Twitter: Case Study

Systems Analysis & Design

Learning Objectives

By the end of this session, you will have acquired the following information:

- Unique ID Generator
- Network Time Protocol (NTP)

Unique ID Generator

- Twitter needed to generate tens of thousands of IDs per second in a highly available manner.
- These IDs needed to be roughly sortable.
  - If tweets A and B were posted around the same time, they should have IDs in close proximity to one another.
- Twitter announced Snowflake in 2010.

Snowflake

- **Sign bit: 1 bit.** It will always be set to 0. This bit is reserved for future use.
- **Timestamp: 41 bits.** This represents the number of milliseconds since the epoch.
    - Twitter uses a default epoch of 1288834974657, which is equivalent to Nov 04, 2010, 01:42:54 UTC.
- **Datacenter ID: 5 bits.**
- **Machine ID: 5 bits.**
- **Sequence number: 12 bits.** This number is incremented by 1 for every ID generated on that machine/process.
    - The number is reset to 0 every millisecond.

1 bit	41 bits	5 bits	5 bits	12 bits
0	timestamp	datacenter ID	machine ID	sequence number

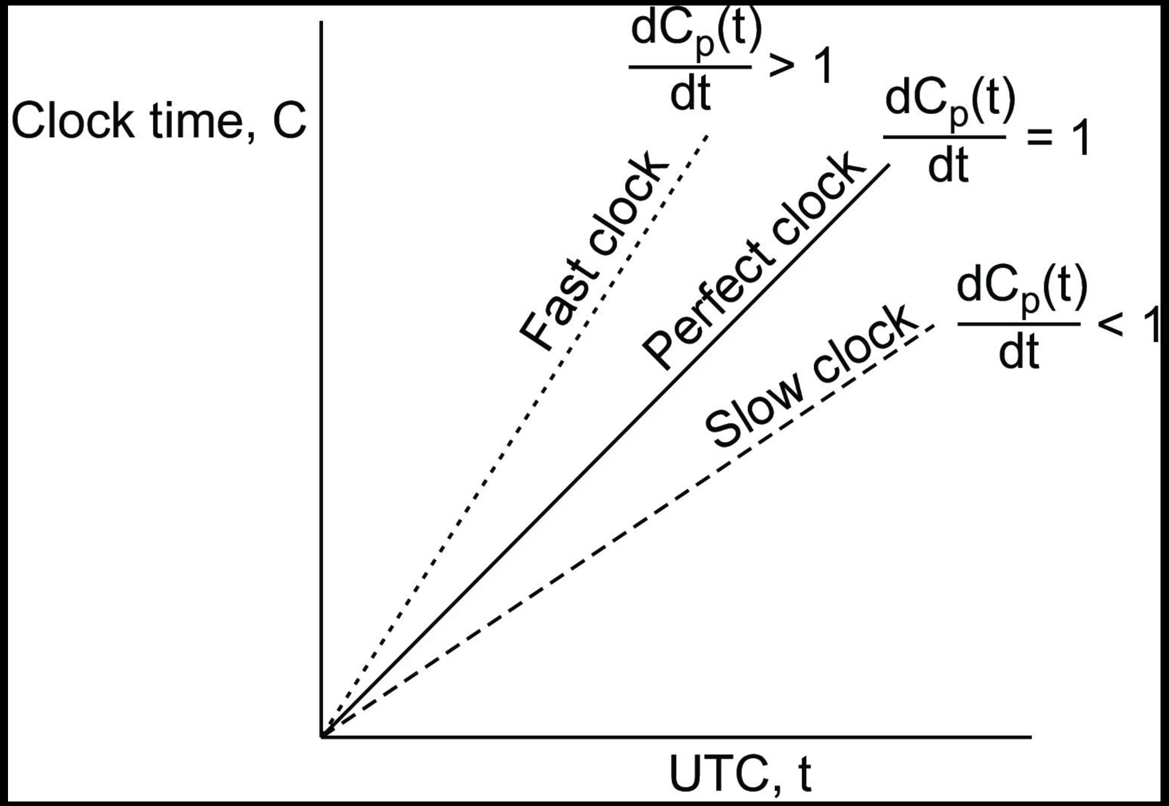
Clock Synchronization

- We assume that ID generation servers have synchronized clocks.
- The Network Time Protocol (NTP) is the most commonly used solution to this problem.

- Problem: Sometimes, we need the exact time, not just an ordering.
- Solution: Universal Coordinated Time (UTC).
  - based on the number of transitions per second of the cesium 133 atom, which is highly accurate.
  - Currently, real time is determined as the average of approximately 50 cesium clocks worldwide.
  - introduces a leap second occasionally to compensate for the gradual lengthening of days.
  - broadcast through short-wave radio and satellites. Satellites can provide an accuracy of about ±0.5 ms.

- A clock is specified with its maximum clock drift rate, denoted by ρ.
- $F(t)$ denotes oscillator frequency of the hardware clock at time t
- F is the clock's ideal (constant) frequency $\Rightarrow$ living up to specifications:

$$\forall t : (1 - \rho) \leq F(t)/F \leq (1 + \rho)$$



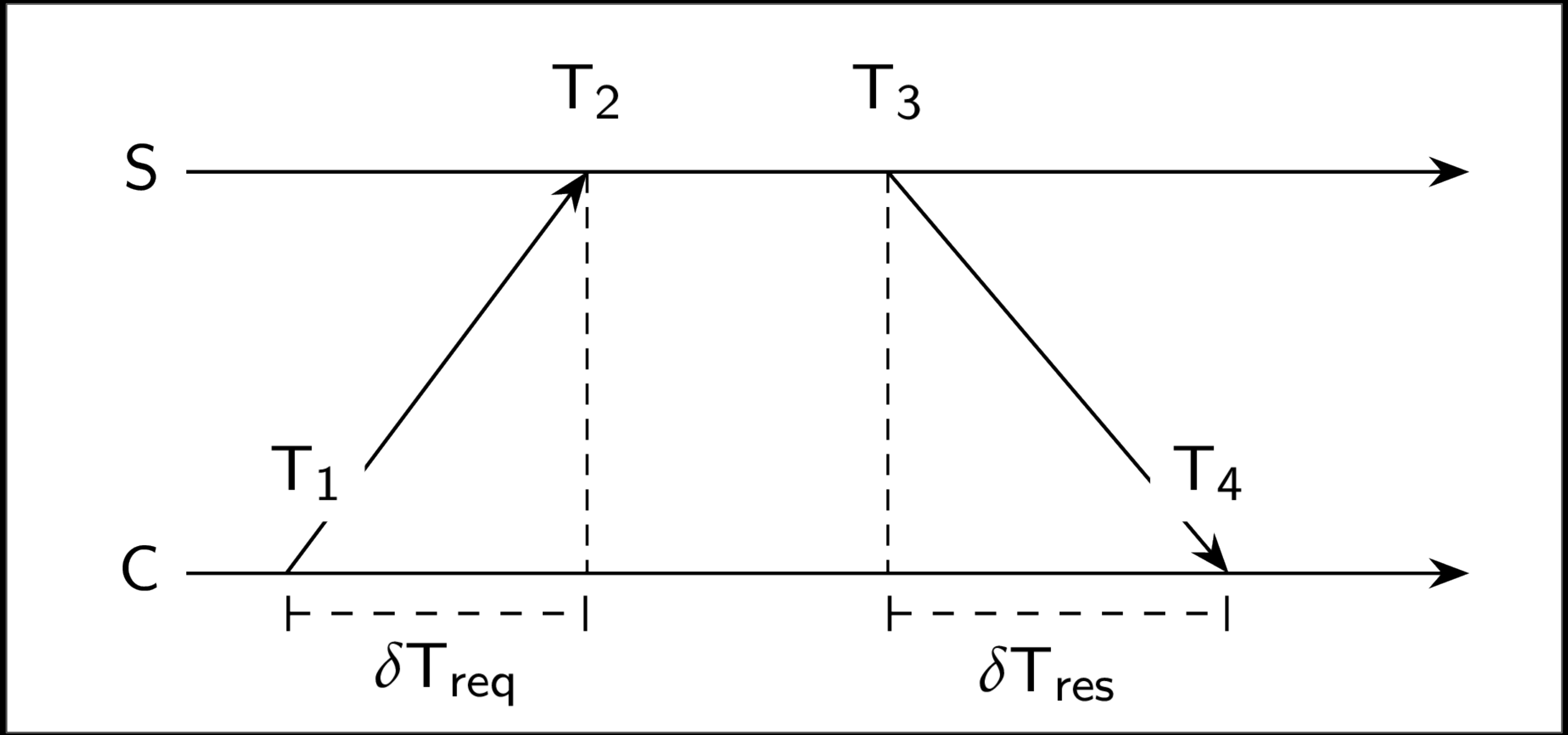
- By using hardware interrupts, we couple a software clock to the hardware clock, thereby also coupling its clock drift rate.

$$C_p(t) = \frac{1}{F}\int_0^t F(t)dt \Rightarrow \frac{dC_p(t)}{dt} = \frac{F(t)}{F}$$

$$\Rightarrow \forall t : 1 - \rho \leq \frac{dC_p(t)}{dt} \leq 1 + \rho$$

- **Precision**
  - The goal is to keep the deviation between two clocks on any two machines within a specified bound, known as the precision π: $$\forall t, \forall p, q : |Cp(t) - Cq(t)| \leq \pi$$
  - with Cp(t) the computed clock time of machine p at UTC time t .
- **Accuracy**
  $$\forall t, \forall p : |Cp(t) - t| \leq \alpha$$
  - In the case of accuracy, we aim to keep the clock bound to a value α:
- **Synchronization**
  - Internal synchronization: keep clocks precise
  - External synchronization: keep clocks accurate

Detecting and adjusting incorrect times



Computing the relative offset θ and delay δ

Assumption: $\delta T_{req} = T_2 - T_1 \approx T_4 - T_3 = \delta T_{res}$

$$\theta = T_3 + ((T_2 - T_1) + (T_4 - T_3))/2 - T_4 = ((T_2 - T_1) + (T_3 - T_4))/2$$

$$\delta = ((T_4 - T_1) - (T_3 - T_2))/2$$

Network Time Protocol

Collect (θ, δ) pairs. Choose θ for which associated delay δ was minimal.

Chrony

```
# Controller Node
# /etc/chrony/chrony.conf
server NTP_SERVER iburst
# Replace NTP_SERVER with the hostname or IP address of
# a suitable more accurate (lower stratum) NTP server.
allow 10.0.0.0/24

# Other Nodes
# /etc/chrony/chrony.conf
server controller iburst
```

Further Resources

- System Design Interview — An Insider's Guide (pages: 110 - 118)
- Clock Synchronization