

38 | WebSocket：沙盒里的TCP

2019-08-23 Chrono

《透视HTTP协议》

课程介绍 >



讲述：Chrono

时长 11:36 大小 13.28M



在之前讲 TCP/IP 协议栈的时候，我说过有“TCP Socket”，它实际上是一种功能接口，通过这些接口就可以使用 TCP/IP 协议栈在传输层收发数据。

那么，你知道还有一种东西叫“WebSocket”吗？

单从名字上看，“Web”指的是 HTTP，“Socket”是套接字调用，那么这两个连起来又是什么意思呢？

所谓“望文生义”，大概你也能猜出来，“WebSocket”就是运行在“Web”，也就是 HTTP 上的 Socket 通信规范，提供与“TCP Socket”类似的功能，使用它就可以像“TCP Socket”一样调用下层协议栈，任意地收发数据。

领资料



HTML



WebSockets

更准确地说，“WebSocket”是一种基于 TCP 的轻量级网络通信协议，在地位上是与 HTTP“平级”的。

为什么要有 WebSocket

不过，已经有了被广泛应用的 HTTP 协议，为什么要再出一个 WebSocket 呢？它有哪些好处呢？

其实 WebSocket 与 HTTP/2 一样，都是为了解决 HTTP 某方面的缺陷而诞生的。HTTP/2 针对的是“队头阻塞”，而 WebSocket 针对的是“请求 – 应答”通信模式。

那么，“请求 – 应答”有什么不好的地方呢？

“请求 – 应答”是一种“**半双工**”的通信模式，虽然可以双向收发数据，但同一时刻只能一个方向上有动作，传输效率低。更关键的一点，它是一种“被动”通信模式，服务器只能“被动”响应客户端的请求，无法主动向客户端发送数据。

虽然后来的 HTTP/2、HTTP/3 新增了 Stream、Server Push 等特性，但“请求 – 应答”依然是主要的工作方式。这就导致 HTTP 难以应用在动态页面、即时消息、网络游戏等要求“**实时通信**”的领域。

在 WebSocket 出现之前，在浏览器环境里用 JavaScript 开发实时 Web 应用很麻烦。因为浏览器是一个“受限的沙盒”，不能用 TCP，只有 HTTP 协议可用，所以就出现了很多“变通”的技术，“**轮询**”（polling）就是比较常用的一种。

领资料



简单地说，轮询就是不停地向服务器发送 HTTP 请求，问有没有数据，有数据的话服务器就用响应报文回应。如果轮询的频率比较高，那么就可以近似地实现“实时通信”的效果。

但轮询的缺点也很明显，反复发送无效查询请求耗费了大量的带宽和 CPU 资源，非常不经济。

所以，为了克服 HTTP“请求 – 应答”模式的缺点，WebSocket 就“应运而生”了。它原来是 HTML5 的一部分，后来“自立门户”，形成了一个单独的标准，RFC 文档编号是 6455。

WebSocket 的特点

WebSocket 是一个真正“**全双工**”的通信协议，与 TCP 一样，客户端和服务器都可以随时向对方发送数据，而不用像 HTTP“你拍一，我拍一”那么“客套”。于是，服务器就可以变得更加“主动”了。一旦后台有新的数据，就可以立即“推送”给客户端，不需要客户端轮询，“实时通信”的效率也就提高了。

WebSocket 采用了二进制帧结构，语法、语义与 HTTP 完全不兼容，但因为它的主要运行环境是浏览器，为了便于推广和应用，就不得不“搭便车”，在使用习惯上尽量向 HTTP 靠拢，这就是它名字里“Web”的含义。

服务发现方面，WebSocket 没有使用 TCP 的“IP 地址 + 端口号”，而是延用了 HTTP 的 URI 格式，但开头的协议名不是“http”，引入的是两个新的名字：“**ws**”和“**wss**”，分别表示明文和加密的 WebSocket 协议。

WebSocket 的默认端口也选择了 80 和 443，因为现在互联网上的防火墙屏蔽了绝大多数的端口，只对 HTTP 的 80、443 端口“放行”，所以 WebSocket 就可以“伪装”成 HTTP 协议，比较容易地“穿透”防火墙，与服务器建立连接。具体是怎么“伪装”的，我稍后再讲。

下面我举几个 WebSocket 服务的例子，你看看，是不是和 HTTP 几乎一模一样：



复制代码

```
1 ws://www.chrono.com
2 ws://www.chrono.com:8080/srv
3 wss://www.chrono.com:445/im?user_id=xxx
```

要注意的一点是，WebSocket 的名字容易让人产生误解，虽然大多数情况下我们会在浏览器里调用 API 来使用 WebSocket，但它不是一个“调用接口的集合”，而是一个通信协议，所以我觉得把它理解成“TCP over Web”会更恰当一些。

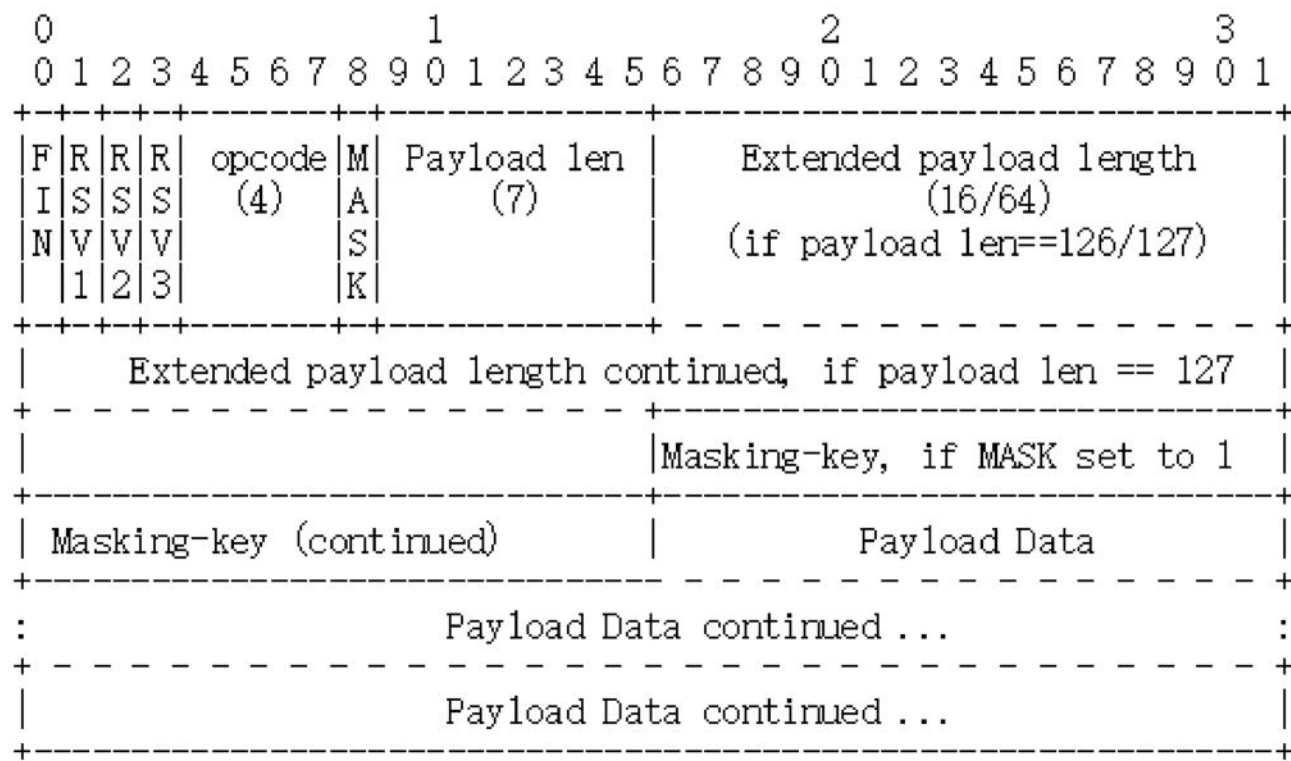
WebSocket 的帧结构

刚才说了，WebSocket 用的也是二进制帧，有之前 HTTP/2、HTTP/3 的经验，相信你这次也能很快掌握 WebSocket 的报文结构。

不过 WebSocket 和 HTTP/2 的关注点不同，WebSocket 更侧重于“实时通信”，而 HTTP/2 更侧重于提高传输效率，所以两者的帧结构也有很大的区别。

WebSocket 虽然有“帧”，但却没有像 HTTP/2 那样定义“流”，也就不存在“多路复用”“优先级”等复杂的特性，而它自身就是“全双工”的，也就不需要“服务器推送”。所以综合起来，WebSocket 的帧学习起来会简单一些。

下图就是 WebSocket 的帧结构定义，长度不固定，最少 2 个字节，最多 14 字节，看着好像很复杂，实际非常简单。



开头的两个字节是必须的，也是最关键的。

领资料



第一个字节的第一位“**FIN**”是消息结束的标志位，相当于 HTTP/2 里的“END_STREAM”，表示数据发送完毕。一个消息可以拆成多个帧，接收方看到“FIN”后，就可以把前面的帧拼起来，组成完整的消息。

“FIN”后面的三个位是保留位，目前没有任何意义，但必须是 0。

第一个字节的后 4 位很重要，叫“**Opcode**”，操作码，其实就是帧类型，比如 1 表示帧内容是纯文本，2 表示帧内容是二进制数据，8 是关闭连接，9 和 10 分别是连接保活的 PING 和 PONG。

第二个字节第一位是掩码标志位“**MASK**”，表示帧内容是否使用异或操作（xor）做简单的加密。目前的 WebSocket 标准规定，客户端发送数据必须使用掩码，而服务器发送则必须不使用掩码。

第二个字节后 7 位是“**Payload len**”，表示帧内容的长度。它是另一种变长编码，最少 7 位，最多是 7+64 位，也就是额外增加 8 个字节，所以一个 WebSocket 帧最大是 2^{64} 。

长度字段后面是“**Masking-key**”，掩码密钥，它是由上面的标志位“MASK”决定的，如果使用掩码就是 4 个字节的随机数，否则就不存在。

这么分析下来，其实 WebSocket 的帧头就四个部分：“**结束标志位 + 操作码 + 帧长度 + 掩码**”，只是使用了变长编码的“小花招”，不像 HTTP/2 定长报文头那么简单明了。

我们的实验环境利用 OpenResty 的“lua-resty-websocket”库，实现了一个简单的 WebSocket 通信，你可以访问 URI“/38-1”，它会连接后端的 WebSocket 服务“ws://127.0.0.1/38-0”，用 Wireshark 抓包就可以看到 WebSocket 的整个通信过程。

下面的截图是其中的一个文本帧，因为它是客户端发出的，所以需要掩码，报文头就在两个字节之外多了四个字节的“Masking-key”，总共是 6 个字节。

领资料



▼ WebSocket	
1... .. = Fin: True	
.000 = Reserved: 0x0	
.... 0001 = Opcode: Text (1)	
1... .. = Mask: True	
.000 1111 = Payload length: 15	
Masking-Key: 5aa62869	
Masked payload	
Payload	
0000	02 00 00 00 45 00 00 3d 6e 27 40 00 80 06 00 00
0010	7f 00 00 01 7f 00 00 01 e7 30 00 50 fc 73 d0 e5
0020	72 a5 ef 7a 50 18 08 04 75 19 00 00 81 8f 5a a6
0030	28 69 32 c3 44 05 35 86 5f 0c 38 d5 47 0a 31 c3
0040	5c

而报文内容经过掩码，不是直接可见的明文，但掩码的安全强度几乎是零，用“Masking-key”简单地异或一下就可以转换出明文。

WebSocket 的握手

和 TCP、TLS 一样，WebSocket 也要有一个握手过程，然后才能正式收发数据。

这里它还是搭上了 HTTP 的“便车”，利用了 HTTP 本身的“协议升级”特性，“伪装”成 HTTP，这样就能绕过浏览器沙盒、网络防火墙等等限制，这也是 WebSocket 与 HTTP 的另一个重要关联点。

WebSocket 的握手是一个标准的 HTTP GET 请求，但要带上两个协议升级的专用头字段：

- “Connection: Upgrade”，表示要求协议“升级”；
- “Upgrade: websocket”，表示要“升级”成 WebSocket 协议。

另外，为了防止普通的 HTTP 消息被“意外”识别成 WebSocket，握手消息还增加了两个额外的认证用头字段（所谓的“挑战”，Challenge）：

- Sec-WebSocket-Key：一个 Base64 编码的 16 字节随机数，作为简单的认证密钥；
- Sec-WebSocket-Version：协议的版本号，当前必须是 13。

领资料




```
▼ Hypertext Transfer Protocol
  > GET /38-0 HTTP/1.1\r\n
    Upgrade: websocket\r\n
    Host: 127.0.0.1:80\r\n
    Sec-WebSocket-Key: y7KXwBSpVrxtkR00+bQt+Q==\r\n
    Sec-WebSocket-Version: 13\r\n
    Connection: Upgrade\r\n
    \r\n
```

服务器收到 HTTP 请求报文，看到上面的四个字段，就知道这不是一个普通的 GET 请求，而是 WebSocket 的升级请求，于是就不走普通的 HTTP 处理流程，而是构造一个特殊的“101 Switching Protocols”响应报文，通知客户端，接下来就不用 HTTP 了，全改用 WebSocket 协议通信。（有点像 TLS 的“Change Cipher Spec”）

WebSocket 的握手响应报文也是有特殊格式的，要用字段“Sec-WebSocket-Accept”验证客户端请求报文，同样也是为了防止误连接。

具体的做法是把请求头里“Sec-WebSocket-Key”的值，加上一个专用的 UUID “258EAF5E914-47DA-95CA-C5AB0DC85B11”，再计算 SHA-1 摘要。

复制代码

```
1 encode_base64(
2     sha1(
3         Sec-WebSocket-Key + '258EAF5E914-47DA-95CA-C5AB0DC85B11' ))
```

客户端收到响应报文，就可以用同样的算法，比对值是否相等，如果相等，就说明返回的报文确实是刚才握手时连接的服务器，认证成功。

握手完成，后续传输的数据就不再是 HTTP 报文，而是 WebSocket 格式的二进制帧了。

领资料



No.	Time	Source	Destination	Protocol	Info
6	0.033972	127.0.0.1	127.0.0.1	TCP	59184 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=65495 WS=2!
7	0.034022	127.0.0.1	127.0.0.1	TCP	80 → 59184 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=
8	0.034063	127.0.0.1	127.0.0.1	TCP	59184 → 80 [ACK] Seq=1 Ack=1 Win=525568 Len=0
9	0.034155	127.0.0.1	127.0.0.1	HTTP	GET /38-0 HTTP/1.1
10	0.034176	127.0.0.1	127.0.0.1	TCP	80 → 59184 [ACK] Seq=1 Ack=156 Win=525568 Len=0
11	0.035689	127.0.0.1	127.0.0.1	HTTP	HTTP/1.1 101 Switching Protocols
12	0.035725	127.0.0.1	127.0.0.1	TCP	59184 → 80 [ACK] Seq=156 Ack=195 Win=525312 Len=0
13	0.035878	127.0.0.1	127.0.0.1	WebSocket	WebSocket Ping [FIN] [MASKED]
14	0.035902	127.0.0.1	127.0.0.1	TCP	80 → 59184 [ACK] Seq=195 Ack=162 Win=525568 Len=0
15	0.035982	127.0.0.1	127.0.0.1	WebSocket	WebSocket Pong [FIN]
16	0.036004	127.0.0.1	127.0.0.1	TCP	59184 → 80 [ACK] Seq=162 Ack=197 Win=525312 Len=0

>	Transmission Control Protocol, Src Port: 80, Dst Port: 59184, Seq: 1, Ack: 156, Len: 194
▼	Hypertext Transfer Protocol
>	HTTP/1.1 101 Switching Protocols\r\n
	Server: openresty/1.15.8.1\r\n
	Date: Thu, 20 Jun 2019 01:53:05 GMT\r\n
	Connection: upgrade\r\n
	Upgrade: websocket\r\n
	Sec-WebSocket-Accept: VabQ3HadzXWk54h5+9KssbNVS+s=\r\n
	\r\n
	[HTTP response 1/1]
	[Time since request: 0.001534000 seconds]
	[Request in frame: 9]

小结

浏览器是一个“沙盒”环境，有很多的限制，不允许建立 TCP 连接收发数据，而有了 WebSocket，我们就可以在浏览器里与服务器直接建立“TCP 连接”，获得更多的自由。

不过自由也是有代价的，WebSocket 虽然是在应用层，但使用方式却与“TCP Socket”差不多，过于“原始”，用户必须自己管理连接、缓存、状态，开发上比 HTTP 复杂的多，所以是否要在项目中引入 WebSocket 必须慎重考虑。

1. HTTP 的“请求 – 应答”模式不适合开发“实时通信”应用，效率低，难以实现动态页面，所以出现了 WebSocket；
2. WebSocket 是一个“全双工”的通信协议，相当于对 TCP 做了一层“薄薄的包装”，让它运行在浏览器环境里；
3. WebSocket 使用兼容 HTTP 的 URI 来发现服务，但定义了新的协议名“ws”和“wss”，端口号也沿用了 80 和 443；
4. WebSocket 使用二进制帧，结构比较简单，特殊的地方是有个“掩码”操作，客户端发数据必须掩码，服务器则不用；
5. WebSocket 利用 HTTP 协议实现连接握手，发送 GET 请求要求“协议升级”，握手过程中有个非常简单的认证机制，目的是防止误连接。

领资料



课下作业

1. WebSocket 与 HTTP/2 有很多相似点，比如都可以从 HTTP/1 升级，都采用二进制帧结构，你能比较一下这两个协议吗？
2. 试着自己解释一下 WebSocket 里的”Web“和”Socket“的含义。
3. 结合自己的实际工作，你觉得 WebSocket 适合用在哪些场景里？

欢迎你把自己的学习体会写在留言区，与我和其他同学一起讨论。如果你觉得有所收获，也欢迎把文章分享给你的朋友。



== 课外小贴士 ==

- 01 WebSocket 标准诞生于 2011 年，比 HTTP/2 要大上四岁。
- 02 最早 WebSocket 只能从 HTTP/1.1 升级，因为 HTTP/2 取消了 “Connection” 头字段和协议升级机制，不能跑在 HTTP/2 上，所以就有草案提议扩展 HTTP/2 支持 WebSocket，后来形成了 RFC8441。
- 03 虽然 WebSocket 完全借用了 HTTP 的 URI 形式，但也有一点小小的不兼容：不支持 URI 后面的 “#” 片段标识，“#” 必须被编码为 “%23”。

领资料




04 WebSocket 强制要求客户端发送数据必须使用掩码，这是为了提供最基本的安全防护，让每次发送的消息都是随机、不可预测的，抵御“缓存中毒”攻击。但如果运行在 SSL/TLS 上，采用加密通信，那么掩码就没有必要了。

05 WebSocket 协议里的 PING、PONG 帧对于保持长连接很重要，可以让链路上总有数据在传输，防止被服务器、路由、网关认为是“无效连接”而意外关闭。

分享给需要的人，Ta 订阅超级会员，你将得 50 元

Ta 单独购买本课程，你将得 20 元

 生成海报并分享

 赞 9  提建议

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

领资料

上一篇 37 | CDN：加速我们的网络服务

下一篇 39 | HTTP 性能优化面面观（上）



JVM + NIO + Spring

各大厂面试题及知识点详解

限时免费



精选留言 (32)

写留言



许童童

2019-08-23

思考题：

1.WebSocket 和 HTTP/2 都是用来弥补HTTP协议的一些缺陷和不足，WebSocket 主要解决双向通信、全双工问题，HTTP/2 主要解决传输效率的问题，两者在二进制帧的格式上也不太一样，HTTP/2 有多路复用、优先级和流的概念。

2.试着自己解释一下 WebSocket 里的”Web“和”Socket“的含义。

Web就是HTTP的意思，Socket就是网络编程里的套接字，也就是HTTP协议上的网络套接字，可以任意双向通信。

3.结合自己的实际工作，你觉得 WebSocket 适合用在哪些场景里？

IM通信，实时互动，回调响应，数据实时同步。

作者回复: great。



22



领资料



Leon

2019-08-23

工作场景遇到过用户订阅股票的股价，股价波动时实时推送给海量订阅的用户，面试场景被问

到两次，一 千万粉丝的明星发布动态如何推送给粉丝 二 海量用户的主播直播如何推送弹幕
当时回答消息队列，其实web socket才是比较好的方案

作者回复: WebSocket适合实时通信交互的场景，和消息队列其实是两个领域，不冲突，可以互相结合使用。

共 3 条评论 >

👍 10



cugphoenix

2020-02-07

是不是可以这样理解：HTTP是基于TCP的，通过TCP收发的消息用HTTP的应用层协议解析。WebSocket是首先通过HTTP协议把TCP链接建好，然后通过Upgrade字段进行协议转换，在收到服务器的101 Switching Protocols应答之后，后续的TCP消息就通过WebSocket协议解析。

作者回复: 理解的基本正确，把WebSocket和http、tcp的关系理顺了。



👍 6



大土豆

2020-02-28

WebSocket定义为tcp over web，我个人感觉是不妥的，应该是"可靠有序的传输层 + 实现组包协议的应用层的长连接方案 over web"。WebSocket和HTTP已经有包的结构了,业务直接可以用了，浏览器A发一个websocket包，比如说数据是1234，给服务器，服务器可以获取这个包，数据是1234，和http一样了。而tcp还是原始的没头没尾的字节流，想要通讯，还得再自定义一个应用层的协议。

作者回复: 说的很对。

正文里的tcp over web只是一种比喻，强调了WebSocket与http的区别，与tcp的相似性，不是那么严谨。



👍 5

领资料



chao

2019-09-18

1、第二个字节后 7 位是“Payload len”，表示帧内容的长度。它是另一种变长编码，最少 7 位，最多是 7+64 位，也就是额外增加 8 个字节，所以一个 WebSocket 帧最大是 2^{64} 。
2、如果数据的长度小于等于125个字节，则用默认的7个bit来标示数据的长度；
如果数据的长度为126个字节，则用后面相邻的2个字节来保存一个16bit位的无符号整数作为数据的长度；



如果数据的长度大于等于127个字节，则用后面相邻的8个字节来保存一个64bit位的无符号整数作为数据的长度；

老师，2是其它地方看到的，Payload len 这样设计的原因是什么，以及没明白为啥126个字节的长度要用16bit来表示

作者回复: 我个人也觉得WebSocket的变长编码设计的很奇怪。

第二个字节最高位被mask占用，所以低7位表示长度，最多127。

那么125一下在低7位就够了，126用作标志位，表示后续使用两个字节，127又是另外一个标志位，表示后面是四个字节。

所以超过125后低7位就不再是长度的含义了，而是标志位：126=>2 bytes, 127=> 4 bytes。



4



夏目

2019-12-10

两年前我实习的时候公司项目用过，到现在我才搞清楚和http的区别，惭愧...

作者回复: 这两个确实很像，第一次接触的人（包括我）也是容易弄糊涂。



2



Daiver

2020-01-23

还有socket.io，算是websocket的超集了。

作者回复: socket.io是一个开发框架，而WebSocket是传输协议，两者虽然有联系，但不能混在一起。

共 2 条评论 >

1



Cris

2019-08-23

老师，我想问下，uri里的端口号，有什么用？为什么它是和协议对应的（http默认80，https默认443），却又写在域名的后面？

作者回复: 可以参考一下第6讲，端口号是跟tcp协议相关的概念。

因为域名实际上是ip地址的等价替换，所以端口号就可以跟在域名后面。

领资料





1

**徐海浪**

2019-08-23

1. WebSocket 与 HTTP/2 有很多相似点，比如都可以从 HTTP/1 升级，都采用二进制帧结构，你能比较一下这两个协议吗？

差别：HTTP/2是请求与响应的模式，而WebSocket是双向的，服务器也可以主动向客户端发起请求。

2. 试着自己解释一下 WebSocket 里的”Web“和”Socket“的含义。

是基于web服务器，类似于tcp的socket方式来使用的协议。

3. 结合自己的实际工作，你觉得 WebSocket 适合用在哪些场景里？

我在实际工作中还没有用到WebSocket，觉得适合服务器主动推送的客户端的场景，比如站内信或者站内聊天，或者在线页游？

作者回复：

1.在WebSocket里没有请求响应的概念，收发的都是数据帧，通信的双方可以自己解释帧的含义。

2.应该是基于web，也就是http协议。

3.对。

共 2 条评论 >



1

**-W.LI-**

2019-08-23

老师好!websocket单机服务器能支持多少链接啊?之前没用过websocket。看帖子好像是通过key-value形式存储所有链接。需要用得时候通过key拿到链接往外写数据。希望老师科普下websocket的简单应用和实现，性能分析。

需要服务器主动推的感觉都可以用websocket做。

聊天工具:用户A，用户B，

A->服务器(保存聊天记录)->B;B->服务器(保存聊天记录)->A;是这样么？

作者回复：

WebSocket其实就是给tcp加了一层简单的包装，所以它的并发能力取决于服务器，并不是kv的形式，你应该把它理解成运行在http上的tcp，用tcp的思路去考虑它。



1

领资料

**Jasmine**

2021-11-22

第二个字节后 7 位是“Payload len”，表示帧内容的长度。它是另一种变长编码，最少 7 位，最多是 7+64 位，也就是额外增加 8 个字节，所以一个 WebSocket 帧最大是 2^{64} 。——这里最大的帧为什么不说是 2^{71} 呢？

作者回复: 前面的7位是标志位，已经被占用了，所以只能用64位。



CCX

2021-09-30

在 FINTECH 领域工作了几年了，自研发的外汇/数字货币交易系统的行情模块基本都是 web socket 实现的；另外还遇到一个有意思的场景，就是 discord 的机器人 Slash Commands 的实现也是基于 websocket 的。

作者回复: websocket在如今的互联网大环境下非常有用，基于http握手建连让它可以很容易运行在各种web服务上。



Unknown element

2021-06-04

老师问下

- 1.Web socket既然没有流的概念那具体是怎么实现全双工的呢 两边都同时收发那包的顺序就乱了吧
- 2.Web socket帧里帧长度最大是7+64位那最长不应该是 2^{71} 位吗 为什么课程里说最长是 2^{64} ?

作者回复:

1.websocket本质上和tcp是一样的，双工是指通信的双方可以同时发送数据，但一个方向上的数据包还是有序的。

2.注意它用了变长编码，超过127才会用8个字节来表示长度，所以最大长度是 2^{64} 。

共 2 条评论 >



领资料



张峰

2021-04-20

服务器发送事件（Server-sent Events）是基于 WebSocket 协议的一种服务器向客户端发送事件和数据的单向通讯。

HTML5 服务器发送事件（server-sent event）允许网页获得来自服务器的更新。



老师能讲解一下 Server-sent Events 和 websocket的相同和不同。

作者回复: 对这两个不是太了解, 感觉像是js里的编程概念吧, 与传输协议本身没有关系。

抱歉了。



blueBean

2021-04-08

帧那里, 第一个字节的第一位应该就是一个bit吧, 为啥能放下FIN这个字段呢? 想不通

作者回复: FIN只是一个代号, 并不是三个字符, 也就是说0是没结束, 1是结束。



脱缰的野马__

2021-03-20

老师你好, 请问服务端在使用websocket主动推送信息给客户端的时候, 是如何知道客户端的呢? 另外服务端要主动推送的客户端有成千上万个, 哪又如何推送? 不像客户端请求服务端, 客户端请求服务端是使用http, 有ip和端口。

作者回复: WebSocket和http一样, 客户端要发起连接, 接下来就和tcp一样了, 服务器在内存里保持多个与客户端的连接。

如果会Python, 可以找相关的库试一试。



乘风破浪

2021-01-24

罗大师, 所有WS的页面都无法打开, 错误提示如下,
无法访问此网站网址为 ws://www.chrono.com:8080/srv 的网页可能暂时无法连接, 或者它已永久性地移动到了新网址。

ERR_DISALLOWED_URL_SCHEME

无法访问此网站网址为 ws://127.0.0.1/38-0 的网页可能暂时无法连接, 或者它已永久性地移动到了新网址。

ERR_DISALLOWED_URL_SCHEME

领资料



wireshark也抓不到任何包

作者回复: 可能是本地的dns解析缓存的问题, 可以考虑用docker环境试试, 或者用一个干净的虚拟机环境重新搭建。

共 3 条评论 >



火锅小王子

2020-09-18

老师您好, 可以解释一下其中的帧是如何具体划分的吗 比如发送一个大的内容, 分割帧有什么依据, 还是说依赖mss或者滑动窗口这些内容?

作者回复: WebSocket的帧划分是由应用自己定义的, 和http/2差不多, 它的下层还是交给tcp发送, 所以不用考虑传输的底层细节。



Maske

2020-06-29

- 1.都是建立在tcp/ip之上, 默认端口和http(s)一样。不同点: http2专注于解决http1.1的性能问题。websocket是一种全双工通信协议, 解决的是能够让服务器端主动推送数据给客户端的问题, 使得web应用不再局限于‘请求——应答’模式。
- 2.‘web’表示是基于web浏览器层面的, 在前端通过js动态控制网络的连接与断开、数据收发操作。socket是一种应用进程与网络通信协议之间的连接机制。
- 3.实时聊天应用, 金融领域的股价股指等走势图的绘制也需要实时获取最新的数据, 导航应用也需要获取实时路况。

作者回复: good, 快学完了, 加油。

共 2 条评论 >



我母鸡啊!

2020-05-24

1.WebSocket 与 HTTP/2 有很多相似点, 比如都可以从 HTTP/1 升级, 都采用二进制帧结构, 你能比较一下这两个协议吗?

websocket里面有帧的概念, 却没有http2.0里的虚拟流的概念, 也不存在优先级, 多路复用。websocket的出现本质上还是为了解决http的半双工的问题, 变成全双工, 服务器和客户端可以随意通行的问题

领资料



作者回复: 说的很好。

