EXAM Telematics Networks (262000)

27 October 2009, 13:45–17:15

- This is an open-book exam: you are allowed to use the book ("Computer Networking" by Kurose & Ross), the reader, copies of the lecture slides, and a dictionary.
  Use of other written material, such as your own notes, is not allowed, nor is the use of laptops, notebook computers, PDAs, mobile phones, etc. ***Please remove any such material and equipment from your desk, now!***
- Although the questions are only written in English, you are allowed to answer in either English or Dutch.
- Copying pieces of text literally from the book, reader or slides is not allowed: such answers are worth 0 points. You should compose your answers yourself.
- Since there have been some significant changes in the course material this year compared to last year, you have the choice to do an "old-style" exam, based on last year's material. For the old-style exam, skip problem 1 and do problem 6 instead.

1. Communication theory and the physical layer

Skip this problem if you do the "old-style" exam

Suppose we are going to design a system that transports a stream of 1000 messages per second, each of which can be "yes", "no" or "maybe", with probabilities 25%, 50% and 25% respectively, independently of each other.

(a) Calculate Shannon's information rate (i.e., amount of information per unit of time) for this stream.

In the reader on page 17, it says the following:

Given a discrete memoryless source characterized by a certain amount of entropy, the average code-word length for a distortionless source-encoding scheme is upper bounded by the entropy.

Unfortunately, this sentence is wrong, as anyone could have noticed because it contradicts both the rest of the text, the slides, and even common sense.

(b) What is the error? How can it be fixed by replacing just one word?
(Hint: in the lecture we ignored the precise meaning of "discrete memoryless" and "distortionless", but the error is **not** in those details.

Next, suppose we want to use a copper cable to transport these messages, and we have a choice between using unshielded twisted pair, and coaxial cable.

(c) Which of the two will in general have the best signal-to-noise ratio at the receiving side, all other things being equal?

We choose the twisted pair cable. Now we are given the choice to use either
[i] the full signal bandwidth of this cable, or
[ii] only half of the signal bandwidth but with twice the signal-to-noise ratio compared to [i].

(d) Which of these two possibilities has the highest Shannon capacity?

Consider the Hamming code; it is normally used for its ability to *correct* a single bit error in a received message. Alternatively, we can choose to use it as an error *detecting* code: when the normal decoder would correct a bit error it has found, we treat the received message as incorrect (and ask for a retransmission), instead of correcting the error.

(e) Are there situations in which this is useful? Explain.

---

2. Addressing

IPv6 addresses are 128 bit long, sufficient for 2^{128} computers, but just about every computer also has a unique MAC address, and there are only 2^{48} of those.

(a) Why does it still make sense to have much more than 2^{48} IPv6 addresses if we never expect more than 2^{48} computers to be connected?

Suppose the same MAC address is used by two different computers, on different LANs that are interconnected by a bridge.

(b) Does this cause problems? Explain.

Suppose the same MAC address is used by two different computers, on different networks that are only interconnected by IPv4 and IPv6 routers.

(c) Does this cause problems for IPv6 traffic if stateless autoconfiguration is used? Explain.

Suppose at some point in the future the 2^{48}'th ethernetcard has been manufactured, so we have run out of MAC addresses. We don't want to upgrade to longer MAC addresses, because the checking of the MAC destination address in received packets is done in the network interface hardware, so the address length cannot be changed without replacing the hardware. (However, the actual address itself can be configured by software.)

(d) Propose a solution for the problem of running out of MAC addresses that can be implemented without hardware modifications; if your solution has any disadvantages, discuss them.

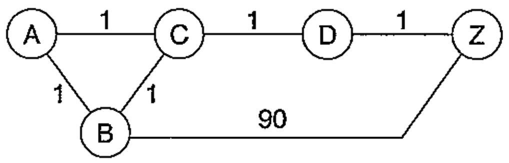
---

3. Distance Vector routing

The main problem of distance vector routing is the *counting to infinity* (also known as *bouncing*) problem. One suggested solution for this problem is colloquially described as "choose a low value of infinity".

(a) Does this completely solve the problem, or not? Why?

Another suggested solution is to use "hold-down timers". Let us explore that further in the following example network, consisting of 5 nodes and 6 links with the indicated costs:



We will focus only on the routing information about the path to node Z, as seen by each of the other nodes.

In this network, the distance vector algorithm with split-horizon and poisoned reverse is used. Distance vectors are exchanged perfectly synchronized (for simplicity) at regular intervals, namely at time t=0, t=1, t=2 and so on; thus, at those times each node gets an update from all its neighbours and can select the best route to node Z.

Furthermore, we use "hold-down timers", which means the following: when a node learns

from the neighbour which previously provided it with the best path to a particular destination, that this destination now is unreachable (i.e., has infinite cost), then it does two things:
- it notes this infinite cost in its routing table;
- it starts a timer (the "hold-down timer"), and until that timer expires, it ignores any further updates about this destination.

Only after the timer has expired are new updates about that destination accepted. Assume the hold-down timer is set to 2.5 seconds.

(b) Suppose the link between D and Z fails at time $t = 0.8$, and this is noticed immediately by both sides of that link. Describe what happens next, by completing the following table, which shows for each node their idea of the best cost to reach node Z *just after* each exchange of routing information; continue until the routing tables have converged again.

time	A's cost to Z	B's cost to Z	C's cost to Z	D's cost to Z
0	3	3	2	1
1				
2				
3				
⋮				

(c) How much does the hold-down rule speed up the convergence in this example? Explain.

(d) Is the hold-down rule always so effective, or can you imagine situations in which it does not prevent counting-to-infinity? Explain.

(e) What is a *disadvantage* of choosing the hold-down timer too large?

4. TCP

Consider the problems that TCP has at high speed. The window problem is solved by introducing the window scaling option, which effectively extends the 16-bit window field in the TCP header.

(a) Explain in your own words how this works.

(b) Why can't we use the same approach to solve the sequence number wrap problem by introducing a sequence number scaling option?

Consider a TCP connection (using standard TCP) over a link with an infinite buffer ahead of it, so packets are never dropped. There is no other traffic over this link, and propagation delays are negligible. Initially, the buffer is empty, and our TCP connection just starts to transmit. The application offers the TCP an unlimited amount amount of data to be transported.

(c) Will the buffer content (queue length) grow infinitely large, or will it stop growing at some point? Why?

Consider the TCP state-transition diagram on page 87 of the reader (old reader: page 78). As the text remarks, a retransmit timer is needed to retransmit SYN or FIN segments if they are lost, but this is not indicated in the diagram.

(d) In which states is there a retransmission timer for a SYN or FIN running? Explain.

Continued on next page...

5. QoS and delay

Let us consider an internet telephony application that transmits audio in packets of 6400 bits each. The audio is sampled with 8000 samples per second of 8 bits each; once 6400 audio bits have been collected, they are sent out as a packet immediately, without any compression.

(a) Describe this particular source's traffic with a leaky-bucket model.
(Formulated more precisely: give leaky bucket parameter values such that the leaky bucket policer will not delay any packet from this source.)
Is this description unique? Why?

Through the internet these packets take a path that goes over several links; one of them has a bit rate of just 1 Mbit/s and is called the bottleneck link. All other links on the path may be assumed to have infinite speed, and all propagation delays may be ignored. Assume the voice packets share the bottleneck link with an unknown amount of data packets of 12000 bits each.

(b) If Fair Queueing is used, with equal shares for data and voice, what is the maximum delay the voice packets may experience?

(c) In the case of Fair Queueing, 0.5 Mbit/s is available for the voice traffic, but our voice flow uses much less. Does this mean the rest of this 0.5 Mbit/s is wasted? Explain.

Telephony is of course a two-way application, but packets going one way in the internet may experience different delays than those going the other way.

(d) Assuming the participants in such a telephone call do *not* have access to perfectly synchronized clocks, can they notice that the delays in both directions are different, or can they only notice the total delay? Explain.

---

6. Long delay in FDDI

Only do this problem if you do the "old-style" exam

Consider the following scenario. We have an FDDI ring with 4 nodes. The ring is short, so we can neglect the propagation delay. TTRT is set to 10 milliseconds. Each node is allowed to transmit at most 2.5 ms of synchronous traffic per round of the token (in the notation of the reader: S(1)=S(2)=S(3)=S(4)=2.5ms). In the first round, each node transmits 1 ms worth of synchronous traffic and 1 ms worth of asynchronous traffic, so the first round takes 8 ms.

(a) When the token returns to node 1, what is the value of TRT measured by this node?

(b) How much synchronous traffic is node 1 allowed to transmit then? And how much asynchronous traffic? Why?

Now assume that in this (second) round, each of the four nodes only transmits 0.1 ms of synchronous and 0.1 ms of asynchronous data.

(c) How much synchronous traffic is node 1 allowed to transmit when it gets the token again (i.e., in the third round)? And how much asynchronous traffic? Why?

(d) Is it possible that this third round lasts (much) longer than TTRT, say, at least 15 ms? If so, how? (Give an example, by specifying valid amounts of synchronous and asynchronous data to be transmitted by each of the nodes in the third round.)
If not, why not? Illustrate your answer with a timing diagram.

(e) Suppose you're allowed to modify the values of S(1)... S(4), within the limitations set by the FDDI protocol. Construct an example scenario that leads to a round with the longest possible duration; how long is this? Also, indicate why you think an even longer round is not possible.