FLP

What about the asynchronous model?

Theorem

There is no deterministic protocol that solves Consensus in a message-passing asynchronous system in which at most one process may fail by crashing

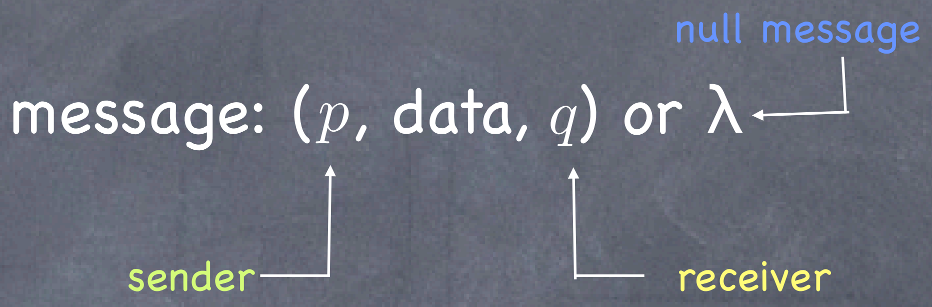
(Fisher, Lynch, and Paterson. Impossibility of distributed consensus with one faulty process. JACM, Vol. 32, no. 2, April 1985, pp. 374-382)

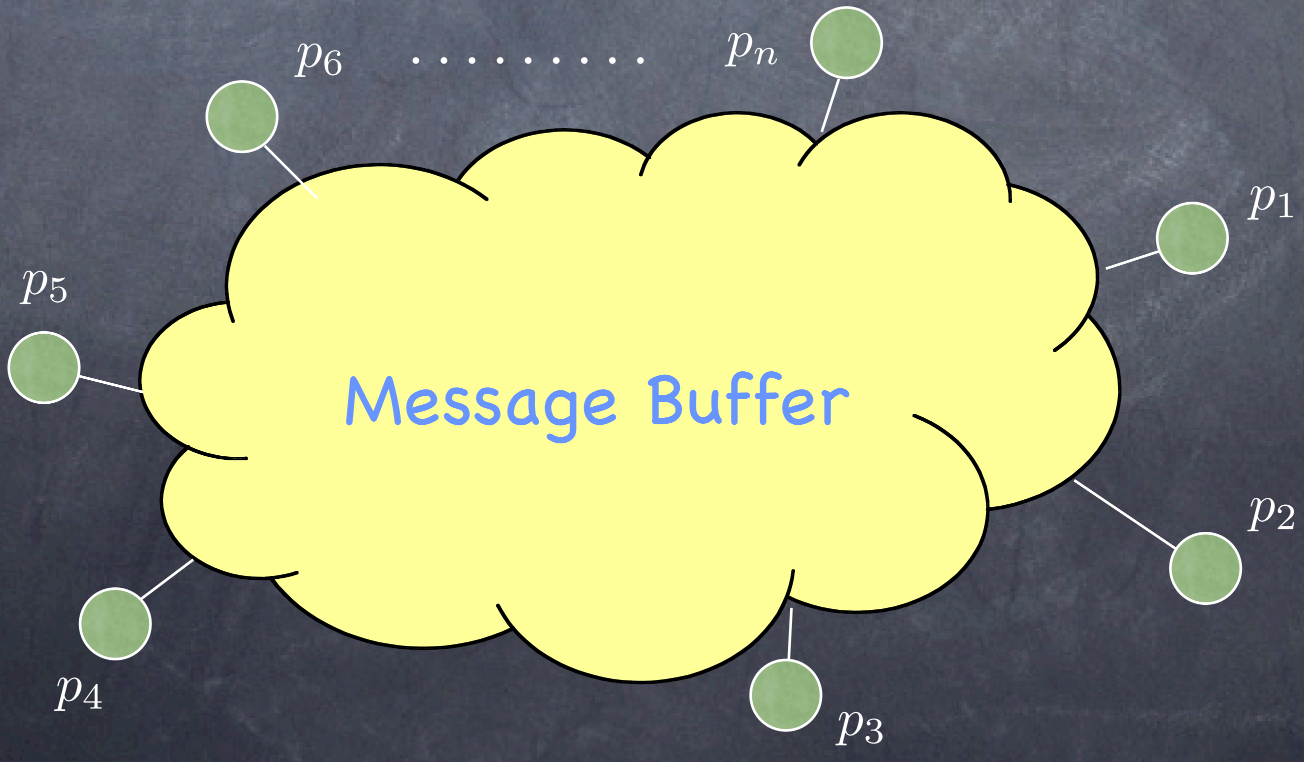
The Intuition

- In an asynchronous system, a process p cannot tell whether a non-responsive process q has crashed or it is just slow
- If p waits, it might do so forever
- If p decides, it may find out later that q came to a different decision

The Model - 1

- n processes
- a message buffer





The Model - 2

- An algorithm A is a sequence of steps
- Each step consists of two phases
  - Receive phase - some p removes from buffer $(x, data, p)$ or λ
  - Send phase - p changes its state; adds zero or more messages to buffer
- p can receive λ <u>even if there are messages for</u> p <u>in the buffer</u>

Assumptions

Liveness Assumption:

Every message sent will be eventually received if intended receiver tries infinitely often

One-time Assumption:

p sends m to q at most once

WLOG, process p_i can only propose a single bit b_i

Configurations

- A configuration C of $\mathcal{A}$ is a pair (s, M) where:
  - s is a function that maps each p_i to its local state
  - M is the set of messages in the buffer
- A step $e \equiv (p, m, \mathcal{A})$ is applicable to $C = (s, M)$ if and only if $m \in M \cup \{\lambda\}$. Note: $(p, \lambda, \mathcal{A})$ is always applicable to C
- $C' \equiv e(C)$ is the configuration resulting from applying to C

Schedules

- A schedule S of A is a finite or infinite sequence of steps of A
- A schedule S is applicable to a configuration C if and only if either
  - ☐ S is the empty schedule $S_\perp$ or
  - ☐ $S[1]$ is applicable to C ;
    $S[2]$ is applicable to $S[1](C)$; etc.
- If S is finite, $S(C)$ is the unique configuration obtained by applying S to C

Schedules and configurations

- A configuration C' is accessible from a configuration C if there exist a schedule S such that $C' = S(C)$
- C' is a configuration of $S(C)$ if $\exists S'$ prefix of S such that $S'(C) = C'$

Runs

- A run of $\mathcal{A}$ is a pair <I, S> where
  - I is an initial configuration
  - S is an infinite schedule of $\mathcal{A}$ applicable to I
- A run is partial if S is a finite schedule of $\mathcal{A}$
- A run is admissible if every process, except possibly one, takes infinitely many steps in S
- An admissible run is unacceptable if every process, except possibly one, takes infinitely many steps in S without deciding

Structure of the proof

- Show that, for any given consensus algorithm $\mathcal{A}$, there always exists an unacceptable run
- In fact, we will show an unacceptable run in which no process crashes!

Classifying Configurations

0-valent: A configuration C is 0-valent if some process has decided 0 in C, or if all configurations accessible from C are 0-valent

1-valent: A configuration C is 1-valent if some process has decided 1 in C, or if all configurations accessible from C are 1-valent

Bivalent: A configuration C is bivalent if some of the configurations accessible from it are 0-valent while others are 1-valent