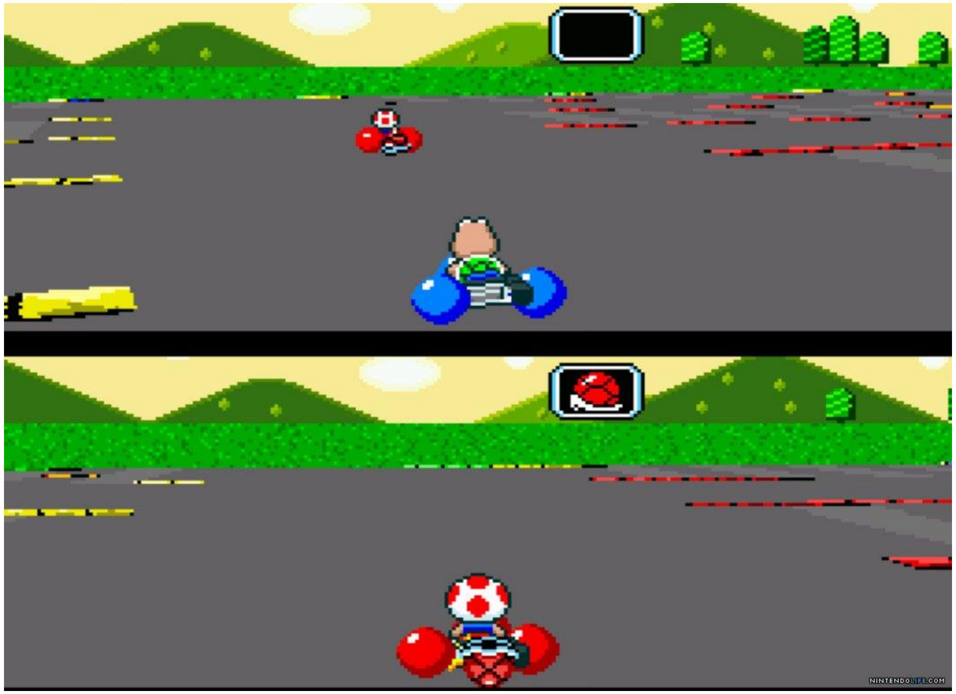
Game Technology

Lecture 12 – 19.12.2015
Multiplayer Games



Super Mario Kart (1991)

Dipl-Inf. Robert Konrad
Dr.-Ing. Florian Mehm

Prof. Dr.-Ing. Ralf Steinmetz
KOM - Multimedia Communications Lab

TECHNISCHE UNIVERSITÄT DARMSTADT

Short multiplayer history

First games – Local multiplayer

- AI not yet ready for use
- Simple to implement
- Lower hurdle for players who don't know video games (= everyone in 1970 ;-)

Pong (1972), Computer Space (1971)

Flash Attack (1980)

Described by Ken Wassermann and Tim Stryker in BYTE, December 1980

https://www.youtube.com/watch?v=9RutllBwoiA

http://archive.org/stream/byte-magazine-1980-12/1980_12_BYTE_05-12_Adventure

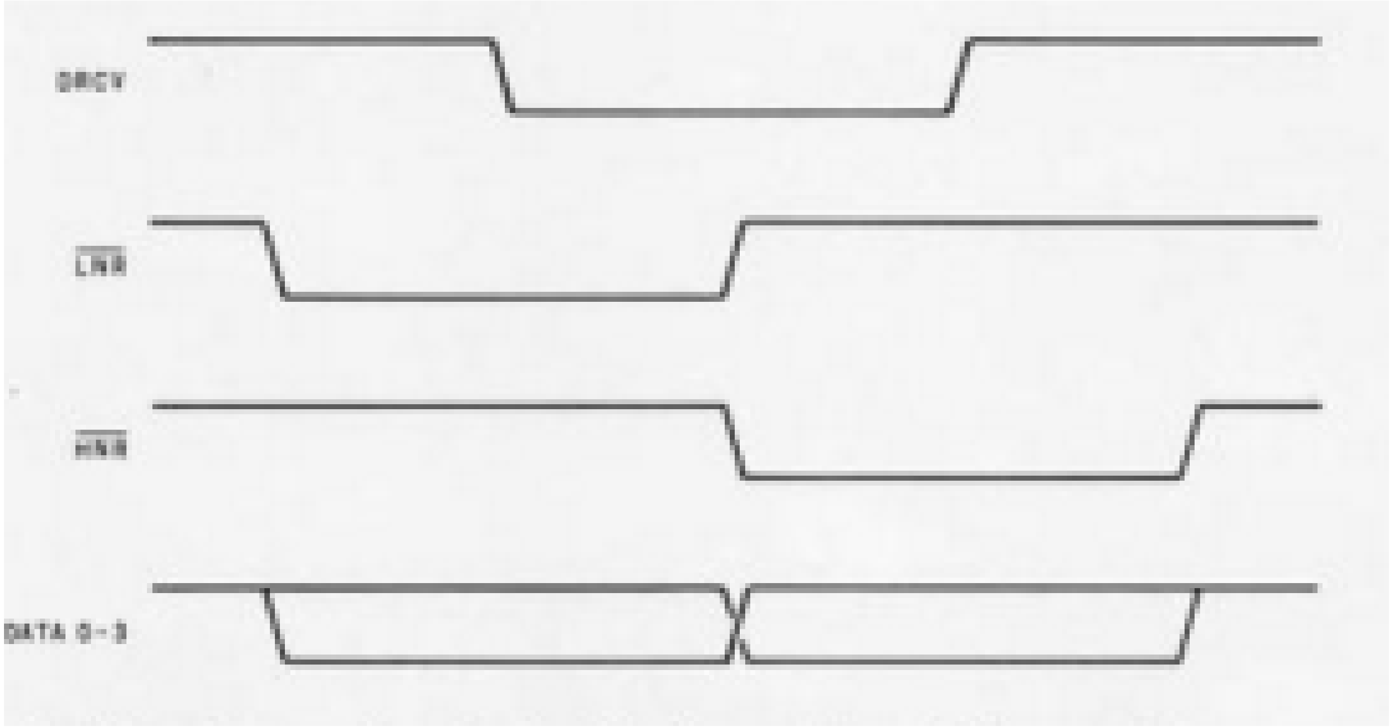
Parallel port multiplayer

Userport (8 bit parallel communication)

Commodore PET (1977)

Parallel port multiplayer

- 2 programs need to coordinate when the bus is used for reading and writing
- Very limited communication possible



MIDI Maze (1987)

Atari ST, Up to 16 players connected via MIDI ports

MIDI Maze GameBoy Port

Faceball 2000 (1991)

Supported 16 player multiplayer (only GB game)

Required 7 4-player adapters (requirement by Nintendo – developers had developed a custom solution for the game)

Doom (1993)

Peer to peer multiplayer

Keyboard commands sampled at tics (1/35 s) and sent to all players

Game proceeds when received inputs by all players

Negative acknowledgements: If tic numbers do not match up, resend

Quake (1996)

Client/Server with no prediction

QuakeWorld (1996)

Update to allow internet multiplayer for Quake

Client/Server with Client-Side prediction

LAN gameplay (1990s) Metrics

Why the switch from Quake to QuakeWorld?

10Base2 Ethernet

- Latency: Minimal
- Bandwidth: 10 Mbps
- Packet loss: Almost non-existent
- Jitter: Almost none
- Fury at the player who interrupted the connection: endless

“an elegant weapon for a more civilized time”

Internet

TECHNISCHE UNIVERSITÄT DARMSTADT

Study by Bungie in 2007

Baseline for 99% of Xbox ownwers

- Latency: 200ms one-way (ping of 400)
- 10% jitter (consistency of the connection – rate of packets arriving same as sending)
- Bandwidth: 8KB/s up, 8 KB/s down
- Packet loss: Up to 5%

→ **Very different challenges**

→ LAN: Low latency, large bandwidth, reliable (except for people stumbling over cables…)

→ Internet: High latency, smaller bandwidth, jitter, unreliable

Multiplayer architectures

Number of players

Networking technology

Gameplay implications

- Social factors
- Network metrics
- Gameplay requirements

The Simpsons Arcade Game (1991)

One computer, multiple players

Trivial implementation

No latencies

Uncompressed realtime 3D video chat

Saturn Bomberman (1996)

Local multiplayer

- Screen space restricted
- Number of controllers restricted
- Number of locally available players who understand Bomberman severely restricted


www.kewlwallpapers.com

Peer-to-Peer Lockstep

Synchronizes game step by step

- Send command data (go forward, move unit,…)
- Receive commands by all other players
- Simulate game step on all computers
- Repeat

Example structure

```
struct MovementCommand {
  unsigned int UnitID;
  float targetLocation[2];
};

size_t s = sizeof(MovementCommand);          //12 Bytes
```

Real-time strategy games about 1 command every 1.5 – 2s

1 command / 1.75 s

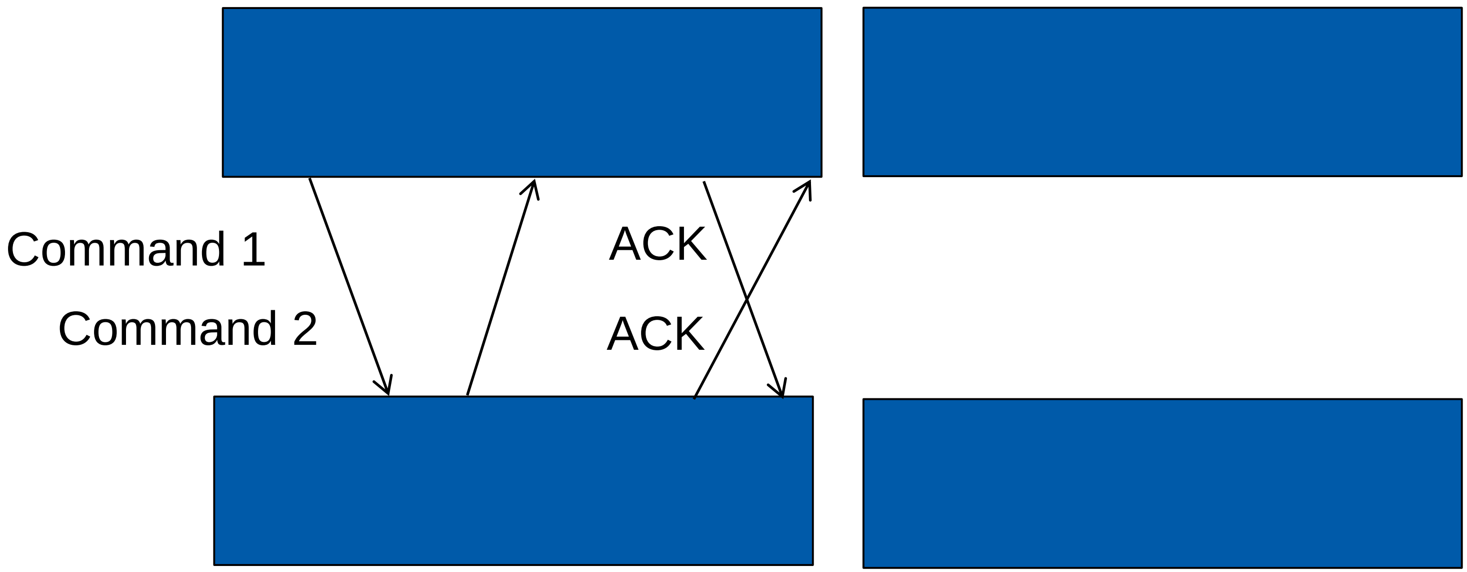
1/1.75 commands per second --> 6.86 Bytes per second per Player

With 8 players: 54.86 Bytes per second

Turns

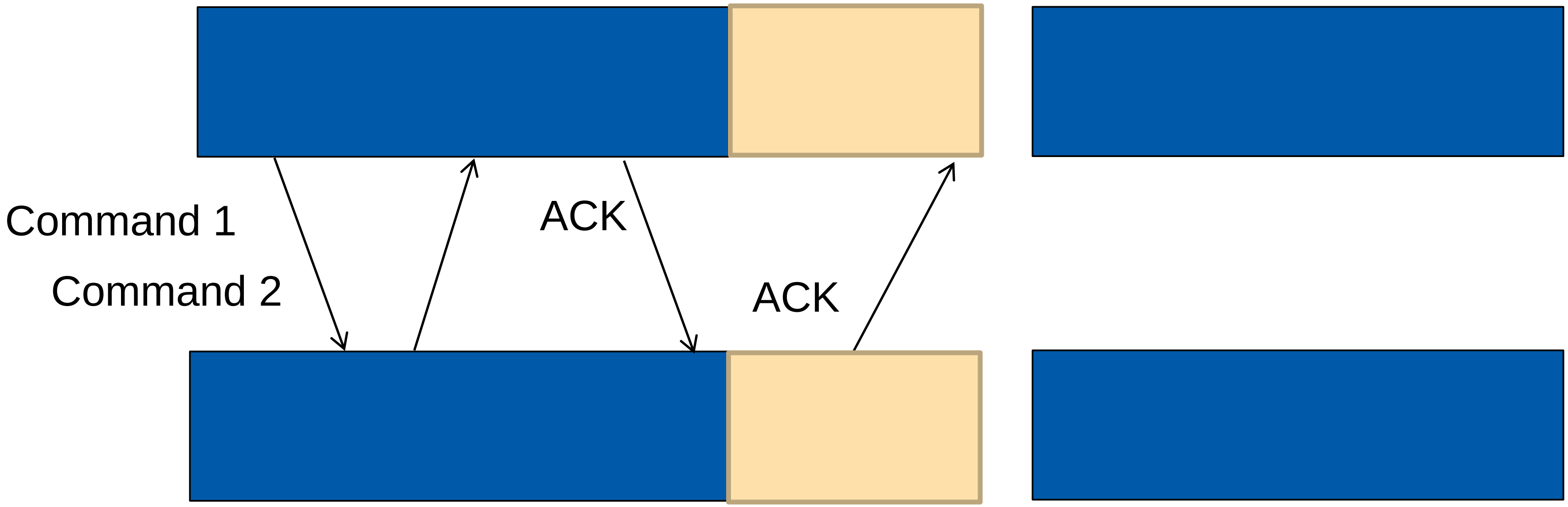
Player 1 and player 2 send a command each

Game continues when all commands are sent and received



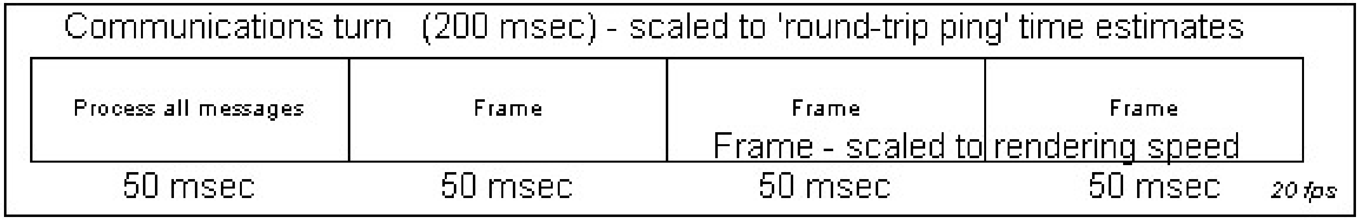
Turns

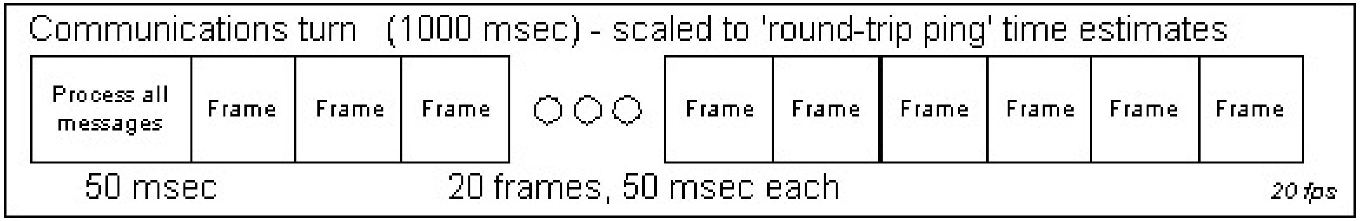
Player 2 is slow → Game runs slower

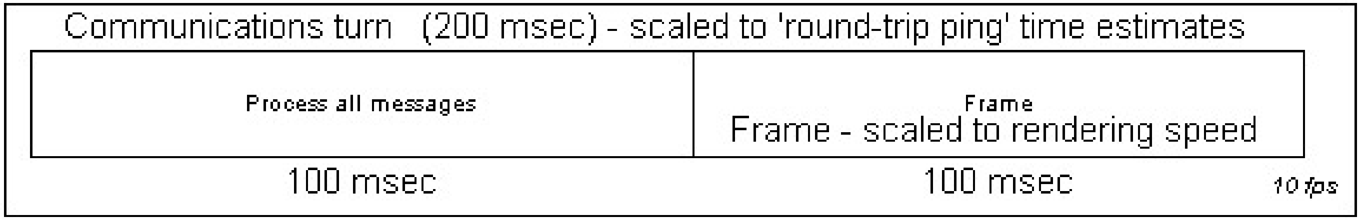


Adjustment of turn lengths

Take the ping and the capabilities of the slowest machine into account – measure constantly and adapt







http://www.gamasutra.com/view/feature/3094/1500_archers_on_a_288_network_.php?print=1

Pro & Contra

Low data rate

- Just high level game commands

Very fragile

- Requires complete determinism
- Requires every client to reliably send data
  - One client hangs -> the game hangs

Maximizes latency

- Game has to wait for every one

Players can‘t join a running game easily

- Would have to rerun all previous game commands

Debugging

TECHNISCHE UNIVERSITÄT DARMSTADT

Desynchronization errors

Serialize game states

- Maybe already needed for replay, save games, …
- Exact, allows resetting the state, debugging
- Larger sizes for snapshots

Implement hashes for game states

- Containing everything relevant to the game
- Ideally can do this quickly
- Small memory footprint

Serializing and debugging

Manually

- Each object can implement functions for Serialization and Deserialization, write and read relevant data

Save memory layout

- Simple, but can break easily
- References need to be fixed

Reflection system

- Not part of C++ natively

Finding which part of the state is corrupted

- During deserialization, compare the states
- Assert at the point where the states differ

Determinism

Make sure to separate between core and other parts

Core: Everything needed to calculate relevant game state

Advantages

- Can determine the game state easier
- Explicit which code needs to have network in mind
- Eleminate cross-talk

Cross-talk

- Imagine a random animation component
- float nextValue = rand(minValue, maxValue);
- Depends on frame rate
- → Might or might take a random value away

Determinism

Randomness

- Save your seeds
- Implement your own rand()
- Done

Calculations

- Integer calculations - easy
- Floating point calculations – a little weird
  - Different optimizations on different compilers
    - There is usually a „strict IEEE 754" option
  - Different CPUs
    - x86 calculates in 80bits, then rounds to 32/64 bit
  - …

Hashing

TECHNISCHE UNIVERSITÄT DARMSTADT

Ideally

- Fast
- Captures all relevant information
- Few collisions (different game states with same hash)

Zobrist Hash

- Developed for chess programming
- Generate a random number for each piece on each field
  - White pawn on A1: 8B8A 616B 8587 1AB6
  - Black pawn on A1: 83AB C69D 2933 4FEC
  - ...
- Encode a state as the XOR combination of all field states
  - A1 XOR A2 XOR ....

Peer-to-Peer Lockstep Today

TECHNISCHE UNIVERSITÄT DARMSTADT

Still used in strategy games

- Even realtime strategy

Not used in action games

- Because the internetz

Game design tricks used to hide latency

- Play an animation/sound immediately
- Move units after all clients agreed
- But: The longer the own units take to react, the more apparent it becomes

"More Work?" – Warcraft 3, 2002

Similar tricks used to hide AI calculations

Client/Server

Server controls everything

Clients are like terminals

Complete game runs only on the server

- Clients send game commands
- Server sends game state

Game State

```
struct {
  vec3 Position;
  vec3 Rotation;
  AnimationID Animation;
  float AnimationState;
}
```

For each player

Server

Simulates the complete game

- Everything that‘s relevant for the game state
- Including physics
- Not including cosmetics like particle effects

Does not depend on clients

- Clients can hang
- Clients can drop in and out
- Does not result in problems for other clients

Client

TECHNISCHE UNIVERSITÄT DARMSTADT

Really dumb client

- Reads input, sends it to the server
- Does not actually run the game
- Just interpolates received game states
- Might run some simulations for effects work
  - Menu animations
  - Particle effects
  - Physics which do not interfere with gameplay
  - …

Interpolation

Client/Server can feel very stop-and-go

Players see individual frames as they come in

Interpolate between states

Pro & Contra

TECHNISCHE UNIVERSITÄT DARMSTADT

Very robust

- Clients can hardly cause any problems
- Lags from one client do not propagate to other clients
- No cheating

Very laggy

- Everything lags
  - Even basic movement lags
  - The server simulates every player
- Size of game state has to be rather small

Client/Server today

Outdated

Client/Server with Client-Side Prediction

Mix of Client/Server and a little bit of Peer-to-Peer

Server is still the boss

- But clients predict the game state

Prediction

King’s Quest V - 1990

Prediction

Just run everything on the client and the server

- But no client-client-communication
- Determinism helps

Most of the time, predictions should be correct

- At least for the player character himself
- Makes controls snappy

For other players pure prediction

- Often incorrect

Failed Predictions

Failed Predictions

Use the corrected data

- Cause the server is the boss

Hide your mistakes

- Interpolate visuals to avoid jumps
- Or let stuff jump around when out of view

Failed Predictions

Clients receive only old data

Compare old received data and old predicted data

- When prediction was wrong
  - Recalculate new current state based on received old state
  - Then interpolate

Failed Predictions

Can cause unfair situations

- Visuals show that an enemy was hit but he really wasn‘t

No real solution possible

- Virtual life is not fair :-(

Physics States

Excellent series of blog posts: „Introduction to Networked Physics“ by Glenn Fiedler

http://gafferongames.com/networked-physics/introduction-to-networked-physics/

GDC Talk available to watch: http://gafferongames.com/2015/04/12/networking-for-physics-programmers-is-now-free-to-view-in-the-gdc-vault/

Also well suited to recap the architectures

Lockstep, Determinism

Effects of lacking determinism

→ Random number generation not synchronized

Lockstep, Determinism

Simulation with fixed determinism

Client/Server

Client/Server

Client/Server with Interpolation

Network Protocols

All IP based

Everything just works like the internet

Much more information

- Communication Networks lectures
- Multimedia Communications Lab (KOM)

IP

TECHNISCHE UNIVERSITÄT DARMSTADT

Internet Protocol

Packet based

- No direct connections
- Much like post packages
- Unreliable

TCP/IP

TECHNISCHE UNIVERSITÄT DARMSTADT

Transmission Control Protocol

Direct connections

Reliable streams of data

Super easy

TCP/IP

Builds on a package based protocol

Makes sure every package arrives

Makes sure all packages stay in the same order

TCP/IP

Reorders packages

Requests missing packages again

→ One missing package can cause huge delays

Missed packages

Unacceptable for many applications

Mostly not important for games

- Positions from 30ms ago are outdated anyway
- Gets new positions all the time anyway

UDP

User Datagram Protocol

Basically IP plus port numbers

Works with packages directly

UDP

Use packages directly for game state

Implement TCP like functionality for other stuff

- Highscore lists,…

UDP

Has additional difficulties

- Applications have to measure transfer rates
- Typical packet sizes (< 512 Bytes) are hopefully enough for one piece of game state

Cheating

Never trust the client.

Never put anything on the client.

The client is in the hands of the enemy.

Never ever ever forget this.

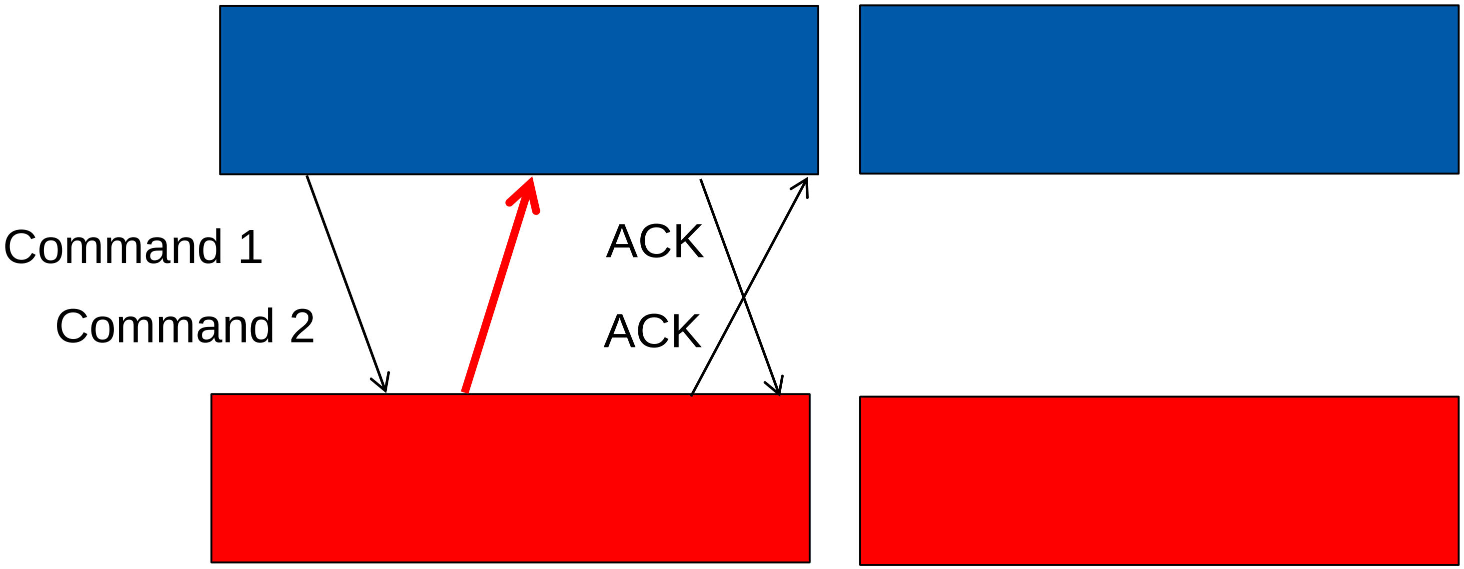
- Raph Koster, "*The Laws of Online World Design*"

Cheating in Lockstep Multiplayer

Cheating client holds back sending commands until it knows the other's commands

- RTS game: Dispatch units to counter enemy movements
- FPS game: Dodge bullets

Client 2 sends a command after it knows what Client 1 does

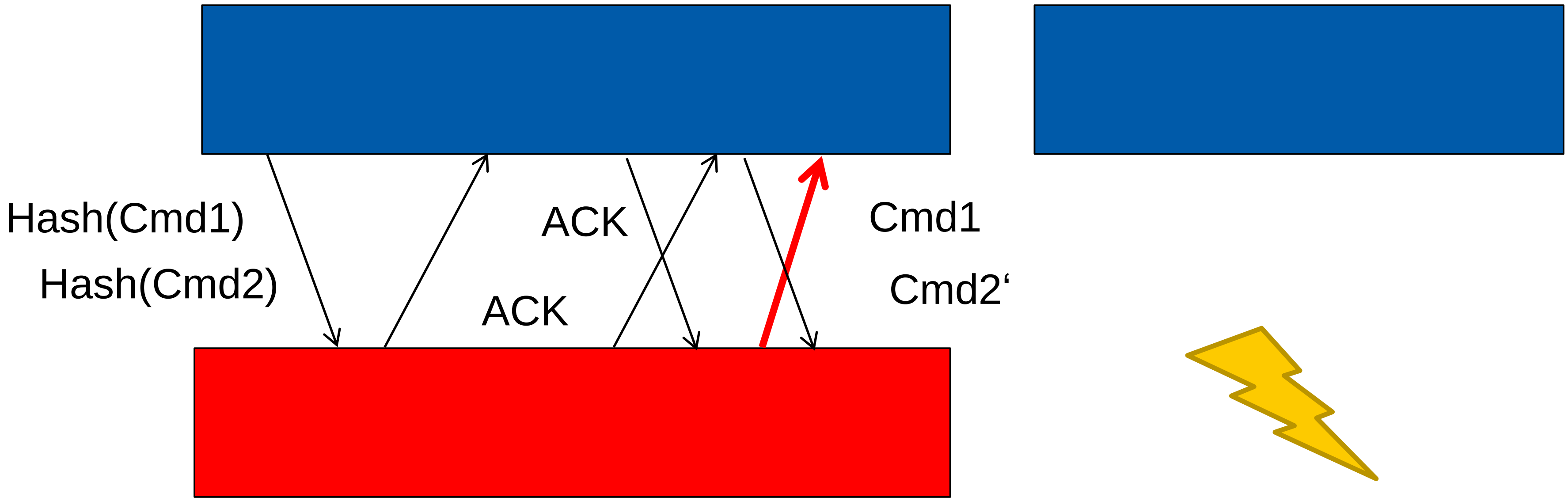


Cheating in Lockstep Multiplayer

Countermeasures

- Send a commitment – hashed value of the command
- When received all commitments: Send commands
- Each peer checks the received commitments and commands
- Cheating players are kicked

Client 2 send a different command than the committed one → Kicked



Client-Server Cheating

Assume client is hacked – Always

Everything is potentially garbage

Don’t use strings without sanitizing them first

- Or you might find users that call themselves “' OR EXISTS(SELECT * FROM users WHERE name='jake' AND password LIKE '%w%') AND ''='”

Client side

- Use knowledge of game data
- Predict wrongly

Server side

- Make incorrect inputs

Client-Side Cheats

Use game data that should not be available or usable for the player

By packet sniffing, changing the game client, memory analysis

- Wall hacks: Change textures to allow players to be seen through walls
- Auto aim: Use exact positioning data to aim automatically
- Access hidden information: Other player's hands in card games, inventories, units hidden by fog of war, …
- → Only send data on a need-to-know basis
- → Can interfere with smooth gameplay (e.g. client has to preload meshes for objects which will come into view soon, other players behind walls, …)

Incorrect predictions

- Report data like position, … incorrectly
- → Server must check reported data for validity

Server-Side cheats

TECHNISCHE UNIVERSITÄT DARMSTADT

Send wrong requests to server

- E.g. MMORPG – Players can choose new skills to learn by clicking them
- Options are grayed out if unavailable
- Hacked client sends all RPCs anyway
- → Server needs to validate that client requests are valid

Attacking the server itself

- E.g. hack the database, …

Cheat prevention

TECHNISCHE UNIVERSITÄT DARMSTADT

Check integrity of game files and executables

- Hashing, comparing hashes to reference

Monitor computer for cheating software

- World of Warcraft Warden

Monitor cheating forums

Analyze data

- Find invalid game states
- Get leads on possible exploits

Game replays, community actions

- Check replays by suspected players
- Vote on cheating players

The Future – More Predictions

Ultima VI, 1990

Game-Streaming

Run game on the server

- Client sends input events
- Server sends video stream

First commercial services

- OnLive
- ... Went out of business in 2015
- PlayStation Now
  - Started 2014

Game-Streaming Pro & Contra

Game works like a split-screen game on the server

- Super easy development

Video compression can look ugly

- But internet connections get faster all the time

Latency is as bad or worse than basic Client/Server

Cheating prevention

Latency

Speed of light is ~300000 km/s

Circumference of the earth ~40000 km

At least one data roundtrip necessary

- > 0.1 seconds for far away servers
  - Too slow

Latency

Streaming Game providers try to place lots of server at different places

- To minimize distance and therefore latency

Typically ends up at speeds that are ok for some persons

- And some genres

Not acceptable for VR

- Super low latency is critical for good VR

Shinra

Research project by Square-Enix

Wants to use streaming to create new types of multiplayer games

Current multiplayer games are restricted by the amount of data that can be transfered

- Doesn't matter when just streaming audio/video data

Plus want to just use more hardware per game

- For more physics or other costly effects

Current state (August 2015)

- Beta in North America for users with Google Fiber connection
- https://www.youtube.com/watch?v=j_Eep-XzxXo

Client/Server Programming

Example: Unreal Engine 4

Architecture

Remote Procedure Calls

- Validation

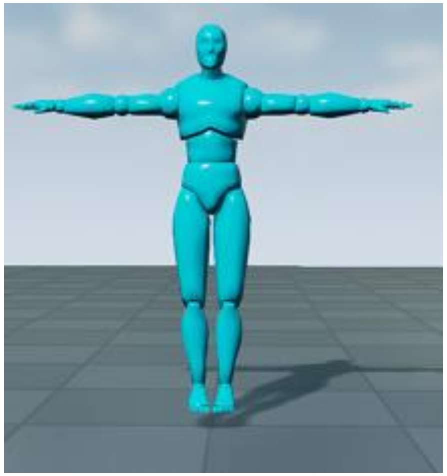
Replication

Prediction, Correction

Cheating strategies and preventions

Unreal Networking

Authoritative Client/Server



Can be dedicated server

- No rendering

Basic methods on Actors

- RPCs
- Property Replication

Actors exist on both the clients and the server

- Ownership: Local player can be the owner of an actor
- Relevant for choosing which objects run remote code

Remote Procedure Calls

Called from the server, runs on the client:

```
UFUNCTION( Client );
void ClientRPCFunction();
```

Called from the client, runs on the server:

```
UFUNCTION( Server );
void ServerRPCFunction();
```

Called from the server, runs on all clients:

```
UFUNCTION( NetMulticast );
void MulticastRPCFunction();
```

Remote Procedure Calls

TECHNISCHE UNIVERSITÄT DARMSTADT

Reliability

- Make sure that the code is eventually run
- E.g. by resending and acknowledging

```
UFUNCTION( Client, Reliable );
void ClientRPCFunction();
```

Validation

- Need to implement a function bool SomeRPCFunction_Validate(…)
- Check if game state allows this function to be called

```
UFUNCTION( Server, WithValidation );
void SomeRPCFunction( int32 AddHealth );
```

Property Replication


```
UPROPERTY( replicated )
float Health;
```

If change on server

- Replicate to client
- Overwrite current value

If change on client

- Nothing
- Clients need to use RPCs to make relevant state changes

References, Priorities, Quantization

Sending pointer values over the network

- Internally serialize to an ID
- Send the ID
- On the receiving side, look up the correct pointer value

Priorities

- Set custom net update intervals
- NetPriority: Objects with higher priority get more share of the bandwidth
- Maximal distance to replicate
- Important for owner only, for all players,…?

Quantization

- FVector_NetQuantize/FVector_NetQuantize10/FVector_NetQuantize100
- Different sizes when sent over the network

Required systems

Preprocessor Magic

"Unreal Header Tool"

Parses all UProperty

Generates meta code/reflections

Allows properties to be serialized

Common pitfalls

Forgetting to replicate properties, e.g. movement

- →Different behaviour, position on client and server
- → Running into invisible barriers, …

Getting properties of the wrong object

- Each player is represented by different pawns
- Want to check against name → different names

Harder pitfalls

Relying on ordering of replication

- The order in which properties are replicated is not guaranteed by default
- Always assume that the state of an object is not completely coherent
- If coherency is needed, ensure it
- E.g. by using RPCs to synchronize data

Summary

TECHNISCHE UNIVERSITÄT DARMSTADT

Multiplayer through the ages

- Local machine multiplayer
- 2-machine multiplayer
- LAN networking
- Internetworking
- Cloud gaming?

Architectures

- P2P Lockstep
- Client/Server (with client-side prediction)
- Cloud

Internet basics

Cheating and Cheat prevention

Merry Christmas ☺