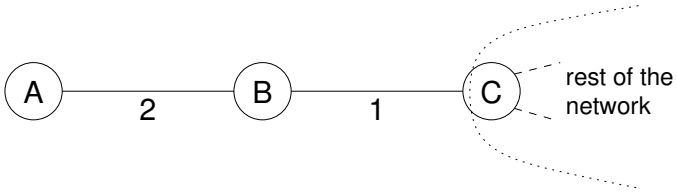
Network Systems (201300179/201400431), Test 3

March 18, 2016, 13:45–15:15

- This is an open-book exam: you are allowed to use the book by Peterson & Davie and the reader that belongs to this module. Furthermore, use of a dictionary is allowed. Use of a simple (non-graphical) calculator is allowed.
- Other written materials, and laptops, tablets, graphical calculators, mobile phones, etc., are not allowed. ***Please remove any such material and equipment from your desk, now!***
- Although the questions are stated in English, you may answer in English or Dutch, whichever you are more comfortable with.
- You should always explain or motivate your answers, with so much detail that the grader can judge whether you understand the material; so just saying "yes" or giving a formula without explanation is not enough.
- Visiting the toilet without explicit permission of the supervisor is not allowed. During the last 30 minutes of the exam, no toilet visits are allowed.

1. Distance-vector routing



Consider a network consisting of six nodes labelled A—F. A small part of the network is shown in the figure; the rest of the network lies beyond node C and no details about it are given. The network uses a simple distance-vector based routing algorithm, without enhancements like split horizon; all nodes simultaneously send a distance vector to their neighbours once per minute.

We focus on what node B does.

Let us assume that initially there is no link between nodes B and C, but A and B are already connected for a couple of minutes. Then B and C get connected.

2 pt (a) What is the distance vector that node B sends to node C?

A at cost 2, and B at cost 0.

Not including B at cost 0 is also ok, as it is implied; one can also argue that C at cost 1 should also be included.

Node C sends to node B the following distance vector:

destination	cost
D	1
E	8
F	4

2 pt (b) How many minutes (i.e., iterations of the algorithm) does it take before also node A knows about destinations D, E and F? Explain.

2 iterations: first node B learns about them, then in the next iteration it tells node A about them.

Of course, it depends a bit on where you start counting; as long as you're clear and consistent with the above, it's ok.

Some time later, after everything has converged, node B gets a different distance vector from node C:

destination	cost
D	1
E	30
F	4

3 pt (c) Describe what will happen next in the network. How long will it take before nodes A and B have correct paths to node E again?

A thought it could reach E at cost 8+1+2 = 11, so it tells that to B. B now prefers A over C, putting its cost to E at 13. Next iteration A thinks cost to E is 15. Then B 17; A 19; B 21; A 23; B 25; A 27; B 29; A 31; then finally B chooses the correct path via C. That's 11 iterations. (+/−1 depending on where you start counting)

Although the above is what I had in mind when composing the question (and was enough for a full score), actually the situation is much more complicated.
For example, the updates could also bounce back and forth between nodes B and C, and since that link cost is only 1, it will take twice as many iterations. Then again, it is strange that node C gave B a cost to E of 30; shouldn't it have given a cost of 10, namely via B? Apparently node C does something to prevent this counting-to-infinity problem.
Furthermore, we do not know anything about what happens in the network behind node C. Perhaps there will be some bouncing back-and-forth over there, and perhaps the 30 that B learnt from C is not the final value...

2 pt (d) Split horizon could have prevented the slow convergence you saw in the previous question. In order to achieve that, would it be sufficient if *only* node A does split horizon?
(Of course, in general it's safest to have all nodes do split horizon; but the question here is whether in this specific case, it would have been enough if only A does it.)

Yes, this would have been sufficient, since then it would never have told B that it could reach E.

If in problem (c) you answered that it bounces back and forth between nodes B and C, then now of course you should now answer that the problem is *not* fixed if only A does split horizon.

Continued on next page...

2. Addressing and routing in the Internet

2 pt (a) Does it make sense to apply the HD ratio to MAC addresses? Explain.

No, because the HD ratio expresses how many addresses are wasted in a convenient hierarchical assignment, while MAC addresses are not assigned hierarchically.

Some pointed out that there is some hierarchy in MAC addresses, namely that the first 24 bits indicated the manufacturer. That's true, but if a manufacturer needs multiple blocks, there's no need for those to be adjacent, while in e.g. IP one would rather not assign say 10 random /24's to a single network.

The IPv6 address block `ff00::/8` is reserved for multicast.

2 pt (b) What does this notation mean? Explain the `ff00`, the `::` and the `/8` parts.

`ff00` are the first 16 bits in hexadecimal notation; `::` means the other 112 bits are 0; `/8` means the addresses belonging to this block are the same in the first 8 bits, the others can be anything.

2 pt (c) Why does multicast need a separate address block? After all, the packets are going to be sent to the same computers that already have unicast addresses.

Each multicast address designates a *group* of computers; listing all their addresses in the packet header would be wasteful or impossible, hence the use of a separate address to represent the entire group.
Or: the sender may not even know the unicast addresses of all computers interested in the transmission.

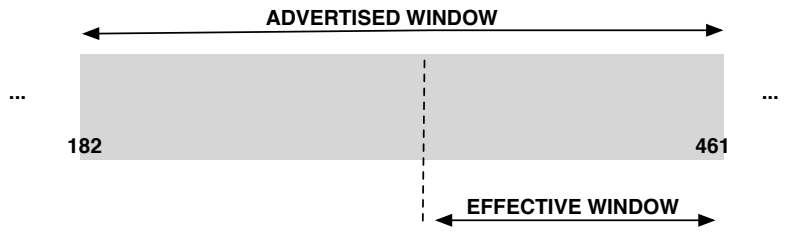
Assume a certain address block, let's say 12.1.2.3/24, is advertised using BGP at two totally different places in the Internet, let's say in Amsterdam and in Madrid.

2 pt (d) What happens if a packet is sent to this address from say Paris? Where will it go, and what determines this?

It will be sent to either Amsterdam or Madrid, not both. Which of the two depends on the number of ASs to cross from Paris; the smallest number will win, unless policies set by the system administrators override this.

---

3. TCP If possible, please answer this problem in English



Consider the state of the sliding window at the sender side of a TCP connection, as depicted in the above figure. The gray area indicates which segments fall into the advertised window. The first byte in the Advertised Window is byte number 182. The last byte is byte number 461. The MSS is 40 bytes and the initial sequence number (used by the SYN packet initiating the connection) is 101.

2 pt (a) Explain the difference between the advertised window and the effective window.

Advertised window: number of bytes the receiver is currently able to receive in the buffer
Effective window: number of bytes the sender can send at this point (considering that some segments are in-flight

2 pt (b) Four segments of size MSS are currently in-flight. How many more segments can be transmitted in the shown state?

bytes_currently_ in-flight = 4 * MSS = 160
First byte of the effective window = start_of_advertized_window +bytes_currently_ in-flight = 182 + 160 = 342
size_of_effective_window_in_bytes /MSS =(461-342+1)/40 = 3.

2 pt (c) Starting with the state shown in the figure, how do the advertised window and the effective window change when a segment with AcknowledgementNumber = 302 and WindowSize = 280 is received?

AcknowledgementNumber = 302 indicates that bytes up to byte 301 have been received. This means that bytes from 182 to 301 that are currently in the advertised window are acked and do not fall into the advertised window anymore. Therefore the Advertised Window will shift forward of 301-182+1 = 120 bytes, from byte 302 to 581. Assuming that no segment has been sent in the meanwhile, the effective window will increase of 120 bytes (240 bytes in total).

2 pt (d) So far we've (silently) assumed the window scaling option was not used. Suppose it was in fact used, with a factor of 16. What change, if any, would this make to the answer to the previous question?

The actual window size will be 280*16 = 4480 bytes, and thus include all sequence numbers up to 4599.

3 pt (e) Consider a regular teardown of a TCP session. Draw the timing diagram of the TCP teardown. In the diagram indicate which segments are sent and, for both the sender and the receiver, which states of the TCP state-transition diagram they traverse (see Figure 5.7 of Peterson & Davie). When drawing the diagram, assume that the host to the left initiates the teardown.

End of this exam.