第一章 ESP32 的 WebSocket 服务器

1. 学习目的及目标

- 掌握 Websocket 原理和工作过程
- 掌握乐鑫 ESP32 的 WebSocket 的程序设计

2. WebSocket 原理

WebSocket 是一种网络通信协议，是 HTML5 开始提供的一种在单个 TCP 连接上进行全双工通讯的协议。

2.1. 为什么需要 WebSocket ？

了解计算机网络协议的人，应该都知道：HTTP 协议是一种无状态的、无连接的、单向的应用层协议。它采用了请求/响应模型。通信请求只能由客户端发起，服务端对请求做出应答处理。

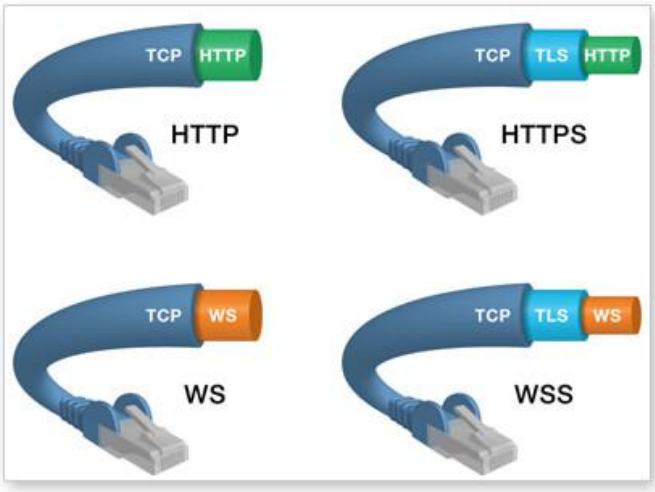
这种通信模型有一个弊端：HTTP 协议无法实现服务器主动向客户端发起消息。

这种单向请求的特点，注定了如果服务器有连续的状态变化，客户端要获知就非常麻烦。大多数 Web 应用程序将通过频繁的异步 JavaScript 和 XML（AJAX）请求实现长轮询。轮询的效率低，非常浪费资源（因为必须不停连接，或者 HTTP 连接始终打开）。

因此，工程师们一直在思考，有没有更好的方法。WebSocket 就是这样发明的。WebSocket 连接允许客户端和服务器之间进行全双工通信，以便任一方都可以通过建立的连接将数据推送到另一端。WebSocket 只需要建立一次连接，就可以一直保持连接状态。这相比于轮询方式的不停建立连接显然效率要大大提高。

2.2. Websocket 特点

- 建立在 TCP 协议之上，服务器端的实现比较容易。
- 与 HTTP 协议有着良好的兼容性。默认端口也是 80 和 443 ，并且握手阶段采用 HTTP 协议，因此握手时不容易屏蔽，能通过各种 HTTP 代理服务器。
- 数据格式比较轻量，性能开销小，通信高效。
- 可以发送文本，也可以发送二进制数据。
- 没有同源限制，客户端可以与任意服务器通信。
- 协议标识符是 ws（如果加密，则为 wss），服务器网址就是 URL，如下图。



2.3. 请求握手包

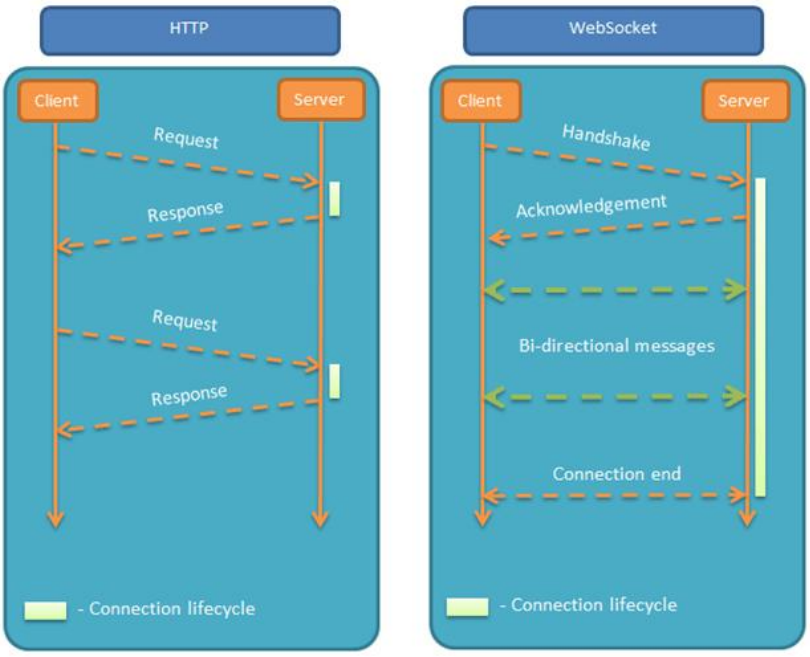
```
GET /chat HTTP/1.1
Host: server.example.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: x3JJHMbDL1EzLkh9GBhXDw==
Sec-WebSocket-Protocol: chat, superchat
Sec-WebSocket-Version: 13
Origin: http://example.com
```

2.4. 接收请求包

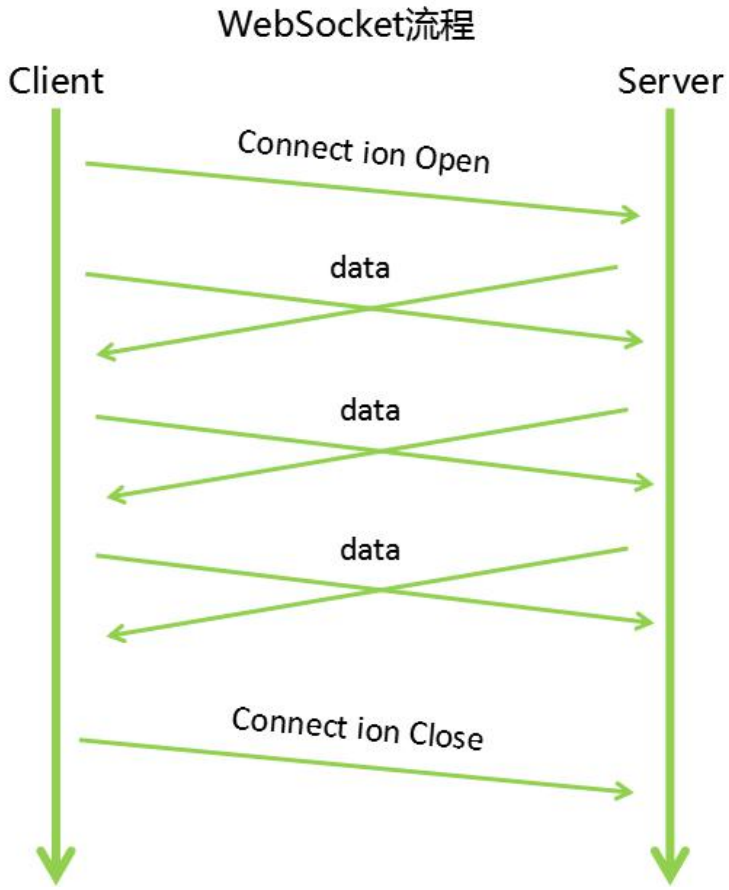
```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: HSmrc0sMlYUkAGmm5OPpG2HaGWk= Sec-WebSocket-Protocol: chat
```

注：[WebSocket 握手详解]

3. Websocket 和 HTTP 连接过程

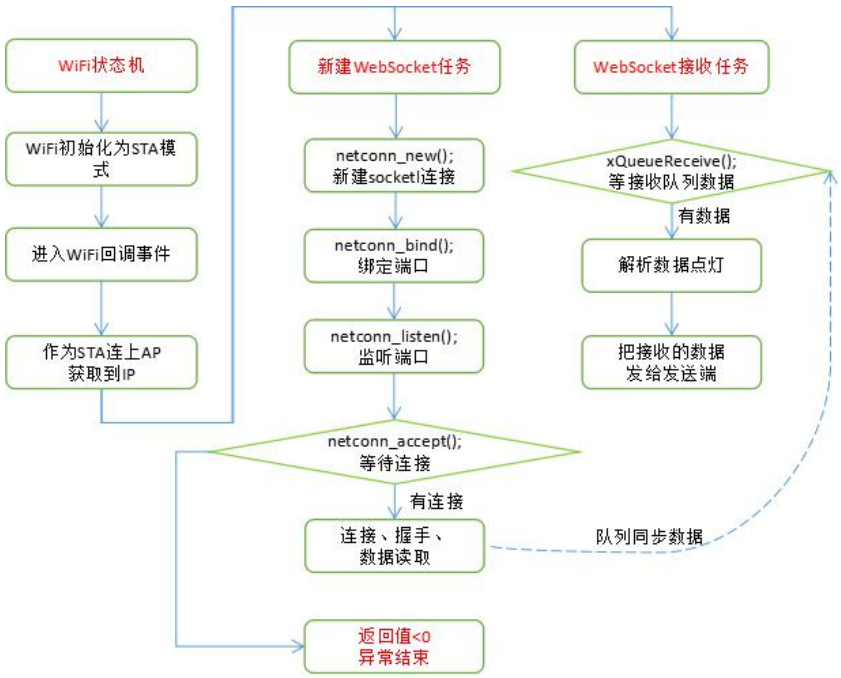


4. Websocket 工作过程



5. 软件设计

5.1. ESP32 的 Websocket 详细过程



5.2. ESP32 的 Websocket 接口介绍

- 连接函数：netconn_new();
- 绑定函数：netconn_bind();
- 监听函数：netconn_listen();
- 获取连接函数：netconn_accept();
- 接收数据函数：netconn_recv();
- 发送数据函数：netconn_write();
- 关闭连接函数：netconn_close();
- 删除连接函数：netconn_delete();

更多更详细接口请参考官方指南。

5.3. Websocket 新建任务编写

```
1  void ws_server(void *pvParameters)
2  {
3       struct netconn *conn, *newconn;
4       //获取 tcp socket connect
5       conn = netconn_new(NETCONN_TCP);
6       //绑定 port
7       netconn_bind(conn, NULL, WS_PORT);
8       //监听
9       netconn_listen(conn);
10      //等待 client 连接
11      while (netconn_accept(conn, &newconn) == ERR_OK)
12      {
13           //新连接：等待连接、连接过程、数据读取
14           ws_server_netconn_serve(newconn);
15      }
16      //关闭 websocket server connect
17      netconn_close(conn);
18      netconn_delete(conn);
19 }
```

5.4. Websocket 连接过程代码

```
1  static void ws_server_netconn_serve(struct netconn *conn) {
2       //申请 SHA1
3       p_SHA1_Inp = pvPortMallocCaps(WS_CLIENT_KEY_L + sizeof(WS_sec_conKey),
4                 MALLOC_CAP_8BIT);
5       //申请 SHA1 result
6       p_SHA1_result = pvPortMallocCaps(SHA1_RES_L, MALLOC_CAP_8BIT);
7       //Check if malloc suceeded
8       if ((p_SHA1_Inp != NULL) && (p_SHA1_result != NULL)) {
9            //接收“连接”过程的数据
10           if (netconn_recv(conn, &inbuf) == ERR_OK) {
11                //读取“连接”过程的数据到 buf
```

```
12                netbuf_data(inbuf, (void**) &buf, &i);
13                //把 server 的 key 传给 SHA1
14                for (i = 0; i < sizeof(WS_sec_conKey); i++)
15                {
16                     //放在后 24 字节
17                     p_SHA1_Inp[i + WS_CLIENT_KEY_L] = WS_sec_conKey[i];
18                }
19                //搜索 client 的 key
20                p_buf = strstr(buf, WS_sec_WS_keys);
21                //找到 key
22                if (p_buf != NULL) {
23                     //get Client Key
24                     for (i = 0; i < WS_CLIENT_KEY_L; i++)
25                     {
26                          //放在前 24 字节
27                          p_SHA1_Inp[i] = *(p_buf + sizeof(WS_sec_WS_keys) + i);
28                     }
29                     // 计算 hash
30                     esp_sha(SHA1, (unsigned char*) p_SHA1_Inp, strlen(p_SHA1_Inp),
31                               (unsigned char*) p_SHA1_result);
32                     //转 base64
33                     p_buf = (char*) _base64_encode((unsigned char*) p_SHA1_result,
34                               SHA1_RES_L, (size_t*) &i);
35                     //free SHA1 input
36                     free(p_SHA1_Inp);
37                     //free SHA1 result
38                     free(p_SHA1_result);
39                     if (p_payload != NULL) {
40                          //准备“握手”帧
41                          sprintf(p_payload, WS_srv_hs, i - 1, p_buf);
42                          //发送“握手”帧
43                          netconn_write(conn, p_payload, strlen(p_payload),NETCONN_COPY);
44                          //free base64
45                          free(p_buf);
46                          //free “握手”内存
47                          free(p_payload);
48                          //websocket 连接成功
49                          WS_conn = conn;
50                          //“接收数据”
51                          while (netconn_recv(conn, &inbuf) == ERR_OK) {
52                     。。。。。。
```

5.5. Websocket 发送代码

```
1  err_t WS_write_data(char* p_data, size_t length) {
2       //websocket 未连接，直接退出
3       if (WS_conn == NULL)
```

```
4           return ERR_CONN;
5       //数据帧长度溢出，直接退出
6       if (length > WS_STD_LEN)
7           return ERR_VAL;
8       err_t result;
9       //报头
10      WS_frame_header_t hdr;
11      hdr.FIN = 0x1;
12      hdr.payload_length = length;
13      hdr.mask = 0;
14      hdr.reserved = 0;
15      hdr.opcode = WS_OP_TXT;
16      //发送报头
17      result = netconn_write(WS_conn, &hdr, sizeof(WS_frame_header_t), NETCONN_COPY);
18      if (result != ERR_OK)
19          return result;
20      //发送数据
21      return netconn_write(WS_conn, p_data, length, NETCONN_COPY);
22  }
```

代码有全部中文注释

6. 测试流程和效果展示

6.1. 测试流程

- 修改 STA 的账号密码
- 使用电脑助手工具进行 WebSocket 测试

6.2. 效果展示

➢ 连接

➢ 发送


WebSocket 在线测试 v13 by 煌
转发场景
WebSocket客户端
蓝林纪录片
点击进
ws://192.168.2.105:9998
连接
断开
1、连接格式为 ws://IP或域名:端口（示例ws://121.40.165.18:8800）
2、对于内网的测试环境，只需填入服务端的内网IP和端口
3、可连接我上面提供的服务端ws地址来测试您自己的客户端
表情
发送的内容
发什么回什么
发送（ctrl+回车）
1、平台兼容IE6及以上的任何浏览器接入
2、服务端未使用任何框架，原生方式实现，更清楚WebSocket底层流程
3、服务端支持多客户端数据收发
4、需要完整的【客户端/服务器】源码请咨询作者
联系作者
消息窗口
等待服务器握手包...
服务器 16:45:40
和服务器断开连接！
你 16:46:13
等待服务器握手包...
服务器 16:46:14
和服务器断开连接！
你 16:46:23
等待服务器握手包...
你 16:46:23
收到服务器握手包.
服务器 16:46:23
连接已建立，正在等待数据...
你 16:47:17
123456789
服务器 16:47:17
123456789

➢ 控灯


WebSocket 在线测试 v13 by 煌
转发场景
WebSocket客户端
蓝林纪录片
ws://192.168.2.105:9998
连接
断开
1、连接格式为 ws://IP或域名:端口（示例ws://121.40.165.18:8800）
2、对于内网的测试环境，只需填入服务端的内网IP和端口
3、可连接我上面提供的服务端ws地址来测试您自己的客户端
表情
发送的内容
发送（ctrl+回车）
1、平台兼容IE6及以上的任何浏览器接入
2、服务端未使用任何框架，原生方式实现，更清楚WebSocket底层流程
3、服务端支持多客户端数据收发
4、需要完整的【客户端/服务器】源码请咨询作者
联系作者
消息窗口
和服务器断开连接！
你 16:46:13
等待服务器握手包...
服务器 16:46:14
和服务器断开连接！
你 16:46:23
等待服务器握手包...
你 16:46:23
收到服务器握手包.
服务器 16:46:23
连接已建立，正在等待数据...
你 16:47:17
123456789
服务器 16:47:17
123456789
你 16:48:19
ON
开灯

➢ 断开连接


WebSocket 在线测试 v13 by 煌
转发场景
WebSocket客户端
蓝林纪录片
点击进入聊天室
ws://192.168.2.105:9998
连接
断开
1、连接格式为 ws://IP或域名:端口（示例ws://121.40.165.18:8800）
2、对于内网的测试环境，只需填入服务端的内网IP和端口
3、可连接我上面提供的服务端ws地址来测试您自己的客户端
表情
发送的内容
发送（ctrl+回车）
1、平台兼容IE6及以上的任何浏览器接入
2、服务端未使用任何框架，原生方式实现，更清楚WebSocket底层流程
3、服务端支持多客户端数据收发
4、需要完整的【客户端/服务器】源码请咨询作者
联系作者
消息窗口
等待服务器握手包...
服务器 16:46:14
和服务器断开连接！
你 16:46:23
等待服务器握手包...
你 16:46:23
收到服务器握手包.
服务器 16:46:23
连接已建立，正在等待数据...
你 16:47:17
123456789
服务器 16:47:17
123456789
你 16:48:19
ON
服务器 16:48:50
和服务器断开连接！

7. WebSocket 总结

- 此源码主要是学习 WebSocket 整个流程，实际中还有很多可以完善的地方，了解 WebSocket 后，就自己飞向天空吧。
- 源码地址：https://github.com/xiaolongba/wireless-tech