Clock Synchronization in Distributed Systems

What is a Distributed System?

- A distributed system is a collection of independent computers connected via a network.
- These systems communicate and coordinate through message passing.
- Each node can share resources like data, hardware, and software.
- **Challenge:** How to maintain consistency and order without a shared clock?

Why Synchronization is Needed

- In distributed systems, each node has its own local clock.
- Clocks can drift over time due to hardware differences

.

- **Without synchronization:**
  - Inconsistent timestamps
  - Incorrect event ordering
  - Resource conflicts
  - System failures in coordination

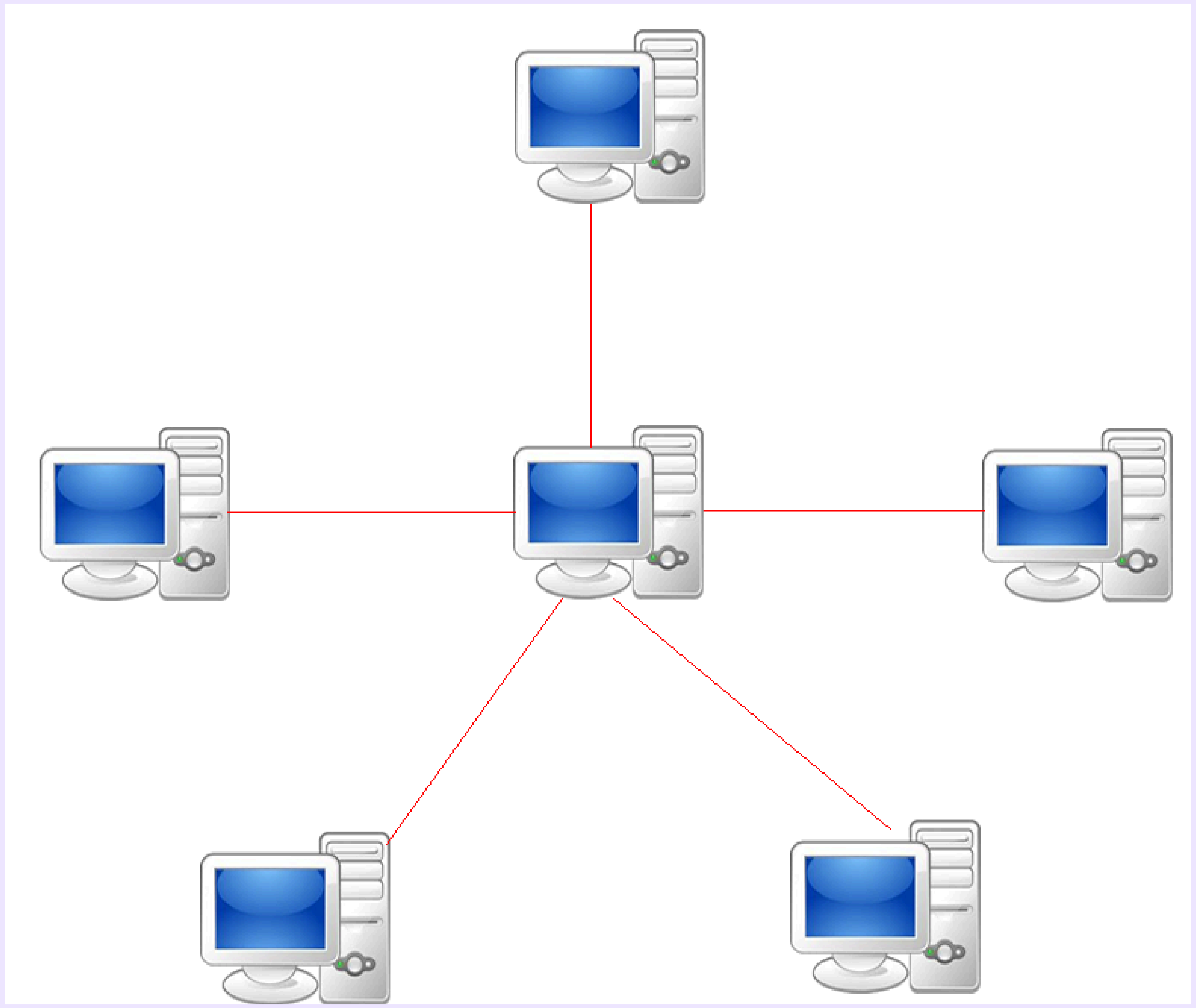
Types of Clock Synchronization

- **External Clock Synchronization**
  - Uses an external reference clock (e.g., UTC).
  - Nodes adjust their time to match this global reference.

- **Internal Clock Synchronization**
  - No external clock is used.
  - Nodes synchronize with each other by exchanging time messages.
  - All nodes adjust to an agreed average time.

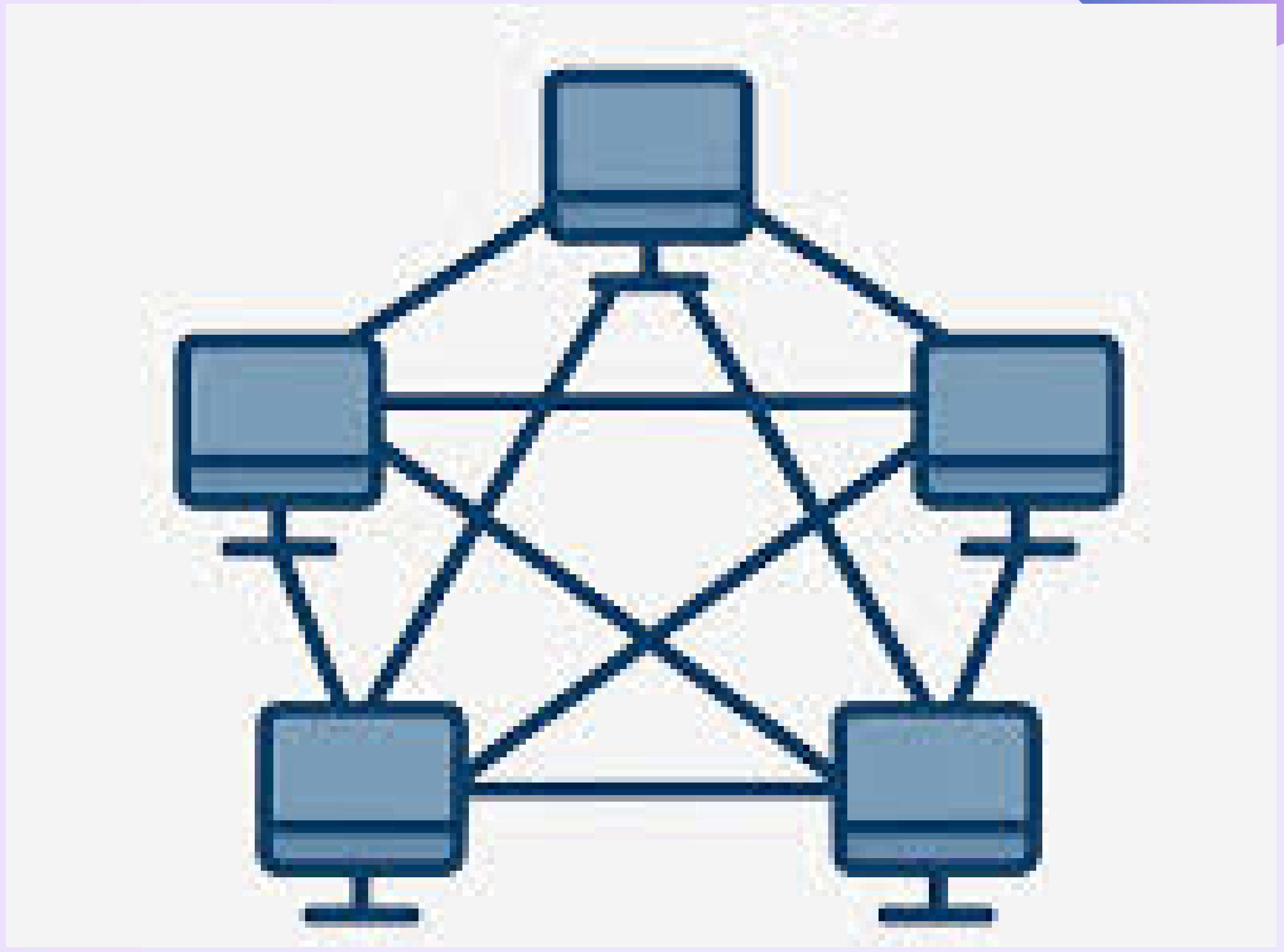
Centralized Clock Synchronization

- Uses a single time server as the reference.
- All other nodes request time from this server.
- **Examples:**
    - Berkeley Algorithm
    - Active/Passive Time Server
- **Advantage:**
  Simple to implement.
- **Disadvantage:**
  Single point of failure, not scalable



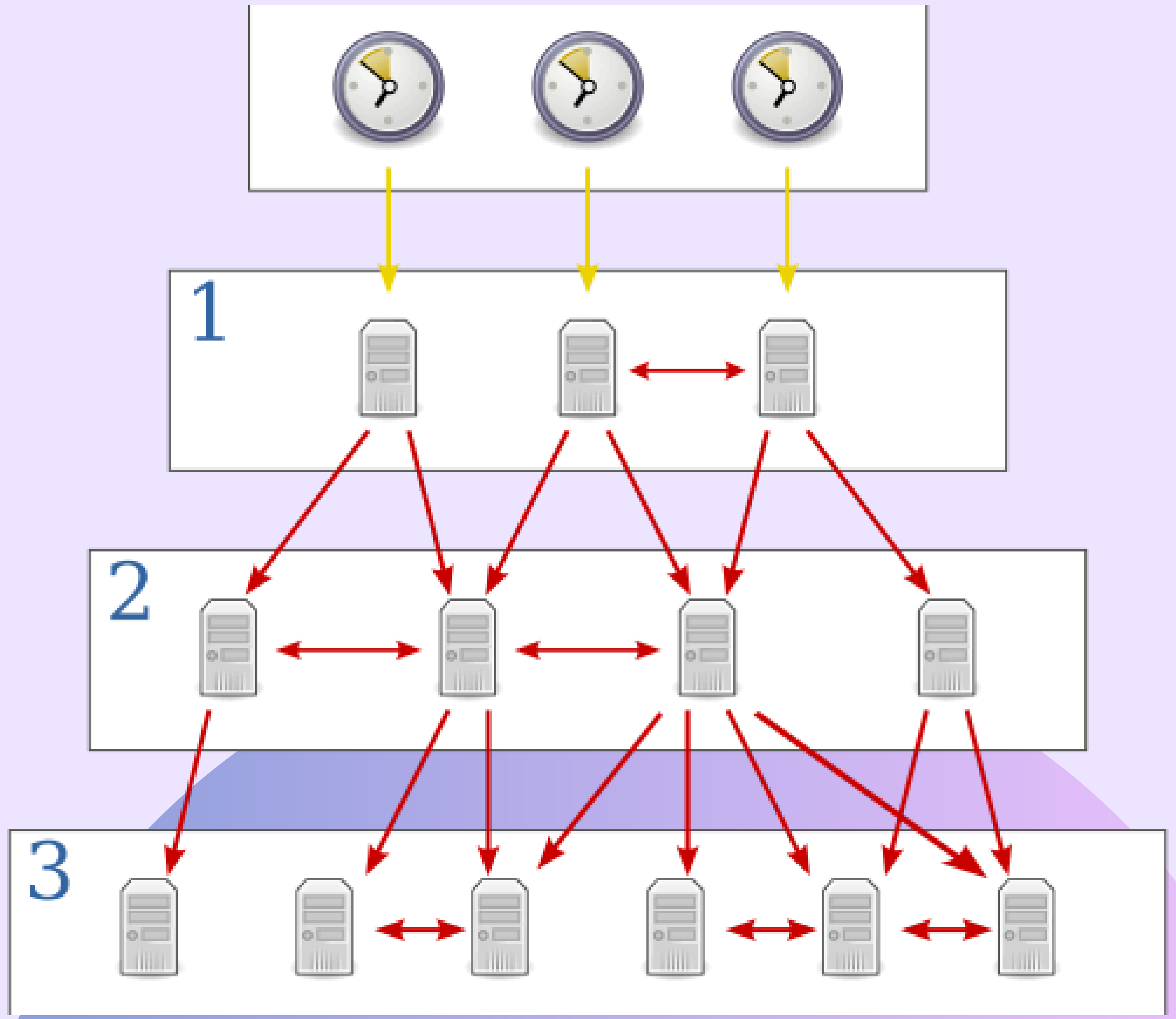
Distributed Clock Synchronization

- No central server; all nodes cooperate.
- Each node exchanges time with others and computes an average.
- **Examples**:
  - Network Time Protocol (NTP)
  - Global Averaging Algorithm
  - Localized Averaging Algorithm
- **Advantages**: Fault-tolerant, scalable, no single point of failure.



Network Time Protocol (NTP) in Detail

- Most widely used distributed time protocol.
- Uses a hierarchy of servers (stratum levels).
- Stratum 0 = atomic clock, Stratum 1 = primary servers, etc.
- Designed to handle network delays and packet loss.
- Provides accuracy within milliseconds over the internet.



Studio
Shodwe

Challenges in **Clock Synchronization**

- Network Delays: Variable latency affects time accuracy.
- Clock Drift: Physical clocks run at slightly different speeds.
- Fault Tolerance: Handling node failures during sync.
- Security: Preventing malicious time attacks (e.g., in blockchain).
- Scalability: Maintaining sync in large, dynamic networks.

Real-World Applications

- **Financial Systems:** Timestamping transactions.
- **Cloud Computing:** Synchronizing distributed databases.
- **Blockchain:** Consensus mechanisms rely on time sync.
- **Telecommunications:** Call logging and billing.
- **IoT Networks:** Coordinating sensor data collection.

Thank You.

Sandeep Gaire