IT4883: Distributed Systems

Spring 2020

Synchronization - 1

School of Information and Communication Technology
Hanoi University of Science and Technology

Synchronization

Until now, we have looked at:

- how entities communicate with each other

In addition to the above requirements, entities in DS often have to *cooperate* and *synchronize* to solve the problem correctly

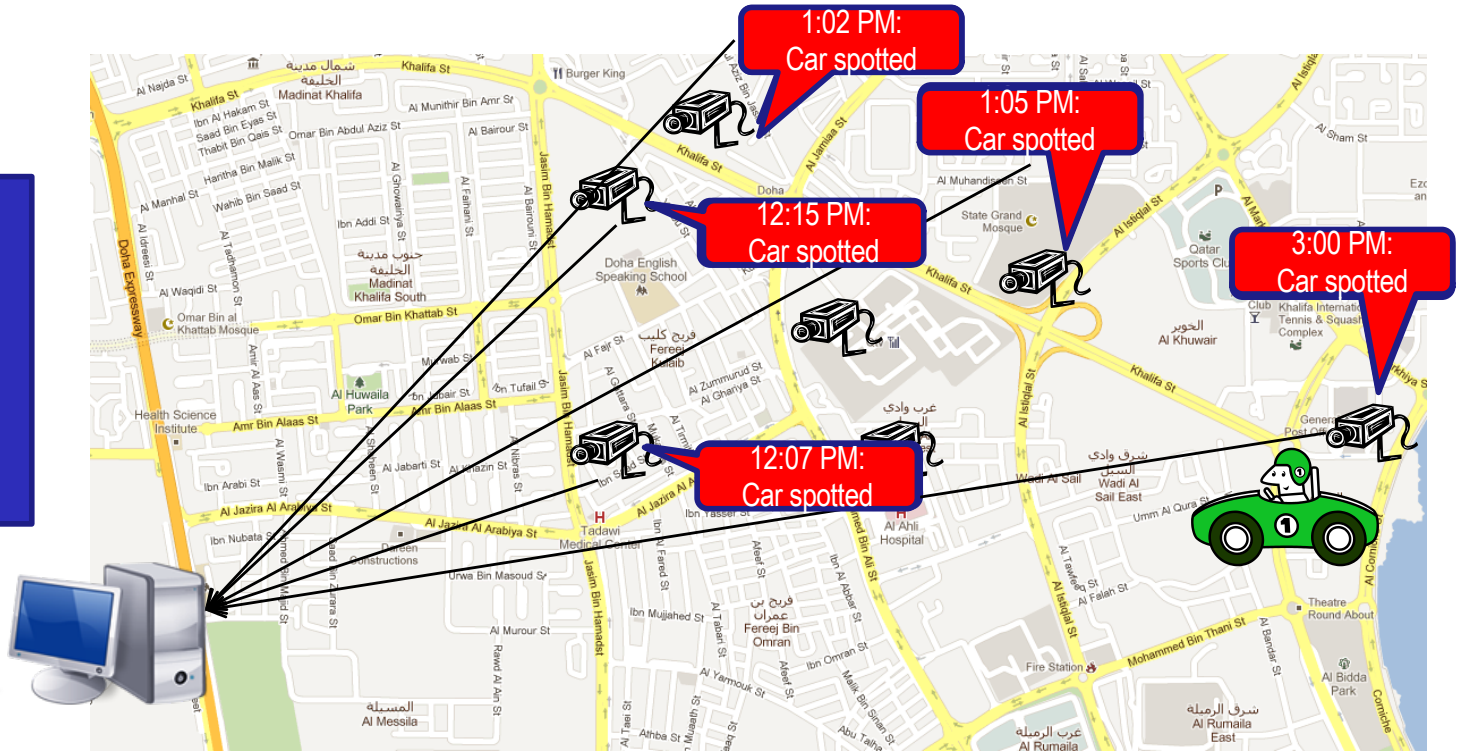
- e.g., In a distributed file system, processes have to synchronize and cooperate such that two processes are not allowed to write to the same part of a file

Need for Synchronization – Example 1

Vehicle tracking in a City Surveillance System using a Distributed Sensor Network of Cameras

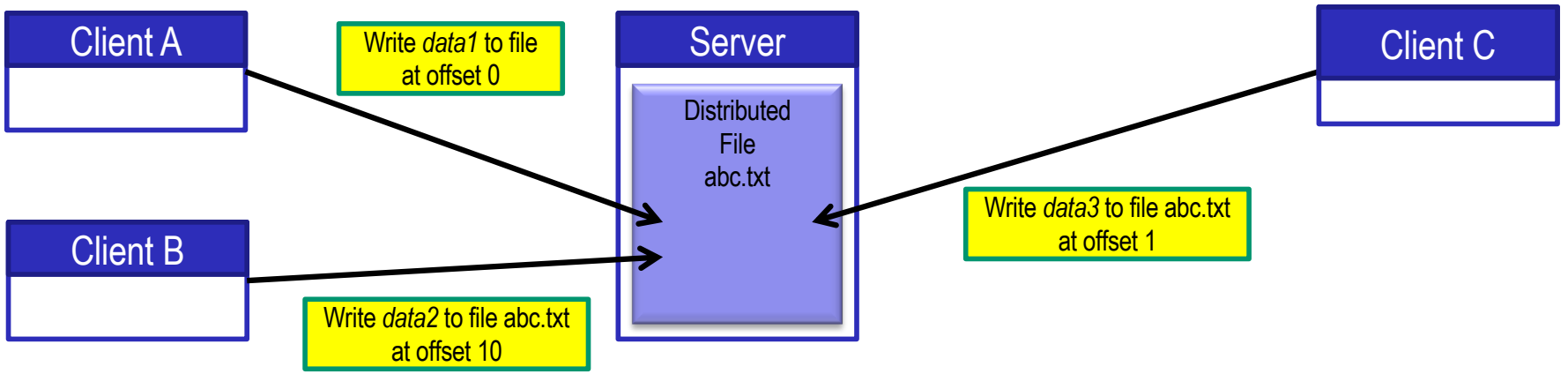
- *Objective:* To keep track of suspicious vehicles
- Camera Sensor Nodes are deployed over the city
- Each Camera Sensor that detects the vehicle reports the time to a central server
- Server tracks the movement of the suspicious vehicle

If the sensor nodes do not have a consistent version of the time, the vehicle cannot be reliably tracked



Need for Synchronization – Example 2

Writing a file in a Distributed File System



If the distributed clients do not synchronize their write operations to the distributed file, then the data in the file can be corrupted

A Broad Taxonomy of Synchronization

Reason for synchronization and cooperation	Entities have to agree on ordering of events	Entities have to share common resources
Examples	e.g., Vehicle tracking in a Camera Sensor Network, Financial transactions in Distributed eCommerce Systems	e.g., Reading and writing in a Distributed File System
Requirement for entities	Entities should have a common understanding of time across different computers	Entities should coordinate and agree on when and how to access resources
Topics we will study	Time Synchronization	Mutual Exclusion

Overview

Today's lecture

Time Synchronization

- Physical Clock Synchronization (or, simply, Clock Synchronization)
  Here, actual time on computers are synchronized
- Logical Clock Synchronization
  Computers are synchronized based on relative ordering of events

Mutual Exclusion

- How to coordinate between processes that access the same resource?

Election Algorithms

- Here, a group of entities elect one entity as the coordinator for solving a problem

Next two lectures

Overview

Time Synchronization

- Clock Synchronization
- Logical Clock Synchronization

Mutual Exclusion

Election Algorithms

Clock Synchronization

Clock Synchronization is a mechanism to synchronize the time of all the computers in a DS

We will study

- Coordinated Universal Time
- Tracking Time on a Computer
- Clock Synchronization Algorithms
  - Cristian's Algorithm
  - Berkeley Algorithm
  - Network Time Protocol

Clock Synchronization

Coordinated Universal Time

Tracking Time on a Computer

Clock Synchronization Algorithms

- Cristian's Algorithm
- Network Time Protocol
- Berkeley Algorithm

Coordinated Universal Time (UTC)

All the computers are generally synchronized to a standard time called Coordinated Universal Time (UTC)

- UTC is the primary time standard by which the world regulates clocks and time

UTC is broadcasted via the satellites

- UTC broadcasting service provides an accuracy of 0.5 msec

Computer servers and online services with **UTC receivers** can be synchronized by satellite broadcasts

- Many popular synchronization protocols in distributed systems use UTC as a reference time to synchronize clocks of computers

Clock Synchronization

Coordinated Universal Time

Tracking Time on a Computer

Clock Synchronization Algorithms

- Cristian's Algorithm
- Berkeley Algorithm
- Network Time Protocol

Tracking Time on a Computer

How does a computer keep track of its time?

- Each computer has a hardware *timer*
  - The timer causes an interrupt '`H`' times a second
- The interrupt handler adds 1 to its Software Clock (`C`)

Issues with clocks on a computer

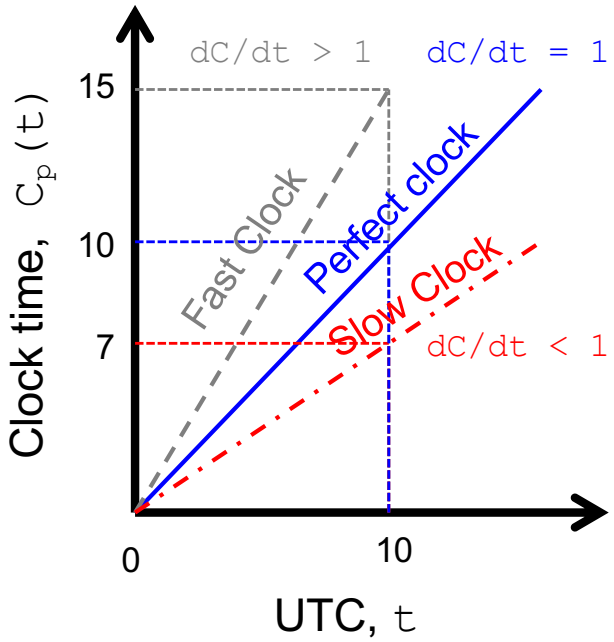
- In practice, the hardware timer is imprecise
  - It does not interrupt 'H' times a second due to material imperfections of the hardware and temperature variations
  - The computer counts the time slower or faster than actual time

Clock Skew

When the UTC time is `t`, let the clock on the computer have a time `C(t)`

Three types of clocks are possible

- Perfect clock:
  The timer ticks 'H' interrupts a second
  `dC/dt = 1`
- Fast clock:
  The timer ticks more than 'H' interrupts a second
  `dC/dt > 1`
- Slow clock:
  The timer ticks less than 'H' interrupts a second
  `dC/dt < 1`



Clock Skew (cont'd)

Frequency of the clock is defined as the ratio of the number of seconds counted by the software clock for every UTC second

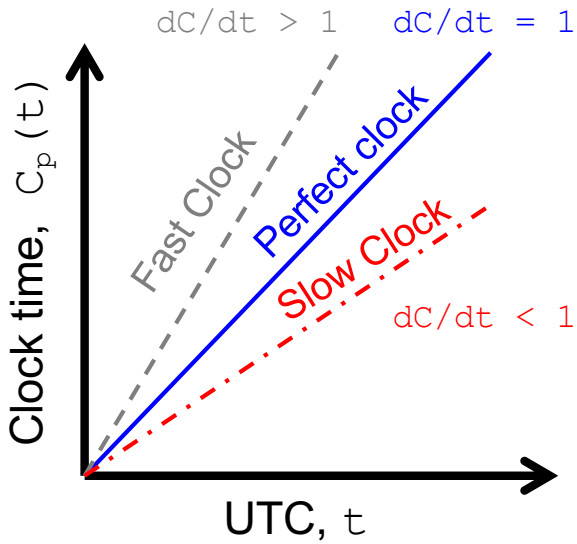
$$\text{Frequency} = dC/dt$$

Skew of the clock is defined as the extent to which the frequency differs from that of a perfect clock

$$\text{Skew} = dC/dt - 1$$

Hence,

$$Skew \begin{cases} > 0 & \text{for a fast clock} \\ = 0 & \text{for a perfect clock} \\ < 1 & \text{for a slow clock} \end{cases}$$



Clock Synchronization

Coordinated Universal Time

Tracking Time on a Computer

Clock Synchronization Algorithms

- Cristian's Algorithm
- Berkeley Algorithm
- Network Time Protocol

Cristian's Algorithm

Flaviu Cristian (in 1989) provided an algorithm to synchronize networked computers with a time server

The basic idea:

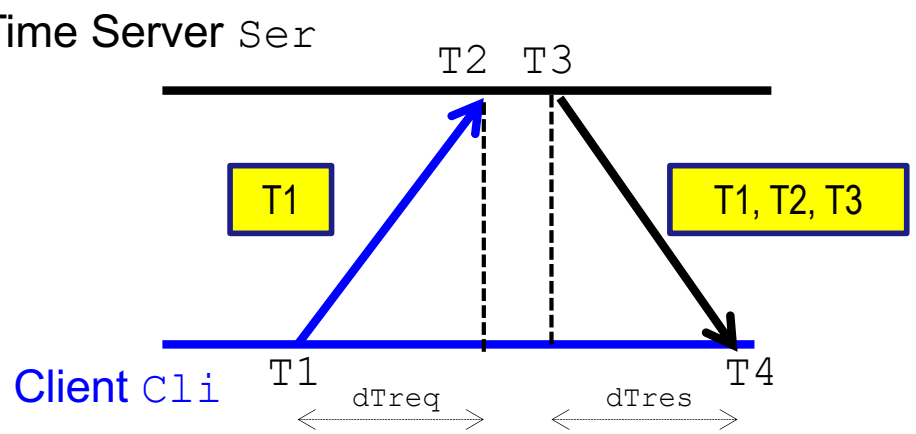
- Identify a network time server that has an accurate source for time (e.g., the time server has a UTC receiver)
- All the clients contact the network time server for synchronization

However, the network delays incurred while the client contacts the time server will have outdated the reported time

- The algorithm estimates the network delays and compensates for it

Cristian's Algorithm – Approach

- Client `Cli` sends a request to Time Server `Ser`, time stamped its local clock time `T1`
- `S` will record the time of receipt `T2` according to its local clock
  - `dTreq` is network delay for request transmission



`Ser` replies to `Cli` at its local time `T3`, piggybacking `T1` and `T2`

`Cli` receives the reply at its local time `T4`

- `dTres` is the network delay for response transmission

Now `Cli` has the information `T1`, `T2`, `T3` and `T4`

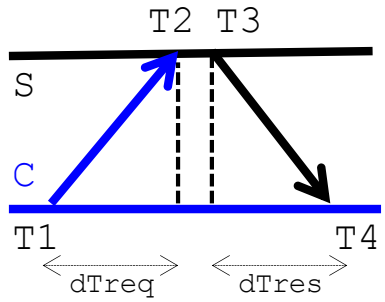
Assuming that the transmission delay from `Cli`→`Ser` and `Ser`→`Cli` are the same

`T2-T1 ≈ T4-T3`

Christian's Algorithm – Synchronizing Client Time

+ Client C estimates its offset θ relative to Time Server S

$$\theta = T3 + dTres - T4$$
$$= T3 + ((T2-T1)+(T4-T3))/2 - T4$$
$$= ((T2-T1)+(T3-T4))/2$$



+ If $\theta > 0$ or $\theta < 0$, then the client time should be incremented or decremented by θ seconds

Gradual Time Synchronization at the client

- Instead of changing the time drastically by θ seconds, typically the time is gradually synchronized
  - The software clock is updated at a lesser/greater rate whenever timer interrupts

Note: Setting clock backward (say, if $\theta < 0$) is not allowed in a DS since decrementing a clock at any computer has adverse effects on several applications (e.g., *make* program)

Cristian's Algorithm – Discussion

1. Assumption about packet transmission delays

- Cristian's algorithm assumes that the round-trip times for messages exchanged over the network is reasonably short
- The algorithm assumes that the delay for the request and response are equal

- Will the trend of increasing Internet traffic decrease the accuracy of the algorithm?
- Can the algorithm handle delay asymmetry that is prevalent in the Internet?
- Can the clients be mobile entities with intermittent connectivity?

Cristian's algorithm is intended for synchronizing computers within intranets

2. A probabilistic approach for calculating delays

- There is no tight bound on the maximum drift between clocks of computers

3. Time server failure or faulty server clock

- Faulty clock on the time server leads to inaccurate clocks in the entire DS
- Failure of the time server will render synchronization impossible

Summary

Physical clocks on computers are not accurate

Clock synchronization algorithms provide mechanisms to synchronize clocks on networked computers in a DS

- Computers on a local network use various algorithms for synchronization
  - Some algorithms (e.g, Cristian's algorithm) synchronize time with by contacting centralized time servers

Next Classes

Remaining topics in Time Synchronization

- Clock Synchronization
  Berkeley algorithm, Network Time Protocol
- Logical Clock Synchronization
  Computers are synchronized based on relative ordering of events

Mutual Exclusion

- How to coordinate between processes that access the same resource?