Network Applications: DNS Details; UDP Network App Programming

Y. Richard Yang

http://zoo.cs.yale.edu/classes/cs433/

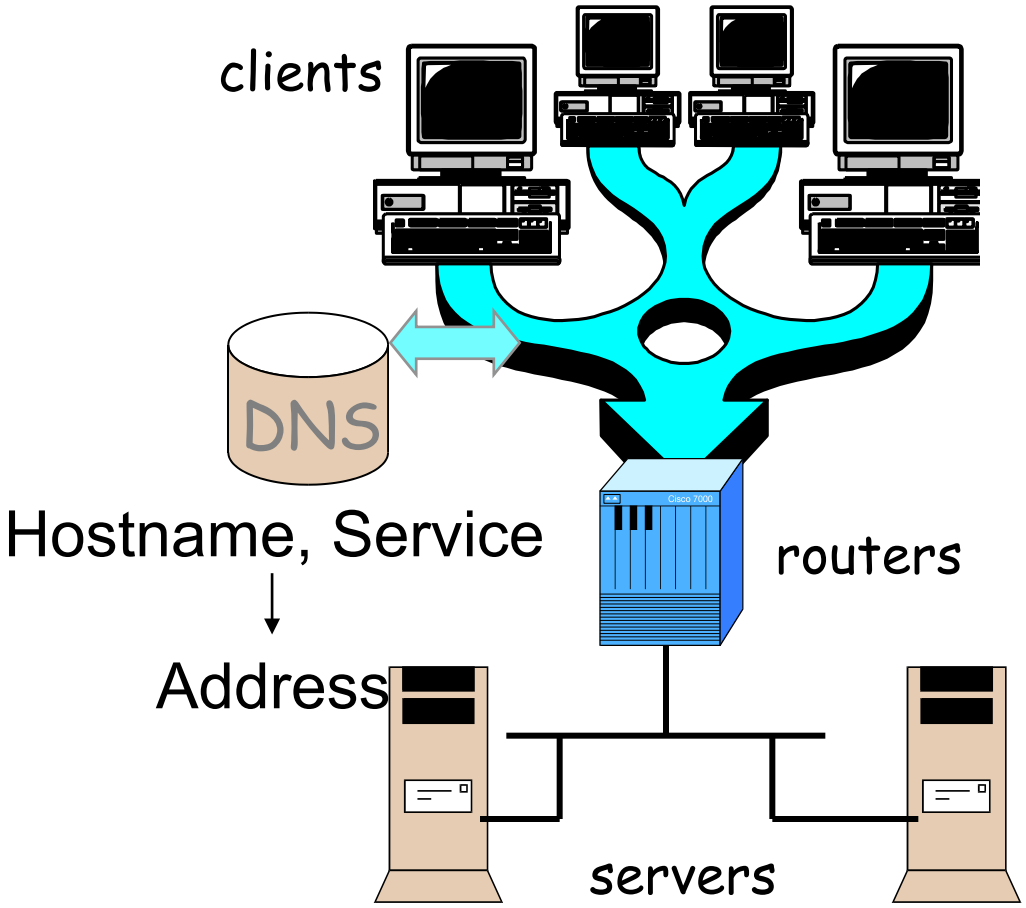
2/3/2016

Outline

- Admin and recap
- Network app programming

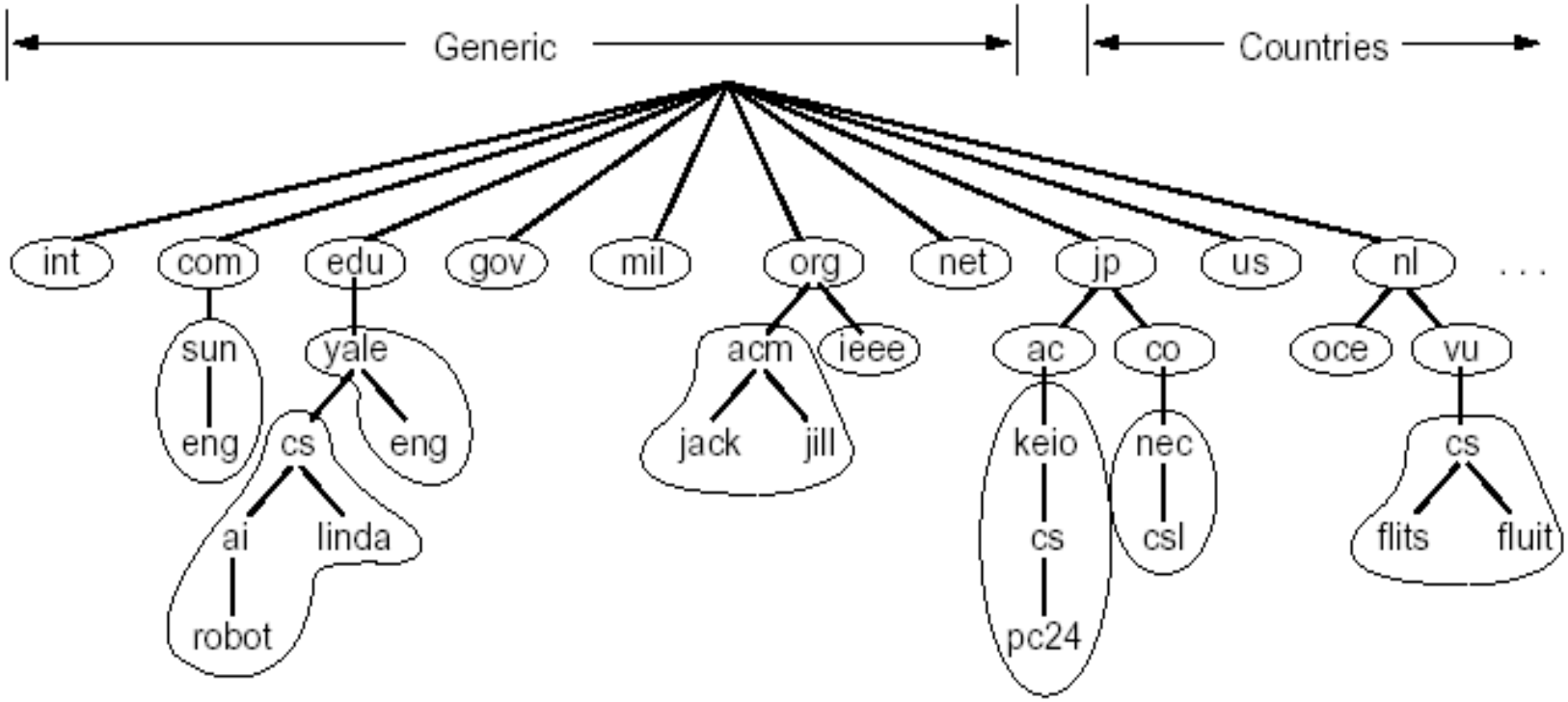
Recap: Domain Name System (DNS)

- Function
  - map between (domain name, service) to value, e.g.,
    - (www.cs.yale.edu, Addr) -> 128.36.229.30
    - (cs.yale.edu, Email) -> netra.cs.yale.edu



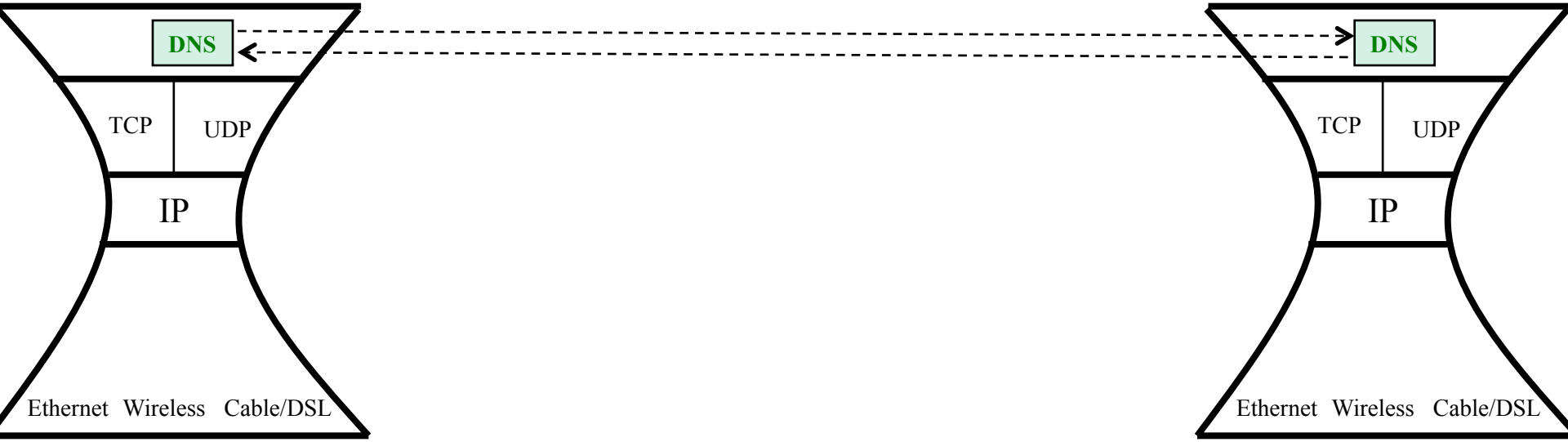
Recap: DNS

- Key design features of DNS
  - Hierarchical domain name space allowing delegation
  - Recursive or iterative queries



DNS Message Format?

Basic encoding decisions: UDP/TCP, how to encode domain name, how to encode answers…



Observing DNS Messages

- Issue DNS query using the command dig:
  - force iterated query to see the trace:

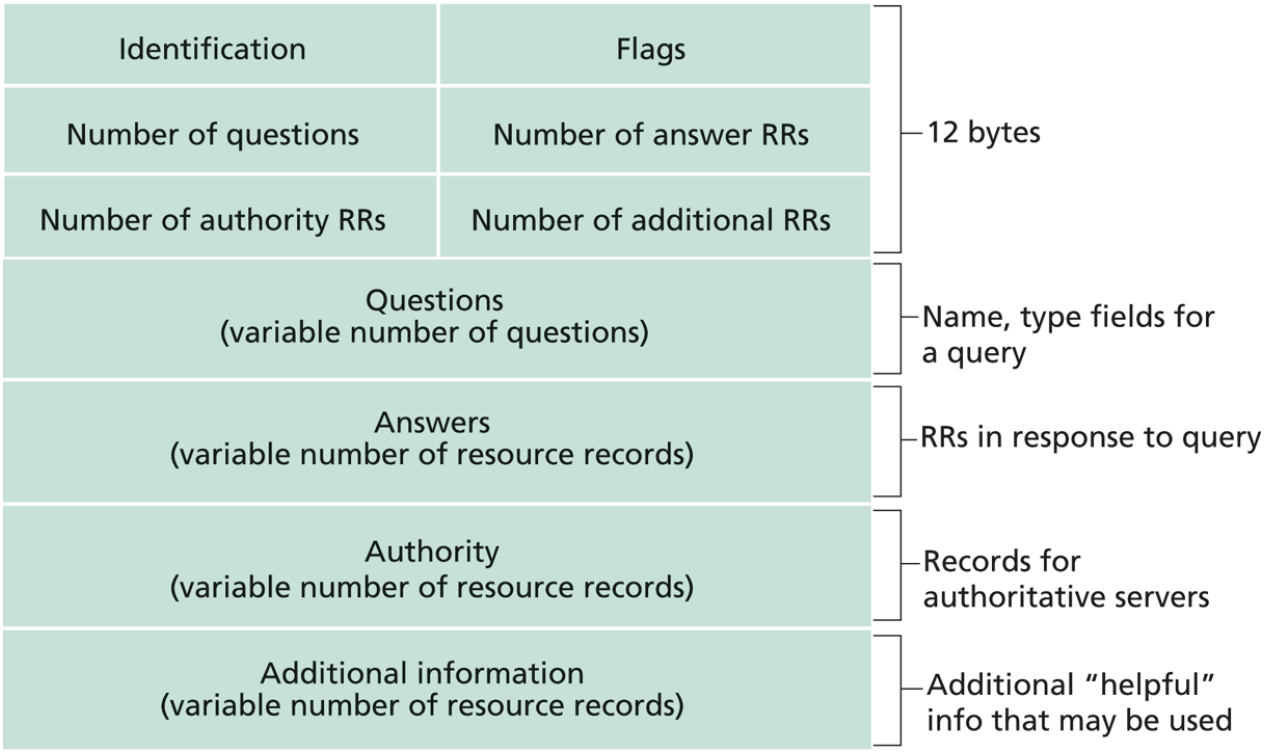
    `%dig +trace www.cnn.com`
    - see the manual for more details

- Capture the messages
  - DNS server is at port 53
    - Display and clear DNS cache
      - https://support.apple.com/en-us/HT202516 (e.g., MAC)
  - Try to load the dns-capture file from class Schedule page, if you do not want live capture

https://www.ietf.org/rfc/rfc1035.txt

DNS Protocol, Messages

DNS protocol : typically over UDP (can use TCP); *query* and *reply* messages, both with the same *message format*



DNS Details

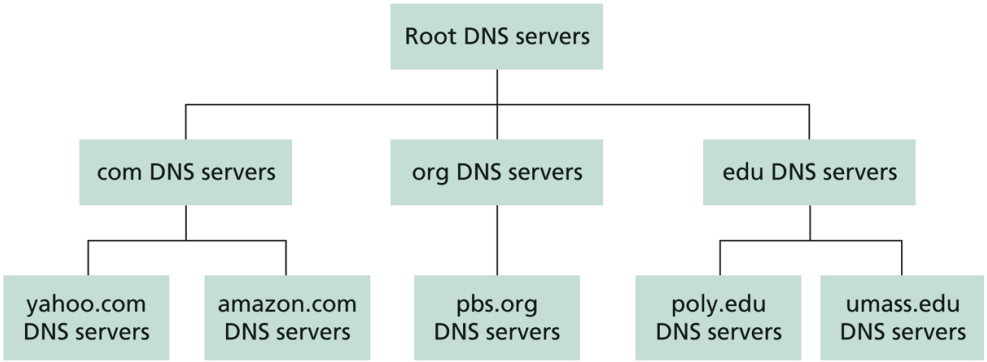
- Header (Sec. 4.1.1 of https://www.ietf.org/rfc/rfc1035.txt)
- Encoding of questions (Sec. 4.1.2):
  - [Label-length label-chars]
- Encoding of answers (Sec. 4.1.3)
  - Pointer format (http://www.iana.org/assignments/dns-parameters/dns-parameters.xhtml)

- See example DNS packets

Evaluation of DNS

Key questions to ask about a C-S application

- **extensible**?
- **scalable**?
- **robust**?
- **security**?



What DNS did Right?

- Hierarchical delegation avoids central control, improving manageability and scalability
- Redundant servers improve robustness
  - see http://www.internetnews.com/dev-news/article.php/1486981 for DDoS attack on root servers in Oct. 2002 (9 of the 13 root servers were crippled, but only slowed the network)
- Caching reduces workload and improves robustness

Problems of DNS

- Domain names may not be the best way to name other resources, e.g. files
- Simple query model makes it hard to implement advanced query
- Relatively static resource types make it hard to introduce new services or handle mobility
- Although theoretically you can update the values of the records, it is rarely enabled
- Early binding (separation of DNS query from application query) does not work well in mobile, dynamic environments
  - e.g., load balancing, locate the nearest printer

Outline

- Recap
- Network app programming

Socket Programming

Socket API

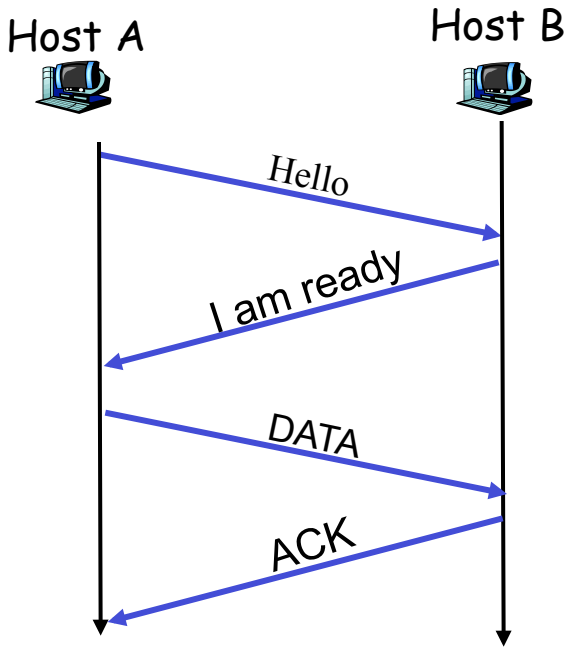
- introduced in BSD4.1 UNIX, 1981
- Two types of sockets
  - Connectionless (UDP)
  - connection-oriented (TCP)

socket

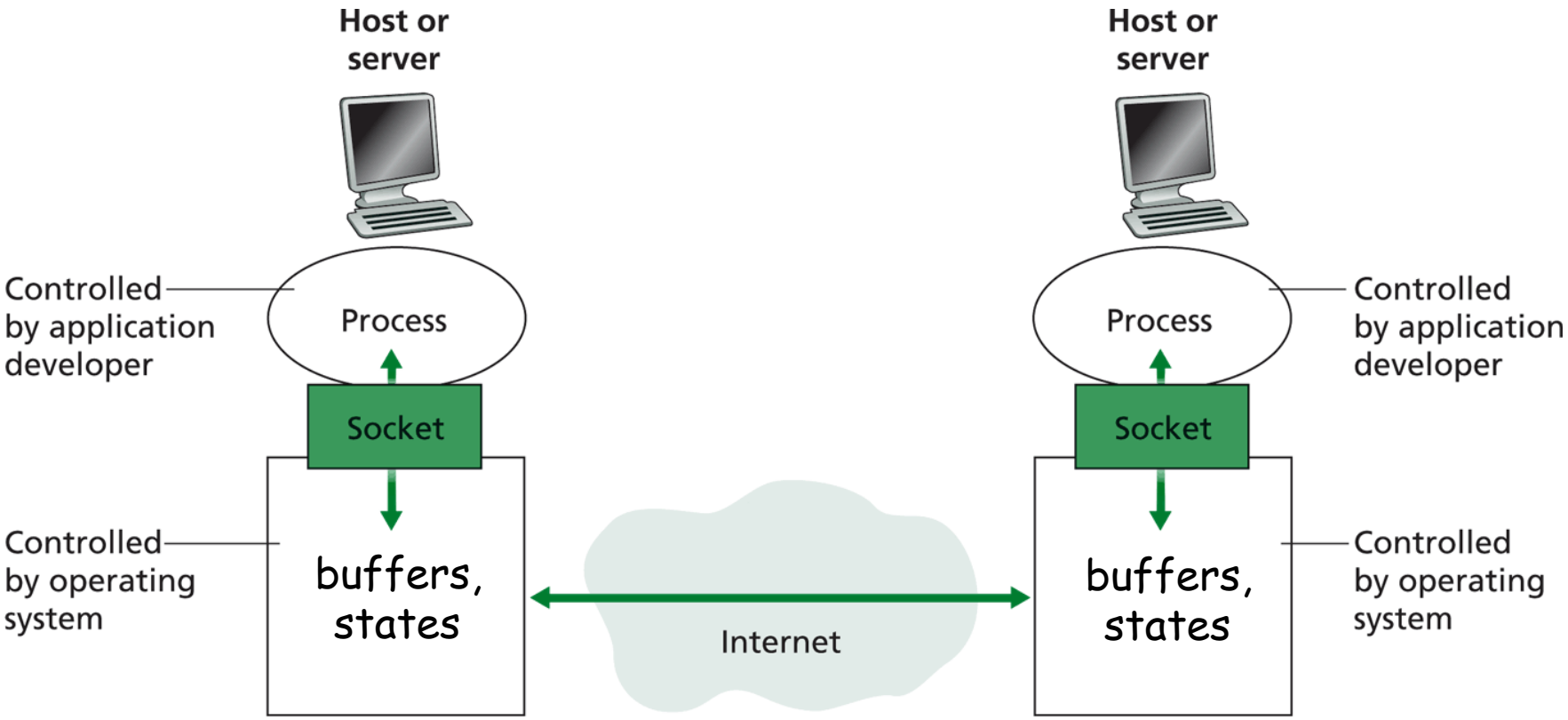
an interface (a "door") into which one application process can both send and receive messages to/from another (remote or local) application process

Services Provided by Transport

- User data protocol (UDP)
  - multiplexing/demultiplexing
- Transmission control protocol (TCP)
  - multiplexing/demultiplexing
  - reliable data transfer
  - rate control: flow control and congestion control



Big Picture: Socket



Outline

- Recap
- Basic network application programming
  - Overview
  - UDP (Datagram Socket)

DatagramSocket(Java) (Basic)

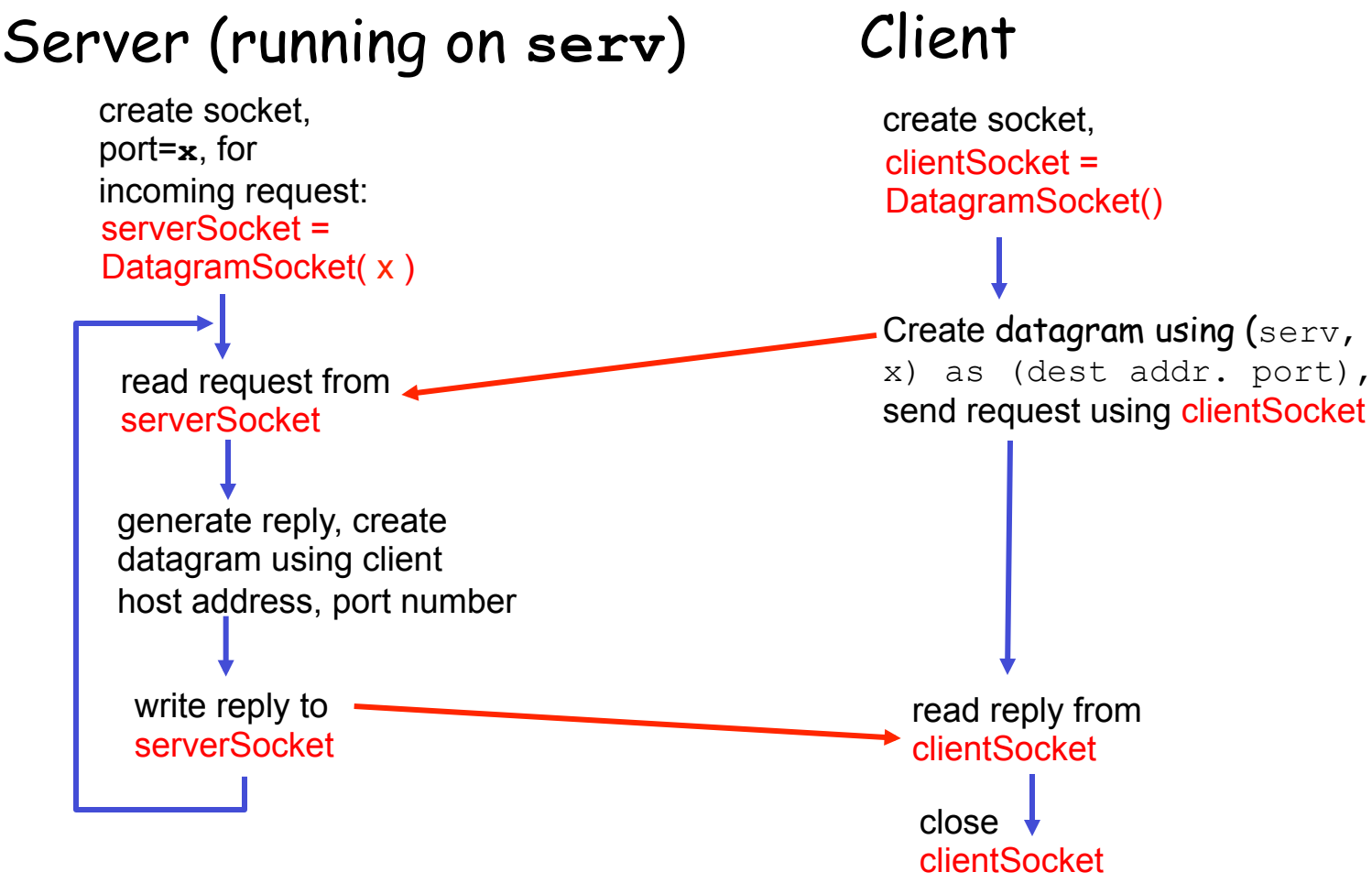
- **DatagramSocket**()
  constructs a datagram socket and binds it to any available port on the local host
- **DatagramSocket**(int lport)
  constructs a datagram socket and binds it to the specified port on the local host machine.

- **DatagramPacket**(byte[] buf, int length)
  constructs a DatagramPacket for receiving packets of length length.
- **DatagramPacket**(byte[] buf, int length, InetAddress address, int port)
  constructs a datagram packet for sending packets of length length to the specified port number on the specified host.

- **receive**(DatagramPacket p)
  receives a datagram packet from this socket.
- **send**(DatagramPacket p)
  sends a datagram packet from this socket.

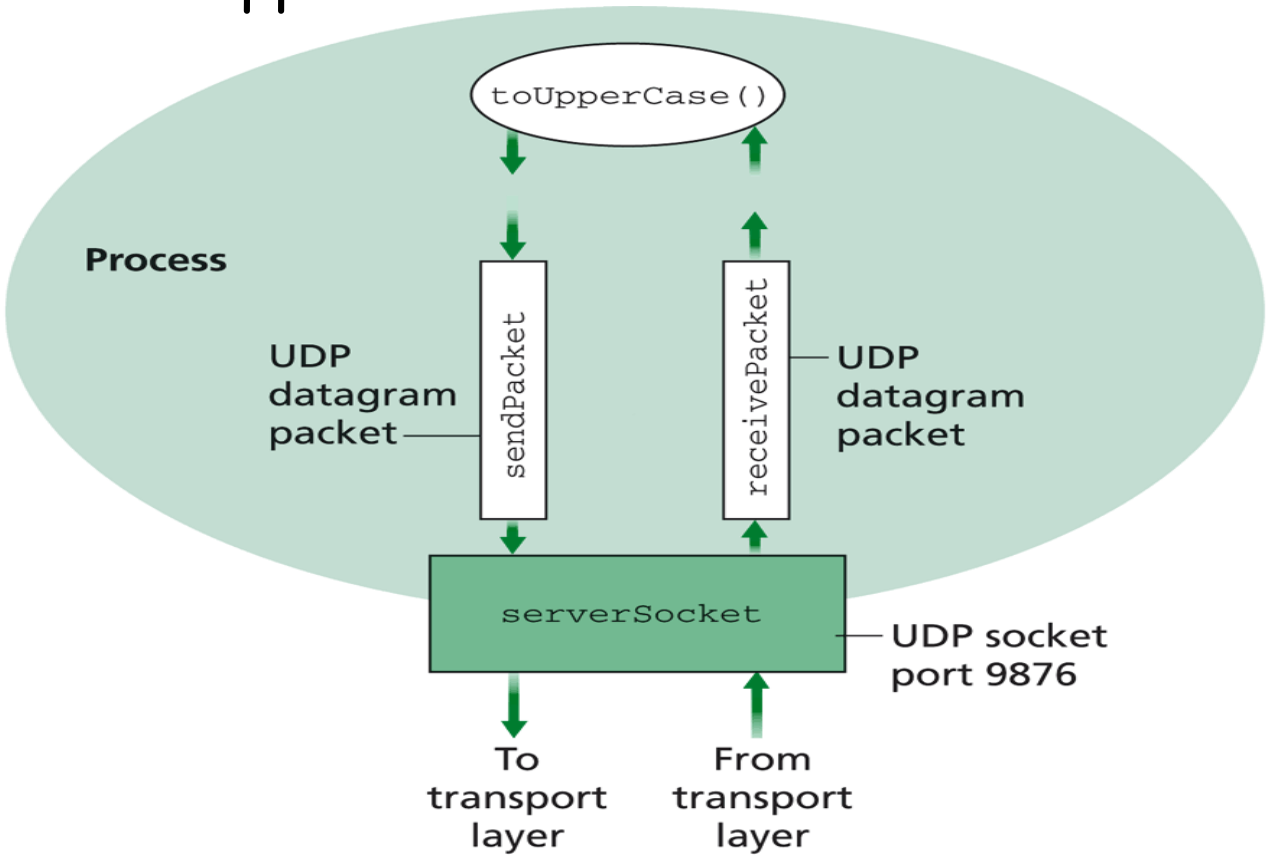
- **close**()
  closes this datagram socket.

Connectionless UDP: Big Picture (Java version)



Example: UDPServer.java

- A simple UDP server which changes any received sentence to upper case.



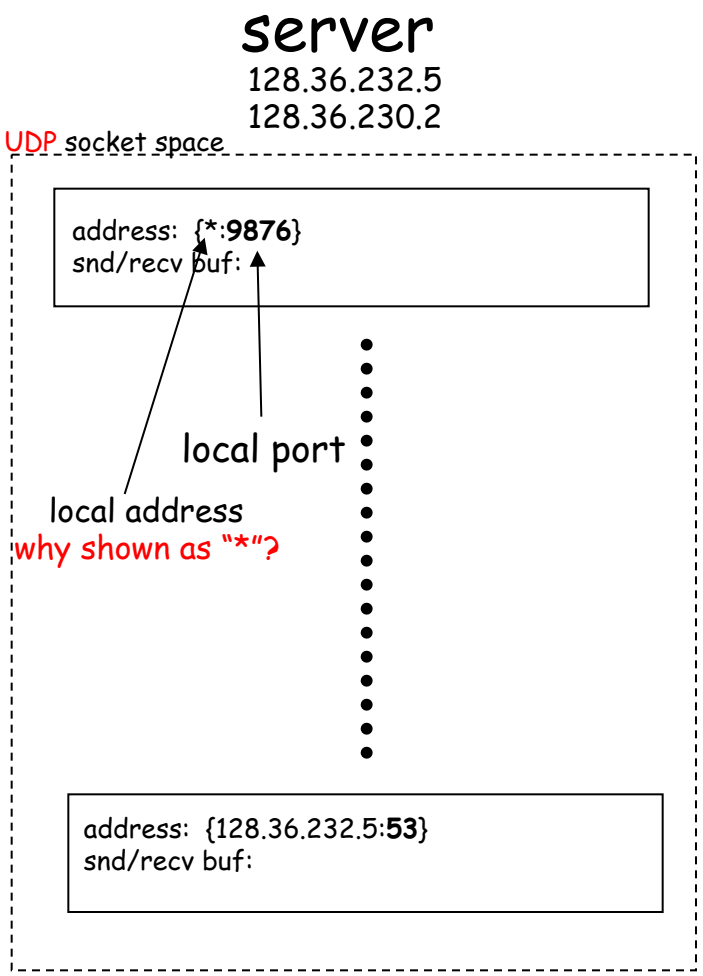
Java Server (UDP): Create Socket

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

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

    DatagramSocket serverSocket = new DatagramSocket(9876);
```

Create datagram socket bind at port 9876

Check socket state:
%netstat –a –p udp –n

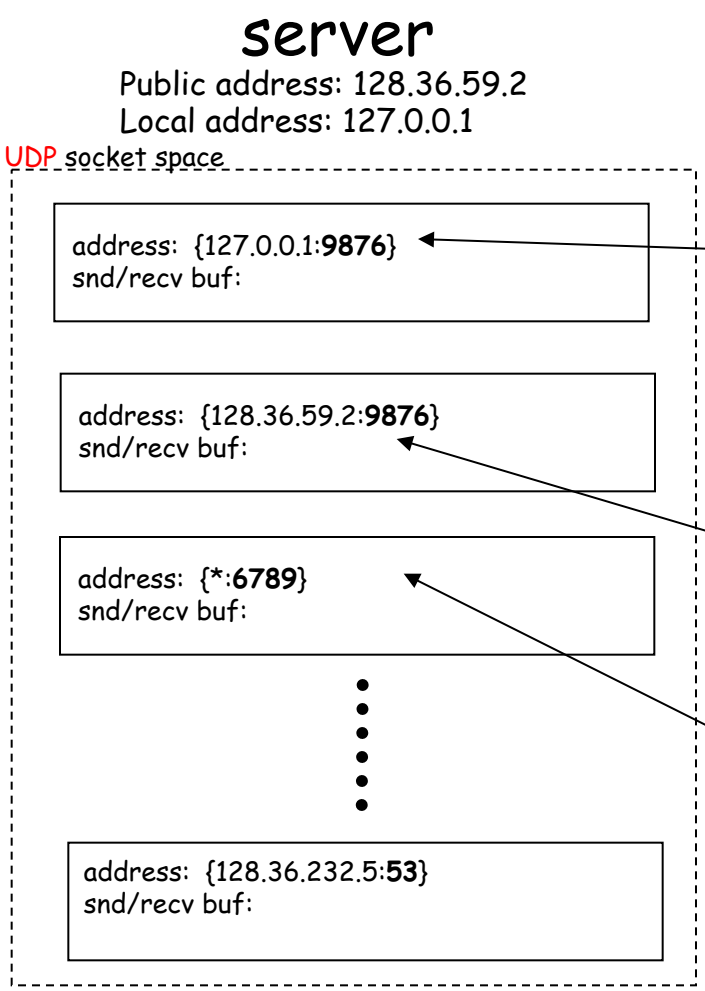
System State after the Call



"*" indicates that the socket binds to all IP addresses of the machine:

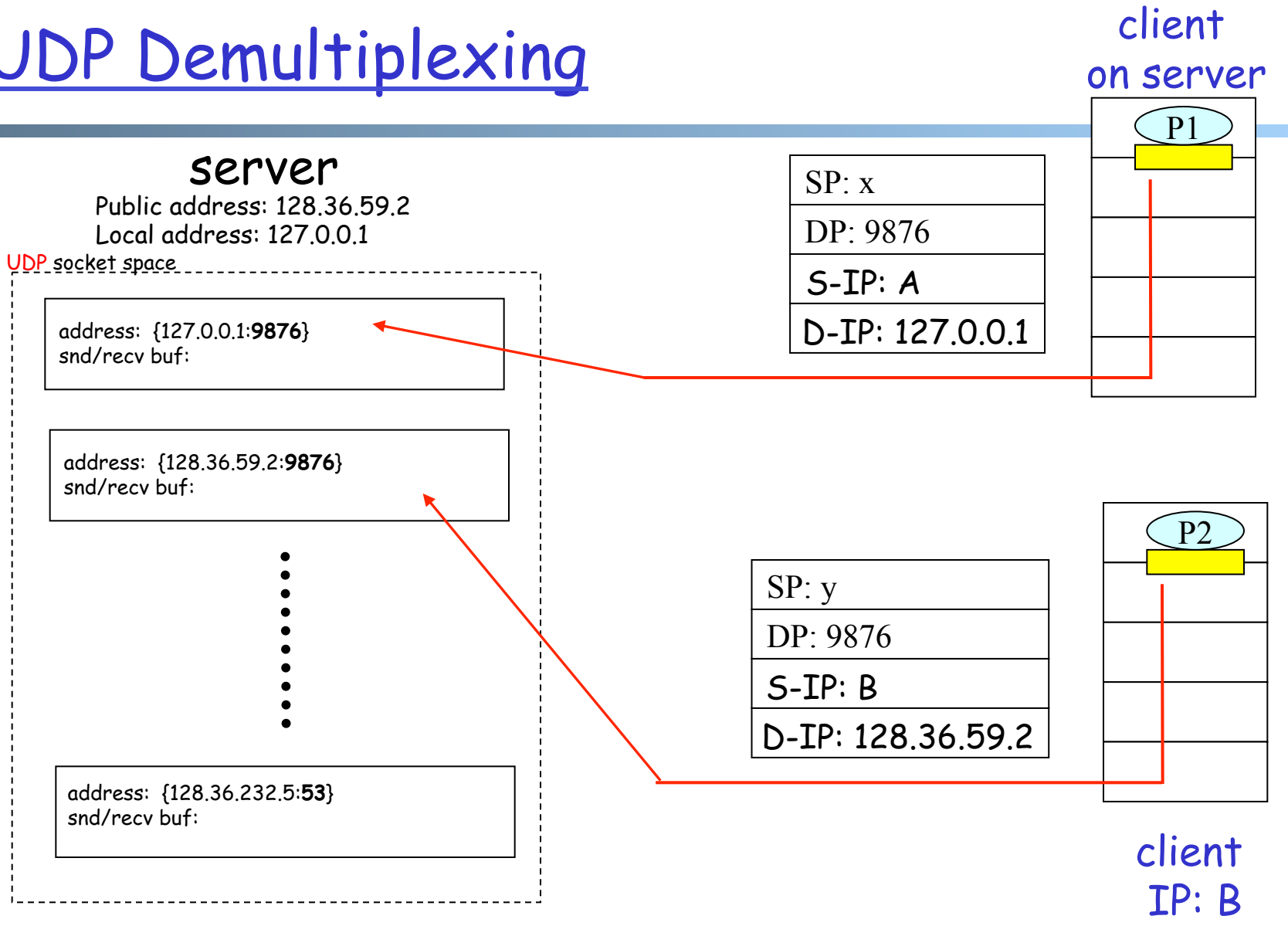
```
% ifconfig -a
```

Binding to Specific IP Addresses



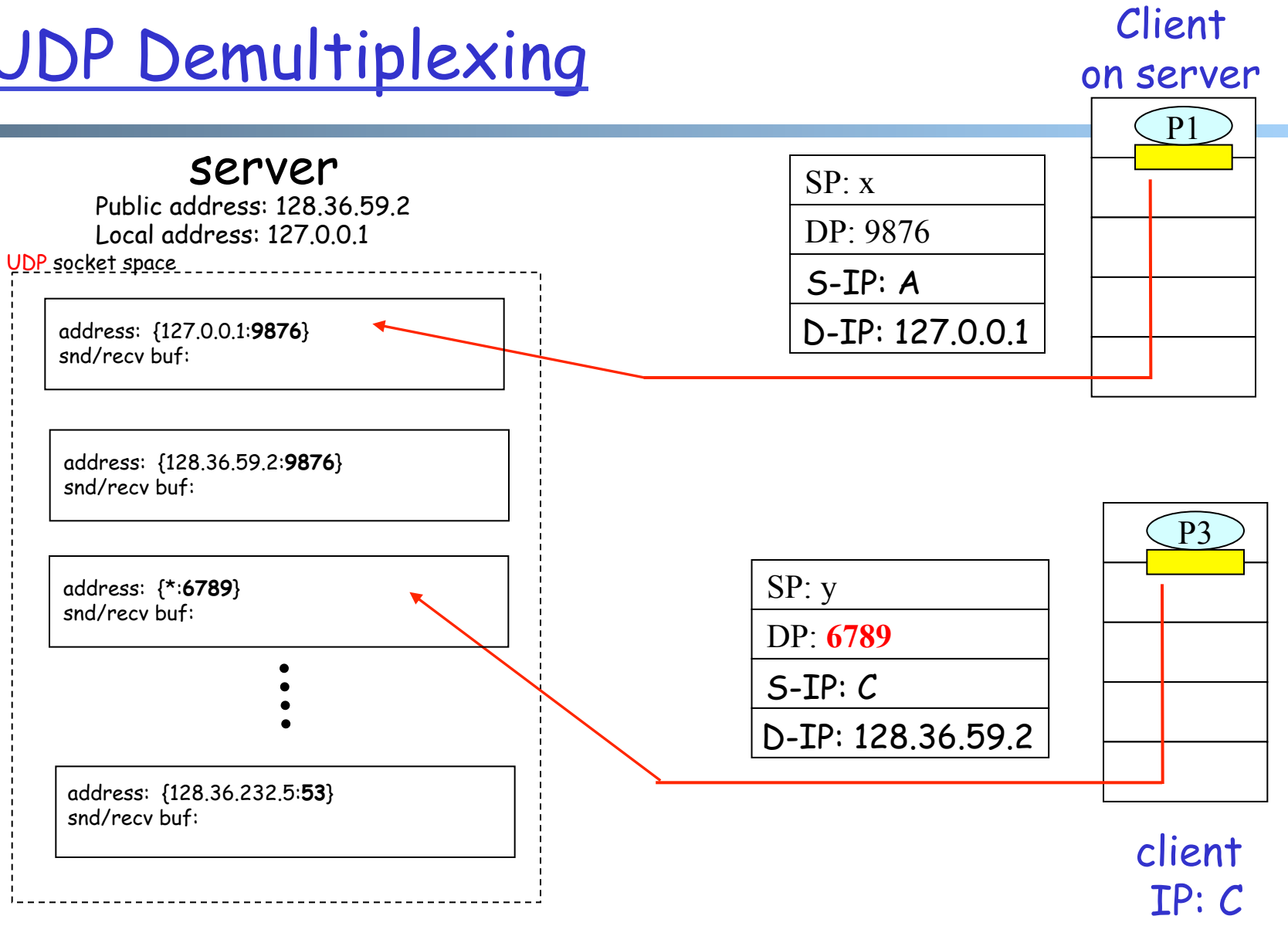

```
InetAddress sIP1 =
   InetAddress.getByName("localhost");
DatagramSocket ssock1 = new
       DatagramSocket(9876, sIP1);

InetAddress sIP2 =
 InetAddress.getByName("128.36.59.2");
DatagramSocket ssock2 = new
       DatagramSocket(9876,sIP2);

DatagramSocket serverSocket = new
DatagramSocket(6789);
```

UDP Demultiplexing



UDP demutiplexing is based on matching (dst address, dst port)

UDP Demultiplexing



UDP demutiplexing is based on matching (dst address, dst port)

Per Socket State

- Each Datagram socket has a set of states:
  - local address
  - send buffer size
  - receive buffer size
  - timeout
  - traffic class

See http://download.java.net/jdk7/archive/b123/docs/api/java/net/DatagramSocket.html

Example: socket state after clients sent msgs to the server

Java Server (UDP): Receiving

```
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 = null;

      while(true)
        {

          DatagramPacket receivePacket =
            new DatagramPacket(receiveData, receiveData.length);

          serverSocket.receive(receivePacket);
```

Create space for received datagram

Receive datagram

DatagramPacket

- **Receiving**
  - **DatagramPacket**(byte[] buf, int length)
    constructs a DatagramPacket for receiving packets of length length.
  - **DatagramPacket**(byte[] buf, int offset, int length)
    constructs a DatagramPacket for receiving packets starting at offset, length length.

- **Sending**
  - **DatagramPacket**(byte[] buf, int length, InetAddress address, int port)
    constructs a datagram packet for sending packets of length length to the specified port number on the specified host.
  - **DatagramPacket**(byte[] buf, int offset, int length, InetAddress address, int port)

Java Server (UDP): Processing

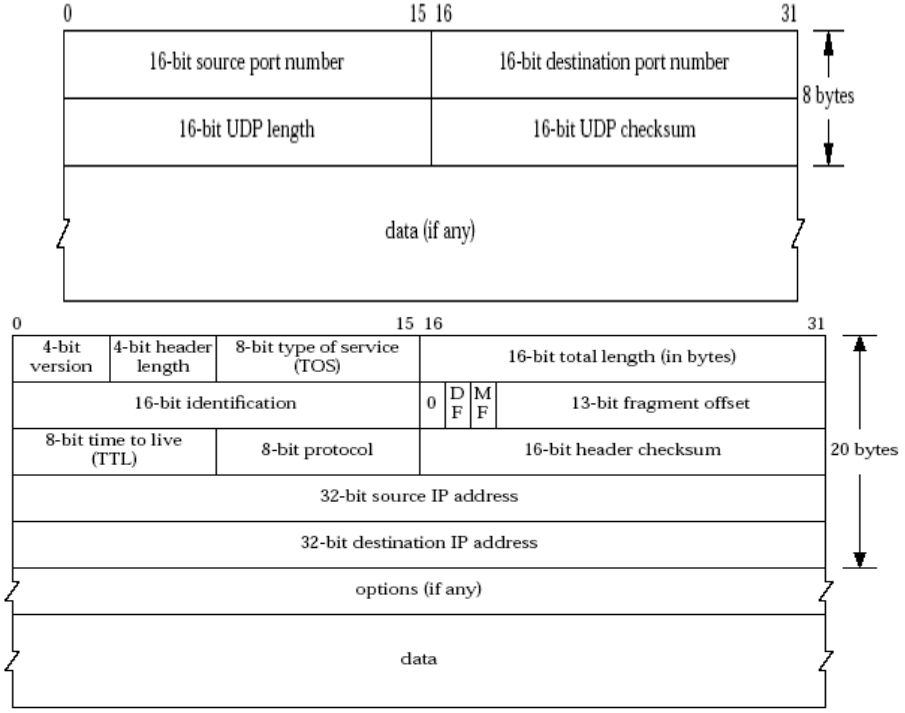
```
import java.io.*;
import java.net.*;

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

…
// process data
String sentence = new String(receivePacket.getData(),
                             0, receivePacket.getLength());
String capitalizedSentence = sentence.toUpperCase();
sendData = capitalizedSentence.getBytes();
```

getData() returns a pointer to an underlying buffer array; for efficiency, don't assume receive() will reset the rest of the array

getLength() returns how much data is valid.

Java Server (UDP): Response

- Java DatagramPacket:
  - `getAddress()/ getPort()` returns the source address/port



Java server (UDP): Reply

Get IP addr port #, of sender

```
InetAddress IPAddress = receivePacket.getAddress();
int port = receivePacket.getPort();
```

Create datagram to send to client

```
DatagramPacket sendPacket =
  new DatagramPacket(sendData, sendData.length,
                     IPAddress, port);
```

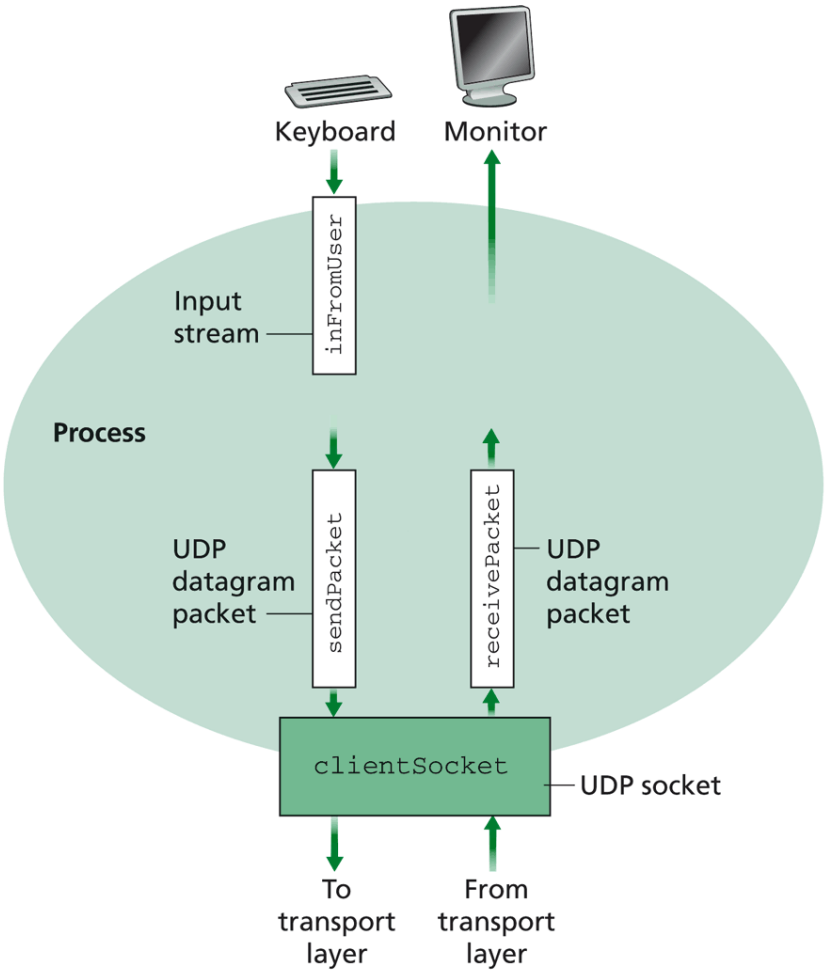
Write out datagram to socket

```
serverSocket.send(sendPacket);
    }
  }
}
```

End of while loop, loop back and wait for another datagram

Example: UDPClient.java

- A simple UDP client which reads input from keyboard, sends the input to server, and reads the reply back from the server.



Example: Java client (UDP)

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

class UDPClient {
   public static void main(String args[]) throws Exception
   {
     BufferedReader inFromUser =
       new BufferedReader(new InputStreamReader(System.in));
     String sentence = inFromUser.readLine();
     byte[] sendData = sentence.getBytes();

     DatagramSocket clientSocket = new DatagramSocket();

     InetAddress sIPAddress = InetAddress.getByName("servname");
```

Create input stream

Create client socket

Translate hostname to IP address using DNS

Example: Java client (UDP), cont.

Create datagram with data-to-send, length, IP addr, port

Send datagram to server

Read datagram from server

```
DatagramPacket sendPacket =
   new DatagramPacket(sendData, sendData.length, sIPAddress, 9876);

clientSocket.send(sendPacket);

byte[] receiveData = new byte[1024];
DatagramPacket receivePacket =
   new DatagramPacket(receiveData, receiveData.length);

clientSocket.receive(receivePacket);

String modifiedSentence =
    new String(receivePacket.getData());

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

}
```

Demo

%mac: java UDPServer
% netstat to see buffer

%cicada: java UDPClient <server>

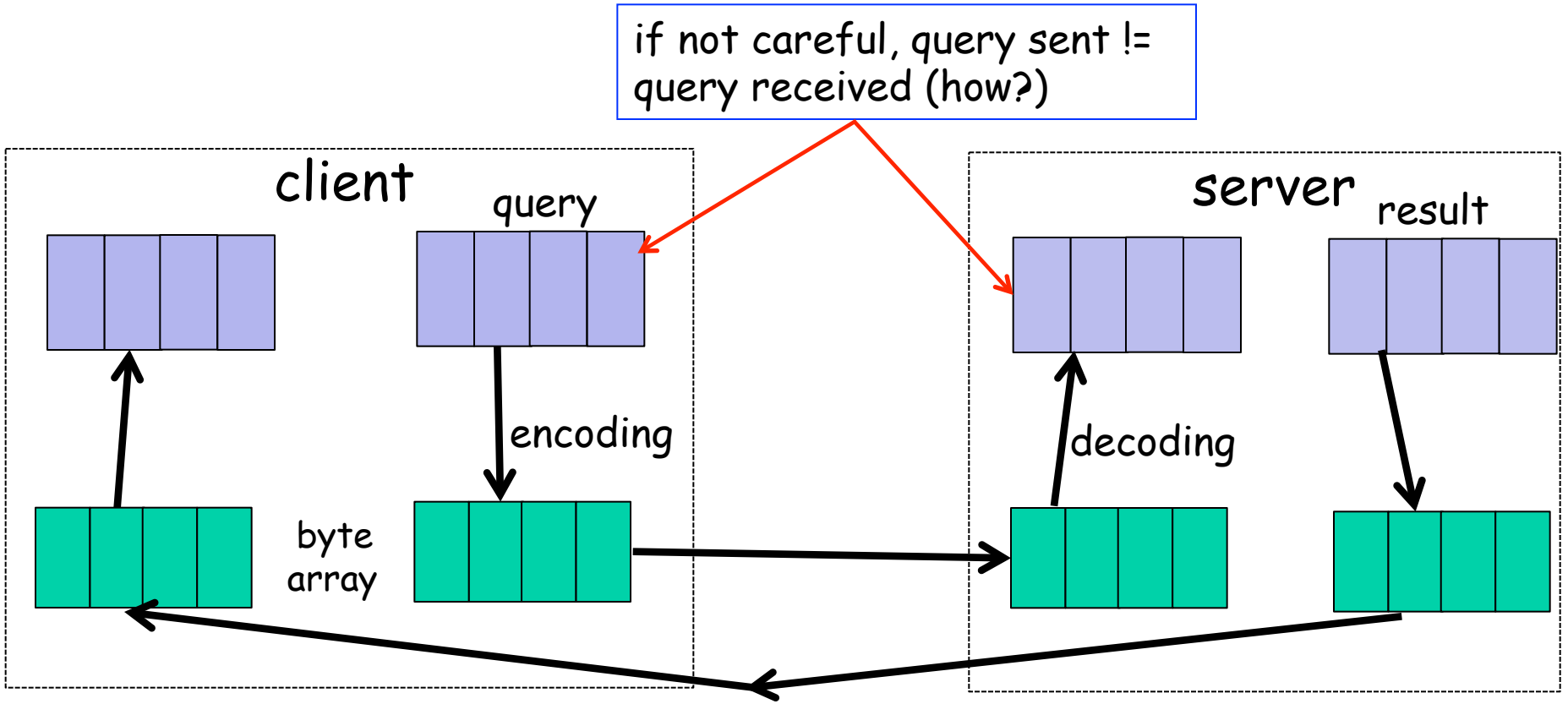
% wireshark to capture traffic

Discussion on Example Code

- A simple upper-case UDP echo service is among the simplest network service.
- Are there any problems with the program?

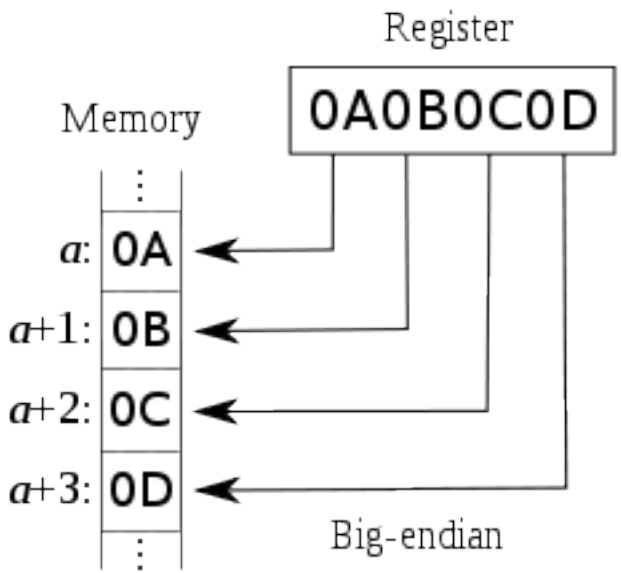
Data Encoding/Decoding

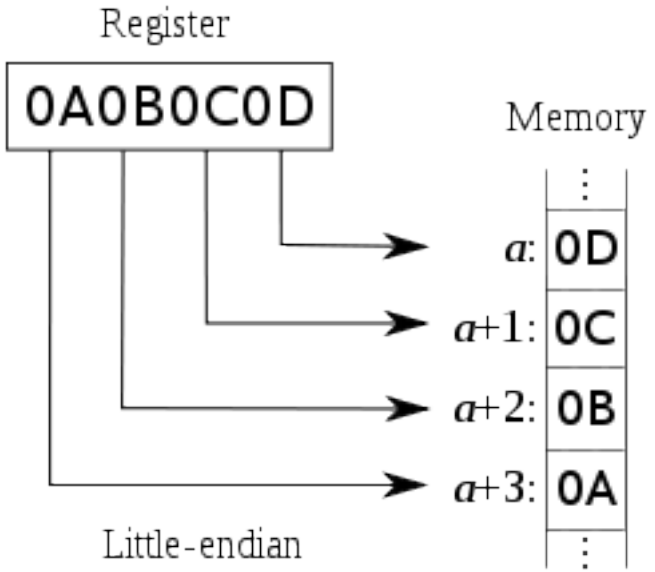
- Pay attention to encoding/decoding of data: transport layer handles only a sequence of bytes



Example: Endianness of Numbers

- int var = 0x0A0B0C0D

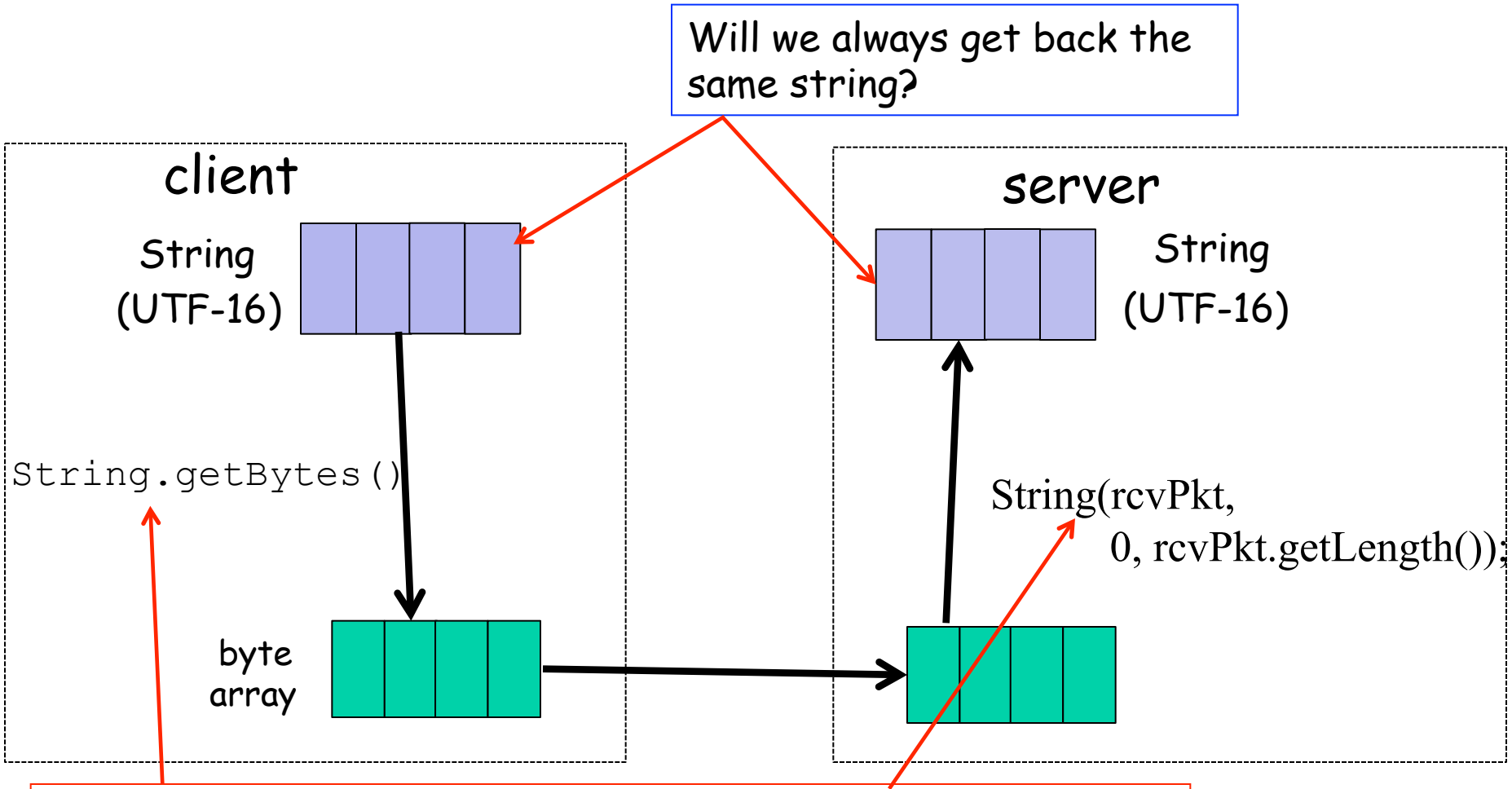




ARM, Power PC, Motorola 68k, IA-64

Intel x86

Example: String and Chars



Depends on default local platform char set (why?) : java.nio.charset.Charset.defaultCharset()

Example: Charset Troubles

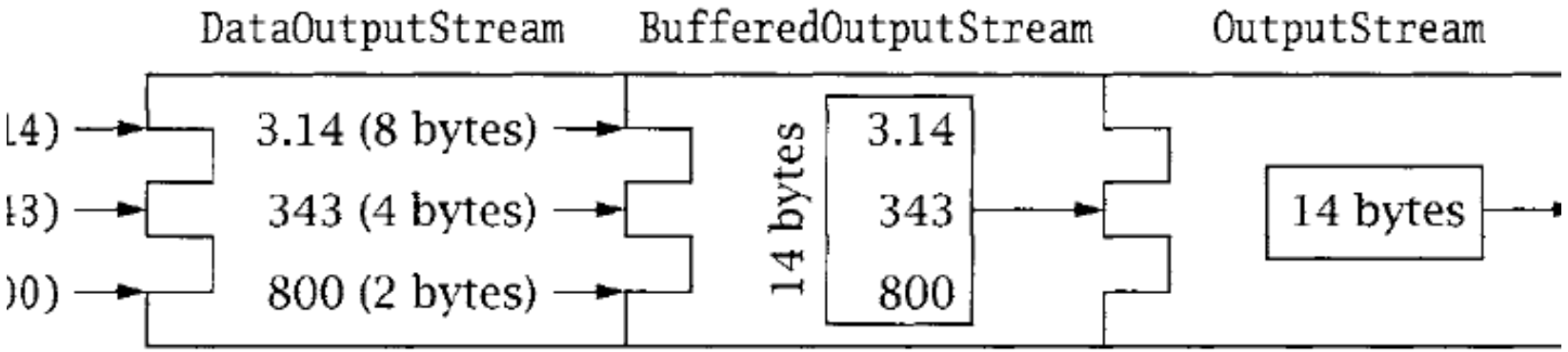
- Try
  - java EncodingDecoding US-ASCII UTF-8

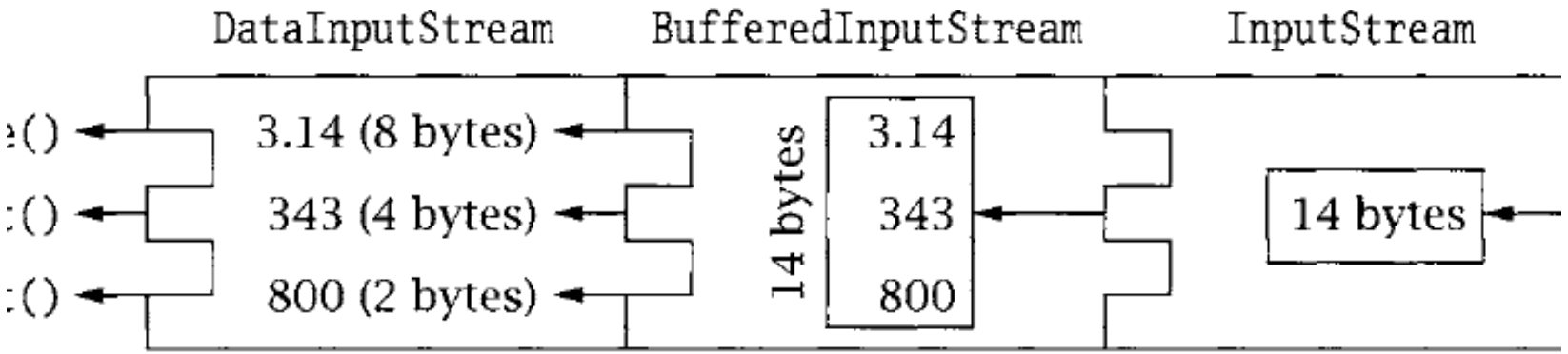
Encoding/Decoding as a Common Source of Errors

- Please read chapter 4 of Java Network Programming for more details

- Common mistake even in many (textbook) examples:
  - http://www.java2s.com/Code/Java/Network-Protocol/UseDatagramSockettosendoutandreceiveDatagramPacket.htm

DataStream


DataOutputStream
BufferedOutputStream
OutputStream
3.14 (8 bytes)
343 (4 bytes)
800 (2 bytes)
14 bytes
3.14
343
800
14 bytes


DataInputStream
BufferedInputStream
InputStream
3.14 (8 bytes)
343 (4 bytes)
800 (2 bytes)
14 bytes
3.14
343
800
14 bytes