

Developing Multiplayer Games

Lars Kroll Kristensen
Director of contracting

The internet, and why it sucks

Arpanet invented
1969 - 4 nodes.

Evolve to IPv6
1995

Broadband goes
nuts

Evolve to IPv4
1981

WWW goes nuts



The internet, and why it sucks



Why Multiplayer ?

Why Multiplayer ?

- Play with or against other people

Why Multiplayer ?

- Play with or against other people
- Replayability

Why Multiplayer ?

- Play with or against other people
- Replayability
- Competition

Why Multiplayer ?

- Play with or against other people
- Replayability
- Competition
- Variation

Why Multiplayer ?

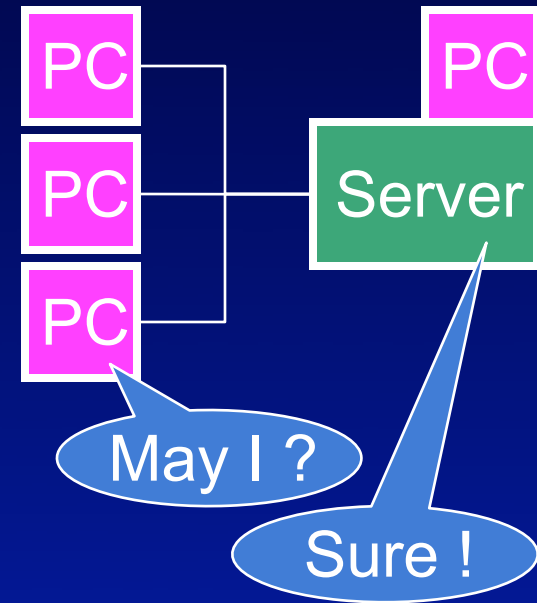
- Play with or against other people
- Replayability
- Competition
- Variation
- Boasting rights

Why Multiplayer ?

- Play with or against other people
- Replayability
- Competition
- Variation
- Boasting rights
- Community

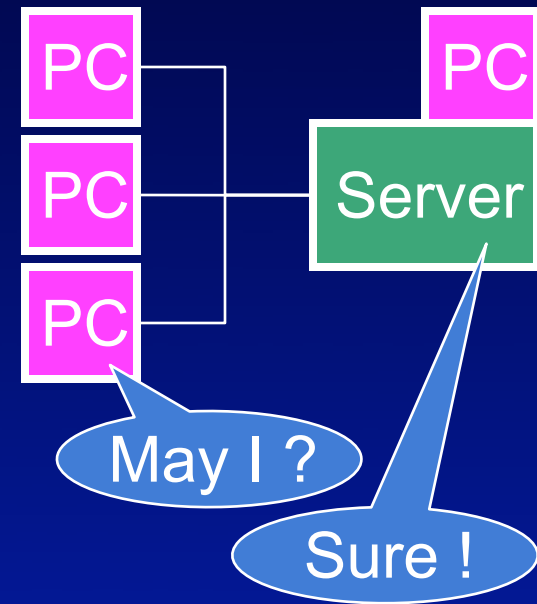
How ?

Authoritative server



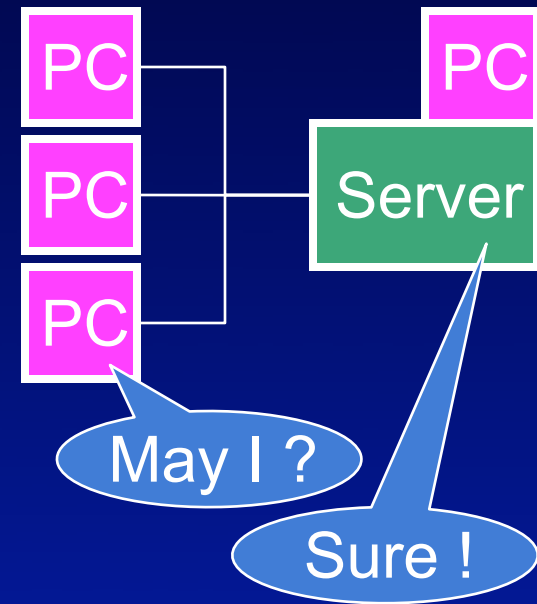
Authoritative server

- One machine arbitrates all disputes - the server



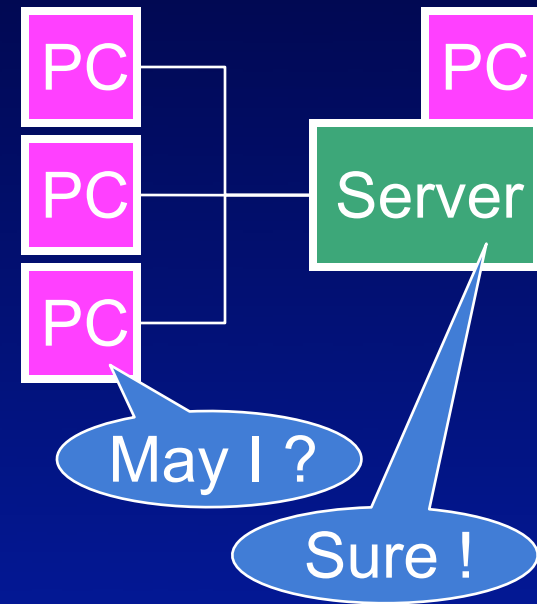
Authoritative server

- One machine arbitrates all disputes - the server
- “The server is always right”



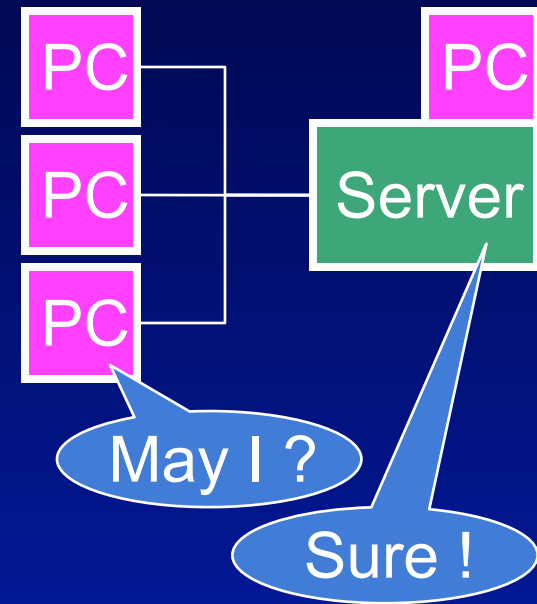
Authoritative server

- One machine arbitrates all disputes - the server
- “The server is always right”
- Server can also be a client

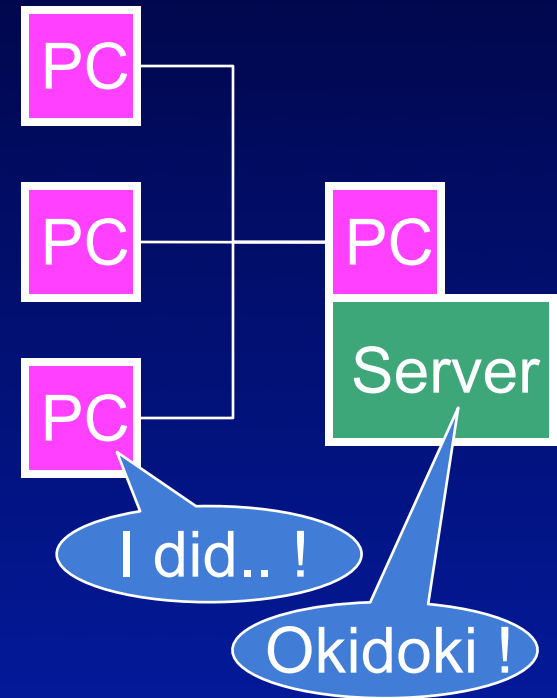


Authoritative server

- One machine arbitrates all disputes - the server
- “The server is always right”
- Server can also be a client
- Cheat counter measure

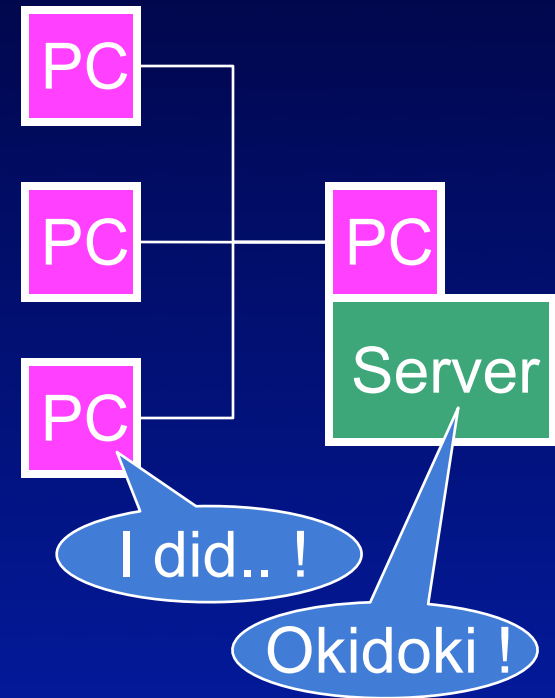


Decentralised authority



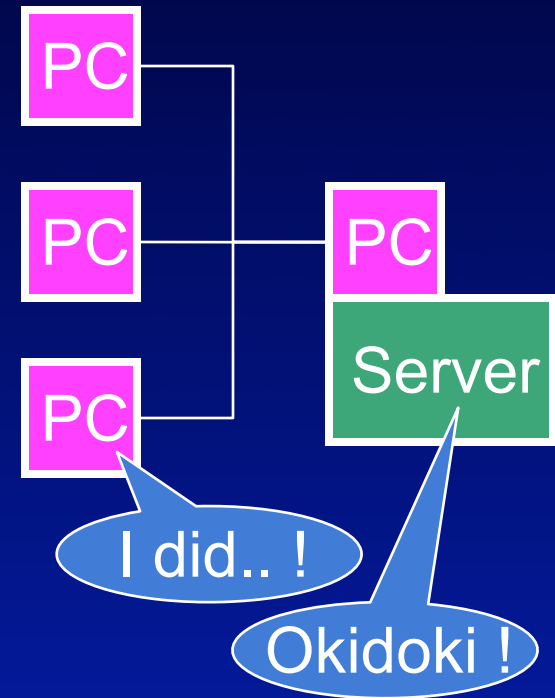
Decentralised authority

- One is still the server



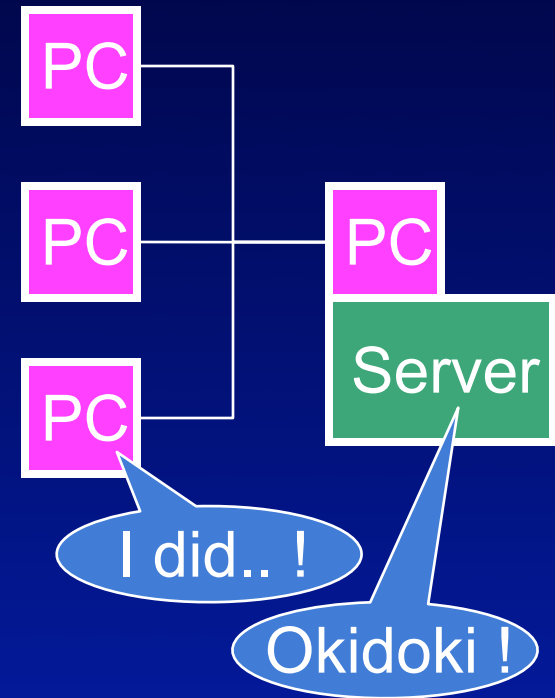
Decentralised authority

- One is still the server
- Different hosts control different objects



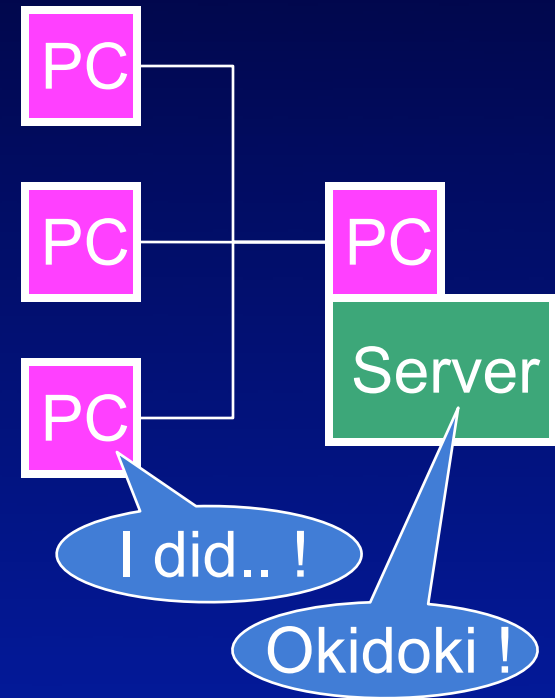
Decentralised authority

- One is still the server
- Different hosts control different objects
- Trust everyone equally. Arbitrate decentrally, but consistently



Decentralised authority

- One is still the server
- Different hosts control different objects
- Trust everyone equally. Arbitrate decentrally, but consistently
- Same code on all hosts



Connectivity

What is it ?

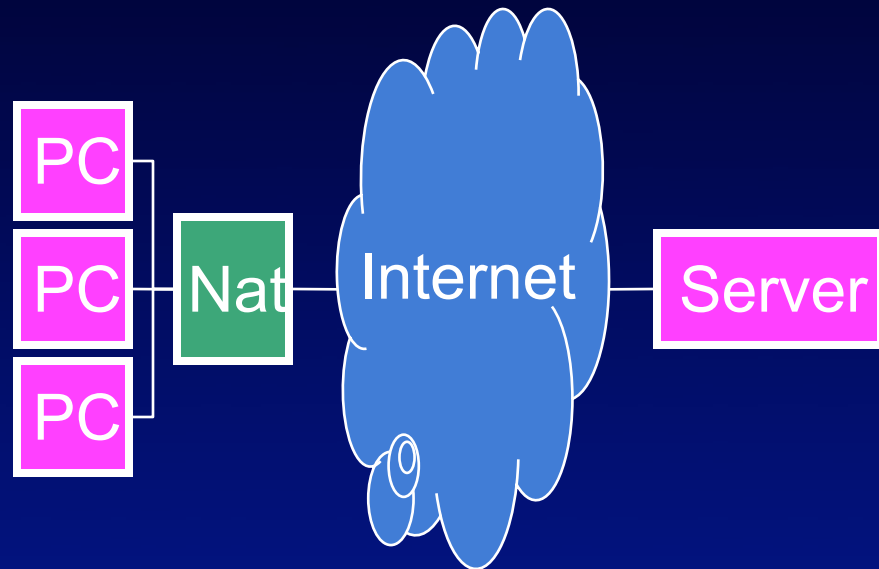
- Inability to create connections between clients
- Connections lost

How to solve it ?

- Buy better internet connection :-)
- Control the server - stay away from P2P
- NAT punch through
- Inform the player

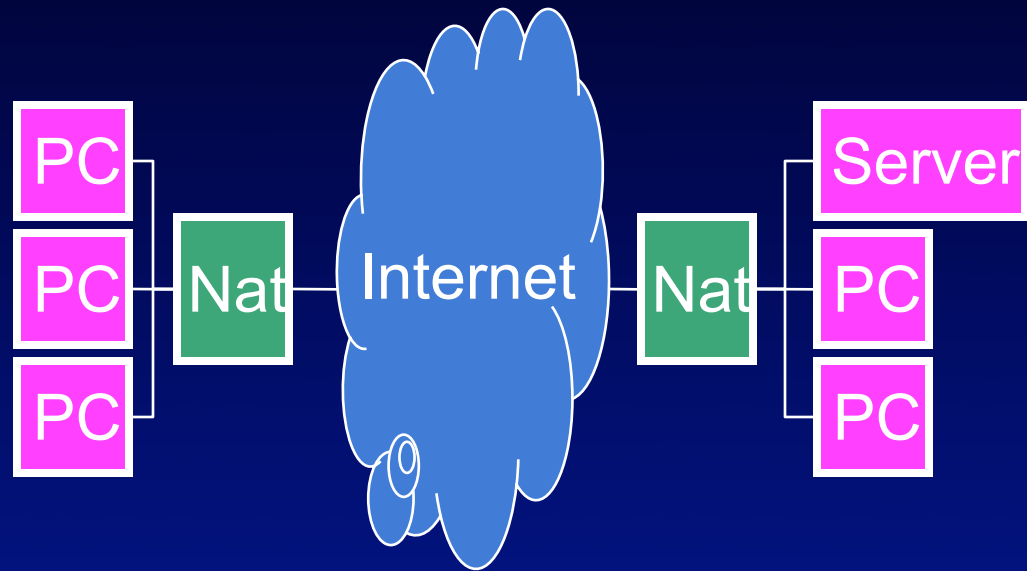
NAT

Too few
addresses
NAT: Network
Address
Translation



NAT

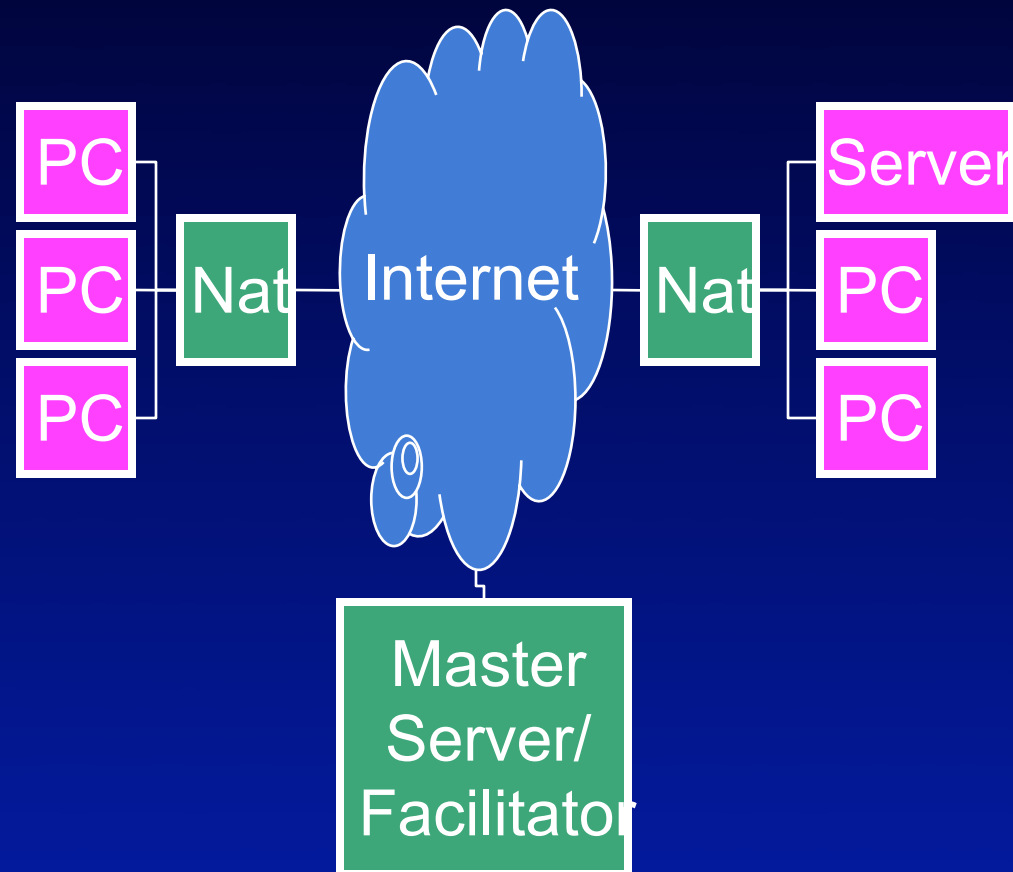
Server behind
NAT causes
trouble



NAT-punch through

Master server facilitates the contact, then play can proceed

Not all NATs support this :-
(Unity can test this for you :-)



Lag

What is it ?

- Delays created by server or client over load,
- Clients or servers waiting for each other
- Lag != Latency

How to solve it ?

- Buy better hardware
- Decrease computational load
- Parallelize where ever possible
- Cap maximum player count
- Decentralize decisions (let the client decide as much as possible).
- Decrease amount of communication

Latency

What is it ?

- Delays created by message travel time
- Latency != lag

How to solve it ?

- Buy better internet connection :-)
- Clever client side prediction
- Clever game design

Clever game design

- Allow, and accept that server and client *will* disagree
- Hide it.

- Turn based game - easy (chess, tic-tac-toe etc) Synchronisation in game design
- RTS (command sending “real time”) Synchronisation hidden in game design
- Most RPG (Diablo, WoW, EVE) Synchronisation hidden in game design

- FPS (Counterstrike etc) Synchronisation issues. Clever prediction
- Racing games (synchronisation of physics)

Network issues

Lag

Delays created by server or client congestion. Solve by improving server performance / decrease server work load

Latency

Delays created by network transport time. Solve by prediction and clever game design

Connectivity

Issues with client being unable to communicate/ losing connections NAT punchthrough, but also clever design

Network view

- Core concept in Unity networking
- Used to bind objects together across the network
- Identified uniquely by networkID
- Used for RPC calls and state synch

RPC in Unity

- Remote procedure call
- Like “SendMessage”
- Float, int, string.
- No return values.
- Buffered or unbuffered

```
@RPC
function PrintText (text : String)
{
    Debug.Log(text);
}

networkView.RPC ("PrintText",
    RPCMode.All, "Hello world");
```

RPC buffer

- Server remembers buffered RPC calls (and instantiates).
- When a new client connects, it gets the buffered RPC calls
- Cool for e.g. Spawning players
- Nice for permanent effects
- Do not buffer volatile data

RPC in Unity

- Remote procedure call
- Like “SendMessage”
- Float, int, string.
- No return values.
- Buffered or unbuffered

```
@RPC
function PrintText (text : String)
{
    Debug.Log(text);
}

networkView.RPC ("PrintText",
    RPCMode.All, "Hello world");
```

State Synchronisation

- Transforms
- Animations
- Rigidbodies (pos, vel, angVel, rot)
- OnSerializeNetworkView
- Drag'n'drop features

Network.Instantiate

- Facilitates instantiating objects on other hosts, with correctly setup network view (s).

Questions ?

- ...also ask Larus :-)