Fault-tolerant Consensus in Distributed Systems

Hong Jiang (Theory Clique Talk)

February 14, 2005

The lunch appointment problem

Formally known as the coordinated attack problem.
In a department with faulty email servers ...

10:00AM: Alice $\overset{\text{email}}{\longrightarrow}$ Bob: "Let's meet at noon for lunch in front of AKW.".

10:20AM: Bob receives the email and checked his calendar. Bob $\overset{\text{email}}{\longrightarrow}$ Alice: "Okay, see you then.".

10:45AM: Alice received Bob's message. A thought occurs to her "If Bob doesn't know that I received his acknowledgment, he might think I won't wait for him. I'd better acknowledge his acknowledgment."

... ...

The lunch appointment problem (continued)

Theorem

Neither Bob nor Alice will make it to AKW, unless at least one of them is willing to risk waiting in the cold without meeting the other.

Lessons learned:

- Consensus can be hard with faults.
- Transmission is easier than mutually coordinated action.

Communication models

- Message passing: processes and channels form a communication graph.
  - Processes communicate by sending messages over channels.
  - The communication graph is assumed to be complete in this talk.
  - The type of channels specifies their characteristics such as FIFO, unbounded/bounded delay.
- Shared memory
  - Processes communicate via a set of shared variables.
  - The type of the shared variables specifies the operations that can be performed on them.

Timing models

- Asynchronous systems
  - No upper bound on message delay or the time that elapses between consecutive steps of a processor.
  - Every message sent is eventually delivered.
  - No process is blocked indefinitely from making progress.
  - Real-world example: the Internet (email services etc.).
- Synchronous system:
  - Processes proceed in lockstep: each process takes a step and can send an message to each neighbor in each round.
  - Messages are delivered at the beginning of the next round.

Failure models

- Fail-stop failures
  - Processes fail by stopping.
- Byzantine failures
  - Processes fail by acting maliciously.
  - A pessimistic model of software failures.

Consensus in asynchronous systems

Back to lunch appointment example:

- The department switched to a stable email server.
- Either Bob or Alice could be fired at any time.

Is there a solution for them?

Consensus in asynchronous systems (continued)

Theorem (FLP Impossibility)

In a distributed system with an unbounded but finite message delay, there is no protocol that can guarantee consensus within a finite amount of time if even a single process can fail by stopping.

This impossibility result was later proven for systems with asynchronous processes and shared-memory supporting only atomic reads and writes.

System parameters

- Processes

  Synchronous There exists a constant $s > 1$ s.t. for every $s + 1$ steps taken by any processor, every other processor will have taken at least one step.

  Asynchronous Otherwise.

- Communication delay

  Bounded Every message sent by a processor arrives at its destination within t real-time steps.

  Unbounded Otherwise

System parameters (continued)

- Delivery order
  - **Ordered** Messages are delivered in the same real-time order in which they are sent.
  - **Unordered** Messages are delivered in an arbitrary order.
- Transmission mechanism
  - **Point-to-point** A processor can send a message in an atomic step to at most one processor.
  - **Broadcast** A processor can send a message to all the processors in an atomic step.

Summary of possibilities

Processes	Message order				Communi-cation
	Unordered		Ordered		
Asyn-chronous	No	No	Yes	No	Unbounded
	No	No	Yes	No	Bounded
Syn-chronous	Yes	Yes	Yes	Yes	
	No	No	Yes	Yes	Unbounded
	Point-to-point	Broadcast		Point-to-point	
	Transmission				

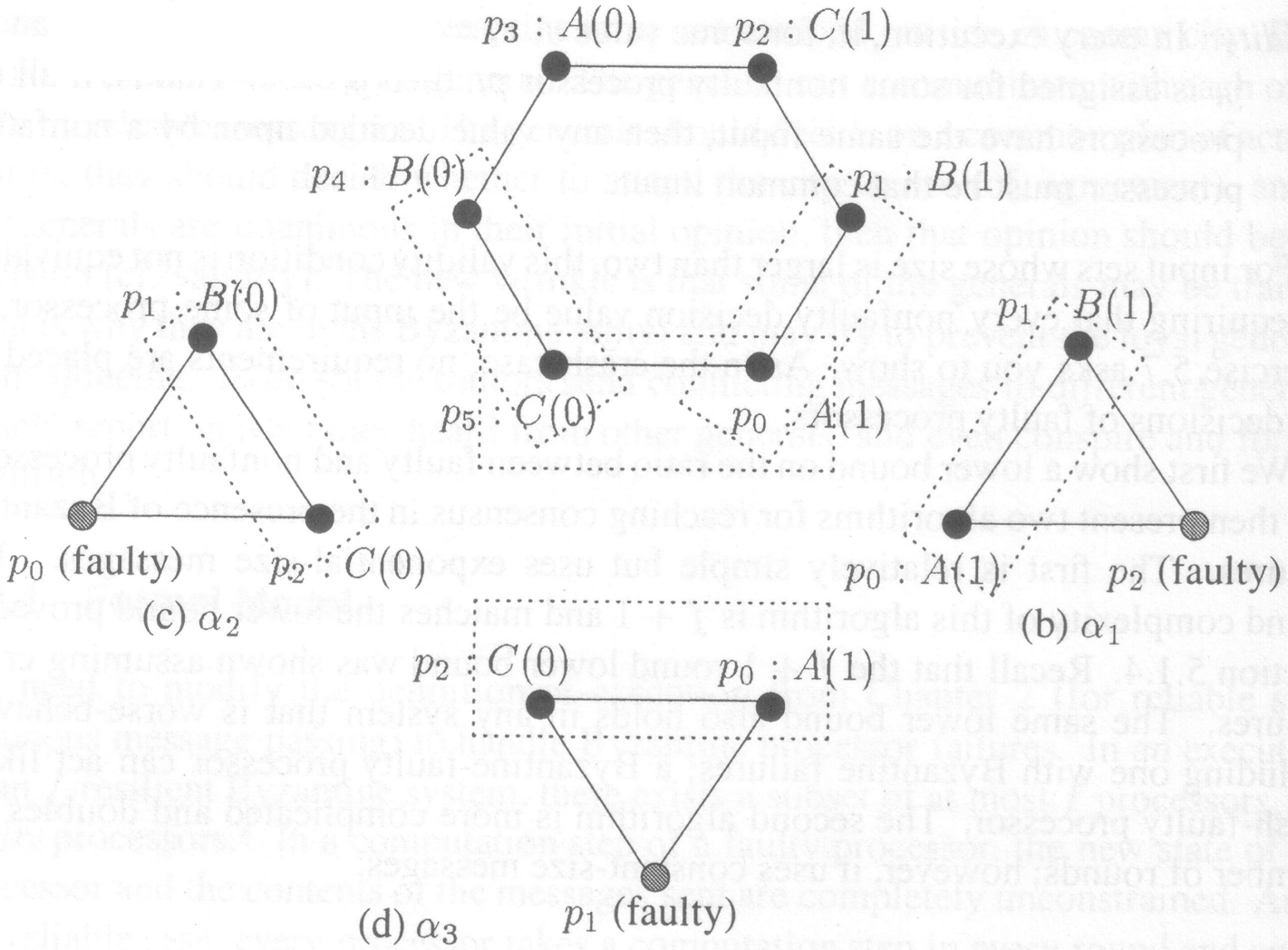
The 3-person lunch appointment problem

- Bob, Alice, and Charlie want to get together for lunch, using the telephone to communicate.
- The three-way-calling service is not available.
- One of them is trying to make one of the other two wait in the snow.

The 3-person lunch appointment problem (cont.)

Could they design a protocol such that ...

1. the two honest people will agree on whether or not to meet;
2. if all honest people want to meet, they will meet;
3. if no honest people want to meet, they won't meet.


$p_3 : A(0)$
$p_2 : C(1)$
$p_4 : B(0)$
$p_1 : B(1)$
$p_5 : C(0)$
$p_0 : A(1)$
$p_1 : B(0)$
p_0 (faulty)
$p_2 : C(0)$
(c) α_2
$p_1 : B(1)$
$p_0 : A(1)$
p_2 (faulty)
(b) α_1
$p_2 : C(0)$
$p_0 : A(1)$
p_1 (faulty)
(d) α_3

A bound for faults

Theorem

Byzantine agreement is possible if and only if there are at least $3k + 1$ *processes when* k *of the processes can fail.*

More recent development

- Fail-stop faults in asynchronous systems:
  - Weakened requirements
  - Probabilistic algorithms
  - Strengthened system model (failure detectors, time-outs)
- Byzantine faults in asynchronous systems:
  - Authentication, signature, and other crypto techniques