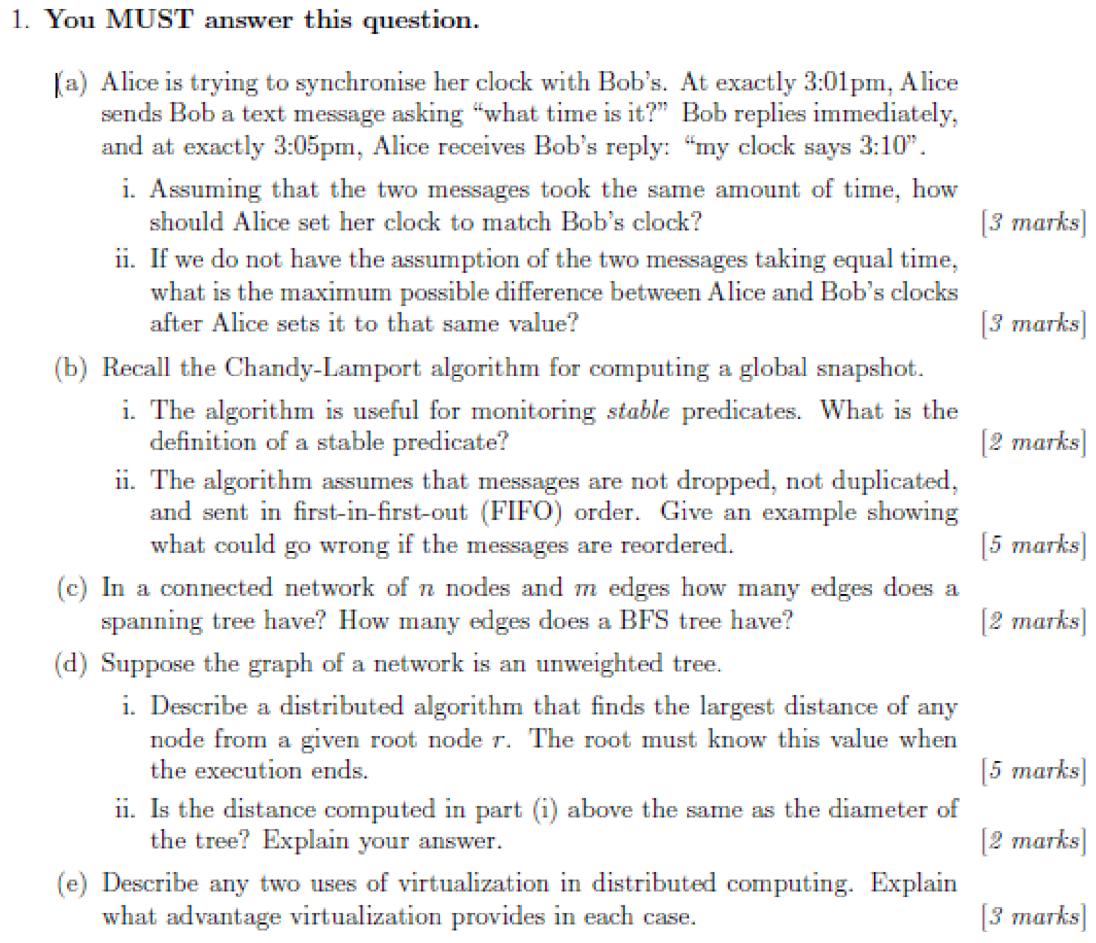
q1



1. **You MUST answer this question.**

(a) Alice is trying to synchronise her clock with Bob's. At exactly 3:01pm, Alice sends Bob a text message asking "what time is it?" Bob replies immediately, and at exactly 3:05pm, Alice receives Bob's reply: "my clock says 3:10".

i. Assuming that the two messages took the same amount of time, how should Alice set her clock to match Bob's clock? [*3 marks*]

ii. If we do not have the assumption of the two messages taking equal time, what is the maximum possible difference between Alice and Bob's clocks after Alice sets it to that same value? [*3 marks*]

(b) Recall the Chandy-Lamport algorithm for computing a global snapshot.

i. The algorithm is useful for monitoring *stable* predicates. What is the definition of a stable predicate? [*2 marks*]

ii. The algorithm assumes that messages are not dropped, not duplicated, and sent in first-in-first-out (FIFO) order. Give an example showing what could go wrong if the messages are reordered. [*5 marks*]

(c) In a connected network of $n$ nodes and $m$ edges how many edges does a spanning tree have? How many edges does a BFS tree have? [*2 marks*]

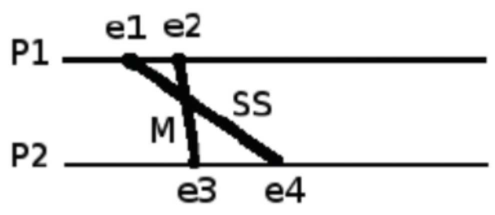
(d) Suppose the graph of a network is an unweighted tree.

i. Describe a distributed algorithm that finds the largest distance of any node from a given root node $r$. The root must know this value when the execution ends. [*5 marks*]

ii. Is the distance computed in part (i) above the same as the diameter of the tree? Explain your answer. [*2 marks*]

(e) Describe any two uses of virtualization in distributed computing. Explain what advantage virtualization provides in each case. [*3 marks*]

1.(a)
i. 3:12 pm
ii. 2 minutaes +1
Is it worth noting something about drift here? This calculation assumes that Bob and Alice's clock are both moving at about the same rate.
(b)
i.
Predicate that once it becomes true, stays true.
ii.
If p1 sends marker `M` (e1) and then a message `m` (e2) but message `m` gets delivered (e3) before `M` at p2 then p2 will record `m` in its state snapshot once marker `M` arrives (e4) but p1 will not record sending `m`.



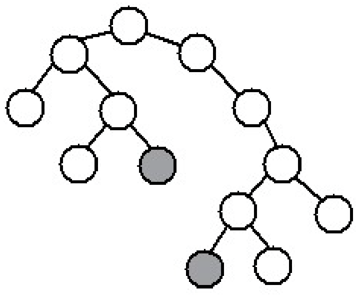
(c)
Spanning tree has n - 1 edges.
BFS also n-1? +1
(d)
i.
Not sure if I understand the question correctly, assuming they mean find the most distant node from the root.
(They are not the same) The diameter is the longest distance between any two leaves in a tree while the distance asked for in the algorithm is the distance to the root.

Use a convergecast from the root:

- Inner nodes recursively tell children to report value and wait for replies.
- Leaves send (1, nodeID) to parent.
- Once inner node gets replies from all children, it sends (max(children values) + 1, corresponding nodeID) to its parent.

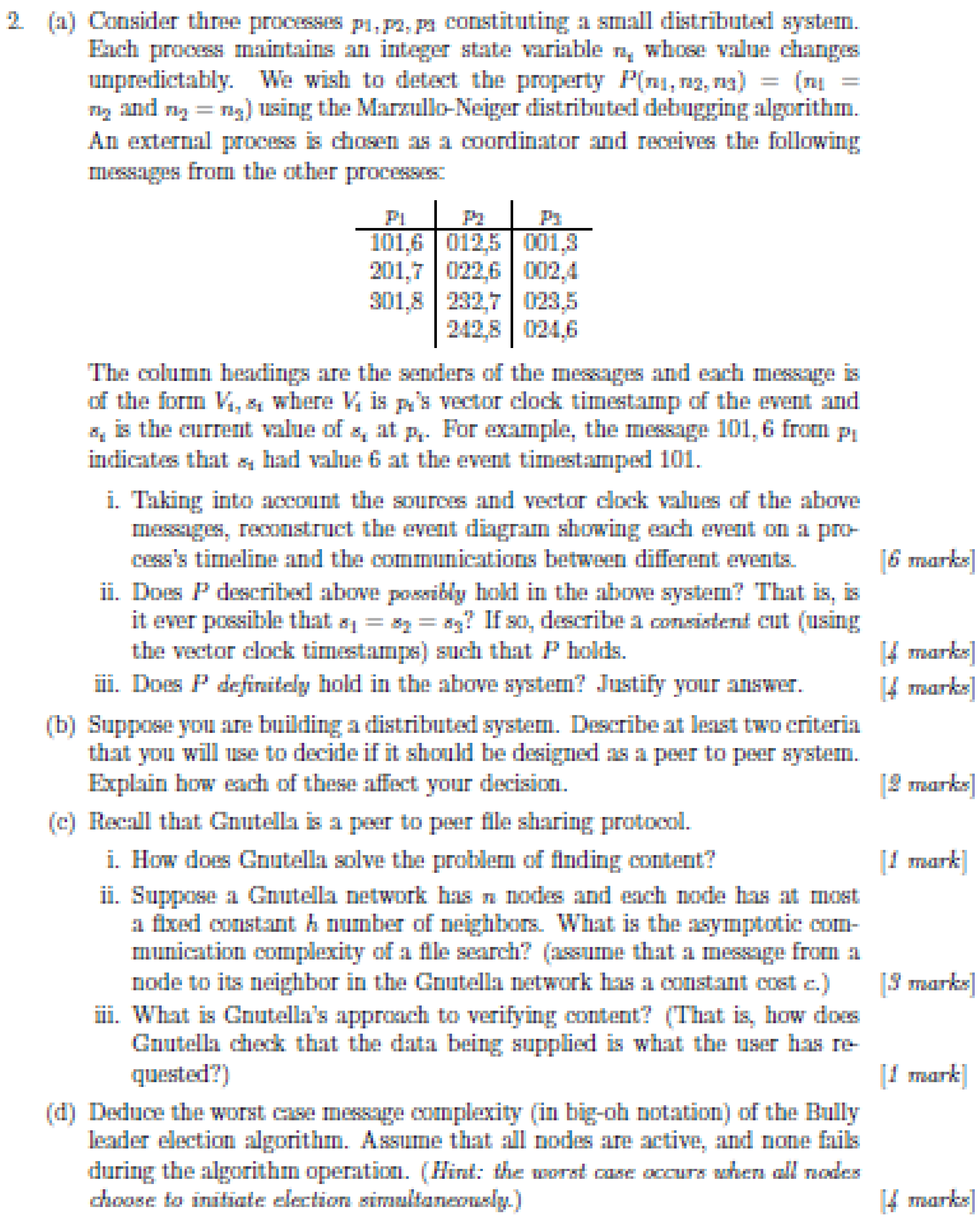
ii.
Assuming same interpretation as previous question.

Not necessarily, consider image below (assuming root is topmost node). The diameter is distance between grey nodes (8) whereas our algorithm would return distance to grey node on the right (5).



Basically, it depends on whether node r is one of the two furthest apart nodes or not. If r were one of the grey nodes, it would be, but there is no way of knowing that.
(e)
**Testing - sandboxing**
Allows you to e.g. test out new OS without affecting hardware.

**Cloud computing - dynamic resource allocation**
Allows multiple cloud service users utilising same physical server. When one user has less traffic on their website other user gets more CPU and the server is well-utilised.



2. (a) Consider three processes $p_1, p_2, p_3$ constituting a small distributed system. Each process maintains an integer state variable $n_i$ whose value changes unpredictably. We wish to detect the property $P(n_1, n_2, n_3) = (n_1 = n_2 \text{ and } n_2 = n_3)$ using the Marzullo-Neiger distributed debugging algorithm. An external process is chosen as a coordinator and receives the following messages from the other processes:

| $p_1$ | $p_2$ | $p_3$ |
|---|---|---|
| 101,6 | 012,5 | 001,3 |
| 201,7 | 022,6 | 002,4 |
| 301,8 | 232,7 | 023,5 |
|  | 242,8 | 024,6 |

The column headings are the senders of the messages and each message is of the form $V_i, s_i$ where $V_i$ is $p_i$'s vector clock timestamp of the event and $s_i$ is the current value of $s_i$ at $p_i$. For example, the message 101, 6 from $p_1$ indicates that $s_1$ had value 6 at the event timestamped 101.

i. Taking into account the sources and vector clock values of the above messages, reconstruct the event diagram showing each event on a process's timeline and the communications between different events. [*6 marks*]

ii. Does $P$ described above *possibly* hold in the above system? That is, is it ever possible that $s_1 = s_2 = s_3$? If so, describe a *consistent* cut (using the vector clock timestamps) such that $P$ holds. [*4 marks*]

iii. Does $P$ *definitely* hold in the above system? Justify your answer. [*4 marks*]

(b) Suppose you are building a distributed system. Describe at least two criteria that you will use to decide if it should be designed as a peer to peer system. Explain how each of these affect your decision. [*2 marks*]

(c) Recall that Gnutella is a peer to peer file sharing protocol.

i. How does Gnutella solve the problem of finding content? [*1 mark*]

ii. Suppose a Gnutella network has $n$ nodes and each node has at most a fixed constant $k$ number of neighbors. What is the asymptotic communication complexity of a file search? (assume that a message from a node to its neighbor in the Gnutella network has a constant cost $c$.) [*3 marks*]

iii. What is Gnutella's approach to verifying content? (That is, how does Gnutella check that the data being supplied is what the user has requested?) [*1 mark*]

(d) Deduce the worst case message complexity (in big-oh notation) of the Bully leader election algorithm. Assume that all nodes are active, and none fails during the algorithm operation. (*Hint: the worst case occurs when all nodes choose to initiate election simultaneously.*) [*4 marks*]

2.

(a)

Not examinable anymore?

(b)

- load balancing
- fault tolerance
- cost efficiency
- hard to take down

B Criteria should be
Budget: making sure p2p is a low budget solution
Trust: trusting other users
Static content: the content should not be dynamic because there will be a cost incurred with updating it..
(c)

i)
a node searching any specific data floods an overlay network with a search request. Nodes having the file respond to the requesting node, which can then download the data from the reponding node
ii)
not sure about this one but worstcase cost of flood O(n^2)
number of neighbors is irrelevant because the search is trying to reach ALL nodes, not just the neighbors
iii)
gnutella does not have any verification of data/content

(d)
When all nodes choose to initiate the election simultaneously, the lowest-ID node will send a message to nodes with higher ID (n-1 messages). The second-lowest-ID node will send a message to nodes with higher ID (n-2 messages) and so on:

$$n-1 \quad + \quad n-2 \quad + \quad \cdots \quad = \quad O(n^2)$$

All nodes are alive, and so they reply for another n(n+1)/2 = O(n^2) messages.
Only node n (did not send any messages) does not hear back, and so it continues to declare itself leader (another O(n) messages)

Complexity is 2*O(n^2)+O(n) = O(n^2).

Of interest: worst case *with failures* is O(n^3) - let each elected node crash upon election (before declaring itself coordinator). Then there will be n elections.

3. (a) Sometimes in a distributed system, it is useful to elect a process as a "leader" for the distributed computation. Give two examples of computations or scenarios where having a leader is useful. *[3 marks]*

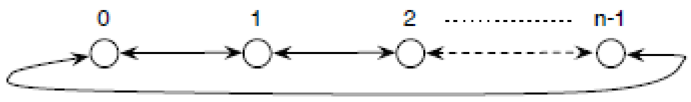


Figure 1: Leader election ring. The numbers next to the processes show their ids.

(b) Consider the ring of $n$ processes shown in Figure 1. Every process has two pointers (or links) called *left* and *right* to its neighbours in the ring. The links at any process naturally point to the processes on its left and right, except at process 0 and $n-1$ where they loop to the other end of the ring as shown in the figure.

Recall that the basic (Chang and Robert's) ring based leader election algorithm sends messages in only one direction and elects the process with the highest id as the leader.

i. Suppose all the processes initiate leader election simultaneously. If every process always sends its message to the process on its *left* link, what will be the asymptotic message complexity of the algorithm (in big-oh notation) on the given ring? (Justify your answer) *[4 marks]*

ii. What will be the message complexity if the processes always send messages to the process on their *right* links? *[3 marks]*

(c) In Bittorrent, a client prefers to download from fast peers, since fast peers usually provide faster downloads. Suppose we modify the protocol so that only peers that are sufficiently fast host the file. What are the advantages and disadvantages of the modified protocol? *[4 marks]*

*QUESTION CONTINUES ON NEXT PAGE*

(d) Recall the Maekawa voting algorithm for ensuring mutual exclusion. Consider four processes $p_1, p_2, p_3, p_4$ with the following voting sets:

$$\begin{aligned} V_1 &= \{p_4, p_1, p_2\} \\ V_2 &= \{p_1, p_2, p_3\} \\ V_3 &= \{p_2, p_3, p_4\} \\ V_4 &= \{p_3, p_4, p_1\} \end{aligned}$$

That is, the four processes are arranged in a square, and each process's voting set consists of itself and its two neighbours.

i. Give a possible run of this system resulting in deadlock, i.e., such that each process is blocked waiting for access to the critical section. *[5 marks]*

ii. Give a possible run of the system that violates fairness, i.e., such that some process $p_i$ makes a request before $p_j$ but $p_j$ is granted access before $p_i$. *[6 marks]*

3.
(a)
**Sequencer process in totally ordered multicast**
We need to order messages in global order. Sequencer process distributes sequence numbers to other processes.

**Central server mutex**
A central server ensures only one process is accessing a resource at a time.

Compute the sum/max using aggregation tree

(b)
i.
In Chang and Roberts algorithm every node has one of two states:

- Non-participant
    - Node p sends max(p.id, p.right.id) to the left
    - Sets itself as participant
- Participant
    - Receives p.right.id, if p.right.id > p.id then it sends it to the left

In the end only the highest ID will remain in circulation and once it gets to the node with that ID the algorithm terminates and we have our leader. Thus:

- Node n-2 will sends its ID left and it will get sent all the way to n-1
- Node n-3 will sends its ID left and it will get sent all the way to n-1
- ....

Hence, we get: $n-1 \quad + \quad n-2 \quad + \quad \cdots \quad = \quad O(n^2)$

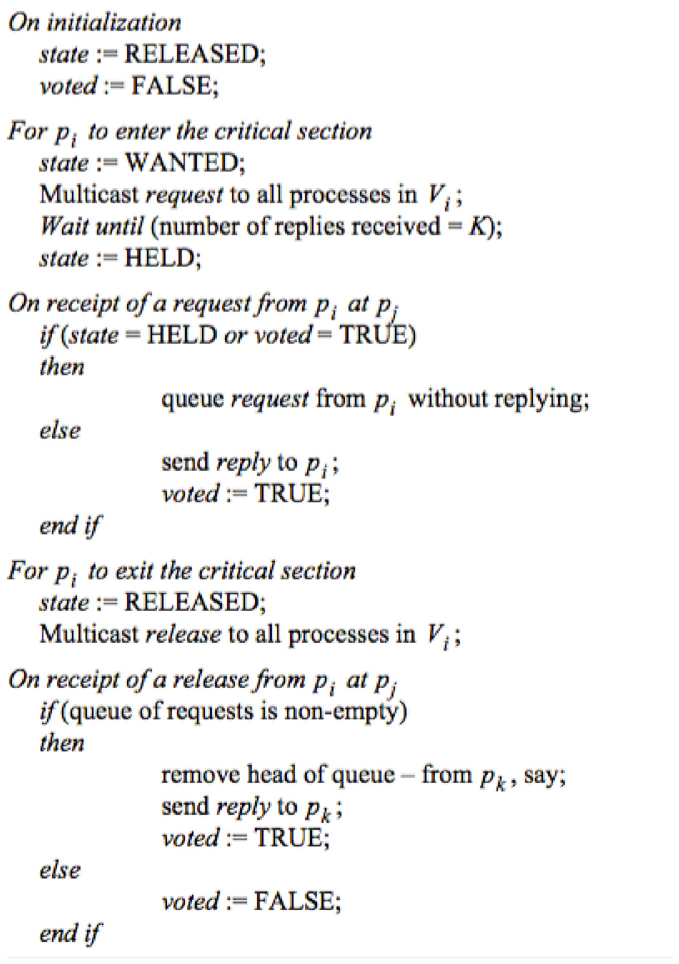
ii.
Every node's election will require only one message except for n-1 which will require n messages. Thus the complexity is $O(n)$ .
(n-1)*1 + n = 2n - 1 = O(n)

(c)

- Less waste of slow peer's resources - what? surely its getting totally wasted now
- more demand on fast peer's resources
- Fewer nodes have a file in case of node failure
- How are we determining speed / speed threshold?

(d)
i.

**Figure 15.6** Maekawa's algorithm



```
On initialization
    state := RELEASED;
    voted := FALSE;

For $p_i$ to enter the critical section
    state := WANTED;
    Multicast request to all processes in $V_i$;
    Wait until (number of replies received = $K$);
    state := HELD;

On receipt of a request from $p_i$ at $p_j$
    if (state = HELD or voted = TRUE)
    then
            queue request from $p_i$ without replying;
    else
            send reply to $p_i$;
            voted := TRUE;
    end if

For $p_i$ to exit the critical section
    state := RELEASED;
    Multicast release to all processes in $V_i$;

On receipt of a release from $p_i$ at $p_j$
    if (queue of requests is non-empty)
    then
            remove head of queue – from $p_k$, say;
            send reply to $p_k$;
            voted := TRUE;
    else
            voted := FALSE;
    end if
```

Every process votes for itself, hence voted = TRUE for all processes. Hence all other mutex responses are queued and all processes wait until they get replies from all neighbours (never).

ii. Assume P3 makes a request at time T1 to P2 and P4 but the requests are on the way. Then P1 makes a request at Time T2 to P2 and P4 and the requests are granted by P2 and P4. When P3's requests make it to P2 and P4, it would be queued up and waits for P1 to finish making use of the CS even though it made its request at an earlier time T1. This violates the Fairness condition.

My answer:
Say P2 is in the CS, P1 wants to enter and sends a message to P2 and P4, but the message doesn't arrive yet. He also now sends a message to P3. P3 gets the message and wants to enter the CS and sends a message to P2 and P4, which arrives before P1-s message. Now P1's messages arrive and are queued after P3's. Now when P2 exits, P3 will enter before P1, breaking happens-before fairness.