

CSE 486/586 Distributed Systems Time and Synchronization

Steve Ko
Computer Sciences and Engineering
University at Buffalo

CSE 486/586

Last Time

- Models of Distributed Systems
 - Synchronous systems
 - Asynchronous systems
- Failure detectors---why?
 - Because things do fail.
- Failure detectors---what?
 - Properties: **completeness & accuracy**
 - Cannot have a perfect failure detector
 - Metrics: **bandwidth, detection time, scale, accuracy**
- Failure detectors---how?
 - Two processes: Heartbeating and Ping
 - Multiple processes: Centralized, ring, all-to-all

CSE 486/586

2

Today's Question

- The topic of time
 - Today and next time
- Why?
 - Need to know when things happen
 - One of the two fundamental challenges (failure & ordering)
- What?
 - Ideally, we'd like to know when exactly something happened.
- How?
 - Let's see!

CSE 486/586

3

Today's Question

- Servers in the cloud need to **timestamp events**
- Server A and server B in the cloud have different clock values
 - You buy an airline ticket online via the cloud
 - It's the last airline ticket available on that flight
 - Server A timestamps your purchase at 9h:15m:32.45s
 - What if someone else also bought the last ticket (via server B) at 9h:20m:22.76s?
 - What if Server A was > 10 minutes ahead of server B? Behind?
 - How would you know what the difference was at those times?

CSE 486/586

4

Physical Clocks & Synchronization

- Some definitions: Clock Skew versus Drift
 - Clock **Skew** = Relative Difference in clock *values* of two processes
 - Clock **Drift** = Relative Difference in clock *frequencies (rates)* of two processes
- A non-zero clock drift will cause skew to continuously increase.
- Real-life examples
 - Ever had "make: warning: Clock skew detected. Your build may be incomplete."?
 - It's reported that in the worst case, there's 1 sec/day drift in modern HW.
 - Almost all physical clocks experience this.

CSE 486/586

5

Synchronizing Physical Clocks

- $C_i(t)$: the reading of the software clock at process i when the real time is t .
- **External synchronization**: For a synchronization bound $D > 0$, and for source S of UTC time,

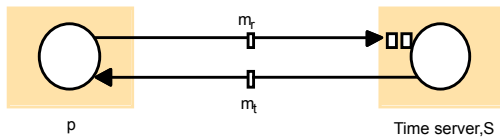
$$|S(t) - C_i(t)| < D,$$
 for $i = 1, 2, \dots, N$ and for all real times t .
Clocks C_i are accurate to within the bound D .
- **Internal synchronization**: For a synchronization bound $D > 0$,

$$|C_i(t) - C_j(t)| < D$$
 for $i, j = 1, 2, \dots, N$ and for all real times t .
Clocks C_i agree within the bound D .
- External synchronization with $D \Rightarrow$ Internal synchronization with $2D$
- Internal synchronization with $D \Rightarrow$ External synchronization with ??

CSE 486/586

6

Clock Synchronization Using a Time Server



- Client: "What time is it?"
- Server: "It's t ."
- Any difficulty?

CSE 486/586

7

Cristian's Algorithm

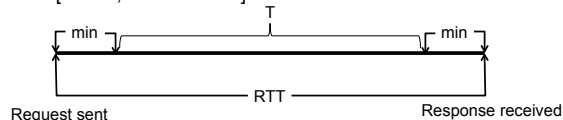
- Uses a **time server** to synchronize clocks
- Mainly designed for LAN
- Time server keeps the reference time (say UTC)
- A client asks the time server for time, the server responds with its current time T , and the client uses the received value T to set its clock
- But network round-trip time introduces an error.
- So what do we need to do?
 - Estimate one-way delay

CSE 486/586

8

Cristian's Algorithm

- Let RTT = response-received-time – request-sent-time (measurable at client)
- Also, suppose we know
 - The minimum value min of the client-server one-way transmission time [Depends on what?]
 - That the server timestamped the message at the last possible instant before sending it back
- Then, the actual time could be between $[T + min, T + RTT - min]$



CSE 486/586

9

Cristian's Algorithm

- (From the previous slide), the accuracy is: $\pm(RTT/2 - min)$
- Cristian's algorithm
 - A client asks its time server.
 - The time server sends its time T .
 - The client estimates the one-way delay and sets its time.
 - » It uses $T + RTT/2$
- Want to improve accuracy?
 - Take multiple readings and use the minimum RTT → **tighter bound**
 - For unusually long RTTs, ignore them and repeat the request → **removing outliers**

CSE 486/586

10

CSE 486/586 Administrivia

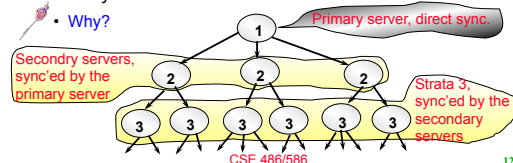
- Please start PA2-A.
- Please use Piazza; all announcements will go there.
 - If you want an invite, let me know.
- Please come to my office during the office hours!
 - Give feedback about the class, ask questions, etc.

CSE 486/586

11

The Network Time Protocol (NTP)

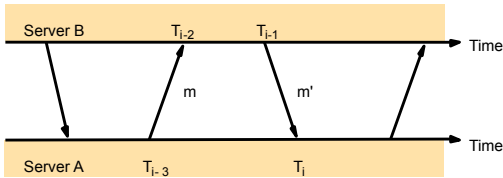
- Uses a **network of time servers** to synchronize all processes on a network.
- Designed for the Internet
 - Why not Cristian's algo?
- Time servers are connected by a **synchronization subnet tree**. The root is in touch with UTC. Each node synchronizes its children nodes.
- Why?



CSE 486/586

12

Messages Exchanged Between a Pair of NTP Peers ("Connected Servers")

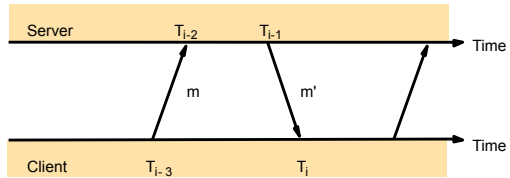


- Each message bears timestamps of recent message events: the local time when the previous NTP message was sent and received, and the local time when the current message was transmitted.

CSE 486/586

13

The Protocol

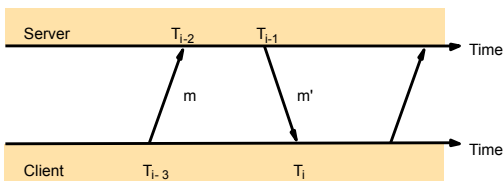


- Compute round-trip delay: $(T_i - T_{i-3}) - (T_{i-1} - T_{i-2})$
- Take the half of the round-trip delay as the one-way estimate: $((T_i - T_{i-3}) - (T_{i-1} - T_{i-2}))/2$

CSE 486/586

14

The Protocol

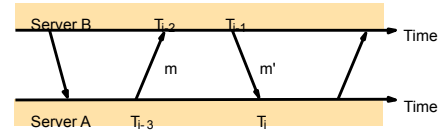


- Compute offset: $T_{i-1} + (\text{one-way estimate}) - T_i = ((T_{i-2} - T_{i-3}) + (T_{i-1} - T_i))/2$
- Do this with not just one server, but multiple servers.
- Do some statistical analysis, remove outliers, and apply a data filtering algorithm.
 - Out of the scope of this lecture

CSE 486/586

15

Theoretical Base for NTP

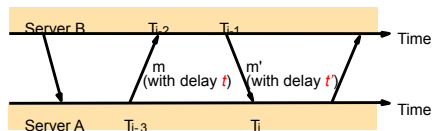


- o_i : estimate of the actual offset between the two clocks
- d_i : estimate of accuracy of o_i ; total transmission times for m and m' ; $d_i = t + t'$

CSE 486/586

16

Theoretical Base for NTP



First, let's get o :

$$T_{i-2} = T_{i-3} + t + o$$

$$T_i = T_{i-1} + t' - o$$

$$\Rightarrow o = (T_{i-2} - T_{i-3} + T_{i-1} - T_i)/2 + (t' - t)/2$$

Then, get the bound for $(t' - t)/2$:

$$-t' - t \leq t' - t \leq t' + t \text{ (since } t', t \geq 0)$$

Finally, we set:

$$o_i = (T_{i-2} - T_{i-3} + T_{i-1} - T_i)/2$$

$$d_i = t + t' = T_{i-2} - T_{i-3} + T_i - T_{i-1}$$

Then we get:

$$o_i - d_i/2 \leq o \leq o_i + d_i/2.$$

CSE 486/586

17

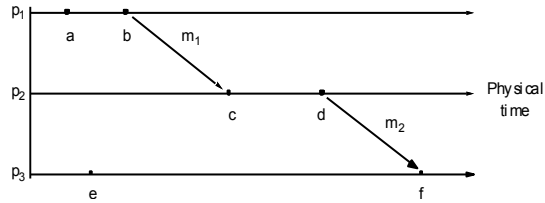
Then a Breakthrough...

- We **cannot** sync multiple clocks **perfectly**.
- Thus, if we want to **order events** happened at **different processes** (remember the ticket reservation example?), we cannot rely on physical clocks.
- Then came **logical time**.
 - First proposed by Leslie Lamport in the 70's
 - Based on **causality of events**
 - Defined relative time, not absolute time
- Critical observation**: time (ordering) **only matters** if two or more processes **interact**, i.e., **send/receive messages**.

CSE 486/586

18

Events Occurring at Three Processes



CSE 486/586

19

Summary

- Time synchronization important for distributed systems
 - Cristian's algorithm
 - Berkeley algorithm
 - NTP
- Relative order of events enough for practical purposes
 - Lamport's logical clocks
- Next: continue on logical clocks

CSE 486/586

20

Acknowledgements

- These slides contain material developed and copyrighted by Indranil Gupta at UIUC.

CSE 486/586

21