Distributed Systems Synchronization

Daniel Andresen

CIS520 – Operating Systems

Distributed Algorithms-- Assumptions:

1. Information on different machines is exchanged by PASSING MESSSAGES.
2. Decisions can only be based on local information.
3. A single machine failure should not cause the algorithm to fail.
4. No global clock exists.

Aside: Clock Synchronization

- *Physical Clocks*: A TIMER (clock) is a precisely machined quartz crystal.
- Each interrupt generated by a clock is called a *CLOCK TICK* - they occur at 60Hz or 100Hz.
- At each clock tick, the INTERRUPT SERVICE PROCEDURE adds 1 to the time stored in memory.
- The difference in the time stored between clocks is called *CLOCK SKEW*.

Physical Clock Synchronization:

How do we synchronize clocks:

1. *With real world clocks?*
   - UTC(Universal Coordinated Time)
2. *With each other?*
   - Recall, each machine has a timer that causes interrupts H times per second, and the clock value C is incremented on each interrupt.
     Let C = machine time and t = UTC timer.

     Ideally, dC/dt = 1.

Error Rate (aka drift):

Clocks with H = 60Hz generate 60^3 = 216,000 ticks/hour.

The maximum relative error is approximately 10^-5 (for a quartz crystal).

==> |actual - expected| / expected <= 10^-5.
==> |actual - expected| <= (10^-5)*(expected)
So |actual - expected| <= (10^-5)*(216,000) = 2.16 ~ 2
==> 215,998 <= actual <= 216,002.

Cristian's Algorithm:

Let

p = rho = MAXIMUM DRIFT RATE = 10^-5.
==> (1 - p) <= dC/dt <= (1 + p).
After Δt seconds, two clocks may be as much as 2 * p * Δt seconds apart.
Suppose that the clocks should differ by a most d seconds.
==> 2 * p * Δt < d ==> Δt < d/(2 * p).
==> After d/(2 * p) seconds it is time to resynchronize.

The Algorithm:

1. After at most d/(2 * p) seconds, each machine asks the time server for the current time C(UTC).
2. The sender estimates the propagation delay:
   PDT = propagation delay time ~ (t1 - t0 - I)/2.
   where I = interrupt time.
3. Finally, the sender sets it's time

   C = C(UTC) + PDT.

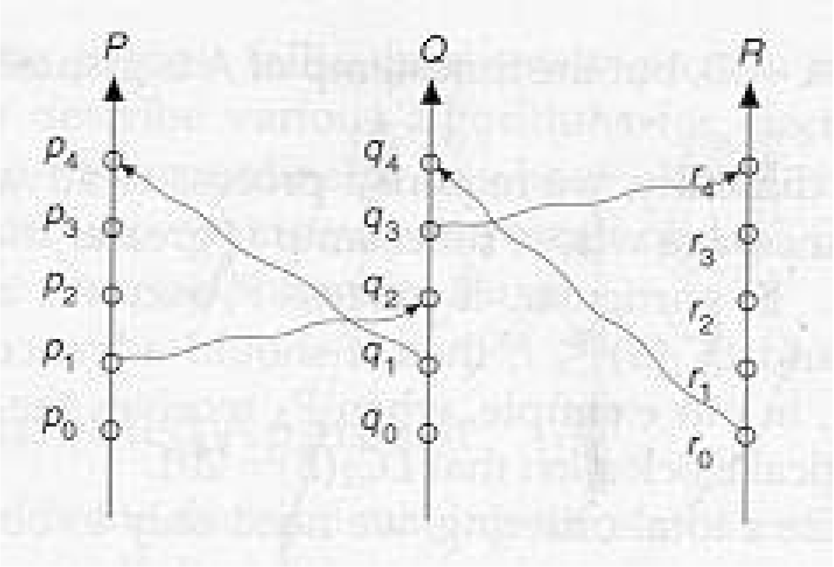
The idea is to take several measurements of PDT to derive a good estimate of propagation delay.

Berkeley Algorithm: The time server polls every machine periodically to ask what time each machines has. Based on the answers, the server broadcasts a message to have the machines set their times to the calculated average.

Logical Clocks

For each process Pi, we define a 'logical clock' Ci to be a function that assigns a number to each event in process Pi; that is, Ci : Ei -> N, where

Ei = events in Pi.

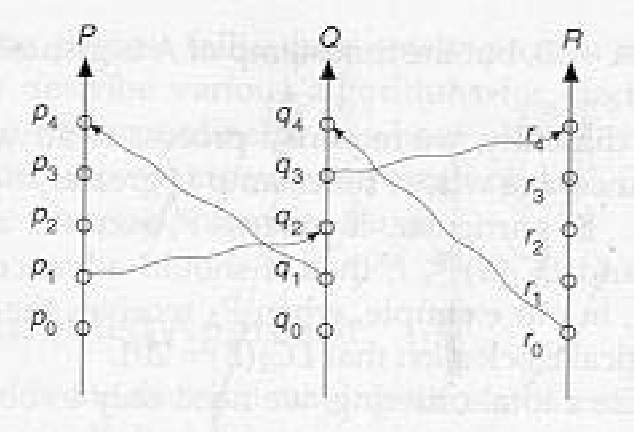


Lamport Clocks

- Lamport showed that logical clock synchronization is possible in a distributed system [1978]. He defined a transitive relation, called 'happened-before', denoted ----->, on events by: a----->b if
  a. a and b are events in the same process and a occurs before b.
  b. a is the event of sending a message and b is the event of receiving the message.
- Events that are not related (a --/--> b and b --/--> a) are called *'concurrent events'*.
- Let C(a) = logical time of event a. We want:

  a --> b => C(a) < C(b)

  This is called the clock condition.



- Proposition: a -----> b => C(a) < C(b).

A short outline of the proof:

Let a and b denote events. Suppose that a -----> b. We proceed by induction on the length of the causal path from a to b.

Basis Step (Length 1): Suppose that a happens immediately before b. That is, there is no event c, such that a -----> c and c -----> b. Then, either events a and b are on the same process and a happens immediately before b, or a is a send event and b is the corresponding receive event. In either case, C(b) will be at lease C(a) + 1 from the definition of the clock shown above.
Therefore C(a) < C(b).

Inductive Hypothesis: a -----> b and the length of the shortest causal path from a to b is k => C(a) < C(b).

Inductive Step: Suppose that the Inductive Hypothesis holds for k = N. We will show that it also holds for k = N+1. Suppose that a -----> b and the length of the shortest causal path from a to b is N+1. Let
A -----> x_1 -----> ... -----> x_N -----> b denote such a path. Then a happens immediately before x_1.
By the argument presented in the Base Step, C(a) < C(x_1). By the Inductive Hypothesis, C(x_1) < C(b) (Note: the path from x_1 to b is length N). Therefore, C(a) < C(b) by the ordinary transitive relation on the natural numbers.
Therefore, the hypothesis must be true for all N, and this proves our claim. QED

Total ordering -- By appending a unique process id, we can define a total ordering on all events in the system (and the total ordering is consistent with the -----> relation e.g. a -----> b ==> TO(a) < TO(b))
Let TOp(a) = (Cp(a), pid(p)) and define TOp1(a) < TOp2(b) if

1. Cp1(a) < Cp2(b) or
2. Cp1(a) == Cp2(b) and pid(p1) < pid(p2).