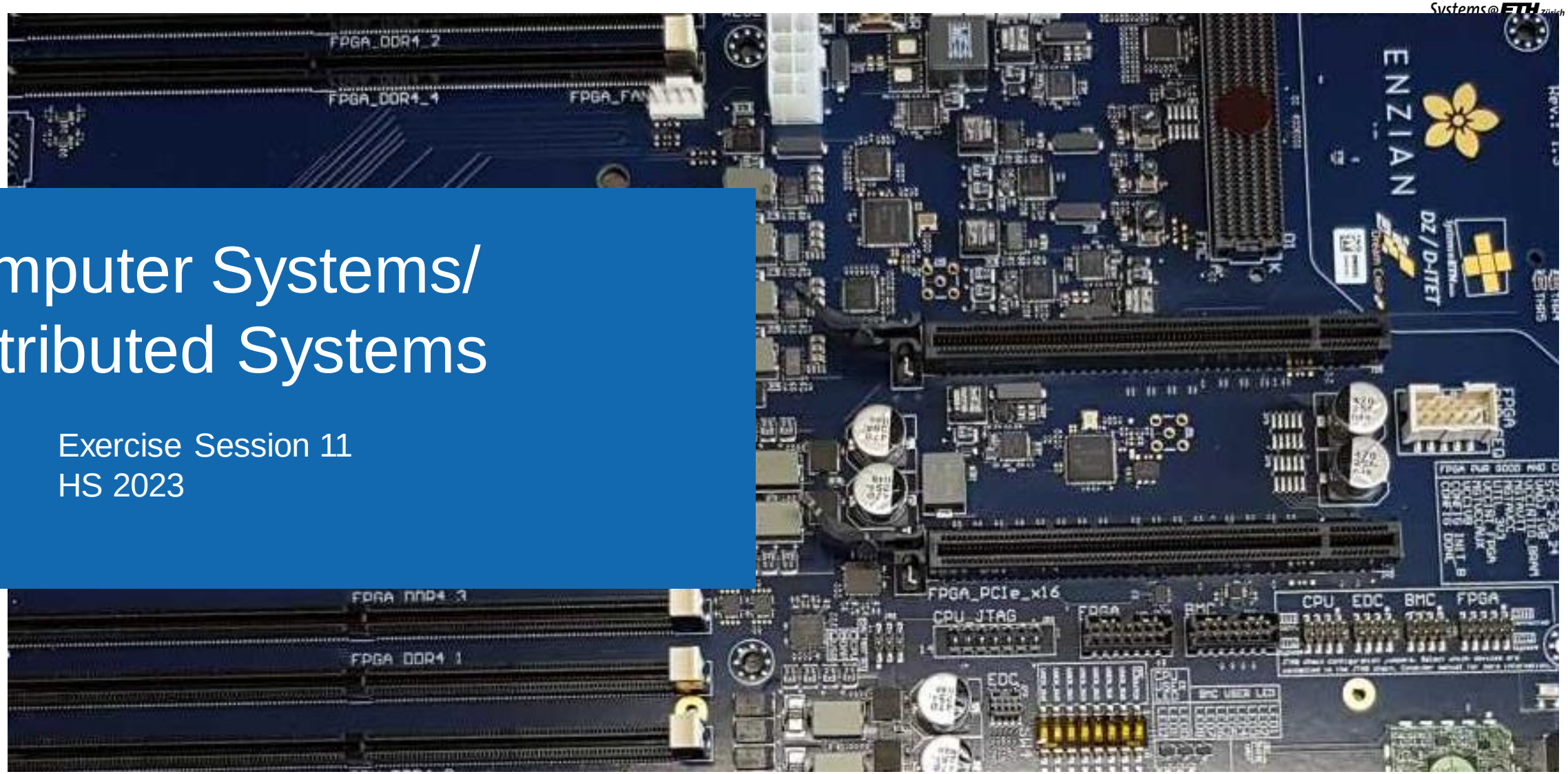
Distributed Computing

Slides last updated:
04.12.2023

Computer Systems/ Distributed Systems

Exercise Session 11
HS 2023



Consistency models

- Linearizable
- Sequentially Consistency
- Quiescent Consistency

Theorem:

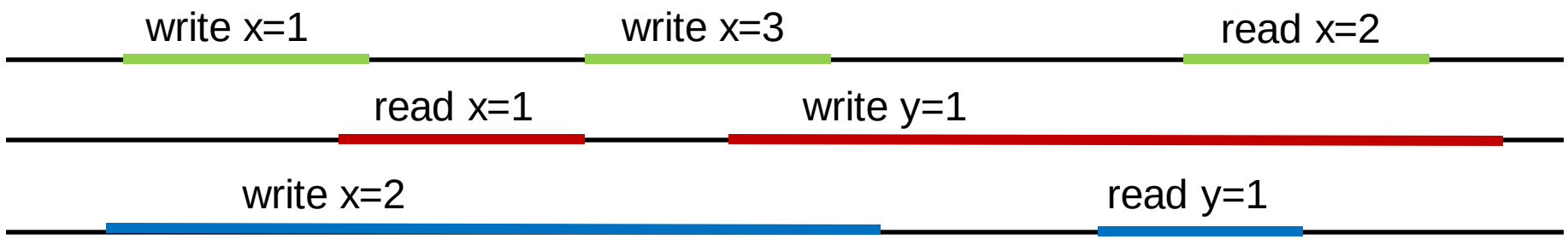
Linearizable implies both sequentially and quiescent consistency.

Linearizable

- “one global order”
- Linearizable → put points on a “line”
- Strongest assumption, implies other two

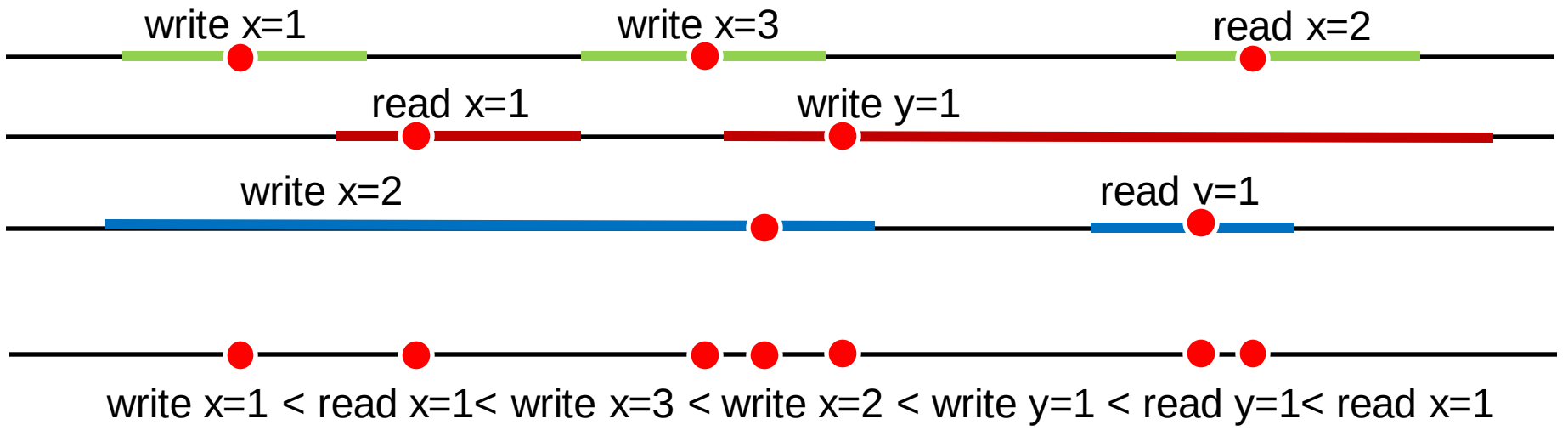
Linearizable

- “one global order”
- Linearizable → put points on a “line”
- Strongest assumption, implies other two



Linearizable

- “one global order”
- Linearizable → put points on a “line”
- Strongest assumption, implies other two

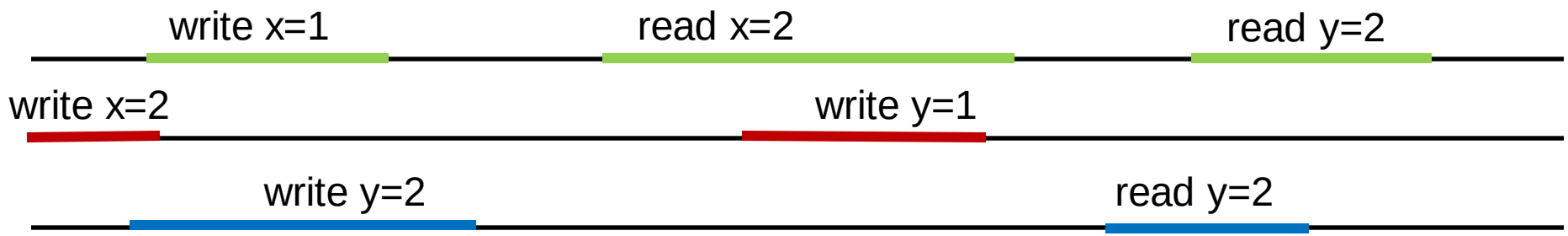


Sequential Consistency

- “per thread order”
- Sequential consistency → build “sequences”

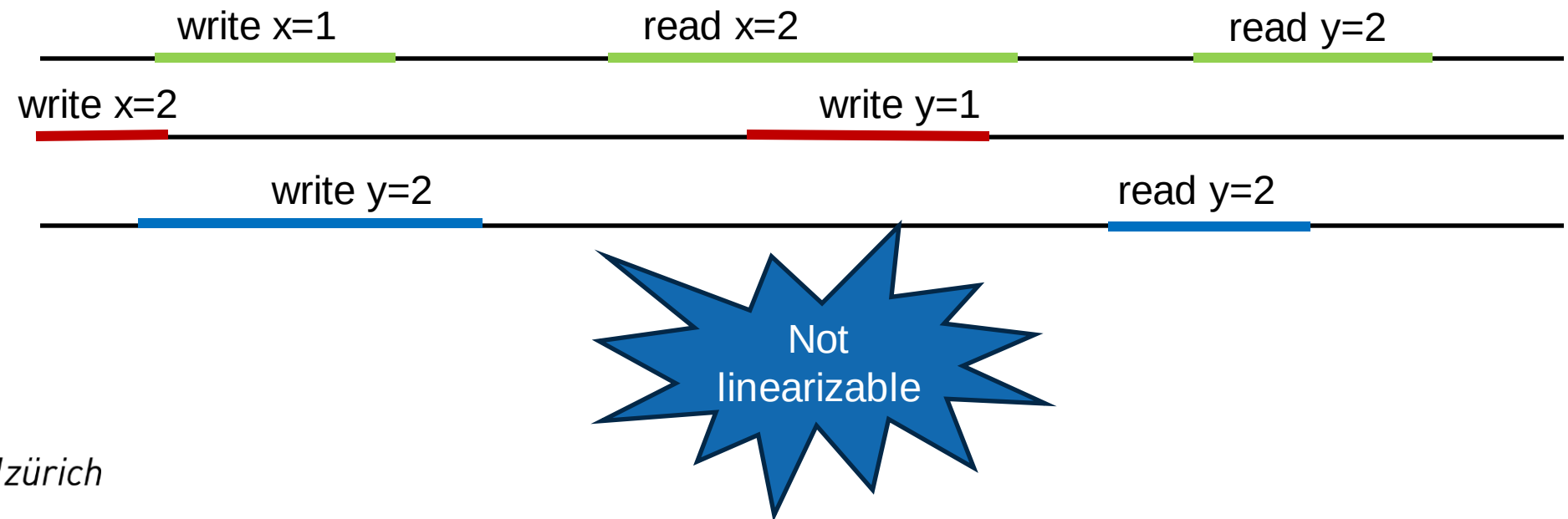
Sequential Consistency

- “per thread order”
- Sequential consistency → build “sequences”



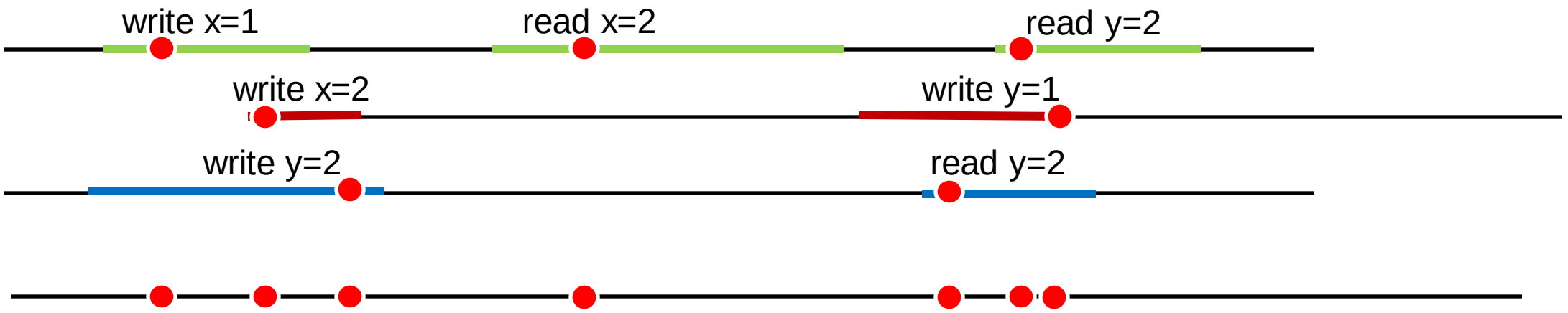
Sequential Consistency

- "per thread order"
- Sequential consistency → build "sequences"



Sequential Consistency

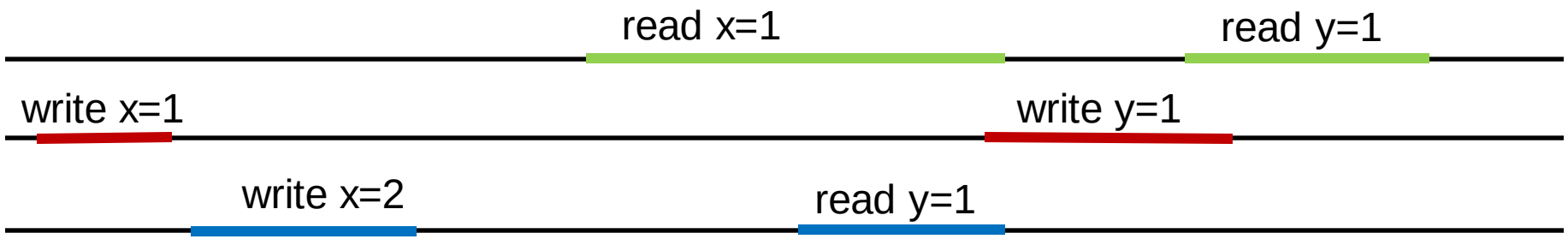
- "per thread order"
- Sequential consistency → build "sequences"



write x=1 < write x=2 < write y=2 < read x=2 < read y=2 < read y=2 < write y=1

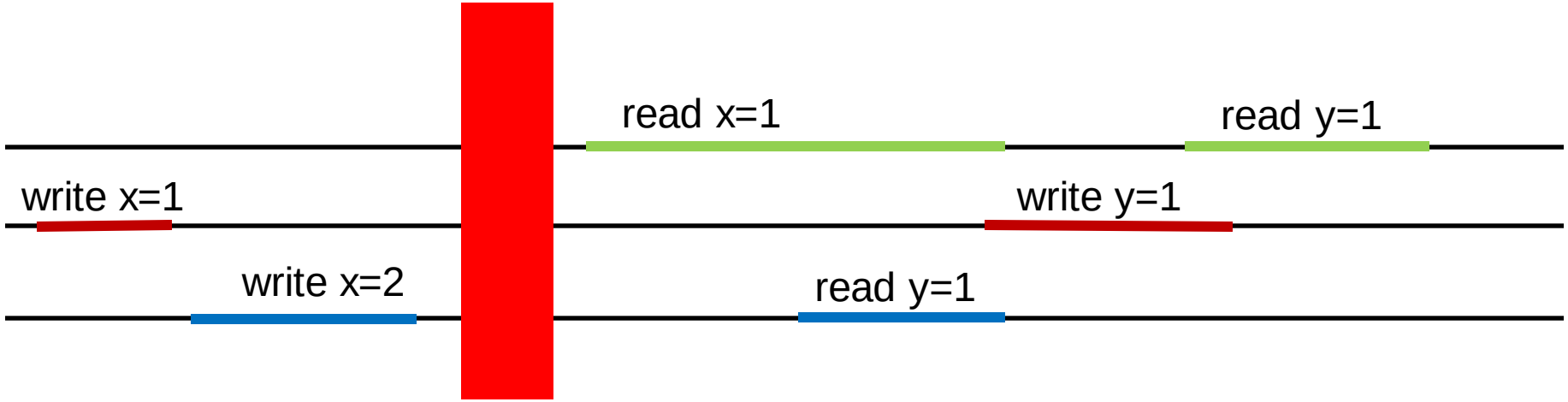
Quiscent Consistency

- Synchronizes all threads on quiescent point, i.e. point where no execution happens



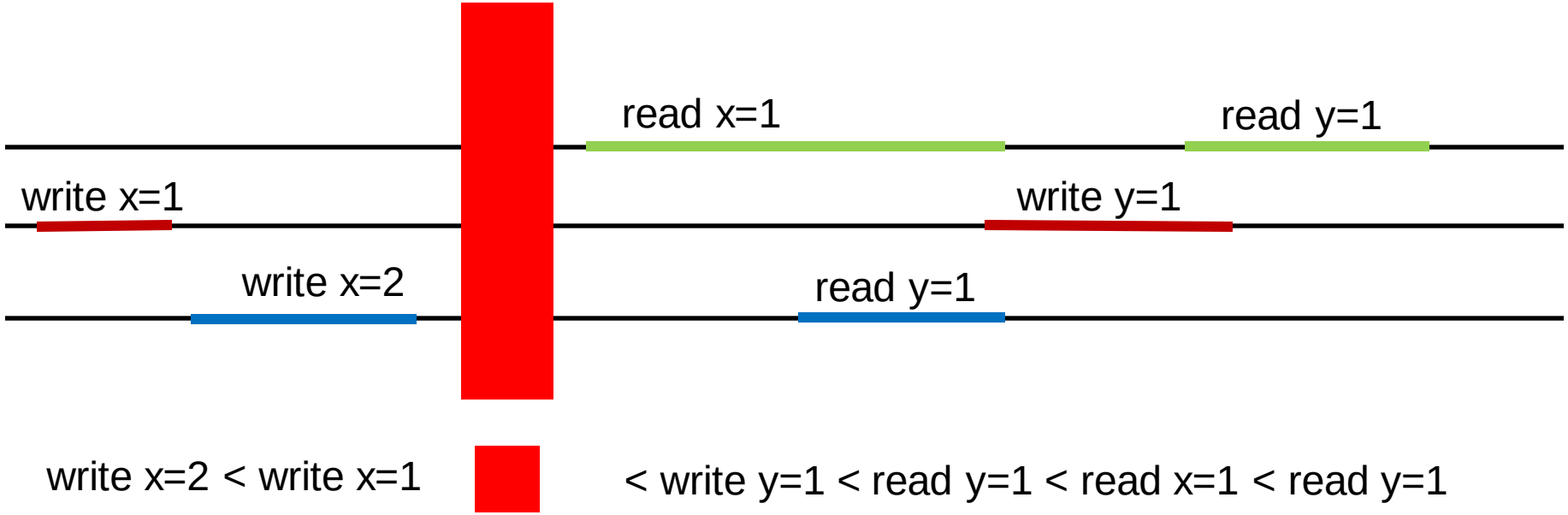
Quiscent Consistency

- Synchronizes all threads on quiescent point, i.e. point where no execution happens



Quiscent Consistency

- Synchronizes all threads on quiescent point, i.e. point where no execution happens

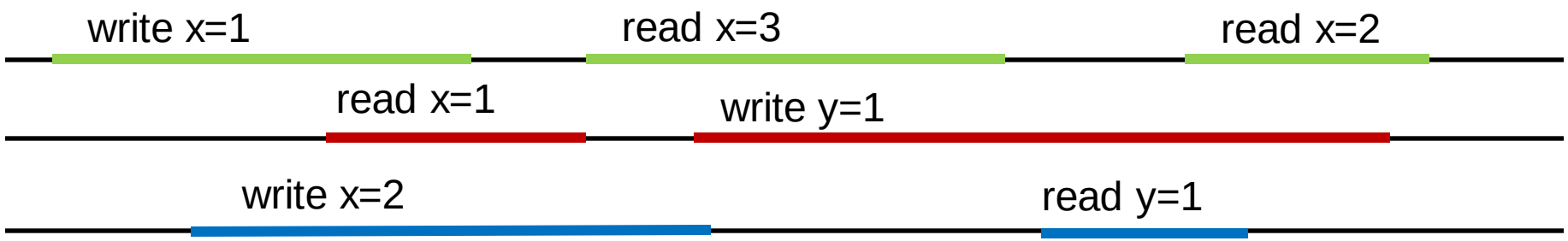


Composable Consistency

- Definition: If you only look at all operations concerning any object and the execution is consistent, then also the whole execution is consistent
- Sequentially consistent is not composable
- Linearizability is composable
- Quiescent consistency is composable

Composable Consistency

- Definition: If you only look at all operations concerning any object and the execution is consistent, then also the whole execution is consistent
- Sequentially consistent is not composable
- Linearizability is composable
- Quiescent consistency is composable



Logical Clocks:

- Happened before relation "→" holds

1) IF f < g on the same node

2) Send happens before receive

3) If f → g and g → h, then f→ h (transitivity)

Logical Clocks:

- Happened before relation "→" holds

1) IF f < g on the same node

2) Send happens before receive

3) If f → g and g → h, then f→ h (transitivity)

- C(a): timestamp of event a

Logical Clocks:

- Happened before relation "→" holds

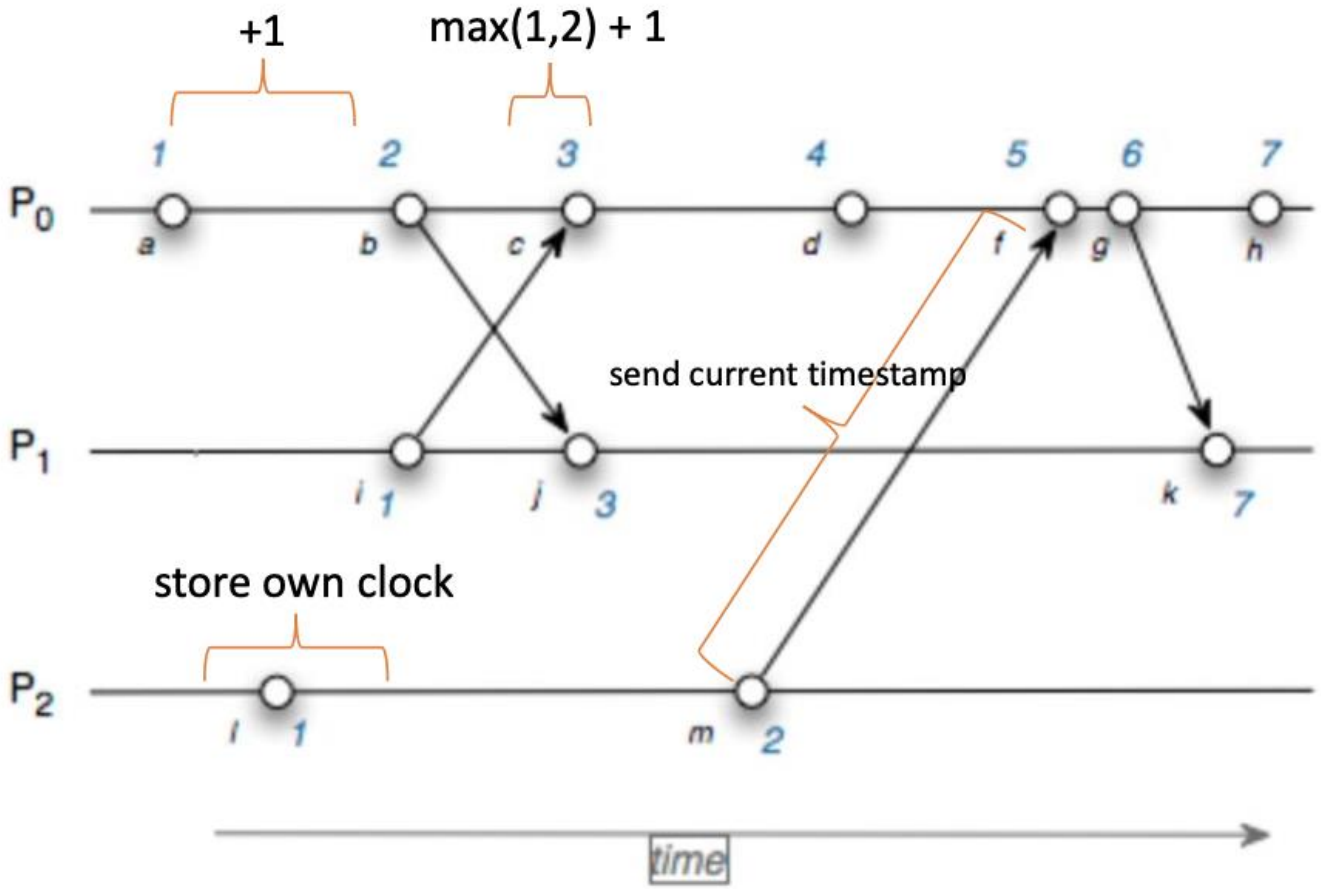
1) IF f < g on the same node

2) Send happens before receive

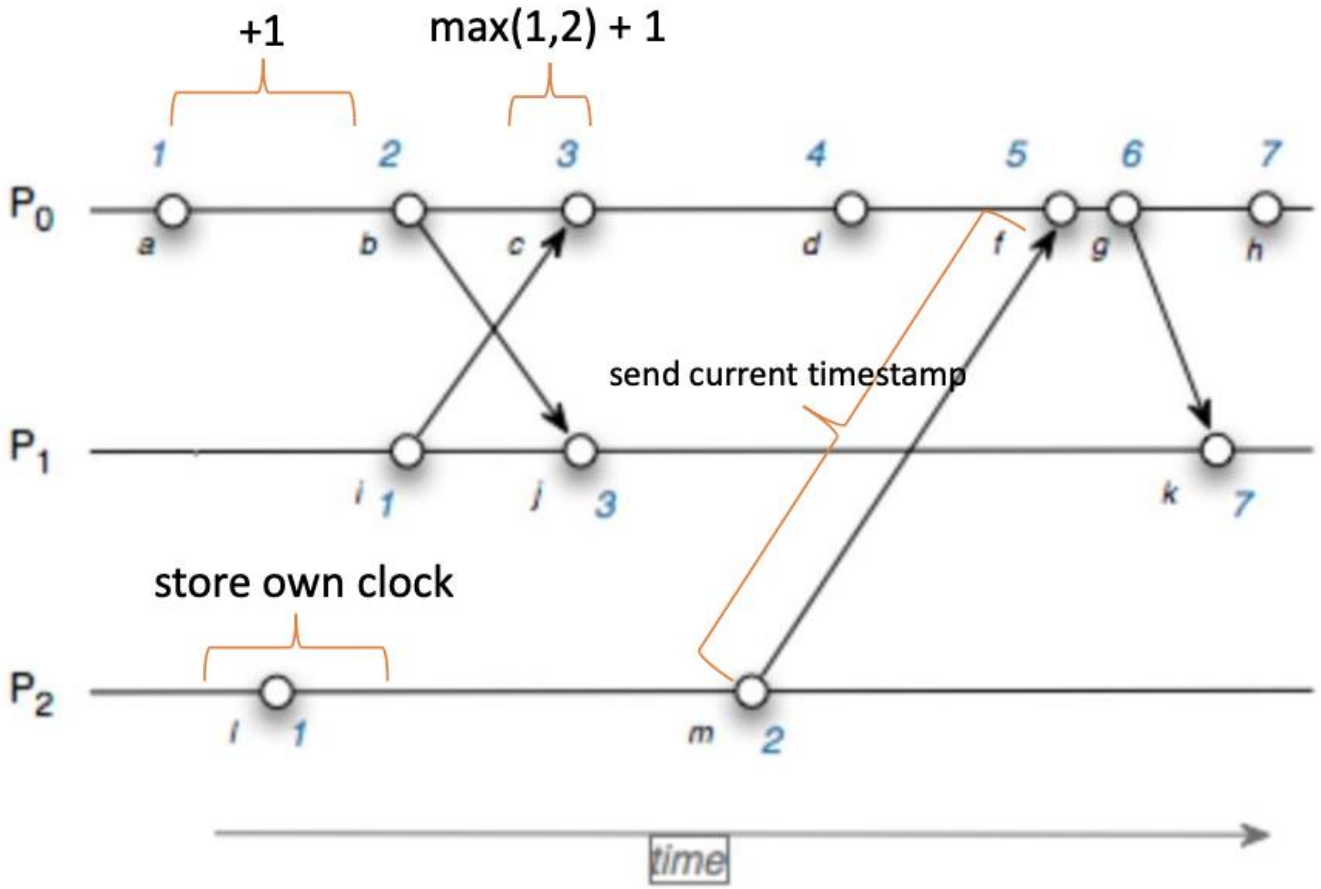
3) If f → g and g → h, then f→ h (transitivity)

- C(a): timestamp of event a
- **logical clocks: a→ b implies c(a) < c(b)**
- **Strong logical clock: c(a) < c(b) implies a→ b** (in addition)

Lamport Clocks:

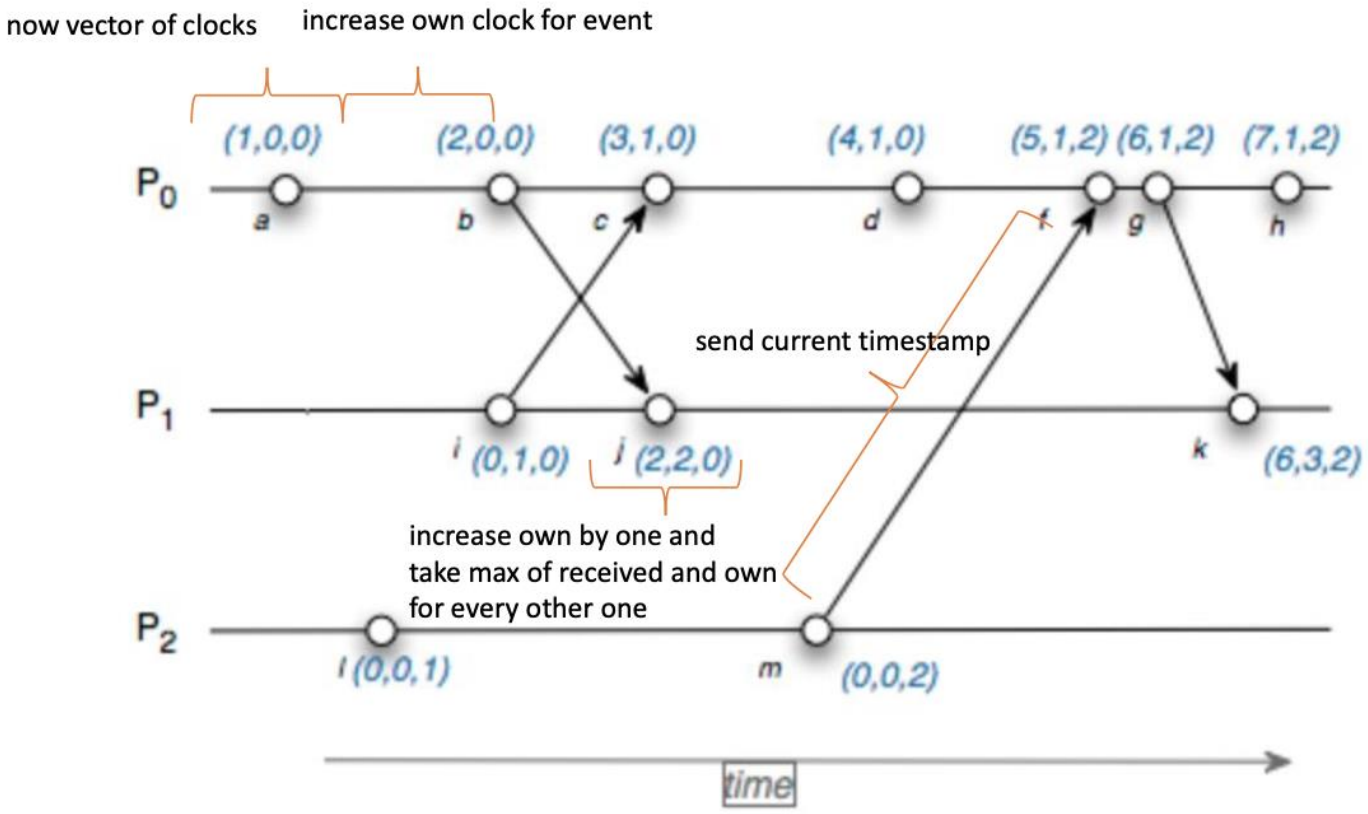


Lamport Clocks:



Weak logical clock: a → b implies c(a) < c(b) but not vice versa

Vector Clocks:



Vector Clocks:

- What **does c(a) < c(b) mean now?**

- if all the entries in a<= b and at least one entry where a < b

- Is a **logical clock** (so if a → b then c(a) < c(b))

- Is also a **strong logical clock** (if c(a) < c(b) -> a → b)

Intuition: because in order to achive c(a) < c(b), all entries have to be at least as big, so a message from a must have reached b (not necessarily directly) so that b has the right value

Consistent Snapshot:

- **Cut:** prefix of a distributed execution
- **Consistent Snapshot:**

a cut where for every operation g in that cut, if f → g, then the cut contains f

→ if all "connected" preceding operations are included

- With number of consistent snapshots, one can make conclusions about degrees of concurrency in system

Time & Clocks

- Wall clock time: “true” time
- clock error

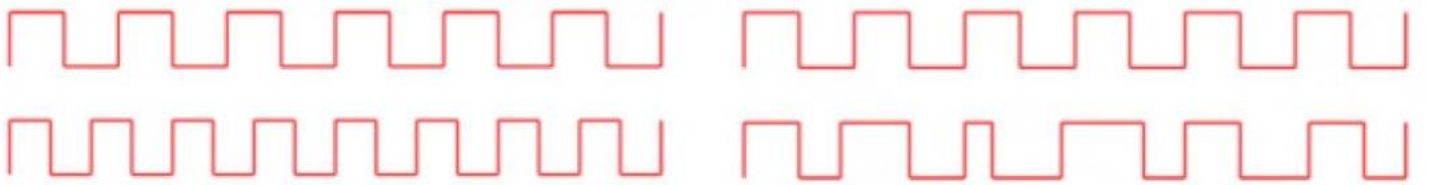
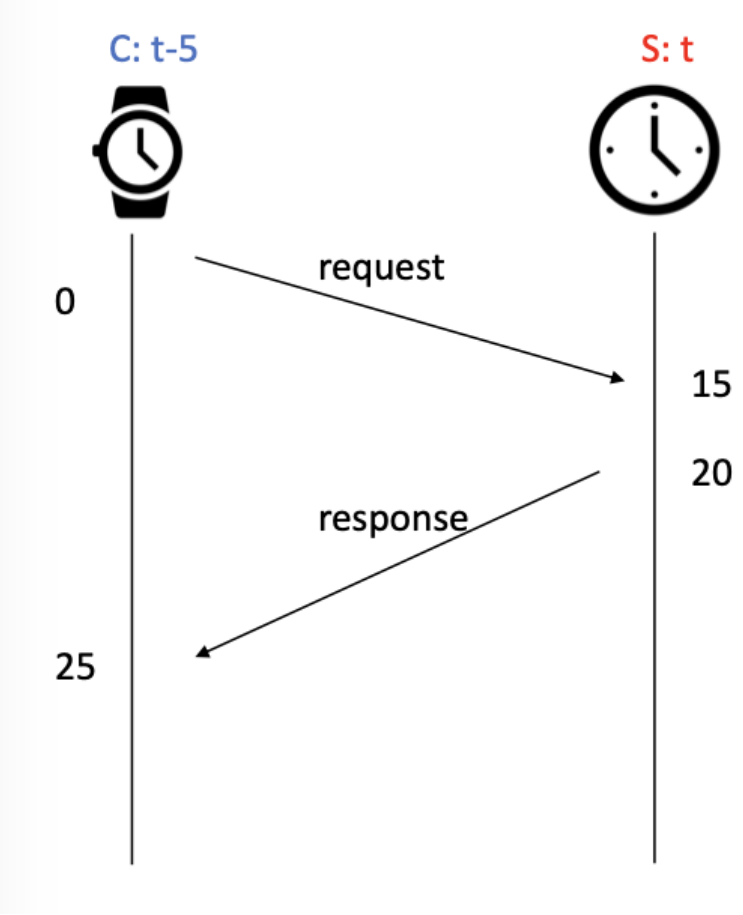


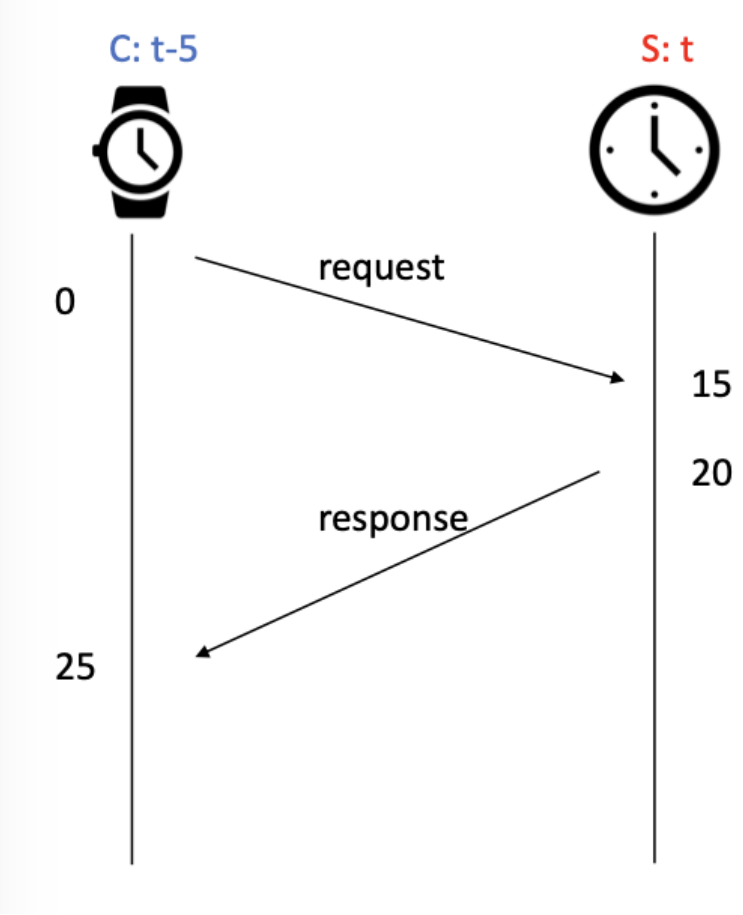
Figure 20.8: Drift (left) and Jitter (right). On top is a square wave, the wall-clock time t^*.

NTP



Propagation delay = ((25-0) – (20-15)) / 2 = 10

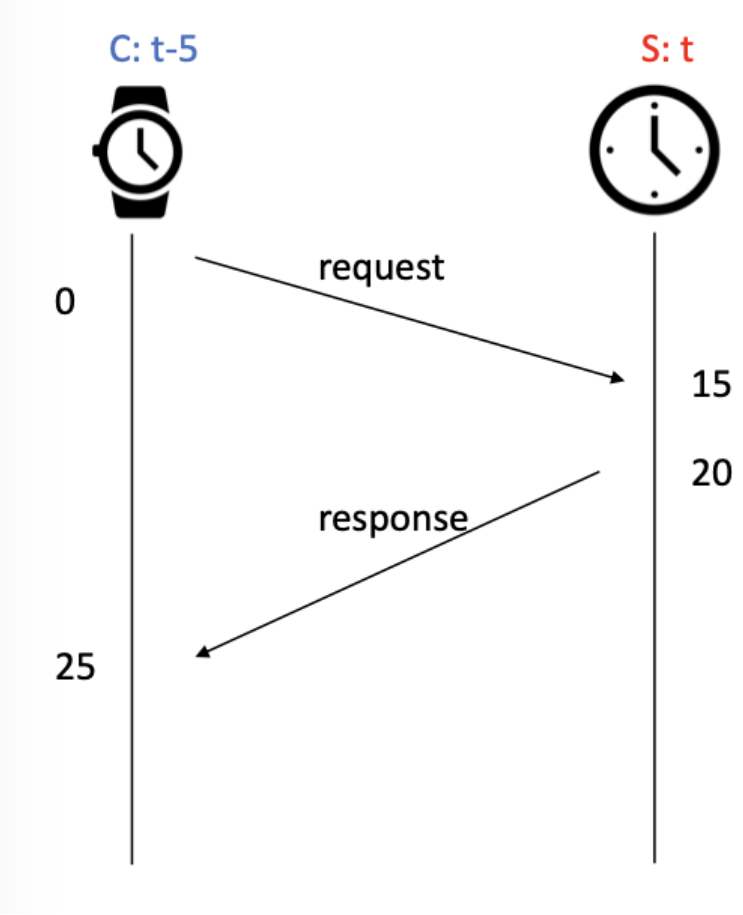
NTP



Propagation delay = ((25-0) – (20-15)) / 2 = 10

Assumed to be symmetric!

NTP



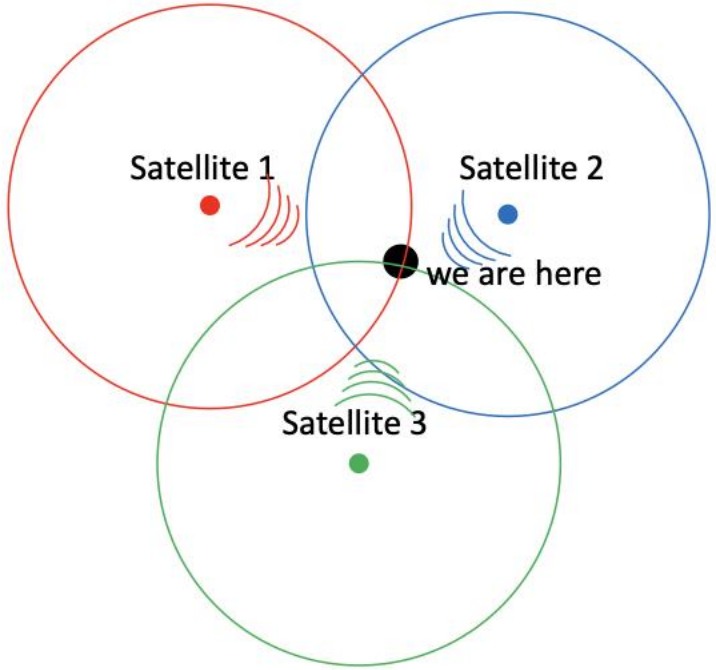
Propagation delay = ((25-0) – (20-15)) / 2 = 10
Assumed to be symmetric!

Skew = ((15-0) + (20-25))/2 = 5
→ Adjust clock by skew

Alternative Synchronization Protocols

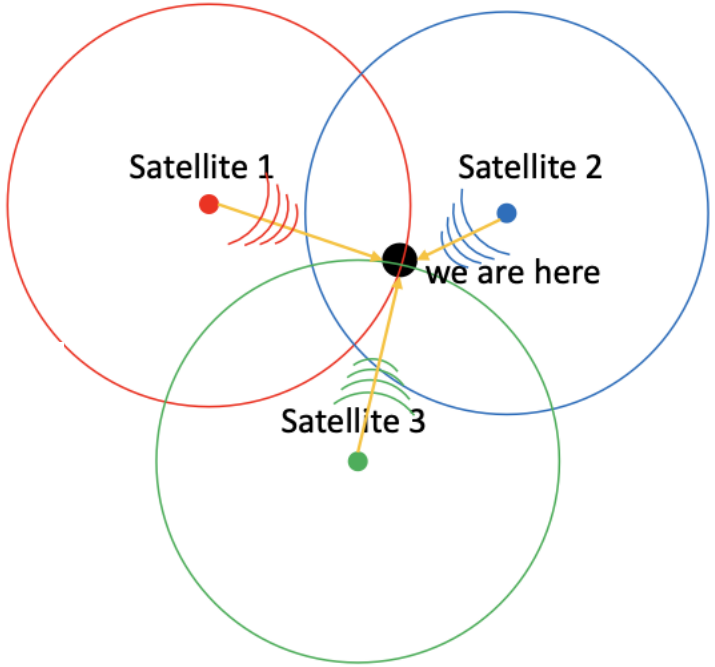
- Precision Time Protocol
- Local Time Synchronization
- Wireless Clock Synchronization with Known Delays

GPS - General Idea



GPS - General Idea

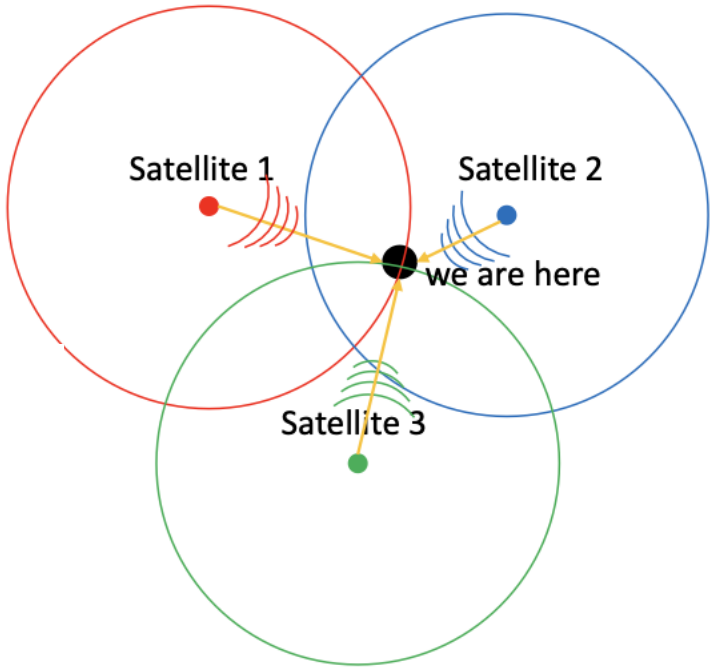
Satellites transmit location and timestamp when sent



GPS - General Idea

Satellites transmit location and timestamp when sent

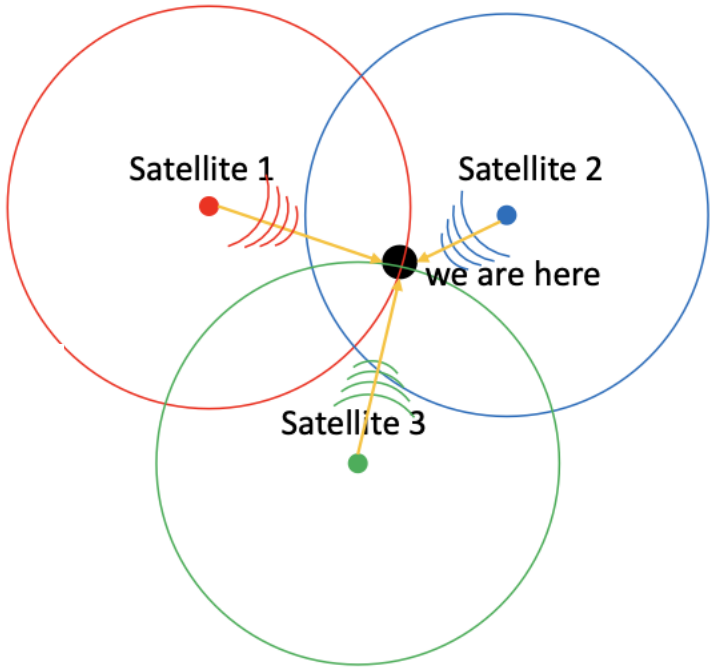
→ Compare satellite timestamp to local timestamp and calculate distance



GPS - General Idea

Satellites transmit location and timestamp when sent

→ Compare satellite timestamp to local timestamp and calculate distance

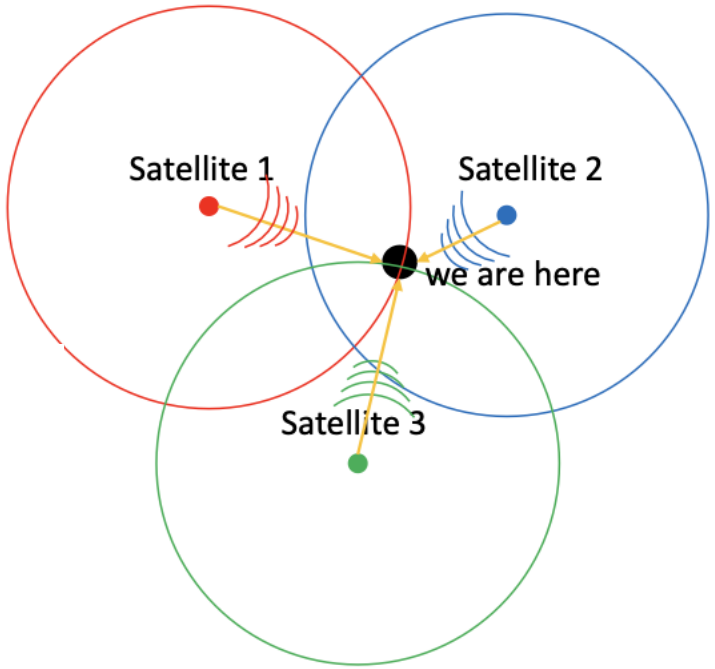


Problem: Quadratic equation results in two possible locations

GPS - General Idea

Satellites transmit location and timestamp when sent

→ Compare satellite timestamp to local timestamp and calculate distance



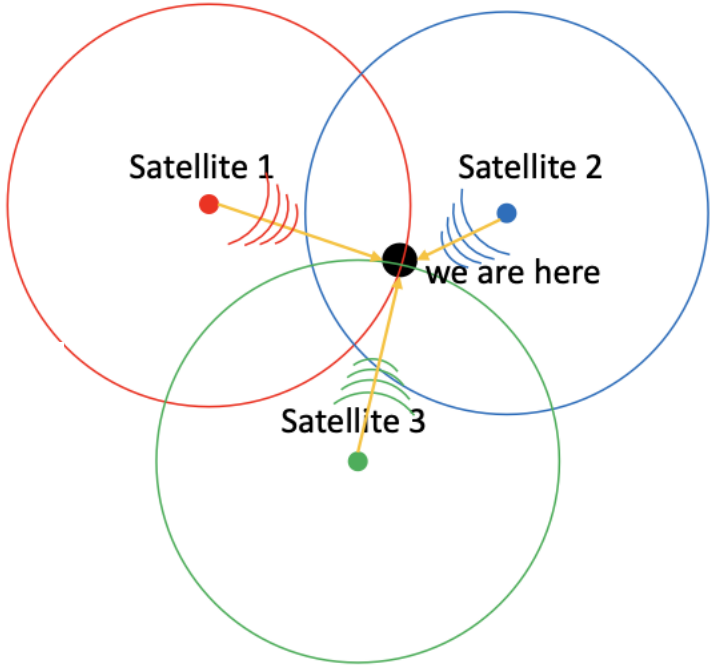
Problem: Quadratic equation results in two possible locations

Solution: Generally, only one close to Earth's surface

GPS - General Idea

Satellites transmit location and timestamp when sent

→ Compare satellite timestamp to local timestamp and calculate distance

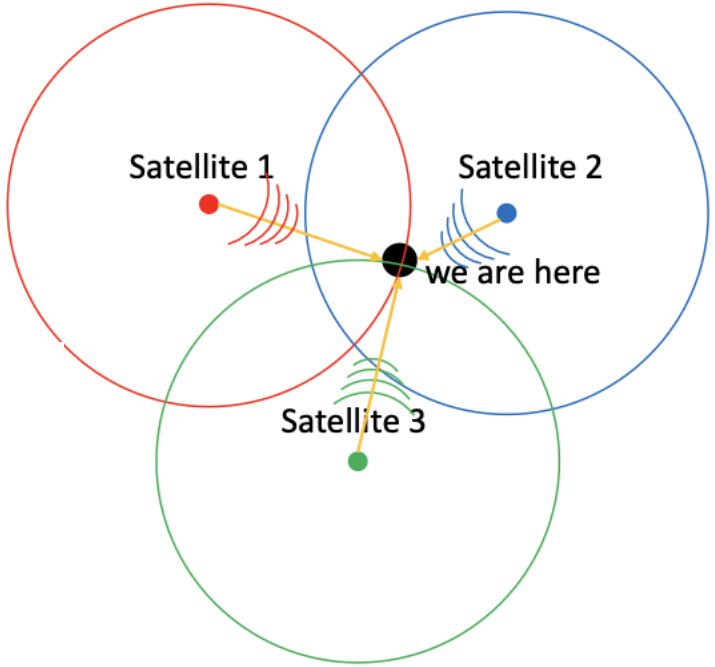


Problem: We do not have the same time as the satellite, so calculating the distance might not be accurate

GPS - General Idea

Satellites transmit location and timestamp when sent

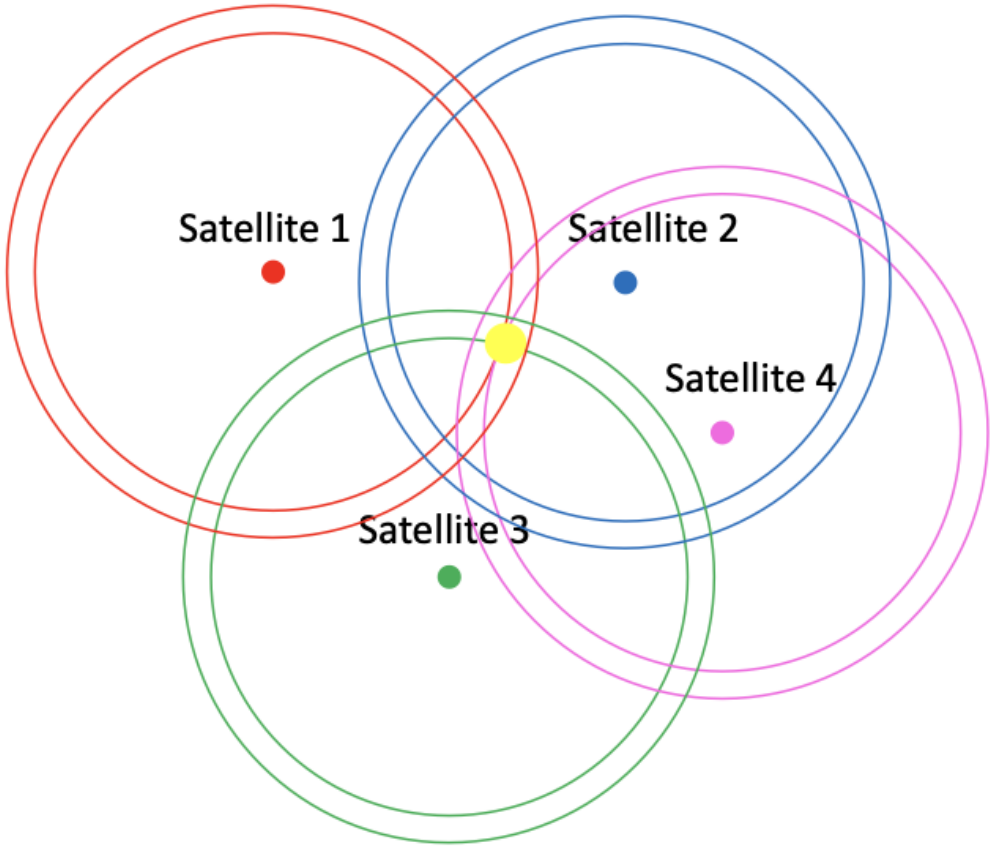
→ Compare satellite timestamp to local timestamp and calculate distance



Problem: We do not have the same time as the satellite, so calculating the distance might not be accurate

Solution: take measurements from forth satellite!

GPS - Redefined



Quiz

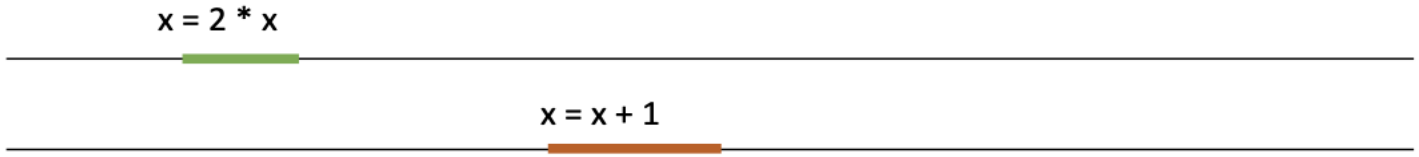
1. Does sequential consistency imply quiescent consistency?

2. Are there guarantees a Lamport clock can achieve a vector clock cannot?

3. Does a high number of consistent snapshots imply a high level of concurrency?

Quiz

1. Does sequential consistency imply quiescent consistency? - **Wrong**



e.g. x=1.5 is a valid outcome for sequential consistency, but not quiescent

2. Are there guarantees a Lamport clock can achieve a vector clock cannot?

No, because the concept of a Lamport clock is included in the vector clock concept

3. Does a high number of consistent snapshots imply a high level of concurrency? **- True**

Quiz

4. What is the difference between jitter and drift?

Quiz

4. What is the difference between jitter and drift?

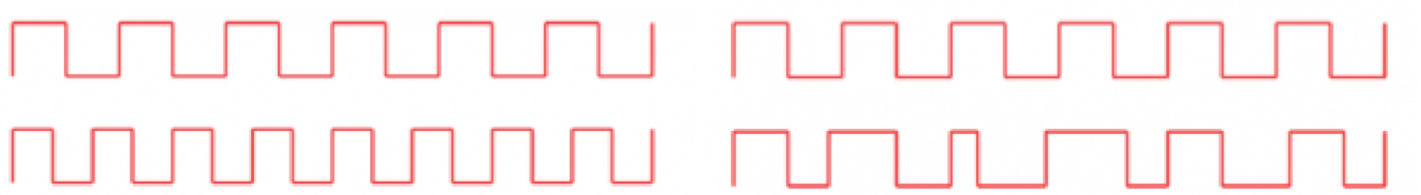


Figure 22.8: Drift (left) and Jitter (right). On top is a square wave, the wall-clock time t^*.

Assignment 9 Outlook:

1 Clock Sync

Quiz

1.1 Clock Synchronization

a) Assume you run NTP to synchronize speakers in a soccer stadium. Each speaker has a radio downlink to receive digital audio data. However, there is no uplink! You decide to use an acoustic signal transmit by the speaker. To synchronize its clock, a speaker first plays back an acoustic signal. This signal is picked up by the NTP server which responds via radio. The speaker measures the exact time that passes between audio playback and radio downlink response. What is likely the largest source of error?

b) What are strategies to reduce the effect of this error source?

c) Prove or disprove the following statement: If the average local skew is smaller than x, then so is the average global skew.

d) Prove or disprove the following statement: If the average global skew is smaller than x, then so is the average local skew.

Assignment 9 Outlook:

1.2 Time Difference of Arrival

Assume you are located on a line $y = -x + 8$ km in the two dimensional plane. You receive the GPS signals from satellites A and B. Both signals are transmitted exactly at the same time t by both satellites. You receive the signal from satellite A 3.3 µs before the signal of satellite B. At time t, satellite A is located at $p_A = (6\text{ km}, 6\text{ km})$ and satellite B is located at $p_B = (2\text{ km}, 1\text{ km})$, in the plane.

a) Formulate the least squares problem to find your location.

b) Are you more likely to be at position (2 km, 6 km) or (4 km, 4 km)?

c) What is the time when receiving the signal from satellite B?

Assignment 9 Outlook:

1.3 Clock Synchronization: Spanning Tree

Common clock synchronization algorithms (e.g. TPSN, FTSP) rely on a spanning tree to perform clock synchronization. Finding a good spanning tree for clock synchronization is not trivial. Nodes which are neighbors in the network graph should also be close-by in the resulting tree. Show that in a grid of $n = m \times m$ nodes there exists at least a pair of nodes with a stretch of at least m. The stretch is defined as the hop distance in the tree divided by the distance in the grid.

Assignment 9 Outlook:

1.4 NTP Programming

Write a Linux program that prints the current UTC time and the maximum error.

Hint: Have a look at the manpage for `adjtimex`.

Assignment 9 Outlook:

1.4 NTP Programming

Write a Linux program that prints the current UTC time and the maximum error.

Hint: Have a look at the manpage for `adjtimex`.

Assignment 9 Outlook:

2 Consistency and Logical Clocks

Quiz

2.1 Different Consistencies

Prove or disprove the following statements:

a) Neither sequential consistency nor quiescent consistency imply linearizability.

b) If a system has sequential consistency **and** quiescent consistency, it is linearizable.

Assignment 9 Outlook:

2.2 Measure of Concurrency from Vector Clocks

You are given two nodes that each have a vector logical clock that additionally logs the clock state upon receiving a message (see Algorithm 1).

Algorithm 1 Vector clocks with logging

1: (Code for node u)
2: Initialize $c_u[v] := 0$ for all other nodes v.
3: Upon local operation: Increment current local time $c_u[u] := c_u[u] + 1$.
4: Upon send operation: Increment $c_u[u] := c_u[u] + 1$ and include the whole vector c_u as d in message.
5: Upon receive operation: Extract vector d from message and update $c_u[v] := \max(d[v], c_u[v])$ for all entries v. Increment $c_u[u] := c_u[u] + 1$. Save the vector c_u to the log file of node u.

Assume that exactly one message gets send from one to the other node. Given the logs and current vector states of both nodes, write a short program that calculates the measure of concurrency as defined in the script (Definition 19.30). You can use your favorite programming language. The example solution will be in Python.

Advanced

Generalize your program to any number of messages exchanged between the nodes.