to 1 MSS*

When a triple duplicate ACK occurs:

- *In TCP RENO, **Threshold** set to **CongWin/2** and **CongWin** set to **Threshold**.*
- *In TCP Tahoe, **Threshold** set to **CongWin/2** and **CongWin** is set to 1 MSS.*

In practice:
TCP will be in Slow Start when CONGWIN <= THRESHOLD.
TCP will be in AIMD when CONGWIN > THRESHOLD.

b) slow start: to start CongWin set to 1MSS bytes;
*in the absence of loss events set (double) CongWin=2*CongWin, every RTT*

In practice:
This stage ends as soon as CongWin = Threshold
This is the initial starting stage of the TCP algorithm.
TCP Tahoe returns to slow-start after every loss event
TCP Reno returns to slow start only after after a timeout

c) AIMD: increase CONGWIN by 1 MSS every RTT in the absence of loss events
decrease CONGWIN by a factor of ½ after a loss event

8) (7 pts)

a) How many bits does an IPv4 address have?

b) What is the difference between *classful addressing* (the original scheme used in the internet) and *classless addressing* (CIDR)?

c) Describe what is meant by *longest-prefix matching* in IP-routing.

a) 32 bits

b) IP addresses are split into a (prefix) network address followed by a (suffix) host address. All machines on the same network will share the same network address.

Classful addressing essentially required that the network address component be one of 8 (class A), 16 (class B), or 24 (class C) bits. Those were the only options allowed.

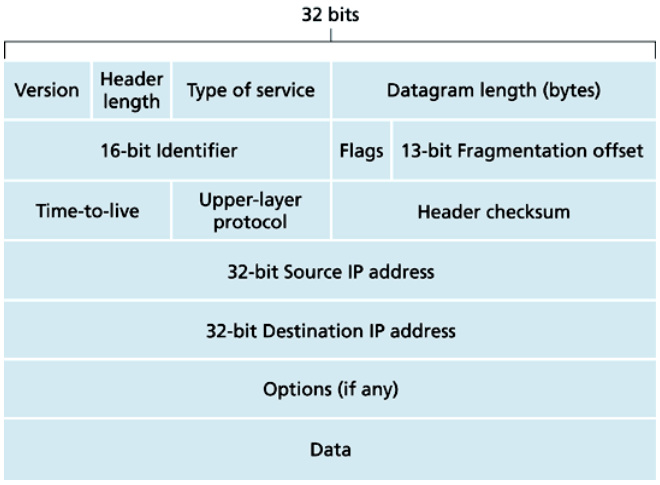
In classless addressing the network part of an IP address can be ***any*** *number of bits long. This is often denoted by the* ***dotted decimal*** *form a.b.c.d/x where x is the number of (leading) bits that compose the network component of the address.*

c) A router (host) contains a routing table. The entries in the routing table are network addresses *along with associated next hops that should be taken if a destination address "matches" the associated network address.*

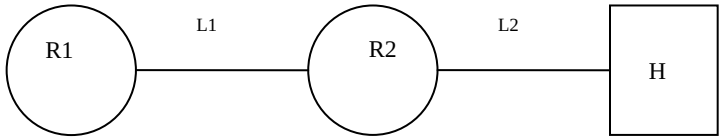
The network address is denoted by the "prefix" of an IP address and a destination is considered to "match" that prefix if the network address is actually a prefix of the destination address.

Suppose a destination address matches two or more "network addresses" in the table. Which next hop should be followed? The answer is to choose the longest *network addresse that matches the destination one and follow its next hop.*

9) (10pts)



The picture above describes the format of an IPv4 datagram. The diagram below illustrates Router R1 sending a datagram to host H through Router R2.



Link L1 only permits a MTU of 1500 bytes. Link L2 only permits a MTU of 1100 bytes.
(MTU= Maximum Transfer Unit)

A is an IP datagram which
i) Has size 4000 bytes (the size of a datagram includes its header)
ii) Is not using any of the option fields in its header.
Because A is larger than the MTUs of Links L1 and L2, A must be *fragmented* as it is sent from R1 to H. Assume that all datagrams sent are received successfully.

a) Into how many IP datagrams is **A** fragmented when it is sent from R1 to R2 over L1?
What is the size (in bytes) of each of these smaller fragments?
Which of these fragments has its *offset-flag bit* set to *0?*

b) In order for Host H to receive the data in **A**, R2 must also send some datagrams to H.
How many IP datagrams does H receive?
What is the size (in bytes) of each of these smaller fragments?
Which of these fragments has its *offset-flag bit* set to *0?*

a) An IP datagram sent over L1 can contain at most 1480 bytes of data (since it uses 20 bytes for the header). Thus A is fragmented into 3 datagrams;
two of size 1500 (1480 + 20 bytes of header) followed by
*one of size 1040 (20 bytes of header & 1020= 3980-2*1480 bytes of data).*

Only the last datagram, the one of size 1100, has its offset-flag bit set to 0. The other 2 datagrams have their offset-flag bits set to 1.

b) An IP datagram sent over L2 can contain at most 1080=1100-20 bytes of data.

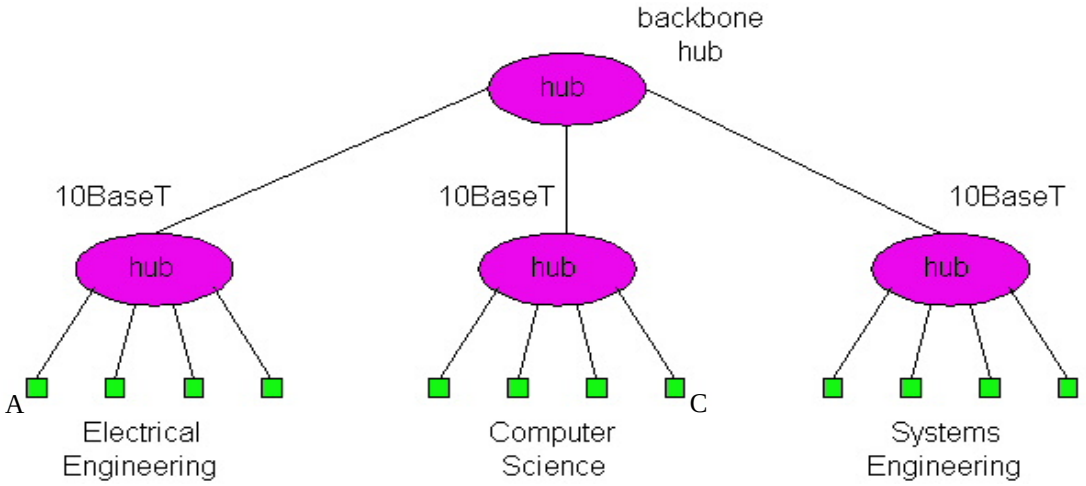
Therefore the two fragments of size 1500 (1480 data bytes) received by R2 must be fragmented further into 2 fragments; each original being fragmented into one of size 1100 (20+1080) and one of size 20+(1480-1080)=420.

The third fragment received by R2, the one of size 1040, is small enough to be forwarded over R2 without further fragmentation

Therefore H will receive 5 fragments in total:
1. Size 1100
2. Size 420
3. Size 1100
4. Size 420.
5. Size 1040

The only fragment with offset-flag bit set to 0 is the fragment of size 1040 originally sent from R1 and then forwarded onto H by R2. The other 4 datagrams have their offset-flag bits set to 1.

10) (6 pts) Consider the following CSMA/CD network configuration. You should assume that each connecting line (connecting a host to a hub or a hub to a hub) is a 50m long twisted pair line. Assume that the one-way propagation delay through a hub is 0.1 μs (recall that a μs is 10^{-6} seconds). Finally, assume that the signal propagation speed in the twisted pair lines is 2 x 10^8 m/s.



Now suppose that A starts transmitting a message and the message will collide with a message sent out by C. Under the above assumptions, what is the worst case time needed from the time that A starts transmitting until it learns about the collision.

The worst case time for a bit to get from a host in one of the departmental domains to a host in a different domain is 4 x 0.25 μs + 3 x 0.1 μs = 1.3 μs. So, the worst case amount of time needed for a collision is twice that which is 2.6μs.