Chapter 2: Basic socket

-Socket address

-TCP socket

+Its client

+Its server

+Its I/O stream

-UDP socket

+DatagramPacket

+Its client

+Its server

+UDP receive and send stream

Socket address:

InetAddress class (2 subclasses Inet4Address and Inet6Address): represent name and numberical destination (name e.g [www.sdadasd.org](http://www.sdadasd.org) and numberical e.g IPv4 and IPv6 format)

InetAddress can illustrate address as string instead of numberical form, and can also check address’ properties

InternetAddress: use IP address to link hosts and internet

NetworkInterface class: getNetworkInterface shows all network interfaces, including loopback interface, link-local one

TCP socket:

2 class: Socket (starting point at client’s TCP) and ServerSocket (Internet/server receiving end)

+TCP client

TCP client procedure:

+Construct socket to link to server

+Doing I/O stream

+Close connection with “close()” of socket

Example:

TCPEchoClient.java for communicating echo server

Initializing socket codes, connecting socket to server, closing socket codes, showing socket’s properties

InetSocketAddress:

Port only code is used for showing address of server

String and port code is used for showing IP address

+TCP server

TCP server procedure:

-Create ServerSocket

-Use accept() to receive client connection

-Using I/O stream of Client’s socket

-using socket’s close() for terminate connection

TCPEchoServer.java: implement echo

Server = endpoint; client = starting point. Endpoint use port number, starting point use IP number

Codes for initializing TCP server

Code for connecting socket to server: bind(), accept() and close()

Code for showing properties of ServerSocket

I/O stream

InputStream read input and OutputStream write input and display it

OutputStream: superclass of output stream, write() transfer input, flush() pushes buffed data to output, close() terminate output stream

InputStream: superclass of input stream, read() receives data transfered, available() return number of byte available (memory used?), close() terminate input stream

UDP socket: UDP doesnt have to establish connection like TCP, but corrupted data get removed instead of stored and adjusted like TCP

No guarantee data is sent and be sent in order, UDP sent data faster than TCP and use for anything other than maintaining reliable byte-stream

DatagramPacket: do similar task of TCP receiving and sending stream of bytes

Codes relating to DatagramPacket: initialize, gaining address, controling data’s length

UDP Client:

Procedure:

+Construct DatagramSocket

+use send() and receive() to send and receive Datagram

+deallocate socket with close()

UDP client communicate with several servers, unlike TCP only communicate with specific server

Example:UDPEchoClientTimeout.java sends a datagram containing the string to be echoed and prints whatever it receives back from the server

Code for initialize, connect, getting address, sending and receiving datagram, setting timeout

Server:

+Create DatagramSocket for prepare receiving datagrams

+receive DatagramPacket with receive() to know client’s address

+Communicate with send() and receive()

Example:UDPEchoServer.java

Receiving and send stream:

TCP store socket into buffer in case of data re-transmission, data treated as 1 continuous sequence of bytes, for UDP, data can come from different senders

UDP: when receive() is called with a DatagramPacket containing a buffer of size n, and the size of the first message in the receive queue exceeds n, only the first n bytes of

the message are returned. The remaining bytes are quietly discarded, with no indication to the receiving program that information has been lost