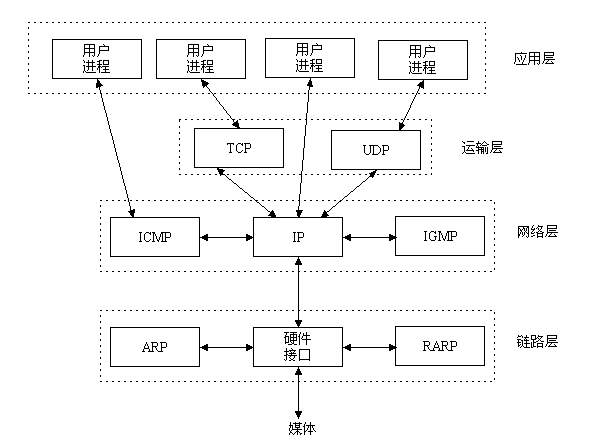
2015/5/19

Basic conception of Socket programming:

Socket is the API for TCP/IP network, which defines functions and demonstrations. Programmer can design the applications on TCP/IP network through socket .



The diagrams above describes the relations between protocols.[[1]](#footnote-1)



**Why socket is chosen for network connection?**

A TCP Socket is an endpoint instance defined by an IP address and a port in the context of either a particular TCP connection or the listening state ,can be described like telephone socket.

Socket users only care about local point and terminal point, how connection is built and how communication is accomplished will not be taken into consideration.

A complete socket holds a unique local socket number distributed by operate system.

Currently socket API is the most widely utilized method for internet visiting.[[2]](#footnote-2)

2015/5/21

This is gonna be a convenient way to synchronize notes on different laptops.

What’s the difference between socket and web server?

Usually web server is application,works on application level.Port ,packet ,protocol have already been defined by the application. Socket works on network level ,for which Port ,packet ,protocol need to be designed by user.

Socket Programming VS. Web service?[[3]](#footnote-3)

Webservices are generally speaking "easier" to do, thanks to the tremendous interest in them and the support for them in developer tools and through libraries and frameworks.

However, especially if your payload is small (think messages the size of a typical SMS or tweet), the overhead you create with webservices is prohibitive: bytes sent over a wireless network like GPRS or UMTS are still very expensive, compared to bytes carried over cable or ADSL. And web services carry several layers of invisible info around that the end customer will also have to pay.

So, if your use case is based on short messages, I'd at least advise to do some bandwidth simulation calculations, and base your decision on bandwidth savings vs increased complexity of your app.

While looking at sockets, also have a look at UDP: if you can live with the fact that basically you throw someone a packet, and without designing some ack mechanism into your protocol you'll never be sure the message arrived, it's very efficient because there is no traffic to create and maintain a connection, and even pretty long messages can very well be transported inside 1 UDP packet.

EDIT based on comment:

• stream socket: not sure how you define streams, but streams and messages are two very distinct concepts for me, a stream is a typically longer sequence of data being sent, whereas a message is an entity that's sent, and optionally acknowledged or answered by the receiver.

• bandwidth simulation: the easiest way to understand what I'm talking about is to get [Wireshark](http://www.wireshark.org/) and add up everything that gets transported across the net to do a simple request of a very short string - you'll see several layers of administrative info (ie invisible, just there to make the different protocol layers work) that are all traffic paid for by the end user. Then, write a small mock service using UDP to transport the same message, or use a tool like [netcat](http://netcat.sourceforge.net/), good tutorial [here](http://www.adamsinfo.com/netcat-tutorial-for-linux-windows-howto-nc/), and add up the bytes that get transported. You'll see pretty huge differences in the number of bytes that are exchanged.

EDIT2, something I forgot to mention: mobile networks used to be open, transparent networks with devices identified by public IP addresses. There's a rapid evolution towards NATed mobile networks which has its impact on how devices inside and outside this "walled garden" can communicate ([NAT traversal](http://en.wikipedia.org/wiki/NAT_traversal)). You'll need to take this into account when designing your communication channel.

As for the use of streams for a chat application: it offers some conceptual advantages, but you can very well layer a chat app on top of UDP, look [here](http://www.codeproject.com/KB/IP/ChatAppAsynchUDPSocks.aspx) or [here](https://decibel.ni.com/content/docs/DOC-16654)

Sending over Socket VS. Http request

A socket is an end of communication channel(e.g between client and server).

The request going through the socket may be of many types depending on the protocol that is implemented(TCP/IP,HTTP…..).

The basic difference is that TCP/IP is protocol of lower layer=TCP(transport=4) and IP (network=3)than HTTP which is layer 6(presentation).

2015/5/22

Review of the 7 Layers of the OSI Model

OSI(Open system interconnection)model defines a networking framework to implement protocols in 7 layers. Control is passed from one layer to the next, starting at the application layer, proceeding to the buttom layer, over the channel to the next station and back up the hierarchy.OSI is nothing more but a conceptual framework so we can better understand complex interactions that are happening.

**So the relation between Web service and Socket is clear.**

**Confliction of server tools working on server**

**Existing RTP libraries, like JRTPLib provides an easy implementation on the application layer.**

**And socket works on abstract layer, connects application layer(=layer 7) and transports layer(=layer 4).**

**So how could I configure JRTPlib? so that I apply socket programming among it.**

2015/5/28

Yesterday(5.27)received introduction of robot design form Florian,which is great helpful.

Interface for robot control:

|  |  |  |
| --- | --- | --- |
| **Byte** | **Alias** | **Description** |
| **0** | VT\_ID | Consecutive ID for every element in the list |
| **1** | VT\_TYPE | Type of object |
| **2,3** | VT\_X1 | Parameter 1 (Function Depending on VT\_TYPE) |
| **4,5** | VT\_Y1 | Parameter 2 (Function Depending on VT\_TYPE) |
| **6,7** | VT\_X2 | Parameter 3 (Function Depending on VT\_TYPE) |
| **8,9** | VT\_Y2 | Parameter 4 (Function Depending on VT\_TYPE) |

C++ Programming in Linux:

GCC(GNU Complier Collection)

Will be discuss later.

6/2/2015

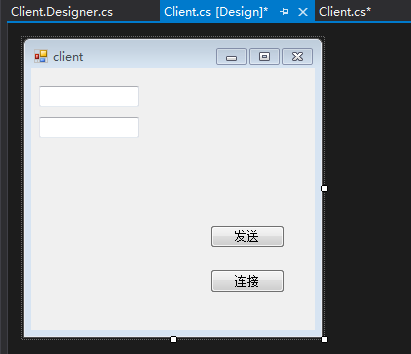
Client will be design by c#.

Advance:

.Net.Socket has been perfectly encapsulated.

Form design is easily implemented.

Windows form ”client ”is initially designed as follows:



Note about socket programming on .Net platform.

* IPEndPoint ipe =new IPEndPoint(ip,port);

//Initialize a instance with port and ip address

* Socket socket = new Socket(parameters…);

socket.Connect(ipe);

//connect socket to the server

* socket.send(‘data as bytes’,’length’,flag)

//send data

* socket.receive(‘data as bytes’,’length’,flag)

//receive data,here “date as bytes ” is to hold the received data

//all the data to send and receive must be format of ‘byte[]’

2015/6/5

Basic windows client form has been set.

Today I overview the webcam for Linux(Raspberrypei) in Lab.

<https://help.ubuntu.com/community/Webcam> is a introduction for how to choose and configure a webcam for Linux.

“Chesse”:A GNOME program for capturing video, therefore can be used to test webcams.

1. Install chesse: sudo apt-get install chesse
2. Run chesse, automatically detects webcam and display live video stream.

d

1. 源码工作室.揭开Socket编程的面纱

   <http://www.cnblogs.com/goodcandle/archive/2005/12/10/socket.html> ,19/05/2015 [↑](#footnote-ref-1)
2. Baidu.Zhidao socket编程是什么

   http://zhidao.baidu.com/question/329094437.html [↑](#footnote-ref-2)
3. <http://stackoverflow.com/questions/7237459/socket-programing-vs-web-service> [↑](#footnote-ref-3)