Last Class: Naming

- Naming
  - Distributed naming
  - DNS
  - LDAP

DNS Implementation

- An excerpt from the DNS database for the zone *cs.vu.nl*.

Name	Record type	Record value
cs.vu.nl	SOA	star (1999121502,7200,3600,2419200,86400)
cs.vu.nl	NS	star.cs.vu.nl
cs.vu.nl	NS	top.cs.vu.nl
cs.vu.nl	NS	solo.cs.vu.nl
cs.vu.nl	TXT	"Vrije Universiteit - Math. & Comp. Sc."
cs.vu.nl	MX	1 zephyr.cs.vu.nl
cs.vu.nl	MX	2 tornado.cs.vu.nl
cs.vu.nl	MX	3 star.cs.vu.nl
star.cs.vu.nl	HINFO	Sun Unix
star.cs.vu.nl	MX	1 star.cs.vu.nl
star.cs.vu.nl	MX	10 zephyr.cs.vu.nl
star.cs.vu.nl	A	130.37.24.6
star.cs.vu.nl	A	192.31.231.42
zephyr.cs.vu.nl	HINFO	Sun Unix
zephyr.cs.vu.nl	MX	1 zephyr.cs.vu.nl
zephyr.cs.vu.nl	MX	2 tornado.cs.vu.nl
zephyr.cs.vu.nl	A	192.31.231.66
www.cs.vu.nl	CNAME	soling.cs.vu.nl
ftp.cs.vu.nl	CNAME	soling.cs.vu.nl
soling.cs.vu.nl	HINFO	Sun Unix
soling.cs.vu.nl	MX	1 soling.cs.vu.nl
soling.cs.vu.nl	MX	10 zephyr.cs.vu.nl
soling.cs.vu.nl	A	130.37.24.11
laser.cs.vu.nl	HINFO	PC MS-DOS
laser.cs.vu.nl	A	130.37.30.32
vucs-das.cs.vu.nl	PTR	0.26.37.130.in-addr.arpa
vucs-das.cs.vu.nl	A	130.37.26.0

X.500 Directory Service

- OSI Standard
- Directory service: special kind of naming service where:
  - Clients can lookup entities based on attributes instead of full name
  - Real-world example: Yellow pages: look for a plumber

LDAP

- Lightweight Directory Access Protocol (LDAP)
  - X.500 too complex for many applications
  - LDAP: Simplified version of X.500
  - Widely used for Internet services
  - Application-level protocol, uses TCP
  - Lookups and updates can use strings instead of OSI encoding
  - Use master servers and replicas servers for performance improvements
  - Example LDAP implementations:
    - Active Directory (Windows 2000)
    - Novell Directory services
    - iPlanet directory services (Netscape)
    - OpenLDAP
    - Typical uses: user profiles, access privileges, network resources

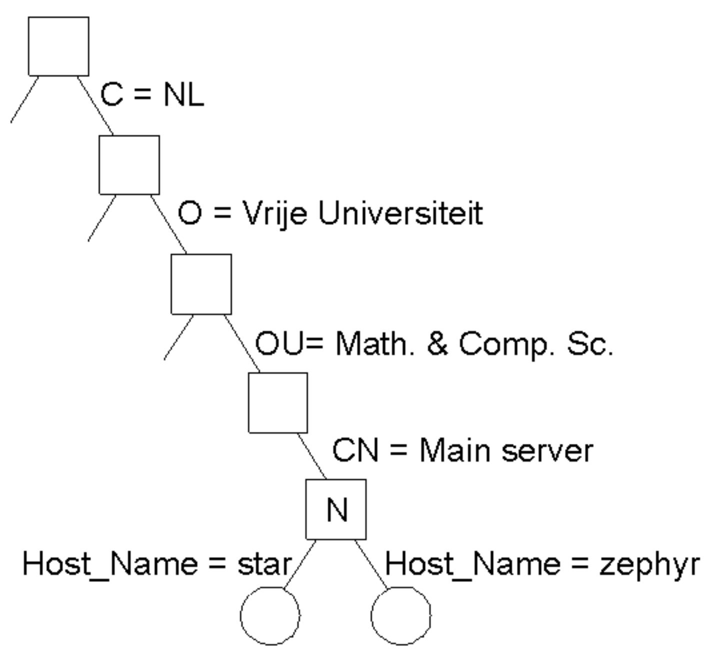
The LDAP Name Space

Attribute	Abbr.	Value
Country	C	NL
Locality	L	Amsterdam
Organization	L	Vrije Universiteit
OrganizationalUnit	OU	Math. & Comp. Sc.
CommonName	CN	Main server
Mail_Servers	--	130.37.24.6, 192.31.231,192.31.231.66
FTP_Server	--	130.37.21.11
WWW_Server	--	130.37.21.11

- A simple example of a LDAP directory entry using X.500 naming conventions.

The LDAP Name Space (2)

- Part of the directory information tree.

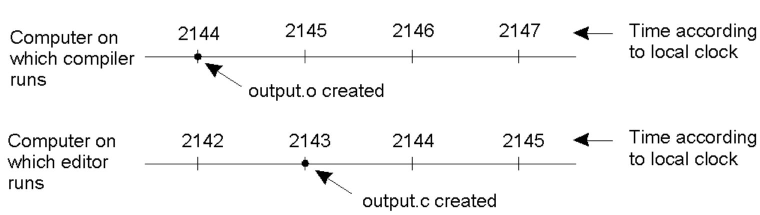


Today: Canonical Problems in Distributed Systems

- Time ordering and clock synchronization
- Leader election
- Mutual exclusion
- Distributed transactions
- Deadlock detection

Clock Synchronization

- Time in unambiguous in centralized systems
  - System clock keeps time, all entities use this for time
- Distributed systems: each node has own system clock
  - Crystal-based clocks are less accurate (1 part in million)
  - *Problem:* An event that occurred after another may be assigned an earlier time

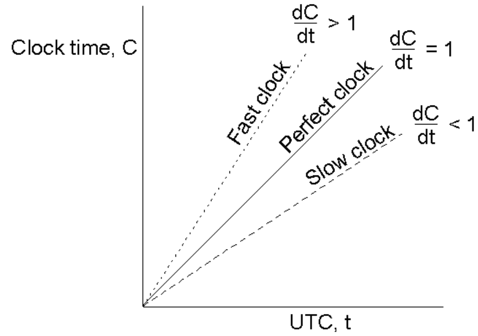


Physical Clocks: A Primer

- Accurate clocks are atomic oscillators (one part in 10^{13})
- Most clocks are less accurate (e.g., mechanical watches)
  - Computers use crystal-based blocks (one part in million)
  - Results in *clock drift*
- How do you tell time?
  - Use astronomical metrics (solar day)
- Coordinated universal time *(UTC)* – international standard based on atomic time
  - Add leap seconds to be consistent with astronomical time
  - UTC broadcast on radio (satellite and earth)
  - Receivers accurate to 0.1 – 10 ms
- Need to synchronize machines with a master or with one another

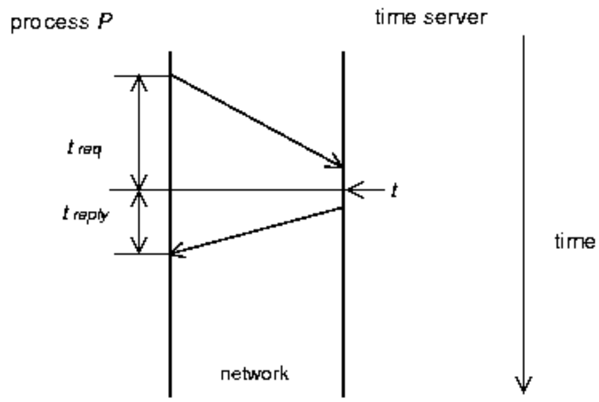
Clock Synchronization

- Each clock has a maximum drift rate ρ
  - $1-\rho <= dC/dt <= 1+\rho$
  - Two clocks may drift by $2\rho\ \Delta t$ in time Δt
  - To limit drift to δ => resynchronize every $\delta/2\rho$ seconds



Cristian's Algorithm

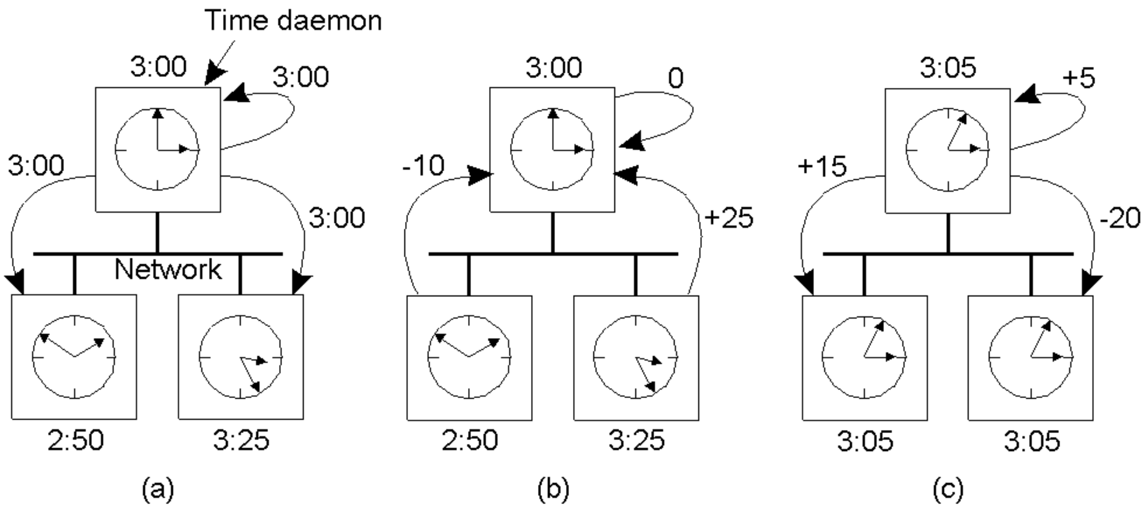
- Synchronize machines to a *time server* with a UTC receiver
- Machine P requests time from server every $\delta/2\rho$ seconds
  - Receives time t from server, P sets clock to $t+t_{reply}$ where t_{reply} is the time to send reply to P
  - Use $(t_{req}+t_{reply})/2$ as an estimate of t_{reply}
  - Improve accuracy by making a series of measurements



Berkeley Algorithm

- Used in systems without UTC receiver
  - Keep clocks synchronized with one another
  - One computer is *master*, other are *slaves*
  - Master periodically polls slaves for their times
    - Average times and return differences to slaves
    - Communication delays compensated as in Cristian's algo
  - Failure of master => election of a new master

Berkeley Algorithm

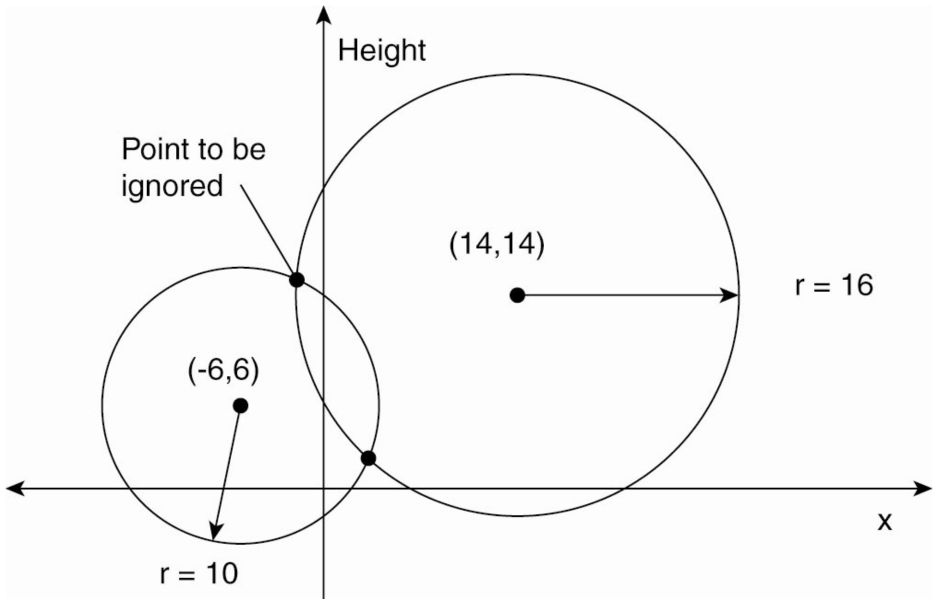


a) The time daemon asks all the other machines for their clock values
b) The machines answer
c) The time daemon tells everyone how to adjust their clock

Distributed Approaches

- Both approaches studied thus far are centralized
- Decentralized algorithms: use resync intervals
  - Broadcast time at the start of the interval
  - Collect all other broadcast that arrive in a period *S*
  - Use average value of all reported times
  - Can throw away few highest and lowest values
- Approaches in use today
  - *rdate*: synchronizes a machine with a specified machine
  - Network Time Protocol (NTP) - discussed in a later slide
    - Uses advanced techniques for accuracies of 1-50 ms

Global Positioning System



- Computing a position in a two-dimensional space.

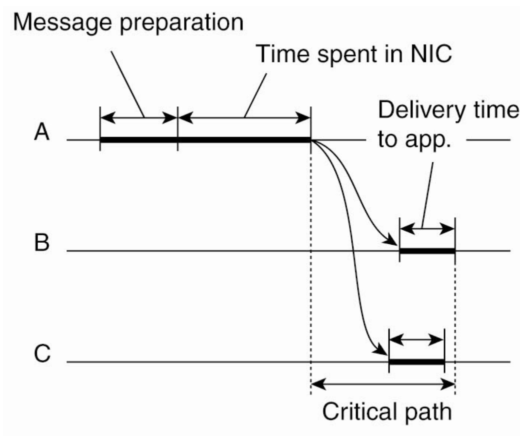
Global Positioning System

- 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.

GPS Basics

- D_r – deviation of receiver from actual time
- Beacon with timestamp T_i received at T_{now}
  - Delay $D_i = (T_{now} - T_i) + D_r$
  - Distance $d_i = c\,(T_{now} - T_i)$
  - Also $d_i = \text{sqrt}[\,(x_i-x_r)^2 + (y_i-y_r)^2 + (z_i-z_r)^2\,]$
- Four unknowns, need 4 satellites.
-

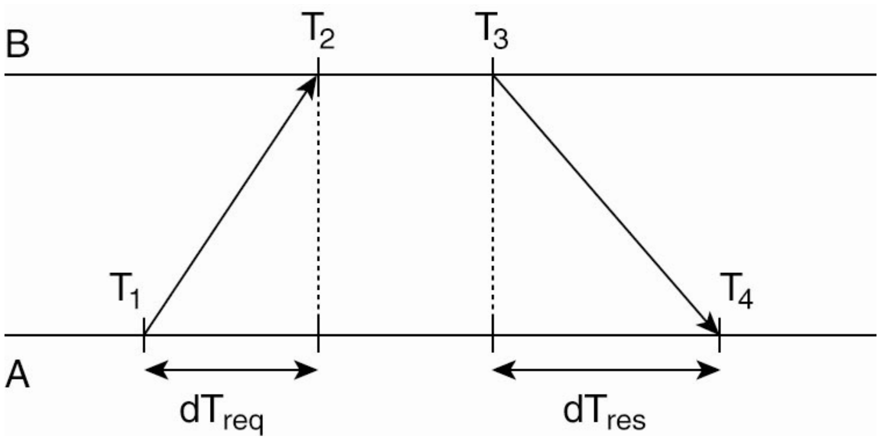
Clock Synchronization in Wireless Networks



(b)

- Reference broadcast sync (RBS): receivers synchronize with one another using RB server
  - Mutual offset = $T_{i,s} - T_{j,s}$ (can average over multiple readings)

Network Time Protocol



- Widely used standard - based on Cristian's algo
  - Uses eight pairs of delays from A to B and B to A.
- Hierarchical – uses notion of stratum
- Clock can not go backward

Logical Clocks

- For many problems, internal consistency of clocks is important
  - Absolute time is less important
  - Use *logical* clocks
- Key idea:
  - Clock synchronization need not be absolute
  - If two machines do not interact, no need to synchronize them
  - More importantly, processes need to agree on the *order* in which events occur rather than the *time* at which they occurred