Assignment 1

username: wendongc1

student ID:931018 name: Chen Wendong

1. The total header size: 20+20+20+100+150=310 bytes.

   The total size of the messages after encapsulation: M+310 bytes.

   So the fraction of the headers is 310/(M+310).

2. The image's size: 1920×1080×3=6220800 bytes.

   6220800 bytes = 49766400 bits.

   For 56-kbps model channel: 49766400 ÷ 56,000 ≈ 888.686 sec.

   For 1-Mbps cable modem: 49766400 ÷ 1,000,000 ≈ 49.766 sec.

   For 100 Mbps Ethernet: 49766400 ÷ 100,000,000 ≈ 0.498 sec.

   For gigabit Ethernet: 49766400 ÷ 1,000,000,000 ≈ 49.766 ms

3. In the first approach, the file is sent by dividing it into many small packets while the second method is sending the entire file once. After dividing the file into packets, the packets get bigger by adding protocol overheads so that the size of the file becomes larger because of the portion of headers. Not only that, for each packet the receiver should individually acknowledge and those many "ACK" messages will also occupy bandwidth so that it will take more time to transmit a file. On the contrary, in the second approach the whole file is sent directly and only need to be acknowledged once which is more efficient but imprecise. For bandwidth utilization, there're not many attached overheads so the speed is faster but it's not very safe. If the network condition is very good (There is no error in the process), the second approach is better than the first one. If the network condition is not good, once a problem occurs (like loss of bytes, out of order), for the first method just need to resend the packets which are not acknowledged correctly. But for the second approach the whole file need to be resent again. So the first approach is better for the bad network condition.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.0.5	4000v-eng-web-depa…	TCP	78	62143 → 80 [SYN] Seq=0 Win…
2	0.067836	4000v-eng-web-depa…	192.168.0.5	TCP	74	80 → 62143 [SYN, ACK] Seq=…
3	0.067915	192.168.0.5	4000v-eng-web-depa…	TCP	66	62143 → 80 [ACK] Seq=1 Ack…
4	0.068056	192.168.0.5	4000v-eng-web-depa…	HTTP	152	GET / HTTP/1.1
5	0.135078	4000v-eng-web-depa…	192.168.0.5	TCP	66	80 → 62143 [ACK] Seq=1 Ack…
6	0.139248	4000v-eng-web-depa…	192.168.0.5	TCP	1506	80 → 62143 [ACK] Seq=1 Ack…
7	0.140179	4000v-eng-web-depa…	192.168.0.5	TCP	1506	80 → 62143 [ACK] Seq=1441 …
8	0.140284	192.168.0.5	4000v-eng-web-depa…	TCP	66	62143 → 80 [ACK] Seq=87 Ac…
9	0.141151	4000v-eng-web-depa…	192.168.0.5	TCP	1506	80 → 62143 [ACK] Seq=2881 …
10	0.141275	192.168.0.5	4000v-eng-web-depa…	TCP	66	62143 → 80 [ACK] Seq=87 Ac…
11	0.142070	4000v-eng-web-depa…	192.168.0.5	TCP	148	80 → 62143 [PSH, ACK] Seq=…
12	0.142142	192.168.0.5	4000v-eng-web-depa…	TCP	66	62143 → 80 [ACK] Seq=87 Ac…
13	0.143195	4000v-eng-web-depa…	192.168.0.5	TCP	1506	80 → 62143 [ACK] Seq=4403 …
14	0.143770	4000v-eng-web-depa…	192.168.0.5	TCP	1506	80 → 62143 [ACK] Seq=5843 …
15	0.143849	192.168.0.5	4000v-eng-web-depa…	TCP	66	62143 → 80 [ACK] Seq=87 Ac…
16	0.144348	4000v-eng-web-depa…	192.168.0.5	TCP	1506	80 → 62143 [ACK] Seq=7283 …
17	0.144447	192.168.0.5	4000v-eng-web-depa…	TCP	66	62143 → 80 [ACK] Seq=87 Ac…
18	0.144939	4000v-eng-web-depa…	192.168.0.5	TCP	305	80 → 62143 [PSH, ACK] Seq=…
19	0.144992	192.168.0.5	4000v-eng-web-depa…	TCP	66	62143 → 80 [ACK] Seq=87 Ac…
20	0.148952	4000v-eng-web-depa…	192.168.0.5	TCP	1506	80 → 62143 [ACK] Seq=8962 …
21	0.149486	4000v-eng-web-depa…	192.168.0.5	TCP	1506	80 → 62143 [ACK] Seq=10402…
22	0.149565	192.168.0.5	4000v-eng-web-depa…	TCP	66	62143 → 80 [ACK] Seq=87 Ac…
23	0.207505	4000v-eng-web-depa…	192.168.0.5	TCP	923	80 → 62143 [PSH, ACK] Seq=…
24	0.207605	192.168.0.5	4000v-eng-web-depa…	TCP	66	62143 → 80 [ACK] Seq=87 Ac…

No.	Time	Source	Destination	Protocol	Length	Info
23	0.207505	4000v-eng-web-depa…	192.168.0.5	TCP	923	80 → 62143 [PSH, ACK] Seq=…
24	0.207605	192.168.0.5	4000v-eng-web-depa…	TCP	66	62143 → 80 [ACK] Seq=87 Ac…
25	0.621015	4000v-eng-web-depa…	192.168.0.5	TCP	1506	80 → 62143 [ACK] Seq=12699…
26	0.621834	4000v-eng-web-depa…	192.168.0.5	TCP	1506	80 → 62143 [ACK] Seq=14139…
27	0.621920	192.168.0.5	4000v-eng-web-depa…	TCP	66	62143 → 80 [ACK] Seq=87 Ac…
28	0.622939	4000v-eng-web-depa…	192.168.0.5	TCP	1506	80 → 62143 [ACK] Seq=15579…
29	0.623101	192.168.0.5	4000v-eng-web-depa…	TCP	66	62143 → 80 [ACK] Seq=87 Ac…
30	0.623452	4000v-eng-web-depa…	192.168.0.5	TCP	819	80 → 62143 [PSH, ACK] Seq=…
31	0.623519	192.168.0.5	4000v-eng-web-depa…	TCP	66	62143 → 80 [ACK] Seq=87 Ac…
32	0.624297	4000v-eng-web-depa…	192.168.0.5	TCP	1506	80 → 62143 [ACK] Seq=17772…
33	0.625139	4000v-eng-web-depa…	192.168.0.5	TCP	1506	80 → 62143 [ACK] Seq=19212…
34	0.625211	192.168.0.5	4000v-eng-web-depa…	TCP	66	62143 → 80 [ACK] Seq=87 Ac…
35	0.626435	4000v-eng-web-depa…	192.168.0.5	TCP	1291	80 → 62143 [PSH, ACK] Seq=…
36	0.626516	192.168.0.5	4000v-eng-web-depa…	TCP	66	62143 → 80 [ACK] Seq=87 Ac…
37	0.626756	4000v-eng-web-depa…	192.168.0.5	HTTP	109	HTTP/1.1 200 OK (text/htm…
38	0.626790	192.168.0.5	4000v-eng-web-depa…	TCP	66	62143 → 80 [ACK] Seq=87 Ac…
39	0.627107	192.168.0.5	4000v-eng-web-depa…	TCP	66	62143 → 80 [FIN, ACK] Seq=…
40	0.694102	4000v-eng-web-depa…	192.168.0.5	TCP	66	80 → 62143 [FIN, ACK] Seq=…
41	0.694214	192.168.0.5	4000v-eng-web-depa…	TCP	66	62143 → 80 [ACK] Seq=88 Ac…

4. This is the trace as well as flow graph I captured. Before the connection, the server executed primitive “LISTEN” which represented waiting for an incoming connection. Then my laptop executed primitive “CONNECT” to establish a connection with the server. As you can see in the picture, “SYN”(synchronous) is the connection request sent from my laptop to the server. When the server got my connection require(SYN), it executed primitive “ACCEPT” which means accept connection and it sent a “SYN,ACK” message to my laptop. “ACK” means acknowledgement and this “SYN,ACK” message represented the server has accepted the connection request. Then my laptop sent “ACK” message to respond the server and at that time

the connection has been totally established(So-called three-way handshake). After that the server executed primitive “RECEIVE” to accept the request then my laptop executed primitive “SEND” to send the request. In the picture, “GET/HTTP” is the request sent from my laptop to the server and after that the server began to send packets to my laptop. To process my request, the server executed primitive “SEND” and return the packets to my laptop and my laptop executed primitive “RECEIVE” to receive the packets from the server. When the job was done, the server sent a “HTTP/OK” message to my laptop which means all packets have been sent successfully. Then my laptop sent a “ACK” message and executed primitive “DISCONNECT” by sending a “FIN,ACK”(finish) message to the server. The server also executed primitive “DISCONNECT” and sent a “FIN,ACK” to respond to my request. The last “ACK” message from my laptop to the server means this connection was ended.

5. i) Generally speaking, the file size is large, so the bandwidth is an important factor. Although distance between Melbourne and USA is very far and it’s necessary to consider the latency. But comparing to the ability of transporting a large number of bits per second(bandwidth), the time delay of transporting bits on the way is relatively less crucial. I think the suitable example of this application is fiber optics, which has high bandwidth with low latency.

   ii) For the bandwidth intensive interactive gaming, the latency is depended on situations. If the interactive gaming is online game (long distance), real-time interaction must be ensured so the latency must be very small. Bandwidth is also a key issue because the game is “bandwidth intensive”. The suitable example of this application is fiber optics, because it has high bandwidth and low latency. If interactive gaming is proceeding face to face, the latency is not important because the distance between players is very close. In this case WLAN is a good choice. Players can interact with each other with high-speed wireless network.

   iii) Just as the question says the data size is small so the bandwidth is not a problem. The data transmissions are frequent and the sensors require critical event notification so that the alarm needs to be transmitted as soon as possible. Information timeliness is the most important thing so the lower latency is, the better. I think the suitable example of this application is communication satellite. Whenever an unexpected situation is encountered (like earthquake, flood, fire), satellite can always send information in real time (or recover communication in the quickest time), while physical medium may be destroyed. Another advantage of satellite is that it can broadcast to large area.

iv) People use broadband network to do various things like watching videos, playing games and download files, so broadband connections require high bandwidth and low latency. Due to geographical reason, there are some difficulties in laying fiber optics and it's hard to reduce latency to a very low level. I think communication satellite is a suitable application for this situation. VSAT have enough bandwidth for daily application and it can easily cover large area with low prices.

v) The data flow of video streaming is large and people don't want to wait the buffered video so it requires high bandwidth and low latency. Meanwhile, this application also requires mobility of the network. In this case the suitable example of this application is cellular network (3G or 4G). Actually, 4G cellular network has become widespread and it satisfies people's social needs very well.