

Time and Global States

- 1 Introduction
- 2 Global states
- 3 Clocks, events and process states
- 4 Synchronizing physical clocks


Global States

- ▶ A distributed system consists of multiple processes
- ▶ Each process is located on a different computer
 - No sharing of processor or memory

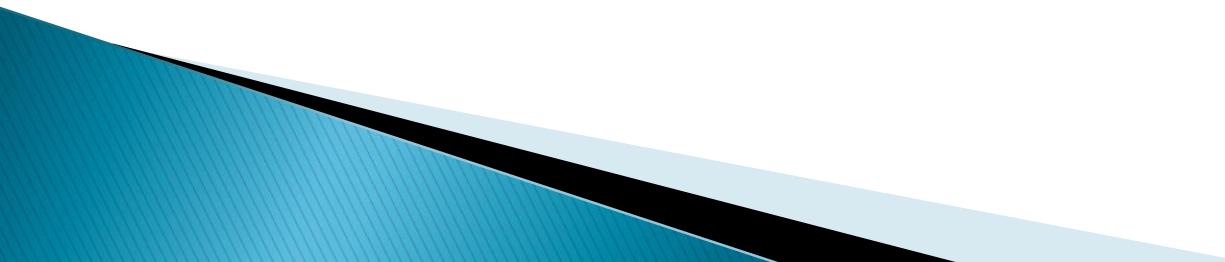
Global States

- ▶ Each process can only determine its own “state”
- ▶ Problem: How do we determine when to garbage collect in a distributed system?
 - How do we check whether a reference to memory still exists?

Global States

- ▶ A distributed system consists of multiple processes
 - ▶ Each process is located on a different computer
 - ▶ Each process consists of “events”
 - ▶ An event is either sending a message, receiving a message, or changing the value of some variable
 - ▶ Each process has a communication channel in and out
- 

Global States

- ▶ In order to test whether a certain property of our system is true, we cannot just look at each process individually
 - ▶ A “snapshot” of the entire system must be taken to test whether a certain property of the system is true
 - ▶ This “snapshot” is called a Global State
- 

Global States: Definition

The global state of a distributed system is the set of local states of each individual processes involved in the system plus the state of the communication channels.

Introduction to clocks in DS(1)

Importance of time in distributed systems:


A quantity to timestamp events accurately

- To know what time a particular event occurs
- i.e. Recording when an e-commerce transaction occurs

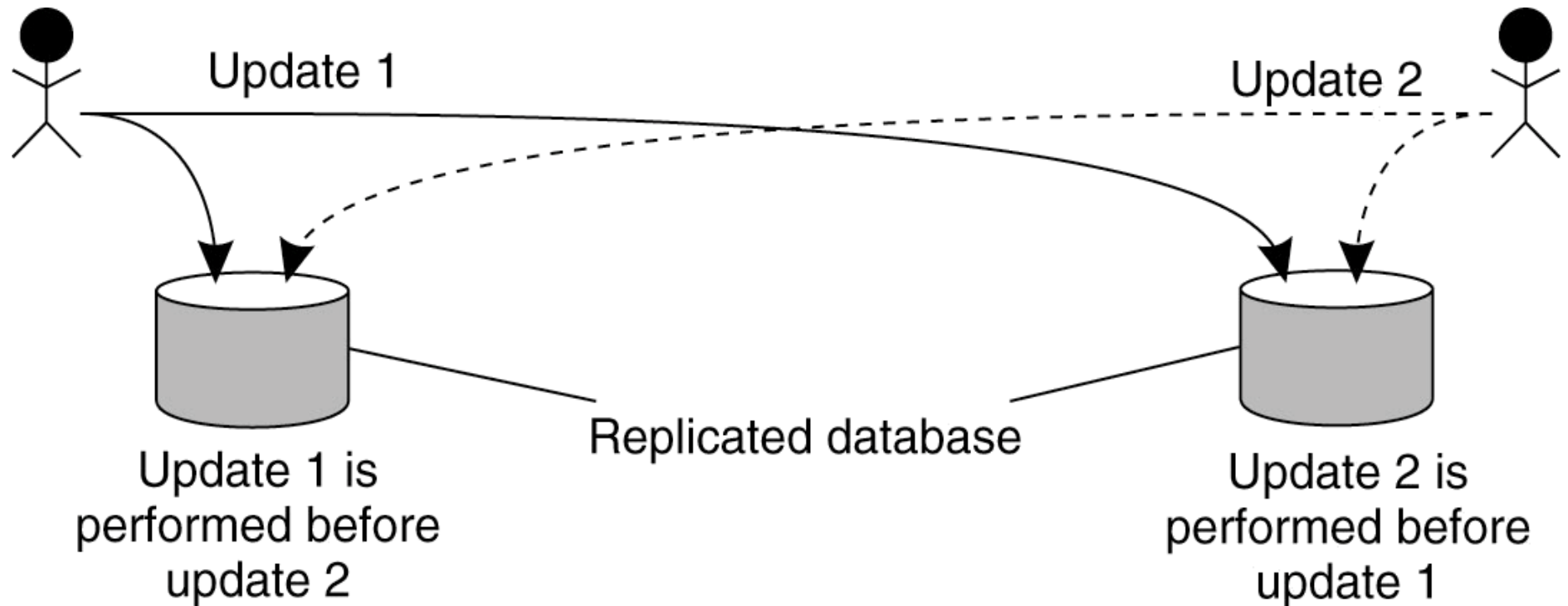
A synchronization source for several distributed

- To maintain consistency of distributed data
- i.e. Eliminating duplicate updates

A timing source for multiple events

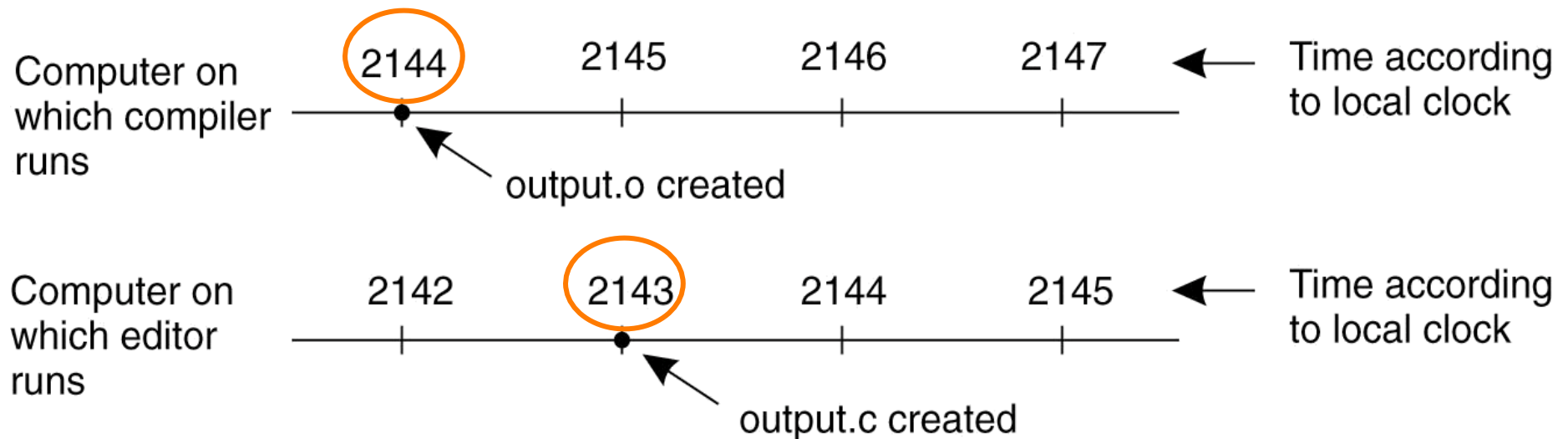
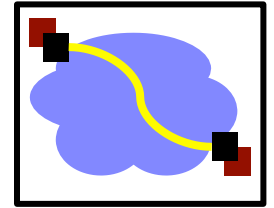
- To provide relative order of two events
 - i.e. Ensuring the order of cause and effect
- 

Replicated Database Update



- Updating a replicated database and leaving it in an inconsistent state

Impact of Clock Synchronization



- When each machine has its own clock, an event that occurred after another event may nevertheless be assigned an earlier time.

Introduction to clocks in DS(1)

Clocks in computers to establish:

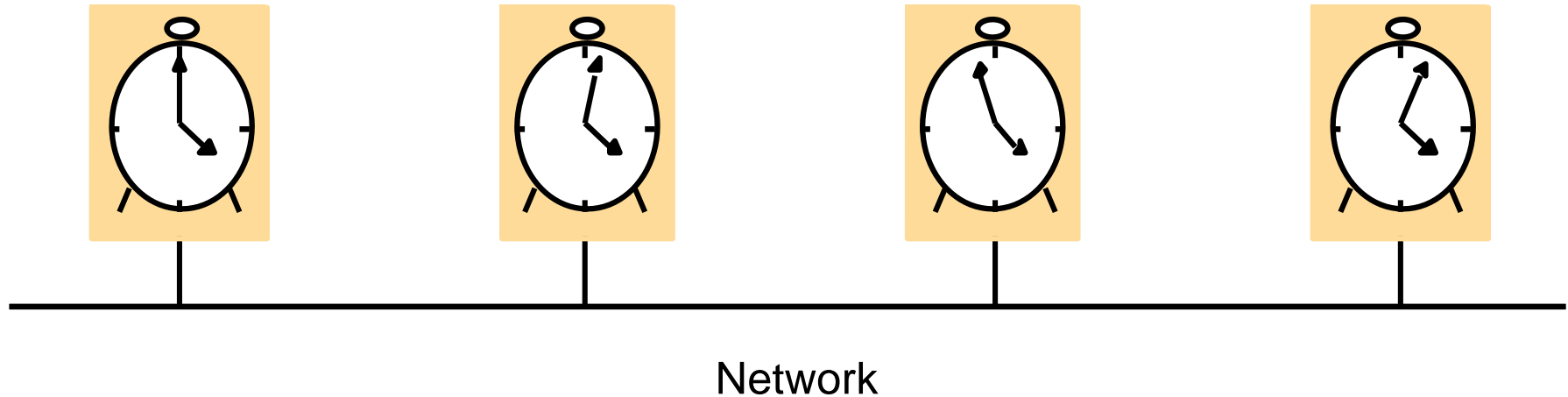
Time at which an event occurred

Duration of an event or interval between two events

Sequence of a series of events or the order in which events occurred

- To timestamp events, use the computer's clock
- At Real time, t , the OS reads the time on the computer's hardware clock $H_i(t)$
- It calculates time in its software clock $C_i(t) = \alpha H_i(t) + \beta$
- In general, the clock is not completely accurate
- but if C_i behaves well enough, it can be used to timestamp events at process p_i

Skew between computer clocks in a distributed system

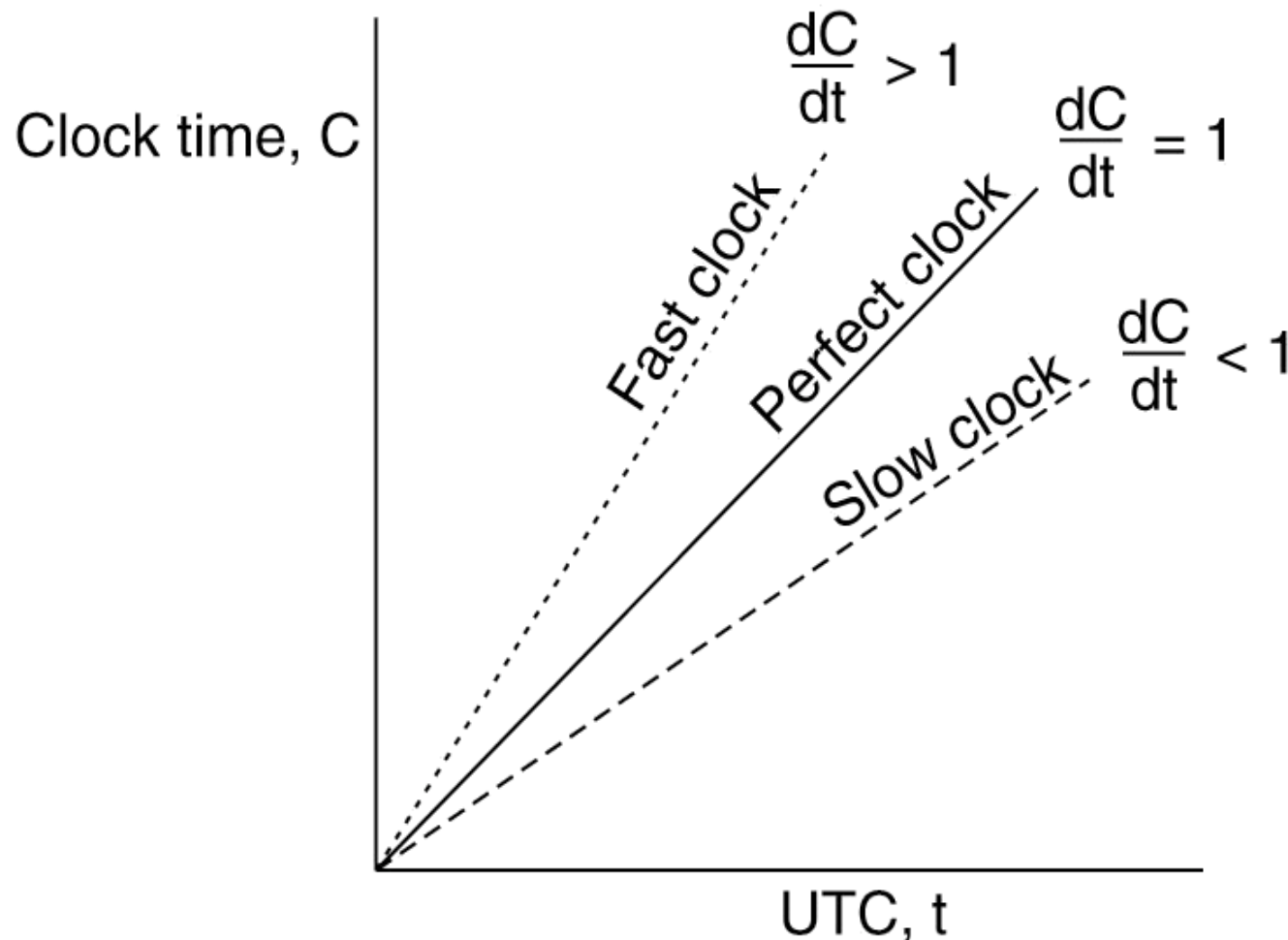
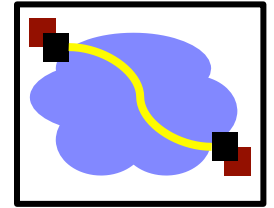


- Each computer in distributed system have their own internal clock
- used by local processes to obtain the value of the current time
- processes on different computers can timestamp their events
- but clocks on different computers may give different times
- computer clocks drift from perfect time and their drift rates differ from one another.
- **clock drift rate**: the relative amount that a computer clock differs from a perfect clock
- **Clock skew**: the difference between the times on two clocks (at any instant)

Computer clocks and timing events

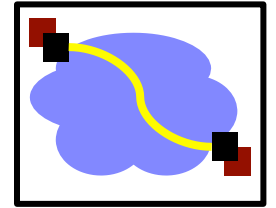
- Even if clocks on all computers in a DS are set to the same time, their clocks will eventually vary quite significantly unless corrections are applied
- Therefore, We need to measure time accurately to know the time an event occurred at a computer to do this we need to synchronize its clock with an authoritative external clock

Clock Synchronization Algorithms



- The relation between clock time and UTC when clocks tick at different rates.

Time Standards



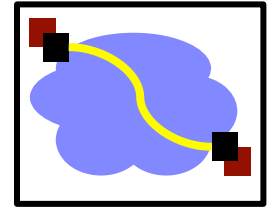
- UT1 (universal time)
 - Based on astronomical observations
 - ~ “Greenwich Mean Time” (GMT)
- TAI (international atomic time)
 - Started Jan 1, 1958
 - Each second is 9,192,631,770 cycles of radiation emitted by Cesium atom
 - Has diverged from UT1 due to slowing of earth's rotation
- UTC (coordinated universal time)
 - TAI + leap seconds to be within 0.9s of UT1
 - Currently 36s, Most recent update: June 30, 2015

Coordinated Universal Time (UTC)

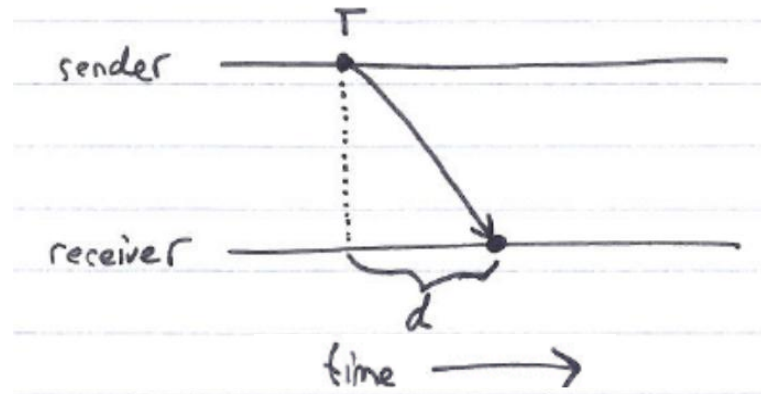
- y International Atomic Time is based on very accurate physical clocks (drift rate 10^{-13})
- y UTC is an international standard for time keeping
- y It is based on atomic time, but occasionally adjusted to astronomical time
- y It is broadcast from radio stations on land and satellite (e.g. GPS)
- y Computers with receivers can synchronize their clocks with these timing signals
- y Signals from land-based stations are accurate to about 0.1-10 millisecond
- y Signals from GPS are accurate to about 1 microsecond

Why can't we put GPS receivers on all our computers?

Perfect networks

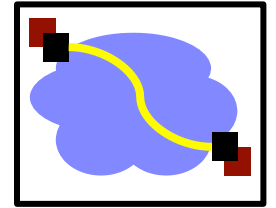


- Messages always arrive, with propagation delay **exactly** d

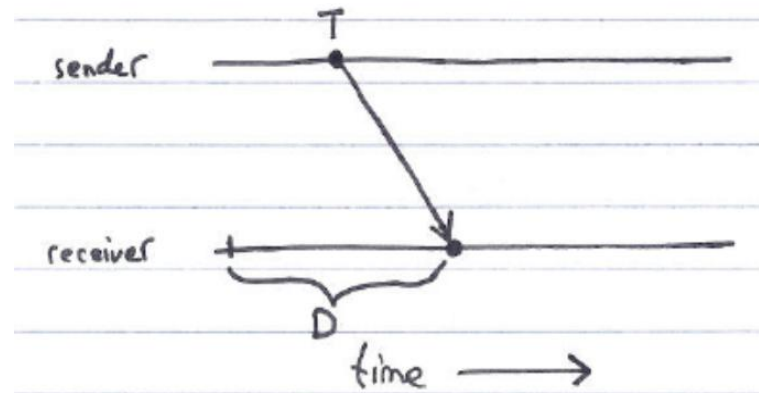


- Sender sends time T in a message
- Receiver sets clock to $T+d$
 - Synchronization is exact

Synchronous networks

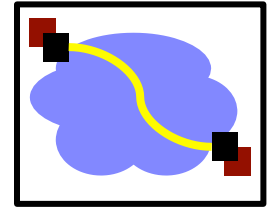


- Messages always arrive, with propagation delay *at most* D



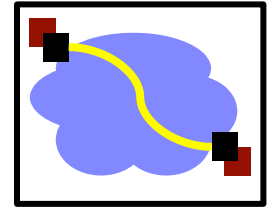
- Sender sends time T in a message
- Receiver sets clock to $T + D/2$
 - Synchronization error is at most $D/2$

Synchronization in the real world

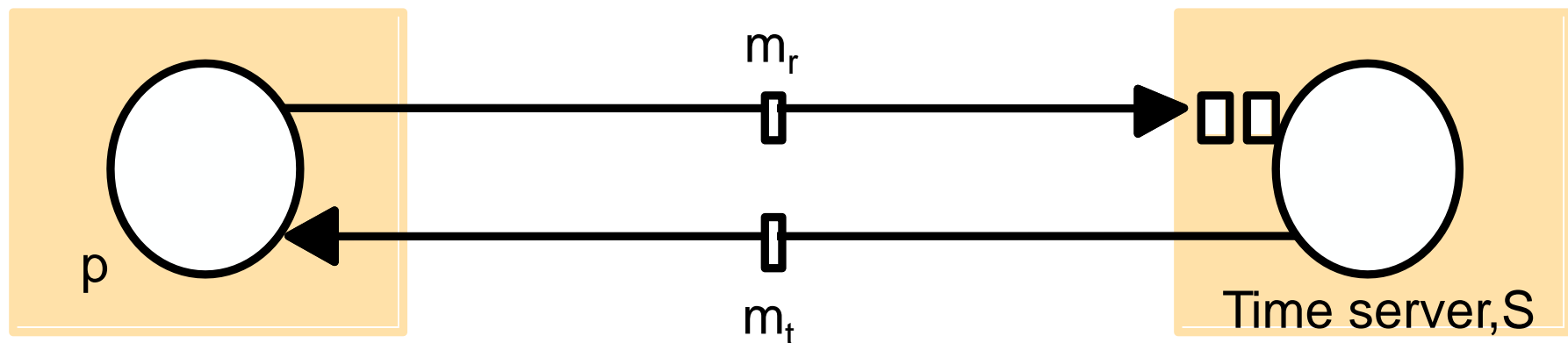


- Real networks are asynchronous
 - Message delays are arbitrary
- Real networks are unreliable
 - Messages don't always arrive

Cristian's Time Sync ('89)

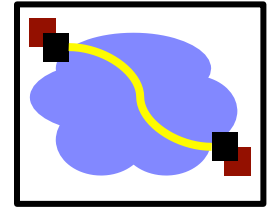


- A time server S receives signals from a UTe source
 - Process p requests time in m_r and receives t in m_t from S
 - p sets its clock to $t + T_{\text{round}}/2$
 - Accuracy $\pm (T_{\text{round}}/2 - \text{min})$:
 - because the earliest time S puts t in message m_t is min after p sent
 - m_r .
 - the latest time was min before m_t arrived at p
 the time by S 's clock when m_t arrives is in the range $[t + \text{min}, t + T_{\text{round}} - \text{min}]$



T_{round} is the round trip time recorded by p
 min is an estimated minimum on way delay

Berkeley algorithm



- Cristian's algorithm -

- a single time server might fail, so they suggest the use of a group of synchronized servers it does not deal with faulty servers

- Berkeley algorithm (also 1989)

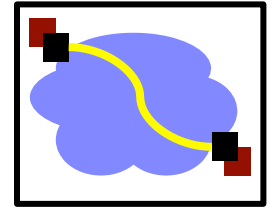
- An algorithm for internal synchronization of a group of computers
- A *master* polls to collect clock values from the others (*slaves*)

- The master uses round trip times to estimate the slaves' clock values It takes an average (eliminating any above average round trip time or faulty clocks) with
- It sends the required **adjustment** to the slaves (better than sending the time which depends on the round trip time)

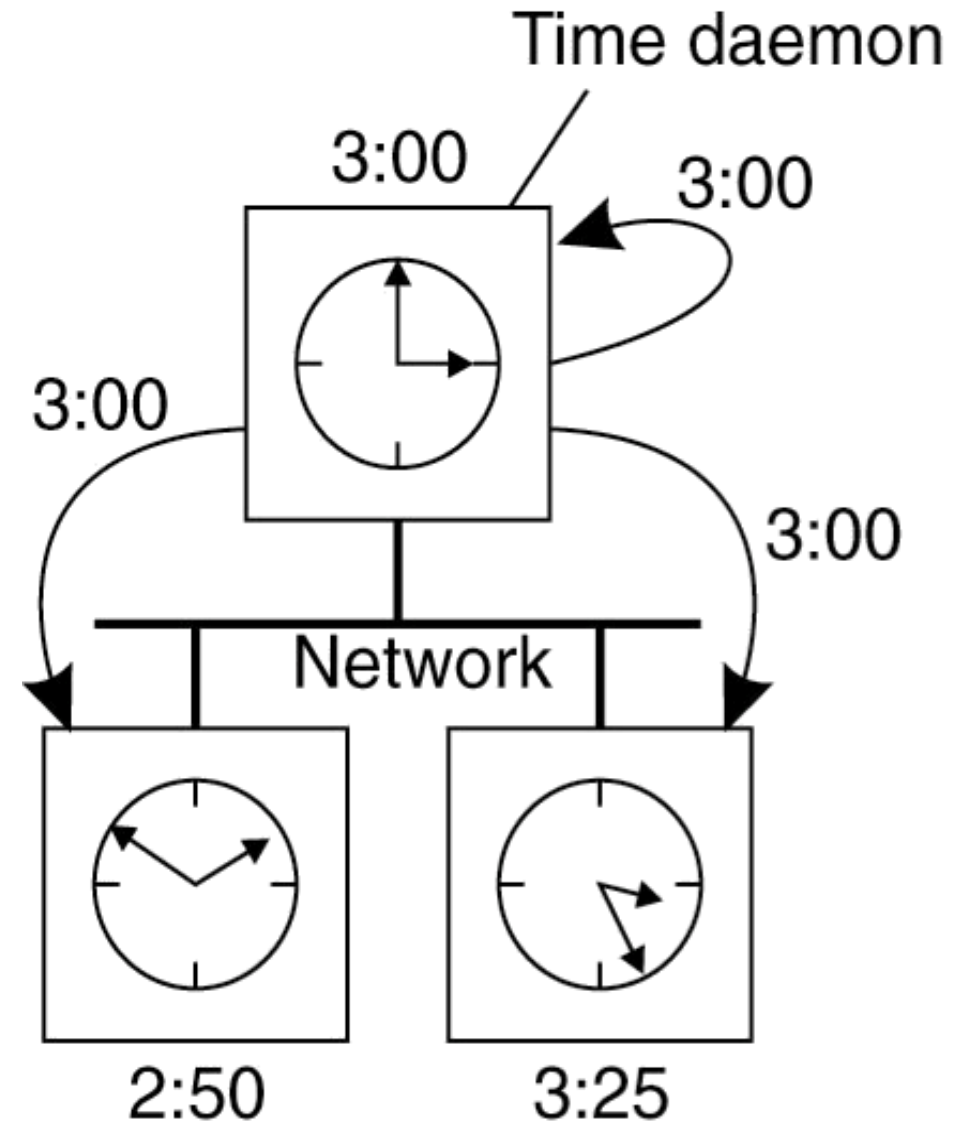
Measurements

- 15 computers, clock synchronization 20-25 millisecs drift rate $< 2 \times 10^{-5}$
- If master fails, can elect a new master to take over (not in bounded time)

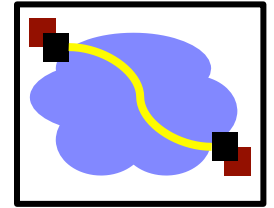
The Berkeley Algorithm (1)



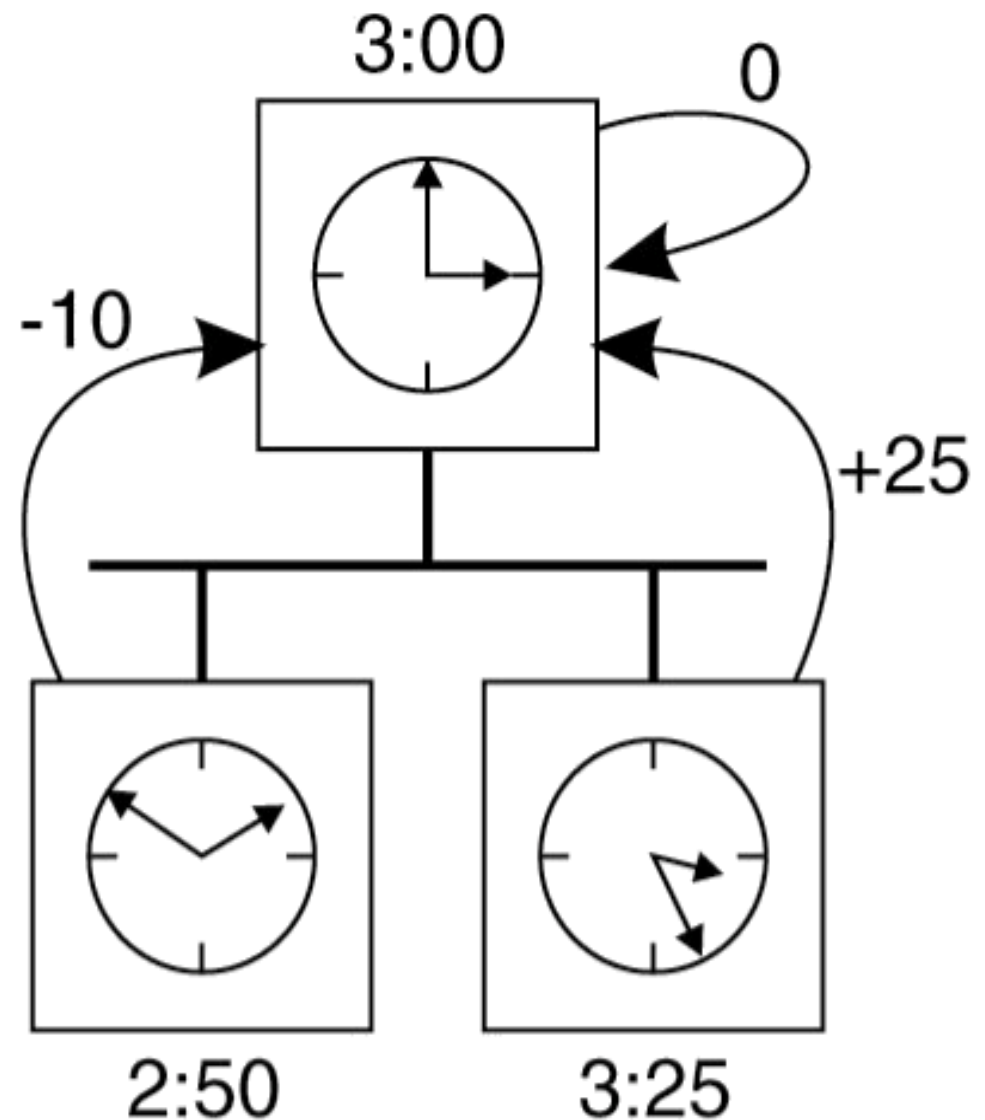
- The time daemon asks all the other machines for their clock values.



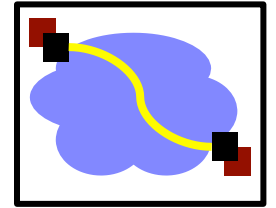
The Berkeley Algorithm (2)



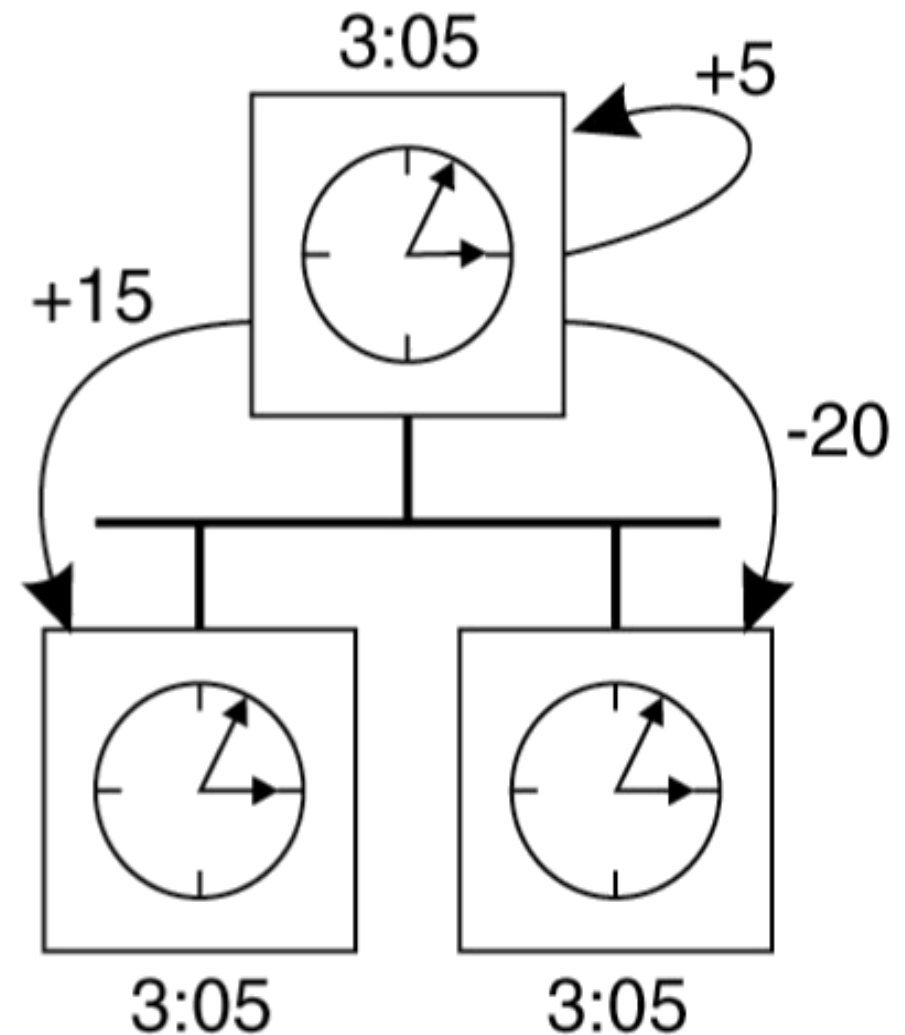
- The machines answer.



The Berkeley Algorithm (3)

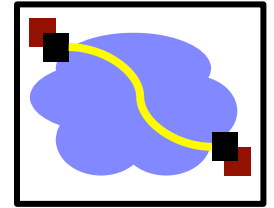


- The time daemon tells everyone how to adjust their clock.



Network Time Protocol (NTP)

(invented by David Mills, 1981)



- A time service for the Internet - synchronizes to clients

Reliable time source Primary servers are connected to UTC sources and communicate

Secondary servers are synchronized to primary servers

Synchronization subnet - lowest level servers in users' computers

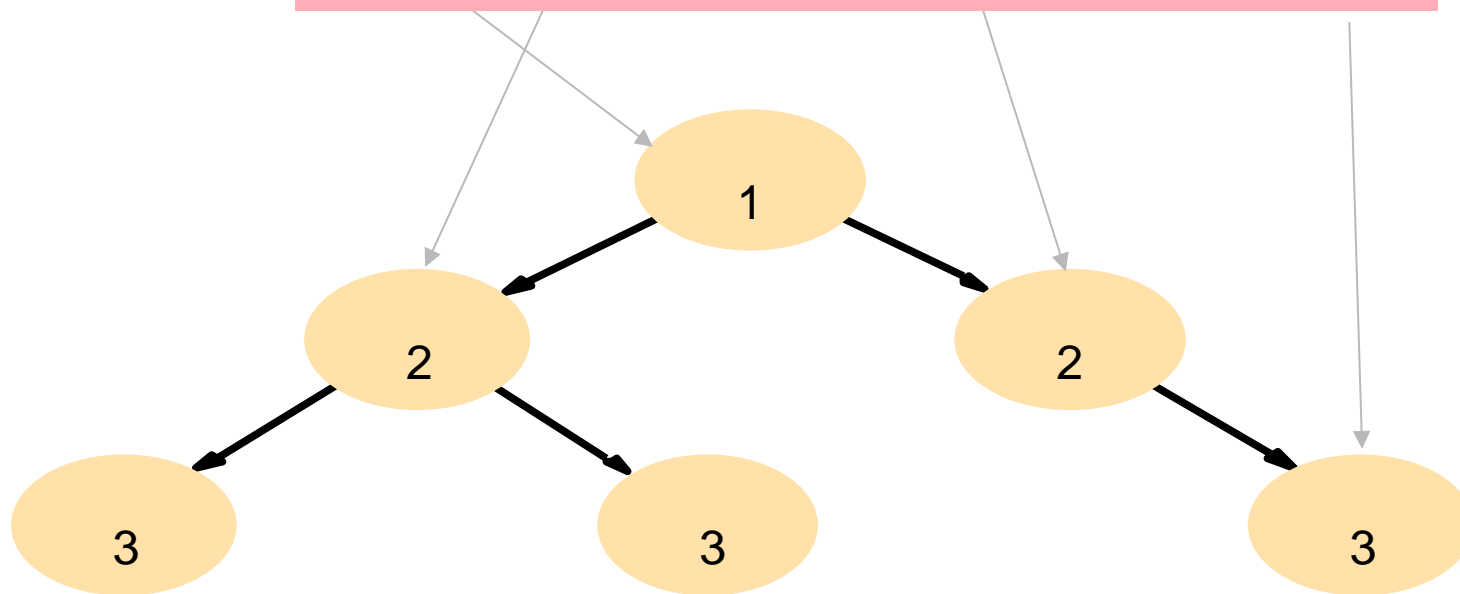


Figure 10.3

NTP - synchronisation of servers

y 3 Modes of synchronization:

y Multicast

ŠA server within a high speed LAN multicasts time to others which set clocks assuming some delay (not very accurate)

y Procedure call

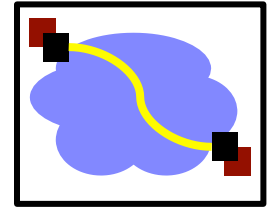
ŠA server accepts requests from other computers (like Cristian's algorithm). Higher accuracy. Useful if no hardware multicast.

y Symmetric

ŠPairs of servers exchange messages containing time information

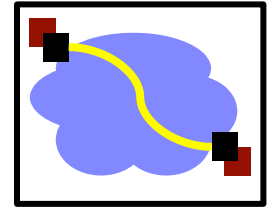
ŠUsed where very high accuracies are needed (e.g. for higher levels)

The Network Time Protocol (NTP)



- Uses a hierarchy of time servers
 - class 1 servers have highly-accurate clocks
 - connected directly to atomic clocks, etc.
 - class 2 servers get time from only class 1 and class 2 servers
 - class 3 servers get time from any server (usually 3) to Cristian's alg.
- Synchronization similar
 - Modified to use multiple one-way messages instead of immediate round-trip
- Accuracy: Local ~1ms, Global ~10ms

U. Delaware Master Time Facility (MTF) (from January 2000)



Spectracom 8170 WWVB Receiver

Spectracom 8183 GPS Receiver

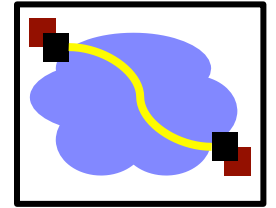
Spectracom 8170 WWVB Receiver

Spectracom 8183 GPS Receiver

Hewlett Packard 105A Quartz
Frequency Standard

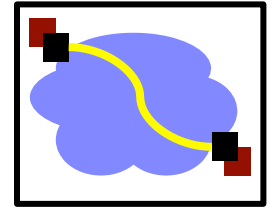
Hewlett Packard 5061A cesium Beam
Frequency Standard

How To Change Time



- Can't just change time
 - Why not?
- Change the update rate for the clock
 - changes time in a more gradual fashion
 - Prevents **inconsistent** local timestamps

Important Lessons



- Clocks on different systems will always behave differently Skew and drift between clocks
- Time disagreement between machines can result in undesirable behavior
- Clock synchronization
 - Rely on a time-stamped network messages
 - Estimate delay for message transmission
 - can synchronize to UTC or to local source
 - clocks never exactly synchronized
- Often inadequate for distributed systems
 - might need totally-ordered events
 - might need millionth-of-a-second precision