PsychicsNet

Application layer UDP networking protocol based on fixed timesteps.

Communication is based on a single server with many clients.

Packets are sent at a fixed rate, based on a server-determined timestep.

Holepunching:

* Both server and client can holepunch by sending empty packets
* Holepunching is unnecessary once a connection is established due to fixed timestep packets
* Therefore, holepunching will occur until a timeout, or a connection is made

Connection protocol:

* Client initiates connection by sending client information to the server
  + If acceptance is not received then this is repeated until a timeout occurs
* Server accepts the connection and responds with server information, including the timestep length and the time of the first timestep
  + Client will now have estimate of RTT
  + Server will wait for an acknowledgement from the client, containing which timestep the client will first participate in
  + If ack not received after X amount of time, the acceptance is resent
  + This is repeated until Y timeouts occurs
  + Server will now have estimate of RTT

Once the acknowledgement is received by the server, the client is considered connected.

* If the acknowledgement is lost and the client sends data at a timestep, it will not yet be considered connected and will need to send another acknowledgement when the server next requests it

Upon connection:

* The server will expect the client to send input packets each time it completes a timestep Tx
* Server will receive the packet at time Tx + D. As long as delay D is not larger than some maximum half ping (e.g. 300 ms), the server will trust the input and send it out.
* When server receives input, it will buffer for a short time (e.g. half of a timestep) to see if it receives other input
* Then the server will broadcast the input to all players, including the player that sent it

Upon receipt of input from the server, the client will accept it as authoritative.

If a received packet completes a new timestep’s input, then the client/server can forget a previous authoritative timestep and use the new timestep for all further predictions.

Synchronisation:

In a perfect world, the server and client would have timesteps that occur at the exact same time.

If the clocks of the server and client are synchronised, then as long as the server and client know when to start the first timestep, and the time between timesteps, then the server and client will have synchronised timesteps.

However, it is possible that the server and client are not synchronous. In this case, the client must modify its timesteps to make it as close as possible to the server.

Let’s say that Tc = Ts + d (some delta)

Every timestep, the client will send its input to the server at what it believes to be Tx (Server time Tx – d)

* The server will receive the packet at server time Tx – d + D (delay)
* Server recognises the total difference in time as D – d
* When server responds to client, the server will include this time difference

When server receives a client message, it will respond with an acknowledgement containing the time it was sent. This allows the client to calculate the RTT, and estimate what time the server

Packet format:

General packet format:

0

+--------+

|CSAN0000|

+--------+

C: Client->Server Connection Request

S: Server->Client Connection Acceptance

X: Connection Data

N: NAK

Client->Server Connection Request

0 8 16 24

+--------+--------+--------+--------+

|10000000| VERS | VERSION |

+--------+--------+--------+--------+

|LENNAME | NAME... |

+--------+--------+--------+--------+

| NAME... |

+--------+--------+--------+--------+

VERS: Version number of the PsychicsNet communication protocol

VERSION: Version number of the client

LENNAME: Length of the name (in bytes) – max of 32

NAME: ASCII-Encoded name

Server->Client Connection Acceptance

0 8 16 24

+--------+--------+--------+--------+

|01000000| VERS | VERSION |

+--------+--------+--------+--------+

| |

+ + TIMESTEP0 + +

| |

+--------+--------+--------+--------+

| LENTIMESTEP |00000000000000000|

+--------+--------+--------+--------+

| TIMESTAMP |

+--------+--------+--------+--------+

VERS: Version number of the PsychicsNet communication protocol

TIMESTEP0: Milliseconds since epoch at which the first timestep happened

LENTIMESTEP: The time in milliseconds of each timestep

TIMESTAMP: Milliseconds since TIMESTEP0 when this packet was sent

General Connected Packet

0 8 16 24

+--------+--------+--------+--------+

|00010000| TIMESTEP |

+--------+--------+--------+--------+

| TIMESTAMP | DELAY |

+--------+--------+--------+--------+

| SEG | TOTALSEGS +

+--------+--------+-----------------+

TIMESTEP: The timestep this packet is relevant to

TIMESTAMP: Time after timestep when the packet was sent in milliseconds

DELAY (signed): The average delay the sender experiences for the receiver packets (delay = receive time – timestamp)

SEG: The segment number of this packet

TOTALSEGS: The total number of segments for this timestep

No Acknowledgement (NAK)

0 8 16 24

+--------+--------+--------+--------+

|00001000| TIMESTEP |

+--------+--------+--------+--------+

| SEGS... |

+--------+--------+

TIMESTEP: The timestep this packet is not acknowledging

SEGS: An array of segments that weren’t received:

* If no segments are listed, then the timestep has not yet been received at all
* If segments are listed, then these are the segments that are missing

Note: NACKS will be sent when:

* Packets are received out of order, and the packet was not found after a short delay
* There has been a period without packets where packets were expected