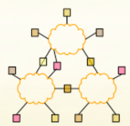
15-446 Distributed Systems Spring 2009



L-6 Synchronizing Physical Clocks

1

Last Lecture – RPC Important Lessons

- Procedure calls
  - Simple way to pass control and data
  - Elegant transparent way to distribute application
  - Not only way…
- Hard to provide true transparency
  - Failures
  - Performance
  - Memory access
  - Etc.
- How to deal with hard problem → give up and let programmer deal with it
  - "Worse is better"

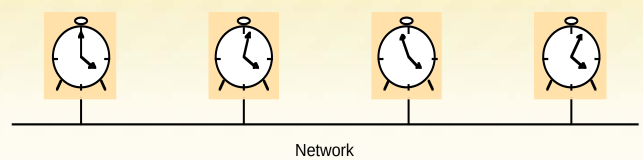
2

Today's Lecture

- Need for time synchronization
- Time synchronization techniques
- Lamport

3

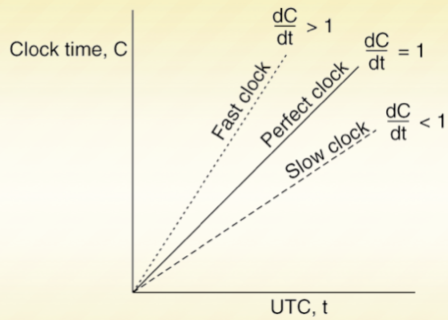
Clocks in a Distributed System



- 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)
  - **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

4

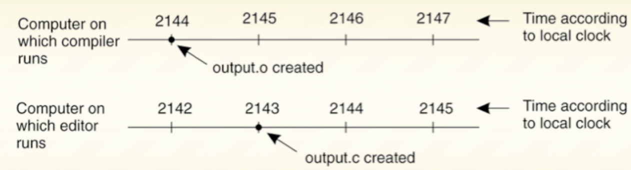
Clock Synchronization Algorithms



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

5

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.

6

Need for Precision Time

- Distributed database transaction journalling and logging
- Stock market buy and sell orders
- Secure document timestamps (with cryptographic certification)
- Aviation traffic control and position reporting
- Radio and TV programming launch and monitoring
- Intruder detection, location and reporting
- Multimedia synchronization for real-time teleconferencing
- Interactive simulation event synchronization and ordering
- Network monitoring, measurement and control
- Early detection of failing network infrastructure devices and air conditioning equipment
- Differentiated services traffic engineering
- Distributed network gaming and training

7

Coordinated Universal Time (UTC)

- International Atomic Time is based on very accurate physical clocks (drift rate 10^{-13})
- UTC is an international standard for time keeping
- It is based on atomic time, but occasionally adjusted to astronomical time
- It 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 put GPS receivers on all our computers?

8

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

9

Udel Master Time Facility (MTF) (from January 2000)

Spectracom 8170 WWVB Receiver

Spectracom 8183 GPS Receiver

Spectracom 8170 WWVB Receiver

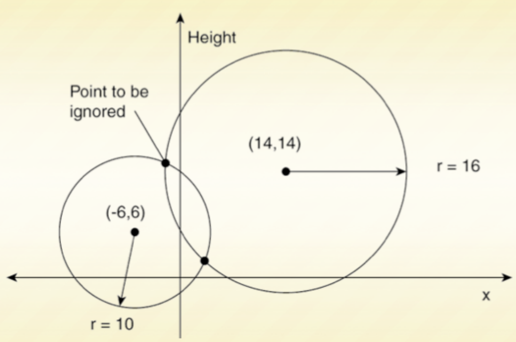
Spectracom 8183 GPS Receiver

Hewlett Packard 105A Quartz Frequency Standard

Hewlett Packard 5061A Cesium Beam Frequency Standard

10

Global Positioning System (1)



- Computing a position in a two-dimensional space.

11

Global Positioning System (2)

Real world facts that complicate GPS

- It takes a while before data on a satellite's position reaches the receiver.
- The receiver's clock is generally not in synch with that of a satellite.

12

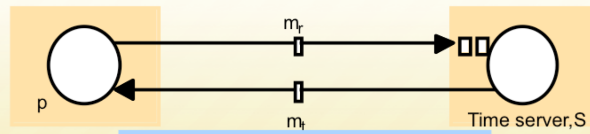
Today's Lecture

- Need for time synchronization
- Time synchronization techniques
- Lamport

13

Cristian's Time Sync

- 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 $t + T_{round}/2$
  - Accuracy $\pm (T_{round}/2 - min)$:
    - 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 $[t+min, t + T_{round} - min]$



T_{round} is the round trip time recorded by *p*
min is an estimated minimum round trip time

14

Network Time Protocol (NTP)

- A time service for the Internet - synchronizes

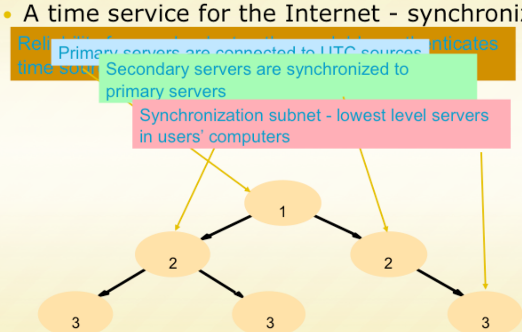
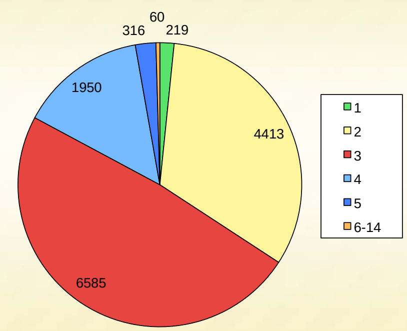


Figure 10.3

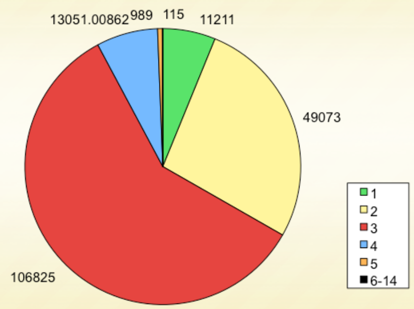
15

Server population by stratum (1997 survey)



16

Client population by stratum (1997 survey)



17

NTP - synchronisation of servers

- The synchronization subnet can reconfigure if failures occur, e.g.
  - a primary that loses its UTC source can become a secondary
  - a secondary that loses its primary can use another primary

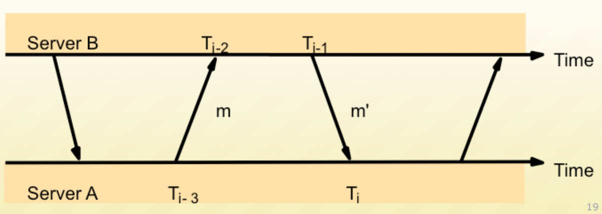
Modes of synchronization:

- Multicast
  - A server within a high speed LAN multicasts time to others which set clocks assuming some delay (not very accurate)
- Procedure call
  - A server accepts requests from other computers (like Cristiain's algorithm). Higher accuracy. Useful if no hardware multicast.
- Symmetric
  - Pairs of servers exchange messages containing time information
  - Used where very high accuracies are needed (e.g. for higher levels)

18

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_i (we have T_{i-3}, T_{i-2}, T_{i-1}, T_i)
- In symmetric mode there can be a non-negligible delay between messages



19

Accuracy of NTP

- For each pair of messages between two servers, NTP estimates an offset o, between the two clocks and a delay d_i (total time for the two messages, which take t and t')
  $T_{i-2} = T_{i-3} + t + o$ and $T_i = T_{i-1} + t' - o$
- This gives us (by adding the equations) :
  $d_i = t + t' = T_{i-2} - T_{i-3} + T_i - T_{i-1}$
- Also (by subtracting the equations)
  $o = o_i + (t' - t)/2$ where $o_i = (T_{i-2} - T_{i-3} + T_{i-1} - T_i)/2$
- Using the fact that t, $t'>0$ it can be shown that
  $o_i - d_i/2 \leq o \leq o_i + d_i/2$.
  - Thus o_i is an estimate of the offset and d_i is a measure of the accuracy
- NTP servers filter pairs <o_i, d_i>, estimating reliability from variation, allowing them to select peers
- Accuracy of 10s of millisecs over Internet paths (1 on LANs)

20

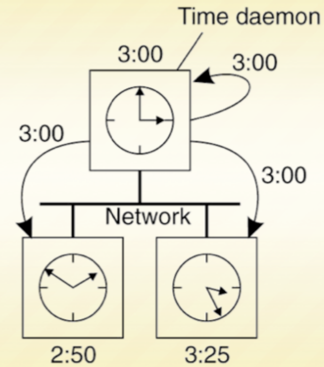
Berkeley algorithm

- 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 some 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)
  - Measurements
    - 15 computers, clock synchronization 20-25 millisecs drift rate < $2x10^{-5}$
    - If master fails, can elect a new master to take over (not in bounded time)

21

The Berkeley Algorithm (1)

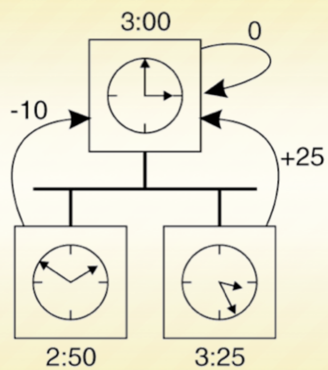
- The time daemon asks all the other machines for their clock values.



22

The Berkeley Algorithm (2)

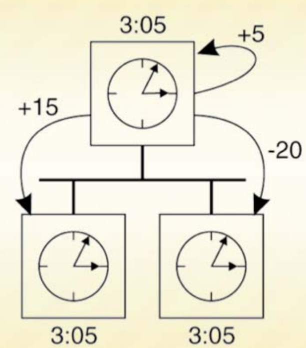
- The machines answer.



23

The Berkeley Algorithm (3)

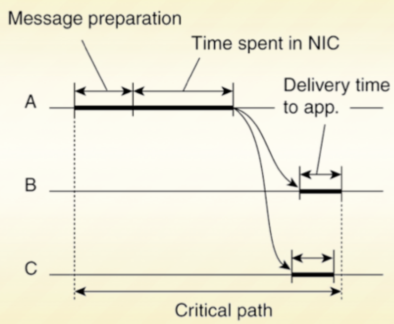
- The time daemon tells everyone how to adjust their clock.



24

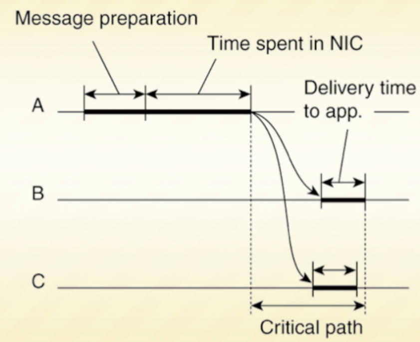
Clock Synchronization in Wireless Networks (1)

- The usual critical path in determining network delays.



Clock Synchronization in Wireless Networks (2)

- The critical path in the case of RBS.



Today's Lecture

- Need for time synchronization
- Time synchronization techniques
- Lamport

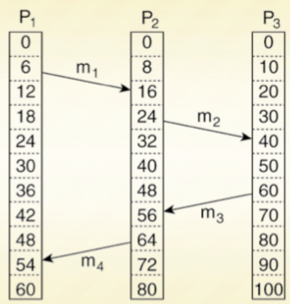
Lamport's Logical Clocks (1)

The "happens-before" relation → can be observed directly in two situations:

- If a and b are events in the same process, and a occurs before b, then a → b is true.
- If a is the event of a message being sent by one process, and b is the event of the message being received by another process, then a → b

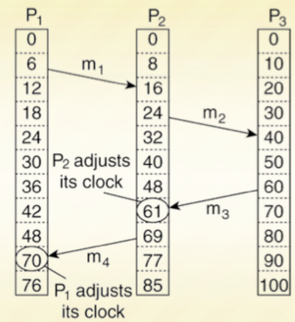
Lamport's Logical Clocks (2)

- Three processes, each with its own clock. The clocks run at different rates.



29

Lamport's Logical Clocks (3)

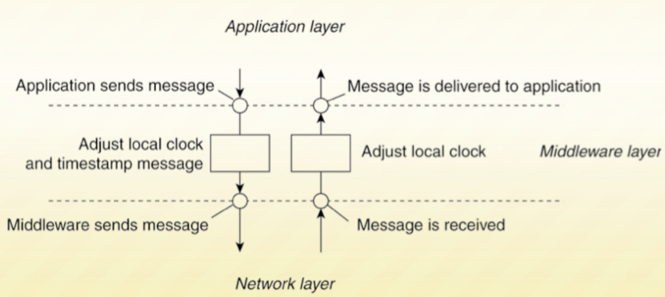


- Lamport's algorithm corrects the clocks.

30

Lamport's Logical Clocks (4)

- The positioning of Lamport's logical clocks in distributed systems.



31

Lamport's Logical Clocks (5)

- Updating counter C_i for process P_i

1. Before executing an event P_i executes $C_i \leftarrow C_i + 1$.
2. When process P_i sends a message m to P_j, it sets *m*'s timestamp *ts (m)* equal to C_i after having executed the previous step.
3. Upon the receipt of a message *m*, process P_j adjusts its own local counter as $C_j \leftarrow \max\{C_j, ts(m)\}$, after which it then executes the first step and delivers the message to the application.

32

Important Lessons

- Clocks on different systems will always behave differently
  - Skew and drift between clocks
- Time disagreement between machines can result in undesirable behavior
- Two paths to solution: synchronize clocks or ensure consistent clocks
- Clock synchronization
  - Rely on a time-stamped network messages
  - Estimate delay for message transmission
  - Can synchronize to UTC or to local source

33