CSE 461 – Autumn 2012
Midterm Key

Please do not read beyond this cover page until told to start.
There are 51 points in all.

1. *[6 points]*
An Ethernet device you might buy today typically supports three rates: 1Gbps, 100Mbps, and 10Mbps. When an Ethernet cable is plugged into the device, it communicates with the other end, chooses a rate to use, and then sticks with it.
802.11 devices also support multiple rates. When they come online they are informed of the set of rates the AP is willing to support. They then engage in a continuing process of choosing a specific rate to use for the next short while, adjusting that rate up and down as they please.

(A) Briefly argue why it's a good idea for 802.11 to repeatedly choose transmission rates **AND** why it's not a good idea for Ethernet to do so.

The 802.11 signal-to-noise ratio can change dramatically over time, which strongly affects the possible transmission rates. Unless 802.11 adapted, it would have to choose between wide coverage at low rates and high rates at low coverage. Dynamic adaptation lets it try to achieve both. Ethernet operates in a much more constrained environment, with strict limits on signal quality imposed by the specification. The environment does not change dynamically.

(B) Identify (by name or short description) a feature of the 802.11 protocol that (a) is motivated by the fact that 802.11 supports clients operating at potentially significantly different bit rates, **and** (b) does not exist as part of the Ethernet protocol.

Rate adaptation. The protocol provides mechanisms that allow senders to dynamically alter transmission rate to try to maximum current goodput (although it doesn't prescribe a specific technique to find that rate).

2. *[3 points]*
In Project 1 your code sent UDP packets from a client to a server. If we were to look at the bits actually being carried on the wire (assuming we're using a wired network), we'd find the destination IP address and port were part of the bits being sent.
Did the bits on the wire also carry a destination MAC address, or not? Briefly explain your answer.

Yes. All data delivery happens at the link layer, which uses MAC addresses. Higher level protocols, like UDP, are encapsulated in link layer frames.

3. *[9 points]*
The text defines the following simplex stop-and-wait protocol:

```
void sender() {
 while(true) {
  from_network_layer(&buffer);
  s.info = buffer;
  to_physical_layer(&s);
  wait_for_event(&event);
 }
}
```

```
void receiver() {
 while(true) {
  wait_for_event(&event);
  from_physical_layer(&r);
  to_network_layer(&r.info);
  to_physical_layer(&s);
 }
}
```

(A)This protocol doesn't provide reliable delivery if frames can be lost. Briefly explain why not.

The protocol relies on strict alternation between the sender and the receiver. If a frame is lost in either direction, the other side will wait forever, since there is no provision for retransmissions.

(B) Suppose I can't change the receiver, but I can do anything I want to the sender. Can I change (just) the sender so that the protocol works even when frames are lost? Briefly explain.

No. If the sender employs a retransmission scheme, it leads to possible duplicates. The receiver, as is, isn't capable of eliding duplicates. If the sender doesn't retransmit, there's no recovering from a lost frame in the forward direction.

(C) Starting again with the implementation exactly as shown above (i.e., ignoring any modification from part (B)), does this protocol provide reliable delivery if frames can be delivered in a different order than they were sent (for example, the third frame sent can arrive before the second frame sent)? Briefly explain.

Yes. There is at most one frame in transit at a time, so even if the underlying network can reorder frames, it won't matter to this protocol.

4. *[2+3+3 = 8 points]*
(A) What is the principle advantage of a sliding window protocol over stop-and-wait?

Performance, in particular, throughput.

(B) Sliding window protocols specify a sending window and a receiving window. Can it ever be useful for the sending window to be larger than the receiving window? Briefly explain your answer.

Yes, possibly. The sending window limits the amount of data that can in flight but unacknowledged. The receiver window is a limit on the memory avilable for buffering and reordering. If the receiver is capable of consuming each incoming frame basically as it arrives, it could have a very small receive window. In that case, the send window could be larger than the receive, to allow frames to be in flight (on the wire).

(C) A sliding window protocol implementation must be able to match ACKs with the frames being ACKed. The book indicates that frames should be assigned sequence numbers (that is, consecutive integers), and that an ACK should indicate which frame(s) it is ACKing by using those sequence numbers.
Does the protocol rely on the use of sequence numbers, or could some other form of frame identifier by used instead? In particular, suppose the system has a clock that increments every nanosecond and never rolls over, and that sending a frame takes more than a nanosecond (and that only one frame can be sent at a time). Could the value of that clock be used in place of sequence numbers?
Briefly explain.

No. Besides facilitating ordering, the sequence numbers in sliding window are used to detect when some frame is missing. With timestamps, that's not possible. For instance, if the reciever sees frames sent at times T1 < T2, it can't know whether or not there is a not yet received frame sent at time T3, T1 < T3 < T2, and so can't ever deliver T2 to its client. (The same objection goes for T1 in fact; even if T1 is the earliest timestamp packet the receiver has seen, it can't deliver it because there might be an even earlier one still in flight.)

5. *[6 points]*
Suppose I'm using a data encoding that sends codewords of exactly three bits. The code has three symbols. It sends a single 0 bit as 000; it sends a single 1 bit as 010; and it sends the two bit sequence 00 as 110. So, to send data 101001 it would send 010 000 010 110 010.

(A) What is the Hamming distance of this code? Briefly explain your answer.

1. (For instance, code 000 → code 010 by flipping one bit.)

(B) Will this code always detect errors in which exactly two bits are flipped (in a single received 3-bit codeword)? Briefly explain.

No. Flipping the first two bits of code 110 results in 000, which is a valid code.

(C) Is there ever a case where the receiver can correct an error, under the assumption that there is never more than one bit flipped in a received codeword? Briefly explain.

Yes. If it receives 001 the only valid code within one bit is 000, so the data sent was a 0.

6. *[3 points]*
The Shannon Theorem limit on information rate (bits per second) goes to infinity as the noise goes to zero (so long as the received signal strength and the available bandwidth are both greater than 0). Given an informal argument for how the bit rate could grow without bound as the noise level approaches zero.

As the noise goes to zero, the potential for the receiver to reliably use small thresholds in distinguishing signal level increases. So, while the maximum symbol rate doesn't increase, the potential bits per symbol does.

7. *[3 points]*
Under what conditions might a static partitioning scheme (e.g., time division multiplexing) outperform a multi-access scheme (like Ethernet's CSMA/CD)? Briefly explain your answer.

Under high load. In that case there is no resource wasted because of the partitioning (all nodes want to use the resource each time its their turn), and there is no overhead lost due to dynamic coordination (collision resolution). So, the static scheme makes full use of the resource with overhead lower than a multi-access scheme can achieve.

8. *[3 points]*
IP is "the narrow waist" of the Internet protocol stack. How has standardization near the middle of the stack aided the growth of the Internet, relative to the alternative of standardizing at the bottom of the stack?

By standardizing at the middle, it allows for evolution of both higher level and lower level protocols. In contrast to standardizing at the bottom, this has made it relatively easy to deploy significantly different, and faster, networking hardware.

9. *[6 points]*
Pipelining is a general approach to mitigating the performance impact of high latency.
(A) What performance measure does pipelining improve?

(Maximum) throughput.

(B) What (interesting and important) performance measure does it not improve?

Latency.

(C) How does pipelining come up in CSE 461?

Sliding window is basically trying to pipeline transmissions (to achieve higher peak throughput).

10. *[4 points]*
(A) Give a single example of functionality that is provided by an RPC infrastructure that an application building directly on TCP sockets would likely have to implement itself.

Data encoding. (For example, how a string is represented on the wire.)

(B) Not every application that can be written on top of TCP sockets can be conveniently written on top of RPC. Why not?

The request-reply semantics may not be appropriate. For instance, if what you want to do is stream a video, you really don't want to have the entire video delivered to the client (the reply) before you can start operating on it (i.e., viewing it).