MONASH University

FIT3143 Tutorial Week 7

Lecturers: ABM Russel (MU Australia) and Vishnu Monn (MU Malaysia)

SYNCHRONISATION

OBJECTIVES

- The purpose of this tutorial is to introduce the concept of Synchronisation, Mutex, Deadlock
- Understand the concept of Averaging algorithm
- Understand the concept of Chandy-Misra-Haas algorithm
- Describe Berkeley's synchronization algorithm and compare it with Cristian's algorithm.

Note: Tutorials are not assessed. Nevertheless, please attempt the questions to improve your unit comprehension in preparation for the labs, assignments, and final assessments.

QUESTIONS

1. Discuss the difference between logical clock and real clock synchronization.

   A sensible discussion that includes points around, synchronisation, causation and ordering. The clocks that agree among certain computers but not necessarily with the real clock are logical clocks. Clocks that agree on time and within a certain time limit, are physical clocks.

2. Outline when and why you might need to use a centralised or distributed MUTEX algorithm.

   When multiple processes need to access shared resources in a distributed environment. MUTEX ensures there are no collisions between processes for a given resource.

3. Discuss Averaging algorithm

   "Averaging algorithm" is a decentralized algorithm. This algorithm divides time into resynchronization intervals with a fixed length R. Every machine broadcasts the current time at the beginning of each interval according to its clock. A machine

collects all other broadcasts for a certain interval and sets the local clock by the average of their arrival times.

4. Discuss the relationship between the Chandy-Misra-Haas algorithm and the wait for graph in the context of deadlock identification.

   CMH essentially passes a message along the “wait-for-graph.” If a dead-lock exists, that is, if a cycle exists in the wait-for-graph then the message will return to whence it started.

5. Describe Berkeley’s algorithm by write its pseudo-code. Compare this algorithm with Cristian’s algorithm.
   Solution from: https://www.quora.com/How-do-I-implement-a-Berkeley-clock-synchronization-algorithm-in-C++

Hereinafter, all UDP network messages are represented by $\Pi_{<tag>}[< contents >]$, where $< tag >$ is one of:

- req — “here’s my current time, what’s yours?”
- $resp$ — “I see your current time, and I’ll tack on my own”
- adj — “here’s your time adjustment going forward”

Step 0: Choose a leader from among the n nodes in your cluster. The specific leader election protocol depends greatly on your circumstances, so I’ll just point you to this overview for some options.

Step 1: For each “client” $x = 1..(n-1)$, the leader will:

- Get its current time, T_{M_x}
- Send C_x its current time: $\Pi_{req}[T_{M_x}]$

Step 2: When each client receives $\Pi_{req}[T_{M_x}]$, it will:

- Get its current time, T_{C_x}
- Send M both the time received from M, and its current time: $\Pi_{resp}[T_{M_x}, T_{C_x}]$

Step 3: When the leader receives $\Pi_{resp}[T_{M_x}, T_{C_x}]$, it will:

- Get its current time, T_{R_x}
- Calculate C_x's one-way communications delay: $\delta_{C_x} = \frac{T_{R_x} - T_{M_x}}{2}$
- Calculate C_x's time offset: $\Delta_{C_x} = T_{C_x} - T_{M_x} - \delta_{C_x}$

Step 4: Once it has accumulated Δ_{C_x} from all $n-1$ clients, it then calculates the average time offset for the entire cluster:

$$\Delta_\tau = \frac{\sum_{n-1}^{x=1} \Delta_{Cx}}{n}$$

(Note that we’re summing $n-1$ terms but dividing by n, because the leader’s own offset Δ_M is, by definition, 0.)

Step 5: The leader will then:

- Record its own time adjustment: $\Phi_M = -\Delta_\tau$
- Calculate each client’s time offset: $\Phi_{C_x} = \Delta_{C_x} - \Delta_\tau$
- Send C_x its time adjustment: $\Pi_{adj}[\Phi_{C_x}]$

From this point forward, all nodes will add their Φ_n to every time measurement to get *Coordinated Cluster Time* (CCT).