Today

- perl
- winsock
- java

perl sockets

- interpreted language
- multi OS
- CGI scripts and such
- conventional BSD socket
- perl5.004, object-oriented sockets

TCP client

```
#!/usr/local/bin/perl5.004
    use Socket;
    my ($remote,$port, $iaddr, $paddr, $proto, $line);

    $remote  = shift || 'localhost';
    $port    = shift || 2345;  # random port
    if ($port =~ /\D/) { $port = getservbyname($port, 'tcp') }
    die "No port" unless $port;
    $iaddr   = inet_aton($remote)               || die "no host: $remote";
    $paddr   = sockaddr_in($port, $iaddr);

    $proto   = getprotobyname('tcp');
    socket(SOCK, PF_INET, SOCK_STREAM, $proto)  || die "socket: $!";
    connect(SOCK, $paddr)    || die "connect: $!";
    while (defined($line = <SOCK>)) {
        print $line;
    }

    close (SOCK)            || die "close: $!";
    exit;
```


TCP server

```
#!/usr/local/bin/perl5.004
    use strict;
    use Socket;
    use Carp;

    sub logmsg { print "$0 $$: @_ at ", scalar localtime, "\n" }

    my $port = shift || 2345;
    my $proto = getprotobyname('tcp');
    $port = $1 if $port =~ /(\d+)/; # untaint port number

    socket(Server, PF_INET, SOCK_STREAM, $proto)        || die "sock
    setsockopt(Server, SOL_SOCKET, SO_REUSEADDR,
                                         pack("l", 1))   || die "sets
    bind(Server, sockaddr_in($port, INADDR_ANY))        || die "bind
    listen(Server,SOMAXCONN)                            || die "list

    logmsg "server started on port $port";
```

```
my $paddr;
for ( ; $paddr = accept(Client,Server); close Client) {
    my($port,$iaddr) = sockaddr_in($paddr);
    my $name = gethostbyaddr($iaddr,AF_INET);

    logmsg "connection from $name [",
            inet_ntoa($iaddr), "]
            at port $port";

    print Client "Hello there, $name, it's now ",
                    scalar localtime, "\n";
}
```


IO module

```
use IO::Socket;
$remote = IO::Socket::INET->new(
                    Proto    => "tcp",
                    PeerAddr => "localhost",
                    PeerPort => "daytime(13)",
                  )
                or die "cannot connect to daytime port at localhos
while ( <$remote> ) { print }
```


windows sockets

- console C (blocking)
- event driven C
- C++ (MFC classes)

console C

- borland or visual C++
- nmake from console
- run from console

generic.mak

for single source module

```
CC=cl
CFLAGS=-c -DSTRICT -DWIN32 -D_CONSOLE -O2  -Zp

LINKER=link
GUIFLAGS=-SUBSYSTEM:console

LIBS=kernel32.lib wsock32.lib
$(APP).exe : $(APP).obj
     $(LINKER) $(GUIFLAGS) -OUT:$(APP).exe $(APP).obj $(LIBS)
```

plus i have a little bld.bat

```
@ echo off
set app=%1
nmake generic.mak
```


C mods

- don't need all those includes, just windows.h and winsock.h
- initialize WinSock DLL and cleanup when done WSAStartup()/WSACleanup()
- various winsock DLL's provide different services/implementations
- WSAGetLastError() to get "errno"
- replace read/write with recv/send

tcp server

```
#include <stdio.h>
#include <stdlib.h>
#include <windows.h>
#include<winsock.h>
main(argc,argv)
int argc;
char *argv[];
{
           sockinit();
           tcprecv();
           Cleanup();
}
```



```
sockinit()
{
        WORD wVersionRequested;
        WSADATA wsaData;
        int err;

        wVersionRequested = MAKEWORD( 2, 0 );

        err = WSAStartup( wVersionRequested, &wsaData );
        if ( err != 0 ) {
                /* Tell the user that we couldn't find a usable */
                /* WinSock DLL.                                  */
                printf("startup failed %d\n",err);
                return;
        }
}
```

```
tcprecv()
{
        int lth;
        SOCKET lsd, sd;
        SOCKADDR_IN in,fr;

        lsd =socket(AF_INET,SOCK_STREAM,0);
        in.sin_family = PF_INET;
        in.sin_port = htons(port);
        in.sin_addr.s_addr = htonl(INADDR_ANY);
        bind(lsd,&in,sizeof(in));
        lth = sizeof(fr);
        listen(s,5);
        sd = accept(lsd, &fr, &lth);
        while(lth > 0){
                lth=recv(sd,buff,MAXL,0);
                send(sd,buff,lth,0);
        }
        shutdown(lsd,2);
        closesocket(lsd);
        shutdown(sd,2);
        closesocket(sd);
 }
```


TCP client

```
...
        s =socket(AF_INET,SOCK_STREAM,0);
        in.sin_family = PF_INET;
        in.sin_port = htons(port);
        in.sin_addr.s_addr = GetAddr(dst);
        err=connect(s,(LPSOCKADDR)&in,sizeof(in));
        if (err<0)
         {
           printf("connect err %d host %s\n",WSAGetLastError(),dst);
           return;
         }
...
        send(s,buff,rlth,0);
        rlth = mread(s,buff,lth);  /* recv til lth*/
...
        shutdown(s,2);
        closesocket(s);
```



```
GetAddr (LPSTR szHost) {
  LPHOSTENT lpstHost;
  u_long lAddr = INADDR_ANY;

  /* check that we have a string */
  if (*szHost) {

    /* check for a dotted-IP address string */
    lAddr = inet_addr (szHost);

    /* If not an address, then try to resolve it as a hostname */
    if ((lAddr == INADDR_NONE) &&
        (strcmp (szHost, "255.255.255.255"))) {

      lpstHost = gethostbyname(szHost);
      if (lpstHost) {  /* success */
        lAddr = *((u_long FAR *) (lpstHost->h_addr));
      } else {
        lAddr = INADDR_ANY;   /* failure */
      }
    }
  }
  return (lAddr);
} /* end GetAddr() */
```


porting from UNIX

may vary by winsock version

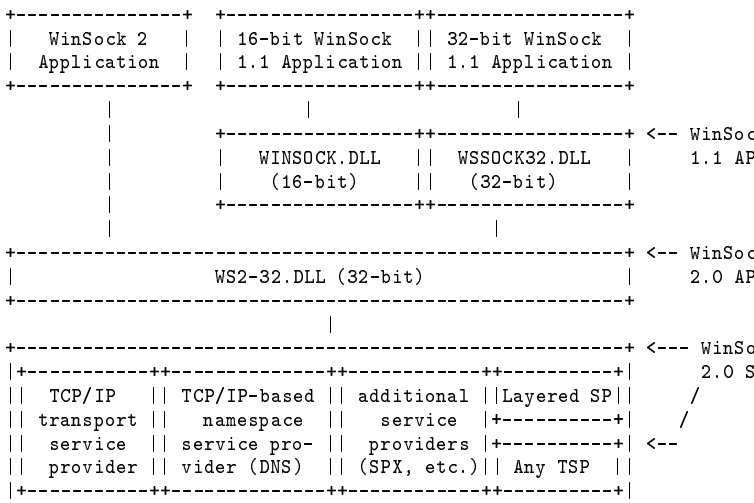
- only need windows.h
- socket are NOT file handles, use send/recv not write/read
- no fork(), use threads: _beginthread() and link with -MT
- no syslog(), or errno/perror
- replace bcopy() etc. with mem-cpy() etc.
- select() can't be used as timer
- worry about byte-order – ntohl/htonl
- no UNIX domain (winsock 1.1)
- no RPC (not included)
- no signal()
- event driven I/O WSAxxxx()
- most setsockopt()

yes: multicast, RAW/ICMP

winsock 2

wsock32.dll

- multi protocol support (ATM)
- scatter/gather
- quality of service
- overlapped I/O (more WSAcalls())
- probably need other packages for RAW sockets or data-link layer support (windump)



timing

- GetTickCount() returns milliseconds
- microseconds from performance counter

```
double
seconds()
{
        static unsigned int mhz=0;
        LARGE_INTEGER t;
        double s;

        if (mhz ==0 ){
                QueryPerformanceFrequency(&t);
                mhz = t.LowPart;
        }
        QueryPerformanceCounter(&t);
        s = t.QuadPart;
        return(s/mhz);
}
```


asynch I/O

non-console

- windows are objects that communicate with messages
- asynch sockets uses Windows messages
- WSAAsynchGetHostByName(), WSAAsynSelect()
- handle with Windows event handlers
- GUI friendly
- typically slower
- then there are the C++ socket classes, CSocket:: and CAsyncScoket::

java networking

- interpreted language
- applets or jre
- applets restricted to communicating to server only
- RPC and more with RMI
- more network features in java 1.2
- object-oriented interface

TCP echo client

```
/* invoke with target hostname */
import java.io.*;
import java.net.*;

public class EchoClient {
    public static void main(String[] args) throws Exception {

    Socket echoSocket = new Socket(args[0], 7);
    PrintWriter out = new PrintWriter(echoSocket.getOutputStream(),
    BufferedReader in = new BufferedReader(new InputStreamReader(
                            echoSocket.getInputStream()));
        BufferedReader stdIn = new BufferedReader(
                                new InputStreamReader(System.in))
        String userInput;

        while ((userInput = stdIn.readLine()) != null) {
            out.println(userInput);
            System.out.println("echo: " + in.readLine());
        }
        echoSocket.close();
    }
}
```


TCP server

```
import java.io.*;
import java.net.*;

class TCPServer {

  public static void main(String argv[]) throws Exception
    {
      String clientSentence;
      String capitalizedSentence;

      ServerSocket welcomeSocket = new ServerSocket(6789);

      while(true) {

           Socket connectionSocket = welcomeSocket.accept();

           BufferedReader inFromClient =
             new BufferedReader(new InputStreamReader(connectionSocket.getInputS
tream()));
           DataOutputStream  outToClient =
             new DataOutputStream(connectionSocket.getOutputStream());
           clientSentence = inFromClient.readLine();
           capitalizedSentence = clientSentence.toUpperCase() + '\n';
           outToClient.writeBytes(capitalizedSentence);
           connectionSocket.close();
        }
    }
}
```


UDP client

```
import java.io.*;
import java.net.*;
class UDPClient {
    public static void main(String args[]) throws Exception
    {
      BufferedReader inFromUser =
        new BufferedReader(new InputStreamReader(System.in));
      DatagramSocket clientSocket = new DatagramSocket();

      InetAddress IPAddress = InetAddress.getByName(args[0]);
      byte[] sendData = new byte[1024];
      byte[] receiveData = new byte[1024];
      String sentence = inFromUser.readLine();
      sendData = sentence.getBytes();

      DatagramPacket sendPacket =
         new DatagramPacket(sendData, sendData.length, IPAddress, 98
      clientSocket.send(sendPacket);
      DatagramPacket receivePacket =
         new DatagramPacket(receiveData, receiveData.length);
      clientSocket.receive(receivePacket);
      String modifiedSentence =
          new String(receivePacket.getData());
      System.out.println("FROM SERVER:" + modifiedSentence);
      clientSocket.close();
    }
}
```

UDP server

```
import java.io.*;
import java.net.*;

class UDPServer {
  public static void main(String args[]) throws Exception
    {
      DatagramSocket serverSocket = new DatagramSocket(9876);
      byte[] receiveData = new byte[1024];
      byte[] sendData  = new byte[1024];

      while(true)
        {
          DatagramPacket receivePacket =
             new DatagramPacket(receiveData, receiveData.length);
          serverSocket.receive(receivePacket);
          String sentence = new String(receivePacket.getData());
          InetAddress IPAddress = receivePacket.getAddress();
          int port = receivePacket.getPort();

          String capitalizedSentence = sentence.toUpperCase();

          sendData = capitalizedSentence.getBytes();
          DatagramPacket sendPacket =
             new DatagramPacket(sendData, sendData.length, IPAddress,
                                port);
          serverSocket.send(sendPacket);
        }
    }
}
```


socket options

various Socket methods for socket options

- setTcpNoDelay(boolean on)
- setSoLinger(boolean on, int linger)
- setSendBufferSize(int size)
- setReceiveBufferSize(int size)

RMI – remote objects/methods (RPC)

next time

IPv6