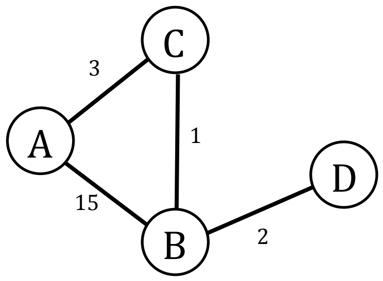
# Network Systems (201300179), Test 3

March 21, 2014, 15:45–17:15

- 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 the following network, where the nodes represent routers, and the labelled links represent links between routers with their associated link costs. We assume the use a distance vector routing algorithm without split horizon or poisoned reverse.



In the following, you will be asked to give the distances stored at each node in the form of a table. This table has the following form (this is also the format used in Table 3.10 in Peterson & Davie):

| Information | Distance to reach node | | | |
|---|---|---|---|---|
| stored at node | A | B | C | D |
| A | | | | |
| B | | | | |
| C | | | | |
| D | | | | |

3 pt (a) Give, in table form, the distances stored when each router only knows the distance to its immediate neighbours.

| Information | Distance to reach node | | | |
|---|---|---|---|---|
| stored at node | A | B | C | D |
| A | 0 | 15 | 3 | ∞ |
| B | 15 | 0 | 1 | 2 |
| C | 3 | 1 | 0 | ∞ |
| D | ∞ | 2 | ∞ | 0 |

3 pt (b) Give, in table form, the distances stored when each router has reported the information from the previous subquestion to its immediate neighbours, i.e., after one iteration of the distance vector algorithm

| Information | Distance to reach node | | | |
|---|---|---|---|---|
| stored at node | A | B | C | D |
| A | 0 | 4 | 3 | 17 or 6 |
| B | 4 | 0 | 1 | 2 |
| C | 3 | 1 | 0 | 3 |
| D | 17 | 2 | 3 | 0 |

Let us now assume that the distance vector algorithm has converged, and all nodes know the shortest path and distance to all other nodes.

1 pt (c) Give, in table form, the distances stored at this moment.

| Information | Distance to reach node | | | |
|---|---|---|---|---|
| stored at node | A | B | C | D |
| A | 0 | 4 | 3 | 6 |
| B | 4 | 0 | 1 | 2 |
| C | 3 | 1 | 0 | 3 |
| D | 6 | 2 | 3 | 0 |

Now, the link between B and C disappears (breaks), which both B and C get to know.

3 pt (d) Give, in table form, the distances stored after B and C have recalculated[1] their routing table, but before they have sent any updates to their neighbours.

| Information | Distance to reach node | | | |
|---|---|---|---|---|
| stored at node | A | B | C | D |
| A | 0 | 4 | 3 | 17 |
| B | 8 | 0 | 5 | 2 |
| C | 3 | 7 | 0 | 9 |
| D | 6 | 2 | 3 | 0 |

2 pt (e) How many updates will node B send to node D before node D learns the real cost to node C? Explain your answer.

In its 1st update, B will report a cost of 5 to C. D will report back that his cost to C has increased to 7. In the 2nd update, B will report a cost of 9 to C, and so on. The sequence of reported costs to C (as sent by B to D) is: 5, 9, 13, 17, 18. This last value comes from the cost reported by A (3) + the link cost to A (15), which is the real cost. So, B will send node D 5 updates before D learns the real cost to C.

*Continued on next page...*

[1] you may assume they have kept a copy of the last distance vectors they received from their neighbours

**2. Addressing issues**

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, so we cannot have more than $2^{48}$ computers.

2 pt (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?

Because we want to assign the addresses hierarchically for efficient routing.

3 pt (b) Which of the following are correct resp. incorrect notations for IPv6 addresses? Why? Which of these addresses is/are in the `2001:0600::/23` range?

```
2001:0600::1
2001::0600:1
2001::0600::1
2001:0700::abcd:1
::1
```

Only the 3rd one is incorrect (it has two :: marks).
The 1st and the 4th are in the 2001:0600::/23 range.

3 pt (c) Consider an organization which has a /20 block of IPv4 addresses. Using the Host-Density ratio, argue how many computers the network of this organization could comfortably contain.

In a /20 address block, the first 20 address bits identify the network, leaving 12 for identifying the host: $2^{12}$ addresses. HD-ratio of about 0.87 is the upper threshold for comfortable assignment. Denote the maximum comfortable number of hosts by $N$, then solve $0.87 = \log(N)/\log(2^{12})$, so $N = 2^{0.87\cdot 12} \approx 1400$ hosts.

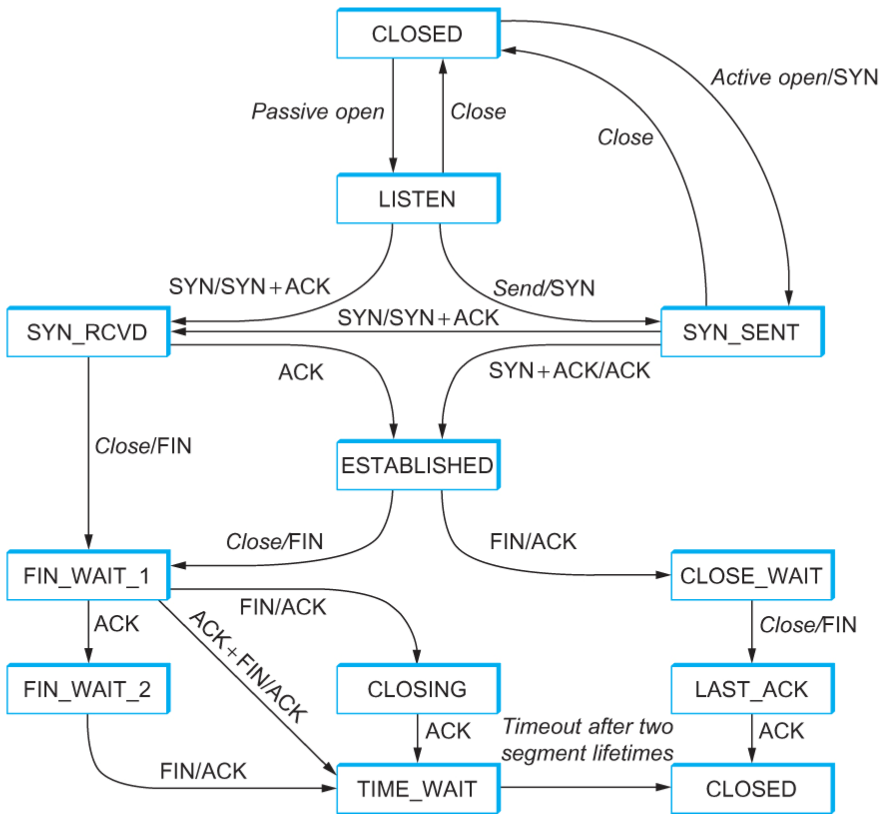
3 pt (d) Suppose an internet service provider has run out of IPv4 addresses and decides to use NAT (Network Address Translation) to solve this problem. Can an unlimited number of computers share a single IPv4 address using NAT, or are there practical limitations? If you do find one or more limitations, discuss it/them both qualitatively (i.e., what causes the limitation) and quantitatively (how many computers).

It's limited: if all those computers behind the NAT want to connect to a single TCP port on a single server outside the NAT, the NAT has to distinguish the connections by port numbers, of which there are only 65536. So if it is to be expected that all computers may want to connect to say google's web service simultaneously, 65536 is an absolute maximum. In practice, you'd probably want to allow multiple such connections per computer, reducing the maximum even further.
Also, the number of available IPv4 addresses for local use is limited; the biggest address block for local use is 10.0.0.0/8, which has 16 million addresses (although this could be overcome by multiple levels of NAT). But anyway, the 65536 port numbers mentioned above cause a much lower limit.

*Continued on next page...*

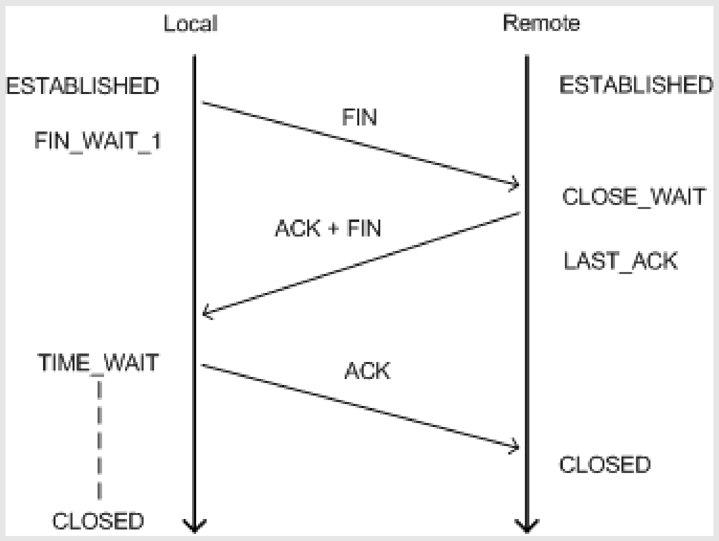
**3. TCP**

Figure 5.7 of Peterson & Davie displays the state-transition diagram of TCP. For your convenience, this figure is reprinted below.



3 pt (a) Give a time-sequence diagram (similar to Figure 5.6 of Peterson Davie) that shows how a connection can be closed in such a way that the local side goes from the state `ESTABLISHED` to the state `CLOSED` using the diagonal transition labelled `ACK+FIN/ACK`.

Note: draw a time-sequence diagram where the local side is depicted on the left, where the messages transferred are labelled with the type of message (or the flag set). Denote on both on the outer side of both time-lines all states the local (left) or remote (right) side is going through.



3 pt (b) In which of the states in the figure above can a node receive new user data and pass that data to the application. Explain your answer.

In the state `ESTABLISHED` a TCP node can expect data from the remote side. Also, when it has sent a `SYN+ACK` in response to a `SYN` (in state `SYN_RCVD`), the remote side may send data together with the `ACK`. Finally, until the local side has received a `FIN` message from the remote side, it can expect new data that has to be passed to the application. So these are the states `FIN_WAIT_1` and `FIN_WAIT_2`.

3 pt (c) In its acknowledgements to the sender, the TCP receiver sets the `AdvertisedWindow` to the value

$$\texttt{AdvertisedWindow} = \texttt{MaxRcvBuffer} - ((\texttt{NextByteExpected} - 1) - \texttt{LastByteRead})$$

This is basically the difference between the capacity of the buffer (`MaxRcvBuffer`) and the bufferspace already filled with data. However, some out-of-sequence data may also be present in the buffer (in that case `NexByteExpected` - 1 does not equal `LastByteRcvd`). Why does that not play a role in the setting of `AdvertizedWindow` (i.e., why can `LastbyteRcvd` not be found in the formula above)?

This out-of-sequence data has not yet been acknowledged to the sender, so for the sender it is part of the amount of data that may be outstanding, and that should fit in the receiver's buffer.

---

*End of this exam.*