Synchronizing Clocks (Wallclock Time)

Richard M. Fujimoto
Professor

Computational Science and Engineering Division
College of Computing
Georgia Institute of Technology
Atlanta, GA 30332-0765, USA

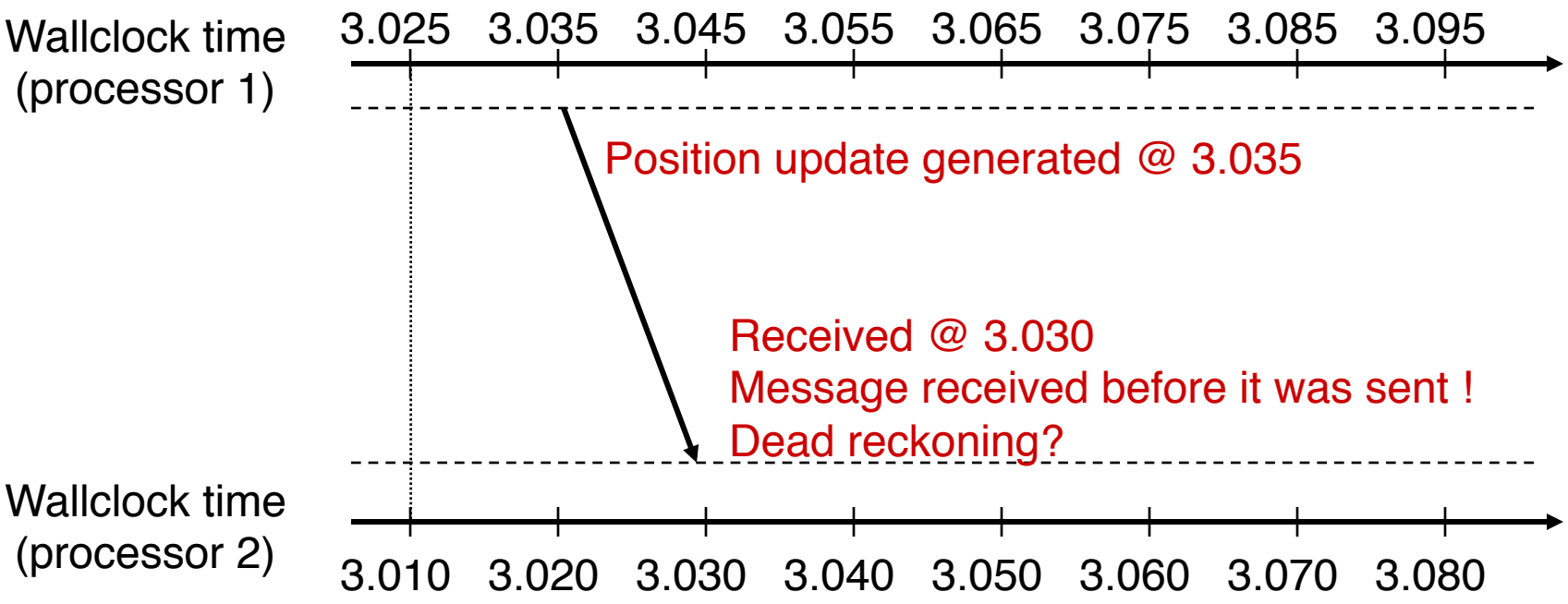
http://www.cc.gatech.edu/~fujimoto/

Outline

- Clock Synchronization Problem
- Wallclock Time: Definitions
- Synchronization algorithms
- Correcting clock errors

Problems With Unsynchronized Clocks

- Each processor has a *local* oscillator providing wallclock time values
- At any instant clocks in distinct processors will differ
- Clocks in different processors drift relative to each other



A mechanism to synchronize hardware clocks is needed.

Time Definitions

- Historically, time definitions based on periodic astronomical events
  - Solar day: time duration that elapses between event where the sun is at the highest point in the sky
  - 24 hours per solar day, 3600 seconds per hour
  - Greenwich mean time: solar time at the Greenwich meridian (observatory near London)
- Today, atomic clocks define standard time sources
  - Time duration based on transitions of cesium atom

Time Definitions (cont.)

- Earth's rotation slowing down slightly
- Duration of a "day" is increasing
- Slowing down effect not accounted for by atomic clocks
- Solution: leap seconds are periodically added to time produced by cesium clock
  - Resulting time called Coordinated Universal Time (UTC)
  - Time standard accepted world wide
  - Disseminated by standards organizations
    - National Institute of Standards and Technology (NIST) in U.S.
    - Available via satellites, radio broadcasts, phones

Clock Synchronization Problems

- Ensuring different processors maintain clocks that are synchronized with each other
  - δ = maximum offset allowed between two clocks
- Ensuring that clocks are synchronized with UTC
  - Only an issue if interacting with UTC-based devices

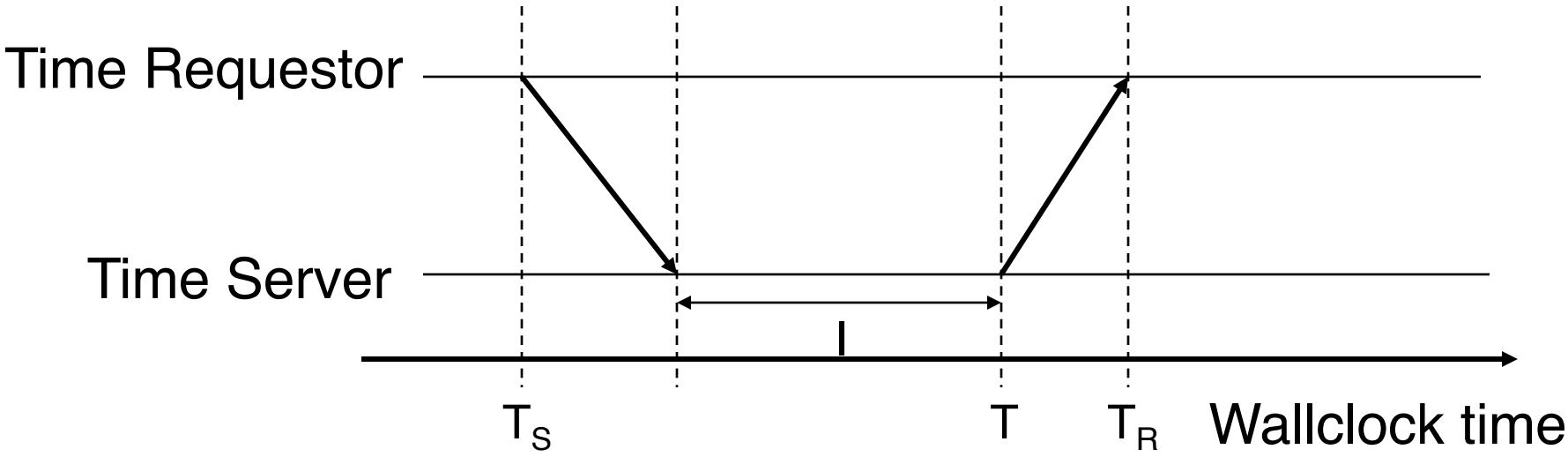
Synchronized Wallclock Time

- Notation
  - C_i = hardware clock in processor i
  - T = UTC
  - Max clock drift ρ: $1-\rho \leq dC_i/dT \leq 1+\rho$
  - Clocks in two processor may drift 2ρ per unit UTC
- Processors must periodically resynchronize their clocks
- How often?
  - δ = maximum offset allowed between two clocks
  - Clocks can drift δ time units in $\delta/2\rho$ time
  - Resynchronization must be done at least every $\delta/2\rho$ time units

Clock Synchronization Algorithms

- Centralized, pull algorithms
  - Each processor periodically requests current time from a central time server
  - Server reads local clock and returns current time value
- Centralized, push algorithms
  - Central time server periodically sends current time to other processors
- Distributed algorithms
  - No central time server
  - Processors use distributed algorithm to synchronize their local clock
- Basic problem: communication latency time is unpredictable

Centralized, Pull Algorithm



- T_S = local time in requestor when request sent
- I = interrupt service time
- T = time value provided by server
- T_R = local time in requestor when response received
- $T_R = T + L$ where L is communication latency
- Estimate $L = (T_R - T_S - I) / 2$
- Collect several estimates of L, discard outliers, average
- Latency often symmetric in LANs, typically not in WANs

Push and Distributed Algorithms

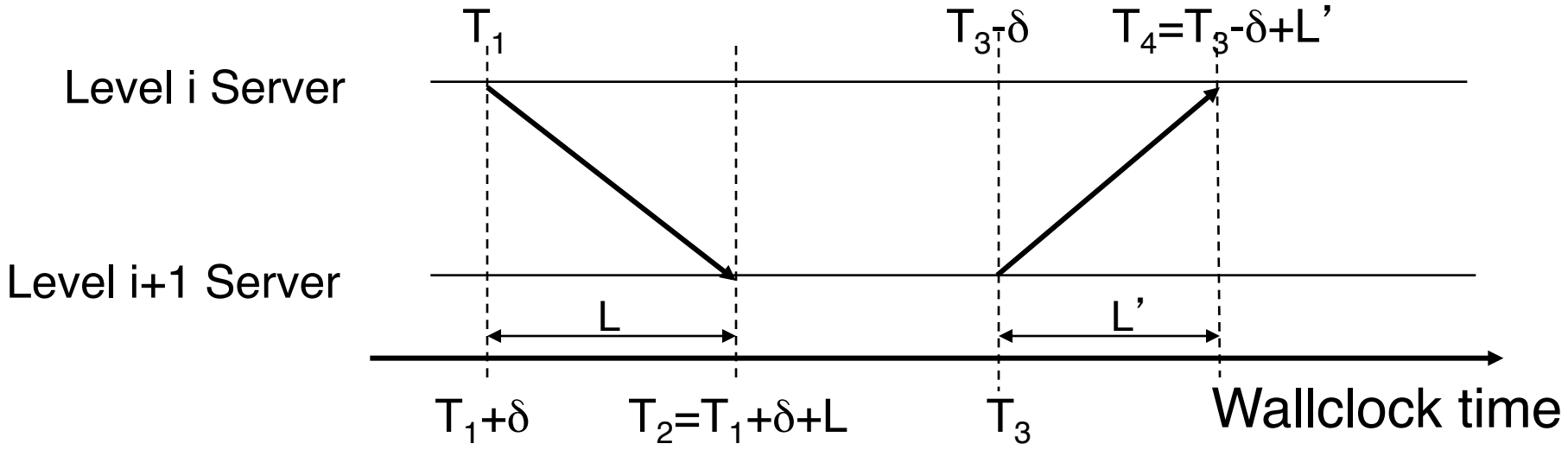
- Push algorithms
  - Server periodically broadcasts time values
  - Requires assumption of message latency
- Distributed algorithms
  - Avoids single point of failure, bottleneck for large systems
  - A push algorithm
    - Each processor periodically sends time values to a subset of the other processors
    - Incoming time values: discard outliers, average received values

Network Time Protocol

- Distributed clock service implemented on Internet
- Hierarchical subnetwork of clock servers
  - Clock information flows down hierarchy
  - Servers at level i synchronize with servers at level i+1
- Primary servers
  - Level 1 of tree
  - Synchronized with national time servers
- Secondary servers
  - Level 2, 3, …

NTP Latency and Offset Estimation

- Four time stamps stored in message: T_1, T_2, T_3, T_4
- δ = offset between clocks (unknown)
- L, L' = message latency (send and reply)



- $A = T_2 - T_1 = \delta + L > \delta$; A is an upper bound on δ
- $B = T_3 - T_4 = \delta - L' < \delta$; B is a lower bound on δ
- Estimate offset as (A+B)/2
- Round trip delay = L+L' = A –B (max estimate error for δ)

NTP: Other Aspects

- Collect clock estimates from several servers
- Clock selection algorithm: reduce to set of "trusted clocks"
  - Interval selection sub-algorithm eliminates outliers
  - Clustering algorithm selects best K estimates among the ones that remain
- Clock combining algorithm
  - Weighted average of clock estimates used to adjust local oscillator

Correcting (Resynchronizing) Clocks

- Reset clock to new time value
  - Could cause clock to go backwards in time
  - Abrupt changes forward also undesirable
- Phase in clock change
  - Suppose clock is 10 milliseconds ahead
  - Interrupt generated every 30 milliseconds
  - Each interrupt increment clock by 29 milliseconds rather than 30
  - Perform above operation 10 times

Summary

- Clock synchronization needed for features like latency compensation in dead reckoning algorithms
- Coordinated Universal Time (UTC) standard reference
- Synchronization: Unknown message latency source of problems
- Several styles of clock synchronization available
  - Centralized vs. distributed
  - Push vs. pull
- Clock corrections must be phased in