参考：<http://blog.csdn.net/jenminzhang/article/details/47017741>

### 题外话

前几天和朋友聊天，朋友问我怎么最近不写博客了，一个是因为最近在忙着公司使用的一些控件的开发，浏览器兼容性搞死人；但主要是因为这段时间一直在看**[HTML5](http://lib.csdn.net/base/html5" \t "_blank" \o "HTML5知识库)**的东西，看到web socket时觉得很有意思，动手写几个demo，但web socket需要特定的服务器支持，由于标准制定工作还没完成，所以没有多少主流的服务器支持，自己在网上下载了几个实现，包括**[PHP](http://lib.csdn.net/base/php" \o "PHP知识库" \t "_blank)**的、C#的、甚至**[Node.js](http://lib.csdn.net/base/nodejs" \o "Node.js知识库" \t "_blank)**的，但一个是协议变化比较大，很多代码已经过时了，再就是有一些支持最新的标准，但是我想稍微改造一下，看人家源代码的时候云里雾里，看看别人的代码行数也不多，决定自己实现一个。

悲剧由此开始，虽然哥们儿国内非知名工科大学毕业，但好歹也是科班CS出身，但大学得过且过，什么TCP/IP协议，什么socket了都没概念。为了做出一个简单的支持广播的websocket server，在网上找了很多相关代码，左抄一句，右抄一句，弄了一个星期竟然还是漏洞百出，调试不起来，只好从头来过了，先补一些基本知识，然后再一步步根据原理实现，今天终于实现了绝大部分功能，由此真的感受到了，搞计算机必须得有理论指导实践，否则只能像个没头苍蝇到处乱撞。

### TCP/IP

要想理解socket首先得熟悉一下TCP/IP协议族， TCP/IP（Transmission Control Protocol/Internet Protocol）即传输控制协议/网间协议（TCP指的就是传输层，ip指的就是网络层），定义了主机如何连入因特网及数据如何在它们之间传输的标准，

从字面意思来看TCP/IP是TCP和IP协议的合称，但实际上TCP/IP协议是指因特网整个TCP/IP协议族。不同于ISO模型的七个分层，TCP/IP协议参考模型把所有的TCP/IP系列协议归类到四个抽象层中：数 网 传 应《-》物 数 网 传 会 表 应

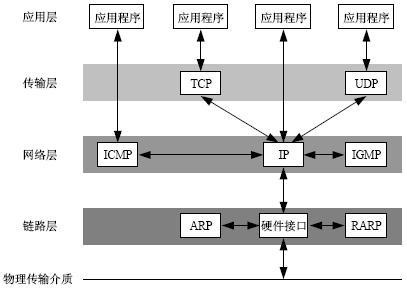
应用层：TFTP，HTTP，SNMP，FTP，SMTP，DNS，Telnet 等等

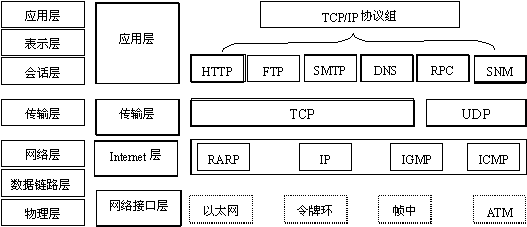
传输层：TCP，UDP

网络层：IP，ICMP，OSPF，EIGRP，IGMP

数据链路层：SLIP，CSLIP，PPP，MTU

每一抽象层建立在低一层提供的服务上（类似docker构建镜像的层叠），并且为高一层提供服务，看起来大概是这样子的



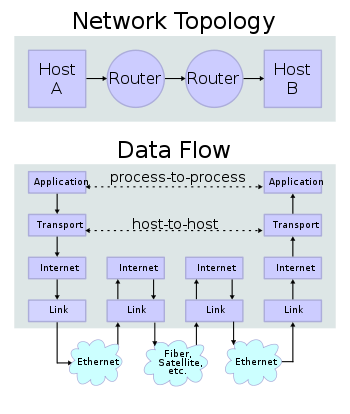


估计有兴趣打开此文的同学都对此有一定了解了，加上我也是一知半解，所以就不详细解释，有兴趣同学可以上网上搜一下资料

[维基百科](http://zh.wikipedia.org/wiki/TCP/IP)

[百度百科](http://baike.baidu.com/link?url=KJleEWJUBxe3XaxEaTU1lcQrg9qYN7FYzuPhoWVyoPFNk79NIURxAO6HSWC1MfrCU1Dh7QRsNDWOaZ-4bg5HRa)

在TCP/IP协议中两个因特网主机通过两个路由器和对应的层连接。各主机上的应用通过一些数据通道相互执行读取操作



### socket

我们知道两个进程如果需要进行通讯最基本的一个前提能能够唯一的标示一个进程，在本地进程通讯中我们可以使用PID来唯一标示一个进程，但PID只在本地唯一，网络中的两个进程PID冲突几率很大，这时候我们需要另辟它径了，我们知道IP层的ip地址可以唯一标示主机，而TCP层协议和端口号可以唯一标示主机的一个进程，这样我们可以利用ip地址＋协议＋端口号唯一标示网络中的一个进程。

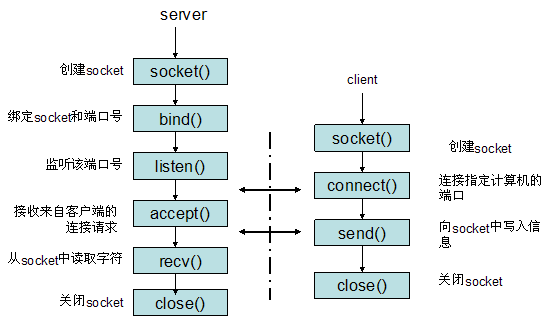
能够唯一标示网络中的进程后，它们就可以利用socket进行通信了，什么是socket呢？我们经常把socket翻译为套接字，socket是在应用层和传输层（如TCP和UDP）之间的一个抽象层，是一个接口，它把TCP/IP层复杂的操作抽象为几个简单的接口供应用层调用已实现进程在网络中通信。



socket起源于UNIX，在Unix一切皆文件哲学的思想下，socket是一种"打开—读/写—关闭"模式的实现，服务器和客户端各自维护一个"文件"，在建立连接打开后，可以向自己文件写入内容供对方读取或者读取对方内容，通讯结束时关闭文件。

### socket通信流程

socket是"打开—读/写—关闭"模式（对比HTTP的请求-响应模式）的实现，以使用TCP协议通讯的socket为例，其交互流程大概是这样子的



服务器根据地址类型（ipv4,ipv6）、socket类型（见下方解释）、协议创建socket

服务器为socket绑定ip地址和端口号

服务器socket监听端口号请求，随时准备接收客户端发来的连接，这时候服务器的socket并没有被打开

客户端创建socket

客户端打开socket，根据服务器ip地址和端口号试图连接服务器socket

服务器socket接收到客户端socket请求，被动打开，开始接收客户端请求，直到客户端返回连接信息。这时候socket进入**阻塞**状态，所谓阻塞即accept()方法一直到客户端返回连接信息后才返回，开始接收下一个客户端连接请求

客户端连接成功，向服务器发送连接状态信息

服务器accept方法返回，连接成功

客户端向socket写入信息

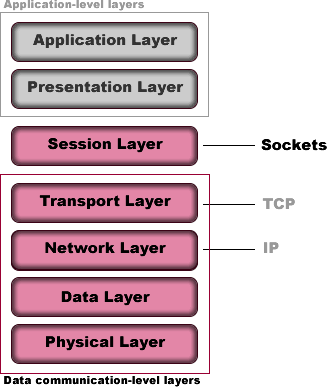
服务器读取信息

客户端关闭（socket是长连接，如果客户端或者服务端不主动关闭则会一直保持连接而不会关闭）

服务器端关闭

socket类型：

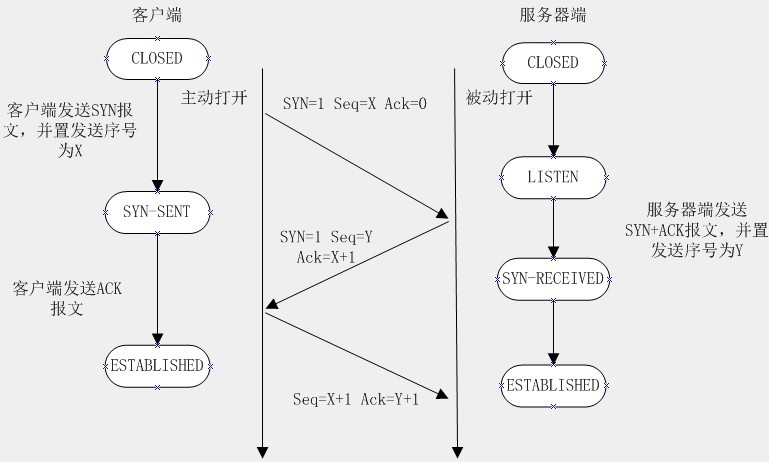
Socket是一组编程接口（API，面向应用层）, 是对TCP/IP协议（传输层和网络层）的封装和应用。介于传输层和应用层,大致驻留在 OSI 模型（也协作iso模型）的会话层（物 数 网 传 会 表 应对应TCP/IP系列协议 数 网 传 应），向应用层提供统一的编程接口。应用层不必了解TCP/IP协议细节。直接通过对Socket接口函数的调用完成数据在IP网络的传输。



**基于传输层差异，4种类型的Socket:**  
    (1)基于TCP的Socket:提供给应用层可靠的流式数据服务，使用TCP的Socket应用程序协议：BGP，HTTP，FTP，TELNET等。优点：基于数据传输的可靠性。  
   (2)基于UDP的Socket：适用于数据传输可靠性要求不高的场合。基于UDP的Socket应用程序或协议有：RIP，SNMP，L2TP等。  
  (3)基于RawIp的Socket:非连接，不可靠的数据传输。特点：能使应用程序直接访问网络层。基于RawIp的Socket有ping ,tracert,ospf等。  
  (4)基于链路层的Socket。为IS-IS协议提供的Socket接口。使IS-IS协议可通过Socket直接访问链路层。非连接，不可靠通信服务。

### 三次握手

在TCP/IP协议中，TCP协议通过三次握手建立一个可靠的连接

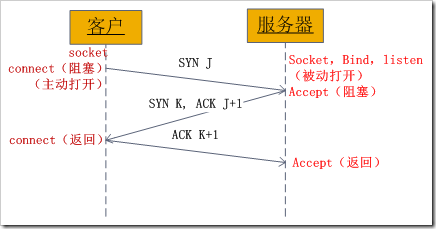


第一次握手：客户端尝试连接服务器，向服务器发送syn包（同步序列编号***Synchronize Sequence Numbers***），syn=j，客户端进入SYN\_SEND状态等待服务器确认

第二次握手：服务器接收客户端syn包并确认（ack=j+1），同时向客户端发送一个SYN包（syn=k），即SYN+ACK包，此时服务器进入SYN\_RECV状态

第三次握手：第三次握手：客户端收到服务器的SYN+ACK包，向服务器发送确认包ACK（Acknowledgement）(ack=k+1），此包发送完毕，客户端和服务器进入ESTABLISHED状态，完成三次握手

定睛一看，服务器socket与客户端socket建立连接的部分其实就是大名鼎鼎的三次握手



**socket编程API**

前面提到socket是"打开—读/写—关闭"模式的实现，简单了解一下socket提供了哪些API供应用程序使用，还是以TCP协议为例，看看Unix下的socket API，其它语言都很类似（PHP甚至名字都几乎一样），这里我就简单解释一下方法作用和参数，具体使用有兴趣同学可以看看博客参考中的链接或者上网搜索

1、创建socket

int socket(int domain, int type, int protocol);//套接字的一部分，host+socket类型+协议类型

根据指定的地址族、socket类型和协议来分配一个socket的描述字及其所用的资源。

domain:协议族，常用的有AF\_INET、AF\_INET6、AF\_LOCAL、AF\_ROUTE其中AF\_INET代表使用ipv4地址

type:socket类型，常用的socket类型有，SOCK\_STREAM、SOCK\_DGRAM、SOCK\_RAW、SOCK\_PACKET、SOCK\_SEQPACKET等

protocol:协议。常用的协议有，IPPROTO\_TCP、IPPTOTO\_UDP、IPPROTO\_SCTP、IPPROTO\_TIPC等

2、绑定地址和端口

int bind(int sockfd, const struct sockaddr \*addr, socklen\_t addrlen);

把一个地址族中的特定地址赋给socket

sockfd:socket描述字，也就是socket引用

addr:要绑定给sockfd的协议地址

addrlen:地址的长度

通常服务器在启动的时候都会绑定一个众所周知的地址（如ip地址+端口号），用于提供服务，客户就可以通过它来接连服务器；而客户端就不用指定，有系统自动分配一个端口号和自身的ip地址组合。这就是为什么通常服务器端在listen之前会调用bind()，而客户端就不会调用，而是在connect()时由系统随机生成一个（随机在客户端生成一个连接端口供客户端连接服务器使用，所以客户端的socket每次connect的端口是不固定的）。

3、监听端口

int listen(int sockfd, int backlog);

监听socket

sockfd:要监听的socket描述字

backlog:相应socket可以排队的最大连接个数 （socket连接可能导致阻塞）

4、连接

int connect(int sockfd, const struct sockaddr \*addr, socklen\_t addrlen);

连接某个socket

sockfd:客户端的socket描述字

addr:服务器的socket地址

addrlen:socket地址的长度

4、接受请求

int accept(int sockfd, struct sockaddr \*addr, socklen\_t \*addrlen);

 TCP服务器监听到客户端请求之后，调用accept()函数取接收请求

sockfd:服务器的socket描述字

addr:客户端的socket地址

addrlen:socket地址的长度

5、获取数据

ssize\_t read(int fd, void \*buf, size\_t count);

读取socket内容

fd:socket描述字

buf：缓冲区

count：缓冲区长度

6、写数据

ssize\_t write(int fd, const void \*buf, size\_t count);

向socket写入内容，其实就是发送内容

fd:socket描述字

buf：缓冲区

count：缓冲区长度

7、关闭socket连接

int close(int fd);

socket标记为以关闭 ，使相应socket描述字的引用计数-1（采用引用计数来关闭连接。php中对变量也是引用计数），当引用计数为0的时候，触发TCP客户端向服务器发送终止连接请求。

Server端：

create socket

bind

listen

accept

Client端：

create

connect

send

### 参考

[Linux Socket编程（不限Linux）](http://www.cnblogs.com/skynet/archive/2010/12/12/1903949.html)

[揭开Socket编程的面纱](http://goodcandle.cnblogs.com/archive/2005/12/10/294652.aspx)

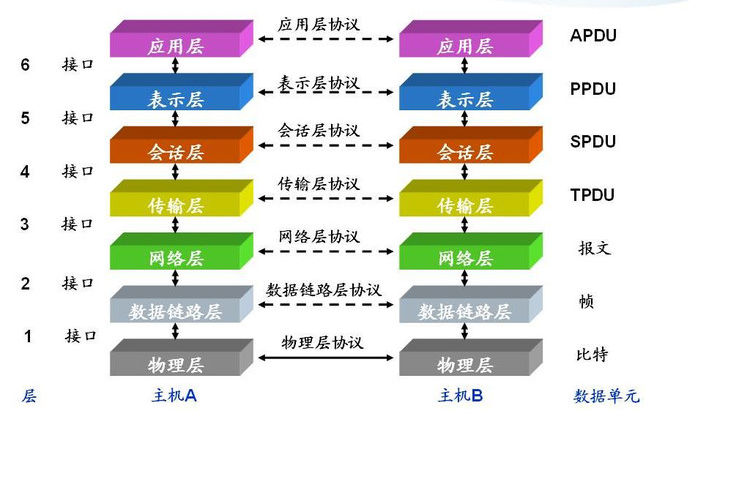
PS. 有同学看完后发现没有demo示例，参考中的示例已经很不错了，我就不班门弄斧了，而且我用C#实现了一个websocket server，接下来的博客中会有介绍。另外由于刚刚实际接触socket，文中谬误较多，还望大家批评指正，文章内容主要参考上面两个博文，图片全部来源于网络，在百度图片搜索得来，无法注明第一源地址，如有版权问题请站内信联系，第一时间处理。

原文地址为；http://www.cnblogs.com/dolphinX/p/3460545.html

参考：<http://www.cnblogs.com/meier1205/p/5971313.html>

[HTTP 和 Socket 的区别](http://www.cnblogs.com/meier1205/p/5971313.html)

要弄明白 http 和 socket 首先要熟悉网络七层：物 数 网 传 会 表 应，如图1（发送传输的时候一层层的加（类似docker构建镜像加层），获取时一层层减）



如图1

HTTP 协议:超文本传输协议，对应于应用层，用于如何封装数据.

TCP/UDP 协议:传输控制协议，对应于传输层，主要解决数据在网络中的传输。

IP 协议:对应于网络层，同样解决数据在网络中的传输。

传输数据的时候只使用 TCP/IP 协议(传输层TCP-传输层，IP-网络层)，如果没有应用层来识别数据内容，传输后的协议都是无用的。

应用层协议很多 FTP,HTTP,TELNET等，可以自己定义应用层协议。

web 使用 HTTP 作传输层协议，以封装 HTTP 文本信息，然后使用 TCP/IP 做传输层协议，将数据发送到网络上。

**一、HTTP 协议**

**http 为短连接：**客户端发送请求都需要服务器端回送响应.请求结束后，主动释放链接，因此为短连接。通常的做法是，不需要任何数据，也要保持每隔一段时间向服务器发送"保持连接"的请求。这样可以保证客户端在服务器端是"上线"状态。

HTTP连接使用的是"请求-响应"方式，不仅在请求时建立连接，而且客户端向服务器端请求后，服务器才返回数据。

**二、Socket 连接**

要想明白 Socket，必须要理解 TCP 连接。

TCP 三次握手：握手过程中并不传输数据，在握手后服务器与客户端才开始传输数据，理想状态下，TCP 连接一旦建立，在通讯双方中的任何一方主动断开连接之前 TCP 连接会一直保持下去。

Socket 是对 TCP/IP 协议的封装，Socket 只是个接口不是协议，通过 Socket 我们才能使用 TCP/IP 协议，除了 TCP，也可以使用 UDP 协议来传递数据。

创建 Socket 连接的时候，可以指定传输层协议，可以是 TCP 或者 UDP，当用 TCP 连接，该Socket就是个TCP连接，反之。

***Socket 原理***

*Socket 连接,至少需要一对套接字，分为 clientSocket，serverSocket 连接分为3个步骤:*

*(1) 服务器监听:服务器并不定位具体客户端的套接字，而是时刻处于监听状态；*

*(2) 客户端请求:客户端的套接字要描述它要连接的服务器的套接字（如host:192.168.xxx.xxx,port:xxxx），提供地址和端口号，然后向服务器套接字提出连接请求；*

*(3) 连接确认:当服务器套接字收到客户端套接字发来的请求后，就响应客户端套接字的请求,并建立一个新的线程,把服务器端的套接字的描述发给客户端。一旦客户端确认了此描述，就正式建立连接。而服务器套接字继续处于监听状态，继续接收其他客户端套接字的连接请求.*

传递客户端的套接字

clinet server

传递服务端的套接字

发送确认包建立连接

**Socket为长连接（对比http的短连接）：**通常情况下Socket 连接就是 TCP 连接，因此 Socket 连接一旦建立,通讯双方开始互发数据内容，直到双方断开连接。在实际应用中，由于网络节点过多，在传输过程中，会被节点断开连接，因此要通过轮询高速网络，该节点处于活跃状态。

很多情况下，都是需要服务器端向客户端主动推送数据，保持客户端与服务端的实时同步。

若双方是 Socket 连接，可以由服务器直接向客户端发送数据。

若双方是 HTTP 连接，则服务器需要等客户端发送请求后，才能将数据回传给客户端（服务端发送完数据节关闭连接）。

因此，客户端定时向服务器端发送请求，不仅可以保持在线，同时也询问服务器是否有新数据，如果有就将数据传给客户端。

参考原文：http://www.cnblogs.com/xclidongbo/p/4032904.html?utm\_source=tuicool&utm\_medium=referral

参考：<http://blog.csdn.net/zeng622peng/article/details/5546384>

相信不少初学手机联网开发的朋友都想知道Http与Socket连接究竟有什么区别，希望通过自己的浅显理解能对初学者有所帮助。

**1、TCP连接**

 手机能够使用联网功能是因为手机底层实现了TCP/IP协议，可以使手机终端通过无线网络(ip层)建立TCP连接（tcp传输）。TCP协议可以对上层网络提供接口，使上层网络数据的传输建立在“无差别”的网络之上。（tcp是传输层，ip是网络层；网络七层=>物 数 网 传 会 表 应）

建立起一个TCP连接需要经过“三次握手”：

第一次握手：客户端发送syn包(syn=j)到服务器，并进入SYN\_SEND状态，等待服务器确认；

第二次握手：服务器收到syn包，必须确认客户的SYN（ack=j+1），同时自己也发送一个SYN包（syn=k），即SYN+ACK包，此时服务器进入SYN\_RECV状态；

第三次握手：客户端收到服务器的SYN＋ACK包，向服务器发送确认包ACK(ack=k+1)，此包发送完毕，客户端和服务器进入ESTABLISHED状态，完成三次握手。

这里的j和k仅仅是为了确认的一种验证规范（如这个k+1对应k，j+1对应j），类似：

client：你在吗？

server:我在，你还在吗？

client:我还在，我们开始连接吧！

握手过程中传送的包里不包含数据，三次握手完毕后，客户端与服务器才正式开始传送数据。

理想状态下，TCP连接一旦建立，在通信双方中的任何一方主动关闭连接之前，TCP 连接都将被一直保持（socket是tcp连接，是长连接）下去。断开连接时服务器和客户端均可以主动发起断开TCP连接的请求，断开过程需要经过“四次握手”（过程就不细写了，就是服务器和客户端交互，最终确定断开）

2、HTTP连接

HTTP协议即超文本传送协议(Hypertext Transfer Protocol )，是Web联网的基础，也是手机联网常用的协议之一，HTTP协议是建立在TCP协议之上的一种应用。

HTTP连接最显著的特点是客户端发送的每次请求都需要服务器回送响应，在请求结束后，会主动释放连接（请求-响应模式）。从建立连接到关闭连接的过程称为“一次连接”（也称为短连接）。

1）在HTTP 1.0中，客户端的每次请求都要求建立一次单独的连接，在处理完本次请求后，就自动释放连接。  
   
2）在HTTP 1.1中则可以在一次连接中处理多个请求，并且多个请求可以重叠进行，不需要等待一个请求结束后再发送下一个请求。  
   
由于HTTP在每次请求结束后都会主动释放连接，因此HTTP连接是一种“短连接”，要保持客户端程序的在线状态，需要不断地向服务器发起连接请求。通常的做法是即使不需要获得任何数据，客户端也保持每隔一段固定的时间向服务器发送一次“保持连接”的请求，服务器在收到该请求后对客户端进行回复，表明知道客户端“在线”（如redis的master和slave的ping-pong以及mysql的heart beat）。若服务器长时间无法收到客户端的请求，则认为客户端“下线”，若客户端长时间无法收到服务器的回复，则认为网络已经断开。

3、SOCKET原理

3.1套接字（socket）概念

套接字（socket）是通信的基石，是支持TCP/IP协议的网络通信的基本操作单元。它是网络通信过程中端点（Endpoint）的抽象表示，包含进行网络通信必须的五种信息：连接使用的协议，本地主机的IP地址，本地进程的协议端口，远地主机的IP地址，远地进程的协议端口。

协议和两套ip即端口

应用层通过传输层（如Tcp）进行数据通信时，TCP会遇到同时为多个应用程序进程提供并发服务的问题。多个TCP连接或多个应用程序进程可能需要通过同一个 TCP协议端口传输数据。为了区别不同的应用程序进程和连接，许多计算机[操作系统](http://lib.csdn.net/base/operatingsystem" \o "操作系统知识库" \t "_blank)为应用程序与TCP／IP协议交互提供了套接字(Socket)接口。应用层可以和传输层通过Socket接口，区分来自不同应用程序进程或网络连接的通信，实现数据传输的并发服务。

3.2 建立socket连接

建立Socket连接至少需要一对套接字，其中一个运行于客户端，称为ClientSocket ，另一个运行于服务器端，称为ServerSocket 。

 套接字之间的连接过程分为三个步骤：服务器监听，客户端请求，连接确认。

服务器监听：服务器端套接字并不定位具体的客户端套接字，而是处于等待连接的状态，实时监控网络状态，等待客户端的连接请求。

客户端请求：指客户端的套接字提出连接请求，要连接的目标是服务器端的套接字。为此，客户端的套接字必须首先描述它要连接的服务器的套接字，指出服务器端套接字的地址和端口号，然后就向服务器端套接字提出连接请求。

连接确认：当服务器端套接字监听到或者说接收到客户端套接字的连接请求时，就响应客户端套接字的请求，建立一个新的线程，把服务器端套接字的描述发给客户端，一旦客户端确认了此描述，双方就正式建立连接。而服务器端套接字继续处于监听状态，继续接收其他客户端套接字的连接请求。

4、SOCKET连接与TCP连接

创建Socket连接时，可以指定使用的传输层协议，Socket可以支持不同的传输层协议（TCP或UDP），当使用TCP协议进行连接时，该Socket连接就是一个TCP连接。

5、Socket连接与HTTP连接

由于通常情况下Socket连接就是TCP连接，因此Socket连接一旦建立，通信双方即可开始相互发送数据内容，直到双方连接断开。但在实际网络应用中，客户端到服务器之间的通信往往需要穿越多个中间节点，例如路由器、网关、防火墙等，大部分防火墙默认会关闭长时间处于非活跃状态的连接而导致 Socket 连接断连，因此需要通过轮询告诉网络，该连接处于活跃状态（socket是长连接，其他的中间节点可能会关闭不活跃的socket连接促使需要轮询使socket处于活跃状态）。

而HTTP连接使用的是“请求—响应”的方式，不仅在请求时需要先建立连接，而且需要客户端向服务器发出请求后，服务器端才能回复数据（每次请求结束后都会释放连接）。

很多情况下，需要服务器端主动向客户端推送数据，保持客户端与服务器数据的实时与同步。此时若双方建立的是Socket连接，服务器就可以直接将数据传送给客户端；若双方建立的是HTTP连接，则服务器需要等到客户端发送一次请求后才能将数据传回给客户端，因此，客户端定时向服务器端发送连接请求（轮询），不仅可以保持在线，同时也是在“询问”服务器是否有新的数据，如果有就将数据传给客户端。