Distributed Systems

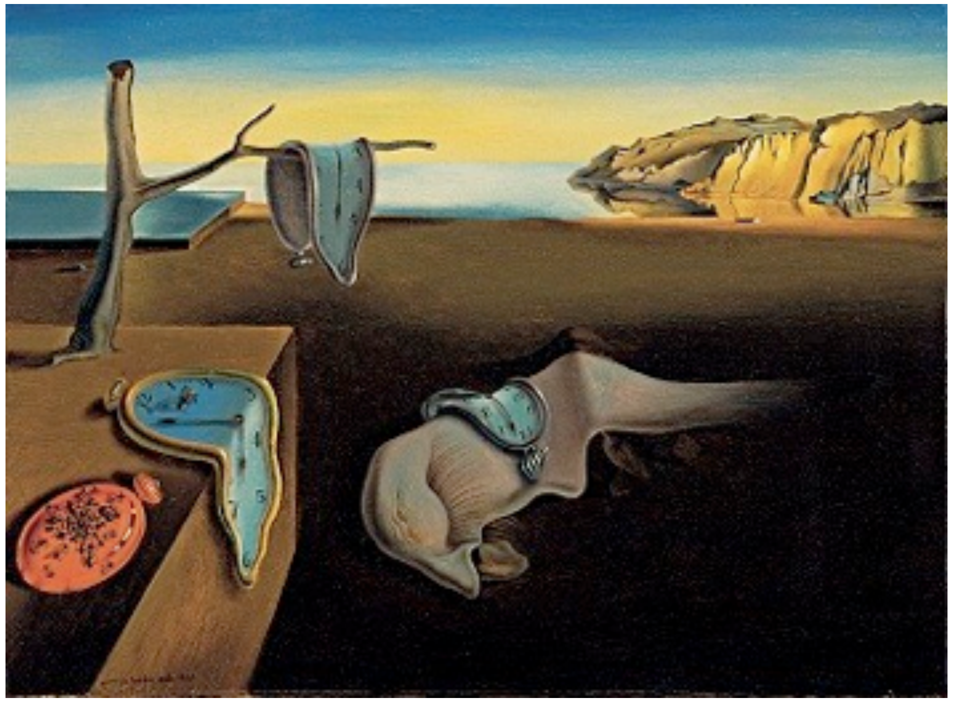
Rik Sarkar
James Cheney

Time and Synchronization
January 27, 2014

Introduction

- In this part of the course we will cover:
- Why time is such an issue for distributed computing
- The problem of maintaining a global state in a distributed system
- Consequences of these two main ideas
- Methods to get around these problems

Clocks

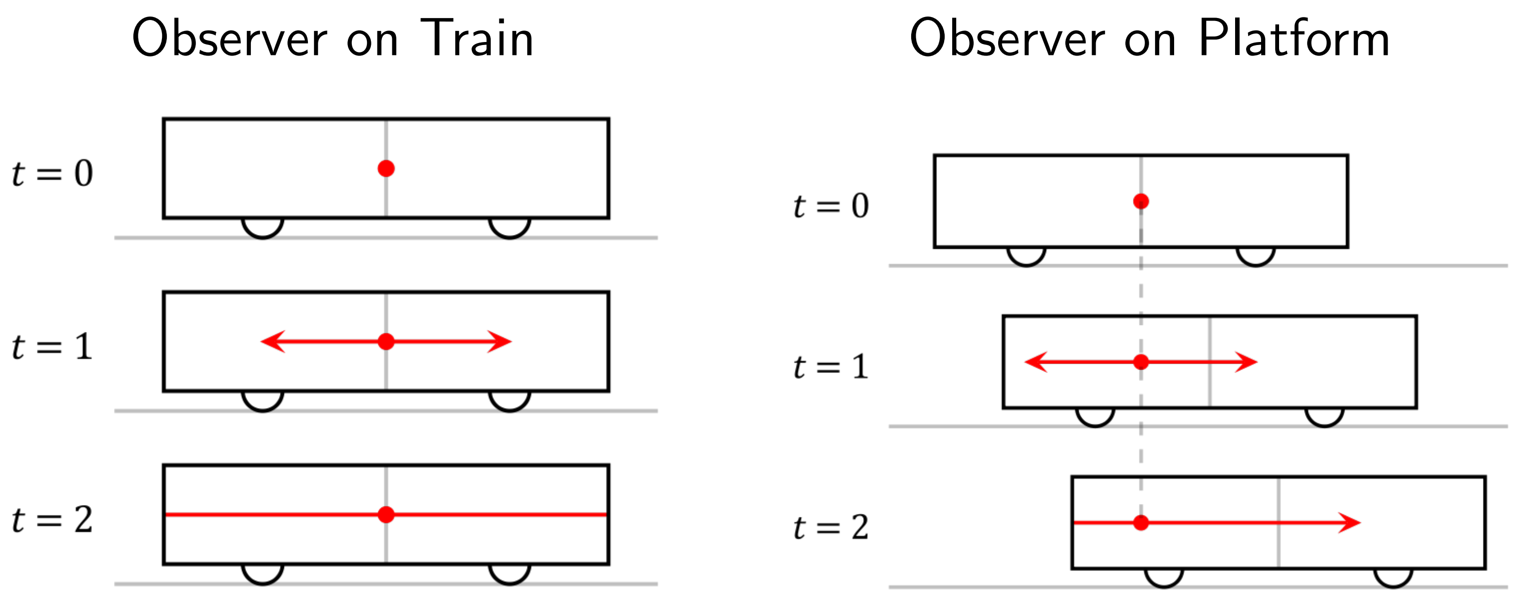


£20,000 (1714)
£2.6m (2014)

Global notion of time

- Einstein showed that the speed of light is constant for all observers regardless of their own velocity
- He (and others) have shown that this forced several other (sometimes counter-intuitive) properties including:
  1. length contraction
  2. time dilation
  3. relativity of simultaneity
- Contradicting the classical notion that the duration of the time interval between two events is equal for all observers
- It is impossible to say whether two events occur at the same time, if those two events are separated by space
- A drum beat in Japan and a car crash in Brazil
- However, if the two events are causally connected — if A causes B — the RoS preserves the causal order

Global notion of time



- However, if the two events are causally connected — if A causes B — the relativity of simultaneity preserves the causal order
- In this case, the flash of light happens before the light reaches either end of the carriage for all observers

Global Notion of Time

- We operate as if this were not true, that is, as if there were some global notion of time
- People may tell you that this is because:
- On the scale of the differences in our frames of references, the effect of relativity is negligible
- But that’s not really why we operate as if there was a global notion of time
- Even if our theoretical clocks are well synchronized, or mechanical ones are not
- We just accept this inherent inaccuracy & build that into our (social) protocols

Physical Clocks

- Computer clocks tend to rely on the oscillations occuring in a crystal
- The difference between the instantaneous readings of two separate clocks is termed their “skew”
- The “drift” between any two clocks is the difference in the rates at which they are progressing. The rate of change of the skew
- The drift rate of a given clock is the drift from a nominal “perfect” clock, for quartz crystal clocks this is about 10−6
- Meaning it will drift from a perfect clock by about 1 second every 1 million seconds — 11 and a half days.

Coordinated Universal Time and French

- The most accurate clocks are based on atomic oscillators
- Atomic clocks are used as the basis for the international Standard International Atomic Time
- Abbreviated to TAI from the French Temps Atomique International
- Since 1967 a standard second is defined as 9,192,631,770 periods of transition between the two hyperfine levels of the ground state of Cesium-133 (Cs133).
- Time was originally bound to astronomical time, but astronomical and atomic time tend to get out of step
- Coordinated Universal Time — basically the same as TAI but with *leap seconds* inserted
- Abbreviated to UTC again from the French Temps Universel Coordonné

Correctness of Clocks

- What does it mean for a clock to be correct?
- The operating system reads the node's hardware clock value, $H(t)$, scales it and adds an offset so as to produce a software clock $C(t) = aH(t) + \beta$ which measures real, physical time t
- Suppose we have two real times t and t' such that $t < t'$
- A physical clock, H, is correct with respect to a given bound 'p' if:

$$(1-p)(t' - t) \leq H(t') - H(t) \leq (1+p)(t' - t)$$

- $(t' - t)$ — The true length of the interval
- $H(t') - H(t)$ — The measured length of the interval
- $(1-p)(t'-t)$ — The smallest acceptable length of the interval
- $(1+p)(t'-t)$ — The largest acceptable length of the interval

Correctness of Clocks

- $(1-p)(t'-t) \leq H(t')-H(t) \leq (1+p)(t'-t)$
- An important feature of this definition is that it is *monotonic*
- Meaning that:
  - If $t<t'$ then $H(t)<H(t')$
  - Assuming that $t < t'$ with respect to the precision of the hardware clock

Monotonicity

- What happens when a clock is determined to be running fast?
- We could just set the clock back:
  - but that would break monotonicity
- Instead, we retain monotonicity:
  - $C_i(t) = aH(t) + \beta$
  - decreasing β such that $C_i(t) \leq C_i(t')$ for all $t < t'$

External vs Internal Synchronization

- Intuitively, multiple clocks may be synchronized with respect to each other, or with respect to an external source.
- Formally, for a synchronization bound $D > 0$ and external source S:
  - Internal Synchronization: $|C_i(t) - C_j(t)| < D$
    - No two clocks disagree by D or more
  - External Synchronization: $|C_i(t) - S(t)| < D$
    - No clock disagrees with external source S by D or more
- Internally synchronized clocks may not be very accurate at all with respect to some external source
- Clocks which are externally synchronized to a bound of D though are automatically internally synchronized to a bound of $2 \times D$.

Synchronizing clocks (synchronous case)

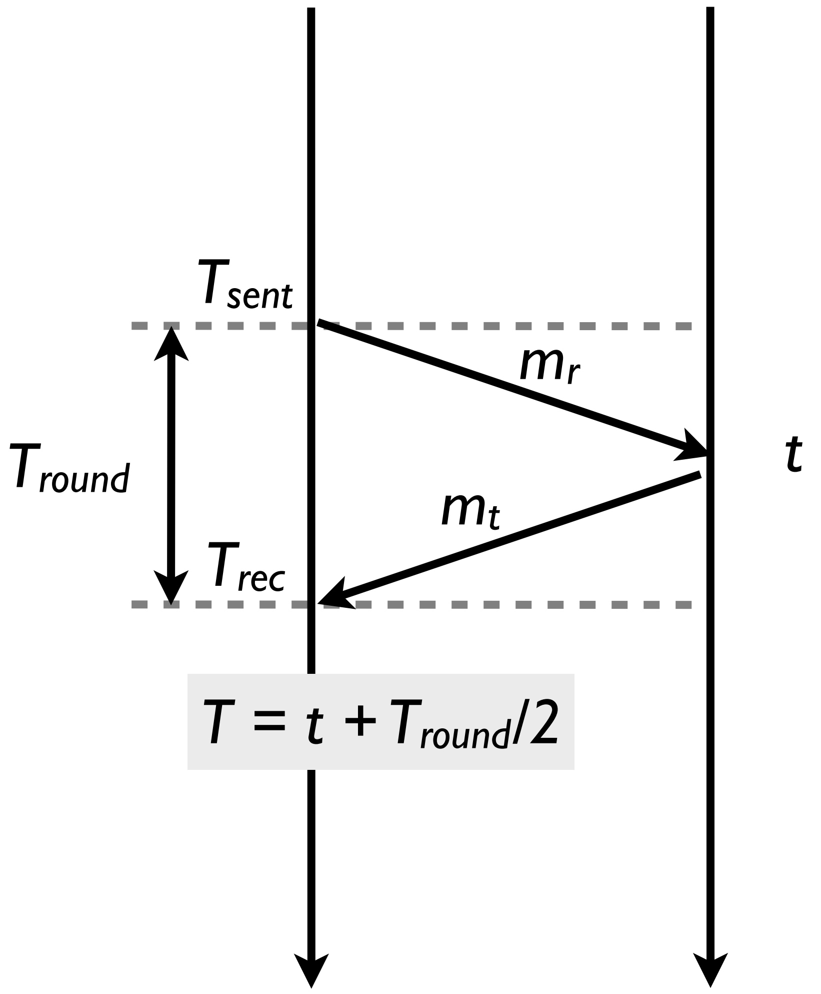
- Imagine trying to synchronize watches using text messaging
- Except that you have bounds for how long a text message will take
- How would you do this?
  1. Mario sends the time *t* on his watch to Luigi in a message m
  2. Luigi should set his watch to $t + T_{trans}$ where T_{trans} is the time taken to transmit and receive the message *m*
  3. Unfortunately T_{trans} is not known exactly
  4. We do know that $min \leq T_{trans} \leq max$
  5. We can therefore achieve a bound of $u = max - min$ if the Luigi sets his watch to $t + min$ or $t + max$
  6. We can do a bit better and achieve a bound of $u = (max-min)/2$ if Luigi sets his watch to $t + (max+min)/2$
  7. More generally if there are N clocks (Mario, Luigi, Peach, Toad, ...) we can achieve a bound of $(max-min)(1-1/n)$
  8. Or more simply we make Mario an external source and the bound is then $max - min$ (or $2 \times (max-min)/2$)

Cristian’s Method

- The previous method does not work where we have no upper bound on message delivery time, i.e. in an asynchronous system
- *Cristian’s method* is a method to synchronize clocks to an external source.
- This could be used to provide external or internal synchronization as before, depending on whether the source is itself externally synchronized or not.
- The key idea is that while we might not have an upper bound on how long a single message takes, we can have an upper bound on how long a round-trip took.
- However it requires that the round-trip time is sufficiently short as compared to the required accuracy.

Cristian’s Method

- Luigi sends Mario a message m_r requesting the current time, sent at time T_{sent} according to Luigi’s clock
- Mario responds with his current time in the message m_t.
- Luigi receives Mario’s time t in message m_t at time T_{rec}
  - according to his own clock the round trip took $T_{round} = T_{rec} - T_{sent}$
- Luigi then sets clock to $t + T_{round}/2$
- Assumes that the elapsed time was split evenly
  - (so may be less accurate in case of asymmetric latency)



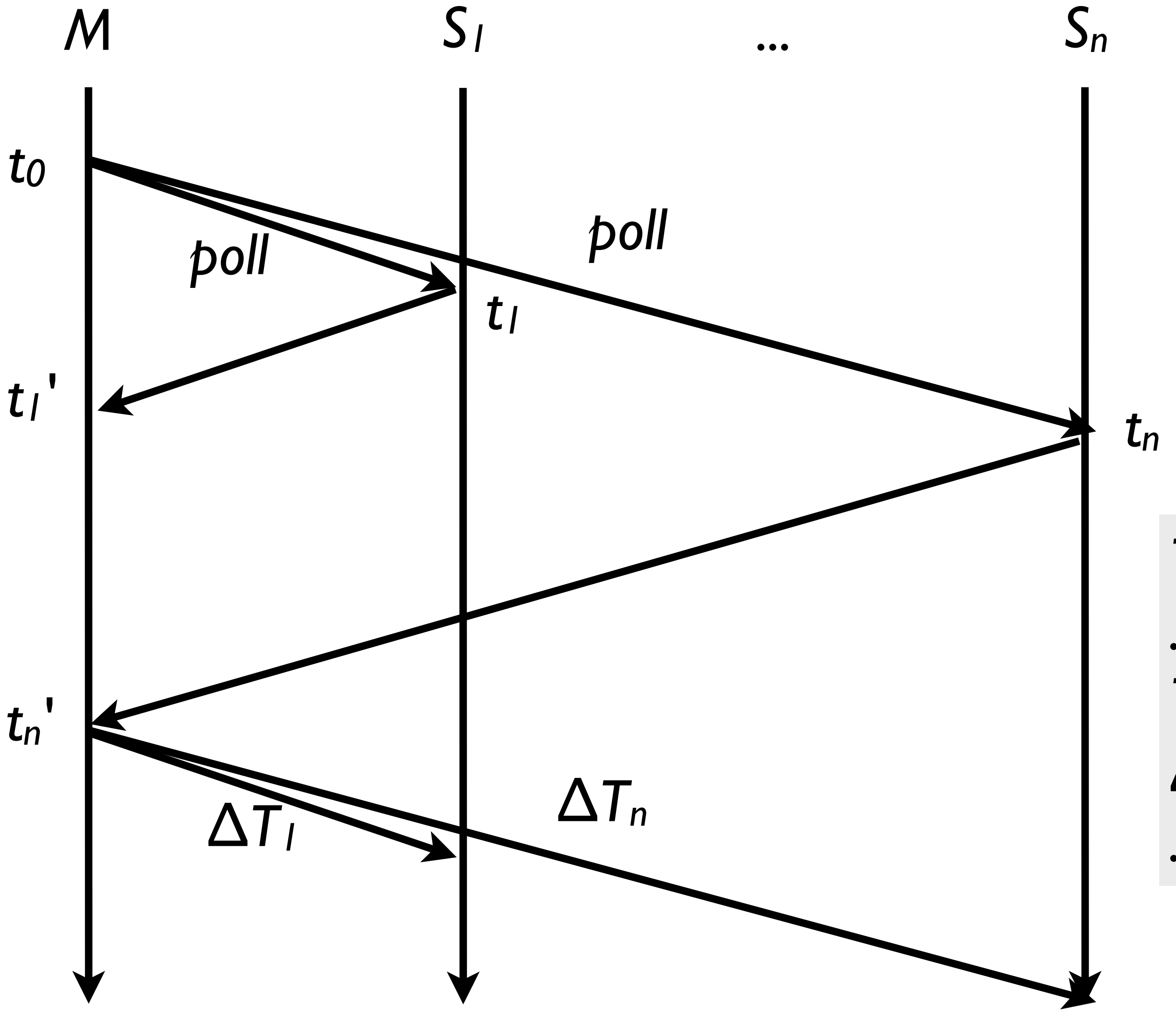
Cristian’s Method

- How accurate is this?
- We often don’t have accurate upper bounds for message delivery times but frequently we can at least guess conservative lower bounds
- Assume that messages take at least *min* time to be delivered
- The earliest time at which Mario could have placed his time into the response message m_t is min after Luigi sent his request message m_r.
- The latest time at which Mario could have done this was *min* before Luigi receives the response message m_t.
- The time on Mario’s watch when Luigi receives the response m_t is:
    - At least $t + min$
    - At most $t + T_{round} - min$
    - Hence the width is $T_{round} - (2 \times min)$
- The accuracy is therefore $T_{round}/2 - min$

The Berkeley Algorithm

- Like Cristian’s algorithm this provides either external synchronization to a known server, or internal synchronization via choosing one of the players to be the master
- Unlike Cristian’s algorithm though, the master in this case does not wait for requests from the other clocks to be synchronized, rather it periodically polls the other clocks.
- The others then reply with a message containing their current time.
- The master estimates the slaves current times using the round trip time in a similar way to Cristian’s algorithm
  - Then averages those clock readings together with its own to determine what should be the current time.
  - Finally replies to each of the other players with the amount by which they should adjust their clocks

The Berkeley Algorithm



$T_i = t_i + (t_i' - t_0)/2$

...

$T = (t_n' + T_1 + \ldots + T_n)/(n+1)$

$\Delta T_i = T_i - T$

...

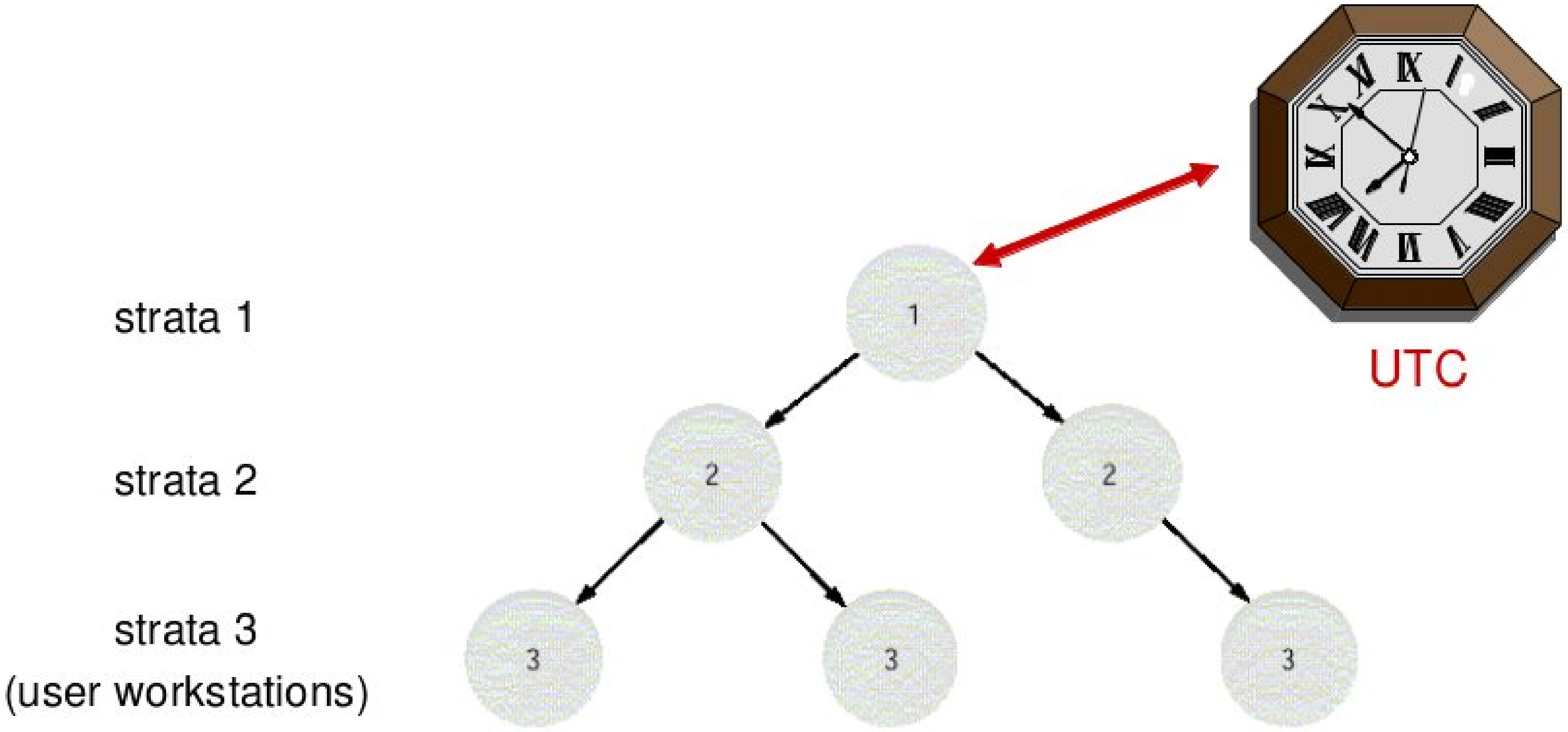
The Berkeley Algorithm

- If a straightforward average is taken, a faulty clock could shift this average by a large amount
  - therefore a *fault tolerant average* is taken
- This just averages all the clocks that do not differ by a chosen maximum amount *M*
  - (discarding clocks that are off by more than *M*)
- Synchronized ~15 computers to within 20-25ms

Network Time Protocol

- Network Time Protocol (actually abbreviated was NTP) is designed to allow clients to synchronize with UTC over the Internet.
- NTP is provided by a network of servers located across the Internet.
- Primary servers are connected directly to a time source such as a radio clock receiving UTC.
- Other servers are connected in a tree, with their strata determined by how many branches are between them and a primary server
- Strata *N* servers synchronize with Strata *N - 1* servers
- Eventually a server is within a user's workstation
- Errors may be introduced at each level of synchronization and they are cumulative, so the higher the strata number the less accurate is the server

Network Time Protocol



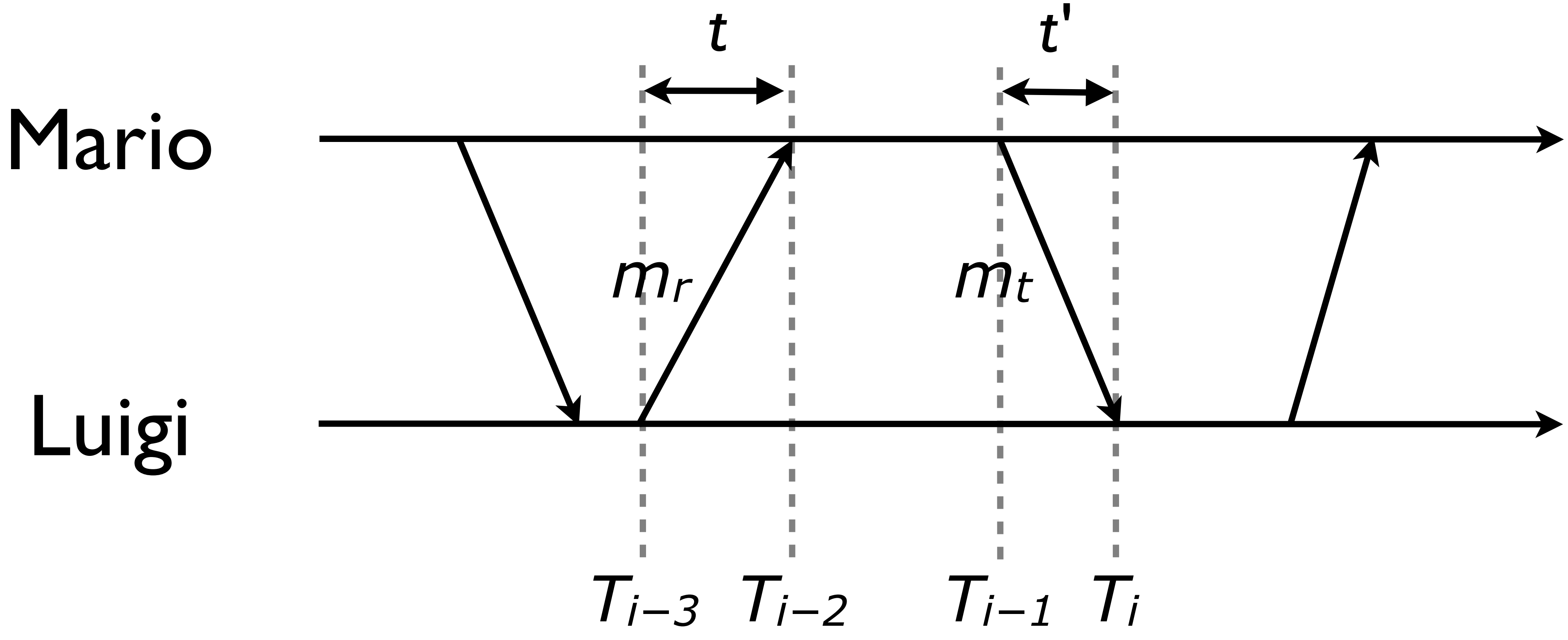
Note: Arrows denote synchronization control, numbers denote strata.

© Pearson Education 2001

- Note: this picture does not show synchronization between servers at the same strata, but this does occur

Network Time Protocol

- Synchronization between strata is pairwise
- Uses multiple rounds of messages



Pairwise synchronization

- Similar to Cristian's method, however:
- Four times are recorded as measured by the clock of the process at which the event occurs:
  1. T_{i-3} — Time of sending of the request message m_r
  2. T_{i-2} — Time of receiving of the request message m_r
  3. T_{i-1} — Time of sending of the response message m_t
  4. T_i — Time of receiving of the response message m_t
- So if Luigi is requesting the time from Mario, then T_{i-3} and T_i are recorded by Luigi and T_{i-2} and T_{i-1} are recorded by Mario
- Note that because Mario records the time at which the request message was received and the time at which the response message is sent, there can be a non-negligible delay between both
- In particular then messages may be dropped

Network Time Protocol

- If we assume that the true (unknown) offset between the two clocks is O_{true}:
- And that the actual transmission times for the messages m_r and m_t are t and t' respectively then:

  $T_{i-2} = T_{i-3} + t + O_{true}$ and $T_i = T_{i-1} + t' - O_{true}$

- T_{round} is the measure of accuracy (based on how long the messages were in transit)

  $T_{round} = (t+t') = (T_i - T_{i-3}) - (T_{i-1} - T_{i-2})$

- O_{guess} is the guess as to the offset

  $O_{guess} = [(T_{i-2} - T_{i-3}) + (T_{i-1} - T_i)] / 2$

Network Time Protocol

- This is the non-trivial line:

  $O_{guess} = [(T_{i-2} - T_{i-3}) + (T_{i-1} - T_i)]/2$

  $T_{i-2} - T_{i-3} = t + O_{true}$

  $T_{i-1} - T_i = O_{true} - t'$

  Hence $O_{guess} = [(t + O_{true}) + (O_{true} - t')] / 2$

  $= [(t - t') + (2 \times O_{true})]/2 = (t - t')/2 + O_{true}$

  That is: $O_{true} = O_{guess} + (t - t') / 2$

- Since we know that $T_{round} > |t - t'|$:

  $O_{guess} - T_{round} \leq O_{true} \leq O_{guess} + T_{round}$

Network Time Protocol (modes)

1. Multicast (broadcast to group) mode
   - Not considered very accurate
   - Intended for use on a high-speed LAN
   - Can be accurate enough nonetheless for some purposes
2. Procedure call mode
   - Similar to Cristian's method
   - Servers respond to requests from higher-strata servers
   - Who use round-trip times to calculate the current time to some degree of accuracy
   - Used for example in network file servers which wish to keep as accurate as possible file access times
3. Symmetric mode
   - Used where the highest accuracies are required
   - In particular between servers nearest the primary sources, that is the lower strata servers
   - Essentially similar to procedure-call mode except that the communicating servers retain timing information to improve their accuracy over time

Aside: Message reliability and TCP vs. UDP

- We will consider a number of different algorithms/protocols
  - making different **assumptions** about process failure and reliability of messages
- Transmission Control Protocol (TCP)
  - reliable, first-in-first-out streams
  - most Internet traffic (SMTP (mail), HTTP (Web), etc.)
  - but carries overhead due to latency, error detection/correction
- User Datagram Protocol (UDP)
  - messages may be dropped, reordered; error detection only
  - useful for faster traffic where reliability less important (or dealt with using other algorithms)
  - Including NTP, DNS, voice, video, games

Network Time Protocol

- In all three modes messages are delivered using the standard UDP (unreliable, broadcast) protocol
  - Hence message delivery is unreliable
- At the higher strata servers can synchronize to high degree of accuracy over time
- But in general NTP is useful for synchronizing accurately to UTC, whereby accurate is at the human level of accuracy
  - Wall clocks, clocks at stations etc
- In summary: we can synchronize clocks to a bounded level of accuracy, but for many applications the bound is simply not tight enough

Summary

- We noted that even in the real world there is no global notion of time
- We extended this to computer systems noting that the clocks associated with separate machines are subject to differences between them known as the skew and the drift.
- We nevertheless described algorithms for attempting the synchronization between remote computers
    - Cristian’s method
    - The Berkeley Algorithm
    - Pairwise synchronization in NTP
- **Next time:**
    - Despite these algorithms to synchronize clocks it is still impossible to determine for two arbitrary events which occurred before the other.
    - We will look at ways in which we can impose a meaningful order on remote events even without perfect synchronization