Communication Networks
The University of Kansas EECS 780
Introduction to Socket Programming

Anh Nguyen, Mohammed J.F. Alenazi, Egemen K. Çetinkaya, and James P.G. Sterbenz

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

jgps@eecs.ku.edu

http://www.ittc.ku.edu/~jgps/courses/nets
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 dynamic 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

Diagram:

```
Application
    | Socket
    | Transport
Network
Link
Physical
End system
Channel
Router
Channel
End system
```
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

Client program → Server program

Client process → Server process

read/write from/to sockets
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 connection 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 between `connectionSocket-clientSocket`
Socket Programming Stages

Connection-Oriented Flow Diagram

server

wait for connection
socket()
bind()
listen()
accept()

connection establishment

process request
read()
write()

data (request)
write()
read()

connection establishment
socket()
connect()

data (reply)
write()
read()

client

[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

socket()
bind()
recvfrom()

wait for client data

process request

sendto()

server

[Stevens-1990]

socket()
bind()
sendto()
recvfrom()

data (request)
data (reply)

client
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

```python
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

```python
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()
```
from socket import *

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

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

clientSocket = socket(AF_INET, # Internet
                      SOCK_DGRAM) # UDP
clientSocket.sendto(message, (serverIP, servPort))
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
• What is the largest datagram size can you send and receive using UDP socket?
  – verify it experimentally
Socket Programming Assignment

Extra Credit

• In the UDP client program on Foil 17:
  – 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

• Follow instructions on submission requirements page
  – http://www.ittc.ku.edu/~jpgs/courses/homework.html

• You need to submit:
  – client file for reliable byte-stream service
  – server file for reliable byte-stream service
  – client file for datagram service
  – server file for datagram service
  – your test cases (screenshots in a single PDF file)
  – PDF file for additional questions

• Use Python
Socket Programming Assignment

Submission Requirements

- Folder name: LastName_socket_programming
- Folder contains:
  - LastName_TCP_server.py
  - LastName_TCP_client.py
  - LastName_UDP_server.py
  - LastName_UDP_client.py
  - LastName_test_cases.pdf
  - LastName_report.pdf
- Subject: EECS780 – socket programming
- attachment: LastName_socket_programming.zip
  - zip the whole folder
Socket Programming

Acknowledgements

Some material in these foils comes from the textbook supplementary materials:

  https://media.pearsoncmg.com/bc/abp/cs-resources/products/product.html#product,isbn=0133594149

- Stevens, *UNIX Network Programming*
  *Prentice Hall, 1990*
  http://www.kohala.com/start/unp.html
Socket Programming

Acknowledgements


- Python.org http://wiki.python.org/moin/TcpCommunication
- Python.org http://wiki.python.org/moin/UdpCommunication
Some material in these foils comes from the textbook supplementary materials:

- 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.iana.org/assignments/port-numbers