Physical Clocks

- need for time in distributed systems
- physical clocks and their problems
- synchronizing physical clocks
  - coordinated universal time (UTC)
  - Cristain's algorithm
  - Berkeley algorithm
  - network time protocol (NTP)

Why Do We Care About "Time" in a Distributed System?

- may need to know the time of day some event happened on a specific computer
  - need to synchronize that computer's clock with some external authoritative source of time (external clock synchronization)
    - How hard is this to do?
- May need to know the time interval, or relative order, between two events that happened on different computers
  - If their clocks are synchronized to each other to some known degree of accuracy (called internal clock synchronization), we can measure time relative to a local clock
    - Is this always consistent?
- Will ignore relativistic effects
  - Cannot ignore network's unpredictability

Physical Clocks

- Every computer contains a physical clock
- A clock (also called a timer) is an electronic device that counts oscillations in a crystal at a particular frequency
- Count is typically divided and stored in a counter register
- Clock can be programmed to generate interrupts at regular intervals (e.g., at time interval required by a CPU scheduler)
- Counter can be scaled to get time of day
- This value can be used to timestamp an event on that computer
- Two events will have different timestamps only if clock resolution is sufficiently small
- Many applications are interested only in the order of the events, not the exact time of day at which they occurred, so this scaling is often not necessary

Physical Clocks in a Distributed System

- Does this work?
  - Synchronize all the clocks to some known high degree of accuracy, and then
  - measure time relative to each local clock to determine order between two events
- Well, there are some problems…
  - It's difficult to synchronize the clocks
  - Crystal-based clocks tend to drift over time — count time at different rates, and diverge from each other
    - Physical variations in the crystals, temperature variations, etc.
    - Drift is small, but adds up over time
    - For quartz crystal clocks, typical drift rate is about one second every 106 seconds =11.6 days
    - Best atomic clocks have drift rate of one second in 1013 seconds = 300,000 years

Coordinated universal time

- The output of the atomic clocks is called International Atomic Time
  - Coordinated Universal Time (UTC) is an international standard based on atomic time, with an occasional leap second added or deleted
- UTC signals are synchronized and broadcast regularly by various radio stations (e.g., WWV in the US) and satellites (e.g., GEOS, GPS)
  - Have propagation delay due to speed of light, distance from broadcast source, atmospheric conditions, etc.
  - Received value is only accurate to 0.1–10 milliseconds
- Unfortunately, most workstations and PCs don't have UTC receivers

Synchronizing physical clocks

- Use a time server with a UTC receiver
- Centralized algorithms
  - Client sets time to Tserver + Dtrans
    - Tserver = server's time
    - Dtrans = transmission delay
      - Unpredictable due to network traffic

Cristian's algorithm

- Send request to time server, measure time Dtrans taken to receive reply Tserver
- Set local time to Tserver + (Dtrans / 2)
  - Improvement: make several requests, take average Tserver value
- Assumptions:
  - Network delay is fairly consistent
  - Request & reply take equal amount of time
- to offset variations in time delay – client may average over several requests
- Problems:
  - Doesn't work if time server fails
  - Not secure against malfunctioning time server, or malicious impostor time server

7

Berkeley algorihtm

- Choose a coordinator computer to act as the master
- Master periodically polls the slaves — the other computers whose clocks should be synchronized to the master
  - Slaves send their clock value to master
- Master observes transmission delays, and estimates their local clock times
  - Master averages everyone's clock times (including its own)
    - Master takes a fault-tolerant average — it ignores readings from clocks that have drifted badly, or that have failed and are producing readings far outside the range of the other clocks
  - Master sends to each slave the amount (positive or negative) by which it should adjust its clock

8

Network Time Service protocol (NTP)

- Provides time service on the Internet
- Hierarchical network of servers:
  - Primary servers (100s) — connected directly to a time source
  - Secondary servers (1000s) — connected to primary servers in hierarchical fashion
    - ns.mcs.kent.edu runs a time server
  - Servers at higher levels are presumed to be more accurate than at lower levels
- Several synchronization modes:
  - Multicast — for LANs, low accuracy
  - Procedure call — similar to Cristian's algorithm, higher accuracy (file servers)
  - Symmetric mode — exchange detailed messages, maintain history
- All built on top of UDP (connectionless)

9

Compensating for clock drift in NTP

- Compare time Ts provided by time server to time Tc at computer C
- If Ts > Tc (e.g., 9:07am vs 9:05am)
  - Could advance C's time to Ts
  - May miss some clock ticks; probably OK (maybe not)
- If Ts < Tc (e.g., 9:07am vs 9:10am)
  - Can't roll back C's time to Ts
    - Many applications (e.g., make) assume that time always advances!
  - Can cause C's clock to run slowly until it resynchronizes with the time server
    - Can't change the clock oscillator rate, so have to change the software interpreting the clock's counter register
    - Tsoftware = a Thardware + b
    - Can determine constants a and b

10

Is It Enough to Synchronize Physical Clocks?

- In a distributed system, there is no common clock, so we have to:
  - Use atomic clocks to minimize clock drift
  - Synchronize with time servers that have UTC receivers, trying to compensate for unpredictable network delay
- Is this sufficient?
  - Value received from UTC receiver is only accurate to within 0.1–10 milliseconds
    - At best, we can synchronize clocks to within 10–30 milliseconds of each other
    - We have to synchronize frequently, to avoid local clock drift
  - In 10 ms, a 100 MIPS machine can execute 1 million instructions
    - Accurate enough as time-of-day
    - Not sufficiently accurate to determine the relative order of events on different computers in a distributed system

11