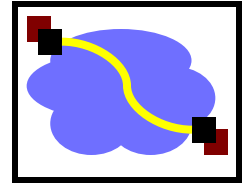


# 416 Distributed Systems

Time Synchronization

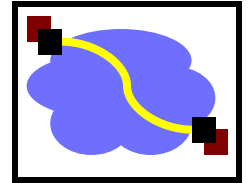
Feb 2 2017

# Today's Lecture



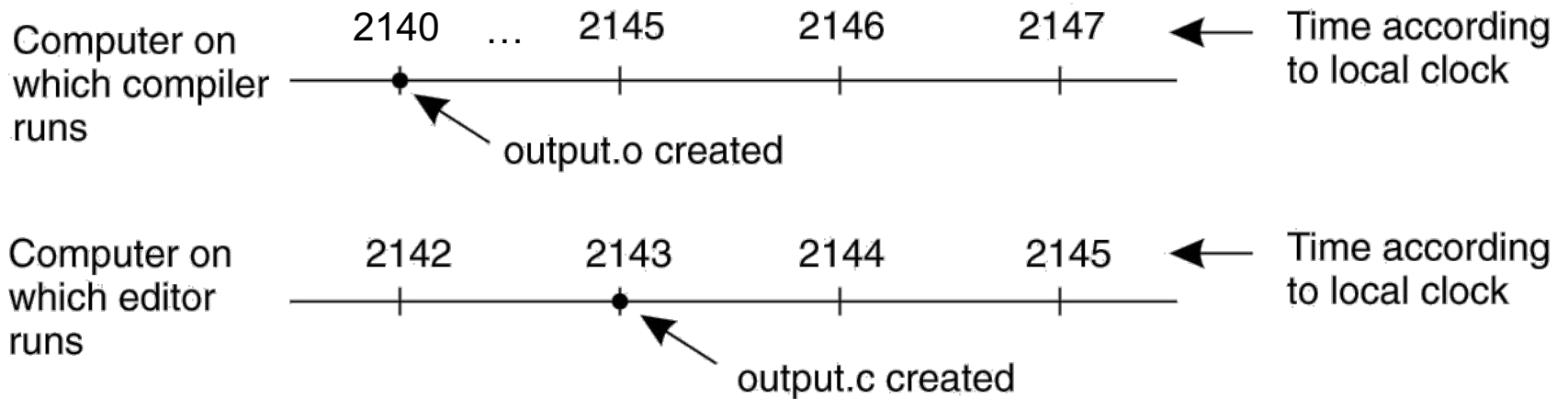
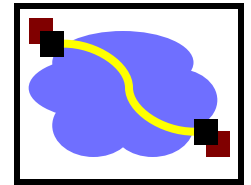
- Need for time synchronization
- Time synchronization techniques
- Lamport Clocks
- Vector Clocks

# Why Global Timing?

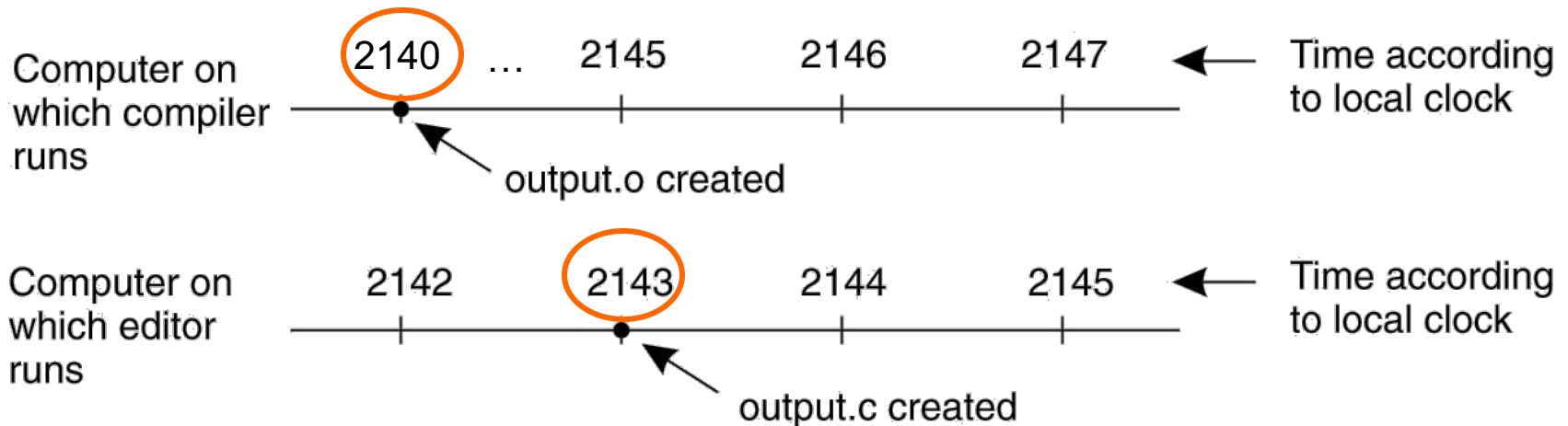
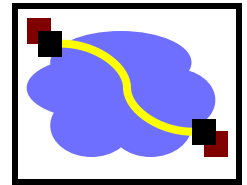


- Suppose there were a globally consistent time standard
- Would be handy
  - Who got last seat on airplane?
  - Who submitted final auction bid before deadline?
  - Did defense move before snap? (football reference)
  - DFS:
    - Did Read(chunk0) happen before Write(chunk0)?
  - BlockArt:
    - Did AddShape(s1) happen before AddShape(s2)?

# Impact of Clock Synchronization

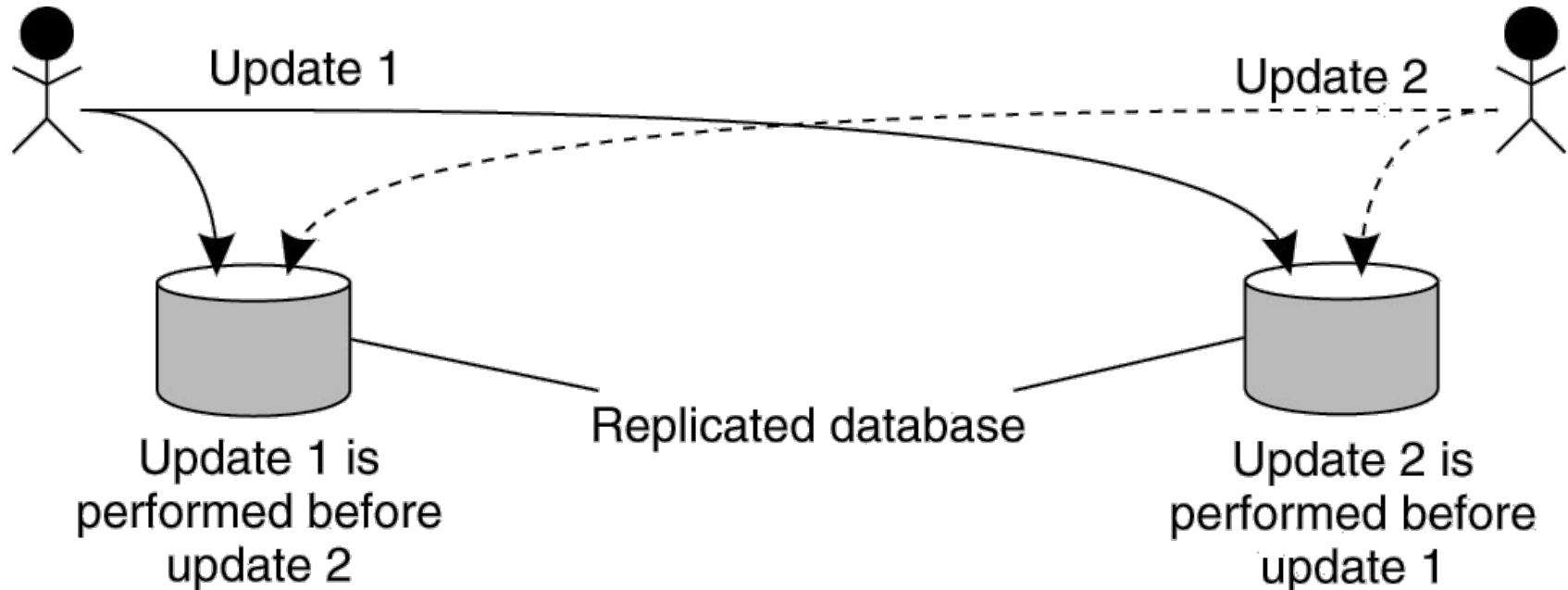
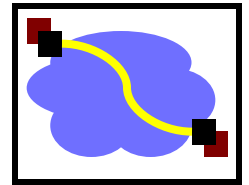


# Impact of Clock Synchronization



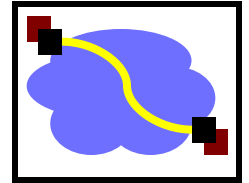
- When each machine has its own clock, an event that occurred after another event may nevertheless be assigned an earlier time.

# Replicated Database Update



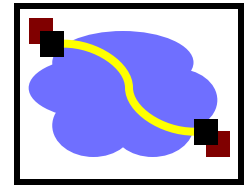
- Updating a replicated database and leaving it in an inconsistent state

# Time Standards

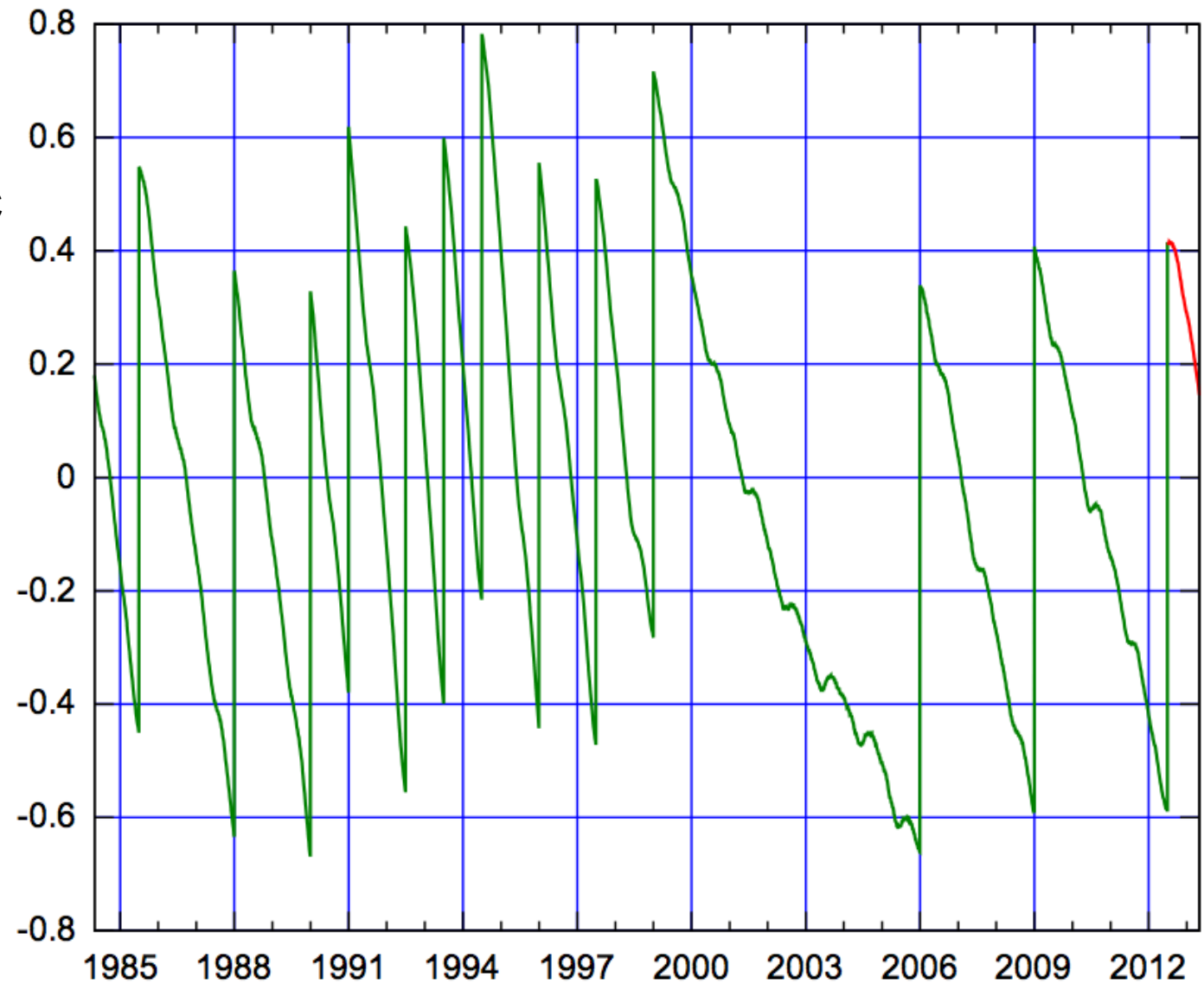


- UT1 (universal time)
  - Based on astronomical observations
  - ~ “Greenwich Mean Time” (GMT)
- TAI (international atomic time)
  - Started Jan 1, 1958
  - Each second is 9,192,631,770 cycles of radiation emitted by Cesium atom
  - Has diverged from UT1 due to slowing of earth’s rotation
- UTC (coordinated universal time)
  - TAI + leap seconds to be within 0.9s of UT1
  - Currently 37s
  - Most recent update: Dec, 2016

# Comparing Time Standards

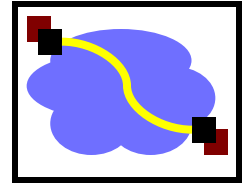


UT1 - UTC



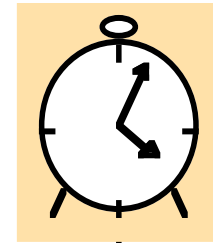
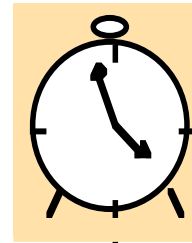
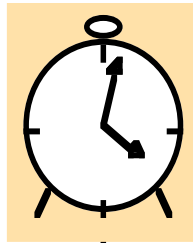
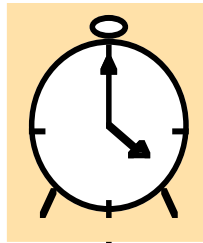
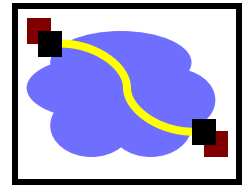


# Coordinated Universal Time (UTC)



- Is broadcast from radio stations on land and satellite (e.g. GPS)
- Computers with receivers can synchronize their clocks with these timing signals
- Signals from land-based stations are accurate to about 0.1-10 millisecond
- Signals from GPS are accurate to about 1 microsecond
  - Why can't we use GPS receivers on all our computers?

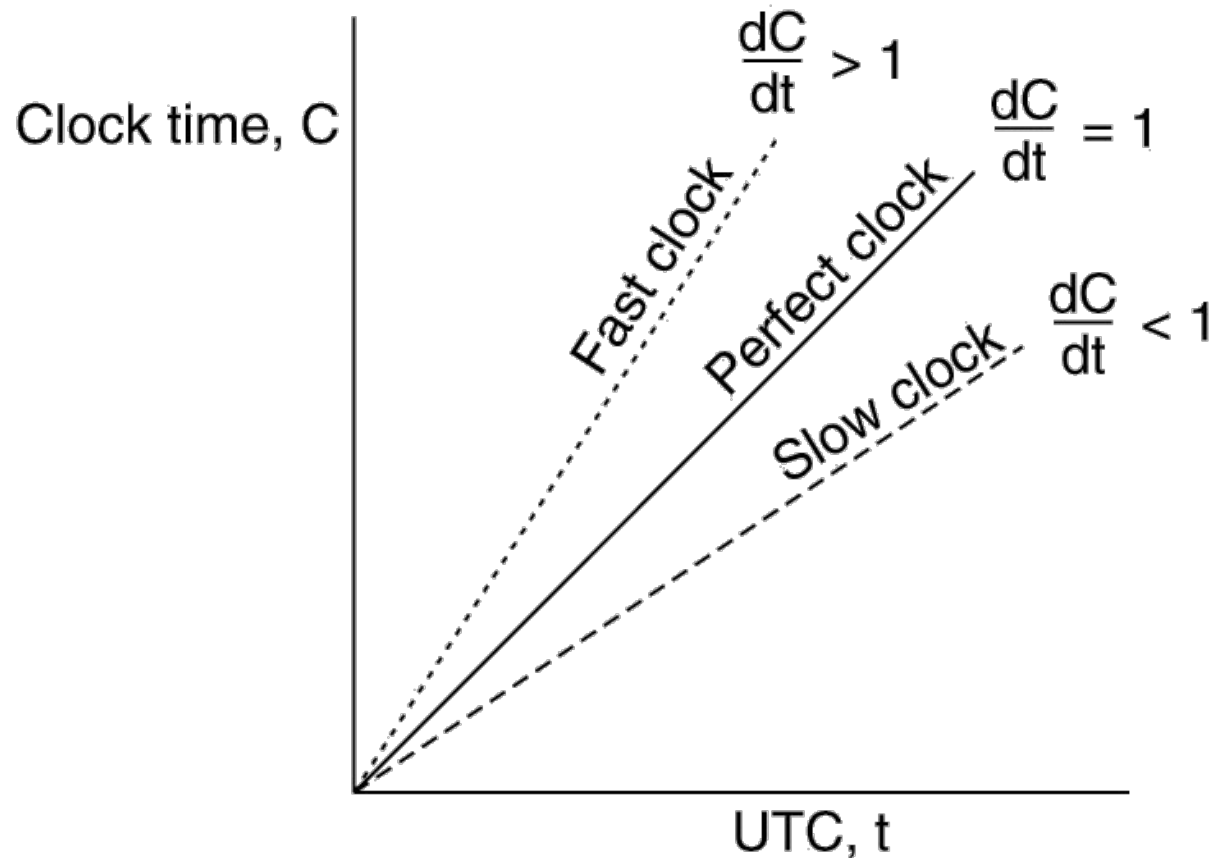
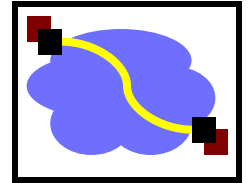
# Clocks in a Distributed System



Network

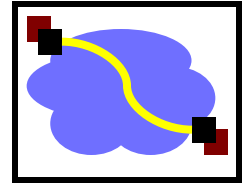
- Computer clocks are **not** generally in perfect agreement
  - **Skew**: the difference between the times on two clocks (at any instant)
- Computer clocks are subject to clock drift (they count time at different rates; consider batteries)
  - **Clock drift rate**: the difference per unit of time from some ideal reference clock
  - Ordinary quartz clocks drift by about 1 sec in 11-12 days. ( $10^{-6}$  secs/sec).
  - High precision quartz clocks drift rate is about  $10^{-7}$  or  $10^{-8}$  secs/sec

# Clock Synchronization Algorithms



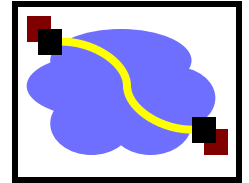
- The relation between clock time and UTC when clocks tick at different rates.

# Today's Lecture

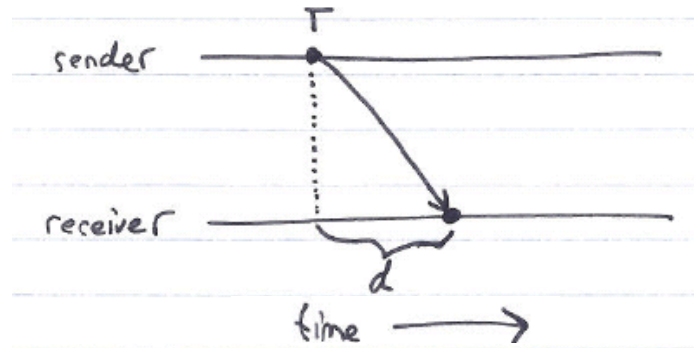


- Need for time synchronization
- Time synchronization techniques
- Lamport Clocks
- Vector Clocks

# Perfect networks

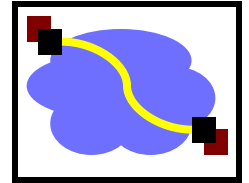


- Messages always arrive, with propagation delay **exactly**  $d$

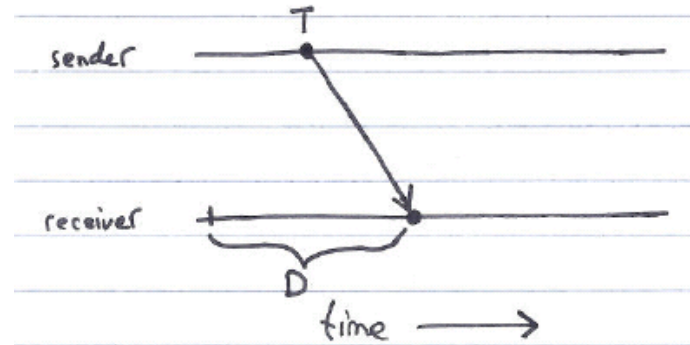


- Sender sends time  $T$  in a message
- Receiver sets clock to  $T+d$ 
  - Synchronization is exact

# Synchronous networks

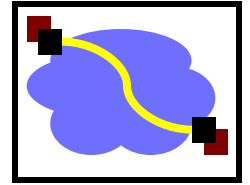


- Messages always arrive, with propagation delay *at most*  $D$



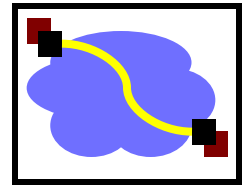
- Sender sends time  $T$  in a message
- Receiver sets clock to  $T + D/2$ 
  - Synchronization error is at most  $D/2$

# Synchronization in the real world

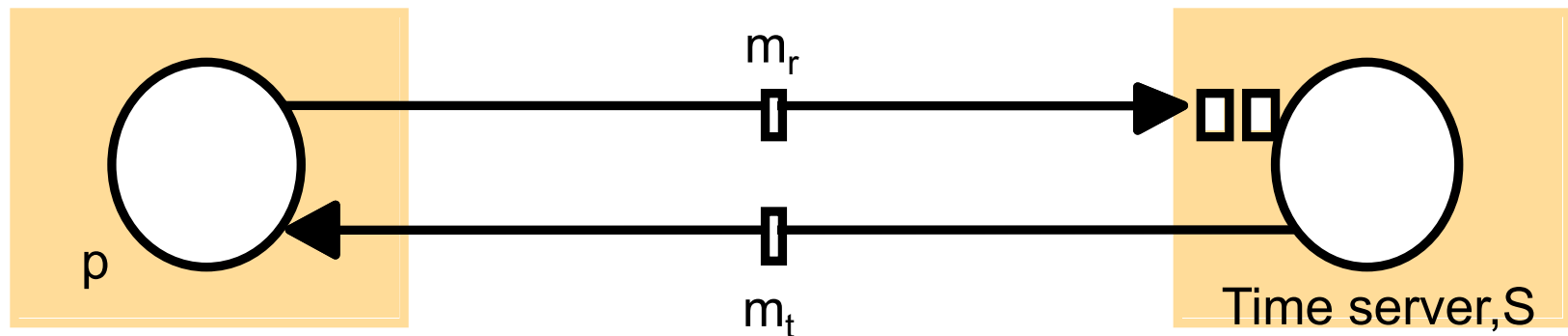


- Real networks are asynchronous
  - Message delays are arbitrary
- Real networks are unreliable
  - Messages don't always arrive

# Cristian's Time Sync ('89)



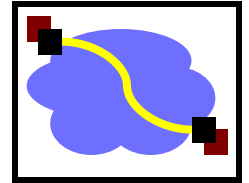
- A time server  $S$  receives signals from a UTC source
  - Process  $p$  requests time in  $m_r$  and receives  $t$  in  $m_t$  from  $S$
  - $p$  sets its clock to  $t + T_{\text{round-trip}}/2$
  - Accuracy  $\pm (T_{\text{round-trip}}/2 - \text{min})$  :
    - Where  $\text{min}$  is minimum one-way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



$T_{\text{round}}$  is the round trip time recorded by  $p$   
 $\text{min}$  is an estimated minimum one way delay

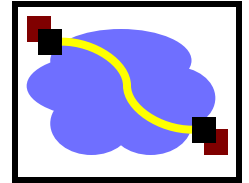


# Berkeley algorithm

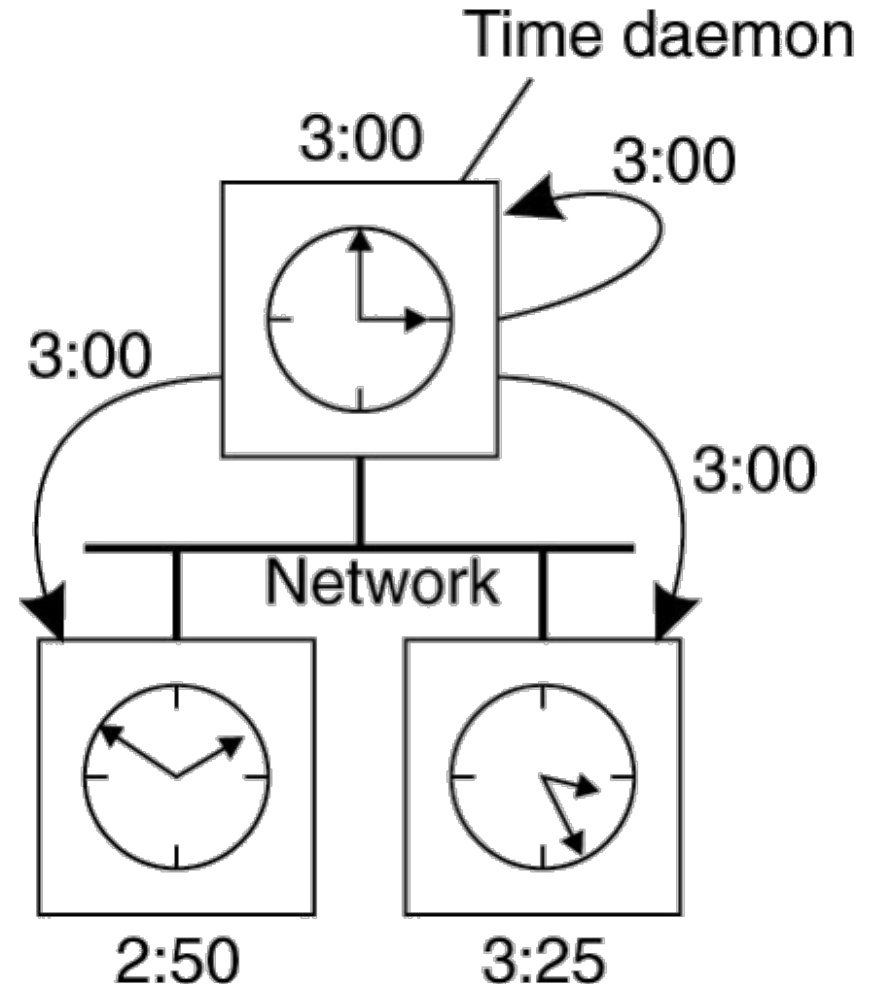


- Cristian's algorithm -
  - a single time server might fail, so they suggest the use of a group of synchronized servers
  - it does not deal with faulty servers
- Berkeley algorithm (also 1989)
  - An algorithm for internal synchronization of a group of computers
  - A *master* polls to collect clock values from the others (*slaves*)
  - The master uses round trip times to estimate the slaves' clock values (only master computes RTT)
  - It takes an average (eliminating any above average round trip time or with faulty clocks)
  - It sends the required **adjustment** to the slaves (better than sending the time which depends on the round trip time)
  - Failures
    - If master fails, can elect a new master to take over (not in bounded time)

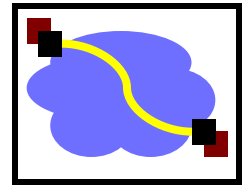
# The Berkeley Algorithm (1)



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



# The Berkeley Algorithm (2)



- The machines answer.

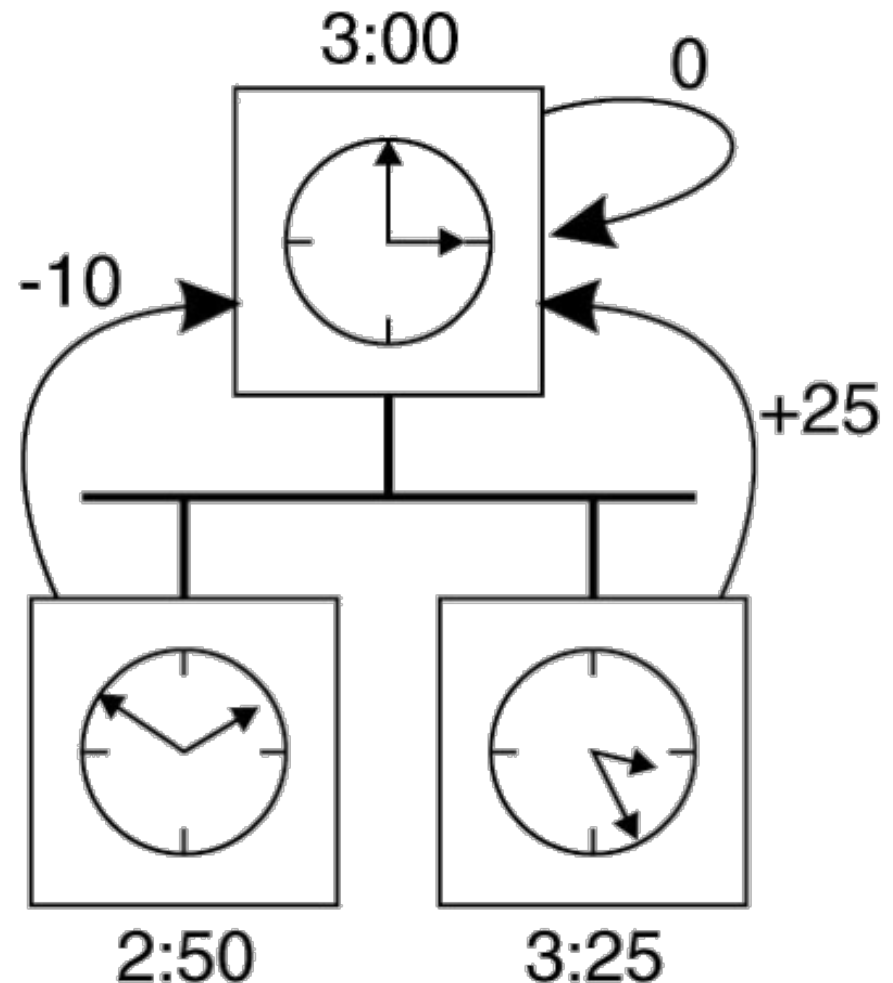
Compute avg:  
 $+15 / 3 = +5$

Adjustment:

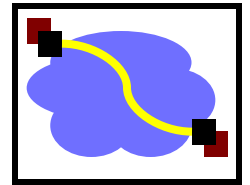
$0 \rightarrow +5 = +5$

$-10 \rightarrow +5 = +15$

$+25 \rightarrow +5 = -20$



# The Berkeley Algorithm (3)



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

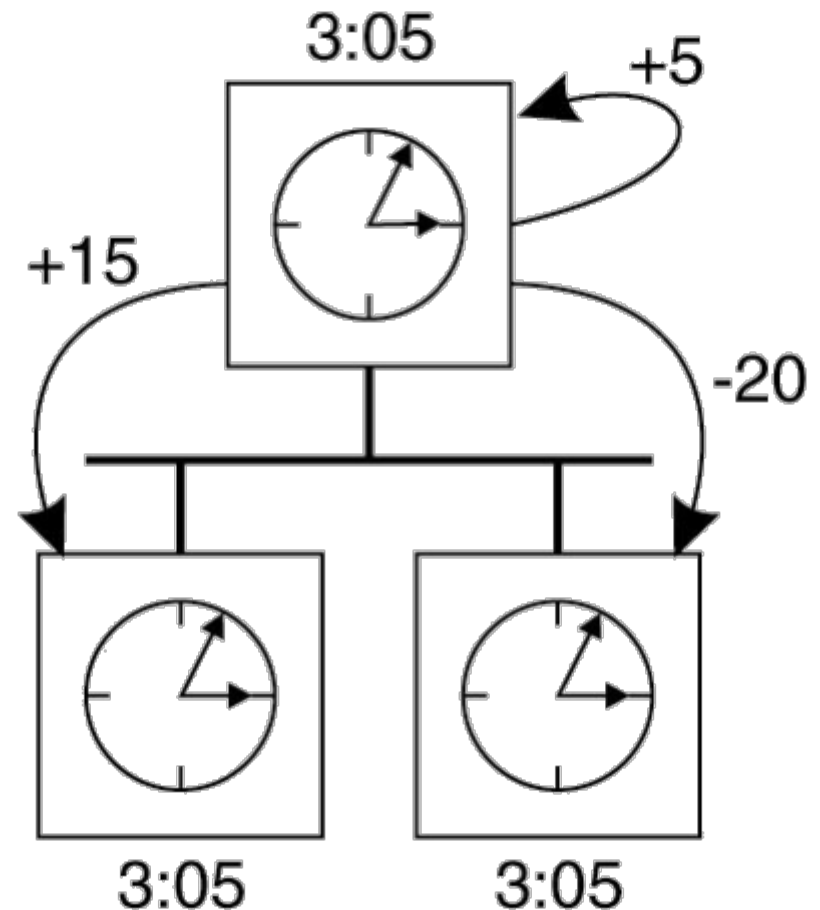
Compute avg:  
 $+15 / 3 = +5$

Adjustment:

$0 \rightarrow +5 = +5$

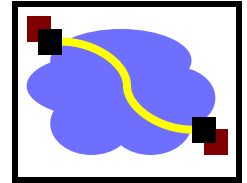
$-10 \rightarrow +5 = +15$

$+25 \rightarrow +5 = -20$



# Network Time Protocol (NTP)

(invented by David Mills, 1981)



- A time service for the Internet - synchronizes clients to

Reliability of time source is maintained by authenticating

time source

Primary servers are connected to UTC

Secondary servers are synchronized to primary servers

Synchronization subnet - lowest level servers in users' computers

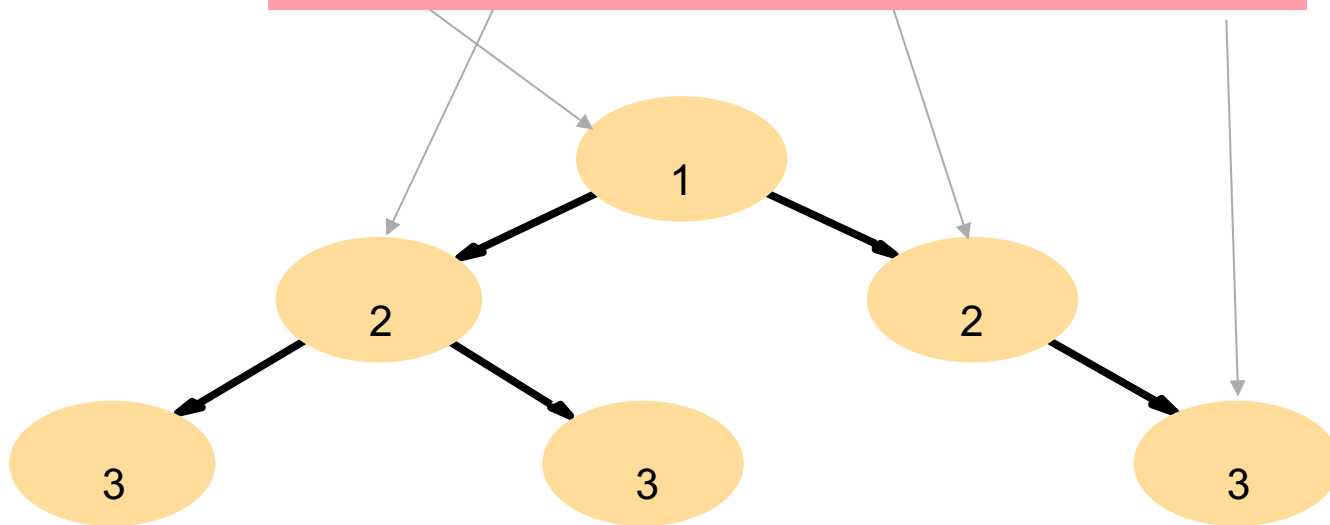
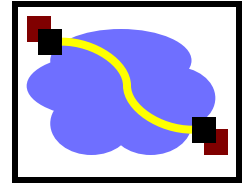


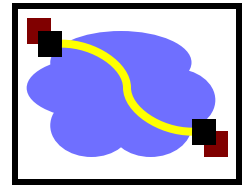
Figure 10.3

# The Network Time Protocol (NTP)



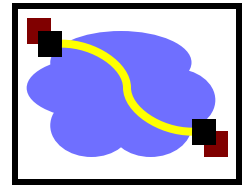
- Uses a hierarchy of time servers
  - Class 1 servers have highly-accurate clocks
    - connected directly to atomic clocks, etc.
  - Class 2 servers get time from only Class 1 and Class 2 servers
  - Class 3 servers get time from any server (usually 3)
- Synchronization similar to Cristian's alg.
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

# NTP Reference Clock Sources (1997 survey)



- In a survey of 36,479 peers, found 1,733 primary and backup external reference sources
- 231 radio/satellite/modem primary sources
  - 47 GPS satellite (worldwide), GOES satellite (western hemisphere)
  - 57 WWVB radio (US)
  - 17 WWV radio (US)
  - 63 DCF77 radio (Europe)
  - 6 MSF radio (UK)
  - 5 CHU radio (Canada)
  - 7 modem time service (NIST and USNO (US), PTB (Germany), NPL (UK))
  - 25 other (precision PPS sources, etc.)
- 1,502 local clock backup sources (used only if all other sources fail)
- For some reason or other, 88 of the 1,733 sources appeared down at the time of the survey

# U. Delaware Master Time Facility (MTF) (from January 2000, an original!)



Spectracom 8170 WWVB Receiver

Spectracom 8183 GPS Receiver

Spectracom 8170 WWVB Receiver

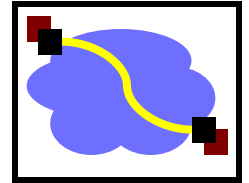
Spectracom 8183 GPS Receiver

Hewlett Packard 105A Quartz  
Frequency Standard

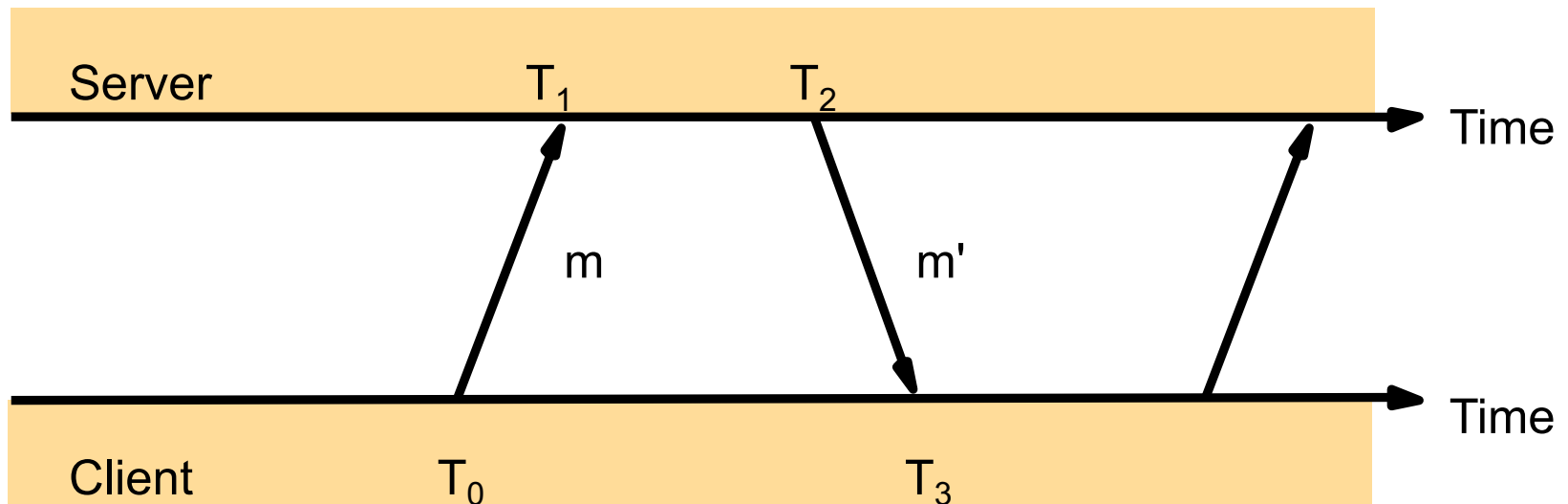
Hewlett Packard 5061A Cesium Beam  
Frequency Standard



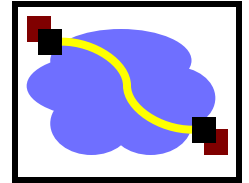
# NTP Protocol



- All modes use UDP
- Each message bears timestamps of recent events:
  - Local times of Send and Receive of previous message
  - Local times of Send of current message
- Recipient notes the time of receipt  $T_3$  (we have  $T_0$ ,  $T_1$ ,  $T_2$ ,  $T_3$ )

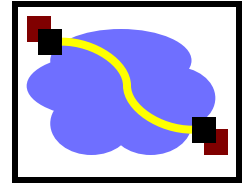


# Accuracy of NTP



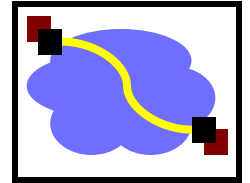
- Timestamps
  - $t_0$  is the client's timestamp of the request packet transmission,
  - $t_1$  is the server's timestamp of the request packet reception,
  - $t_2$  is the server's timestamp of the response packet transmission and
  - $t_3$  is the client's timestamp of the response packet reception.
- RTT = wait\_time\_client – server\_proc\_time  
=  $(t_3 - t_0) - (t_2 - t_1)$
- Offset =  $((t_1 - t_0) + (t_2 - t_3)) / 2$
- NTP servers filter pairs  $\langle rtt_i, offset_i \rangle$ , estimating reliability from variation, allowing them to select peers
- Accuracy of tens of millisecs over Internet paths (1ms on LANs)

# How To Change Time



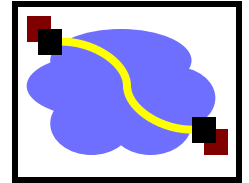
- Can't just change time
  - Why not?

# 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