Peerchat Protocol

This document describes the peerchat protocol specification. Peerchat is a decentralized, peer to peer chat system. The sections below will describe both the protocol that peers communicate with, and implementation details on how peers use the protocol.

Table of Contents

1 Protocol Specification

1.1 Packet Format

For the sake of implementation, all messages, also referred to as packets, sent between peers are of a fixed size (260 bytes). The first 4 bytes of each packet are used to describe the type of payload the packet is delivering, and how to parse the payload.

Below is the layout of an individual packet.

Packet Format			
Size	**Type**	**Name**	**Description**
4 Bytes	Unsigned Integral	Payload Type	Indicator for the type of the payload.
256 Bytes	Fixed Length	Payload	The data for the payload.

1.2 Payload Format

1.2.1 Payload Types

There are two types of payloads in the protocol: message payloads and identity payloads. They are denoted with the payload types 0x0001 and 0x0002 respectively. Payloads for unknown types are to be dropped and ignored.

Payload Types		
Value	**Name**	**Description**
0x0001	Message	Denotes the payload contains message data.
0x0002	Identity	Denotes the payload contains identity data.

1.2.2 Message Payload

If the payload is marked with the message type, then the payload contains message data. The message payload has one field, the message, which contains the message from the sender of the packet. The message is sent in a 256 byte array, and must be null terminated. If the message is not null terminated, the 256th byte is set to a null terminator when the packet is being read. Messages cannot be longer than 256 characters.

Below is the layout of the message payload.

Message Payload Format			
Size	**Type**	**Name**	**Description**
256 Bytes	Unsigned Integral	Message	The null terminated message. Limited to 256 characters.

1.2.3 Identity Payload

If the payload is marked with the identity type, then the payload contains identity data. The first field in the identity payload is the username field. This is a 32 byte array for storing the username of the sender of the packet. This field must be null terminated. If the username is not null terminated, the 32nd byte is set to a null terminator when the packet is being read. Usernames cannot be longer than 32 characters. The second field is the port that the sender accepts connections on, and is sent as a 16 bit unsigned integer. The third field is the zip code of the sende, and is sent as a 32 bit unsigned integer. The forth field is the age of the sender, and is sent as a 8 bit unsigned integer. The fifth field is an array of 32 peers. The sender can populate up to 32 of their peers into this array. The sixth field denotes the number of peers in the peers array, and is sent as a 32 bit unsigned integer. If this value is higher than 32, it is set to 32 when being read.

Below is the layout of the identity payload.

Identity Payload Format			
Size	**Type**	**Name**	**Description**
32 Bytes	Character	Username	The sender’s null terminated username.
2 Bytes	Unsigned Integer	Port	The port the sender accepts connections on.
4 Bytes	Unsigned Integer	Zip Code	The zip code of the sender.
1 Byte	Unsigned Integer	Age	The age of the sender.
192 Bytes	Peer	Peers	List with room for 32 of the sender’s peers.
4 Bytes	Unsigned Integer	Peer count	Total number of peers in the peers field. Cannot exceed 32.

1.2.4 Peer Format

Peers being sent in the identity payload describe the peers that the sender is connected to. This does not include the sender themself, or the peer that the identity payload is being sent to. There are two fields in this payload: the ipv4 address that the peer can be connected to from, and the port that the peer is listening for connections on.

Below is the layout of a peer.

Peer Format			
Size	**Type**	**Name**	**Description**
4 Bytes	Unsigned Integer	IPv4 Address	The address of the peer.
2 Bytes	Unsigned Integer	Port	The port the peer accepts connections on.

2 Implementation

Each peer in the network maintains a connection with every other peer in the network.

2.1 Terminology

The terms client and host peer are used in the implementation section to distinguish the communication between two peers. The host peer is the peer who is accepting the connection, while the client peer is the peer who established the connection.

2.2 Peer States

Peers can be in two distinct states in regards to their connection with other peers: pending and active. These states describe the state of the connection and decide how the protocol is handled for packets being sent and received from that peer.

2.2.1 Pending

The pending state for a peer is reserved for newly connected peers. In this state, the client peer has connect to the host peer, but had not yet sent their identity via an identity payload to the host peer. Any payload received by a host peer by the client peer while in the pending state, except the identity payload, is dropped by the host peer. Client peers in the pending state are also not included in the host peers list when the host peer sends its identity payload to another peer.

2.2.2 Active

The active state for a peer is reserved for client peers that have sent their identity payload. Once a host peer has received the identity payload from the client peer, the client peer is now marked as active for the host peer. Peers in the active state are no longer able to send identity payloads to the host peer. If an identity payload is sent, the host peer will drop it. Packets, except the identity packet, are now able to be sent from the client peer to the host peer without being dropped.

2.3 Connections

This section will describe the flow of information when a connection is established between a client and host peer. When two peers connect, they will initially mark each other with the pending peer state and immediately send each other their identity payload. When the identity payload is received by a peer, it marks the other peer as active (and vice versa). In addition to this, it attempts to establish a connect with all peers included in the identity payload's peer list and this process is repeated. The implementation is responsible for not connecting to peers that it has already connected to.

2.4 Messages

This section will describe how messages are sent between peers. When a peer creates a message payload, they will send the packet to all active peers that they are tracking with the message. If the receiving peer has the sending peer in the active state, they will process and display the message.