

Carnegie Mellon  
Computer Science Department  
**15-441 Spring 2009**  
**Final**

Name: \_\_\_\_\_

Andrew ID: \_\_\_\_\_

**INSTRUCTIONS:**

There are 19 pages (numbered at the bottom). Make sure you have all of them.

Please write your name on this cover and at the top of each page in this booklet **except the last**.

If you find a question ambiguous, be sure to write down any assumptions you make.

It is better to partially answer a question than to not attempt it at all.

Be clear and concise. Limit your answers to the space provided.

A	/ 7
B	/ 20
C	/ 12
D	/ 8
E	/ 15
F	/ 12
G	/ 15
H	/ 12
I	/ 9
J	/ 10
Total	/ 120

## A TCP versus Ethernet versus UDP

1. Please indicate on the right whether the following statements are Correct or False.

The goal of TCP slow start is to avoid queue overflow in routers

TCP has stronger detection of bit corruption than UDP

Ethernet uses stop and wait error control

UDP flow control does not use sequence numbers

Ethernet has stronger detection of bit corruption than UDP

Ethernet does not use slow start

TCP fast retransmit avoids timeouts

TCP fast recovery avoids timeout

UDP guarantees in order delivery of packets

<b>Solution:</b> False, False, False, False, True, True, True, True, False
--

## B Switching versus Routing

2. You are running a private local area network based on switched Ethernet. Your network has no routers. Since you took 441 at CMU, you are asked to design the network.

3. The first design question is whether you need IP, considering that you do not have routers?

(a) Give one advantage of not running IP (i.e. transport runs directly on the Ethernet)?

**Solution:** Saves header space.

(b) Give one disadvantage of not running IP?

**Solution:** Will be nearly impossible to use higher layer protocols and to use existing applications.

4. You meet with your boss to go over your analysis, and he decides to skip the IP layer. He now wants you to evaluate the message overhead of two different algorithms for establishing forwarding tables in the switches. The first algorithm is the standard spanning tree algorithm. The second algorithm is a proprietary link state routing algorithm. Please use the following variables:

- N - number of client nodes
- S - number of switches
- L - number of links connecting switches
- D - diameter of the network (switches only)
- T - number of links in the spanning tree
- O - the average out degree of the switches (number of outgoing links to other switches)

Note that  $S \times O = 2 \times L$ . The message overhead is defined as the cost of sending a message over one link, i.e. one transmission by one node over one link.

(a) Please calculate the maximum message overhead of establishing the spanning tree across the switches.

**Solution:** Each switch sends a message on all its outgoing links; it may take up to D iterations.  
 $D \times S \times O$

(b) In order to establish the forwarding tables, each client node sends a short packet to a randomly generate MAC address. Please calculate the message overhead.

**Solution:** The random MAC address will likely not exist so the message needs to be broadcasted across the spanning tree  $N \times (T + N)$

5. The company building the switches also developed a custom link state algorithm that uses the MAC addresses (standard IEEE 48 bit addresses without hierarchy). The first part of the algorithm consists of all nodes (clients and switches) identifying themselves to all their neighbors using a one way HELLO messages. The second phase consists of the switches executing a standard link state algorithm.

(a) What is the message overhead of the HELLO phase?

**Solution:** All switches send on all their outgoing links (to other switches),  $S \times O + 2 \times N$

(b) What is the message overhead of the link state phase?

**Solution:** Each switch will flood its link state, where for each flooding operation, each switch floods all its links except the incoming link:  $S \times (S \times (O - 1) + 1)$

6. You need to formulate a recommendation for your boss for what algorithm to use based on message overhead. The set up costs look pretty scary, so you focus your analysis on the costs of the operation that needs to be performed periodically: maintaining the spanning tree and the routing tables. Please calculate the recurring cost and formulate your recommendation:

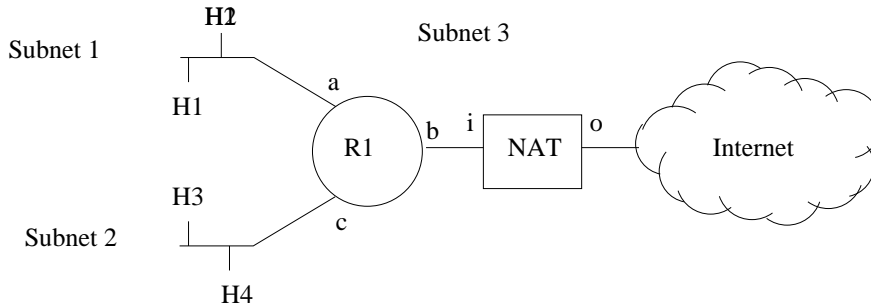
**Solution:** Spanning tree:  $S \times O$

Link state:  $S \times (S \times (O - 1) + 1)$

Clearly spanning tree is more efficient on the long run.

## C IP Layer

7. Sally Student runs a large network at her house and wants to subnet it to separate her work computers from the network that controls her toaster and fish tanks. She has purchased a NAT box, and divided her network up as follows:



Her ISP has given her an IP address that she assigns to NAT-o (the outside or “o” interface on the NAT box). Sally knows that RFC1918 specifies three different address ranges that she could use for private addresses inside her home:

10.0.0.0 - 10.255.255.255 (10/8 prefix)  
 172.16.0.0 - 172.31.255.255 (172.16/12 prefix)  
 192.168.0.0 - 192.168.255.255 (192.168/16 prefix)

- (a) Your job is to assign addresses to the subnets, routers, and NAT box inside her house. Use addresses from the 10.x netblock.

Subnet	Number	Netmask
Subnet 1		
Subnet 2		
Subnet 3		

Interface	IP Address
H1	
H2	
H3	
H4	
R1a	
R1b	
R1c	
NAT-i	

**Solution:** There are many solutions, here is one:

Subnet	Number	Netmask
Subnet 1	10.0.1.0	255.255.255.0
Subnet 2	10.0.2.0	255.255.255.0
Subnet 3	10.0.3.0	255.255.255.0

Interface	IP Address
H1	10.0.1.1
H2	10.0.1.2
H3	10.0.2.1
H4	10.0.2.2
R1a	10.0.1.3
R1b	10.0.3.1
R1c	10.0.2.3
NAT-i	10.0.3.2

Note that broadcast addresses and network addresses can NOT be assigned to interfaces.

- (b) Give one reason that wide-spread deployment of IPv6 would let Sally get rid of her NAT device.

**Solution:** IPv6 would allow Sally to place a virtually unlimited number of devices on her home network without special IP address allocation from her provider.

- (c) Give one reason that Sally might want to continue using her NAPT even if she could use IPv6.

**Solution:**  
A NAPT provides a measure of security by not permitting unsolicited inbound connections.

- (d) Assuming that the NAT box has no special support for any protocols, and merely translates TCP and IP ports and addresses, give an example of an application that would not work through this NAT, and very briefly explain why.

**Solution:** FTP would not work through this NAPT because it requires that the server open a connection back to the client. (Passive mode FTP would work—it has the client open the connection instead). Any protocol that embeds IP or TCP-layer information in the application stream is likely to be broken by a basic NAT.

## D Virtual Circuits

8. The following question focuses on virtual circuits, how we can benefit from them, and how they are implemented.
- (a) The delivery model of IP is known as best effort delivery service. This differs strongly from the delivery model provided by *virtual circuits*. Compare and contrast the guarantees provided by IP and virtual circuits in terms of: latency, bandwidth, reliability.

**Solution:**

IP is best effort delivery. There are absolutely no guarantees on latency, bandwidth, or reliability. It is all dependent on the route the information takes and the current state of the network (e.g., congested). Latency can be extremely variable from packet to packet, bandwidth can change drastically, and there is no reliability mechanism.

On the other hand, virtual circuits can provide guarantees through reservation at the routers in the network. For example, virtual circuits can guarantee 1Mbps to 10 different connections, if the bandwidth at the router is 10Mbps. Since the path in the network is fixed and there is less buffering (link is often underutilized), better latency guarantees can be provided in terms of variability. Since the path is relatively fixed, there will be very little jitter in the latency which is good for streaming services. Virtual circuits do not necessarily explicitly implement reliability, I will take most answers for this.

- (b) Why is there additional latency to get the first data packet across from the source to the destination in a virtual circuit network than an IP network? What is this latency in networking terminology? (think end to end, not an actual number)

**Solution:**

There is additional latency in a virtual circuit network because of the connection setup stage. The connection setup is used to reserve the bandwidth at each router along the way. The connection setup delay is 1 RTT (round trip time), which will vary based on the distance of the end points.

- (c) What is the difference between the information needed to route an IP packet through a network, and a packet in a virtual circuit switched network? Is routing with virtual circuits more simple, or more complex, than IP routing?

**Solution:**

Routing with IP requires a source and destination field in the header so that the packet can get to its destination (destination field), and the destination knows who the packet came from (source field). Since there is a connection setup stage with virtual circuits, each packet only needs to carry a virtual circuit ID tag to get from the source to the destination, and the destination knows who the source was based on this tag also.

Routing in virtual circuit networks is more simple than IP networks. The router only needs to do a 1:1 comparison of the tag to determine the output port. To determine the output port in an IP router, shortest prefix matching must be done, which is more complex in terms of processing.

- (d) Throughout the second half of 441, you have learned plenty about a transport layer protocol which implements virtual circuit like functionality on top of IP. What is this protocol, and how is it similar to virtual circuits and how does it differ?

**Solution:**

The one, and only, TCP! It has a connection setup phase, it transfers packets reliably between the two end hosts, and it has a connection teardown stage. It differs in that it does not actually provide any latency and bandwidth guarantees, like virtual circuits do.

## E Crypto Schmipto

9. Below are several scenarios describing simple uses of cryptographic schemes we have covered in 441. For each scenario, circle “correct” if the scenario describes a valid use of the mechanism as described in class. Otherwise circle “incorrect” and write one sentence explaining the vulnerability it exposes.

- (a) Bruce wants to send the top-secret solutions for the final exam to Peter, but Peter’s email server is down. However, Peter promised to check the 441 bboard regularly to see if Bruce posted any messages for him. During the first day of class, Peter gave everyone at the lecture (including Bruce) his public key  $K_{Peter}$ . Only Peter knows his private key,  $K_{Peter}^{-1}$ . Knowing that Peter will recognize the correct answer key based on their past discussions, Bruce encrypts the answer key with  $K_{Peter}$  and posts it to the bboard.

correct / incorrect

(4 points)

**Solution:** Correct. While a student could encrypt a fake solution set, the question specified that Peter would recognize a fake.

- (b) David wants to transmit project2 grades from his home computer to Bruce at CMU. He is worried that some enterprising 441 student may have hacked a router along the path and might modify the message to improve their grade and win the project2 contest. So when David sends a message  $M$  to Bruce, he also calculates  $H = \text{Hash}(M)$  and appends  $H$  to the message. Bruce receives  $M$  and  $H$ , and calculates  $H' = \text{Hash}(M)$ , only accepting the message as valid if  $H' = H$ . You can assume that Hash is a secure hash function that is one-way, collision resistant, and pre-image resistant.

correct / incorrect

(4 points)

**Solution:** Incorrect. The enterprising student could change the contents and recalculate the hash. Bruce would never know that anything had changed.

- (c) George and Jacob both share a secret key with a Key Distribution Center (KDC). We call these keys  $K_{George,KDC}$  and  $K_{Jacob,KDC}$  respectively. George wants to establish a shared symmetric key with Jacob, so George authenticates to the KDC using  $K_{George,KDC}$  and the KDC replies with  $Encrypt_{K_{George,KDC}}(K_{Jacob,KDC})$ . George and Jacob then communicate using the shared secret key  $K_{Jacob,KDC}$ .

correct / incorrect

(4 points)

**Solution:** Incorrect. George would now be able to pretend to be Jacob because he has the secret key that Jacob shares with the KDC.

10. Bruce and Peter need to communicate to decide which TA is going to grade the next homework. They have a shared secret,  $K_{Prof}$  that allows them to create unforgeable message authentication codes (MAC) so that Bruce can verify that Peter did in fact create any message that is received. Bruce and Peter have a simple protocol: Bruce sends a “Who grades HWX?” message to Peter in plain text, and Peter replies with one of three messages:  $M1 = MAC_{K_{Prof}}(\text{“George”})$ ,  $M2 = MAC_{K_{Prof}}(\text{“David”})$ , or  $M3 = MAC_{K_{Prof}}(\text{“Jacob”})$ . When Bruce receives either  $M1$ ,  $M2$ , or  $M3$ , he verifies the MAC using  $K_{Prof}$  and knows who will grade the next homework.

- (a) This protocol is insecure. A malicious TA on a router between Bruce and Peter may be able to avoid ever having to grade a homework! In one sentence, describe the attack. (5 points)

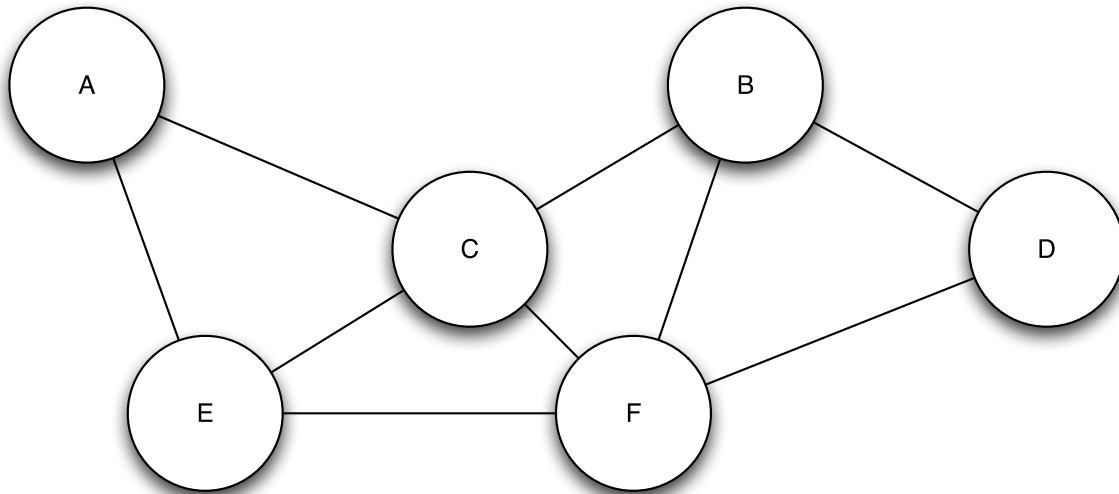
**Solution:** It is subject to a replay attack. The TA could replay an earlier answer for a different TA's name.

- (b) What simple change to the above protocol could defend against this attack? (2 points)

**Solution:** Use a nonce. Along with the "Who grades HWX" message, Bruce should also send a random string that must be included in the MAC to ensure that the answer is unique.

## F Wireless Networks

11. Consider the following wireless network arrangement. Nodes connected with a line can receive transmissions from one another; unconnected nodes are too far to communicate. Link strengths are proportional to the length of the line.



- (a) Recall that a hidden terminal problem can lead to unexpected collisions. What are some nodes which might encounter a hidden terminal issue in transmitting?

**Solution:** Any two nodes which connect to a common node but do not connect to one another are hidden terminals; A-C-F, E-F-D, and so on.

- (b) What are two pairs of nodes which should be able to transmit to one another simultaneously, but cannot, because of an exposed terminal problem?

**Solution:** C to A and B to D, for example: C and B can see each other, but not A-B or C-D.

- (c) Suppose this is an ad-hoc network using dynamic source routing. Node A wishes to send a packet to node D. All nodes have empty route caches; therefore, the first thing to happen is a route discovery step. How does route discovery in a DSR network propagate through the network?

**Solution:** A will send an RR packet to E and C, looking to reach D; it will then be rebroadcast by B and F, and then reach D. Once a route is found, a reply will propagate from D back to A, and intermediate nodes will add this information to their route caches. Then, A can send a suitably addressed packet across the network.

- (d) The DSR algorithm presented in lecture does not specify what should occur if two routes with the same number of hops exist. Suppose we change the discovery algorithm such that when a node receives a route request, it waits very briefly (for just enough time to receive another RR from another link, if any is arriving near-simultaneously) and then determines which route to offer when it rebroadcasts or replies to the request. Do not consider collisions; assume all packets are properly received.

How might a node choose 'optimally' which request to forward? What does this optimize for? What route would your algorithm decide upon to route a message from A to D?

**Solution:** Only forward the “strongest” request. This should mean that the most direct path will be taken. In the above network, F will forward from C, not E; D will choose the path via B and C.

## G Socket Programming

12. In recitation, we discussed some of the tradeoffs between nonblocking network I/O and threads. Network assignments in 15-213 generally use threads; in 15-441, your projects were based on nonblocking I/O. One technique for high-performance applications is *user-space threading*: writing utility functions to create an environment in which the nonblocking system calls “act like” blocking calls running in threads. This grants the same ease of expressing control flow as normal threading, but is lighter-weight than kernel threads; since user-space threading in networking applications can be made non-preemptible, it is also much less susceptible to race conditions. User-space threading can be implemented in C with some assembly-language work, or in languages supporting `call/cc` such as SML and Lisp.

Suppose you are writing such a system in C. We will not ask you to write the actual threading implementation, but instead the glue code that interfaces this to the BSD socket API. For simplicity, we will also not concern ourselves with how threads are created or destroyed. During operation, the threading system provides one function for use:

```
enum waitcondition_t { READABLE, WRITABLE };
extern void block(int fd, enum waitcondition_t condition);
```

When you call `block()` from within a thread, the calling thread does not continue executing until the given socket has become readable or writable (as by functions like `select()`).

- Task 1: Write a function `ssize_t b_recv(int socket, void *buffer, size_t length, int flags)`, which expects to be given a nonblocking socket. It should act exactly as `recv()` would act if the socket were set as blocking. You may call `block()` and the real `recv()` to implement this functionality. Make sure you handle errors not related to blocking properly, by simply returning them from `b_recv()`.

**Solution:** Something like this:

```
ssize_t b_recv(int socket, void *buffer, size_t length, int flags) {
    int rv = 0;
    while(1) {
        rv = recv(socket, buffer, length, flags);
        if (rv == EAGAIN) {
            block(socket, READABLE);
        } else {
            return rv;
        }
    }
}
```

Now, the threading library must have some way to know when a given socket becomes readable again. We will use the `select()` system call for this, just like in your applications. Recall the headers for `select()` and the `fd_set` macros:

```
void FD_CLR(fd, fd_set *fdset);
void FD_COPY(fd_set *fdset_orig, fd_set *fdset_copy);
int FD_ISSET(fd, fd_set *fdset);
void FD_SET(fd, fd_set *fdset);
void FD_ZERO(fd_set *fdset);
int select(int nfd, fd_set *readfds, fd_set *writefds,
          fd_set *errorfds, struct timeval *timeout);
```

The thread library operates as such: first, it loops through all runnable threads and runs each of them until it `block()`s. Then it calls `select()` to determine which thread IDs are ready to continue, and repeats the process. You will need to implement two helper functions for this:

```
void add_sock(int fd, enum waitcondition_t condition);
void check_sockets();
```

Calling `add_sock()` should “flag” a socket, such that future `check_sockets()` include that socket in their `select()` call. The `block()` function will call `add_sock()` before switching to another thread.

When `check_sockets()` is invoked, it calls `select()` to determine which flagged sockets are now readable or writable (as specified by the condition argument to `add_sock`). For each such socket, it invokes an function `void ready(int fd)` which will be provided by the thread library. Once activity is detected on a socket, it is unflagged; future `check_sockets()` calls will not detect it again unless it is re-added with another call to `add_sock()`. If no activity is detected on a flagged socket, however, it should remain flagged for future `check_sockets()` calls.

- Task 2: Implement `add_sock()` and `check_sockets()` according to the specification given. You do not need to handle errors in `select()`, or worry about what happens if a given socket is `add_sock()`ed again when it is already flagged. You will need to store some sort of state about which sockets are flagged between calls to `add_sock()` and `check_sockets()`; use global variables.

13. (a) Describe two advantages of using cash to complete a transaction.
- (b) Why is electronic cash covered in a course on computer networking?
- (c) As described in lecture, when RSA is used as a signature scheme, Alice can blind a coin's serial number by selecting a blinding factor  $r$ , multiplying the serial number by  $r^e$ , where  $e$  is her bank's public key, and later dividing the signed serial number by  $r$ .  
Why does Alice "blind" the serial number of the coins that she asks her bank to sign?
- (d) There's a flaw with the simple "blinding" scheme described above. Suppose that instead of blinding the serial number with one blinding factor, Alice chooses two,  $r_1$  and  $r_2$ , and then multiplies the serial number by  $r_1^e r_2^e$ . What's wrong?

**Solution:**

- (a) Transactions can take place off-line and anonymously.
- (b) Unlike ordinary cash, electronic cash can be used in transactions that take place over a network.
- (c) Alice blinds the serial number so that when the coin is spent, the bank will not know who the coin was originally issued to.
- (d) If Alice uses two blinding factors, she can create three signed coins with different serial numbers that the bank has never seen, first, the original serial number, then one in which the serial number is multiplied by  $r_1$  and the other in which the serial number is multiplied by  $r_2$ . Alice can then go and spend these three coins and the bank will treat them all as distinct and valid.

14. Freenet is a peer-to-peer file distribution system that aims to be scalable while providing some level of anonymity and deniability. In Freenet, a query contains a unique 64-bit ID, a hops-to-live count, and a SHA-1 hash of a human-readable description of the file that is sought.

## Part I

### W

Why does each query contain a unique 64-bit ID?

**Solution:**

So that a node does not process the same query more than once.

## Part II

### A

Freenet query does not contain the address of the node that originated the query. How does a response make it back to the originator?

**Solution:**

## Part III

### W

When a node receives a query, it remembers the identity of the node that sent it the query, and then (if necessary) forwards the query on. If a response comes back, it then forwards that response back to the node that sent it the query.

## Part IV

### F

Freenet suggests that files should be encrypted using the human readable description of the file as a key. Give two good reasons for using this key.

**Solution:**

First, the originator of the query knows the human readable description and will be able to decrypt the file. Second, the nodes that store the file don't see the human readable description, and hence don't know what they are storing. This perhaps protects them from culpability in some cases.

15. Skype uses a mysterious protocol to determine whether a user is logged in, where they are located, and what ports they are listening on. Suppose that you have set out to build your own peer-to-peer telephony system.

## Part V

### V

ery briefly, how could you use a system such as Chord to maintain a directory of users?

**Solution:**

Chord could be used to build a dictionary, where a user's name was used as a key, and the value associated with that key included information about the user such as whether they were currently logged on, and if so, where, and on what ports were they listening.

## Part VI

### Y

our telephony system catches on very quickly, in part because you actually pay people to use it. (At least until it is discovered that the funds are coming out of your research advisor's grant.) Soon you have tens of thousands of users worldwide. Now users are starting to complain that it takes a long time to find their friends! Give at least two good reasons why this might be.

**Solution:**

Chord requires up to  $\log n$  messages to complete a lookups, and when  $n$  gets to be in the tens of thousands, this is ten to fifteen lookups.

Second, there's no locality in chord, so when the users are located all over the world, each message may have to travel a very long distance.

Third, it might be that users are not staying on the system very long, and a there's a lot of overhead (e.g.,  $\log^2 n$  messages) when someone leaves or joins just to repair pointers in Chord, and on top of this you have to deal with moving data around.

## Part VII

### W

hen a node suddenly fails in Chord, all of the information in that node may be lost. Given that you've already got an implementation of Chord up and running, describe an easy way to provide some protection in the event that a single node fails.

**Solution:**

One possibility would be to set up two distinct instances of Chord, keeping two copies of each key-value pair, one in each instance. To be truly resilient against single-node failures, you'd want to address the problem of both instances storing the record on the same node.

Another solution would be to have each node keep a complete back-up copy of its predecessor's data.