Computer Science and Engineering
Elad Michael Schiller
Chalmers University of Technology

2017-10-27

Written exam in EDA387/DIT663 Computer Networks 2017-10-27. Exam time: 4 hours.

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

Examiner: Elad Michael Schiller, phone: 073-6439754 and 031-7721052

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 (10 points)
Please remember that you should use DNS terminology in your answers of the following sub-questions.
1a. (3 points)

- Which type of server will normally perform recursion for DNS-clients within their own network?

- Explain carefully the advantages and why it is important and useful to have such a server in an organization's network.

1b. (4 points)

- Suppose that you are using one of the computers connected to the Chalmers network; like the computer you used during the course labs. Suppose also that you want to access the web site www.tue.nl for the first time (the homepage not in cache). Explain **how and why** DNS will be involved immediately after entering the name of the site in your browser. Your answer should; **specifically and technically**, describe the necessary operation, including:

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

1c. (3 points)
dig *(domain information groper) is a useful command-line tool for querying the name system of the Internet. This tool has been used extensively during one of the course labs.*
Consider that a PC-user issues the <dig> command in order to get a specific DNS-information. Examine the output carefully and then answer the question given below using DNS-terminology and concepts.

```
C:\dig>dig www.wittenborg.eu @ns1.google.com

; <<>> DiG 9.3.2 <<>> www.wittenborg.eu @ns1.google.com
; (1 server found)
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: REFUSED, id: 1778
;; flags: qr rd; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;www.wittenborg.eu.              IN      A

;; Query time: 43 msec
;; SERVER: 216.239.32.10#53(216.239.32.10)
;; WHEN: Wed Oct 18 16:26:39 2017
;; MSG SIZE  rcvd: 35
```

Task & Questions:

- Explain and discuss the syntax of the issued command as well as the shown output after executing it.
- What DNS information does the user **specifically** want to know? Give the full name of the object **type** that has been queried.
- Did the user get answer containing the queried information? Explain why or why not in this case.

Question 2: IPv6 and ICMPv6 (12 points)

2a. (2 points)

- What does it mean that a host (or a node) has **IP dual stack**?
- Which "stack" is considered here as being dual?
- Explain carefully and it is beneficial if your answer is combined with a sketch of the dual stack.

.....

2b. (8 points)

Assume that an IPv6 node has an Ethernet-interface with the MAC address D4-85-64-5F-AB-EC.

- Is EACH of the following a correct and valid representation of an IPv6 address for the interface?

  (i) 2001:06b0:0001:1300:D685:64FF:FE5F:ABEC

  (ii) FF02:0000:0000:0000:0000:0001:FF5F:ABEC

  (iii) FE80:0000:0000:0000:D685:64FF:FE5F:ABEC

  (iv) FF02:0000:0000:0000:0000:0000:0000:0001

  - If NOT, explain clearly "why not" and "what is wrong".
  - If YES; for **each** correct and valid address:

    - rewrite the IPv6 address using optimal zero-compression.

    - give the **type** of the address explaining what the TYPE does imply when the address is used in IPv6 packets sent to the node.

    - give the **scope** of the address explaining what the SCOPE does imply when the address is used in IPv6 packets sent to the node.

2c. (2 points)

Suppose now that the same node (in question **2b.**) is preparing to send IPv6 packets containing data from a certain application. The target will be another node with the destination address:

2001:6B0:1:1300:20C:F1FF:FE6C:14DD

Assume also that there has not been any earlier communication with this destination.

What will the source node do firstly before it will be able to send the packets encapsulated in Ethernet frames? **Describe** the operation in detail with relationship to **this case** in question.

* *Please note that general statements in your answer will give no points.*

Question 3: Vertex coloring (12 points)

Below please find two algorithms for vertex coloring. In Figure 1, you can find a self-stabilizing algorithm and in Figure 2, you can find a super-stabilizing one. Note that δ refers to the number of nodes and ID(m) returns the unique identifier of node p_m. The operator \\ on set A refers to the completely set of A with respect to the color palette.

```
01 Do forever
02        GColors := {}
03        For m:=1 to δ do
04                lr_m:=read(r_m)
05                If ID(m)>i then
06                   GColors := GColors U {lr_m.color}
07        od
08        If color_i  in   GColors then
09          color_i:=choose(\\ GColors)
10        Write r_i.color := color_i
11 od
```

Figure 1 A Self-Stabilizing Algorithm for Graph Coloring

```
01  Do forever
02          AColors := ∅
03          GColors := ∅
04          For m:=1 to δ do
05                  lr_m:=read(r_m)
06                  AColors:=AColors U {lr_m.color}
07                  If ID(m)>i then GColors := GColors U {lr_m.color}
08          od
09          If color_i = ⊥ or color_i ∈ GColors then
10                  color_i:=choose(\\ AColors)
11          Write r_i.color := color
12  od
13  Interrupt section
14          If recover_ij and j > i then
15                  Color_i := ⊥
16                  Write r_i.color := ⊥
```

Figure 2 A Super-Stabilizing Algorithm for Graph Coloring

3a. (1 point) Please define the task of vertex coloring.
3b. (4 points) Please show the time it takes the self-stabilizing algorithm to recover (in terms of asynchronous cycles) when starting from an arbitrary starting configuration. Prove your claims.
3c. (4 points) Please show the time it takes the super-stabilizing algorithm to recover (in terms of asynchronous cycles) when starting from an arbitrary starting configuration. Prove your claims.
3d. (3 points) What are the advantages that a super-stabilizing algorithm has over a self-stabilizing one? Please use the above two algorithms as an example that illustrates your claims.

Question 4: Token circulation in a directed ring with a distinguished node (12 points)

4a. (2 points) Define the task of token circulation.
4b. (5 points) Show that, without a distinguished processor, there is no deterministic solution for the problem of self-stabilizing token circulation.

Below (in Figure 3) please find Dijkstra algorithm for token circulation in directed rings as well as the proof.

```
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
```

Figure 3 Dijkstra algorithm for token circulation

- 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, the value of 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)

4c. (5 points) Both the proof of Lemma 2.2 and 2.4 make arguments about the propagation of x_1's value within n-1 asynchronous cycles. Please compare the different ways in which the proofs of these two lemmas use this propagation argument. What is similar? What is different?

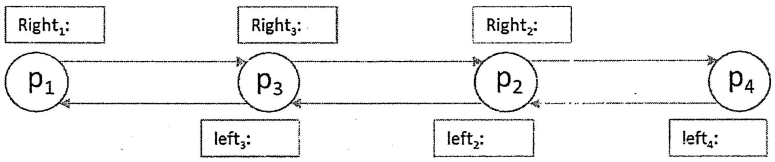
Question 5: Self-stabilizing maximum matching on a directed ring with a distinguished node (12 points)
Let $p_0, \ldots, p_{n-1}$ be n processors on a directed ring that Dijkstra considered in his self-stabilizing token circulation algorithm (in which processors are semi-uniform, i.e., there is one distinguished processor, p_0, that runs a different program than all other processors but the processors have no unique identifiers). Recall that in Dijkstra's directed ring, each processor p_i can only read from his predecessor's shared variable and each processor can use shared variables of constant size.

Design a deterministic self-stabilizing matching algorithm for an asynchronous system that, after the recovery period, provides an optimal solution with respect to the number of matched edges. Please provide a clear system description and *list clearly all of your assumptions*. In particular, describe the fields that each register uses and which of these fields encode the matching solution. Note that the matched edges are directed and the encoding scheme might be (slightly) different than the one presented in class for general undirected graphs.

5a. (3 points) Please give a clearly written pseudo-code.
5b. (2 points) Please define the set of legal executions.
5c. (4 points) Please provide a correctness proof with all the arguments needed to convince the reader that the algorithm is correct and self-stabilizing.
5d. (1 points) Show that your algorithm is always optimal after convergence with respect to the number of matched edges. Prove your claims.
5e. (2 points) What is the number of states (in the finite state machine that represents each processor) that your algorithm needs? In case your solution uses more than O(1) bits, can you show that there is no O(1) bits solution? Prove your claims.

Question 6: Self-stabilizing counting algorithm on an anonymous and oriented path graph (12 points)

Consider a path graph P_n that has n nodes, where $n < N$ is a finite number that is unknown to the algorithm and N is an upper-bound on n such that only $\lceil \log_2 N \rceil$ is known. The network is based on shared memory where each processor can write to a single register that has up to $O(\lceil \log_2 N \rceil)$ bits. The node p_i can read its neighbours registers, $left_i$ and $right_i$ (when they exists). The nodes do not have unique identifiers and all nodes run the same program without the presence of a distinguished processor. However, the path is oriented from left to right in the sense that each node has (at most one) right neighbour and starting from the most left node (who has no left neighbour), and then taking exactly n hops to the right brings us to the most right node (who has no left neighbour). Below please find an example for the case of n=4.



Design a self-stabilizing algorithm for an asynchronous system that counts the number of nodes in the system. That is, after the recovery period, at least one specific processor would know the exact number of nodes in the system. Please provide a clear system description and *list clearly all of your assumptions*.

6a. (3 points) Please give a clearly written pseudo-code.
6b. (1 points) Please define the set of legal executions.
6c. (4 points) Please provide a correctness proof with all the arguments needed to convince the reader that the algorithm is correct and self-stabilizing.
6d. (4 points) Is there a self-stabilizing naming algorithm for the system that this question considers? In case there is, please provide an algorithm and a correctness proof. In case there is not, please provide the poof of an impossibility result.
