CHALMERS
EXAMINATION / TENTAMEN

Course code/kurskod	Course name/kursnamn			
EDA387	EDA387 - Computer Networks			
Anonymous code Anonym kod		Examination date Tentamensdatum	Number of pages Antal blad	Grade Betyg
EDA387-0003-NYE		2023-01-04	13	3

* I confirm that I've no mobile or other similar electronic equipment available during the examination.
Jag intygar att jag inte har mobiltelefon eller annan liknande elektronisk utrustning tillgänglig under eximinationen.

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	13	
2	6	
3	5½	
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		
16		
17		
Bonus poäng		
Total examination points Summa poäng på tentamen	(24½)	25

3

1a) packet routing:
determing a route for a packet to take. Figuring out which nodes can take you where in a network

½

packet forwarding:
Transmitting a packet forward using nothing but local information. Associating ips with physical connections etc.

½

3

1b)

for each node x in a d-dimension hypercube its neighbours ids are the set of ids with by 1 digit

neighbour[i].id = X.id xor (1 << i) where $i \in [0, d)$

eg:) X.id = 0000 d = 4

neig[0].id = 0001
neig[1].id = 0010
neig[2].id = 0100
neig[3].id = 1000

Since the assignment talks about packets i will assume a high level packet aknowledgements protocol (such as TCP) to handle transient faults

```
on recieve(packet)
    packet is for me = true
    dst = packet.dst
    for i in [0, d)
        if dst[i] ≠ this.id[i]
            transmit packet to neighbours[i]
            packet is for me = false
            break!

if packet is for me ...
```

3

3

1c) the algorithm forwards packets to the node where the lowest bit matches then the second and so on...

after 1 step the packet will be on a node where the first bit matches and so on.

P(i) = bit i matches after i+1 steps

assuming P(i) ⇒ if dst[i+1] ≠ self[i+1]
send to node where P(i+1) is true
P(i+1)

P(i) ⇒ P(i+1)

P(0): if dst[0] ≠ self[0]
send to node where P(0)
true

2

⇒ P is true

Transient faults hadled by TCP

1d) Software-defined Networking. 1

Instead of every router/switch working on their own and broadcasting there is a server (controller) that knows the entire network structure and decides how routing tables etc. should be set up.

Packet-IN: a switch sends a packet it does not know how to transmit to the controller 1

Packet-out: the controller sends the packet to a switch that should handle the packet (or endpoint) 1

Flow-mod: the controller sending new forwarding table information to a switch 1

3

1e)

in the SDN (and normal networks) each switch does not know the whole network topology. 1

The controller can hard code the forwarding tables if there is enough room. (flow-mod) otherwise it can packet-IN and packet-out to the correct node dynamically updating tables once it recieves packets 1

1

3

2a) vertex coloring gives each vertex a color such that it doesn't have the same color as any of its neighbours

Why there is no (deterministic) algorithm for anonymous networks:

example: a=1 b=2

For n_1 to stabilize it depends on n_2 to be stable. Vice versa for n_2.
There is no deterministic way to decide which node goes first (or if they go at the same time) the system will

A (if simultaneous steps)
never stabilize, each node would deterministically switch to identical (and invalid) states

B (if random order steps)
a random node will get the defadt color and the other follows suit
⇒ non deterministic

3

2b)

(lowest = leader)

Each node uses the studied (candidate, distance) leader election. It sets its color equal to distance mod 2.

Non corner nodes set their candidate id to N^k. Due to the corners beeing unique ther will always be one and only one lowest id which will become the leader

2

2c)

The leader election is already proven to be self stabilising and will correct transient faults.

A grid with color alterating between odd and even distances is trivially a correct vertex coloring.

1

3

2d) the stabilisation time of this leader election is equal to the furthest distance between nodes. for a grid this is a+b

within d cycles nodes d units from the leader will have stabilized.

1/2

3

2e) Each nodes need enough bits to store two copies of <candidate, distance>

memory = $O(2 * (\#bits(candidate) + \#bits(distance)))$

$= O(\log(n) + \log(a+b))$

$n = a \times b$

$\log(a+b) < \log(a \times b)$

$\Rightarrow \log(a+b) < \log(n)$

$= O(\log n)$

2½

3

3a) maximum matching:

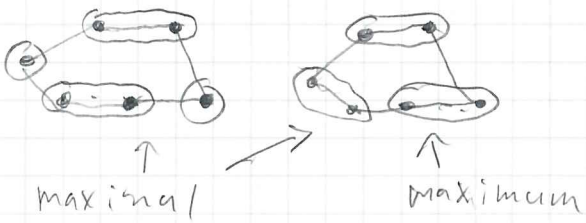
a graph split in the fewest number of partitions such as every partion contains at most X nodes.

maximal matching:

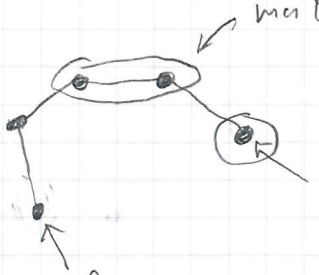
a graph split in partions such that no partions could add a neighbouring partion without exceeding X nodes in the partion.

maximum is a subset of maximal matches

ex) X = 2



2 1/2



matched node:

a node that is part of a partion

single node:

free node:

a node that don't have any neighbours that it could join

a node that have neigbours it could join with to make a portion

3

3b. Non optimal

First BST the tree, the use the studied topology collection so the root knows the entire tree structure.

Solve the problem locally continously using the collected structure

3c) the topology collection and bst is already proven. Nothing to

1

1/2

3

3d same as 3b but with the following changes

each node self stabilizingly calculates its smallest distance from a leaf and which direction it is

if leaf
  Ldistance = 0
else
  Ldistance = max (read(neighbours)..distance) +1

to find when to stop (is centered) check all neighbours neighbours, to prevent going back and forth

3e fair composition, split task for half tree each in case of two centers, once that is stable check if center partions can be merged