

8 - Multi-Player Games & Networking

Overview

Application side:

- Game Network Architectures
- Networked Virtual Environments
- Game Communication Protocols

Game Engine side:

- Networking support in RAGE
- Protocol Stacks
- Java Sockets
- UDP / TCP client and server tasks

2

Game Network Architectures

type	advantage	disadvantage
Peer-to-peer (all talk to each other)	sometimes simpler, no server needed	limited by slowest machine, easier to cheat/hack
Client/Server (one server per game)	more secure, centralized control	powerful server needed, single point of failure

Floating Server:

peer-to-peer, but one "peer" is the server

Distributed Server:

multiple services for managing a very large world

3

Networked Virtual Environments (NVE)

"A computer-based artificial world of 3D spaces and objects visited by geographically dispersed users who interact and collaborate with each other and with objects/entities local to the world" [1]

Common terms: **NVE, MMG, MMOG, MMORPG, Virtual World**

Many different types:

- Gaming
- Military
- Business / Training ("Serious Games", "performance engineering")
- Education

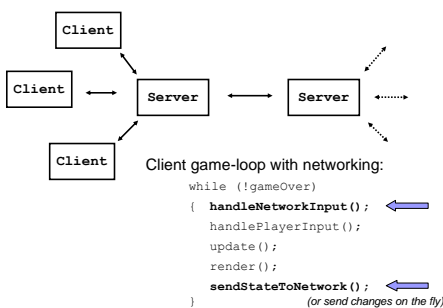
Examples: Overwatch, Counter-Strike, World of Warcraft, League of Legends, etc...

mmorpg.com (multi-player online game directory) lists thousands of MMO's.

[1] Davison, *Killer Game Programming in Java*, O'Reilly

4

Client-Server Organization



5

Game Communication

What to send?

- Entire world state is usually too much
- Can usually just send *user actions*
- also impacted by NPCs (we will see later)

"Fat Client" (each client runs world simulation)

Advantages:

- Fast local updates
- Server handles message-switching

Drawbacks:

- Code/data duplicated on each client
- Need to *synchronize* client worlds
- Clients must be *deterministic*

"Thin Client" (server runs world simulation)

Advantages:

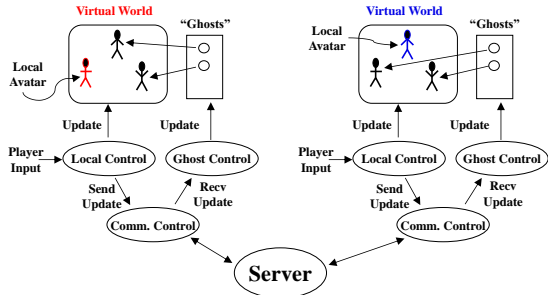
- Client can be *non-deterministic*
- Synchronization easier

Drawbacks:

- More network traffic
- Server bottleneck for all activity

6

NVE Fat-client Organization



Davidson, Killer Game Programming in Java, O'Reilly

7

Example Game Protocol

Messages from Client to Server :

- **CREATE (name, position)**
Informs server of a new world participant
- **MOVE/ROTATE (amount)**
Informs server about a change in a client avatar
- **DETAILSFOR (addr, port, position, orient)**
Informs server of local avatar position/orientation
(intended for forwarding to another client)
- **BYE (name)**
Informs server that client is leaving

Davidson, Killer Game Programming in Java, O'Reilly

8

Messages from Server to Client :

- **CREATE (name, position)**
Informs client that a new remote avatar exists
- **MOVE/ROTATE (senderName, amount)**
Informs client of a change in status of a remote avatar
- **WANTSDETAILS (addr, port)**
Informs client that a remote client wants a local status update
- **DETAILSFOR (senderName, pos, orient)**
Provides client with updated status of a remote avatar
- **BYE (name)**
informs all clients with the name of a client who quit

9

Example Protocol Processing

Server Handling:

- **CREATE (name, position)**
 - Save name and corresponding IP/port
 - Forward **CREATE** message to all other clients
 - Send **WANTSDETAILS (addr, port)** to all other clients
- **MOVE/ROTATE (name, amount)**
 - Look up sender name
 - Send **MOVE/ROTATE** message to all other clients
- **DETAILSFOR (addr, port, position, orientation)**
 - Look up sender name
 - Send **DETAILSFOR (sender, pos, orient)** to <addr, port>

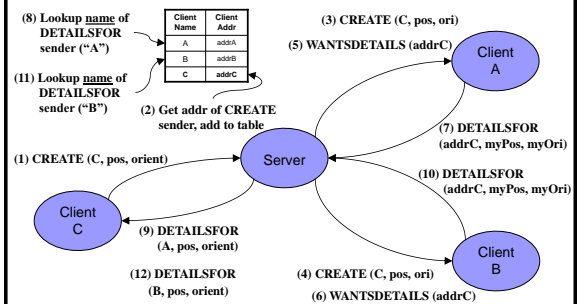
10

Client Handling:

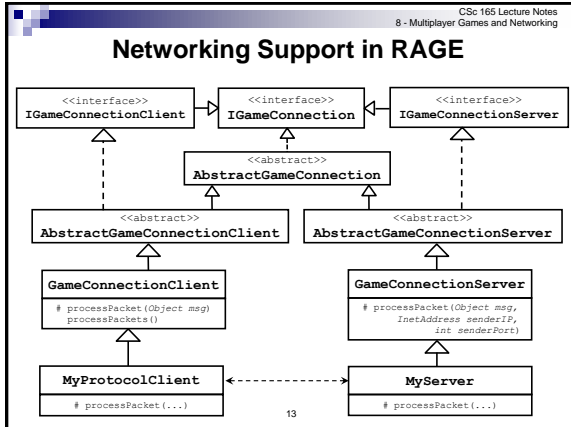
- **CREATE (name, position)**
 - Create Ghost avatar
- **MOVE/ROTATE (senderName, amount)**
 - Update Ghost avatar for sender
- **WANTSDETAILS (addr, port)**
 - Get local avatar position/orientation
 - Send **DETAILSFOR (addr, port, mypos, myorientation)**
- **DETAILSFOR (senderName, position, orientation)**
 - Update Ghost avatar for sender

11

Sample Protocol Sequence



12



CSc 165 Lecture Notes
8 - Multiplayer Games and Networking

Networking in RAGE (continued)

server side:

- extend `GameConnectionServer`
- override constructor, calling parent class constructor with port and type (typically UDP)
- override `processPacket (...)`, implementing desired protocol
- if the message represents a first message from a new client, use `addClient (...)`, getting the client's info using `createClientInfo (...)`
- To send messages to clients, use `sendPacket (...)`, or `sendPacketToAll (...)`, etc.

client side:

- extend `GameConnectionClient`
- override constructor, calling parent class constructor with address, port and type.
- override `processPacket (...)`, implementing desired protocol. This method is called each time another node (typically the server) sends it a message. The message is contained in a java Object passed as a parameter (typically a string).
- The game's update() method should call `client.processPackets (...)`

14

CSc 165 Lecture Notes
8 - Multiplayer Games and Networking

NVE issues

Naming

- Need some way to identify clients uniquely

Server Discovery

- Clients need a way of finding servers

Synchronization

- Need to keep the various simulations "in step"

15

CSc 165 Lecture Notes
8 - Multiplayer Games and Networking

Creating Unique Names

Class `UID` ("Unique ID")

- Unique with respect to the current host
- Components:
 - Time, VM ID, Counter

Class `UUID` ("Universally Unique ID")

- 128-bit value, guaranteed unique world-wide
- Components:
 - Time, VM ID, Counter, MAC address
- Potential security issue: publishing MAC addresses
 - Partial solution: random UUIDs

```

UUID newID = UUID.randomUUID();
  
```

16

CSc 165 Lecture Notes
8 - Multiplayer Games and Networking

Synchronization issues

- Latency can cause "freezing"
- Solution: *data prediction*
 - Clients retain position "history" for ghosts
 - Extrapolate "likely next position" from history (typically use a quadratic polynomial)

17

CSc 165 Lecture Notes
8 - Multiplayer Games and Networking

Correcting Prediction Errors

18

Server Discovery

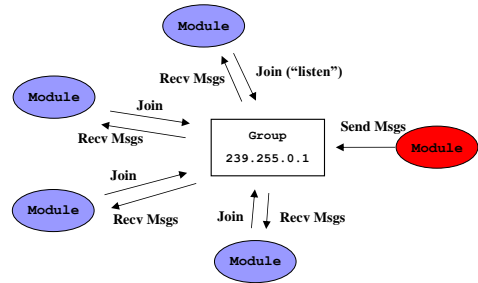
Client must know how to contact server

Various approaches:

- o "Hard-coded" in client
- o Player-provided to client at startup
 - Command line
 - Interactive prompt
- o "Multicast" groups

19

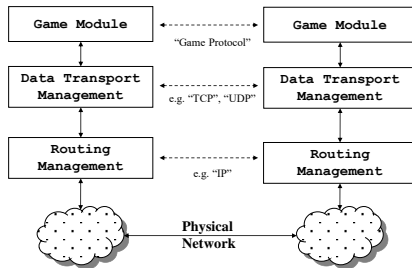
Multicast Groups



20

Networking Internals

Protocol Stacks



21

TCP vs. UDP

- **TCP** – sender verifies packet is received
 - more reliability
 - good for file transfers
- **UDP** – sender just sends; no verification of receipt
 - faster speed
 - good for real-time broadcasts, or games

22