Problem 1

Suppose two packets arrive to two different input ports of a router at exactly the same time. Also suppose there are no other packets anywhere in the router.

(a) Suppose the two packets are to be forwarded to two different output ports. Is it possible to forward the two packets through the switch fabric at the same time when the fabric uses a shared bus?

(b) Suppose the two packets are to be forwarded to two different output ports. Is it possible to forward the two packets through the switch fabric at the same time when the fabric uses switching via memory?

(c) Suppose the two packets are to be forwarded to two different output ports. Is it possible to forward the two packets through the switch fabric at the same time when the fabric uses a crossbar?

(a) No, it is not possible to forward two packets to different output ports through a switching fabric with a shared bus at the same time. In this shared bus, the packets must be sent one at a time.

(b) No. Even if the fabric uses memory switching, you can only read/write one packet at a time in this shared system bus implementation. Thus, only one packet can be forwarded at a time.

(c) Yes. If the fabric is using a crossbar, it is possible to send two packets at the same time to two different output ports. This interconnected network has a matrix topology in which every single input port has a unique connection to each output port. As long as the packets are going to different output ports, it is possible to send both through the switching fabric at the same time. As long as an individual output port is free, it will not be delayed in receiving any packets, thus two different outputs can receive packets in parallel.

Problem 2

Consider a router that interconnects three subnets: Subnet 1, Subnet 2, and Subnet 3. Suppose all of the interfaces in each of these three subnets are required to have the prefix 224.1.17.0/24. Also suppose that Subnet 1 is required to support at least 60 interfaces, Subnet 2 is to support at least 90 interfaces, and Subnet 3 is to support at least 8 interfaces. Provide three subnet addresses (of the form a.b.c.d/x) that satisfy the constraints. You may use the following link to help verify your result: `http://jodies.de/ipcalc`.

I tried to make sure that whichever subnets that I chose did not overlap.
Subnet 1:
Need 60 possible addresses for interface; can get this by allowing the last 6 bits to be changed, after setting the first two bits to 10). This means using a mask of size 26. The first 24 bits are provided in the prefix, and the next two bits could be set to 1 0. The remaining 6 bits can be used to support over 60 interfaces (62 available, to be exact).
Subnet 2:
Need 90 possible addresses for this interface; get this by allowing the last 7 bits free, but have to make sure does not overlap any of the previous interface possibilities. Use a mask of size 25, so the first 24 bits are from the prefix and the next 1 bit is set to make sure that it would not overlap with other subnet(s). Here, I set that bit to be 0 so that, clearly, it will not overlap with subnet 1 (had the 25th bit set to 1). The remaining 7 bits can be used to support over 90 interfaces (126, to be exact).
Subnet 3:
Need 8 possible addresses for this interface; get this by allowing the last 4 bits free, but have to make sure does not overlap any of the interface possibilities of the other two subnets. This means using a mask of size 28. The first 24 bits are from the prefix, and the next 4 are fixed to make sure that they do not overlap with the previous ones. I chose the values 1 1 1 1. This was to ensure that there is no overlap with the other subnets, as it is clear to see that just based on the first two bit values, their would not be any overlap. This leaves the last 4 bits free, to support over 8 interfaces (support 14 to be exact.
Final subnet addresses:
Subnet 1: 224.1.17.128/26
Subnet 2: 224.1.17.0/25
Subnet 3: 224.1.17.240/28

Problem 3

Consider sending a datagram with total length 2400 B into a link that has an MTU (maximum transmission unit) of 800 B. Suppose the original datagram is stamped with the identification number 421 and all IP headers are 20 bytes.

(a) How many fragments are generated?

(b) What are the values in the various fields (header length, total length, identification, MF flag, and fragment offset) in the IP datagram(s) generated related to fragmentation?

(a) The total number of fragments generated depends on the total payload, as well as the max payload that can be carried by each MTU.
The total payload: from the original datagram, payload was 2400 - 20 (header size) = 2380. A single MTU's Max payload = 800-20 = 780, BUT this payload must be divisible by 8 since we need to store the correct offset value (multiples of 8) in the data fragments' headers. Thus, we reduce this payload to 776. Total number of fragments:(2380)/776 = 3.05.
We must round this value up, thus the total number of fragments generated = 4.

(b) IP datagrams' field values:
All of these datagrams will have the same header length, 20 (Bytes).
All of these datagrams will have the same identification number, 421 (same as original datagram).
The first 3 datagrams will have MF Flag = 1(True); the 4th will have MF Flag =0(False), since it is the last datagram and there are no more after it.
The Fragment offset values are as follows:
1st datagram: 0
2nd datagram: 97
3rd datagram: 194
4th datagram: 291
The total length values are as follows, all in bytes:
1st datagram: 796
2nd datagram: 796
3rd datagram: 796
4th datagram: 72 (just the leftover bytes, doesn't have to be a full datagram)

Problem 4

Please answer the following questions regarding checksum.

(a) Why is the IP header checksum recalculated at every router?

(b) What is covered by IP checksum and TCP checksum?

(a) This header checksum is recomputed at every router to check for the case that the IP Packet header is corrupted in transmission along a link ("hop") between routers. If this happens, incorrect information that adversely affects transmission might happen, such as an improper change in the destination IP of the packet due to corruption. Thus, this checksum is recalculated at every router in order to mitigate these kinds of errors.

(b) The IP Checksum only covers the IP Packet Header. Meanwhile, The TCP Checksum covers the entire TCP segment, meaning both the header as well as the payload. Since this is exactly what is encapsulated inside of the IP Packet header (this is what consists of an IP packet's payload), then it is sufficient for the IP Checksum to only cover the IP Packet Header, assuming that the TCP Checksum can handle the rest.

Problem 5

In this problem we will explore the impact of NATs on P2P applications. Suppose a peer with username Arnold discovers through querying that a peer with username Bernard has a file it wants to download. Also suppose that Bernard and Arnold are both behind a NAT. Try to devise a technique that will allow Arnold to establish a TCP connection with Bernard without application-specific NAT configuration. If you have difficulty devising such a technique, discuss why.

It would not be possible to create such a TCP Connection without using an application-specific NAT configuration. Whichever host first creates the connection must begin by interacting with the NAT covering the other. If a host tried to create the connection with the other behind this NAT, it will not be able to since it won't be able to access the other host's private IP. Therefore, any SYN packet sent to establish a connection would not make it to the end host on its own. This is why an application-specific NAT configuration would be necessary, to allow these connection messages to go through. These configurations are what is used by the NAT's to handle these transport layer protocols, TCP and UDP, specifically in this case to allow this TCP connection between Bernard and Arnold. One configuration we discussed in class is a relaying node, which is used in applications such as Skype to create these connections and abstract away the NAT in creating a connection, while the entire time forwarding all of the information received to the correct device behind it's corresponding NAT. In this, a NAT can act as a sort of "firewall" to block these connections.
Therefor, it is not possible for Arnold to create a TCP connection with Bernard without any application-specific NAT configuration.