THE UNIVERSITY OF AUCKLAND

Department of Computer Science
COMPSCI 314 S1 C Assignment 2.
Due Friday 2 April 2004, 4 pm.

- This assignment will contribute $^{5}/_{30}$ of the coursework mark and 5% of the overall course mark.
- The prefix M means Mega or 10^6;
  bps is bits per second
  Bps is byte per second
- "User data rate" means the number of bytes that a user sees sent or received in one second.

Q 1 A user message is to be transmitted over a 100 Mbps Ethernet LAN, using TCP (transport layer), IP (network layer).

- Each user message is a single TCP "segment", with a TCP header of 20 octets.
- Each segment (including its header) is carried as one or more IPv6 packets or "datagrams" (but as few as possible).
- Each IPv6 packet has a 40 octet header and may be as large as the physical medium will carry. (1500 octets is the maximum size of an IP packet *including* any IP headers.)
- Allow for the preamble and all other parts of the Ethernet frame, but ignore the inter-frame gap.

◇ *For now the only thing you need to know about IPv6 is that each packet has a 40 octet overhead Similarly, each TCP segment has a 20 octet overhead, and may be "fragmented" among several IP packets.*

◇ *The TCP, IP and Ethernet layers form a layered protocol stack, as discussed in lectures, with each adding a header (and possibly trailer) as the message passes down from the user, for transmission. While you are welcome to read ahead to really understand what is going on, the description given here is adequate for the present.*

(i) Estimate the user data rate for user packets of 20, 200, 2000 and 10,000 bytes. **[6 marks]**

Considering the 2000 byte user message —

- A TCP header is added to give a TCP segment of 2020 bytes
- An Ethernet frame (with no SNAP header) can carry an IP packet of 1460+40 bytes (data+header)
- The TCP segment of 2020 bytes becomes 2 IP packets, of (1460+40) and (560+40) bytes (the first including the TCP header of 20) bytes.
- Each Ethernet frame has an overhead of (8+6+6+2+4) = 26 octets.
- To send the 2000 bytes of user data needs frames of (26+1460+40) and (26+560+40) = 1526+626=2152 octets = 17216 bits. (Actually all that we need is to send the TCP segment (2020 bytes) plus 2 frame overheads (26 Ethernet + 40 IPv6 = 66 bytes each)
- At a speed of 100 Mbps = 100b/µs, the time taken is 172.16 µs.
- The data rate seen by the user is then 2000/172.16 = 11.6 bytes/µs = 92.94 Mbit/s.

Repeating for the other message lengths gives

User message (bytes)	20	200	2,000	10,000	500	0
TCP segment length(bytes)	40	220	2,020	10,020	520	20
IP packets (or 802.3 frames)	1	1	2	7	1	1
Frame o'head (@ 66 octets/frame)	66	66	132	462	66	66
Total bytes transferred	106	286	2152	10482	586	86
Total transfer time (µs)	8.48	22.88	172.16	838.56	46.88	6.88
User transfer rate (Mbyte/s)	**2.358**	**8.741**	**11.617**	**11.925**	**10.666**	
User transfer rate (Mbit/s)	**18.868**	**69.930**	**92.937**	**95.402**	**85.324**	

Basic method 2 marks, result for each speed, 1 mark each

(ii) If each transmitted TCP segment is itself acknowledged by another TCP segment (header only, with no data), estimate the user data rates for user messages of 500 and 10,000 bytes. Assume that each user message is sent as a sequence of segments and packets, and that the sender then waits for an acknowledgement before starting the next user message.

◇ *Using a LAN means that we can ignore propagation delays.*

[4 marks]

We can repeat the calculations of part (i) for message lengths of 500 and 0, and thence get the total time to transfer the user data as —

User message (bytes)	500	10,000
Time to transfer user message	46.88	838.56
Time to send reply	6.88	6.88
Total time (send & reply) µs	53.76	845.44
User data rate (Mbyte/s)	**9.301**	**11.828**
User data rate (Mbit/s)	**74.405**	**94.625**

2 marks for method, including time for empty reply, 1 mark for each result

(iii) *Comment only* (no full calculations) on the probable effect on user data rate if the IPv6 traffic must at some stage be "tunnelled" through an IPv4 link (IPv4 header = 20 octets).

[2 marks]

The IPv6 packet must be enclosed in an IPv4 packet (the entire IPv6 packet, including header, becomes the payload for the IPv4 packet) , thus adding 20 bytes to the amount of data to be sent in both directions. This is 320 bits in total, taking **3.20µs** more than the times of part (ii).
This extra time decreases the visible user speed by about 3µs in 50µs for the 500 byte case (about 6%) and about 0.3% for 10,000 bytes. **The user data rate becomes about 8.78 Mbyte/s (70.2Mbit/s) for 500 byte messages, with negligible change for 10,000 bytes**.

Q 2 It is desirable in data communications to avoid character-by-character processing of messages wherever possible, including message copying and, unfortunately, programmed calculation of checksums. This question tries to show you why!
Assume here that each TCP segment (with its header) is first checksummed and then divided into IPv6 datagrams of an appropriate size (again each with a header). All of the IP datagrams for the segment are sent in sequence and the whole segment is acknowledged by a similar reply segment before the next segment is sent. (The user message must be verified as correct before it is acknowledged; if the received segment is 1160 octets, then the acknowledgement will also use a segment of 1160 octets.)
Each checksum or copying operation must be completed before proceeding to the next stage of processing.
The situation is similar to that of Question 1 above, with a 1400 byte user message which will be transmitted over Ethernet as an IPv6 datagram.

Memory bandwidth overheads.

Some operations must process the entire message, traversing it byte by byte (or possibly word by word). Assume the following times —

- checksums, where used, take 100 ns per 2 bytes (16 bits),
- copying a message memory to memory takes 200 ns for each 4 bytes (32 bits). (Where possible this copying should be replaced by data structures and pointers into the original data area, so that a byte is transmitted from the original user data area.)
- copying from memory when sending the Ethernet frame is performed "invisibly" by

hardware and has no extra overhead; the same applies to the calculation of the Ethernet checksum.

- The final Ethernet transmission has zero cost, either for memory transfers or for checksum calculation.

Possible memory intensive operations include —

a. Calculate the TCP checksum – *this must done for every segment*

b. Copy the TCP segment into the IP packet – *this may be done for every segment*

c. Copy the IP packet into the Ethernet frame – *this may be done for every packet*

(i) The results may differ by as much as 10% from the correct values. What simplifications does this allow in the calculations (or what quantities may be omitted)? **[2 marks]**

For this look at "large values" and "small values" that are added together. The small values include the headers etc. Consider values from Question 1. As the frame overheads are 66 octets per frame, we can ignore them for any Ethernet frame over about 700 octets. Thus in the rest of Q2 just ignore all headers, as the 1400 user data certainly exceeds the threshhold.
Apply these simplifications to the remaining answers.

(iii) Assuming that there are no memory bandwidth overheads at all, calculate the time to transfer a single user segment, for an Ethernet speed of 100 Mb/s, and hence the "user transfer rate" in this case (the number of bytes transferred per second as seen by the user). **[5 marks]**

Ignore the time to generate the message, or any and all processing overheads. As half the time is spent sending in one direction and half the time waiting for a reply, the user sees a rate of half the Ethernet rate, or 50 Mb/s.
(Including the headers and other overheads gives a total traffic of 2992 bytes to transfer 1400 data bytes, and a visible user rate of 46.79 Mbit/s, which is well within the allowable 10% tolerance.) Observe that we have not worked out how long a transfer actually takes – it is not needed!

5 easy marks!

(iv) Calculate the user data rate with different degrees of copying. Assume that there is no copying for physical transmission. Also assume that there is no overlap between the stages of the processing, so that for example all of the IP datagrams must be constructed before the first one is processed.

Consider two cases —

1. The message is checksummed before transmission and after reception, with minimal copying. Transmission may be overlapped with other processing, but a frame cannot start until its IP packet is completely processed.

   • Checksumming a 1400 byte message (at 100 ns per 2 bytes) takes 70µs.
   • The time to send the message is 1400*8/100 = 112 µs.
   • To get a verified message to the receiver then takes
   70 (sender checksum) + 112 (sending message) + 70 (receiver checksum) = 252 µs.
   • It takes a similar time to renerate, send and receive the acknowledgement, giving a time of 504 µs to send the 1400 bytes, or **2.77 Mbyte/s** or **22.2 Mbit/s**.
   • (Within the 10% accuracy, we can say **2.8 Mbyte/s** or **22 Mbit/s** – 3 Mbyte/s or 20 Mbit/s is not quite god enough)
   This is about half the rate from ignoring checksum overheads.

   1 mark for checksum cost, 1 mark for <u>both</u> ends, 1 mark for result [total 2]

2. As well as checksumming, the SDUs at each level are copied first by the Transport layer to form its Transport-PDU, and then by the Network Layer to give the Network-PDU. There is similar copying at the receiver to recover the user data.

• Start with the earlier time of 504 µs to send the 1400 bytes, or **2.77 M byte/s** or **22.2 Mbit/s**.

• Each copying operation takes 70 µs, and there are four involved in each data transmission and four in each acknowledgement, a total of 560 µs for copying on each transfer.

• The send-acknowledge cycle increases from 504 µs to 1064 µs for each 1400 bytes.

• The user transfer rate is then 1400/1064µs = **1.32 Mbyte/s = 10.5 Mbit/s**.

Comparing the rates shows how much processing overheads can affect data transfer rates

• Ignore all processing costs	50.0 Mbit/s	
• Include the essential TCP checksum	22.2 Mbit/s	44% original
• Include message copying	**10.5** Mbit/s	21% original

[6 marks]

1 mark for copying cost, 1 mark for <u>both</u> ends, 1 mark for result [total 3]

[Assignment total = 25 marks]

Explanation of the "minimal copying" referred in the assignment.

This assumes that the Ethernet system includes a form of I/O processor, placed between the processor and the Ethernet interface proper, that is given a table IO instructions or "IO program". The IO processor scans the IO program, delivering a sequence of commands to the Ethernet interface.

In the simplest case with no I/O processor the Ethernet controller might be given a command "send this data as a single frame". Here the user program constructs a table of IO instructions and the passes the table's address to the Ethernet interface. The Ethernet interface "executes" those instructions in sequence to send several frames. Only when it has finished (or when an error occurs) does it notify the main processor.

Most "IO instructions" have the general form "*Send* *n* bytes from address *A*", but with other operations besides *send*. (A real implementation might have to handle complications from real or virtual addresses, but these can be ignored for now.)

In this example we want to send 3000 user octets, and for simplicity say that each Ethernet frame includes at most 1400 user octets (this leaves space, and spare, for headers).

Then construct small memory areas, all correctly formatted, for each of the TCP header, all 3 IP headers, and the Ethernet header (destination & type, source is supplied by hardware).

The IO program then looks like

operation	bytes	address
start frame	6 bytes	Ethernet header
send	40 bytes	IP header 1
send	20 bytes	TCP header
send	1400 bytes	UserData[0]
close Frame		
start frame	6 bytes	Ethernet header
send	40 bytes	IP header 2
send	1400 bytes	UserData[1400]
close Frame		
start frame	6 bytes	Ethernet header
send	40 bytes	IP header 3
send	200 bytes	UserData[2800]
close Frame		
finish		