CLOCK SYNCHRONIZATION IN DISTRIBUTED SYSTEMS

Published Article: https://www.irjet.net/archives/V9/i3/IRJET-V9I3350.pdf

Presentation by:

Amey Thakur & Mega Satish

AGENDA

1. What is a Distributed Systems?
2. What is Clock Synchronization?
3. Clock Synchronization Algorithms
4. Conclusion
5. References

DISTRIBUTED SYSTEMS & ITS TYPES

Distributed System (DS) is a collection of computers connected via a high-speed communication network.

Types of Distributed System

1. Homogeneous Distributed Systems:
   - It is a distributed system such that all nodes have identical hardware, the same type of architecture, and operating system.

2. Heterogeneous Distributed Systems:
   - It is a distributed system such that each node has its own operating system and machine architecture.

NEED TO RESYNCHRONIZE THE CLOCK

- Need for proper allocation of available resources to preserve the state of resources and coordination between processes.
- Clock synchronization is critical for resolving these problem and can be implemented by using the physical clock and logical clock.
- Synchronizing clocks helps us
  - ➔ Time-stamping events (provides ‘Fairness’)
  - ➔ Ordering events (provides ‘Correctness’)

CLOCK SYNCHRONIZATION

- In a centralized system, time is unambiguous.
  - ➔ One system clock that keeps time, all others nodes follow this time.

- In a decentralized system, each node has its own time.
  - ➔ Problem: an event that occured after another event may not be assigned because of the lack of synchronization of time among the different nodes.

CLOCK SYNCHRONIZATION

- We need to measure time accurately:
  - ➔ To know the time an event occurred at a computer.
  - ➔ To do this we need to synchronize its clock with an authoritative external clock.
- Algorithms for clock synchronization useful for:
  - ➔ Concurrency control based on timestamp ordering.
  - ➔ Authenticity of requests e.g. in Kerberos.
- There is no global clock in a distributed system.
- Logical time is an alternative:
  - ➔ It gives ordering of events - also useful for consistency of replicated data

TYPES OF CLOCKS IN SYNCHRONIZATION

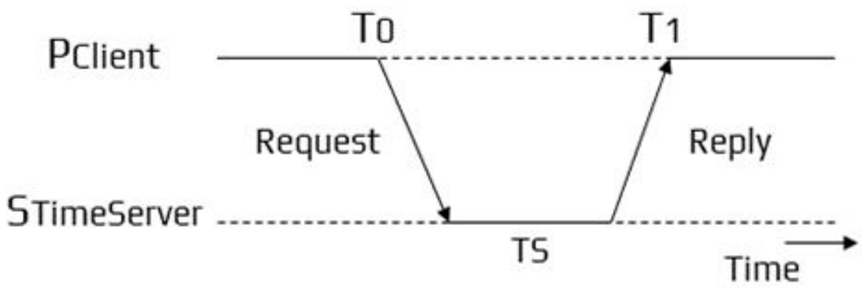
- **Physical clock -** refers to the time based on UTC, which is used as a reference time clock.
  - ➔ There are two aspects:
    - Obtaining an accurate value for physical time.
    - Synchronizing the concept of physical time throughout the distributed system.

- **Logical clock -** refers to the relative time and maintain logical consistency.
  - ➔ The essence of logical clocks is based on the happened-before relationship.

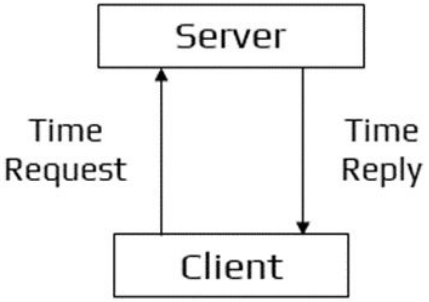
HAPPENED BEFORE RELATIONSHIP

- If two events, *a* and *b*, occurred at the same process, they occurred in the order of which they were observed, i.e., $a > b$.
- If a sends a message to *b*, then $a > b$. i.e., you cannot receive something before it is sent. This relationship holds regardless of where events *a* and *b* occur.
- The happen-before relationship is transitive.
- If *a* happens before *b* and *b* happens before *c*, then *a* happens before *c*. i.e., if $a > b$ and $b > c$, then $a > c$.

CRISTIAN'S ALGORITHM



$$T_{new} = T_{server} + T_1 - T_0 / 2$$

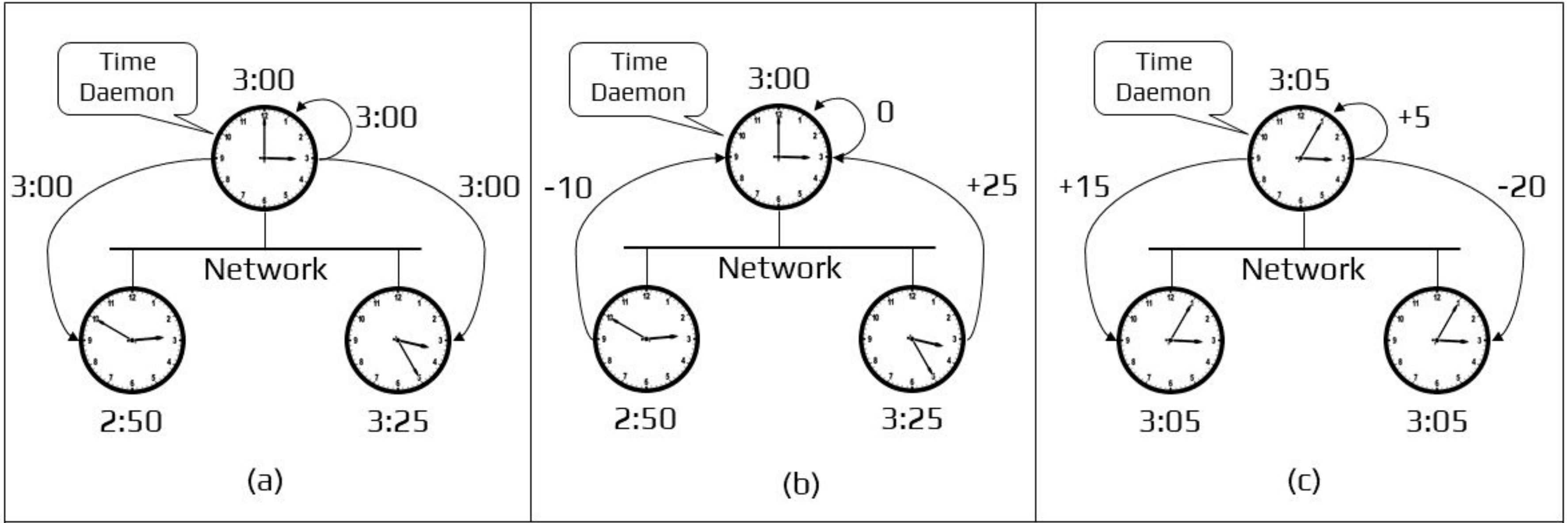


Here, the client approaches the server.

Algorithm:

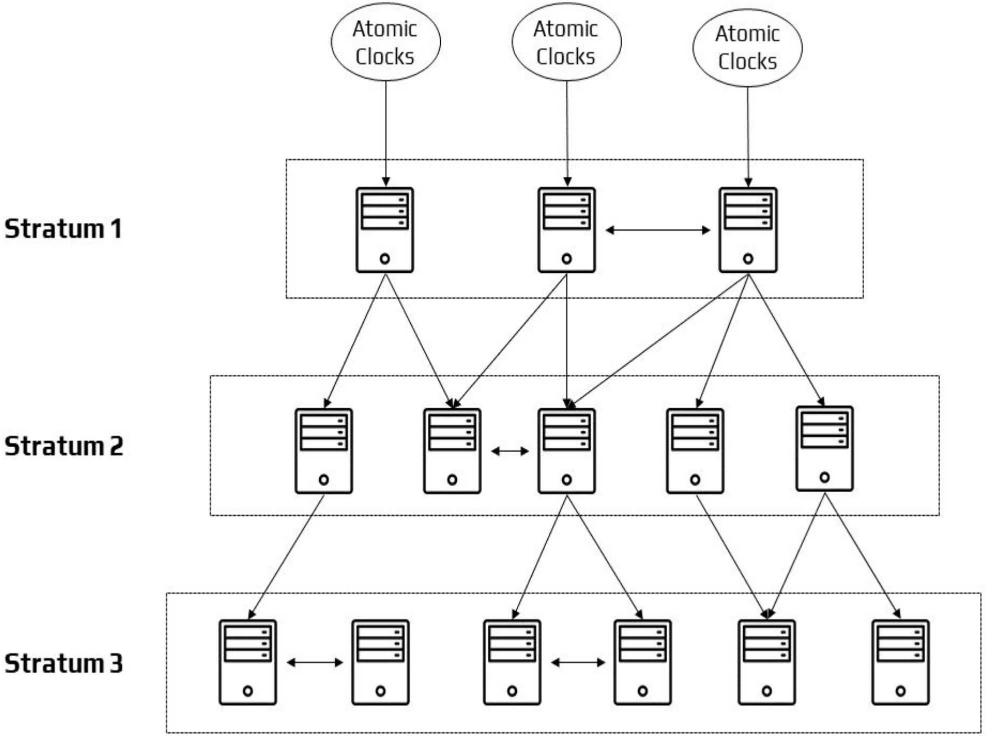
- Let S be the time server and Ts be its time.
- Process P requests the time from S.
- After receiving the request from P, S prepares a response and appends time Ts from its own clock and then sends it back to P.

BERKELEY'S ALGORITHM



(a) The time daemon asks all the other machines for their clock values.
(b) The machines answer.
(c) The time daemon tells everyone how to adjust their clock.

NETWORK TIME PROTOCOL



- Provides UTC synchronization service across Internet
- Uses time servers to synchronize networked processes.
- Time servers are connected by synchronized subnet tree.
- The root is adjusted directly.
- Each node synchronizes its children nodes.

LAMPORT'S CLOCK

- Lamport algorithm assigns logical timestamps to events.
- Each process has a counter (logical clock).
- Initially logical clock is set to 0.
- Process increments its counter when a *send* or a computation (*comp*) step is performed.
- Counter is assigned to event as its timestamp.
- Send(message) event carries its timestamp.
- On recv(message) event, the counter is updated by *max(local clock, message timestamp) + 1*

VECTOR CLOCK

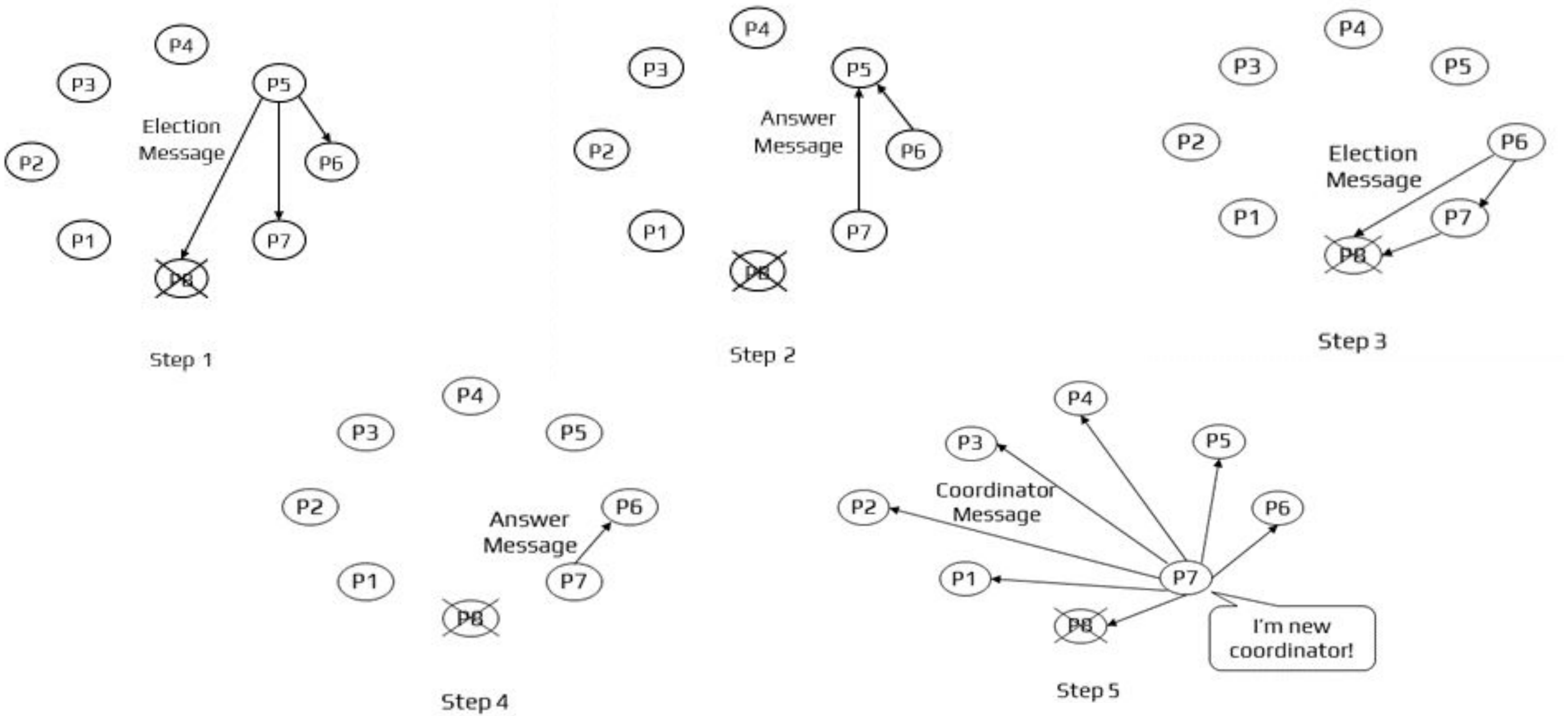
- One integer can't order event in more than one process.
- So, a Vector Clock (VC) is a vector of integers, one entry for each princess in the entire distributed system.
  - Label event *e* with VC(*e*) = [$C1, C2, \ldots, Cn$]
  - Each entry *Ck* is a count of events in process *k* that causally precede *e.*

ELECTION ALGORITHM

- Many distributed algorithms need one process to act as coordinator.
- It doesn't matter which process does the job, just need to pick one.
- Election algorithms: technique to pick a unique coordinator (leader election).
- Types of election algorithms: Bully and Ring algorithms.

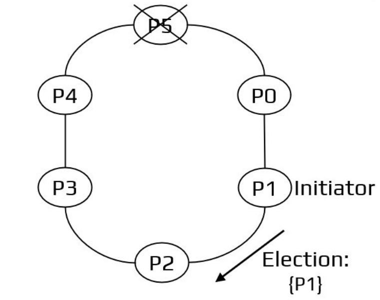
BULLY ALGORITHM

Bully algorithm requires one process to act as the coordinator. Suppose that there are 8 processes in the system, which are numbered from 1 to 8. Initially, process 8 was the coordinator. However, it has just crashed. Process 5 is the first one to notice this failure. The behaviour of the bully algorithm in this situation is illustrated below:

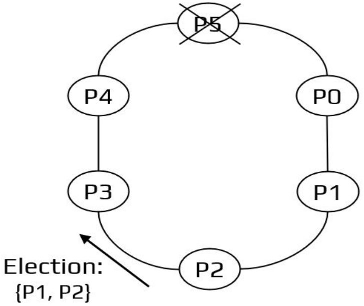


RING ALGORITHM

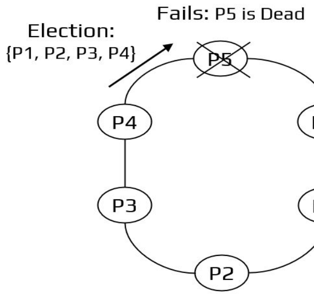
Ring algorithm requires one process to act as the coordinator. Suppose there are 6 processes in the system, which are numbered from 0 to 5. Initially, process 5 was the coordinator. However it has just crashed. Process 1 notices this and starts an election. The behavior of the ring algorithm in this situation is illustrated below:



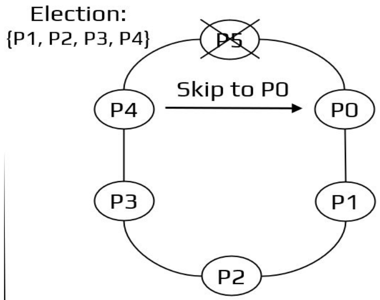
Step 1



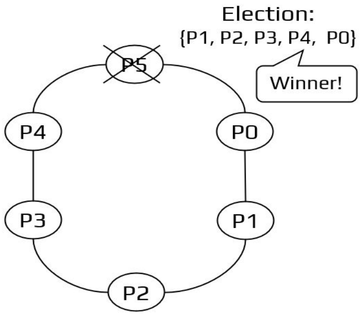
Step 2



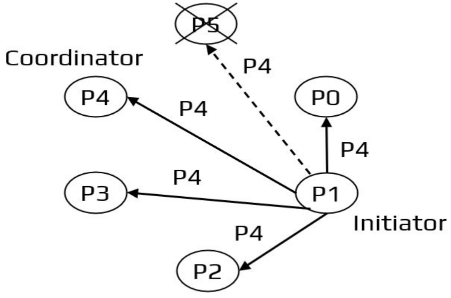
Step 3



Step 4



Step 5



Step 6

CONCLUSION

- In terms of algorithms, we can conclude that for clock synchronization, both centralized and distributed algorithms must account for the propagation time of messages among each node.
- The sequencing of processes and the preservation of resource status requires clock synchronization.
- When it comes to the concept of time in distributed systems, the most essential element is to get the events in the right sequence.
- Events can be positioned either in chronological order with Physical Clocks or in a logical order with Lamport's Logical Clocks and Vector Clocks along the execution timeline.

REFERENCES

[1] Latha, C. A., and H. L. Shashidhara. "Clock synchronization in distributed systems." In *2010 5th International Conference on Industrial and Information Systems*, pp. 475-480. IEEE, 2010.

[2] Horauer, Martin. "Clock synchronization in distributed systems." PhD diss., 2004.

[3] Sampath, Amritha, and C. Tripti. "Synchronization in distributed systems." In *Advances in Computing and Information Technology*, pp. 417-424. Springer, Berlin, Heidelberg, 2012.

[4] Biradar, Shripad, Santosh Durugkar, and Subhash Patil. "Handling Clock synchronization Anomalies in Distributed System."

[5] Simons, Barbara. "An overview of clock synchronization." *Fault-Tolerant Distributed Computing* (1990): 84-96.

[6] Welch, Jennifer Lundelius, and Nancy Lynch. "A new fault-tolerant algorithm for clock synchronization." *Information and computation* 77, no. 1 (1988): 1-36.

[7] Arghavani, A., E. Ahmadi, and A. T. Haghighat. "Improved bully election algorithm in distributed systems." In *ICIMU 2011: Proceedings of the 5th international Conference on Information Technology & Multimedia*, pp. 1-6. IEEE, 2011.

[8] Soundarabai, Paulsingh & Thriveni, J. & Manjunatha, H. & K R, Venugopal & Patnaik, Lalit. (2013). Message Efficient Ring Leader Election in Distributed Systems.

[9] Baldoni, Roberto, and Michel Raynal. "Fundamentals of distributed computing: A practical tour of vector clock systems." *IEEE Distributed Systems Online* 3, no. 2 (2002): 12.

THANK YOU