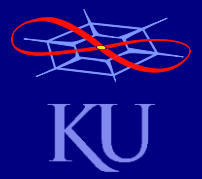
Introduction to Communication Networks
The University of Kansas EECS 563
Introduction to Socket Programming

Trúc Anh N. Nguyễn,
Egemen K. Çetinkaya, Mohammed Alenazi, and
James P.G. Sterbenz



Department of Electrical Engineering & Computer Science
Information Technology & Telecommunications Research Center
The University of Kansas

jpgs@eecs.ku.edu

https://www.ittc.ku.edu/~jpgs/courses/intronets

Socket Programming
Outline

SP.1 Motivation and overview
SP.2 Socket programming stages
SP.3 Socket programming examples
SP.4 Socket programming assignment

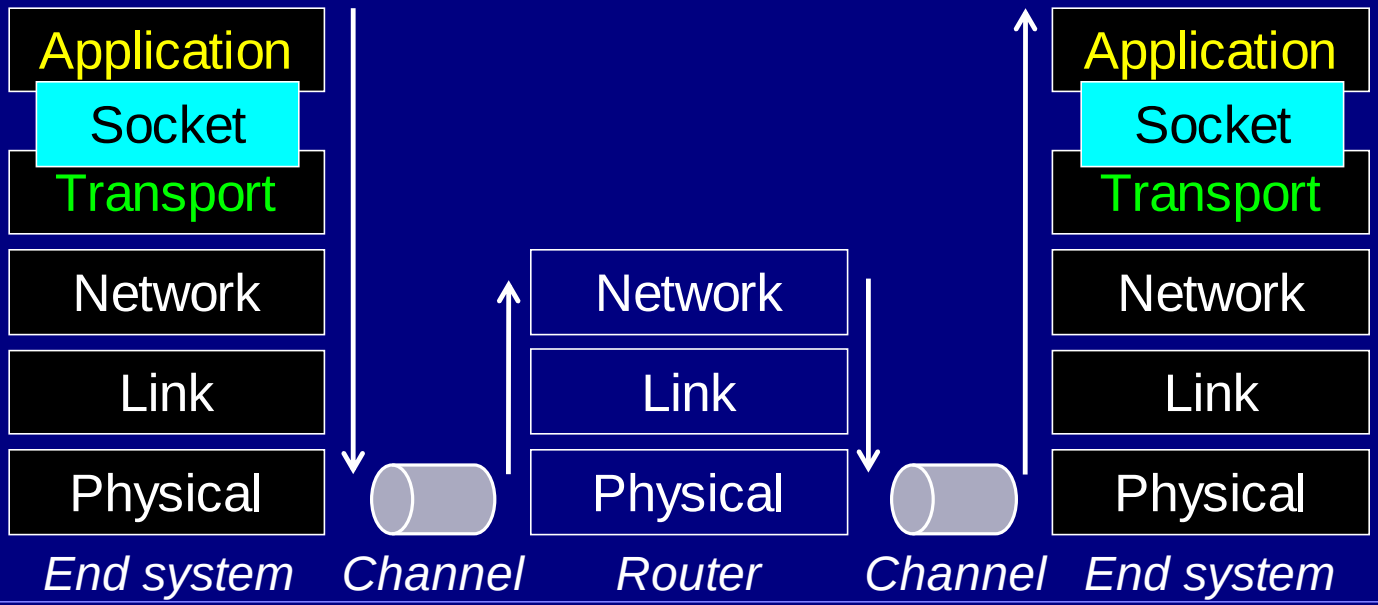
Motivation and Overview
Socket Programming and Applications

- Introduced in 4.1 BSD UNIX 1982
- Network application implementations
  - standard network application
    - based on RFCs (e.g. TCP and UDP)
    - port numbers 0–1023 should conform to IANA registry `https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml`
  - proprietary network application
    - don't conform to RFCs
    - use registered port numbers 1024–49151

      or
    - use dynmic port numbers 49152–65535

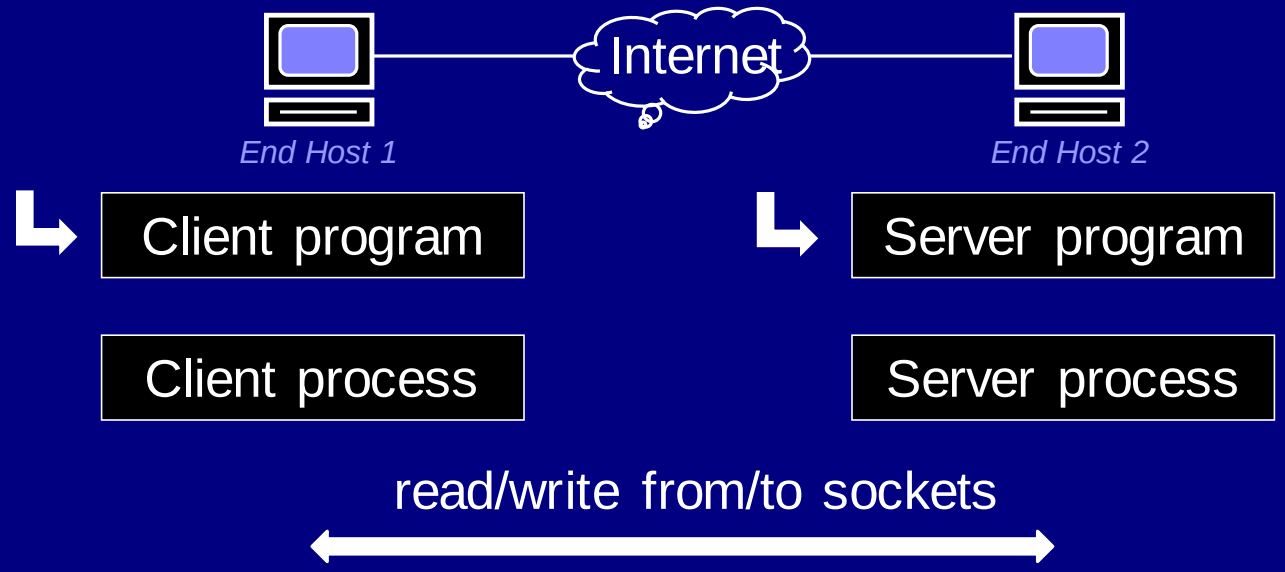
Motivation and Overview
E2E Application Data Flow and Sockets

- Application process send messages via sockets
  - controlled by the developer
- Transport layer (TCP,UDP) is controlled by the OS



Motivation and Overview
Sockets and Processes

- Socket is a method for interprocess communication
- Processes are created via client/server programs
- IPC can be done on a single host as well



Socket Programming
Socket Programming Stages

SP.1 Motivation and overview
SP.2 Socket programming stages
SP.3 Socket programming examples
SP.4 Socket programming assignment

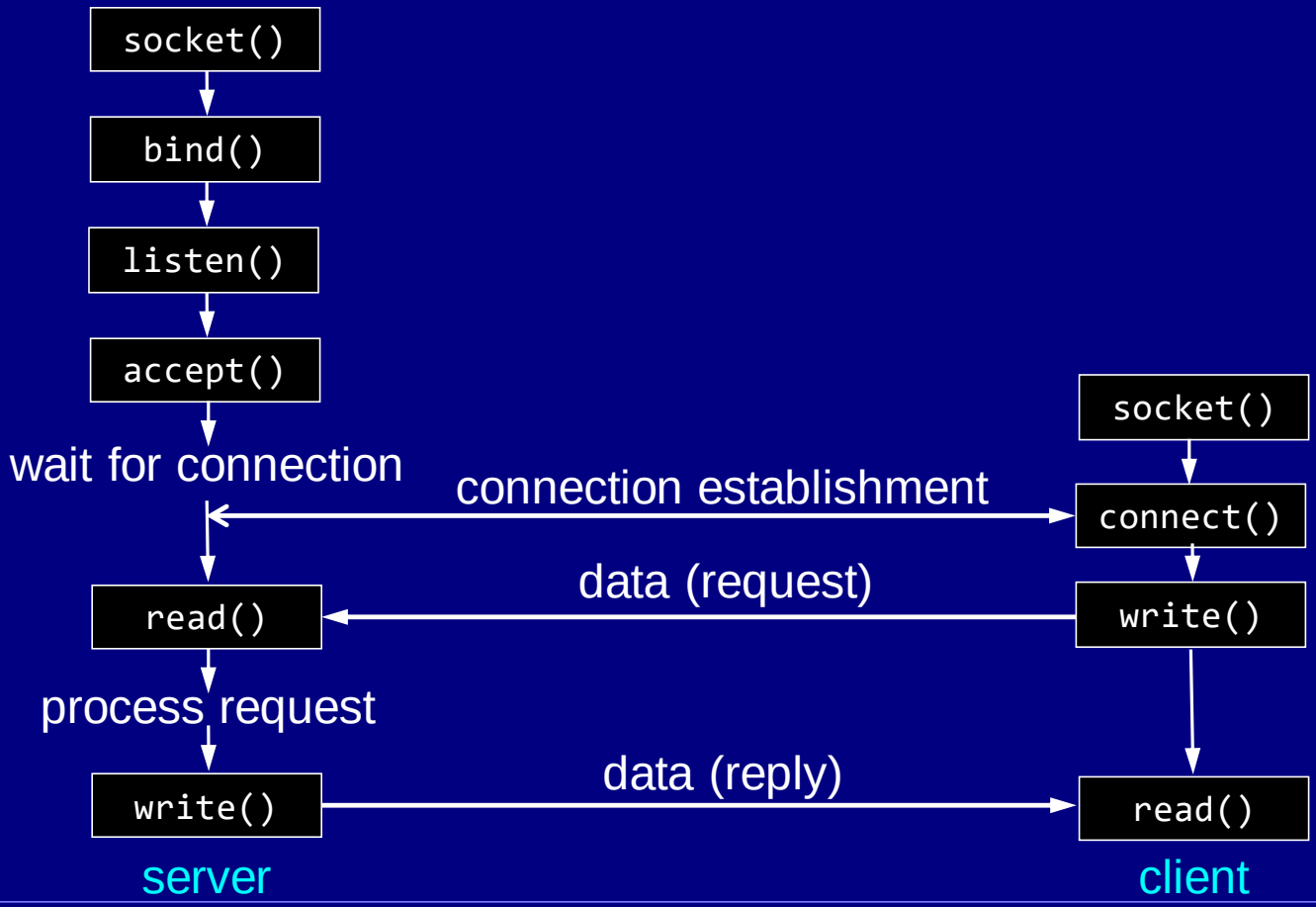
Socket Programming Stages
Planning Phase

1. Developer decides programming language and OS
   - Python, C, Java etc. and UNIX, Linux, MS etc.
2. Developer should decide whether:
   - to run the application on TCP
     - TCP is connection oriented, reliable byte stream channel
   - to run the application on UDP
     - UDP is connectionless service, best effort, no guarantee
   - skip transport layer to run the application over IP
     - e.g. ICMP or on routers
     - also called raw sockets
3. Developer implements the code

Socket Programming Stages
Socket Programming with TCP

- In order to establish connect. between client & server
- Server process
  - has to be ready to respond client's requests
  - server has to have a welcoming socket
- Client process
  - creates socket
  - specifies the destination
  - 3-way handshake occurs
    - client invokes server's `welcomeSocket accept()` method
    - server responds this by creating dedicated `connectionSocket`
    - TCP establishes pipe betw. `connectionSocket-clientSocket`

Socket Programming Stages
Connection-Oriented Flow Diagram

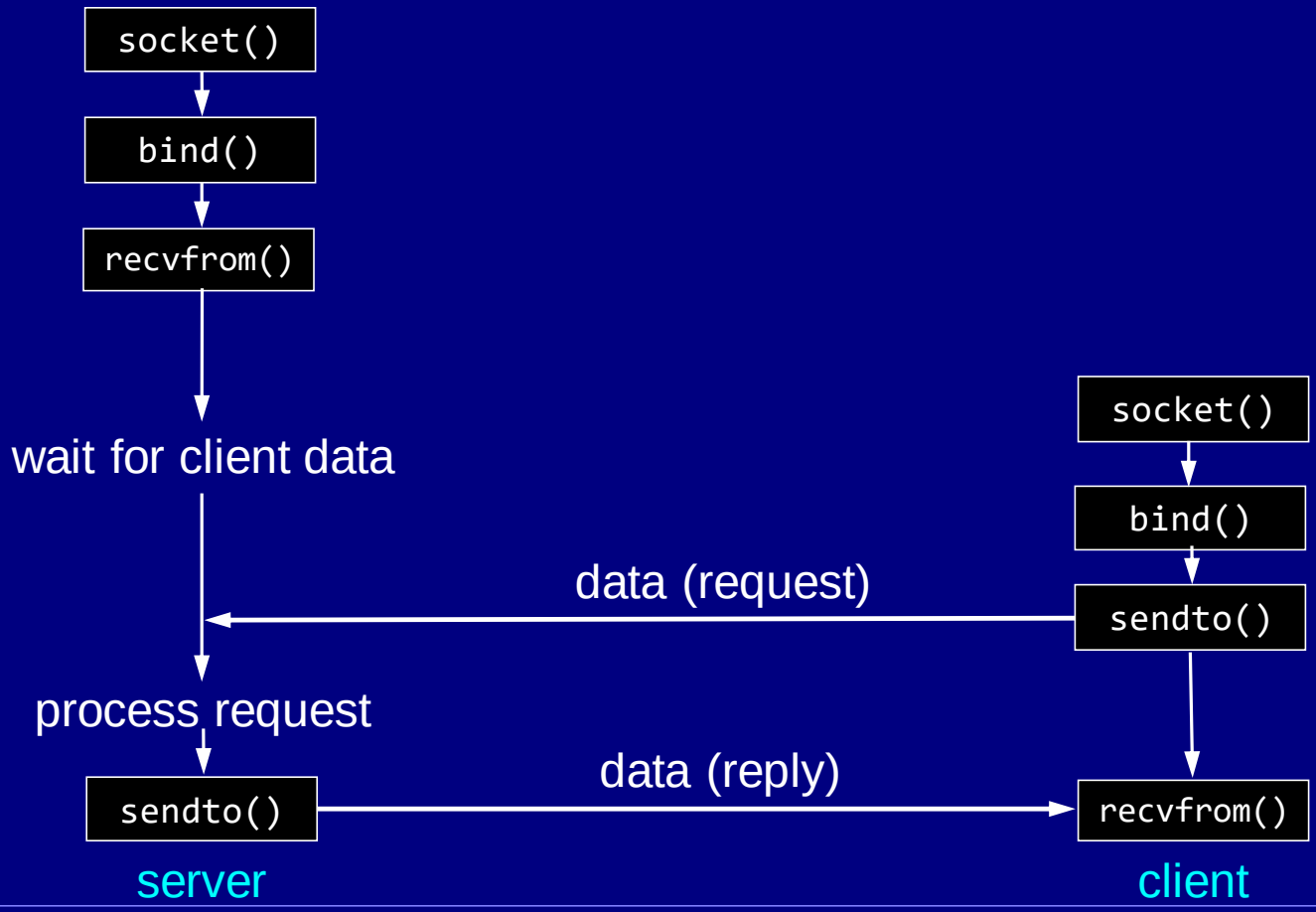


[Stevens-1990]

Socket Programming Stages
Socket Programming with UDP

- Connectionless transport between client & server
  - no initial handshaking
  - unlike TCP client can be started first
- Client attaches destination address to each packet
- Client process
  - creates `clientSocket` of type `DatagramSocket`
  - in TCP `clientSocket` is of type `Socket`
- Server process
  - creates `serverSocket` of type `DatagramSocket`
  - no `welcomeSocket` as in TCP

Socket Programming Stages
Connectionless Flow Diagram



[Stevens-1990]

Socket Programming
Socket Programming Examples

SP.1 Motivation and overview
SP.2 Socket programming stages
SP.3 Socket programming examples
SP.4 Socket programming assignment

Socket Programming
Socket Programming Examples

- Simple client-server application using TCP and UDP
  - client sends a message to the server
  - server echoes the message back to the client
  - program is written in Python 2.7.13

Socket Programming Examples
Python TCP Client

```
from socket import *

serverIP = '127.0.0.1'
serverPort = 5005
bufferSize = 1024
message = "Hello, World!"

print "TCP server IP address:", serverIP
print "TCP server port number:", serverPort
print "Message to be sent to server:", message

clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverIP, serverPort))
clientSocket.send(message)
recvMessage = clientSocket.recv(bufferSize)
print “Message received from server:", recvMessage

clientSocket.close()
```

Socket Programming Examples
Python TCP Server

```
from socket import *

serverPort = 5005
bufferSize = 1024

serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind(('', serverPort))
serverSocket.listen(1)
print('The server is ready to receive')

while True:
    connectionSocket, addr = serverSocket.accept()
    message = connectionSocket.recv(bufferSize)
    print "Message received from client:", message
    connectionSocket.send(message)  # echo
    connectionSocket.close()
```

Socket Programming Examples
Python UDP Client

```
from socket import *

serverIP = "127.0.0.1"
serverPort = 5005
bufferSize = 1024
message = "Hello, World!"

print "UDP server IP address:", serverIP
print "UDP server port number:", serverPort
print "Message to be sent to server:", message

clientSocket = socket(AF_INET, # Internet
                      SOCK_DGRAM) # UDP
clientSocket.sendto(message, (serverIP, serverPort))
recvMessage, serverAddress = clientSocket.recvfrom(bufferSize)
print "Message received from server:", recvMessage
clientSocket.close()
```

Socket Programming Examples
Python UDP Server

```
from socket import *

serverPort = 5005
bufferSize = 1024

serverSocket = socket(AF_INET, # Internet
                      SOCK_DGRAM) # UDP
serverSocket.bind(('', serverPort))
print('The server is ready to receive')

while True:
    message, clientAddress = serverSocket.recvfrom(1024)
    print "Message received from client:", message
    serverSocket.sendto(message, clientAddress)
```

Socket Programming
Lab Report Submission Requirement

SP.1 Motivation and overview
SP.2 Socket programming stages
SP.3 Socket programming examples
SP.4 Socket programming assignment

Socket Programming Assignment
Program Output Requirement

- Create an application that will
  - convert lowercase letters to uppercase
    - e.g. [a…z] to [A…Z]
    - code will not change any special characters, e.g. &*!
  - if the character is in uppercase , the program must not alter
- Create socket API both for
  - reliable byte stream
  - datagram services
- Must take the server address and port from the CLI

Socket Programming Assignment
Example for UDP

- Server side:
```
$ python LastName_UDP_server.py  5050
```
- Client side:
```
$ python LastName_UDP_server.py 127.0.0.1 5050
$ Please enter the statement: eecs563
$ Return text from the server: EECS563
$ Please enter the statement:
```

- Program should keep asking user for input
  - until forced to exit, e.g ctrl-c

Socket Programming Assignment
Questions to Answer

- What are example applications using TCP and UDP?
  - give two examples for each protocol
  - what are the port numbers those applications use?
  - use examples from well-known port number range
- In the example:
  - why the UDP server needs only one socket, whereas the TCP server needs two sockets?
  - if the TCP server needs to support *n* simultaneous connections, each from a different client host, how many sockets would the TCP server need?

Socket Programming Assignment
Extra Credit

- For client/server programs
  - print local and foreign address using functions
- Run client and server programs on different machines
- In the UDP client program on Foil 16:
  - suppose we add the following line after we create the socket

    ```
    clientSocket.bind(('', 5432))
    ```
  - will it become necessary to change the UDP server program on Foil 16?
  - what are the port numbers for the sockets in the UDP client and UDP server programs?
  - what were the port numbers before making this change?

Socket Programming Assignment
Submission Requirements

- Use Python
- Your submission should include:
  - printout of client program for datagram service
  - printout of server program for datagram service
  - printout of client program for reliable byte-stream service
  - printout of server program for reliable byte-stream service
  - screenshots of your test cases
  - your answers to additional questions
    - and optional extra credit
- Retain your source files
  - in case you are asked to demonstrate your code

Socket Programming
Acknowledgements

Some material in these foils comes from:

- Kurose & Ross,
  Computer Networking:
  A Top-Down Approach, 7th ed.
  `http://www-net.cs.umass.edu/kurose-ross-ppt-7e/`
- Stevens,
  UNIX Network Programming
  Prentice Hall, 1990
  `http://www.kohala.com/start/unp.html`

Socket Programming
Acknowledgements

- Donahoo & Calvert,
  TCP/IP Sockets in C: Practical Guide for Programmers, 2nd ed.
  http://cs.ecs.baylor.edu/~donahoo/practical/CSockets/

- Donahoo & Calvert,
  TCP/IP Sockets in Java: Practical Guide for Programmers, 2nd ed.
  Morgan Kaufmann, 2008
  http://cs.ecs.baylor.edu/~donahoo/practical/JavaSockets/

- Python
  http://wiki.python.org/moin/TcpCommunication
  http://wiki.python.org/moin/UdpCommunication

Socket Programming
Additional Reading

- http://www.cs.utsa.edu/~korkmaz/teaching/cn-resources/programs/capitalize-tcp/
- http://www.cs.utsa.edu/~korkmaz/teaching/cn-resources/programs/capitalize-udp/
- http://gaia.cs.umass.edu/ntu_socket/
- http://beej.us/guide/bgnet/
- http://java.sun.com/docs/books/tutorial/networking/sockets/
- http://www.faqs.org/faqs/unix-faq/socket/
- http://www.iana.org/assignments/port-numbers
- http://docs.freebsd.org/44doc/psd/20.ipctut/paper.pdf
- http://docs.freebsd.org/44doc/psd/21.ipc/paper.pdf