Communications in Distributed Systems

UNIT-3: Synchronization and Processes

- Clock Synchronization
- Clock Synchronization Algorithms
- Mutual Exclusion Algorithms
- Election Algorithms
- Atomic Transactions & Modeling
- Atomic Transaction Implementation
- Concurrency Control Algorithms in Atomic Transaction

Some critical challenges and principles of synchronization in distributed systems

1. Information Scattered Across Multiple Machines

- Decentralization: In a distributed system, instead of having one central computer do all the work, the tasks are spread across many computers. These computers must work together without needing to send all the information to a single spot, which would slow everything down and create a weak point where failure could cause big problems.
- Local Decision Making: Each computer makes decisions based on the information it has. These computers talk to each other to stay in sync, but there's no central boss telling them what to do. They need to work together smoothly, even though they're not always directly connected.
- .

2. Decisions Based on Local Information

- Autonomy: Each computer or process in the system makes its own decisions based on what it knows at the moment. This includes the current state of its tasks and any recent updates from other computers. It operates independently but stays aware of what others are doing.
- Consistency Models: To keep the system working as one, distributed systems use rules (called consistency models) that ensure all the computers are on the same page, even if they make decisions on their own. For example, "eventual consistency" means that while the system might be temporarily out of sync, it will eventually become consistent as all the updates spread throughout the network.

3.Avoiding Single Points of Failure:

- Fault Tolerance: Distributed systems are designed to tolerate failures by ensuring that the failure of one machine does not bring down the entire system. This is achieved through redundancy, replication, and robust failure recovery mechanisms.
- Failover Mechanisms: Techniques such as leader election and backup nodes ensure that if one node fails, another can take over its responsibilities.

4.Lack of Global Clock or Precise Time Source:

- Logical Clocks: Since it's tough to have all computers in sync with the same clock, distributed systems use logical clocks, like Lamport timestamps. These help keep track of the order of events and ensure consistency, even without exact time synchronization.
- Vector Clocks: For more precise tracking of the order of events, vector clocks are used. They allow the system to better understand the sequence and relationships between events across different computers.

Centralized Resource Allocation: Problems

1. **Heavy Burden on a Single Process:**

- Scenario: In a centralized system, one main process handles all requests for resources, like deciding who gets access to certain devices.

- Problem: If too many requests come in, this single process gets overwhelmed, slowing everything down. In a large system, this creates a bottleneck, making it inefficient as the system grows.

2.Single Point of Failure:

- Scenario: If the central manager process fails, the whole system can stop working because all resource decisions depend on it. For example, if the manager crashes, other processes can't get what they need, causing everything to freeze or fail.

- Problem: This makes the system unreliable. A failure in one part (the manager) can bring down the entire system, which goes against the goal of having a reliable and resilient distributed system that can keep running even when some parts fail.

Challenges in a Distributed System:

- **Inconsistent Timestamps**: In a distributed environment where multiple machines may be involved, each machine could have its own local clock. If the clocks are not synchronized perfectly, the timestamps on files could be inconsistent or misleading. For example, one machine might think a file was modified at a certain time, while another might see a different timestamp due to clock discrepancies.

- **Inconsistent Build Results**: If make were running across a distributed system, inconsistent timestamps could lead to incorrect build decisions, such as not recompiling a file that should be updated or unnecessarily recompiling files that haven't changed.

Logical Clocks

Introduction to Computer Timers

Computer Timer Basics: Every computer has a special circuit called a timer. This timer includes a quartz crystal that vibrates at a specific rate, which helps the computer keep track of time.

Structure of a Computer Timer

How It Works: A computer timer has two main parts: a counter and a holding register. The crystal's vibrations reduce the counter's value. When the counter hits zero, it sends a signal called an interrupt, then resets using the holding register. This setup allows the computer to create regular time signals, like 60 times per second, known as clock ticks.

Initialization and Updating of the Software Clock

Starting the Clock: When a computer starts, the current date and time are entered and converted into clock ticks (tiny time units) since a specific starting point. The system clock then updates with each tick, keeping track of time.

Single vs. Multiple Clocks

One Clock vs. Many Clocks: On a single computer, having a slightly inaccurate clock isn’t a big deal because everything uses the same timer. However, when multiple computers have their own clocks, even tiny differences can cause them to drift apart, leading to issues with timing and consistency in programs that rely on accurate timestamps across different machines.

Logical Clocks vs. Physical Clocks

- **Logical clocks** are used in computer systems to keep track of the order of events rather than the exact real-world time. They help ensure that processes or operations in a distributed system are consistent with each other.
- **Physical clocks** are designed to keep track of the exact real-world time. They synchronize with standard time sources to ensure that the time shown by the clock matches the current time in the real world.

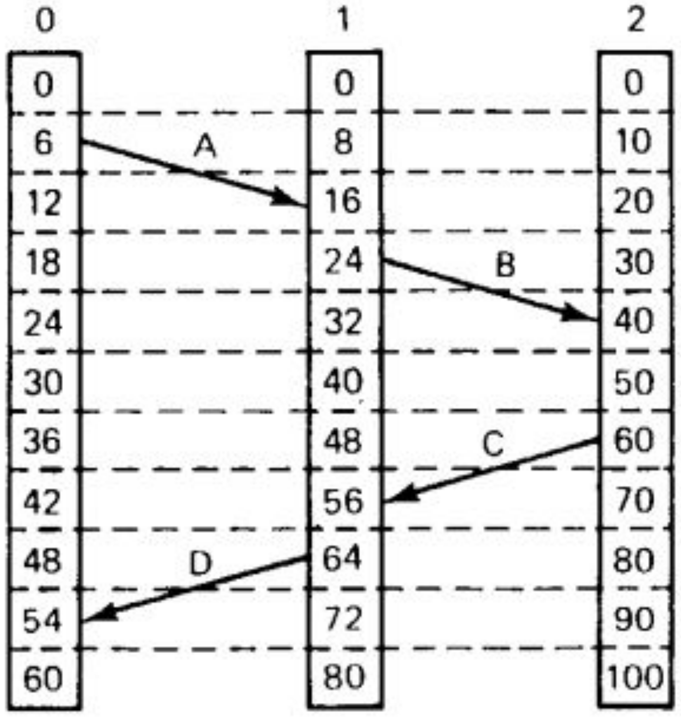
Lamport's Happens-Before Relation

To synchronize logical clocks, Lamport introduced the happens-before relation, denoted as "a happens before b" (a → b). This relation is observed in two scenarios:

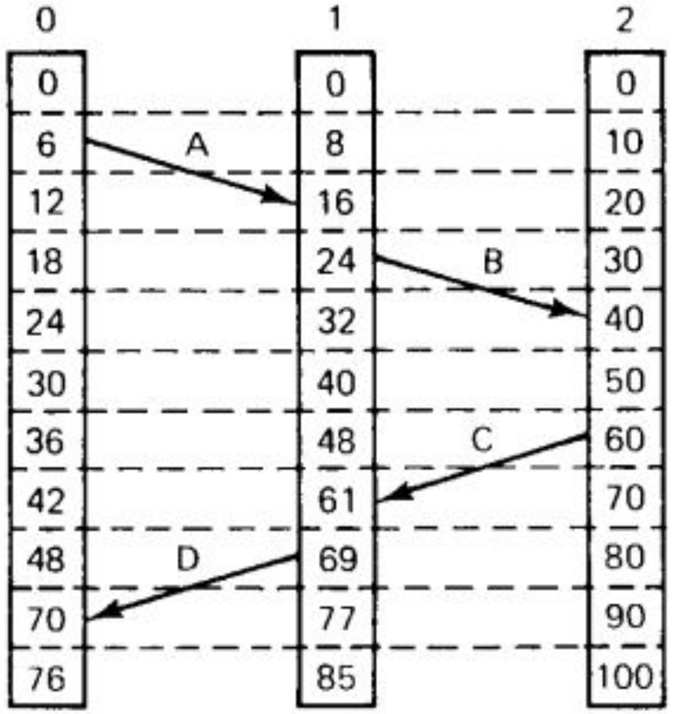
- If events a and b occur in the same process and a happens before b, then a → b.
- If event a is the sending of a message by one process and event b is its receipt by another process, then a → b since a message cannot be received before it is sent.

The happens-before relation is transitive, meaning if a → b and b → c, then a → c. If events x and y occur in different processes without exchanging messages, they are concurrent, meaning neither x → y nor y → x holds.

Lamport solution



a)clock run at different rate



b)Lamport algo correct the clocks

Way to assign times to events in a distributed system

- If a happens before b in the same process, then $C(a) < C(b)$.
- If a and b are the sending and receiving of a message, then $C(a) < C(b)$.
- All events a and b must have unique times, $C(a) \neq C(b)$.

Physical Clocks

Importance of Physical Clocks

While Lamport's algorithm provides an unambiguous ordering of events, the assigned time values do not necessarily reflect the actual occurrence times. In systems where real-time accuracy is critical, such as in real-time systems, external physical clocks are essential. Having multiple physical clocks enhances efficiency and redundancy, but it introduces two significant challenges:

- Synchronizing these clocks with real-world time.
- Synchronizing the clocks with each other.

Measurement of Time

Accurate time measurement is complex. Historically, time measurement was astronomical, based on the solar day—the interval between two consecutive transits of the sun, which reaches its highest point in the sky at noon each day. A solar second was defined as exactly 1/86400th of a solar day.

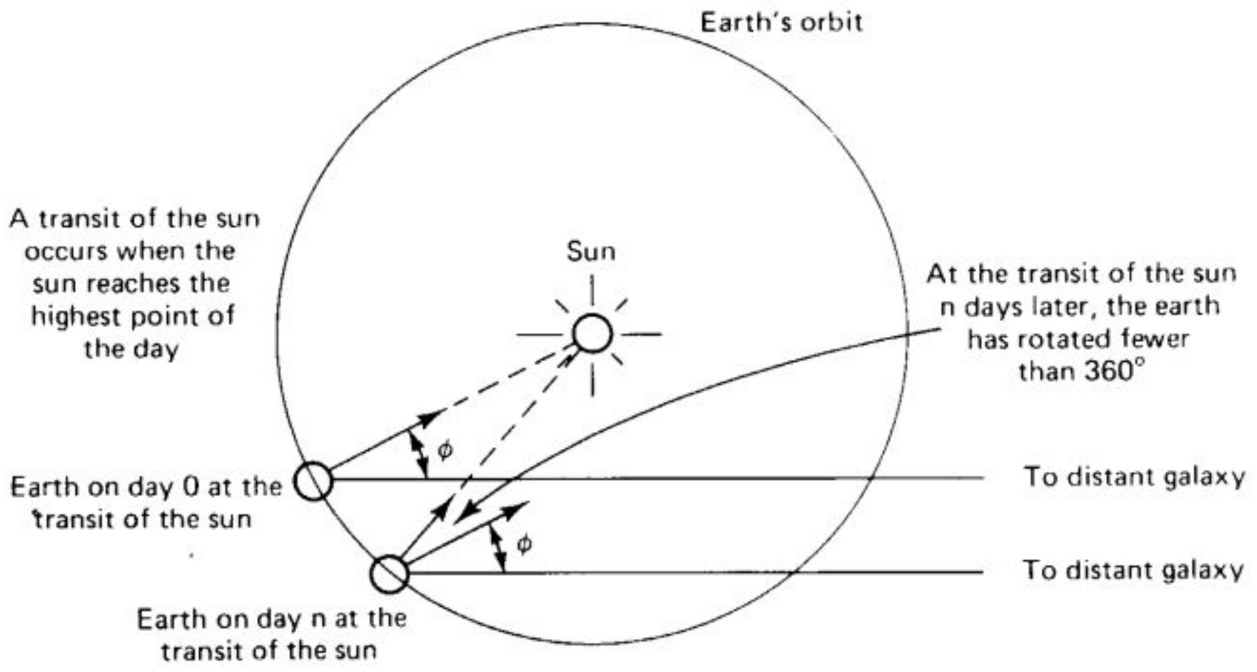
Variations in Earth's Rotation

In the 1940s, it was discovered that the Earth's rotation period is not constant due to tidal friction and atmospheric drag. Geological studies suggest that 300 million years ago, there were about 400 days in a year, indicating that the Earth's rotation has slowed down. Additionally, short-term variations in day length occur due to turbulence in the Earth's core.

Mean Solar Second and Atomic Clocks

Astronomers calculated the mean solar second by averaging the length of many days. With the invention of the atomic clock in 1948, timekeeping became more precise. The atomic second was defined based on the cesium 133 atom's transitions, making the atomic second equal to the mean solar second at the time of its introduction.

Computation of the mean solar day


Earth's orbit
A transit of the sun occurs when the sun reaches the highest point of the day
Sun
At the transit of the sun n days later, the earth has rotated fewer than 360°
ϕ
Earth on day 0 at the transit of the sun
To distant galaxy
ϕ
To distant galaxy
Earth on day n at the transit of the sun

International Atomic Time (TAI)

- Approximately 50 laboratories worldwide use cesium 133 clocks to measure time. These laboratories report their clock ticks to the Bureau International de l'Heure (BIH) in Paris, which averages them to produce International Atomic Time (TAI). TAI is the mean number of cesium 133 clock ticks since January 1, 1958, divided by 9,192,631,770.

Leap Seconds and Universal Coordinated Time (UTC)

- Due to the Earth's slowing rotation, 86,400 TAI seconds are now about 3 milliseconds shorter than a mean solar day. To prevent discrepancies, BIH introduces leap seconds when the difference between TAI and solar time reaches 800 milliseconds. This adjustment maintains synchronization with the sun's apparent motion, resulting in Universal Coordinated Time (UTC), the modern civil timekeeping standard.

Distribution and Accuracy of UTC

- UTC is disseminated through various means, including shortwave radio stations like WWV in Fort Collins, Colorado, and MSF in Rugby, Warwickshire. These broadcasts offer UTC with an accuracy of about ±1 millisecond, though atmospheric conditions can reduce practical accuracy to ±10 milliseconds. Satellites also provide UTC services with high accuracy, requiring precise knowledge of sender-receiver positions to account for signal propagation delays.

Q&A

Homework Questions

1) What is synchronization in distributed system?
2) What is clock synchronization?
3) Logical clock vs Physical clock.
4) Name at least three sources of delay that can be introduced between WWV broadcasting the time and the processors in a distributed system setting their internal clocks.
5) Consider the behavior of two machines in a distributed system. Both have clocks that are supposed to tick 1000 times per millisecond. One of them actually does, but the other ticks only 990 times per millisecond. If UTC updates come in once a minute, what is the maximum clock skew that will occur?

Thank You!