Presentation Notes:

The Game:

The game is a first-person shooter where clients can join a match and shoot slow-moving projectiles at each other. Upon getting hit, the hit client is teleported to the centre of the match (has great use for testing later).

The Network Architecture:

The network architecture for this game is client-server. All clients connect to one server. The server has the most up to date information about client positions and projectile positions as well as a stored information of client positions up to one second.

This was done to provide the fairest experience to all clients playing the FPS game, such as fair hit detection and enemy visibility/updates.

Layer Protocols:

Transport layer is UDP. Why:

* UDP is more lightweight, it requires less packets to be sent which is more efficient for games networking.
* The negatives of IP, such as packet loss or out of order packets, are able to be delt with uniquely instead of automatically by TCP.

Network layer is IPv4 (internet protocol).

Network API:

The network API is SFML networking.

How Networking Code is Structured and Implemented:

The server is set up operate at a specified tick rate. The server is required to keep track of time so it also determines how long it took to perform the calculations of the tick (roughly) and dynamically adjusts the sleep duration to keep the tick rate more consistent.

The client’s program has a network manager class which is updated per frame by the main thread of the game. Because of this, no consideration is needed towards keeping specific information threadsafe however the sockets are set up to be non-blocking to run smoothly.

Prediction and Interpolation:

The prediction and interpolation mechanics where adapted from Valve’s networking article (cite).

* On a client’s machine, all other clients are rendered back in time and are interpolated linearly between the two closest packets (in time). This time is definable in terms of ticks or seconds.
* This ensures that whatever movements are made by the enemy are seen by our player, albeit more delayed than necessary.
* This was done due to the fast past movement of the players, prediction logic would introduce significant flaws that would make the game very unenjoyable.
* However, if packet loss occurs than prediction is used for a defined length of time by extrapolating data from the previous most up-to-date packets. After the defined length of time, the client stands still for all other clients to stop prediction errors becoming too serious.

Other Decisions Relating to Networking:

Hit Detection:

* The players can fire projectiles at each other.
* Server side hit detection is used as it is an established standard to help stop cheaters.
* Because players see all other players back in time for interpolation reasons, the projectile will only hit clients back in time as well. This means that clients can perfectly aim and shoot the clients that they are looking at however, players can be hit in their past. This can be minimised by increasing the tick rate.
* The mathematics for hit detection and different depending on whether the hit player is stationary or not, this is easily determined.

Ping:

* Clients need to know the server time for use in hit detection and interpolation/prediction. Due to packet travel times the ping can be used to estimate the true server time.
* On the server, there is a thread that is blocked on the receive function on a unique socket watching a unique port. As soon as this port receives a message, the thread checks that it is a ping message and immediately sends the message back to the sender before looping and blocking on the receive function again.
* The client only pings the server once a connection has been established. It creates a new thread and sends a ping message and blocks to receive the returned message (with a timeout for packet loss). 10 pings are sent sequentially, averaged, and halved to get an approximation of the time taken for a message to be sent from the server to the client.
* On local host using ethernet, the ping was found to be around 100 microseconds.

Disconnecting:

* If the server doesn’t receive a packet from a client for the timeout duration (10 seconds) than the server disconnects this player and tells all other clients that this player has disconnected.
* Clients can manually tell the server they are disconnecting.
* Clients will also manually delete other clients if they don’t receive a packet within the timeout. If they receive a packet from the client again they are re-created.

Critical Evaluation of Effectiveness:

Results of Testing:

Tests were conducted using Clumsy. The following aspects of the game can be affected:

* Interpolation and prediction
* Hit detection
* Projectile positioning

Lag:

* Introducing lag can cause the prediction/interpolation logic to go into prediction when it should just interpolate (because even with the delay, new packets aren’t coming in fast enough). With lag being inconsistent it is impossible to entirely remediate this by accounting for the ping delay however this does help significantly. This can be solved by increasing the interpolation time delay but the amount depends on the ping.
* The lag only affects where the projectile starts for other clients since it’s trajectory is linear and defined by the first packet. i.e., a large ping causes the projectile to appear later in its journey.

Packet Loss:

* If the packet for firing a projectile is lost then the server never fires it and there is no check, but the client shows the projectile regardless
* Enough packet loss can cause the interpolation delay to expire and induce prediction logic which can introduce prediction errors.

Duplication:

* Duplication has no negative affect on prediction or interpolation due to checks on the packets’ times. This can be seen in code and seen by checking the server output messages.
* Duplication has no negative affect on projectiles since the server checks the packets’ times for duplication. This can be seen in code

Out of Order:

* Out of order packets does not affect projectiles negatively.
* It doesn’t affect prediction/interpolation logic significantly either. It can cause slight hitches although the delay between position packets is so large, out of order packets are extremely unlikely and the algorithm finds more suitable packets most often.
* This can sometimes cause problems with clients attempting to join the server however.

Tampering:

* Having random bits changed caused flickering of the player positions and few other negative effects.

References:

<https://developer.valvesoftware.com/wiki/Source_Multiplayer_Networking>

<http://geomalgorithms.com/a07-_distance.html> shortest distance between two skew line segments