Intro to WebSockets

The WebSocket Protocol – RFC 6455

The WebSocket Protocol enables two-way communication between a client running untrusted code in a controlled environment to a remote host that has opted-in to communications from that code.
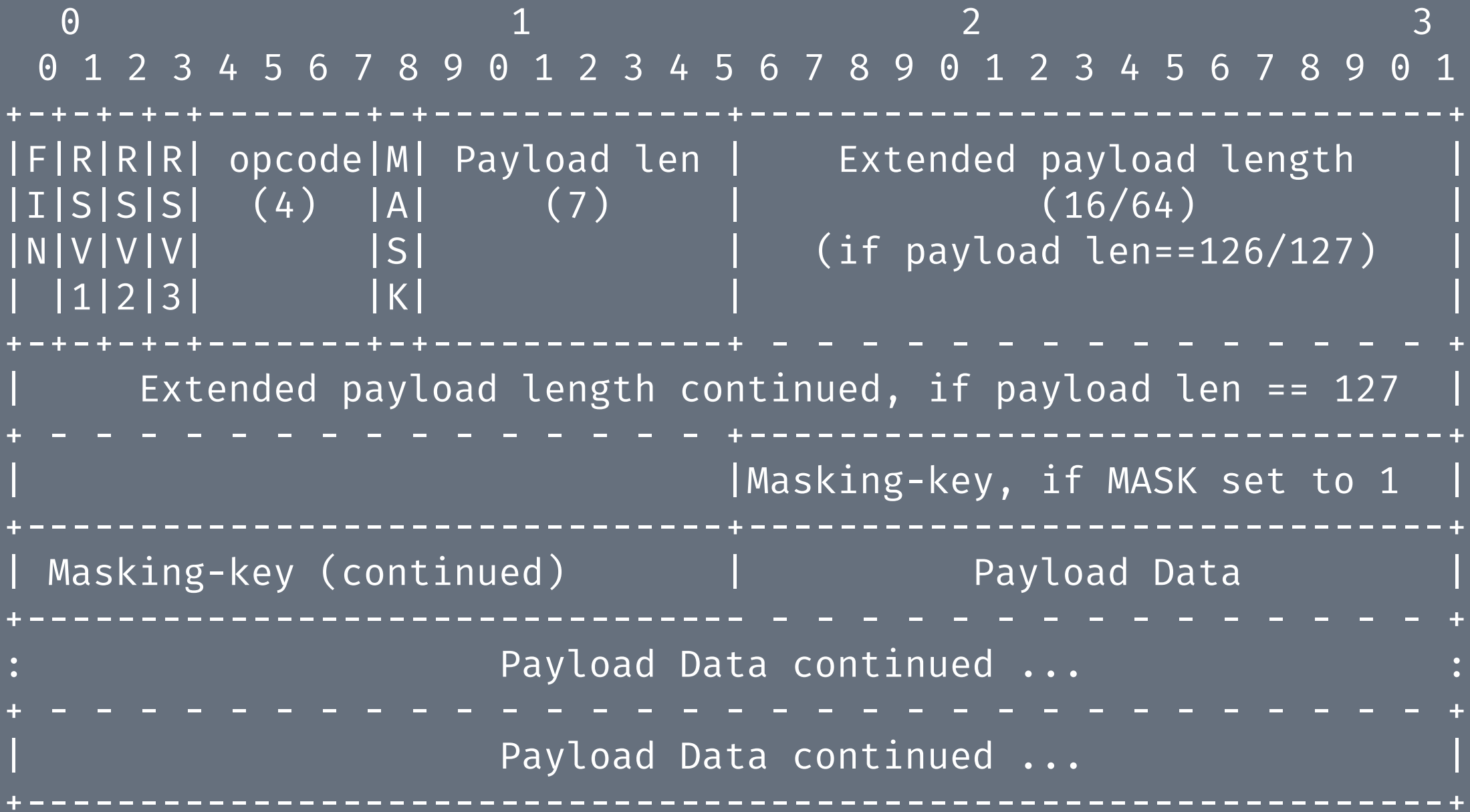
RFC 6455

Basics

→ Binary Protocol

→ Upgrades HTTP 1.1

→ Uses frames

Frames

→ Used for controlling the socket and sending data

→ Contains a header that explains the data message

Frames (cont.)

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-------+-+-------------+-------------------------------+
|F|R|R|R| opcode|M| Payload len |    Extended payload length    |
|I|S|S|S|  (4)  |A|     (7)     |             (16/64)           |
|N|V|V|V|       |S|             |   (if payload len==126/127)   |
| |1|2|3|       |K|             |                               |
+-+-+-+-+-------+-+-------------+ - - - - - - - - - - - - - - - +
|     Extended payload length continued, if payload len == 127  |
+ - - - - - - - - - - - - - - - +-------------------------------+
|                               |Masking-key, if MASK set to 1  |
+-------------------------------+-------------------------------+
| Masking-key (continued)       |          Payload Data         |
+-------------------------------- - - - - - - - - - - - - - - - +
:                     Payload Data continued ...                :
+ - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - +
|                     Payload Data continued ...                |
+---------------------------------------------------------------+
```

Basic Flow

→ Normal HTTP Connection

→ HTTP 1.1 Upgrade to Websocket

→ Frames are sent back and forth

Backends

Cowboy WebSocket Handler

→ Low level websockets

→ Dealing with frames directly

Phoenix Channels

→ Higher Level

→ Abstraction around websockets

→ Implements its own protocol on top of websockets

```elixir
defmodule Web.CharacterSocket do
  use Phoenix.Socket

  channel("chat:*", Web.ChatChannel)

  def connect(%{"token" => token}, socket) do
    case Phoenix.Token.verify(socket, "character socket", token) do
      {:ok, character_id} ->
        {:ok, assign(socket, :character_id, character_id)}

      {:error, _reason} ->
        :error
    end
  end

  def id(socket), do: "characters:#{socket.assigns.character_id}"
end
```

```elixir
defmodule Web.ChatChannel do
  use Phoenix.Channel

  def join("chat:" <> channel, _message, socket) do
    case Map.has_key?(socket.assigns, :character_id) do
      true ->
        assign_character(socket, channel)

      false ->
        {:error, %{reason: "character required"}}
    end
  end
end
```

Clients

Elixir

→ WebSockex

→ Gun

WebSockex

→ Callback inspired approach

→ Only a websocket client

```elixir
defmodule Gossip.Socket do
  use WebSockex

  def start_link() do
    WebSockex.start_link(url(), __MODULE__, state, [name: Gossip.Socket])
  end

  def handle_connect(_conn, state) do
    {:ok, state}
  end

  def handle_frame({:text, message}, state) do
    {:ok, state}
  end
end
```

JavaScript

Phoenix Socket JS

```javascript
let socket = new Socket("/socket", {params: {token: token}});
socket.connect();

let channel = socket.channel("chat:global", {})
channel.on("broadcast", (data) => {
  console.log(data);
});
channel.join().
  receive("ok", (e) => {
    console.log("Connected!");
  });;
```

Inspecting WebSockets

→ Open the dev tools -> Networking -> WS

→ Click on the connection -> Frames

Questions?