基于 HTML5 WebSocket 的 Web 实时通信机制的研究与实现*

陈丽枫，郑力新，王佳斌
（华侨大学 工学院，福建 泉州 362021）

摘　要： 随着互联网技术的不断发展，Web 技术在各个领域得到了不同程度的运用，人们对于 Web 应用的实时性提出了更高的要求，HTML5 WebSocket 协议因此得到了广泛的关注。通过对基于 HTTP 的传统 Web 实时通信方案进行分析，针对其中的不足与缺点，深入介绍了基于 HTML5 WebSocket 协议的实时通信机制以及相对于传统方案的优势，并通过使用 Node. js 的 Express 框架和 HTML5 WebSocket 协议的第三方应用程序编程接口 Socket. io 类库实现了一个基于 WebSocket 协议的 Web 应用。经实验表明，所描述的研究能成功地在客户端和服务器端完成基于 HTML5 WebSocket 协议的实时通信过程并建立连接。

关键词： Web 应用；WebSocket；实时通信；Socket. io
中图分类号： TP39　　**文献标识码：** A　　DOI：10.19358/j.issn.1674-7720.2016.10.030
引用格式： 陈丽枫，郑力新，王佳斌. 基于 HTML5 WebSocket 的 Web 实时通信机制的研究与实现[J]. 微型机与应用，2016，35(10)：88-91.

Study and implementation of Web real-time communication mechanism based on HTML5 WebSocket

Chen Lifeng，Zheng Lixin，Wang Jiabin
（College of Engineering，Huaqiao University，Quanzhou 362021，China）

Abstract: With the development of Internet technology，Web technology has been applied in various fields，and people put forward higher requirements to the real-time of Web application. So HTML5 WebSocket protocol gets a lot of attention. Through the analysis of the traditional Web real-time communication scheme based on HTTP，for the deficiencies and disadvantages of HTTP，this paper introduces the real-time communication mechanism based on the HTML5 WebSocket protocol and the advantages compared with the traditional solutions. By using the Express framework of Node. js and third party application programming interface Socket. io library of HTML5 WebSocket protocol，it implements a Web application based on WeBsocket protocol. The experiment shows that it can successfully complete the process of real-time communication based on the HTML5 WebSocket protocol and establish a connection between the client and server.

Key words: Web application；WebSocket；real-time communication；Socket. io

0　引言

随着互联网技术的高速发展，人们对 Web 应用的实时性要求越来越高，传统的 Web 实时通信方案已经无法满足一些现实应用的需求。在长期的 Web 应用过程中该传统方案逐渐露出资源浪费、实时性不高等问题，这些问题的出现对一些实时性要求较高的 Web 应用（如在线游戏、在线证券、设备监控等）造成了不好的用户体验。除此之外，这些不足还会制约 Web 实时通信的性能，对通信效率造成影响。面对这种情况，HTML5 规范中定义了 WebSocket 协议来实现更好的用户体验和实时通信功能，并针对传统的 Web 实时通信方案在实际运用中产生的资源浪费问题进行改善，提高通信效率。

目前，WebSocket 协议的实现主要分为客户端和服务器端两部分。对于其客户端而言，许多的主流浏览器（包括个人电脑和移动终端）如谷歌、火狐、IE 等都在不同的版本上支持 WebSocket 客户端应用程序编程接口。而对于其服务器端而言，也有许多常见的应用服务器如 WebSphere、WebLogic、Tomcat 等在不同的版本上支持 WebSocket 服务器端应用程序编程接口。综上所述，本文从传统的 Web 实时通信方案出发，针对其在 Web 应用中所体现的不足与缺点，深入研究 WebSocket 协议在 Web 实时通信方面的原理与优势，并根据该协议的通信机制进行实现。

1　传统的 Web 实时通信方案

1.1　轮询

在早期的 Web 应用中，所采用的 Web 实时通信方案是轮询。在使用轮询时，客户端需要频繁地向服务器端发送 HTTP 请求来保持客户端和服务器端的同步以便不断地刷新客户端所要呈现的信息。在这个过程中，客户端无

* 基金项目：华侨大学研究生科研创新能力培育计划项目（1400222001）

法确定合适的时间间隔向服务器端发送 HTTP 请求。若间隔的时间太短，客户端频繁的请求将会给服务器端造成巨大的压力；若间隔的时间太长，就无法满足客户端和服务器端实时通信的要求。由于客户端在频繁地发送请求时服务器端的数据可能还未进行更新，导致服务器端返回的大部分应答包中的数据域为空，因而产生了很多无谓的网络传输，浪费了大量的带宽资源和其他网络资源。对于每次的 HTTP 请求而言，过长的 HTTP 头信息也会占用不必要的带宽资源。因此，这是一种缺乏灵活性又低效的 Web 实时通信方案。其中客户端和服务器端的交互过程如图 1(a) 所示。

1.2 Comet 技术

目前，Comet 技术[1]的实现方式包括基于异步 JavaScript 和可扩展标记语言(Asynchronous JavaScript and Extensible Markup Language，AJAX) 的长轮询方式和基于 Iframe 的流方式。这两种方式针对轮询都做出了较大的改进。

1.2.1 基于 AJAX 的长轮询方式

基于 AJAX 的长轮询方式[2]通过采用 AJAX 技术让客户端向服务器端发送 HTTP 请求，进而与服务器端建立连接，且该连接会在服务器端保持一段时间。若服务器端检测到有新数据产生，那么它会将这些数据通过连接发送至客户端，然后关闭连接；若服务器端在连接存在期间都没有产生新的数据发送至客户端，那么它将会向客户端发送一个超时信息，然后关闭连接。无论服务器端的数据是否还在更新，在连接关闭之后，客户端都需要重新向服务器端发送 HTTP 请求来建立连接。其中客户端和服务器端的交互过程如图 1(b) 所示。

虽然这种方式能够对客户端的部分页面进行更新，减少服务器端发送的数据量，降低客户端请求的频率，减少无效的网络传输，但当服务器端更新数据的速度较快时，基于 AJAX 的长轮询方式将变成普通的轮询，不仅会降低其性能，而且还会对服务器端造成较大的处理压力。除此之外，为了保持 HTTP 连接长时间处于打开状态，服务器端也需要消耗一定的服务器资源。因此，使用基于 AJAX 的长轮询方式会产生资源浪费的问题。

1.2.2 基于 Iframe 的流方式

基于 Iframe 的流方式[3]通过客户端页面上内嵌的一个 Iframe 标签向服务器端发送 HTTP 请求，服务器端在响应该请求后与客户端建立一条长连接。连接建立后，服务器端通过不断地更新该连接的状态以保持其不过期。当服务器端检测到有新数据产生时，它会将新数据通过该连接发送给客户端；当客户端和服务器端之间的通信出现问题导致连接出现错误或者关闭时，客户端会立即发出连接请求与服务器端重新建立连接，否则该连接会一直持续，不会关闭。其中客户端和服务器端的交互过程如图 1(c) 所示。

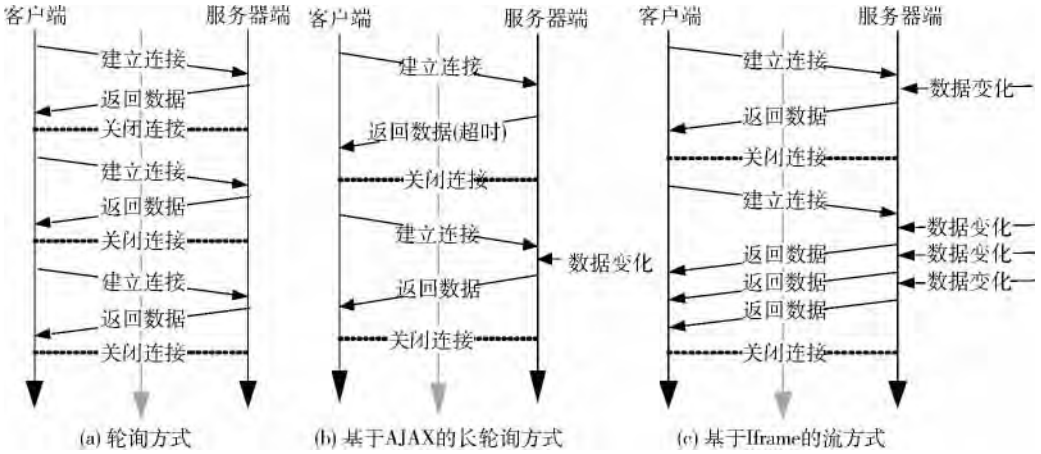

图 1 客户端与服务器端的交互图

虽然这种方式有利于减少客户端的请求次数，减轻客户端和服务器端之间的网络负担，避免因频繁的建立连接和关闭连接所带来的资源浪费，但由于基于 Iframe 的流方式在连接过程中始终只维持一个长连接，因此客户端页面会一直处于加载过程中而无法显示页面加载完成，从而影响用户体验。且当有多个客户端同时向服务器端发送 HTTP 请求时，由于服务器端长期只维持一个连接，因此会导致服务器端在这种高并发状态下的处理能力降低，造成大量的服务器资源和其他网络资源被消耗。

由于基于 AJAX 的长轮询方式和基于 Iframe 的流方式在通信过程中一直采用 HTTP 作为通信协议，因此每次的 HTTP 请求和应答所携带的完整的 HTTP 头信息不仅增加了实时更新信息时的数据传输量，还造成带宽资源的浪费。此外，为了维持和协调通信过程中 HTTP 连接随时处于可用状态，服务器端也需要消耗资源。对于 HTTP 连接的建立和关闭过程而言，服务器端新产生的数据有可能会因为无法及时发送到客户端而导致客户端的数据丢失。由于这两种方式对 Web 应用中的实时信息和非实时信息的请求/响应方式都未发生改变，因此，当实时信息的请求较为频繁时，可能会造成服务器端较大的处理压力，从而影响非实时信息的呈现。其中基于 HTTP 的 Web 实时应用模型如图 2 所示。

2 传统的 Web 实时通信方案

WebSocket 协议[4-5]是 HTML5 规范中的一种新的通信协议，是能够在客户端和服务器端进行异步通信的一种方法。它支持客户端与服务器端通过全双工通信的方式实

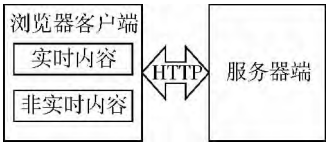

图2 基于 HTTP 的 Web 实时应用模型

现实时通信，本质上是一个基于传输控制协议的协议。因此，WebSocket 连接的建立过程与传输控制协议连接的建立过程有些相似，客户端和服务器端需要通过“握手”来建立 WebSocket 连接。

首先由客户端向服务器端发送一个 HTTP 请求，该请求不同于一般的 HTTP 请求，它包含了一些附加的 HTTP 头信息，其中一条信息“Upgrade: WebSocket”表明这是一个申请将当前 HTTP 协议升级为 WebSocket 协议的 HTTP 请求。若服务器端收到该请求后能正确解读其 HTTP 头信息，那么它会返回一个基于 HTTP 的应答报文给客户端，此时连接建立成功[6]，之后，客户端和服务器端便可以通过该连接主动向对方发送或者接收数据，直到其中一方主动关闭该连接。其中客户端和服务器端的交互过程如图 3 所示。

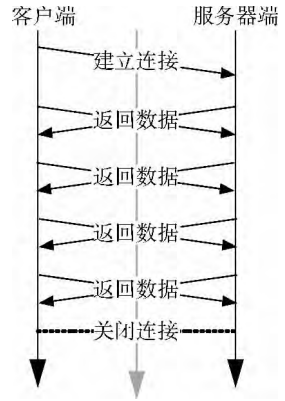

图3 基于 WebSocket 协议的交互图

通过 WebSocket 协议，客户端和服务器端之间只要做一个“握手”的动作就可以建立一条双向通信的通道。这不仅让服务器端可以主动与客户端互发信息，而且还避免了因客户端频繁请求而造成的网络资源浪费、实时通信效率低、服务器处理压力大等问题[7]。由于 WebSocket 连接采用 WebSocket 协议作为通信协议，因此在传输过程中数据帧的头部信息所占的字节数将大大降低，从而有效地减小了通信过程中传输的数据量和网络负载，节约了带宽资源。在基于 WebSocket 协议的实时通信方案中，Web 应用中的实时部分和非实时部分被加以区分。客户端使用 WebSocket 协议获取实时内容，使用 HTTP 协议获取非实时内容。而服务器端则采用两种不同的模块来处理实时的 WebSocket 请求和非实时的 HTTP 请求，其应用模型如图 4 所示。

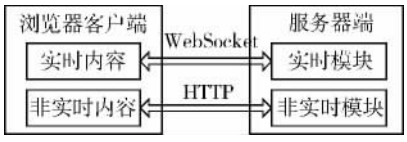

图4 基于 WebSocket 的 Web 实时应用模型

通过上述模型可以看出，该实时通信方案使服务器端的结构更加明确，不仅让 WebSocket 协议和 HTTP 协议各司其职、互不干扰，而且还降低了系统的耦合性，在最大程度上发挥了两个模块的功能。此外，由于采用以传输控制协议为基础的 WebSocket 协议来处理实时服务，因此可以保证传输数据过程中的稳定性和及时性，在较大程度上提高了实时通信的性能。相对于传统方案来说，该方案不仅减小了对服务器资源的浪费，也减轻了服务器端的处理压力。

3 基于 WebSocket 的 Web 实时通信应用实例

本文采用基于 Node. js[8] 的 Express 框架和 Socket. io 类库来实现基于 WebSocket 的 Web 实时通信应用。其中，Node. js 是一个 JavaScript 运行平台，可用于构建响应速度快、容易扩展的网络程序。但由于 Node. js 中只提供了大量的低端功能，因此文中将使用 Express 框架进行 Web 实时通信应用的开发。Express 是一个能够在 Node. js 中使用的 Web 应用程序开发框架，它提供的一系列强大的特性，能够让 Web 应用程序的开发变得更加方便、快速。

Socket. io 是一个开源、跨平台且支持客户端和服务器端进行实时双向通信的 WebSocket 库[9-10]。它包括客户端的 JavaScript 库和服务器端的 Node. js 模块。它能够根据不同的客户端自动在一些实时通信机制中选择合适的一个来实现 Web 实时应用。当使用支持 HTML5 技术的浏览器客户端进行实时通信时，Socket. io 会选译效率最高、消耗服务器资源最少的 WebSocket 协议来实现实时通信，并在浏览器客户端发生变化时自动选择其他方式进行通信。因此，Socket. io 能有效解决跨平台的实时通信问题。

3.1 在线聊天室的设计

在线聊天室的设计分为客户端与服务器端两个部分，其实时通信过程如图 5 所示。

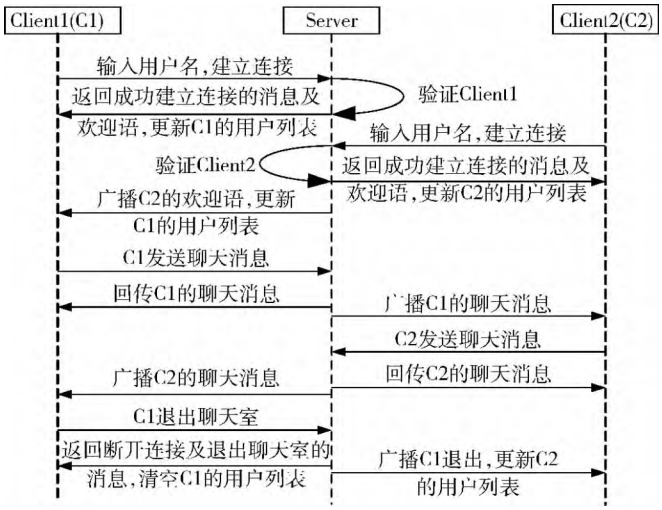

图5 客户端与服务器端的实时通信过程

3.2 在线聊天室的实现

在线聊天室的实现也分为客户端和服务器端两个部分。其中客户端通过使用 HTML5、层叠样式表以及 JavaScript 来实现用户名的验证功能、消息显示功能和数据传送功能。服务器端通过 JavaScript 来实现与客户端的实时通信功能、广播功能以及在线用户列表的管理功能。

3.2.1 客户端的实现过程

当有新的客户端用户加入聊天室时，已在聊天室的用户将会接收到新用户加入聊天室的消息且用户列表会被即时更新以显示新加入的用户名。新用户所在页面也会被更新以显示所有在线用户。当有客户端用户在聊天室发送聊天消息时，该消息会被即时广播给所有在线用户。当有客户端用户退出聊天室时，其他在线用户将会接收到该用户退出聊天室的消息且用户列表会被实时更新以移除下线用户的用户名。下线用户所在的页面也会进行相应的调整。若用户在聊天过程中直接退出聊天室页面，则所有在线用户都会收到该用户退出聊天室的消息。客户端的具体实现流程如图 6 所示。

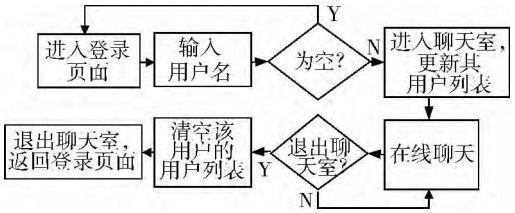

图 6 客户端流程图

3.2.2 服务器端的实现过程

当有多个客户端用户存在时，服务器端的主要功能包括管理所有在线用户的用户列表以及广播它们之间的聊天消息。服务器端的具体实现流程如图 7 所示。

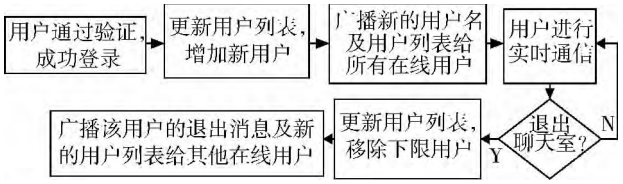

图 7 服务器端流程图

3.2.3 客户端和服务器端的交互过程

本文主要针对用户成功登录进聊天室的情况进行介绍。当用户成功登录在线聊天室时，客户端和服务器端通过触发事件进行实时交互，其具体交互过程如图 8 所示。

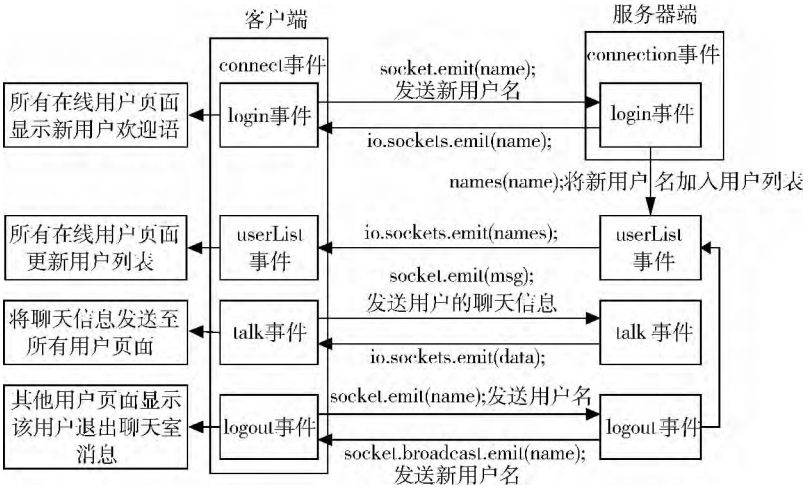

图 8 用户登录成功时客户端与服务器端的交互图

4 结论

传统的 Web 实时通信方案是在长期的应用实践中发展出来的，其中比较常用的是基于 AJAX 的长轮询方式和基于 Iframe 的流方式。但由于这两种方案都是采用基于 HTTP 的通信方式，因此当 Web 实时应用采用这两种方案时会产生难以解决的问题。而 WebSocket 协议的出现适时地提供了一种新的 Web 实时通信方案，它能够更加快捷有效地构建出简单高效的 Web 实时应用。因此，本文通过分析传统的 Web 实时通信方案的不足之处，不仅从理论层面分析了基于 WebSocket 的 Web 实时通信方案的优势，而且还通过使用 HTML5、层叠样式表和 JavaScript 编写了具体的应用实例简单的实现了该方案。随着 WebSocket 协议的不断发展，基于 WebSocket 的 Web 实时通信方案将会被广泛应用。

参考文献

[1] 蔡骥然，曹海传. B/S 架构下基于 OPC 与 Comet 技术的实时监控系统[J]. 计算机应用，2012，32(z2)：214-216.

[2] 文爱平，文德民. 基于 IE 浏览器的 Ajax Comet 架构[J]. 电脑知识与技术，2010，6(17)：4646-4648.

[3] 张家爱，孙飞. Comet 技术在 Web 开发中的研究与应用[J]. 煤炭技术，2011，30(12)：153-154.

[4] 陆晨，冯向阳，苏厚勤. HTML5 WebSocket 握手协议的研究与实现[J]. 计算机应用与软件，2015，32(1)：128-131，178.

[5] 李代立，陈榕. WebSocket 在 Web 实时通信领域的研究[J]. 电脑知识与技术，2010，6(28)：7923-7925，7935.

[6] 周东仿，孟宁. 基于 WebSocket 的网络设备自发现机制[J]. 计算机工程与设计，2013，34(2)：392-396，438.

[7] 温照松，易仁伟，姚寒冰. 基于 WebSocket 的实时 Web 应用解决方案[J]. 电脑知识与技术，2012，8(16)：3826-3828.

[8] 王金龙，宋斌，丁锐. Node. js：一种新的 Web 应用构建技术[J]. 现代电子技术，2015，38(6)：70-73.

[9] 李广文. 基于 Socket. io 的互动教学即时反馈系统的设计与实现[J]. 中国现代教育装备，2012(18)：10-12.

[10] 黄经赢. 基于 Socket. io + Node. js + Redis 构建高效即时通讯系统[J]. 现代计算机(专业版)，2014(19)：62-64，69.

(收稿日期：2016-01-28)

作者简介：

陈丽枫(1991 -)，女，硕士研究生，主要研究方向：Web 应用研究、光电信息检测与智能计算。

郑力新(1967 -)，男，博士，教授，主要研究方向：人工智能、工业自动化技术。

王佳斌(1974 -)，男，硕士，副教授，主要研究方向：嵌入式系统、物联网技术、云计算应用。