



THE UNIVERSITY *of* EDINBURGH
informatics

Distributed Systems

Time, Clocks, and Ordering of Events

Björn Franke

University of Edinburgh
2016/2017

Today

- Last lecture: Basic Algorithms
- Today:
 - Time, clocks, NTP
 - Ref: CDK
 - Causality, ordering, logical clocks:
 - Ref: VG, CDK

Time

- Ordering of events are important:
 - Which happened first
- Need synchronisation between sender and receiver
- Coordination of joint activity etc..

Coordinated Universal Time

- Coordinated universal time (UTC)
 - Time maintained for civil use (on atomic clock)
 - Kept within 0.9 seconds of exact mean time for Greenwich

Clocks

- Piezoelectric effect:
 - Squeeze a quartz crystal:
generates electric field
 - Apply electric field: crystal bends
- Quartz crystal clock:
 - Resonation like a tuning fork
 - Accurate to parts per million
 - Gain/lose $\frac{1}{2}$ second per day

Challenges

- Two clocks do not agree perfectly
- **Skew:** The time difference between two clocks
- Quartz oscillators vibrate at different rates
- **Drift:** The difference in rates of two clocks
- If we had two perfect clocks:
 - Skew = 0
 - Drift = 0

When we detect a clock has a skew

- Eg: it is 5 seconds behind
- Or 5 seconds ahead

- What can we do?

When we detect a clock has a skew

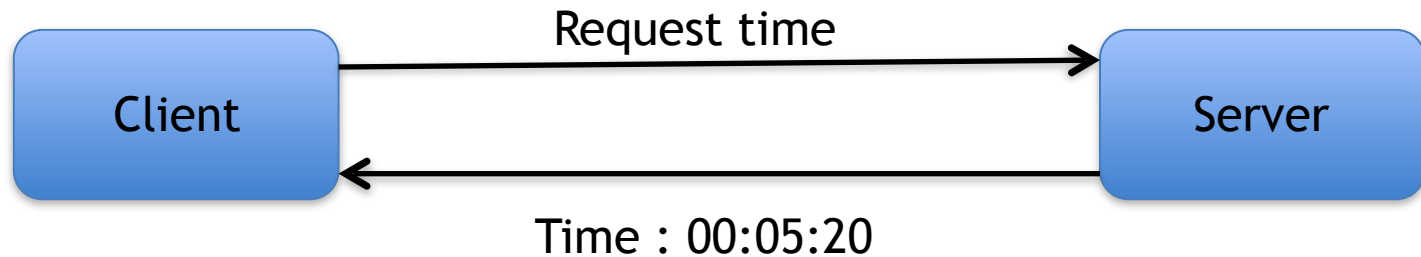
- e.g. it is 5 seconds behind
 - We can advance it 5 seconds to correct
 - Might skip over event scheduled in-between
- Or 5 seconds ahead
 - Pushing back 5 seconds is a bad idea
 - Message was received before it was sent
 - Document closed before it was saved etc..
 - We want **monotonicity**: time always increases
 - We want **continuity**: time doesn't make jumps

When we detect a clock has a skew

- e.g. it is behind
 - Run it faster until it catches up
- It is ahead
 - Run it slower until it catches up
- This does not guarantee correct clock in future
 - Need to check and adjust periodically

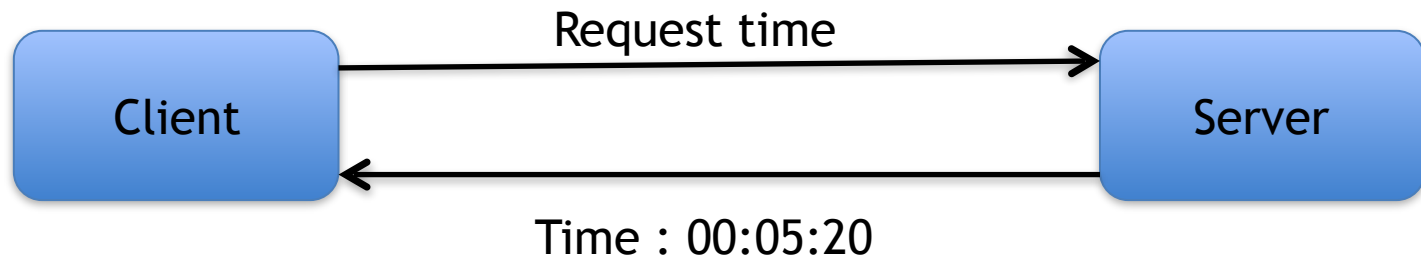
How clocks synchronise

- Obtain time from time server:



How clocks synchronise

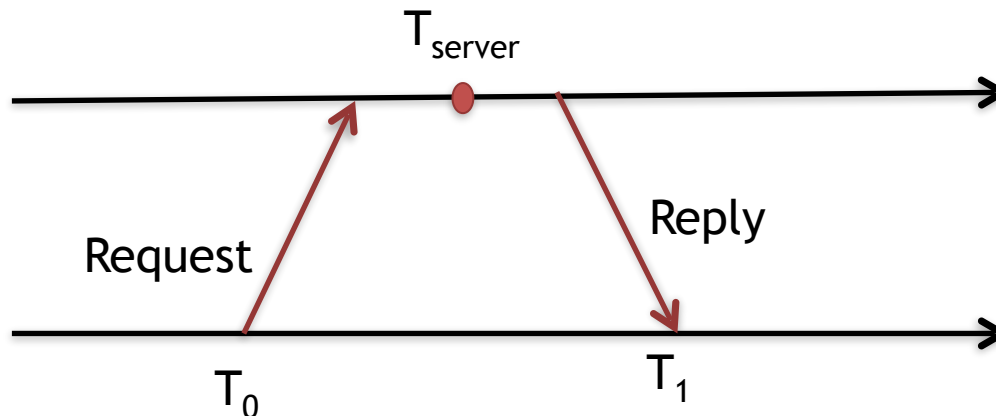
- Obtain time from time server:



- Time is inaccurate
 - Delays in message transmission
 - Delays due to processing time
 - Server's time may be inaccurate

Christian's algorithm

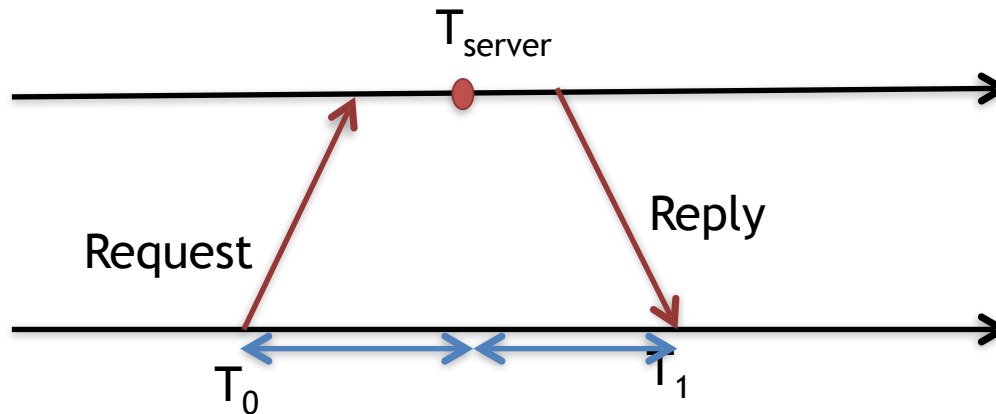
- Compensate for delays
 - Request sent at T_0
 - Reply received at T_1



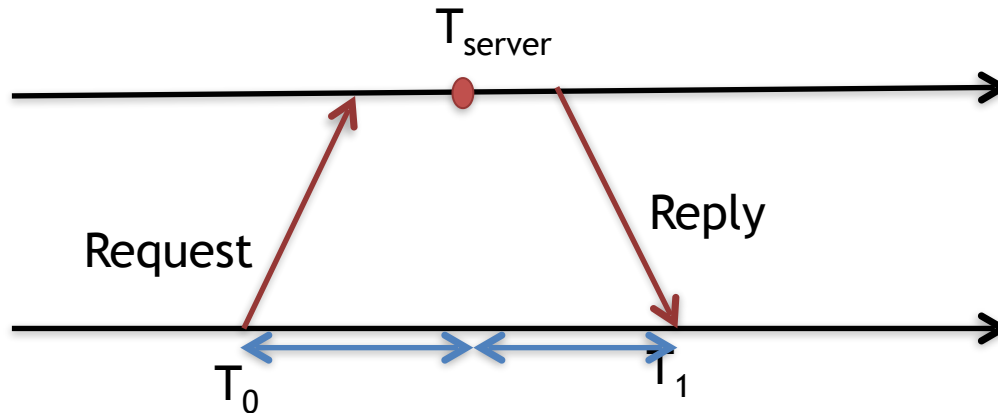
- Assume delays are symmetric

Christian's algorithm

$$T_{\text{new}} = T_{\text{server}} + (T_1 - T_0)/2$$



Christian's algorithm



$$T_{\text{new}} = T_{\text{server}} + (T_1 - T_0)/2$$

Example:

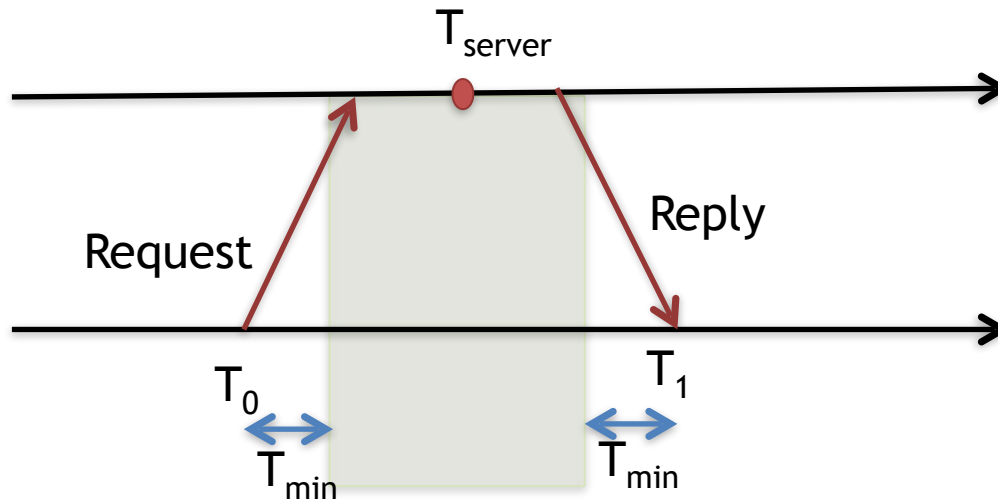
$$T_0 = 5:05:08.100, T_1 = 5:05:9.500$$

$$T_{\text{server}} = 5:05:9.100$$

$$T_{\text{new}} = 5:05:09:800$$

Christian's algorithm

- If minimum message transit time T_{\min} is known
- Range = $T_1 - T_0 - 2T_{\min}$
- Accuracy of result: $(T_1 - T_0 - 2T_{\min})/2$



Berkeley algorithm

- Assumes no machine has perfect time
- Takes average of participating computers
- Sync all clocks to average

Berkeley algorithm

- One computer is elected as server (master)
 - Others are slaves
- Master polls each machine for time
- Compute average
 - Key idea: average will cancel out skews
- Send each clock the offset by which it needs to adjust time
 - Sending time itself is susceptible to network delays

Berkeley algorithm

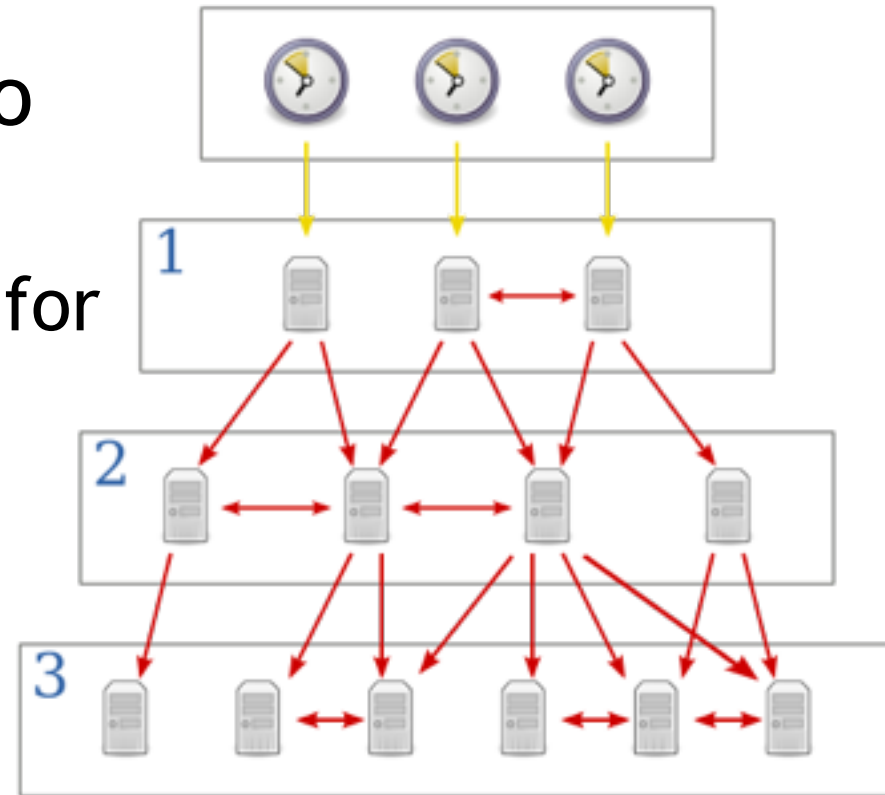
- Fault tolerance
 - Ignore readings of clocks with too large skews
 - If master fails: run an election algorithm and a slave becomes master

Network time protocol (NTP)

- Enable clients to synchronise to UTC with reasonable accuracy
- Reliable:
 - Redundant servers and paths
- Scalable:
 - Enable many clients to synchronise frequently
- Security
 - Authenticate sources

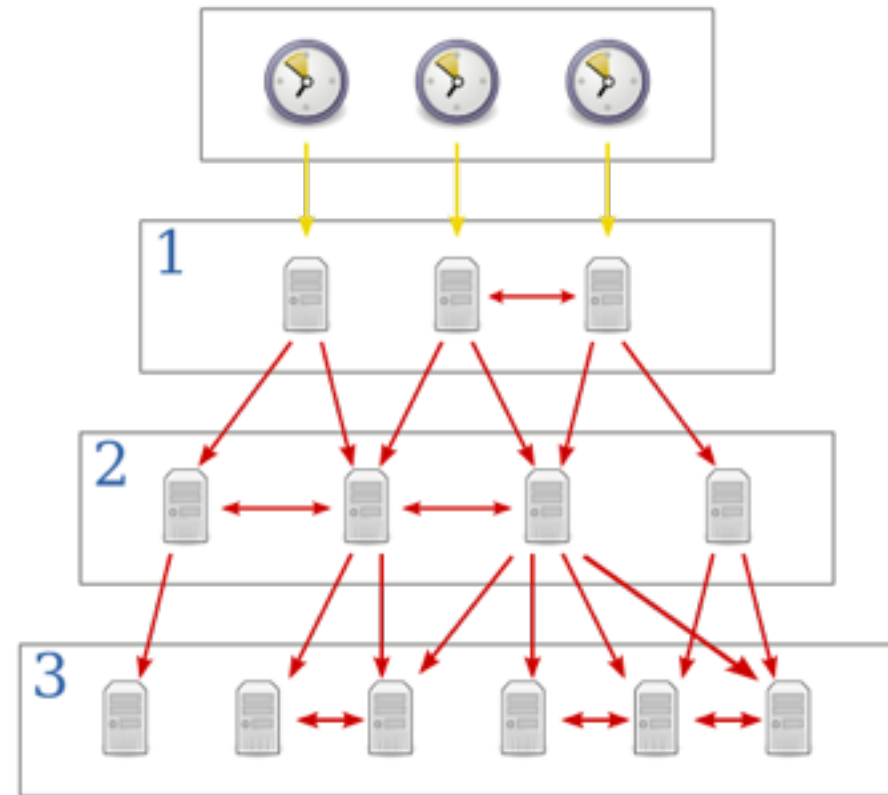
Network time protocol

- Servers in strata
- 1: directly connected to atomic, GPS etc clock
 - May inter-communicate for cross checks
- 2: few microseconds of level 1 etc



Network time protocol

- Uses multiple rounds of messages to get better time
- Large number of servers
- Uses a multiple spanning tree (MST) for inter-server synchronisation

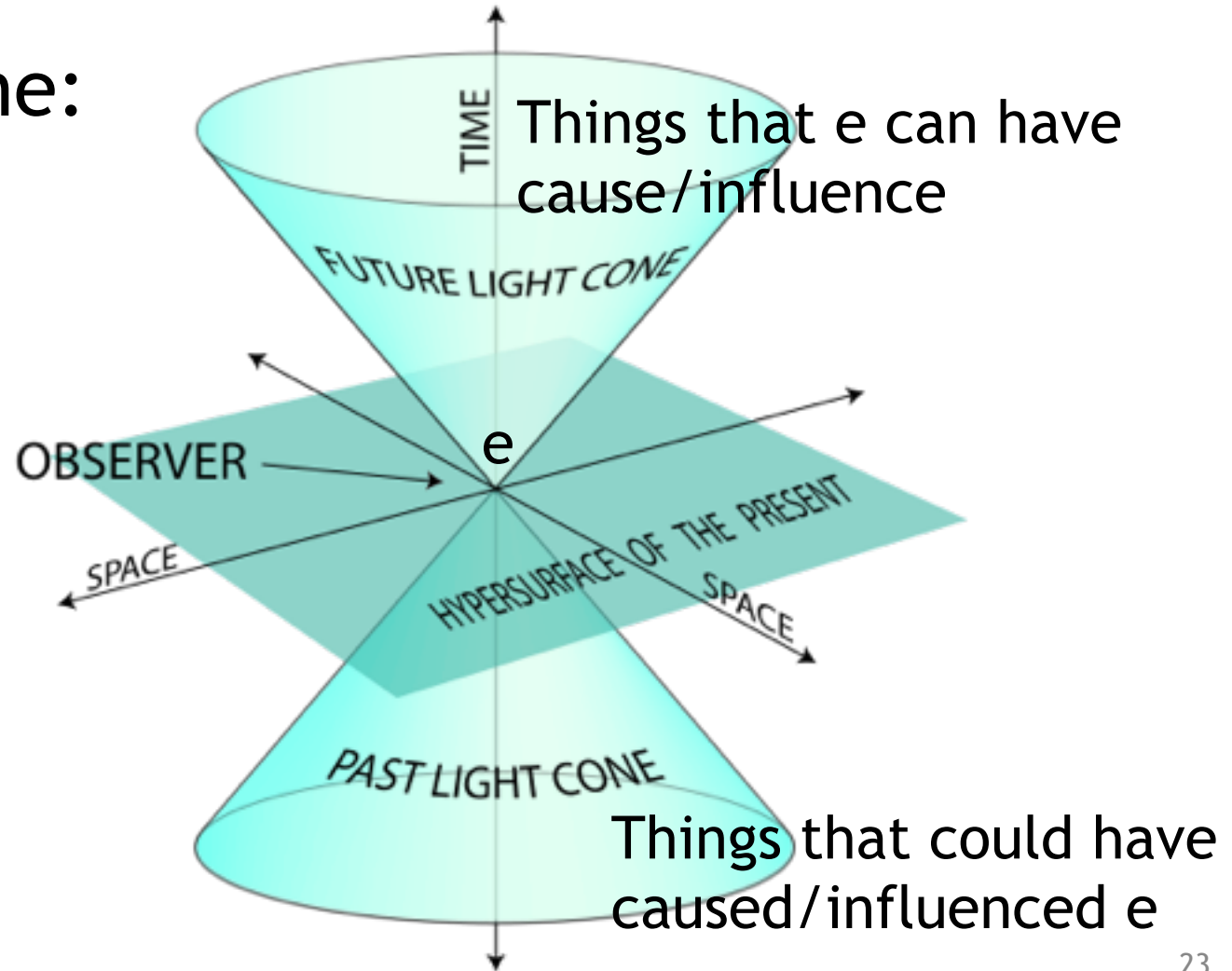


Time and Synchronisation

- Important topic in distributed systems
- Many different methods
 - Depending on systems, requirements...
- No perfect solution

Special Relativity

- Light cone:



GPS

- Satellites: Have very accurate atomic clocks
- Transmit signals: “satID, time T_0, \dots ”
- Receivers measure distance:
 - $(T_1 - T_0) * c$ [c = speed of light]
 - Distance from multiple satellites gives location
 - Complex computation, taking into account possible errors, clock drift and skew etc..
- Needs relativistic computation
 - Special relativity: Clocks on fast moving satellites run slow (microseconds per day drift for satellites)
 - General relativity: Clocks far from heavy bodies run fast (microseconds per day)