Computer Science 425
Distributed Systems

CS 425 / ECE 428

Physical Clocks & Synchronization

- **In a distributed system, each process has its own physical clock.**

- **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 causes skew to increase***

Synchronizing 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,...,N*** **and for all real times** ***t*****.**

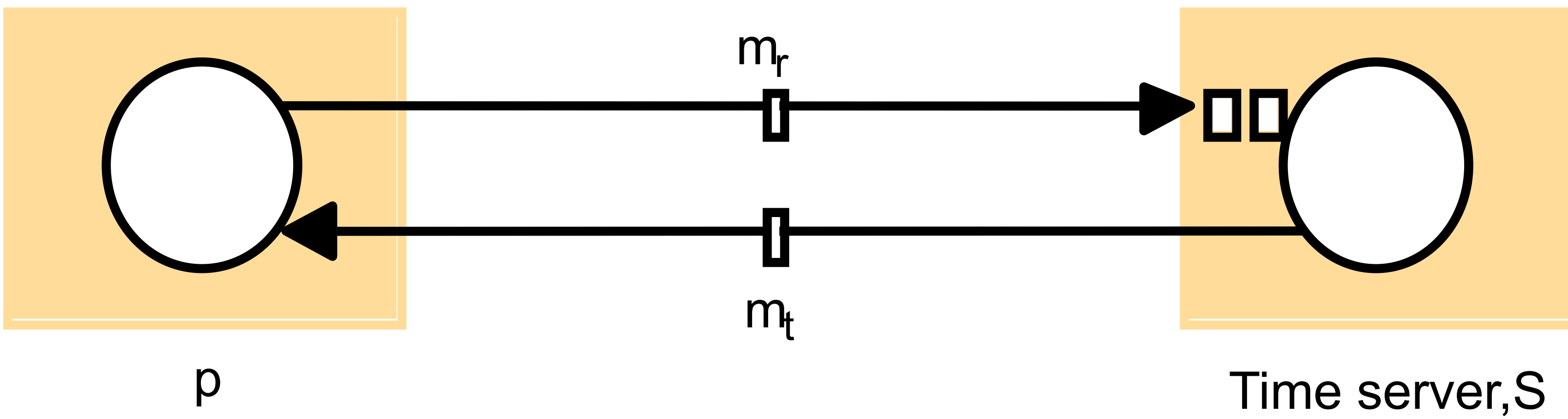
  Clocks C_i **are externally accurate to within the bound** ***D*****.**
- **Internal synchronization: For a synchronization bound** ***D>0*****,**

  $$|C_i(t) - C_j(t)| < D$$

  for ***i, j=1,2,...,N*** **and for all real times** ***t.***

  Clocks C_i **are internally accurate within the bound** ***D*****.**

Clock Synchronization Using a Time Server



Cristian's Algorithm

- Uses a *time server* to synchronize clocks
- Time server keeps the reference time
- A client asks the time server for time, the server responds with its current time *T*, and the client uses this received value to set its clock
- But network round-trip time introduces an error…

Let *RTT = response-received-time – request-sent-time* (measurable at client)

Also, suppose we know: (1) the minimum value *min* of the client-server one-way transmission time [Depends on what?]

(2) and 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]

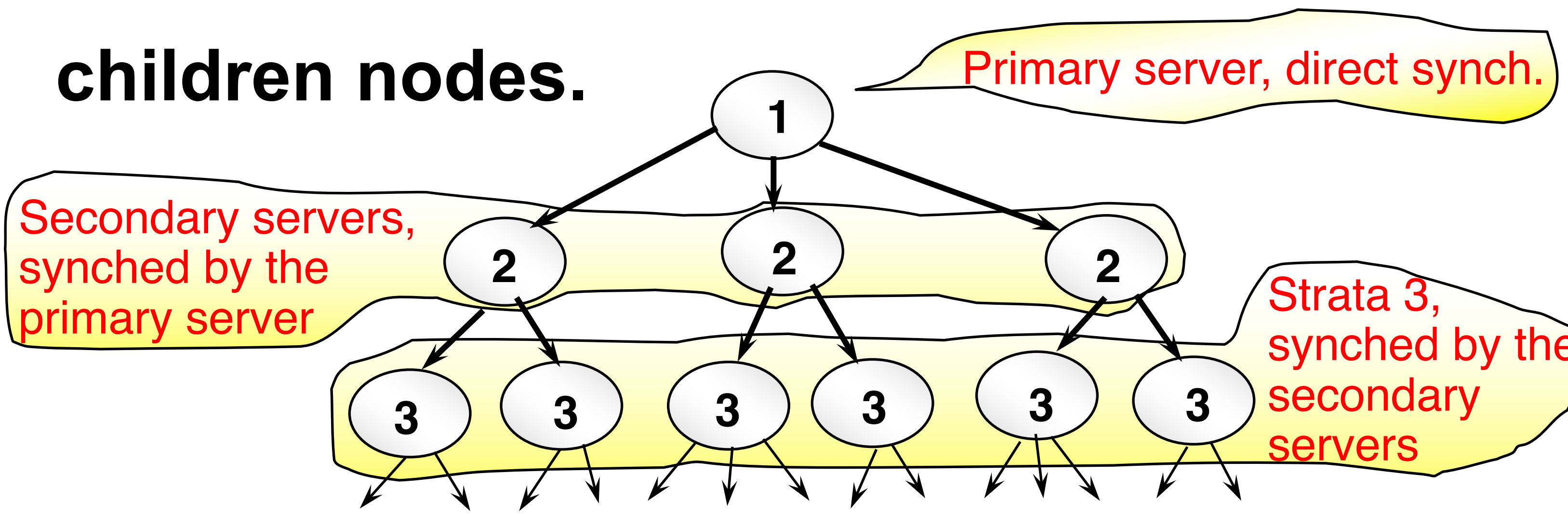
What are the two extremes?

Cristian's Algorithm (2)

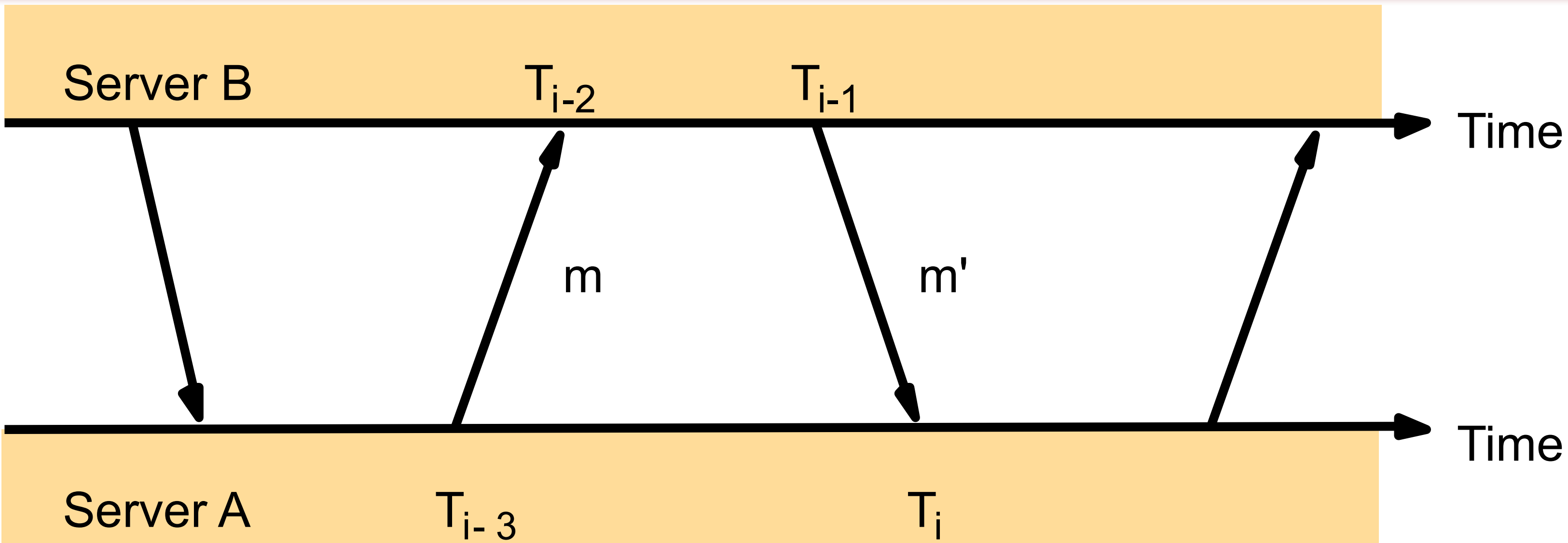
- Client sets its clock to halfway between T+min and T+RTT– min i.e., at T+RTT/2
  - Worst case skew in client clock time will be = half of this interval = (RTT/2 – *min*)
- Can increase clock value, but should *not* decrease it – Why?
- For unusually long RTTs, repeat the time request

The Network Time Protocol (NTP)

- **Uses a network of time servers to synchronize all processes on a network.**
- **Time servers are connected by a synchronization subnet tree. The root is in touch with UTC. Each node synchronizes its children nodes.**

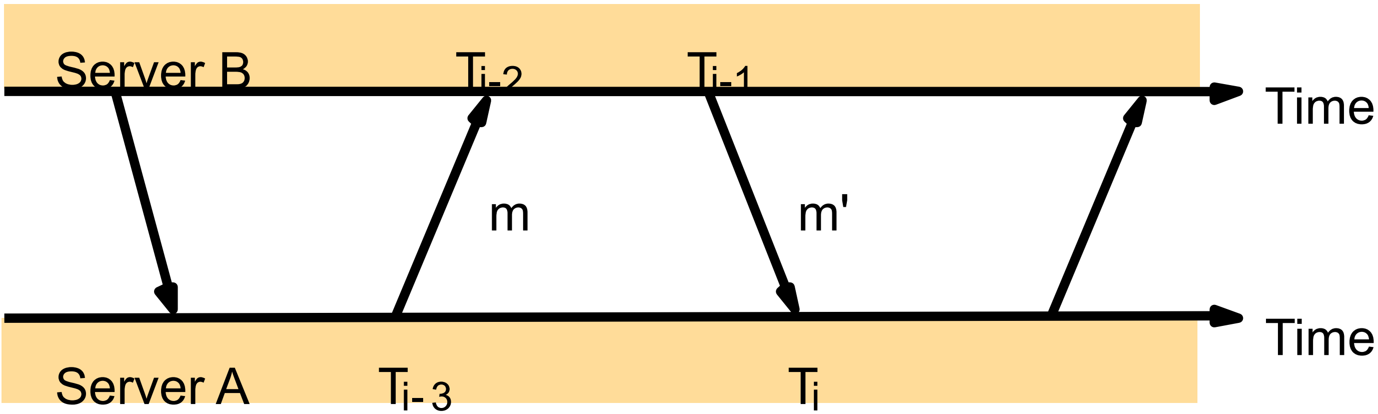


Messages Exchanged Between a Pair of NTP Peers ("Connected Servers")



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.

Theoretical Base for NTP



- *t* and *t'*: actual transmission times for *m* and *m' (unknown)*
- *o*: true offset of clock at *B* relative to clock at *A*
- o_i: estimate of actual offset between the two clocks
- d_i: estimate of accuracy of o_i ; total transmission times for *m* and *m'*; $d_i = t + t'$

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

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

This leads to

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

$$o = o_i + (t' - t)/2, \quad \text{where}$$

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

It can then be shown that

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

Summary

- **Time synchronization important for distributed systems**
  - **Cristian's algorithm**
  - **NTP**