# Clock Synchronization Algorithms

# Clock Synchronization

**Introduction:**

- Goal of Clock Synchronization:
  - **With WWV Receiver:** If one machine has a WWV receiver (a time signal receiver), the goal is to synchronize all other machines to it.
  - **Without WWV Receiver:** Each machine keeps its own time, and the goal is to synchronize all machines as closely as possible.

**Basic Model of the System:**

- **Software Clock:**
  - Every machine has a timer that interrupts a certain number of times per second (denoted as H).
  - Every time the timer interrupts, the machine's software clock (C) is updated. This clock tracks the number of ticks (interrupts) since a starting time.
  - **Ideal Scenario:** If the Universal Time Coordinated (UTC) time is t, then the clock of machine p should ideally be $C_p(t) = t$ (meaning the clock perfectly matches UTC).

**Real-World Timer Behavior:**

- **Timer Imperfection:**
  - In reality, timers don't interrupt exactly H times per second. For instance, if H = 60, the timer should generate 216,000 ticks in an hour.
  - Due to imperfections, the actual ticks may slightly vary (e.g., between 215,998 and 216,002 ticks per hour).
- **Maximum Drift Rate (p):**
  - The timer's deviation from the ideal number of ticks is determined by a constant p, known as the **maximum drift rate**.
  - If the clock ticks within a certain range (defined by p), it's considered to be working correctly.

## Clock Drift and Synchronization:

- Clock Drift:
  - If two clocks are drifting in opposite directions from UTC, after a certain time $\Delta t$, they could be as much as $2p\Delta t$ ticks apart.
- Resynchronization:
  - To ensure that no two clocks differ by more than a certain amount (denoted as $\delta$), the clocks need to be resynchronized in software at least every $\delta/2p$ seconds.

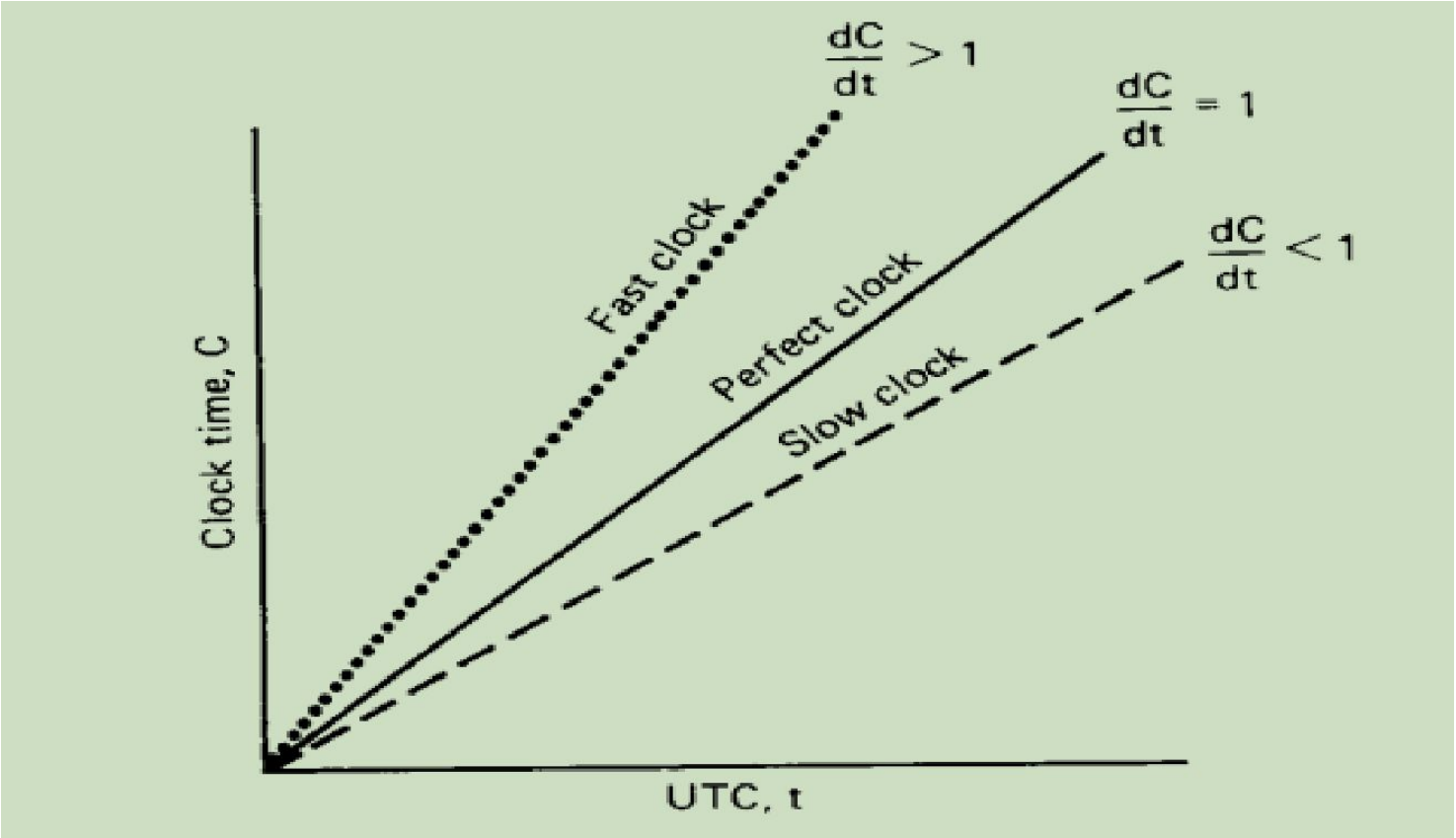



**Fig:**Not all clocks tick precisely at the correct rate

# Cristian's Algorithm

1. **Purpose of Cristian's Algorithm:**
    - Synchronizes machines in a network with a central time server (the machine with a WWV receiver).
2. **How It Works:**
    - **Request Time:** Each machine periodically sends a request to the time server asking for the current time.
    - **Receive Time:** The time server responds with its current time (C_UTC).
    - **Set Clock:** The requesting machine sets its clock to the time received, but with some adjustments.

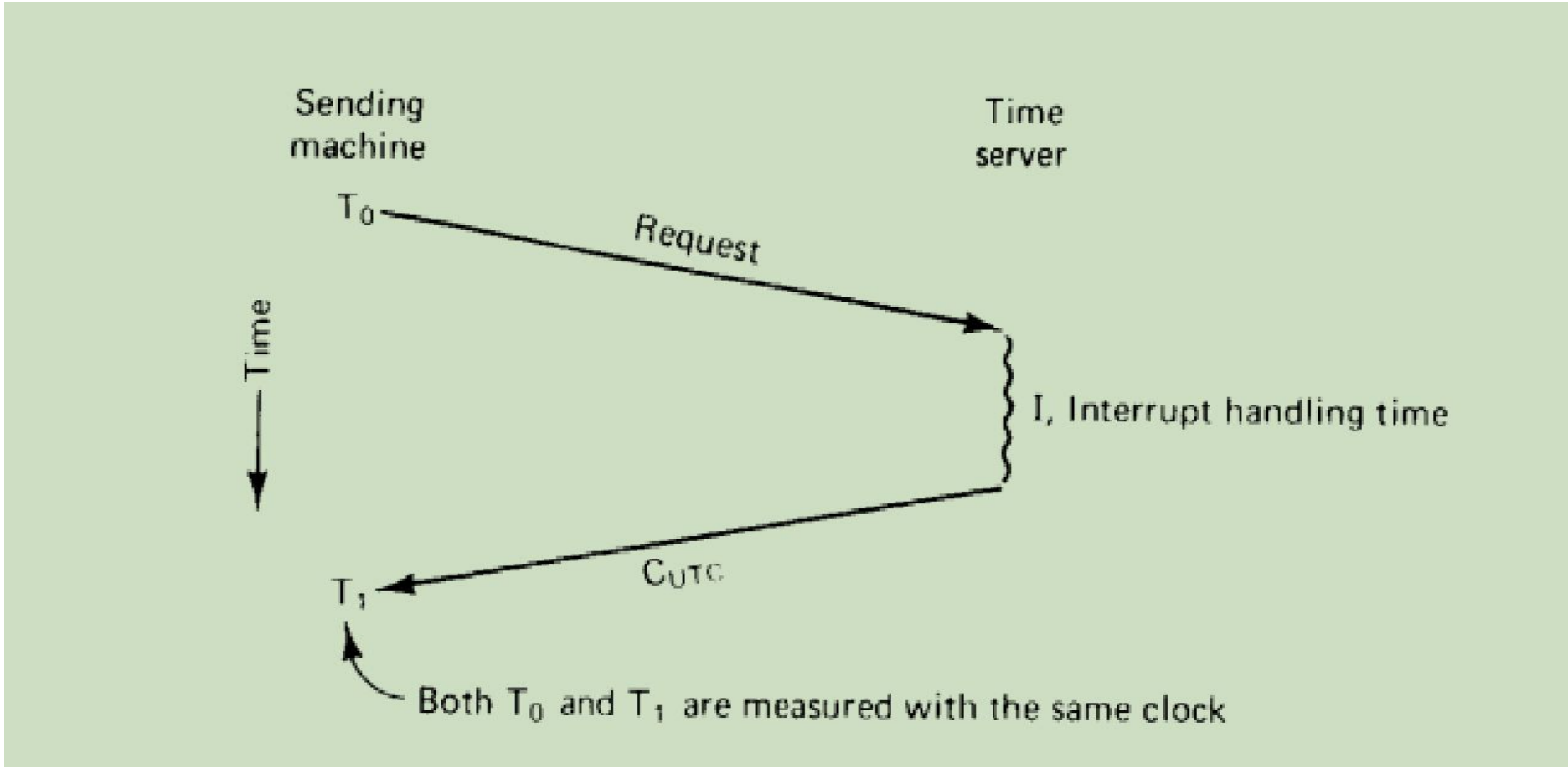



**Fig**:Getting the current time from the time server

3. **Key Challenges:**
    - **No Backward Time:**
        - If the requesting machine's clock is ahead of C_UTC, setting it back could cause problems (e.g., timestamps being out of order).
        - **Solution:** Adjust the clock gradually by slightly slowing down or speeding up the tick rate until it matches C_UTC.
- **Network Delay:**
    - The time taken for the time server's response to reach the requesting machine can vary due to network load.
    - **Estimate Delay:** Measure the round-trip time between sending the request ($T_0$) and receiving the reply ($T_1$).
    - **Best Estimate:** Assume the one-way delay is half of the round-trip time, then adjust the received time by adding this estimate.

4. **Improving Accuracy:**

   - **Multiple Measurements:** Take several measurements to account for network variability.
   - **Discard Outliers:** Ignore measurements with unusually high delays (due to network congestion).
   - **Use Best Measurement:** Average the valid measurements or use the fastest one (least network delay) for the most accurate time estimate.

Cristian's algorithm is effective in synchronizing clocks across a network by accounting for potential delays and ensuring time adjustments are smooth and reliable.

## 1. Cristian's Algorithm:

- **Passive Time Server:** In Cristian's algorithm, the time server is called "passive." This means the server doesn't actively seek out information; instead, it just responds to requests. So, when a computer wants to know the correct time, it asks the server, and the server replies with the time.

## 2. Berkeley Algorithm:

- **Active Time Server (Time Daemon):** In the Berkeley Algorithm, the time server is more involved, known as the "Time Daemon." It doesn't just wait for requests. Instead, it actively collects time data from all the computers in the network.

## 3. **No WWV Receiver:**

- **No Access to a Reference Time:** WWV is a radio station that broadcasts the official time. If no machine has a WWV receiver, it means none of the computers have access to the exact official time from an outside source.

## 4. **Polling Technique:**

- **Collecting Time from Computers:** The time server (Time Daemon) periodically asks all the computers in the network what time they think it is. This process of asking each computer is called "polling."

## 5. Determine the Average Time:

- **Calculating the Average:** After collecting the times from all computers, the Time Daemon calculates the average of these times. This average time represents what the network agrees to be the correct time.

## 6. Communicating the Difference:

- **Adjusting the Clocks:** Once the average time is determined, the Time Daemon doesn't directly set each computer's clock. Instead, it tells each computer how much they need to adjust their clocks. For example, it might say, "Your clock is 5 seconds fast; slow it down," or "Your clock is 3 seconds slow; speed it up."

## 7. **Advancing or Slowing Down the Clocks:**

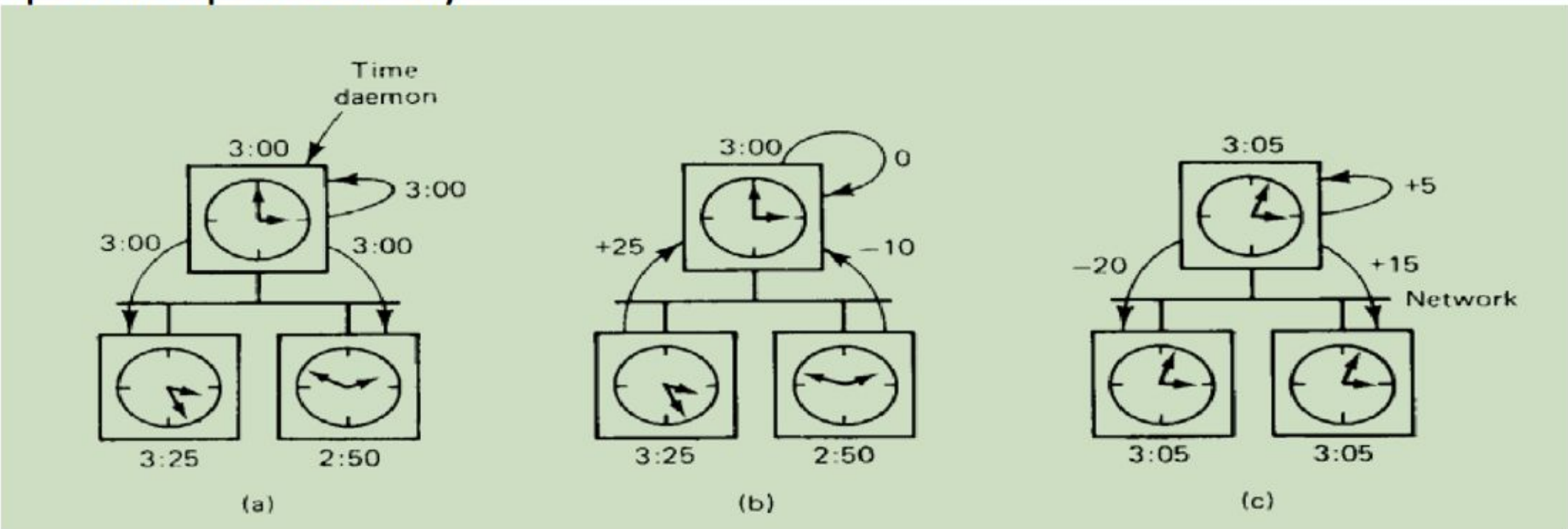
- **Synchronizing the Time:** The computers then make small adjustments to their clocks (either speeding up or slowing down) until they match the new, agreed-upon time.

In summary, Cristian's algorithm has a passive server that responds to time requests, while the Berkeley Algorithm has an active server (Time Daemon) that gathers time data from all computers, calculates an average time, and helps the computers sync up to this average time by adjusting their clocks.

# Berkeley Algorithm cont..

- The time daemon's time must be set mannually by the operator periodically.

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



# Averaging Algorithms

## 1. Decentralized Algorithm:

- **What Does "Decentralized" Mean?** In a decentralized system, there isn't just one master computer (or server) that decides the time for everyone. Instead, all computers work together to decide the correct time.

## 2. Resynchronization Intervals:

- **Fixed Length Intervals:** The computers agree to check and synchronize their clocks regularly after fixed intervals.

- **How It Works:**
    - The first interval starts at a specific time, let's call it $T_0$.
    - After that, each interval lasts for a fixed amount of time, denoted by $R$.
    - For example, if $T_0$ is 12:00 PM and $R$ is 10 minutes, the intervals start at 12:00 PM, 12:10 PM, 12:20 PM, and so on.

## 3. Broadcasting Time:

- **Sharing the Current Time:** At the start of each interval, every computer sends out a message to the others, broadcasting what time it thinks it is according to its clock.
- **Different Speeds:** Since different clocks run at slightly different speeds, each computer might report a slightly different time.

## 4. Collecting Time Broadcasts:

- **Waiting and Collecting:** After broadcasting its own time, each computer starts a timer and waits for messages from all the other computers. It collects these messages during a small time window $S$.

## 5. Computing the New Time:

- **Averaging the Times:** Once all the time messages are collected, the computer calculates the average of these times. This average is then used as the new time for the next interval.