Chalmers University of Technology
Computer Science and Engineering
Elad Michael Schiller

2015-08-27

Written exam in EDA387/DIT663 Computer Networks 2015-08-27. 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 mx ntnu.no @ns1.ntnu.no

; <<>> DiG 9.3.2 <<>> mx ntnu.no @ns1.ntnu.no
; (1 server found)
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 1780
;; flags: qr aa rd; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 3

;; QUESTION SECTION:
;ntnu.no.                       IN      MX

;; ANSWER SECTION:
ntnu.no.                60      IN      MX      10 mx.ntnu.no.

;; AUTHORITY SECTION:
ntnu.no.                3600    IN      NS      ns2.ntnu.no.
ntnu.no.                3600    IN      NS      ns1.ntnu.no.

;; ADDITIONAL SECTION:
mx.ntnu.no.             600     IN      A       129.241.56.67
ns1.ntnu.no.            3600    IN      A       129.241.0.208
ns2.ntnu.no.            3600    IN      A       129.241.0.209

;; Query time: 32 msec
;; SERVER: 129.241.0.208#53(129.241.0.208)
;; WHEN: Mon Aug 17 11:52:11 2015
;; MSG SIZE  rcvd: 240
```

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? Describe in detail.

Question 2: DNS (6 points)

The cache-only local name server (**res1.chalmers.se**) has received a standard query from a local client about type **ns** for the name "**ibm.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 DNS lookup.

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

Question 3: IPv6 Addresses and Neighbor Discovery (10 points)

These three addresses are given with IPv6 representation:

(i) 2001:6b0:2:10::1
(ii) FF02::1:ff6c:14dd
(iii)FE80::20c:f1ff:fe6c:14dd

Note: Please answer the following sub-questions (3a,3b,3c, and 3d) separately and in relation to the above addresses. In addition you are allowed to use any of the given addresses as examples in your answers to sub-questions 3e and 3f.

3a. (1 point) Decompress and rewrite each of the given addresses showing all hexadecimal digits.

3b. (2 points) What is the "type" of each of these IPv6 addresses? Explain what each type does imply.

3c. (1 point) Which of the given addresses can**not** be used as valid source address in IPv6 packet? Explain why?

3d. (2 points) What is the "scope" of each of these IPv6 addresses? Explain what each scope does imply.

3e. (2 points) What is the **main** purpose of IPv6 Neighbor Discovery? Explain **clearly** the operation.

3f. (2 points) What are the messages deployed in IPv6 Neighbor Discovery? Explain how these messages will be encapsulated and addressed in layer-2 and layer-3 PDUs (i.e. packets and frames).

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 (6 points)

9.a **(2 p)** Define the task of wait-free self-stabilizing clock synchronization. Given a fixed integer k, once a processor p_i works correctly for at least k time units and continues working correctly, the following properties hold:

- Adjustment: p_i does not (1)____ its clock.
- Agreement: p_i's clock (2)____ with the clock of (3) ____ that has also (4) ____ for at least k time units.

9.b **(4 p)** We learned in class an algorithm for wait-free self-stabilizing clock synchronization for the fully connected graph, please find below its code. Each processor P has the following two variables: (1) P.clock $\in \{0 \ldots M-1\}$ and (2) $\forall$ Q : P.count[Q] $\in \{0,1,2\}$. We say that processor P is behind Q if P.count[Q]+1 (mod 3) = Q.count[P].
Suppose the processor P executes more than k=2 successive steps. Show that the set *NB*, which is *R* in the code to the right, is not empty following P's first step.

The program for P:

1) Read every count and clock
2) Find the processor set R that are not behind any other processor
3) If R $\neq \emptyset$ then P finds a processor K with the maximal clock value in R and assigns P.clock := K.clock + 1 (mod M)
4) For every processor Q, if Q is not behind P then P.count[Q] := P.count[Q] + 1 (mod 3)

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 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)~~

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 $lr_m := \text{read}(r_m)$
3. Leader := (id == *maximum* $\{lr_m.\text{id} \mid 1 \le m \le n\}$)
4. (* if *Leader* == True then *act_like_a_leader()* *)

Question 10 (2.5 points)

10.a **(1 p)** Define the task of vertex coloring.

10.b **(1.5 p)** Please find to the right one of the self-stabilizing algorithms for vertex coloring that we learned in class. How long does it takes for the algorithm to convergence. Please give an example for a particularly long convergence period.

```
01 Do forever
02      GColors := ∅
03      For m:=1 to δ do
04          lrm:=read(rm)
05          If ID(m)>i then
06              GColors := GColors U {lrm.color}
07      od
08      If colori ∈ GColors then
09        colori:=choose(\\ GColors)
10      Write ri.color := colori
11  od
```

Chalmers University of Technology
Computer Science and Engineering
Elad Michael Schiller

2013-01-17

Written exam in EDA387/DIT663 Computer Networks 2013-01-17. 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)

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.

A PC-user issues the <dig> command in order to get DNS-information. Examine the output carefully and then answer the questions given below (appears on next page) using DNS-terminology and concepts. Please answer each question separately.

C:\dig>dig ns chalmers.se @dns.uu.se

```
; <<>> DiG 9.3.2 <<>> ns chalmers.se @dns.uu.se
; (1 server found)
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 1628
;; flags: qr aa rd; QUERY: 1, ANSWER: 4, AUTHORITY: 0, ADDITIONAL:7

;; QUESTION SECTION:
;chalmers.se.                   IN    NS

;; ANSWER SECTION:
chalmers.se.          172800  IN    NS      ns1.chalmers.se.
chalmers.se.          172800  IN    NS      ns2.chalmers.se.
chalmers.se.          172800  IN    NS      dns.uu.se.
chalmers.se.          172800  IN    NS      ns3.chalmers.se.

;; ADDITIONAL SECTION:
dns.uu.se.            14400   IN    A       130.238.7.10
dns.uu.se.            14400   IN    AAAA    2001:6b0:b:242::10
ns1.chalmers.se.      172800  IN    A       129.16.2.40
ns1.chalmers.se.      172800  IN    AAAA    2001:6b0:2:10::1
ns2.chalmers.se.      172800  IN    A       129.16.253.252
ns2.chalmers.se.      172800  IN    AAAA    2001:6b0:2:20::1
ns3.chalmers.se.      172800  IN    A       192.36.120.11

;; Query time: 44 msec
;; SERVER:130.238.7.10#53(130.238.7.10)
;; WHEN: Thurs. Dec 10 20:19:56 2013
;; MSG SIZE  rcvd: 252
```

Note: Please answer the following questions with the aid of the issued command and the information given in each section of the answer.

1.a (1p) What DNS-information does the user specifically want to know? Relate your answer to the issued command.

1.b (1p) To which DNS-server is the query sent? What is the IP address of this server?

1.c (1p) Explain whether the user gets exactly the queried information or not. ***Note:*** NOT enough with answering "YES" or "NO".

1.d (2p) How can you verify whether the above answer is authoritative or non-authoritative? Explain clearly in two different ways.

1.e (2p) How many Resource Records are there in the answer section? Describe clearly the information and the content of **one** of these RRs.

1.f (1p) Within each raw in the output, there is a given number (e.g. 14400 or 172800), explain what this number is telling and what it is used for.

1.g (2p) What are the shown abbreviations (NS, A, AAAA) standing for? Explain the meaning of each.

Question 2 IPv6 Addresses (6 points)

An IPv6 node has an Ethernet-interface with MAC address 5C-26-0A-66-77-7C.

2.a (3p) Rewrite each of the following IPv6 addresses using optimal zero-compression.

(i) 2001:06B0:0000:0000:5E26:0AFF:FE66:777C

(ii) FF02:0000:0000:0000:0000:0001:FF66:777C

(iii) FE80:0000:0000:0000:5E26:0AFF:FE66:777C

2.b (3p) What is the type and the scope of each one of the given addresses in question (**2.a**).

Question 3 IPv6 operations (6 points)

3.a (3p) There is neither broadcasting nor ARP in the new version of the Internet Protocol IPv6. Describe the substituting operation, its purpose, the protocol and the messages used by an IPv6 node and its neighbors which are attached to an Ethernet LAN.

Hint: You are allowed to make use of the address (es) in the previous question if you would like to include example(s) in your answer.

3.b (3p) Describe clearly the stateless autoconfiguration of an IPv6 node that has an interface attached to an Ethernet LAN. In your answer mention in detail how the address will be configured, the protocol, the messages, and their relevant content, utilized in order to achieve this type of IPv6 configuration.

Question 4 (2 point)

6.a (1 p) An _____ E = (c(1), a(1), c(2), a(2), ...) , an _____ sequence, such that the _____ a(i) is _____ to _____ c(i-1) and results in _____ c(i), i.e., c(i-1) $\xrightarrow{a(i)}$ c(i) (i>1).

6.b (1 p) Please complete the definition of the mutual exclusion task, ME. (1) _____ one _____ can _____ it's _____ in any _____, and (2) every _____ can _____ it's _____ in _____ many _____ in every _____ in ME.

Question 5 (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 ∈ N(i) do send(i,clock_i)
03     clock_i := clock_i
04     forall p ∈ N(i) do
05         receive(clock_p)
06         if clock_p > clock_i then clock_i := clock_p
07     od
08     clock_i := clock_i mod (B · lg(n) + 1)
```

4.a (2 p) The algorithm considers bounded set of values for the variable *clock*. This bound depends on the number of nodes in the network, *n*. Why that property is considered to be unattractive? Is the same true for the network diameter, d? In the context of the Internet, where is the difference?

4.b (4 p) Prove the correctness of the algorithm above.

Question 6 (4 points)

Below please find the spanning-tree construction algorithm. A variant of that algorithm is proposed. In this version every processor repeatedly checks whether the value of the *dist* variable of its parent in the tree is smaller than the value of its own *dist* variable. Processor pi does not execute lines 8 to 16 of the code when the above condition holds. Is the proposed algorithm a self-stabilizing spanning tree construction algorithm? Prove your answer.

```
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 write 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
```

Question 7 (8 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 *no need to prove Lemmas 2.2 and 2.3*!

01 P_1: do forever

02 if $x_1 = x_n$ then

03 $x_1 := (x_1 + 1) \bmod (n+1)$

04 $P_i (i = 1)$: do forever

05 if $x_i \neq x_{i-1}$ 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)~~

(4 pt) 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)

(4 pt) 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) `select()`

Each of the following parts of a program contains a flaw. Identify and describe the flaw in a 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.)

The following program has two sockets, `sockA` and `sockB`, from which the system expects to receive 24 byte messages. Each message is to be stored in a variable of type `message_t`, which is sufficiently large to contain the message. The messages are processed using the `process_message()` method. It is assumed that this method is capable of authenticating the message and checking that the message has the correct format. You can also assume that the `handle_*_error()` methods do something sensible.

```
/* includes, declarations, etc. */

static bool receive_message_from_socket( int sock ) {
      message_t msg;
      ssize_t receivedBytes = 0;
      // make sure that we receive the whole message
      while( receivedBytes < sizeof(msg) ) {
            ssize_t ret = recv( sock,
                  ((char*)&msg) + receivedBytes,
                  sizeof(msg) - receivedBytes,
                  0
            );
            if( 0 == ret ) {
                  // peer disconnected; close socket and return false to
                  // indicated that the connection has closed.
                  close( sock );
                  return false;
            }
```

```
            if( -1 == ret ) handle_recv_error();
            receivedBytes += ret;
        }
        // OK, process message and return success
        process_message( msg );
        return true;
}

int main() { /* initialization code has been omitted */
        while( sockA != -1 || sockB != -1 ) {
            // initialize read set
            fd_set readfds;
            FD_ZERO( &readfds );
            if( sockA != -1 ) FD_SET( sockA, &readfds );
            if( sockB != -1 ) FD_SET( sockB, &readfds );
            // call select(). parameters that are NULL (=0) are ignored by
            // select() - i.e. this is not an error!
            int maxfd = sockA;
            if( sockB > maxfd ) maxfd = sockB;
            int ret = select( maxfd+1, &readfds, 0, 0, 0 );
            if( -1 == ret ) handle_select_error();
            // check sockA for messages
            if( FD_ISSET( sockA, &readfds ) ) {
                bool stillOpen = receive_message_from_socket( sockA );
                if( !stillOpen )
                    sockA = -1;
            }

            // check sockB for messages
            if( FD_ISSET( sockB, &readfds ) ) {
                bool stillOpen = receive_message_from_socket( sockB );
                if( !stillOpen )
                    sockB = -1;
            }
        }
        /* de-initialization code has been omitted */
        return 0;
}
```

Question 9 TCP/IP (12 points):

9.a (3 p) What is the main idea of the TCP-friendly congestion control? Explain carefully (you do not need to write and explain the exact formula of the data-rate); outline the goal and how it is achieved.

9.b (3 p) Explain how can overlays be used in peer-to-peer file-sharing applications to address (i) searching and (ii) fetching of content. Accompany your explanations with examples.

9.c (2 p) Explain how can a connected dominating set be used in ad-hoc wireless networks in order to route messages between nodes.

9.d (4 p) Describe the marking algorithm for computing a connected dominating set and show its correctness.
