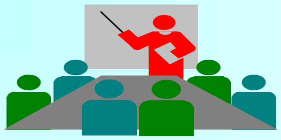
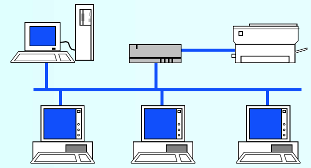




Distributed Systems 601.417

Synchronous Models for Consensus

Department of Computer Science
The Johns Hopkins University

Synchronous Models For Consensus

Lecture 2

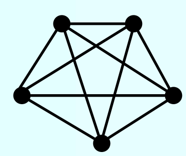
Further reading:

Distributed Algorithms
Nancy Lynch,
Morgan Kaufmann Publishers.

Distributed Consensus

No Faults
Problem Description

Assumptions:



- *n* processes connected by a full graph.
- Each process starts with an initial value {0, 1}.
- Synchronous settings: every message is received (if not lost) in the same epoch in which it is sent.
- No Faults case: No process faults or message omissions.
- solution is required within *r* rounds for some fixed *r*.

No Faults
Problem Description (cont.)

Requirements:

- **Agreement:** All processes decide on the same value.
- **Validity:** If a process decides on a value, there was a process that started with that value.

What if we eliminate the validity requirement ?

No Faults
Problem Description (cont.)

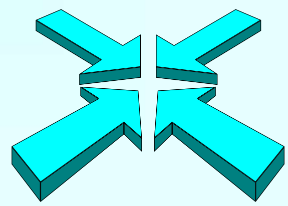
Requirements:

- **Agreement:** All processes decide on the same value.
- **Validity:** If a process decides on a value, there was a process that started with that value.

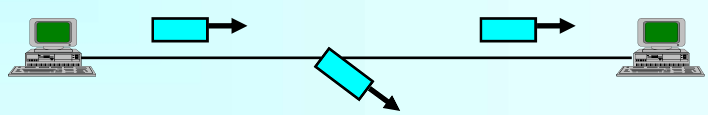
The validity requirement eliminates trivial meaningless solutions.

No Faults
One-Round Algorithm

- Send your value to all the processes.
- If all the values you have (including your own) are 1 then decide 1. Otherwise decide 0.



Message Omissions
Problem Description



Assumptions:

- *n* processes connected by a full graph.
- Each process starts with an initial value {0, 1}.
- Synchronous setting - solution is required within *r* rounds for some fixed *r*.
- Any number of messages may be lost.

Message Omissions
Problem Description (cont.)

Requirements:

- **Agreement:** All processes decide on the same value.
- **Validity:** If all processes start with 0, then the decision value is 0; if all processes start with 1 and no message is lost, then the decision value is 1.

Notice that the validity requirement is **weaker** then the original validity requirement.

Message Omissions
Consensus is Not Solvable!

Theorem: There is no algorithm that solves the consensus problem for even 2 processes.

Definition: Execution α is **indistinguishable** from execution β with respect to process *p* if in both α and β, *p* has the same initial state and receives exactly the same messages at the same rounds.

$$\alpha \overset{p}{\sim} \beta$$

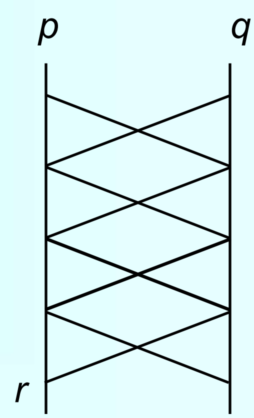
Proof

Assume there is a correct algorithm that solves consensus

$\alpha 1$: Both processes start with 1 and no message is lost.

$\alpha 2$: Similar to $\alpha 1$ except that the last message from *p* to *q* is lost.

$\alpha 3$: Similar to $\alpha 2$ except that the last message from *q* to *p* is lost.



$\alpha 1 \overset{p}{\sim} \alpha 2 \quad \alpha 2 \overset{q}{\sim} \alpha 3$

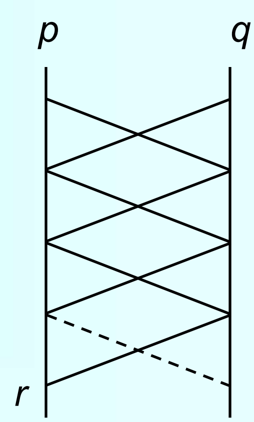
Proof

Assume there is a correct algorithm that solves consensus

$\alpha 1$: Both processes start with 1 and no message is lost.

$\alpha 2$: Similar to $\alpha 1$ except that the last message from *p* to *q* is lost.

$\alpha 3$: Similar to $\alpha 2$ except that the last message from *q* to *p* is lost.



$\alpha 1 \overset{p}{\sim} \alpha 2 \quad \alpha 2 \overset{q}{\sim} \alpha 3$

Proof

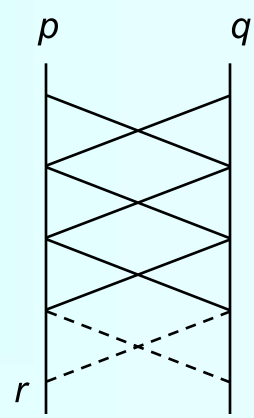
Assume there is a correct algorithm that solves consensus

α1: Both processes start with 1 and no message is lost.

α2: Similar to α1 except that the last message from *p* to *q* is lost.

α3: Similar to α2 except that the last message from *q* to *p* is lost.

$\alpha 1 \overset{p}{\sim} \alpha 2 \quad \alpha 2 \overset{q}{\sim} \alpha 3$



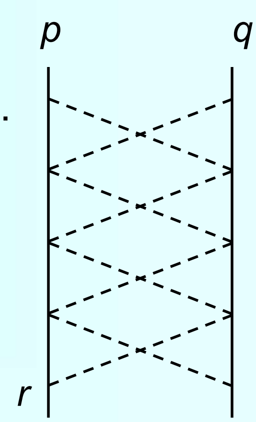
Proof (cont.)

αx: Both processes start with 1 and all messages are lost.

βx: Similar to αx except that *q* starts with 0.

βy: Similar to βx except that *p* starts with 0.

$\alpha x \overset{p}{\sim} \beta x \quad \beta x \overset{q}{\sim} \beta y$



Contradiction

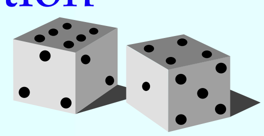
Message Omissions Randomized Consensus

An Adversary is an arbitrary choice of:

- Initial values for all processes.
- Subset of { (*p*1, *p*2, *i*) } where *p*1, *p*2 are processes and *i* is a round number.

The subset represents which messages are lost.

Message Omissions Randomized Solution



Requirements:

- **Agreement:** For any adversary A:

  The probability that some process decides 0 and some process decides 1 is less or equal to ε.

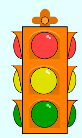
- **Validity:** If all processes start with 0, then the decision value is 0; if all processes start with 1 and no message is lost, then the decision value is 1.

Message Omissions
A Randomized Algorithm

At initialization one specific process, *p*, chooses a *key* at **random**, uniformly from the range [1..*r*].

At each round the processes send the following:

- Initial value.
- *key* (for process *p* only).
- *color*

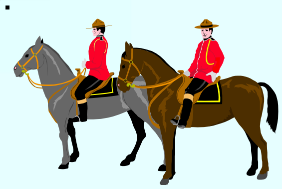


Each process holds a variable *color* initialized to **green**. If red message was received, or a message was missed, the process sets *color* to red.

Message Omissions
A Randomized Algorithm (cont.)

Decision Rule:

A process decides 1 after *r* rounds if it knows that at least one process started with 1, it knows the value of *key*, and it has received all the messages in all the first *key* rounds and all of them were **green**. Otherwise, it decides 0.

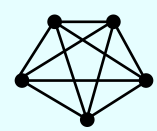


Correctness Proof

Set r to be an integer that is bigger or equal to the desired $1/\varepsilon$. The algorithm satisfies the agreement and validity requirements because **for any adversary**:

- If no message is lost then all processes get all messages and decisions will be identical.
- Look at the first message omitted by the adversary: only if this message is omitted at round *key* there might be disagreement.
- Remember that *key* is selected uniformly at random from the range [1..r].

Fail-Stop Faults Problem Description



Assumptions:

- n processes connected by a full graph.
- Each process starts with an initial value {0, 1}.
- Synchronous setting - solution is required within r rounds for some fixed r.
- The number of Fail-Stop faults is bounded in advance to f. A process may fail in the middle of message sending at some round. Once a process fails, it never recovers.
- No omission failures.

Fail-Stop Faults
Problem Description (cont.)

Requirements:

- **Agreement:** All correct processes do decide on the same value.
- **Validity:** If a correct process decides on a value, there was a process that started with that value.

Fail-Stop Faults
f+1 Rounds Algorithm

Each process maintains a vector containing a value for each process {0,1, u}. u = undefined.

round
- Send your vector to all processes.
- Update local vector according to the received vectors (in case local vector has a "u", and any of received vectors contain "0" or "1").

- After *f+1* rounds decide according to the local vector. If you have 1 in the vector then decide 1, otherwise decide 0.