Distributed System 1

Francesco Penasa

February 18, 2020

18/02/2020

1 Synchronization in Distributed Systems

Time plays a fundamental role in many applications. At some point the clock diverge
clock drift ρ costant of variation of time.
For example

$$\rho = 10^{-6}\frac{s}{s}$$

1s every 11.6 days
clock skew δ maximum clock drift allowed.

accuracy: synchronize all clocks against a single one, which is usually the more accurate
agreement synchroniza all clocks among themselves.
Monocity must be preserved

1.1 Protocols

1. Server-based soution: periodically each client update himself with a server clock adding the offset θ.
2. Network Time Protocol (NTP): protocol to synch time over large-scale networks `ntp.org`. The NTP servers are in Hierarchical structure (as DNS). Synchronization mechanisms: multicast, procedure-call mode, symmetric mode Symmetric Mode: server exchange their roles and exchange the time.

1.2 Observation

1. Often is sufficient to agree on a time, even if it is not accurate.
2. what matters is ordering and causality (relative order). (example: allarm -¿ fire; fire -¿ allarm).
3. if there is no interaction, no sunchronization is required.

2 Modeling a distributed execution

A distributed algorithm can be modeled as a collection of distributed automata.

relevant event in distributed algorithms

1. send event: *send(m,p)*
2. receive event: *receive(m)*
3. local event: everything else (set a variable, write a file...)

Histories

1. local history: history if a process p_i
2. partial history: ...

Happens-before causality, it is useful only if we don't have global time.

Logical clocks enable coordination among processes without synchronization of physocal clocks, essetially a counter. **Definition:** Logical clock LC is a map function for the events e of the history H to an element of a time domain T

$$LC : H \to T$$

Clock consistency: events could be concurrent

$$e \to r' \Rightarrow LC(e) < LC(e')$$

Strong clock consistency events can not be concurrent

$$e \to e' \Leftrightarrow LC(e) < LC(e')$$

Scalar Clocks How to assign logical clocks in a way that guarantees **clock consistency**.
Definition Scalar logical clocks: an increasing counter
Update rule... which guarantees clock consistency by design.

Partial vs Total Order I can add the name of the process to the time to disambiguate the order of the events.

3 Questions

Scalar clock EXERCISE put the number on the events...

what are the problems on the server-based solutions?

1. **Minor:** There is already a mismatch for the "travel" time, to fix this we can use timestamps T1, T2, T3, T4 to measure the RTT.
2. **Major:** We could broke monotonicity of time, to fix the clock we should slow down until the time is fixed.

Definition of Happen-before We say that an event e happens-before an event e', and write $e \rightarrow e'$, if one of the following three cases is true:

$$\exists p_i \in \prod : e = e_i^r,\ e' = e_i^s,\ r < s$$

$$e = send(m, *) \wedge e' = receive(m)$$

$$\exists e'' : e \rightarrow e'' \rightarrow e'$$