03 | 轮询与长连接：如何解决消息的实时到达问题？

2019-09-02 袁武林

即时消息技术剖析与实战 进入课程 >

讲述：袁武林
时长 11:27 大小 13.12M

你好，我是袁武林。

我在前面第一篇文章中，从使用场景的需求方面，讲到了 IM 系统的几个比较重要的特性。其中之一就是“消息到达的实时性”。

实时性场景是所有的 IM 系统绕不开的话题，为了支持互联网的“实时互联”的概念，大部分的 App 都需要实时技术的支持。

我们现在使用的聊天类 App、直播互动类 App 都已经在实时性方面做得很好了，消息收发延迟基本都能控制在毫秒级别。

当然这一方面得益于快速发展的移动网络，让网络延迟越来越低、网络带宽越来越高；另一个重要原因是：社交网络 App 在实时性提升方面的技术，也在不断升级迭代。

实时性主要解决的问题是：当一条消息发出后，我们的系统如何确保这条消息最快被接收人感知并获取到，并且尽量让耗费的资源较少。这里关键的几个点是：最快触达，且耗费资源少。

想好了吗？下面我们来看一看，IM 在追求“消息实时性”的架构上，所经历过的几个代表性阶段。

短轮询场景

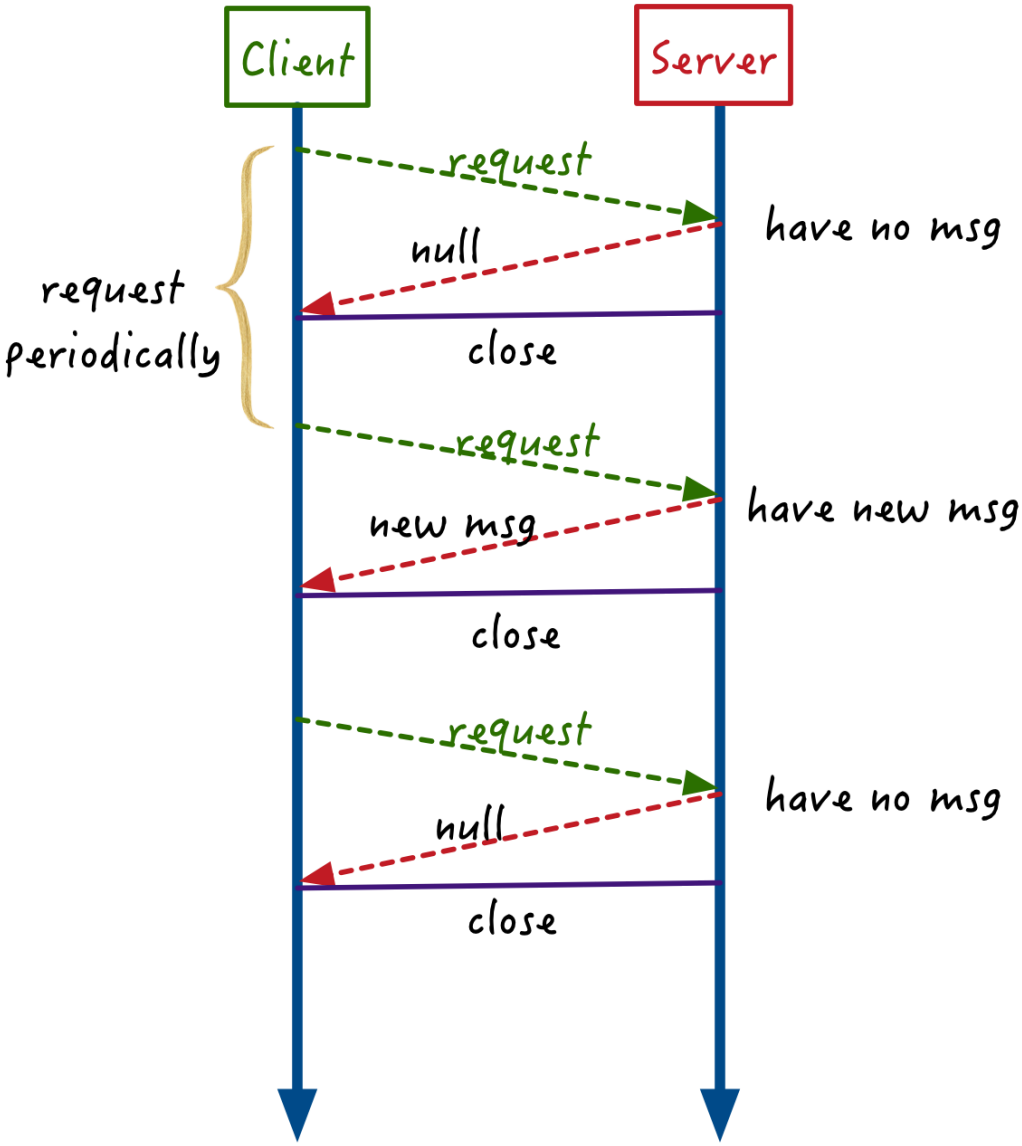
在 PC Web 的早期时代，对于数据的获取，大部分应用采用一问一答的“请求响应”式模式，实际上，像现在我们浏览大部分门户网站的新闻，以及刷微博其实都是采用的“请求响应”模式。

但这种依赖“手动”触发的模式，在即时消息系统中当有新消息产生时并不能很好地感知并获取到，所以明显不适用于对实时性要求高的场景。

因此，这个时期的 IM 软件很多采用了一种“短轮询”的模式，来定期、高频地轮询服务端的新消息。

在短轮询模式中，服务器接到请求后，如果有新消息就会将新消息返回给客户端，如果没有新消息就返回空列表，并关闭连接。

这种短轮询的方式就好像一位焦急等待重要信件的人，每天骑车跑到家门口的邮局去问是否有自己的信件，有就拿回家，没有第二天接着去邮局问。



作为从一问一答的请求响应模式孵化出来的短轮询模式，具有较低的迁移升级成本，比较容易落地。但劣势也很明显：

为了提升实时性，短轮询的频率一般较高，但大部分轮询请求实际上是无用的，客户端既费电也费流量；

高频请求对服务端资源的压力也较大，一是大量服务器用于扛高频轮询的 QPS（每秒查询率），二是对后端存储资源也有较大压力。

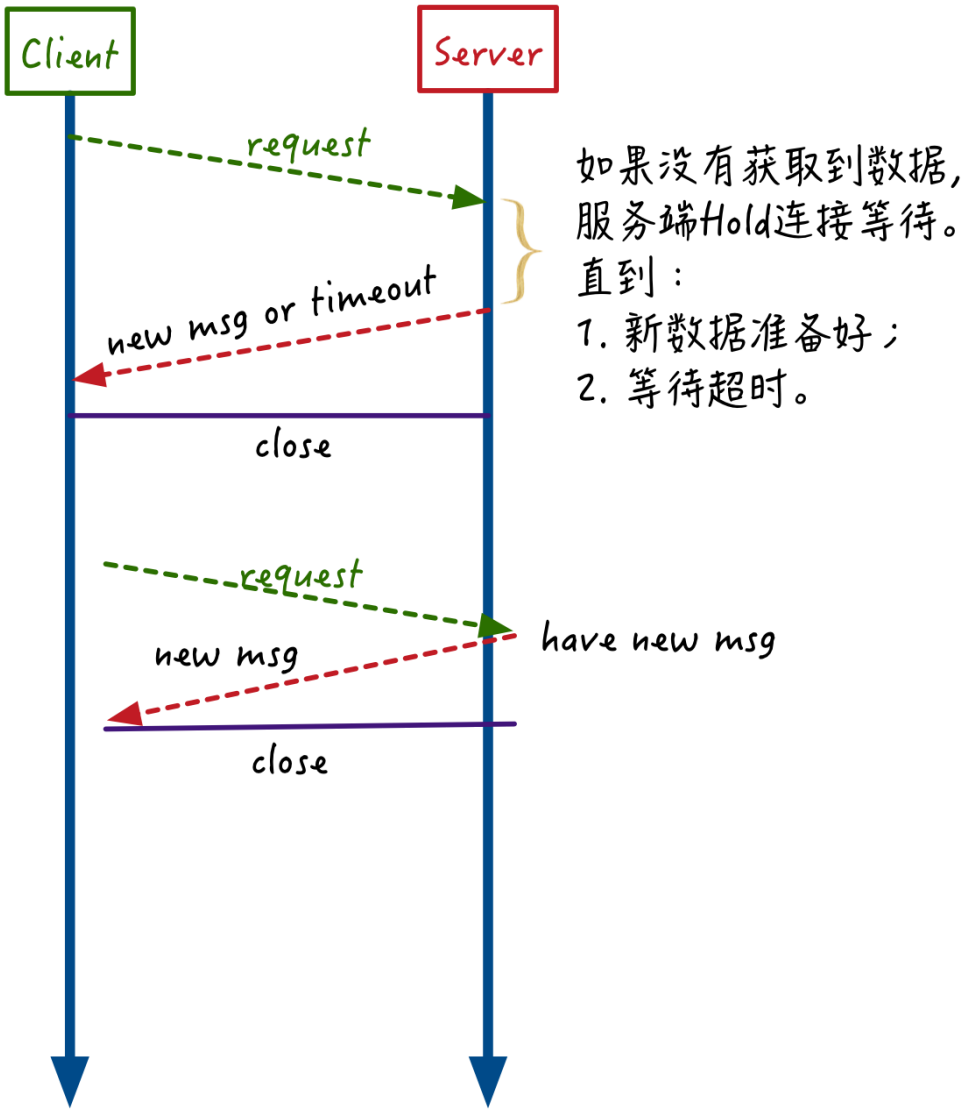
因此，“短轮询”这种方式，一般多用在用户规模比较小，且不愿花费太多服务改造成本的小型应用上。

长轮询场景

正是由于“短轮询”存在着高频无用功的问题，为了避免这个问题，IM 逐步进化出“长轮询”的消息获取模式。

长轮询和短轮询相比，一个最大的改进之处在于：短轮询模式下，服务端不管本轮有没有新消息产生，都会马上响应并返回。而长轮询模式当本次请求没有获取到新消息时，并不会马上结束返回，而是会在服务端“悬挂（hang）”，等待一段时间；如果在等待的这段时间内有新消息产生，就能马上响应返回。

这种方式就像等待收信的人每天跑到邮局去问是否有自己的信件，如果没有，他不是马上回家，而是在邮局待上一天，如果还是没有就先回家，然后第二天再来。



比较之下，我们会发现，长轮询能大幅降低短轮询模式中客户端高频无用的轮询导致的网络开销和功耗开销，也降低了服务端处理请求的 QPS，相比短轮询模式而言，显得更加先进。

长轮询的使用场景多见于： 对实时性要求比较高，但是整体用户量不太大。它在不支持 WebSocket 的浏览器端的场景下还是有比较多的使用。

但是长轮询并没有完全解决服务端资源高负载的问题，仍然存在以下问题。

1. 服务端悬挂（hang）住请求，只是降低了入口请求的 QPS，并没有减少对后端资源轮询的压力。假如有 1000 个请求在等待消息，可能意味着有 1000 个线程在不断轮询消息存储资源。

2. 长轮询在超时时间内没有获取到消息时，会结束返回，因此仍然没有完全解决客户端“无效”请求的问题。

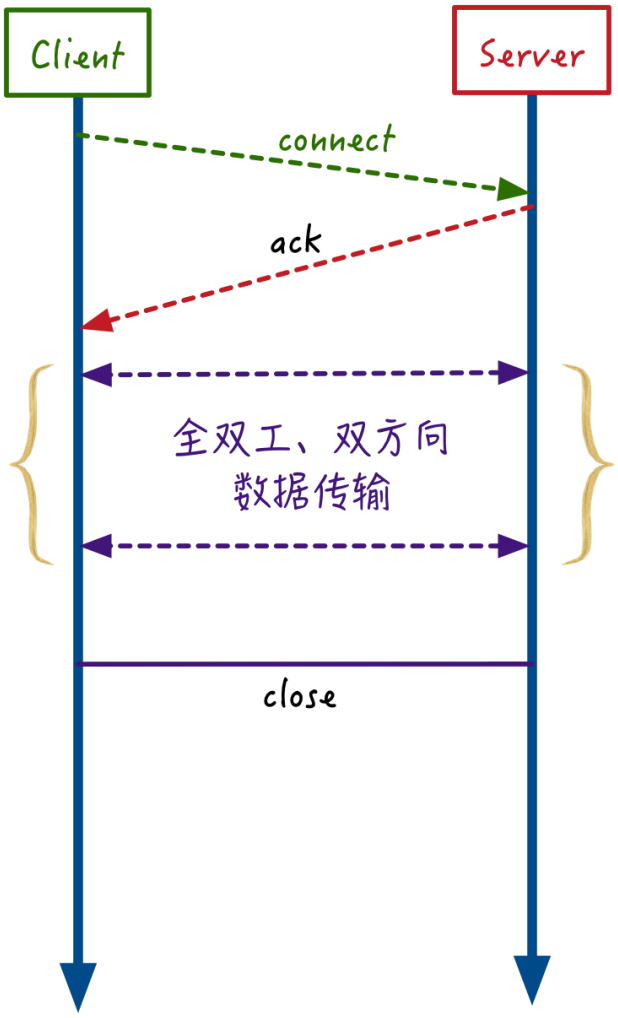
服务端推送：真正的边缘触发

短轮询和长轮询之所以没法做到基于事件的完全的“边缘触发（当状态变化时，发生一个 IO 事件）”，这是因为服务端在有新消息产生时，没有办法直接向客户端进行推送。

这里的根本原因在于短轮询和长轮询是基于 HTTP 协议实现的，由于 HTTP 是一个无状态协议，同一客户端的多次请求对于服务端来说并没有关系，也不会去记录客户端相关的连接信息。

因此，所有的请求只能由客户端发起，服务端由于并不记录客户端状态，当服务端接收到新消息时，没法找到对应的客户端来进行推送。

随着 HTML5 的出现，全双工的 WebSocket 彻底解决了服务端推送的问题。



这就像之前信件处理的逻辑，等待收信的用户不需要每天都跑到邮局去询问，而只要在邮局登记好自己家里的地址。等真正有信件时，邮局会派专门的邮递员按照登记的地址来把信送过去。

同样，当他需要写信给别人时，也只需要填好收件人地址，然后把信交给邮递员就可以了，不需要再自己跑邮局。

WebSocket

WebSocket 正是一种服务端推送的技术代表。

随着 HTML5 的出现，基于单个 TCP 连接的全双工通信的协议 WebSocket 在 2011 年成为 RFC 标准协议，逐渐代替了短轮询和长轮询的方式，而且由于 WebSocket 协议获得了

Web 原生支持，被广泛应用于 IM 服务中，特别是在 Web 端基本属于 IM 的标配通信协议。

和短轮询、长轮询相比，基于 WebSocket 实现的 IM 服务，客户端和服务端只需要完成一次握手，就可以创建持久的长连接，并进行随时的双向数据传输。当服务端接收到新消息时，可以通过建立的 WebSocket 连接，直接进行推送，真正做到“边缘触发”，也保证了消息到达的实时性。

WebSocket 的优点是：

1. 支持服务端推送的双向通信，大幅降低服务端轮询压力；
2. 数据交互的控制开销低，降低双方通信的网络开销；
3. Web 原生支持，实现相对简单。

TCP 长连接衍生的 IM 协议

除了 WebSocket 协议，在 IM 领域，还有其他一些常用的基于 TCP 长连接衍生的通信协议，如 XMPP 协议、MQTT 协议以及各种私有协议。

这些基于 TCP 长连接的通信协议，在用户上线连接时，在服务端维护好连接到服务器的用户设备和具体 TCP 连接的映射关系，通过这种方式客户端能够随时找到服务端，服务端也能通过这个映射关系随时找到对应在线的用户的客户端。

而且这个长连接一旦建立，就一直存在，除非网络被中断。这样当有消息需要实时推送给某个用户时，就能简单地通过这个长连接实现“服务端实时推送”了。

但是上面提到的这些私有协议都各有优缺点，如：XMPP 协议虽然比较成熟、扩展性也不错，但基于 XML 格式的协议传输上冗余比较多，在流量方面不太友好，而且整体实现上比较复杂，在如今移动网络场景下用的并不多。

而轻量级的 MQTT 基于代理的“发布 / 订阅”模式，在省流量和扩展性方面都比较突出，在很多消息推送场景下被广泛使用，但这个协议并不是 IM 领域的专有协议，因此对于很多 IM 下的个性化业务场景仍然需要大量复杂的扩展和开发，比如不支持群组功能、不支持离线消息。

因此，对于开发人力相对充足的大厂，目前很多是基于 TCP（或者 UDP）来实现自己的私有协议，一方面私有协议能够贴合业务需要，做到真正的高效和省流；另一方面私有协议相对安全性更高一些，被破解的可能性小。目前主流的大厂很多都是采用私有协议为主的方式来实现。

小结

这一篇我们介绍了即时消息服务中是如何解决“消息实时性”这个难题。

为了更好地解决实时性问题，即时消息领域经历过的几次技术的迭代升级：

- 从简单、低效的短轮询逐步升级到相对效率可控的长轮询；
- 随着 HTML5 的出现，全双工的 WebSocket 彻底解决了服务端推送的问题；
- 同时基于 TCP 长连接衍生的各种有状态的通信协议，也能够实现服务端主动推送，从而更好解决“消息收发实时性”的问题。

最后给你留一个思考题，TCP 长连接的方式是怎么实现“当有消息需要发送给某个用户时，能够准确找到这个用户对应的网络连接”？

你可以给我留言，我们一起讨论，感谢你的收听，我们下期再见。

上一篇　02 | 消息收发架构：为你的App，加上实时通信功能

精选留言 (19)

写留言

王棕生
2019-09-02

TCP 长连接的方式是怎么实现“当有消息需要发送给某个用户时，能够准确找到这个用户对应的网络连接”？
答： 首先用户有一个登陆的过程： (1)tcp客户端与服务端通过三次握手建立tcp连接；(2)基于该连接客户端发送登陆请求；(3)服务端对登陆请求进行解析和判断，如果合法，就将当前用户的uid和标识当前tcp连接的socket描述符(也就是fd)建立映射关系； (4)这个映...
展开

作者回复: 👍

3

飞翔
2019-09-02

服务器维护一个hashmap key是 userid，value 是socket instance， 这样 用户A 发给用户B的信息里边含有用户B 的id， 用usedBid 到hashmap 里边查到 用户B的socket 就可以用socket 发送信息给用户B了，老师我还有一个问题 就是如果一台机器连接数有限， 如果需要多台机器，如果用户A的socker 和用户B的socket 不在一台机器上，这样怎么解决呀
展开

作者回复: 一种做法是用户上线时维护一个全局的 uid->网关机 的映射，下发的时候就能做到精确定位；另一种方式是：下发时把所有消息下发给所有机器，每台机器如果发现当前用户连接在本机就下发，其他的就丢掉，这种会有一定的资源浪费。

7　3

云师兄
2019-09-02

多终端场景下，用户维度上的应该是一个socket列表。网关接入层是否是有状态的？在进

行推送时候需要定位到某台机器？多终端要定位多台机器？

展开

作者回复: 可以是无状态的，一种做法是用户上线时维护一个全局的 uid->网关机 的映射，下发的时候就能做到精确定位；另一种方式是：下发时把所有消息下发给所有机器，每台机器如果发现当前用户连接在本机就下发，其他的就丢掉，这种会有一定的资源浪费。

1

川杰

2019-09-02

老师您好，请教两个问题：
1、客户端和服务端只需要完成一次握手，就可以创建持久的长连接，并进行随时的双向数据传输；如果是这样的话，那是不是意味着需要服务器资源去存储这个长连接，那当客户端很多的情况下，服务器资源会不会被大量消耗。
2、在一些不是html的场景下，如微服务中消息推送，是不是还是只能通过消息队列等方...

展开

作者回复: 1. 映射关系需要服务端存储，有一定的资源开销。
2. 和html没关系哈，微服务里面的rpc调用也是可以通过tcp长连实现，不一定需要队列。

4 1

Geek_发现

2019-09-02

说一下我的猜想吧:不管是websocket全双工还是tcp有状态链接，都是应用了ack机制，用户访问服务器，ack会保存用户的信息，服务器收到ack后会开辟一块专有内存来保存所有的客户的ack信息；同理，服务器发送信息至客户，客户端也会保存服务器的ack信息。
总的来说，客户端和服务器应该都是采用异步方式来提升效率和客户体验，并且降低服务器连接压力

展开

作者回复: 嗯，ack机制是确保消息被正确接收了，下节课会讲到哈

2 1

糊涂难得

2019-09-02

老师，我想问一下mqtt的遗嘱消息，不是可以实现离线消息吗？

asdf100
2019-09-02

对于连接安全如何搞，如只允许合法的websocket连接到服务器，不允许非法的连接。目前我们是在连接的时候传递一个加密串，在服务器进行解密，不知道是否可行？

作者回复: 防止客户端的连接伪造是比较困难的哈，比如你的app源代码泄露了，稍微改一改就能仿冒APP连接上来。这种情况可以考虑采用人机验证（比如极验，比如手机验证码）来排除“机器行为”，更极端的可以采取网银的方式，通过客户端插入的U盾来提供真实访问的验证。

1

冷笑的花猫
2019-09-02

请问老师，假设mqtt有两个缺点，不支持离线和群组功能，但可以修改源码增加这两类功能，而且mqtt已基本成熟，在mqtt之上开发不更方便和快捷吗？为啥要在tcp基础上自己开发呢？如果仅仅因为这些选择了了基于tcp自己开发，感觉说服力没那么强啊。
所以请老师能详细说下这些的优缺点吗？晚上搜索到的感觉不太靠谱，谢谢。

展开

作者回复: 基于mqtt做二次开发是可以的呀，而且目前也有很多公司已经是这么做的了。只是说有开发能力的大厂更愿意自己来实现一套私有协议，主要是完全根据业务定制化，协议设计上会更高效一些。

liangjf
2019-09-02

一般链接量大了，不用内存缓存连接映射信息。而是放在redis集群中，key是mid，valie是server:channel。就可以在集群中，找到对应server的连接

展开

作者回复: 嗯，是一种方式，不过还需要考虑维护这个全局在线状态时的开销和一致性问题。

1

滚滚
2019-09-02

老师，websocket长连接，服务端推消息给客户端，怎么样确保消息到达了？就是有回调函数吗？

作者回复: 消息的可靠性可以通过应用层实现的类似tcp的ack机制来让消息接收方收到后”确认“给IM服务器。

4

Shuai
2019-09-02

请问，XMPP是基于 XML 格式的协议，那像微信这种成熟的IM软件的私有通信协议是基于什么格式的呢？二进制吗？

展开

作者回复: 据我了解微信是私有二进制协议。

落誓布偶
2019-09-02

最近用java做了一个与物联网设备的连接应用，是将连接存储到map中，然后通过设备的唯一标识为key从中取出连接。将每一个指令状态存到了db中。但这就会有种情况发生，当连接数量过大时，的服务器端cpu资源消耗过大，整个系统出现明显卡顿。这个该怎么解决？

展开

作者回复: 一个是分析一下cpu资源消耗在哪了？另外，指令存储db可以放到异步的进程或者服务中处理，连接层只处理连接相关的轻量操作。

2

Shaoyao•琚
2019-09-02

HTTP 可以是有状态的吧

展开

作者回复: 更准确的说是指：无法从服务端主动发起操作。

2019-09-02

请问袁老师，长轮询实现是不是这样：客户端发起一个请求，服务端通常是在200ms内返回结果，但长轮询的话，是有消息才立马返回；没有消息时，就先让客户端等个30s，然后服务端在这个30s内还是不停地查询消息数据，如果有新消息，就立马返回，如果还是没有新消息，就让他自己超时处理（此时客户端收到timeout响应），也有可能是正常返回空列表?

展开

作者回复: 是这样的

胡胜

2019-09-02

请问本课写的demo是基于mqtt么

展开

作者回复: 代码演示不涉及第三方协议哈

小可

2019-09-02

用户使用客户端与服务器建立连接时携带了userid与clientid，连接建立成功后，服务器端记录用户、客户端及连接之间的关系。一个可以用户使用多个客户端建立连接，一般是不同类型的客户端(网页、APP），后续有消息可根据此关系进行多端推送。

展开

作者回复:

geekymv

2019-09-02

客户端与服务端通过三次握手建立连接后，服务端会保存唯一的socket连接，socket包含了客户端的ip和port，服务端需要保存连接用户的ID和socket的映射关系（<userId，soc

ket>）。这样当服务端需要给某个用户发送消息时，通过映射关系就可以获取到socket，通过socket发送数据给客户端。

展开

作者回复: 连接层的信息映射保存时不需要关心客户端的ip和port哈，一般可以保存为socket对象或者channel对象就可以。

墙角儿的花
2019-09-02

tcp链接和用户id以及设备id 绑定，消息体有发送目标，进行匹配即可.
问下老师，http基础上实现能控制timeout的长轮询，这个怎么做的啊，谢谢

作者回复: 简单实现的话：ajax发起一个请求，超时比如设置为60s，后端servlet接收到之后，while循环判断当前时刻和请求时刻是否到达超时时间，没有的话调用后端service获取数据，如果没有获取到就sleep 1s，继续while循环。

2

超威丶
2019-09-02

请问websocket为什么能实现实时通信

展开

作者回复: websocket支持双向全双工的传输，所以可以做到服务端推送，让消息接收更加实时。

3