Socket Programming

Outline

L2.1  Motivation and overview
L2.2  Socket programming stages
L2.3  Socket programming examples
L2.4  Socket programming assignment
Socket Programming
Motivation and Overview

L2.1  Motivation and overview
L2.2  Socket programming stages
L2.3  Socket programming examples
L2.4  Socket programming assignment
Motivation and Overview
Socket Programming and Applications

• Also called client/server application development
• Introduced in 4.1 BSD in 1982
• Network application implementations
  – standard network application
    • based on RFCs
    • programs conform rules
    • port numbers should be implemented per RFC
  – proprietary network application
    • they don’t conform RFCs
    • code will not interoperate
    • don’t use standard RFC well-known port numbers (0-1023)
Motivation and Overview
E2E Application Data Flow and Sockets

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

Diagram:
- Application
- Socket
- Transport
- Network
- Link
- Physical
- End host
- Channel
- Router
- Network
- Link
- Physical
- End host
Motivation and Overview

Sockets and Processes

• Socket is a method for Inter Process Communication
• Processes are created via client/server programs
• IPC can be done on a single host as well

Internet

End Host 1

End Host 2

Client program

Server program

Client process

Server process

read/write from/to sockets
Socket Programming

Socket Programming Stages

L2.1  Motivation and overview
L2.2  Socket programming stages
L2.3  Socket programming examples
L2.4  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:
   - 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
     - for hop nodes: e.g. ICMP
     - 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

socket() → bind() → listen() → accept() → wait for connection → connection establishment → read() → process request → write() → server

socket() → connect() → write() → data (request) → read() → client

data (reply) → write() → process request → read() → connection establishment → listen() → accept() → bind() → socket() → wait for connection

[Stevens-1990]
Socket Programming Stages

Socket Programming with UDP

- Connectionless transport between client & server
  - there is 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`
  - there is no `welcomeSocket` as in TCP
Socket Programming Stages

Connectionless Flow Diagram

socket() → bind() → recvfrom() → wait for client data → process request → sendto() → server

socket() → bind() → sendto() → data (request) → recvfrom() → data (reply) → client

[Stevens-1990]
Socket Programming
Socket Programming Examples

L2.1 Motivation and overview
L2.2 Socket programming stages
L2.3 Socket programming examples
L2.4 Socket programming assignment
Socket Programming Examples
Python TCP Client

```python
code:
import socket
TCP_IP = '127.0.0.1'
TCP_PORT = 5005
BUFFER_SIZE = 1024
MESSAGE = "Hello, World!"

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((TCP_IP, TCP_PORT))
s.send(MESSAGE)
data = s.recv(BUFFER_SIZE)
s.close()

print "received data:", data
```
import socket
TCP_IP = '127.0.0.1'
TCP_PORT = 5005
BUFFER_SIZE = 1024
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind((TCP_IP, TCP_PORT))
s.listen(1)

conn, addr = s.accept()
print 'Connection address:', addr
while 1:
    data = conn.recv(BUFFER_SIZE)
    if not data: break
    print "received data:'", data
    conn.send(data)  # echo
conn.close()
import socket

UDP_IP = "127.0.0.1"
UDP_PORT = 5005
MESSAGE = "Hello, World!"

print "UDP target IP:", UDP_IP
print "UDP target port:", UDP_PORT
print "message:", MESSAGE

sock = socket.socket(socket.AF_INET, # Internet
                      socket.SOCK_DGRAM) # UDP
sock.sendto(MESSAGE, (UDP_IP, UDP_PORT))
import socket

UDP_IP = "127.0.0.1"
UDP_PORT = 5005

sock = socket.socket(socket.AF_INET, # Internet
    socket.SOCK_DGRAM) # UDP
sock.bind((UDP_IP, UDP_PORT))

while True:
    data, addr = sock.recvfrom(1024) # buffer size is 1024 bytes
    print "received message:", data
Socket Programming
Lab Report Submission Requirement

L2.1 Motivation and overview
L2.2 Socket programming stages
L2.3 Socket programming examples
L2.4 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]
    - the code will not change any special characters
      - e.g. &*!
    - if the character is in uppercase, the program will not alter it

- 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_client.py 127.0.0.1 5050
  $ Please enter the statement: eecs780
  $ Return text from the server: EECS780
  $ 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 example for each protocol
  – what are the port numbers those applications use?
  – use examples from well-known port number range

• In your application:
  – can you use those port numbers? explain why or why not
  – how many simultaneous client/server connections possible for each transport protocol?
  – are port # same for given protocol for each connection?
  – support you answer with `netstat` command output
Socket Programming Assignment
Extra Credit

• For client/server programs
  – print local and foreign address using functions
• What is the largest datagram size can you send and receive using UDP socket?
  – verify it experimentally
• Run client and server programs on different machines
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
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

• Donahoo & Calvert,
  [http://cs.ecs.baylor.edu/~donahoo/practical/CSockets/](http://cs.ecs.baylor.edu/~donahoo/practical/CSockets/)

• Donahoo & Calvert,
  Morgan Kaufmann, 2008

• Python.org [http://wiki.python.org/moin/TcpCommunication](http://wiki.python.org/moin/TcpCommunication)

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