CompSci 131

Parallel and Distributed Systems

Prof. A. Veidenbaum

Today's topics

- **Coordination in DS**
- **Clocks**
- **Clock Synchronization**

- **Reading assignment:**
  - **Today: 6.1**
  - **Next time: 6.2**

  - **Complete the assignment *before* next class**
    -

Last Lecture Covered

- **Naming in DS**

Coordination

- **Activities of multiple processes in DS need to be coordinated.**
  - **Saw an example of synchronization in SMP systems**
- **DS have a much more complex case:**
  - **A general event ordering**
- **How to deal with this?**

Synchronization

- In the context of SMPs
  - Mutual exclusion
    - » Enforce one-at-a-time access to a code section
      - Or object
  - Post/Wait
    - » Ordering enforcement – Post before Wait
- In the context of DS
    - » A somewhat simple case: Send/Recv in MPI
      - very similar to Post/Wait
- Barriers in both types of systems
- DS have a much more complex case:
  - general event ordering

Synchronization in DS

- **One way to achieve event ordering is using time**
  - **Need to know the actual time *very precisely***
- **Often just need to know the relative order**
  - **Does even A occur before event B**
- **The question is, how to figure out this order?**
  - **Will look at some general solutions**

- **One solution: a central coordinator**

- **But – processes *must* agree on which one it is**
  - **Statically**
    - » **What are the issues in using a static coordinator?**
  - **Dynamically**

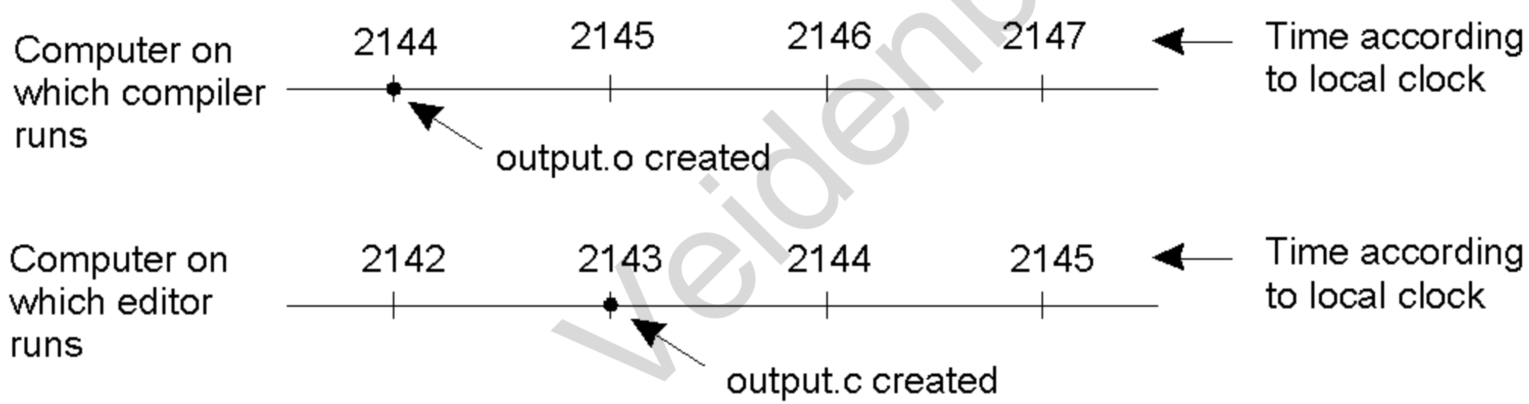
- ***Can <u>elect</u>* one process as a coordinator**
  - **How to run such an election?**

Time-based ordering

- A computer has a clock
  - Driven by complex analog circuitry
    - Includes an accurate oscillator
    - Generates "clock ticks"
  - Ticks are counted for a timer interrupt
  - Today there is also a 64b performance counter
    - Continuously counts *clock cycles*
- Processes on the computer see the same clock
  - Can read clock register or value
- *Thus no need for anything else to determine order*
  - What about multi-cores?

Clock Synchronization in DS

- **Each machine has its own clock, time may differ**
- **Programs like *make* have a problem**
  - **What is local time?**

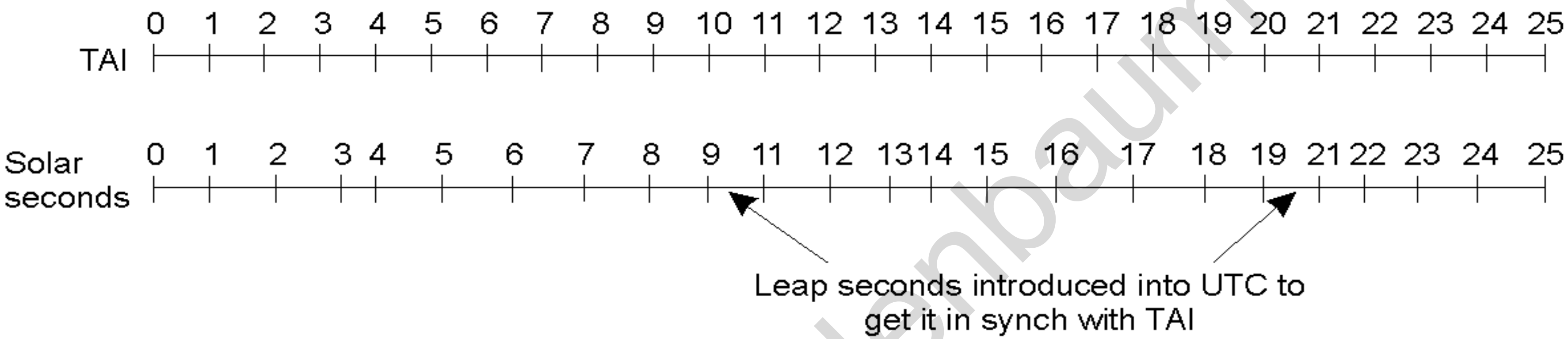


- ***Clock synchronization* makes clocks "the same" on all nodes**
  - **How close is "the same"?**

Measuring time

- Modern clocks are only about 600 years old
  - Before it was just seasons or solar/lunar days and months
  - Romans had a solar “time of day clock”
- Atomic clocks were introduced in 1948
  - Based on Cesium 133 electron state transitions
    - » Under microwave radiation
  - accuracy = 2 to 3 * 10 ^ -14th, i.e. 0.00000000000002 Hz
- US Naval Observatory operates ~70 clocks
  - Coordinated Universal Time is the average

Atomic Clocks



- **TAI (French!) seconds are of constant length, unlike solar seconds**
  - **Leap seconds are introduced to keep in phase with the sun**
- **The result is *Universal Coordinated Time* (UTC)**
  - **Replaced Greenwich mean time**
  - **Is the civilian time keeping**

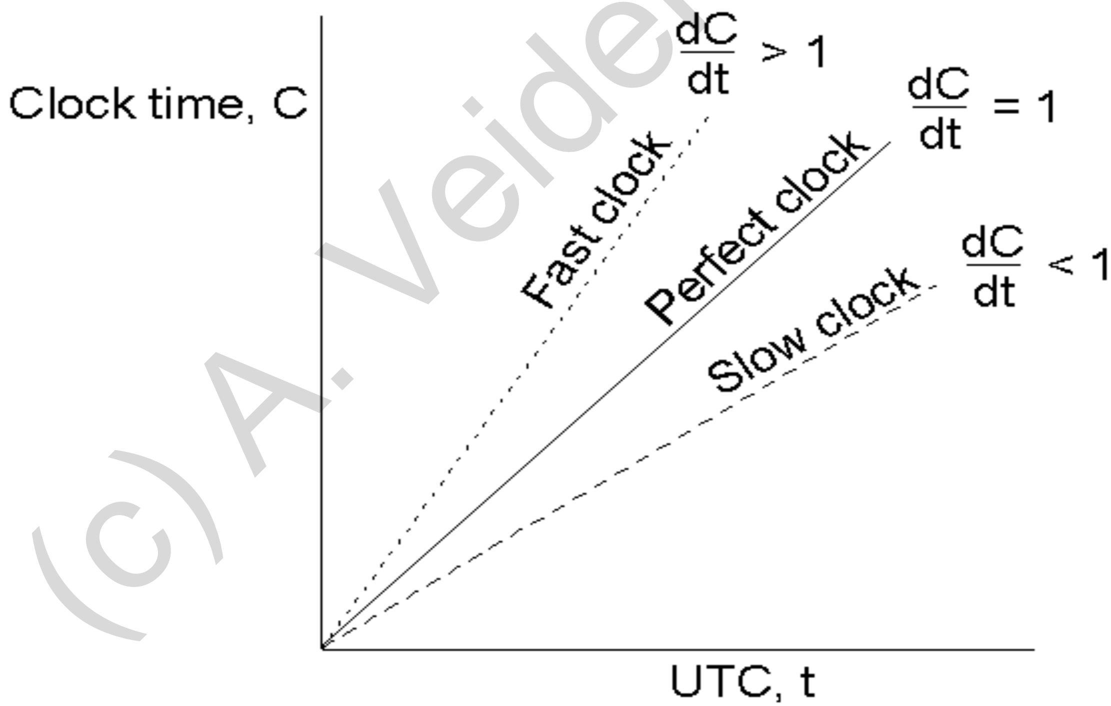
- **Broadcast by NIST on WWW (radio station ID), 1msec accuracy**
  - **But 10msec due to prop delay variations…**

GPS

- Uses satellites with several atomic clocks each
  - These are periodically calibrated from Earth
- A satellite broadcasts its position plus a time stamp
  - They are geostationary
  - Receiver computes delay from satellite i, plus time correction
    - » $\Delta_i = (T_{now} - T_i) + \Delta_r$ => distance $D_i = c*(T_{now} - T_i)$
  - Now triangulate to find a receiver position
- 4 satellites give you position <u>and</u> time

Clock Synchronization

- Clocks drift with respect to UTC time t, i.e. C(t)<>t
- Can define drift rate (max is +,- x)
- Need to check, synchronize clocks frequently to avoid drift
- May be used with WWW (or not)



Back to Computer Clocks

- Variation gets them out of sync

- A crystal oscillator is very precise, *but not perfect*
  – Oscillators with the same nominal frequency differ
    » Error accumulates over time

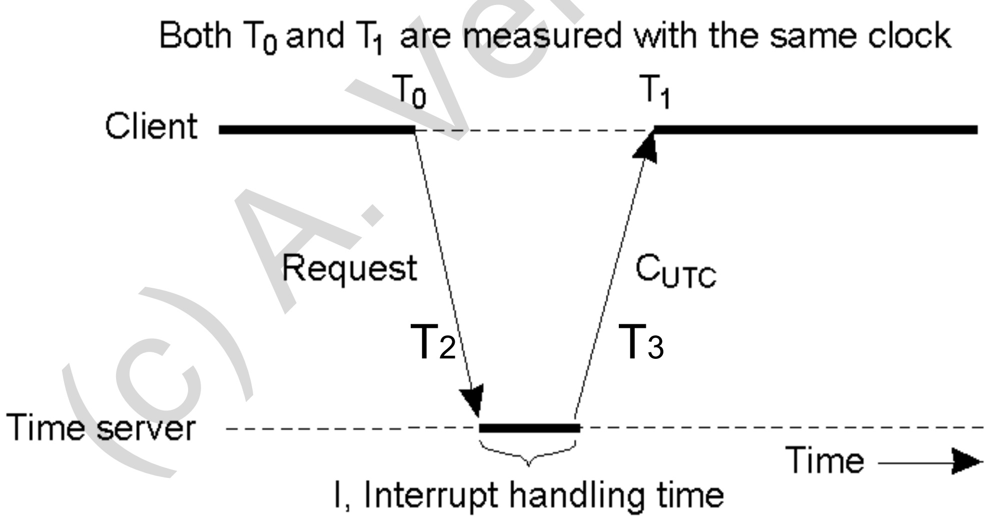
- The counter generates an interrupt after N sec
  – N = 1 or 10 msec today

- Sftw handler adds 1 to counter C. C is the local clock
  – Interrupt service time may also vary!

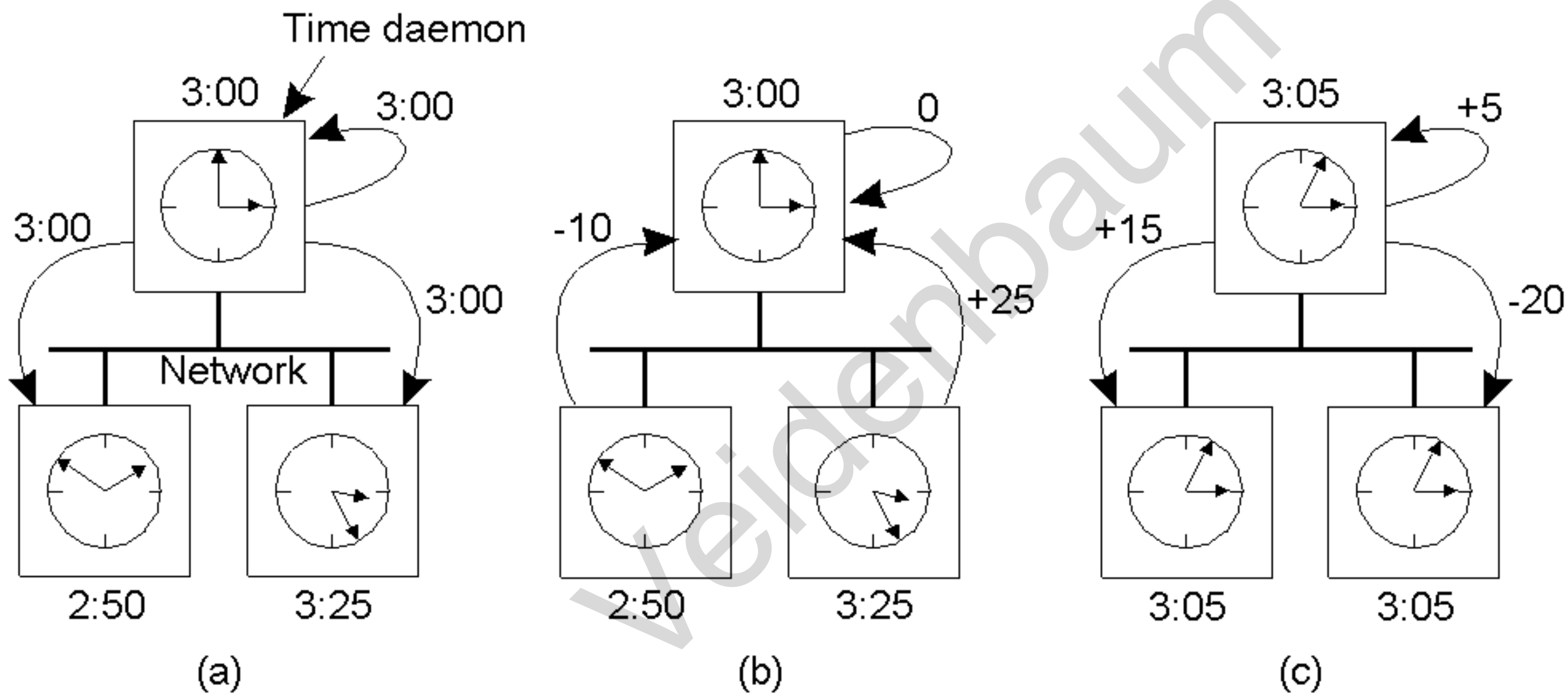
- Thus need clock synchronization!

Network Time Protocol (NTP)

- **Get the current time from a time server**
- **Send time stamp T0, server records arrival time T2, sends at T3**
  - **Server returns T2 and T3**
  - **Δ to Tserver: T2 = (T0 + δ) + Tprop , (T1 + δ) = T3 + Tprop**
    - » **Assumes: Tprop ~= T2-T0 ~= T3-T1, (Ti + δ) is client's adjusted time**
- **Buffer 8 pairs {Δ,δ}, pick minimal δ and corresponding Δ**
- **Adjustment has to be gradual *<u>and</u> cannot go backwards***
  - ***To slow a clock down* add less time on each timer interrupt**



The Berkeley Algorithm



a) The time daemon asks all the other machines for their clock values

b) The machines reply

c) The time daemon tells everyone how to adjust their clock

- **Centralized!**

Synchronization in Wireless nets

- Reference Broadcast Synchronization (RBS)
  - One sender node S, multiple receivers
  - Does not aim to sync to UTC
    - » *Just figure out receiver nodes' pairwise offsets*
- Assumes one-hop routing
  - Thus prop delay is basically constant
- But there is a receiver clock offset δ (to S)
  - all receiver nodes *exchange* Trecv$_i$
  - Can compute the relative (average) offset
    - » As well as individual offsets