Chalmers University of Technology
Computer Science and Engineering
Elad Michael Schiller

2014-08-28

Written exam in EDA387/DIT663 Computer Networks 2014-08-28. Exam time: 4 hours.

Means allowed: Nothing except paper, pencil, pen and English - xx dictionary.

Examiner: Elad Michael Schiller, phone: 073-6439754
Note that student questions can be answered only by phone.

Credits:	30-38	39-47	48-Max
Grade:	3	4	5
Grade (GU)	G	G	VG

1. The answer must be written in English (even for Swedish students). Use proper grammar and punctuation.
2. All answers need to be motivated, unless otherwise stated. Correct answers without motivation or with wrong motivation will not be given full credit.
3. Answer concisely, but explain all reasoning. Draw figures and diagrams when appropriate.
4. Write clearly. Unreadable or hard-to-read handwriting will not be given any credit.
5. Do not use red ink.
6. Solve only one problem per page.
7. Sort and number pages by ascending problem order.
8. Anything written on the back of the pages will be ignored.
9. Do not hand in empty pages or multiple solutions to the same problem. Clearly cross out anything written that is not part of the solution.

Question 1 DNS (12 points)

The Internet **Domain Name System** is hierarchical and implemented as a huge distributed database. Answer the questions given below using **DNS-terminology and concepts**. Please answer each question separately.

(4 points) 1a. Explain the structure of the system by describing clearly **the naming hierarchy** and **the tree of DNS servers**.

(4 points) 1b. Mention and describe the **meaning and contents** of at least **four types** of Resource Records maintained in the DNS database.

(4 points) 1c. Suppose that you are using the Chalmers network to connect your laptop to the Internet. Suppose also that you want to access the web site www.tue.nl for the first time. Explain **how and why** DNS will be involved immediately after entering the name of the site in your browser. The answer should, specifically and technically, explain the necessary operation, including:

- the interaction and communication between the different DNS resolvers and servers,
- the protocols and messages used, and
- the final outcome.

Question 2 IPv6 Addresses (4 points)

(1p) 2a. Explain what is meant by the "scope" of an IPv6 address used as destination address in IPv6 packets.

(3p) 2b. What is the "type" and the "scope" of addresses which are configured out of each of the following IPv6 prefixes:

2001:6b0:2:10:: FF02:: FE80::

Question 3 IPv6 Autoconfiguration (6 points)

An IPv6 node will be connected to the Internet through an Ethernet-based network. The network has an attached router-interface which is periodically advertising the prefix 2001:06b0:2:10::/64. The node is configured to use stateless autoconfiguration of the interface's IPv6 addresses. Assume that the node's interface identifier is a288:b4ff:fe5c:c774.

(1p) 3a. When rebooting, what IPv6-address will be automatically configured for the interface in order to be used in Neighbor Discovery? When answering, please write the address in hexadecimal notation and give the type and scope of its use.

(1p) 3b. After rebooting, what IPv6-address will be automatically configured for the interface in order to be used for accessing the Internet? When answering, please write the address in hexadecimal notation and give the type and scope of its use.

(4p) 3c. Since there is neither broadcasting nor ARP in IPv6, describe the substituting operation, its purpose, the protocols and the messages used by this IPv6 node and its neighbors.

Question 4 (6 points)

Below please find an algorithm for digital clock synchronization. Namely, eventually it holds that we have: (1) identical clock values, and (2) the clock values are incremented by one, once in every pulse.

```
01 upon a pulse
02       forall P_j ∈ N(i) do send (j,clock_i)
03       max := clock_i
04       forall P_j ∈ N(i) do
05              receive(clock_j)
06              if clock_j > max then max := clock_j
07       od
08       clock_i := max + 1
```

4.a (2 p) The algorithm does not consider bounded set of values for the variable *clock*. Explain why in the context of self-stabilization this property is considered unattractive.

4.b (4 p) Is there any deterministic self-stabilizing algorithm (with bounded clock counter values)? If the answer is no, please give a formal impossibility proof. If the answer is yes, please write the code of the algorithm (or just say which line to change) and explain why the algorithm converges from any starting configuration.

Question 5 (5 points)

Please find below Dijkstra's self-stabilizing algorithm for token circulation, as well as the proof outline, see Lemma 2.2 to 2.4 and Theorem 2.1. Please prove one of them, i.e., either Lemma 2.2, 2.3, 2.4 or Theorem 2.1, but *just one of them*!

```
01 P1:          do forever
02                  if x1=xn then
03                      x1:=(x1+1)mod(n+1)
04 Pi(i ≠ 1):   do forever
05                  if xi ≠ xi-1 then
06                      xi:=xi-1
```

A configuration in which all x variables are equal, is a safe configuration for ME (Lemma 2.2)

For every configuration there exists at least one integer j such that for every p_i, x_i is not equal to j (Lemma 2.3)

For every configuration c, in every fair execution that starts in *c*, P_1 changes the value of x_1 at least once in every *n* rounds (Lemma 2.4)

For every possible configuration c, every fair execution that starts in *c* reaches a safe configuration with relation to ME within $O(n^2)$ rounds (Theorem 2.1)

Question 6 (5 point) Multiple Choice and fill in questions

Instructions: Select the single correct choice (labeled i, ii, iii, iv, v or vi) among the available options. Each correct answer will give you the number of points that is written next to the question title. **Zero** points are given for a blank answer, or an **incorrect** answer.

Please write down the **question letter and the full text of the answer** so we can avoid confusion!

6.a (1 p) With respect to Dijkstra's Mutual Exclusion, what is not a safe configuration, where (x1,… xn) denote the system configuration.

i. (0,0, … 0)
ii. (1,…1, 0, … 0)
iii. (0,0,… 1,… 1)
iv. (1, … 1)
v. None of the above.
vi. All of the above.

6.b (1 p) Lemma 2.2 shows that a configuration in which all x variables are equal, is a safe configuration for Dijkstra's Mutual Exclusion. Are these the only safe configuration?

i. Yes
ii. No, because as long as the system has not stabilized in configurations in which all x variables are equal.
iii. No, because there are also configurations in which there are more than just one token in the system.
iv. None of the above.
v. All of the above.

6.c (1 p) Recall Dijkstra's Mutual Exclusion. P1: do forever if x1=xn then x1:=(x1+1)mod(n+1) and Pi(i <> 1): do forever if xi <> xi-1 then xi:=x(i-1)
Lemma 2.3 says that for every configuration there exists at least one integer j such that for every i it holds that xi <>j. What do we need to change in the algorithm if we want to make sure that at any time there are at least two such integer values.

i. Lemma 2.3 actually shows that after convergence we reach a safe configuration in which there are at least two such integers.

ii. It is not possible to change the algorithm to have at least two integer values.

iii. P1: do forever if x1=xn then x1:=(x1+1)mod(n/2) and Pi(i <> 1): do forever if xi <> xi-1 then xi:=xi-1

iv. P1: do forever if x1=xn then x1:=(x1+1)mod(n+2) and Pi(i <> 1): do forever if xi <> xi-1 then xi:=xi-1

v. None of the above.

vi. All of the above.

6.d (1 p) The synchronous consensus task is

i. A set of (synchronous) executions, SC, in which the output variable of every processor is 0, if and only if, the input value of all processors is 0
ii. A set of (synchronous) executions, SC, in which the output variable of every processor is 0, if and only if, there is no input value with value 1
iii. A set of (synchronous) executions, SC, in which the output variable of every processor is 1, if and only if, there exists an input value with value 1
iv. None of the above.
v. All of the above.

6.e (1 p) The task of digital clock synchronization has to make sure that: (1) all ___ have ___ values, and (2) the ___ values are ___ by ___ in every ___.

Question 7 (8 points)

Please find below a self-stabilizing algorithm for leader election.

7.a **(3 p)** Please define the safe configuration of the algorithm. Make sure that you consider all variables and shared registers.

7.b **(5 p)**. Suppose the system execution, *R*, starts in a safe configuration, *c*. Let a_i be a step that processor *pi* takes immediately after *c* and just before c'. Please show that c' is safe.

```
01 do forever
02     ⟨candidate,distance⟩ = ⟨ ID(i), 0 ⟩
03     forall P_j ∈ N(i) do
04         begin
05             ⟨leader_i[j],dis_i[j]⟩ := read⟨ leader_j, dis_j ⟩
06             if (dis_i[j] < N) and ((leader_i[j] < candidate) or
07                 ((leader_i[j] = candidate) and (dis_i[j] < distance))) then
08                     ⟨candidate,distance⟩ := ⟨ leader_i[j], dis_i[j] + 1⟩
09         end
10     write ⟨leader_i , dis_i⟩ := ⟨candidate,distance⟩
11 od
```

Question 8 (6 points)

Take a look at the self-stabilizing maximal matching algorithm. We assume the existence of a central daemon. Given a configuration *c*, we say that a processor p_i is:

- **matched** in *c*, if p_i has a neighbor p_j, such that $pointer_i = j$ and $pointer_j = i$.
- **single** in *c*, if $pointer_i = null$ and every neighbor of p_i is matched.
- **waiting** in *c*, if p_i, has a neighbor p_j such that $pointer_i = j$ and $pointer_j = null$.
- **free** in *c*, if $pointer_i = null$ and there exists a neighbor p_j, such that p_j is not matched.
- **chaining** in *c*, if there exists a neighbor p_j for which $pointer_i = j$ and $pointer_j = k$, $k \neq i$.

We define the variant function *VF(c)* as one that returns a vector (*matched + single, waiting, free, chaining*).

8.a (2 p) Please use the value of *VF(c)* to define the safe configuration, c. Hint: it is a vector that includes values that are either *0* or *n*. For that value show that: (1) $pointer_i = j$ implies that $pointer_j = i$, and (2) if $pointer_i = null$ then there is no neighbor p_j, such that if $pointer_j = null$.

8.b (4 p) Suppose that *c* is safe. Let a_i be a step of processor p_i that is taken immediately after *c*. Moreover, let *c'* be configuration that immediately follows by a_i. Show that c' is safe, i.e., the closure property. Hint: consider the case that a_i includes the execution of either line 02, 03 or 04. For each of these three cases, show that *VF(c) = VF(c')*.

Program for P_i:

01 **do** forever

02 **if** $pointer_i = null$ **and** $(\exists P_j \in N(i) \mid pointer_j = i)$ **then** $pointer_i = j$

03 **if** $pointer_i = null$ **and** $(\forall P_j \in N(i) \mid pointer_j \neq i)$ **and** $(\exists P_j \in N(i) \mid pointer_j = null)$ **then** $pointer_i = j$

04 **if** $pointer_i = j$ **and** $pointer_j = k$ **and** $k \neq i$ **then** $pointer_i = null$

05 **od**

Question 9 Socket API and TCP/IP (8 points):

9.a (2 p) Someone has suggested shortening the TIME_WAIT state duration. What could be the outcome of this suggestion?

9.b (1 p) How many simultaneous socket connections possible? What does it depends on?

9.c (2 p) Is congestion control in the Internet done through an end-to-end method or network-assisted method? Explain your answer.

9.e (3p) What is the effect of TCP's congestion control and error control in real-time traffic? Explain your answer

---