Chalmers University of Technology
Computer Science and Engineering
Elad Michael Schiller

2015-01-05

Written exam in EDA387/DIT663 Computer Networks 2015-01-05. 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 Lab (6 points)

A user issued the following dig-command to find specific DNS information. The output of running the command is shown below.

```
c:\dig>dig -x 129.16.212.14 @ns1.chalmers.se

; <<>> DiG 9.3.2 <<>> -x 129.16.212.14 @ns1.chalmers.se
; (1 server found)
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 945
;; flags: qr aa rd; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 0

;; QUESTION SECTION:
;14.212.16.129.in-addr.arpa.     IN      PTR

;; AUTHORITY SECTION:
16.129.in-addr.arpa. 600 IN SOA ns1.chalmers.se. cth-nic.chalmers.se.
2014100809 14400 3600 1209600 600

;; Query time: 41 msec
;; SERVER: 129.16.2.40#53(129.16.2.40)
;; WHEN: Wed Oct 08 15:01:13 2014
;; MSG SIZE  rcvd: 103
```

1a. (1 point) What did the user want to ask about?

1b. (1 point) Explain the contents of the "QUESTION SECTION". What is the name of the object type that was queried?

1c. (2 point) To which DNS server (hostname and IP-address) was the query sent? Is it an authoritative server or not? Explain and point out how you are able to confirm that it is or not.

1d. (2 point) Is there any answer in the reply? What information did the reply give to the user?

Question 2: DNS (6 points)

The cache-only local name server (**res2.chalmers.se**) has received a standard query from a local client about type **MX** for the name "**msn.com**". The client is running on a laptop using NOMAD-connection. Suppose that the server has no such information in the cache.

2a. (1 point) What DNS-information has the client queried to get answer about?

2b. (4 points) Describe clearly the steps that the server will perform in order to provide a reply of the query to the client. Please remember that you should use **DNS terminology** when describing the different steps that the server has to do when trying to resolve this NS lookup.

2c. (1 point) What will the local server do after obtaining the required DNS information? Explain.

Question 3: IPv6 Addresses and Configuration (10 points)

Note: Please answer these sub-questions separately.

3a. (2 points) What are the types of IPv6 **destination** addresses? Explain briefly the use of three types with regards to delivering IPv6 packets to destinations. (**Note**: IPv6-address type is not the scope).

3b. (1 point) Rewrite the text representation of the following IPv6 address prefix using complete and optimal zero-compression.

FF02:0000:0000:0000:0000:0001:FF00:0000/104

3c. (1 point) Rewrite the text representation of the following IPv6-address prefix using complete and optimal zero-compression.

2001:0DB8:0000:CD30:0000:0000:0000:0000/64

3d. (2 points) What are the different address **scopes** of each of the IPv6 addresses associated with an interface? Describe three important IPv6 address scopes. (**Note**: IPv6-address scope is not the type).

3e. (4 points) Explain the stateless autoconfiguration of IPv6-addresses for an interface that is connecting an IPv6-node to a local Ethernet-based network. When answering, please consider to explain the procedure (during and after rebooting) needed to autoconfigure the necessary IPv6-addresses for the interface so that the node will be able to communicate with other IPv6-nodes using the local connection.

Question 4 (6 points) Socket API: `select()`

The following program part contains at least one flaw. Identify and describe the flaw in few short sentences or points. You do not have to correct the flaw; you should just find and describe it! (Note: you're not looking for, e.g., syntax errors. Find conceptual flaws in the program.)
Hint: The program uses `select()` and they are supposed to be non-blocking. Consider which operations can actually block the processes that execute these programs.
The following program accepts new connections using the listenfd socket. The first byte sent by a client is expected to be an 8-bit ID.

- You may assume that the handle_*_error() methods do something sensible.
- The helper method register_client(client, id) verifies the client ID is acceptable and if that is the case, enters the client into a global list. Otherwise it closes the connection.
- The method add_client_sockets_to_readfds() properly adds all active clients in the global list to the readfds. It returns the largest socket number it encounters.
- handle_registered_clients() handles clients that are ready to send data according to readfds, and removes clients that close their associated connections from the global list. No data is ever sent to the clients, the program only receives and processes data sent to it.

```
int main() { /* includes, declarations, etc. */
    int listenfd = -1; /* initialization code, such as setting up a
listening socket on listenfd, has been omitted - this is not the error you're looking
for */
    while( 1 ) {
        fd_set readfds; // initialize read set
        FD_ZERO( &readfds );
        int maxfd = add_client_sockets_to_readfds( &readfds );
        FD_SET( listenfd, &readfds );
        if( listenfd > maxfd ) maxfd = listenfd;
        int ret = select( maxfd+1, &readfds, 0, 0, 0 ); // call select
        if( -1 == ret ) handle_select_error();
        if(FD_ISSET(listenfd,&readfds)){//is there a waiting client?
            sockaddr_in clientAddr;
            socklen_t clientAddrLen = sizeof(clientAddr);
            int client = accept( listenfd,
                (sockaddr*)&clientAddr,
                &clientAddrLen
            );
            if( -1 == client ) handle_accept_error();
            unsigned char id; // receive 8bit client ID
            int ret = recv( client, &id, sizeof(id), 0 );
            if( 0 == ret ) {
                close( client );
                continue;
            }
            if( -1 == ret ) handle_recv_error();
            register_client( client, id ); // register client
        }
        handle_registered_clients(&readfds);//handle registered clients
    }
    return 0;
}
```

Question 5 (8.5 points)

We learned in class a self-stabilizing algorithm for BFS spanning tree construction, see the code below. We define a *floating distance* in configuration c, as a value stored in $r_{ij}.dis$ that is smaller than the distance of p_i from the root, where *dis* is the distance field of the registers.

```
01 Root: do forever
02         for m := 1 to δ do write r_im := ⟨0,0⟩
03       od
04 Other: do forever
05         for m := 1 to δ do lr_mi := read(r_mi)
06         FirstFound := false
07         dist := 1 + min{lr_mi.dis | 1 ≤ m ≤ δ }
08         for m := 1 to δ
09         do
10             if not FirstFound and lr_mi.dis = dist -1
11                 write r_im := ⟨1,dist⟩
12                 FirstFound := true
13             else
14                 write r_im := ⟨0,dist⟩
15         od
16       od
```

Prove that for every $k > 0$ and for every configuration that follows $\Delta + 4k\Delta$ rounds, it holds that:

- If there exists a floating distance, then the value of the smallest floating distance is at least k.
- The value in the registers of every processor that is within distance k from the root is equal to its distance from the root.

Proof. Note that in every 2Δ successive rounds, each processor reads the registers of all its neighbors and writes to each of its registers. We prove the lemma by (1) _____ k.

Base Case: Proof for k=1. Distances stored in the registers and internal variables are non-negative; thus the value of the smallest floating distance is at least 0 in the first configuration. During the first 2Δ rounds, each non-root processor p_i, computes the value of the variable *dist* (line 7). The result of each such computation must be (2) _____ 1. Let c_2 be the configuration reached following the first computation of the value of *dist* by each processor.

Each non-root processor writes to each of its registers the computed value of *dist* during the 2Δ rounds that follow c_2. Thus, in every configuration that follows the first 4Δ rounds there is no non-root processor with value 0 in its registers. The above proves (3) _____.
To prove (4) _____, note that the root repeatedly writes the (5) _____ to its registers in every (6) _____ rounds. Let c_1 be the configuration reached after these (7) _____ rounds. Each processor reads the registers of the root and *then* writes to its own registers during the 4Δ rounds that follow (8) _____. In this write operation the processor assigns (9) _____ to its own registers. Any further read of the root registers returns the value (10) _____; therefore, the value of the registers of each neighbor of the root is (11) _____ following the first $\Delta + 4\Delta$ rounds. Thus, (12) _____ holds as well.

Induction Step. We assume correctness for k(13) _____ 0 and prove for k + 1. Let $m \geq k$ be the smallest floating distance in the configuration c_{4k} that follows the first $\Delta + 4k\Delta$ rounds. During the 4Δ rounds that follow c_{4k}, each processor that reads m and chooses m as the smallest value assigns (14) _____ to its distance and writes this value. Therefore, the smallest floating distance value is $m + 1$ in the configuration $c_{4(k+1)}$. This proves (15) _____.

Since the smallest floating distance is m (15)_____ k, it is clear that each processor reads the distance of a neighboring processor of distance k and assigns (16)_____ to its distance. ■

Question 6 (8 points)

We learned in class several algorithms for self-stabilizing clock synchronization. Please find below the code of a couple of them, which we call: converge-to-the-min and -max.

Converge-to-the-max

```
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) mod ((n+1)d+1)
```

Converge-to-the-min

```
01 upon a pulse
02      forall P_j ∈ N(i) do send (j,clock_i)
03      min := clock_i
04      forall P_j ∈ N(i) do
05              receive(clock_j)
06              if clock_j < min then min := clock_j
07      od
08      clock_i := (min + 1) mod (2d+1)
```

6.a **(2 point)** What do the constants d and n stand for?
6.b **(1 point)** Please compare these two algorithms with respect to their scalability property. Which one scales better? Why?
6.c **(1 point)** Please compare these two algorithms with respect to the service provided to the application layer. Which one is easier to work with? Why?
6.d **(4 point)** Please complete the correctness proof of the algorithm converge-to-the-min

Suppose that no processor (1)____ during the first (2)_____ pulses. Then we can use simple (3)_____ to show that synchronization is achieved. Otherwise, a processor (4) ____ during the first (5) _____ pulses. Therefore, (6)_____ pulses after this point a configuration c is reached, such that there is no clock value greater than (7)____: the first (8) _____.

Question 7 (7.5 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 both Lemma 2.4 and Theorem 2.1, but there is no need to prove Lemmas 2.2 and 2.3 (and therefore they are ~~strikethrough~~ in the text)!

```
01 P_1:          do forever
02                      if x_1=x_n then
03                              x_1:=(x_1+1)mod(n+1)
04 P_i(i ≠ 1):   do forever
05                      if x_i ≠ x_{i-1} then
06                              x_i:=x_{i-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)~~

7.a (4 point) 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)
7.b (3.5 point) 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 8 (6 points)

Show that in a synchronous uniform (anonymous) ring, there is no deterministic self-stabilizing algorithm for token circulation. (Hint: The answer is similar to the leader election question of the first assignment.)

Question 9 (2 points)

9.a **(1 point)** Write the definition of: the term set of legal executions and the used it to define the term safe configuration.

9.b **(1 point)** Below please find a leader election algorithm. The question is whether this algorithm is self-stabilizing. In case you believe that the answer is positive, please prove your answer. Otherwise, please give a starting configuration, c, such that every fair execution that starts from c does not satisfy the task of self-stabilizing leader election.

1. write *id* to r_i
2. for m := 1 to n do lr_m := read(r_m)
3. Leader := (id == *maximum* $\{lr_m.\text{id} \mid 1 \leq m \leq n\}$)
4. (* if *Leader* == True then *act _like_a_leader()* *)