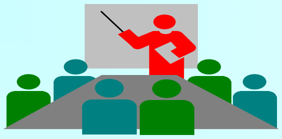
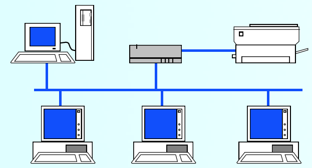




Distributed Systems
600.437
Asynchronous Models for Consensus

Department of Computer Science
The Johns Hopkins University

Asynchronous Models For Consensus

Lecture 5

Further reading:

Distributed Algorithms
Nancy Lynch,
Morgan Kaufmann Publishers, 1996.

[FLP85] Fischer, Lynch and Paterson. Impossibility of Distributed Consensus with One Faulty Process. *Journal of the ACM, 32,* pages 374-382. April 1985.

Distributed Consensus

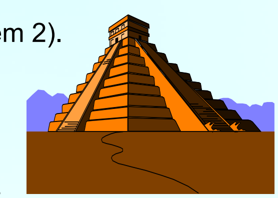
Problem 1- Consensus, synchronous settings, unreliable communication : impossible.

Distributed Consensus

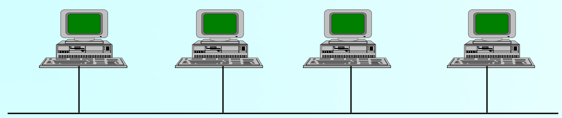
Problem 1- Consensus, synchronous settings, unreliable communication : impossible.

Problem 2 - Consensus, asynchronous settings, unreliable communication : impossible

(Problem 1 is a special case of Problem 2).



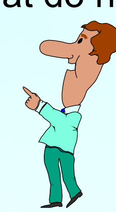
The Asynchronous Model



- Asynchronous setting.
- Complete network graph
- **Reliable** FIFO **unicast** communication.
- Deterministic processes, {0,1} initial values.
- **Fail-stop failures of processes are possible.** (remember that this is solvable in a synchronous setting).

Solution Requirements for Consensus

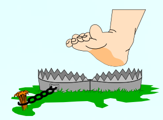
- **Agreement:** All correct processes decide on the same value.
- **Validity:** If a correct process decides on a value, then there was a process that started with that value.
- **Termination:** All processes that do not fail eventually decide.



Impossibility Result (FLP[85])

Definitions:

- **x-fair** execution: executions in which all channels execute fairly, and all processes but at-most x execute fairly.
- **0-RCP** : (0-**resilient consensus protocol**) - a protocol that solves consensus in all 0-fair executions.
- **1-RCP**: a protocol that solves consensus in all 0-fair and 1-fair executions.

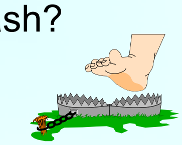


FLP[85] (Cont.)

FLP: There is no 1-Resilient Consensus Protocol !

Question1: Can you think of a 0-Resilient Consensus Protocol?

Question2: what can be problematic if one of the processes may crash?



More Definitions…

-A finite execution α is **0-valent** if 0 is the only decision value in all extensions of α.
-A finite execution α is **1-valent** if 1 is the only decision value in all extensions of α.
- α is **bivalent** if it is neither 0-valent nor 1-valent.

Lemma 1:

In any 1-Resilient Consensus Protocol there is a bivalent initial execution.

Proof of Lemma 1

- If (i1, i2, ..., in) = (0, 0, ..., 0) => decision is 0.
- If (i1, i2, ..., in) = (1, 1, ..., 1) => decision is 1.
- Assume that each vector (i1, i2, ..., in) is univalent.
- Look at: (0, 0, …, 0, 0), (1, 0, …, 0, 0), (1, 1, … 0, 0), … , (1, 1, … ,1, 0), (1, 1, …, 1, 1).
- from all the above, there exists two starting vectors that are identical except for one entry of some processor p, where v0 is 0-valent and v1 is 1-valent.
- Kill p at the beginning to reach a contradiction.

A Decider

A **Decider** for algorithm A consists of execution α of algorithm A and a process p such that:

- Execution α is bivalent.
- There exists 0-valent extension $\alpha 0$ of α such that the suffix after α consists of steps of p only.
- There exists 1-valent extension $\alpha 1$ of α such that the suffix after α consists of steps of p only.

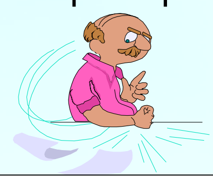
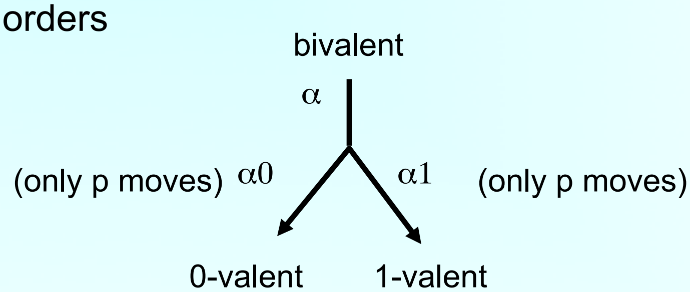


Illustration of a decider

- p may receive a message and then send a message **or** send a message and then receive a message.
- **Alternatively** p may receive 2 messages at different orders



Correctness of FLP

Lemma 2:

Let A be a 1-RCP with a bivalent initial execution. There exists a decider for A.

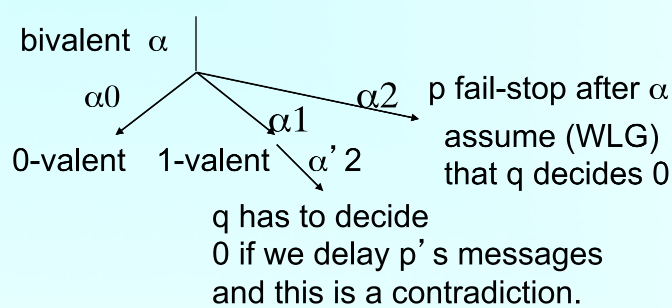
-- FLP is correct if Lemmas 1 and 2 are correct: Why?

Lemma 1: In any 1-RCP there is a bivalent initial execution.

Together they mean that :
in any 1-RCP there exists a decider.

Correctness of FLP (Cont.)

-- FLP is correct if Lemmas 1 and 2 are correct:

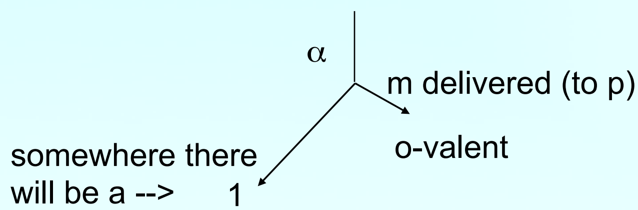


Note: only p moves in $\alpha 0$, $\alpha 1$

Proof of Lemma 2

For 1-RCP, we can delay messages from one process and still expect termination. (!)

Suppose that after α, a bivalent execution, the delivery of m to p yields a univalent execution. WLG assume it yields 0-valent.



Proof of Lemma 2 (cont.)

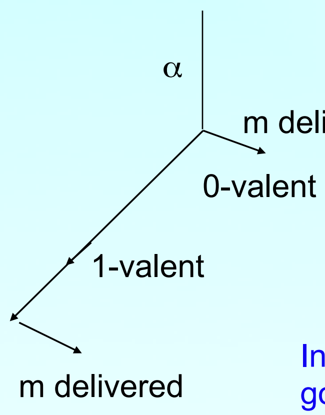
To reach a 1-valent extension of α there are two possibilities:

1. m is not delivered before decision is reached.
2. m is delivered somewhere before decision is reached.

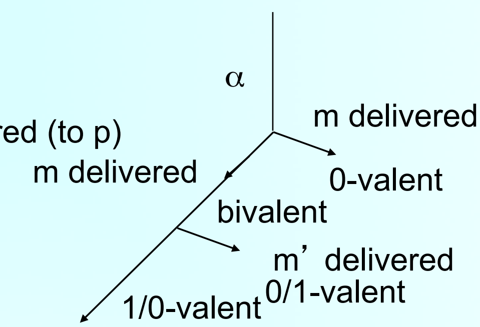
In the first case, we deliver m after the decision is reached. (i.e. after reaching a 1-valent execution).

Proof of Lemma 2 (cont..)

Case 1



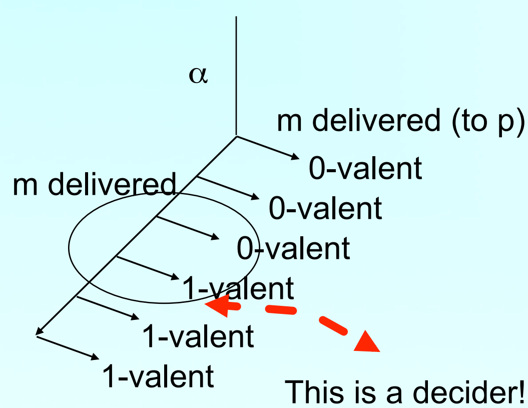
Case 2



In case 2, pick m’. This process of going down has to be finite because of termination.

Proof of Lemma 2 (end)

We stick the delivery of m after each step (look at Case 1)



There has to be a step which before it we have 0-valent and after 1-valent.

This step has to be made by p.

So, What can be done???

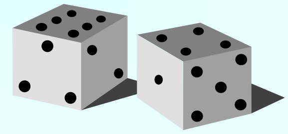
We need to pay something
in order to gain something else.

What can we pay?
(what can we gain?)

A Randomize Protocol for Consensus

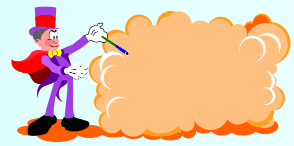
A complete network graph (clique)
n - total number of processes.
f - total number of faulty processes.
Assumption: **n > 5f.**

This algorithm solves a more complex problem where the failure model is **Byzantine**, i.e. the failed processes can send arbitrary messages to arbitrary processes (may lie), or may fail.



The protocol (Ben-Or variation)

Round=0; x = initial value
Do Forever:
    Round = Round + 1
    Step 1
    Step 2



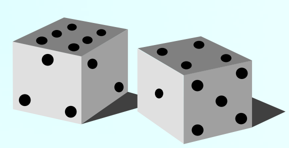
Step 1:
  Send Proposal(Round,x) to all processes
  wait for n-f messages of type Proposal(Round,*)
  if at least n-2f messages have the same value v
      then x = v   (that value)
      else x = undefined

The Protocol (cont.)

Step 2:
  Send Bid(Round,x) to all processes
  wait for n-f messages of type Bid(Round,*)

  v is the real value (0/1) occurring most often
  and m is the number of occurrences of v

  if m >= 3f+1
      then **Decide** (x=v)
  else if m >= f+1
      then x = v
    else x = **random** (0 or 1)



Other Ways to Bypass The Impossibility Result

- To allow the protocol not to guarantee agreement.
- To allow the protocol not to always terminate at all correct members:
  - The Transis membership can exclude live but “slow” processes from the membership, and will reach “agreement” among the connected members.