Logical Clocks and Distributed Snapshots

University of Auckland

28 Aug 2020

Philosophical Questions

- Does time exists? Is it linear? No loops?
- What is causality? Vs correlation ...
-

Time in distributed computing

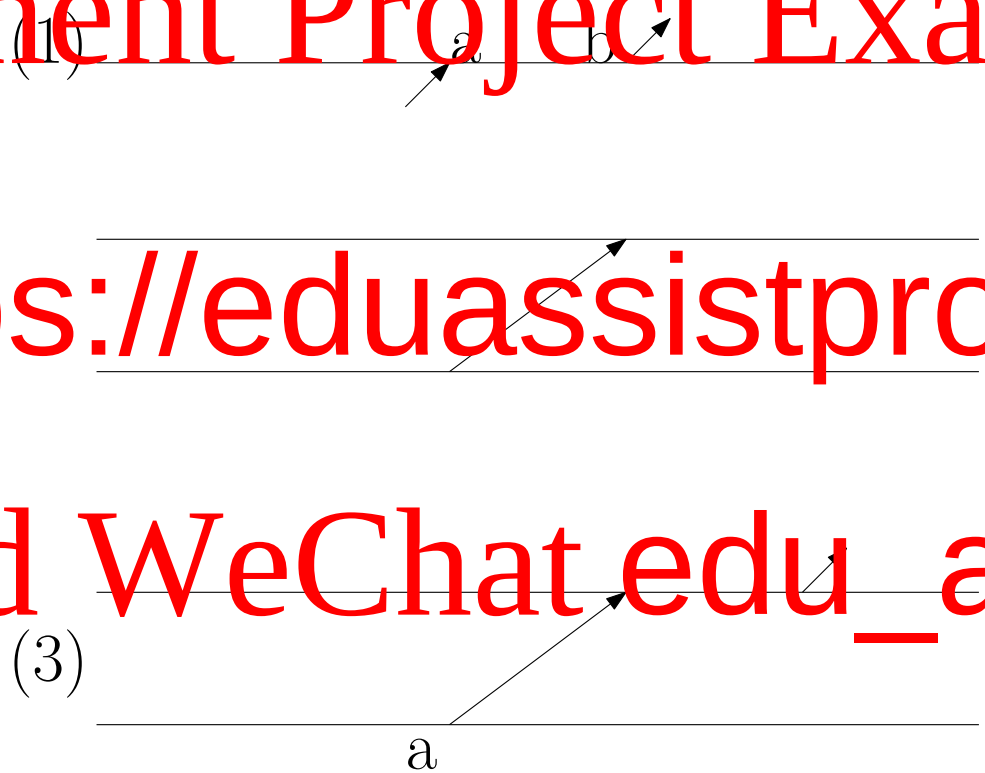
- No global clock
-
- Can we avoid these unreliable physical cloc
-
- Yes, using logical clocks, that are independ time

Happens-before = potential causation

- Event a happens-before event b, $\boxed{a \prec b}$, $\boxed{a \rightarrow b}$, iff
  1. a and b occur in the same node, and a happened before b, according to the local clock
  2. a is the sending of a message by one node, and b is the reception of that message by another node
  3. There is an event c, s.t. $a \prec c \prec b$

- This happens-before ($a \prec b$) determines a partial order (creates a dag)
- For us here, happens-before $\equiv$ potential causality

Happens-before = potential causation



Logical time and logical clocks

- If $a \not\prec \not\succ b$, then a and b are c

Logical time and logical clocks

- Logical clock is a device (algorithm) that makes these mappings in a distributed network
-
-

- Good feature: same times, if "essentially" same execution
- Ideal / exact match feature: $a \prec$ all clocks!

  - Factual: $C(a) < C(b) \Rightarrow a \prec b$

- If the clock maps to a totally ordered set, then it is not "ideal"

Logical time in A#1?

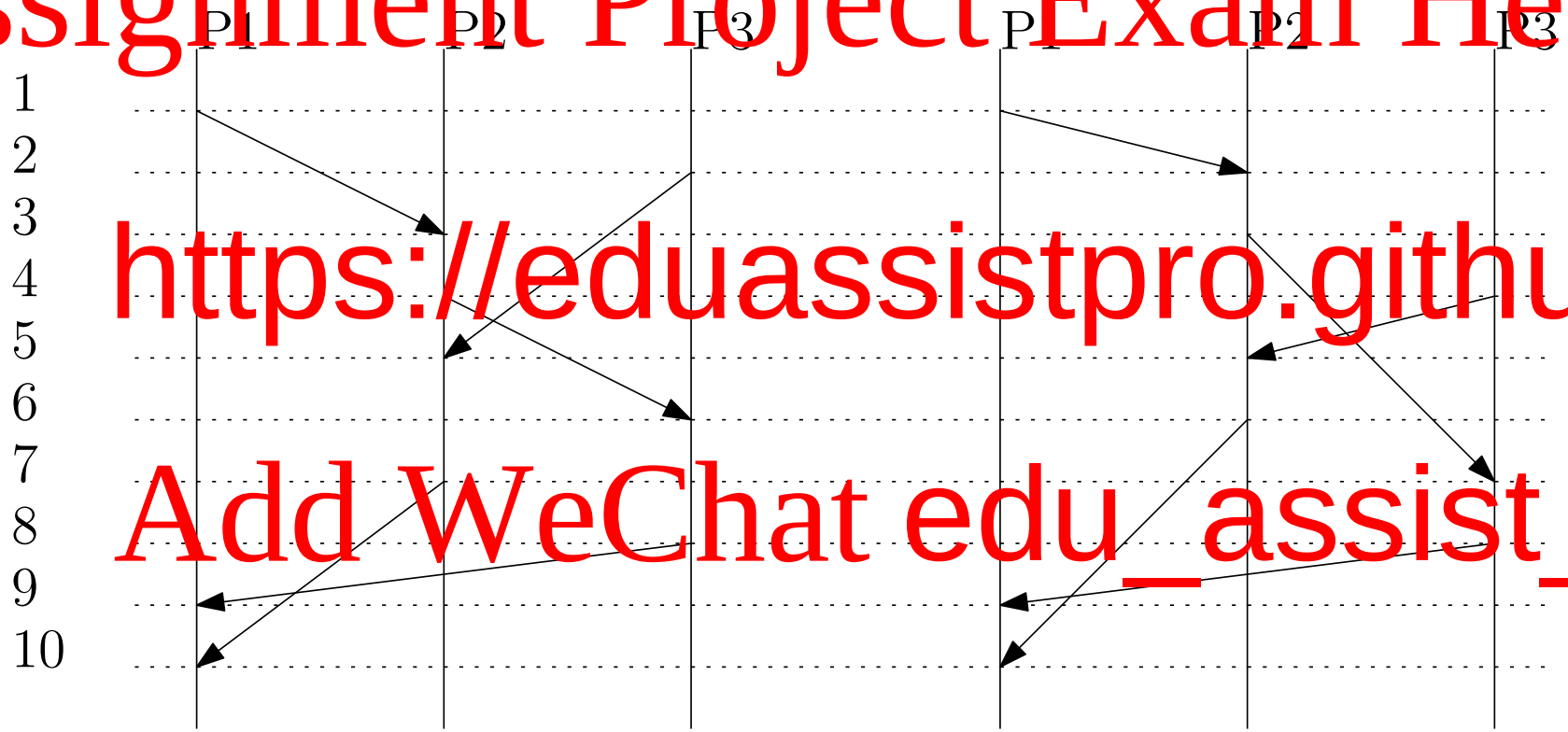
- According to this definition, what is the logical time in A#1?

- Is this "ideal"? Is this total?

- Is this determined by the nodes? Or by Arcs
- Briefly: total logical time, but Arcs is NOT a true logical clock

Sample Scenario – Total orders

Two compatible totals orders for essentially the same execution



A good logical clock should generate same logical timestamps

Lamport Logical Clock – Part I

- Each processing node has its own logical clock, initialised by

  clock = 0

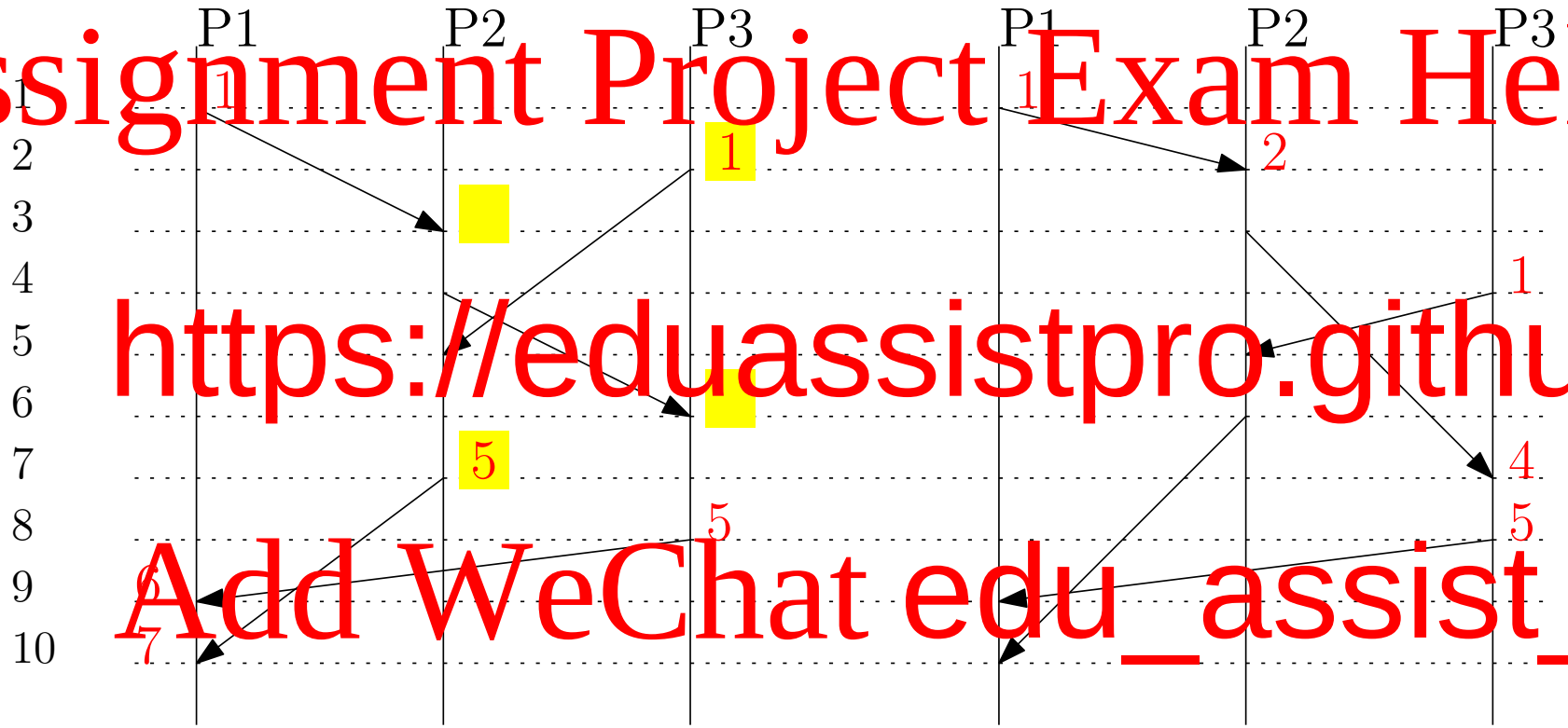
- Each sent messages carries the sender's clo

- After receiving a message, the receiving no

  clock = max (clock, received-clock) + 1

Lamport Logical Clock – Part I

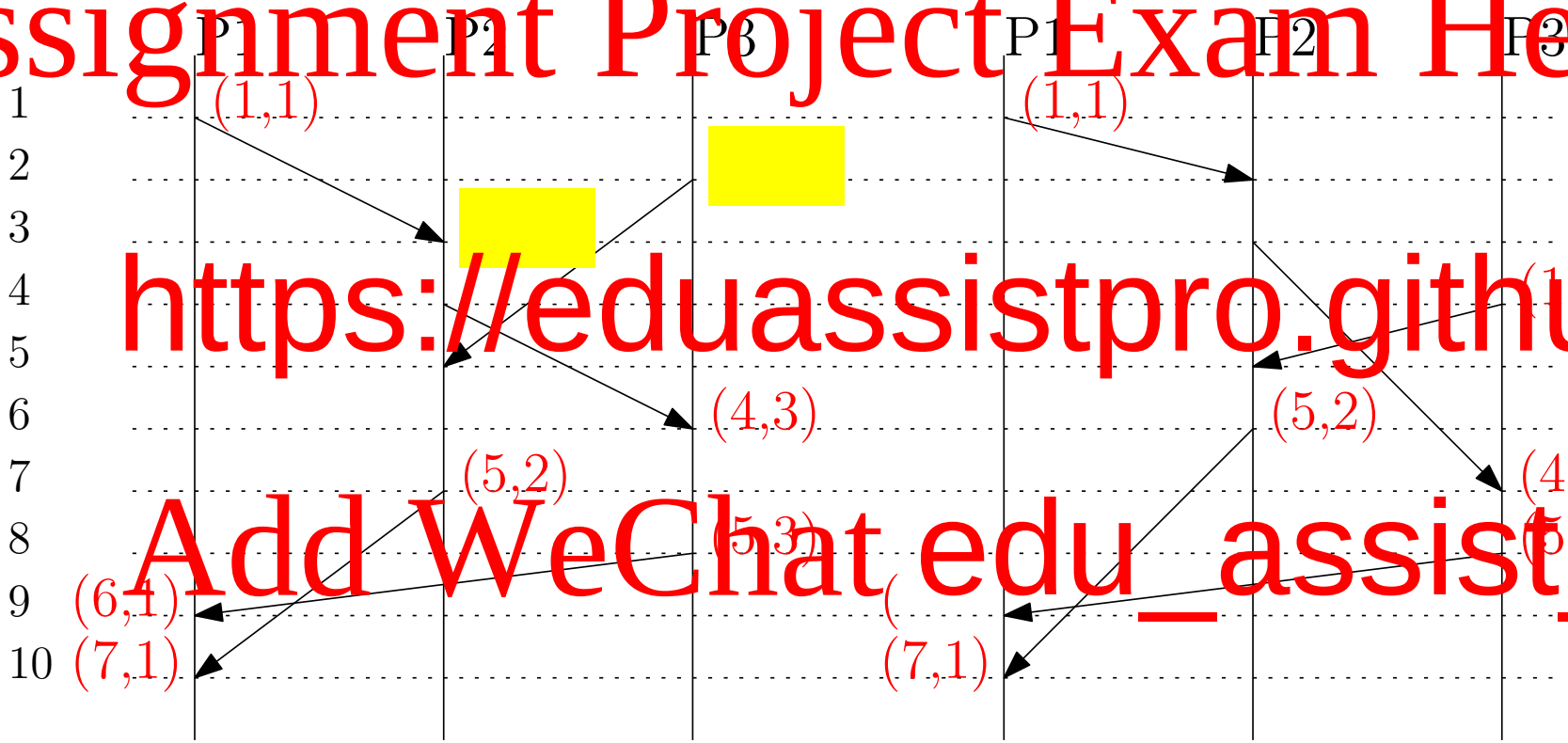
Same logical timestamps for essentially the same execution



- NOT "ideal order": $P3:4 < P2:5$, but $P3:4 \not< P2:5$
- Total order, b/c timestamps have simple numerical values
- More: timestamps are NOT unique!

Lamport Logical Clock – Part II

Ensure uniqueness by adding process IDs – lexicographic order



- Unique, but otherwise same problems, NOT "ideal"
- NOT "ideal order": $P3$: (1,3) $<$ $P2$: (2,2), but
  $P3$: (1,3) $\not<$ $P2$: (2,2) – in fact, these are concurrent

Lamport Logical Clock – Exercise (WanFok)

- Consider the following sequences of events at processes p_1, p_2, p_3:

```
1  p1 : a s1 r3 b
2  p2 : c r2 s3
3
```

-
- for $i = 1, 2, 3$.
- Provide all events with Lamport's clock val
- Answer:

```
1  p1 : 1 2 8 9
2  p2 : 1 6 7
3  p3 : 3 4 5 6
```

Vector Logical Clock – Exact Match

- Each process keeps a vector (tuple) containing all known local logical times, e.g.

1 $V(P_2) = (v_1, v_2, v_3)$

-

1 $(v_1, v_2, v_3) \leq (v'_1, v'_2, v'_3) \Leftrightarrow v_1 \leq v'_1, v$

- The local logical time is incremented befor event, e.g.

1 $V(P_2) : (v_1, v_2, v_3) \Rightarrow (v_1, v_2 + 1, v_3)$

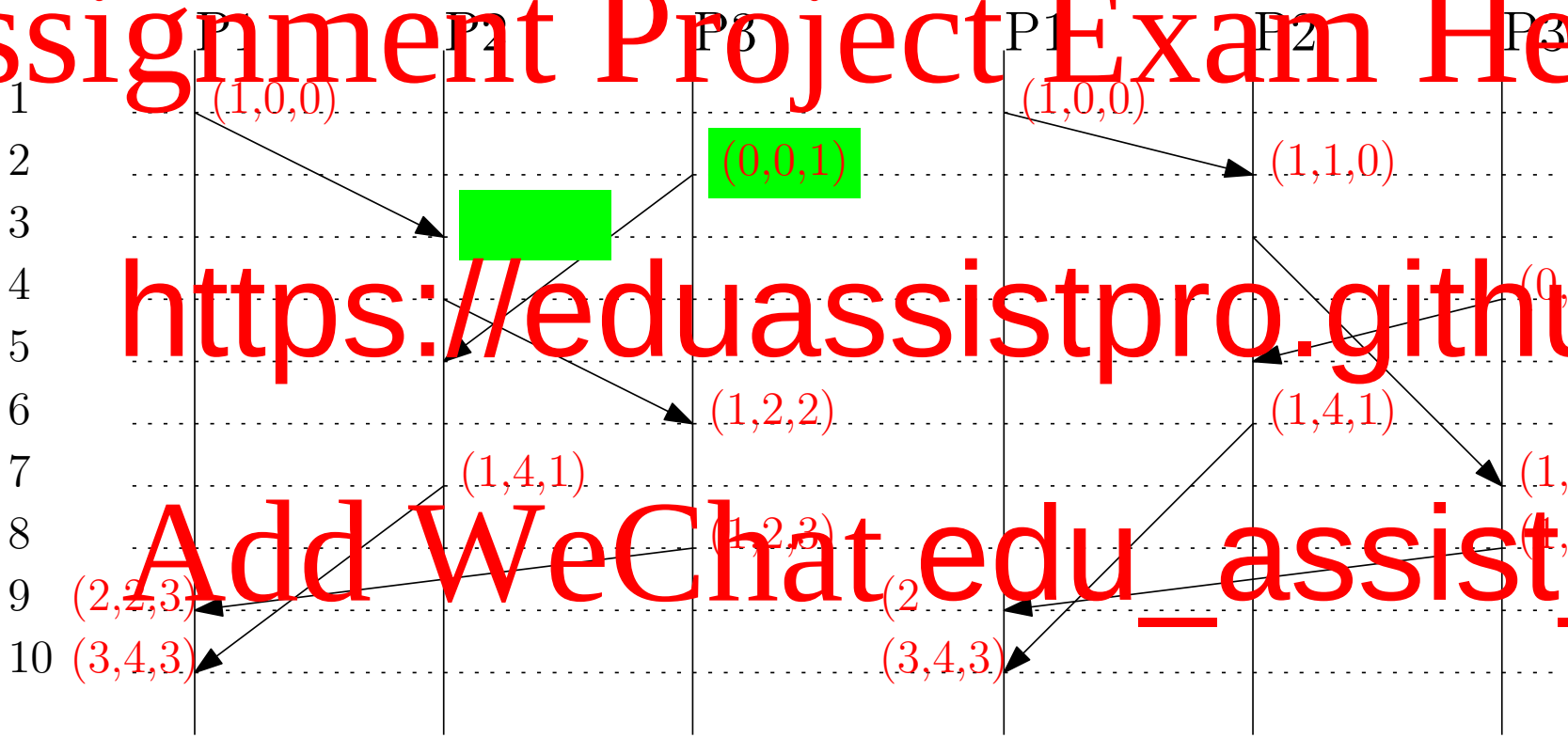
- ...

Vector Logical Clock

- ...
- ...

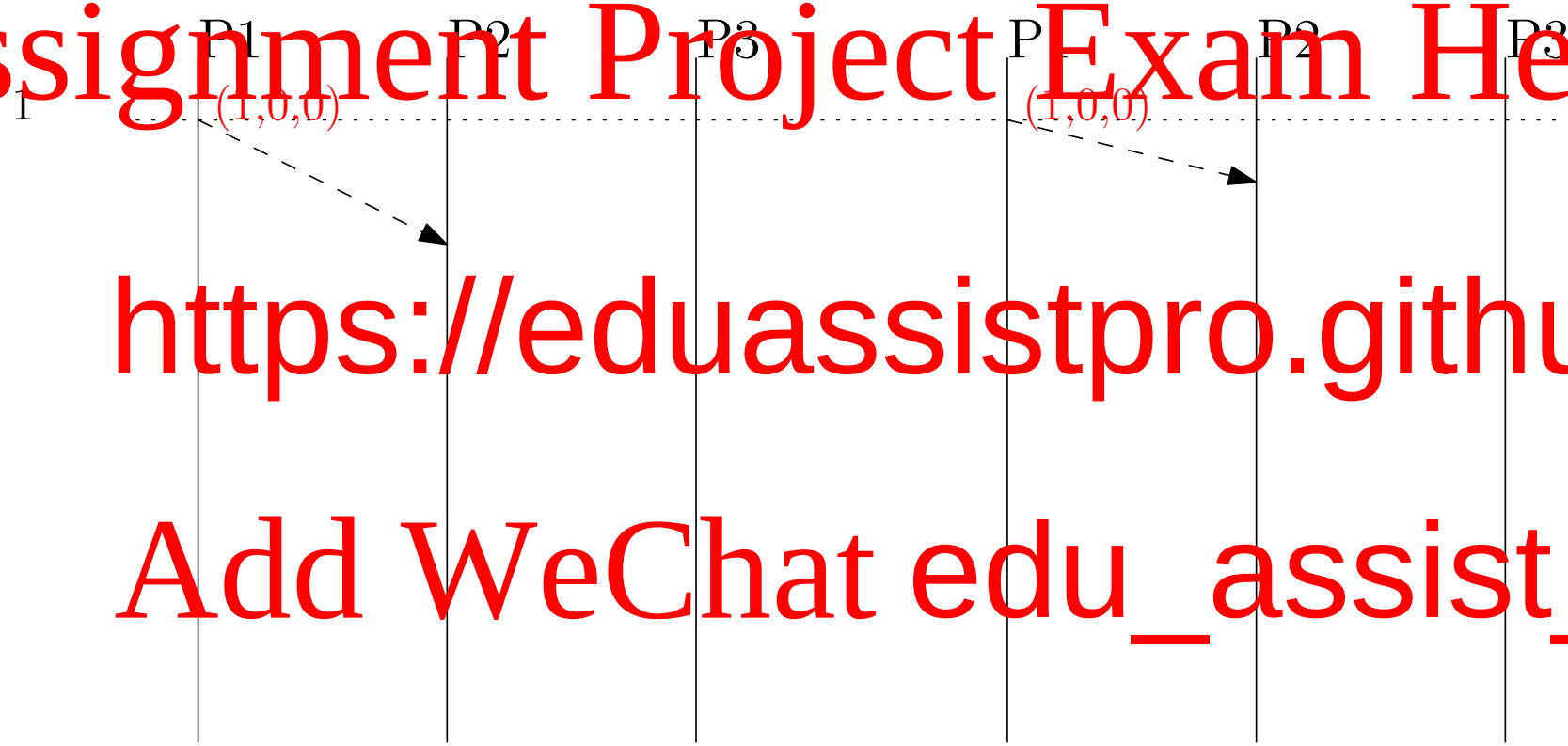
1 $V(P_2) : (v_1, v_2, v_3), (r_1, r_2, r_3)$
2 $\Rightarrow (\max(v_1, r_1), \max(v_2, r_2) + 1$

Vector Logical Clock – Complete example

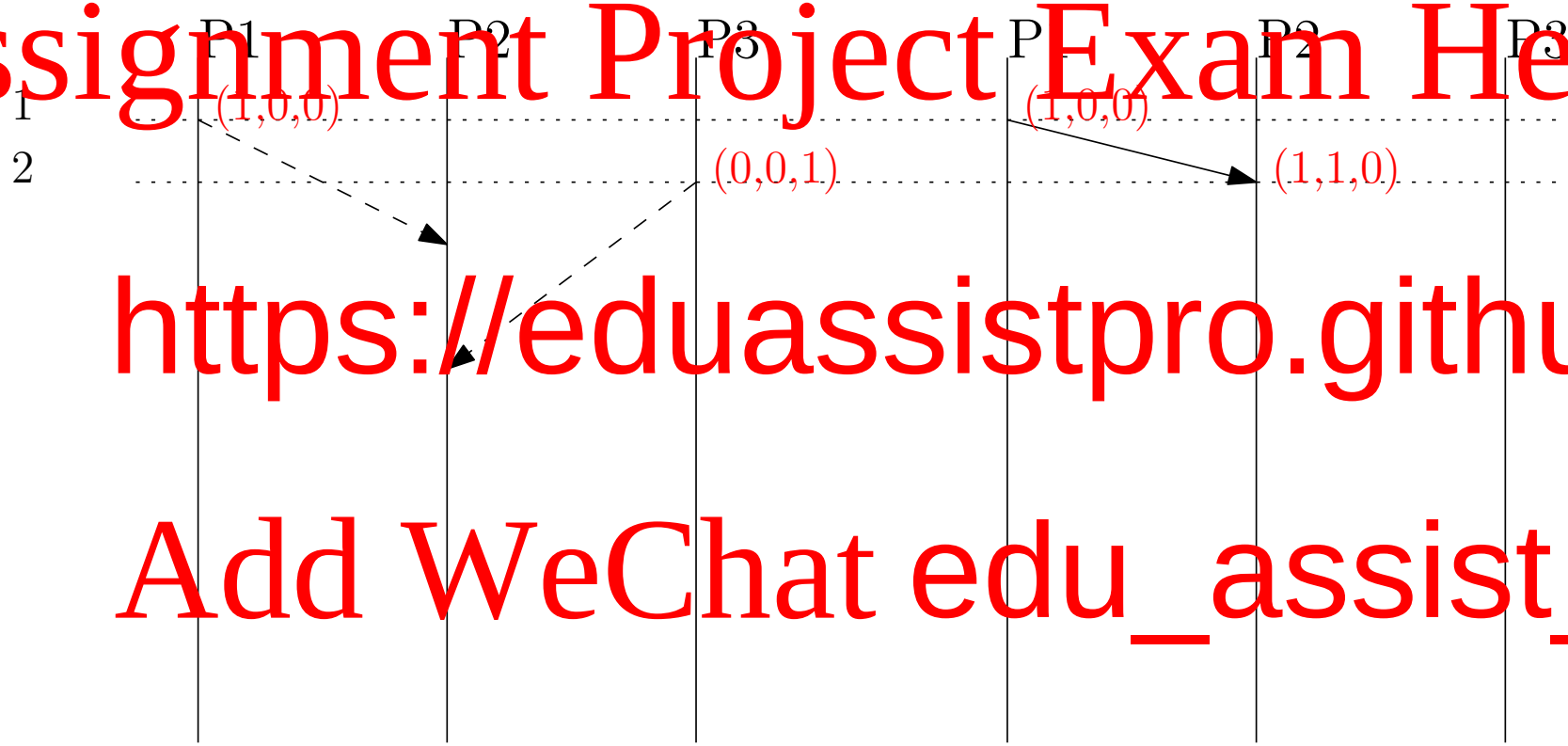


$(1, 1, 0)$ and $(0, 0, 1)$ denote concurrent events and are incomparable!

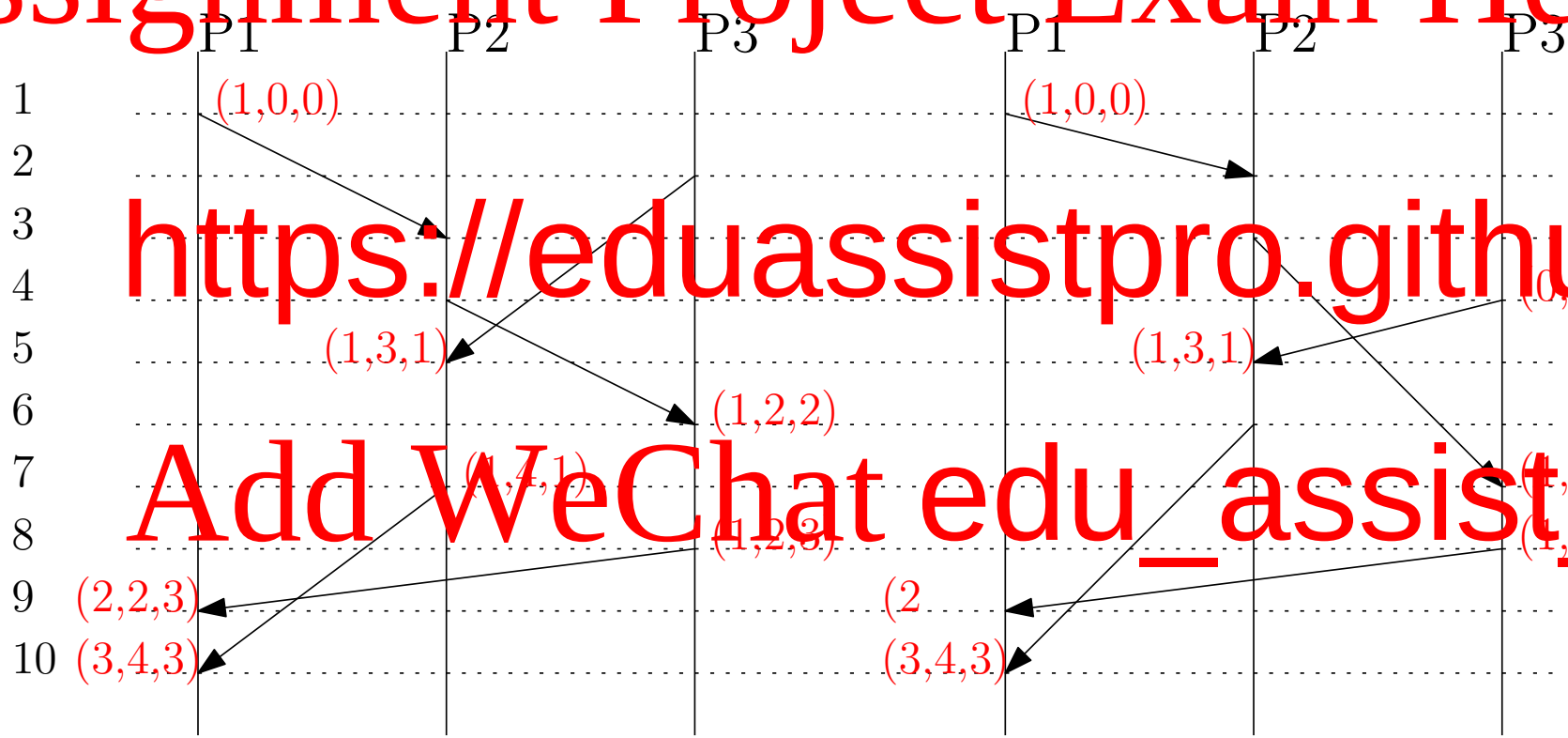
Vector Logical Clock – Steps



Vector Logical Clock – Steps

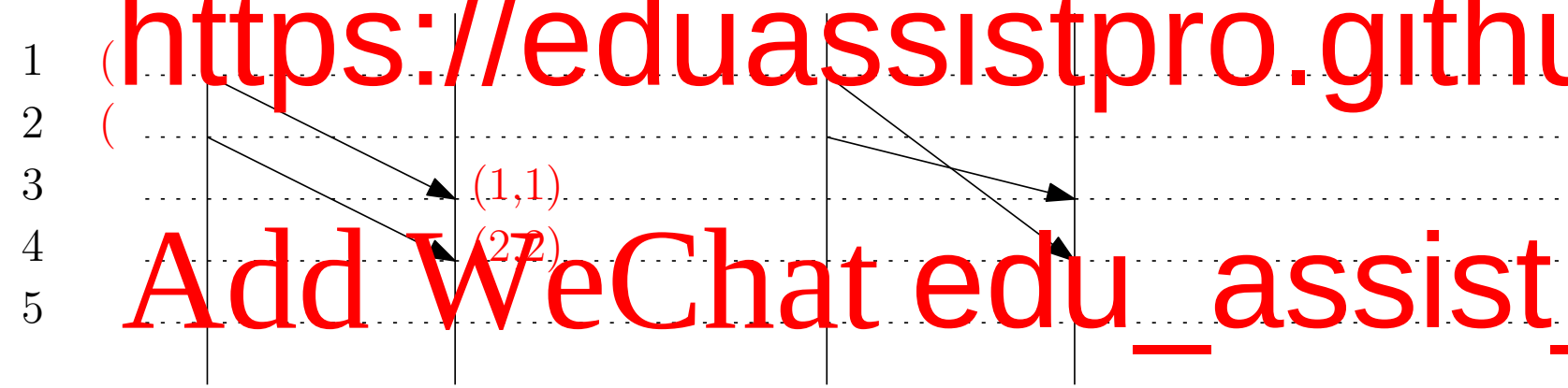


Vector Logical Clock – Steps



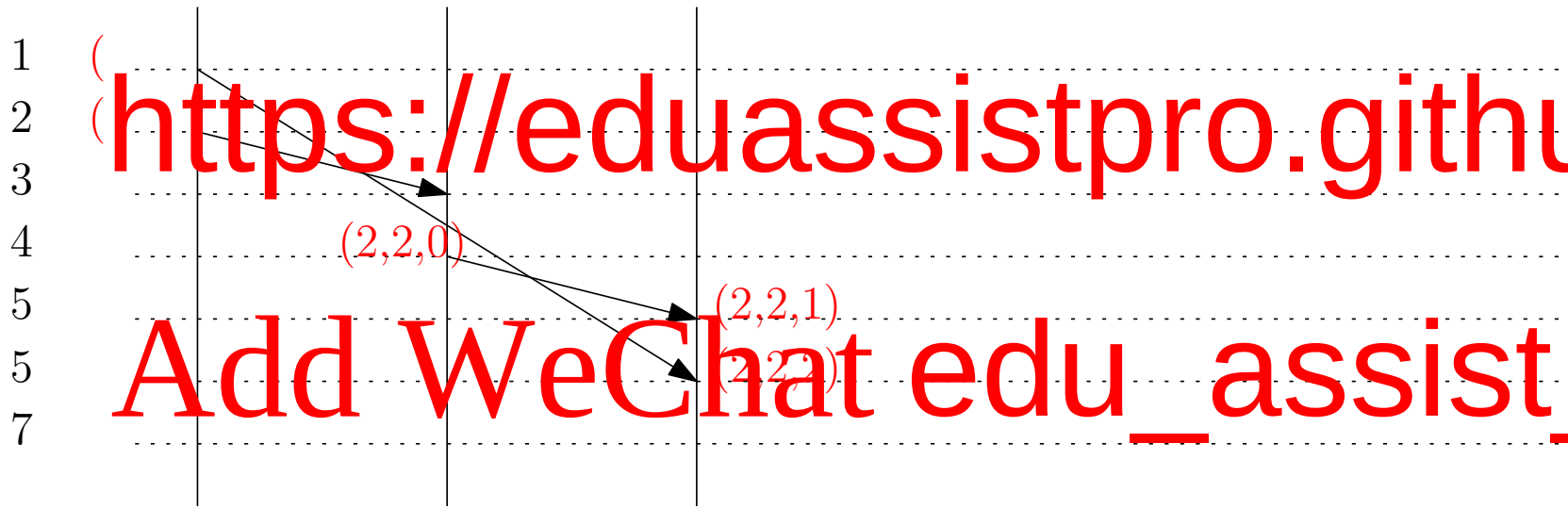
Vector Logical Clock – FIFO example / counter-example

Vector logical clocks can also detect simple FIFO violations, but not Lamport



Vector Logical Clock – Causal ordering counter-example

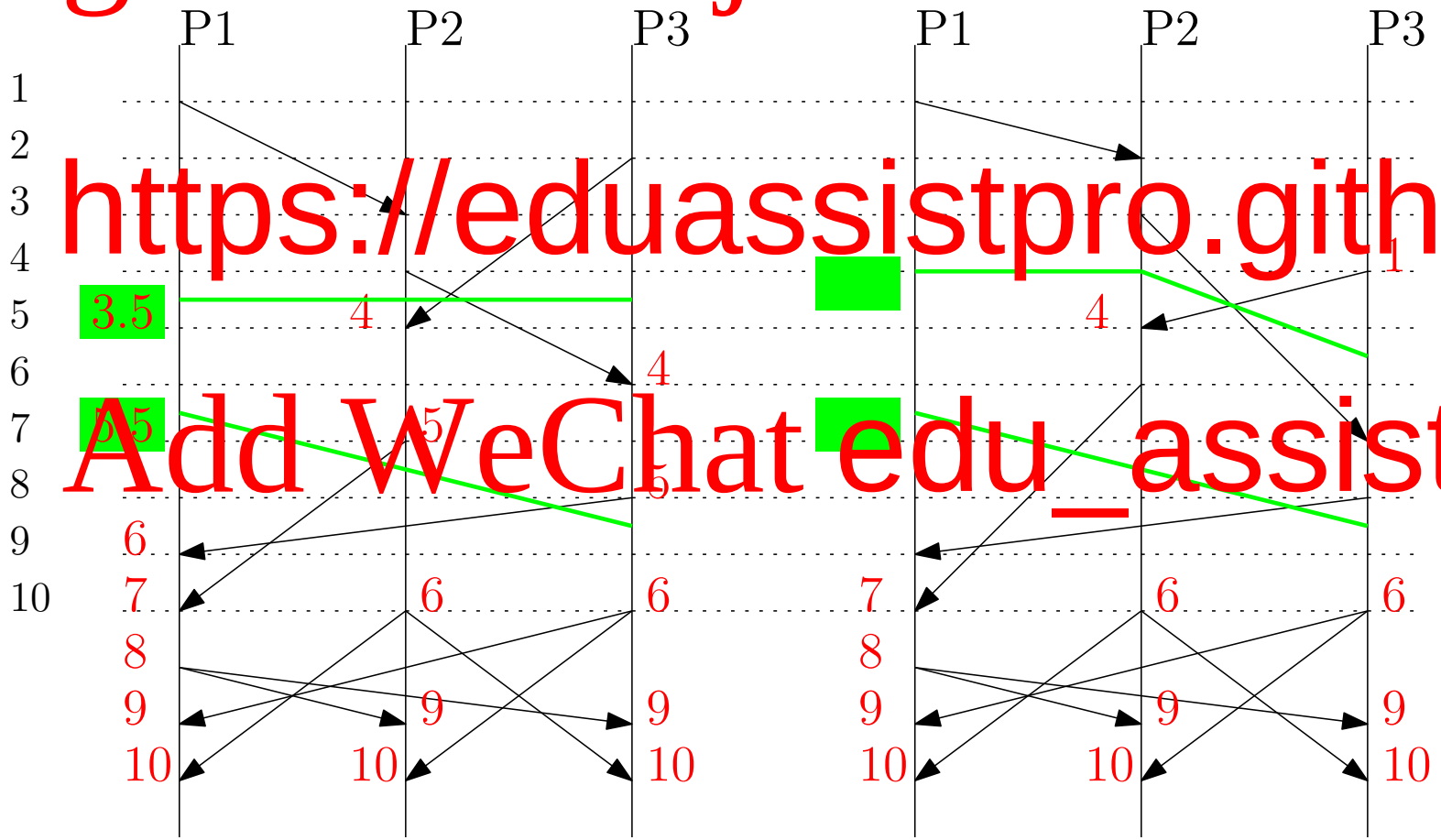
Causal ordering $\simeq$ FIFO++ ☺



How many objects are in these systems?

Count Money Example – How much money in each node?

- Just after logical time 3 (3.5) or 5 (5.5)?
- Assume: initial deposits 100$ each, arrow transfers 10$ each.



Count Money – Algorithm

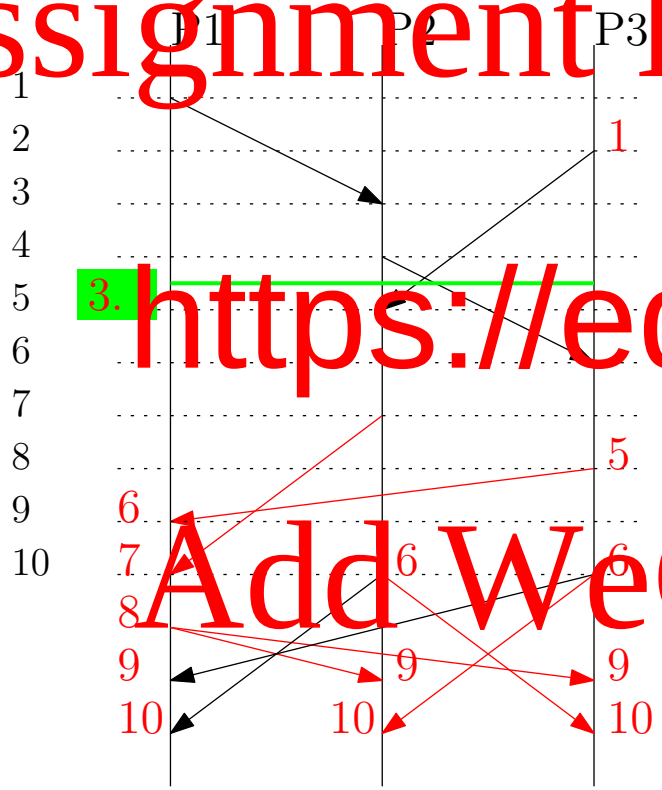
- Given Lamport time t, at each node "bank account", sum
  - The amount already there at time t
- Plus the money already sent to you, but still in transit

message sent after time t – aka marker message

  - Don't bother about money you send out af

- Assumption #1: on each channel, the mess
  potentially infinite (it never stops, both ways)
- Assumption #2: FIFO flows ?
- Can we work around the FIFO restriction?

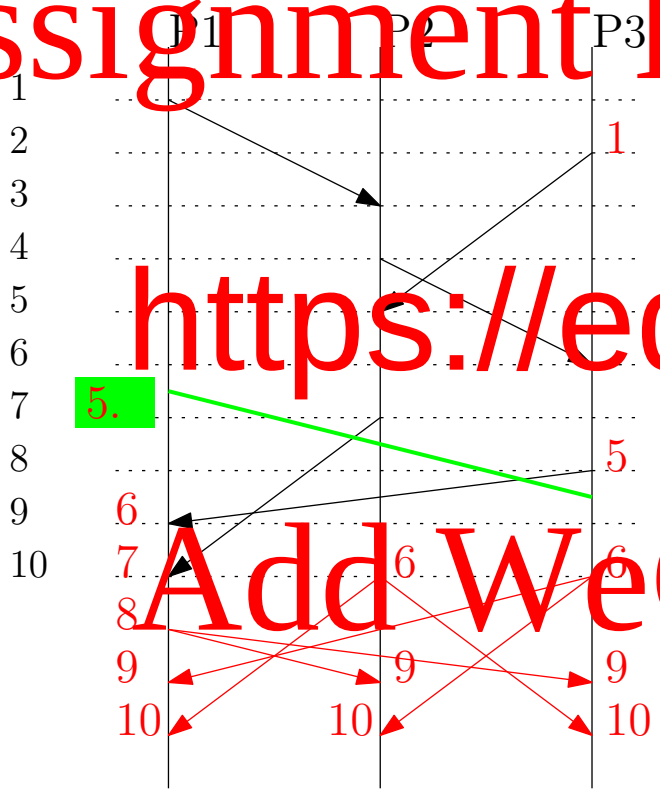
Count Money Example – Amounts at Lamport times 3.5



g

	P			T
0	1			100
3.5 ?	9			280 ?
3.5 ✓	90 @7	110 @10	100 @10	300 ✓

Count Money Example – Amounts at Lamport times 5.5



g

				T
0				300
5.5 ?				280 ?
5.5 ✓	110 @10	100 @10	90 @10	300 ✓

Count Money Example – FIFO Discussion

for each channel! (TCP?)

- FIFO
  problems!
- Ensure t
  sent bef

	P1	P2
0	100	100
1.5 ?	90	90
1.5 ???	100 @5	90 @4

	P1	P2
0	100	100
1.5 ?	90	90
1.5 ???	100 @5	90 @4
1.5 ✓	100 @5	100 @5