Classification of Distributed Systems

- I Distributed Systems can be classified according t several characteristics:

1. **Synchrony degree;**
2. **Type of failures;**
3. **(Network topology)… and more**

Synchronous Systems

- A system is **synchronous** when the following properties hold:

- 1. there exists -and it is known - an upper bound on message delivery delay;
- 2. the drift rate of the local clock of each processor wrt a real time base is bounded and known;
- 3. there exist a known upper bound on the processing speed of each processor.

- Advantages:
  - Mechanisms for fault detection can be implemented (Time-outs).
- Limits:
  - It is **(was??)** far from easy to ensure this property on a large scale system over a long time scale.

Asynchronous Systems

- A distributed system is **ASYNCHRONOUS** if there is no limit EITHER on message delivery delay OR on the local clocks drift OR on the time needed for performing a computational step.
- An asynchronous system is characterized by the lack of whatsoever assumption on TIME (*time-free*).
- Advantages:
- better portability of applications;
- variable or unexpected working loads cause asynchrony, therefore the assumption we make on synchronous model become valid only in a PROBABILISTIC way.
- The two models constitute the extremes of a continuum.

Failure Classification

- Failures hit both the processes (nodes) and the communications.

- Failure Types (by complexity):
- - Crash;
- - Omission;
- - Value;
- - Byzantine;

- If there is a time reference (synchronous systems):
- - Timing: early or late.

Distributed Consensus (1)

- Necessary and used for building reliable and secure distributed applications and services (block chain)
- It allows 2 or more entities (nodes, processes, parties in general) to reach a common agreed decision starting from their initial valies (proposals).

- Exemples :
- leader election;
- agreement on the values of many replicated;
- error detection in redundant systems;
- Bitcoin or smart contracts transactions in the blockchains

Distributed Consensus (2) qui

- Formally defined in terms of the following properties:

- ***Termination:*** every correct process eventually (sooner or later) decides;
- ***Uniform integrity****:* every correct process decides at most only one time;
- ***Agreement****:* two correct process do not decide in a different way;
- ***Uniform validity****: if a* correct process decides for a given value *v*, then *v* has been proposed by some process.

Reliable Broadcast (1)

- Used for an efficient and reliable data exchange.
- If a system has a broadcast mechanism we need to avoid that the occurrence of a (node) failure will make the state of the entire system inconsistent.
- Properties:
- 1. every correct process agree on the message that is delivered;
- 2. all the messages sent by broadcast by correct processes are delivered;
- 3. no spurious message will ever be delivered.

- V. Hadzilacos and S. Toueg, "*Fault-tolerant Broadcasts and Related Problems,*" Distr. Syst., Chap. 5, S. J. Mullender Ed., Addison-Wesley, Reading, Mass., pp. 97-145, 93.

Reliable Broadcast (2)

- Reliable Broadcast algorithm for asynchronous systems.
- Use of the message diffusion principle: when a process receives a message for the first time, it forwards the message to all the other processes and then performs the R-delivery.
- This algorithm satisfies the properties listed in the previous foil in asynchronous systems with at most n-1 crash failures.
  - For each process p:

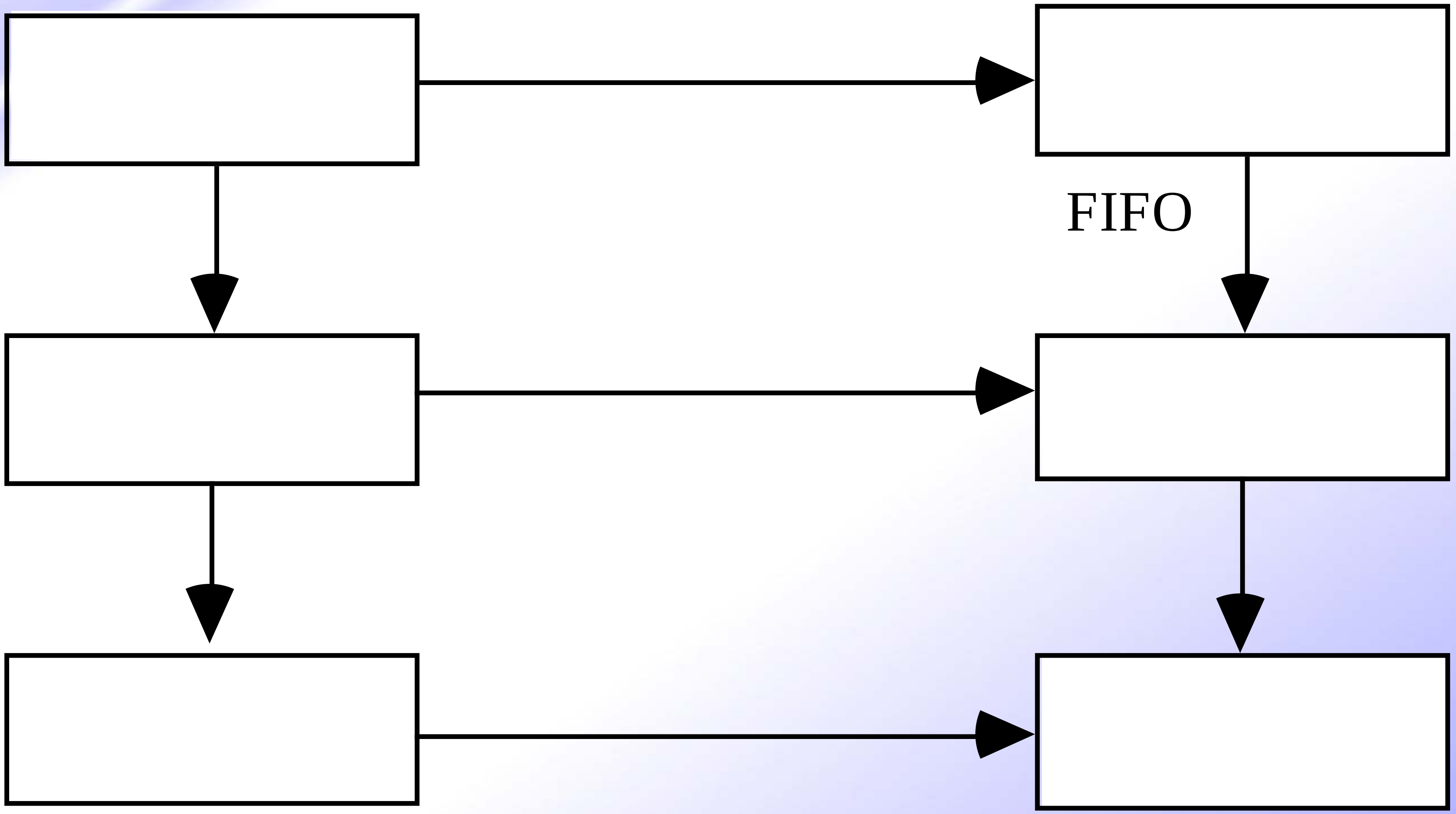
```
R-broadcast(m):        send(m) to all (including p);

R-delivery(m):
        when receive(m) for the first time
          if sender(m) ≠ p then send m to all
          R-deliver(m)
```

Variants of Reliable Broadcast

- By adding further properties we can obtain the following broadcast models:
- **FIFO Broadcast**: messages are delivered in FIFO order;
- **Causal Broadcast**: the delivery order follows causality relationships.
- **Atomic Broadcast**: all messages are delivered in Total Order (same delivery order also for messages not related through the previous relations).

- Combining Total Order with a FIFO o causal ordering we obtain **FIFO Atomic Broadcast** and **Causal Atomic Broadcast** models respectively.

Relations



Atomic Broadcast and Consensus

- It is immediate to prove that Consensus can be reduced to Atomic Broadcast: when a process is willing to propose a value to decide it performs the atomic broadcast when instead it wants to decide for a value it considers the first message that it delivers.

- If we assume to consider only benign failures (non arbitrary failures) allows to implement Reliable Broadcast, FIFO Reliable Broadcast and Causal Broadcast both in synchronous systems and in asynchronous systems (as long as failures do not partition the network).

- **This is not true for Atomic Broadcast as demonstrated by Fischer, Lynch e Paterson**.

FLP theorem

There is **NO** deterministic algorithm
able to solve the consensus problem
in an asynchronous system
which is able to tolerate
even only one crash failure

- *M. Fischer, N. Lynch, and M. Paterson, "Impossibility of Distributed Consensus with One Faulty Process," Journal of ACM, Vol. 32, pp. 374-382, April 1985.*

Proposed approaches to solve consensus

- The bottomline of the consensus problem in asynchronous systems is the impossibility to distinguish between a process hit by a failure and a very slow one.
- Many models have been proposed (since 1985) to overcome this impossibility results :

- Failure Detectors;
- (Partially synchronous systems);
- (quasi-synchronous systems)
- Timed-asynchronous systems.

Essential Bibliografy (1)

- FLP Theorem :
  - M. Fischer, N. Lynch, and M. Paterson, "*Impossibility of Distributed Consensus with One Faulty Process*," Journal of ACM, Vol. 32, pp. 374-382, April 1985.
- Failure detectors:
  - T. Chandra, V. Hadzilacos, and S. Toueg, "*The Weakest Failure Detector for Solving Consensus*," Journal of the ACM, Vol. 43(4), pp. 685-722, July 1996.
  - T. Chandra and S. Toueg, "*Unreliable Failure detectors for Reliable Distributed Systems*," Journal of the ACM, Vol. 43(2), pp. 225-267, Mar. 1996.
- Partially synchronous systems
  - D. Dolev, C. Dwork, and L. Stockmeyer, "*On the Minimal Synchronism Needed for Distributed Consensus,*" Journal of the ACM, Vol. 34(1), pp. 77-97, Jan 1987.
  - C. Dwork, N. Lynch, and L. Stockmeyer, "*Consensus in the Presence of Partial Synchrony,*" Journal of the ACM, Vol. 35(2), pp. 288-323, Feb. 1988.
- Timed Asynchronous Systems:
  - F. *Cristian, and C. Fetzer, "The timed Asynchronous Distributed System Model," 28th Intern. Symp. On Fault-tolerant Computing (FCTS-28), (Munich, Germany), pp. 140-149, IEEE Computer Society Press, 1998.*
  - *C. Fetzer, and F. Cristian, "On the Possibility of Consensus in Asynchronous Systems," in Pacific Rim Intern. Symp. on Fault-tolerant Systems , (Newport beach CA), 1995*