Protocol

INFO

u8 provides 255 possible values
u16 provides 65535 possible values
u32 provides 4,294,967,295 possible values

Wire format

Header

```
u8  | version (TBD)
u8  | flags (unused for now)
u8  | transaction_id
u8  | opcode
u16 | length
```

Payload

Packet Types ("opcode"):

- ACK 0x04
- NACK 0x05
- MSG
- SYNC_MSGS
- FETCH_MSGS
- PRESENCE
- HEARTBEAT 0x08
- STARTUP
- AUTH
- UPDATE_MSG
- PULL_CONFIG_DB
- SERVER_METADATA

Message (plain-text) (`MSG`)

> plain text with support for basic markdown and mentions (@user)

```
u32 | msg id (NULL when client->server)
u16 | channel id
u32 | author id
u16 | content string length
str | msg content
```

`content` gets parsed for markdown and mentions before rendering

Message (rich)

> A rich text message can contain images or videos as well as text
> TODO

Acknowledge Msg (`ACK_MSG`)

```
u32 | msg id
u64 | timestamp (unix epoch)
```

Sync Msgs (`SYNC_MSG`)

```
u16 | channel id
u32 | latest msg id
```

Fetch Msg

```
u32 | msg_id
```

Responds with: MSG | NACK (implies msg doesn't exist)

Fetch Msgs (`FETCH_MSGS`)

```
u16 | channel id
u32 | reference msg id
i32 | number of msgs either side of reference msg id
      to return (+/- of int refers to which side of reference id
      (after/before respectively))
```

Presence (`PRESENCE`)

```
u32  | user id
bool | online/offline
u64  | last seen timestamp
```

Startup (`STARTUP`)

> Kicks off a connection between client & server. Sent immediately after a client connects to a server via TCP. The server responds with various metadata about the server that the client can then use to check for compatibility.

```
u32 | user id
u8  | client version
```

Responses: SERVER_METADATA | NACK

Authenticate (`AUTH`)

```
str (u16 + u8[]) | username
str (u16 + u8[]) | password
```

Heartbeat (`HEARTBEAT`)

```
no-op
```

Update Msg (`UPDATE_MSG`)

```
u32  | msg id
bool | deleted?
u64  | updated_at timestamp
u16  | string len
str  | new msg content
```

Acknowledge (`ACK`)

> General acknowledgement of success

Negative acknowledge (`NACK`)

> General acknowledgement of packet but operation failed / was unable to be completed by recipient

AUTH_ACK (`AUTH_ACK`)

> Tells the client what their user ID is based on username (so they dont need to keep sending it)

```
u32  | user id
128b | session key
```

TBD

Pull the server config db (`CONFIG_DB_REQ`)

> Returns all info about the server such as channel id -> channel name (string) mappings, user names to user ids, etc
> TBD