Socket Programming

A note on the use of these ppt slides:

We're making these slides freely available to all (faculty, students, readers). They're in PowerPoint form so you see the animations; and can add, modify, and delete slides (including this one) and slide content to suit your needs. They obviously represent a *lot* of work on our part. In return for use, we only ask the following:

❖ If you use these slides (e.g., in a class) that you mention their source (after all, we'd like people to use our book!)

❖ If you post any slides on a www site, that you note that they are adapted from (or perhaps identical to) our slides, and note our copyright of this material.
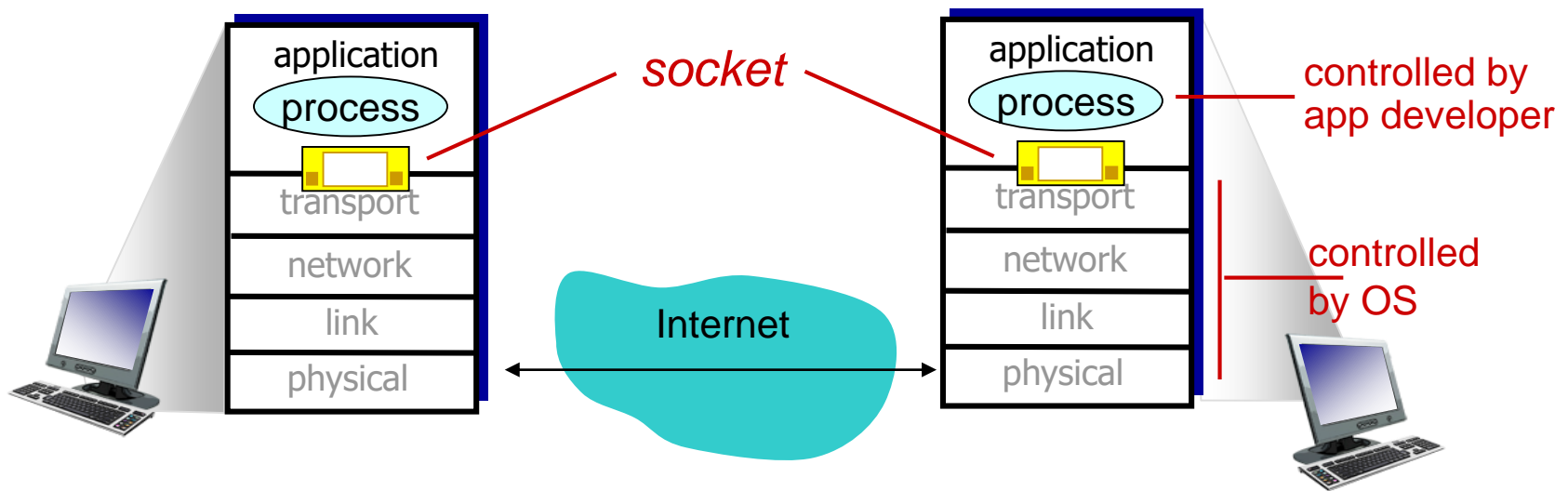
Thanks and enjoy! JFK/KWR

Computer Networking: A Top Down Approach
6th edition
Jim Kurose, Keith Ross
Addison-Wesley
March 2012

Socket programming

goal: learn how to build client/server applications that communicate using sockets

socket: door between application process and end-end-transport protocol

Socket programming

Two socket types for two transport services:

- *UDP:* unreliable datagram
- *TCP:* reliable, byte stream-oriented

Application Example:

1. Client reads a line of characters (data) from its keyboard and sends the data to the server.
2. The server receives the data and converts characters to uppercase.
3. The server sends the modified data to the client.
4. The client receives the modified data and displays the line on its screen.

Socket programming *with UDP*

UDP: no "connection" between client & server

❖ no handshaking before sending data

❖ sender explicitly attaches IP destination address and port # to each packet

❖ rcvr extracts sender IP address and port# from received packet

UDP: transmitted data may be lost or received out-of-order

Application viewpoint:

❖ UDP provides *unreliable* transfer of groups of bytes ("datagrams") between client and server

Categories of Ports

Well known ports	0 - 1023	Can only be used by system (or root) processes or by programs executed by privileged users
Registered ports	1024 - 49151	Can be used by ordinary user processes
Dynamic or private ports	49152 - 65535	Free for anyone to use

Socket programming *with TCP*

client must contact server

- ❖ server process must first be running
- ❖ server must have created socket (door) that welcomes client's contact

client contacts server by:

- ❖ Creating TCP socket, specifying IP address, port number of server process
- ❖ *when client creates socket:* client TCP establishes connection to server TCP

❖ when contacted by client, *server TCP creates new socket* for server process to communicate with that particular client

- ▪ allows server to talk with multiple clients
- ▪ source port numbers used to distinguish clients (more in Chap 3)

application viewpoint:

TCP provides reliable, in-order byte-stream transfer ("pipe") between client and server