the gamedesigninitiative at cornell university

Lecture 9

Networking

Networking Breaks into Two Phases

Matchmaking

- Service to find other players
  - Groups players in a session
  - But does not run session
- Why make your own?
  - Control user accounts
  - Implement skill ladders
- 3^{rd} party services common
  - Apple GameCenter
  - Google OpenMatch
  - CUGL Docker Service

Game Session

- Service to run the core game
  - Synchronizes player state
  - Supports minor adds/drops
- Why make your own?
  - Must tailor to your game
  - You often have no choice
- Limited 3^{rd} party services
  - Often just a networking API
  - For limited class of games
  - **Examples**: Unity, Unreal

Networking Breaks into Two Phases

Matchmaking

- Service to find other players
  - Groups players in a session
  - But does not run session
- Wh…
  - …plement skill ladders
- 3rd party services common
  - Apple GameCenter
  - Google OpenMatch
  - CUGL Docker Service

Simplify if possible

Game Session

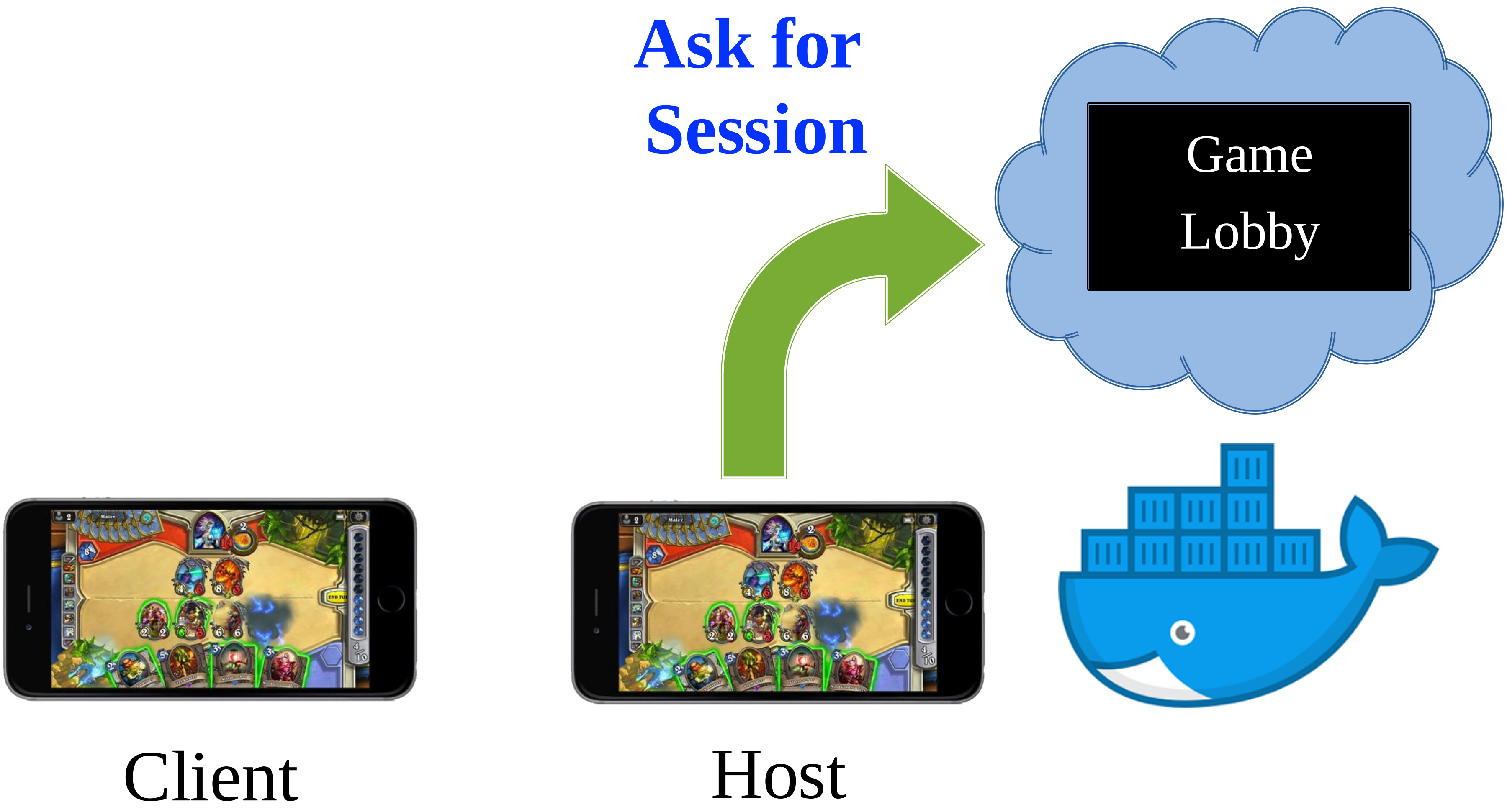
- Service to run the core game
  - Synchronizes player state
  - Supports minor adds/drops
- Wh…
  - …e
  - …often have no choice
- Limited 3rd party services
  - Often just a networking API
  - For limited class of games
  - **Examples**: Unity, Unreal

Our main focus

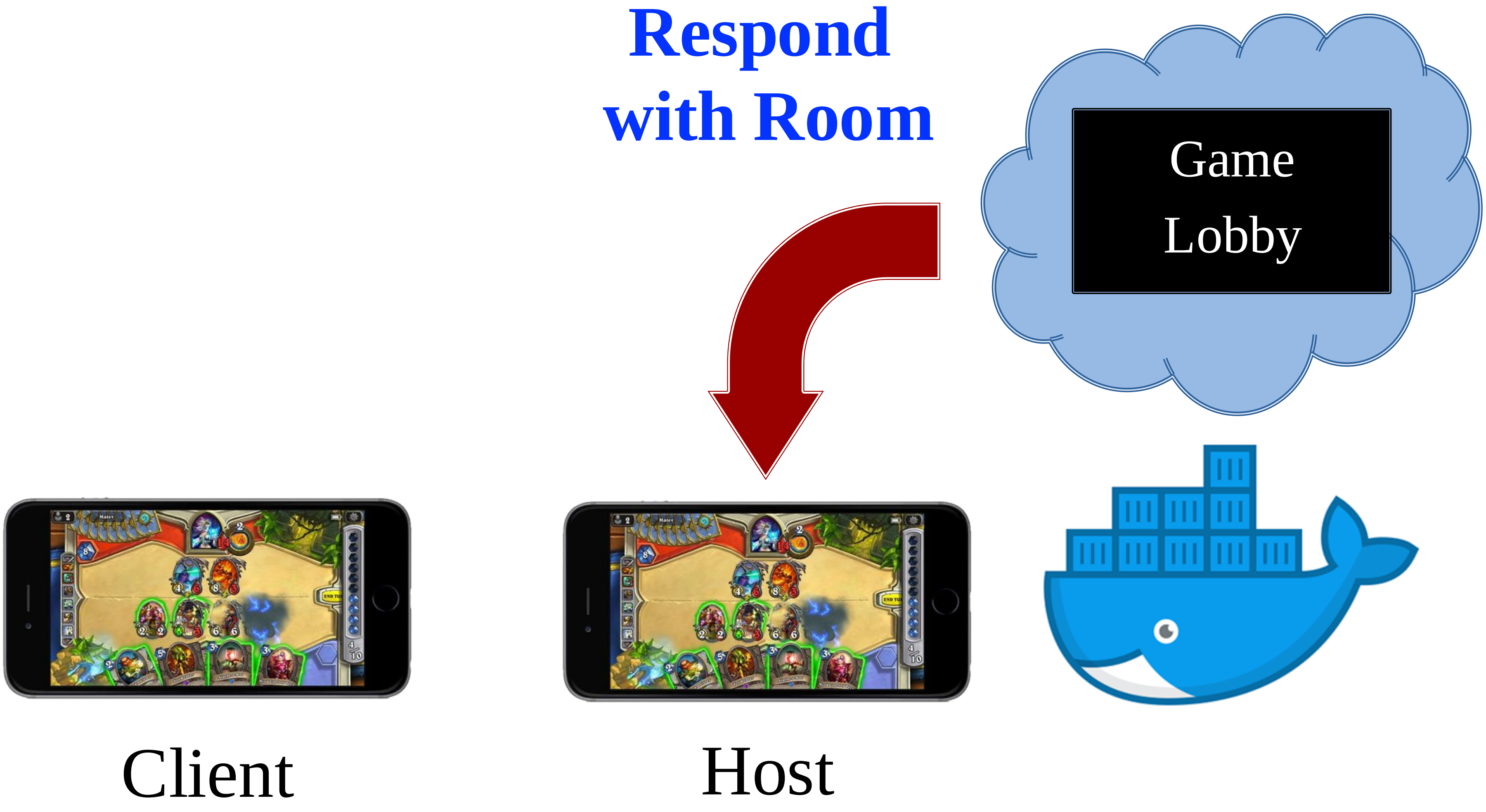
CUGL Matchmaking

- Requires a custom server
  - Needs a fixed IP address
  - IP is coded into the game
  - Or at least put in an asset
- Can leverage **cloud tech**
  - Write a Docker container
  - Deploy only as needed
- **Benefit**: cross-platform play
  - Must for iOS-Android play
  - See also Open Match

CUGL Matchmaking

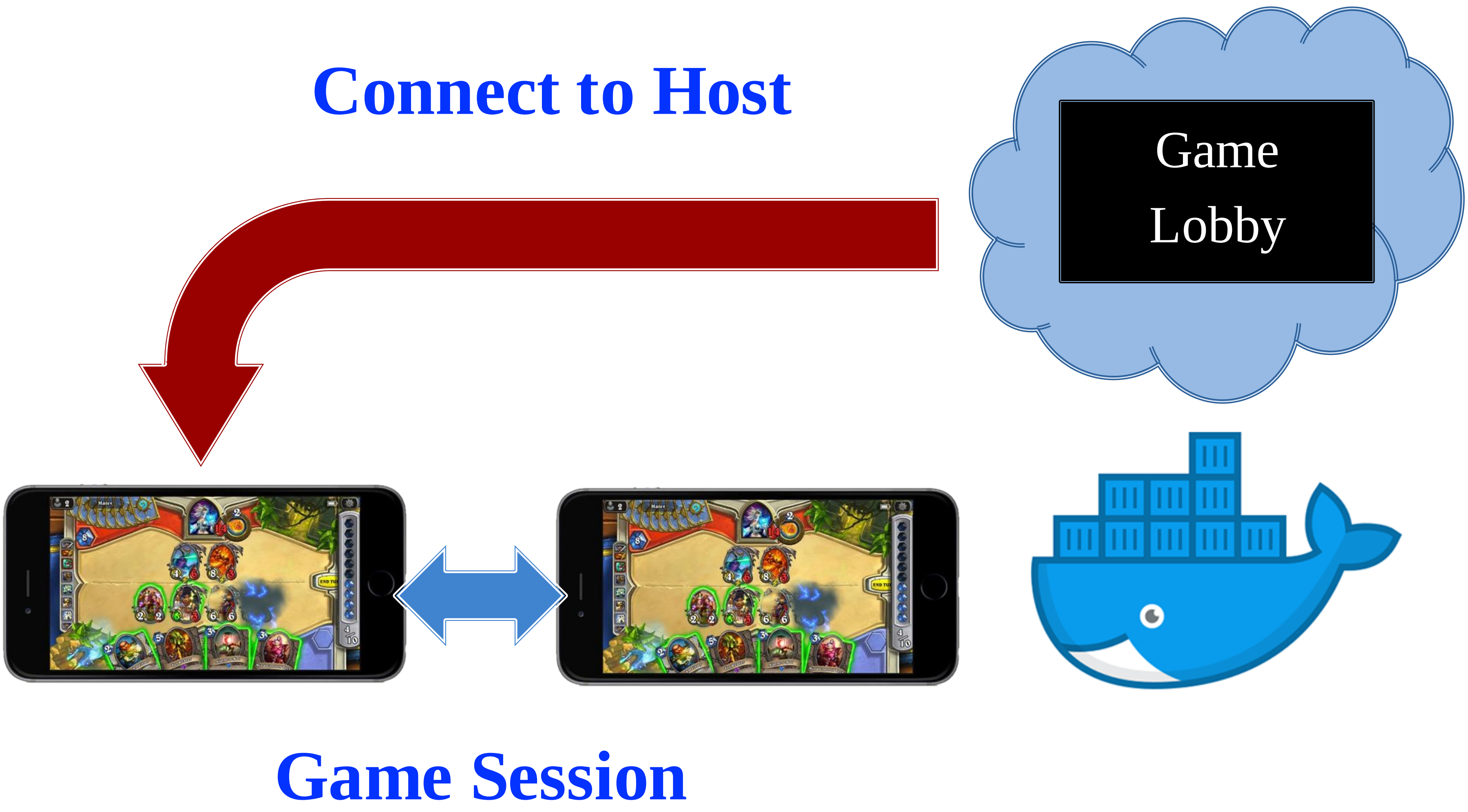


CUGL Matchmaking



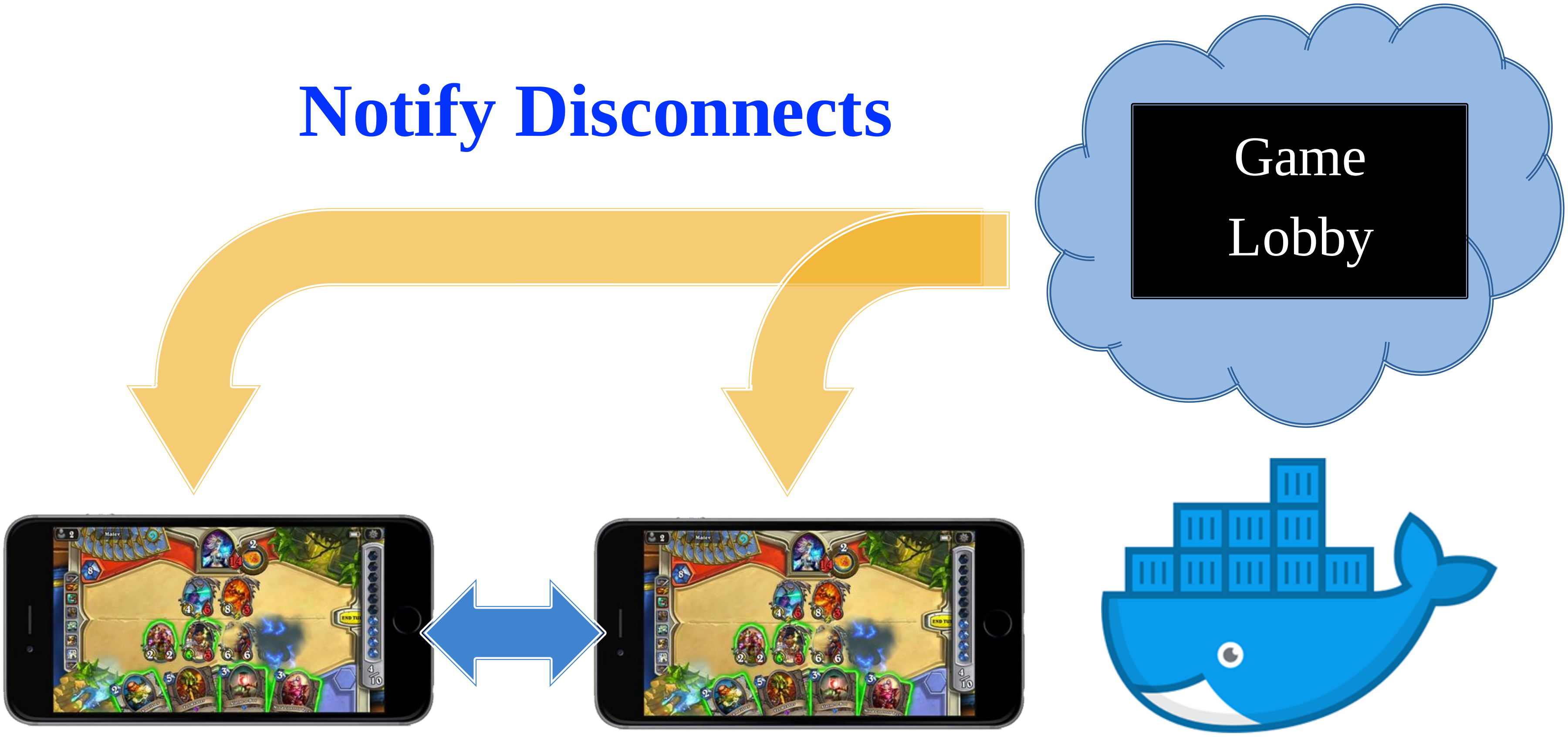
CUGL Matchmaking

CUGL Matchmaking



CUGL Matchmaking

Notify Disconnects



Game Session

Matchmaking in *Family Style*

Why Not Just Direct IPs?

- **Idea:** Just let the host be “the server”
  - Player starts up server instance
  - Player writes down their IP address
  - Everyone else types in that IP address
- **Problem:** Network Address Translation
  - Most networks use NAT to attach many devices
  - This means IP addresses on NAT are not real
- Lobby provides **NAT punchthrough**!
  - Reason why you keep it open for reconnects

NAT Punchthrough: STUN Server

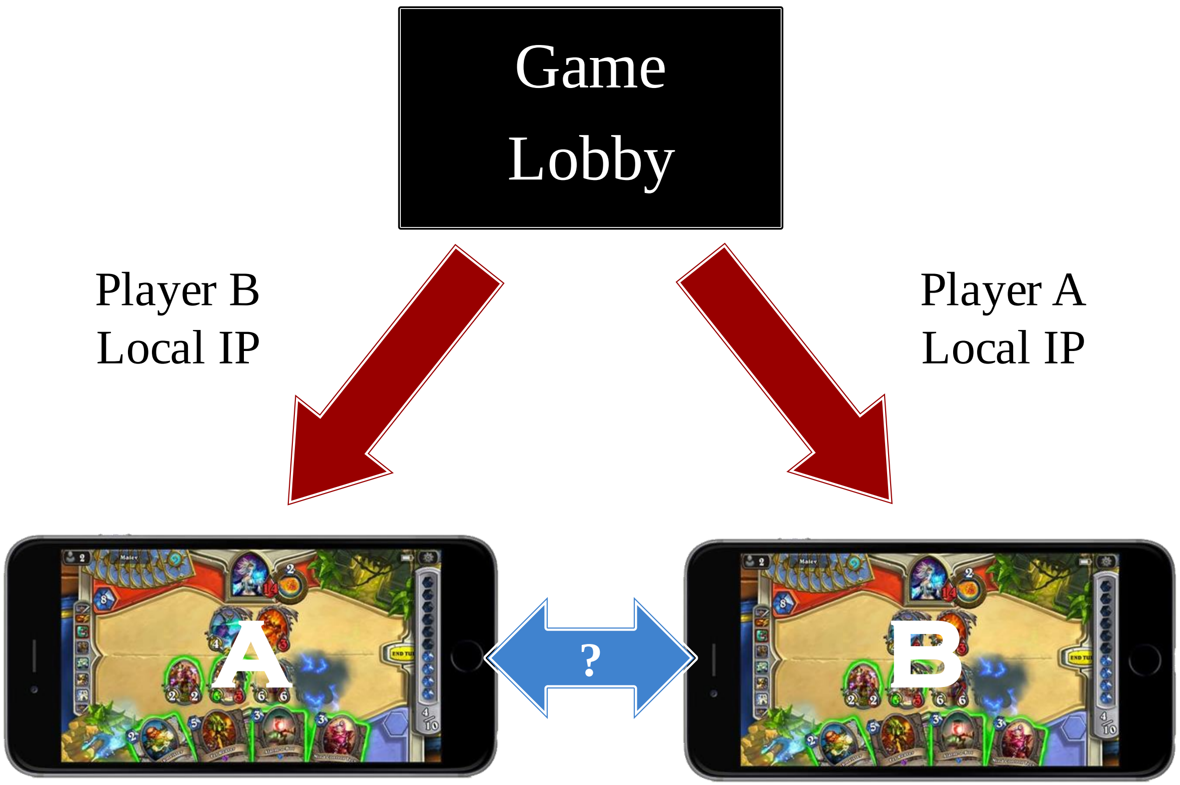
Game Lobby

STUN Server

Player A Local IP

Player B Local IP

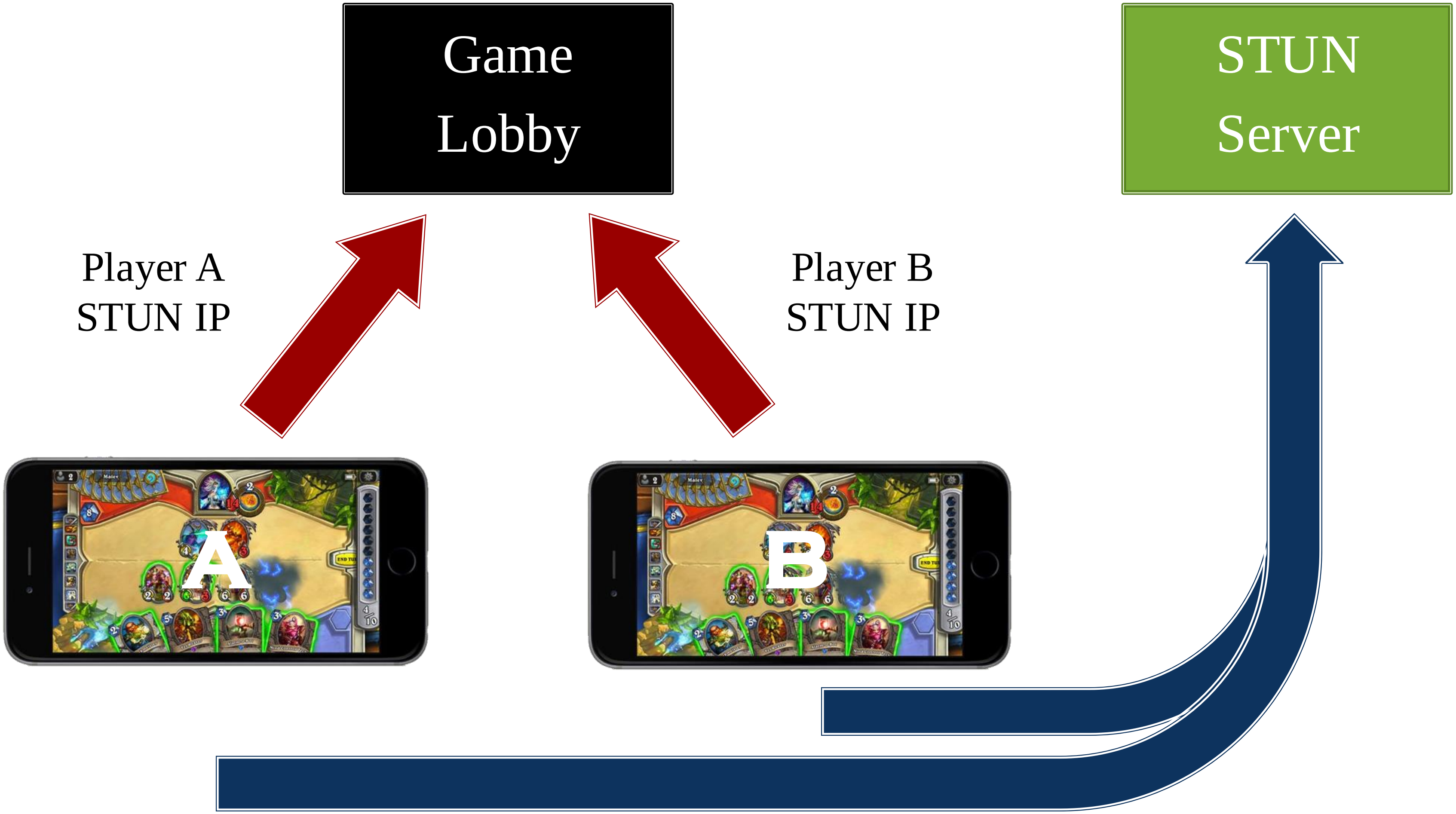
NAT Punchthrough: STUN Server



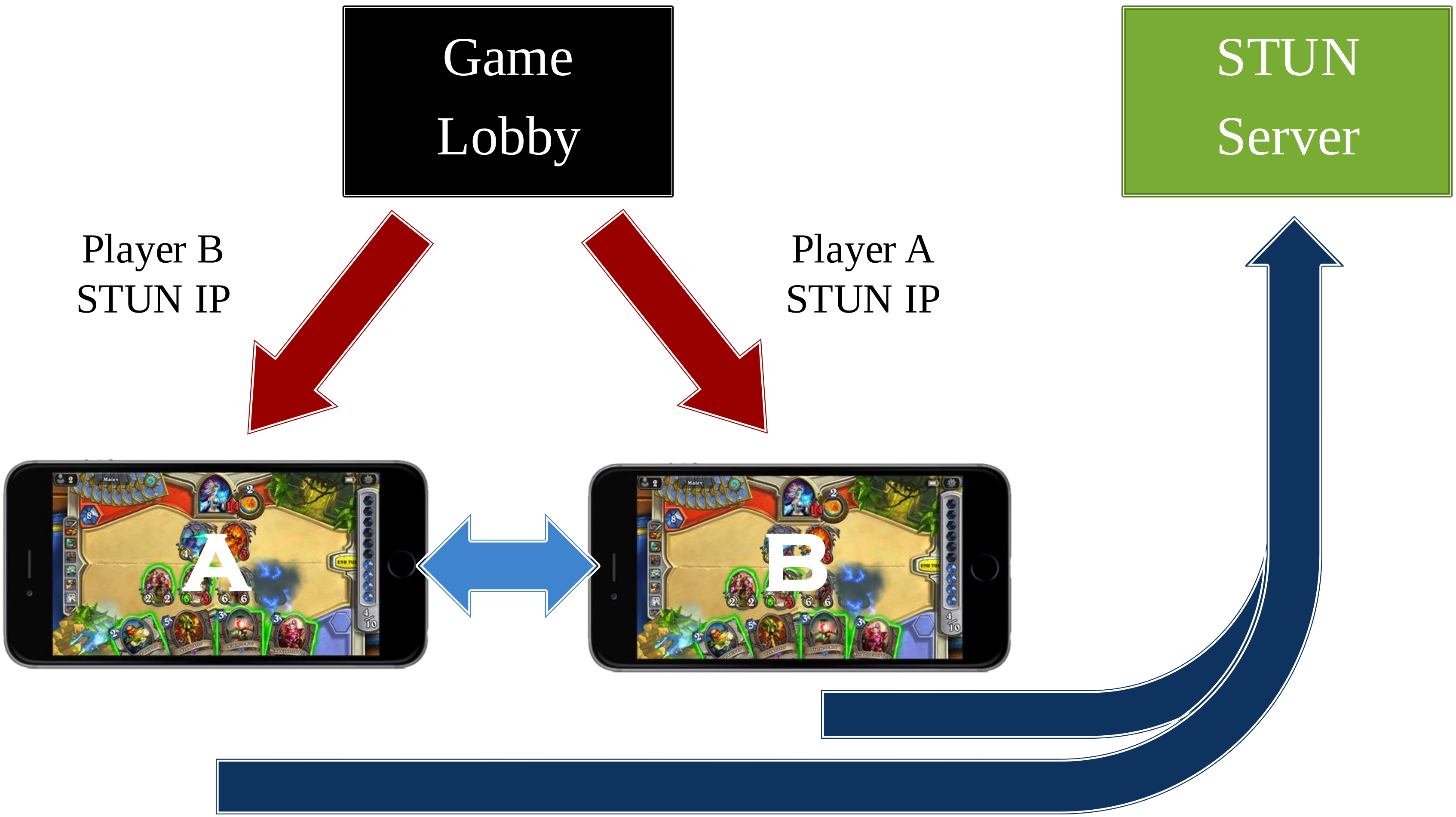
STUN
Server

What if not on same network?

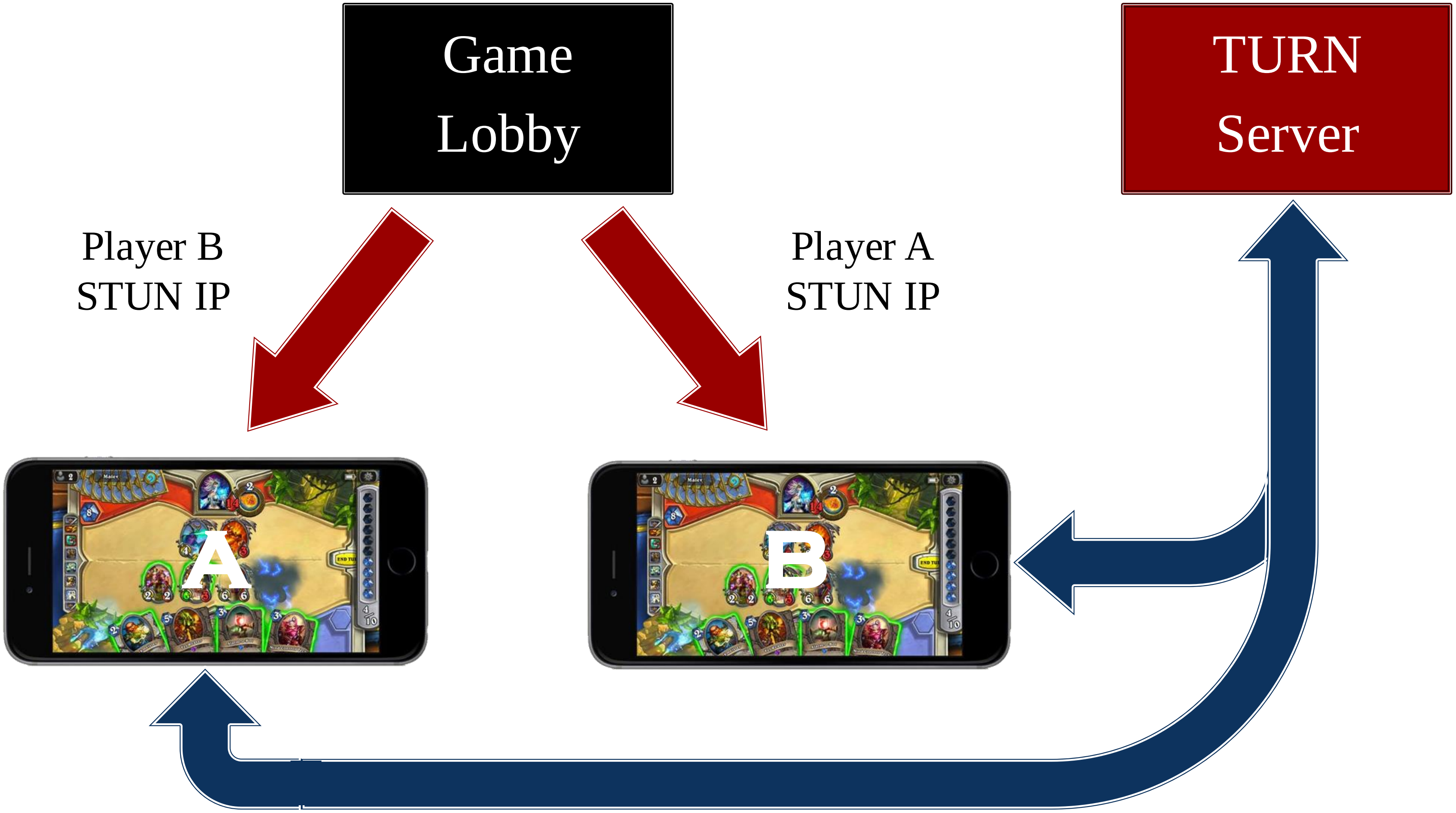
NAT Punchthrough: STUN Server



NAT Punchthrough: STUN Server



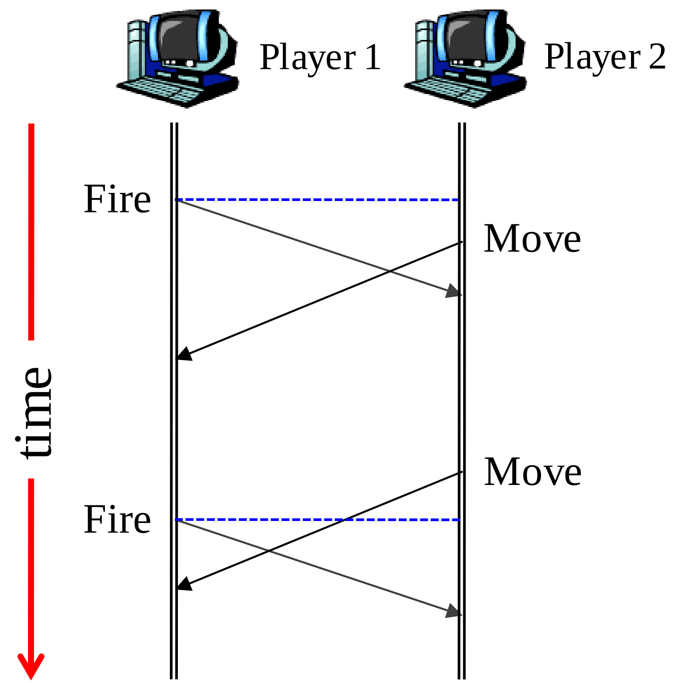
Extreme Firewalls: TURN Server



the gamedesigninitiative at cornell university

Game Session: Consistency

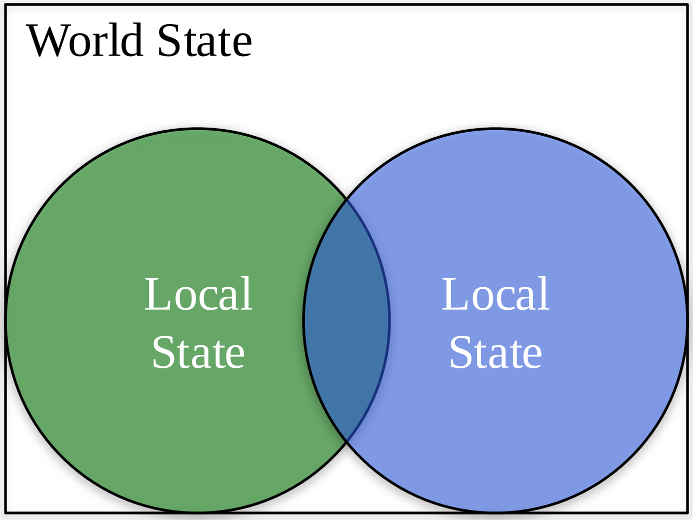
- *Latency* is root of all evil
  - **Local** actions are instant
  - **Network** actions are slow
- **Example**: targeting
  - Want "geometric fidelity"
  - Fire a weapon along ray
  - Hits first object on ray
  - But movement is fast!



How to tell these cases apart?

World State vs. Local State

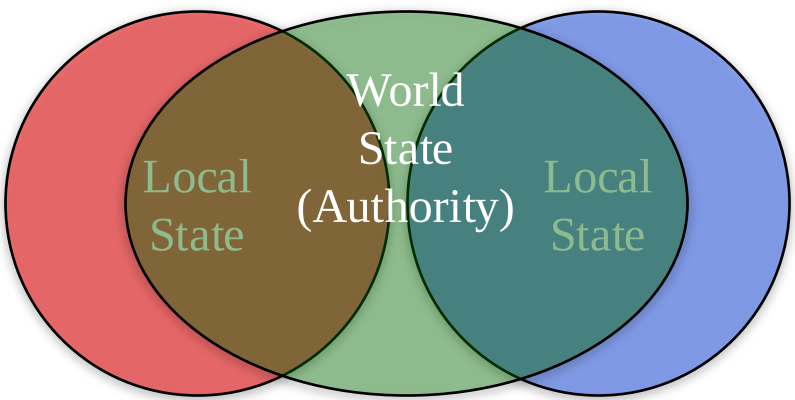
- **State**: all objects in game
  - **Local State**: on a machine
  - **World State**: "true" state
- *Where* is the world state?
  - On a single machine?
  - Union of local states?
- States may be *inconsistent*
  - Local disagrees with world
  - Is this really a problem?
  - What can we do about it?



The Question of Authority

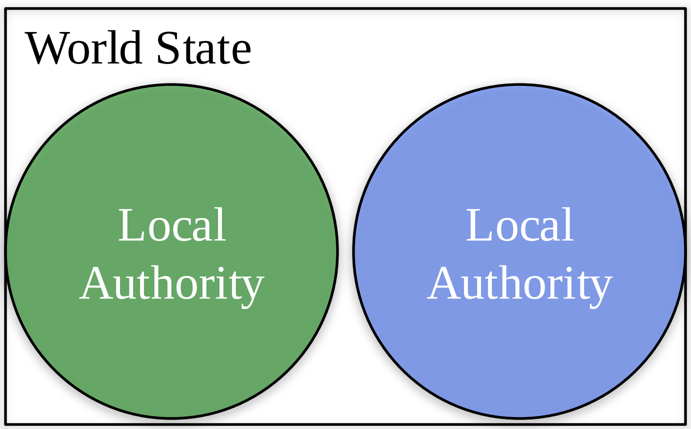
Centralized Authority

- One computer is authority
  - Stores the full world state
  - Local states must match it
- Often call this the "server"

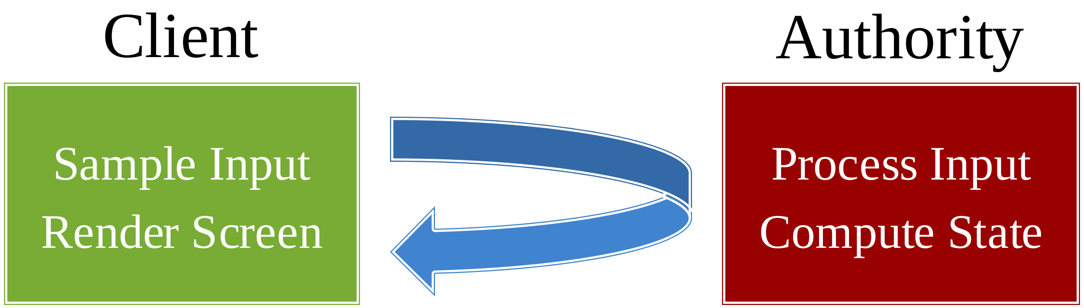


Distributed Authority

- Authority is divided up
  - Each object has an owner
  - Must match if not owner
- Classically call this "P2P"

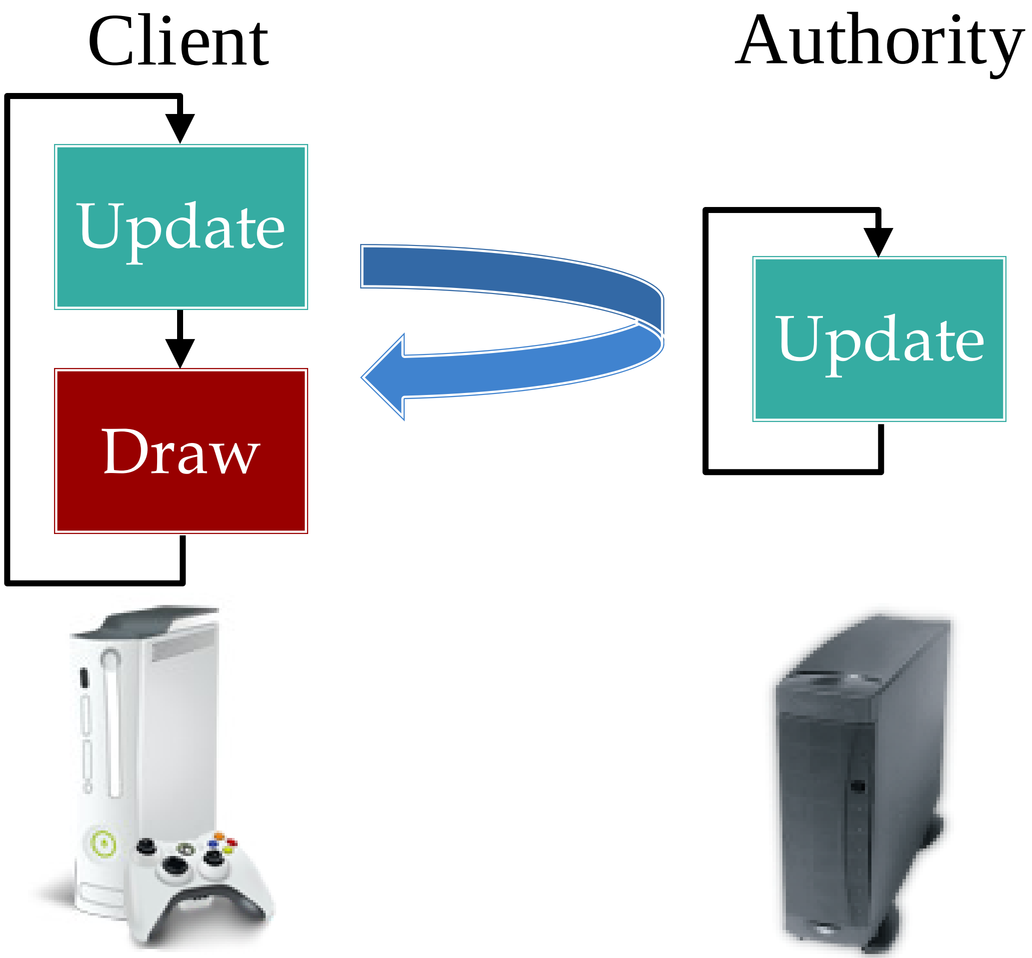


Authority and Latency

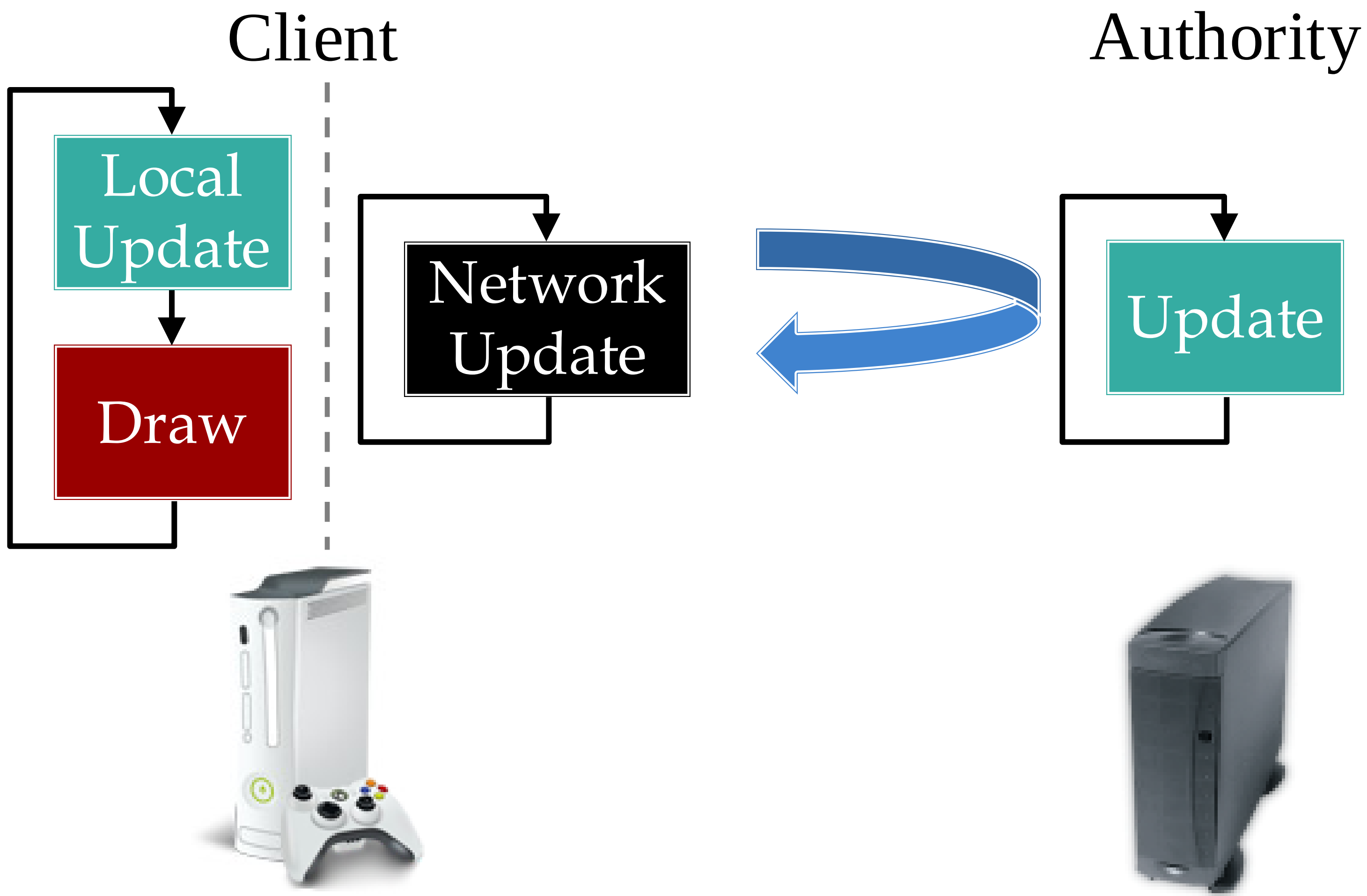


- Lack of authority enforces a delay
  - Only draw what authority tells you
  - Requires round trip from your input
  - Round-trip time (RTT) can be > 200 ms
- This makes the game less responsive
  - Need some way to compensate for this

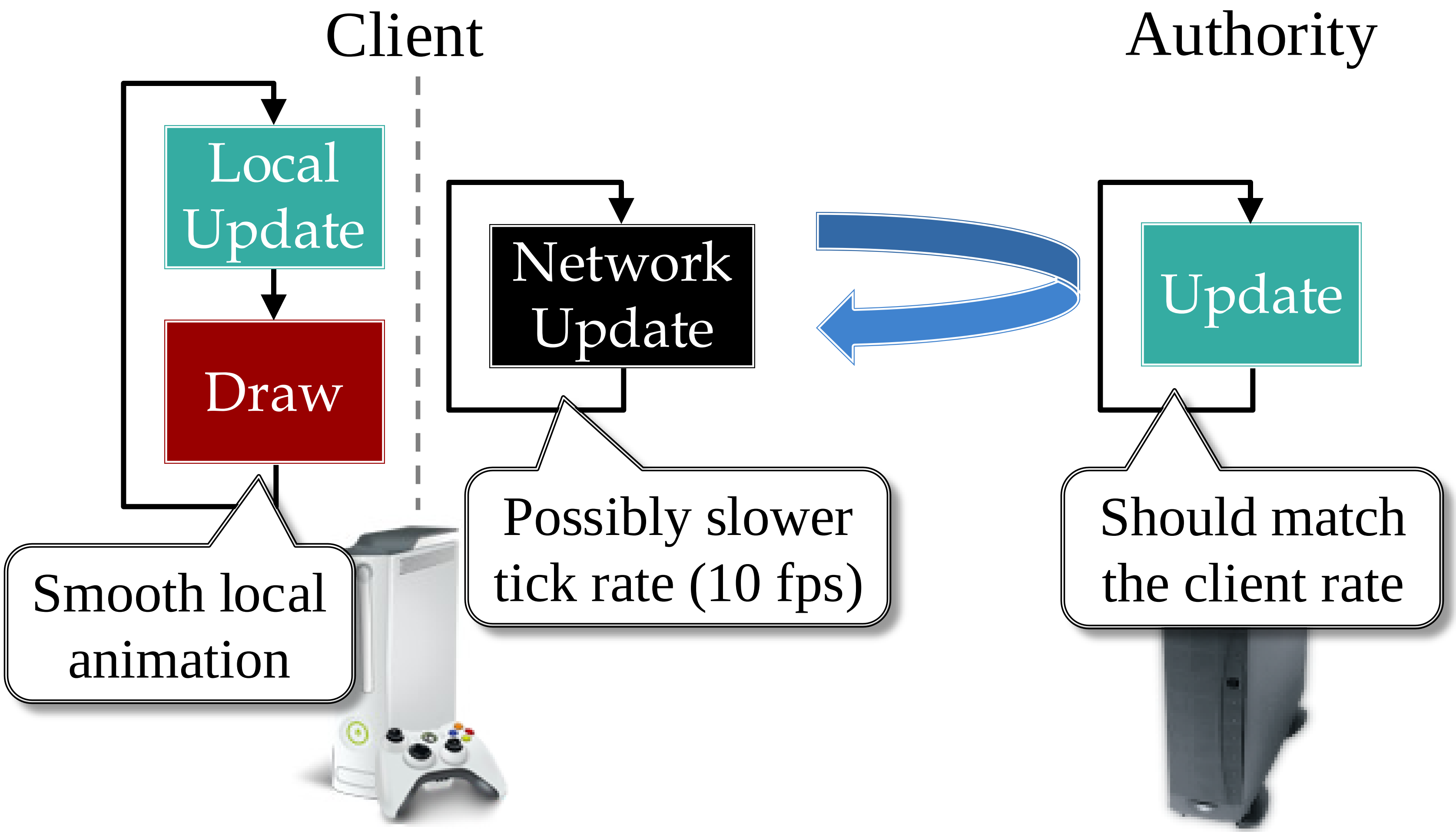
Game Session: Part of Core Loop



Decoupling the Network Loop

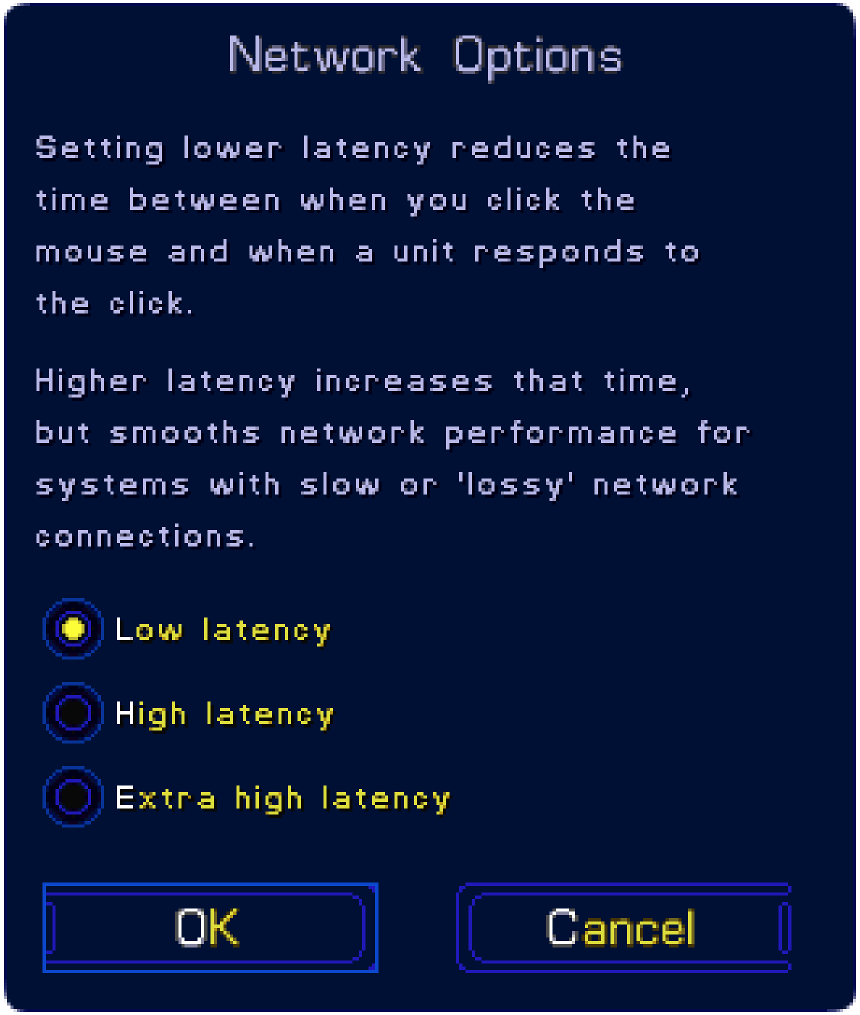


Decoupling the Network Loop



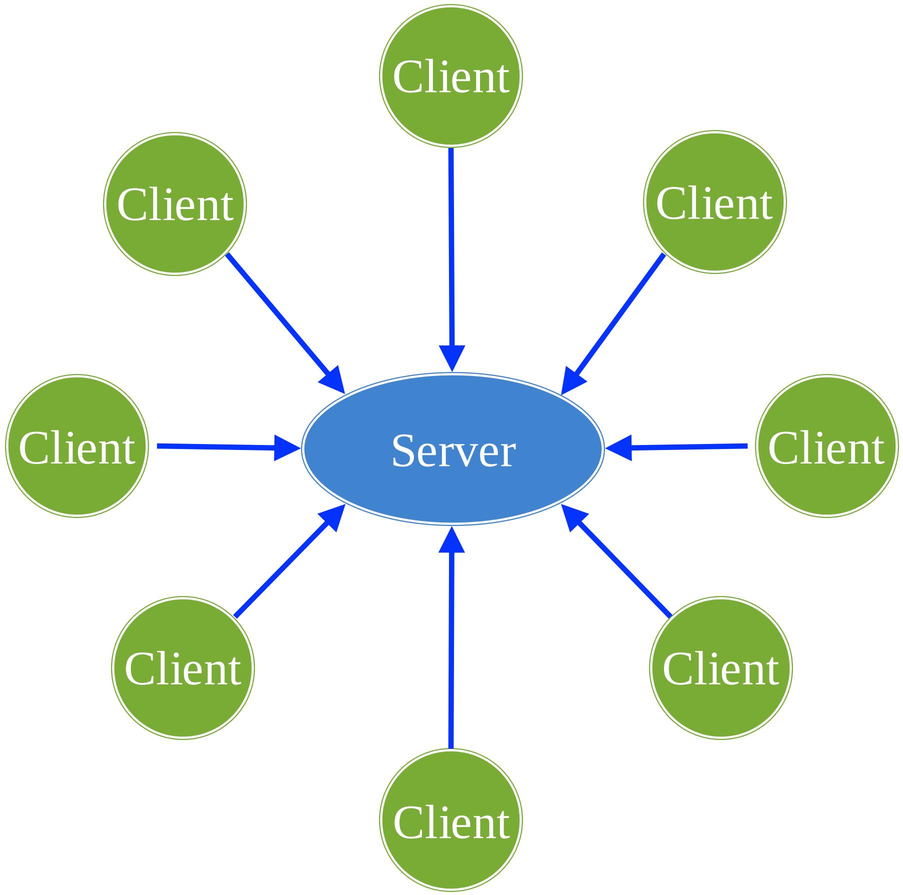
Decoupling Enables Latency Masking

- Animation is "buying time"
  - Looks fast and responsive
  - But no real change to state
  - Animation done at update
- **Examples:**
  - Players wait for elevator
  - Teleportation takes time
  - Many hits needed per kill
  - Bullets have flying time
  - Inertia limits movement



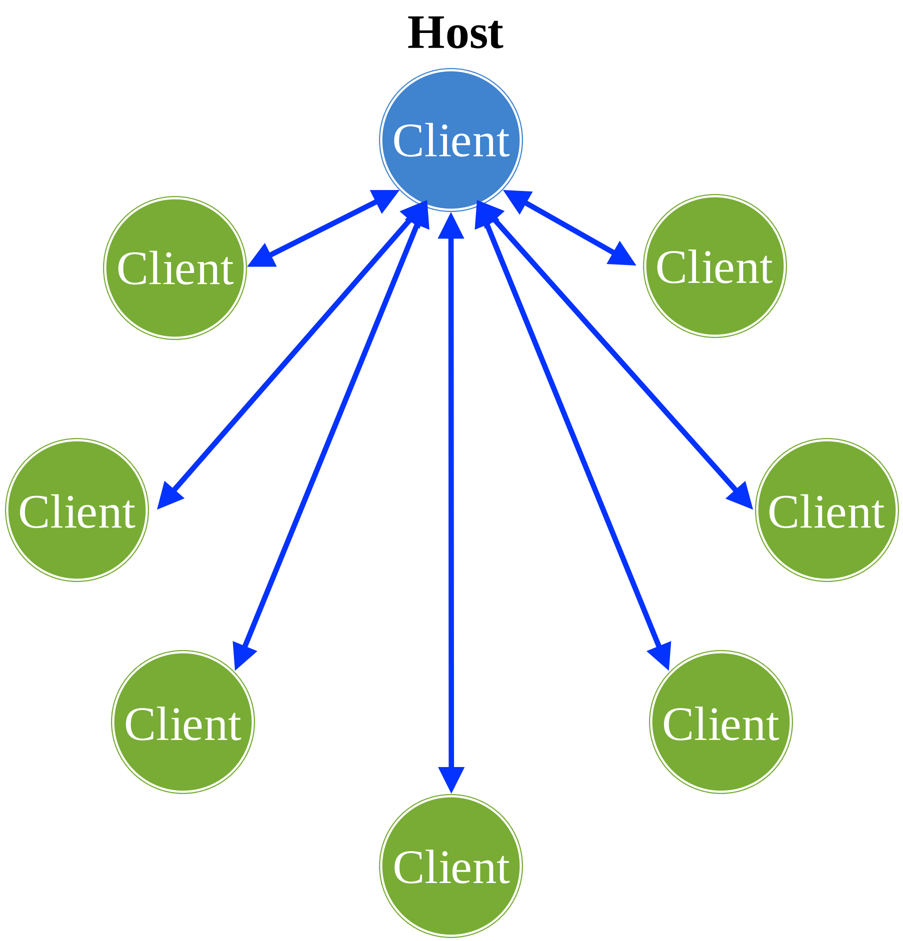
Game Session: Dedicated Server

- Server developer provides
  - Acts as central authority
  - May be several servers
  - May use cloud services
- **Pros:**
  - Could be real computer
  - More power/responsiveness
  - No player has advantage
- **Cons:**
  - Lag if players not nearby
  - Expensive to maintain



Game Session: AdHoc Server

- One client acts as host
  - Acts as central authority
  - Chosen by matchmaker
  - But may change in session
- **Pros:**
  - Cheap long-term solution
  - Can group clients spatially
- **Cons:**
  - Server is a mobile device
  - Host often has advantages
  - Must migrate if host is lost

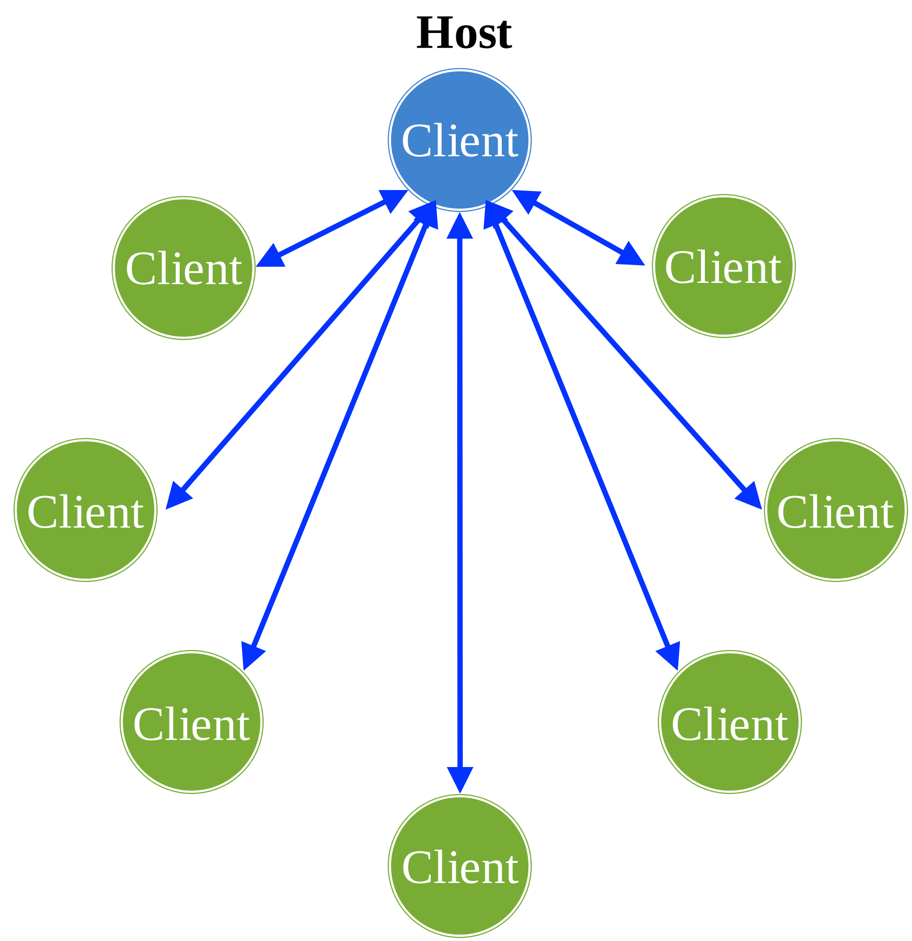


Game Session: AdHoc Server

- One client acts as host
  - Acts as central authority
  - Chosen by matchmaker
  - But may change in session
- **Pr**
  - n
  - lly

Popular in Generation 7

- **Cons:**
  - Server is a mobile device
  - Host often has advantages
  - Must migrate if host is lost

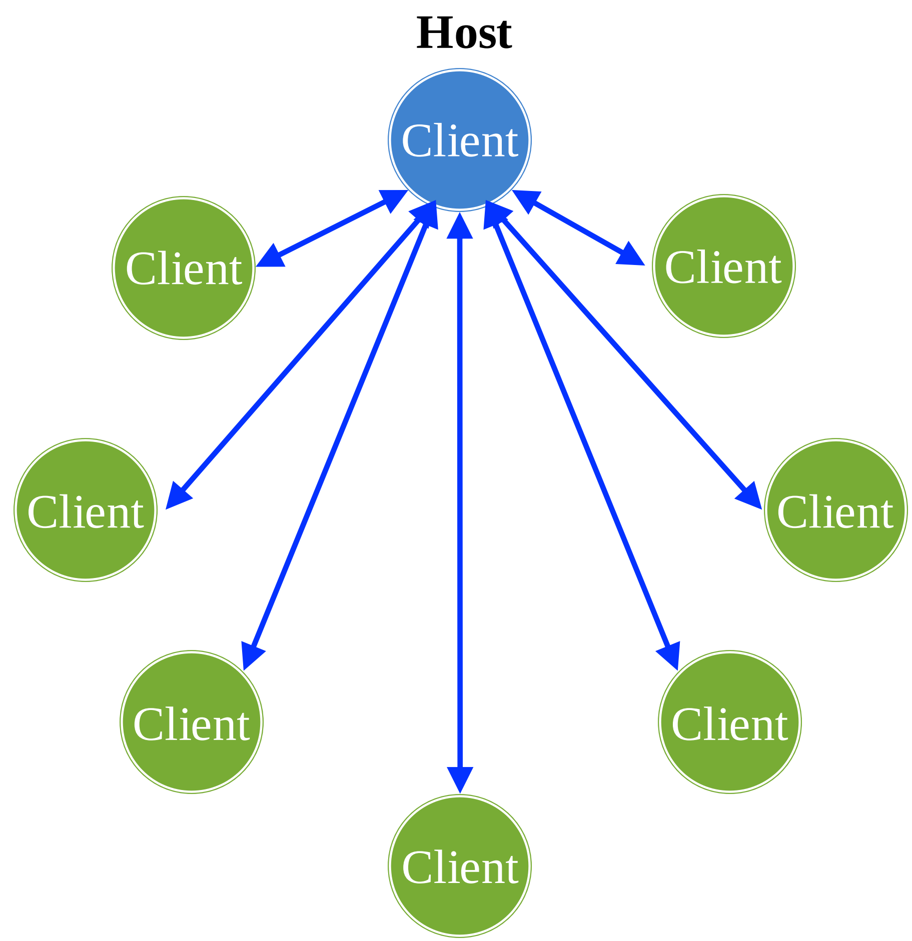


Game Session: AdHoc Server

- One client acts as host
  - Acts as central authority
  - Chosen by matchmaker
  - But may change in session
- **Pr**
  - n
  - lly

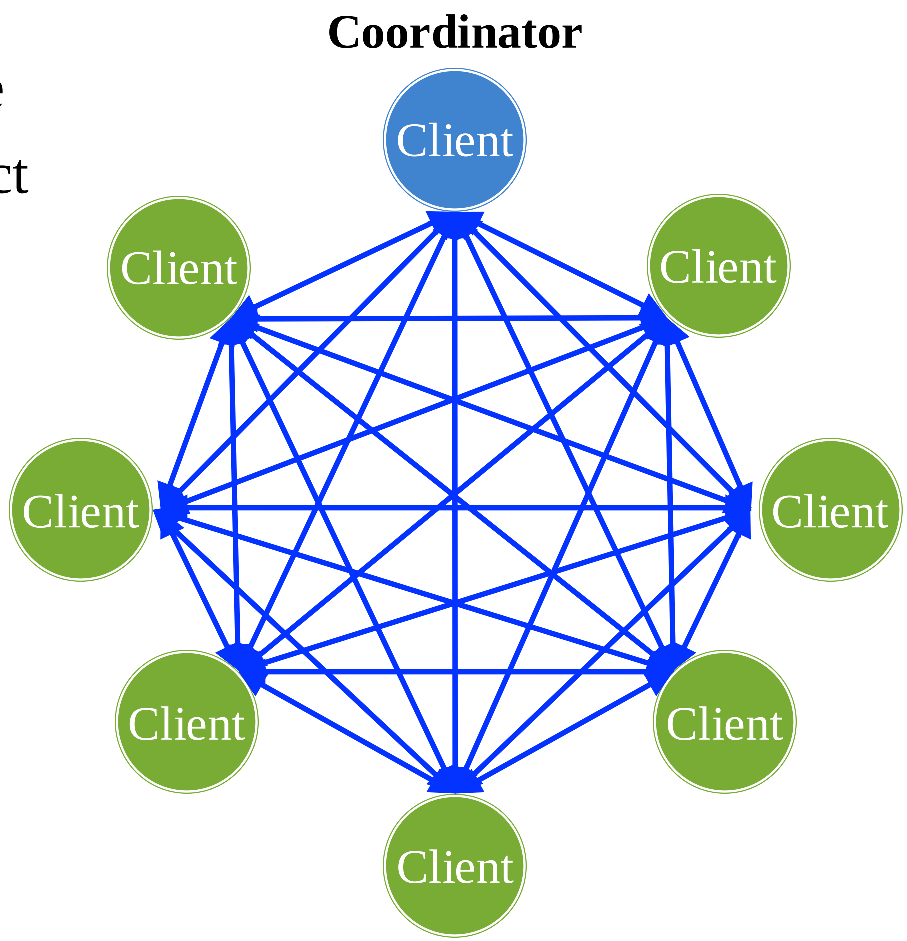
Looks like the CUGL approach?

- **Cons:**
  - Server is a mobile device
  - Host often has advantages
  - Must migrate if host is lost



Game Session: True P2P

- Authority is distributed
  - Each client owns part of state
  - Special algorithms for conflict
  - Coordinator for adds/drops
- **Pros:**
  - No lag on owned objects
  - Lag limited to “attacks”
  - Same advantages as adhoc
- **Cons:**
  - Incredibly hard to implement
  - High networking bandwidth

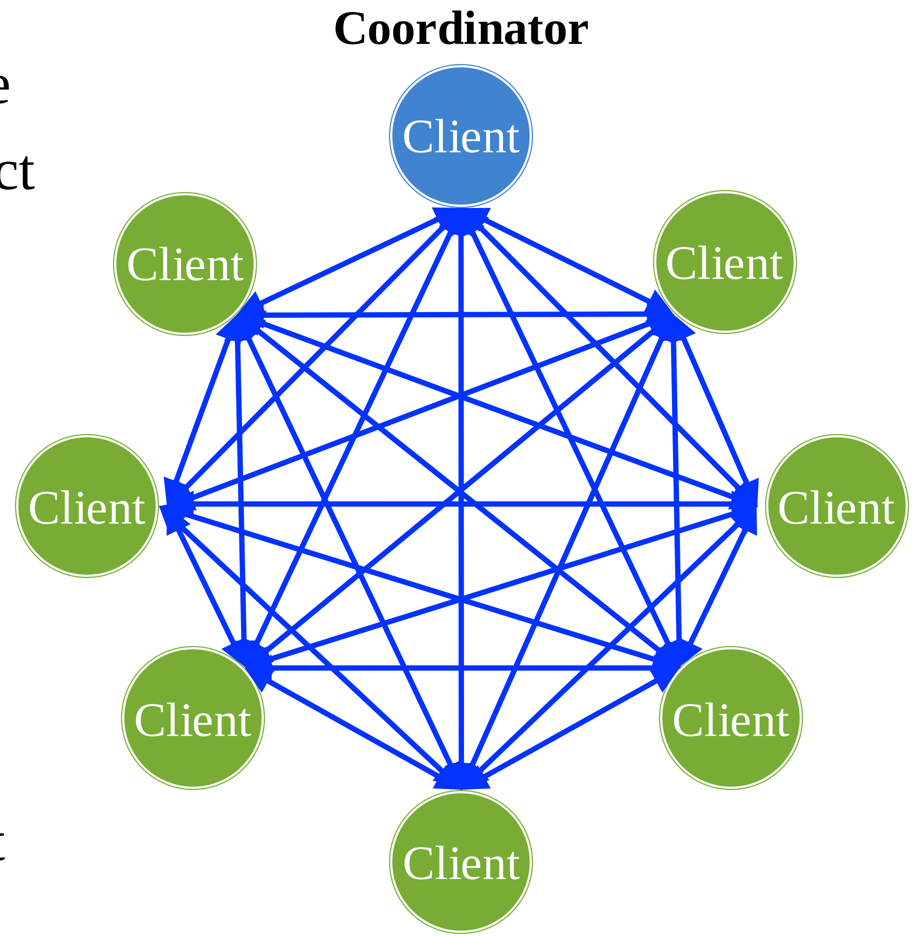


Game Session: True P2P

- Authority is distributed
  - Each client owns part of state
  - Special algorithms for conflict
  - Coordinator for adds/drops
- **Pr**
  - …s
  - …
  - …hoc

> *Almost* no-one does this outside academia

- **Cons:**
  - Incredibly hard to implement
  - High networking bandwidth



the gamedesigninitiative at cornell university

Game Session: True P2P

Game Session: True P2P

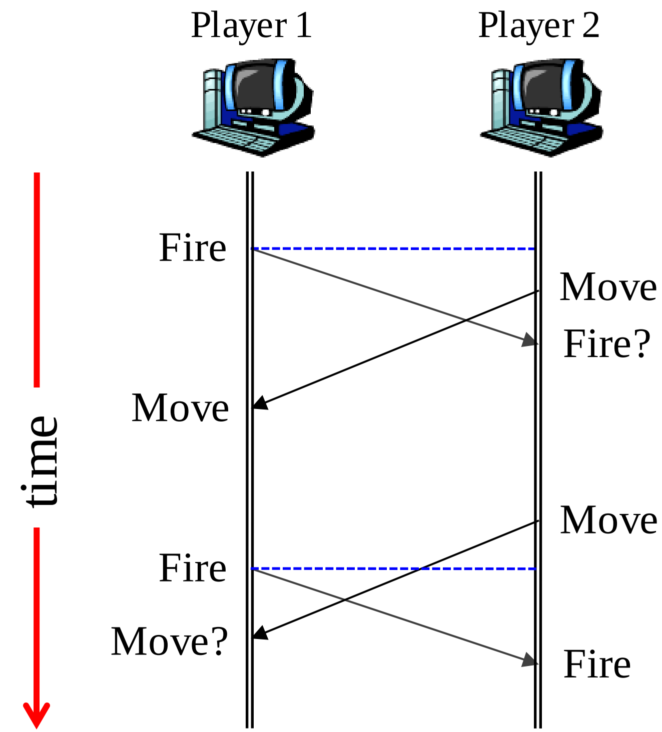
Melee is easy to latency mask!

What Do CUGL Games Use?

- There is a **designated host** in CUGL networking
  - But this used for **matchmaking**, not the session
  - No requirement that host is authoritative
- Library was actually **designed for P2P**
  - Method `broadcast()` broadcasts to all (including host)
  - Worked because *Family Style* spaces were disjoint
- But possible to make **host authoritative**
  - Method `sendToHost()` talks only to host
  - Host **synchronizes** incoming messages
  - Broadcasts back to clients with `broadcast()`

Synchronization Algorithms

- Clients must be **synchronized**
  - Ensure they have same state
  - … or differences do not mattter
- Synchronization != authority
  - Authority determines true state
  - Not *how* clients updated
  - Or *when* clients are updated
- Major concept in networking
  - Lots of complicated algorithms
  - Also a **patent mindfield**
  - Take distributed systems course



Synchronization Algorithms

Pessimistic

- Everyone sees same world
  - Ensure local = world state
  - Forces a drawing delay
- Best on fast networks
  - Local LAN play
  - Bluetooth proximity
- Or games with limited input
  - Real time strategy
  - Simulation games

Optimistic

- Allow some world drift
  - Best guess + roll back
  - Fix mistakes if needed
- Works on any network
  - Lag errors can be fixed
  - But fixes may be distracting
- Works great for shooters
  - Player controls only avatar
  - All else approximated

Synchronization Algorithms

Pessimistic

- Everyone sees same world
  - Ensure local = world state
  - Forces a drawing delay
- Best on fast networks
  - Local LAN play
  - Bluetooth proximity
- Or games with limited input
  - Real time strategy
  - Simulation games

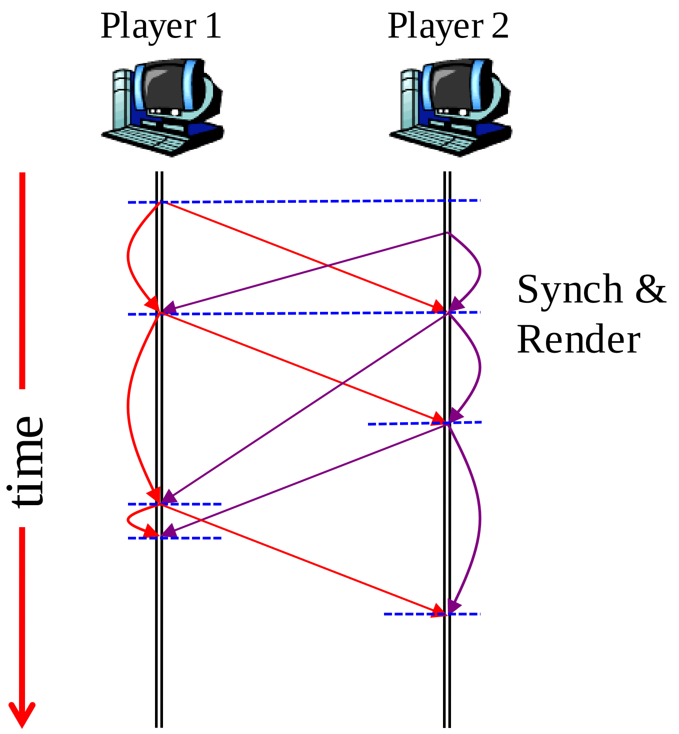
Optimistic

- Allow some world drift
  - Best guess + roll back
  - Fix mistakes if needed
- Works on any network
  - Lag errors can be fixed
  - But fixes may be distracting
- Works great for shooters

Also great for distributed authority

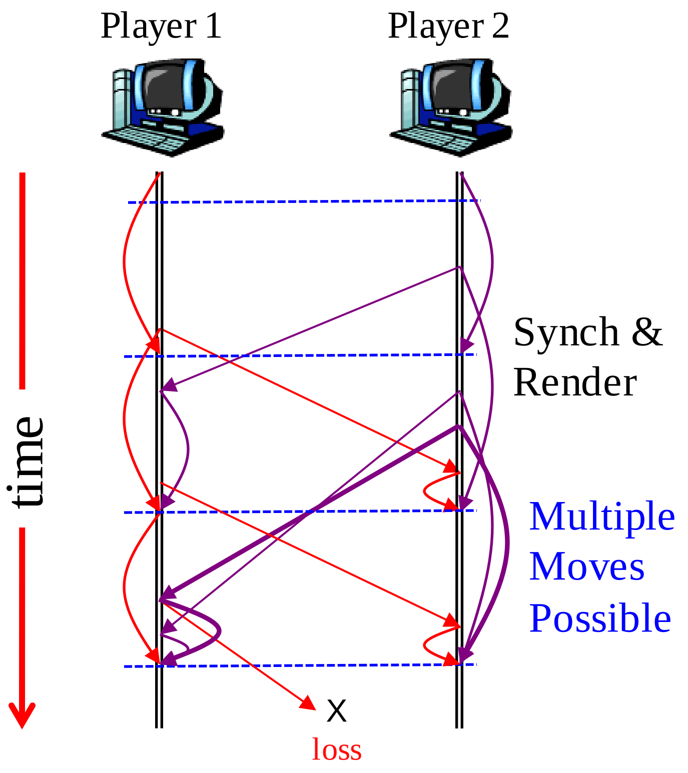
Pessimistic: Lock-Step Synchronization

- **Algorithm**: play by "turns"
  - Players send turn actions
  - Even if no action was taken
  - Wait for response to render
- **Problems**
  - *Long* Internet latency
  - Variable latencies (jitter)
  - Speed set by slowest player
  - What if moves are lost?
- More common in LAN days



Pessimistic: Bucket Synchronization

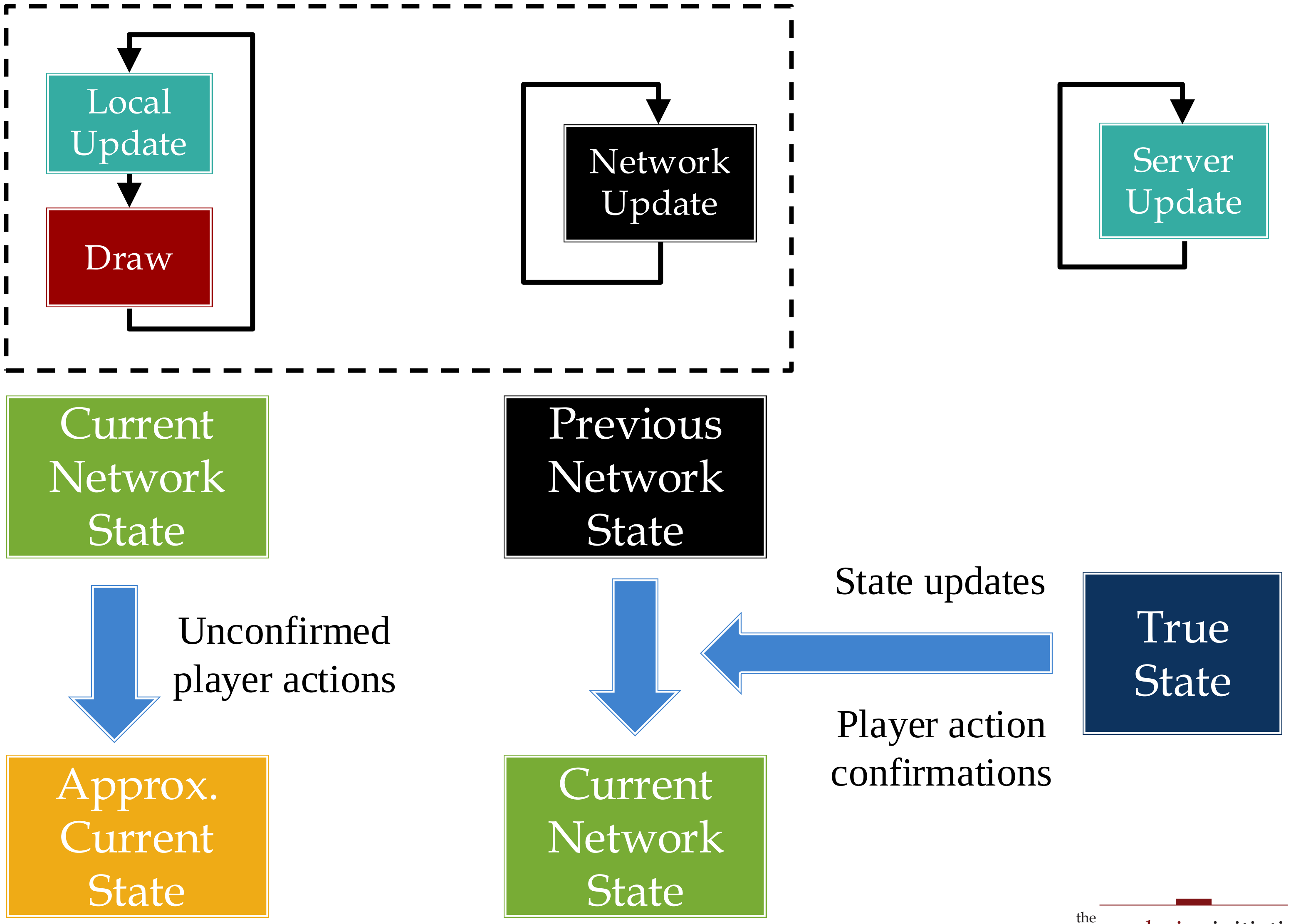
- **Algorithm**: turns w/ timeout
  - Often timeout after 200 ms
  - But can be adapted to RTT
  - All moves are buffered
  - Executed at end of *next* turn
- **Problems**
  - Variable latencies (> a turn)
  - Speed set by slowest player
  - What if moves are lost?
- Used in classic RTS games



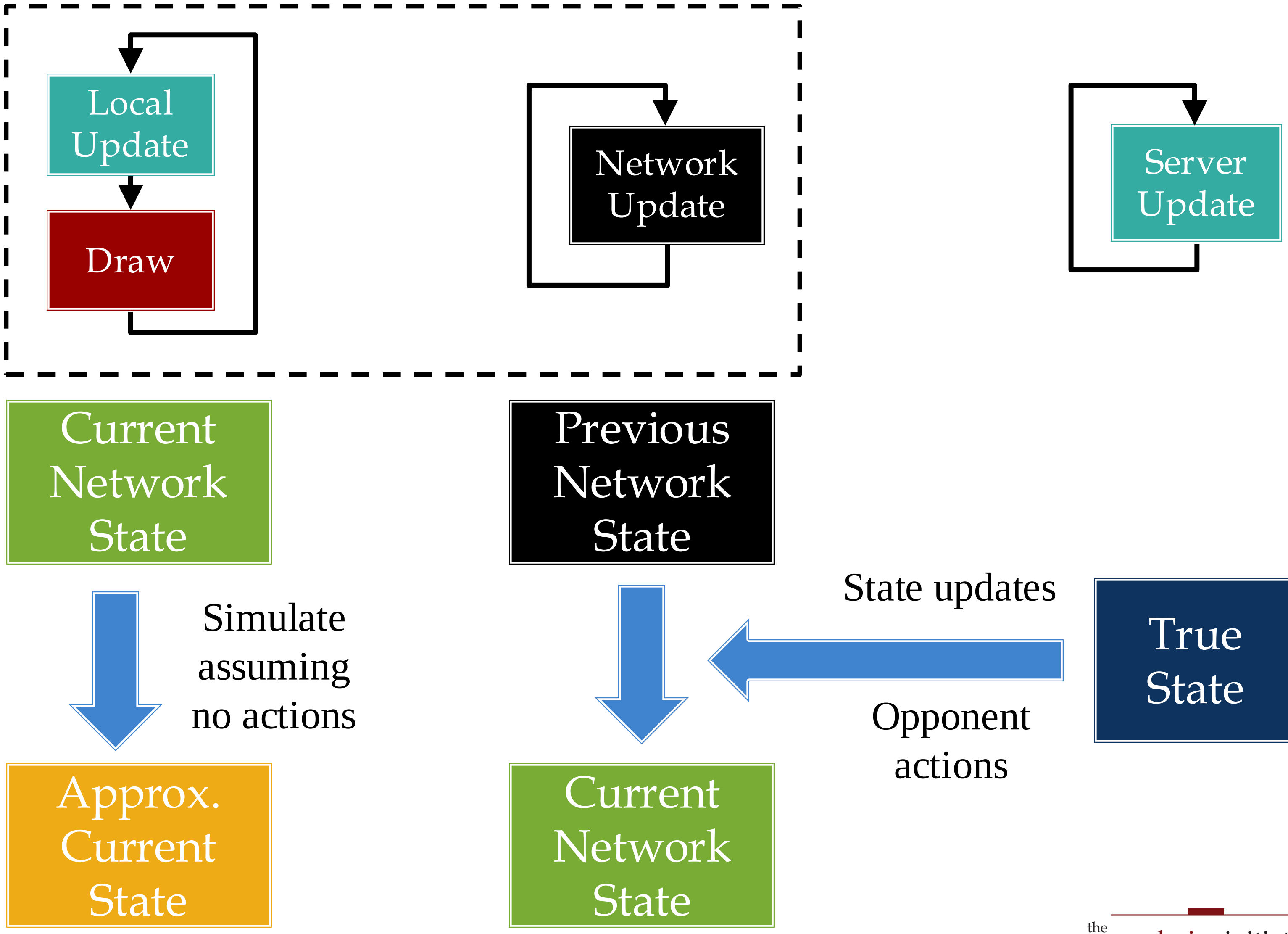
Pessimistic: Bucket Synchronization

- **Algorithm**: turns w/ timeout
  - Often timeout after 200 ms
  - But can be adapted to RTT
  - All moves are buffered
  - Executed at end of *next* turn
- **Problems**
  - Variable latencies (> a turn)
  - Speed set by slowest player
  - What if moves are lost?
- Used in classic RTS games

Optimistic: Personal State



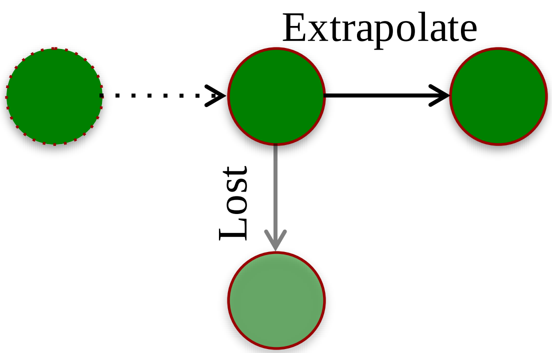
Optimistic: Opponent State



Advantages of Sending Actions

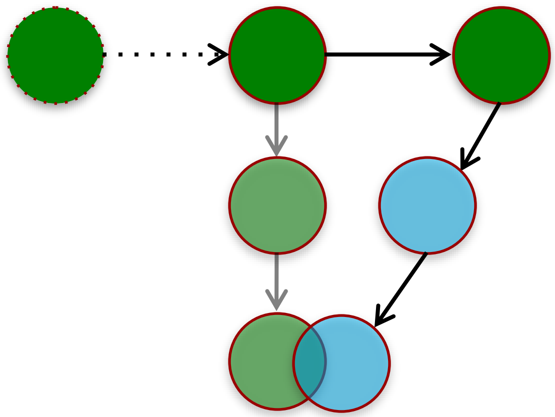
Dead Reckoning

- Assume velocity constant
  - Simulate the new position
  - Treats like physics object
- Generalize to other actions



Error Smoothing

- Can interpolate late actions
  - Create simulation for action
  - Avg into original simulation
- Continue until converge

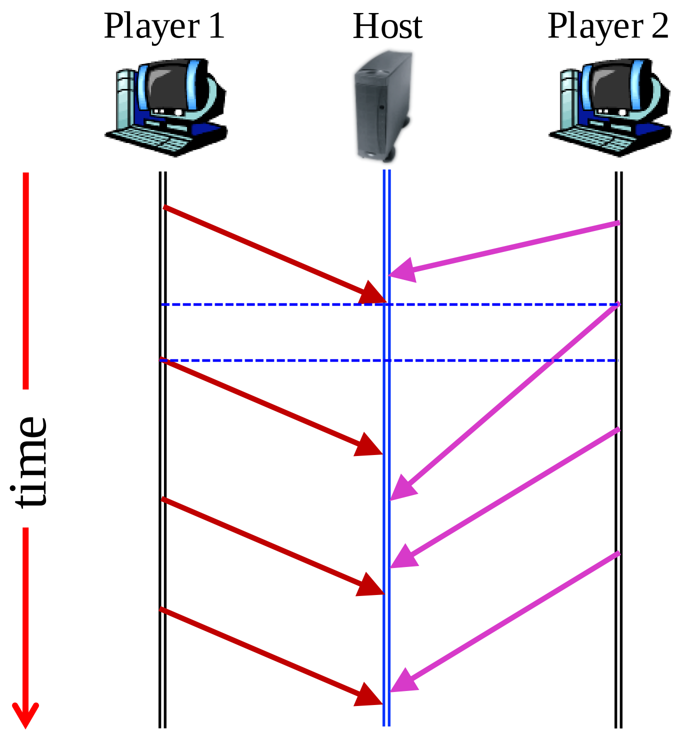


The Perils of Error Correction

the gamedesigninitiative at cornell university

CUGL Networking Guarantees

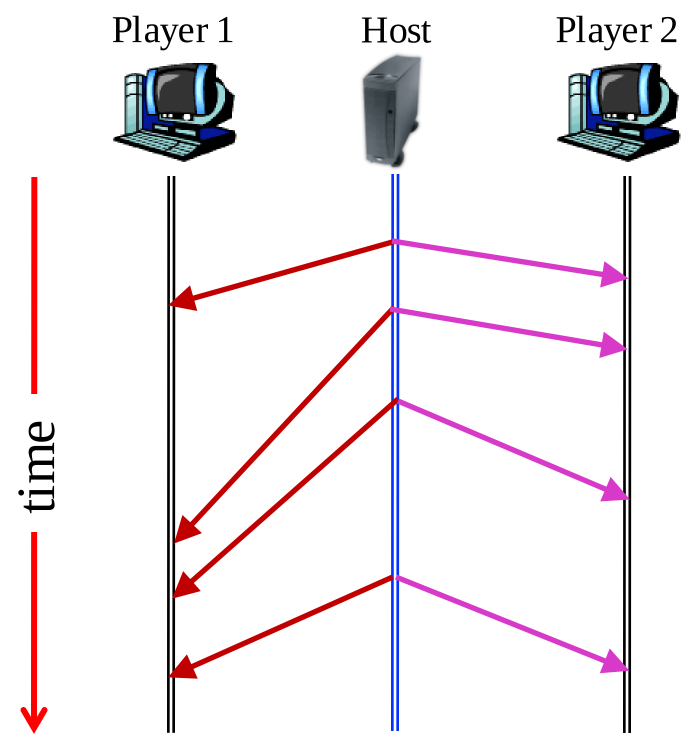
- CUGL built on **WebRTC**
  - Uses **reliable UDP**, not TCP
  - Uses **messages**, not stream
  - Messages are a **byte vector**
- Guarantees **message order**
  - Guarantees are **per client**
  - No guarantee between clients
- Host can **synchronize**
  - Host broadcasts moves to all
  - All clients see in **same order**



the gamedesigninitiative at cornell university

CUGL Networking Guarantees

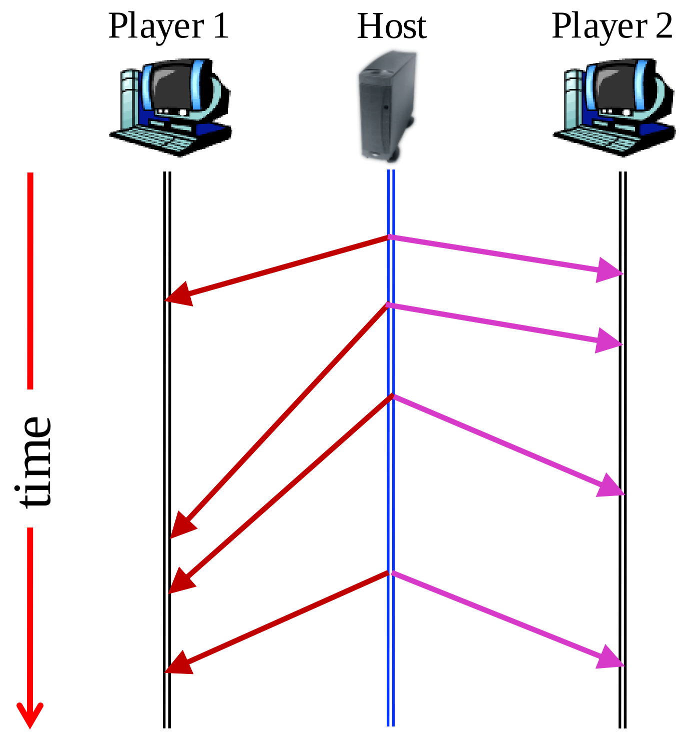
- CUGL built on **WebRTC**
  - Uses **reliable UDP**, not TCP
  - Uses **messages**, not stream
  - Messages are a **byte vector**
- Guarantees **message order**
  - Guarantees are **per client**
  - No guarantee between clients
- Host can **synchronize**
  - Host broadcasts moves to all
  - All clients see in **same order**



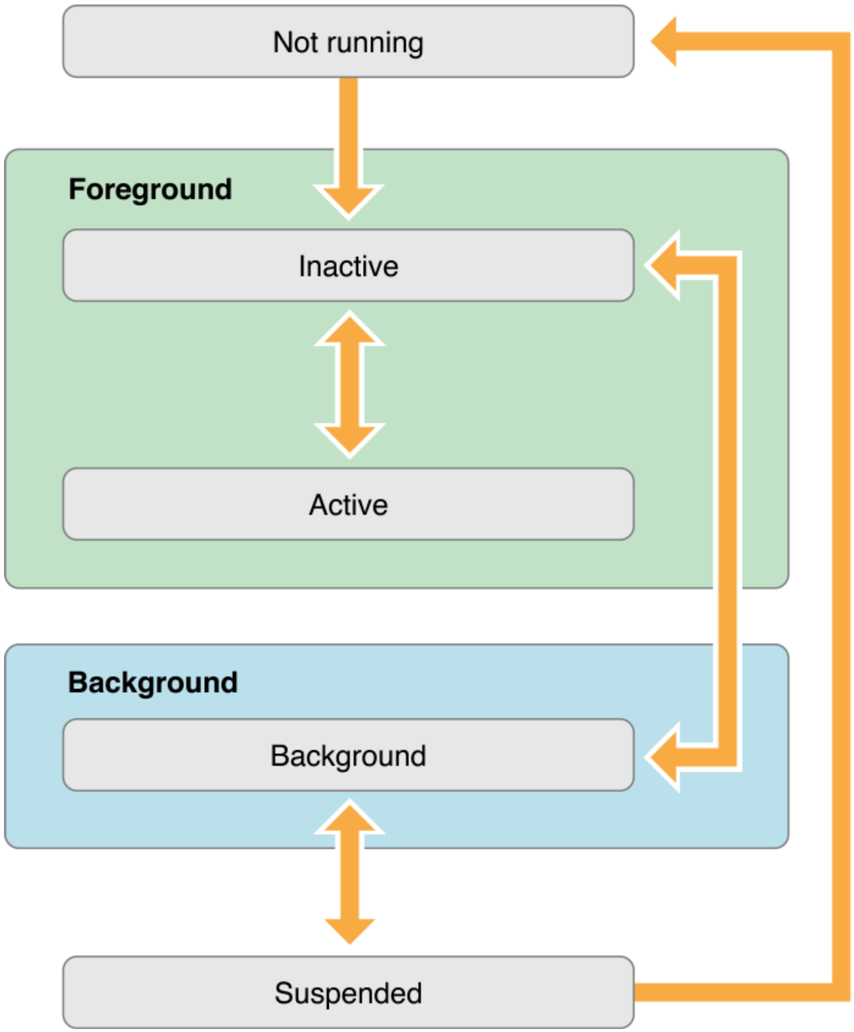
CUGL Networking Guarantees

- CUGL built on **WebRTC**
  - Uses **reliable UDP**, not TCP
  - Uses **messages**, not stream
  - Messages are a **byte vector**
- Guarantees **message order**
  - Guarantees are **per client**
  - No guarantee between clients
- Host can **synchronize**
  - … o all
  - … **rder**

But introduces message *delay*

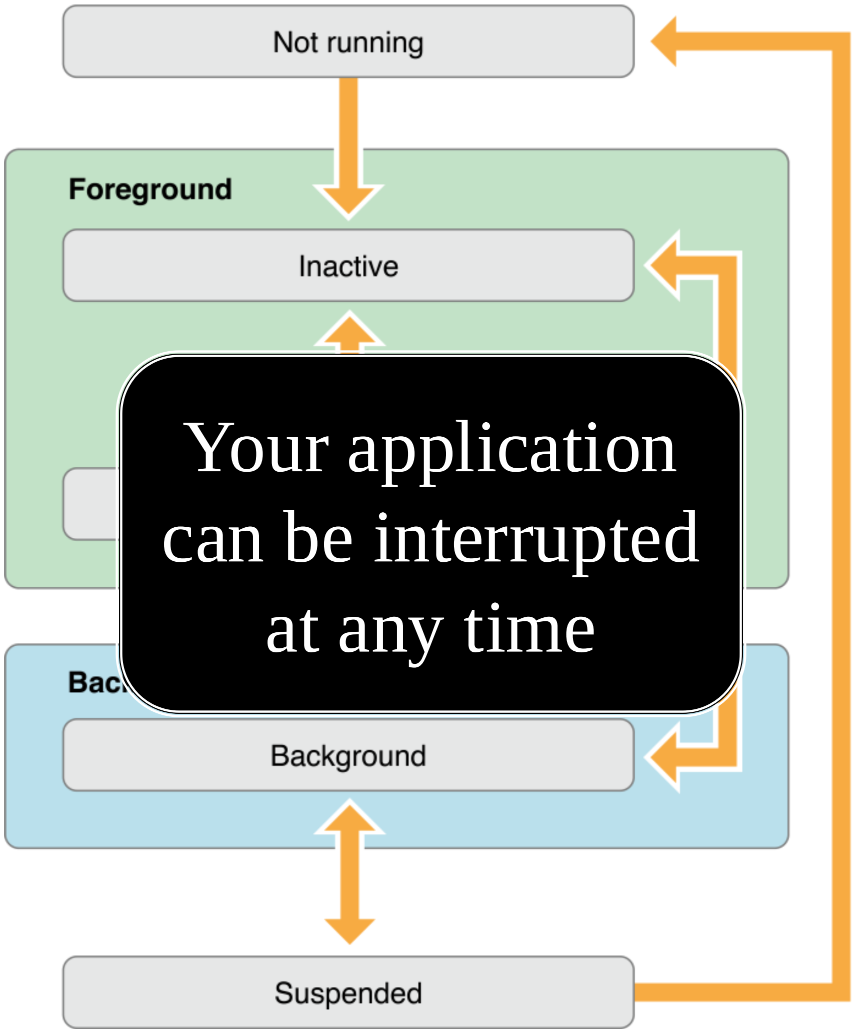


Aside: Application State



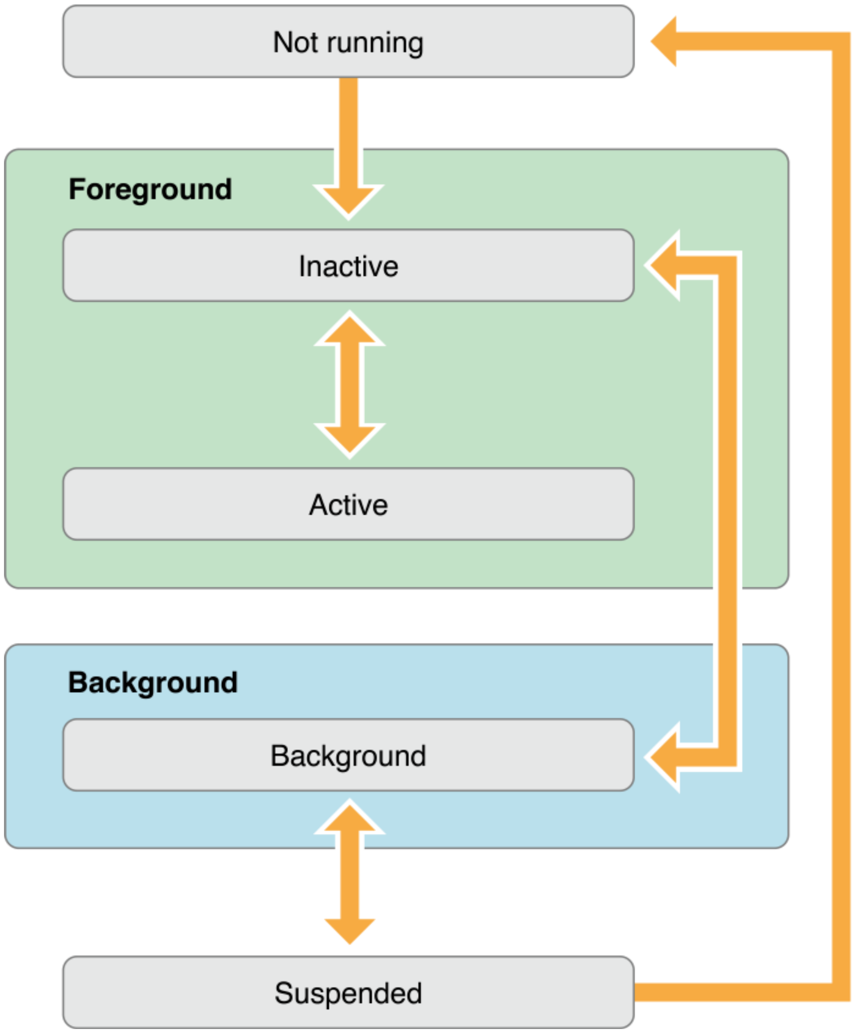
- **Active**
  - Running & getting input
- **Inactive**
  - Running, but no input
  - Transition to suspended
- **Background**
  - Same as inactive
  - But apps can stay here
  - **Example**: Music
- **Suspended**
  - Stopped & Memory freed

Aside: Application State



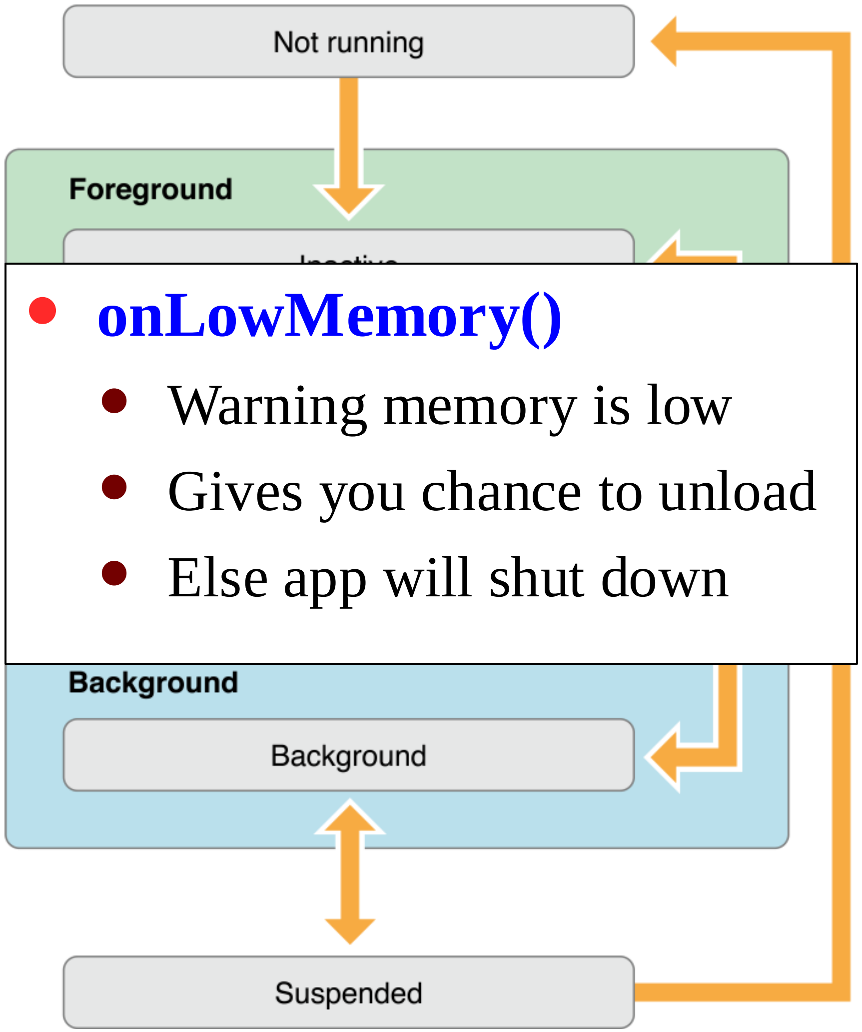
- **Active**
  - Running & getting input
- **Inactive**
  - Running, but no input
  - Transition to suspended
- **Background**
  - Same as inactive
  - But apps can stay here
  - **Example**: Music
- **Suspended**
  - Stopped & Memory freed

The CUGL `Application` Class



- **onStartup()**
  - Initialized and now active
- **onSuspend()**
  - Sent to background
  - Gives you chance to save
  - Also time to pause music
- **onResume()**
  - Returns to app to active
  - Allows you to restore state
- **onShutdown()**
  - Stopped & memory freed

The CUGL `Application` Class



- **onLowMemory()**
  - Warning memory is low
  - Gives you chance to unload
  - Else app will shut down

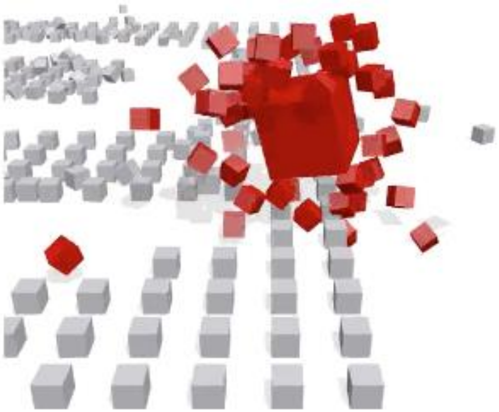
- **onStartup()**
  - Initialized and now active
- **onSuspend()**
  - Sent to background
  - Gives you chance to save
  - Also time to pause music
- **onResume()**
  - Returns to app to active
  - Allows you to restore state
- **onShutdown()**
  - Stopped & memory freed

Application Restoration Plan

- What does your app due when suspended?
  - Kick them out of game immediately?
  - Pause the game until they resume (or timeout)?
  - Temporarily switch the player over to an AI?
- Also important for non-networked games
  - Save the current progress in the level?
  - Restart them at the beginning of the level?
- New feature of **Architecture Specification**

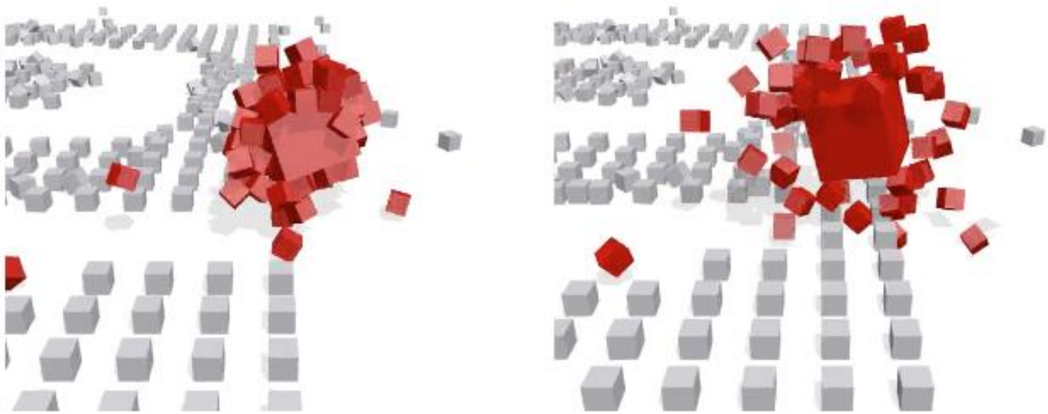
Example: *Slay the Spire*

Physics: Challenge of Synchronization

- Deterministic bi-simulation is very hard
  - Physics engines have randomness (not Box2D)
  - Not all architectures treat floats the same
- Need to mix interpolation with snapshots
  - Like error correction in optimistic concern
  - Run simulation forward from snapshots

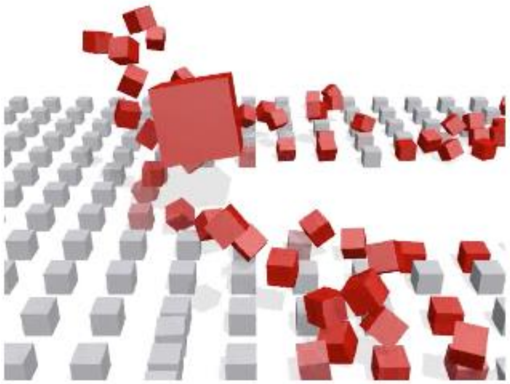
Physics: Challenge of Synchronization



- Deterministic bi-simulation is very hard
  - Physics engines have randomness (not Box2D)
  - Not all ar
- Need to r snapshots
  - Like error correction in optimistic concern
  - Run simulation forward from snapshots

See today's reading

Physics: Challenge of Authority



- Distributed authority is very difficult
  - Authority naturally maps to player actions
  - Physics is a set of interactions
- Who owns an uncontrolled physics object?
  - **Gaffer:** The client that set in motion
  - Collisions act as a form of "authority tag"

Physics: Challenge of Authority

But we have added it to CUGL.
See Tutorial in Code Samples.

- Distrib
  - Autho
  - Physics is a set of interactions
- Who owns an uncontrolled physics object?
  - **Gaffer:** The client that set in motion
  - Collisions act as a form of "authority tag"

Summary

- **Consistency:** local state agrees with world state
  - Caused by latency; takes time for action to be sent
  - Requires complex solutions since must draw now!
- **Authority** is how we measure world state
  - Almost all games use a centralized authority
  - Distributed authority is beyond scope of this class
- **Synchronization** is how we ensure consistency
  - Pessimistic synchronization adds a sizeable input delay
  - Optimistic synchronization requires a lot of overhead