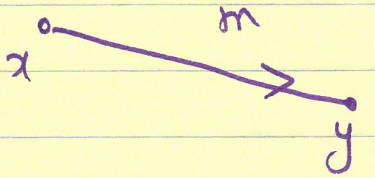
1

10-8-2019

Ch 15. Asynchronous Algorithms

Asynchronous model:

Processes }
links. } asynchronous. ;



links { elapsed time for m to ~~go~~ traverse the link (x,y)
[illegible] (time for m to be delivered to y)
is arbitrary but finite

processes { time for y to process the received message m: arbitrary but finite

Links: FIFO

Failures < Failure Free ✓
~~Failure~~ Fail Stop
Byzantine failure } processes links.

Message Complexity ~~or~~ or bit Complexity

* { Time Complexity: if all messages take 1 time unit to be transmitted and message processing is 0. (delay) elapsed time?

⊛ The asynchronous distributed algorithm must execute correctly regardless of the assumptions about message transmission delays and message processing delays.

Leader election in Rings.

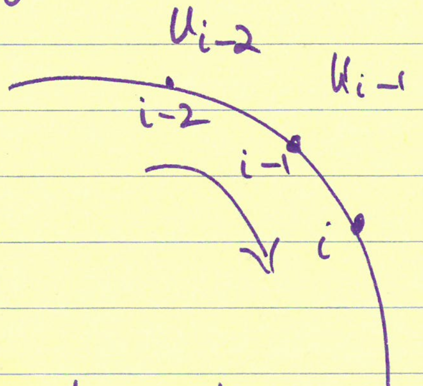
(a) LCR algorithm will work fine.

messages: $O(n^2)$
time: $O(n)$

(b) H S Algorithm., will work fine
messages: $O(n \log n)$
time: ?

Unidirectional rings.
Peterson's algorithm.

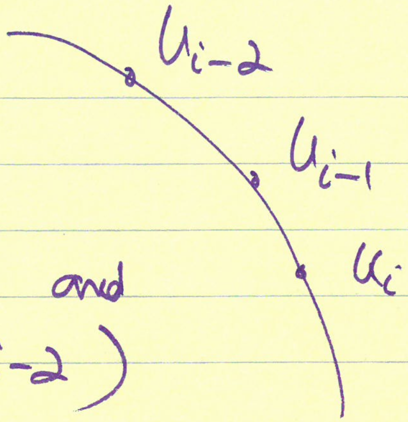
Start: phase 1
Send id for exactly 2 hops.



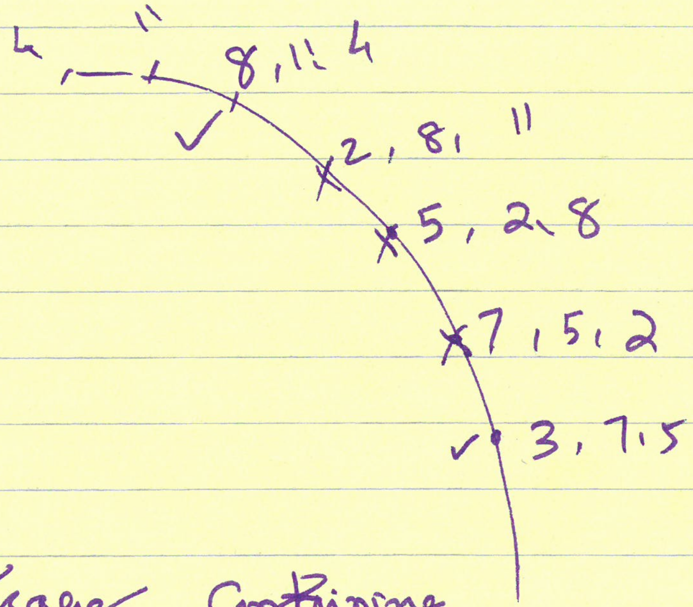
u_i , u_{i-1} , u_{i-2}

u_i proceeds to ~~the~~ next phase (phase 2) if u_{i-1} is max among (u_i, u_{i-1}, u_{i-2})

③



```
if ( $u_{i-1} > u_i$ and
     $u_{i-1} > u_{i-2}$ )
then // $u_i$ ~~st~~ initiates next phase.
        i's (new) temp id = $u_{i-1}$
```



```
     ~~Send message Containing~~
     { // go to next phase : initiate new token
        new temp id = $u_{i-1}$;
        Send new temp id to two
           successors that participate in next
           phase.
              ⋮
     }
else { // become a relay & will not
        initiate any message in future
     }
```

messages: $O(n \log n)$

<u>Lower bound:</u> $\Omega(n \log n)$

Arbitrary Networks (General Network)

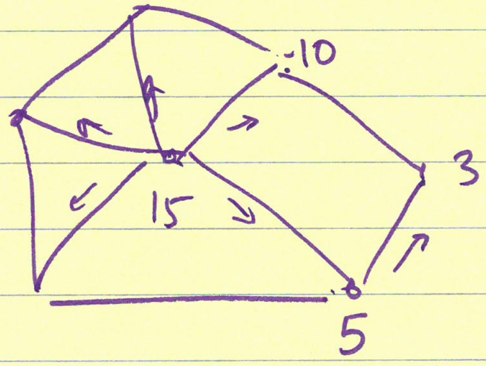
Leader election:

Floodmax ?

Termination ?

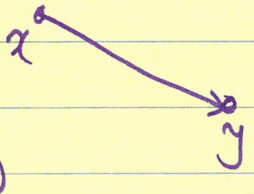
Convergecast acks / NACKs.

Form a tree rooted at u_{max}



not necessarily a BFS tree.

messages : $O(n|E|)$



Can reduce this by making some changes.:

Simulate "rounds"

in Chapter 16 on Synchronizers.

$O(\text{diam}\ |E|)$ messages.

BFS.

i_0 : source / root.

Asynch BFS.

a process may receive a message or a very large # of hops from the root.

Need to perform "relaxation" step