Logical Clocks

CS 451 Lecture

Prof. R.K. Joshi
Dept of CSE
IIT Bombay

Synchronizing between machines

- In a distributed system, machines are spatially separated
- They communicate by exchanging messages
- Delays are not negligible
- It's sometimes impossible to say which of the 2 events occurred first? (*happened before* relation: only partial ordering)

Lamport's Contribution

- Partial ordering defined by happened before relation
- Extend it to provide a consistent total ordering of all events in a distributed system

happened before Relation

- When do you say event a happened before event b?

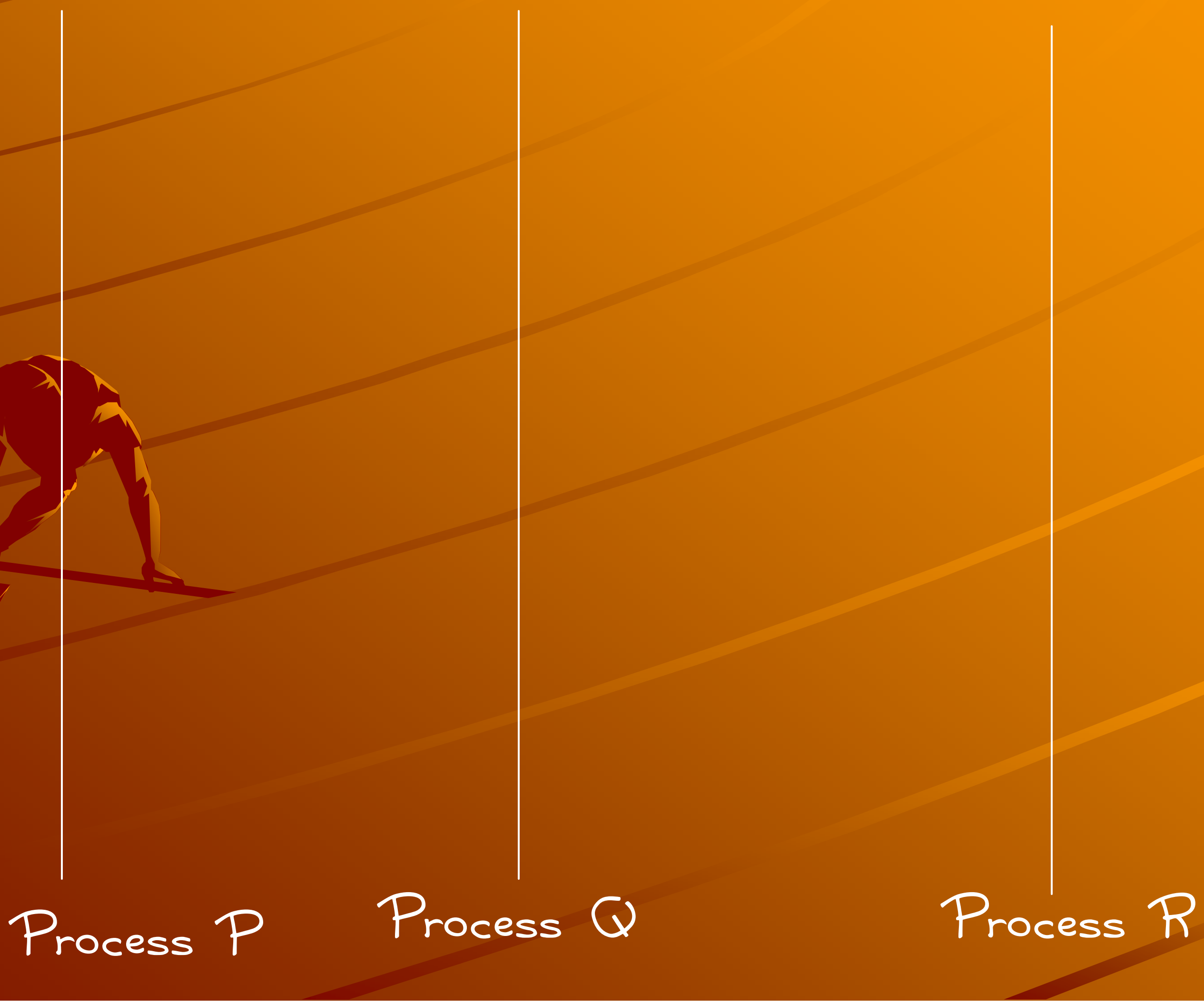
When a's real time stamp is earlier than that of b

problems with this scheme is that all events are not observable from a given system and all real clocks are not synchronized

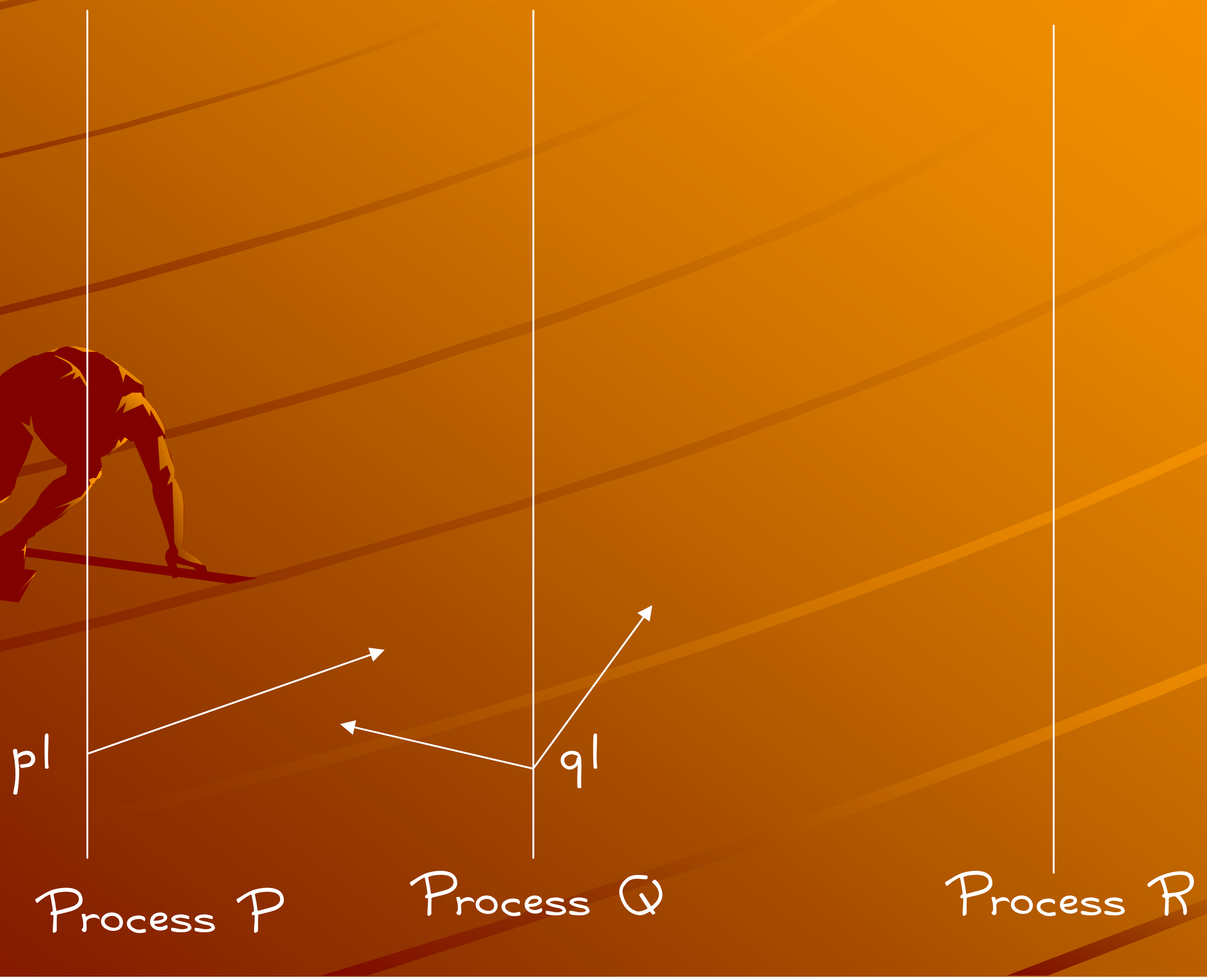
happened before Relation →

- If a, b are events in the same process, and a comes before b, then a→b
- If a is the sending of a message by one process and b is receiving of message by another, a→b
- If a → b and b→ c, then a → c
- If a -/→ b and b -/→ then a and b are said to be concurrent

Example Space-Time Diagram



Example



p1

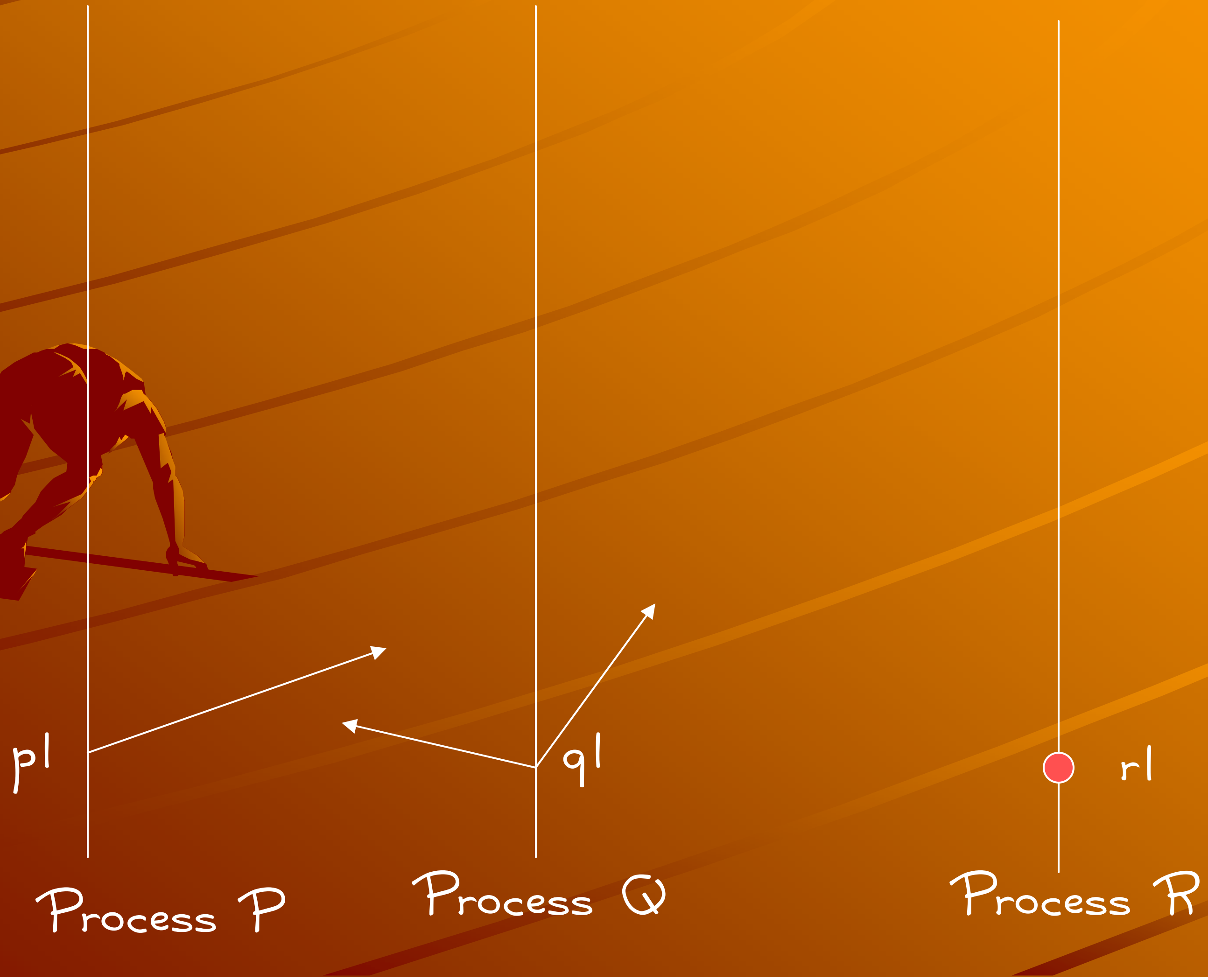
q1

Process P

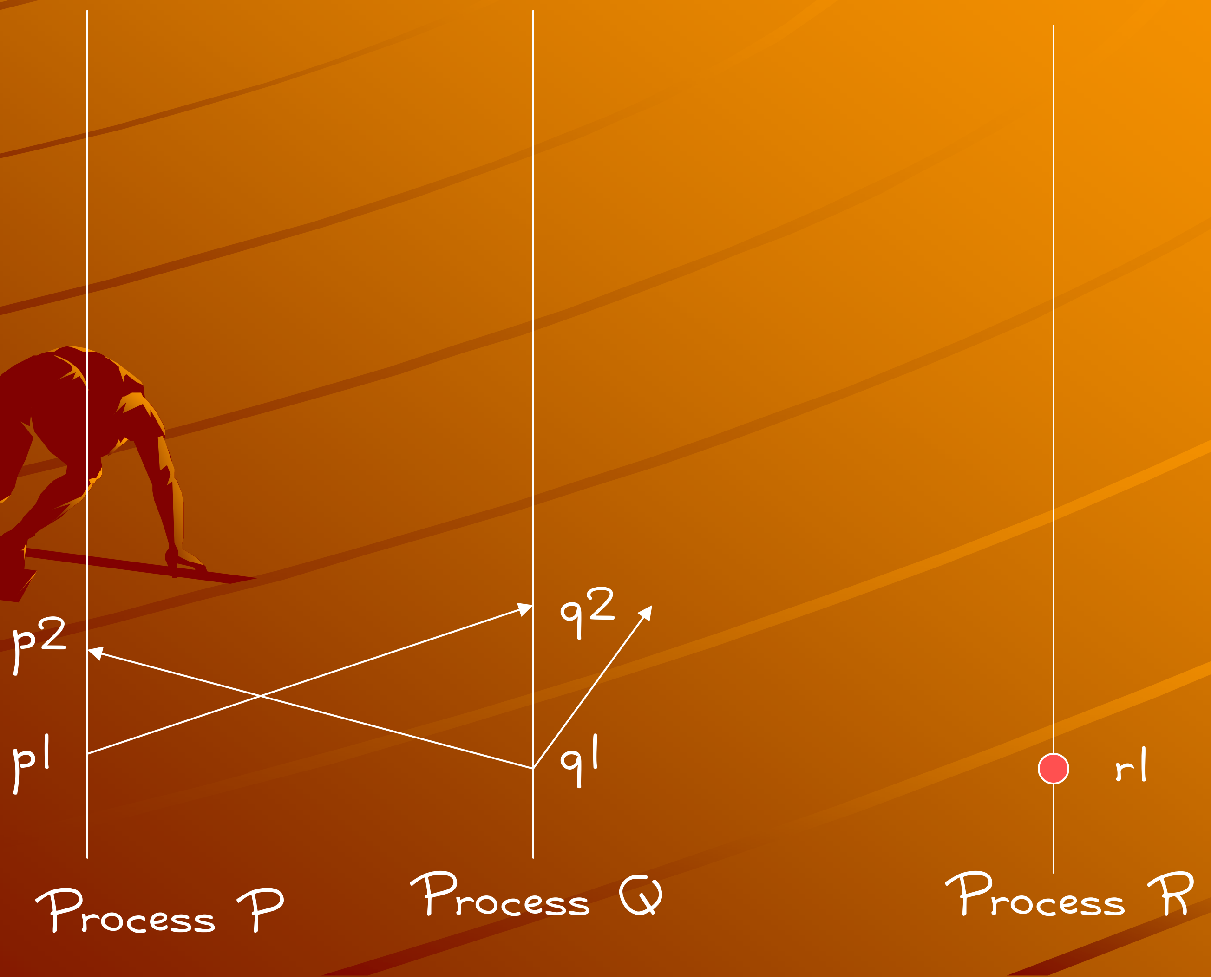
Process Q

Process R

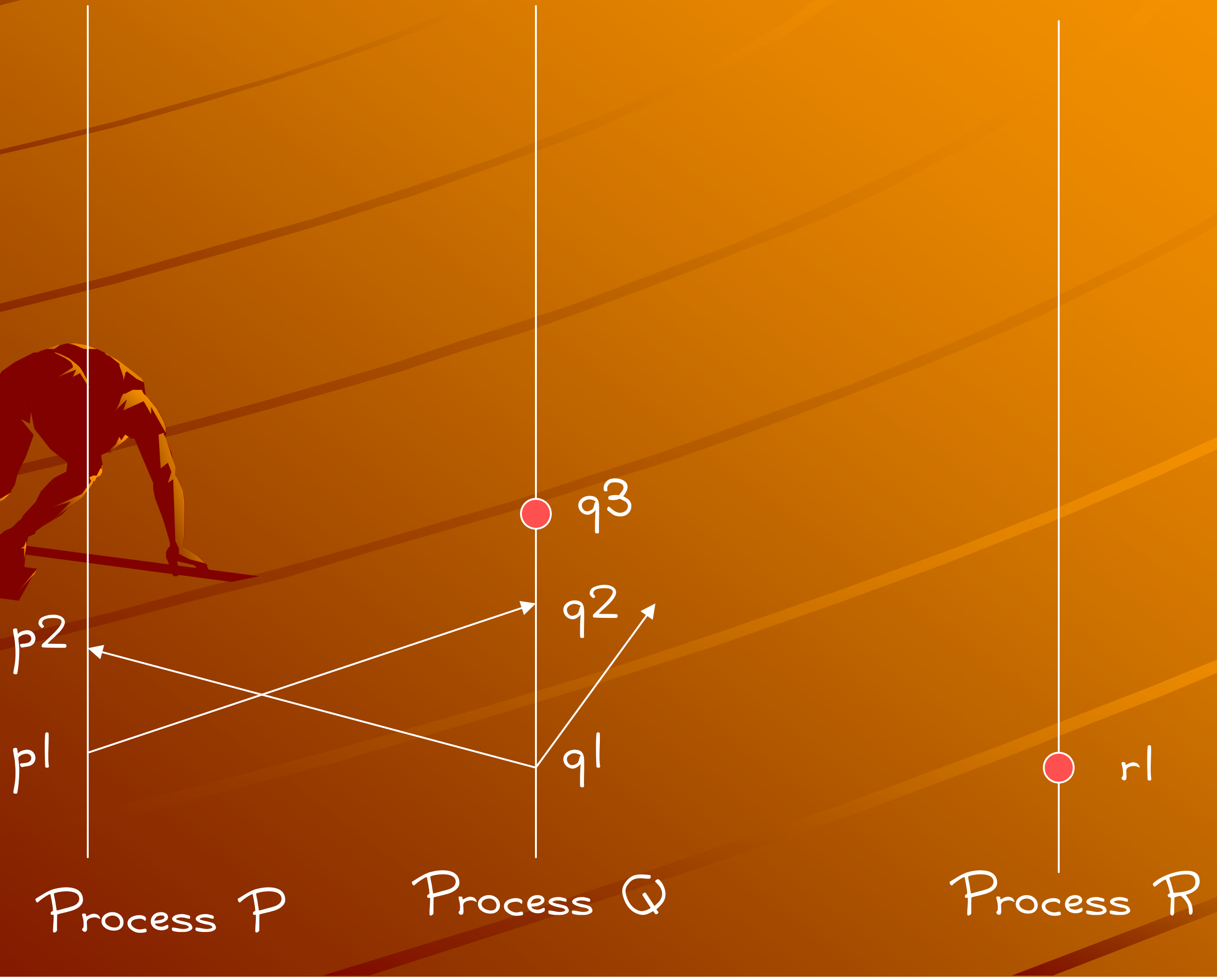
Example



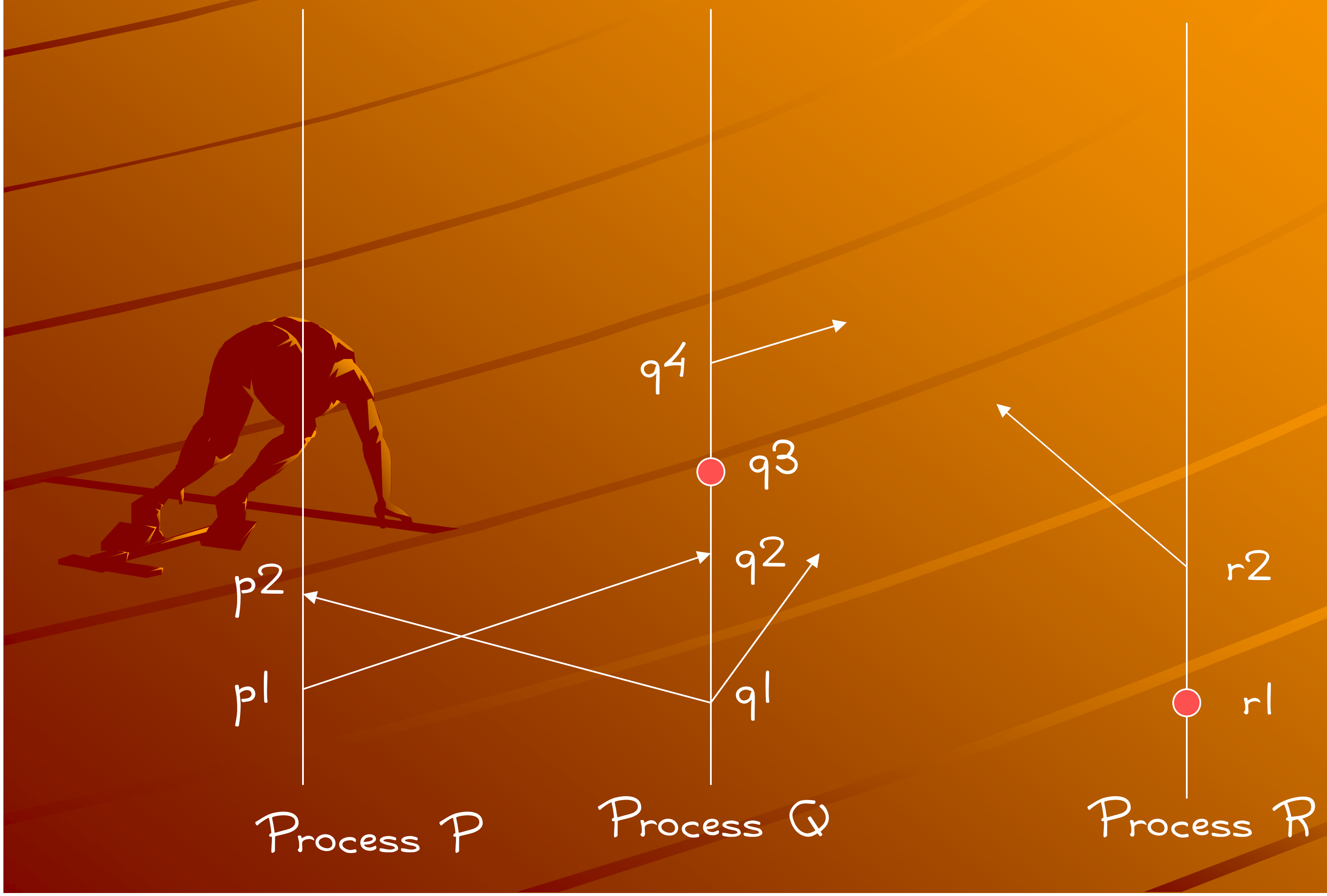
Example



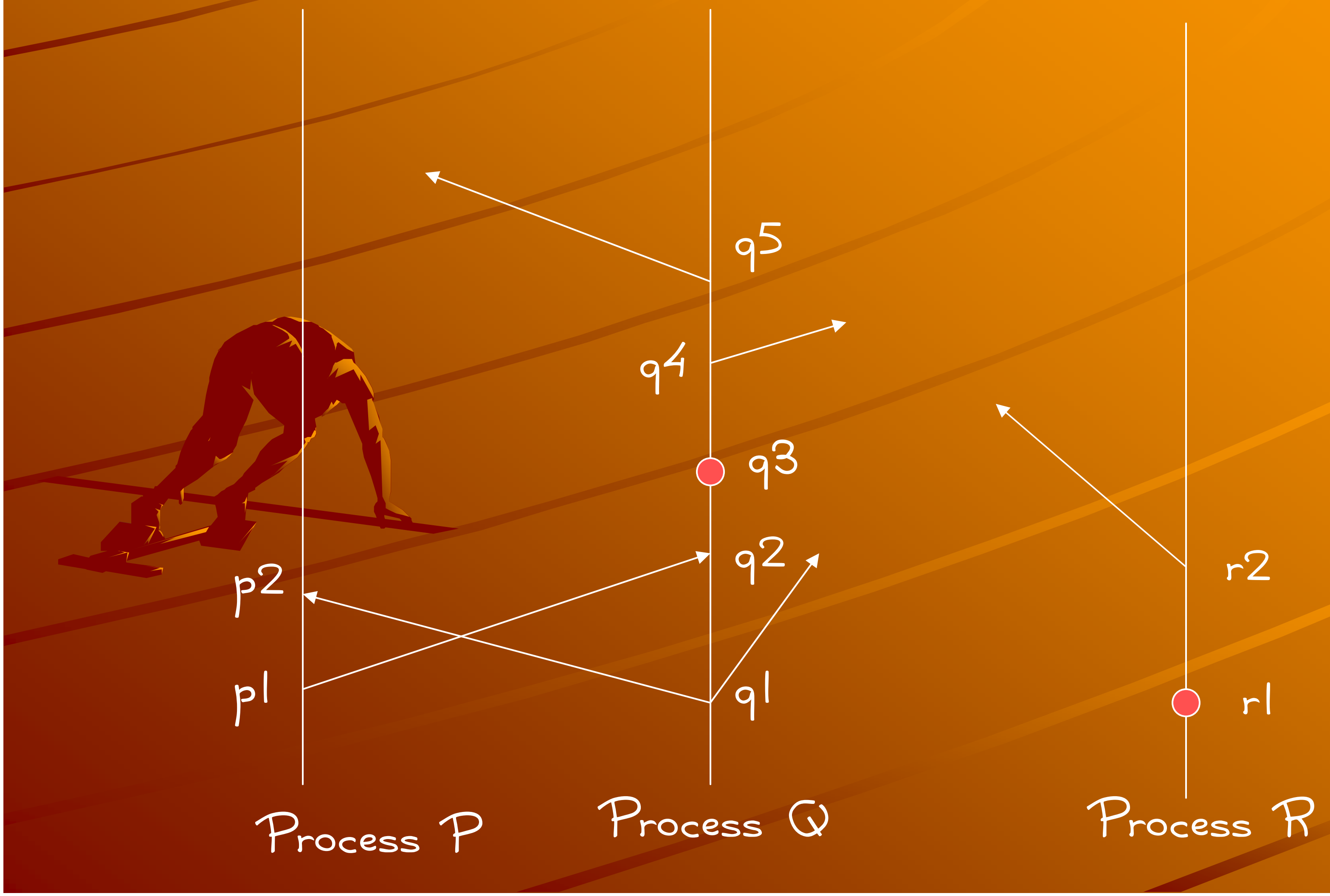
Example



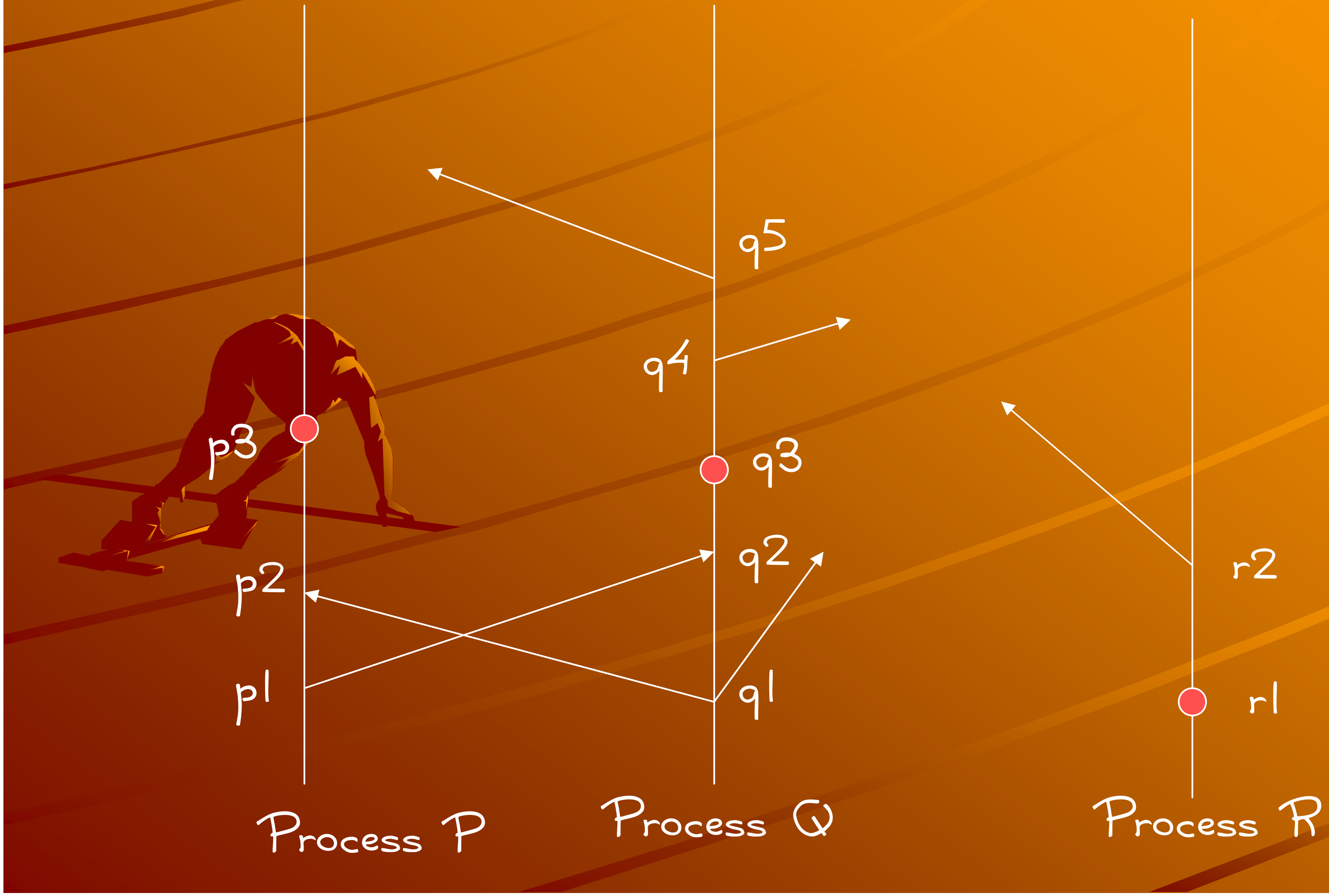
Example



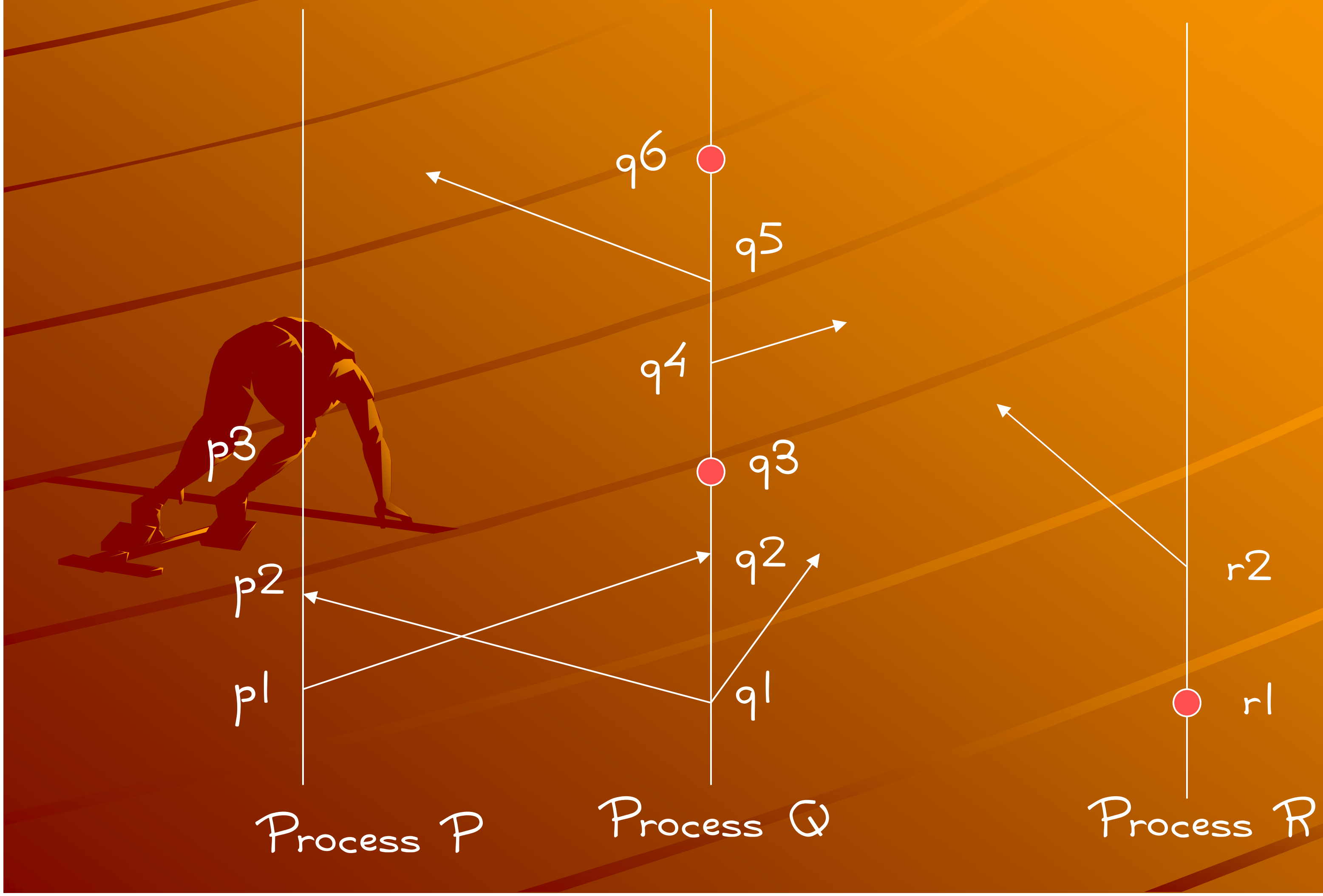
Example



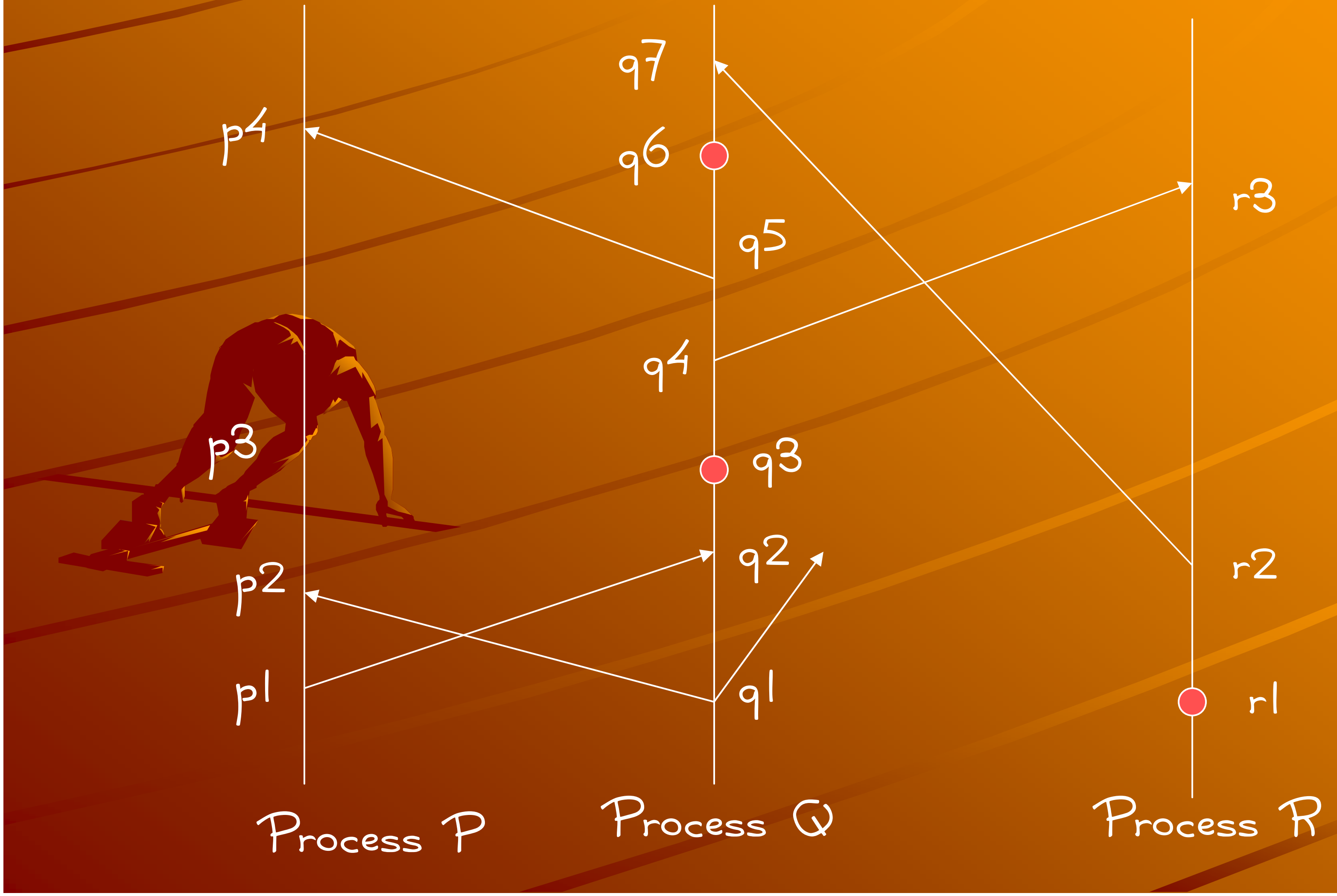
Example



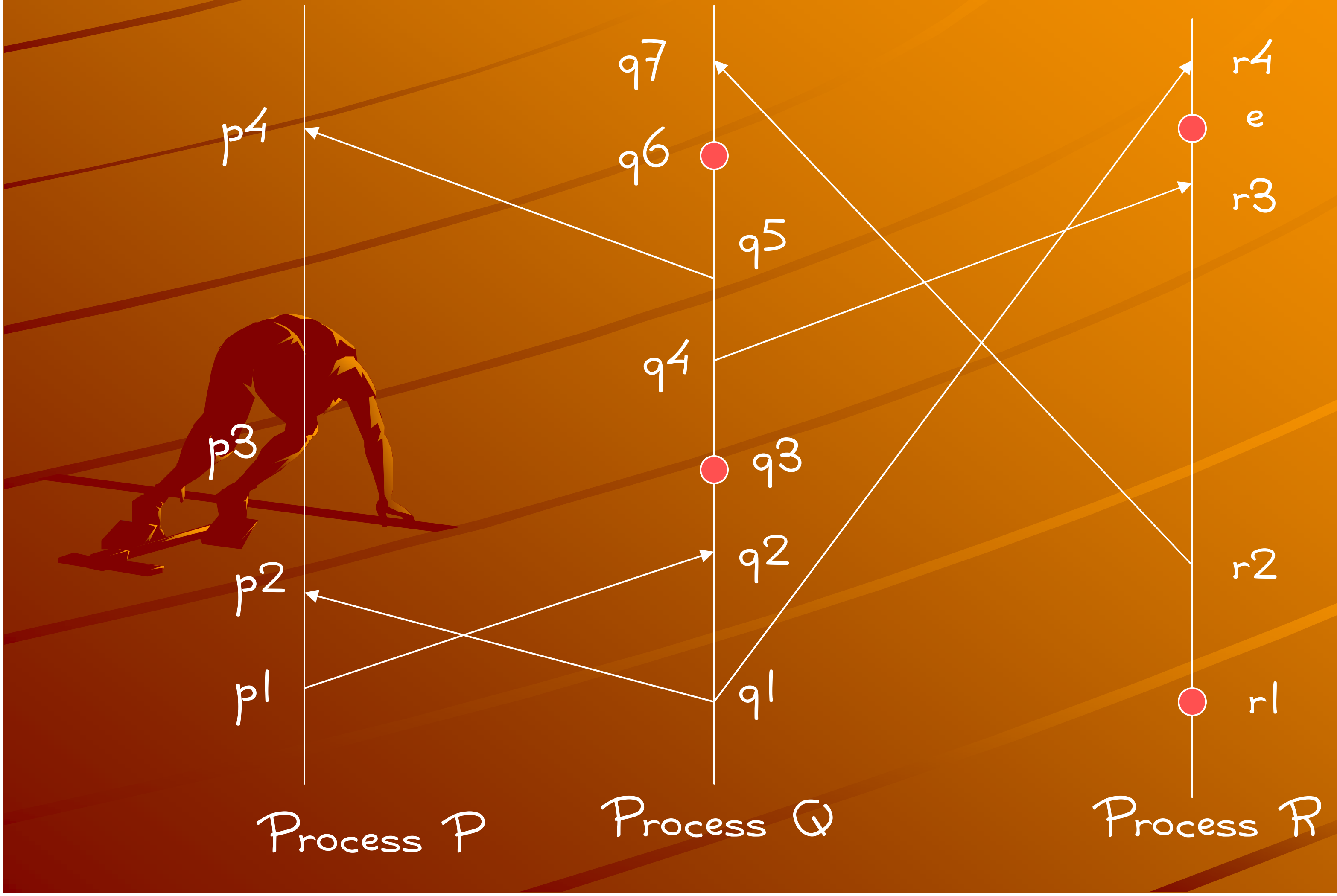
Example



Example


q7
p4
q6
r3
q5
q4
p3
q3
q2
r2
p2
p1
q1
r1
Process P
Process Q
Process R

Example


q7
r4
p4
e
q6
r3
q5
q4
p3
q3
q2
r2
p2
p1
q1
r1
Process P
Process Q
Process R

What can be said about the events in the above example?

- If a → b, then you will be able to move along the time lines and message links from a to b in the space time diagram
- a → b means that it is possible for a to causally affect b
- Two events are concurrent if neither can causally affect the other

Which events are concurrent?

- And which of them can be ordered?
- Not all can be totally ordered
- How to come up with a relation which will allow us to toally order all events in a distributed system

-→ In next class !

Towards Defining a total order: Logical Clocks

- Clock Ci for process Pi assigns a number Ci(a) to any event a in Pi

- This number is called Time Stamp for event a

- No assumptions about relation to physical time : hence logical clock.

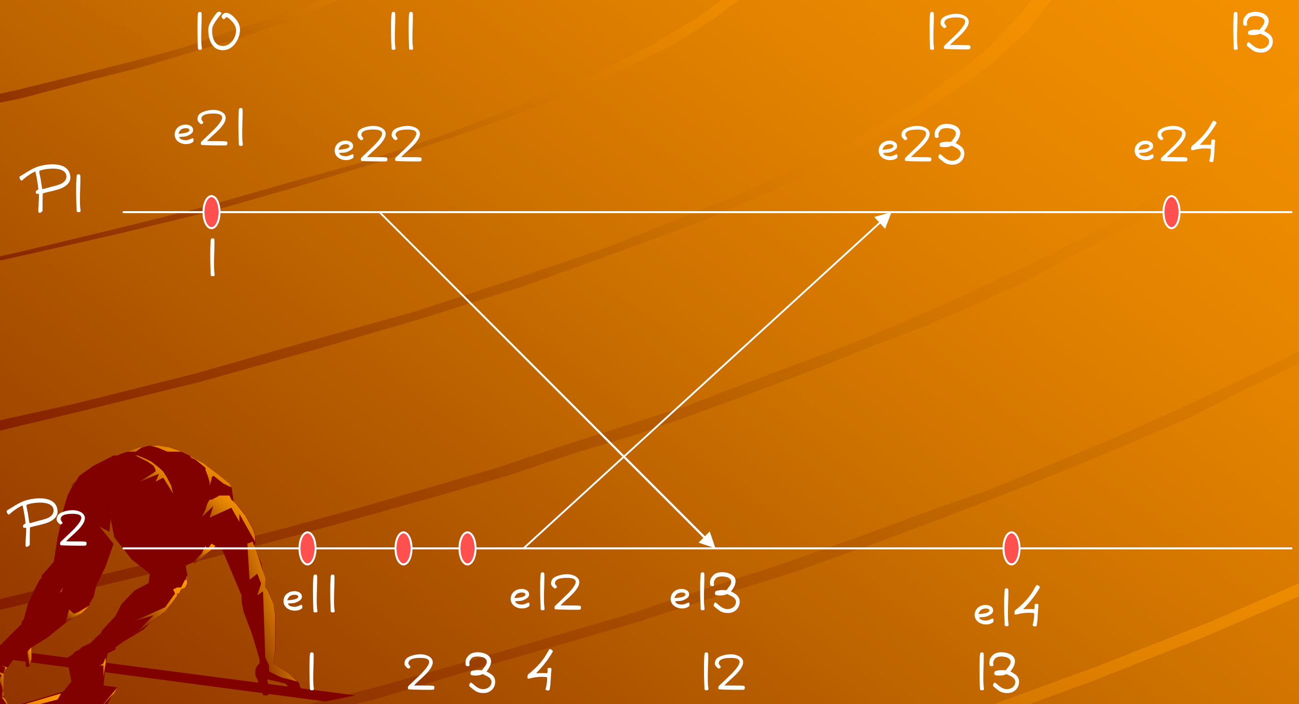
Conditions satisfied by the system of clocks

- If a → b then C(a) < C(b)

  if a, b are in same process Pi
  Ci(a) < Ci(b)

  if a is send event in Pi and
  b is the receive event in Pj
  Ci(a) < Cj(b)

Example



With → relation, there is no total order as there is always a possibility of concurrent events. If you apply clocks to order, equal time stamps are still a problem

How to implement the above conditions for clock values for the happened before relation?

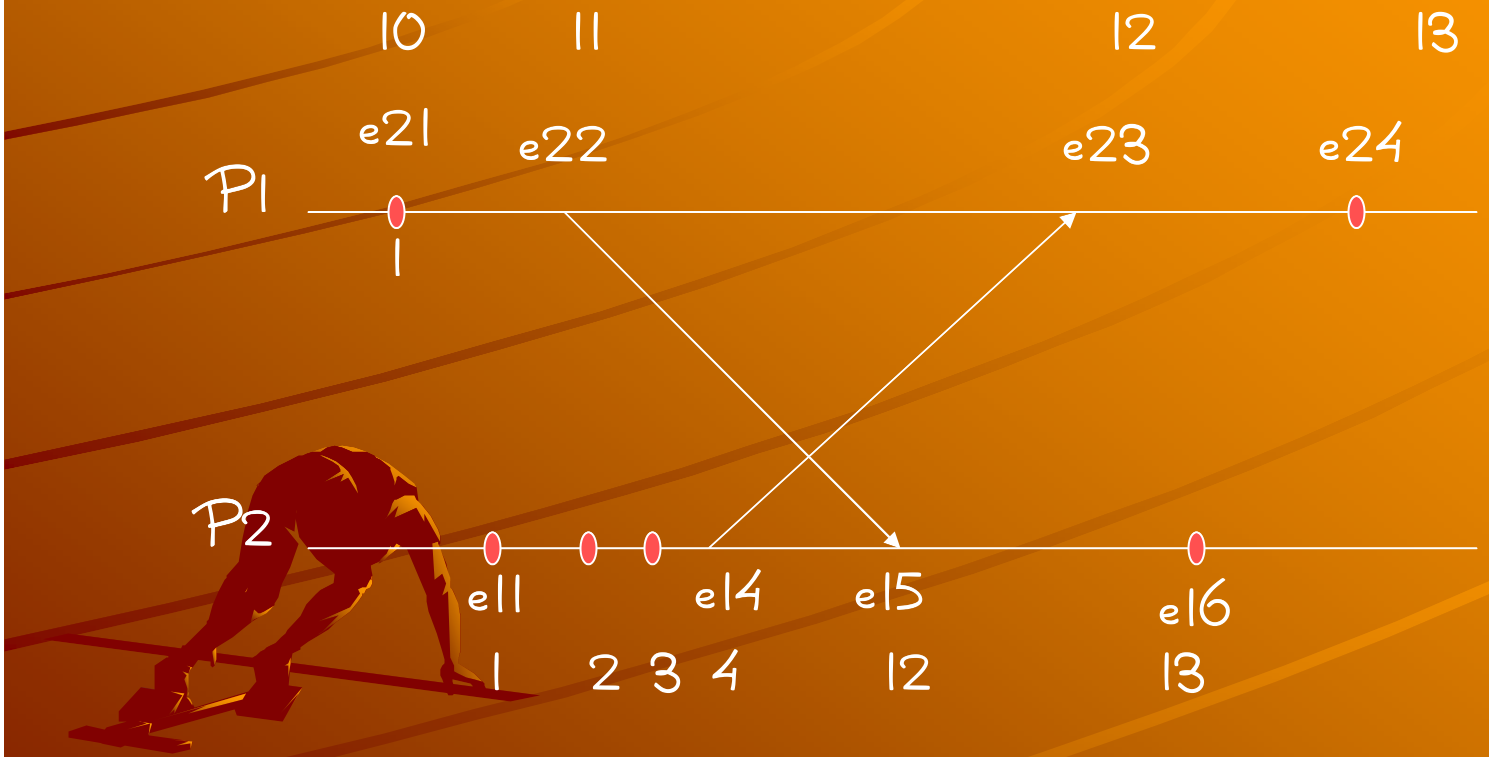
- If a and b are two successive events in same process, a → b implement $C_i(b) = C_i(a) + k$ $(k>0)$

If a is send event in P_i and its corresponding receive event is b in P_j,
implement $C_j(b) = \max(C_i(a), C_j(b-1)) + k$ $(k>0)$

Obtaining Total Order with relation →

- If a is an event in P_i and b is an event in in P_j, then a → b iff
  Either $C_i(a) < C_j(b)$ or
  $C_i(a)=C_j(b)$ and $P_i \prec P_j$ such that
  relation $\prec$ totally orders processes

Example



With → relation, there is total order as there is always a total order:
E11 e12 e13 e14 e21 e22 e23 e15 e24 e16

Reference Reading Material for the course

- Leslie Lamport, Time, Clocks and ordering of events in a distributed system, CACM, July 1978, pp.558-565.