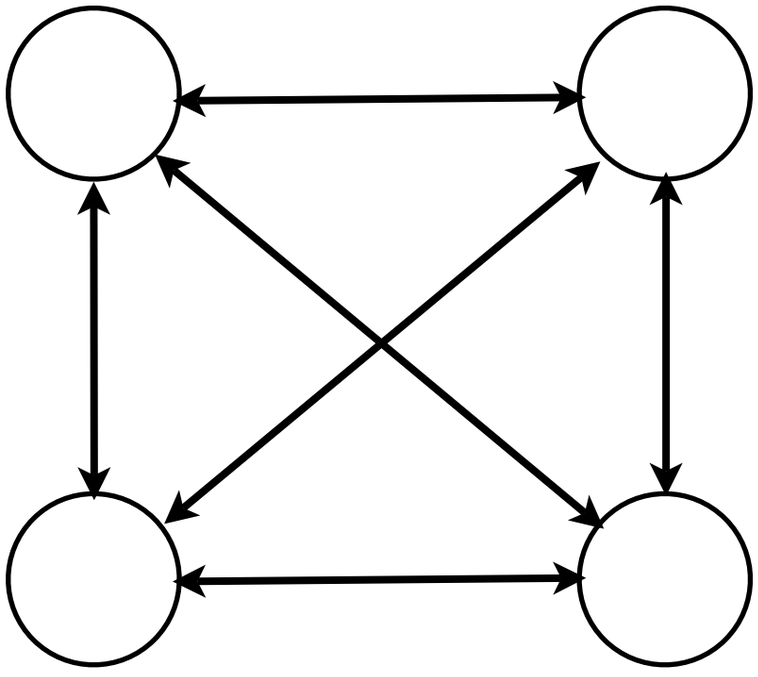
Point-to-Point Architecture

Point-to-Point Architecture



Role of clients

Notify clients

Resolve conflicts

Maintain states

Simulate games

Latency

Robustness

Conflict/Cheating

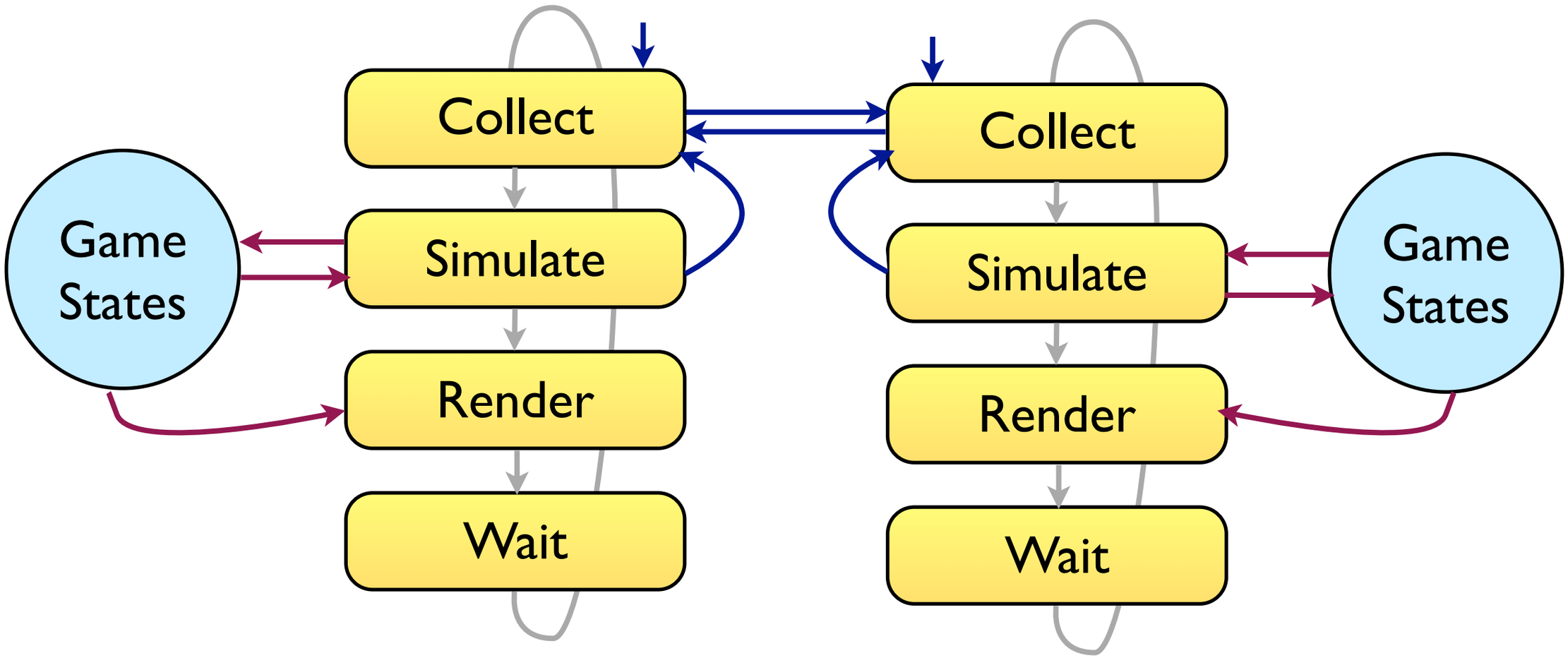
Consistency

Accounting

Scalability

Complexity

Lower Latency
No Single Point of Failure
Low Infrastructure Cost


Collect
Simulate
Render
Wait
Game States
Collect
Simulate
Render
Wait
Game States

AD-HOC MODE

Compete against players on nearby PSP® systems. Make sure your Wi-Fi switch is on and you can communicate with local PSP® systems without an internet connection. You can play together in a house, a backyard, an airport, a lobby - anywhere! Depending on the game, up to 16 PSP® systems within range of each other can be connected using Ad Hoc mode.

For local offline multiplayer gaming (Ad Hoc mode) you need:

- multiple players on PSP® systems within range of each other
- each with a PSP® set up with Ad Hoc on "automatic" or the same channel
- each with a PSP® game that supports multiplayer Ad Hoc mode

INFRASTRUCTURE MODE

Play online with people across the globe, all from the comfort of a Wi-Fi hotspot. Just check the back of the game packaging to see if it supports infrastructure mode, connect to a hotspot, insert the game UMD™, follow the on-screen instructions and start playing online!

For online multiplayer gaming (Infrastructure Mode) you need:

- a Wi-Fi internet hotspot or wireless home network
- a PSP connected to that hotspot or network
- a PSP® game that supports multiplayer Infrastructure mode

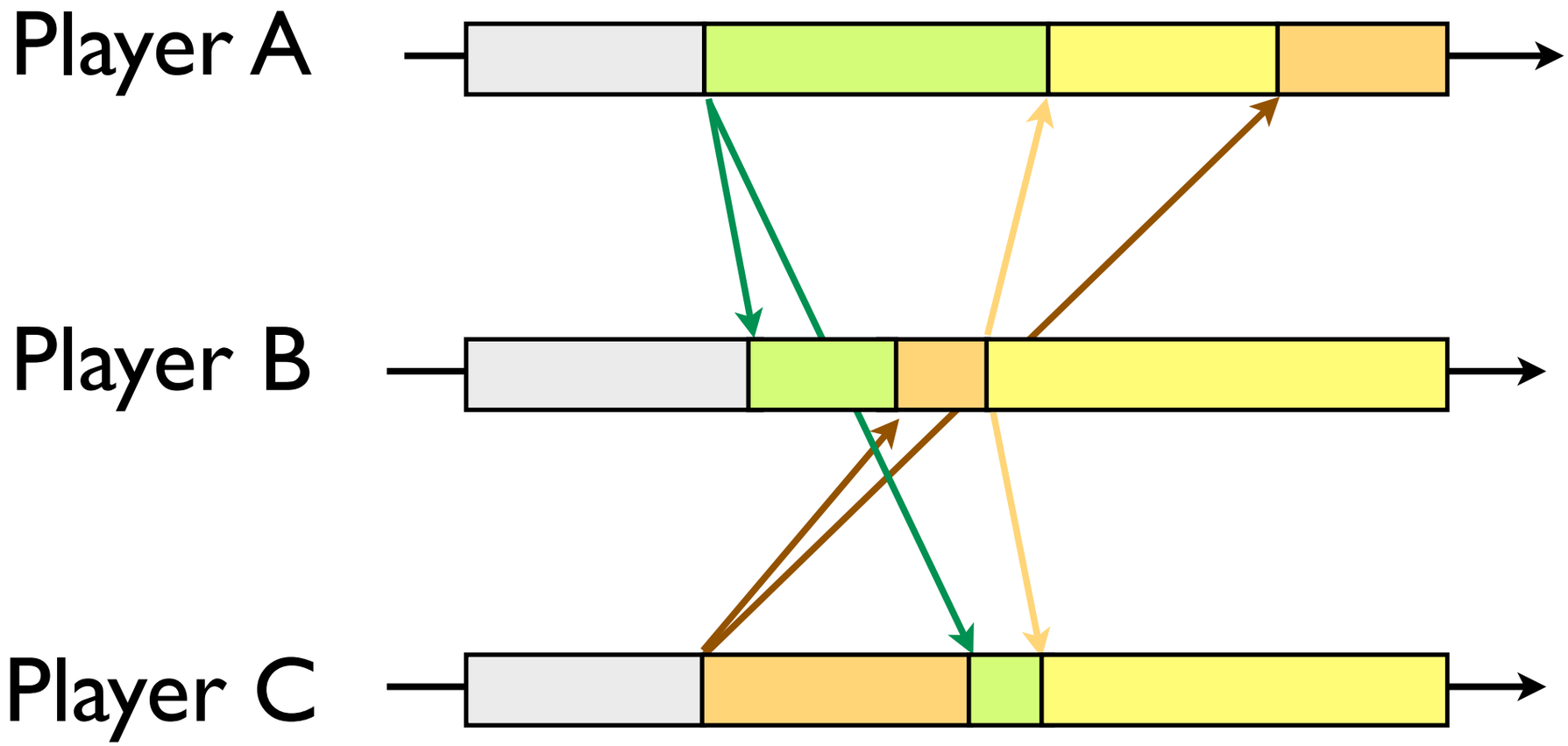
Infrastructure Multiplayer Games

MiMaze from INRIA, France

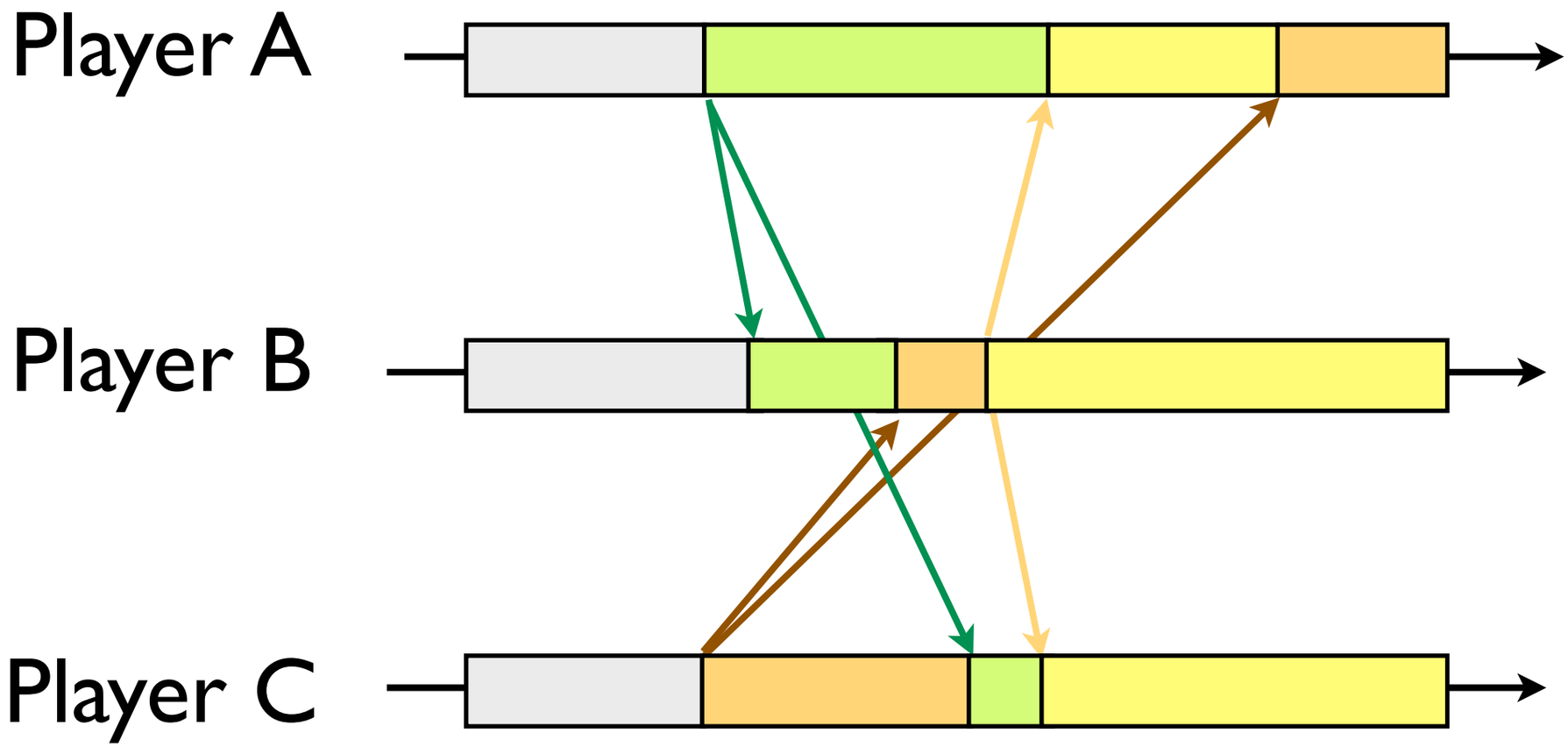
Age of Empire Series

http://compactiongames.about.com/library/games/screenshots/blscreens-ageofkings.htm

How to order events without a server? We can't use received-order delivery anymore.



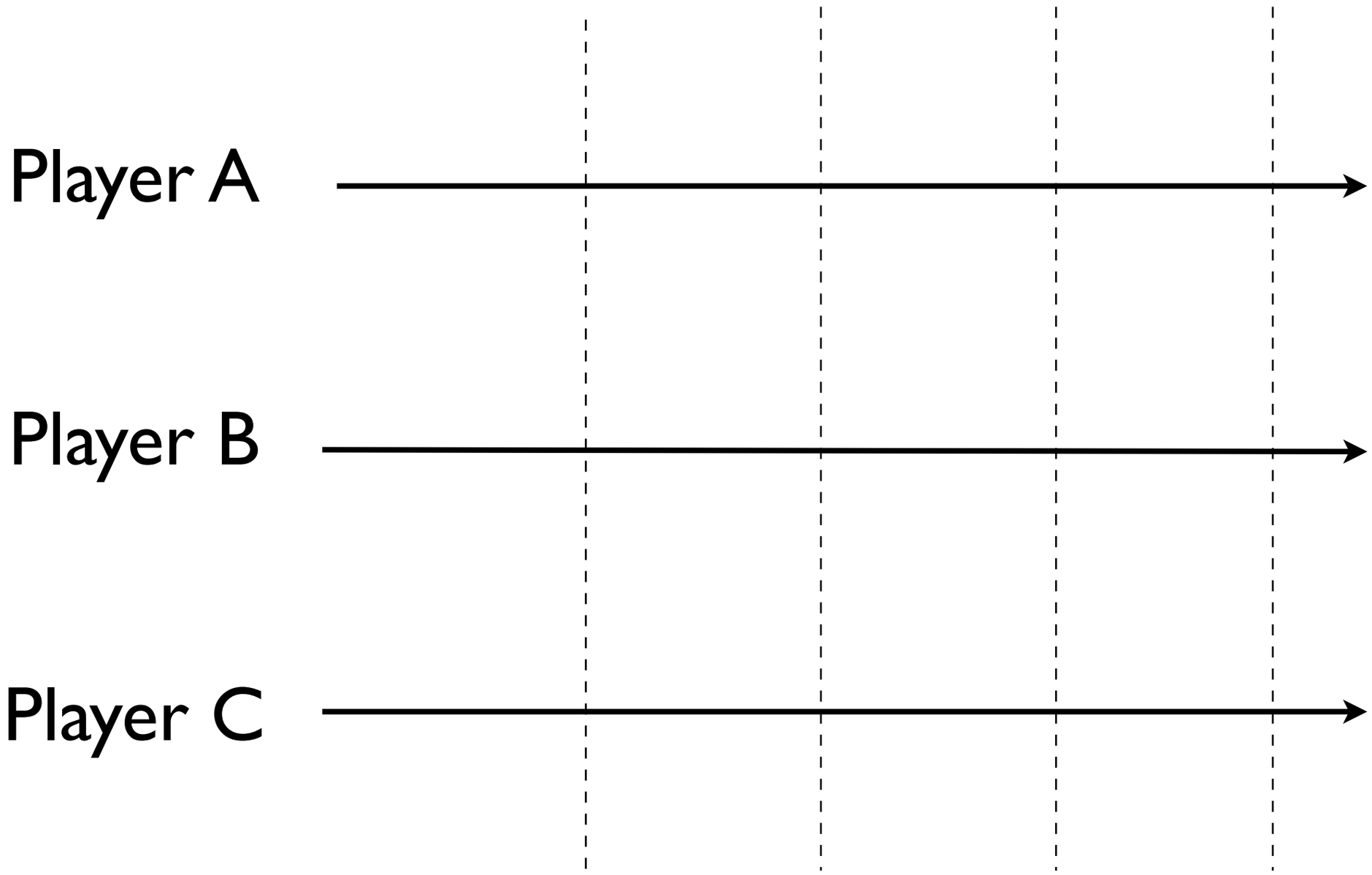
We can timestamp the events and execute in time order, but how do we know if there is another message with an earlier timestamp on the way?



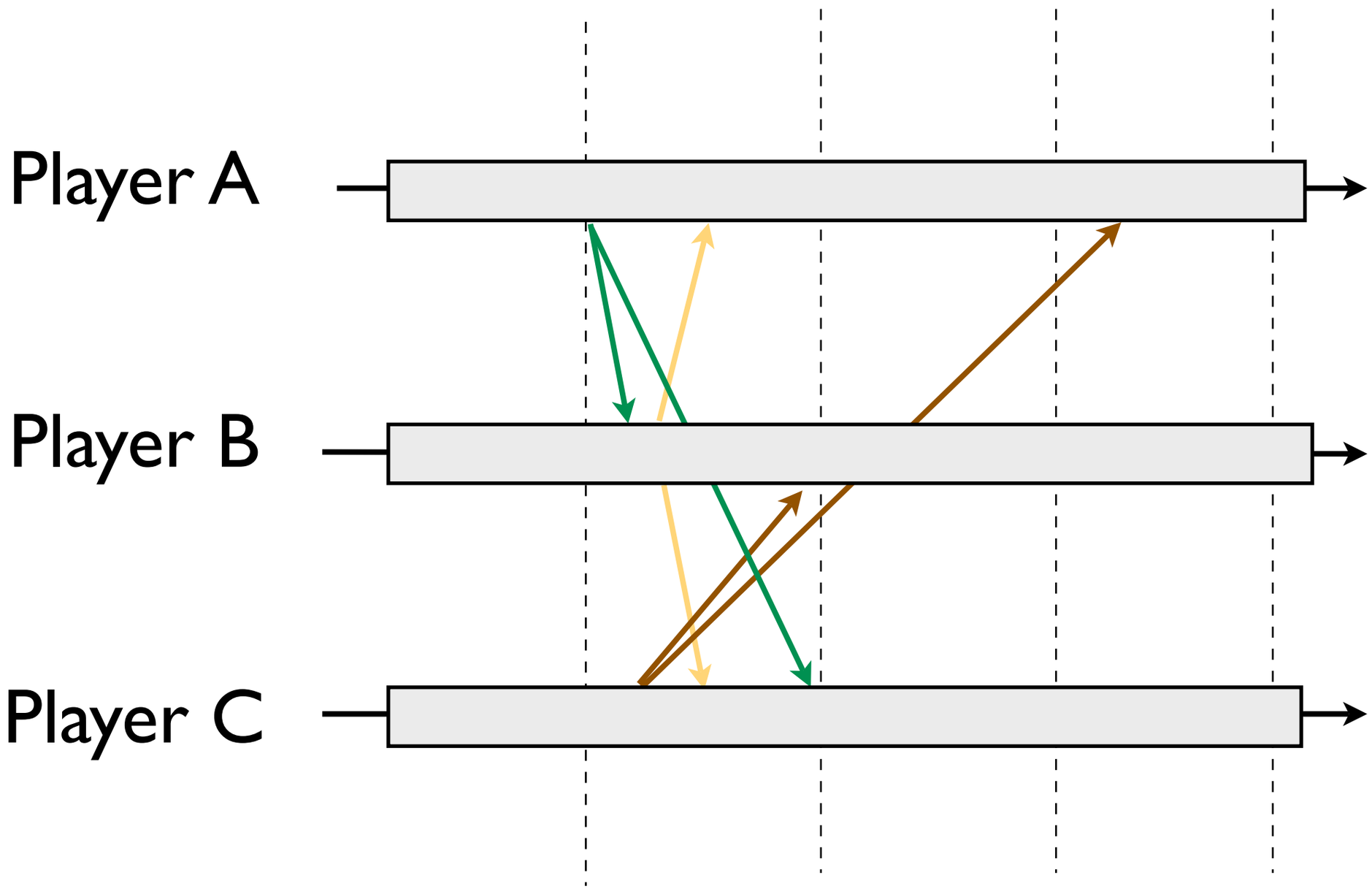
Force everyone to send an event in every round

Bucket Synchronization

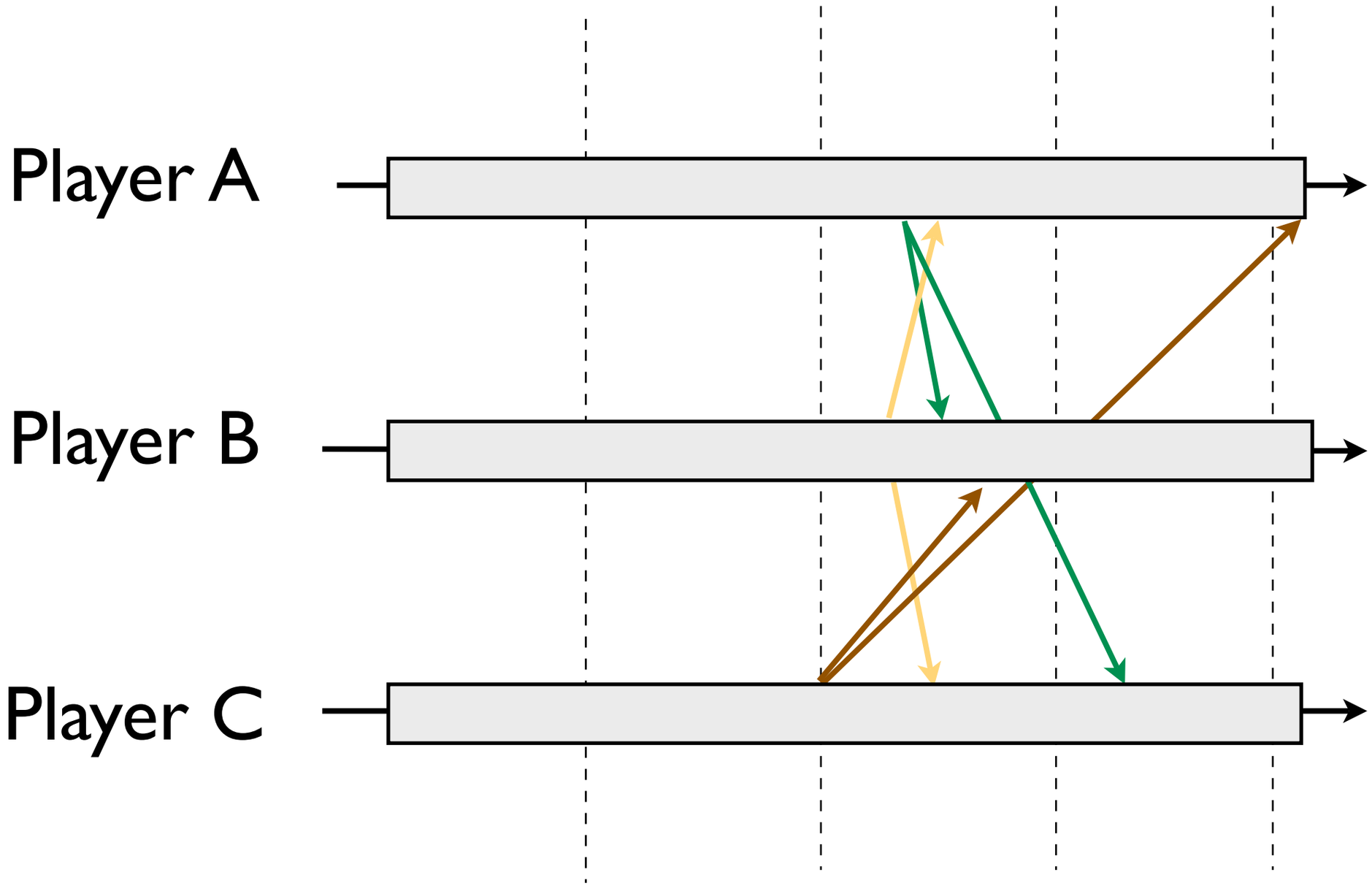
The game is divided into rounds (e.g. 25 rounds per second).



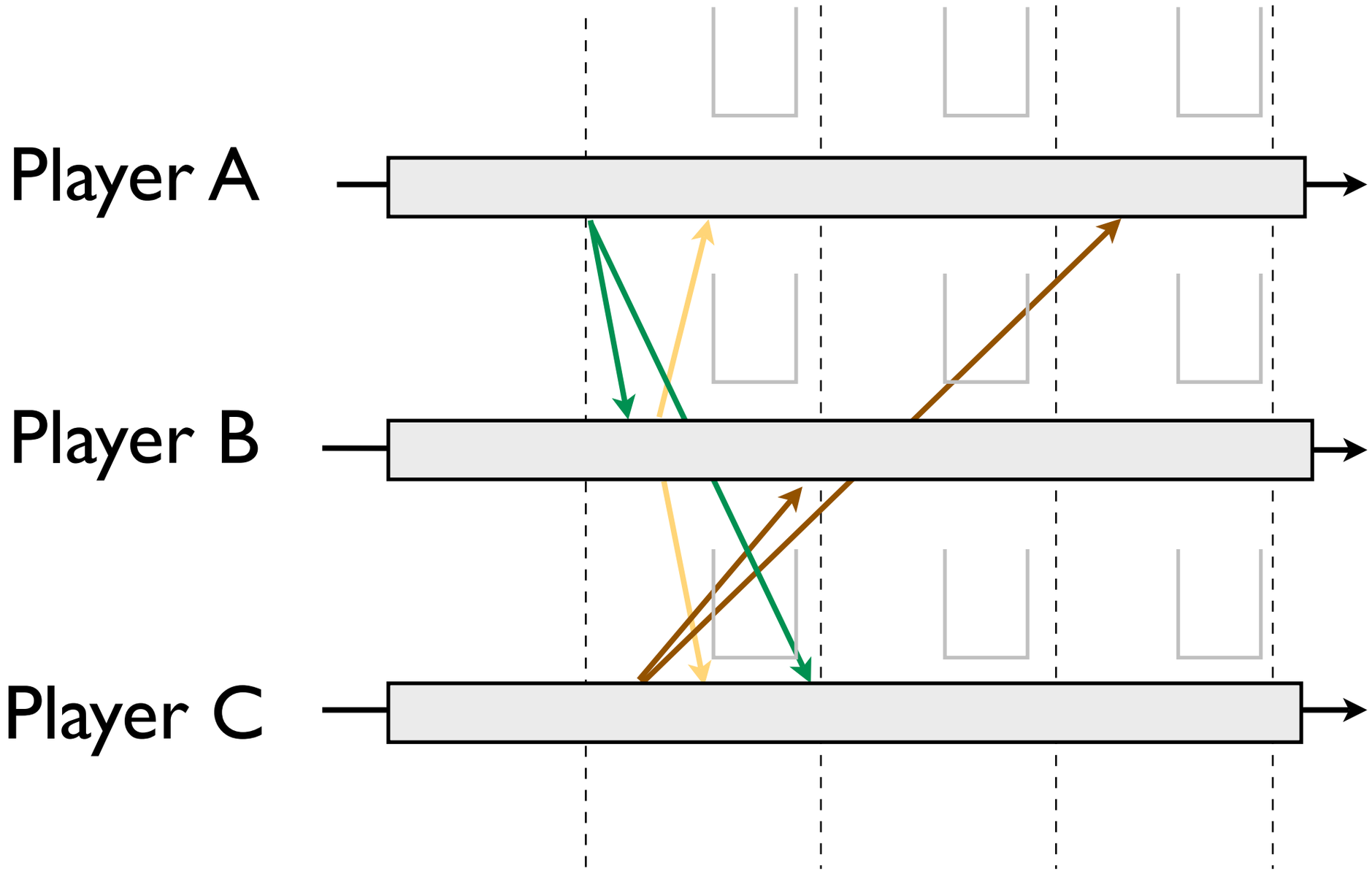
Players are expected to send an event in each round (e.g. 25 updates per second).



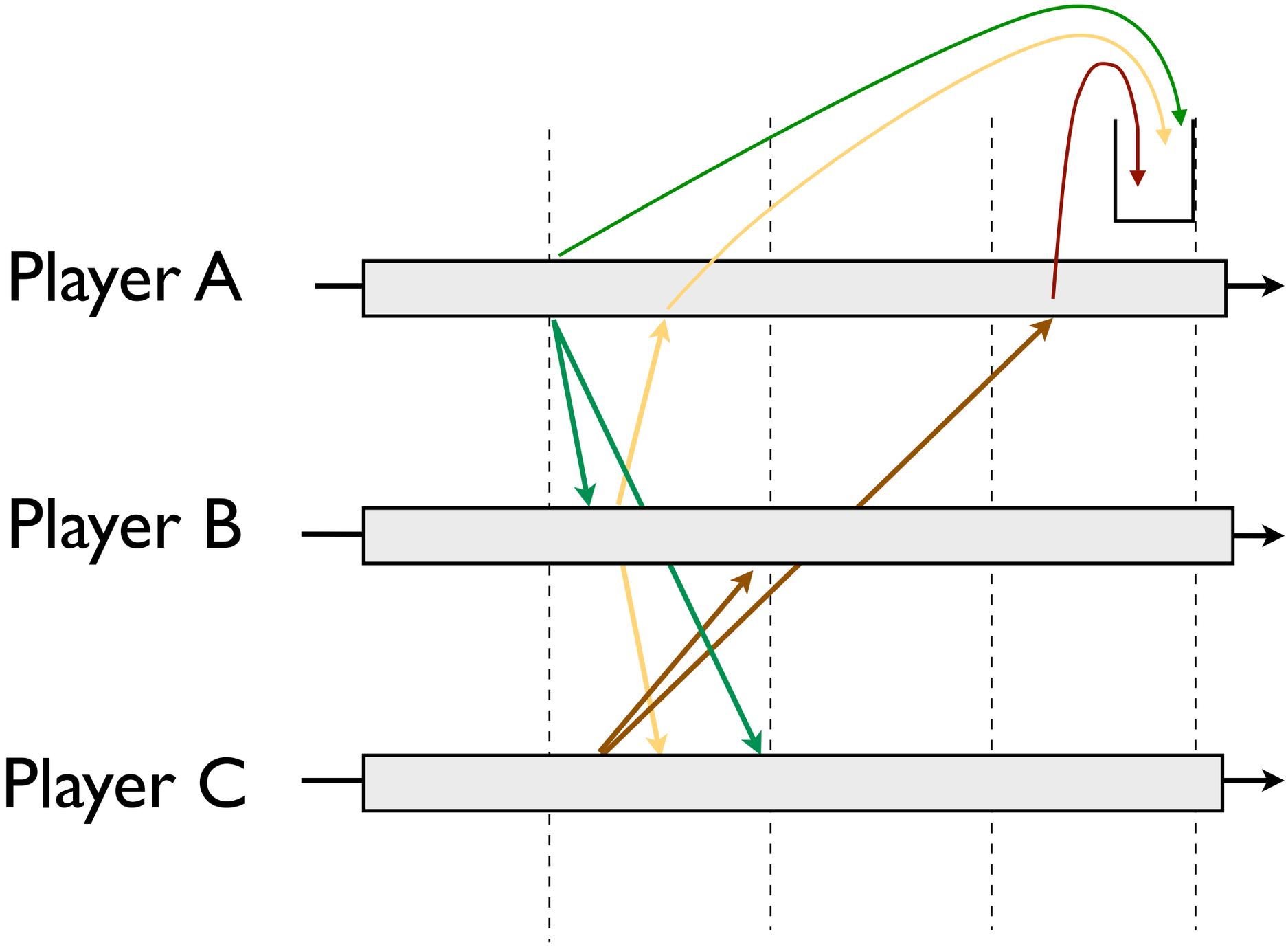
Players are expected to send an event in each round (e.g. 25 updates per second).



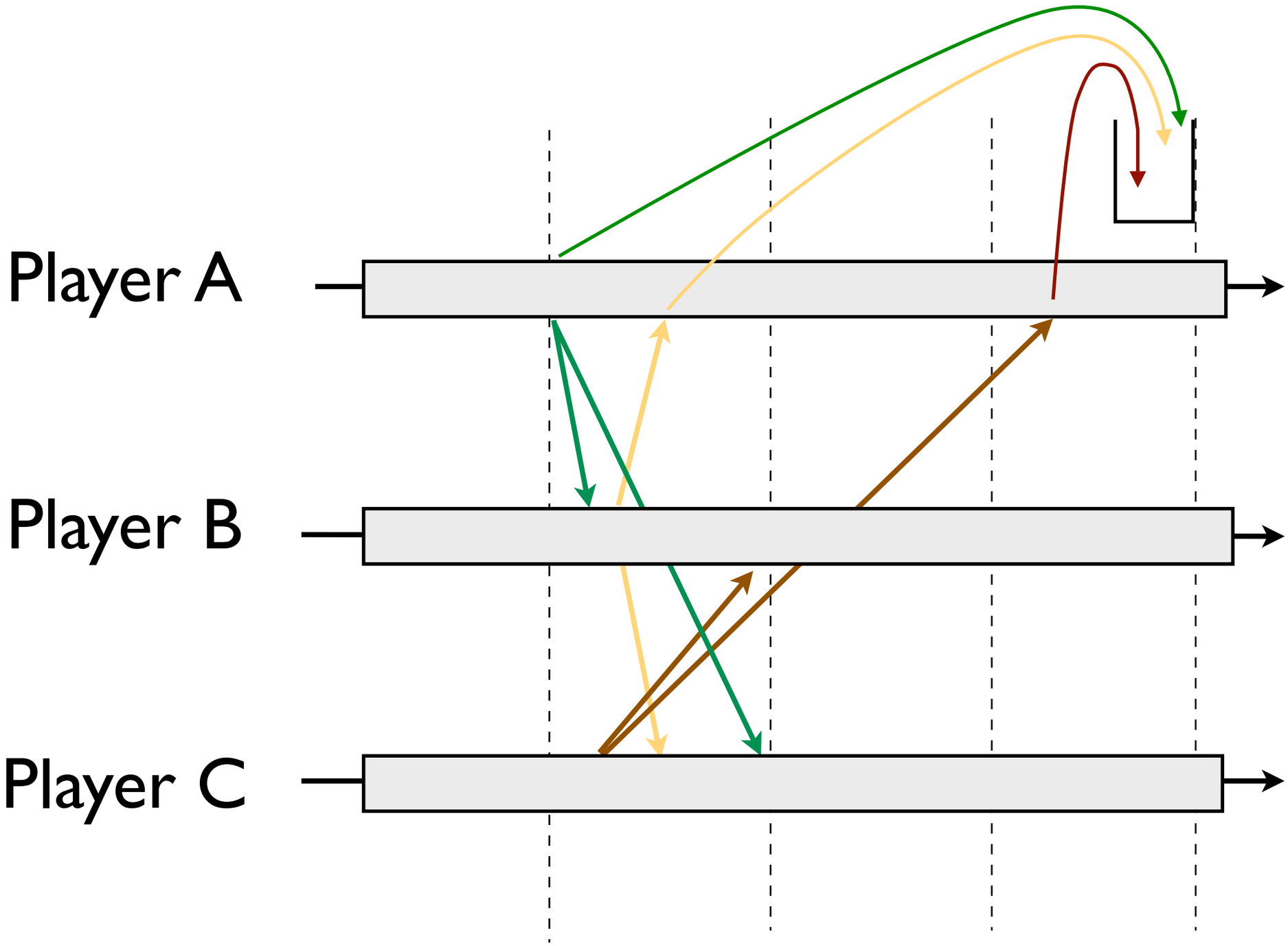
Every round has a “bucket” that collects the events.



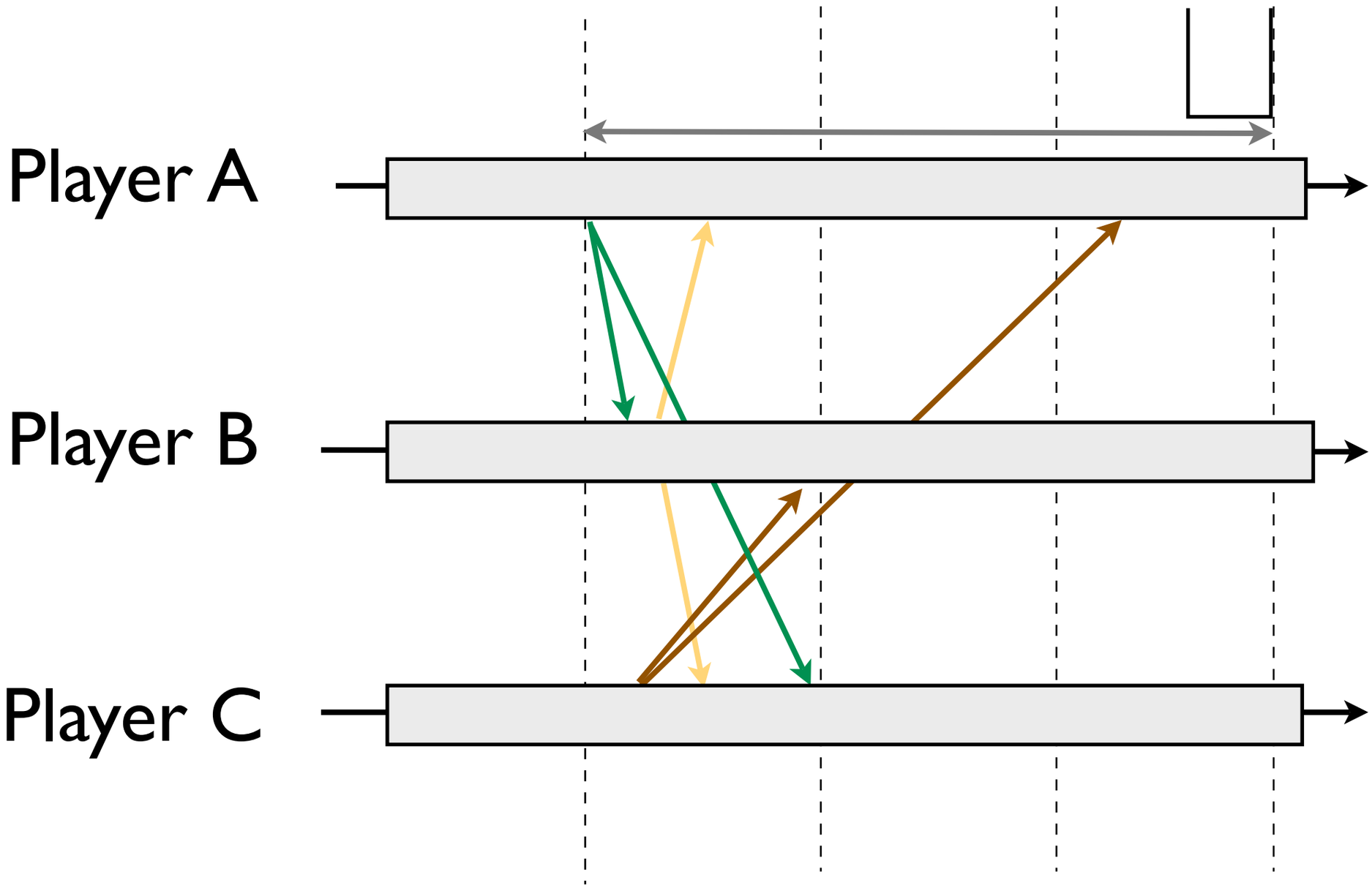
Events generated in the same round go into the same bucket of a future round. We know which round an event is generated in, based on time-stamp.



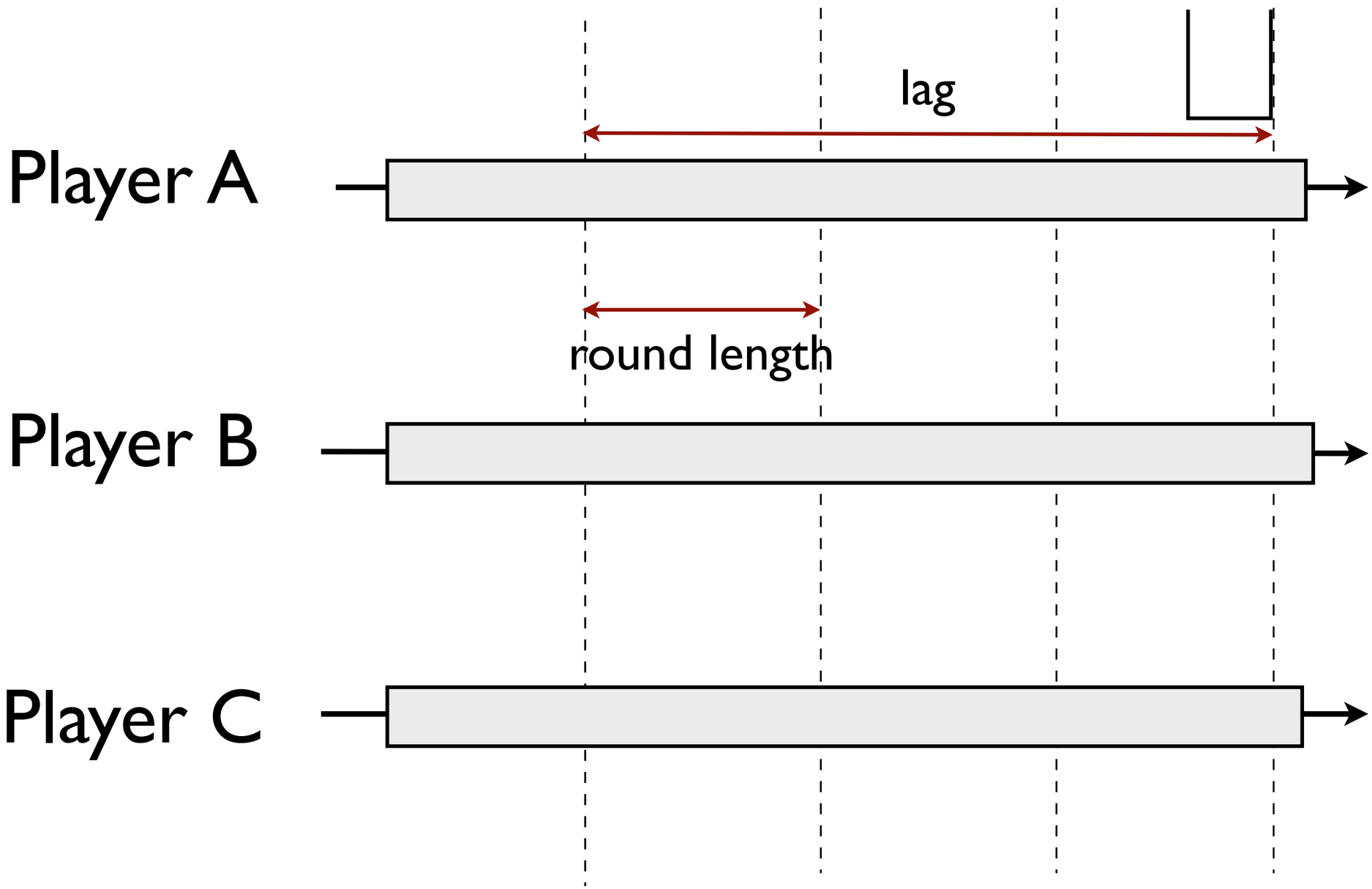
In each round, the events in the bucket are processed (in order of time-stamp).



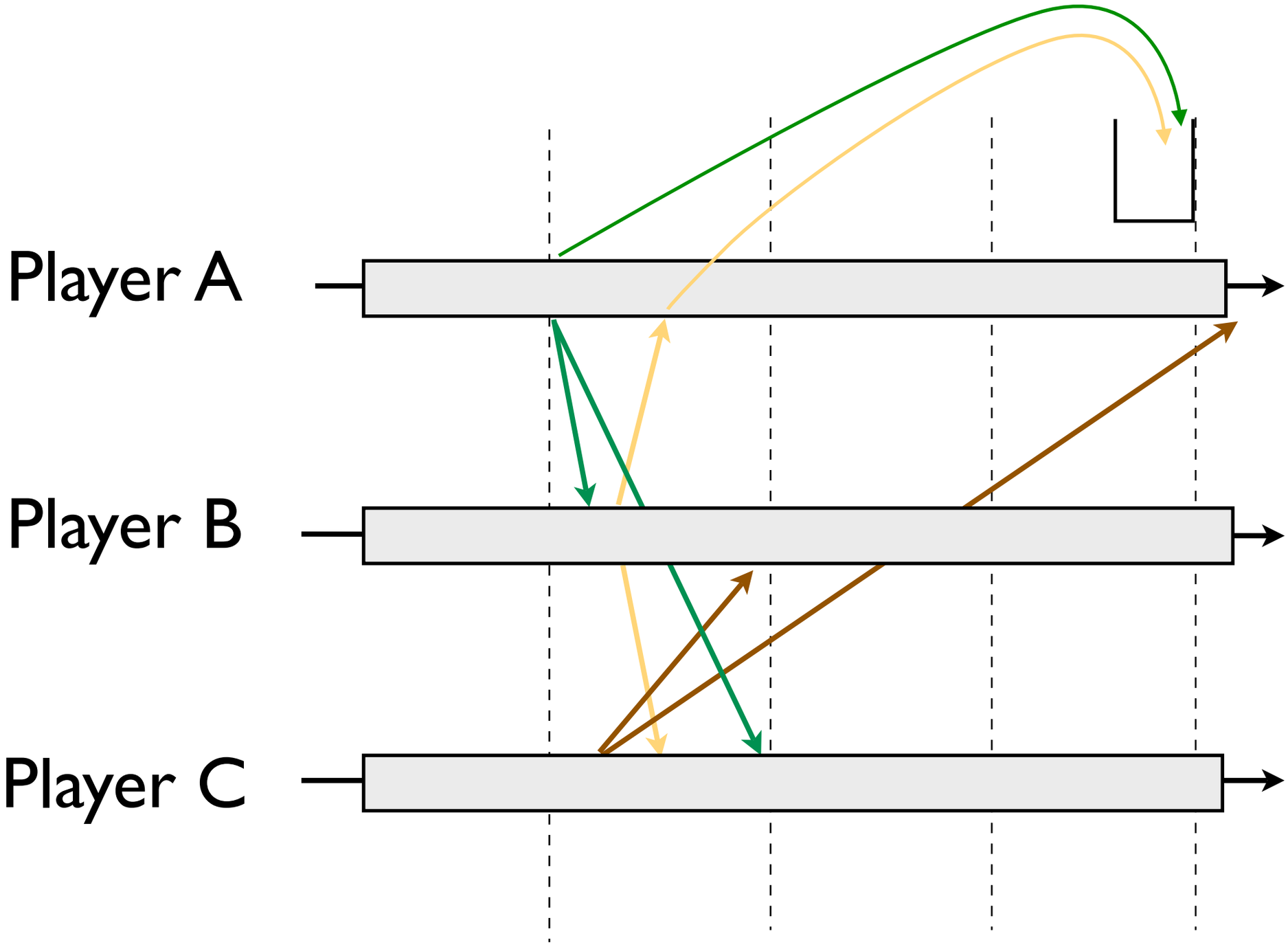
Which future bucket to go into depends on the latency among the players. (Hopefully not too far in the future else responsiveness suffers).



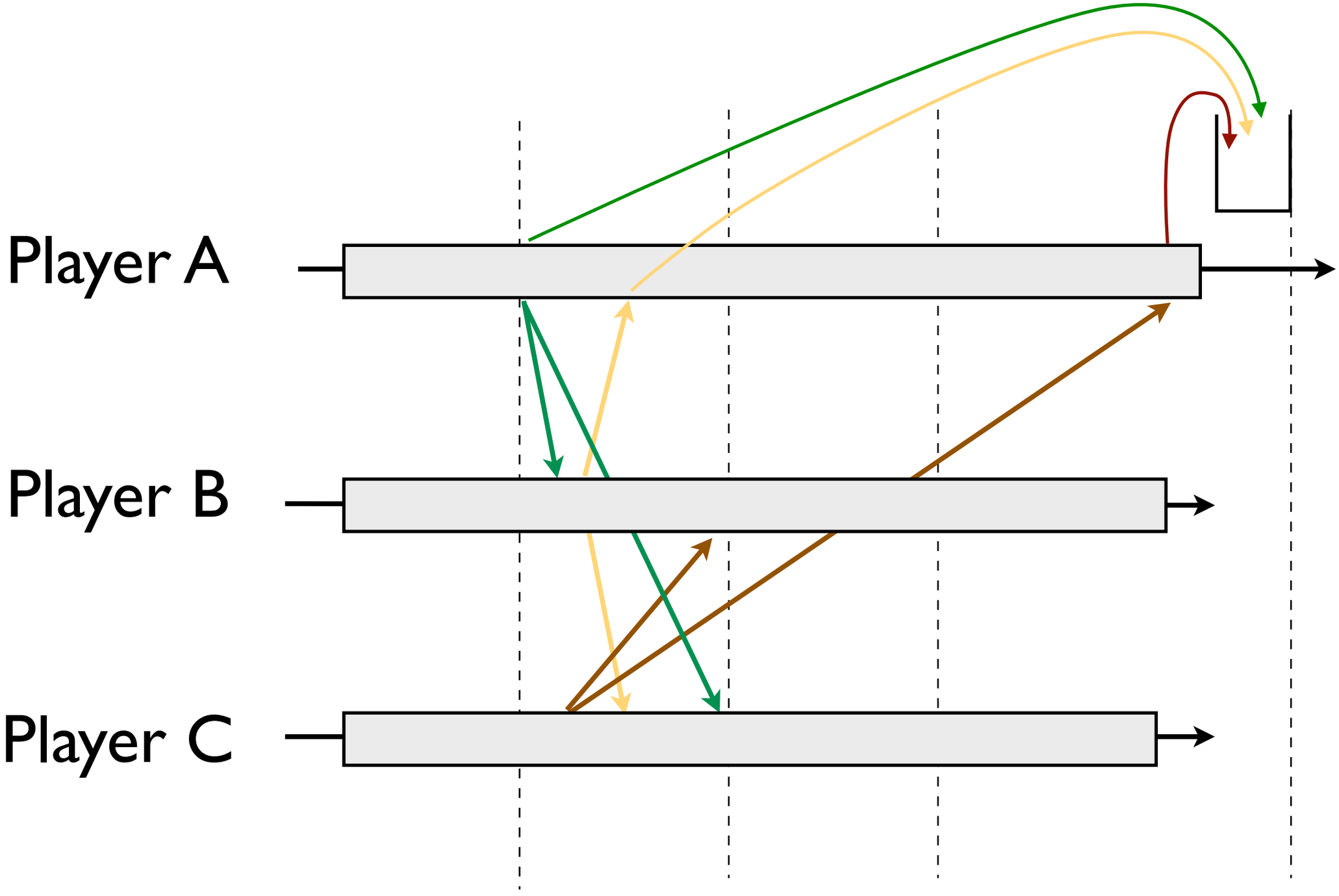
Two parameters needed : round length and lag. Lag depends on network latency; round length depends on rendering speed.



If events from another player is lost (or late), we can predict its update (e.g. using dead reckoning) when possible.



Alternative is to ensure every event is received before executing the bucket.



Stop-and-Wait Protocol

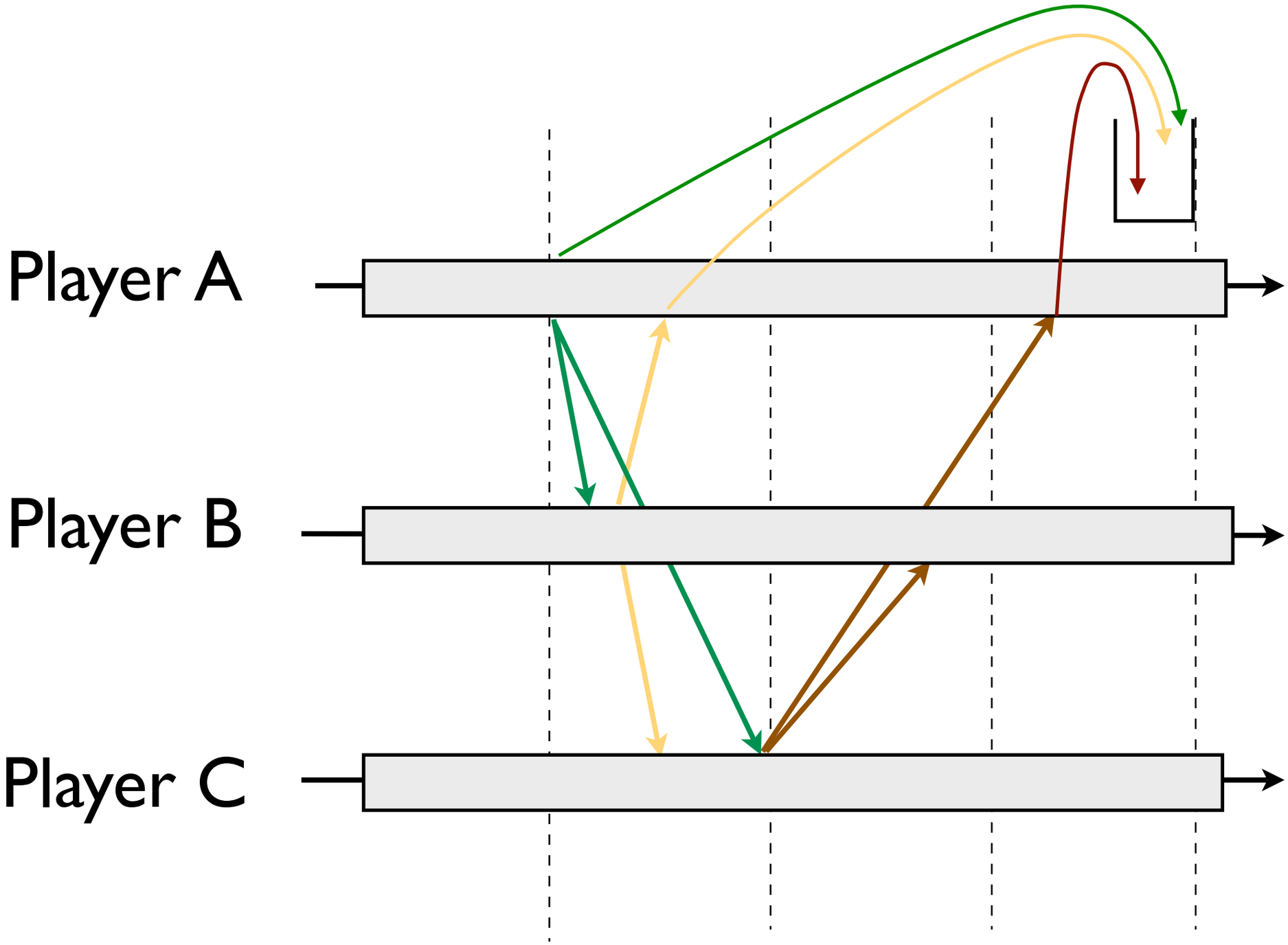
Synchronized Simulations

Every player sees
exactly the same states
(but maybe at different time)

Cheating

Look-Ahead Cheat

Player C (or a bot) can peek at A's and B's actions first, before deciding his/her moves.



Dealing with cheaters:

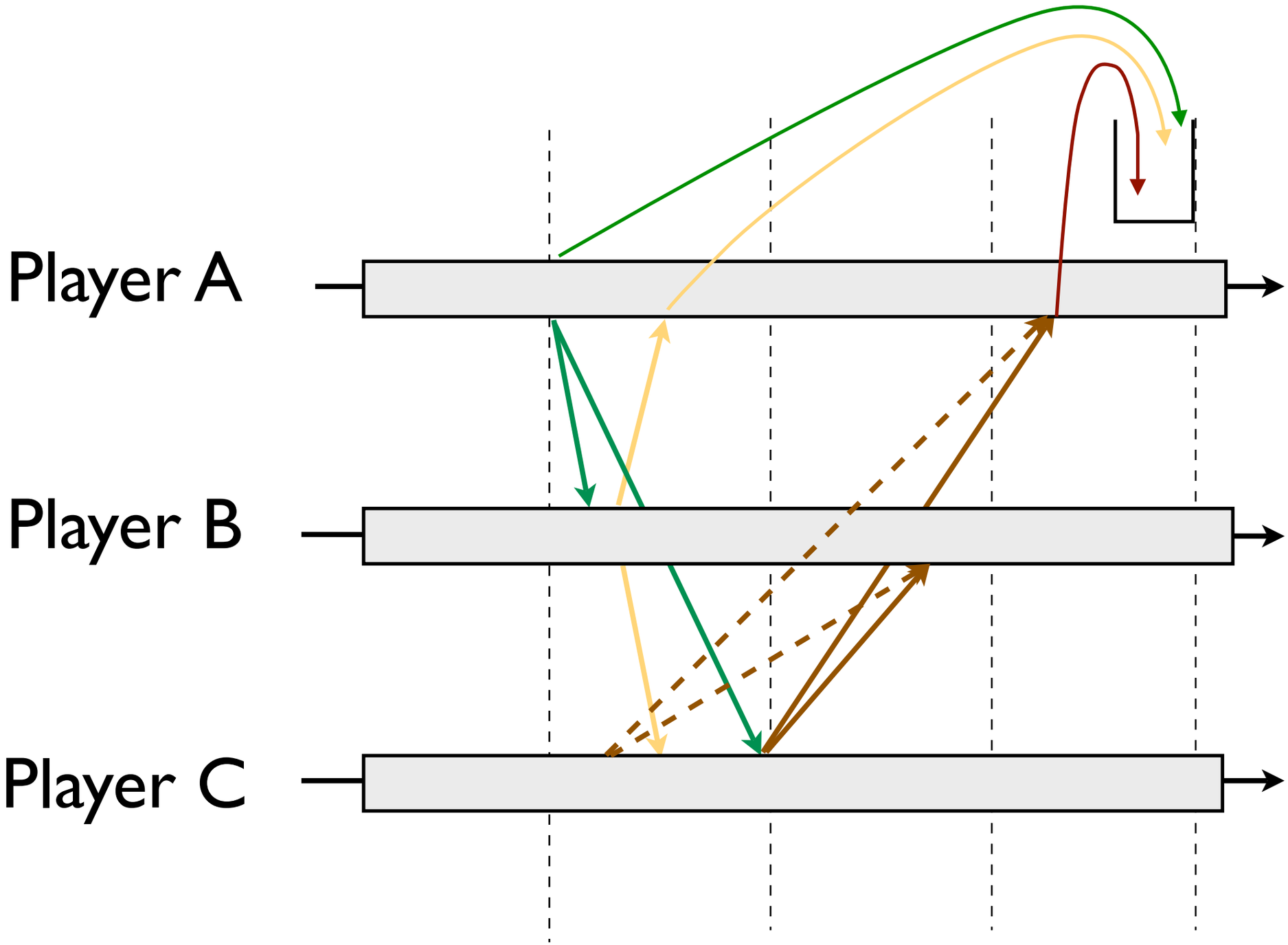
1. Prevent cheats (hard)
2. Detect cheats (easier)

Detecting Look-Ahead Cheats

“Mmm... player C always the last one that makes its move.”

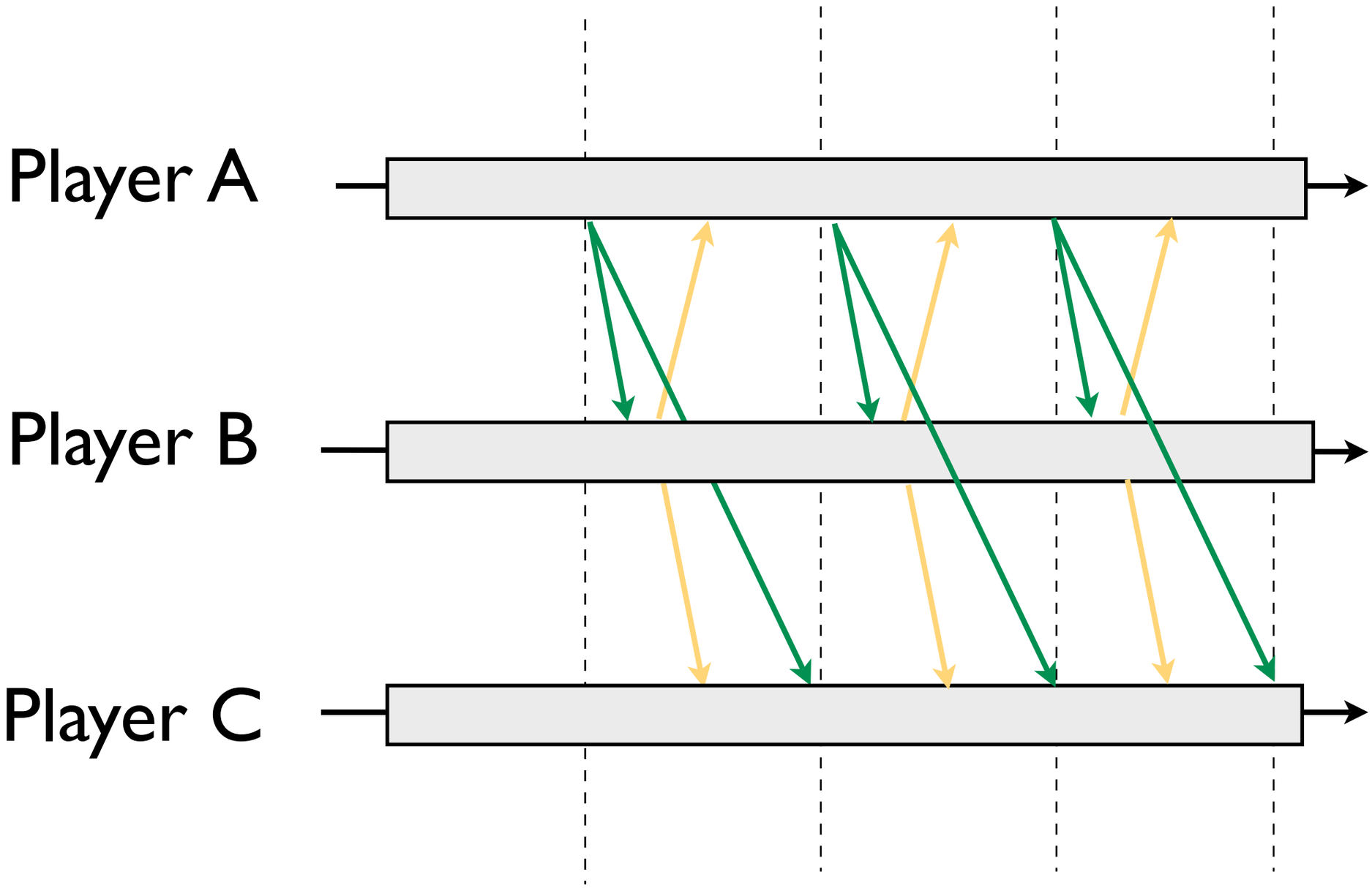
Time-stamp Cheat

Player C (or a bot) can put in an earlier time-stamp in its messages.



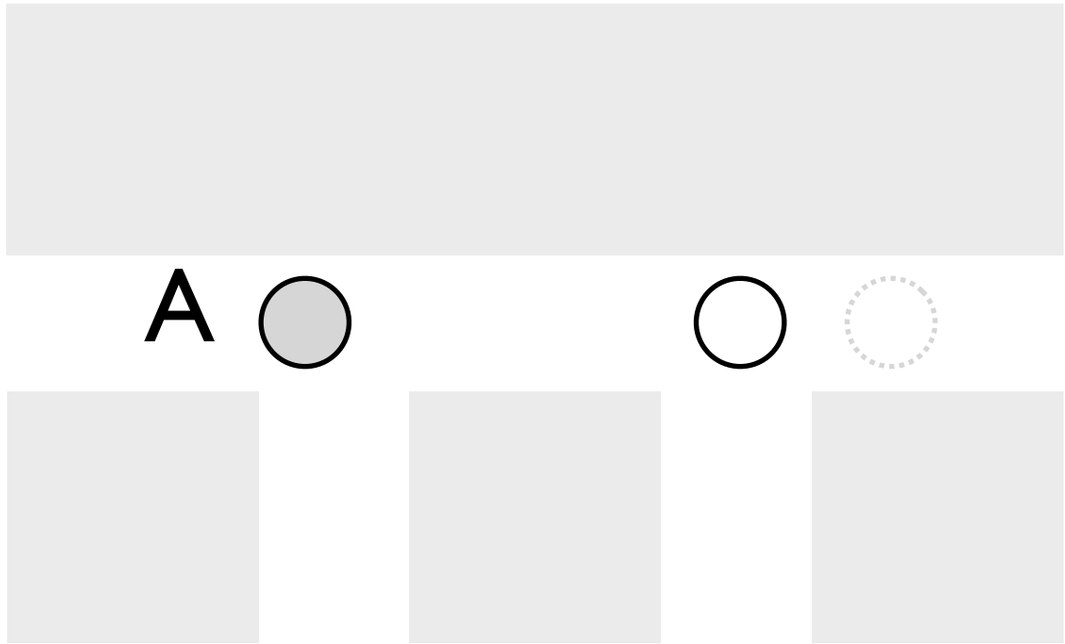
Suppress-Update Cheat

If dead reckoning is used, Player C can stop sending update and let others predict its position. C then sends an update at appropriate time to "surprise" other players.

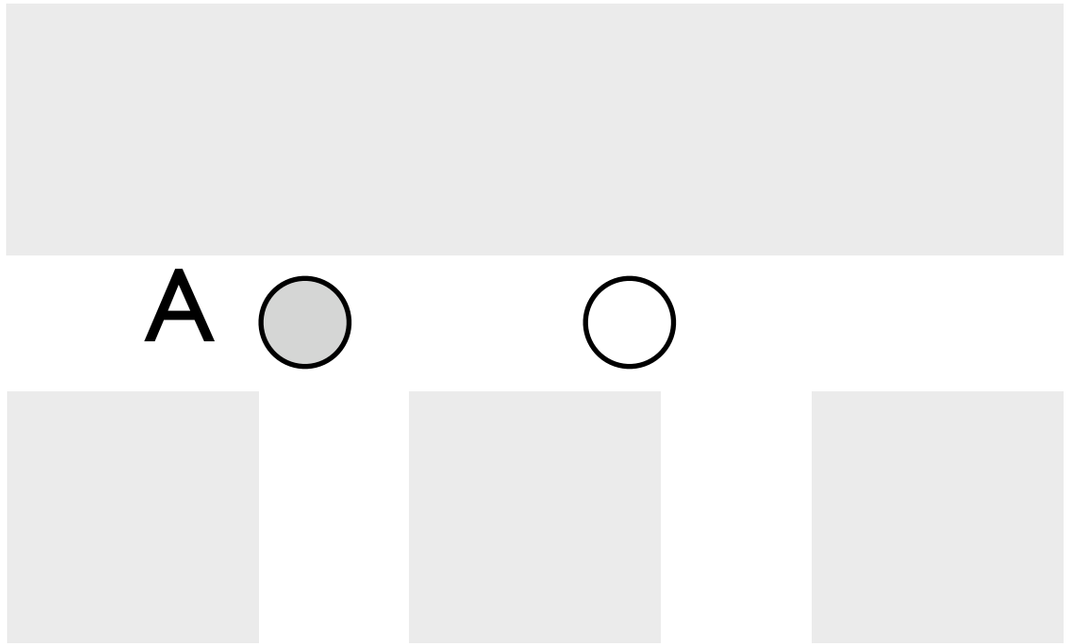



A
C

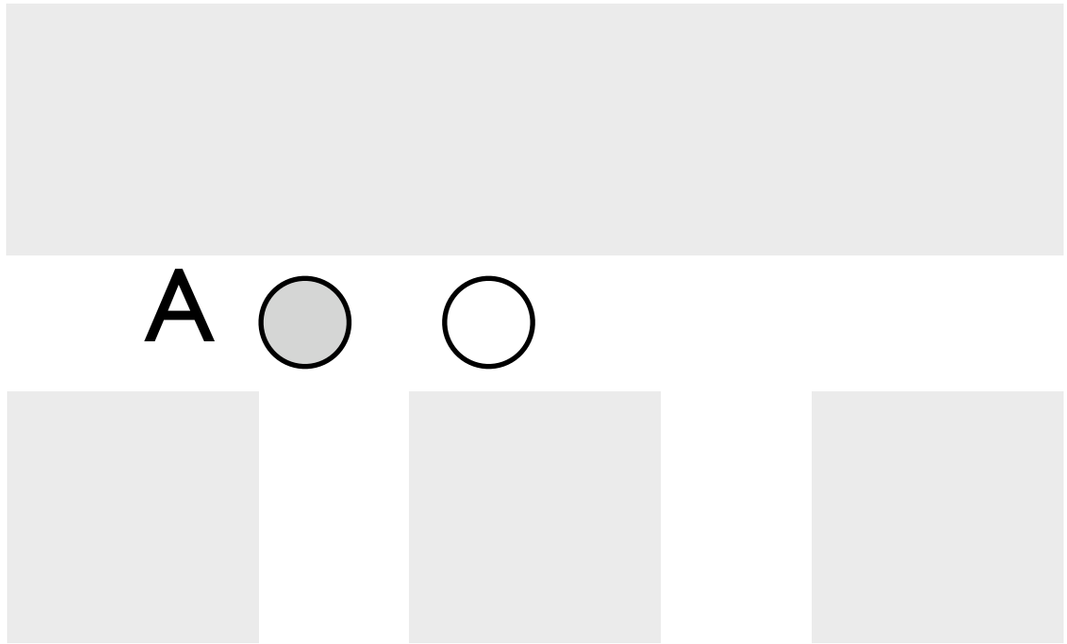
C stops sending update.
A predicts C's position.



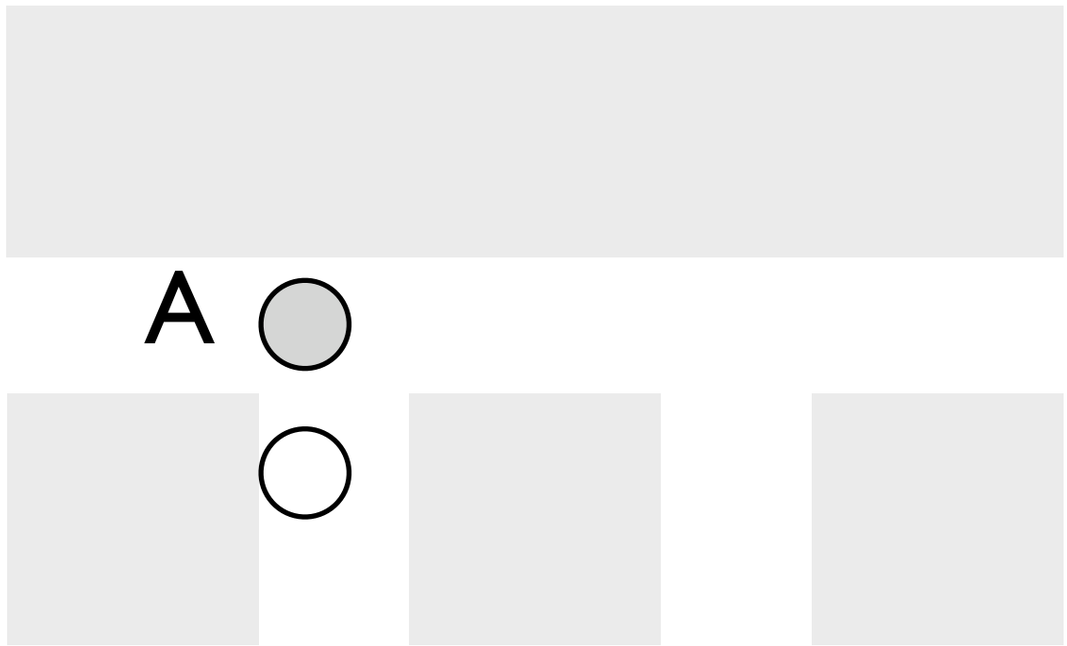
C stops sending update.
A predicts C's position.



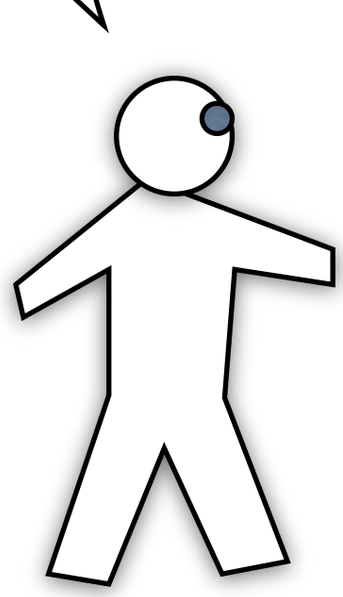
C stops sending update.
A predicts C's position.



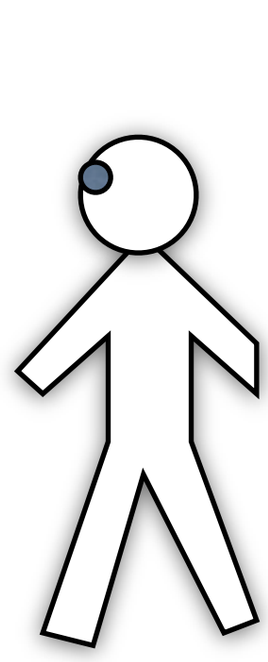
C sends an update and shoots A.



Cheater!

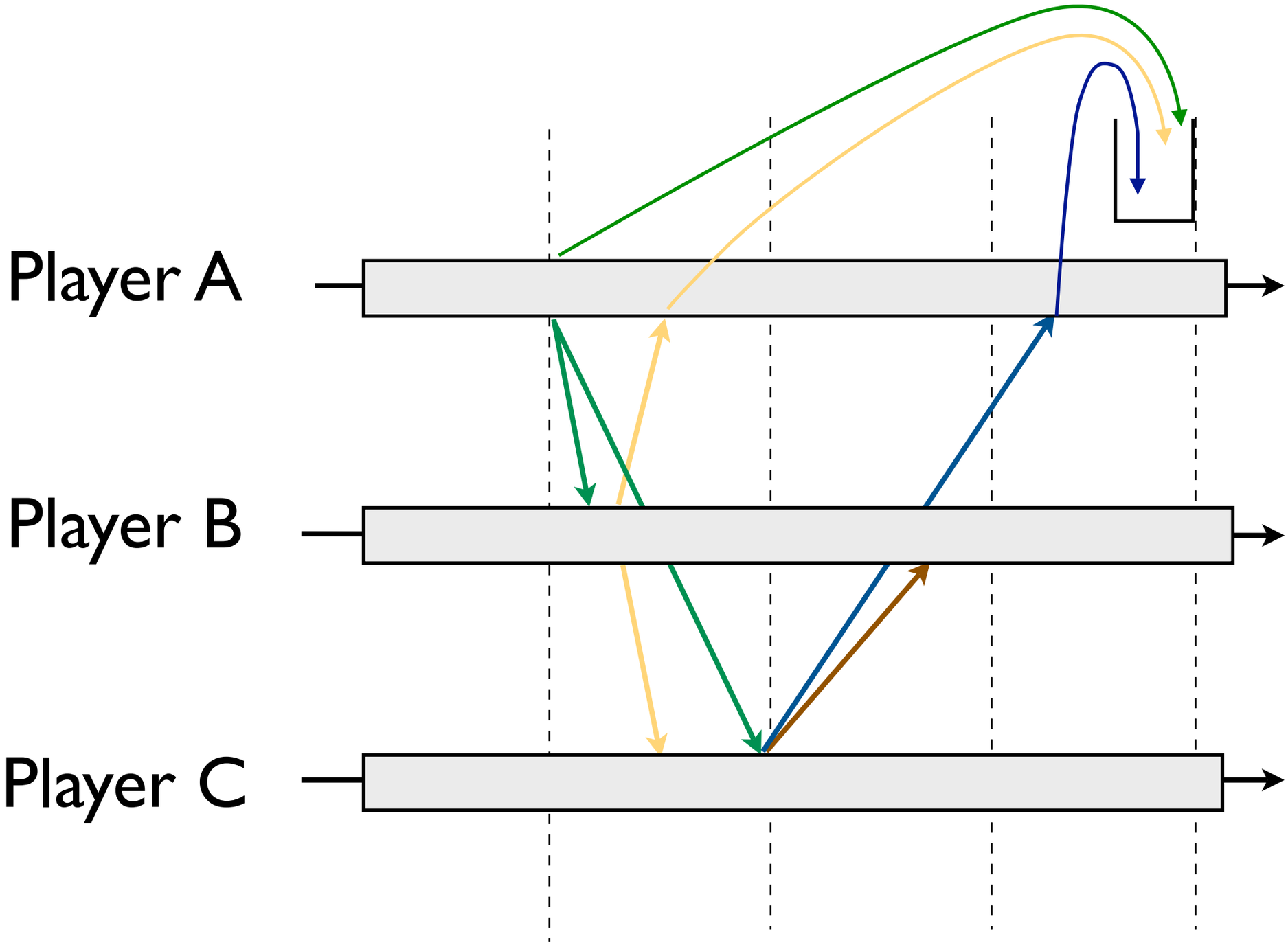


Not my fault! My packets were dropped..



Inconsistent Cheat

Player C can send different events to different players.



With synchronous simulations, we can compare periodically the game states to detect inconsistency cheats

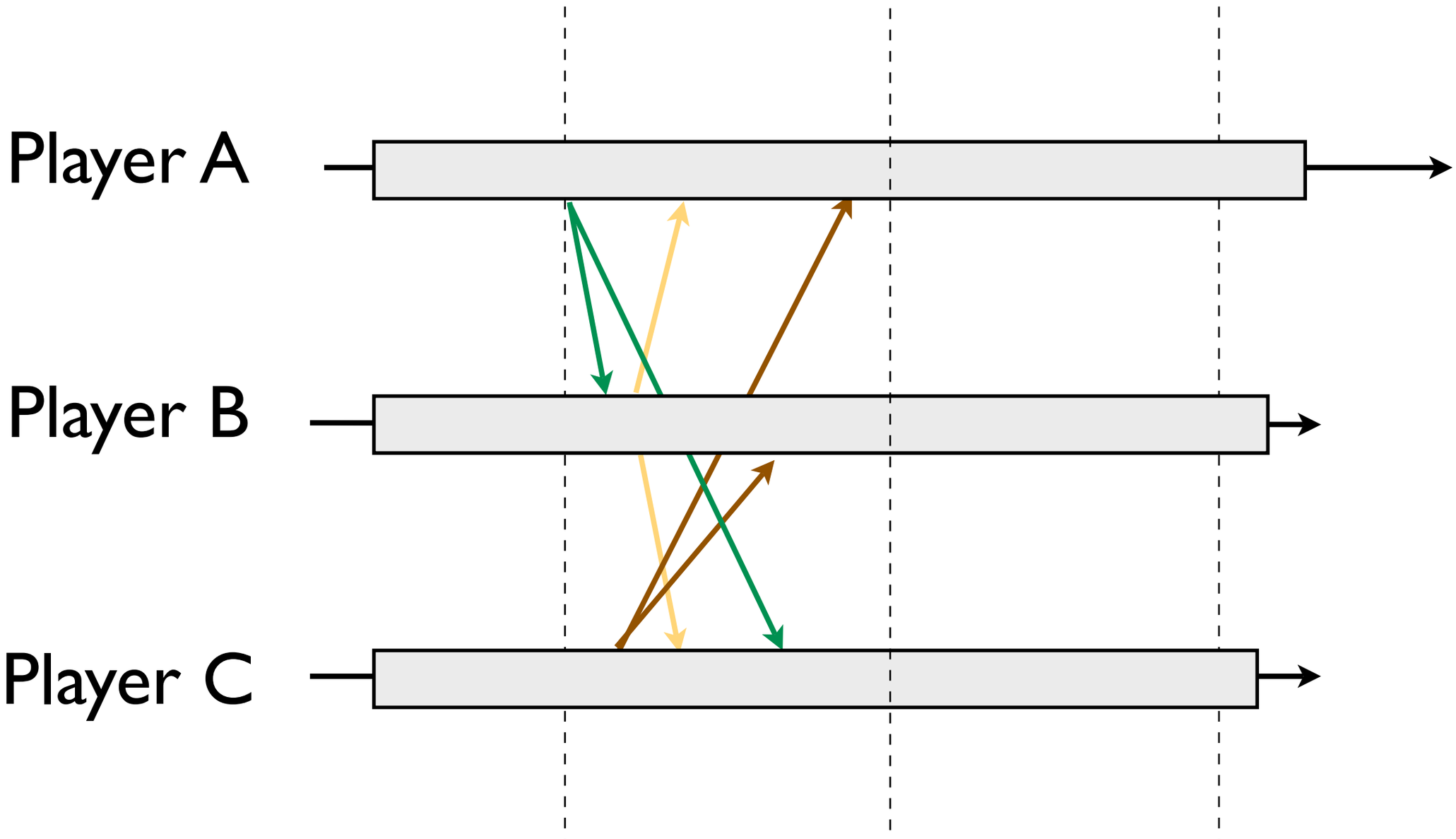
Cheat-Proof Protocol

Lock Step Protocol

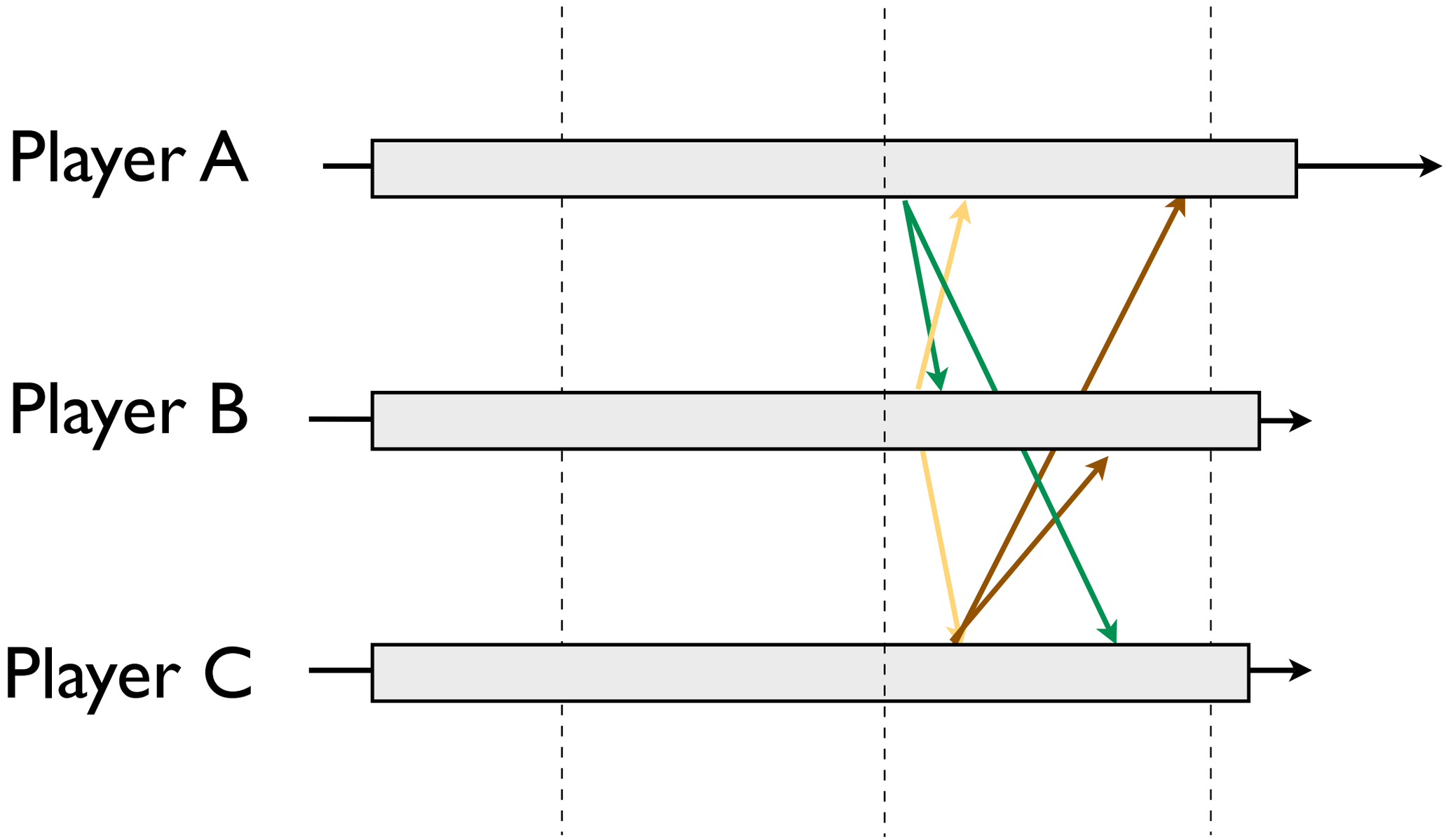
One-Way Function f:
Given x, we can compute f(x) easily. Given f(x) it's hard to find out x if x is random.

Lock Step Protocol

Stage 1 (Commit): Everyone decide on its move x, and send f(x) to each other.



Stage **2** (Reveal): After f(x)s from all other players are received, sends x to each other.



f(x) is known as commitment to x.

A player, once committed to its move, can't change it.

How does lock-step prevent:

look ahead cheat?
timestamp cheat?
suppress-update cheat?

Problem: Lock-step protocol is slow.

l_{max} = maximum latency
r_{max} = maximum frame (round) rate

Lockstep Protocol

$$1/r = \max\{2l_{max},\ 1/r_{max}\}$$

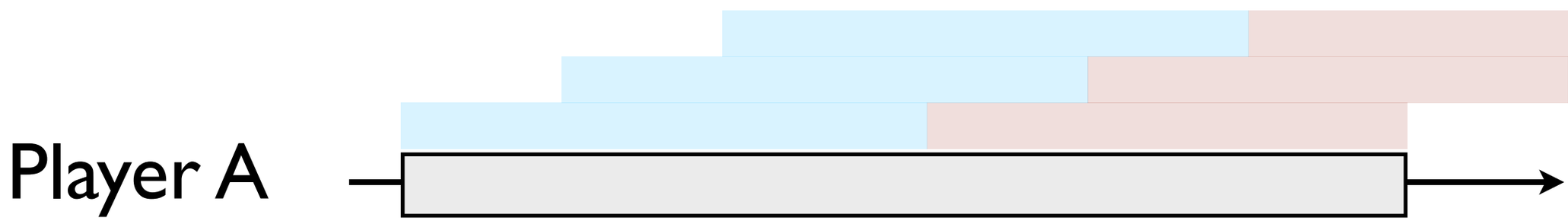
Idea: Use Interest Management

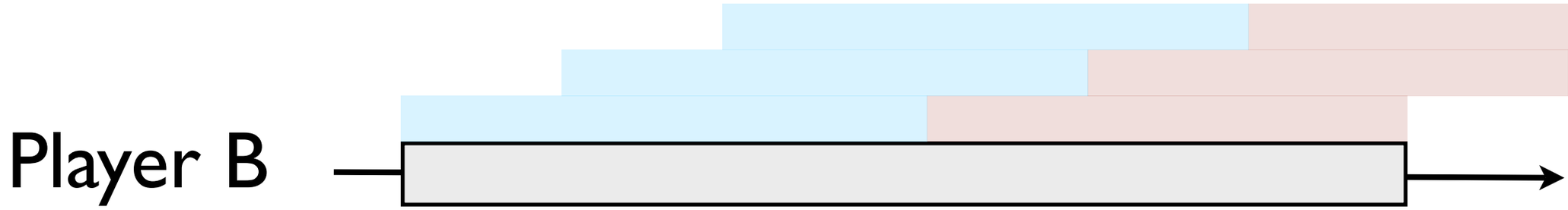
Players only engaged in lock-step protocol when they influence each other. Otherwise their games proceed independently.

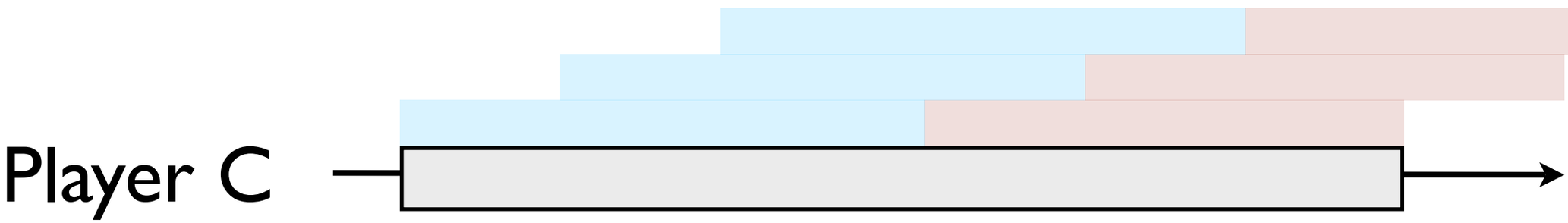
This is known as

Asynchronous Synchronization or Asynchronous Lock-step

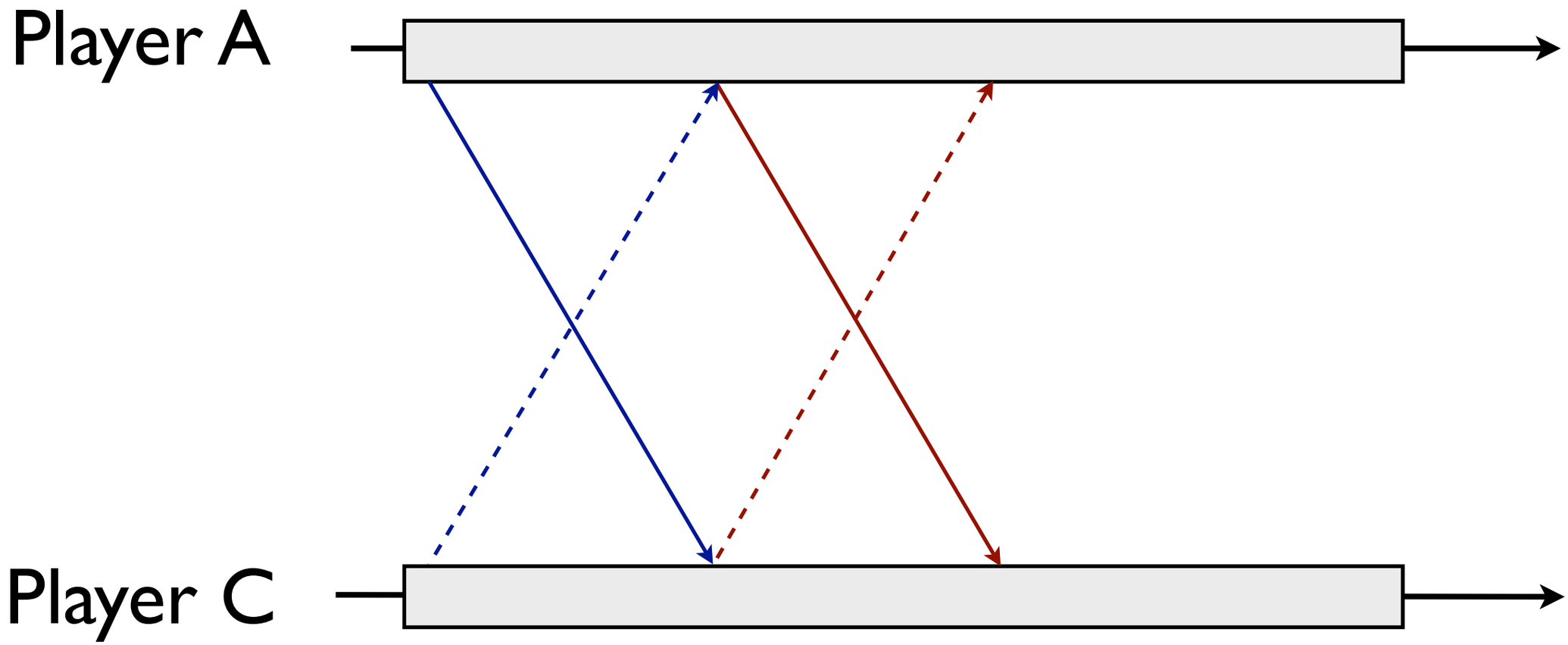
We may also stagger these two stages to improve responsiveness. Multiple commitments can be sent out before we reveal the actions.



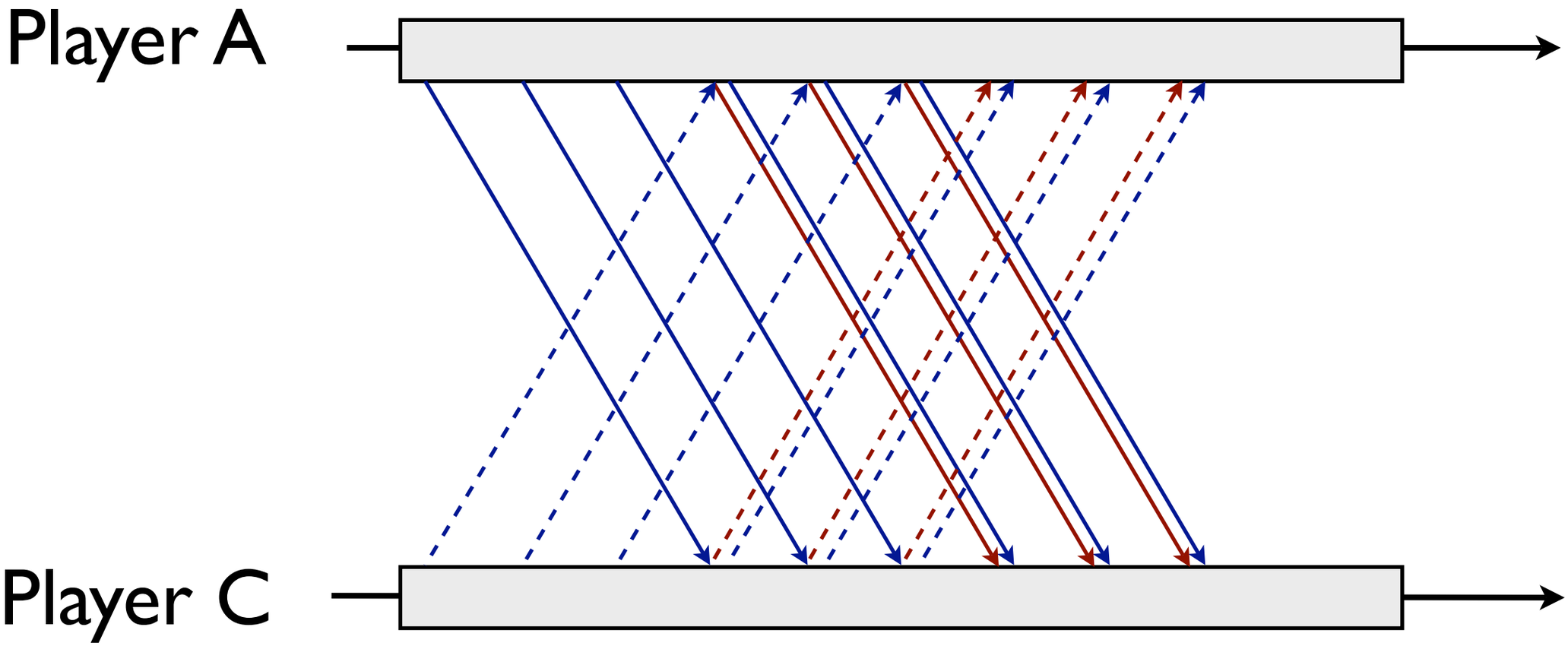




1. A player can reveal its action in round i once it receives all commitment of round i from other players.



2. A player can make p moves (send p commitments) without engaging in lockstep.



This is known as

Pipelined Lock-step

l_{max} = maximum latency
r_{max} = maximum frame (round) rate

Lockstep Protocol

$$1/r = \max\{2l_{max},\ 1/r_{max}\}$$

l_{max} = maximum latency
r_{max} = maximum frame (round) rate

Pipelined Lockstep Protocol

$$1/r = \max\{2l_{max}/p,\ 1/r_{max}\}$$

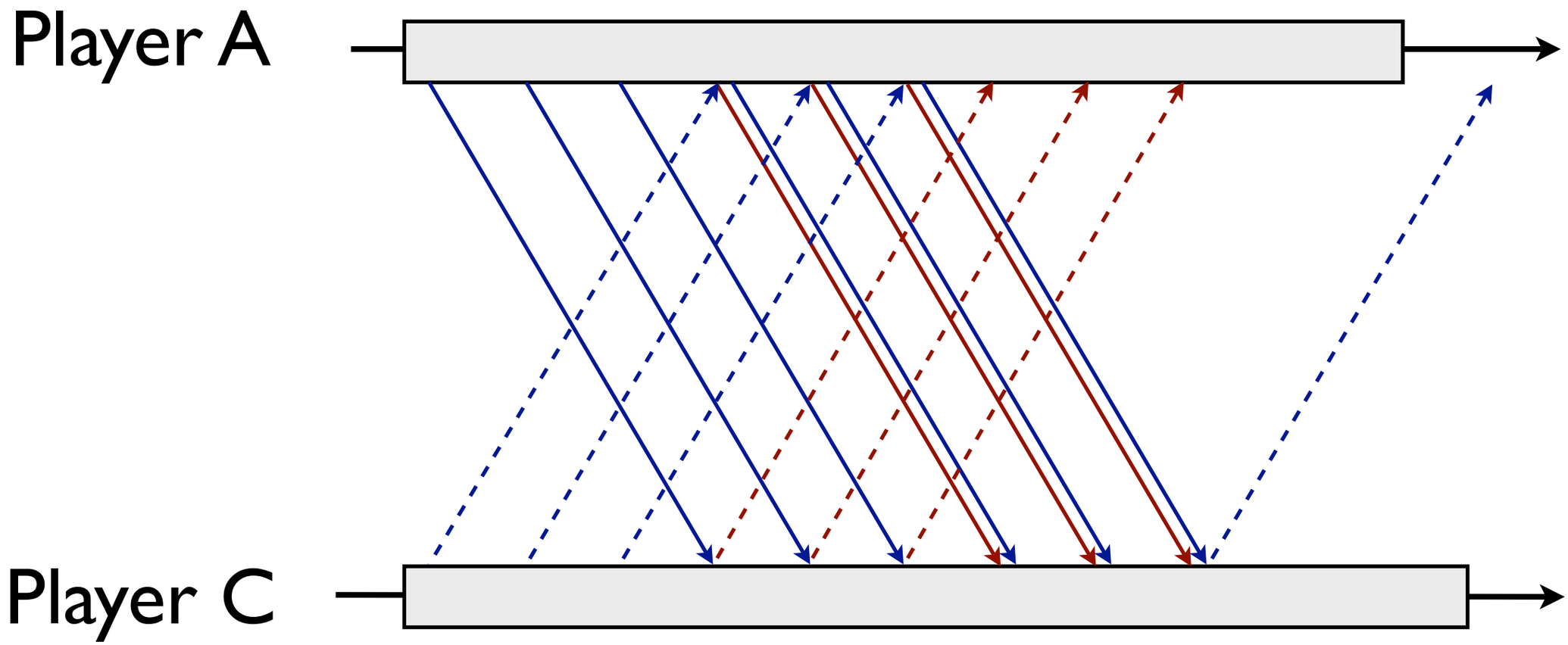
l_{max} = maximum latency
r_{max} = maximum frame (round) rate

We can pick optimal p as

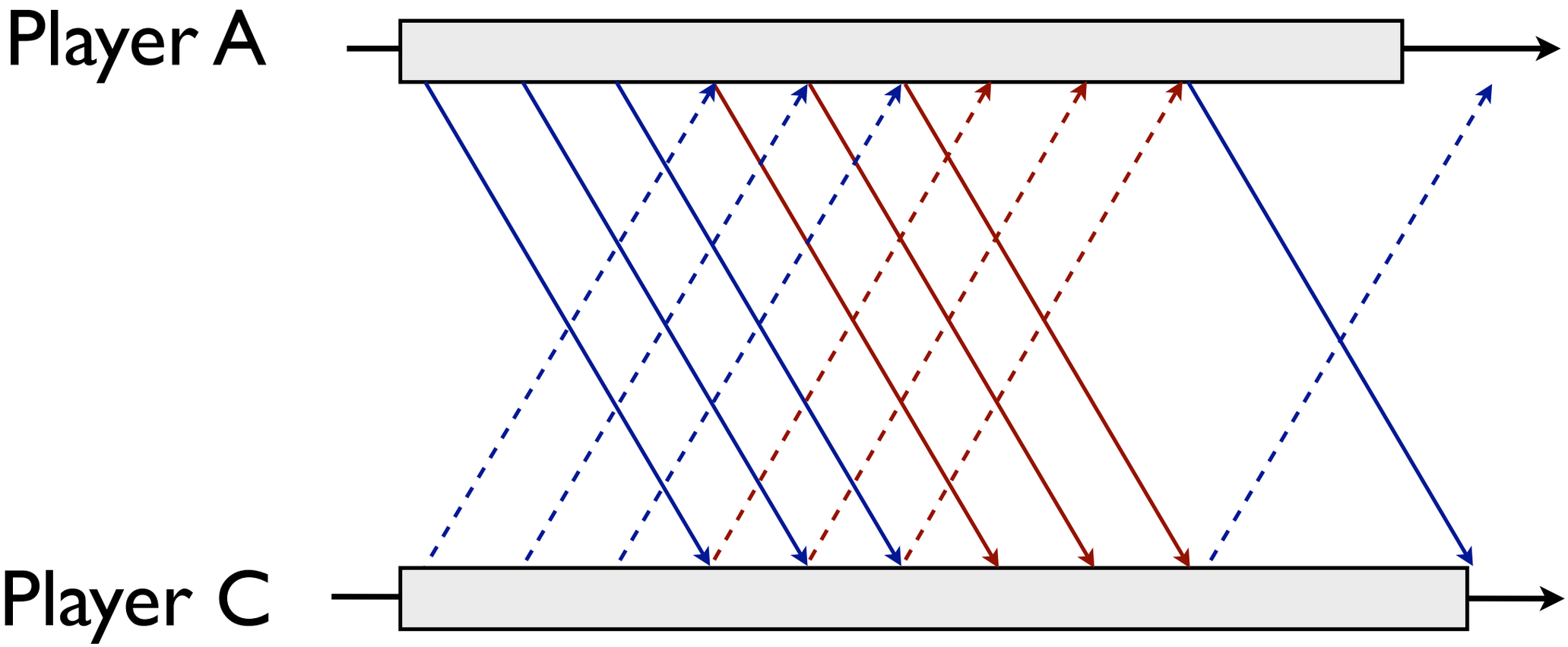
$$p = 2\ l_{max}\ r_{max}$$

Cheating in Pipelined Lock-step

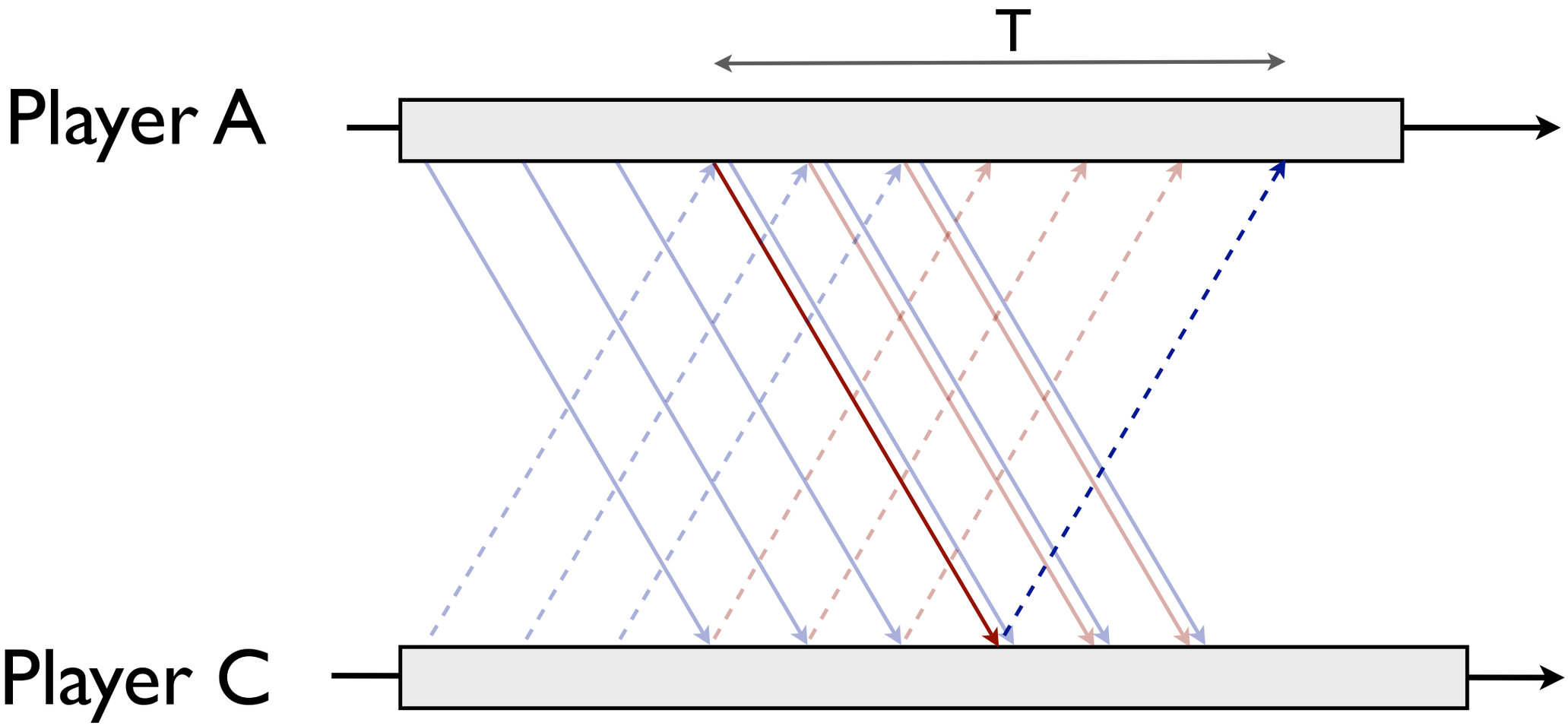
C makes its 4-th move after seeing the first p moves from A. A's first six moves are not based on C's move.



Fair if everyone do the same thing..



C can't peek at extra moves if T < RTT.
Can use this to detect late commit.



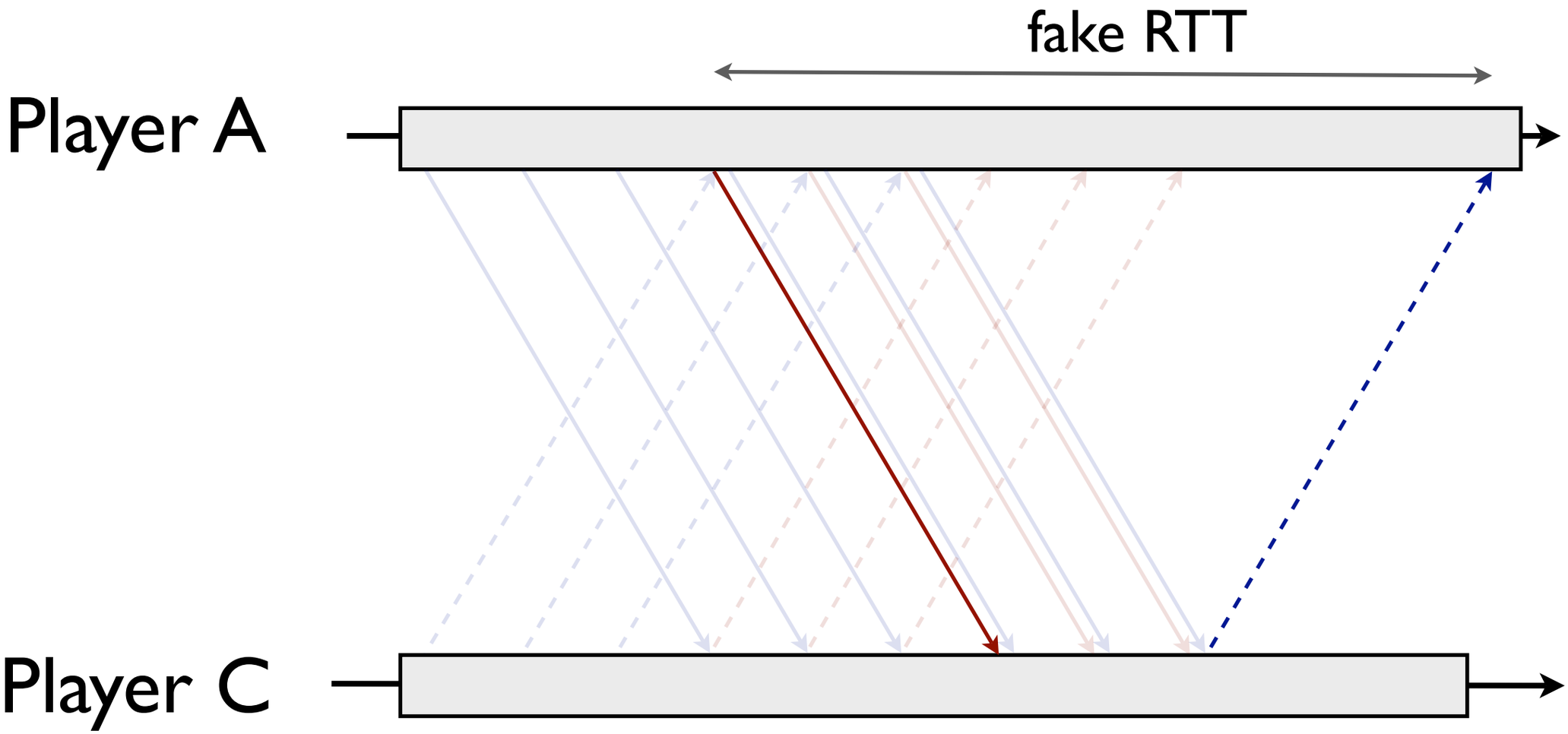
l_{max} = maximum latency
r_{max} = maximum frame (round) rate

Pipeline Lockstep Protocol (without late commit)

$$1/r = \max\{l_{max}/p,\ 1/r_{max}\}$$
$$p = l_{max}\ r_{max}$$

Player can lie about latency!

C can't peek at extra moves if T < RTT.
Can use this to detect late commit.



But, if fake RTT increases l_{max}, p increases, limiting the effects of cheat.