Midterm Exam for Computer Networks (CNT 6215)

Fall 2011

>>> SOLUTIONS <<<

Welcome to the Midterm Exam for *Computer Networks*. Read each problem carefully. There are eight required problems (each worth 12 points – you get 4 points for correctly following these instructions). There is also an additional extra credit question worth 10 points. You may have with you a calculator, pencils and/or pens, erasers, blank paper, and one 8.5 x 11 inch "formula sheet". On this formula sheet you may have anything you want (definitions, formulas, homework answers, old exam answers, etc.) as **handwritten by you in pencil or ink** on both sides of the sheet. Photocopies, scans, or computer generated and/or printed text are not allowed on this sheet. Note to tablet (iPad, etc.) users – you may **not** print-out your handwritten text for the formula sheet. You have 75 minutes for this exam. **Please use a separate sheet of paper for the answer to each question**. Good luck and be sure to show your work!

Problem #1

Each sub-problem worth 4 points.

Answer the following questions regarding the basics principles and concepts of networks.

a) Sketch the 5-layer Internet protocol model – identify the key protocols for each layer. In one sentence, or less, describe the function (or purpose) of each layer.

```
    +--------------+
  5 | Application  | -- User application
    +--------------+
  4 |     TCP      | -- Provides guaranteed delivery of data
    +--------------+
  3 |      IP      | -- Responsible for moving packets from end to end
    +--------------+
  2 |     Link     | -- Provides point-to-point data transport
    +--------------+
  1 |   Physical   | -- Transmission of bit stream
    +--------------+
```

b) State the top-level and second-level goals (or design philosophies) of the Internet as described by Clark.

Top level goal is "effective technique for multiplexed utilization of existing interconnected networks". Second level goals are 1) communications continue despite loss of networks or gateways, 2) support multiple types of communication service, 3) accommodate a variety of networks, 4) permit distributed management, 5) be cost effective, 6) easy host attachment, and 7) resource use accountable

c) What are the four causes of packet delay in the Internet? What are the two causes of packet loss?

The four causes of delay are propagation, transmission, processing, and queueing. The two causes of loss are electrical noise and buffer overflow.

Problem #2

Each sub-problem worth 6 points. Each field in (a) is 1 pt.

Answer the following questions regarding HTTP.

a) Sketch the packet format (that is, identify the position and name of all headers and trailers) of an HTTP GET request sent on an Ethernet link.

```
+--------------+--------+---------+----------+-----------------------------+
| Ethernet hdr | IP hdr | TCP hdr | HTTP hdr | HTTP payload | Ethernet trlr |
+--------------+--------+---------+----------+-----------------------------+
```

b) Describe the contents of an HTTP GET header. Give two possible response codes to a GET.

The HTTP GET header starts with a string `GET /filename HTTP/1.0` where filename is the URI of the file to be downloaded. The 1.0 could also be 1.1 (1.0 and 1.1 are the two versions of HTTP). Following this string are optional header fields for Accept (files types to accept), Cookies, Date (date sent), Host (domain name of host), and many other optional fields.

Two possible response codes are 200 for everything OK and 404 for page not found.

Problem #3

SAW formula is 3 pts, SW formula is 3 pts. Calculations are 6 pts.

Consider a point-to-point 100 Mb/s cross-country link (so, 3000 miles point-to-point). Assume that packets are 500 bytes in length. What is the maximum achievable utilization for this link assuming a single sender if it uses a Stop-And-Wait (SAW) protocol for reliable data transfer? What window size would be needed for a Sliding Window (SW) protocol to achieve 100% link utilization? Make reasonable assumptions as needed (and clearly state them).

The key formulas are (for *W* = window size in number of frames, or packets),

$$U_{SAW} = \frac{t_{fr}}{2 \cdot t_{pr} + t_{fr}}$$ and

$$U_{SW} = \max\left(1, \frac{W \cdot t_{fr}}{2 \cdot t_{pr} + t_{fr}}\right)$$

where we assume that t_{proc} and t_{ack} are negligible. For our case we have t_{pr} = 15 ms (for 5 ms per mile propagation) and t_{fr} = 0.04 ms (500 bytes divided by 100 Mb/s). Thus,

$$U_{SAW} = \frac{0.04}{2 \cdot 15 + 0.04} = 0.133\%$$ (which is really horrible!).

To find the necessary window size for 100% link utilization we solve for:

$$1 = \frac{W \cdot t_{fr}}{2 \cdot t_{pr} + t_{fr}}$$

for which we get *W* = 751.

Problem #4

Formulas are 6 pts, operation is 6 pts.

How does TCP sets its RTO value? You may describe this in words, but best is to give the key formulas and describe how the formulas are used by TCP.

In words… TCP samples the time from a packet send to the receipt of its ACK as RTT. Successive RTT samples are smoothed using exponential smoothing. The error of the smoothed RTT is also determined and used to compute a smooth deviance value. The smooth RTT and smoothed deviance values are used to additively compute the RTO (Retransmission Timeout Value). RTT samples are only taken for non-transmit packets. When a retransmit occurs, the RTO is "backed-off" (doubled).

The formulas…

RTO = Retransmit time-out
RTT(i) = observed RTT for sample i

SRTT(k) = smoothed RTT for first k segments
SERR(k) = smoothed error for first k segments
SDEV(k) = smooth deviance for first k segments

SRTT(k+1) = (1–g)*SRTT(k) + g*RTT(k+1)
SERR(k+1) = RTT(k+1) – SRTT(k)
SDEV(k+1) = (1–h)*SDEV(k) + abs(SERR(k+1))
RTO(k+1) = SRTT(k+1) + f*SDEV(k+1)

g = 1/8, h = 1/4, f = 4

On retransmit use exponential back-off (doubling) of RTO and sample only on non-retransmits

Problem #5 Roughly 3 pts for each step.

Show that the maximum efficiency (or goodput) of an ALOHA channel is approximately 18%. Explain why and how slotted ALOHA doubles the maximum achievable efficiency. Clearly state the assumptions necessary to make the analysis tractable.

Define k = number of users, λ = average rate of packets from a single user (assuming Poisson arrivals), τ = duration of each packet, r = overall new packet rate, and R = new plus retransmit packet rate. Let $r\tau$ = channel utilization and $R\tau$ = channel traffic.

Given Poisson arrivals, the time between arrivals follows an exponential distribution. This (Poisson arrivals) is the key assumption and applies also to the retry packets.

The collision window for unslotted Aloha is 2τ, so the probability that a given packet will be repeated is

$$1 - e^{-2R\tau}$$

The average number of retransmissions per time is $R(1 - e^{-2R\tau})$, so then for (unit time)

$$R = r + R(1 - e^{-2R\tau})$$

which reduces to

$$r\tau = R\tau e^{-2R\tau}$$

If we solve for $R\tau$ to mazimize rt we get $r\tau = 1/2e$, which is approximately 18%.

Slotted ALOHA reduce the window of vulnerability from 2τ to τ resulting in a doubling of maximum $r\tau$ (to 1/e, which is approximately 36%).

Problem #6 Each sub-problem worth 4 points.

Answer the following questions regarding performance evaluation and experimental design.

a) What are the four ways to study a system?

The four ways are, 1) do a mathematical analysis to get a formula to predict performance, 2) build a simulation model and experiment on it, 3) build a prototype and experiment on it, and 4) build the actual system and experiment on it.

b) What is a model (precisely define it) and what is the goal of building a model?

"A model is a representation (physical, logical, or functional) that mimics another object under study" (Molloy 1989). The goal of a building a model is to be able to understand the system under study – to be able to predict its performance.

c) Consider a system with two factors A and B. A has levels 1, 2, and 3. B has levels 1 and 2. Give both a full-factorial and simple design for an experimental study of this system.

A full factorial design would have 6 experiments for all possible combinations of A and B factor levels as follows (A1, B1), (A1, B2), (A2, B1), (A2, B2), (A3, B1), and (A3, B2). A simple design would have 4 experiments for all possible factors values of A and B, one possible such design is (A1, B1), (A2, B1), (A3, B1), (A1, B2)

Problem #7

Each sub-problem worth 3 points.

Answer the following questions regarding Markov models.

a) Precisely define the Markov property?

Past history is completely summarized by the specification of a current state.

b) Which continuous distribution exhibits the memoryless property? Prove it.

The exponential distribution. The memoryless property is $\Pr[x < T + t \mid x > T] = \Pr[x < t]$. We can prove as follows:

$$\Pr[x < T + t \mid x > T] = \frac{\Pr[(x < T + t) \cap (x > T)]}{\Pr[x > T]} = \frac{\Pr[(x < T + t) - \Pr(x < T)]}{\Pr[x > T]} = \frac{-e^{-(T+t)\lambda} + e^{-T\lambda}}{e^{-T\lambda}} = 1 - e^{-\lambda T}$$

c) What is a P matrix? What is a Q matrix? How can a Q matrix be converted into a P matrix?

A P matrix is the matrix showing all <u>state transition probabilities for a DMTC</u>. A Q matrix is the matrix showing all <u>state transition rates for a CMTC</u>.

d) What is a feasible/practical method to solve a large (many hundreds of states) P matrix?

<u>An iterative approach is probably the most feasible approach</u> (to iterate on $\pi = \pi P$) – an iterative approach that does not store or operate on zeroes can be very efficient (in both memory use and processing time).

Problem #8

Q matrix is 6 points, equations are 4 points, and numerical results are 2 points.

Assume that a Wi-Fi interface has three possible operational states which are SLEEP, ON-ACTIVE, and ON-IDLE. Assume that for each state the transition rate out of it is independent of past history. The transition rates are as follows: SLEEP to ON-ACTIVE is 1 time per hour, SLEEP to ON-IDLE is 1 time per hour, ON-ACTIVE to ON-IDLE is 2 times per hour, and ON-IDLE to sleep is 6 times per hour. For a randomly chosen time point, what is the probability that the interface is in the SLEEP, ON-ACTIVE, and ON-IDLE states?

As our first step we write the Q matrix (state 0 is SLEEP, 1 is ON-IDLE, and 2 is ON-ACTIVE):

$$Q = \begin{bmatrix} -2 & 1 & 1 \\ 0 & -2 & 2 \\ 6 & 0 & -6 \end{bmatrix}$$

We then write the three equations in three unknowns and solve for the steady state probabilities as:

$$-2\pi_0 + 6\pi_2 = 0$$
$$\pi_0 - 2\pi_1 = 0$$
$$\pi_0 + \pi_1 + \pi_2 = 1$$

Note that one of the equations is $\pi_0 + \pi_1 + \pi_2 = 1$ to break the linear dependence of the equations. We solve and get $\pi_0 = 6/11, \pi_1 = 3/11,$ and $\pi_2 = 2/11.$

Extra Credit

1 pt for each minor bug found and 4 pts for major bug.

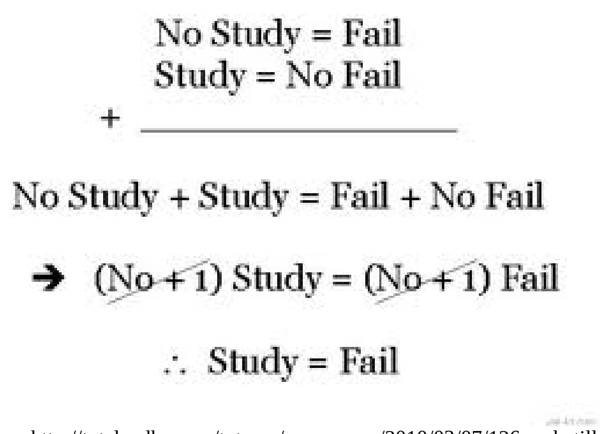
The appendix contains a program intended to measure the response time for an HTTP HEAD, the program is `httpHead.c`. The program contains several bugs – you are to identify and explain how to fix these bugs. There is one very major bug (that is worth more points that the other bugs) for you to find. Note that the program compiles and links successfully (using bcc32 in Windows, at least). Write your answer directly on the code listing (i.e., in the appendix) of this exam handout.

The following are the bugs:

- `#define IP_ADDR "grad.csee.usf.edu"` is wrong (should be an IP address, not host name)
- `client_s = socket(AF_INET, SOCK_DGRAM, 0);` should contain SOCK_STREAMS
- `if (retcode == 0)` following connect should be “!=”
- `HEAD_STRING` is never copied into `out_buf[]` – need a `strcpy(out_buf, HEAD_STRING);`
- Really should check for a `(ret_code == -1)` in receive loop
- Not closing the open socket at end of use – add `closesocket(server_s);` before `ftime(&stop);`
- **The major bug is…** it is not sufficient to time only one HEAD method since the complete time will likely be less than (or, if not less than, then only slightly more than) a single 10 ms clock tick on Windows, at least for lock servers. Need to put a loop around the program and time 100 or 1000 HEADs.

Humor

Hmm…. maybe studying is not such a good idea after all…



No Study = Fail
Study = No Fail
+ ____________

No Study + Study = Fail + No Fail

➔ $(\cancel{No + 1})$ Study = $(\cancel{No + 1})$ Fail

∴ Study = Fail

From: http://totalgadha.com/tgtown/mangoman/2010/03/07/126-and-still-counting/

I hope that everyone did well ☺

Appendix – httpHead.c with errors

```
<< SNIP SNIP – header removed >>
#include <stdio.h>          // Needed for printf()
#include <stdlib.h>         // Needed for exit()
#include <string.h>         // Needed for strcpy() and strlen()
#include <windows.h>        // Needed for all Winsock stuff
#include <sys\timeb.h>      // Needed for ftime() and timeb structure

#define  BUF_SIZE            4096     // Buffer size
#define  PORT_NUM              80     // Web servers are at port 80
#define  IP_ADDR    "grad.csee.usf.edu" // IP address of www.csee.usf.edu
#define  HEAD_STRING "HEAD /~christen/index.html HTTP/1.0\n\n"  // HEAD string

void main(void)
{
  WORD wVersionRequested = MAKEWORD(1,1);    // Stuff for WSA functions
  WSADATA wsaData;                           // Stuff for WSA functions
  unsigned int         server_s;             // Server socket descriptor
  struct sockaddr_in   server_addr;          // Server Internet address
  char                 out_buf[BUF_SIZE];    // Output buffer for GET request
  char                 in_buf[BUF_SIZE];     // Input buffer for response
  unsigned int         retcode;              // Return code
  unsigned int         i;                    // Loop counter
  struct timeb         start, stop;          // Start and stop time structures
  double               elapsed;              // Elapsed time in seconds

  // This stuff initializes winsock
  WSAStartup(wVersionRequested, &wsaData);

  // Start timing
  ftime(&start);

  // Create a socket
  server_s = socket(AF_INET, SOCK_DGRAM, 0);

  // Fill-in the Web server socket's address information
  server_addr.sin_family = AF_INET;                 // Address family to use
  server_addr.sin_port = htons(PORT_NUM);           // Port num to use
  server_addr.sin_addr.s_addr = inet_addr(IP_ADDR); // IP address to use

  // Do a connect (connect() blocks)
  retcode = connect(server_s, (struct sockaddr *)&server_addr,
                    sizeof(server_addr));
  if (retcode == 0)
  {
    printf("ERROR - connect() failed \n");
    exit(1);
  }

  // Send a HEAD to the Web server
  send(server_s, out_buf, strlen(out_buf), 0);

  // Receive from the Web server
  retcode = recv(server_s, in_buf, BUF_SIZE, 0);
  while (retcode != 0)
    retcode = recv(server_s, in_buf, BUF_SIZE, 0);

  // Stop timing, compute and output elapsed time
  ftime(&stop);
  elapsed=((double) stop.time + ((double) stop.millitm * 0.001)) -
          ((double) start.time + ((double) start.millitm * 0.001));
  printf("  Elapsed time = %f \n", elapsed);

  // Clean-up for winsock
  WSACleanup();
}
```