Physical Clocks

Physical Time

Each node in a distributed system has a local clock

Runs at an imprecise rate, close to wall clock rate

Rate can vary over time (e.g., temperature)

Can we synchronize (adjust) local clocks so that every node uses the same time?

- or approximately the same time

Why is Time Important? (Some Examples)

Merging distributed event logs

Consistency in distributed make

Update ordering on social media

Example: Merging Event Logs

You have a large, complex distributed system

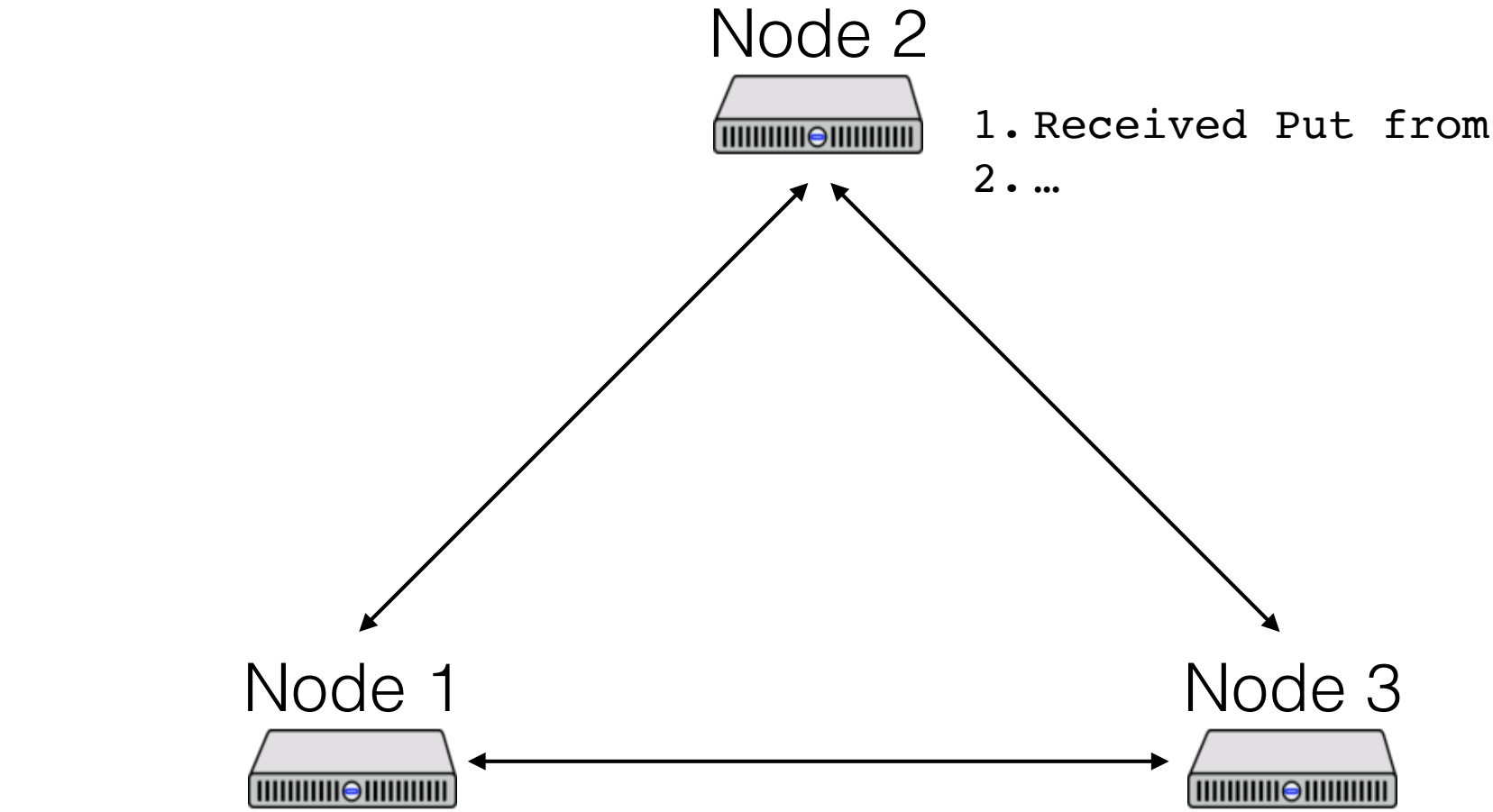
Sometimes, things go wrong—bugs, bad client behavior, etc.

You want to be able to debug!

Ask each node to produce a local log of events

- print statements, for distributed systems

How Do We Merge Event Logs?



Central Log?

Send every event to a centralized logging service

Events will be ordered at the logger

Do nodes keep going in the meantime?

- if so, order at logger != order in real time
- if not, will disturb system behavior (a lot!)

Merging Distributed Logs

Easy if every node knows precise wall clock time

Label each event locally with current time

Sort records after the fact

Example: Distributed Make

Distributed file servers hold source and object files

Clients update files (with modification times)

Make uses timestamps to decide what must be rebuilt

- If object O depends on source S

and O.time < S.time, rebuild O

Depends on correctness of local timestamp; what can go wrong?

Example: Update Ordering

Silently block boss on twitter

Tweet: “My boss is the worst, I need a new job!”

Tweets and block/mute lists sharded

- stored on different servers

Can you guarantee that no one sees the updates in the wrong order?

- easy if every server had wall clock time

Physical Clocks

Server clocks drift apart by 30 parts per million

- temperature sensitive

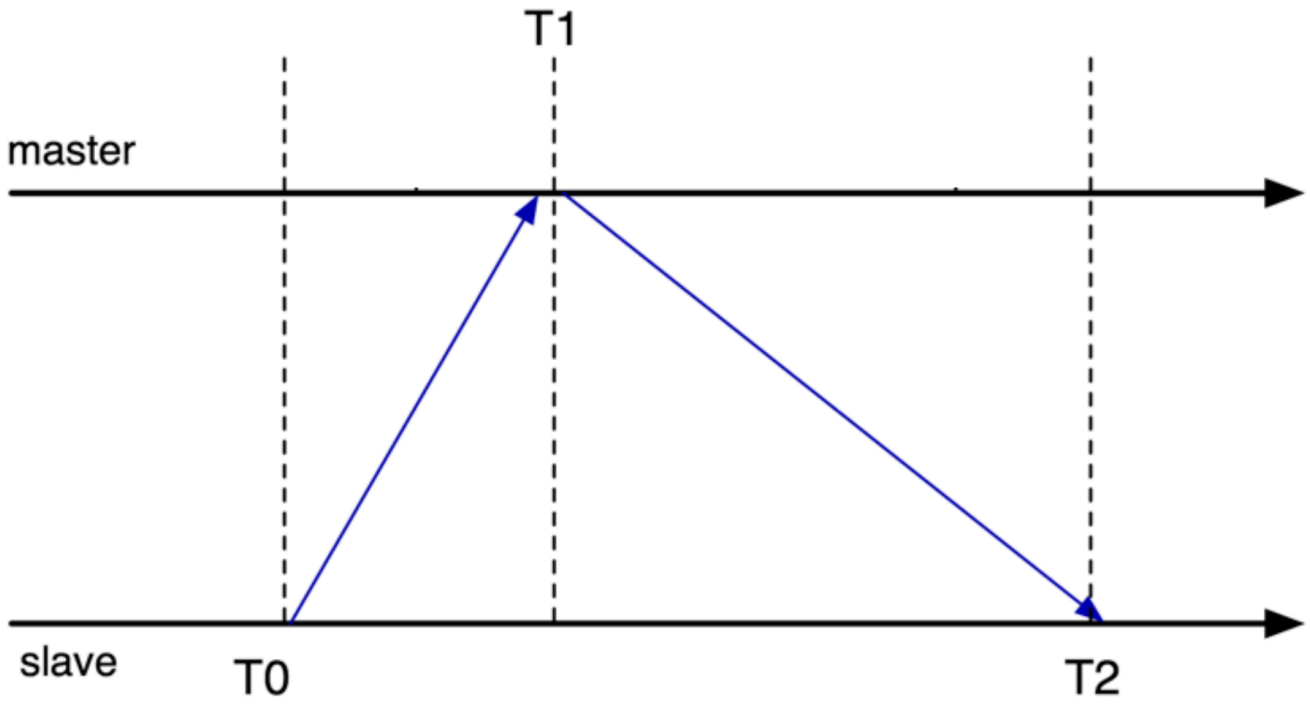
Atomic clock: ns accuracy, expensive

- one per data center?

GPS: 40 ns accuracy, requires antenna

Network packets between servers have variable path length, queueing delay

Client Driven Approach: NTP



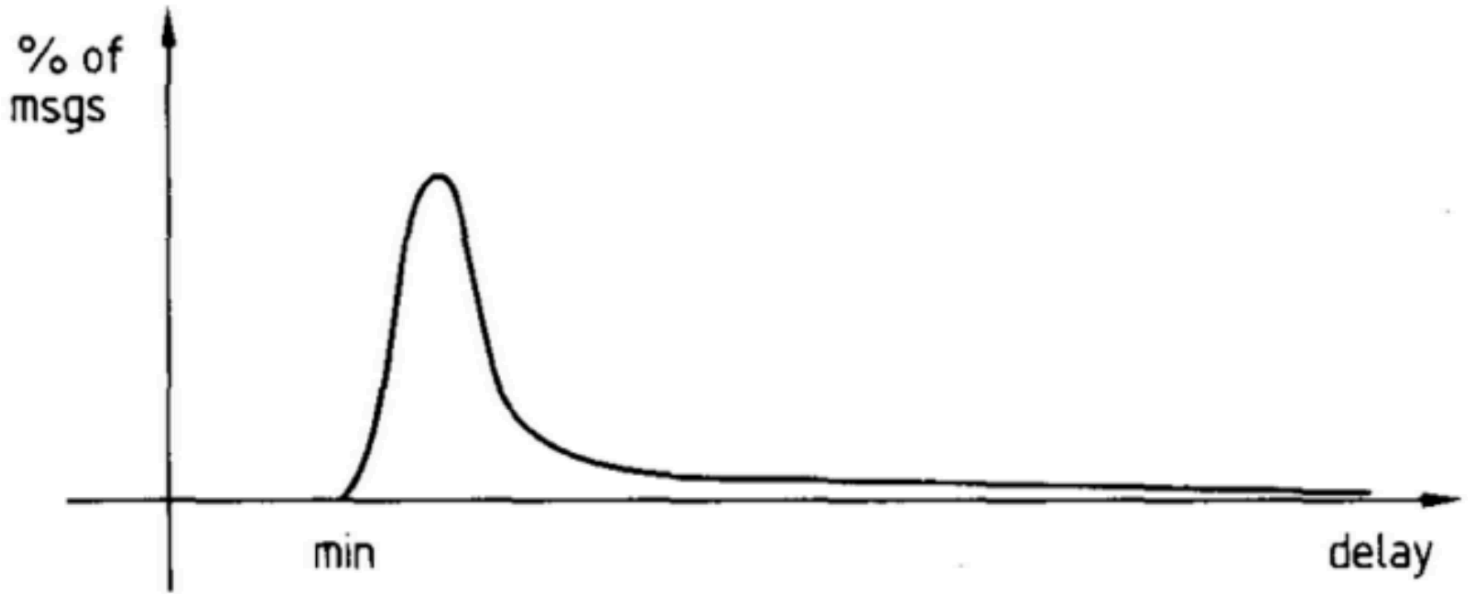
Clients queries time servers

Time = server's clock - 1/2 round trip

Average over several time servers; throw out outliers

In between queries, adjust for measured clock skew

Network Latency



Network latency is unpredictable with a lower bound

NTP vs. Huygens

NTP: sychronize to about 10 usec in data center

- 1/2 minimum round trip time across data center
- GPS/atomic clock (replicated)
- no special hardware on servers

Huygens: synch to 50 nsec, 99% of the time

- requires FGPA hardware per server
- GPS/atomic clock needed to synch to real time

Huygens Techniques

1. Timestamp packets in network interface card hardware
   - avoid OS context switches, OS queueing
2. Sample with pairs of packets, precisely spaced
   - if spacing maintained, likely no network queueing
   - throw out all other samples
2. Estimate relative clock phase + drift between pairs
3. Linear algebra to correct peer-to-peer clock skew

How Close Do We Need?

Huygens: 50 ns clock skew, 99% of the time

100Gbs network: 5ns per packet (min packet)

400Gbs network: 1.2ns per packet