JAVA Socket Programming

What is a socket?

- Socket
  - The combination of an IP address and a port number. (RFC 793 ,original TCP specification)
  - The name of the Berkeley-derived *application programming interfaces* (APIs) for applications using TCP/IP protocols.
  - Two types
    - Stream socket : reliable two-way connected communication streams
    - Datagram socket

- Socket pair
  - Specified the two end points that uniquely identifies each TCP connection in an internet.
  - 4-tuple: (client IP address, client port number, server IP address, server port number)

Client-server applications

- Implementation of a protocol standard defined in an RFC. (FTP, HTTP, SMTP…)
  - Conform to the rules dictated by the RFC.
  - Should use the port number associated with the protocol.

- Proprietary client-server application.
  - A single developer(or team) creates both client and server program.
  - The developer has complete control.
  - Must be careful not to use one of the well-known port number defined in the RFCs.

* well-known port number : managed by the Internet Assigned Numbers Authority(IANA)

Socket Programming with TCP

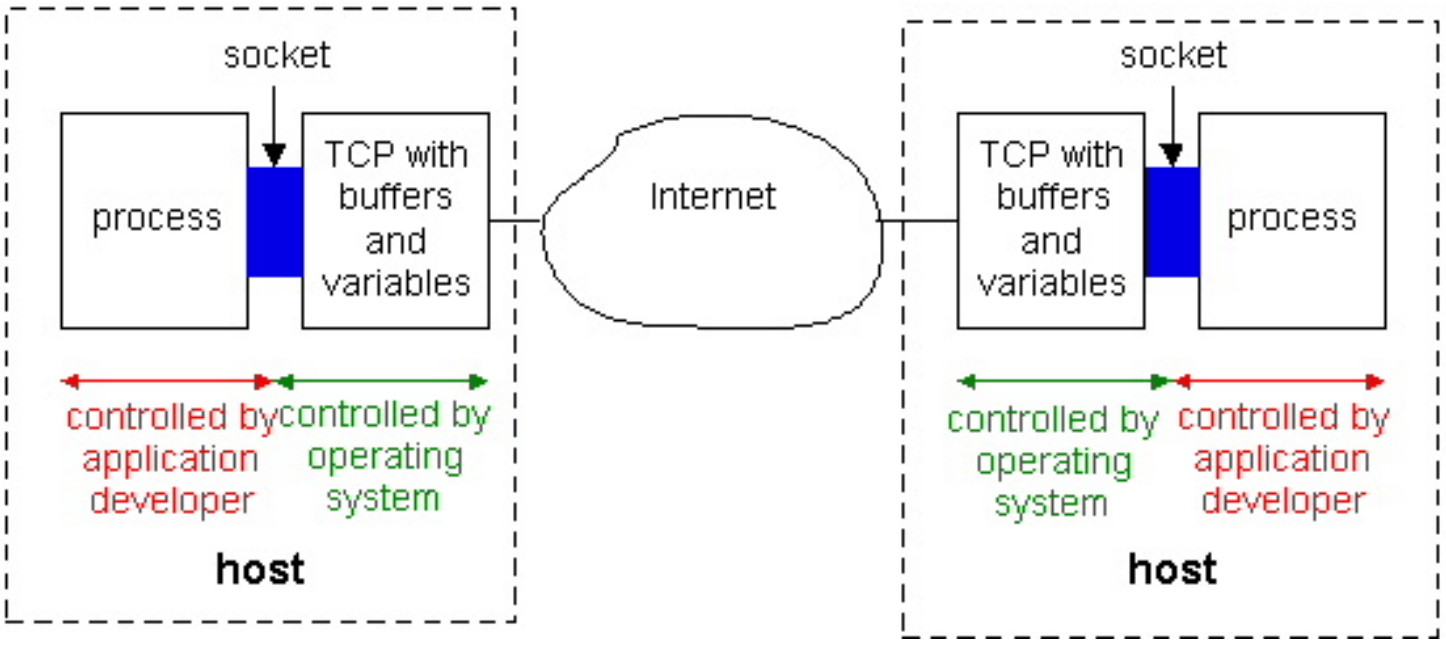


Figure 2.6-1: Processes communicating through TCP sockets

The application developer has the ability to fix a few TCP parameters, such as maximum buffer and maximum segment sizes.

Sockets for server and client

- Server
  - Welcoming socket
    - Welcomes some initial contact from a client.
  - Connection socket
    - Is created at initial contact of client.
    - New socket that is dedicated to the particular client.
- Client
  - Client socket
    - Initiate a TCP connection to the server by creating a socket object. (Three-way handshake)
    - Specify the address of the server process, namely, the IP address of the server and the port number of the process.

Socket functional calls

- socket (): Create a socket
- bind(): bind a socket to a local IP address and port #
- listen(): passively waiting for connections
- connect(): initiating connection to another socket
- accept(): accept a new connection
- Write(): write data to a socket
- Read(): read data from a socket
- sendto(): send a datagram to another UDP socket
- recvfrom(): read a datagram from a UDP socket
- close(): close a socket (tear down the connection)

Sockets

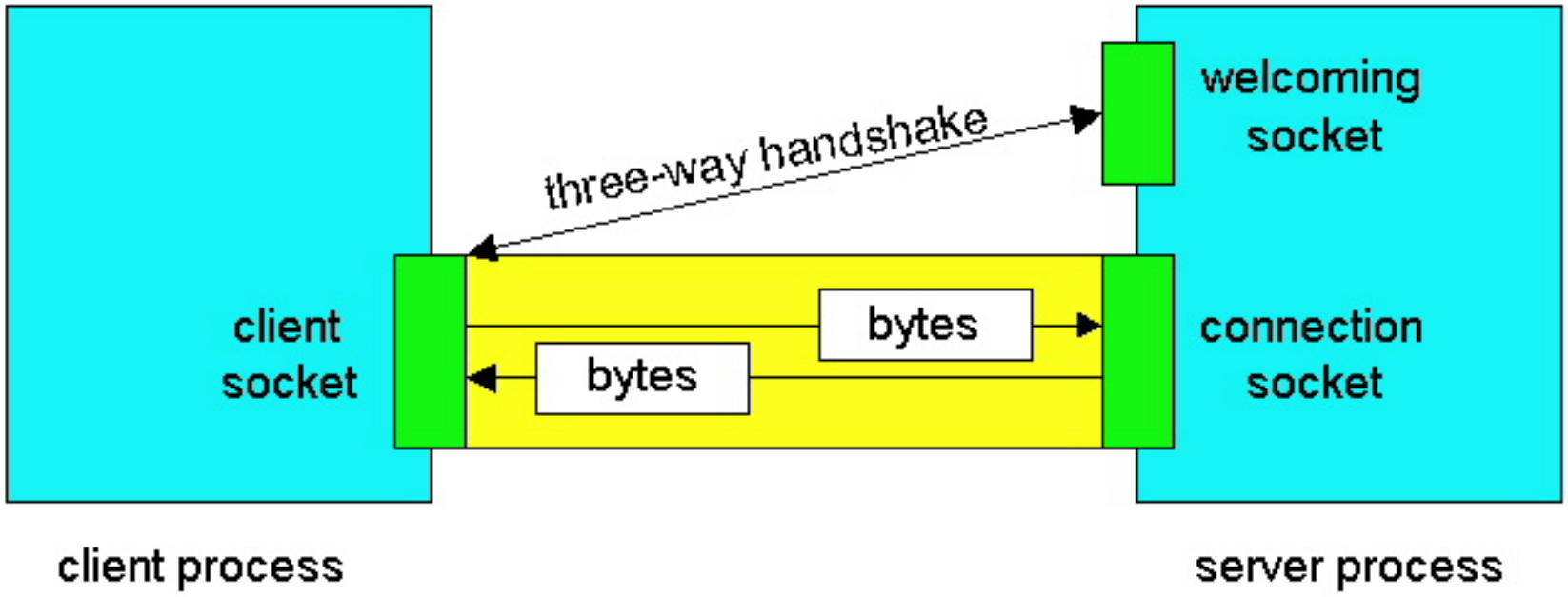
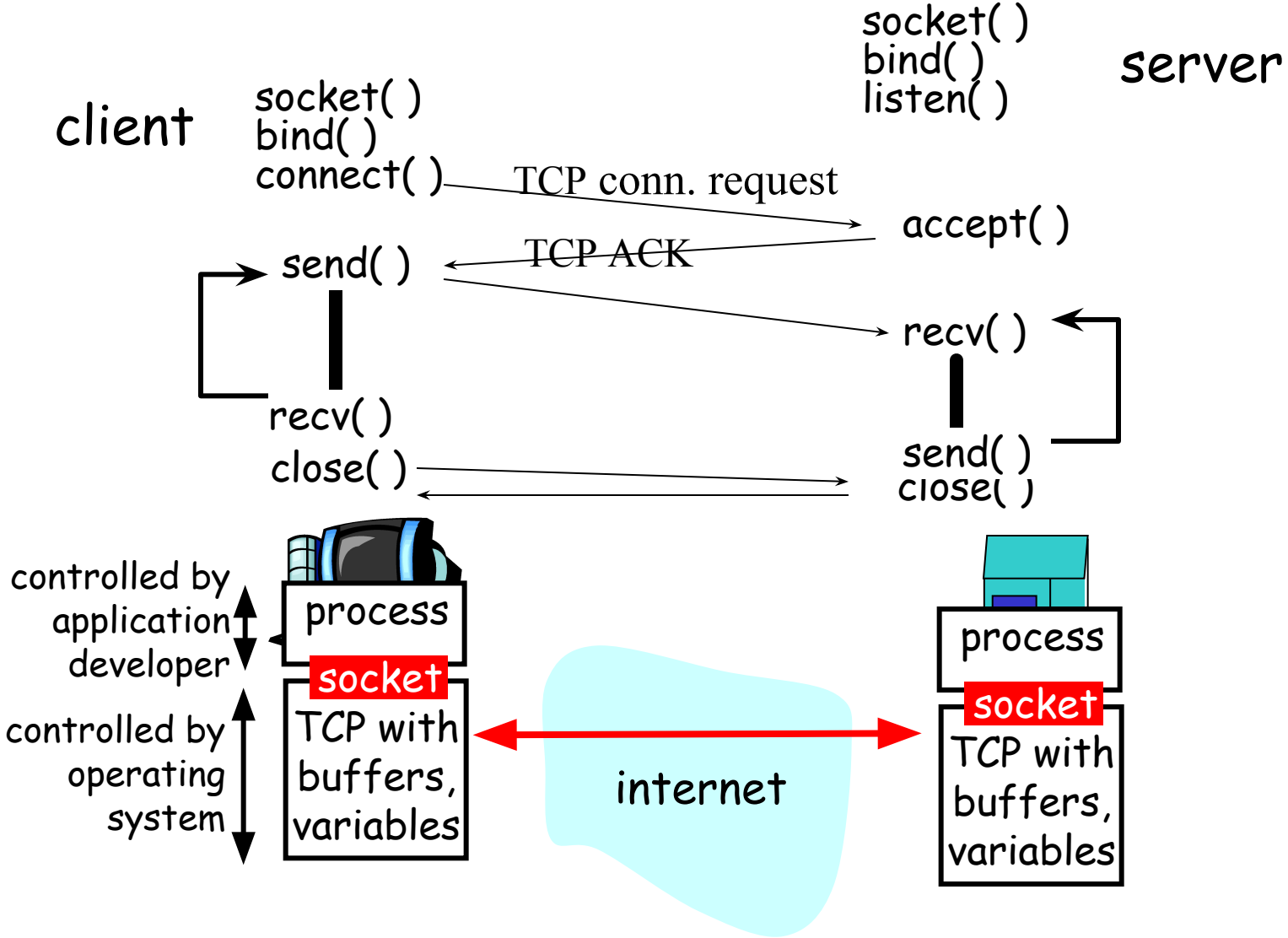


Figure 2.6-2: Client socket, welcoming socket and connection socket

Socket-programming using TCP

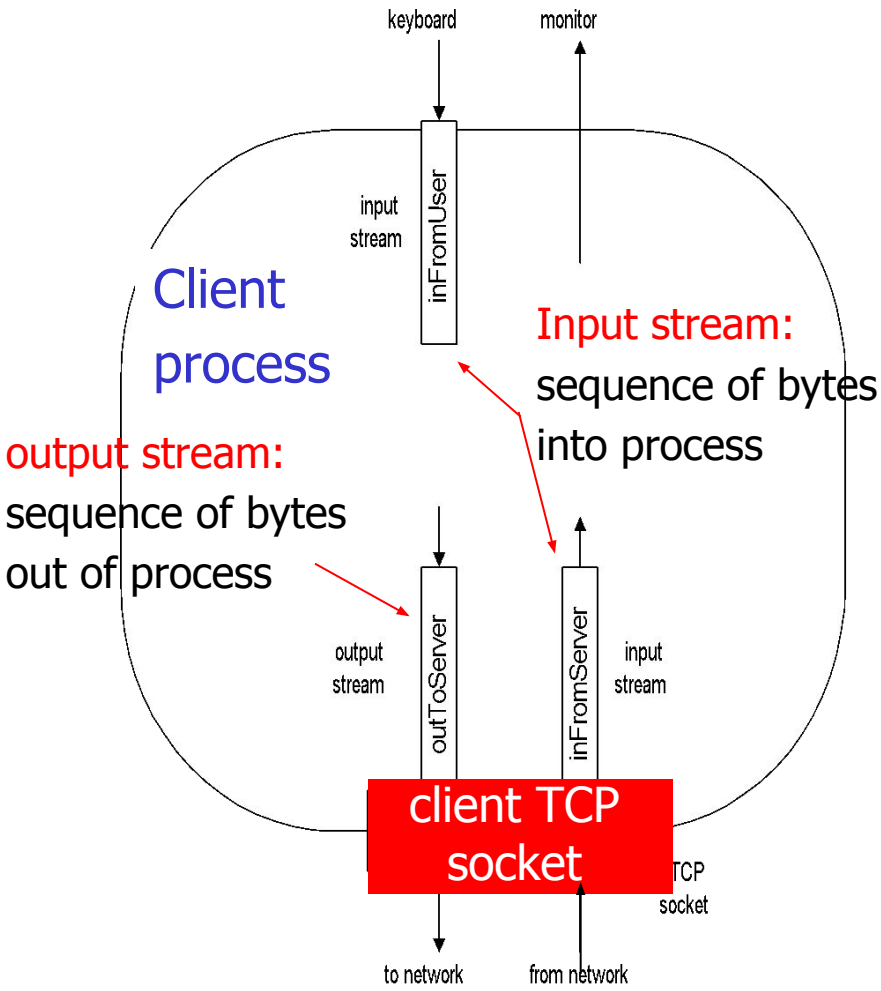
TCP service: reliable byte stream transfer



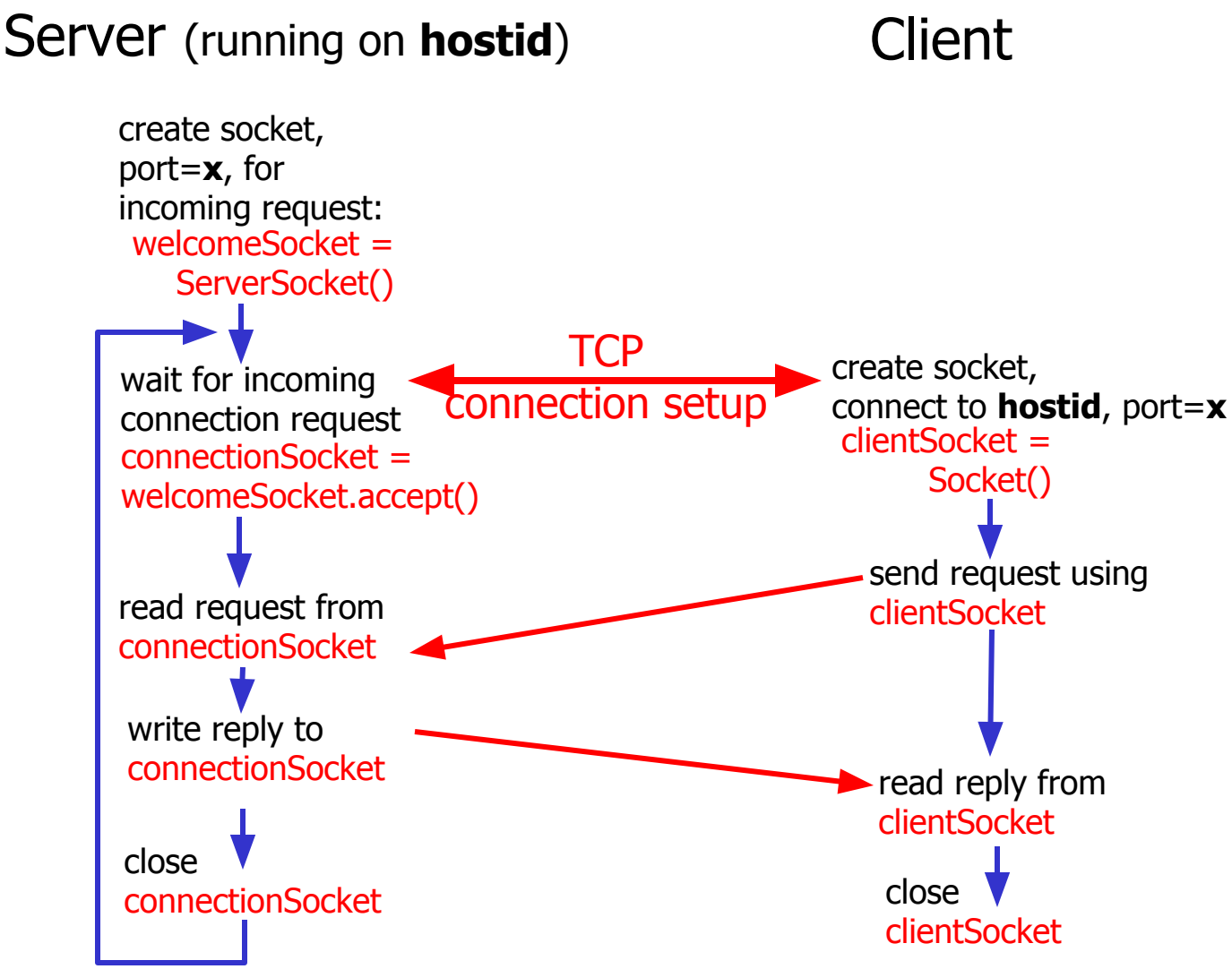
Socket programming with TCP

Example client-server app:

- client reads line from standard input (**inFromUser** stream) , sends to server via socket (**outToServer** stream)
- server reads line from socket
- server converts line to uppercase, sends back to client
- client reads, prints modified line from socket (**inFromServer** stream)



Client/server socket interaction: TCP



JAVA TCP Sockets

- In Package java.net
  - java.net.Socket
    - Implements client sockets (also called just "sockets").
    - An endpoint for communication between two machines.
    - Constructor and Methods
      - Socket(String host, int port): Creates a stream socket and connects it to the specified port number on the named host.
      - InputStream getInputStream()
      - OutputStream getOutputStream()
      - close()
  - java.net.ServerSocket
    - Implements server sockets.
    - Waits for requests to come in over the network.
    - Performs some operation based on the request.
    - Constructor and Methods
      - ServerSocket(int port)
      - Socket Accept(): Listens for a connection to be made to this socket and accepts it. This method blocks until a connection is made.

TCPClient.java

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

class TCPClient {
    public static void main(String argv[]) throws Exception
    {
        String sentence;
        String modifiedSentence;

        BufferedReader inFromUser =
            new BufferedReader(new InputStreamReader(System.in));

        Socket clientSocket = new Socket("hostname", 6789);

        DataOutputStream outToServer =
                new DataOutputStream(clientSocket.getOutputStream());
```

TCPClient.java

```
        BufferedReader inFromServer =
          new BufferedReader(new
    InputStreamReader(clientSocket.getInputStream()));

     sentence = inFromUser.readLine();

     outToServer.writeBytes(sentence + '\n');

     modifiedSentence = inFromServer.readLine();

     System.out.println("FROM SERVER: " + modifiedSentence);

     clientSocket.close();

    }
}
```

TCPServer.java

```
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.getInputStream()));
```

TCPServer.java

```
        DataOutputStream  outToClient =
             new DataOutputStream(connectionSocket.getOutputStream());

        clientSentence = inFromClient.readLine();

        capitalizedSentence = clientSentence.toUpperCase() + '\n';

        outToClient.writeBytes(capitalizedSentence);

      }
    }
}
```

More Info

- http://www.javatpoint.com/java-networking