EECS 591  Handout #6

- Distributed Consensus
  1. Definition of distributed consensus problem
  2. What is impossible? What is possible?
  3. Relationship to Byzantine agreement problem?

- Reading List:
  Turek & Shaha survey, IEEE Computer
  Tanenbaum, section 7.2.3
  Mullender, Chapter 5.6-5.8, pp. 126-132 (recommended)

An Example ... Motivation

Suppose that Group#5 and Farnam are planning to set up a meeting at 4pm Tuesday via e-mail: Assume message loss can occur.

```
“Meeting on Tuesday at 4pm”
Group#5 __________________> Farnam
“OK. Meeting on Tuesday at 4pm”
Group#5 <___________________ Farnam
```

- What do Group#5 and Farnam know about each other at time (1)?
- What do Group#5 and Farnam know about each other at time (2)?
- Does Group#5 know at time (2) that its message was delivered earlier?
- What if we could guarantee bounded message delivery or the detection of failure within bounded time?
- What if each side flooded the other side with messages and the probability of p for each flood?
**Observations:**

- Ensuring reliable delivery is easier than mutual agreement!
- If we knew that messages will be delivered within a known bound, then we can reach agreement.
- Playing the probabilities game pays off! Two-message protocol with HP.

**Examples of Consensus Problem:**

- electing a coordinator
- commit a transaction
- synchronization

---

**Distributed Consensus**

**CONSENSUS**

All correct processes propose a value, and must agree on a value related to the proposed values!

**Definition:** The Consensus problem is specified as follows:

- **Termination:** Every correct process eventually decides some value.
- **Validity:** If all processes that propose a value, propose \( v \), then all correct processes eventually decide \( v \).
- **Agreement:** If a correct process decides \( v \), then all correct processes eventually decide \( v \).
- **Integrity:** Every process decides at most once, and if it decides on \( v \neq NU \), then some process must have proposed it. (\( NU \) is a special value which stands for ‘no unanimity’).
ASSUME NO PROCESSOR FAILURES ...

In a synchronous system in which messages can be dropped (i.e., unreliable communication), consensus is not possible even if none of the processors fail.

Why?
Farnam and Group #5 cannot agree on the time of the meeting.
Impossibility Results for Asynchronous Systems

ASSUME CRASH FAILURES ONLY...

Fischer et. al. Result:

No deterministic algorithm exists that solves Distributed Consensus in an asynchronous system and tolerates even a single crash failure!

- This holds even if the communication network is reliable and completely connected!

Why?

- In an asynchronous systems, a process that has crashed cannot be distinguished from a process that is correct but it is extremely slow!

- What is the significance of this result?
  
  - Asynchronous model of computation (with crash failures) is too restrictive for fault-tolerant applications.
  - The result does not say what can be achieved in practice.
In asynchronous systems, reliable multicast, FIFO multicast, and causal multicast can be implemented.

**Distributed Consensus = Atomic Broadcast**

Distributed Consensus can be reduced to Atomic Multicast with no assumption on the system sync or topology, and any benign failures.

Atomic multicast can be reduced to Distributed Consensus under crash failures.

- Atomic multicast cannot be solved *deterministically* in asynchronous systems, even if we assume that at most one processor can fail, and it can fail by crashing only.
- Atomic multicast can be solved using *randomization* or *failure detectors* in asynchronous systems.

---

Dolev, Dwork, and Stockmeyer (1987) identified a set of system parameters:

- Processors can be synchronous or asynchronous.
- Communication delay can be bounded or unbounded.
- Messages can be ordered or unordered.
- Message transmission can be p-t-p or broadcast.

Conditions under which consensus is possible? Table 1 in survey paper by Turek & Shasha.
What is possible?

Relax Asynchronous Assumptions:
- asynchronous processors,
- unordered messages, or
- unbounded message delay.

- **Case 1**: Processors are synchronous and communication is bounded.

- **Case 2**: Messages are atomically ordered and the transmission mechanism is broadcast.

- **Case 3**: Processors are synchronous and messages are ordered.

Assignment: Propose an algorithm for case 1 and case 2. State informally why each solve the problem.
What can we do in practice?

1. Randomized algorithms:
   - A decision is made with high probability within a bounded amount of time.
   - If a decision is made, it will be correct.

2. Failure Detectors:
   - Build unreliable failure detectors first: which processor has crashed?
   - A failure detector can make a mistake.
   - Group membership problem.
   - Often use timeouts; so it introduces synch. assumption at some level.

3. Partial Synchrony (previous page)

Byzantine Agreement

The problem of reaching Distributed Consensus when processes can have Byzantine failures.

- Why consider Byzantine failures?
  - Model hardware failures.
  - Software design errors.
  - Failure of a layer.

- What do we know about it?
  - May be impossible even under synchronous assumption!
  - In a synchronous system with a reliable and fully connected network, no link failures, and processors subject to Byzantine failures, algorithms exists if $t \geq 3f+1$ (fewer than $1/3$ processors fail);
    no algorithm if $t \leq 3f$.
  - If we use message authentication, solution exists for any number of failures. Why?