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

Student ID_____

COMP 361 Computer Communications Networks

Spring Semester 2005 - Midterm Examination – Solution Key

Date March 22, 2005, Time 20:00 –21:30

Name:_____Student ID:_____Email:_____

Instructions:

1. This is a closed book exam
2. This examination paper consists of 15 pages and 8 questions
3. Please write your name, student ID and Email on this page.
4. For each subsequent page, please write your student ID at the top of the page in the space provided.
5. 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. Each question is on a separate page. This is for clarity and is not meant to imply that each question *requires* a full page answer. Many can be answered using only a few lines.
6. 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.
7. Show all the steps you use in deriving your answer, where ever appropriate.
8. 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 do so.

Question	Points	Score
1	9	
2	10	
3	15	
4	10	
5	5	
6	15	
7	20	
8	16	
TOTAL	100	

1. (9 points) Answer the following true/false questions by circling either T or F

(a) Circuit Switching uses statistical multiplexing	T	**F**
(b) *Doubling the Timeout Interval* in TCP is a limited form of congestion control	**T**	F
(c) FTP uses out-of-band control	**T**	F
(d) The user-agent of the sender of an email message uses the POP3 protocol to send the message to the local mail server.	T	**F**
(e) UDP provides a simple flow-control mechanism.	T	**F**
(f) Napster is a classic example of a P2P file-sharing system with a centralized directory.	**T**	F
(g) If peers A and B are neighbors in a Gnutella overlay network, then A and B must be geographically near to each other.	T	**F**
(h) In TCP, *slow start* refers to the additive-increase multiplicative-decrease (AIDM) changing of the congestion window.	T	**F**
(i) Two distinct web pages, e.g., **www.mit.edu/research.html** and **www.mit.edu/students.html,** can be sent over the same persistent connection.	**T**	F

2. (10 pts)
In class we said that the TCP reliable data transfer protocol is a hybrid (mixture) of *Go Back N (GBN)* and *Selective-Repeat (SR).*

A. Give one way in which TCP is sometimes like a GBN protocol and not like a SR one.
B. Give one way in which TCP is sometimes like a SR protocol and not like a GBN one

In your explanation you should be very clear about which properties you are using, and (A) why they are GBN properties and are not SR properties or (B) why they are SR properties and not GBN properties.

Here are possible solutions (not the only ones)

A. The TCP "sender" maintains only one timer; this timer is associated with the lowest numbered unack'd byte.

This is like GBN since GBN also maintains only one timer (for lowest numbered unack'd packet).

This is unlike SR since SR maintains one timer for every unack'd packet.

B. Many TCP implementations buffer received out-of-order bytes.

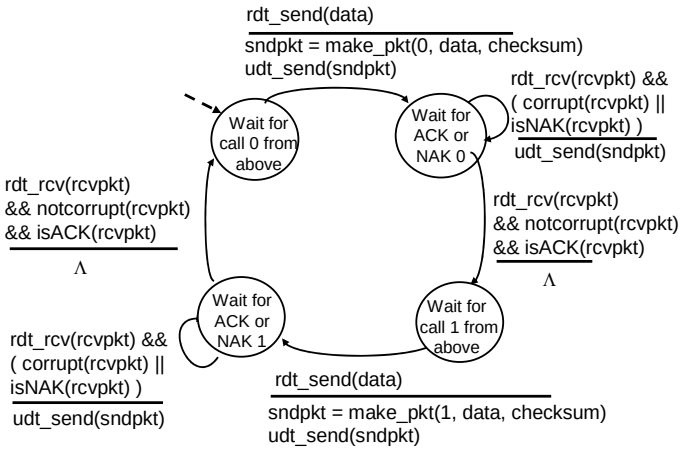
This is like SR since the SR receiver buffers received out-of-order packets (as long as they are within the receiver window).

This is unlike GBN since the GBN receiver does not buffer out-of-order packets.

*Note: If you gave the above example you must explicitly mention that TCP implementations **can** perform receiver out-of-order buffering, but that they are not **required** to do so (or that some implementations do it and some don't).*

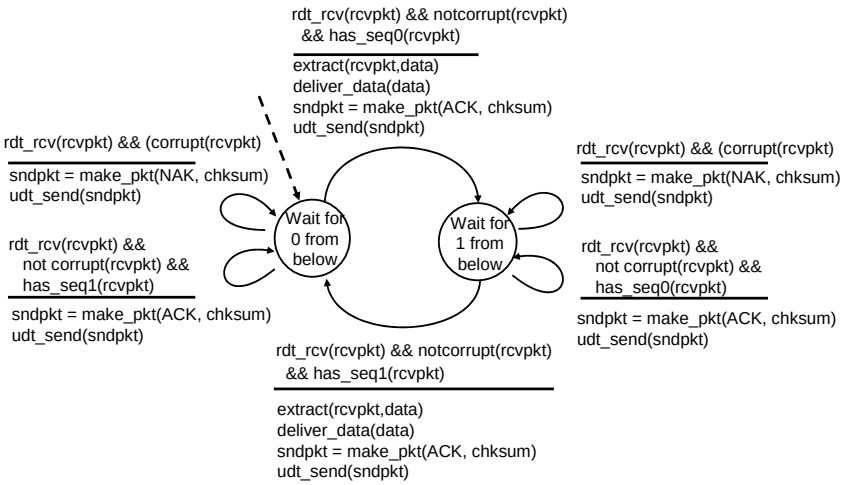
3. (15 pts) Recall protocol rdt 2.1 learnt in class (finite state machines given below). In this protocol both ACKs and NAKs are sent; each *data packet* contains a sequence number 0 or 1, but ACKs/NAKs do not contain sequence numbers. Also, recall that in designing this protocol we assumed that data packets and ACKs/NAKs are never lost but that they can arrive corrupted at their intended recipient. For the purpose of this problem assume that packets sent at time t always arrive at the other side at time t+1.

rdt2.1: sender, handles garbled ACK/NAKs



Comp 361, Spring 2005 3: Transport Layer 1

rdt2.1: receiver, handles garbled ACK/NAKs



Comp 361, Spring 2005 3: Transport Layer 32

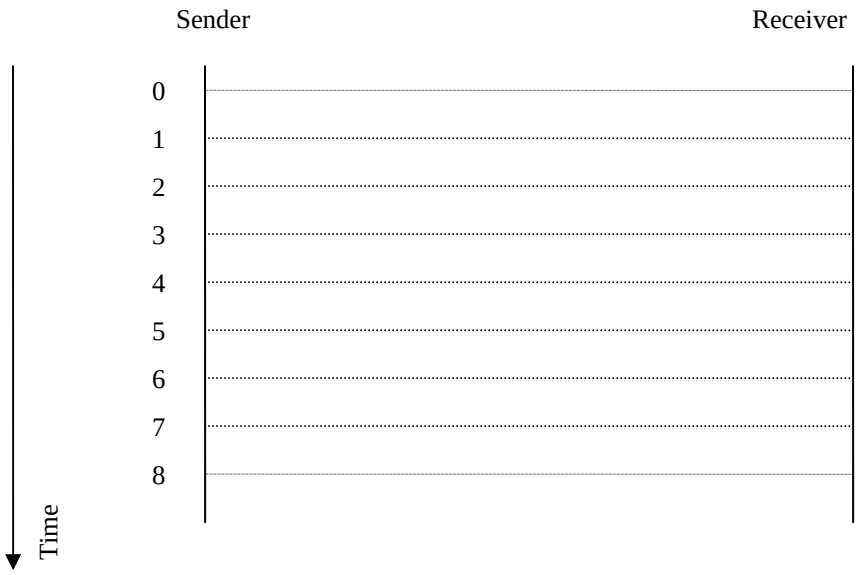
 Student ID_____

In our discussion in class we pointed out that

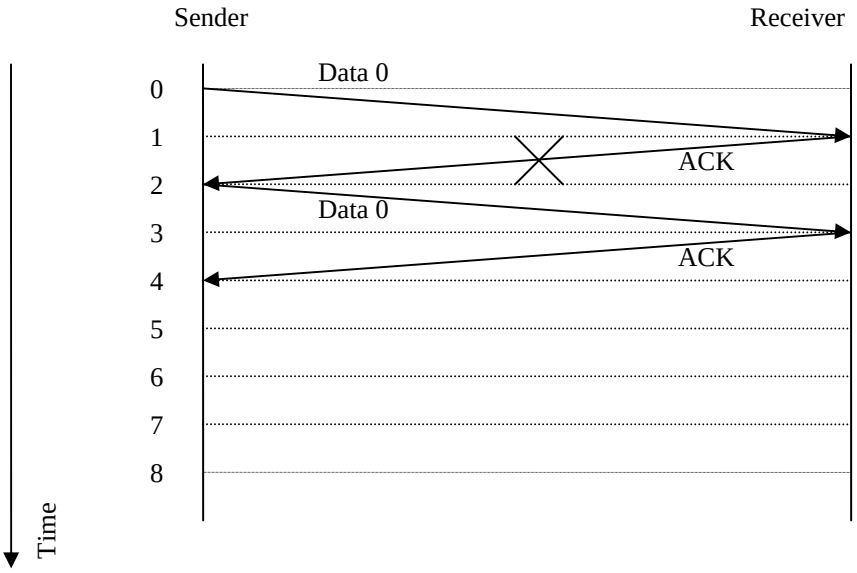
> If the receiver receives a data packet that is valid and has the same sequence number as the last correctly received packet, then the receiver should send an ACK.

Construct a scenario in which this can happen. That is, on the chart below, start from the beginning of the session and draw the sender and receiver sending data packets and ACK/NAKs back and forth under rdt 2.1 in a way that results in a receiver receiving a data packet that is valid and has the same sequence number as the last correctly received packet. Then draw the receiver sending an ACK.

For every packet (data or ACK/NAK) sent make very clear whether the packet arrives properly or arrives garbled (corrupted). Note that it is only necessary to construct a proper scenario. Your proper scenario might not need all 8 time units drawn below.



 Student ID_____



All packets sent during this session are received properly EXCEPT for the ACK sent by the receiver at time 1. That ACK was corrupted during transmission.

4. (10 pts)
You are given the three bytes below. Calculate their checksum (assuming that you are using 8 bit checksums) and write it in the appropriate space.

Byte 1	1 0 0 0 1 0 0 1
Byte 2	1 1 0 1 0 0 1 1
Byte 3	1 0 1 1 1 0 0 0
Checksum	**1 1 1 0 1 0 0 1**

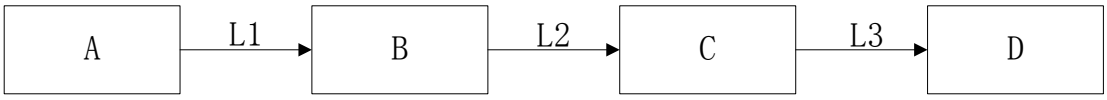
5. (5 pts) In class we described http as a *pull*-protocol and smtp as a *push*-protocol. Why did we use these terms to describe these protocols?

To answer this question you must first (a) describe what a push-protocol is and what a pull-protocol is and then (b) explain why http is a pull-protocol and smtp is a push-protocol.

A pull-protocol is one in which the client "pulls" data from the server, e.g., by requesting it. http is classic case of a pull-protocol since, after initiating the connection, the client requests data from the server which is then sent.

A push-protocol is one in which the client "pushes" data over to the server.
Smtp is a classic case of a push protocol since after initiating the connection, it "pushes" the email-message over to the mail server.

6. (15 points)
Transmission/Propagation
Consider sending a file of **40960 bits** from A to D over the network below, consisting of four nodes and three links.



(a) First assume that the network is a packet-switched network and that the file is split into packets each containing **1024 bits** of data. Each packet is then given an additional **64 bits** of header.

Each *link* transmits at **512 bps** (bits per second). Assume that propagation delay on the links is negligible

How much time does it take from when the first bit of the file leaves A to when the last bit arrives at D? Explain how you derived your answer.

(b) Next, suppose that the network is a circuit-switched network. Further suppose that the transmission rate of the *circuit* between source A and destination D is **1024 bps** and that propagation delay on the links is negligible

Assuming **2 seconds** set-up time to set up the circuit and **512 bits** of header appended to the entire file, how long does it take from when the first bit of the file leaves A to when the last bit arrives at D? Explain how you derived your answer.

a) This is very similar to problem 2 of the week 3 tutorial; as done ther,e we can derive time to be (Q+M-1)(L+h)/R where

F= File size = 40960 bits
L = Packet size (data) = 1024 bits
M= No of packets = F/L =40960/1024 = 40
Q= No of links = 3
h = header size = 64 bits
R = transmission speed = 512bps

One full packet has size L+h, so it takes time (L+h)/R to transmit it over one link.
The 1st packet therefore arrives at the final destination at time Q(L+h)/R/
Since 2nd packet started one time unit after 1st packet it arrives at final destination at time (Q+1)(L+h)/R
Since 3rd packet started one time unit after 2n packet it arrives at time (Q+2)(L+h)/R.
Continuing in this fashion we see that final packet arrives at time (Q+M-1)(L+h)/R

*So solution is 42*1088/512 s = 89.25 sec*

b. Time is time to setup + time to send file + header over link which is t_s *+ (F+h)/R where*

t_s *= setup time = 2 sec*
F = file size = 40960 bits
h = header size = 512 bits
R = transmission speed = 1024 bps

So solution is

2 + (40960+512)/1024 sec = 42.5 sec

7. (20 pts) Consider transferring an enormous file of L bytes from host A to host B. Assume an MSS of 2048 bytes.

A. What is the maximum value of L such that TCP sequence numbers are not exhausted? Recall that the TCP sequence number field has four bytes.

B. For the L you obtain in (a), find how long it takes to transmit the file. Assume that a total of 64 bytes of transport, network, and data-link header are added to each segment before the resulting packet is sent out over a 10 Mbps (10^7 bits per second) link. Ignore flow control and congestion control, so A can pump out the segments back-to-back and continuously. (note: if you do not have a calculator you can write the answer out as *a/b* where *a* and *b* are integers).

A. Recall that TCP sequence numbers refer to bytes and not packets. Therefore, packet size is no important here. The maximum value of L is simply the maximum no of bytes allowed by TCP which is 2^{32} *bytes.*

B.

The file is split into $M= 2^{32}/2048 = 2^{21}$ *packets.*
Each packet has size S= (2048+64)=2112 total bytes

Therefore totals # of bytes transmitted is
$B=M*S =2112 * 2^{21}$ *bytes or* $B= 2112 * 2^{24}$ *bits.*

Total transmission time is therefore $B/10^7$ *sec*
which is approximately = 3543 sec =59.05 min

8. (16pts) Suppose that the top-level domain (TLD) server for the .edu domain contains the following two records:

(umass.edu, dns.umass.edu, NS)
(dns.umass.edu, 128.119.40.111, A)

Also, suppose that machine dns.umass.edu has the following resource record:
(gaia.cs.umass.edu, 128.119.40.200, A)

Now suppose that you are working at HKUST and your local host machine needs to resolve the name *gaia.cs.umass.edu*. The first step is for your local host to pass the request to the local name server. Your local name server then performs a sequence of queries and, after completion, sends your local host the resolved IP address. Describe the sequence of queries between your local name server and the other servers involved in the query process. List all of the servers involved and who talks with whom. What is the number of communications needed to obtain the IP address of gaia.cs.umass.edu?

You should assume that there are no IP addresses cached at any of the intermediate servers. The only records kept at intermediate servers are those that are required to be kept there. You should also assume that the processing of all queries is **iterative** (i.e., **not recursive)**.

Let LNS be the local name server
ROOT be a root server
TLD be the Top Level domain server for .edu

Then the sequence of operations is

1. *LNS asks ROOT for address of* ***gaia.cs.umass.edu***
2. *ROOT send LNS address of TLD*
3. *LNS asks TLD for address of* ***gaia.cs.umass.edu***
4. *TLD sends LNS address of* ***dns.umass.edu***
5. *LNS asks* ***dns.umass.edu*** *for address of* ***gaia.cs.umass.edu***
6. ***dns.umass.edu*** *sends LNS address of* ***gaia.cs.umass.edu***

At this point the LNS server sends your local host the address of ***gaia.cs.umass.edu***

Note: for full credit you needed to mention ***both*** *the root server* ***and*** *the edu TLD server.*