CHALMERS
EXAMINATION / TENTAMEN

Course code/kurskod	Course name/kursnamn			
EDA 387	Computer Networks			
Anonymous code Anonym kod		Examination date Tentamensdatum	Number of pages Antal blad	Grade Betyg
EDA 387 – 73		2014-10-31	12	5

Solved task Behandlade uppgifter No/nr		Points per task Poäng på uppgiften	Observe: Areas with bold contour are to completed by the teacher. Anmärkning: Rutor inom bred kontur ifylles av lärare.
1	X	7	
2	X	5	
3	X	7	
4	X	6	
5	X	9,5	
6	X	6	
7	X	~~6~~ 6	
8	X	7,5	
9	X	6	
10	X	8	
11			
12			
13			
14			
15			
16			
17			
18			
Total examination points Summa poäng på tentamen		68	42

5

CHALMERS | Anonymous code / Anonym kod: EDA 387-73 | Points for question (to be filled in by teacher) / Poäng på uppgiften (ifylles av lärare): 7 | Consecutive page no. / Löpande sid nr: 1 | Question no. / Uppgift nr: 1

Question 1 RR (name, type, value)

1a) (i) type = A: IPv4 address
gives the IPv4 address mapped to a specific host name

(ii) type = MX: mail exchanger
gives the IP address of a mail server with a domain specific name, along with a priority value

(iii) type = NS: name server
gives the authoritative name server for a specific domain name

(iv) type = PTR: pointer
gives the domain name for a specific IP address host

(v) type = AAAA: IPv6 address
gives the IPv6 address mapped to a specific name

(vi) type = CNAME: canonical name
gives the canonical name for a specific name

3

1b) When trying to access the web site www.tue.nl for the first time, the name must be resolved to an IP address. Therefor, the local host sends a DNS request to the local name server, containing the name of the webserver. Unfortunately, the local name server has no cache entry for the name in its database. However, it is a nice name server and decides to resolve the request recursively. Thus, it sends a DNS request for the name to a root server. The root server doesn't know the address-name mapping, either but it knows which name server is authoritative for the requested name. Thus, the root server replies with a message that contains the name and IP address of the authoritative name server. Now, the local name server sends a DNS request for the name to the authoritative server. The authoritative name server answers with a DNS reply that contains the requested name-address mapping. The local name server saves this in its cache and sends a DNS reply with the name-address mapping to the host. Now, the host knows the IP address of the web server and can connect to it to get the website.

not really

It is possible that after the root name server, a top-level domain name server must be queried first to get the address of the authoritative name server. I left that out to keep the text shorter. ?

DNS is an application layer protocol that uses UDP as transport service. The two message types are DNS request and DNS reply, which share the same message format.

root → TLD → Auth DNS-servers

4+

5

CHALMERS | Anonymous code / Anonym kod: EDA387-73 | Points for question (to be filled in by teacher) / Poäng på uppgiften (ifyllas av lärare): 5 | Consecutive page no. / Löpande sid nr: 2 | Question no. / Uppgift nr: 2

Question 2

2a) (i) 2001:0660:0002:0010:0000:0000:0000:0001

(ii) FF02:0000:0000:0000:0000:0001:ff6c:14dd 1

(iii) FE80:0000:0000:0000:020c:f1ff:fe6c:14dd

2b) (i) (Global) unicast address, i.e., address of a specific interface with global scope type?

(ii) Multicast address, i.e., address of a group of interfaces 1

(iii) (Local) unicast address, i.e., address of a specific interface (with local scope)

2c) The multicast address (ii) cannot be used as valid source address, because the source must always relate to a distinct interface, not to a group of interfaces. 1

2d) (i) Global scope, i.e., the address is globally valid, in the whole internet means?

(ii) + (iii) Link-Local scope, i.e., the address is only valid in the subnetwork it belongs to

2+

5

CHALMERS | Anonymous code / Anonym kod: EDA387-73 | Points for question (to be filled in by teacher) / Poäng på uppgiften (ifylles av lärare): 7 | Consecutive page no. / Löpande sid nr: 3 | Question no. / Uppgift nr: 3

Question 3

Router Discovery!

3a) The IPv6 Neighbor Discovery Protocol (is used for auto-configuration on system startup.) It makes it possible for a node to find its neighbors in the local network and determine their physical addresses. Additionally, the node can test whether a specific neighbor is still reachable or not. Another thing is, that with NDP, (a node can discover the routers in the local network, along with their physical addresses and information like the MTU size or the network prefix, which can be used to configure the interface address.) 2(

/ +

operation

3b) For Neighbor Discovery, two messages are used: Neighbor Solicitation and Neighbor Advertisement. Both messages are ICMPv6 messages that are encapsulated in IPv6 packets, which are encapsulated in Ethernet ~~packets~~ frames.

A node A that wants to know the physical address of a node B in the same network sends a Neighbor Solicitation message. The Neighbor Solicitation message will be addressed like the following: The source address of the IPv6 packet is node A's IPv6 unicast address, the destination address of the IPv6 packet is node B's solicitated address. The source address of the Ethernet ~~packet~~ frame is node A's MAC address, the destination address of the Ethernet ~~packet~~ frame is the multicast MAC address 33-33-FF-xx-xx-xx, where xx-xx-xx corresponds to the lowest 24 bits of node B's solicitated address.

Answer only one question on this page. Do not write on the back of this paper
Behandla endast en uppgift på detta blad. Skriv ej på baksidan

Question 3b) - part 2

Node B then answers with a Neighbor Advertisement message that contains its physical address. This message is addressed like the following: The source address of the IPv6 packet is node B's IPv6 unicast address, the destination address of the IPv6 packet is node A's IPv6 unicast address. The source address of the ethernet frame is node B's MAC address and the destination address of the ethernet frame is node A's MAC address. 2

3c)

The routers in a network send Router Advertisements approximately all three minutes. This way, they inform the other nodes in the network about their presence and give important information.
A node can also request a Router Advertisement by sending a Router Solicitation message. Information contained in a Router Advertisement is for example

1) The network prefix that is to be used. With this information, a node can configure a globally valid unicast address by using its modified MAC address as interface ID.
2) The Maximum Transmission Unit size that is to be used. With this information, a node knows how to fragment the IPv6 packets it sends appropriately.
3) The physical address of the router. With this information, a node can forward IPv6 packets that it wants to send to a node outside the local network to the router.

4

CHALMERS

Anonymous code / Anonym kod: EDA 387-73

Points for question (to be filled in by teacher) / Poäng på uppgiften (ifylles av lärare): 6

Consecutive page no. / Löpande sid nr: 5

Question no. / Uppgift nr: 4

Question 4

The flaw here is that after calling accept(), recv() is called without testing whether the new client is ready to send data. This can make the server block forever if a client opens a connection to the server and then ~~[illegible]~~ never sends anything.

5

CHALMERS

Anonymous code / Anonym kod: EDA 387 - 73

Points for question (to be filled in by teacher) / Poäng på uppgiften (ifylles av lärare): 9,5

Consecutive page no. / Löpande sid nr: 6

Question no. / Uppgift nr: 5

Question 5

Transient faults can corrupt the values in the read/write registers and in the local variables. Thus, a node has wrong assumptions about its actual distance to the root and choses wrong nodes as parents. This can mean that the <parent, dist> values do not encode a correct BFS spanning tree.

Proof:

(1) induction over
(2) at least
(3) the first assumption (concerning the smallest floating distance)
(4) the second assumption
(5) distance 0
(6) Δ
(7) Δ
(8) c_1
(9) distance 1
(10) 0
(11) 1
(11) the second assumption (concerning correct distances)
(12) >
(13) m+1
(14) the first assumption
(15) $\geq$ k
(16) k+1

Question 6

6a) We say that configuration c is safe if every fair execution that starts in configuration c belongs to the set of legal executions.

6b) The algorithm depicted here consists of an initialization phase and a phase in which the nodes shall reach consensus by exchanging information with their neighbors.
However, this can go wrong if variables like $pulse_i$ or O_i are set to wrong values because of transient faults, or if the program counter is corrupted, causing a node to start execution in the middle of the program.
Imagine a situation where all inputs are 0, but one output value is set to 1 due to a fault, and then all nodes continue program execution from line 4. In this case, they will set their output variables O_i all to 1, which is wrong and is only caused by the corrupted value in one output variable.

Question 7

7a) A safe configuration for the algorithm is a configuration in which:

1) the register <leader$_i$, dis$_i$> written by processor i contains the minimum value of a processor ID in the network in the leader$_i$ part and the distance of processor i to the processor with minimum ID in the dis$_i$ part, $\forall 1 \le i \le n$

2) the local ~~register~~ variable <candidate, distance> ~~named~~ of processor i is properly initialized to <ID(i), 0>, $\forall 1 \le i \le n$

3) the local variables <leader$_i$[j], dis$_i$[j]> ~~correspond~~ hold the same values as the registers <leader$_j$, dis$_j$>, i.e., leader$_i$[j] = minimum ID in the system, dis$_i$[j] = distance of processor j to the processor with minimum processor ID, $\forall 1 \le i \le n$, $P(j) \in N(i)$

4) implied by 1), after each cycle it holds that candidate = min ID in system & distance = distance of P_i to the processor with the min. ID

7(b) Assume that "step" means a whole iteration of the for loop here ~~otherwise P_i just reads the correct register values of its first neighbor and we are done with the proof.~~

In one iteration, P_i initializes its <candidate, distance> variable first to <ID(i), 0> (line 02). Thus, requirement 2) is fulfilled. Then, for each neighbor P_j, P_i reads the register of P_j and saves the values in <leader$_i$[j], dis$_i$[j]> (line 05). Since those values are correct - because c is a safe configuration - it holds afterwards that leader$_i$[j] = minimum ID in the system, and dis$_i$[j] = distance of P_j to the processor with minimum ID. Thus, requirement 3) is fulfilled. For each read pair <leader$_i$[j], dis$_i$[j]>, P_i checks if dis$_i$[j] < N. This is the case since dis$_i$[j] is the correct distance of processor P_j to the processor with minimum ID, i.e., at most N-1. ~~Afterwards, P_i checks if (leader$_i$[j] < candidate) which is the case iff P_i is not the processor with minimum ID or if (leader$_i$[j] = candidate and dis$_i$[j] < distance) which can never happen because be true because leader$_i$[j] = candidate means that P_i is the processor with minimum ID, but then dis$_i$[j] cannot be smaller than distance$_i$ because distance is 0.~~

Thus, after the complete execution of the for loop (lines 03 - 09) it holds that - if ID(i) is not minimal in the system -

candidate = min {leader$_i$[j] | $P_j \in N(i)$} and

distance = 1 + min {dist$_i$[j] | $P_j \in N(i)$ and leader$_i$[j] = min {leader$_i$[k] | $P_k \in N(i)$}}

Since requirement 3) is true, it holds that

candidate = min ID in the system and

distance = true distance of P_i to the processor with min ID

This is especially true if ID(i) is minimal in the system, because in this case, candidate and distance are left unmodified. Afterwards, <candidate, distance> is written to <leader$_i$, dis$_i$> and requirements 1) and 4) are fulfilled.

Question 8

8a) d is the diameter of the communication graph, i.e., the length of the longest shortest path between a pair of nodes in the graph

2

n is the number of nodes in the communication graph, i.e., the number of processors

8b) The converge-to-the-min algorithm scales better because it uses only 2d+1 different clock values compared to (n+1)d + 1 clock values. That means that it only needs a register with $O(\log_2 d)$ bits instead of $O(\log_2 n \cdot d)$ bits to store the clock value.

1

8c) As soon as a safe configuration is reached, there is no difference observable between the executions of the different algorithms. ~~Thus, the application layer shouldn't care too much about which of them is used.~~ (*)

1/2

8d) (1) resets its clock value to zero
(2) d
(3) induction over the distance to the node with the minimal value
(4) resets its clock value to zero
(5) d
(6) d
(7) d
(8) case holds ☒

4

(*) belonging to 8c)

However, if a transient error occurs, the clock value might jump to a greater (converge-to-max) or smaller (converge-to-min) clock value. In the first case, computation steps could be left out, in the second step, computation steps could be reexecuted. It depends on the application with which scenario it can deal better.

Question 9

9a) (1) adjust
(2) agrees
(3) each processor
(4) worked correctly

9b) After the first step, P is not behind any other processor. This is because in line 4), P increments its count for each processor Q that is not behind P, i.e.,

Q.count[P] = P.count[Q] (mod 3) or (P.count[Q] + 1) = (Q.count[P]) mod 3

Afterwards, it holds for all processors that

Q.count[P] + 1 = P.count[Q] (mod 3) or

Q.count[P] = P.count[Q] mod 3, thus

P is not behind any processor.

Since P consequently belongs to NB afterwards, this set is not empty.

Question 10

10a) Given a graph, assign each vertex a color such that there are no neighboring vertices with the same color. You can use $\Delta+1$ colors where Δ is an upper bound on the number of neighbors a vertex has. It is not required to minimize the number of different colors used.

10b) It can be shown by induction, that after the k-th cycle, the colors of the k vertices with the highest IDs are fixed and legal in the sense that in the set of those k vertices there are no neighboring vertices with the same color.
Thus, convergence is reached after at most n cycles, where n is the number of processors.

In the following situation, convergence actually takes n cycles:

Given a linear array of n processors and colors 0, 1, ..., processor i is assigned color $i \bmod 2$ for $1 \le i \le n-1$ and processor n is assigned color $(n \bmod 2)+1$, thus processor n and processor n-1 have the same color. If now in the first cycle, processor n-1 adjusts its color to $((n-1) \bmod 2)+1 \in \{0,1\}$, then processor n-2 and n-1 have the same color. Thus, processor n-2 has to adjust and chooses $((n-2) \bmod 2)+1 \in \{0,1\}$, thus processor n-3 and n-2 have the same color, and so on.
In cycle i, processor n-i and processor n-i+1 have the same color, and processor n-i changes in a way that it has the same color as processor n-i-1 in the next cycle.
Thus, each processor has to adjust its color once.

Question 10

10c)
In class we saw a superstabilizing algorithm with shorter convergence time:

```
Do forever
   A Colors := ∅
   G Colors := ∅
   For m := 1 to δ do
        r_m := read(r_m)
        AColors := AColors ∪ {r_m.color}
        If ID(m) > i then GColors := GColors ∪ {r_m.color}
   od
   If color_i = ⊥ OR color_i ∈ GColors then
        color_i := choose(\AColors)
   write r_i.color := color_i
od
```

Considering the example given in exercise 10 b), the convergence time is only one cycle when using the improved algorithm: In the first cycle, processor n-1 will chose a color different from 0 and 1. Afterwards, there are no neighboring nodes with the same color anymore.

Note that it is always possible to find a color that is different from all colors used in the neighboring nodes since there are $\Delta + 1$ colors available.