Distributed Systems
601.417
Synchronous Models for Consensus

Department of Computer Science
The Johns Hopkins University

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 \( \{0, 1\} \).
- \textbf{Synchronous settings}: every message is received (if not lost) in the same epoch in which it is sent.
- \textbf{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** than 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 $\alpha$ is **indistinguishable** from execution $\beta$ with respect to process $p$ if in both $\alpha$ and $\beta$, $p$ has the same initial state and receives exactly the same messages at the same rounds.

$$\alpha \sim_p \beta$$
Proof

Assume there is a correct algorithm that solves consensus

\[ \alpha_1: \text{Both processes start with 1 and no message is lost.} \]

\[ \alpha_2: \text{Similar to } \alpha_1 \text{ except that the last message from } p \text{ to } q \text{ is lost.} \]

\[ \alpha_3: \text{Similar to } \alpha_2 \text{ except that the last message from } q \text{ to } p \text{ is lost.} \]

\[ \alpha_1 \sim_p \alpha_2 \quad \alpha_2 \sim_q \alpha_3 \]
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.

\( \alpha_1 \sim^p \alpha_2 \quad \alpha_2 \sim^q \alpha_3 \)

Proof (cont.)

\( \beta_x: \) Both processes start with 1 and all messages are lost.

\( \beta_y: \) Similar to \( \beta_x \) except that \( q \) starts with 0.

\( \beta_x \sim^p \beta_y \quad \beta_x \sim^q \beta_y \)

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 \( \varepsilon \).

• **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.