CMP302 Coursework

Overview

- Made in Unity using .NET sockets.
- Server-Client Hybrid (One client also has the server – which relays information between all clients).
  - Easy hosting for clients.
  - No extra downloads if wanting to host.
  - But one player has advantage of time (their matches up perfectly with server – they're likely to win all conflicts).
  - No dedicated server for people wanting to host big servers (performance overhead).
- Uses UDP for everything.
  - As its fast and integrates with event system well.
  - 3 way handshake and packet delivery verification added when needed.
- Transfers equivalent of structs for packets.
  - Fast & small.
  - Not much error correction.
- Event Based - Packets are decoded then passed onto events for the relevant GameObjects to deal with.

Packet Structure

- Contain a Header.
  - In form of class – due to C# technicality.
  - Contains Time sent, and Type.
    - Type is an Enum to signify what type the packet is for easier debugging and lower overhead casting (C# has reflection capabilities).
- Network Packets
  - Packets about network events – connect, disconnect, time sync etc.
  - Contain acknowledgement versions.
  - New Connection packet contains the Player ID for example.
- Position Packets / Gameplay event packets
  - Position packet is only gameplay event packet.
  - Contains position, whether the player was moving, and speed.

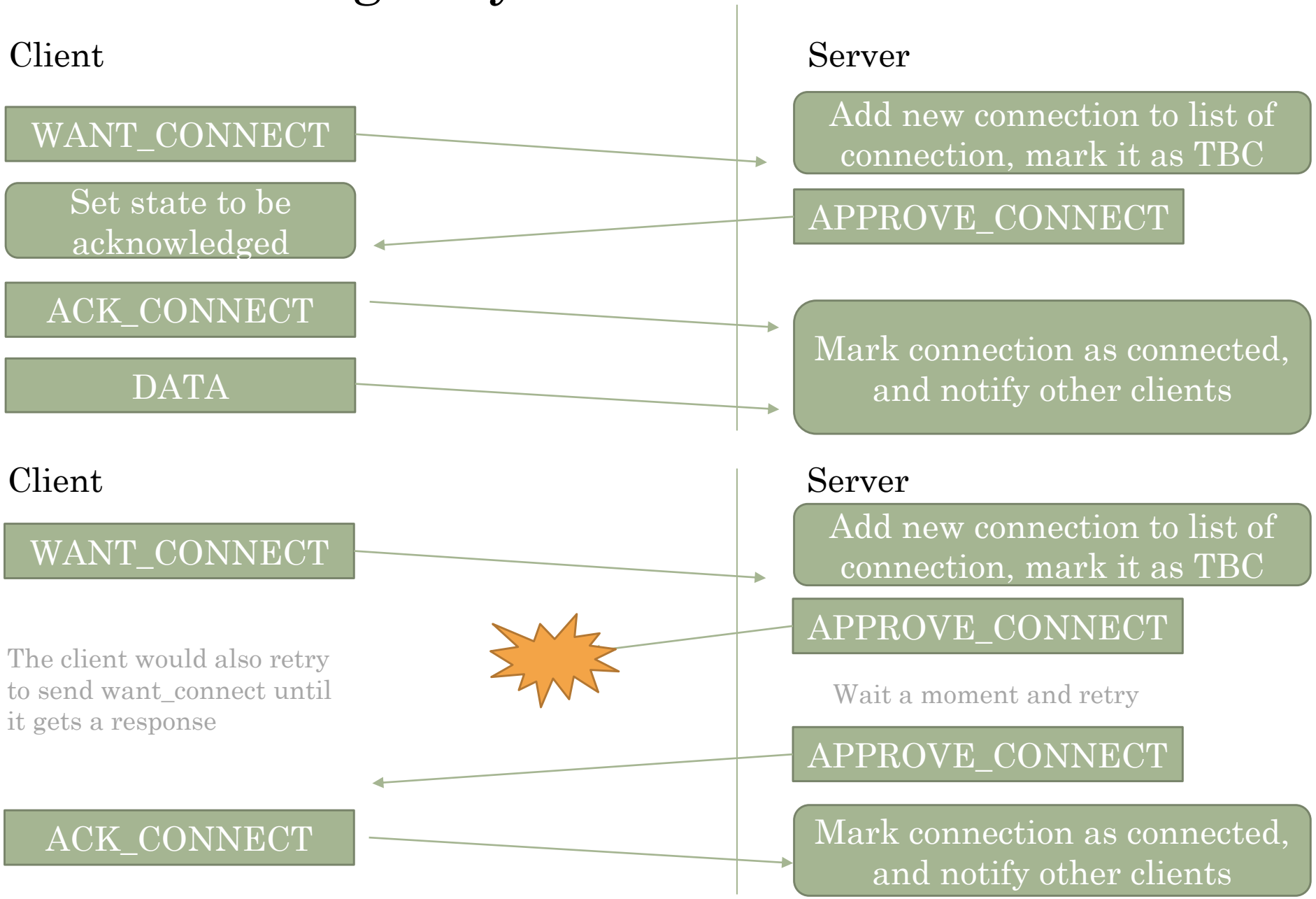
Styles of packets

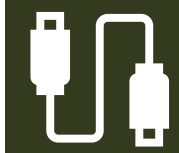
- Three way handshake equivalent.
  - Currently only used when connecting – would be implemented with coins.
- Confirm awareness, signal changes.
  - Server asks player if they're aware of new player.
    - Only asks if server is not sure.
    - Player would not send back ACK if its not sure, or if its lost.
    - Knows if asked again, and the client is aware, the ACK gets sent back as last ACK must've gotten lost.
- One time.
  - Low priority or high frequency events.
  - Position updates are high frequency events.
    - Send packet but don't verify that its delivered.
  - Trees would be an example of a low priority event (Not implemented).
    - Shaking a tree has no functional purpose – so what if it gets lost.
    - Just for aesthetics.

Player Connections

- Server keeps track of connections.
  - Who are they aware of.
  - Their connection status.
  - Last time sent and received packet.
- Max 32 connections.
  - Theoretically max is how much the PC can handle.
  - Slow downs cause rubber-banding due to time syncs.
- Server has array of packets that have to be forwarded on to other players.
- Server sends time syncs every 250ms to all.
  - Non guaranteed delivery.
  - Ideally would be sent when its detected that latency and server time seem out of sync or server slows down (Improvement).

Connecting Player





Server

Client

New Client Connected, notify others.

NEW_CONNECTION

Dispatch Network Event to recipients

PlayerManager receives and create new player

Wait a some time and send another packet stating a new connection.

NEW_CONNECTION_ACK

NEW_CONNECTION

Dispatch event, PlayerManager already know its instanced the player, but send confirmation regardless (how to make sure that the client got the packet)

Mark connection as aware of the new connection.

NEW_CONNECTION_ACK

Event Structure

- The client interfaces the server with the ‘ClientConnectionManager’ class
  - Handles:
    - Connecting.
    - Network events (player connect/disconnect/time sync).
    - Sending/receiving and decoding packets.
  - Passes non connection events onto the NetworkEventDispatcher class.
- NetworkEventDispatcher is the primary ‘interface’ class for any GameObjects wanting to subscribe to events.
  - Bottom up subscription to make it easier to add new events.
  - Although for network events they need to access the ClientConnectionManager (as you cant subscribe to a reference of a event – C# technicality).
  - Has PositionPacketEvent which is dispatched when a new packet is received.
- Sending packets is not handled via event as a non owner cannot dispatch event.
  - Therefore its just a function call.

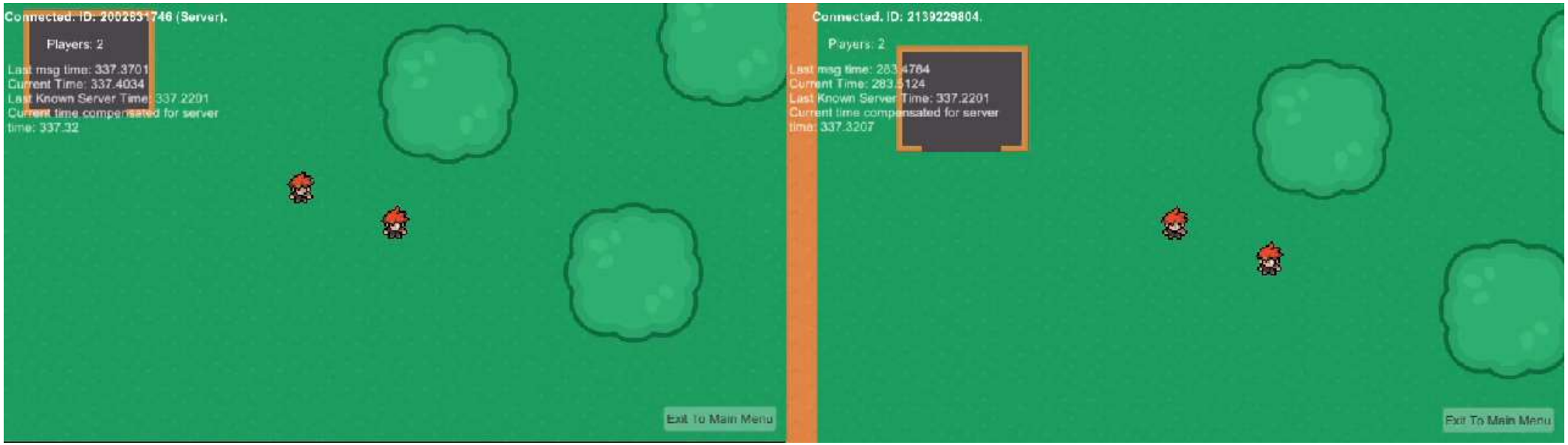
Prediction

- Based off of linear interpolation.
- A new position packet is sent every 30ms.
  - Every ~2 frames so the client isn't more than 4-6 frames behind and doesn't jitter too much.
  - Also doesn't use as much bandwidth + CPU as a sending every 16ms.
  - Game isn't too intensive so a target of 60FPS/16.66ms per frame isn't unrealistic.
- Due to lack pause between frames interpolation is needed.
- Old packets are discarded – only latest ones are stored.
- Packet sends whether the player is moving.

Testing – Ideal Conditions

- Ideal conditions (no interferences or delay)
- Slight jitter when syncing time.
  - Can be fixed by having a time lerp client-side.

Testing – Typical Conditions



- 60ms, 0.5% packet loss, 0.2% chance of throttle of 60ms, 0.1% chance of out of order.
- Jitter is intensified due to higher time desyncs.

Testing – Horrendous Conditions

- 250ms, 15% packet loss, 10% chance of throttle of 60ms, 1% of 2 duplicate, 5% out of order.
- Jitter + Rubber banding – partly due to time syncs, and partly due to missing packets.

Testing – “A cabin in the woods”

- 1000ms, 60% chance of packet dropped, 50% throttle of 400ms, 40% 4 duplicate, 80% out of order
- Unplayable, still updates eventually.
- Clamping prediction speed to max of player (5u/s) would help the ‘fly away’ effect.

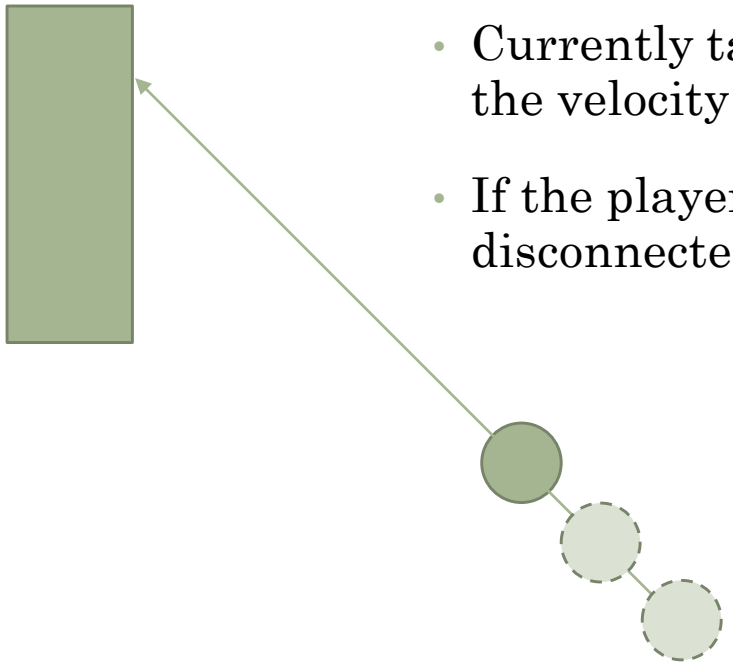
No interpolation

- Done on an earlier build with a time sync bug – but no interpolation.
  - Time is used only on interpolation currently.

Fitness for purpose

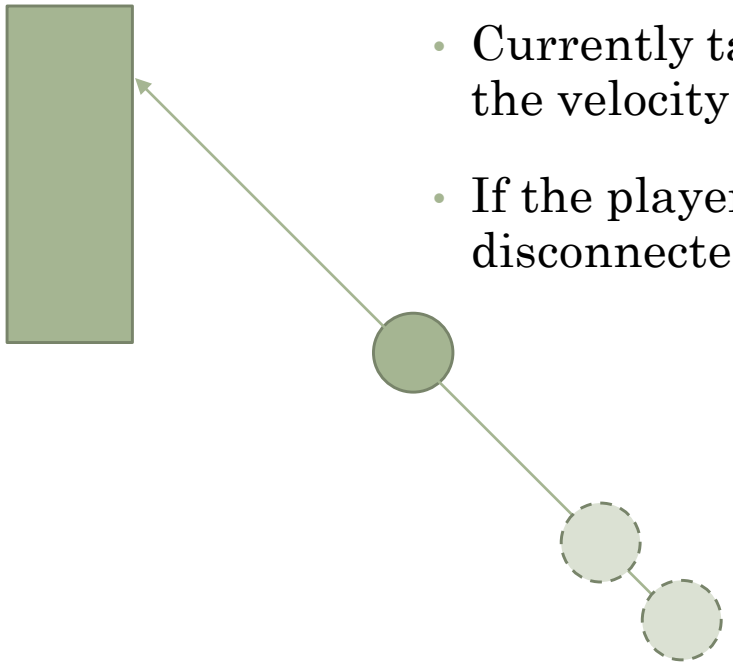
- Syncs both work well.
- Would be perfect without jitter.
  - ‘Dilatating time’ between times would be a very good update.
  - Clamping max speed.
- Connection handshake works in all network conditions – as long as 10 attempts aren’t reached.
- Firewall needing to be disabled would need to be fixed.
  - Third party matchmaker to help poke a wall?
  - Or adding ports that the client can connect on and sending out packets to poke a hole in the firewall.
- Improve prediction with a cubic spline…

Current Prediction



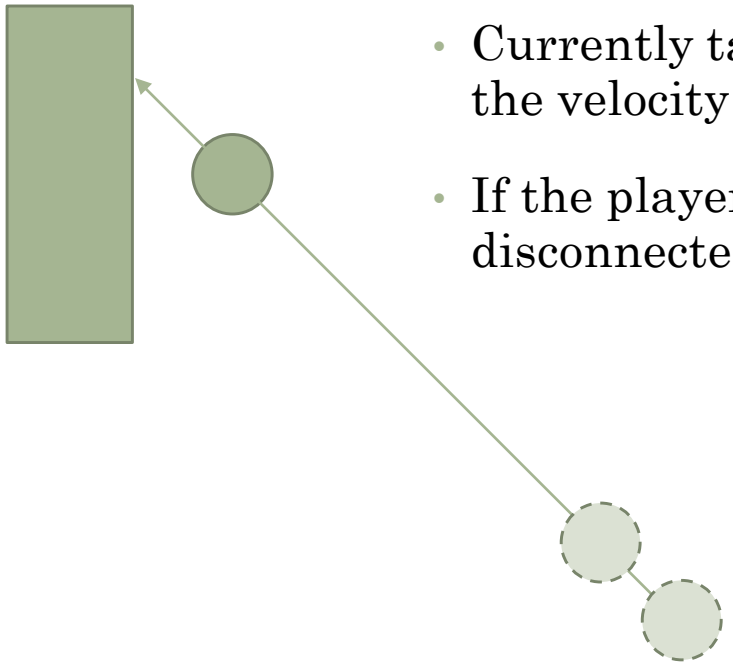
- Currently takes average of last two positions, and bases the velocity and moves the player forward.
- If the player doesn't send anything in 7.5 seconds it will be disconnected.

Current Prediction



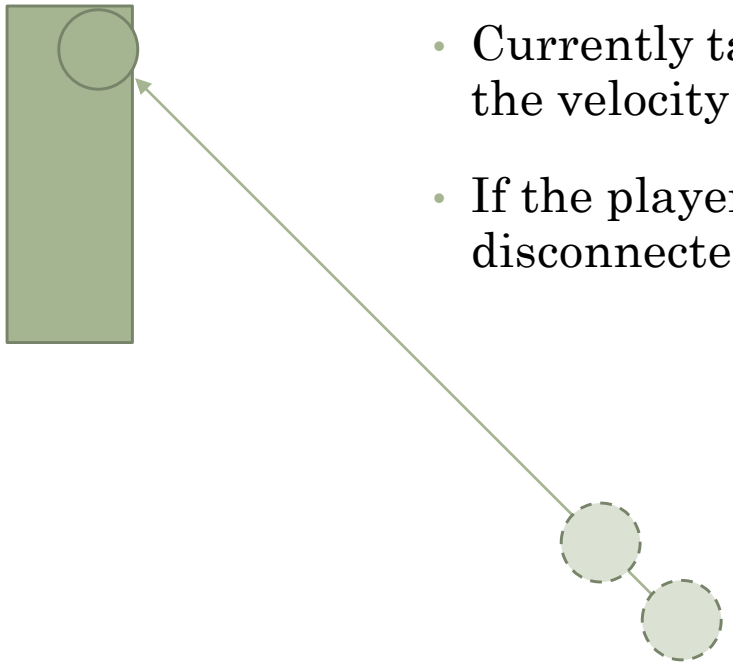
- Currently takes average of last two positions, and bases the velocity and moves the player forward.
- If the player doesn't send anything in 7.5 seconds it will be disconnected.

Current Prediction



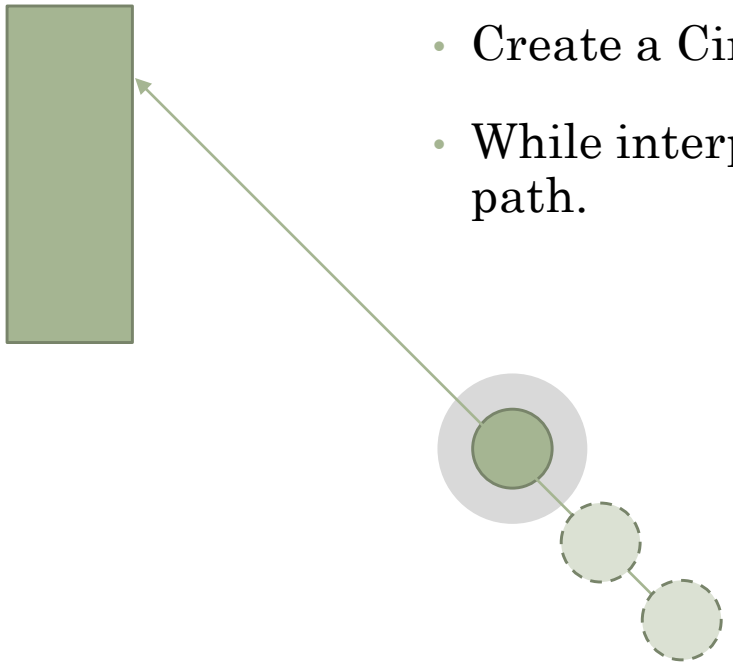
- Currently takes average of last two positions, and bases the velocity and moves the player forward.
- If the player doesn't send anything in 7.5 seconds it will be disconnected.

Current Prediction



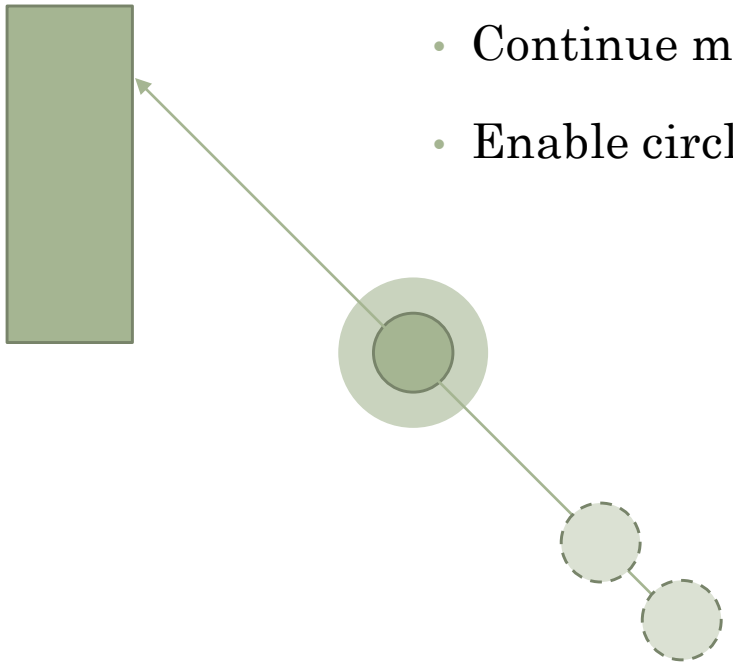
- Currently takes average of last two positions, and bases the velocity and moves the player forward.
- If the player doesn't send anything in 7.5 seconds it will be disconnected.

Improvements to Prediction



- Create a Circle collider around the player of some amount.
- While interpolating position, move forward on predicted path.

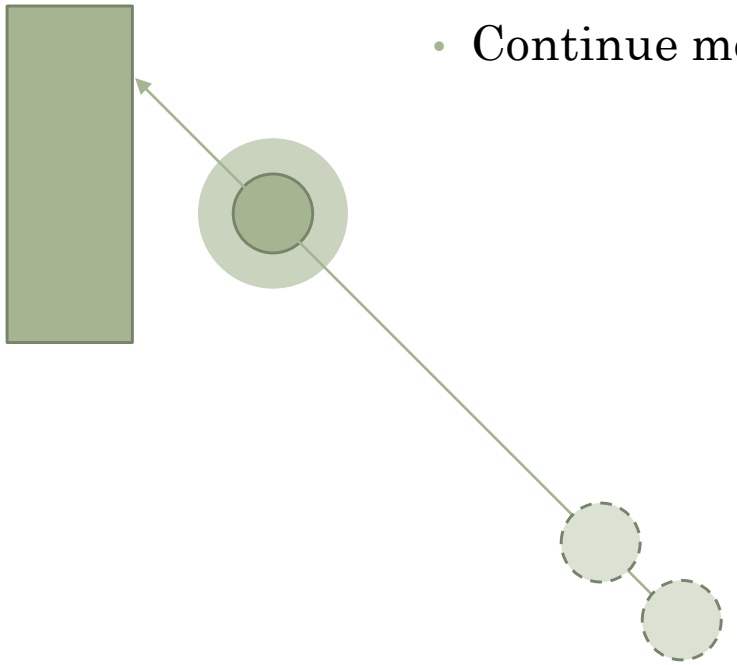
Improvements to Prediction



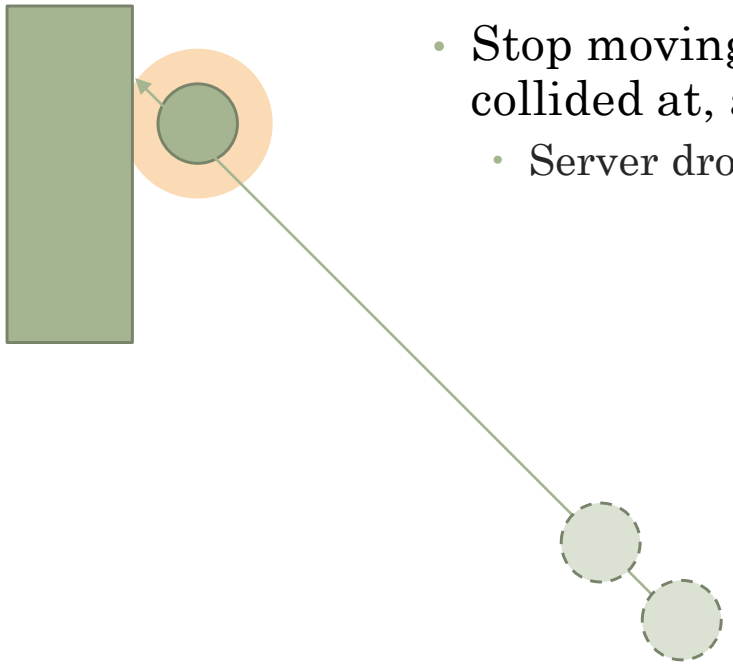
- Continue moving.
- Enable circle collider.

Improvements to Prediction

- Continue moving checking for collisions.

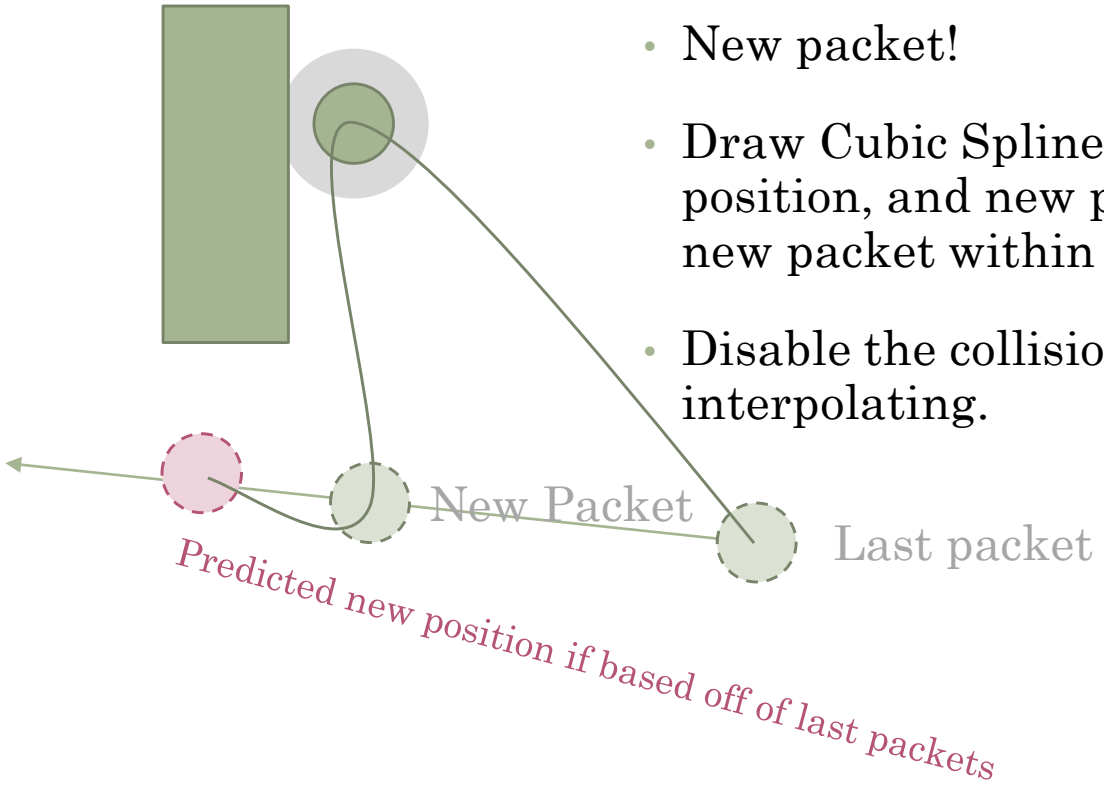


Improvements to Prediction



- Stop moving/interpolating when detected wall or thing collided at, and wait for new packets…
  - Server drops client automatically after 5 seconds.

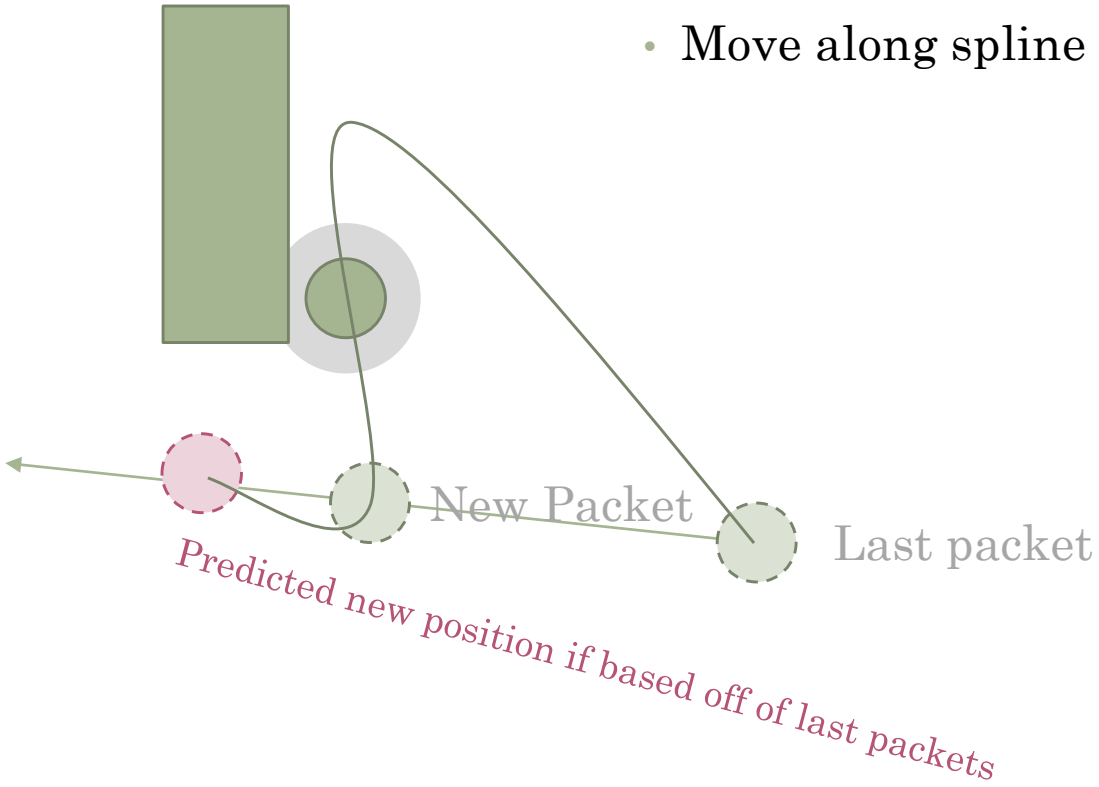
Improvements to Prediction



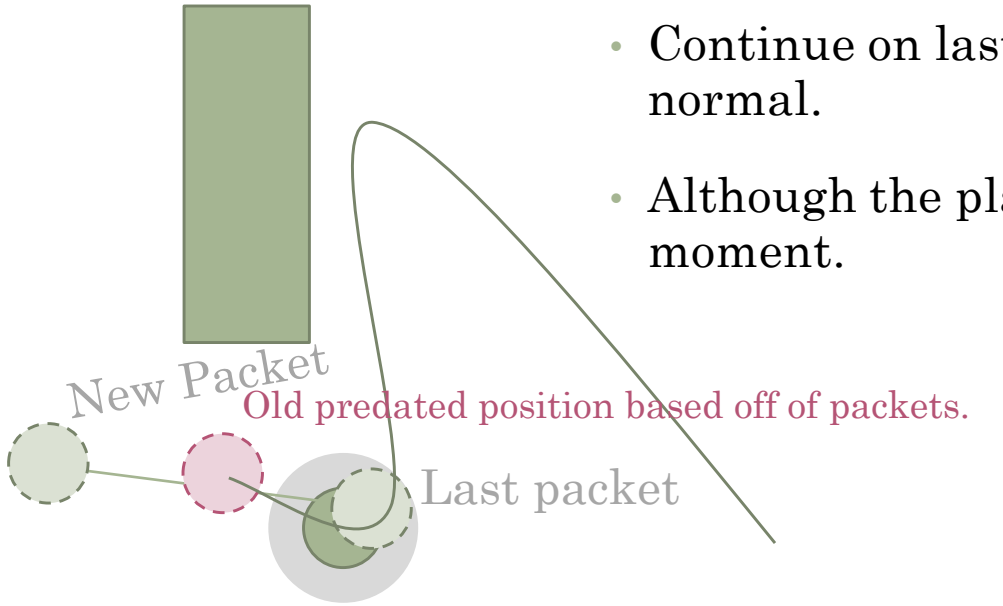
- New packet!
- Draw Cubic Spline between Last packet, current position, and new predicted packet, and interpolate to new packet within 30ms.
- Disable the collision constraint because no longer interpolating.

Improvements to Prediction

- Move along spline as normal.



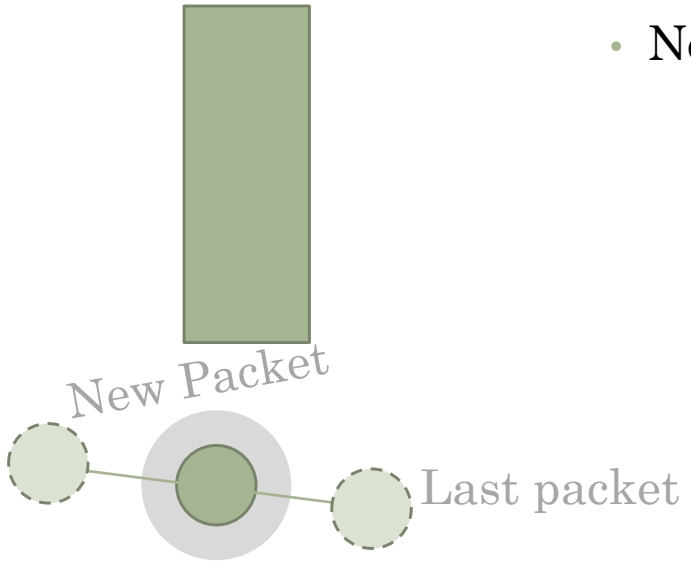
Improvements to Prediction



- Continue on last path until complete, then interpolate as normal.
- Although the players world are now desynced for a moment.

Improvements to Prediction

- Normal interpolation again!



Prediction Improvement Summary

- Change interpolation to use cubic splines (makes life easier – and transition neater).
  - Can easily predict next point.
- Add a collision circle and stop moving if hit something.
  - Triggered after 3 packets missed (90ms) behind.
  - Will behave strangely otherwise – stopping when near a wall.
  - Edge if player stops, and remains next to a wall when no packets are received.
    - Bounding box should be off.
- State machine is perfect for this.
  - INTERPOLATING
  - STOPPED
  - MOVING_ALONG_SPLINE

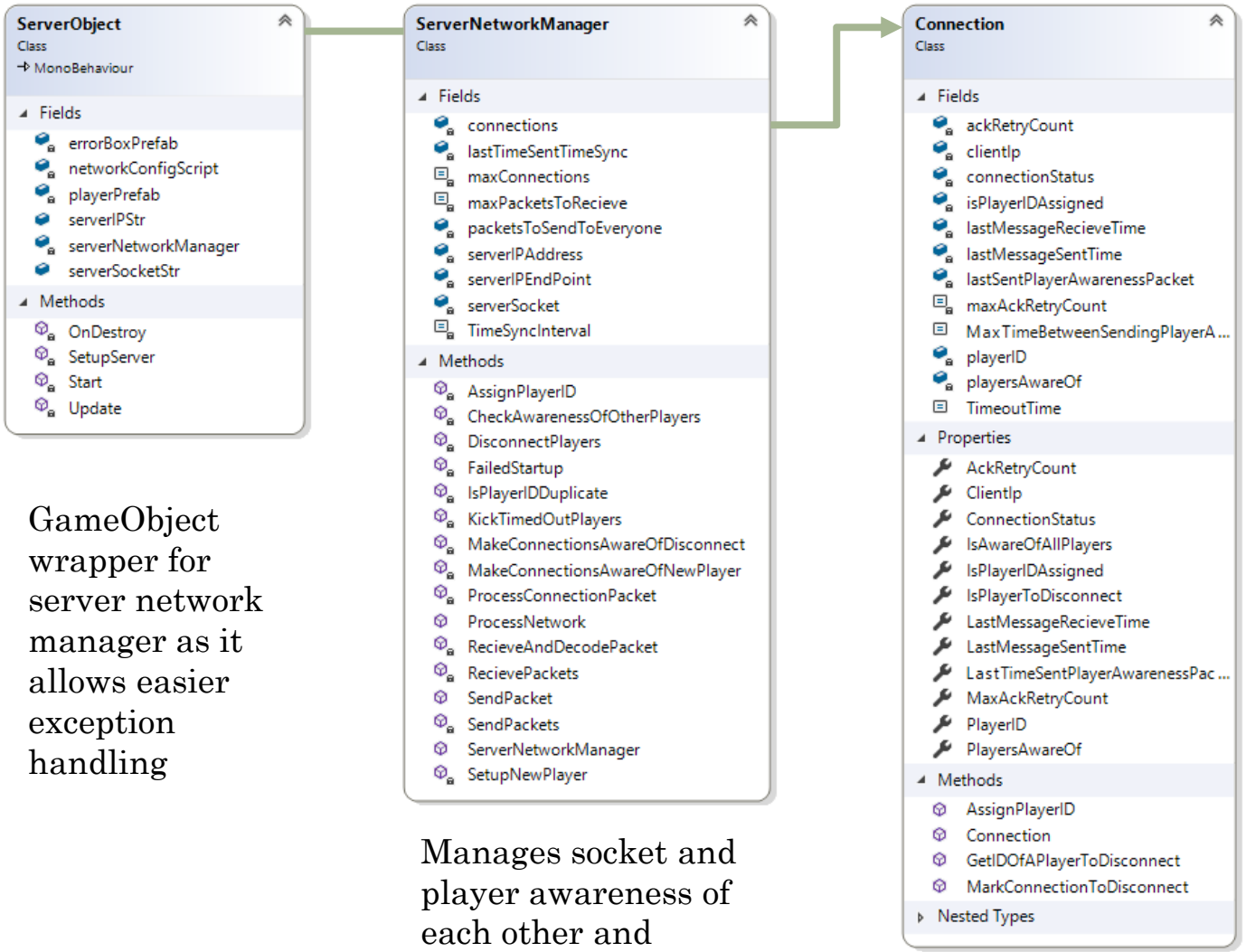
Improvements

- Overhaul of the way packets are into three distinct generic types mentioned at start.
  - Three way handshake.
  - Confirmation.
  - Unimportant.
- Implement ticks.
  - Bandwidth conservation – lesser load on socket.
- Firewall penetration.
  - Setup a way to say that the server is expecting connections at specified port.
- Time sync.
- Code wise:
  - Clean up ConnectionEventArgs.
  - Clean up namespaces.
  - Make a send/receive queue class. (Remove some responsibilities from ConnectionMgrs).
  - Connection Managers reusable across server and client.

Server UML

Wrapper for

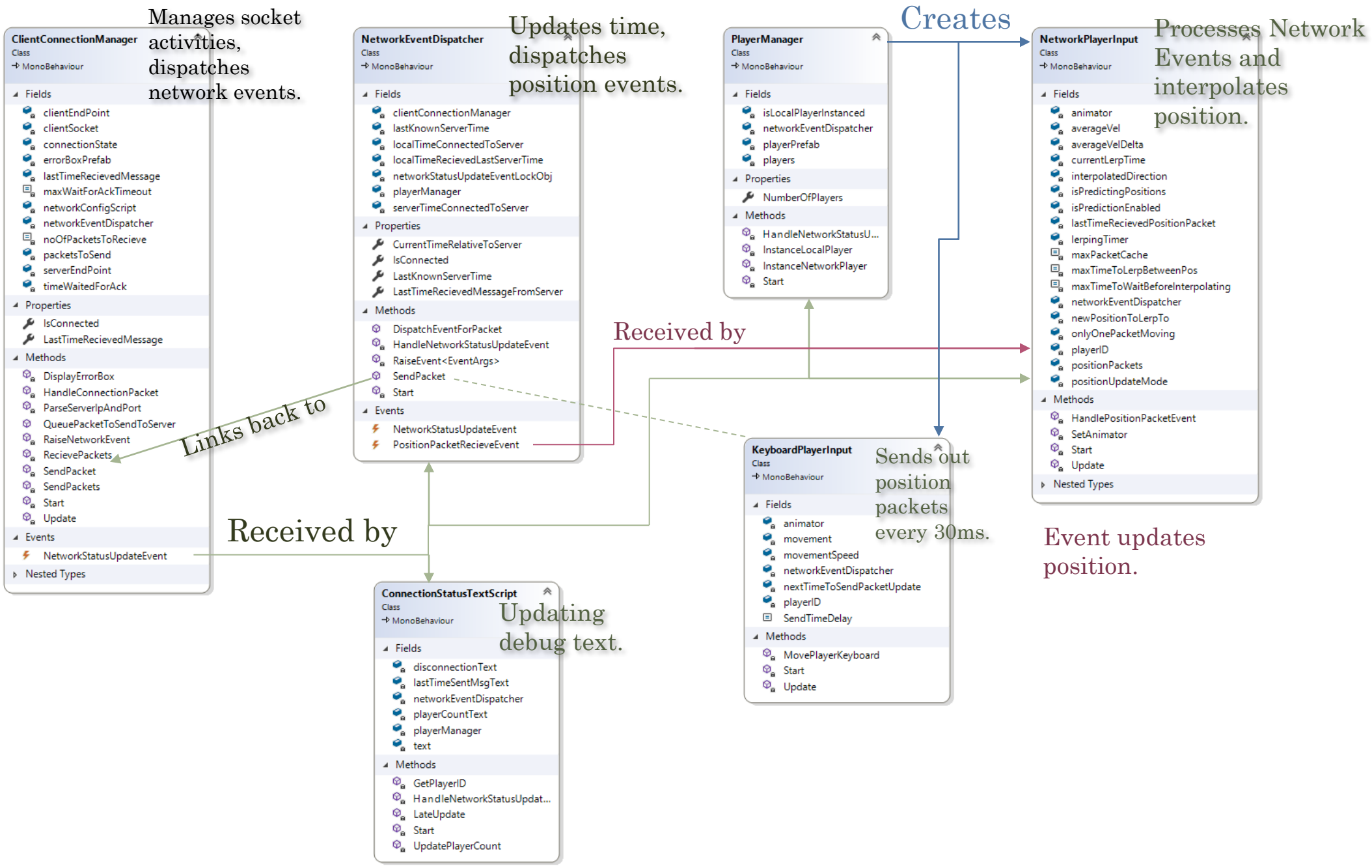
Contains



GameObject wrapper for server network manager as it allows easier exception handling

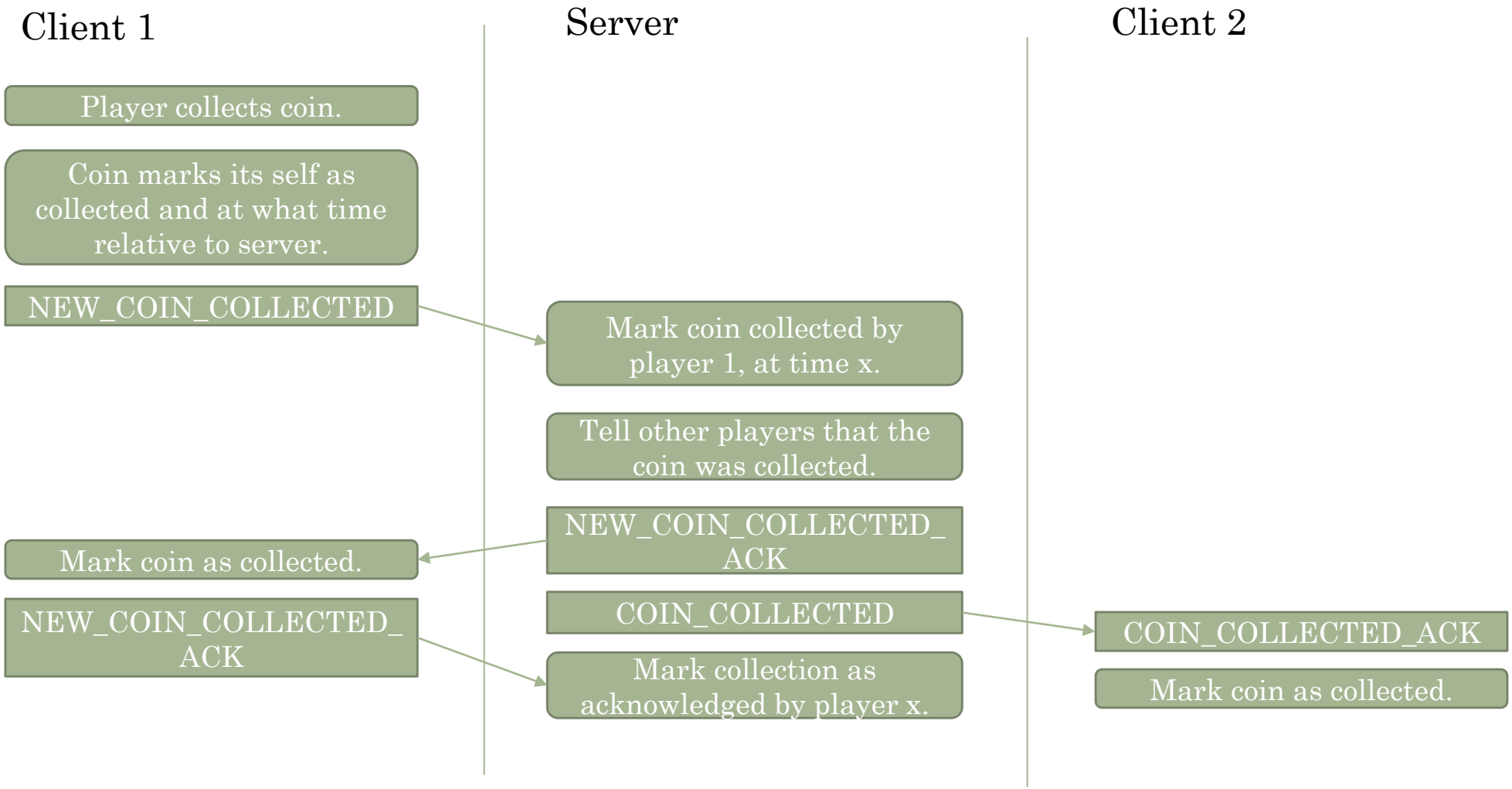
Manages socket and player awareness of each other and relaying positions.

Holds details about connection, and what other players its aware of.


Manages socket activities, dispatches network events.
Updates time, dispatches position events.
Creates
Processes Network Events and interpolates position.
Received by
Links back to
Received by
Sends out position packets every 30ms.
Event updates position.
Updating debug text.
ClientConnectionManager
Class
MonoBehaviour
Fields
clientEndPoint
clientSocket
connectionState
errorBoxPrefab
lastTimeRecievedMessage
maxWaitForAckTimeout
networkConfigScript
networkEventDispatcher
noOfPacketsToRecieve
packetsToSend
serverEndPoint
timeWaitedForAck
Properties
IsConnected
LastTimeRecievedMessage
Methods
DisplayErrorBox
HandleConnectionPacket
ParseServerIpAndPort
QueuePacketToSendToServer
RaiseNetworkEvent
RecievePackets
SendPacket
SendPackets
Start
Update
Events
NetworkStatusUpdateEvent
Nested Types
NetworkEventDispatcher
Class
MonoBehaviour
Fields
clientConnectionManager
lastKnownServerTime
localTimeConnectedToServer
localTimeRecievedLastServerTime
networkStatusUpdateEventLockObj
playerManager
serverTimeConnectedToServer
Properties
CurrentTimeRelativeToServer
IsConnected
LastKnownServerTime
LastTimeRecievedMessageFromServer
Methods
DispatchEventForPacket
HandleNetworkStatusUpdateEvent
RaiseEvent<EventArgs>
SendPacket
Start
Events
NetworkStatusUpdateEvent
PositionPacketRecieveEvent
PlayerManager
Class
MonoBehaviour
Fields
isLocalPlayerInstanced
networkEventDispatcher
playerPrefab
players
Properties
NumberOfPlayers
Methods
HandleNetworkStatusU...
InstanceLocalPlayer
InstanceNetworkPlayer
Start
NetworkPlayerInput
Class
MonoBehaviour
Fields
animator
averageVel
averageVelDelta
currentLerpTime
interpolatedDirection
isPredictingPositions
isPredictionEnabled
lastTimeRecievedPositionPacket
lerpingTimer
maxPacketCache
maxTimeToLerpBetweenPos
maxTimeToWaitBeforeInterpolating
networkEventDispatcher
newPositionToLerpTo
onlyOnePacketMoving
playerID
positionPackets
positionUpdateMode
Methods
HandlePositionPacketEvent
SetAnimator
Start
Update
Nested Types
KeyboardPlayerInput
Class
MonoBehaviour
Fields
animator
movement
movementSpeed
networkEventDispatcher
nextTimeToSendPacketUpdate
playerID
SendTimeDelay
Methods
MovePlayerKeyboard
Start
Update
ConnectionStatusTextScript
Class
MonoBehaviour
Fields
disconnectionText
lastTimeSentMsgText
networkEventDispatcher
playerCountText
playerManager
text
Methods
GetPlayerID
HandleNetworkStatusUpdat...
LateUpdate
Start
UpdatePlayerCount

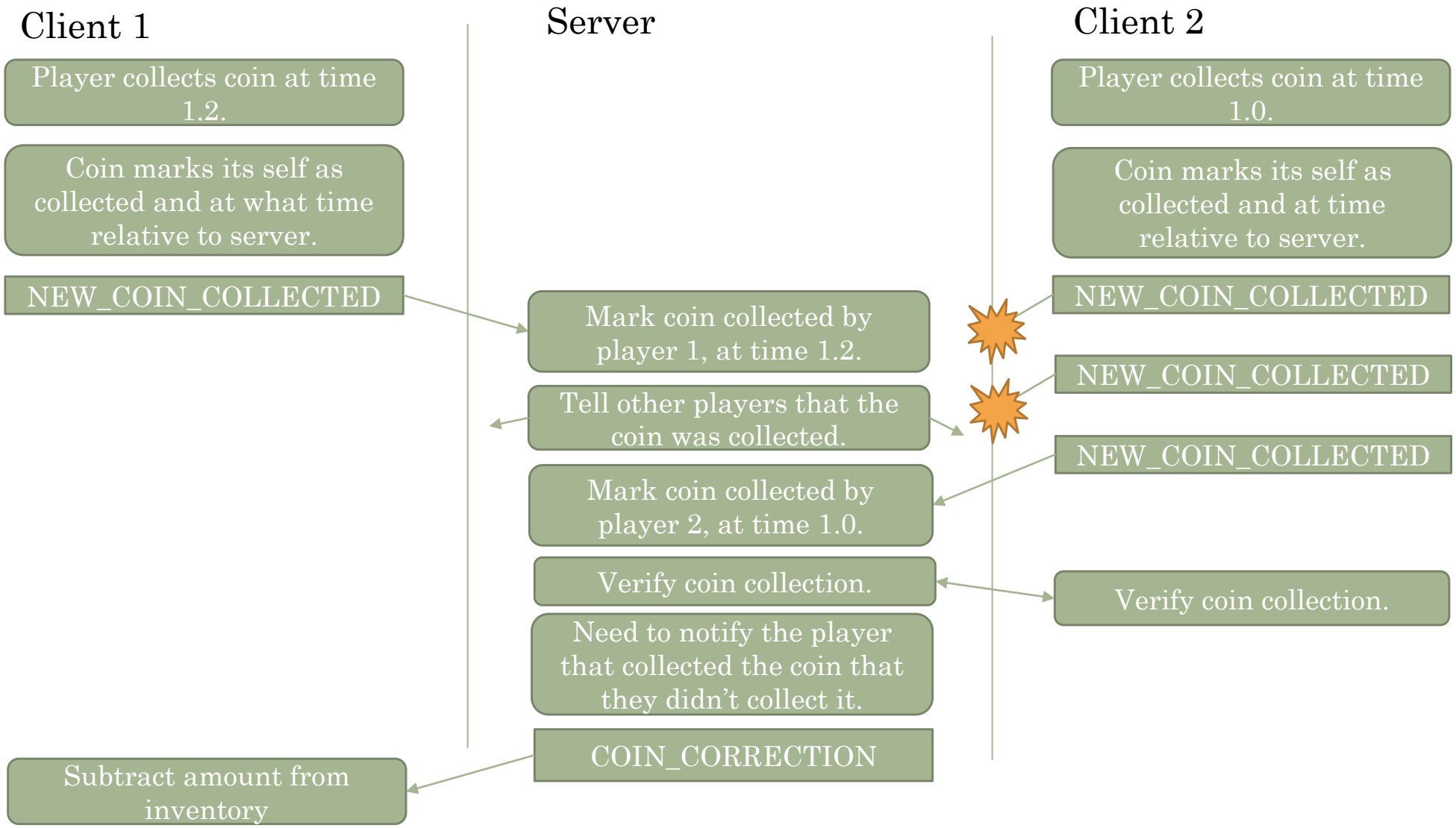
Implementing Coins

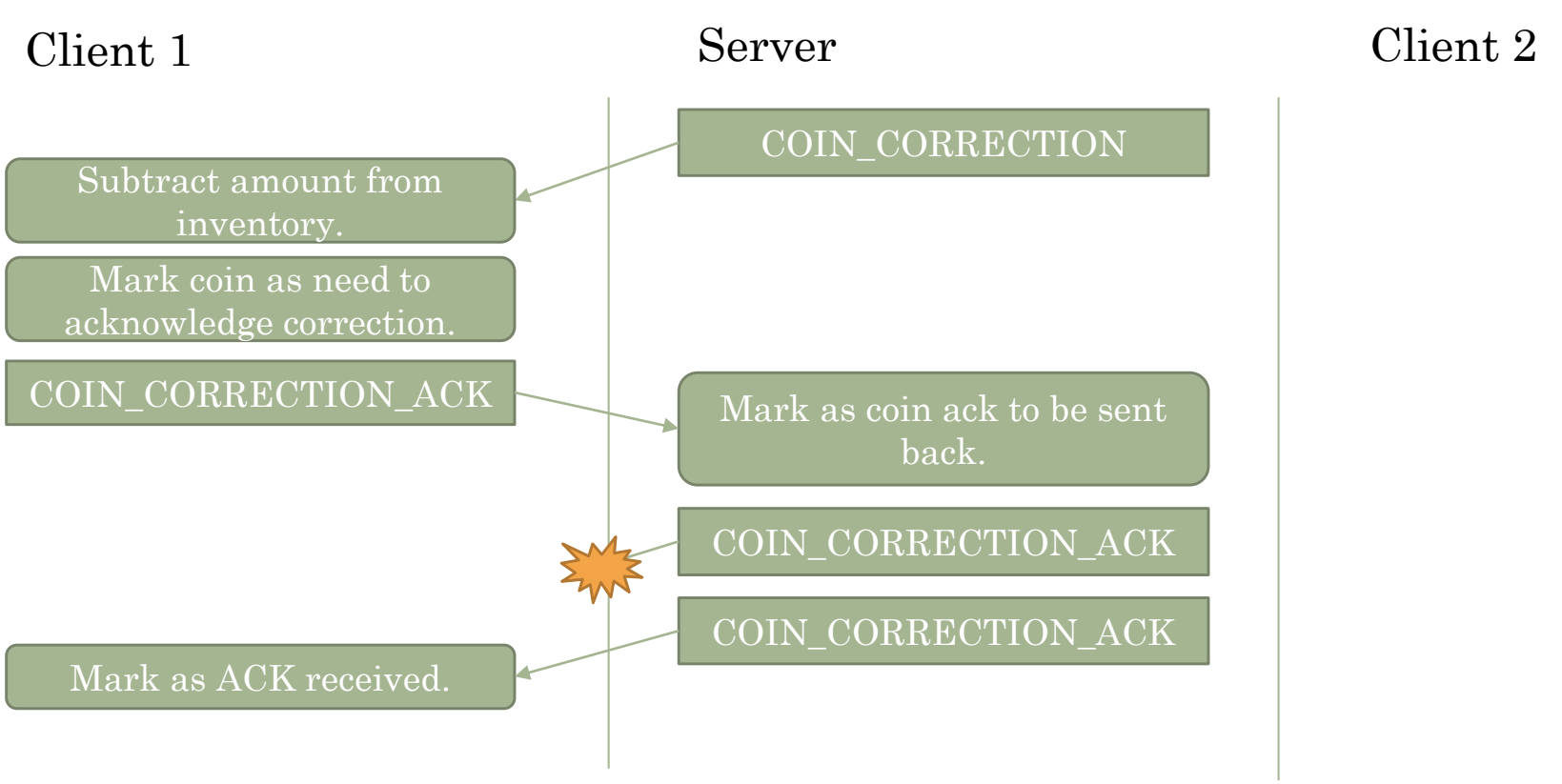
- Coins weren't added due to time constraints. Some inactive code is present.



Resolving Coin Collection Conflicts

- Previous example only was a typical example.





- Time is used to determine when a coin was collected.
- Additional freeze would need to be added if the player is trying to buy something it makes the client wait until it got a confirmation for sometime.
  - Would need to tell other client that if their sending confirmation took too long they don't get the coin.
- Uses both three way handshake and confirmation.

Cheating and Tampering

- Basic protections exist.
- Players' connections are indexed by their IPEndPoint/IP + port.
- Verify that the IP that sent packet the same one as the ID is assigned to.
- Verify that the IP is actually connected.
- Throw away packets that failed to decode.
- Nothing stopping attackers from pretending they're the other player/server and messing with timing.
  - Could be fixed with public/private (SSL) encryption.
    - Not that much overhead.
- Hackers hosting servers would be another risk – they have the master version of the game.
  - And who gets what packets, and who is connected or disconnected.
  - Not much can do except from encrypting memory and stopping with Denuvo or similar.