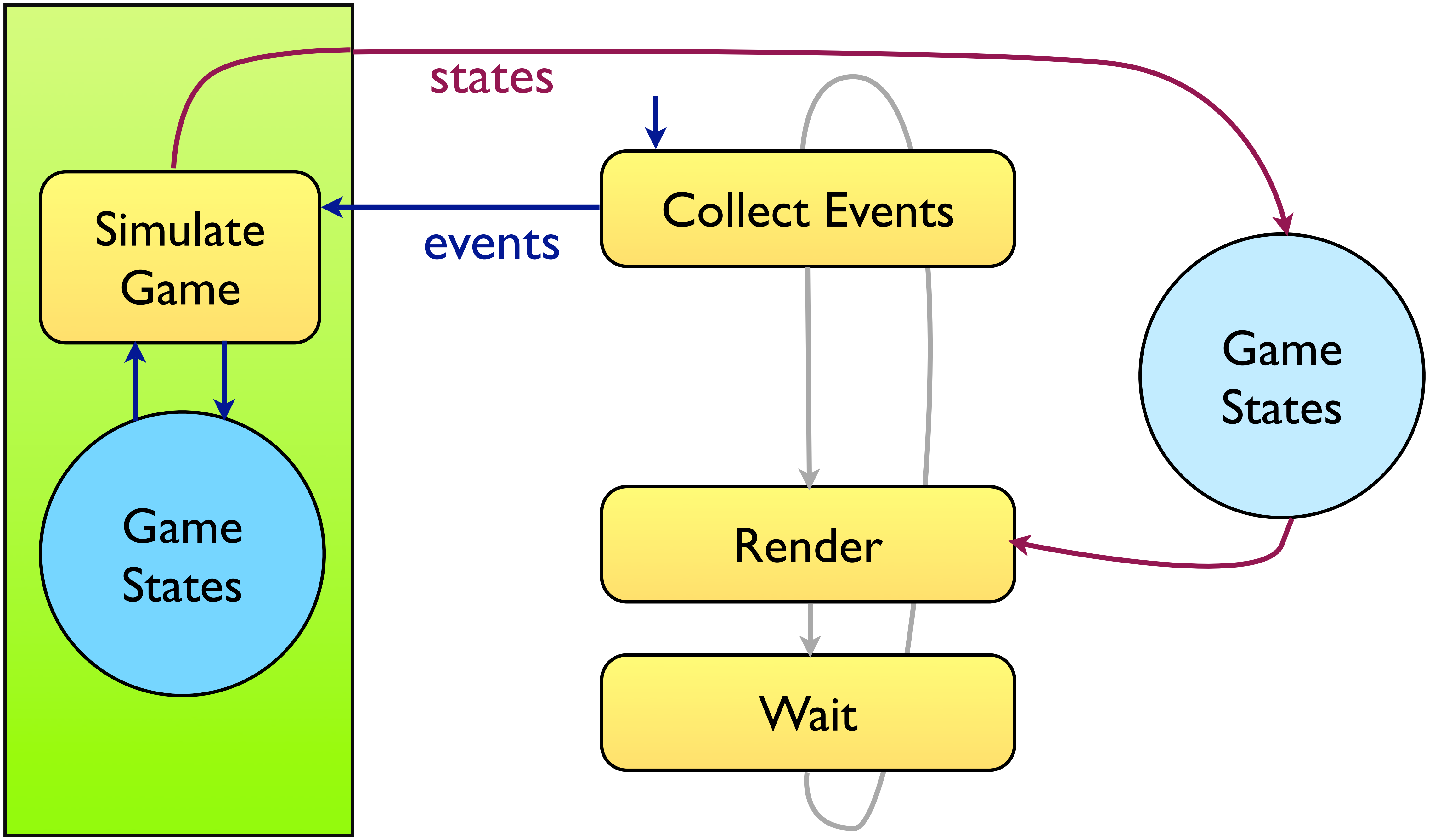
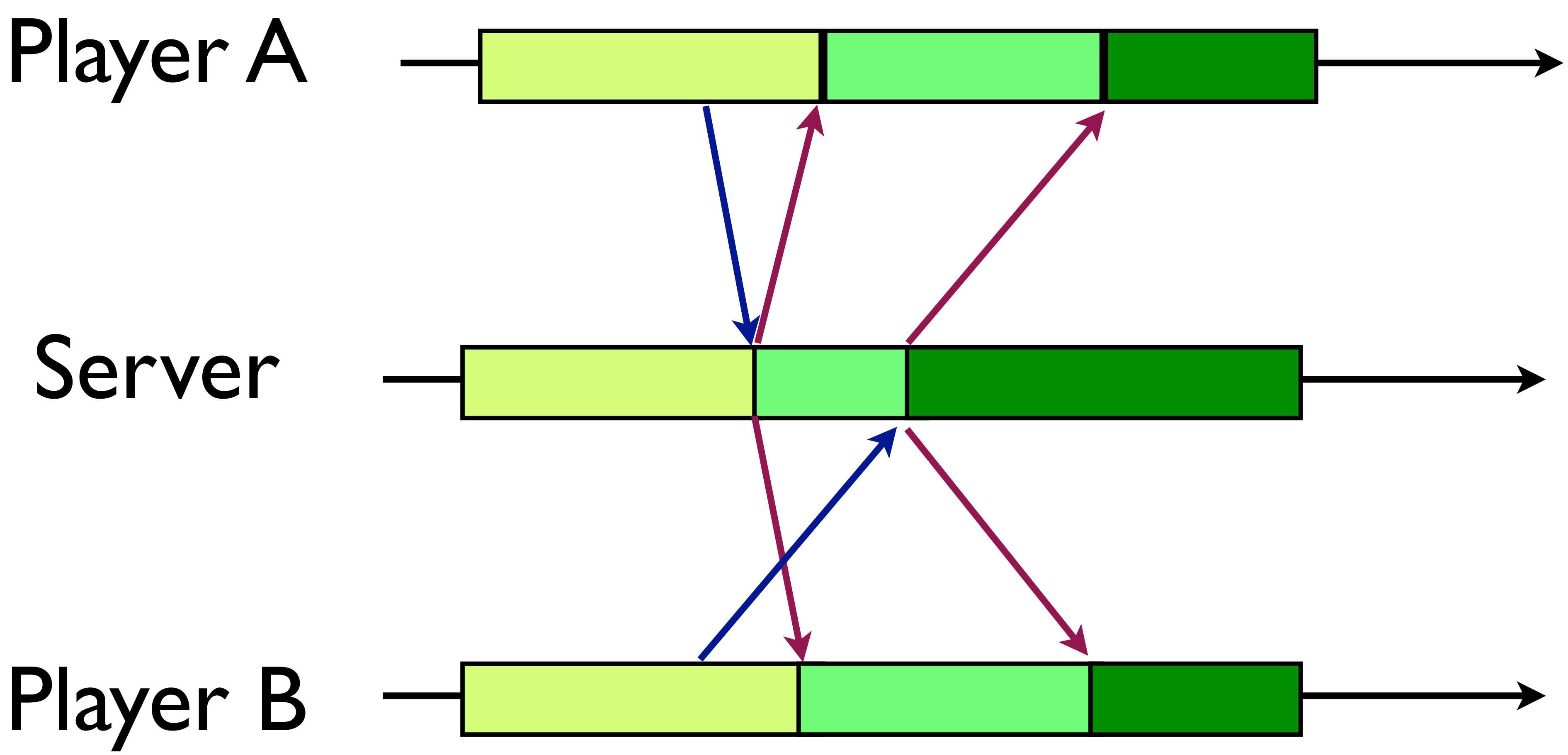
Centralized Server Architecture

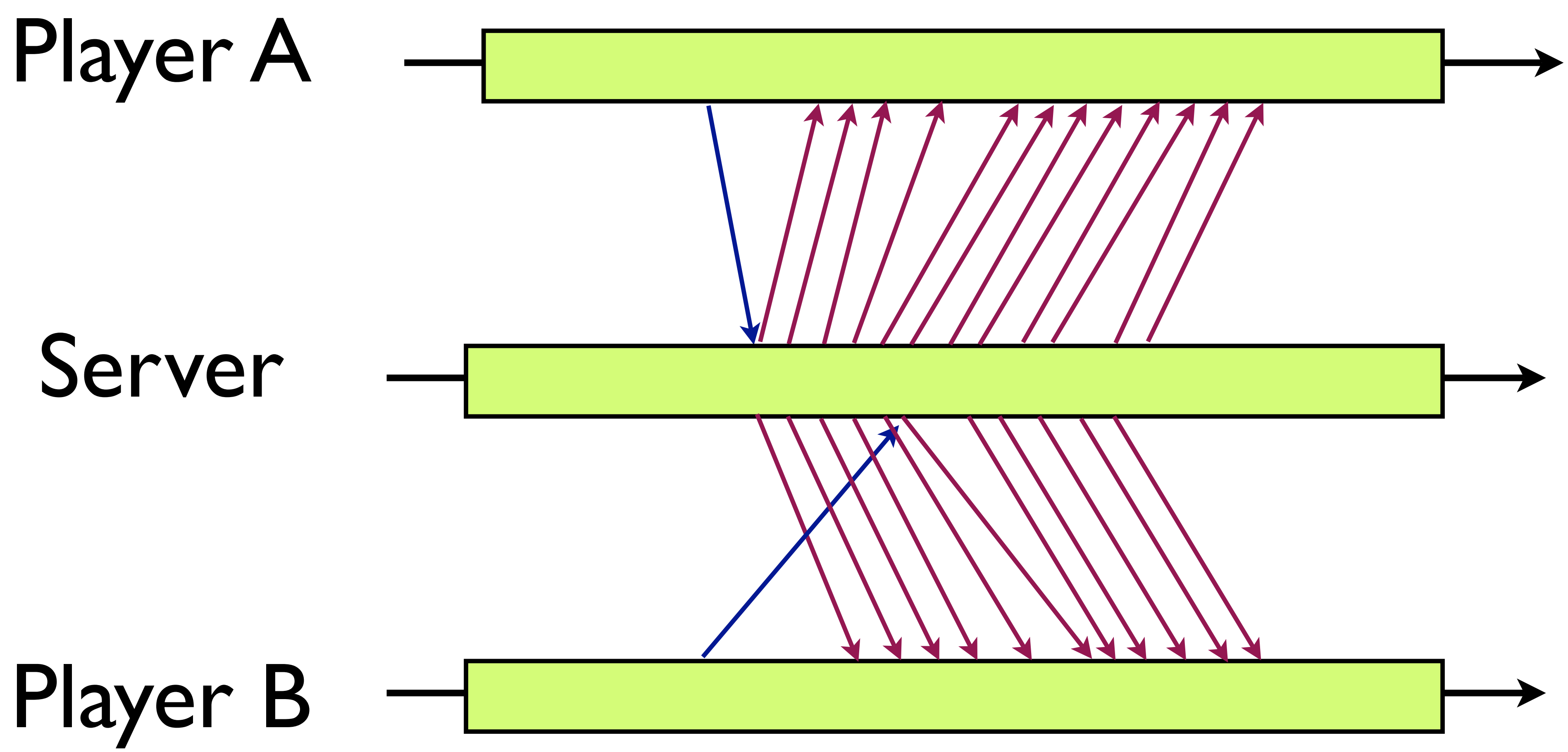


states
Simulate Game
events
Collect Events
Game States
Render
Wait
Game States

What if a state changes continuously (i.e., is a function of time) ?



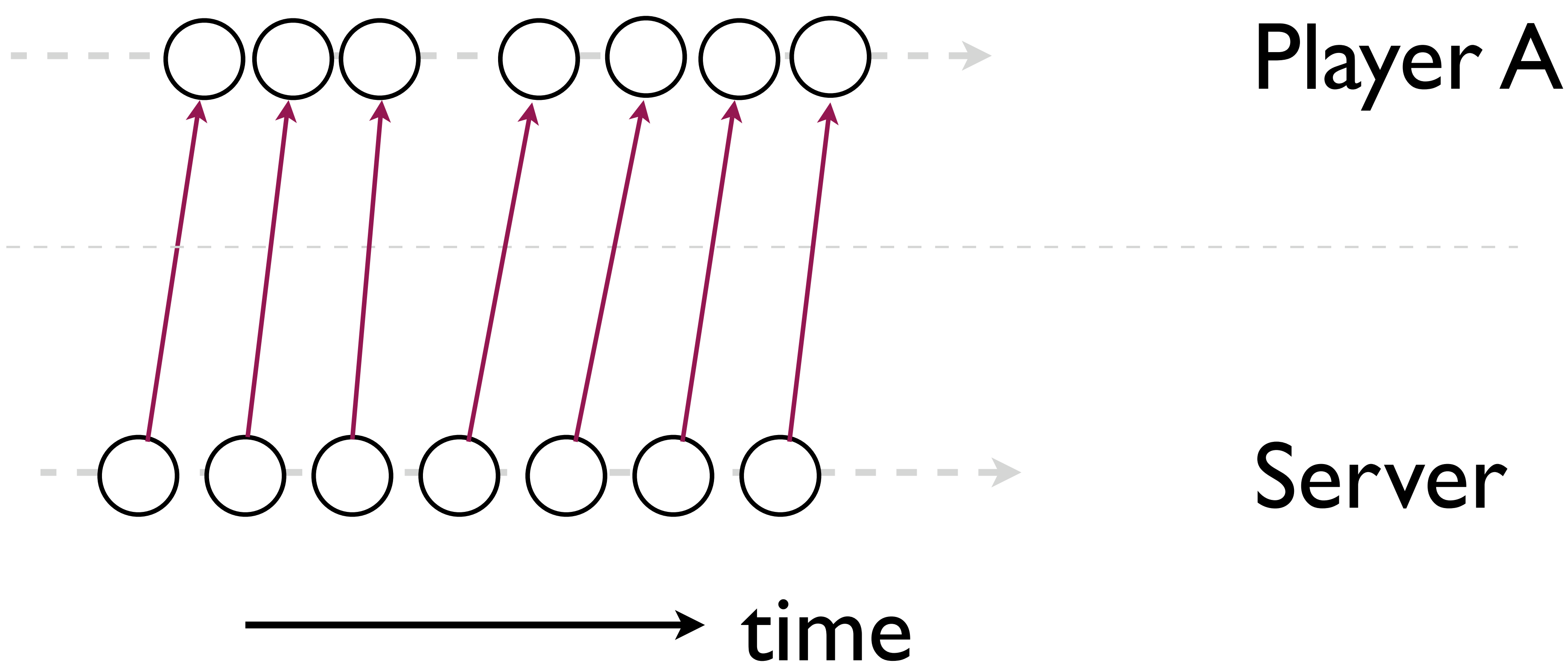
Consider position updates of players. Players send move command. Server replies with positions periodically.



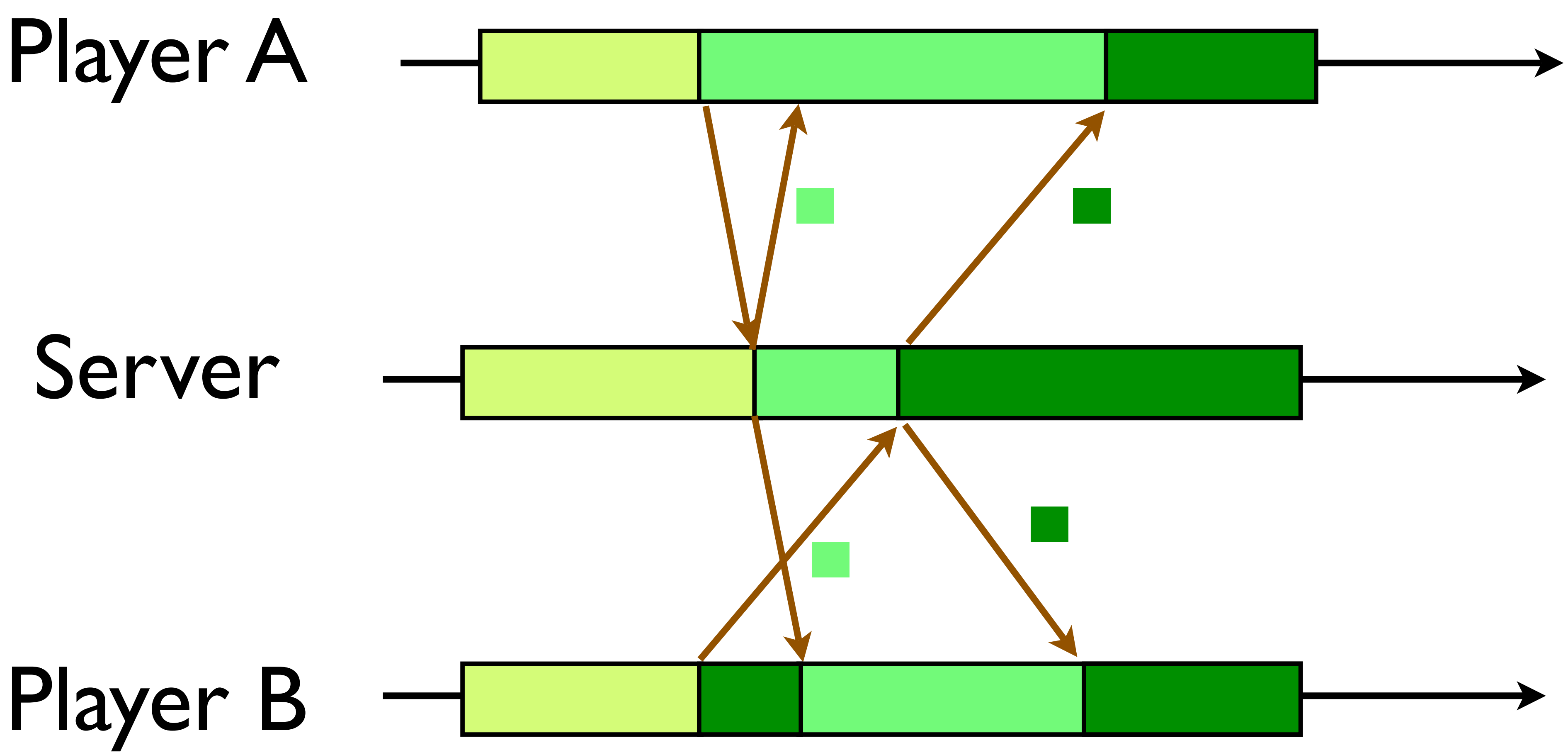
Two issues:

1. Message overhead
2. Delay jitter

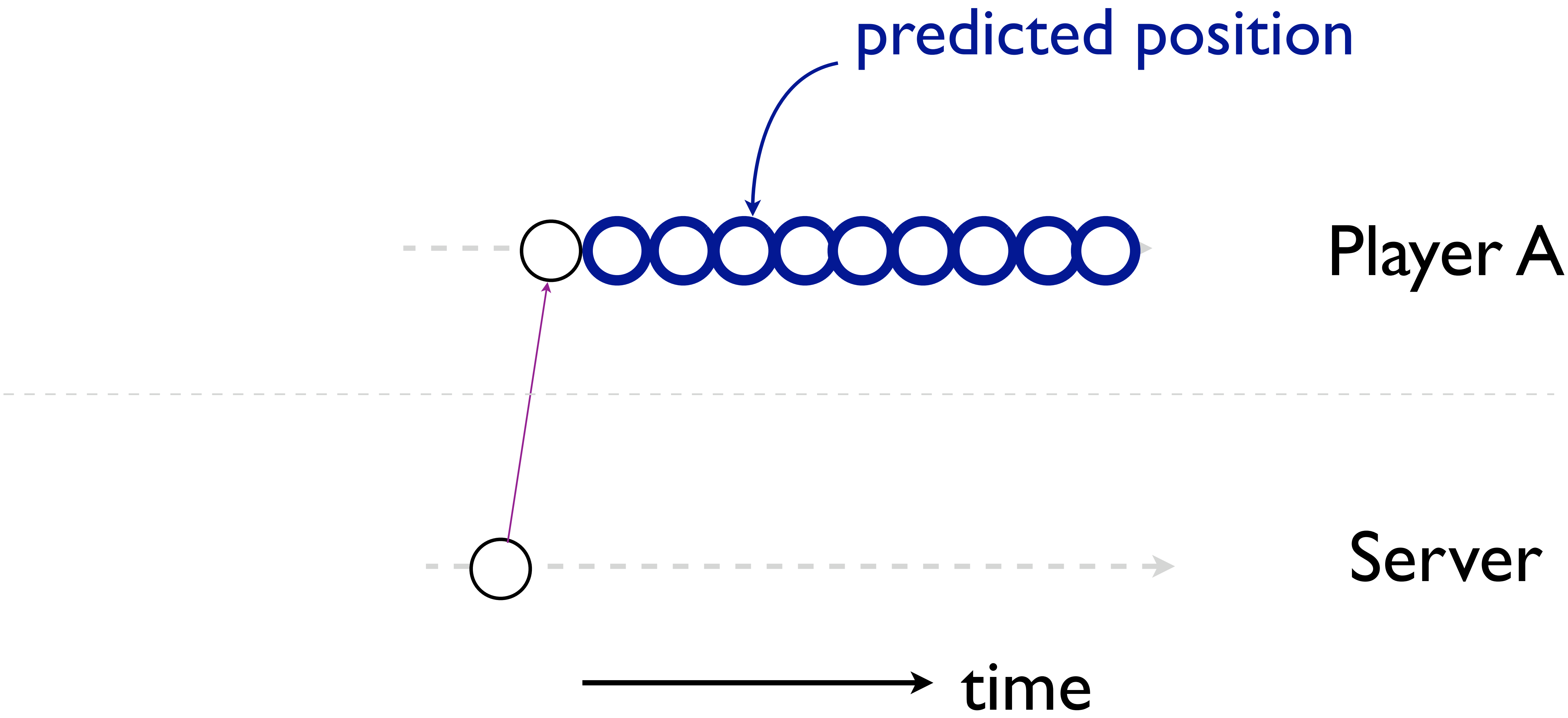
Delay jitter causes player's movement to appear erratic.



Recall: Short circuiting is used to predict own states. We can similar predict opponent's state.



Suppose the velocity remains constant, then we can predict every position at all time.

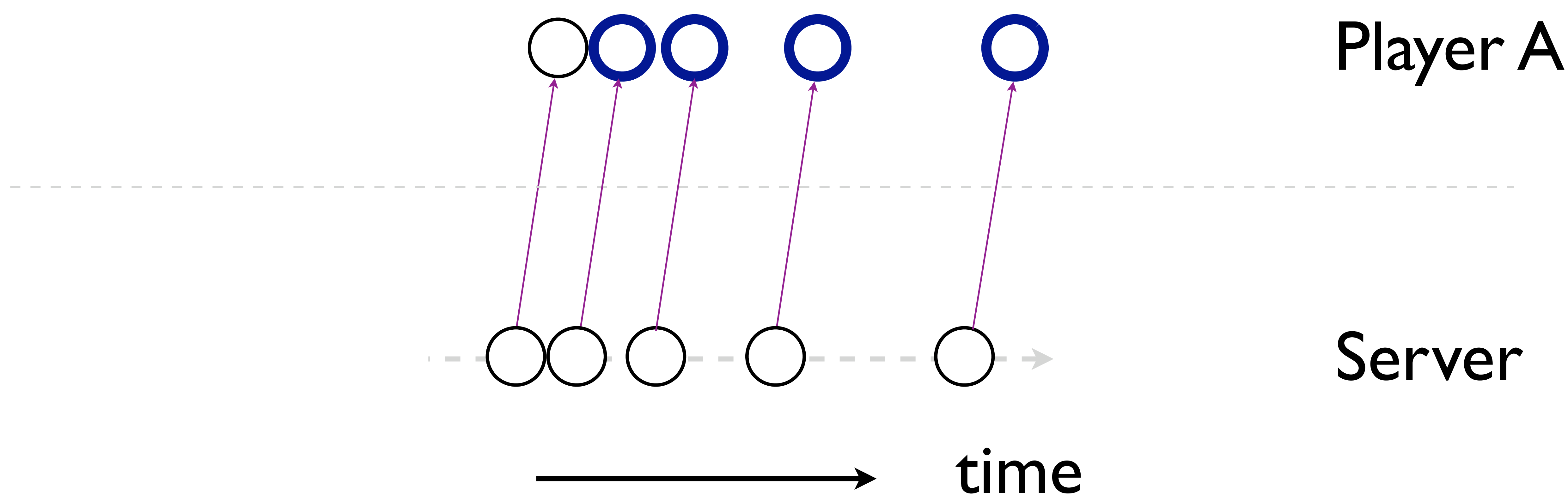


$x[t]$ position of entity at time t

v velocity of the entity

$$x[t_i] = x[t_{i-1}] + v \times (t_i - t_{i-1})$$

But velocity may change all the time (e.g. a car accelerating). To counter this, we send position, velocity, and acceleration as update.



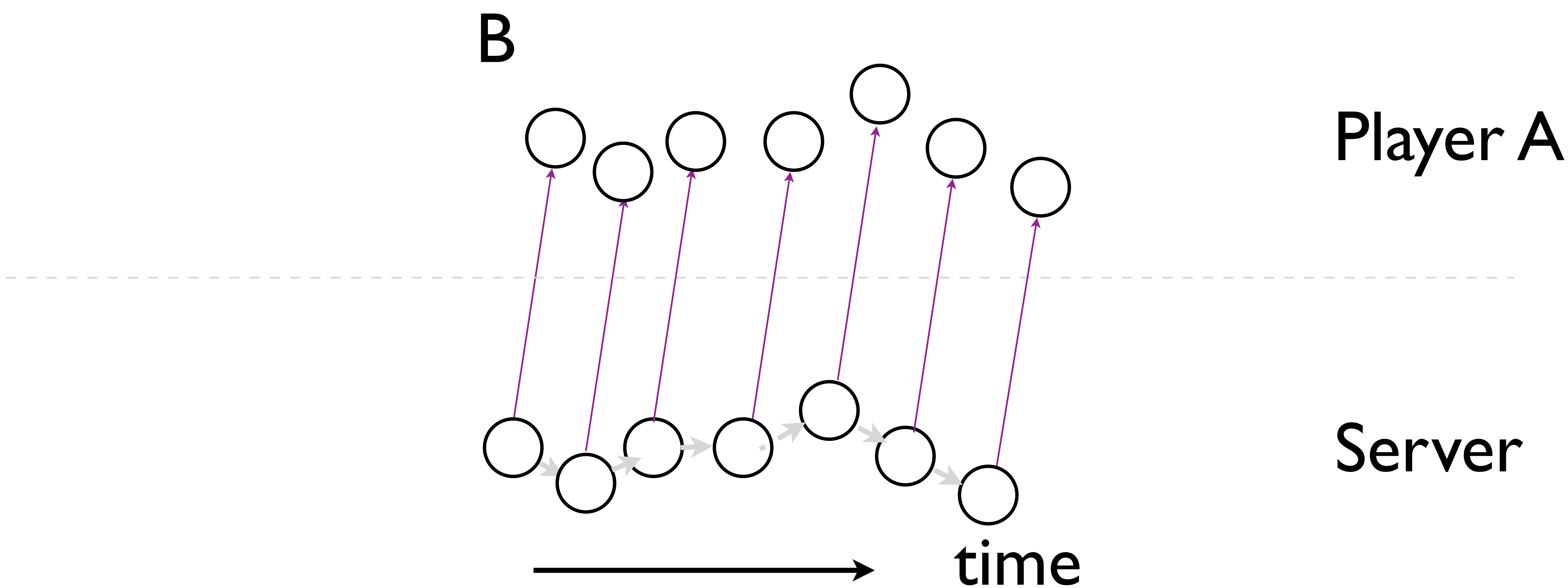
$x[t]$ position of entity at time t

v velocity of the entity

a acceleration of the entity

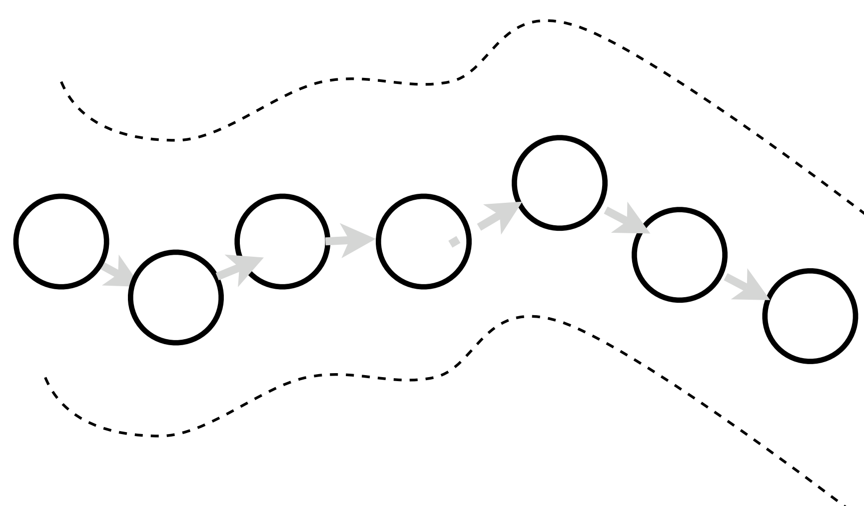
$$x[t_i] = x[t_{i-1}] + v(t_i - t_{i-1}) + \frac{1}{2}a(t_i - t_{i-1})^2$$

We will still need substantial number of updates if the direction changes frequently (e.g. in a FPS game).



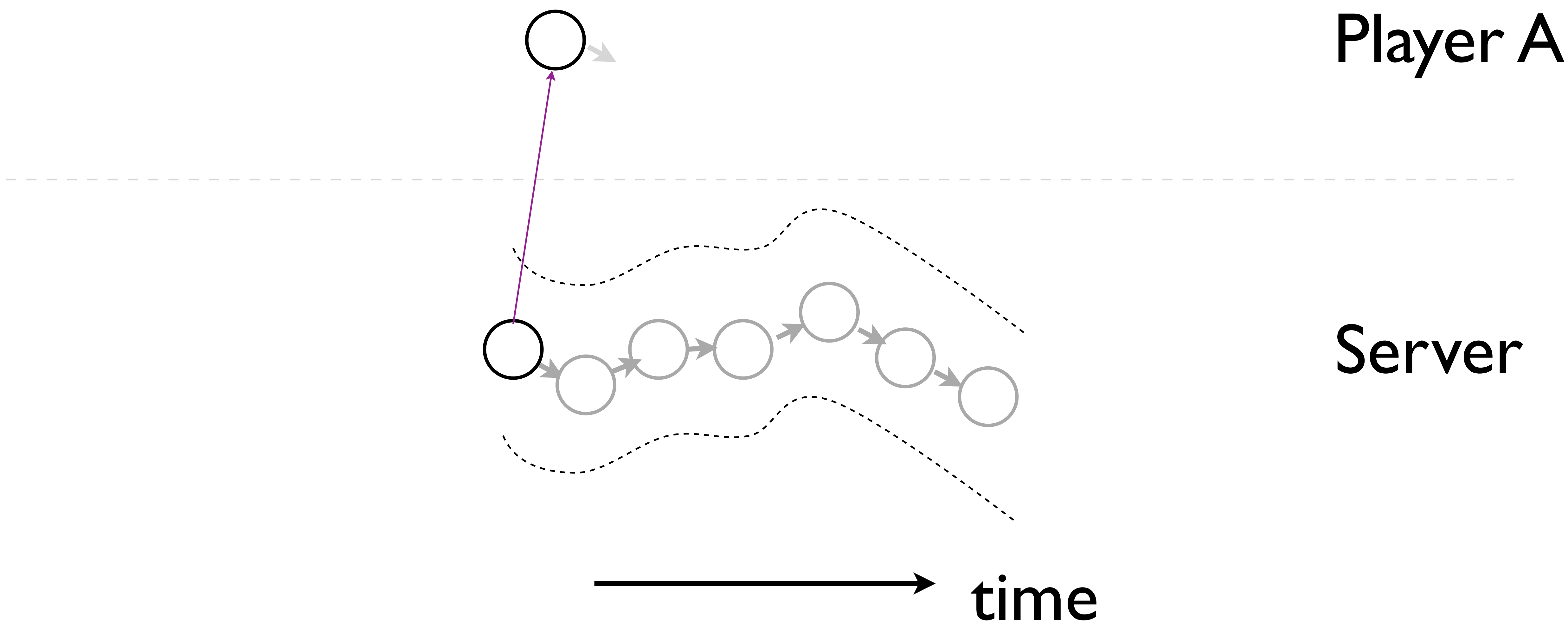
idea: trade-off message overhead and accuracy. No need to update if error is small.

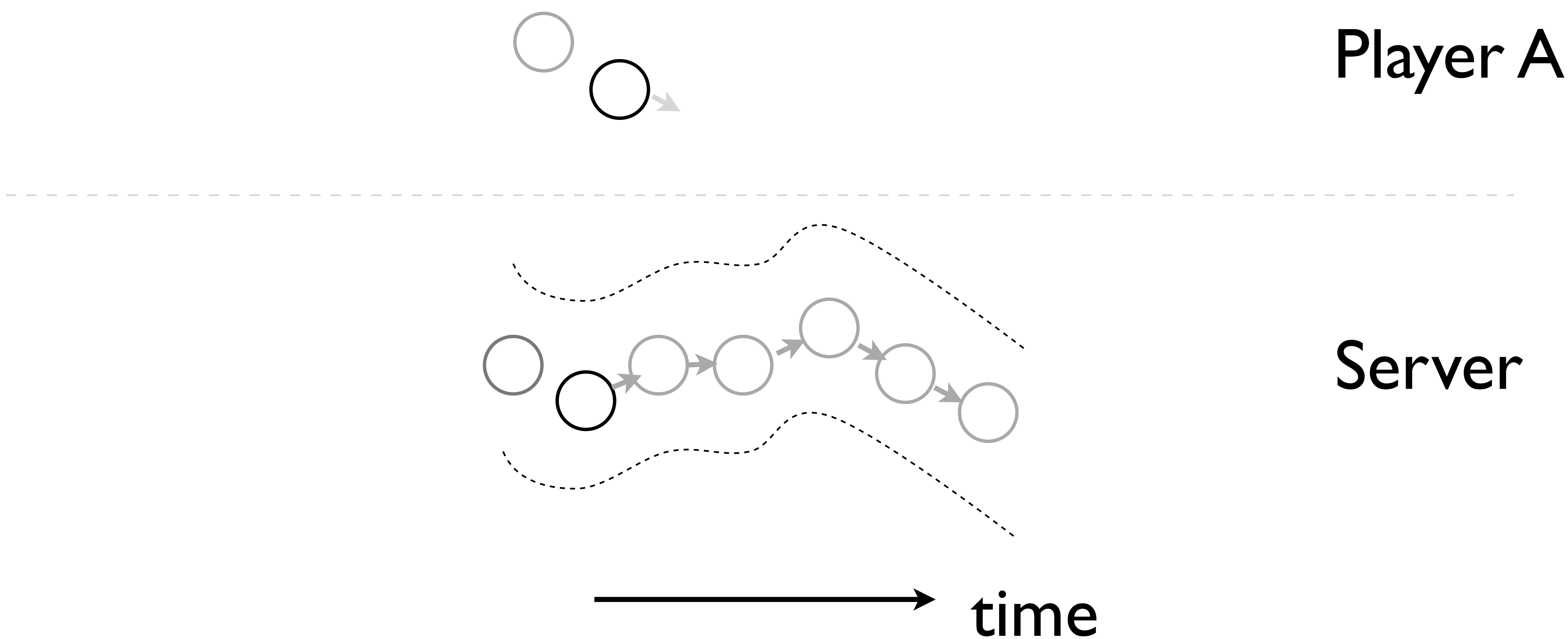
Player A



Server

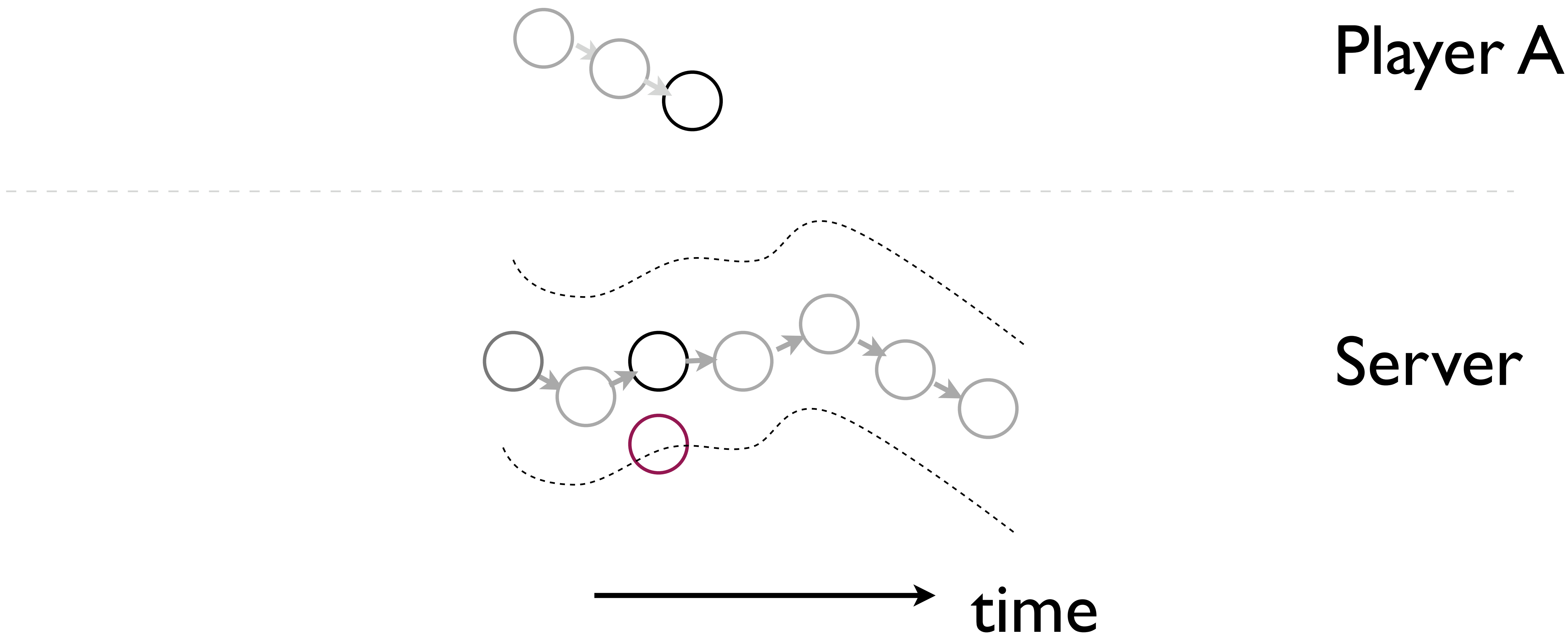
time


Player A
Server
time

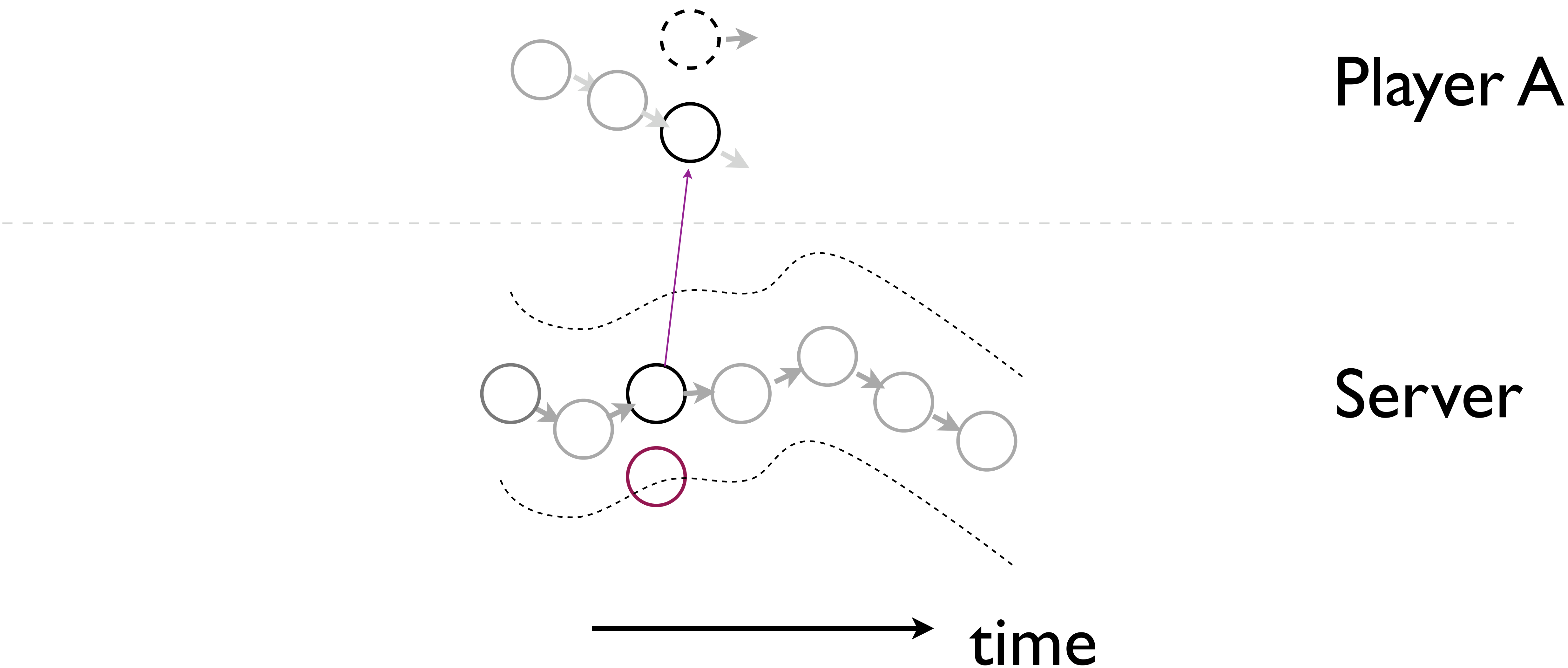

Player A
Server
time

○ predicted position at A

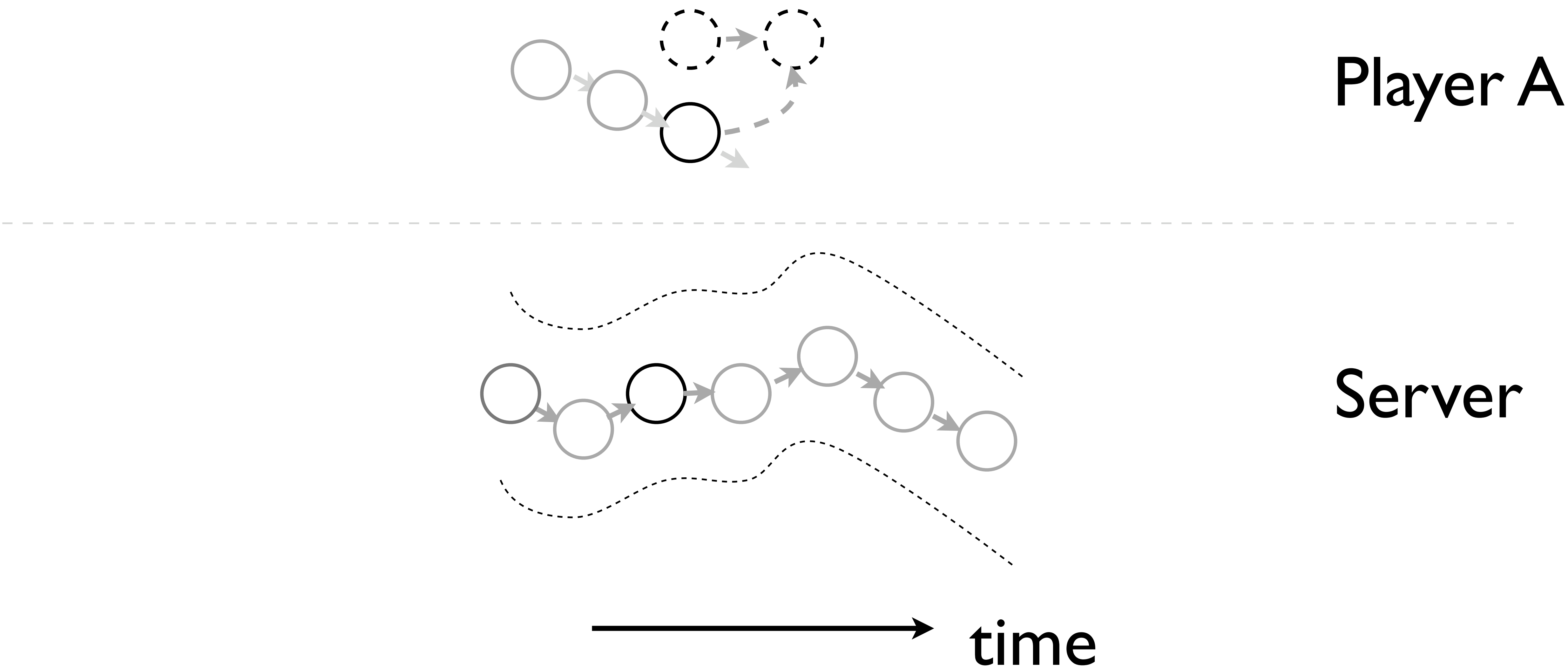
○ A’s predicted position at the server



A's version of the entity's position is now too far away from the correct position.   Server updates A with the new velocity and position.



A converges the entity to the correct position smoothly.



How to set threshold?

adapt based on game requirement

(e.g. distance to other players)

Any drawbacks?

higher CPU cost

(needs to simulate other players)

unfair

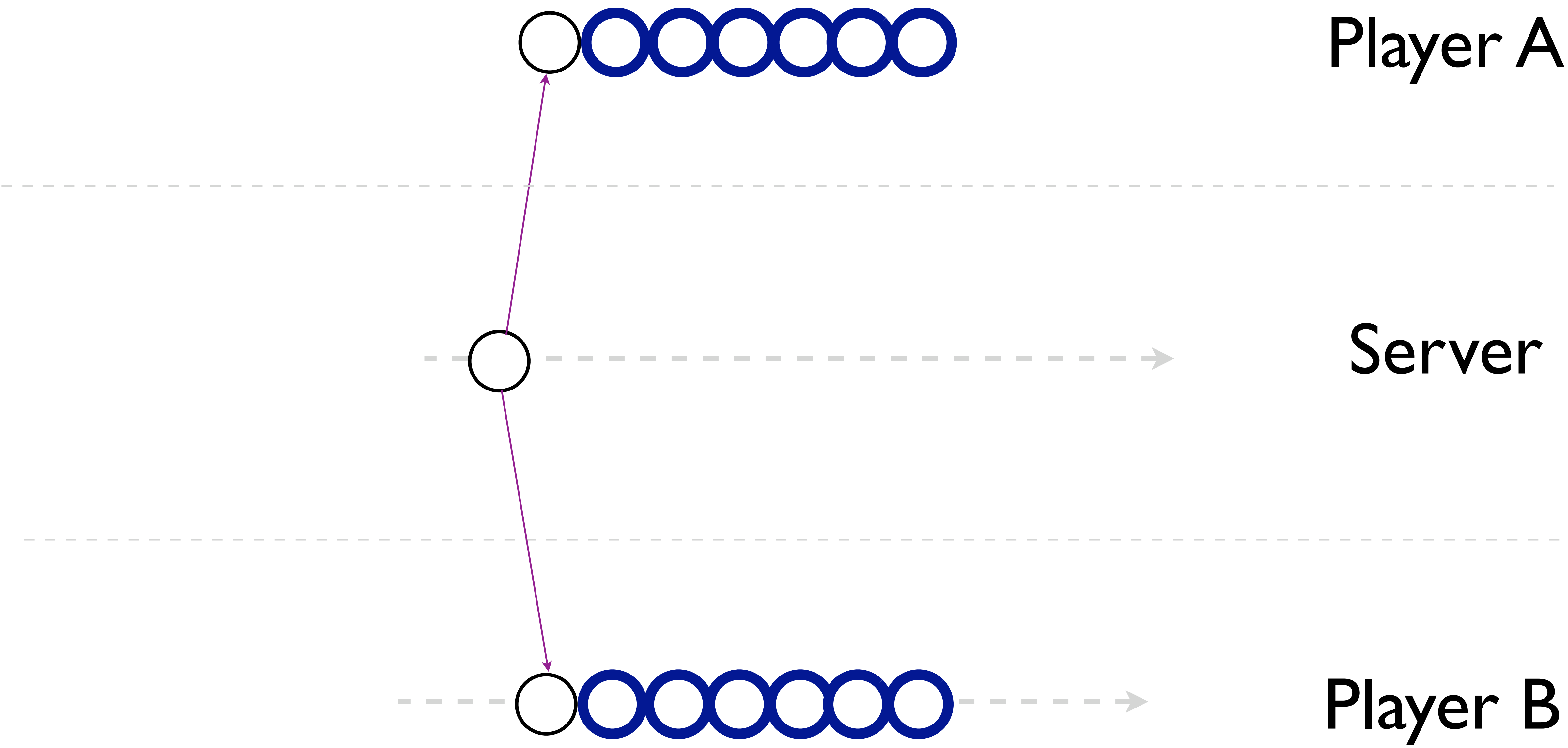
(higher latency leads to larger error)

Dead Reckoning

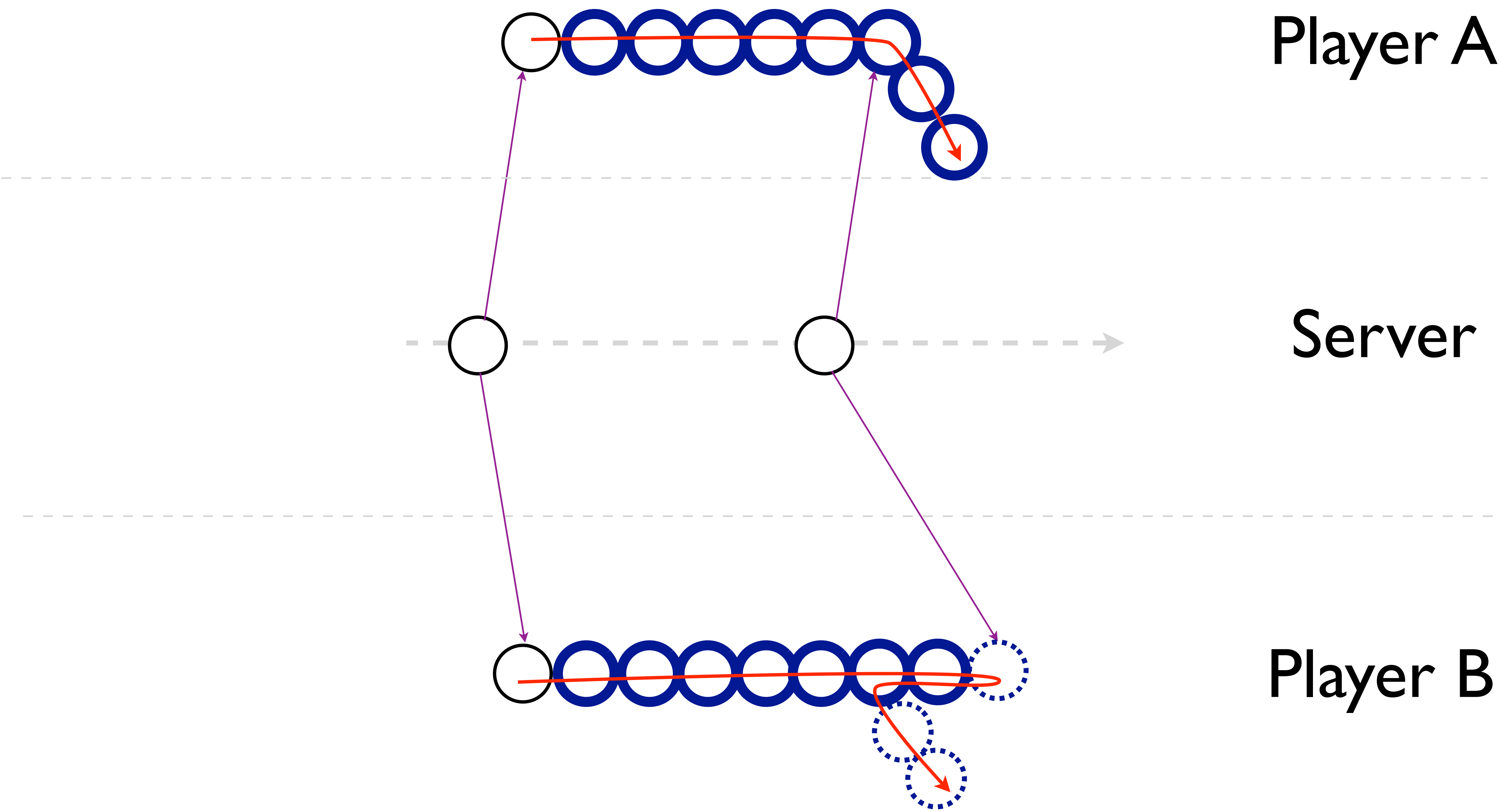
Generalized Dead Reckoning : Prediction Contract

“return to base”
“drive along this road”

Delay jitter would still cause problem.



Solution: add timestamp

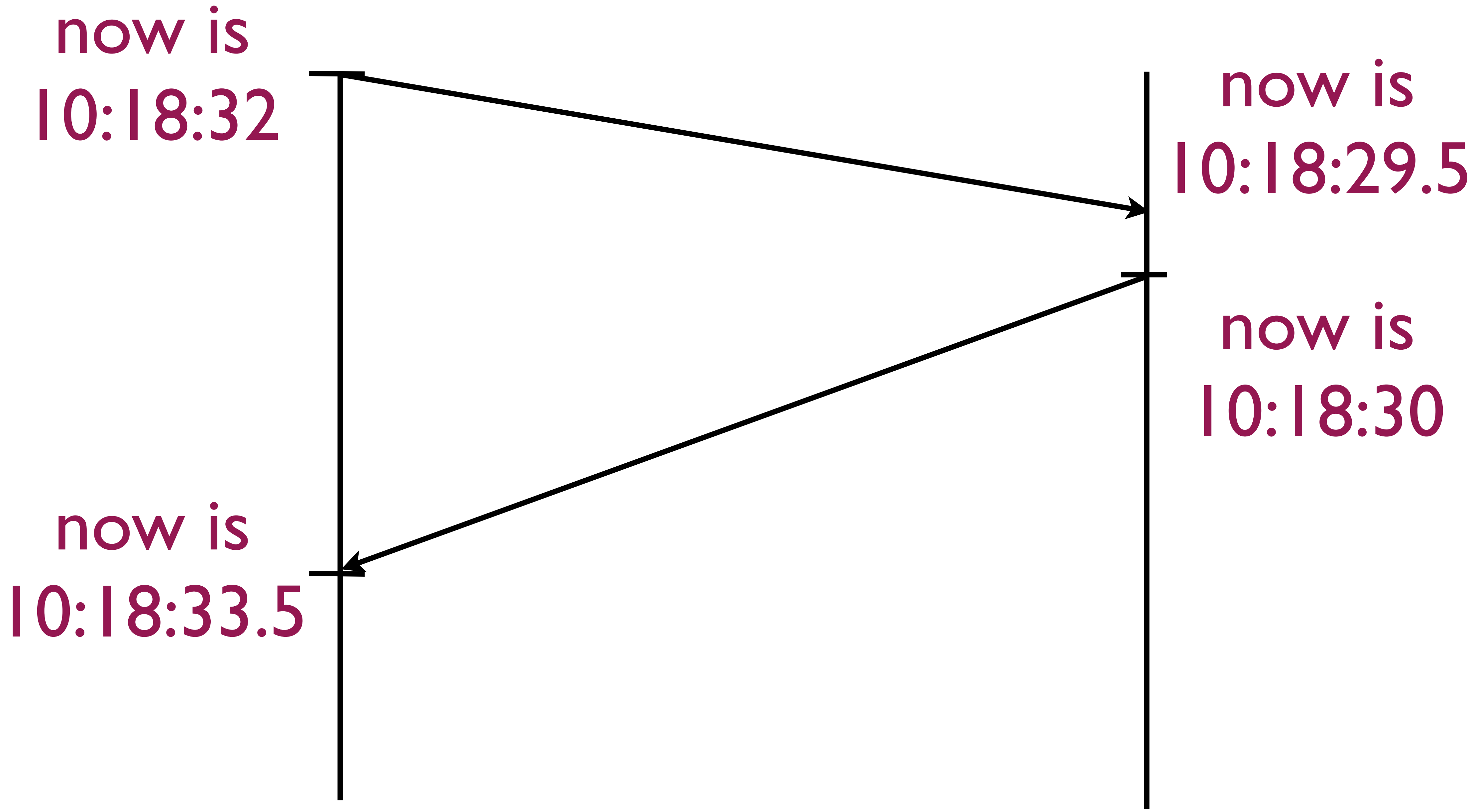


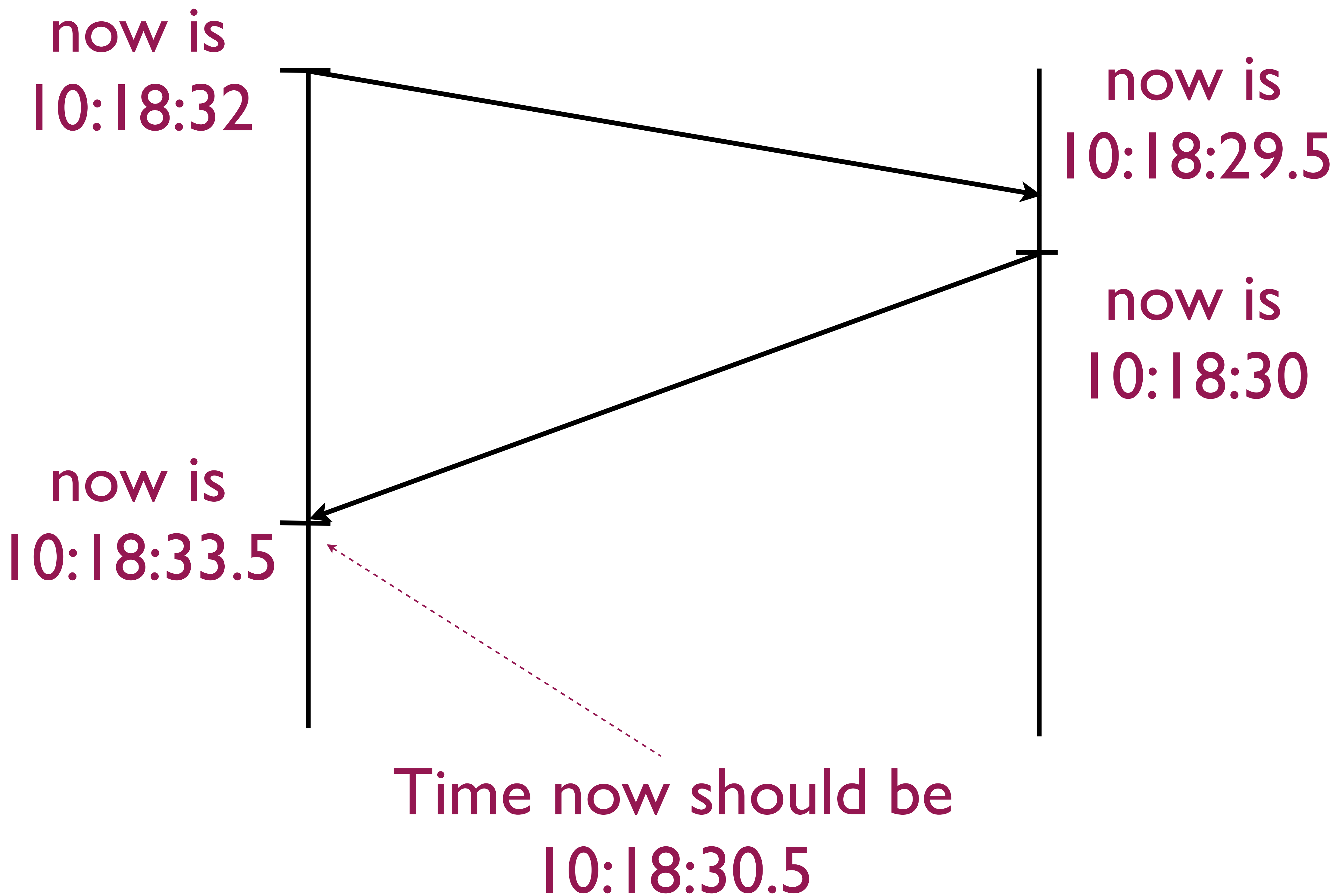
Server and clients must have a common notion of "time"

Two choices:

Wall Clock
Game Clock

Wall Clock: clients and server have to synchronize their physical clocks using NTP or SNTP.

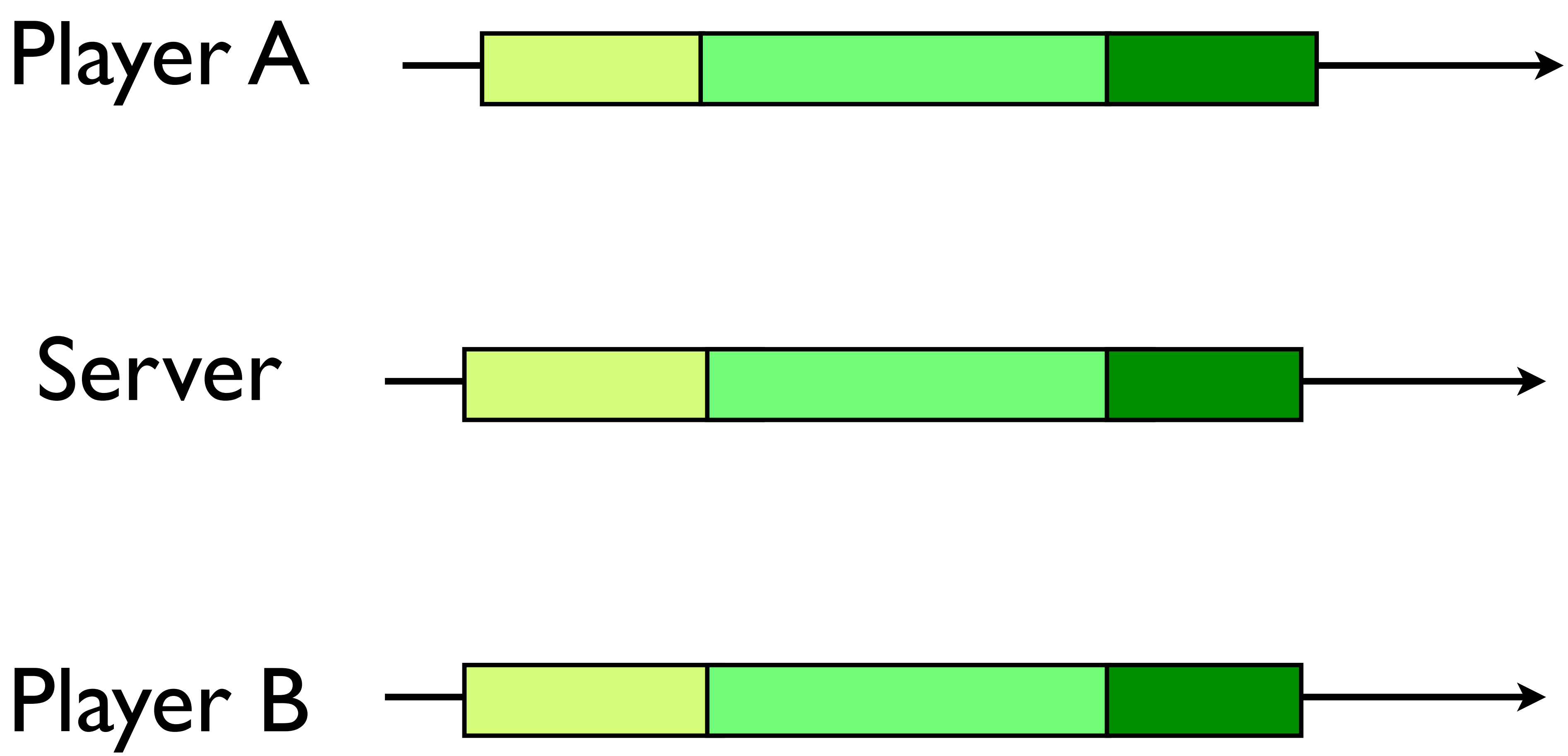

now is
10:18:32
now is
10:18:29.5
now is
10:18:30
now is
10:18:33.5



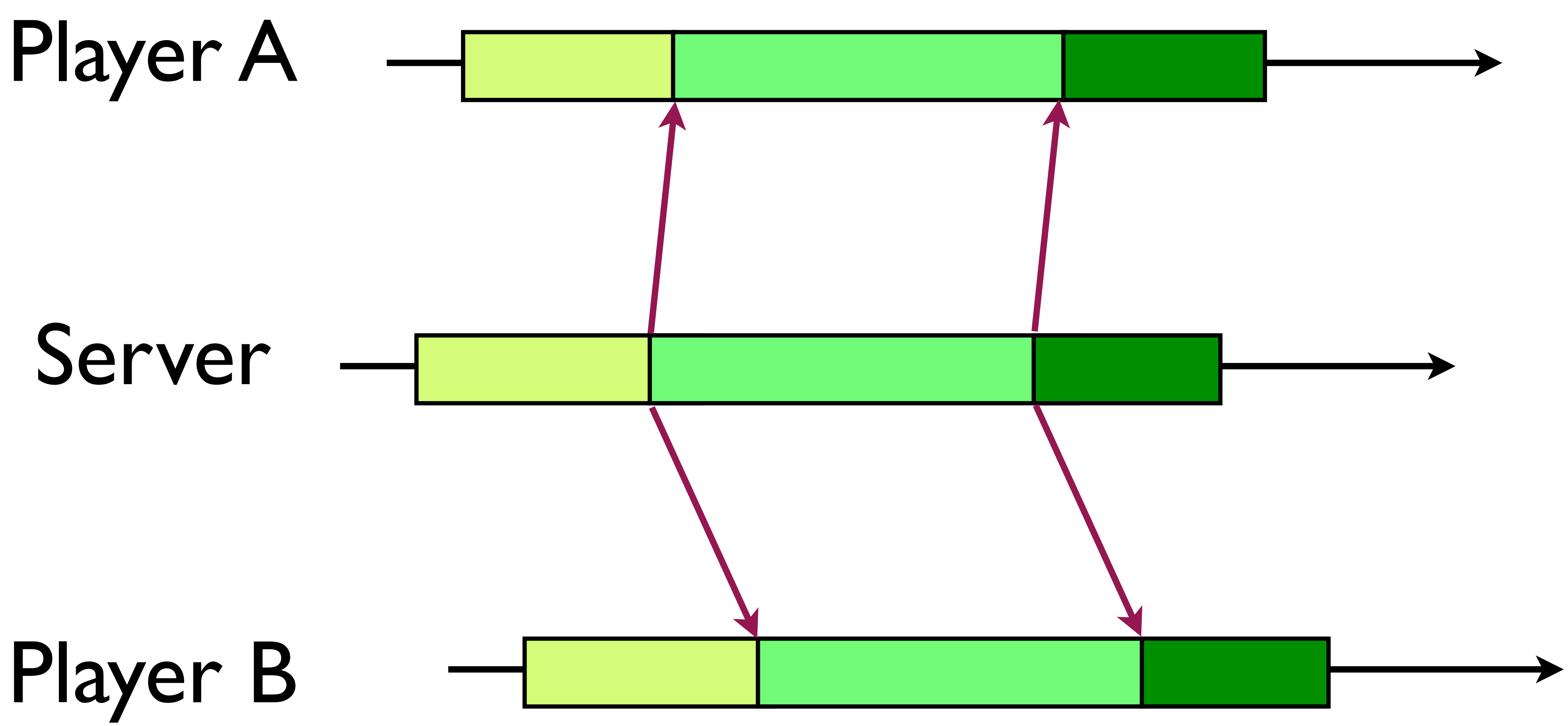
RTT = 1s

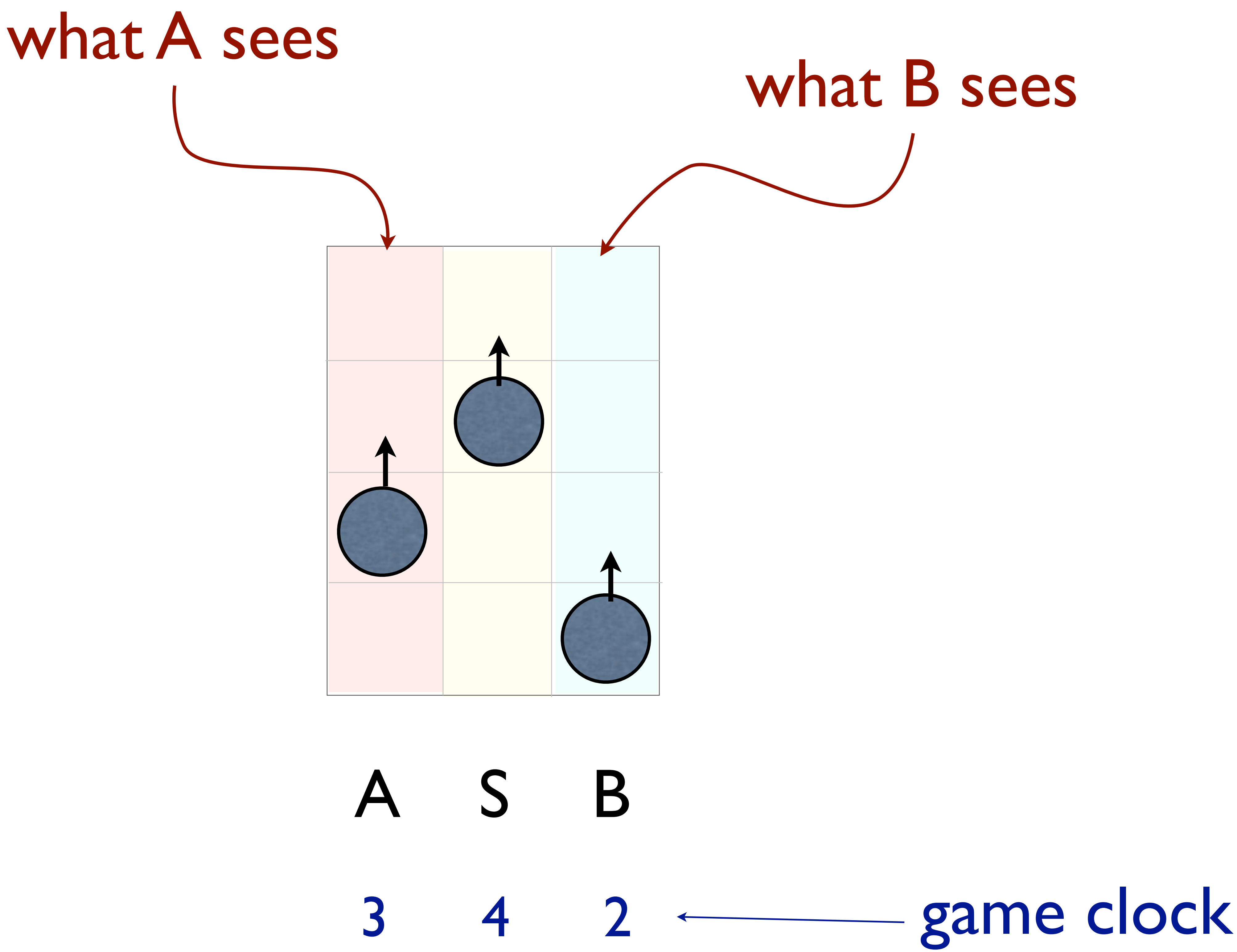
OWD = 0.5s

Players try to sync the game states with the server at the same wallclock time, and predict ahead with an amount of time equal to one-way delay.



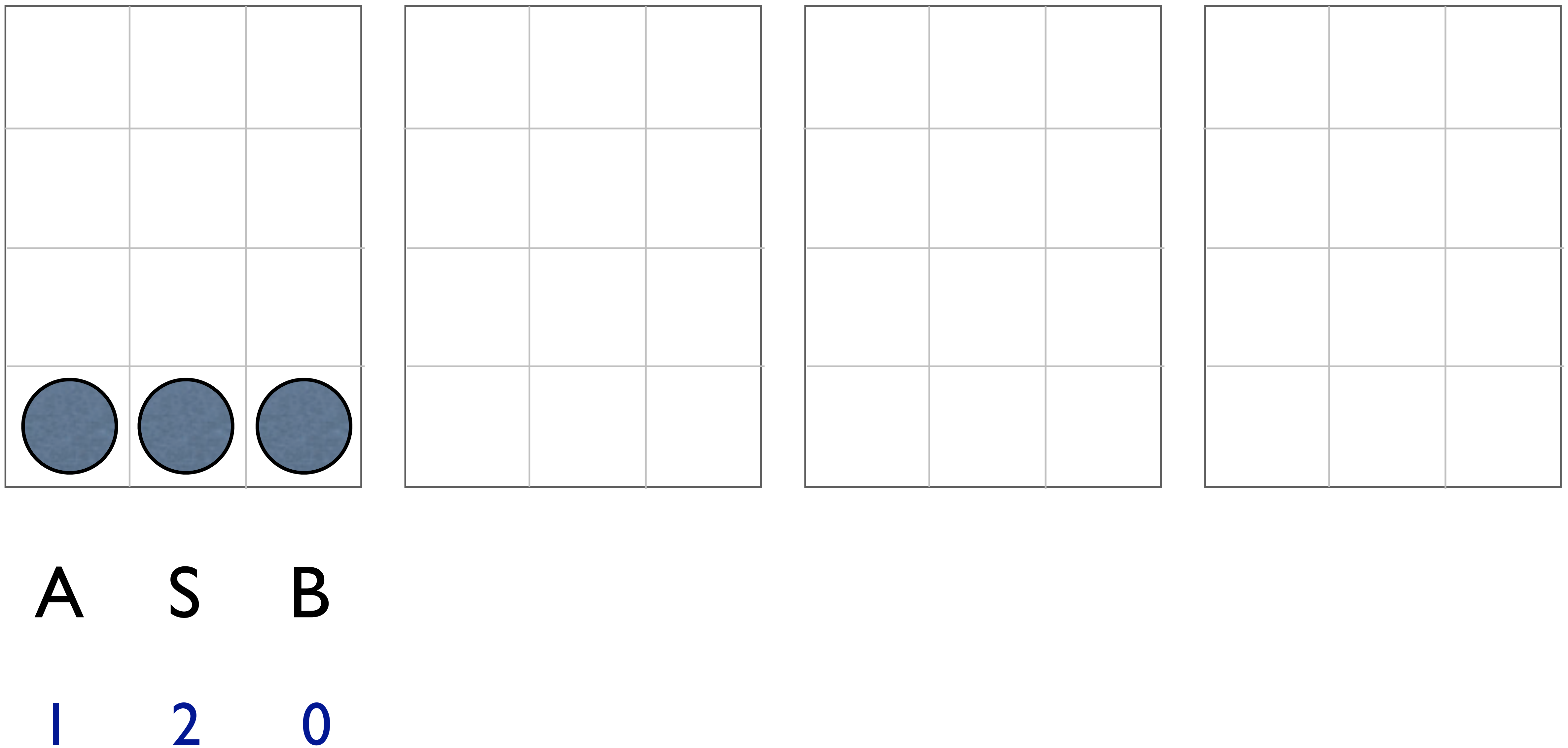
or: synchronize states to game clock, which runs behind the server by an amount of time equals to the one-way delay.



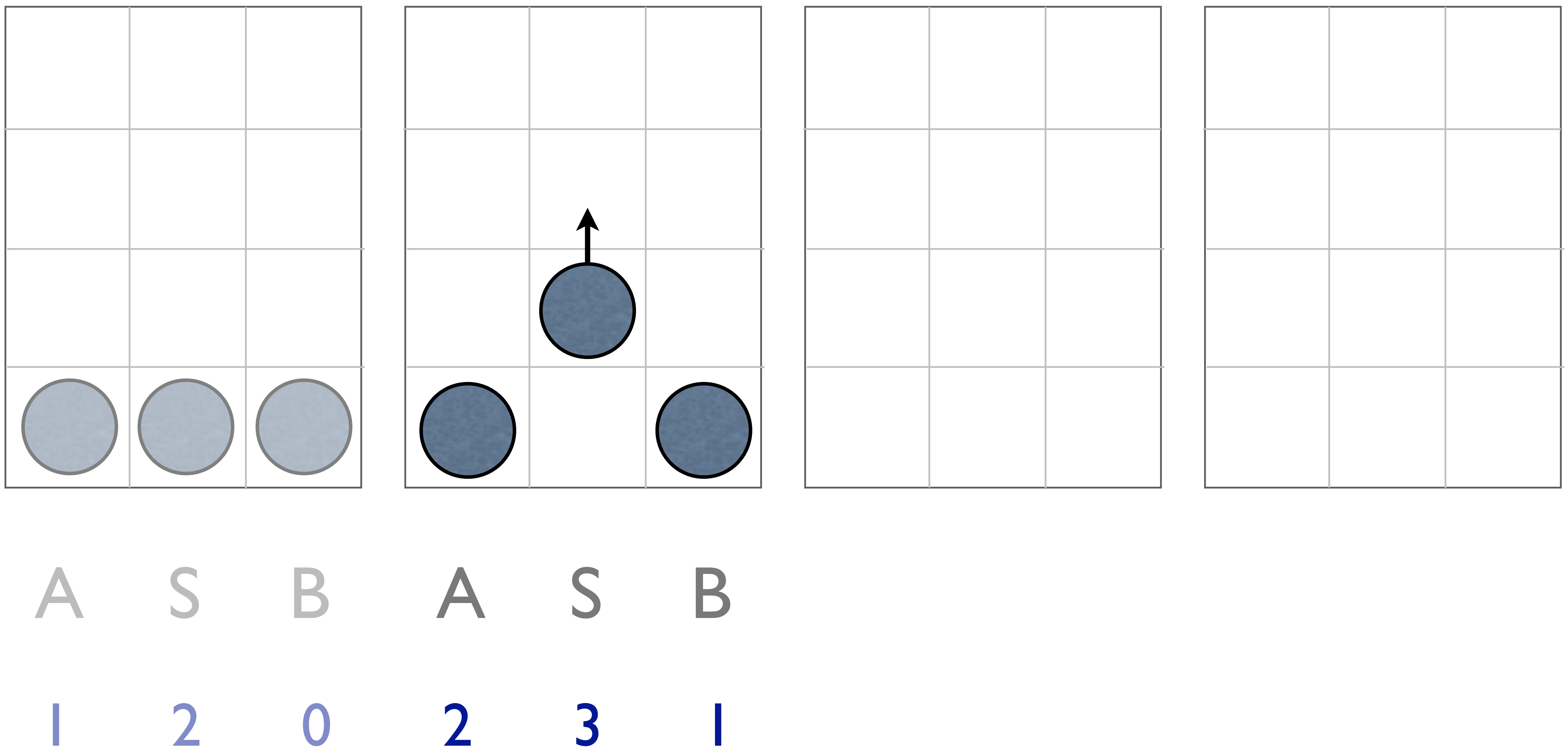

what A sees
what B sees
A
S
B
3
4
2
game clock

Example: using game clock

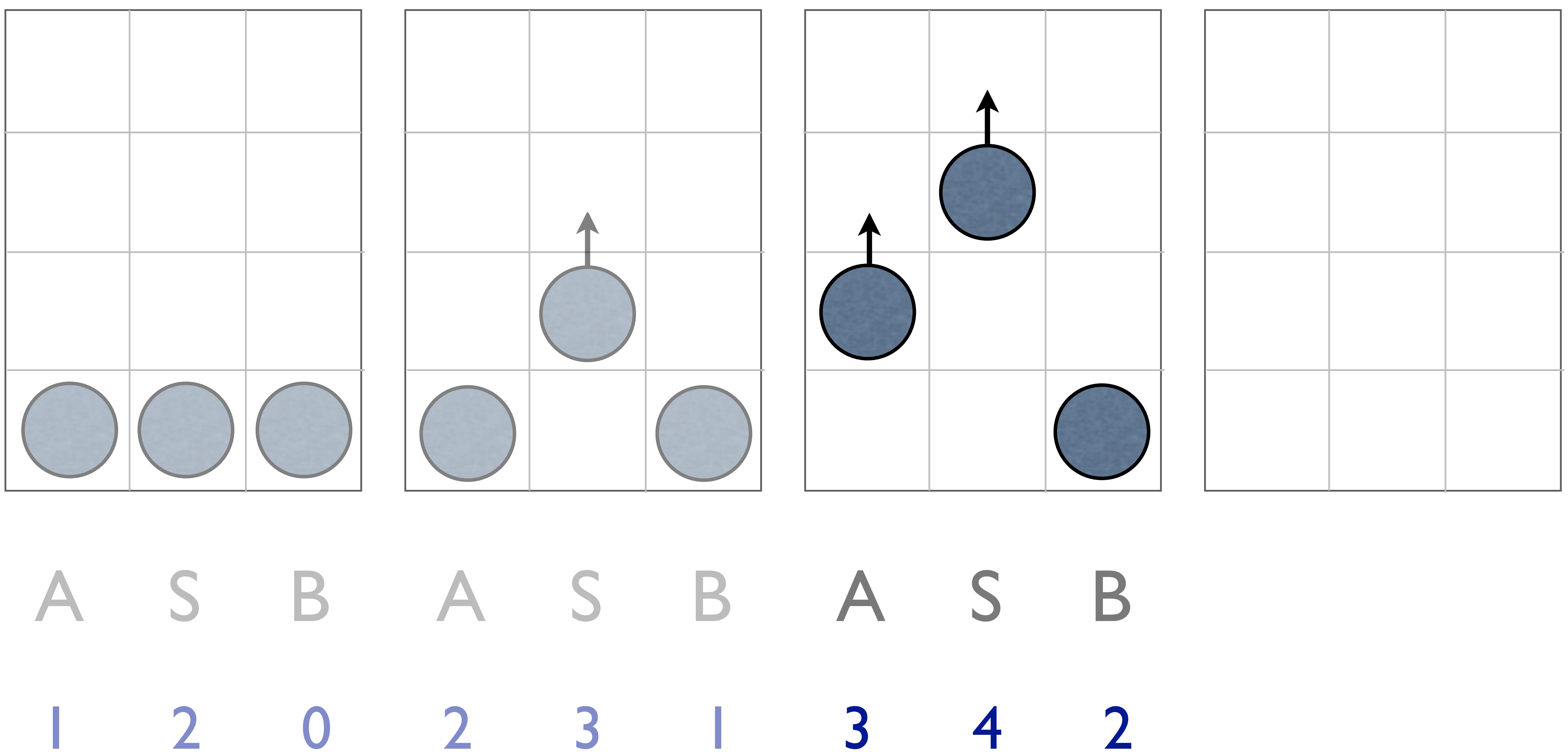
A decides to move. Send event to S.



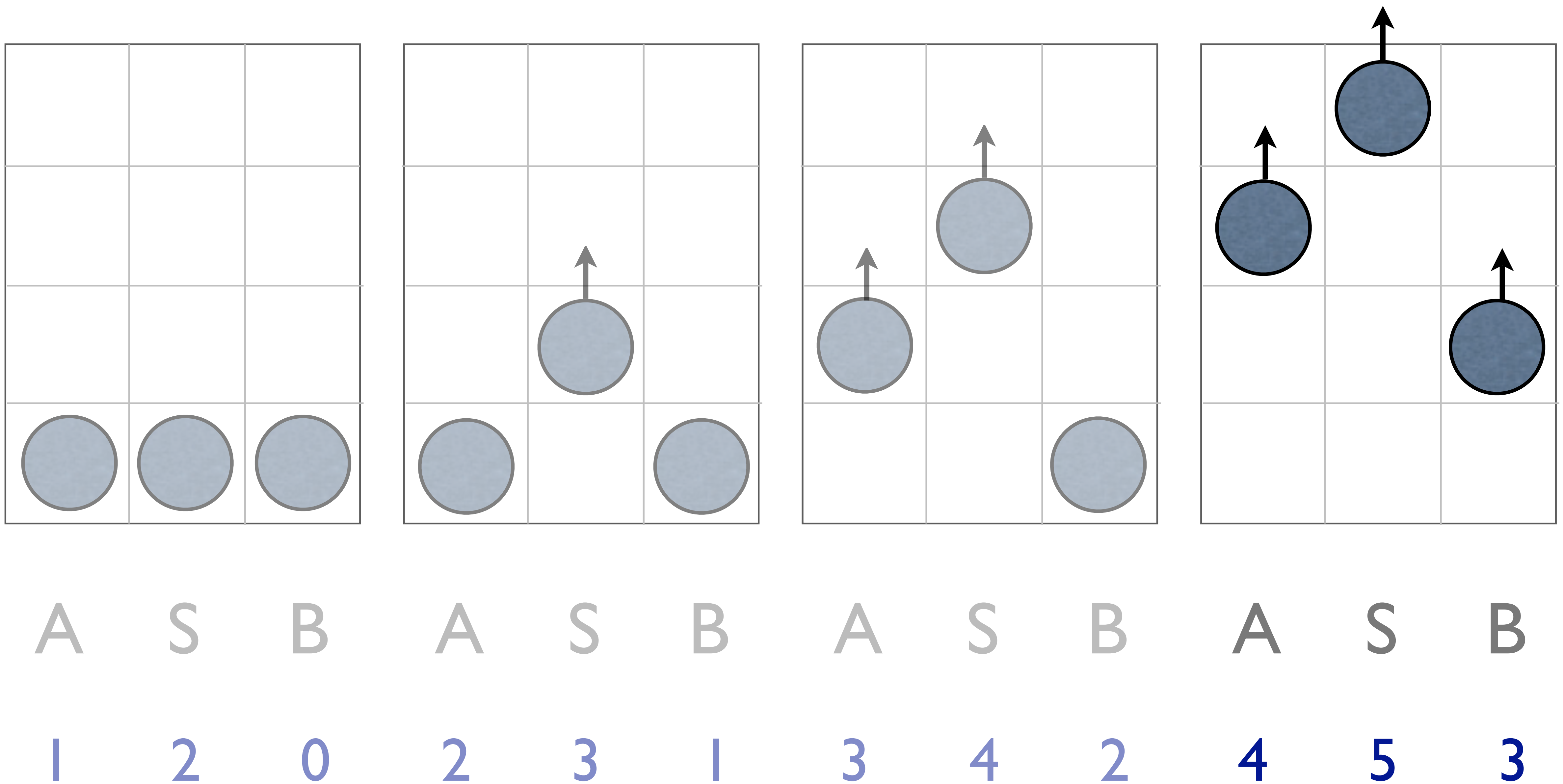
S receives event, moves A, and tells A and B that A is moving at t = 3



A moves itself at t = 3.

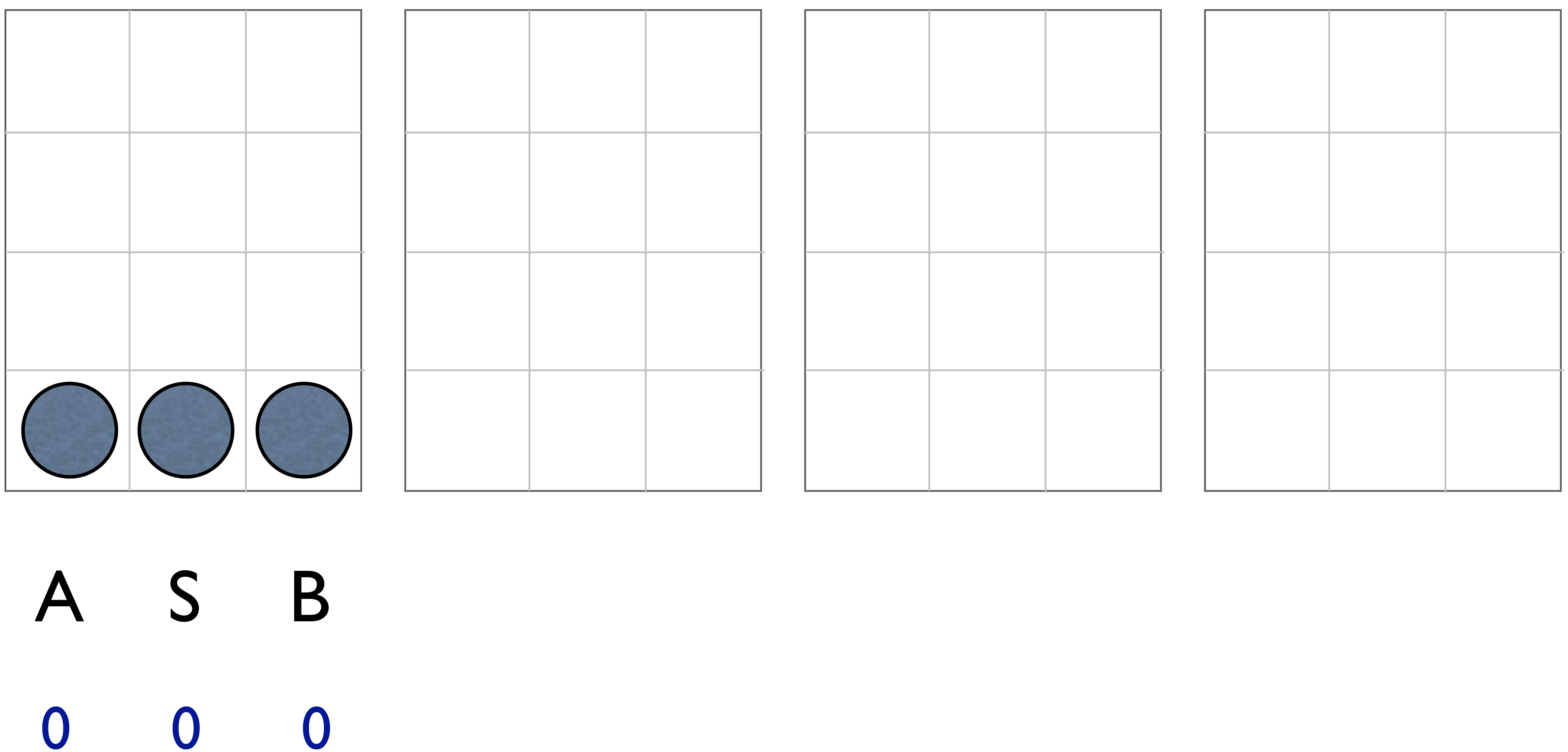


B moves A at t = 3.

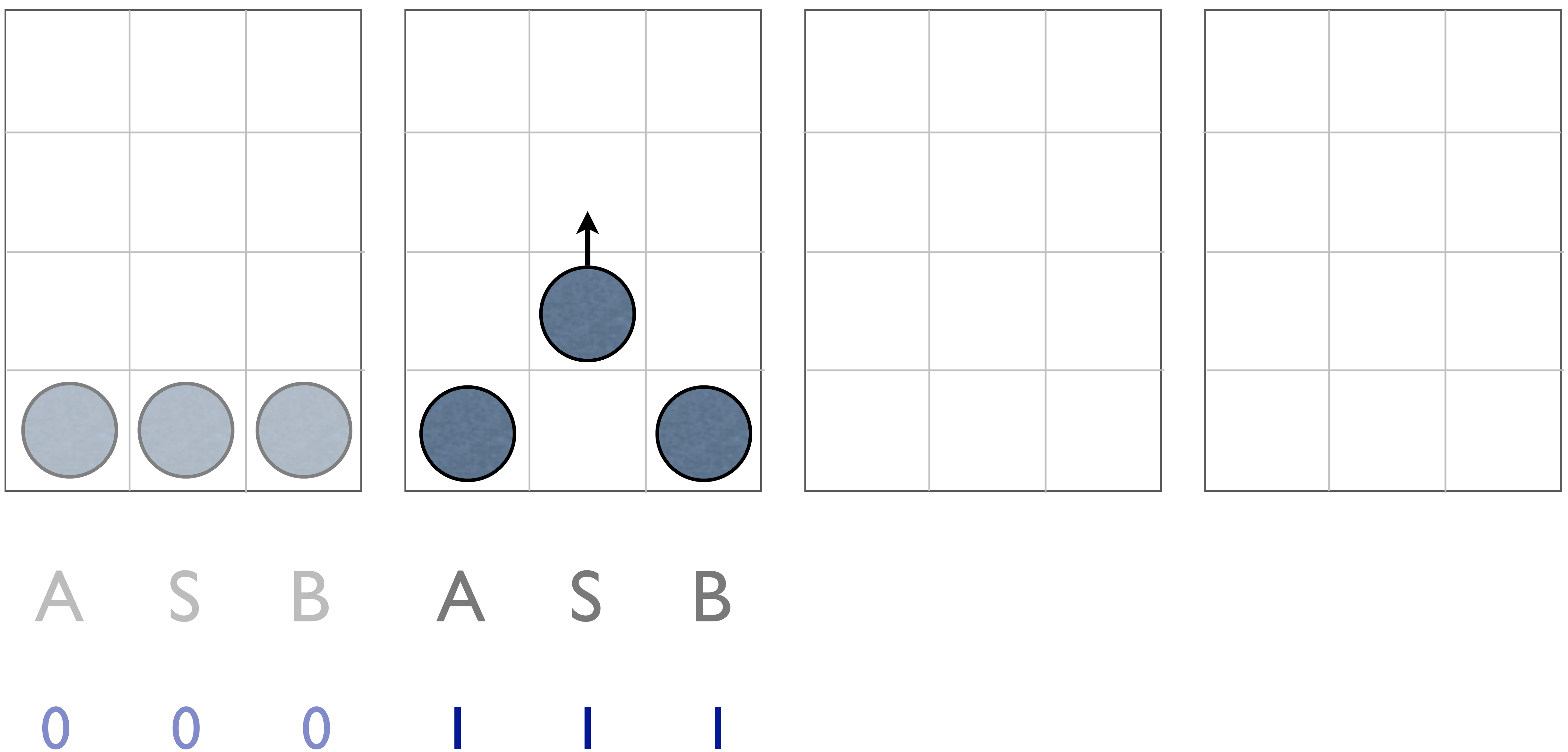


Example: using wallclock

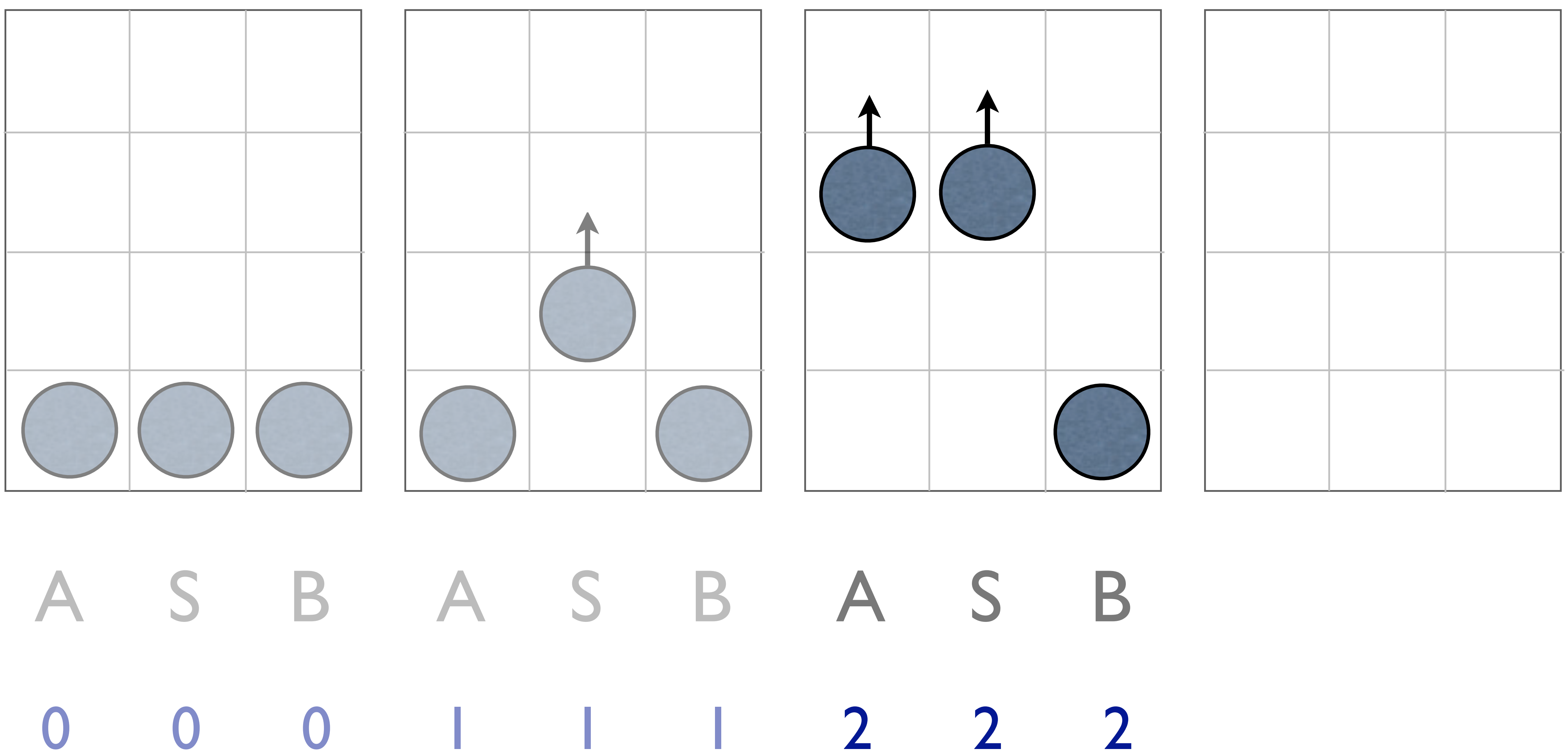
A decides to move. Send event to S.



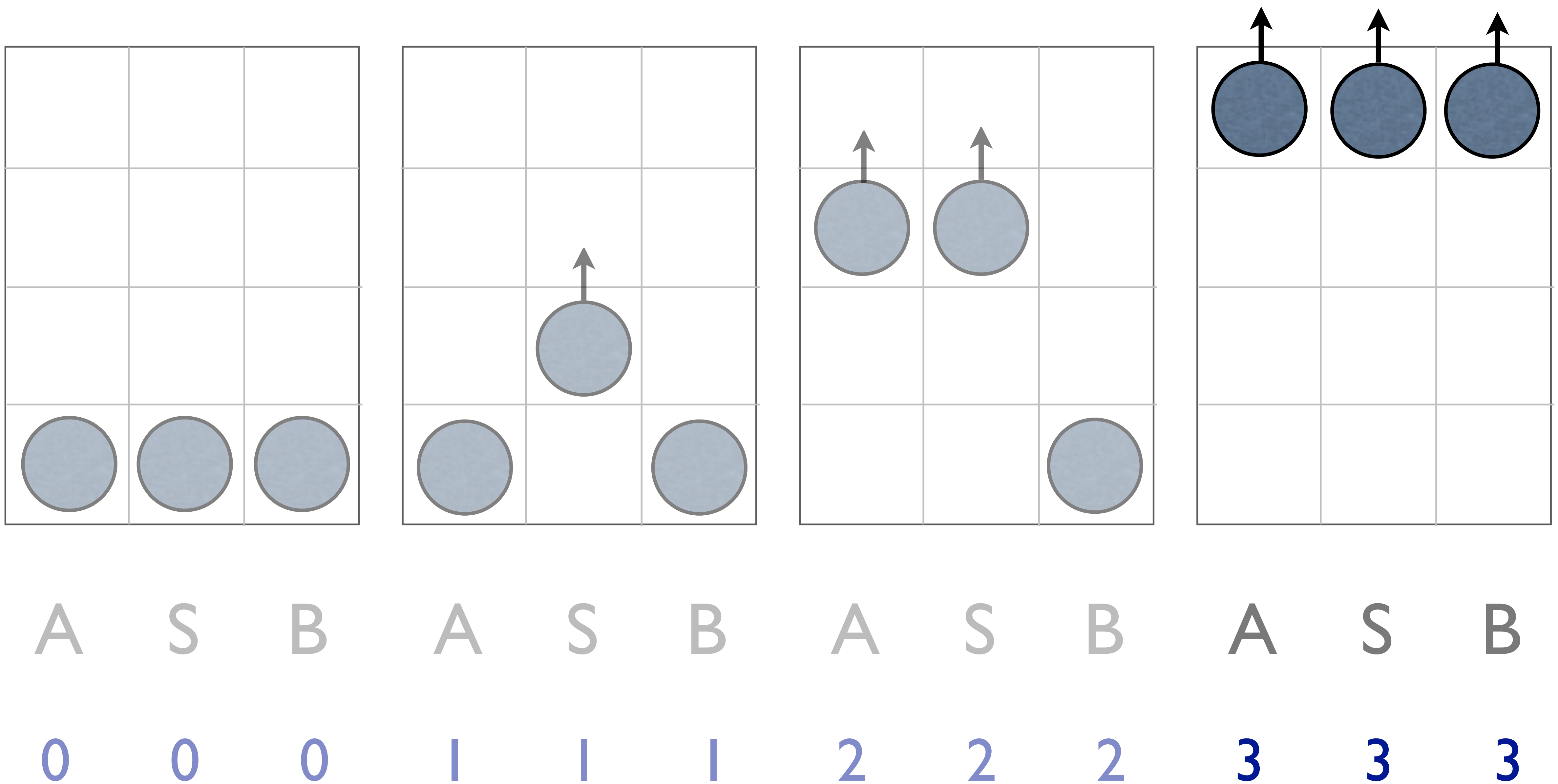
S receives event, moves A, and tells A and B that A is moving at t = 1



A moves itself to where it should be at t = 2.



B finds out A moves at t = 1 and moves A to where A should be at t = 3



Tight synchronization allows interaction but can lead to visual disruptions.

Asynchronization allows smooth movement but hinder interaction.

Local Perception Filter

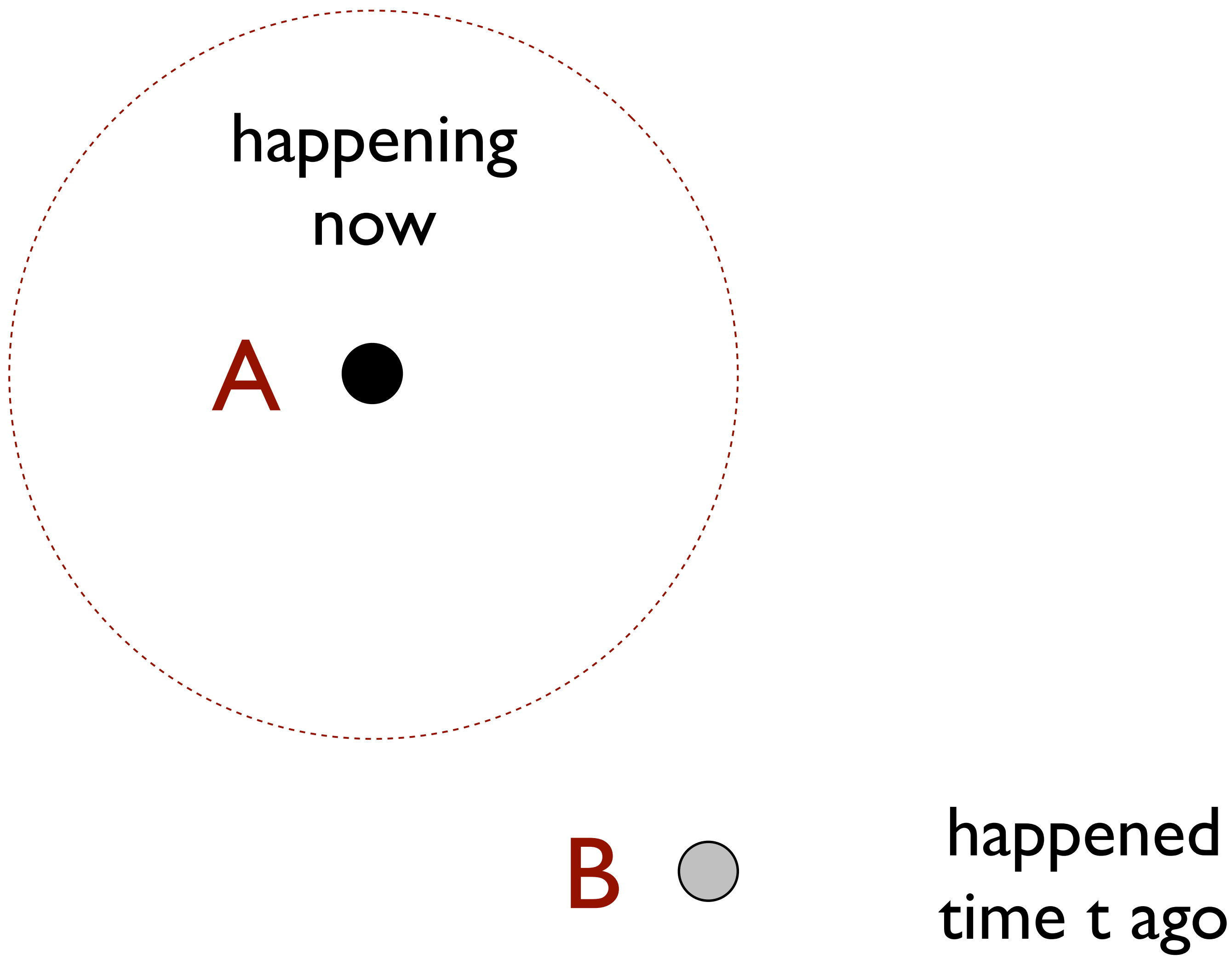
Hybrid Model:

Render objects within real-time interaction range in real time, other objects in delayed time.

Two Kinds of Entities

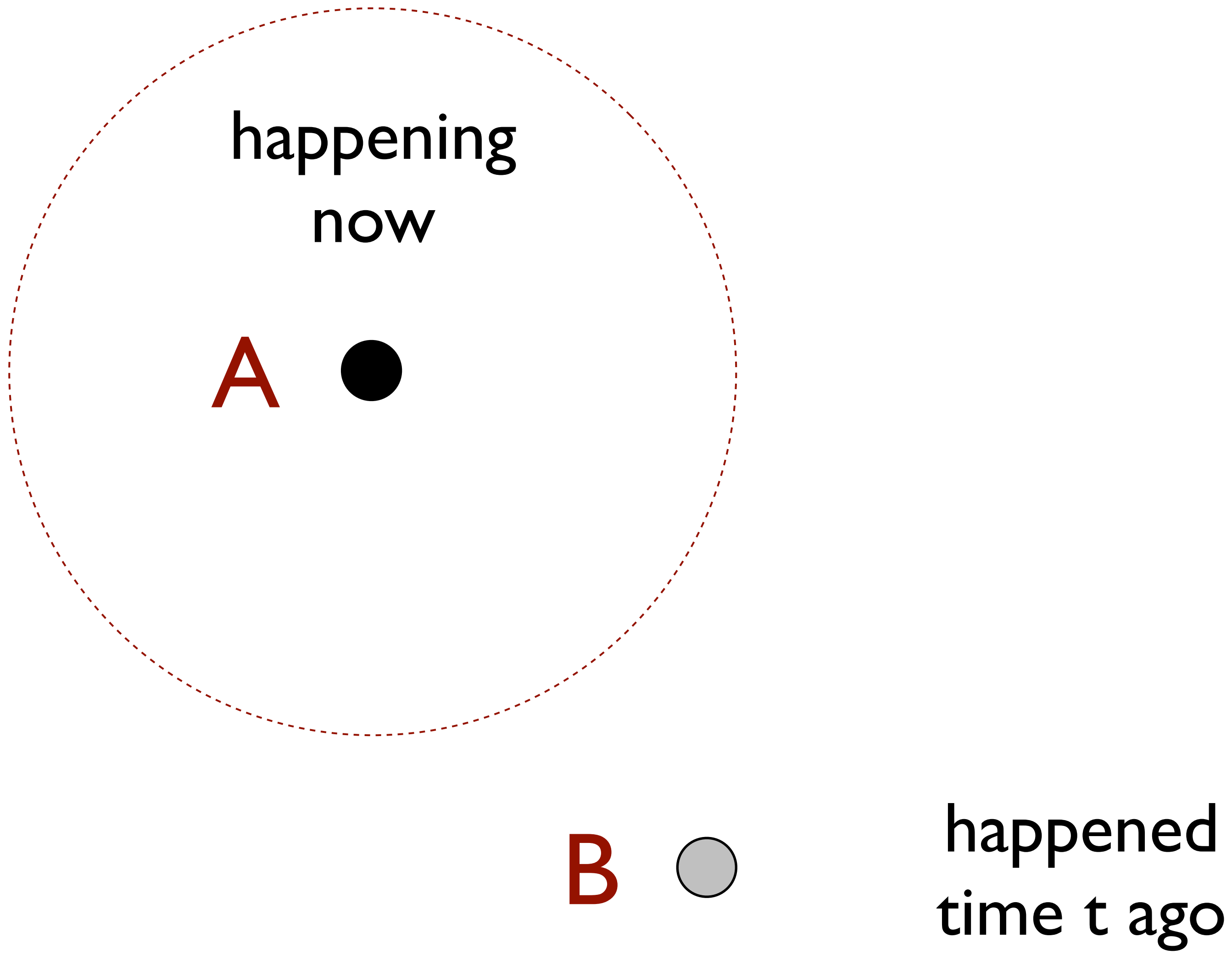
Active:
players (unpredictable)

Passive:
ball, bullet (predictable)


happening now
A
B
happened time t ago

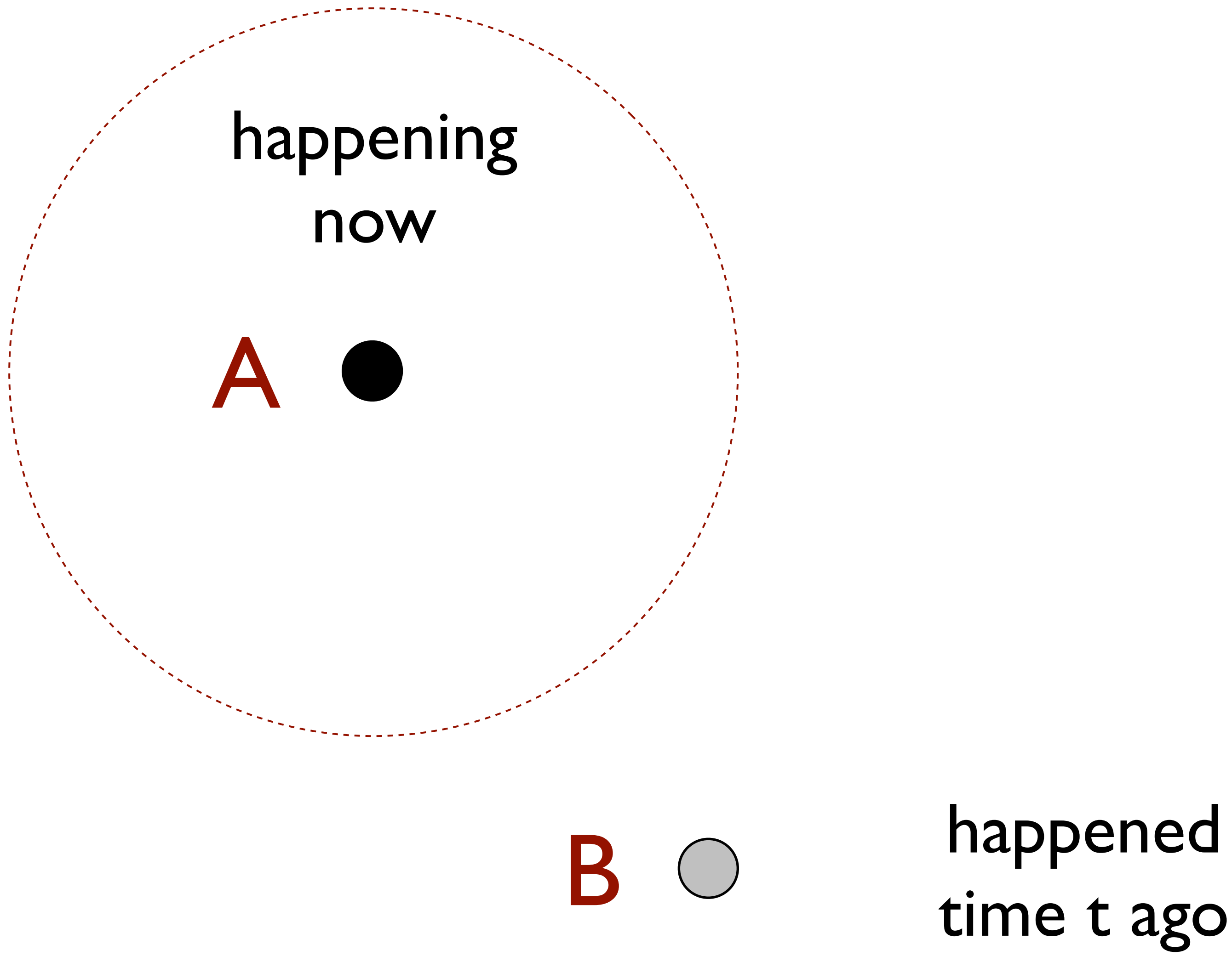
Question:

What if a player A throws a ball at player B?

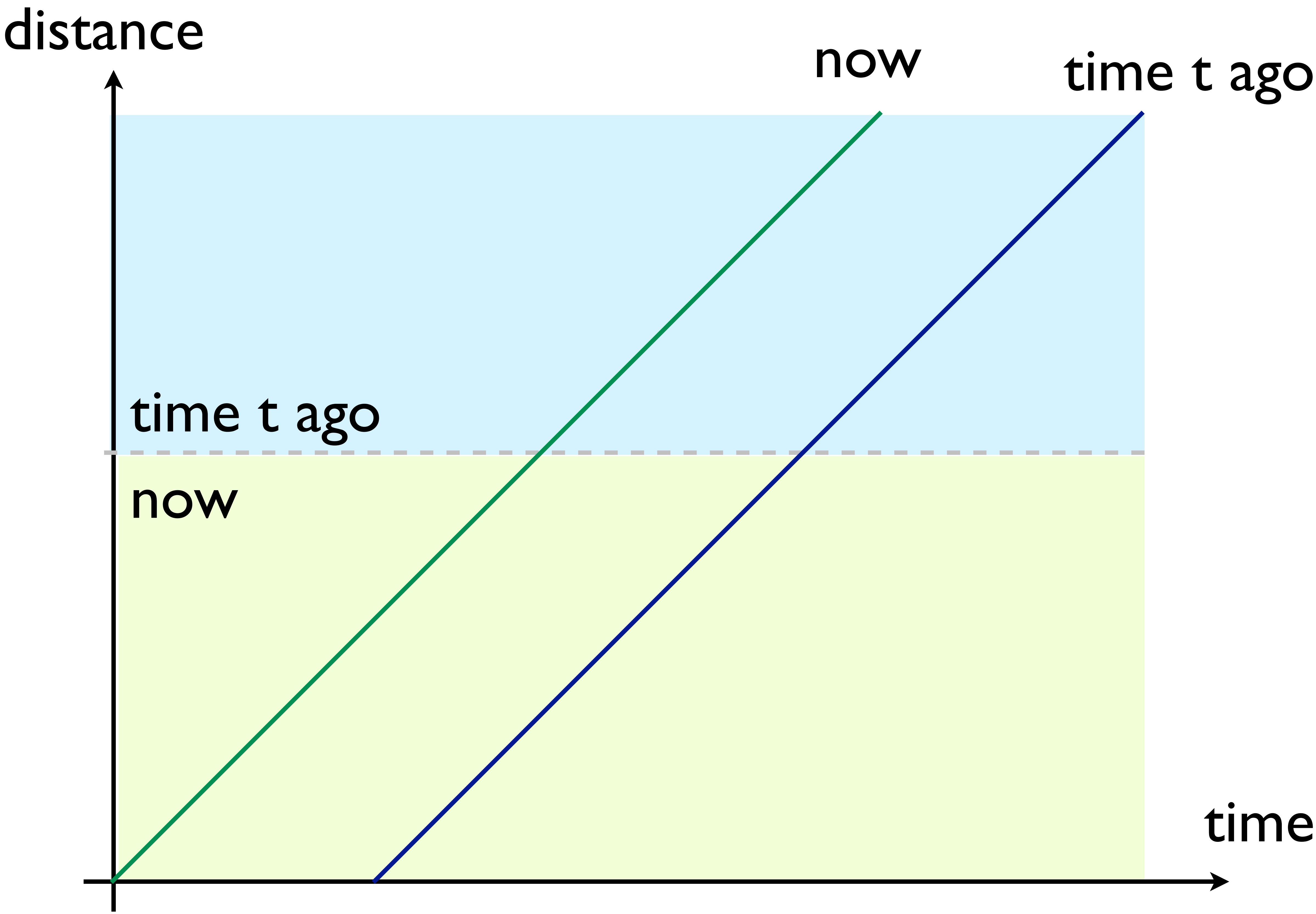

happening
now
A
B
happened
time t ago

Question:

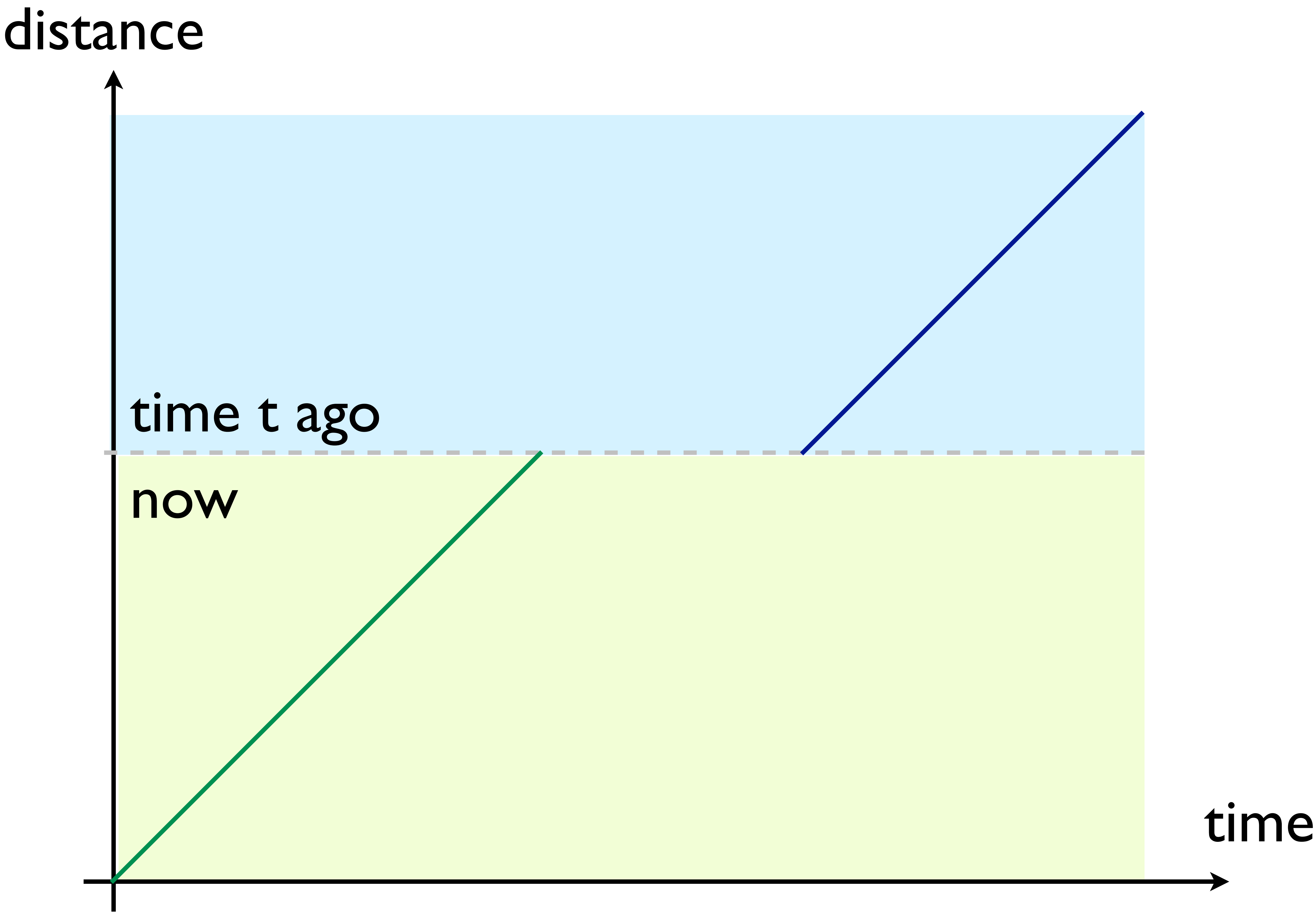
What if a player B throws a ball at player A?


happening
now
A
B
happened
time t ago

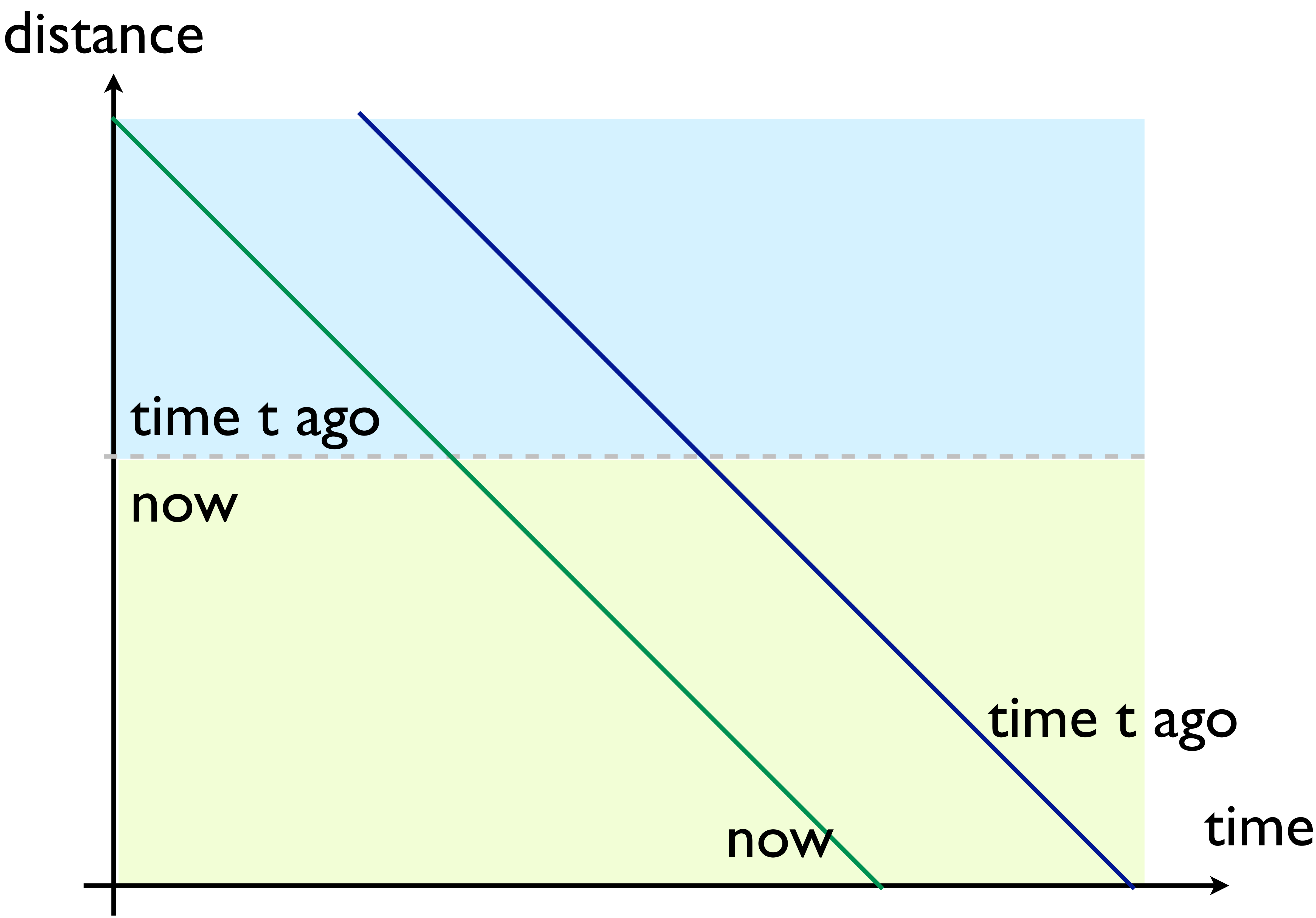
Distance of ball (thrown by A) from A versus time.



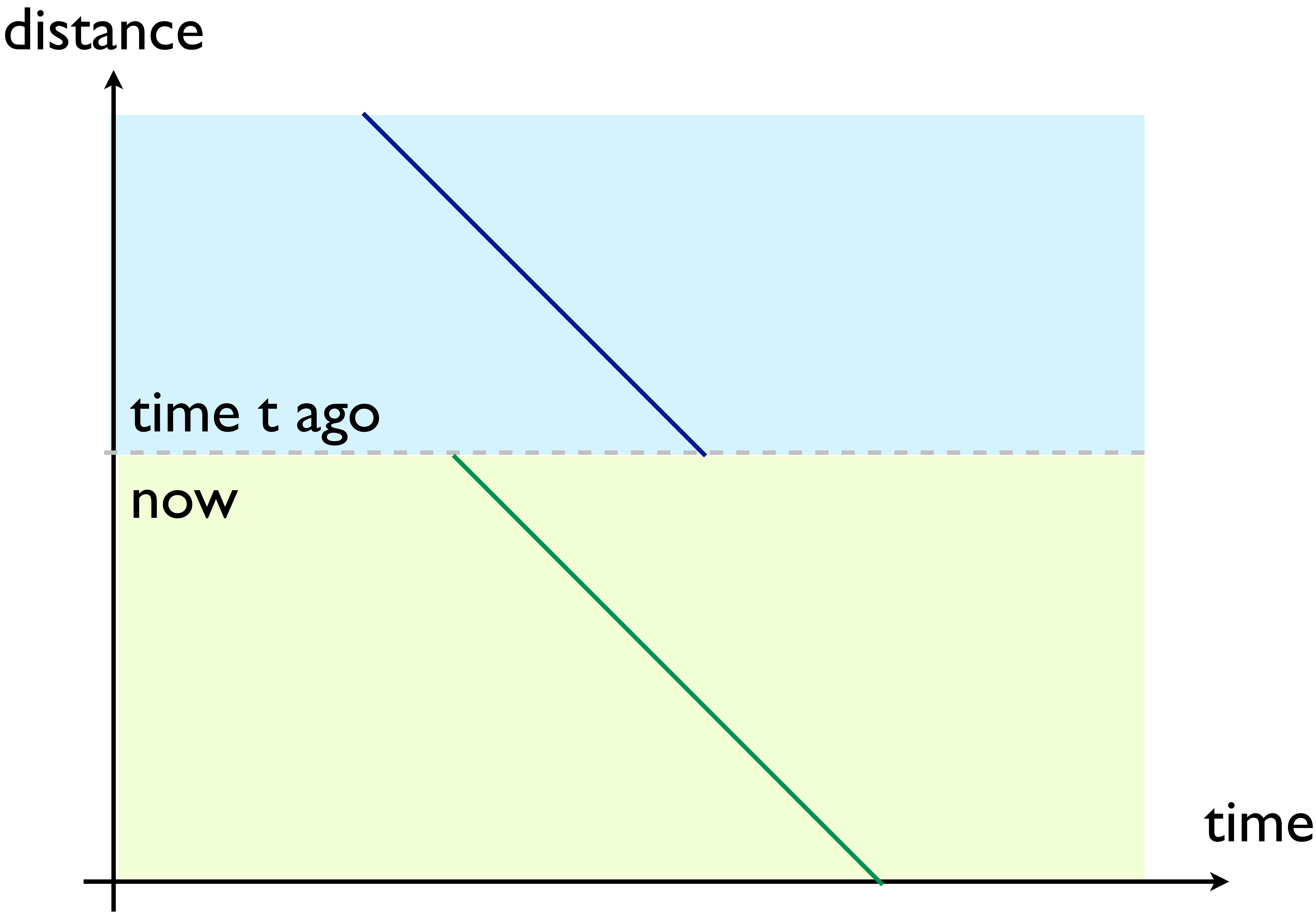
What A sees..

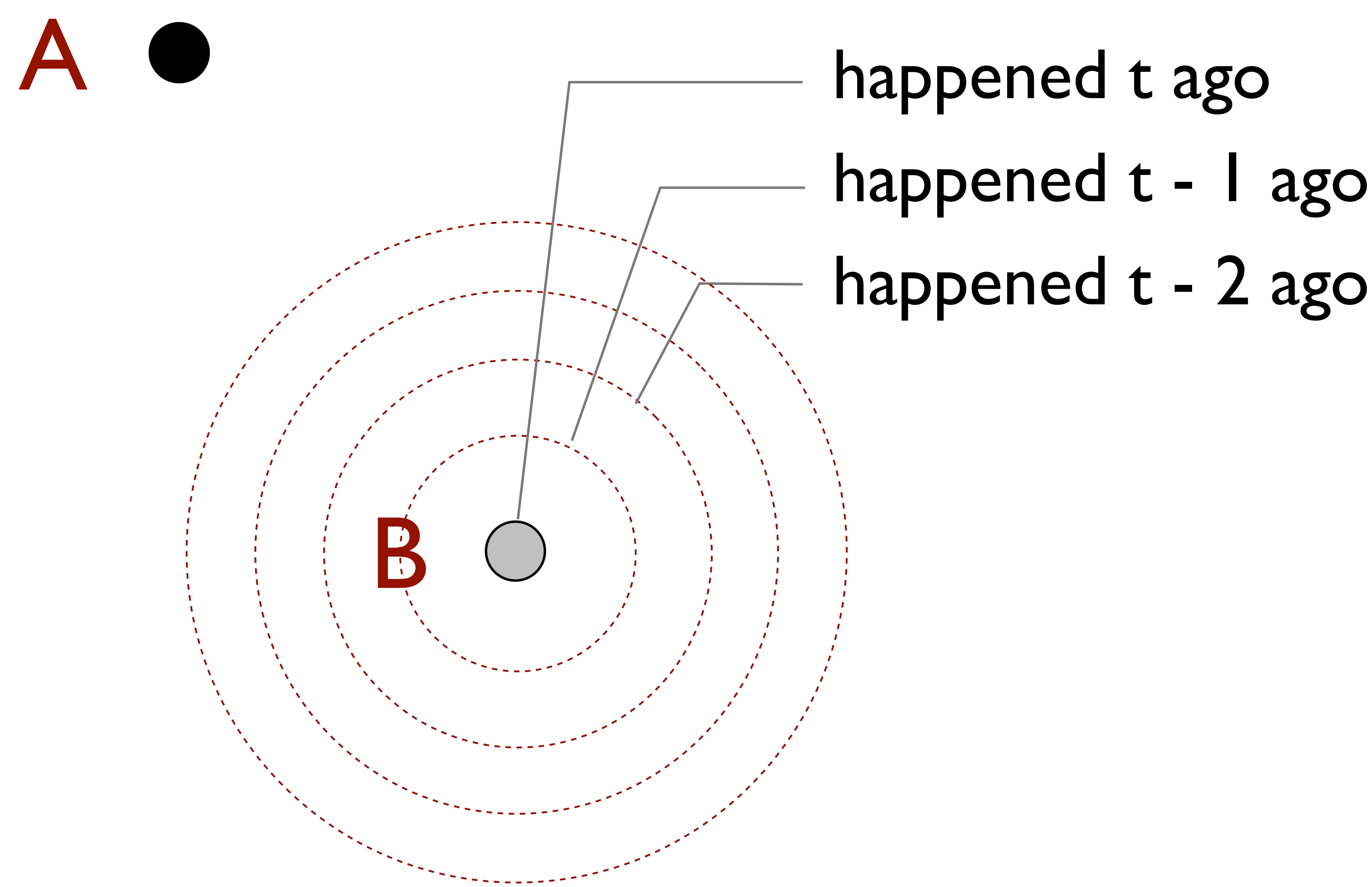


Distance of ball (thrown by B) from A versus time.



Distance of ball (thrown by B) from A versus time.

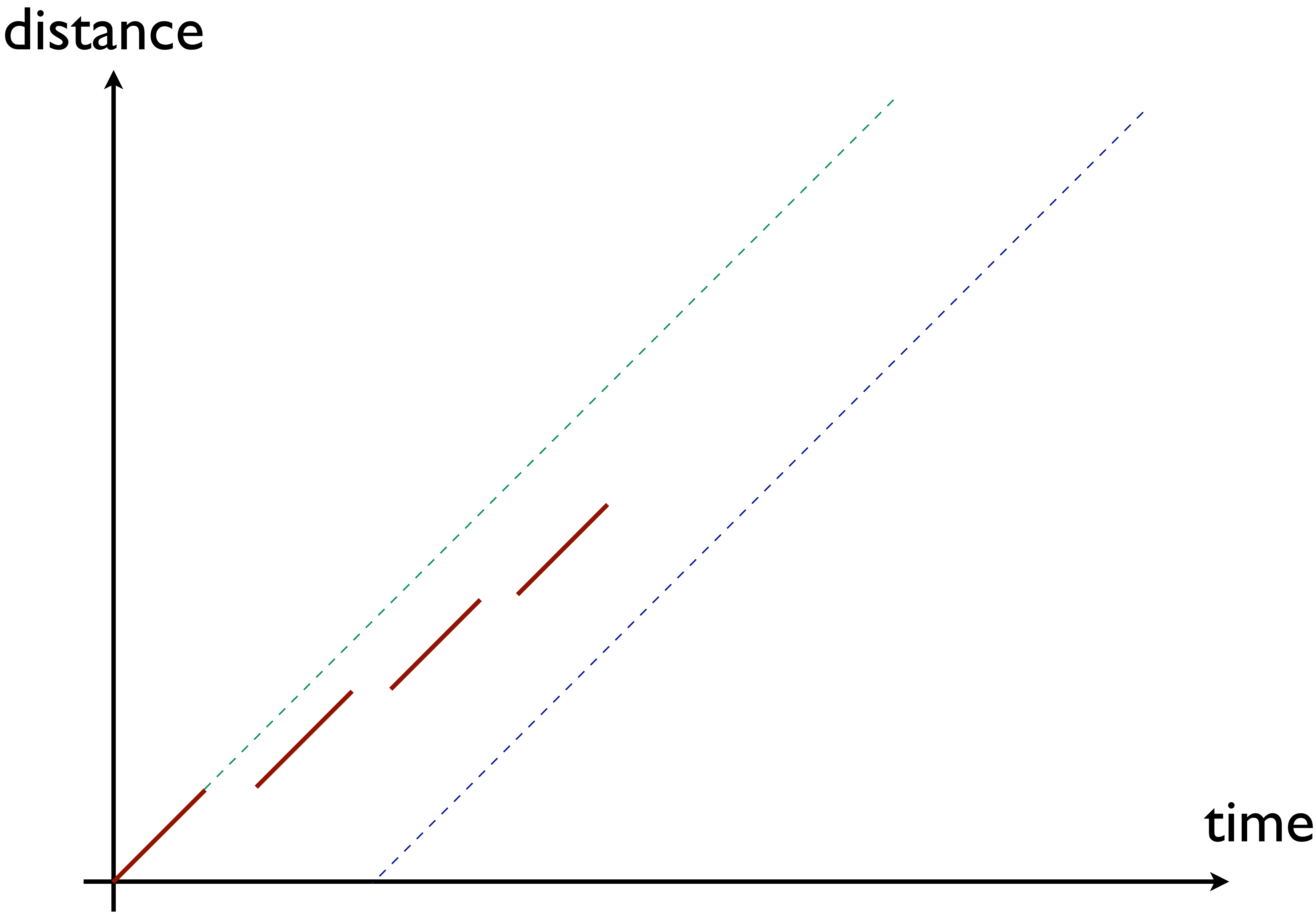



A
B
happened t ago
happened t - 1 ago
happened t - 2 ago

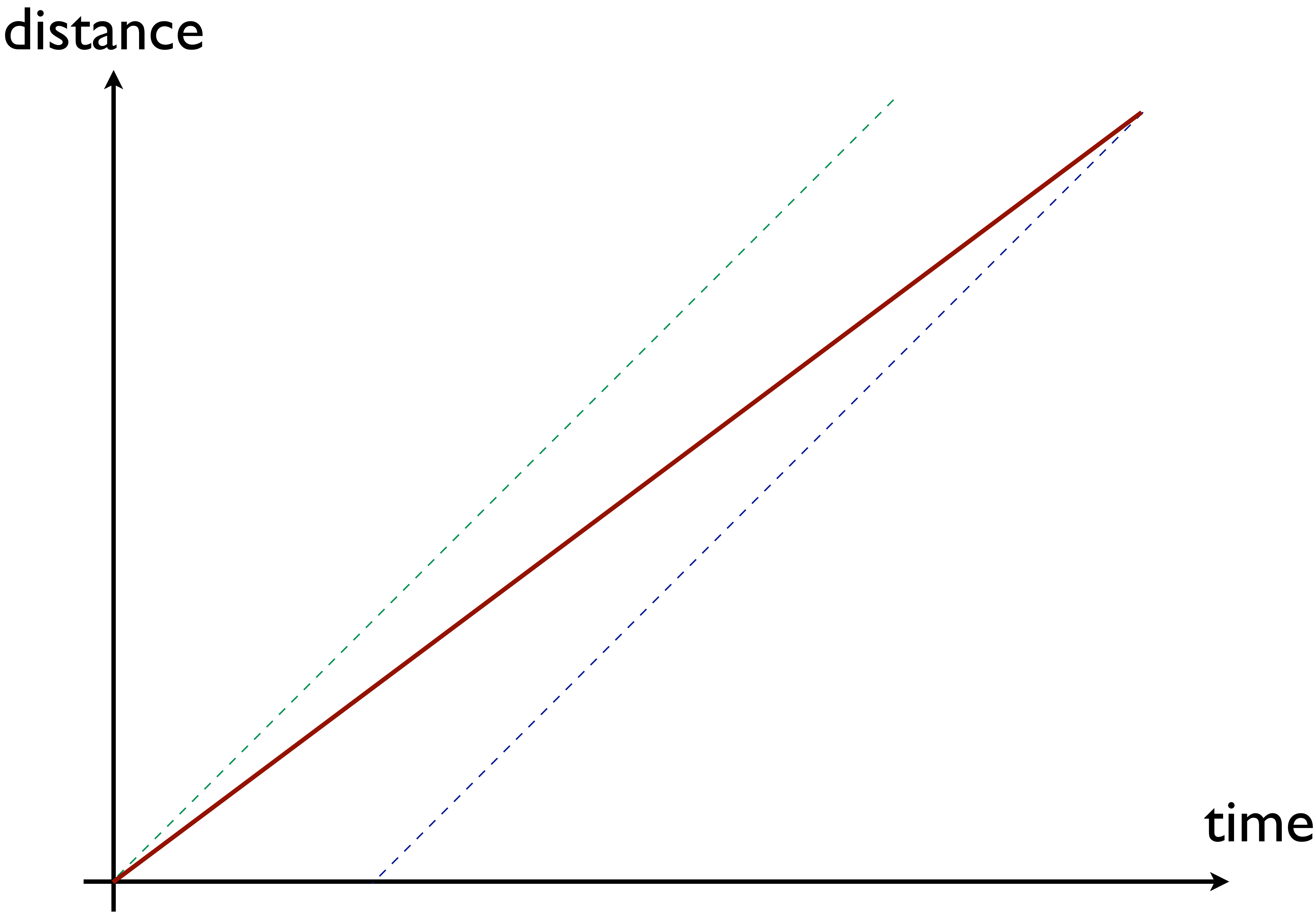
Question:

What if a player A throws a ball at player B?

What A sees..



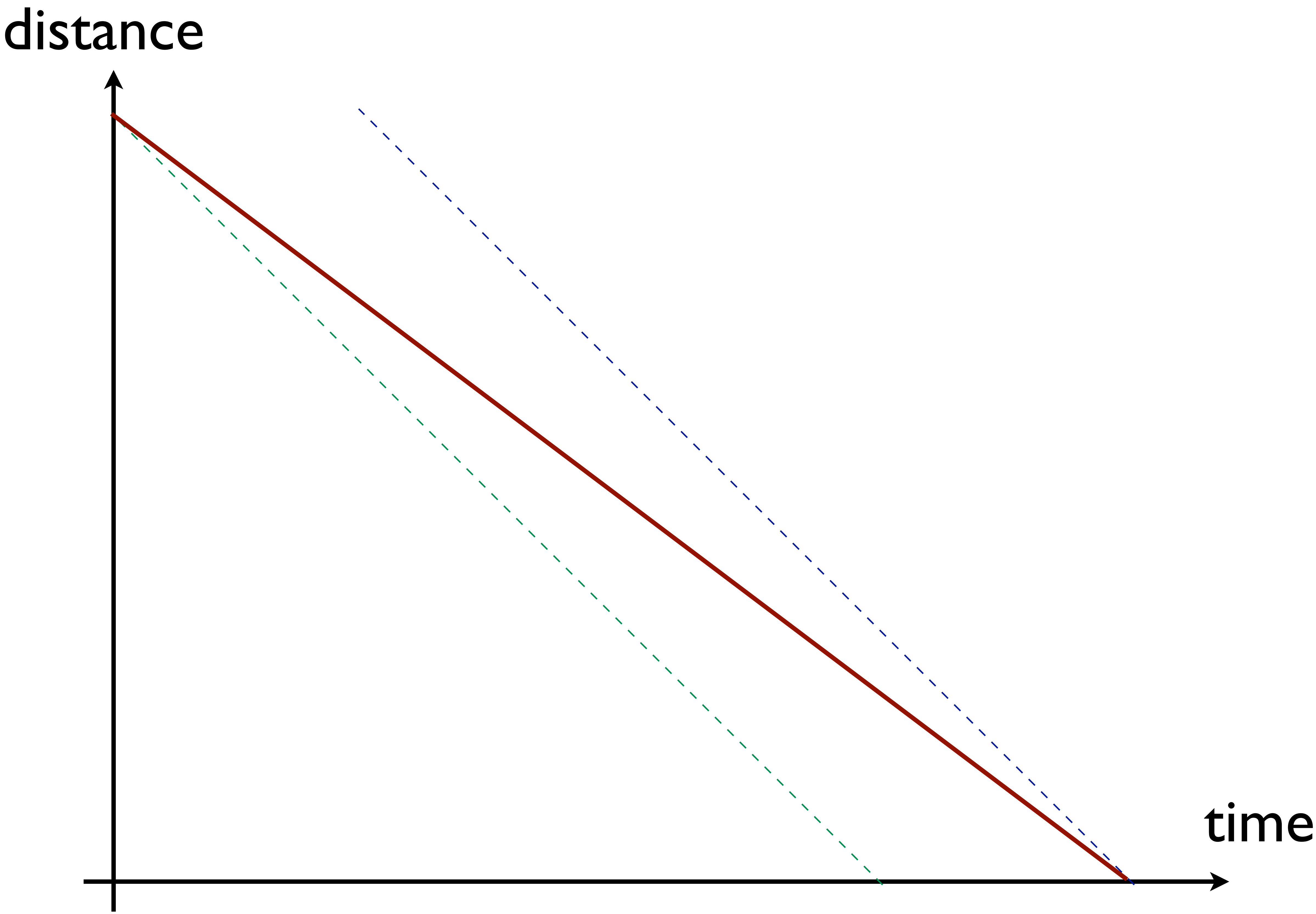
What A sees..

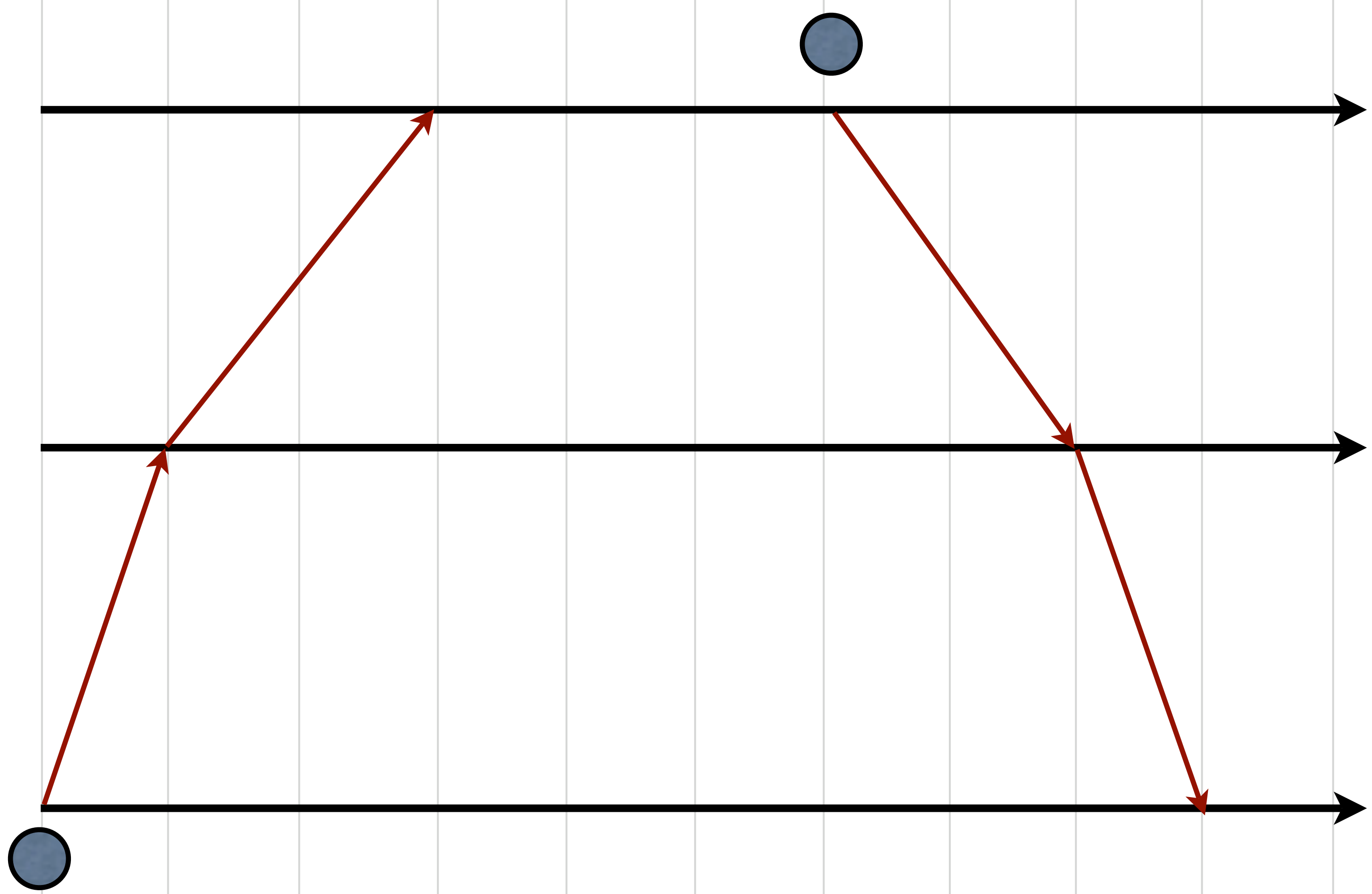


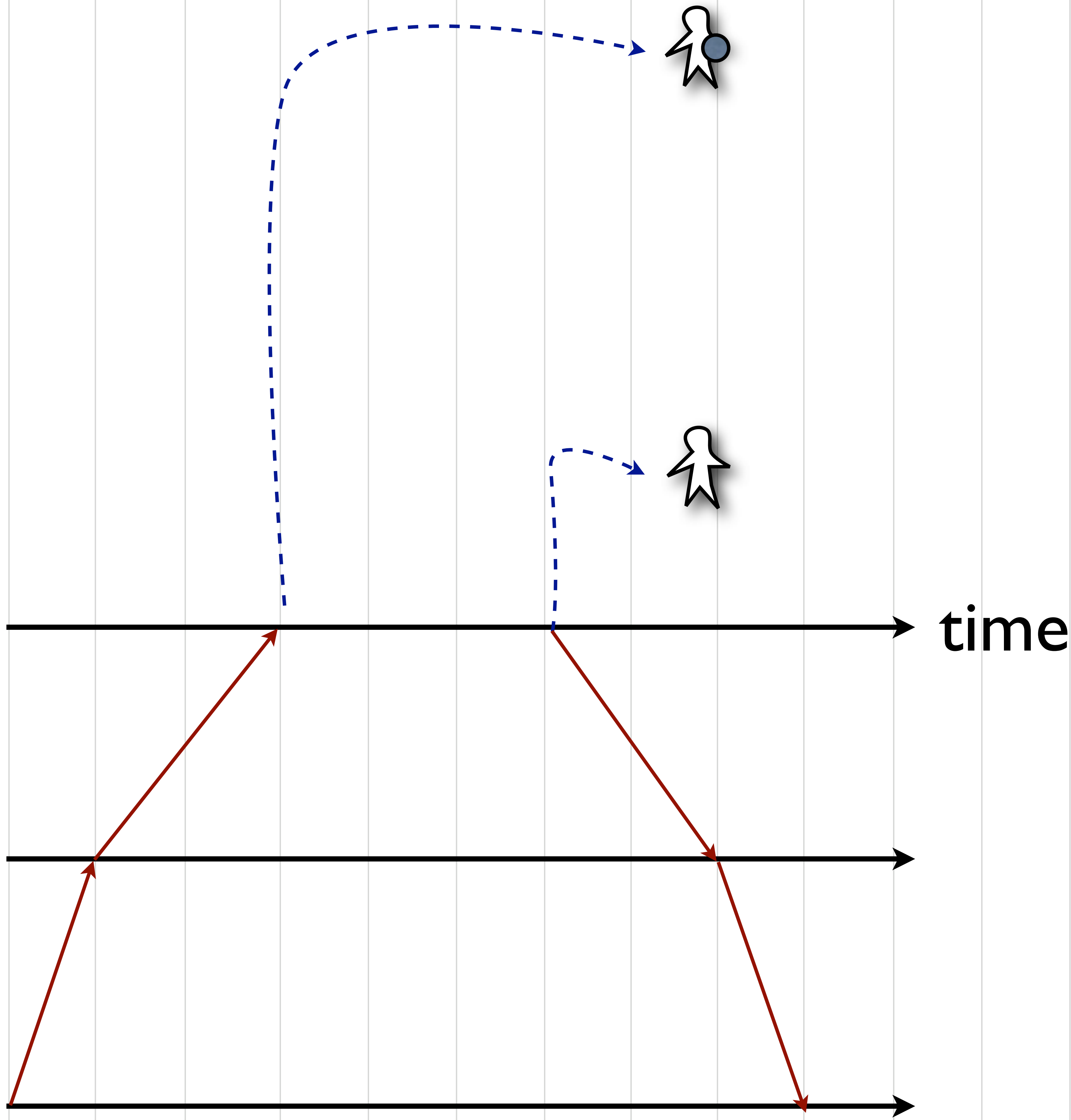
Question:

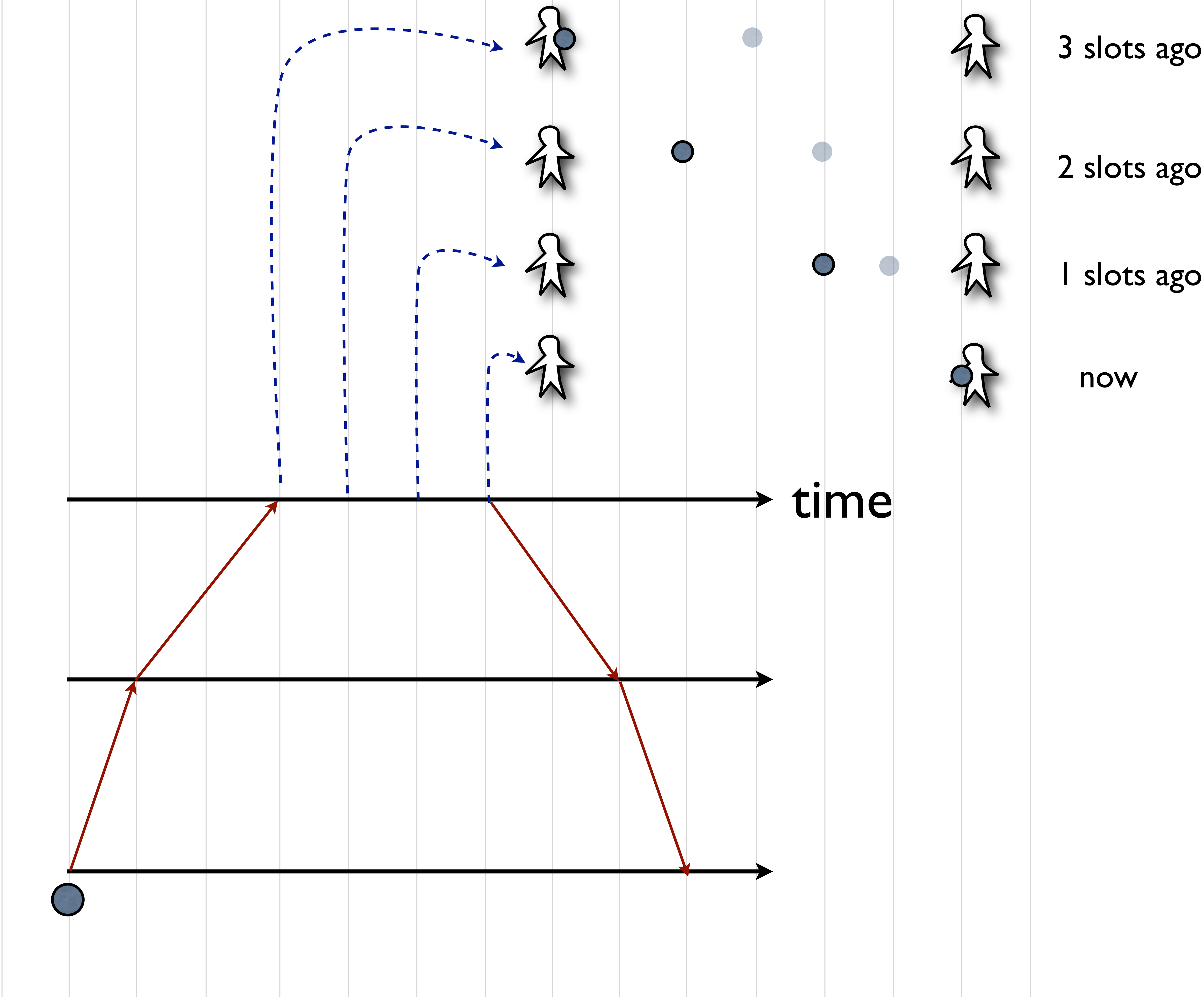
What if a player B throws a ball at player A?

Distance of ball (thrown by B) from A versus time.






time

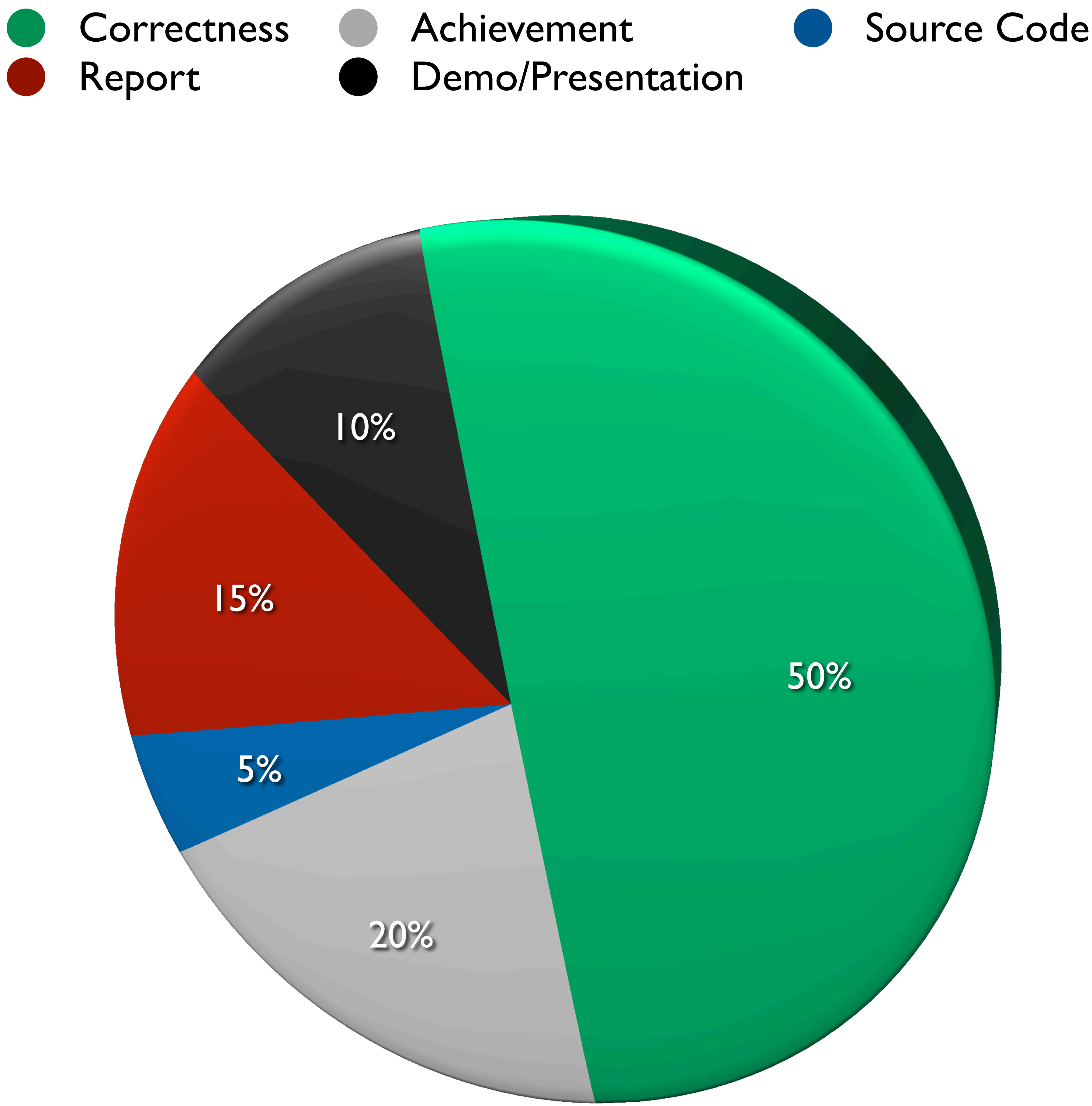

3 slots ago
2 slots ago
1 slots ago
now
time

Local Perception Filter

Assignment 1

15 September, 2008

1-2 students/team


Correctness
Achievement
Source Code
Report
Demo/Presentation
10%
15%
50%
5%
20%