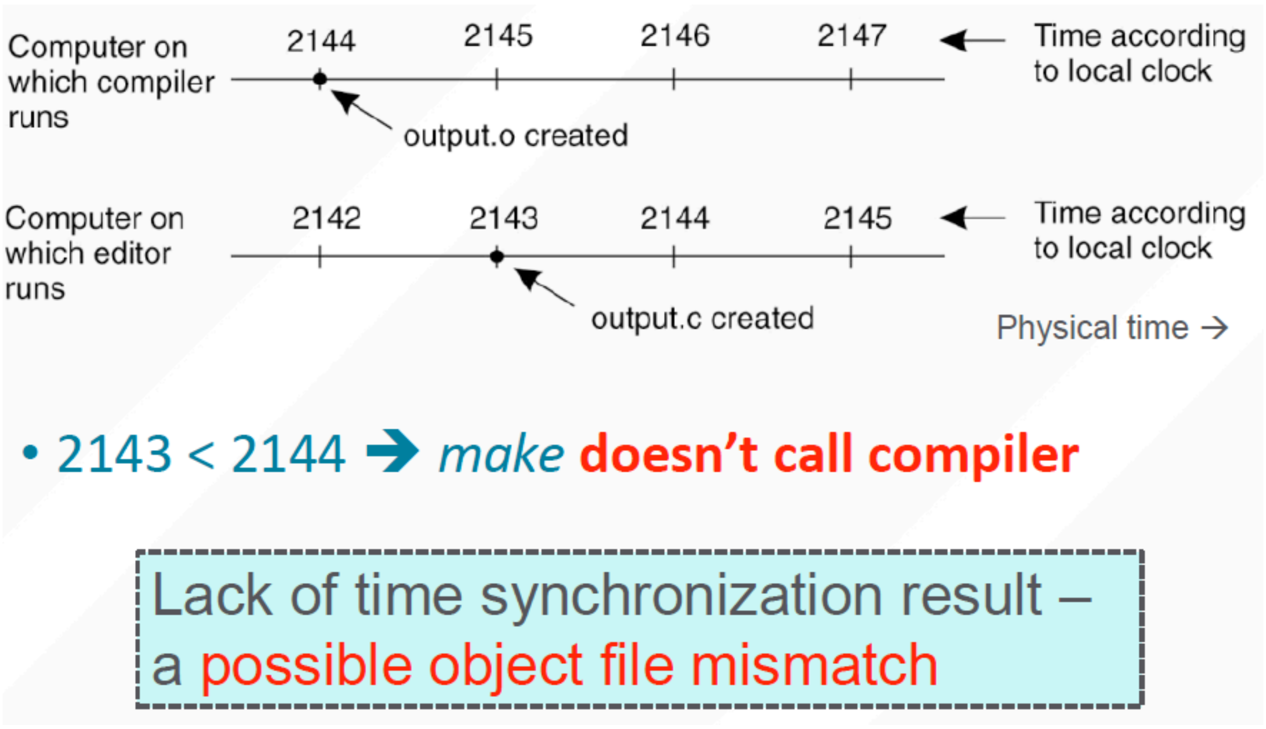
L10 TIME SYNCHRONIZATION, CRISTIAN'S ALGORITHM, BERKELEY ALGORITHM, NTP

Local Clock issues


Computer on which compiler runs
2144
2145
2146
2147
Time according to local clock
output.o created
Computer on which editor runs
2142
2143
2144
2145
Time according to local clock
output.c created
Physical time →
• 2143 < 2144 ➔ make doesn't call compiler
Lack of time synchronization result – a possible object file mismatch

If we only judge based on individual's local clock, it may causes serious problems.

Time synchronization hard

- Quartz oscillator sensitive

  temperature, age, vibration, radiation
- Internet condition

  Asynchronous: arbitrary time delays

  Best efforts: messages don't always arrive

Is it okay to use universal time?

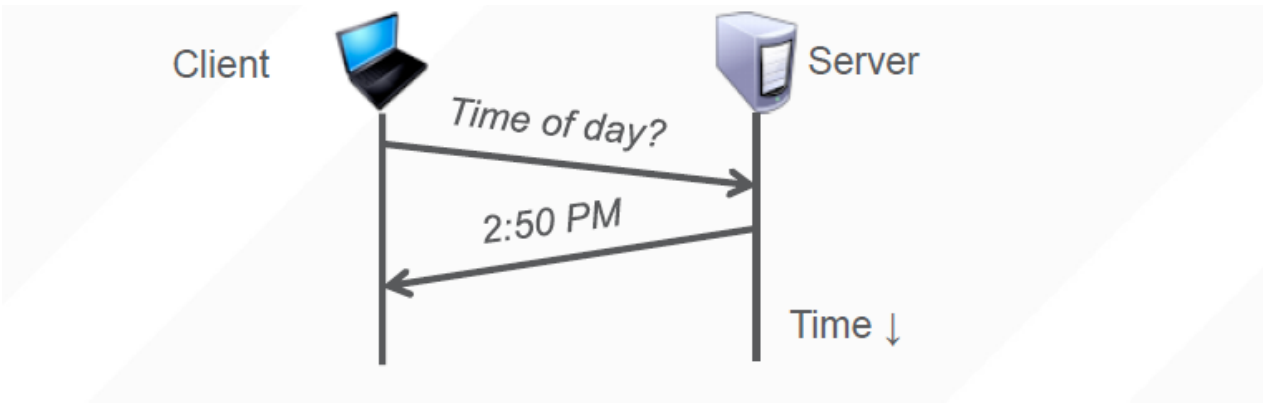
Computers with receivers can synchronize their clocks with signals broadcast from radio stations on land and statelite to set UTC.

Signals from GPS are accurate to about 1microsecond, better compared with signals from land-based stations.

Client synchronization

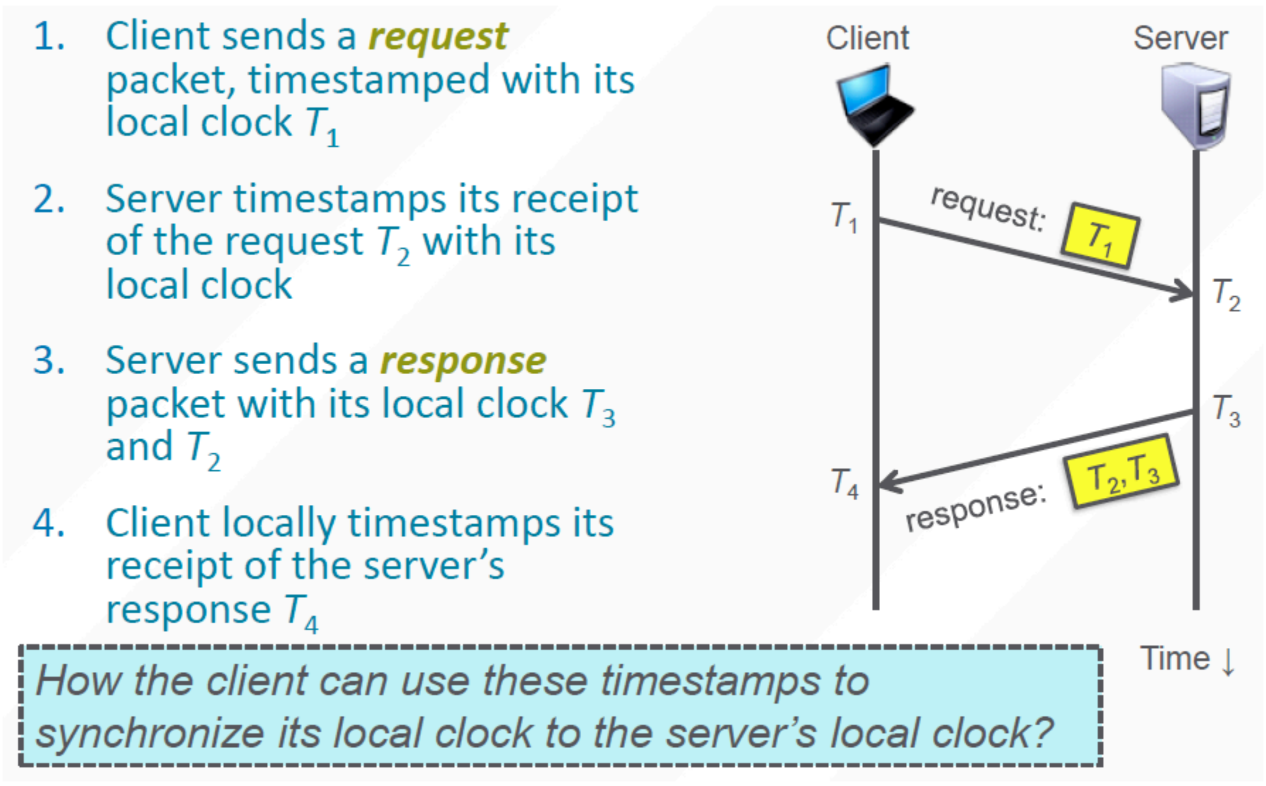
If a server has an accurate clock, how to get the accurate time from the server?

Can simple RPC work?



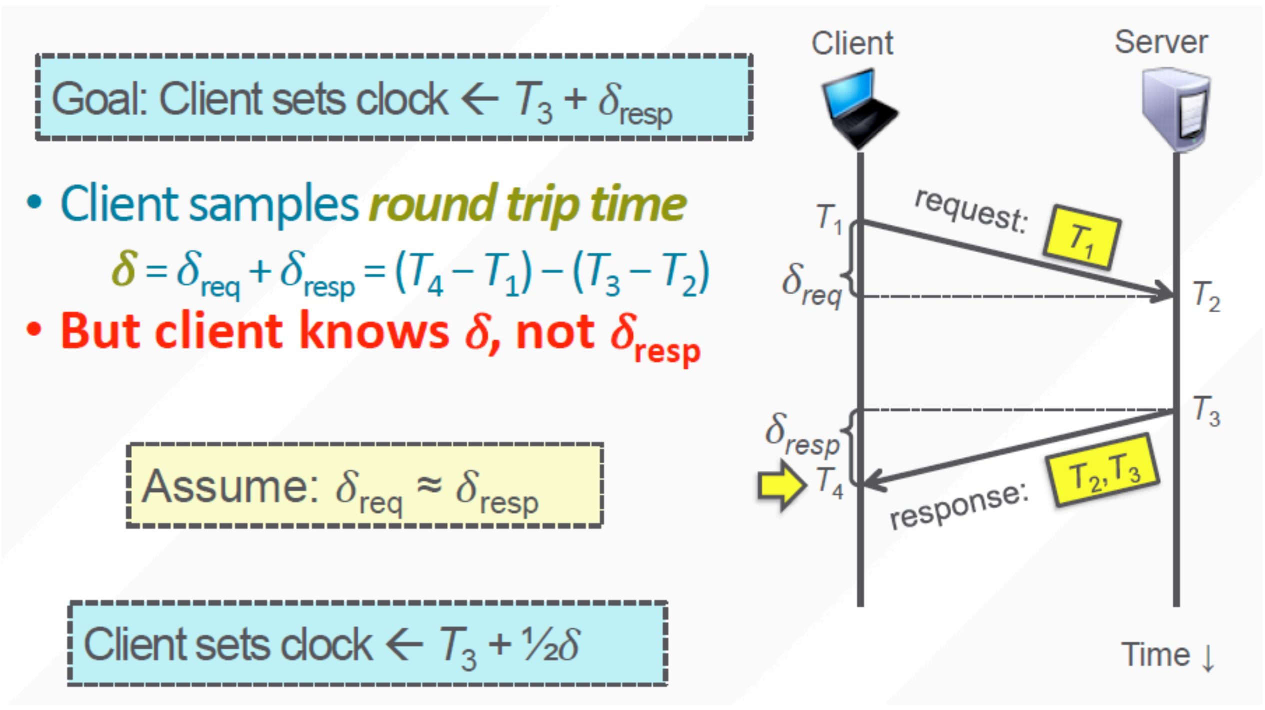
Nope, message delays(network latency) will have outdated server's answer

Cristian's algorithm: Outline



1. Client sends a ***request*** packet, timestamped with its local clock T_1
2. Server timestamps its receipt of the request T_2 with its local clock
3. Server sends a ***response*** packet with its local clock T_3 and T_2
4. Client locally timestamps its receipt of the server's response T_4

How the client can use these timestamps to synchronize its local clock to the server's local clock?



Goal: Client sets clock $\leftarrow T_3 + \delta_{resp}$

- Client samples *round trip time*

$$\delta = \delta_{req} + \delta_{resp} = (T_4 - T_1) - (T_3 - T_2)$$

- **But client knows δ, not δ_{resp}**

Assume: $\delta_{req} \approx \delta_{resp}$

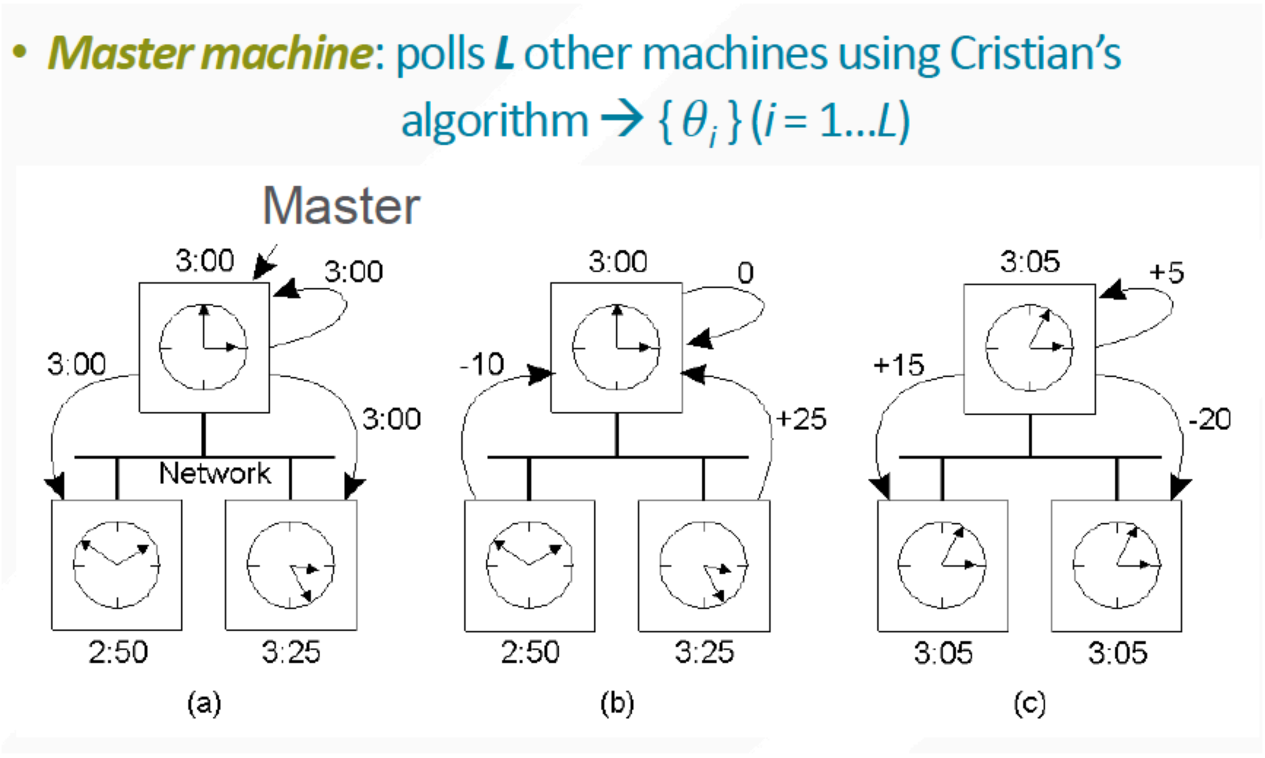
Client sets clock $\leftarrow T_3 + ½\delta$

However, a single time server can fail.

Berkeley Algorithm

Berkeley algorithm is a distributed algorithm for timekeeping.

- It assumes all machines have equally-accurate local clocks
- Obtains average from participating computers and synchronizes clocks to the average

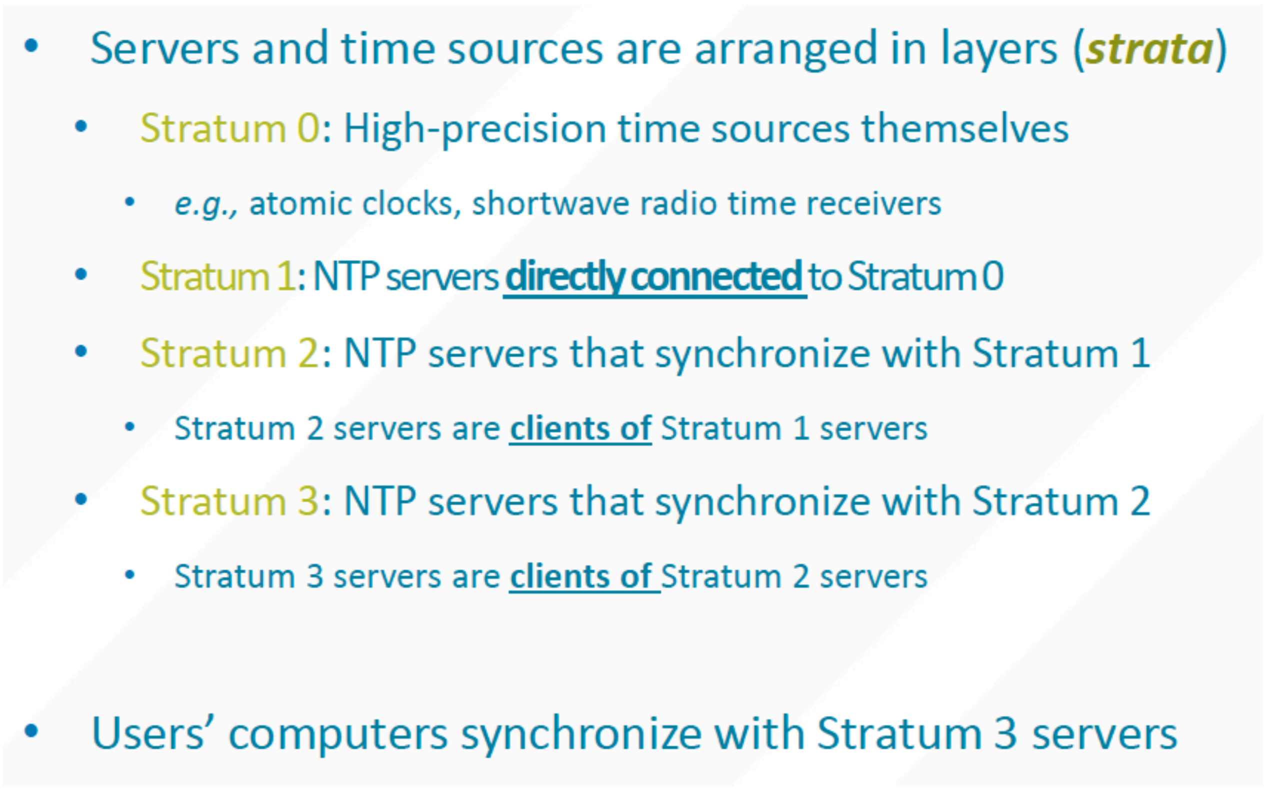


- ***Master machine***: polls ***L*** other machines using Cristian's algorithm → $\{\theta_i\}$ $(i = 1 \dots L)$

NTP (Network Time Protocol)

- Enables clients to be accurately synchronized to UTC despite message delays
- Provides **reliable** service
  - Survives lengthy losses of connectivity
  - Communicates over redundant network paths
- Provides an **accurate** service

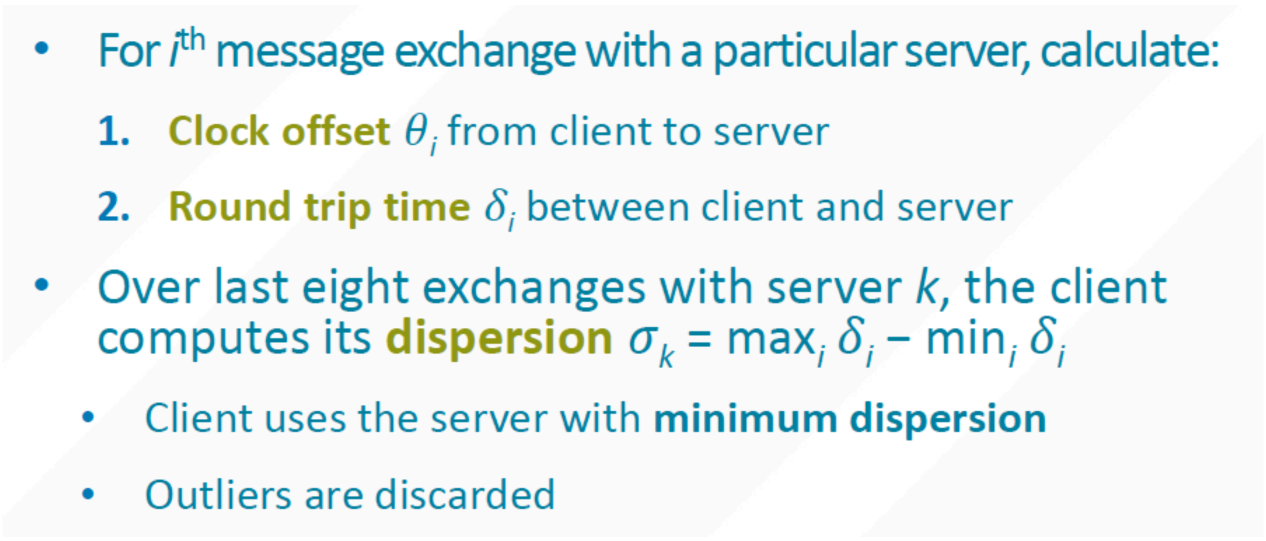
System Structure



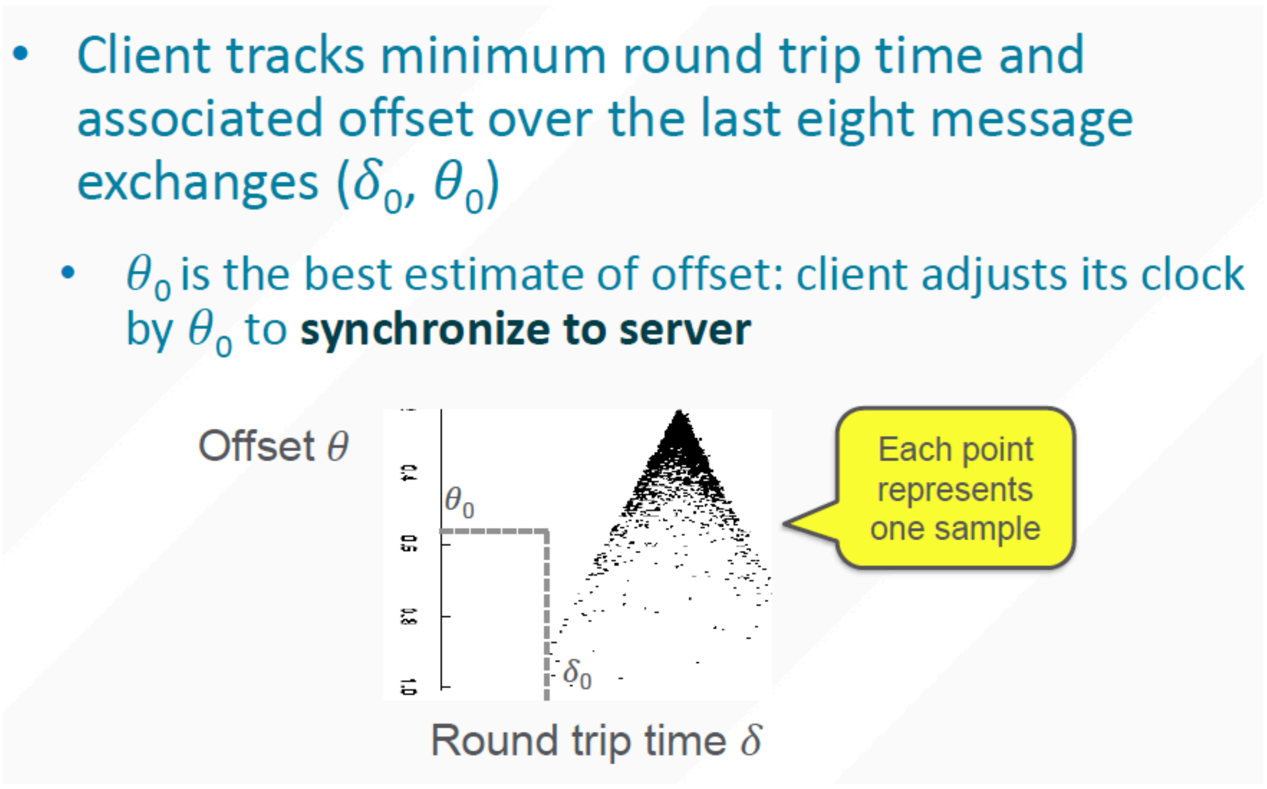
- Servers and time sources are arranged in layers (***strata***)
  - Stratum 0: High-precision time sources themselves
    - *e.g.*, atomic clocks, shortwave radio time receivers
  - Stratum 1: NTP servers **directly connected** to Stratum 0
  - Stratum 2: NTP servers that synchronize with Stratum 1
    - Stratum 2 servers are **clients of** Stratum 1 servers
  - Stratum 3: NTP servers that synchronize with Stratum 2
    - Stratum 3 servers are **clients of** Stratum 2 servers
- Users' computers synchronize with Stratum 3 servers

Message between an NTP client and server are exchanged in pairs: request and response.

Using Cristian's algorithm



- For i^{th} message exchange with a particular server, calculate:
  1. **Clock offset** θ_i from client to server
  2. **Round trip time** δ_i between client and server
- Over last eight exchanges with server k, the client computes its **dispersion** $\sigma_k = \max_i \delta_i - \min_i \delta_i$
  - Client uses the server with **minimum dispersion**
  - Outliers are discarded



- Client tracks minimum round trip time and associated offset over the last eight message exchanges (δ_0, θ_0)
  - θ_0 is the best estimate of offset: client adjusts its clock by θ_0 to **synchronize to server**

Change Time

Instead, change the update rate for the clock

- Changes time in a more gradual fashion
- Prevents inconsistent local timestamps