Distributed Systems

Exercise Session 3

Selma Steinhoff: selmas@ethz.ch

Disclaimer: these are not official slides, there might be mistakes, treat them this way

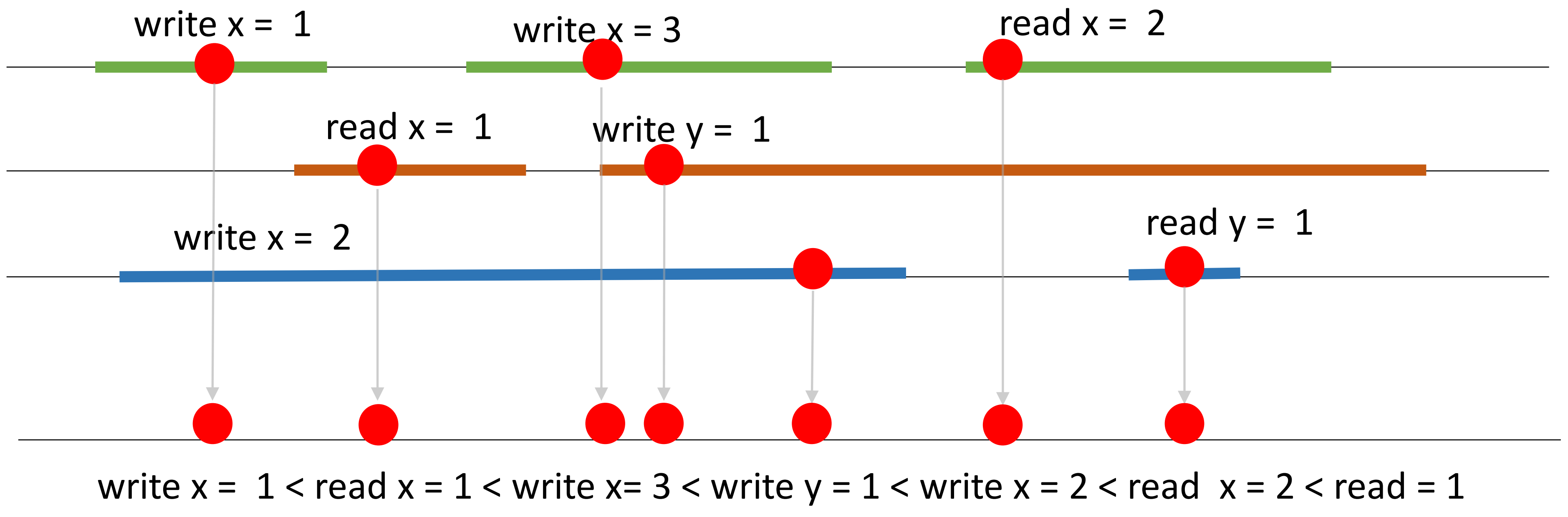
Consistency Models

- **Linearizability** (implies both others)
- **Sequential Consistency**
- **Quiescent Consistency**

- If you are confused or need an overview:
  - http://coldattic.info/post/88/

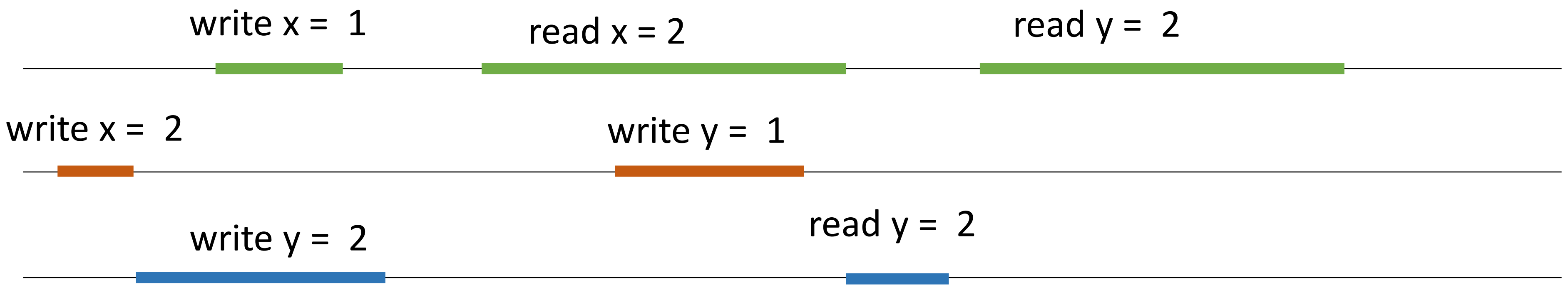
Linearizability

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


write x = 1
write x = 3
read x = 2
read x = 1
write y = 1
write x = 2
read y = 1
write x = 1 < read x = 1 < write x= 3 < write y = 1 < write x = 2 < read x = 2 < read = 1

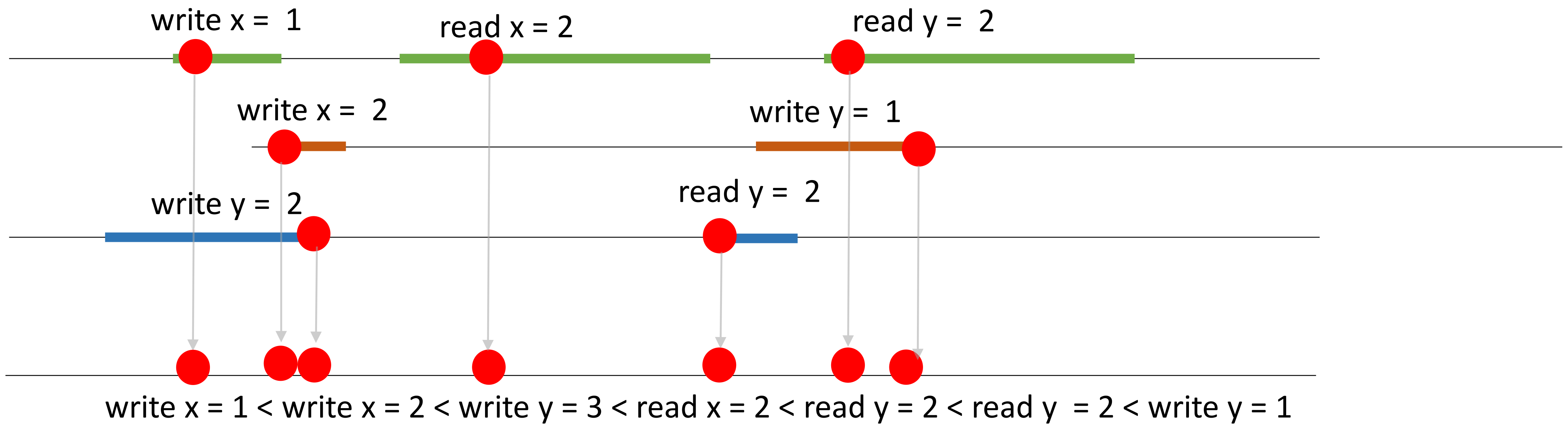
Sequential Consistency

- similar as linearizability, but can ”shift” and “squeeze” threads compared to each other
- sequential consistency -> build “sequences”



Sequential Consistency

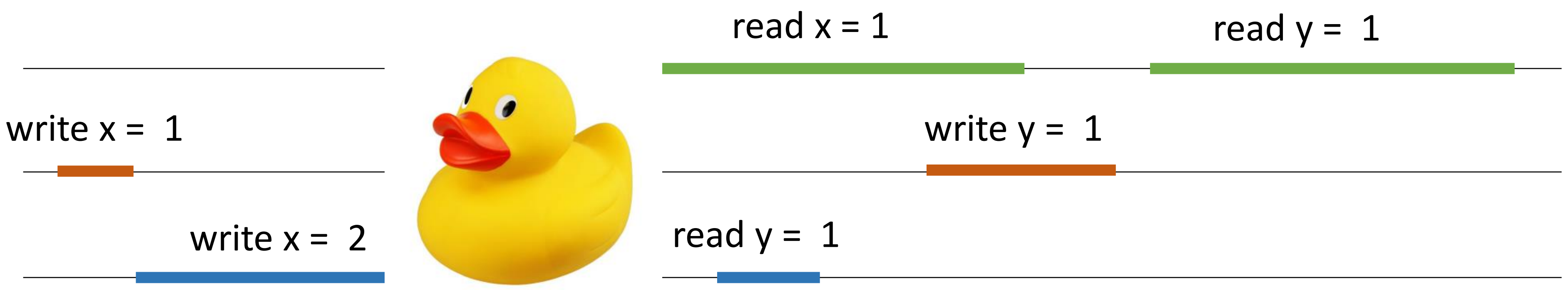
- similar as linearizability, but can ”shift” and “squeeze” threads compared to each other
- sequential consistency -> build “sequences”



write x = 1
read x = 2
read y = 2
write x = 2
write y = 1
write y = 2
read y = 2
write x = 1 < write x = 2 < write y = 3 < read x = 2 < read y = 2 < read y = 2 < write y = 1

Quiescent Consistency

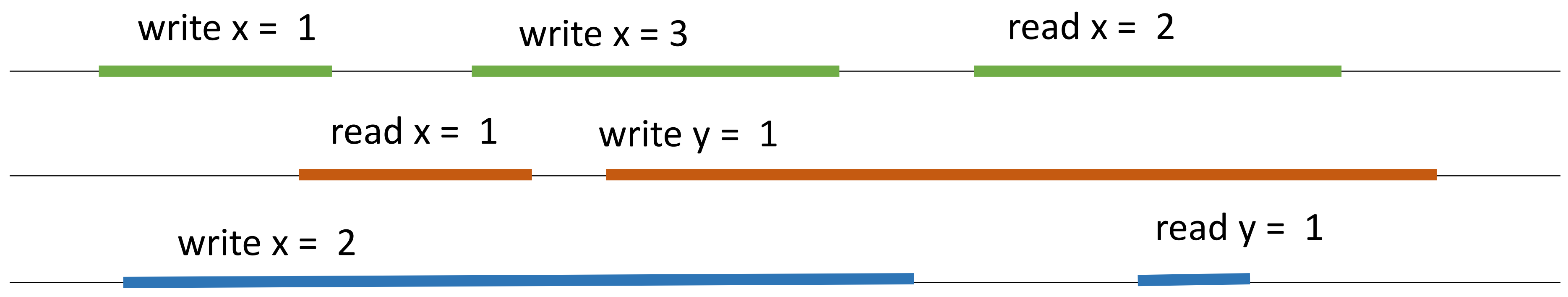
- synchronizes all threads whenever there is a time when there is no possible execution



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

Composable (applies to consistency models)

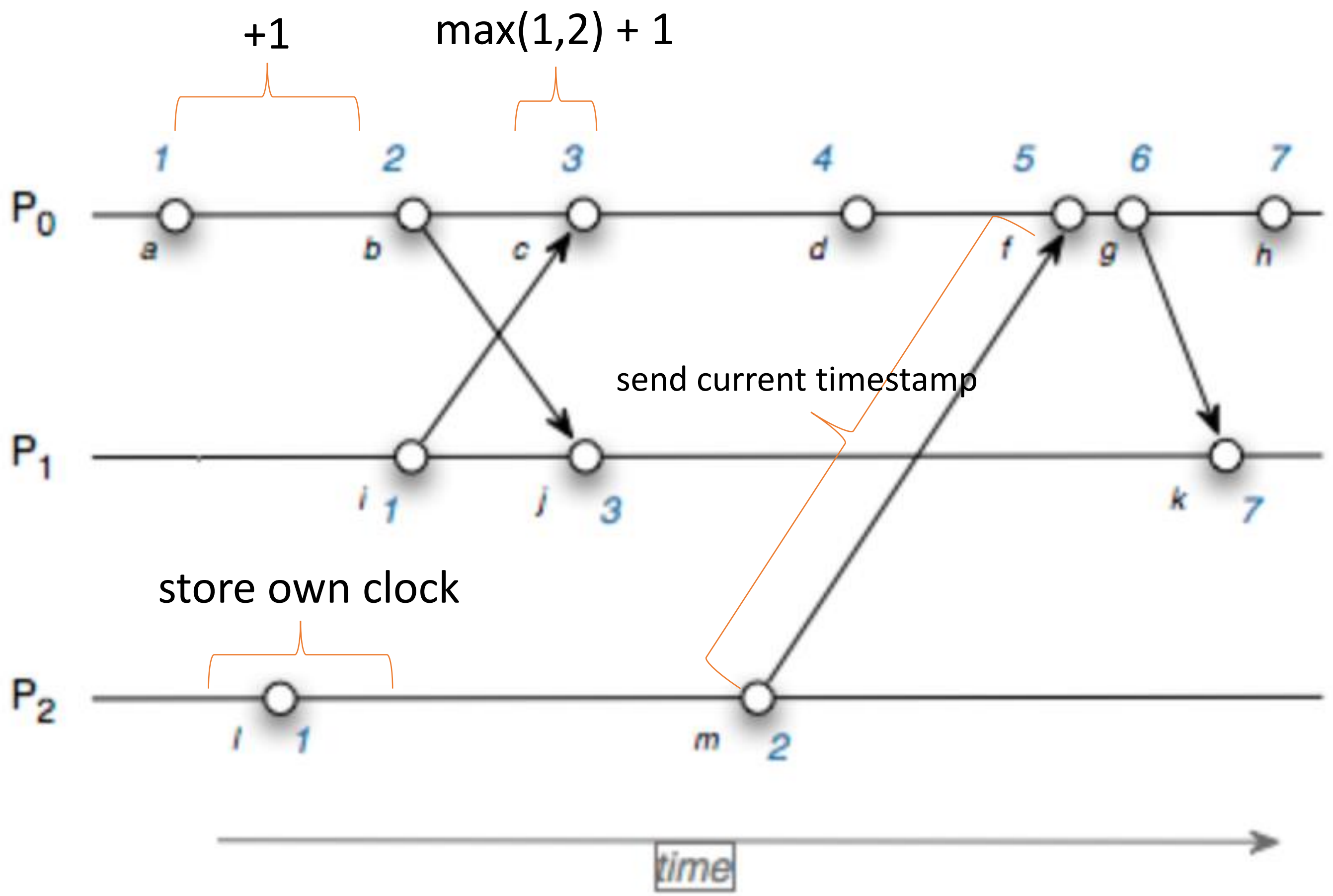
- Definition: If you only look at all operations concerning any object and the execution is consistent, then also the whole execution is consistent
- sequential consistency is **not** composable
- linearizability is composable
- quiescent consistency is composable



Logical Clocks

- happened before relation „->“ holds:
  - If f < g on the same node
  - Send happens before receive
  - If f -> g and g -> h then f -> h (Transitivity)
- c(a) means timestamp of event a
- **logical clock: if a -> b, then c(a) < c(b)**
- **strong logical clock: if c(a) < c(b), then a -> b** (in addition)

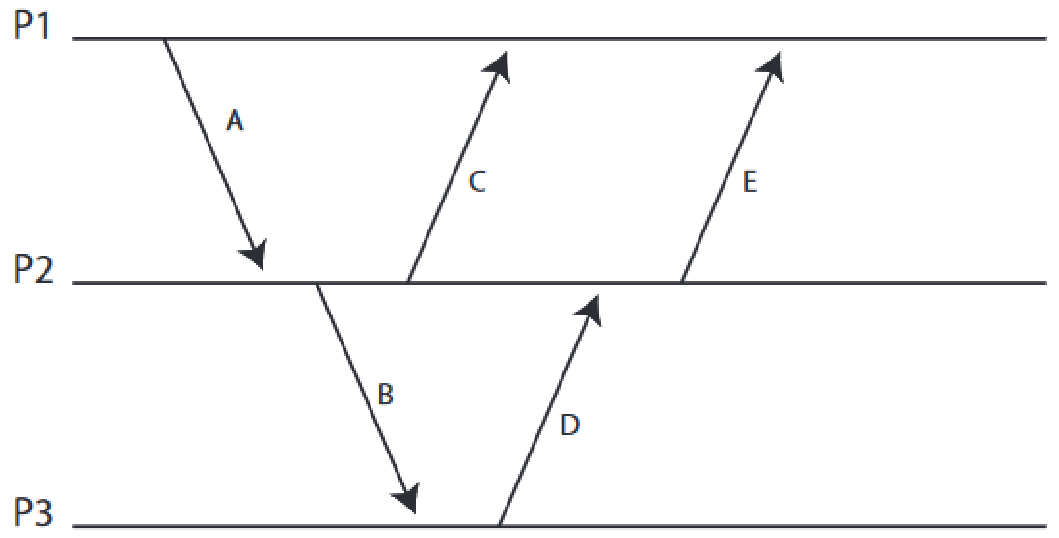
Lamport Clock



Lamport Clock

- Is a logical clock (so if a -> b then c(a) < c(b))
- but the reverse does not hold, so not a strong logical clock

Lamport Clock Hands-on



1) Find a pair of events for which we cannot make a statement about their causal dependencies
2) Insert a message N such that:

   A.receive < N.send ^ C(N.receive) < C(E.send)

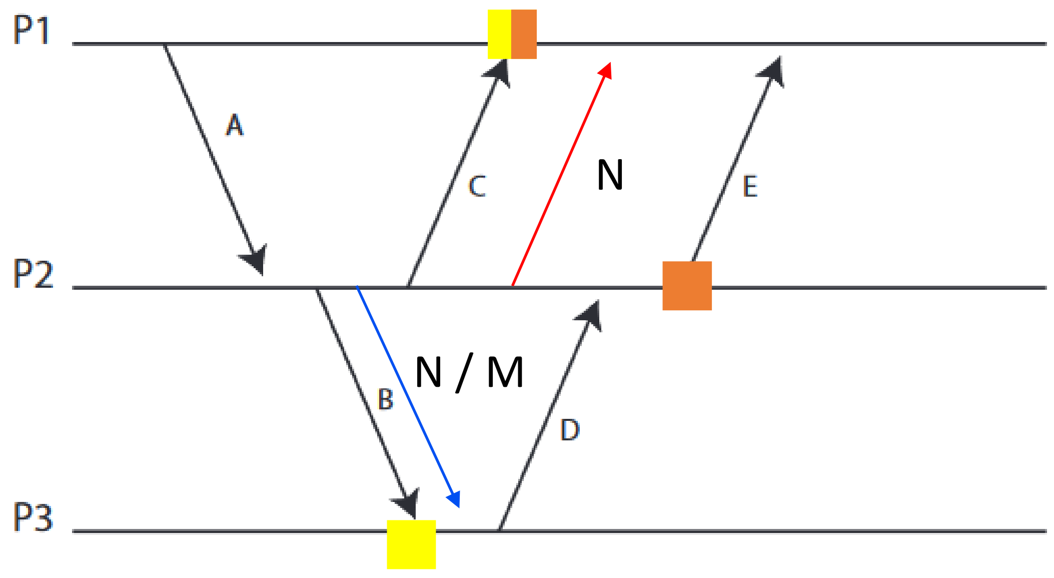
   Sender != receiver
3) Insert a message M such that:

   M.Send < C.send ^ C(B.receive) < C(M.receive)

   Send by P2, received by P3

Exercise from Distributed Systems by Prof. Mattern 2016

Lamport Clock Hands-on: Solutions



1) Find a pair of events for which we cannot make a statement about their causal dependencies 🟨 🟧
2) Insert a message N such that:

   A.receive < N.send ^ C(N.receive) < C(E.send)

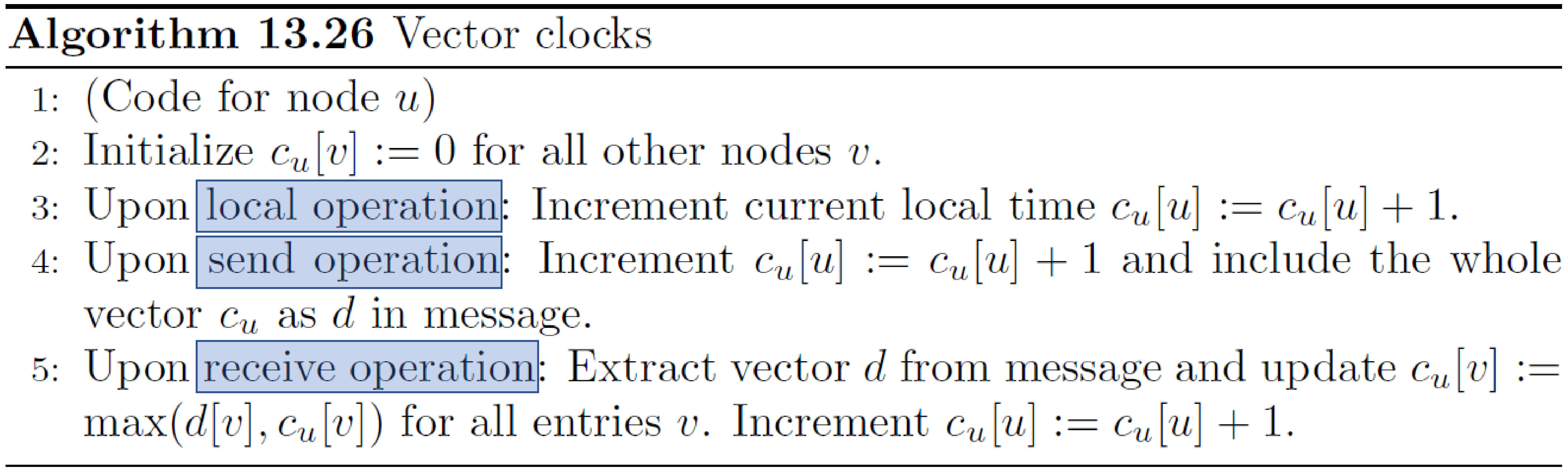
   Sender != receiver
3) Insert a message M such that:

   M.Send < C.send ^ C(B.receive) < C(M.receive)

   Send by P2, received by P3

Exercise from Distributed Systems by Prof. Mattern 2016

Vector Clock



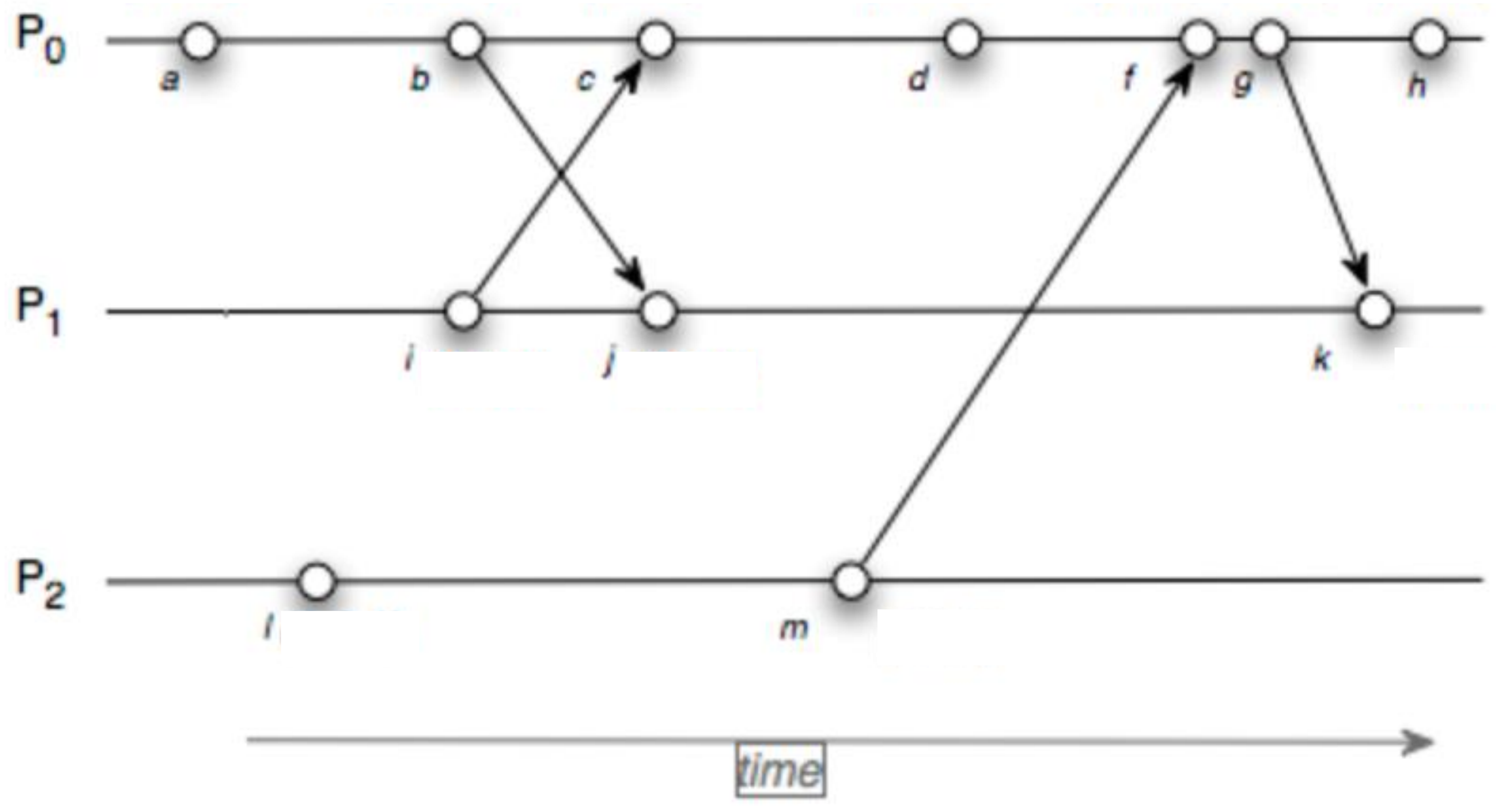
Algorithm 13.26 Vector clocks

```
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 $c_u[u] := c_u[u] + 1$.
```

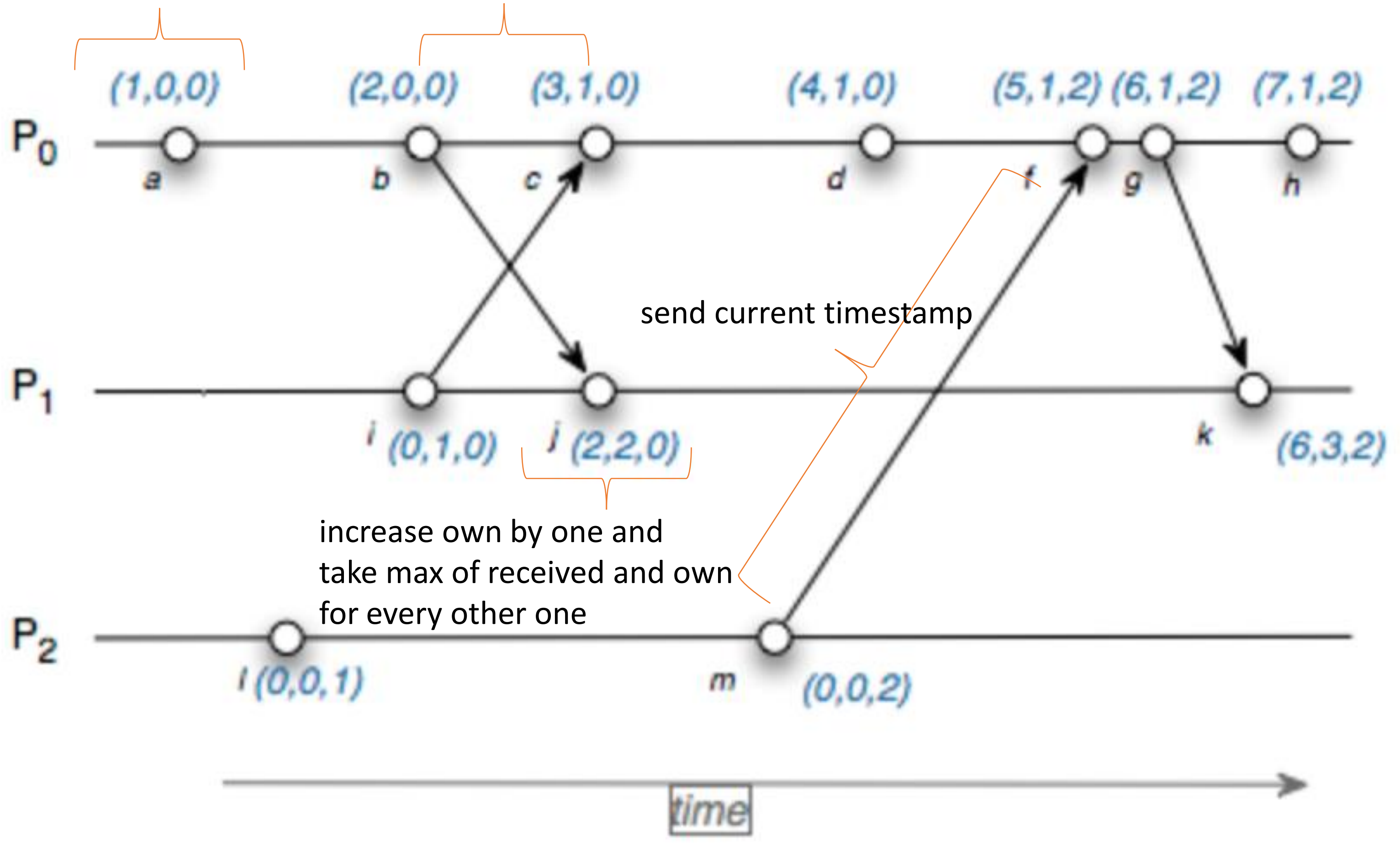
Vector Clock

- what does c(a) < c(b) mean now?
  - if all the entries are 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 achieve 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 a value

Vector Clock Hands-on



Vector Clock Hands-on: Solution



now vector of clocks

increase own clock for event

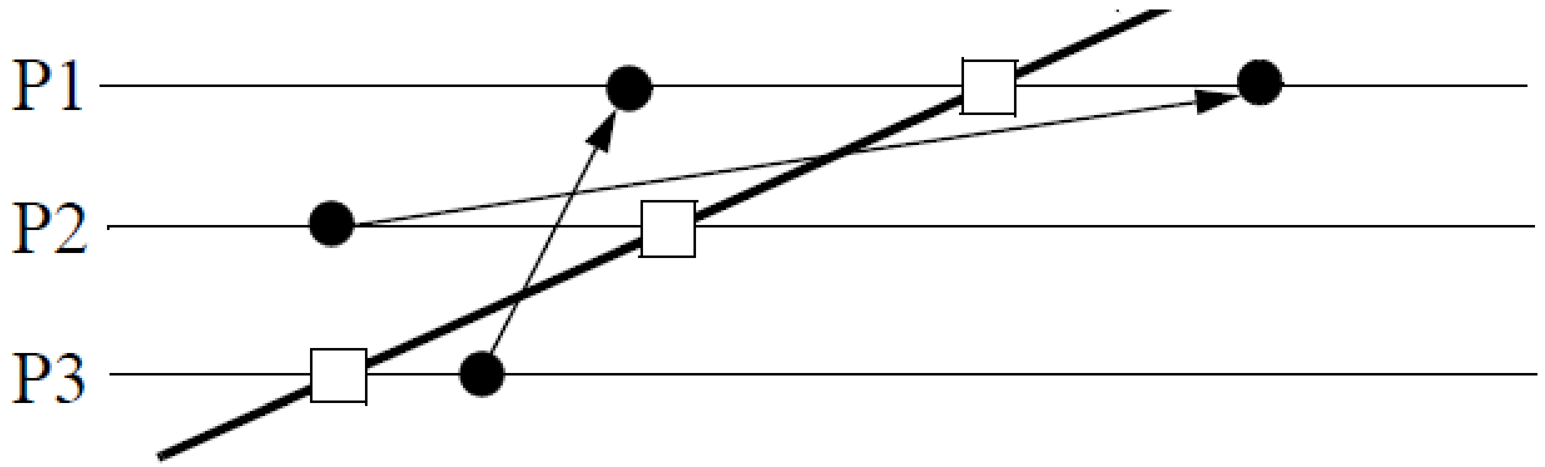
send current timestamp

increase own by one and
take max of received and own
for every other one

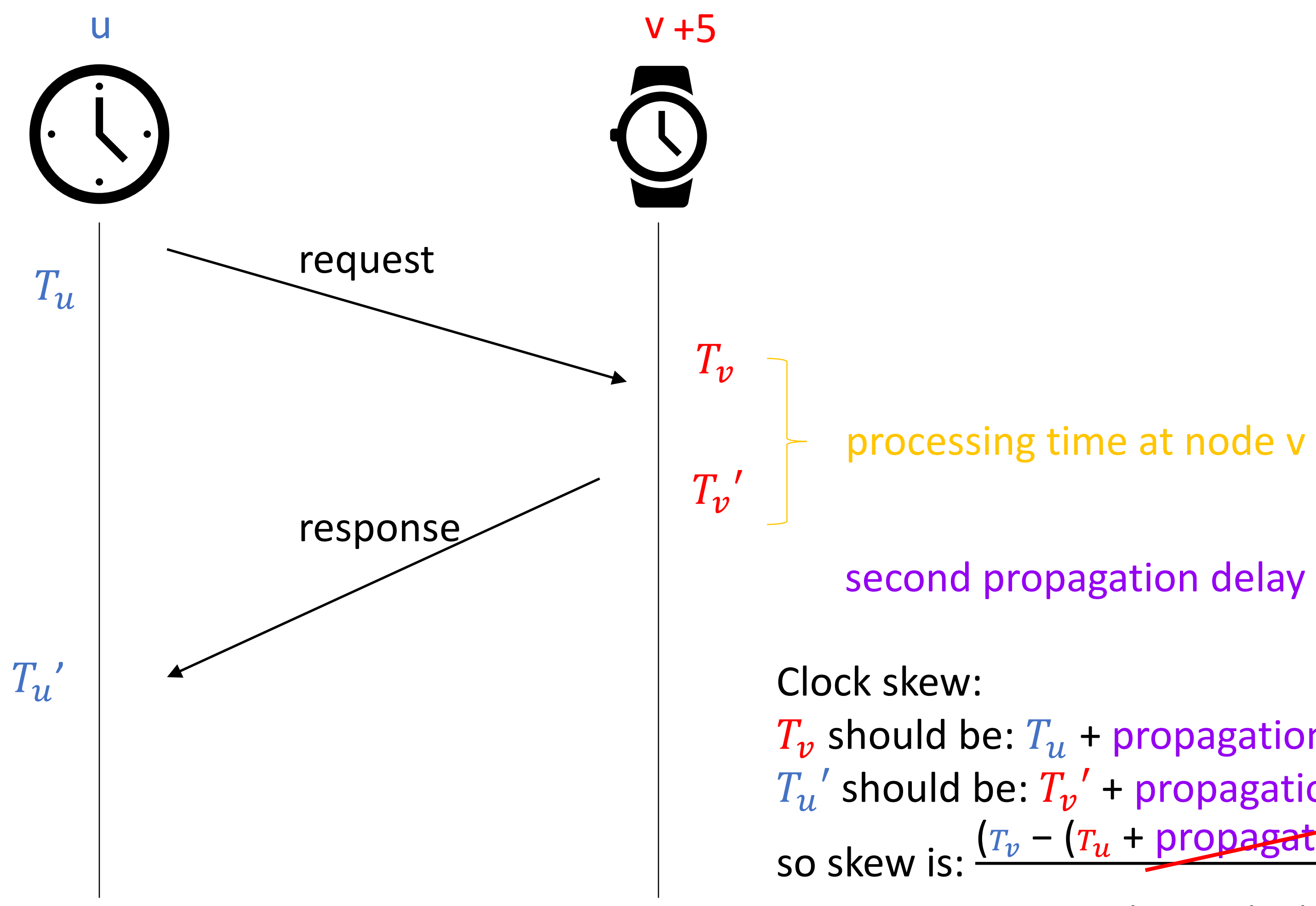
Consistent Snapshot

- Cut
  - prefix of a distributed execution
- Consistent Snapshot
  - a cut for which holds that for every operation g in that cut, if f->g, then also f is there
  - -> if all “connected” preceding operations are included
- with number of consistent snapshots, one can make conclusions about degrees of concurrency in system

Consistent Snapshot



Network Time Protocol (NTP)



Clock skew:

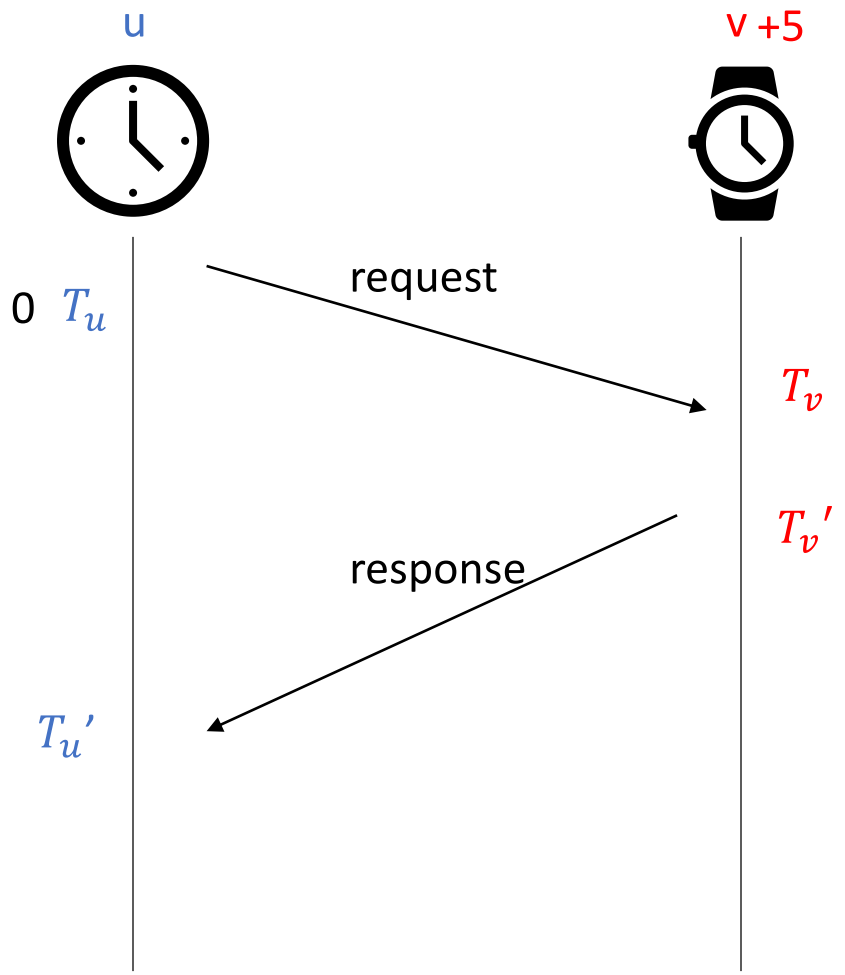
T_v should be: T_u + propagation delay

T_u' should be: T_v' + propagation delay

so skew is: $\frac{(T_v - (T_u + \cancel{\text{propagation delay}})) - (T_u' - (T_v' + \cancel{\text{propagation delay}}))}{2} =$

$\frac{(T_v - T_u) - (T_u' - T_v')}{2} = \frac{(T_v - T_u) + (T_v' - T_u')}{2}$

NTP Hands-on

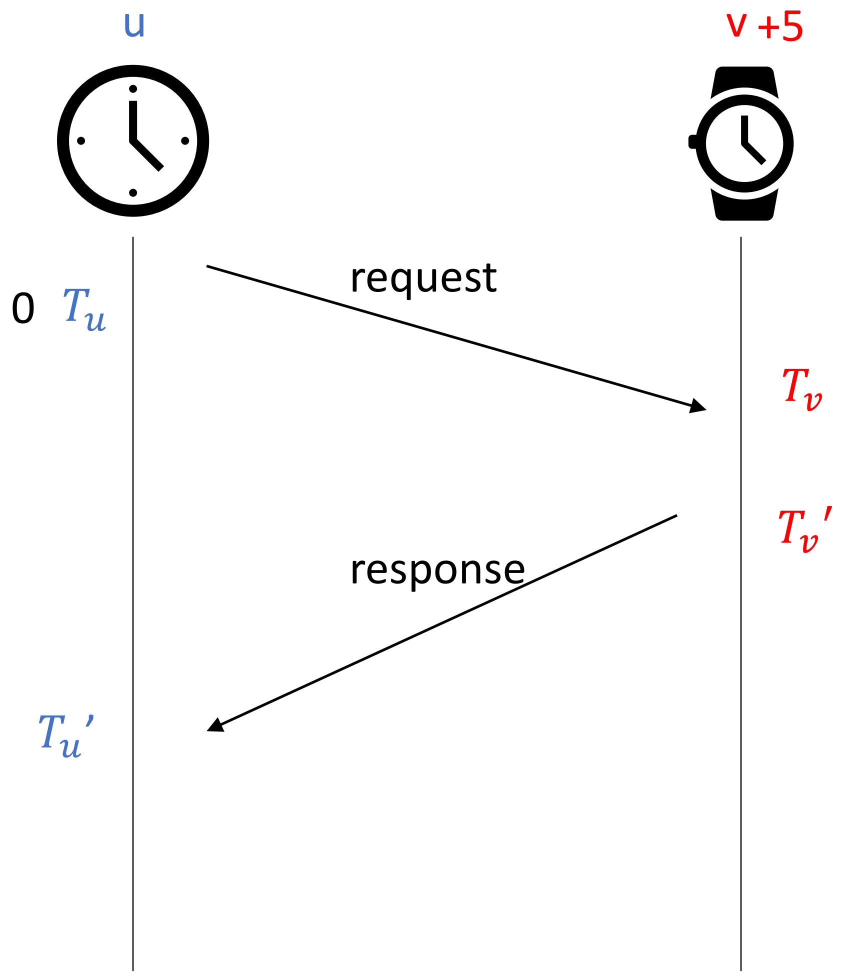


propagation delay = 10

processing time = 5

propagation delay = 10

NTP Hands-on: Solution



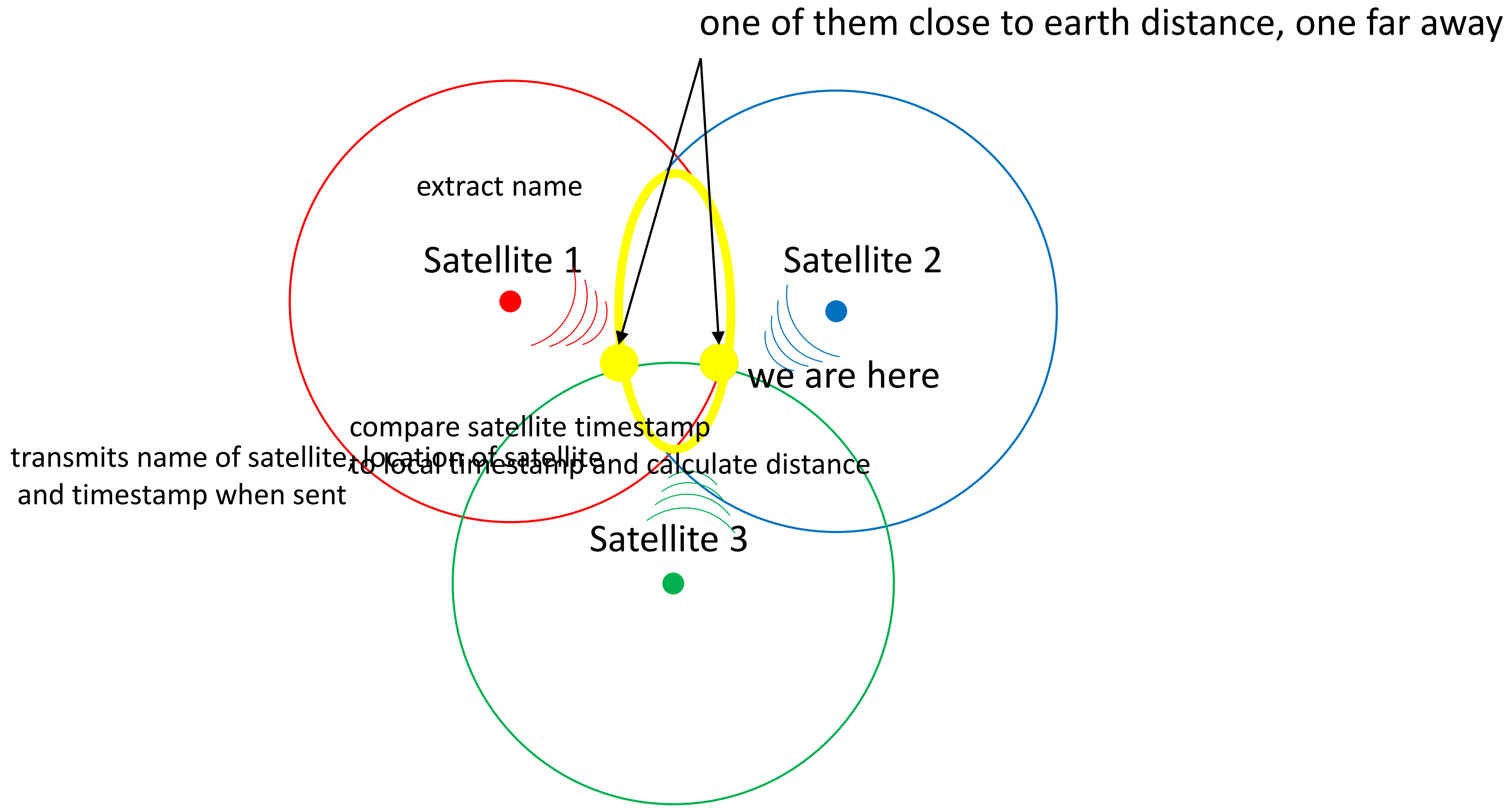
propagation delay = 10

processing time = 5

propagation delay = 10

skew = (15-0) +(20-25)/2 = 5

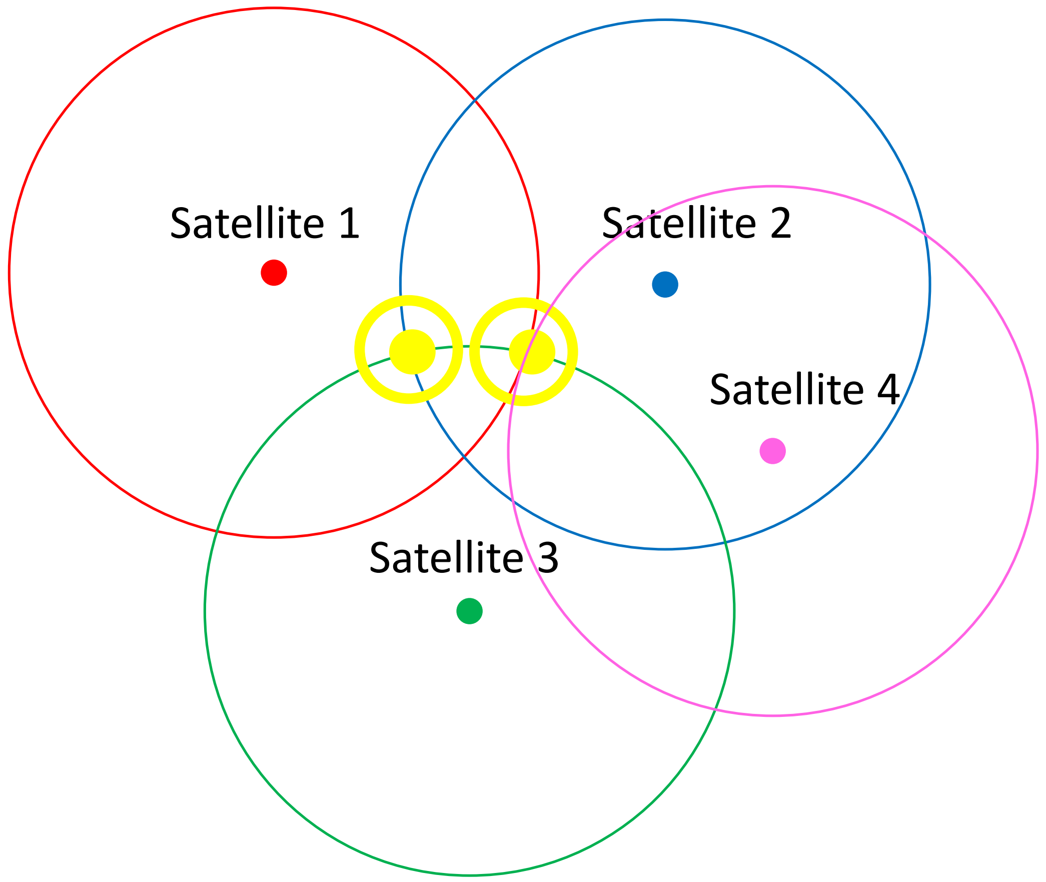
GPS – General idea


one of them close to earth distance, one far away
extract name
Satellite 1
Satellite 2
we are here
compare satellite timestamp
to local timestamp and calculate distance
transmits name of satellite, location of satellite
and timestamp when sent
Satellite 3

GPS - Problem

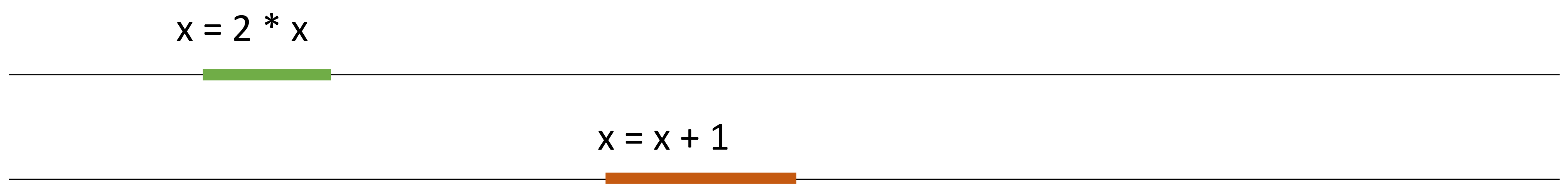
- Problem: we do not have the same time as the satellite, so calculating the distance might not be accurate
- Solution: take measurement from forth satellite!

GPS - Refined



Quiz

- Sequential Consistency implies Quiescent consistency
  - *wrong. E.g. x = 1. 5 is a valid outcome for sequential consistency, but not quiescent*



- Are there guarantees a lamport clock can achieve and a vector clock cannot?
  - *No, because the concept of a lamport clock is included in the vector clock concept*
- A high number of consistent snapshot implies a high level of concurrency
  - true

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.

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.