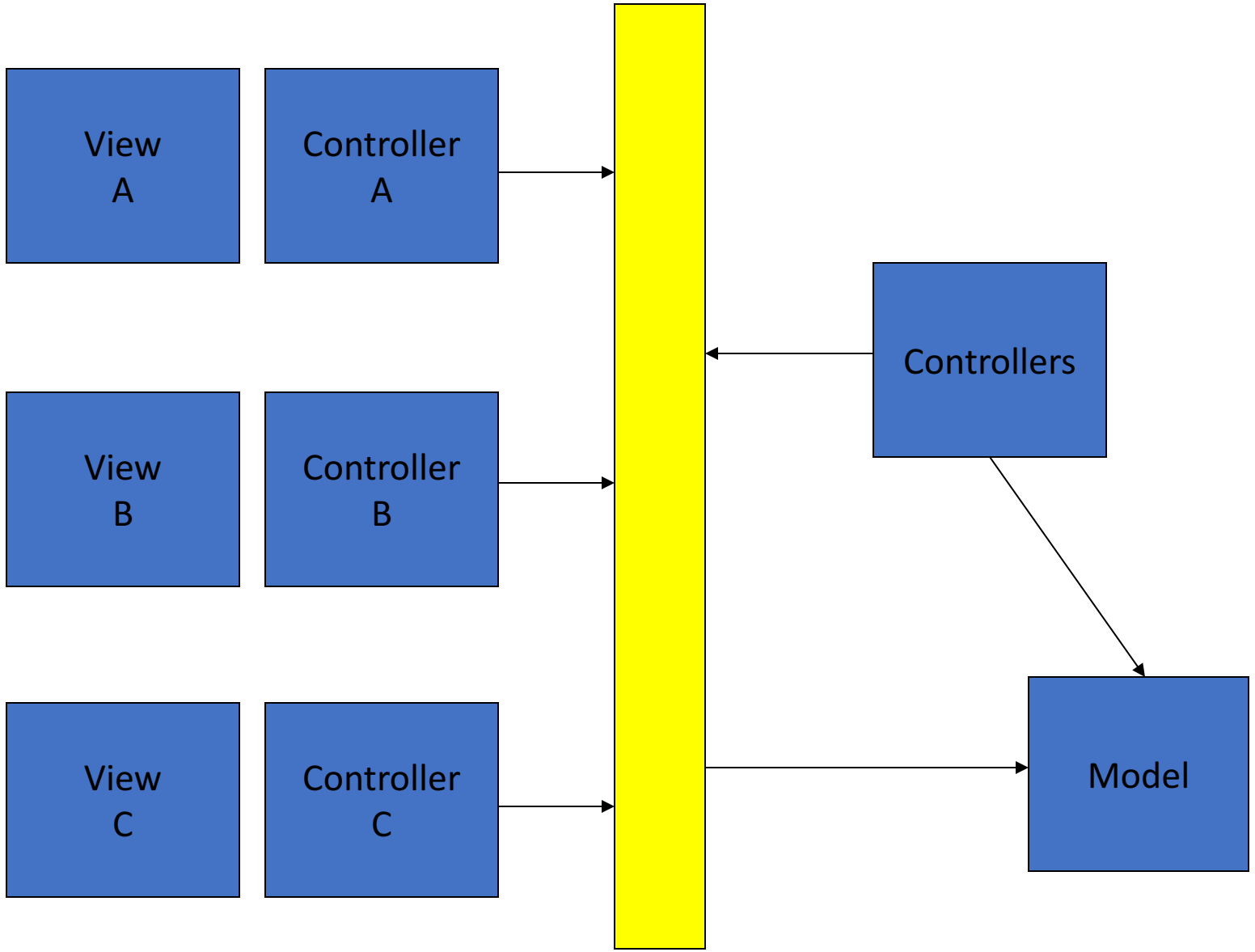
G54GAM Games

Building Games

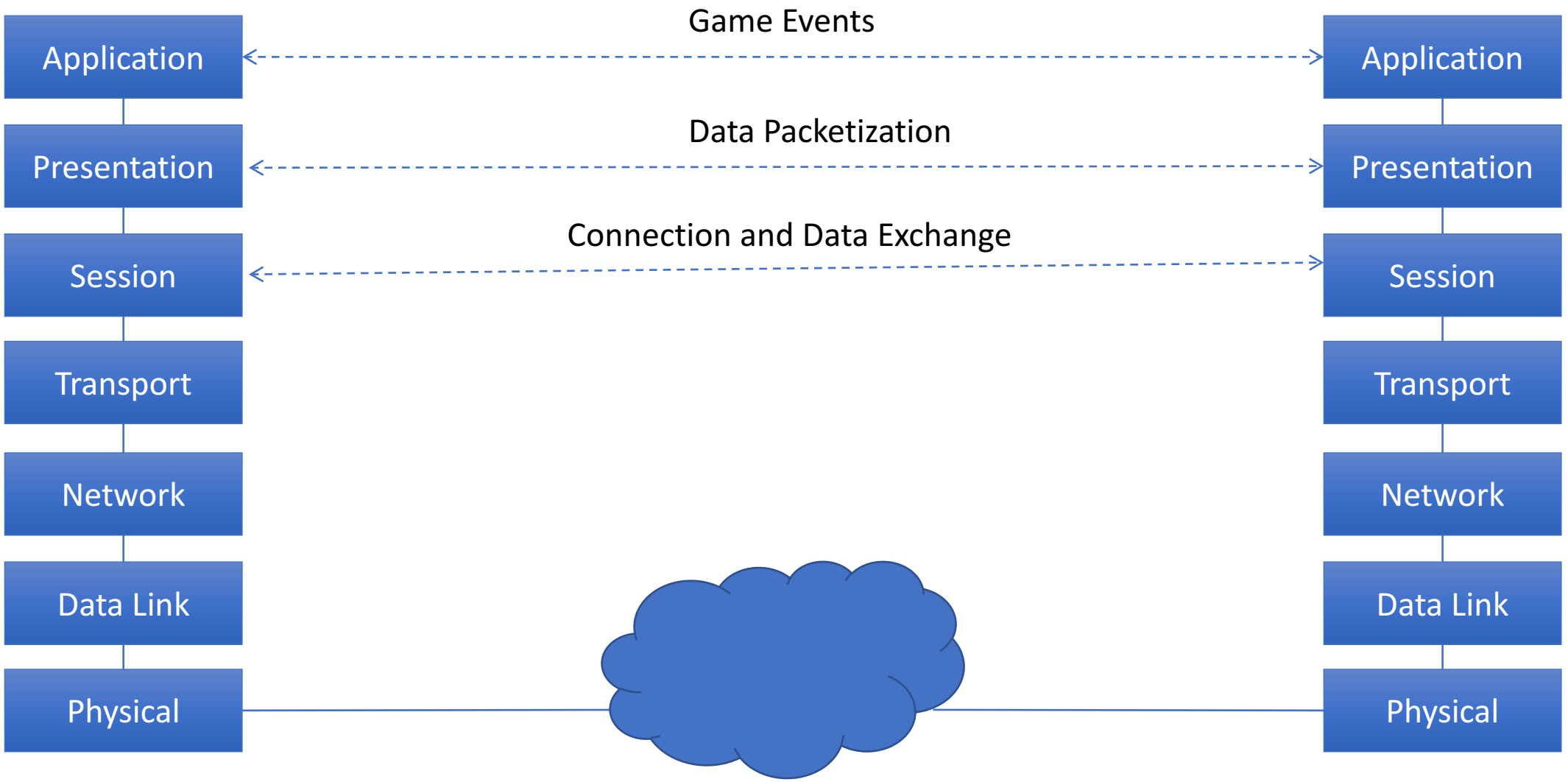
Multiplayer Architectures (2)


View
A
Controller
A
View
B
Controller
B
View
C
Controller
C
Controllers
Model

Networks are subject to delays

- **Real time simulation**
  - Game loop runs at 30 Hz
  - Renderer redraws at 50-100 Hz
- Each input has to…
  - Travel from the client to the server
  - Be processed by the server
  - Wait for the server loop to update the model
  - Travel from the server to the client
  - Be drawn onscreen
  - **Round trip time**
- For several players on the Internet playing a fast paced game
  - **Latency** – Networks can be slow
  - **Bandwidth** - Each client needs to know about the current state of the model every loop
  - **Reliability** - Packets get lost or delayed or arrive out of order

Protocol stack



Lag / Consistency

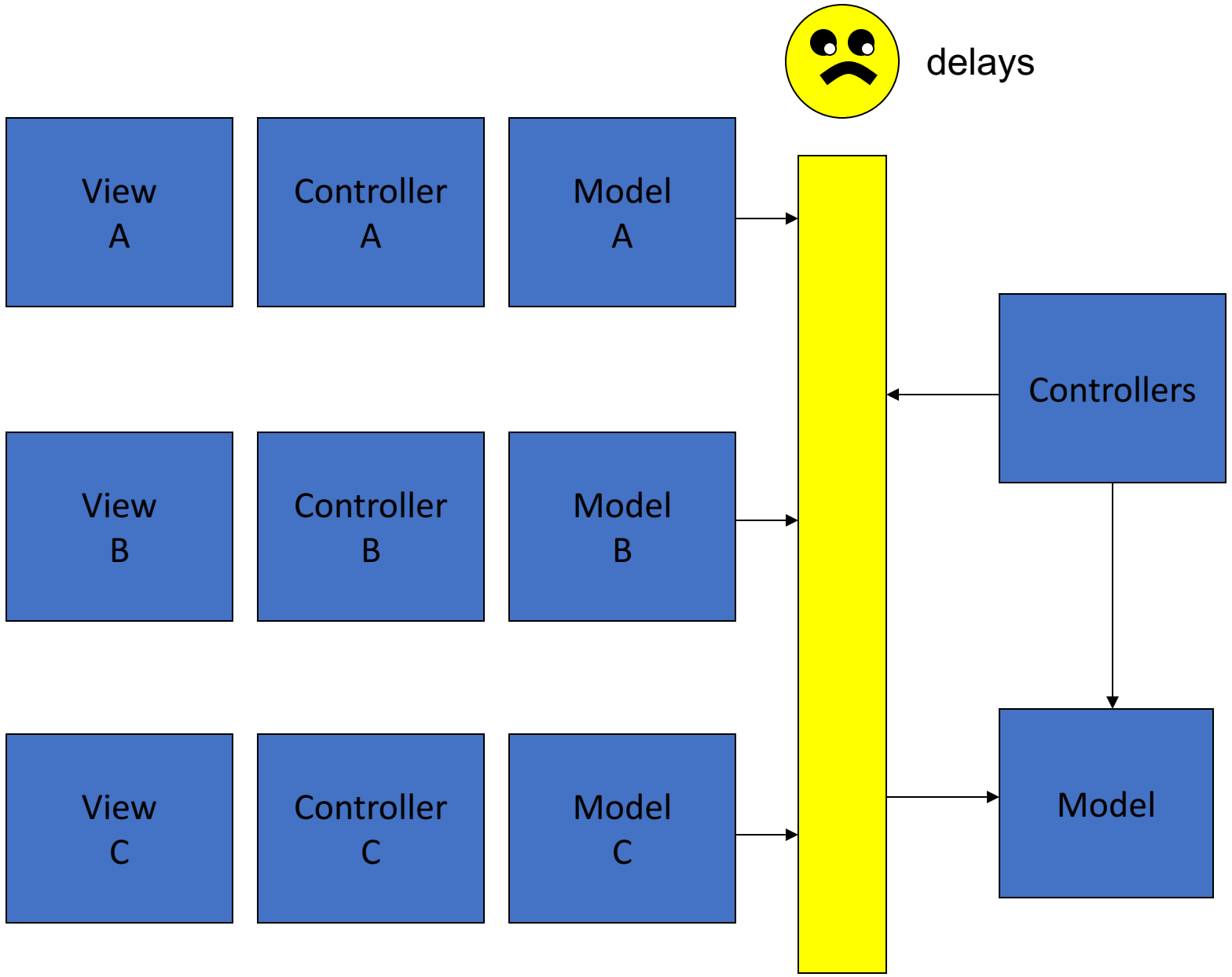
- Network delays lead to logical Inconsistencies
- A player shoots at another player
  - The player/client thinks they should have hit
  - The **fire** command takes some time to get to the server
  - The server thinks the player missed
  - How do we resolve this?
- Two players try to pick up the same object
  - Does the player with the lowest latency get it?
  - What does the player with the highest latency see?
  - How do we resolve the inconsistency of both players trying to pick it up?
- Implementation needs to have
  - Speed (otherwise it feels slow and jerky)
  - Synchronisation (to avoid logical inconsistencies)
  - Not use too much bandwidth (remember dial-up)
  - Cope with packet loss

Key Implementation Details

- Consistent Replication
  - The communication of state or events to a remote peer
  - *Replicating* an object causes it to appear to be created and updated on a remote peer
    - Transfer sufficient data to reflect changes to the local object remotely
    - Replication is the application layer of the networking stack
- Authority
  - Manage *permission* to update the persistent state of an object
    - The game server is *authoritative* over dealing damage
    - Resolve inconsistencies
- Prediction
  - Extrapolating the current properties of an entity based on historical authoritative data local guesses about the future
    - The opposite of *authoritative*

Common Approaches

- Lockstep
  - Deterministic / literal input passing
    - Turn taking, waiting for all clients to have sent input
  - Common for games with a strict split between input and simulation
    - RTS
- Reliable transport protocols (TCP)
  - Requires high bandwidth or simple networked state
  - High-game tolerance to latency
- Delay mitigation
  - Bandwidth Reduction
  - Entity Interpolation
  - Dead-Reckoning
  - Client Authority Prediction
  - Server-Side Lag Compensation

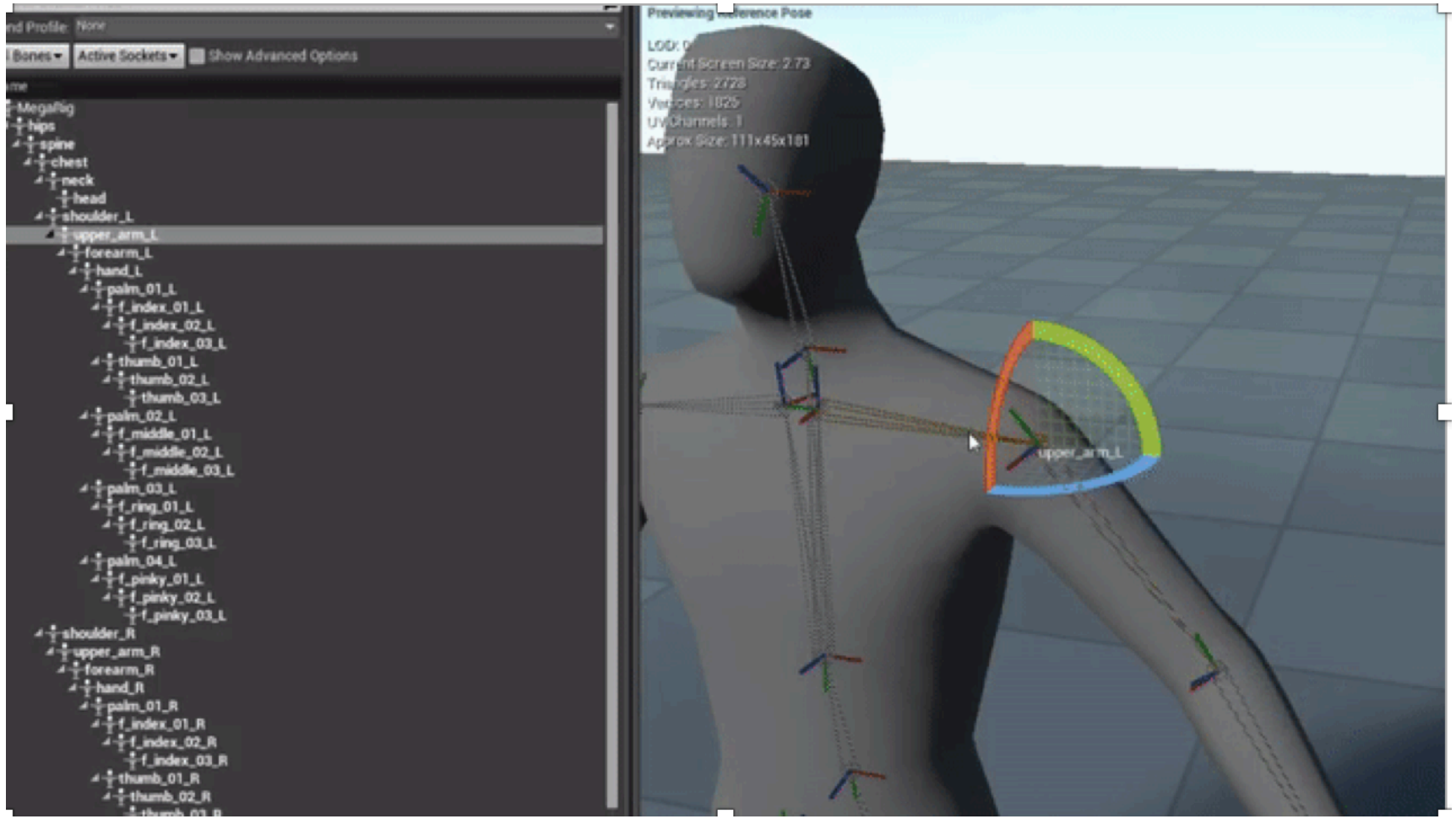

delays
View A
Controller A
Model A
View B
Controller B
Model B
View C
Controller C
Model C
Controllers
Model

Client-Side Replication

- Give each client its own replica of the model to use
  - The server remains the authority on what is happening
- The client
  - Receives regular snapshots of the server model
    - UDP (fast, but unreliable @~20Hz)
    - Ignore old / out-of-data packets
  - Has a local replica of the server's logic
    - Not a static snapshot, but includes logic regarding how the simulation should evolve
      - Local "model"
  - Between snapshots uses the local replica of the model to calculate the current state to render the view
  - Tells the server what we wish to performing, so the server can update the master model
    - The server is ***authoritative***
    - Further changes are cascaded to the client

Bandwidth Reduction

- What is the model and what is the view?
  - What does the server need to tell the client?
  - Position / orientation of underlying capsule
  - An animation has *started*
    - Vs continuously communicating at what point in the animation we are
- Optimise replication communication
  - Only tell the client what has changed, rather than send the full model snapshot
    - Reduce frequency of snapshots
    - {t=1, x { 0, 5 }, y { 5, 5 }}, {t=2, x { 0, 5 }, y { 10, 10 }}
    - {t=1, x { 0, 5 }, y { 5, 5 }}, {t=2, y { 10, 10 }}
  - Send full snapshot after a network delay so that the client gets back in sync

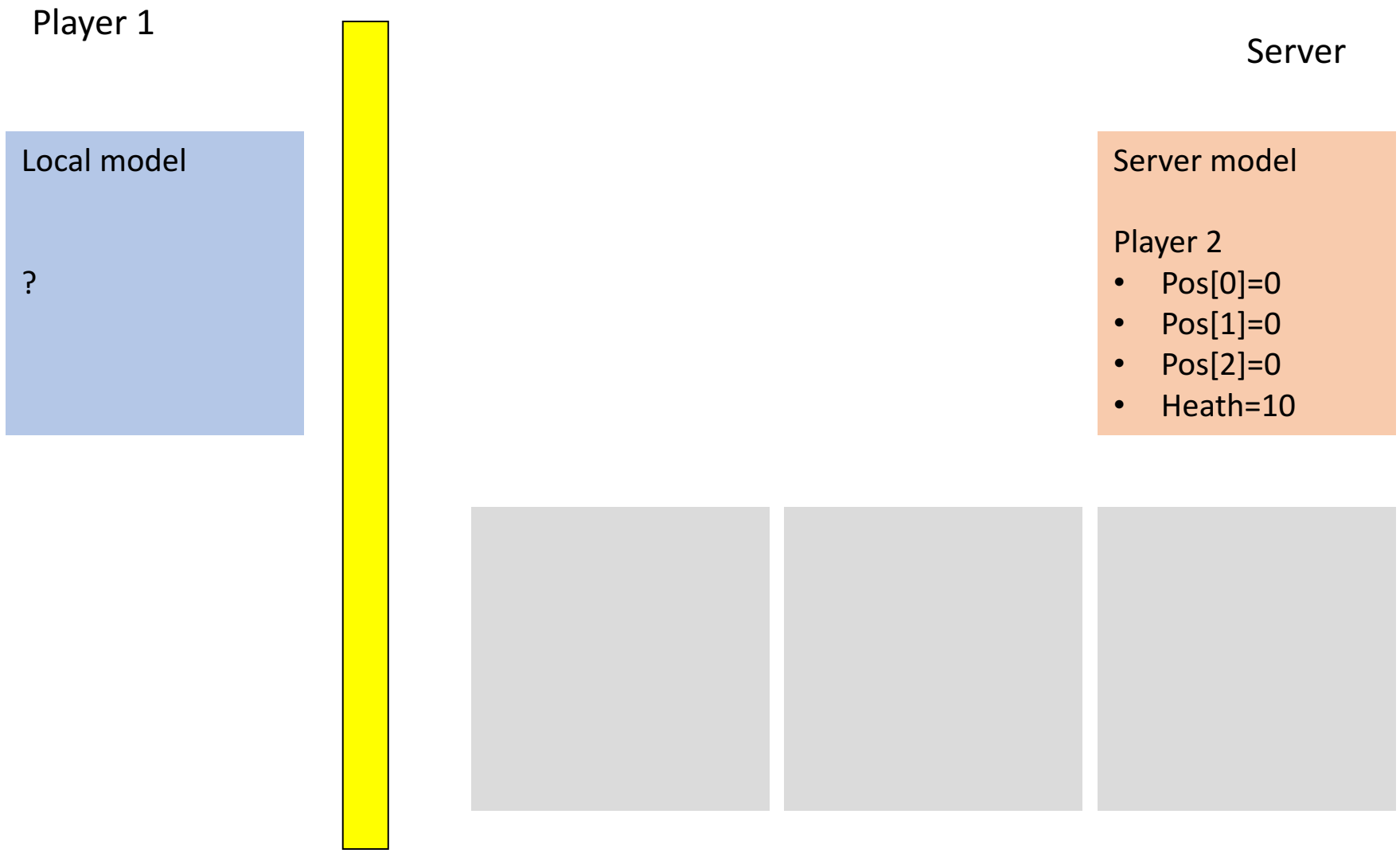

Profile: None
Bones
Active Sockets
Show Advanced Options
Megarig
hips
spine
chest
neck
head
shoulder_L
upper_arm_L
forearm_L
hand_L
palm_01_L
f_index_01_L
f_index_02_L
f_index_03_L
thumb_01_L
thumb_02_L
thumb_03_L
palm_02_L
f_middle_01_L
f_middle_02_L
f_middle_03_L
palm_03_L
f_ring_01_L
f_ring_02_L
f_ring_03_L
palm_04_L
f_pinky_01_L
f_pinky_02_L
f_pinky_03_L
shoulder_R
upper_arm_R
forearm_R
hand_R
palm_01_R
f_index_01_R
f_index_02_R
f_index_03_R
thumb_01_R
thumb_02_R
Previewing Reference Pose
Current Screen Size: 2.73
Triangles: 2728
Vertices: 1825
UV Channels: 1
Approx Size: 111x45x181
upper_arm_L

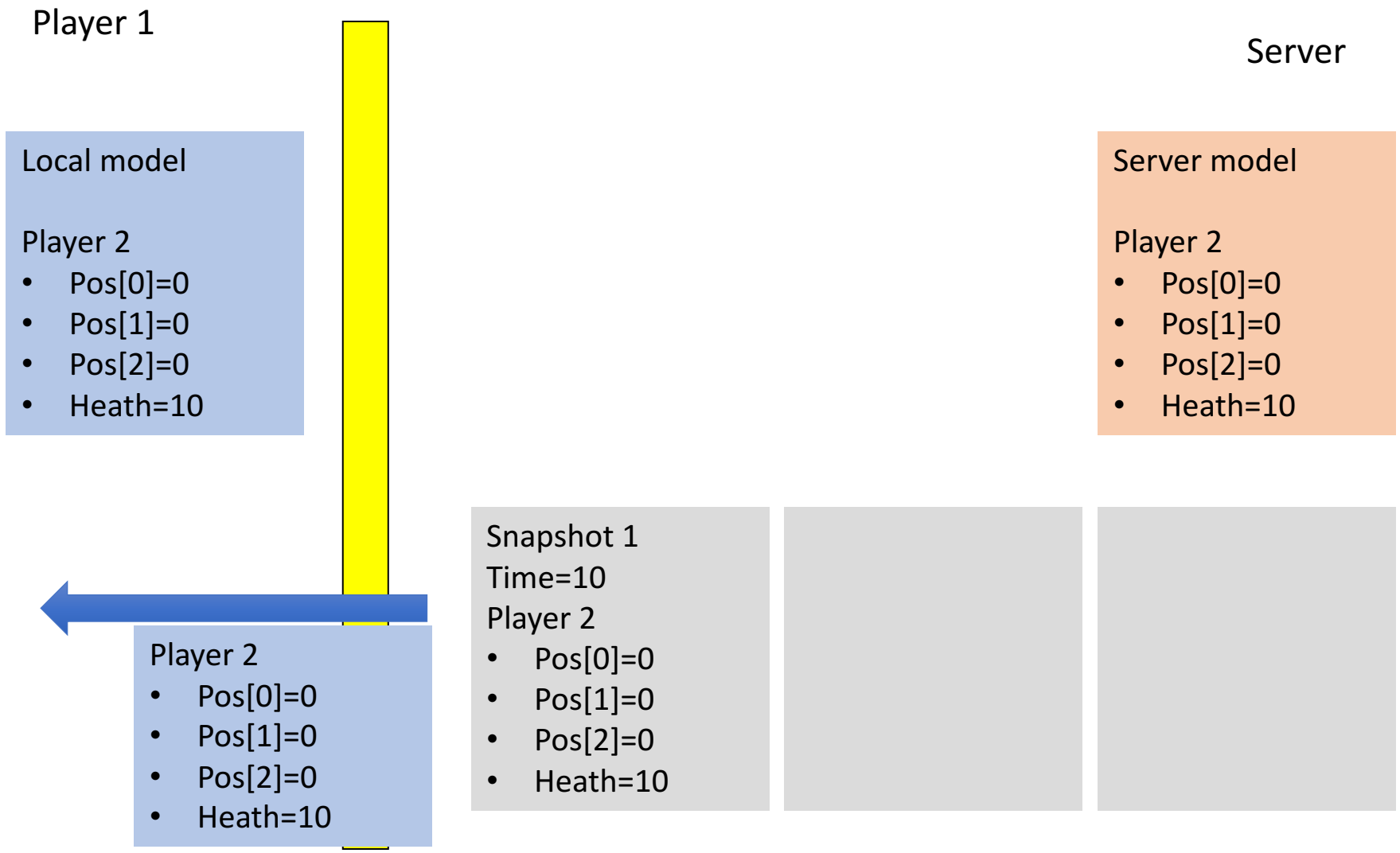
Entity Interpolation

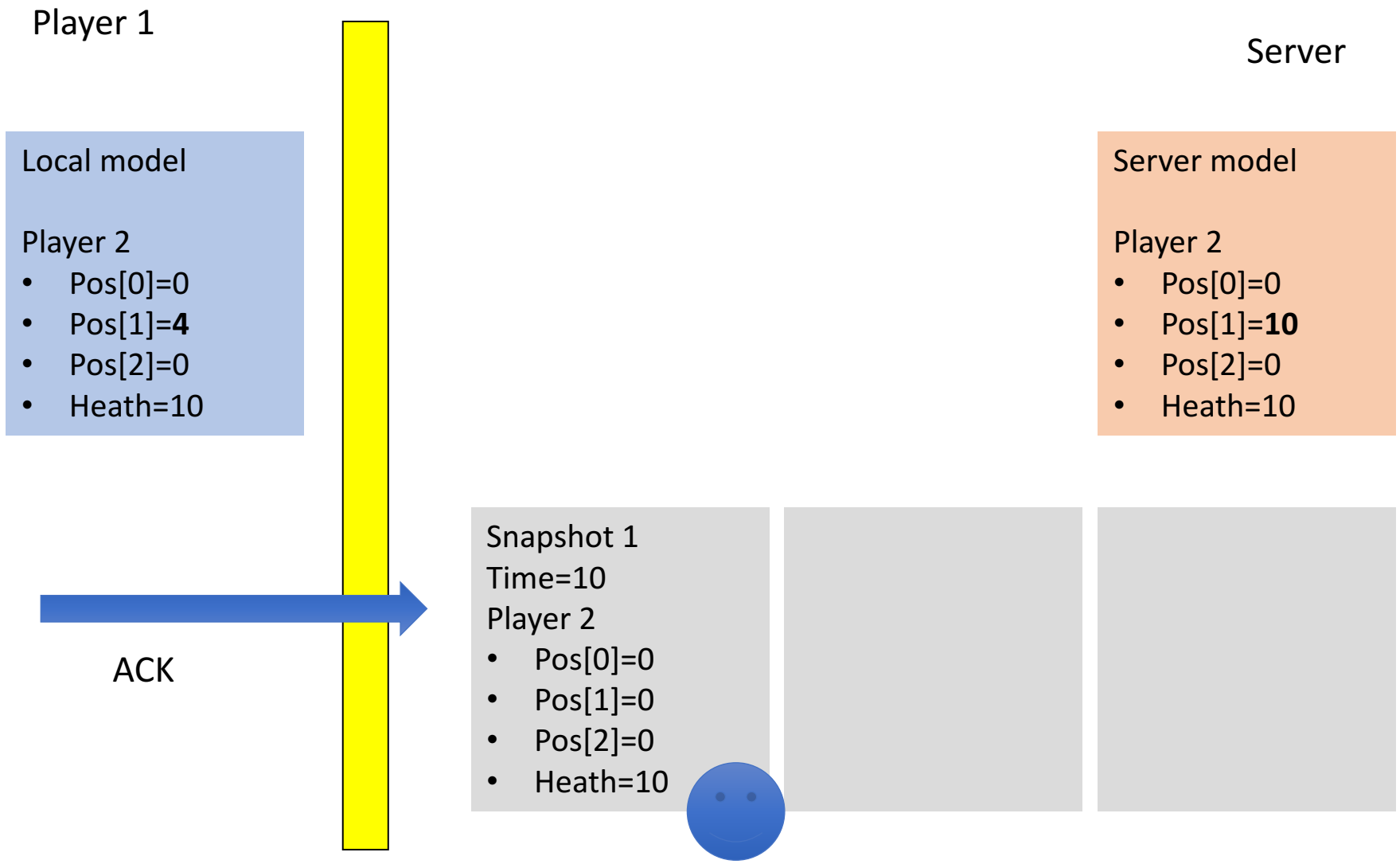
- Receive regular snapshots from the server
  - 20Hz
    - 1) Position of all objects at time t
    - 2) Position of all objects at time t+50ms
    - 3) Position of all objects at time t+100ms
    - …
  - Much slower than our rendering loop frequency
    - Objects would “jump” between snapshots
- Interpolate between past and current snapshots
  - When 2 is received
  - Calculate intermediate positions between 1,2
    - 1) Position of all objects at time t
    - 2) Position of all objects at time t+50ms
      - Move object to t+5
      - Move object to t+10… ← render smooth motion at 200fps

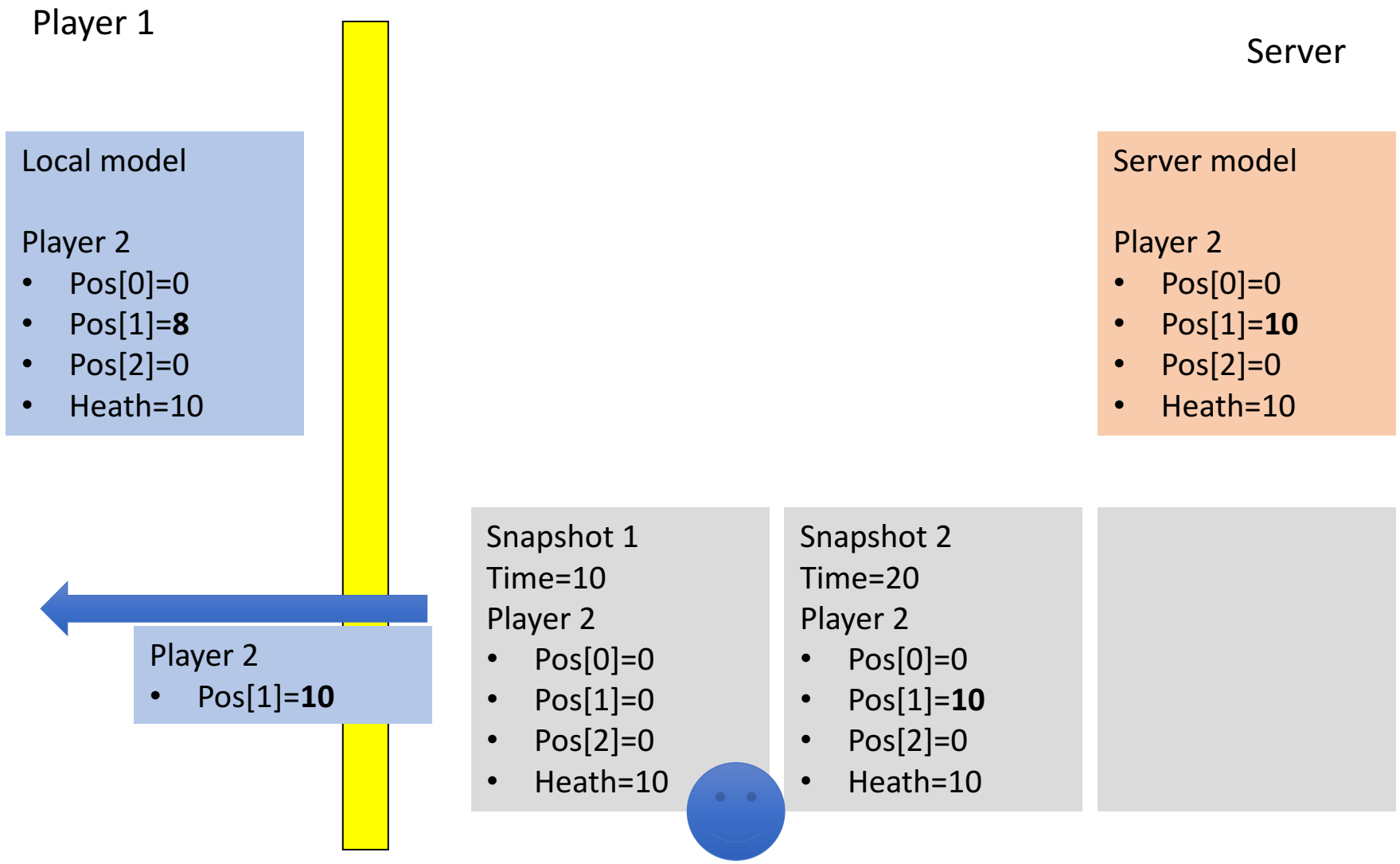
Dead Reckoning / Opponent Predication

- Interpolate / predict motion into the future using linear physics
  - Moving objects are likely to continue to move
  - Static objects are likely to continue being static
  - Make guesses about future states when appropriate
- Client is responsible for frame-to-frame movement simulation and rendering
  - Game critical events are comparatively *infrequent*
    - 100Hz vs ? (how often do you pick something up / shoot?)
  - X is moving at speed Y with direction Z, so move it a bit in direction Z
- Server is responsible for the bigger picture, the context and the consequences
  - Don’t predict significant events, wait for authoritative model
  - Player A has picked up the gold, B hasn’t
  - A has fired a gun and hit B


Player 1
Server
Local model
?
Server model
Player 2
• Pos[0]=0
• Pos[1]=0
• Pos[2]=0
• Heath=10


Player 1
Server
Local model
Player 2
• Pos[0]=0
• Pos[1]=0
• Pos[2]=0
• Heath=10
Server model
Player 2
• Pos[0]=0
• Pos[1]=0
• Pos[2]=0
• Heath=10
Player 2
• Pos[0]=0
• Pos[1]=0
• Pos[2]=0
• Heath=10
Snapshot 1
Time=10
Player 2
• Pos[0]=0
• Pos[1]=0
• Pos[2]=0
• Heath=10


Player 1
Server
Local model
Player 2
• Pos[0]=0
• Pos[1]=4
• Pos[2]=0
• Heath=10
Server model
Player 2
• Pos[0]=0
• Pos[1]=10
• Pos[2]=0
• Heath=10
ACK
Snapshot 1
Time=10
Player 2
• Pos[0]=0
• Pos[1]=0
• Pos[2]=0
• Heath=10


Player 1
Server
Local model
Player 2
• Pos[0]=0
• Pos[1]=8
• Pos[2]=0
• Heath=10
Server model
Player 2
• Pos[0]=0
• Pos[1]=10
• Pos[2]=0
• Heath=10
Player 2
• Pos[1]=10
Snapshot 1
Time=10
Player 2
• Pos[0]=0
• Pos[1]=0
• Pos[2]=0
• Heath=10
Snapshot 2
Time=20
Player 2
• Pos[0]=0
• Pos[1]=10
• Pos[2]=0
• Heath=10

Player 1

Server

Local model

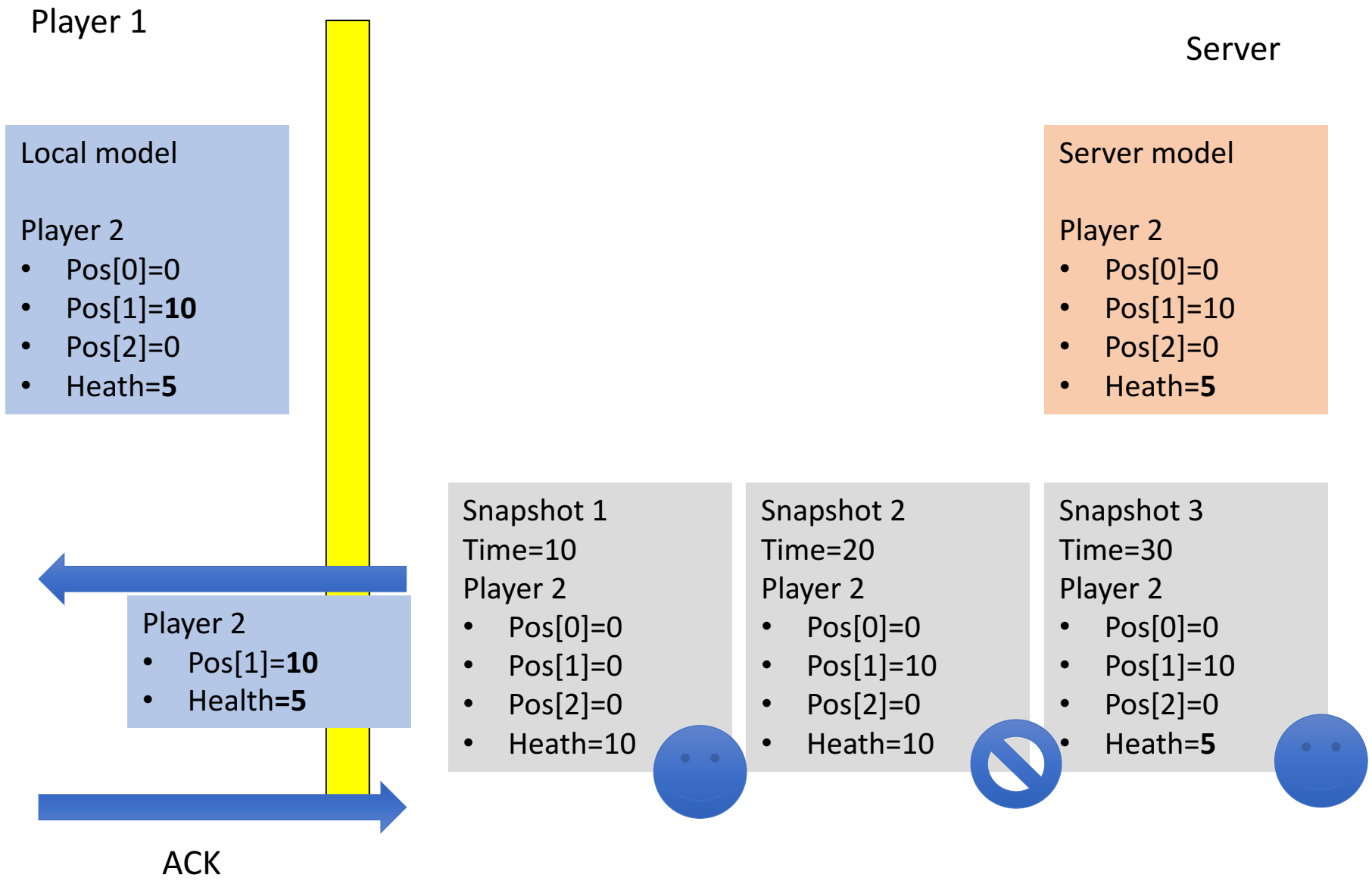
Player 2
- Pos[0]=0
- Pos[1]=**12**
- Pos[2]=0
- Heath=10

Server model

Player 2
- Pos[0]=0
- Pos[1]=10
- Pos[2]=0
- Heath=**5**

Snapshot 1
Time=10
Player 2
- Pos[0]=0
- Pos[1]=0
- Pos[2]=0
- Heath=10

Snapshot 2
Time=20
Player 2
- Pos[0]=0
- Pos[1]=10
- Pos[2]=0
- Heath=10

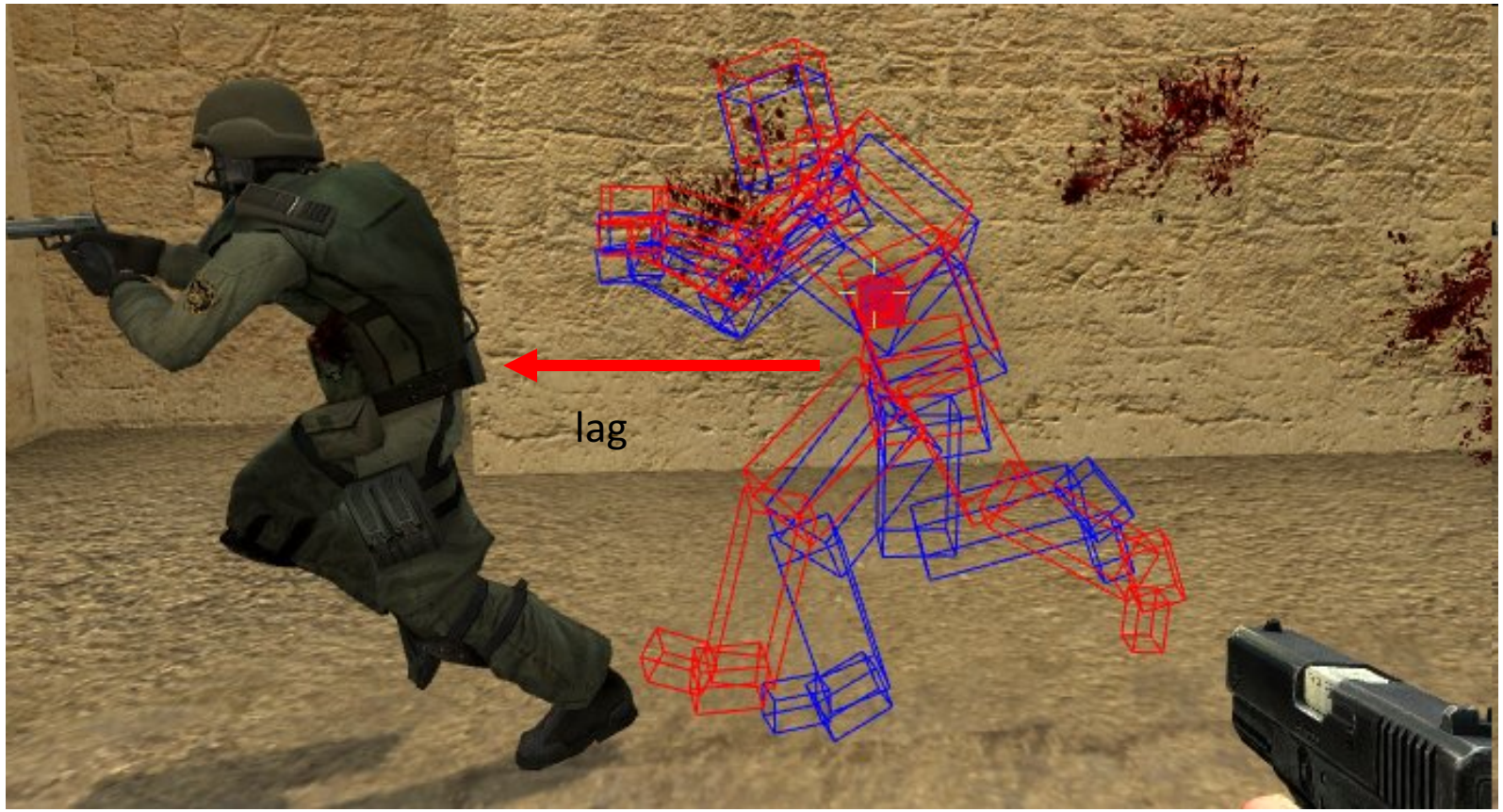

Player 1
Server
Local model
Player 2
• Pos[0]=0
• Pos[1]=10
• Pos[2]=0
• Heath=5
Server model
Player 2
• Pos[0]=0
• Pos[1]=10
• Pos[2]=0
• Heath=5
Player 2
• Pos[1]=10
• Health=5
Snapshot 1
Time=10
Player 2
• Pos[0]=0
• Pos[1]=0
• Pos[2]=0
• Heath=10
Snapshot 2
Time=20
Player 2
• Pos[0]=0
• Pos[1]=10
• Pos[2]=0
• Heath=10
Snapshot 3
Time=30
Player 2
• Pos[0]=0
• Pos[1]=10
• Pos[2]=0
• Heath=5
ACK

Client Authority Prediction

- The player pressed forwards
  - Round trip time
    - Client->Server->Server Game Loop->Client
    - Significant latency for “twitching”
      - onLive
- Assume that we will be given authority
  - That our local model will match future server model
- Send the command to the server controller
- Start moving immediately
- Rewind if we got it wrong
  - Generally most commands will be accepted as the model should be largely in sync
- …the game feels more responsive than the network allows
  - Atomic movements smoother than 20Hz network rate

Server-side Lag Compensation

- Server maintains
  - A per-client stack of snapshots
    - Remember where everything was up to a second ago
  - Current round-trip-time for all the clients
    - Time for a packet to travel from client->server->client
- When a new command is received, estimate when it was **sent** rather than when it was **received**
- Use the historical snapshot of the model to work out what the client state was at this time
- …the command is executed hopefully as if there was no lag
  - Assumes that report of latency is correct


lag

Networking design strategies

- Which parts of the gameplay need to be adjudicated by a single authority?
  - I.e. what can be allowed to be briefly out of sync
  - More adjudication = greater consistency
    - Game changing events
      - Deliver reliably via TCP
  - Less adjudication = more prediction, responsiveness
    - Immediate movements
      - Delivery unreliably, quickly via UDP
- How is the lag hidden?
  - Prediction, interpolation, dead reckoning
- Change game mechanics to improve networking
  - Players doing something at exactly the same time is going to cause problems
    - Can’t heavily rely on highly synchronized state / view

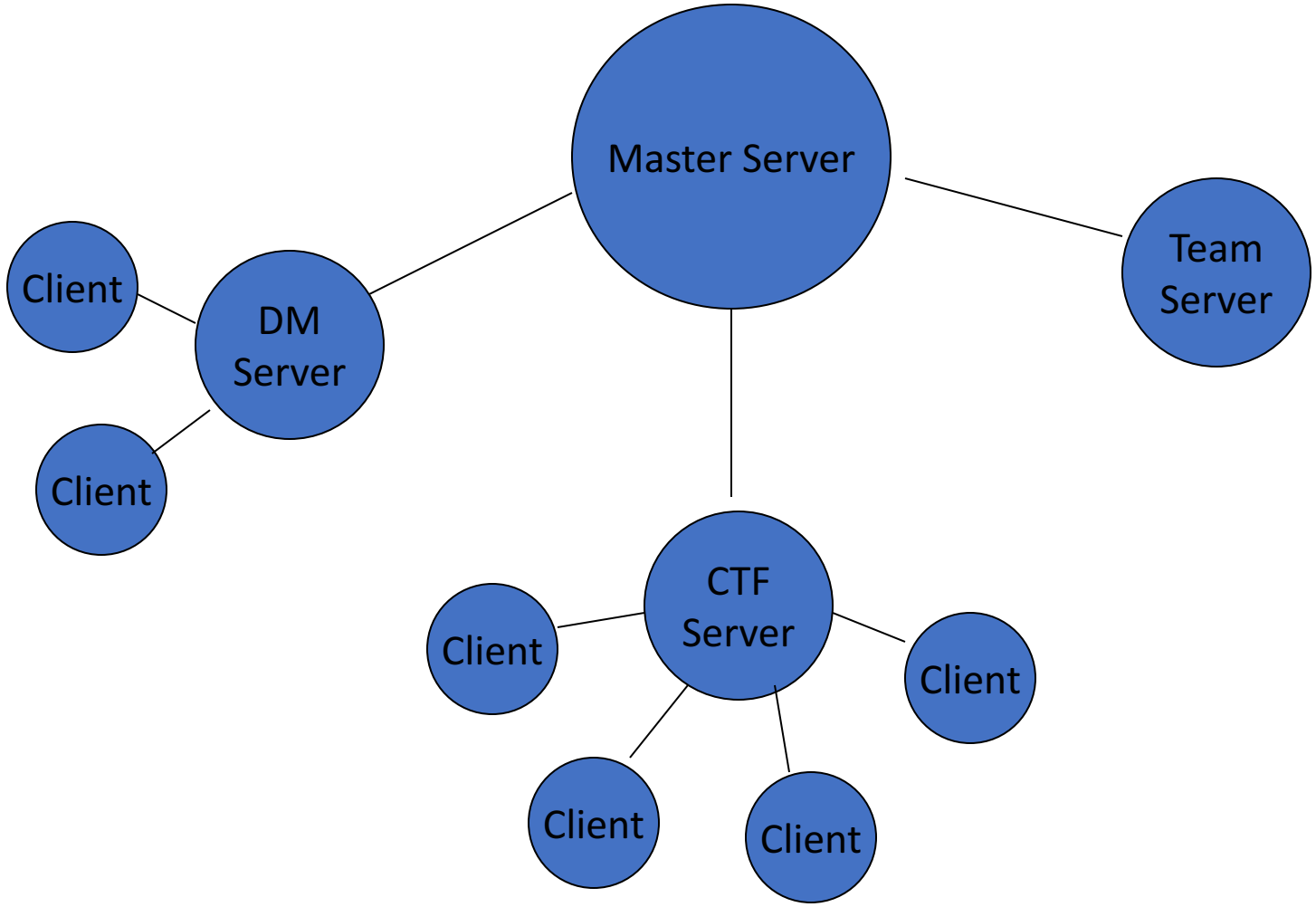
Scalability and Discovery

- Basic client / server model
  - Is reasonable for a small numbers of players
  - 2 to 128 simultaneous clients
- Not applicable to the whole player-base
  - Fortnite
    - 3.2 million concurrent
  - Second Life (at least as of 2010)
    - 600 square miles of land
    - 390TB of content
    - Average 10000 players at any one time?
  - World of Warcraft
    - 11 million paying subscribers peak
- Strategy
  - Divide players explicitly / covertly into manageable simulations

Explicit Scalability

- Advertise individual “game instances”
  - Servers running a particular simulation
  - Distributed geographically
  - Choose based on
    - Game type
    - Reported connection latency
    - Number of players
- Lobbies
  - A master server maintains a browse-able list of all active servers that a player can join
    - Known (hard coded) address
    - Directory system
  - Individual servers register themselves with the master server

Lobby

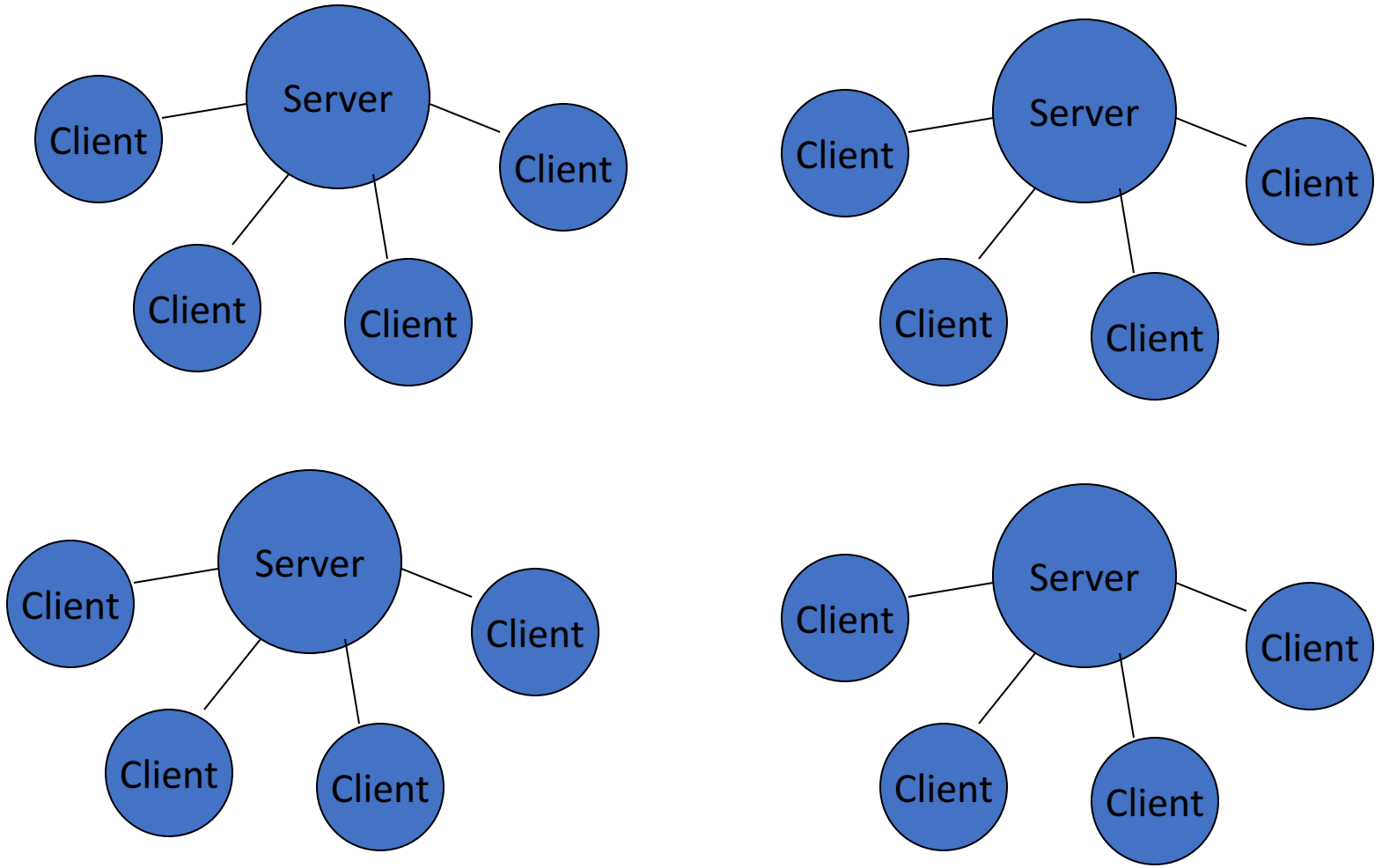

Master Server
Team Server
DM Server
Client
Client
CTF Server
Client
Client
Client
Client

1.	CRACKED SERVER by K7	JOIN	34/50		108.61.210.253:28961	mp_dawnville
2.	[CRACKED]ALLWeapons~Verindra~Public	JOIN	26/48		148.251.81.10:28574	mp_chateau
3.	|MM|Playground!	JOIN	29/40		69.31.134.189:28960	mp_harbor
4.	porno.SerVer /Cracked!	JOIN	23/26		84.200.210.95:28910	mp_pavlov
5.	[Nulled]|www.recod.ru| COD 1.4	JOIN	15/32		81.198.135.160:27795	mp_carentan
6.	EURO RIFLES S&D	JOIN	24/40		94.23.147.114:28960	mp_harbor
7.	[Nulled]|www.recod.ru| COD 1.2	JOIN	23/37		81.198.135.160:28930	mp_depot
8.	`'^Surv!voRxTDM! visit us at : www.survivor-gamerz...	JOIN	16/30		46.105.34.98:28963	mp_harbor
9.	`'^Surv!voRxHarBoR! Visit us at : www.survivor-gam...	JOIN	21/30		46.105.34.98:28960	mp_HarBoR
10.	*CLUB DE AMIGOS*(MG)DM	JOIN	8/16		190.210.176.248:28960	mp_rocket
11.	porno.SerVer /Cracked	JOIN	17/22		84.200.210.95:28900	mp_carentan
12.	`Play DM Harbor 24/7	JOIN	11/24		178.62.195.247:28960	mp_harbor
13.	Creat!ve RIFLES S&D	JOIN	14/32		94.23.147.114:28971	mp_stalingrad
14.	=[BM]=bluemonkeyAWE	JOIN	7/24		69.28.210.30:28960	mp_carentan
15.	[Rifles]|www.recod.ru| COD 1.2	JOIN	8/39		81.198.135.160:27770	mp_harbor

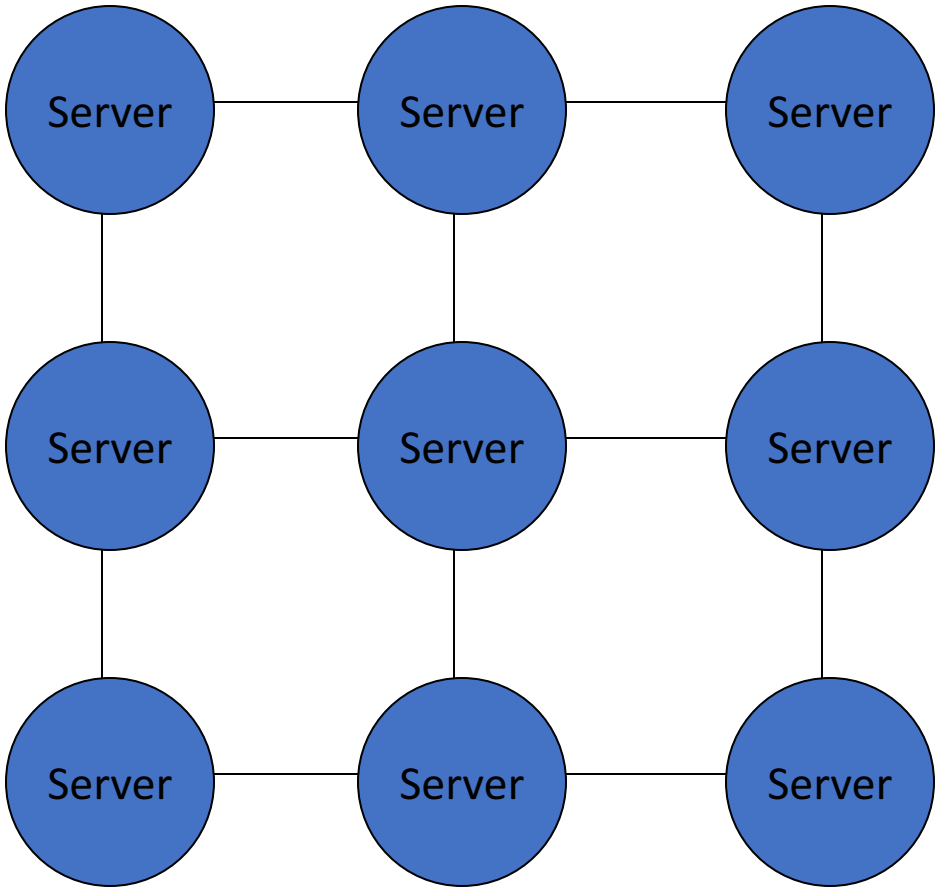
Hidden Scaling

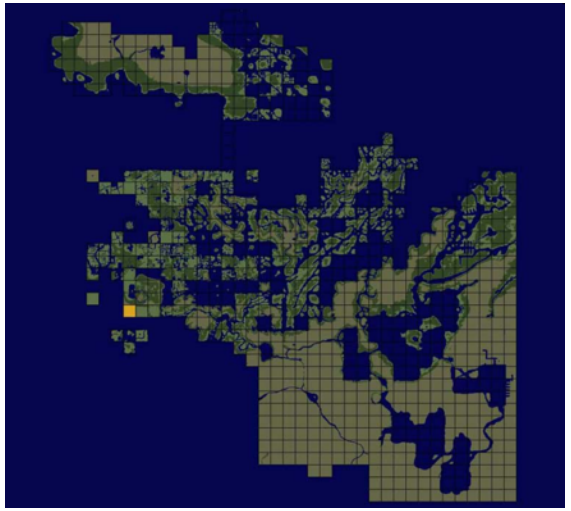
- Moderated Awareness
  - Server(s) maintains the complete model for all players
  - Client receives only a subset of the model relevant to the player
  - Where the player is and what is nearby
    - Geographical segmentation
    - Load segmentation
    - What is visible
- Shards
  - Multiple identical instances of the server model
  - Each player inhabits only one
  - Create new instances as required
- Make the game bigger
  - Scale up
    - Add more server hardware
  - Scale out
    - Spinning up more instances does not affect the clients

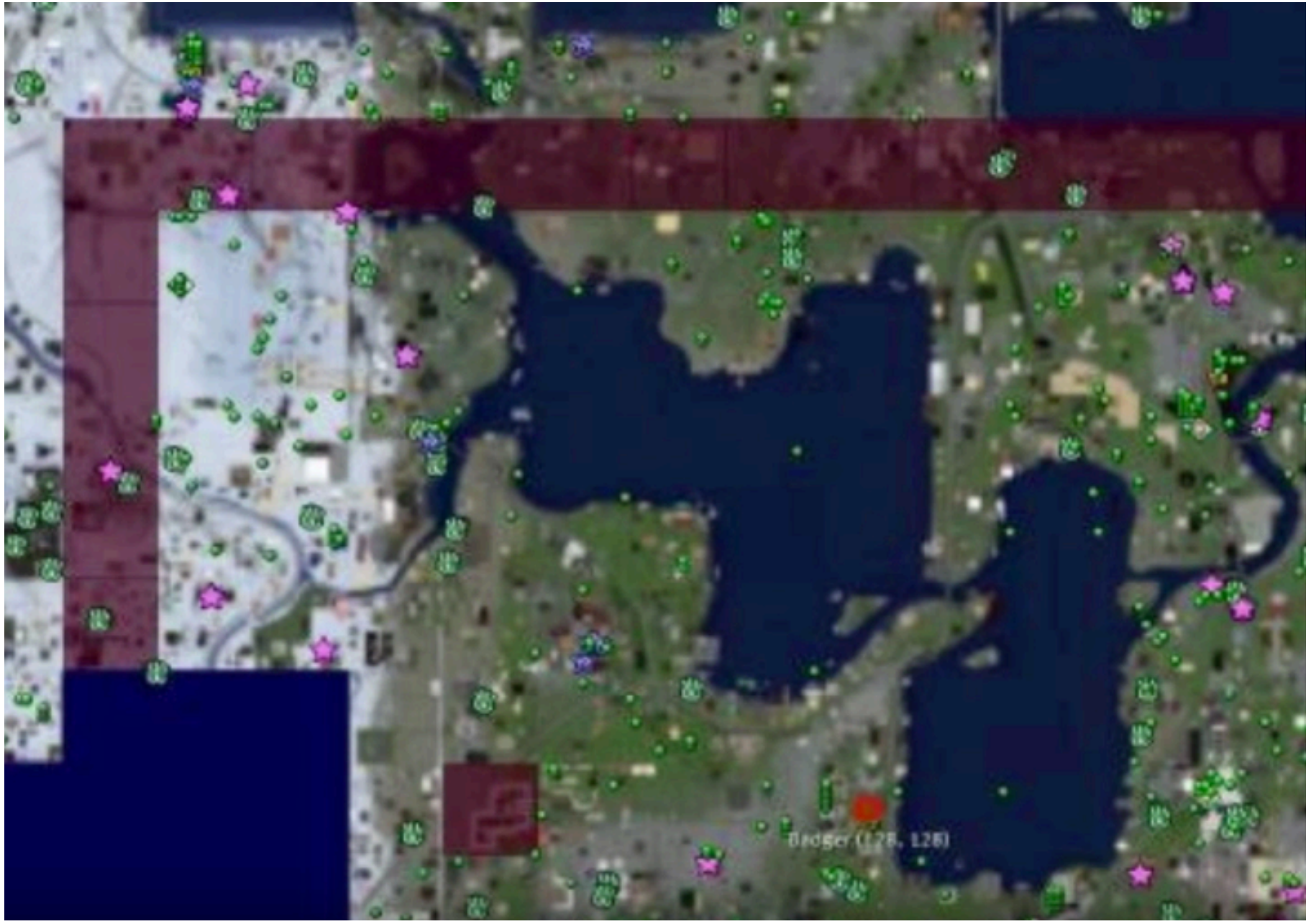
Shards


Server
Client
Client
Client
Client
Server
Client
Client
Client
Client
Server
Client
Client
Client
Client
Server
Client
Client
Client
Client

Second Life Grid *Locales*






Badger (128, 128)

Eve Online – Time Dilation

start game

while(true)

{

how much time has elapsed?

get user input

simulate game world(**elapsed time)**

resolve collisions

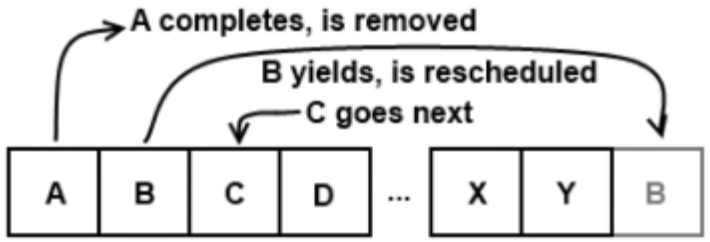
move objects

sleep(desired-elapsed time)

}

exit

Simulation > Real-time



Different Attitudes to the Game

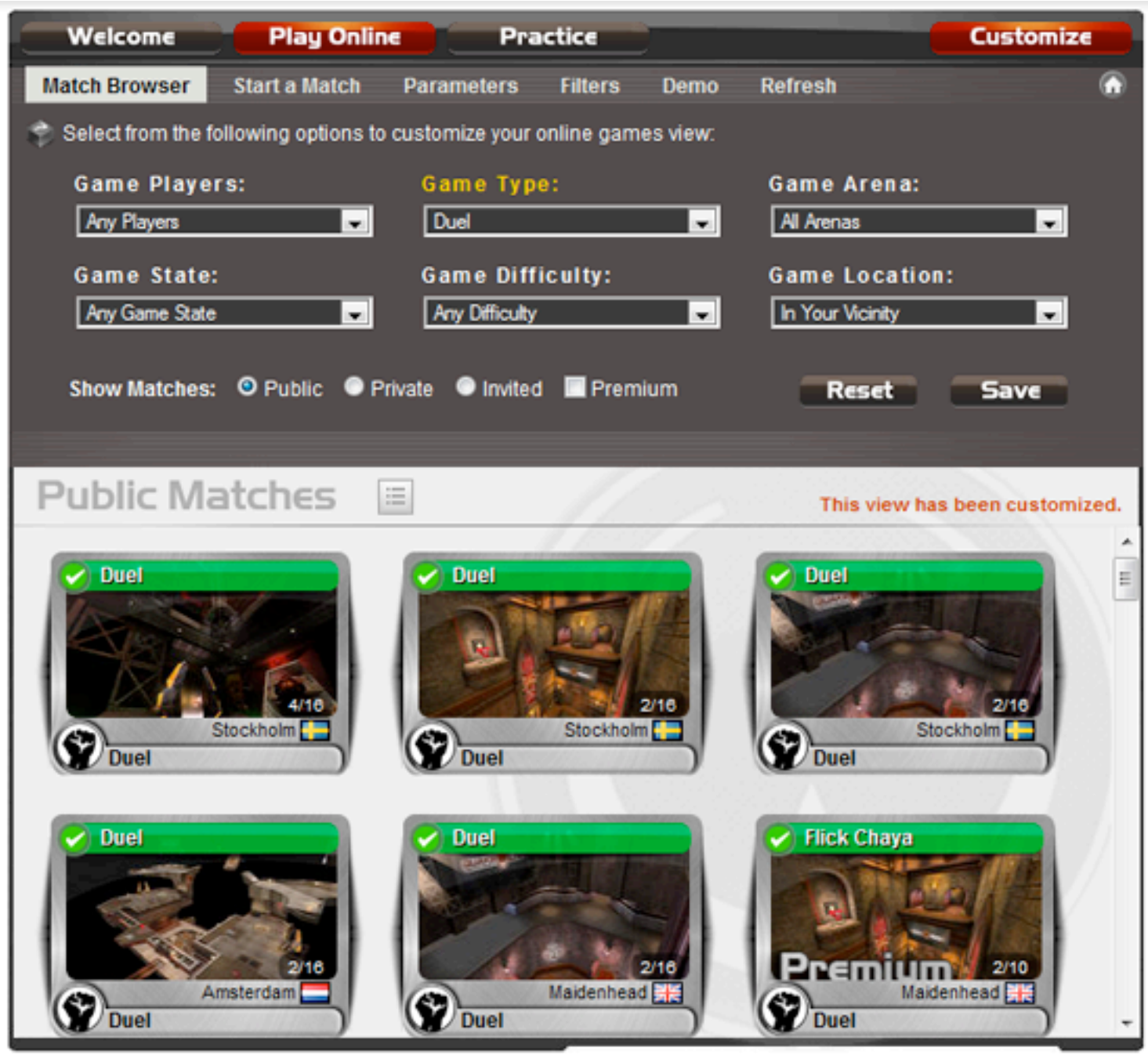
- Degree of lusory attitude
  - Taking pleasure in playing the game
  - Accepting that rules lead to limitations
  - Accepting the rules
    - Play is an end in itself
- Relationship to the rules
  - How much the player adheres to the rules
    - Operational – the formal rules
    - Implicit – the social rules
  - Acknowledges the authority of the rules
    - Can/should they be broken?
- Interest in winning
  - How much the player wants to win

Types of Players

- Standard Player
  - Acknowledges authority of the rules
  - Typical interest in winning
- Dedicated Player
  - Extra-zealous lusory attitude
  - Special interest in understanding and mastering the rules
  - Intense interest in winning
- Unsportsmanlike Player
  - Sometimes dedicated, sometimes a cheat
  - Adheres to operational rules, violates implicit rules
  - Intense interest in winning
- Cheat
  - Pretends to possess lusory attitude
  - Violates operational rules in secret
  - Intense interest in winning
- Spoil-Sport
  - No pretense about lack of lusory attitude
  - No interest in adhering to any rules
  - No interest in winning

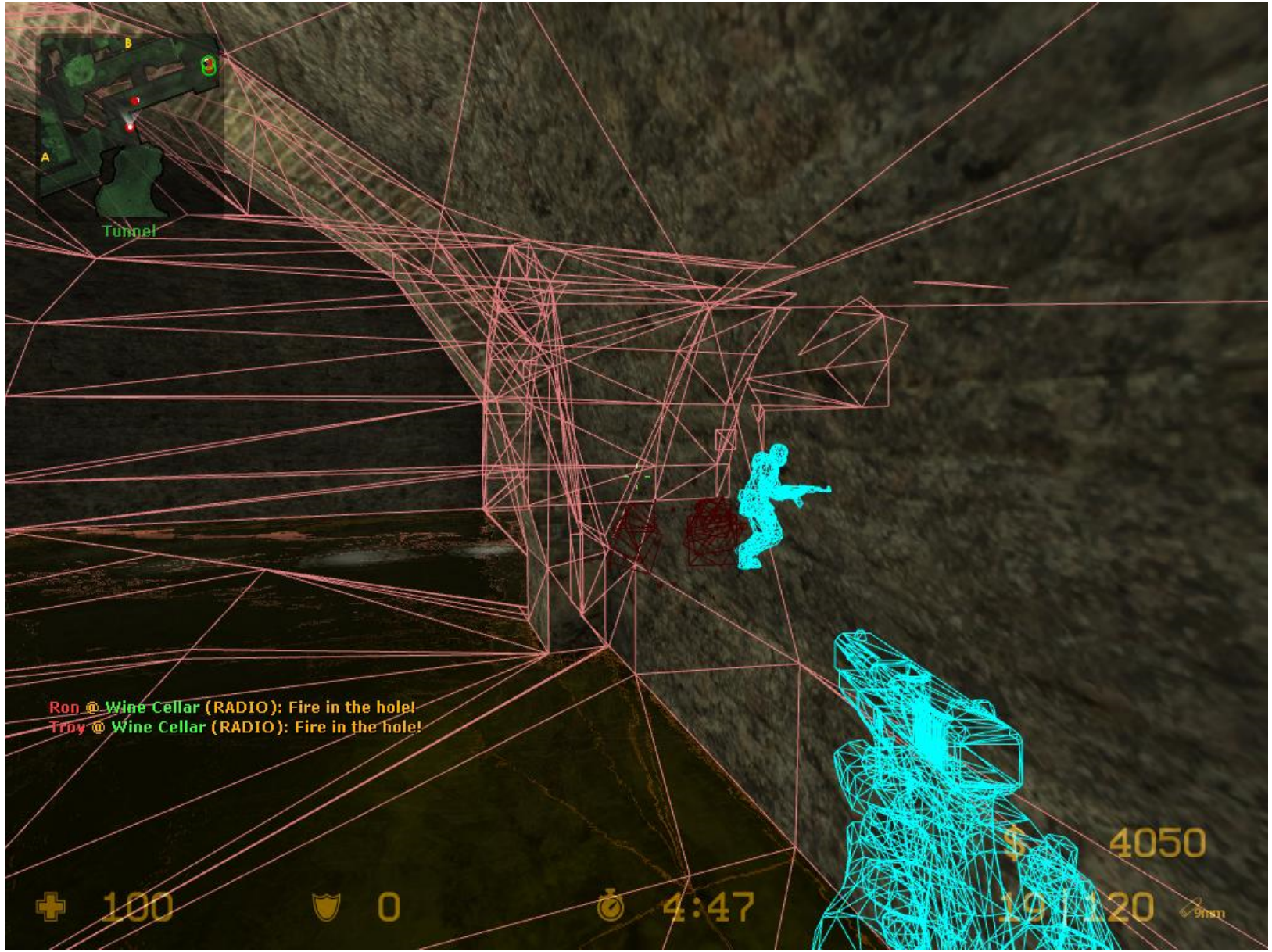
Breaking the Rules

- A significant issue in online games
- Making use of dominant strategies in order to win
  - Trick jumping, camping, weapon imbalances, short cuts
- Standard player vs dedicated player
  - Imbalanced due to skill
    - Playing below your level
  - Automatic skill determination and restrictions


Welcome
Play Online
Practice
Customize
Match Browser
Start a Match
Parameters
Filters
Demo
Refresh
Select from the following options to customize your online games view:
Game Players:
Any Players
Game Type:
Duel
Game Arena:
All Arenas
Game State:
Any Game State
Game Difficulty:
Any Difficulty
Game Location:
In Your Vicinity
Show Matches:
Public
Private
Invited
Premium
Reset
Save
Public Matches
This view has been customized.
Duel
4/16
Stockholm
Duel
Duel
2/16
Stockholm
Duel
Duel
2/16
Stockholm
Duel
Duel
2/16
Amsterdam
Duel
Duel
2/16
Maidenhead
Duel
Flick Chaya
Premium
2/10
Maidenhead
Duel

Breaking Operational Rules

- Reflex augmentation
  - Circumvent user-input moving and aiming
  - Aimbots
    - Simulate user input
    - Control and automate repetitive tasks
- Information exposure
  - Gain access to information the player is not entitled to
    - Opponents health, resources
    - Extracted from the client’s memory
  - Remove walls, fog, foliage
- Protocol level cheats
  - Insert, destroy, modify packets
  - Suppress client’s updates, receive server snapshots


B
A
Tunnel
Ron @ Wine Cellar (RADIO): Fire in the hole!
Troy @ Wine Cellar (RADIO): Fire in the hole!
100
0
4:47
4050
120

Breaking Implicit Rules

- Multiple concurrent sessions
  - Players are not who they appear to be
- Killing team members
  - ...lead to “friendly fire” mechanics
- Collusion, spying for the other team
- Subverting the game economy (“gold farming”)
- Disrupting the play of other players via in game mechanics
- Formalise implicit rules as operational rules
  - Violations are punished within the scope of the game
  - Killing a team-mate = lose a point
  - Gold farming
    - Violates EULA (“operational rule”)
    - Account is banned

Reducing Rule Breaking

- Technical Measures
  - **Never** trust the client
    - Consider authoritative distinctions
  - Attempt to hinder the use of bots / client hacks
    - PunkBuster / The Warden / Valve Anti-Cheat System et al search the clients memory for known hacks, scripts, behaviours
    - Impossible to completely stop client hacking
    - Occasionally punishes innocent players
    - Arguably overly invasive
  - Ban offenders by email, IP address, MAC address, device ID, credit card
    - Increased efficacy due to integrated services such as Steam, Xbox Live, PSN
- Social Measures
  - Player empowerment
    - Allow players to vote out or mute offensive players
  - Employ **moderators** to police the game

References

- http://developer.valvesoftware.com/wiki/Source_Multiplayer_Networking
- https://www.gdcvault.com/play/1014345/I-Shot-You-First-Networking
- https://www.youtube.com/watch?v=0OiamBxxoXA
- https://www.youtube.com/watch?v=WAh-6nEHwqI