CS 336 Final — Spring 2001

May, 2001

Name:

This is an open-book examination. You may consult the text, your notes, or any other inanimate source of information while completing this examination. You have 2 1/2 hours to complete the examination.

1. In their paper on the end-to-end argument, Saltzer, Reed and Clark summarize the impact of non-end-to-end implementations of certain forms of communications technology by saying:

   > Sometimes an incomplete version of the function provided by the communication system may be useful as a performance enhancement.

   In our discussion of the end-to-end argument, we focused on the issue or reliable delivery based on retransmission. We saw that the "performance enhancement" provided by implementing mechanisms to ensure reliable delivery within the communication system (i.e. at the data link layer) could be quite significant. In fact, without the "enhancement", end-to-end reliability mechanisms might be pointless since message delay would become so high that communications would be impractical.

   In their paper on the subject, Saltzer, Reed and Clark also discuss the implications of the basic argument to several other functions including network security. In their discussion of the issue of encryption they assume that if it were provided within the communication system it would be provided as close to end-to-end as possible. That is, if the communication system were the Internet, they would be comparing the usefulness of end-to-end encryption (i.e. encryption performed by the code of client and server applications) to encryption performed by TCP.

   This is quite different from our discussion of the end-to-end argument in the context of reliability. We compared the effectiveness of end-to-end retransmission schemes to a router-to-router retransmission scheme. So, I would like you to perform a similar comparison in the area or security/encryption. In particular, compare the effectiveness of client/server application level encryption to an encryption scheme implement between pairs of Internet routers.

   In the former case, assume that the applications involved use TCP and provide TCP with data in encrypted form expecting TCP to deliver the data to the remote application still in encrypted form. In the latter case, assume that each router shares keys with its adjacent routers so that it can decrypt each message it receives and then encrypt it appropriately with a new key before forwarding it to the next router on the path to its destination.

   (a) What (if any) forms of security/privacy can the end-to-end approach provide that the router-to-router approach can not ensure?

   (b) What (if any) forms of security/privacy can the router-to-router approach provide that the end-to-end approach can not ensure?

2. Alicia and Roberto have just invented a new alternative to traditional cryptography. Their "invention" is based on the idea of a commutative encryption scheme using secret (i.e. symmetric)

keys rather than public-private key pairs. An encryption scheme is said to be commutative if when multiple encryption steps are performed using multiple keys, decryption requires using a matching key for each of the encryption keys but allows the decryption steps to be performed in any order. Thus, if $E_A(M)$ represents the encryption of a message M using Alicia's key and $D_A(C)$ represents the process of decrypting a piece of cyphertext using Alicia's key (and similarly for R and Roberto), then

$$D_R(D_A(E_R(E_A(M)))) = D_A(D_R(E_R(E_A(M)))) = M$$

(In a non-commutative encryption system, only the final equality would apply).

Given such an encryption system, their proposal is as follows. If Alicia wants to send a message M to Roberto, they pick (but do not exchange) secret keys A and R and then

(a) Alicia sends $E_A(M)$ to Roberto. Roberto can not read this message, since he does not know A. In fact, no one except Alicia should be able to read this message.

(b) Roberto sends $E_R(E_A(M))$ to Alicia.

(c) Alicia sends $D_A(E_R(E_A(M)))$ to Roberto.

(d) Roberto computes $D_R(D_A(E_R(E_A(M)))) = M$ allowing him to read Alicia's message. Only Roberto can do this since only Roberto knows key R.

Explain to Alicia and Roberto why their apparently secure scheme would not actually ensure the privacy of their communications. In particular, explain precisely how some evil individual, Carlos, could gain access to the contents of M (assuming that Carlos has complete control over some router through which all messages between Alicia and Roberto must flow).

3. When a web browser loads a new page, it uses TCP to communicate with the server. The browser first requests the HTML file that describes the overall layout of the page. Then, after examining the contents of the HTML file to determine what other files (mainly image files) are needed to display the page, it requests these additional files from the server. In general, the image files may come from servers other than the one from which the HTML file was fetched, but for this problem we will assume (as is frequently true) that the HTML and the image files come from a single server.

   Web browsers interact with web servers by connecting to TCP port 80. There are actually several ways a browser can use TCP to fetch web content.

   sequential For each item to be fetched (i.e. each .html file and each image file and each applet .jar file and so on), the client can open a new TCP connection, send a request for the needed file, receive the file and then close the connection. Then, the client repeats the process to fetch the next item. In this mode, only one TCP connection is open at any given time and only one file is retrieved using each connection.

   parallel For each item to be fetched, the client creates a new thread which opens a connection, sends a request, receives the file and then closes the connection. Several of these threads may be active at the once. In this mode, only one item is fetched through each connection, but many connections may be open at the same time.

   persistent The client opens one connection to the server for each web page to be accessed. After down-loading the .html file, the client checks the contents to see if any image files must be fetched for the page from the same server. If so, it keeps the connection open and sequentially sends requests for the images and receives the needed image data from the server. Once all

the files required have been fetched, the connection is closed. In this mode, only one TCP connection is open at any given time, but many items may be fetched through that single connection.

I would like you to investigate the ways in which these different approaches interact with TCP congestion control.

(a) First, determine a formula for how many packets would actually be sent by the client and by the server using each of the three approaches. Assume that

- All requests and the HTML source file fit in a single packet,
- Each image file requires P packets, and
- The HTML file references N distinct image files.

Justify your answer. In fact, it may be useful for you to draw a little diagram of the packets that would be exchanged. You may ignore things like DNS and ARP requests. Just focus of the TCP segments exchanged. (Hint: your answers for two of the three techniques should be identical.)

(b) Now, assuming that the only factor limiting the rate at which packets can be sent is the TCP congestion control mechanism, provide a formula to estimate the time required to complete the process of fetching the page and all the associated images using each of the three approaches described above (sequential, parallel and persistent). That is, assume that the link connecting the client and server is perfectly reliable and has a very high transmission rate and bandwidth. So, no packets ever timeout. Instead, each packet is acknowledged in a time approximately equal to the value RTT. Basically, assume that the congestion control algorithm is being applied even though the network links used have such high capacity that there is no congestion. To keep things simple, assume that the congestion window is measured in packets (rather than in bytes).

If you find it difficult to produce a general formula, you can instead (for not quite full credit) pick some particular values for I and P and show me how to estimate the time required.

(c) Next, assume that the capacity of the links used is such that congestion is a factor and packets do get discarded as a result of congestion. Such situations are sufficiently complex that a precise, quantitative analysis is not practical. So, instead of asking you to provide a formula describing the behavior of the three approaches in this case, I simply ask that you predict which of the three approaches will complete the task of down loading a web page and its associated image files most quickly (and, of course, that you briefly justify your answer).

As you could probably easily guess, sequential is going to always be the slowest. So, the real question is whether parallel or persistent is faster. It may be the case that in some situations parallel is fastest and in others persistent is fastest. If so, you should describe scenarios where each method would be best. In any case, you should make sure that in your answer you state any assumptions about factors such as the number of images fetched, the size of the images or the point at which packet loss due to congestion becomes an issue.

4. On many networks whose average load is very low, there will be intervals where one station tries to send a large quantity of data to another. Ignoring the reality of acknowledgments, this will lead to a temporary high load situation caused by just one station.

   For this problem, I'd like you to consider how several of the protocols we consider handle this situation.

(a) First, imagine a FDDI-like token ring (i.e. a token ring in which a station releases the free token as soon as it is finished sending data). If a token ring allowed a station to transmit for as long as it had data, then the utilization of the network in the "one busy machine" scenario would be one. In real token rings, however, there is a limit on how long a station can transmit before it must release a free token known as the token holding time. As a result, the efficiency of the protocol will be less than one. I would like you to derive a simple formula for the maximum efficiency in this situation.

Assume that the time for a signal to propagate around the network is "a", and the time to send a maximum length packet is "1". That is, "1" is also the token holding time (THT).

(b) Now, consider the case of a CSMA/CD protocol. While Ethernet-like protocols don't have anything corresponding exactly to the THT, there is an upper limit on the size of the packets allowed on an Ethernet. Putting a limit on the packet size a station can send isn't quite enough. A station could get around this limit by just sending one packet right after another. So, we also need to enforce some minimum idle period, p, between the end of the transmission of one packet by a given station and the beginning of that station's next attempt to transmit.

So, letting "p" denote the length of the minimum pause required and assuming that "a" is the propagation time and that the length of a maximum size packet is 1, determine a formula for the maximum efficiency of a CSMA/CD protocol in the one-busy-station scenario.

(c) Next, let's think a bit harder about this pause time "p" used in our CSMA/CD protocol. How big should it really be? Obviously, it should be at least a few bit transmission times (long enough for other stations to detect an idle period on the cable). It might, however, be better to make it some larger value, such as a, 2*a, of 1.

Discuss the advantages/disadvantages of these four possible choices for the actual value of p. Which do you think would be the best choice? Which would be the worst choice? Why? When evaluating possible choices for "p" do not limit your attention to the one-busy-machine scenario discussed in the other sections of this problem. Instead, consider how the network would behave under a variety of load levels using your choice for p.