

Networking Tutorial 1: Introduction to Networking

New Concepts

- ▶ Multi-Player Gaming
- ▶ Multi-Player Gaming over Networks

- ▶ Sockets
- ▶ Server and Client
- ▶ ENet

Multi-Player Games

- ▶ Games are multi-player
- ▶ Always have been
- ▶ Except Solitaire
- ▶ And Chess against yourself
- ▶ ForeverAlone

Multi-Player Games

- ▶ Early PC Games were fairly innovative in that regard - the most fun you could have by yourself (apart from a tell-your-own-adventure book)
- ▶ But even in arcades, Multi-Player was a Thing.
- ▶ Game to home consoles, too.
- ▶ How do we categorise multi-player experiences?

Multi-Player Games

- ▶ Championship Manager around the same PC
- ▶ Mario Kart in the lounge
- ▶ CounterStrike
- ▶ Overwatch
- ▶ The Old Republic (if you can ever find another player...)

Network Multi-Player

- ▶ Most modern games expect multi-player outside of your lounge
- ▶ Players are not geographically co-located
 - ▶ What does this mean?
 - ▶ Well, there are levels...
- ▶ In order to provision this functionality, we need to understand the tech we're going to be leveraging (this lecture), and its limitations and how we work around them (next lecture)

Sockets

- ▶ Sockets allow communication between processes -multiple processes running on the same machine, or multiple processes running on different machines
- ▶ Heterogeneous, distributed platforms - remember our discussion of the PC in terms of heterogeneous architecture
- ▶ Internet sockets allow us quick and simple access to the OS-managed network protocol stack, to communicate with other machines connected to 'the internet'

Types of Socket

- ▶ Many different types of socket are available to us, but we only care about two:
- ▶ Stream sockets
- ▶ Datagram sockets
- ▶ These socket types are differentiated by their transport protocol

Stream Sockets

- ▶ These use Transmission Control Protocol (TCP) to send messages
- ▶ TCP provides ORDERED and RELIABLE connection between two hosts
- ▶ When a message is sent, TCP guarantees it will arrive in the right order, and ensures that if the message isn't received, we know about it
- ▶ All of this is handled by the OS, so you don't need to manage it yourself

Datagram Sockets

- ▶ These use User Datagram protocol
- ▶ UDP is a much simpler protocol
- ▶ NO delivery guarantees, NO ordering guarantees
- ▶ Very fast updates, but if data goes missing we won't know about it

Which to pick?

- ▶ Dependent on the purpose of the communication
- ▶ Many networking applications will employ both guaranteed/ordered and fast based on the needs of the element (you need to know if a connection has been established for a log-in server, you don't need to know if an individual avatar position update was received)
- ▶ Generally, though, game traffic is UDP
- ▶ Except WoW.

Programming with Sockets

- ▶ Address information
- ▶ Binding the Socket
- ▶ Connecting to the Socket
- ▶ Listening for Connections
- ▶ Accepting Connections

Programming with Sockets

- ▶ Sending
- ▶ Receiving
- ▶ Differences between TCP and UDP for Send/Receive
- ▶ Closing Sockets

Clients and Servers

- ▶ Different models of network architecture
- ▶ Games employ many forms
 - ▶ Server-Client
 - ▶ Peer to Peer
 - ▶ Peer-Servers
- ▶ Cloud Distribution
- ▶ Geographical subdivision

ENet Library

- ▶ Creating Servers and Clients
- ▶ Connecting to a Server
- ▶ Managing Events
 - ▶ Complex responses
- ▶ Packets
 - ▶ Types and what we can do with them

Implementation

- ▶ Look at Framework sample code
- ▶ Think about how you would go about having a client-controlled object on your host physics engine

Summary

- ▶ Overview of multi-player gaming
- ▶ Introduced socket programming
- ▶ Introduced ENet