

- [一、理解长连接](#)
- [二、深入理解WebSocket](#)
- [三、HTTP vs WebSocket](#)

## 一、IM核心技术详解

阶段目标：理解IM系统与之前的微服务体系的区别。理解长连接与WebSocket。

# 一、理解长连接

---

之前开发的系统中，所有的数据请求都是固定的从客户端往服务端发起请求。但是在IM场景中，数据请求不光需要从客户端往服务器端发起，同样也需要服务端向客户端发起。因此，这种场景下，就不能简单的使用HTTP请求来构建，而需要使用长连接。

关于HTTP和长连接的区别，可以简单的对比为发短信和打电话。

- HTTP协议就像是发短信，所有请求只允许从客户端向服务端主动发起(发短信)，然后客户端等待服务端响应(回短信)。如果没有客户端发起的请求，服务端就无法向客户端主动推送消息。这种方式不太灵活，而且时效性也不太高。
- 长连接的工作方式就像是打电话。只要客户端和服务端之间建立了连接(电话接通了)。那么双方都可以主动进行通话。就算客户端什么都不做，服务端也可以主动向客户端推送消息(说话)。这种方式交互更灵活，时效性通常也更高。但是显然，相对会比较麻烦。

对于IM场景，客户端要随时可以接收到其他人发的聊天信息。虽然也可以基于HTTP协议，采用定期轮训的方式去服务端查询自己的信息，但是，不管是时效性，还是服务端的负担，都是不太合适的。所以长连接是更合适的选择。

典型的长连接技术有WebSocket和TCP长连接两种。其中：

- TCP协议是一种可靠的、基于连接的通信协议。他确保网络层的数据安全有效传递。TCP连接直接通过三次握手过程建立，不需要HTTP协议的参与。适用于需要可靠数据传输的应用场景。通常用于服务器后端，或者APP与服务器之间互连的场景。
- Websocket协议则是基于TCP的一种新的网络协议。他实现了浏览器与服务器全双工(full-duplex)通信，即允许服务器主动发送消息给客户端。Websocket的连接可以由HTTP协议升级建立，因此非常实用于前端浏览器与后端服务器之间进行持续稳定的数据传递。

因此，对于仿抖音直播项目来说，可以选择Websocket协议来实现IM场景。

# 二、深入理解WebSocket

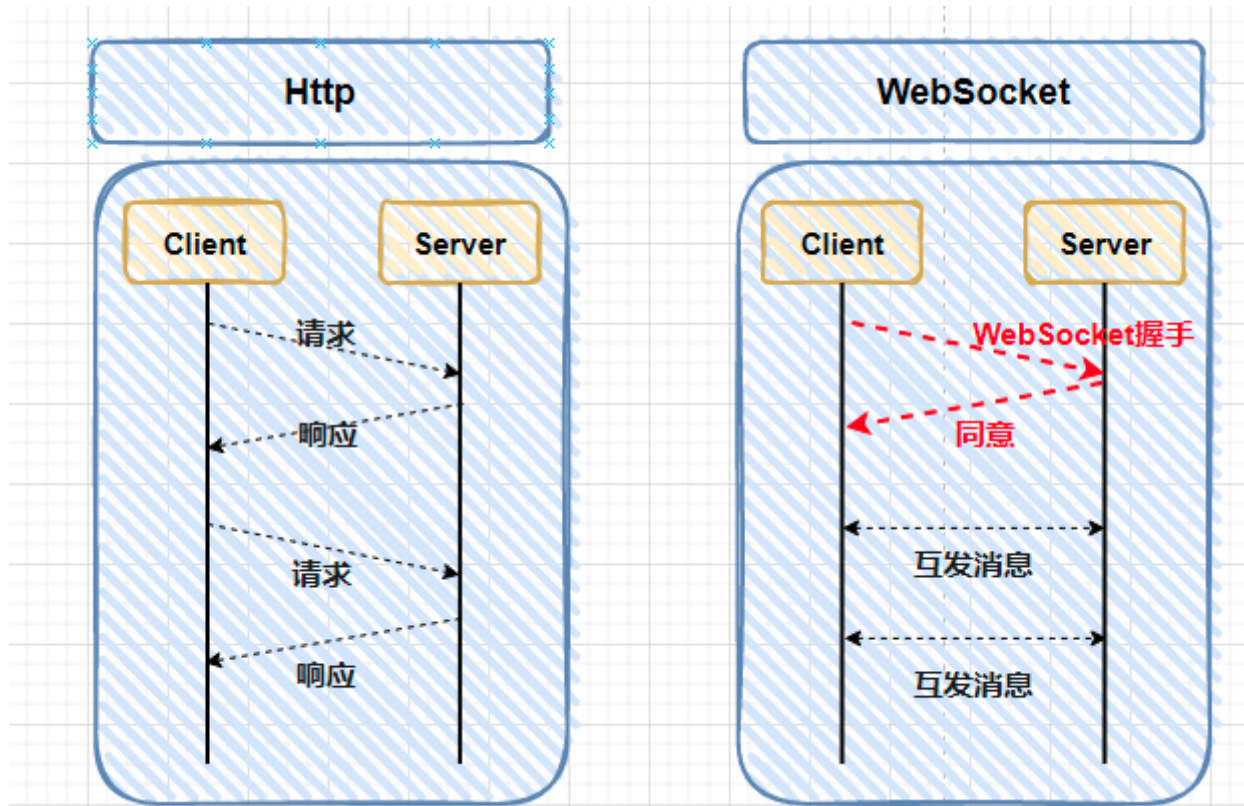
---

WebSocket协议提供了一种标准的执行方式，可以在客户端与服务器之间通过一个单独的TCP连接，建立一个全双工、双向数据传输的通道。

全双工（Full Duplex）是通讯传输的一个术语。通信允许数据在两个方向上同时传输，它在能力上相当于两个单工通信方式的结合。全双工指可以同时（瞬时）进行信号的双向传输（A→B且B→A）。指A→B的同时B→A，是瞬时同步的。单工就是在只允许甲方向乙方传送信息，而乙方不能向甲方传送。（比喻汽车的单行道。）（查看更多：全双工——百度百科）

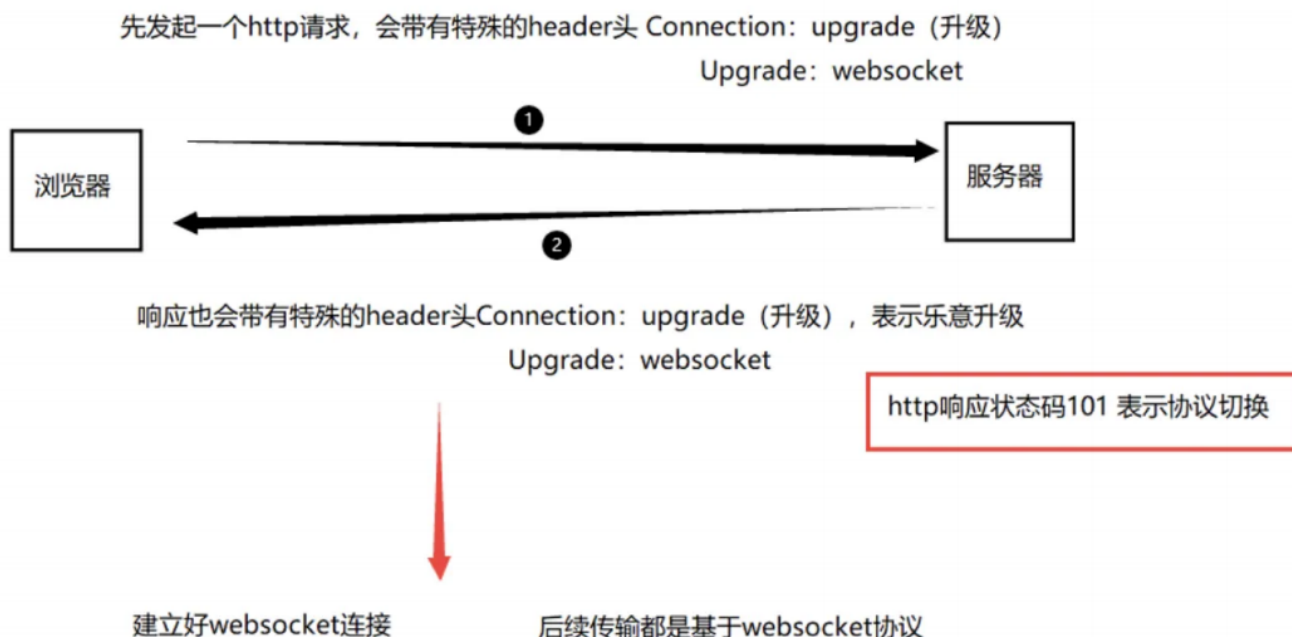
在WebSocket协议中，浏览器与服务器只需要完成一次握手，两者之间就直接可以创建持久性的连接，并进行双向数据传输，从而使得客户端与服务器之间的数据交互编的更加简单。

注意，这里所谓的“浏览器和服务器只需要完成一次握手”，意思是说在三次握手完成TCP连接之后，还会传输一次握手数据：



WebSocket是一种与HTTP协议不同的TCP协议，但是WebSocket却被设计为基于HTTP工作。默认使用端口80和443，并允许重复使用现有的防火墙规则。

WebSocket交互从HTTP请求开始，该请求使用HTTP Upgrade头进行升级，或者在这种情况下，切换到WebSocket协议。



客户端与服务端成功完成握手请求后，HTTP升级而来的TCP连接将会保持打开状态，以便客户端和服务端之间继续发送和接收消息。

### 三、HTTP vs WebSocket

尽管WebSocket被设计为兼容HTTP协议，但是，你必须理解这两种协议有非常不同的应用架构以及编程模型。

在HTTP或者REST中，应用程序是通过多个URL进行组织的。客户端通过请求-响应的风格与服务端进行交互。而服务端根据HTTP URL、方法和请求头等信息，将请求路由到适当的处理程序，并进行处理。

相比之下，在WebSocket中，初始连接通常只有一个URL。随后，所有应用程序信息都在同一个TCP连接上传递。这就指向了一个完全不同的异步、事件驱动的消息传递体系结构。

WebSocket也是一种低级传输协议，与HTTP不同，他不对消息的内容规定任何语义。这意味着，除非客户端和服务端在消息语义上达成一致，否则无法路由或处理消息。

WebSocket客户端和服务端都可以通过HTTP握手请求上的Sec-WebSocket-Protocol协议头协商使用更高级的消息传递协议。

WebSockets可以使网页具有动态性和交互性。然而，在许多情况下，AJAX和HTTP流或长轮询的组合可以提供简单有效的解决方案。

例如，新闻、邮件和社交订阅源需要动态更新，但每隔几分钟更新一次可能是完全可以的。另一方面，游戏和金融应用程序需要更接近实时。

延迟本身并不是决定性因素。如果消息量相对较低（例如，监视网络故障），HTTP流式传输或轮询可以提供有效的解决方案。正是低延迟、高频率和高容量的结合，才是使用WebSocket的最佳理由。

还要记住，在Internet上，您无法控制的限制性代理可能会阻止WebSocket交互，原因可能是它们未配置为传递Upgrade标头，也可能是因为它们关闭了看起来空闲的长期连接。这意味着对防火墙内的内部应用程序使用WebSocket比对面向公众的应用程序使用更简单。

更多信息参考Spring官网对于WebSocket的介绍: <https://docs.spring.io/spring-framework/reference/web/websocket.html>