SOCKET编程实验

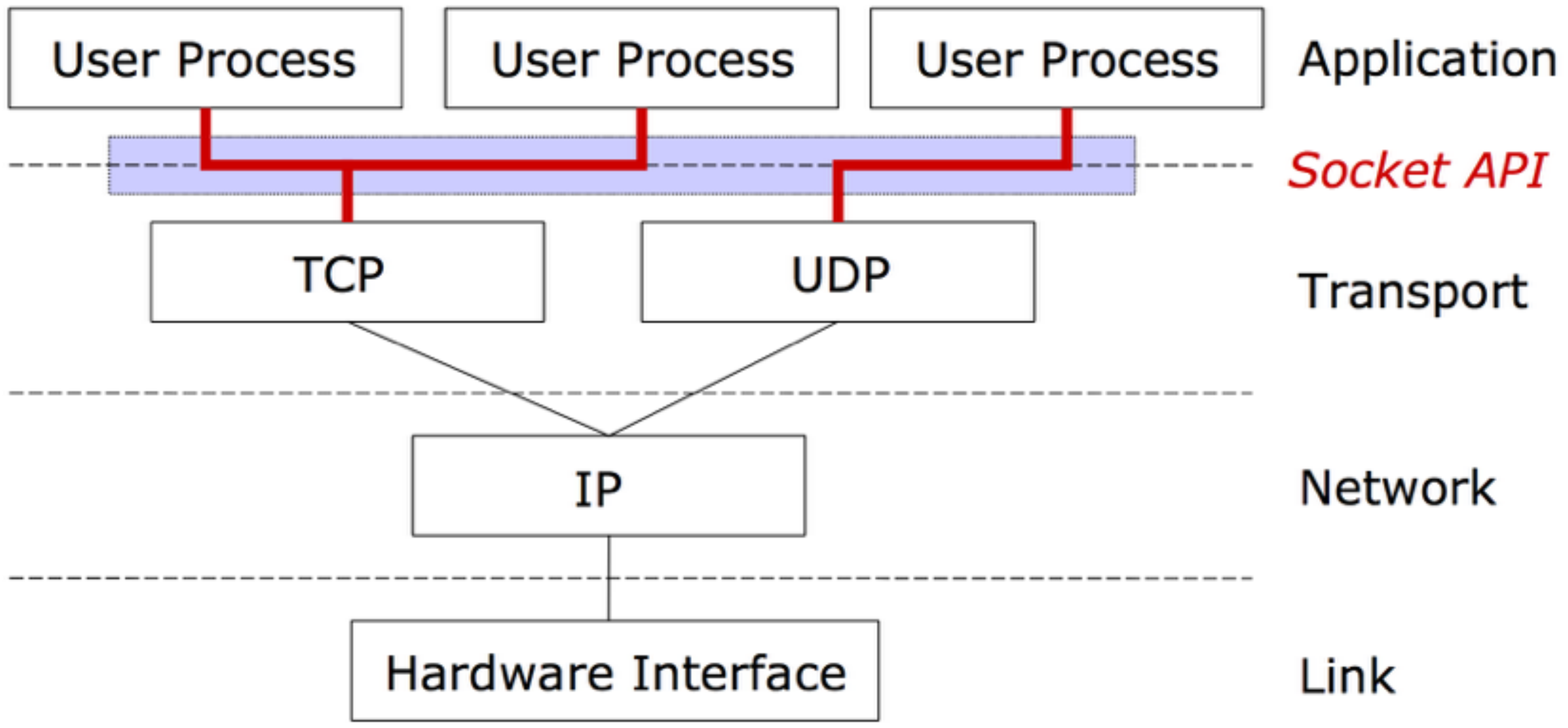
设计一个具体的协议（建议是应用层协议），采用标准Socket API编程来实现协议的功能

- 协议的设计可以参考http等，语法语义相似
- 头部域自己设计，需要有完备的功能，服务器端要能理解客户端的各种请求，并有一定的错误处理机制
- 如无特殊情况，要求使用Python编程
- 如无特殊情况，不能使用额外封装的库

TO WRITE A NETWORK PROGRAM AT USER-LEVEL

SOCKET

SOCKETS AND THE TCP/IP SUITE



MAKE A RAW HTTP CONNECTION

```
#rawConn - making raw connection to Google Maps

import httplib
import json
import urllib #for encode URL

path = ('/maps/api/geocode/json?address=%s&sensor=false&region=%s') % \
        (urllib.quote('Fudan University', safe=''), \
         urllib.quote('Shanghai', safe=''))

#make a HTTP connection and send GET request
connection = httplib.HTTPConnection('maps.google.com')
connection.request('GET', path)

#get the JSON response and paste it
rawreply = connection.getresponse().read()
reply = json.loads(rawreply)

lat = reply['results'][0]['geometry']['location']['lat']
lng = reply['results'][0]['geometry']['location']['lng']

print lat, lng
```

SOCKET

- Instead of making an HTTP request through *httplib* package
    - We can make the request by using *socket*
- *socket* – support basic network communications on an IP network
    - Low-level approach
- Raw network communication
    - A matter of sending and receiving strings

SOCKET

- The layers below the *socket()* are:
  - Transmission Control Protocol (TCP)

    One alternative to TCP is UDP
  - Internet Protocol (IP)
  - Link layer

MAKE A REQUEST BY SOCKET

```python
#socConn - make a request by using socket
import socket

sock = socket.socket()
sock.connect(('maps.google.com', 80)) #port number 80 - HTTP(web)

sock.sendall(
    'GET /maps/api/geocode/json?address=Fudan%20University'
    '&sensor=false&region=Shang%20Hai HTTP/1.1\r\n'
    'Host: maps.google.com:80\r\n'
    'User-Agent: socConn.py\r\n'
    'Connection: close\r\n'
    '\r\n'
    )
rawreply = sock.recv(4096)
print rawreply
```

SOCKET

- Used to identify particular processes (programs) on particular machines
- Network Socket is composed of 3 parameters:
  - Protocol

    TCP, UDP etc.
  - IP address – machine identifier
  - Port number – process identifier

    Well-known ports: 25 – SMTP (email), 80 - HTTP (web), 110 – POP3 (email), 443 – HTTPS (secure web)

EXAMPLE

A SIMPLE TCP CLIENT AND SERVER

A SIMPLE TCP CLIENT AND SERVER

```
import socket, sys
s = socket.socket(socket.AF_INET,
socket.SOCK_STREAM)

HOST = '127.0.0.1' #localhost
PORT = 8888

def recv_all(sock, length):
    data = ''
    while len(data) < length:
        more = sock.recv(length - len(data))
        if not more:
            raise EOFError('socket closed %d
bytes into a %d-byte message’  % (len(data),
length))
        data += more
    return data

s.connect((HOST, PORT))
print 'Client has been assigned socket name',
s.getsockname()
s.sendall('Hello!! Server!!') #16 characters
reply = recv_all(s, 16)
print 'The server said', repr(reply)
s.close
```

```
import socket, sys
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

HOST = '127.0.0.1' #localhost
PORT = 8888

def recv_all(sock, length):
    data = ''
    while len(data) < length:
        more = sock.recv(length - len(data))
        if not more:
            raise EOFError('socket closed %d bytes into a %d-
byte message’  % (len(data), length))
        data += more
    return data

s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
s.bind((HOST, PORT))
s.listen(1) #listen 1 client only
while True:
    print 'Listening at', s.getsockname()
    sc, sockname = s.accept() #wait here until there is a
request
    print 'We have accepted a connection from ', sockname
    print 'Socket connects', sc.getsockname(), 'and',
sc.getpeername()
    message = recv_all(sc, 16)
    print 'The incoming 16-octet message says', repr(message)
    sc.sendall('Bye Bye Client..') #16 characters
    sc.close()
    print 'Reply sent, socket closed'
```

LIMITATIONS

- At the moment, the message must be fixed in 16 characters
- Solution:
  - Client side, before sending the message
    - ① Determine the length of the message *L* [Assume max. number of length is 255]. Add *L* at the beginning of the message
    - ② Send the new message to server
  - Server side, after received the message
    - ① Extract the length of the message (first 3 characters)
    - ② Read the rest of the message with the proper length

MODIFIED TCP CLIENT AND SERVER

```python
import socket, sys
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

HOST = '127.0.0.1' #localhost
PORT = 8888

def recv_all(sock, length):
     data = ''
     while len(data) < length:
          more = sock.recv(length - len(data))
          if not more:
               raise EOFError('socket closed %d bytes
into a %d-byte message'
                                                   % (len(data),
length))
          data += more
     return data


s.connect((HOST, PORT))

msg = raw_input('Please enter your message: ')

#determine the message length (max 255 characters,
i.e. 3 digits), pad with leading zeroes
msg_length_in_str = str(len(msg))
msg_length_in_str = msg_length_in_str.zfill(3)

s.sendall(msg_length_in_str + msg) #add the length at
the beginning of the message

reply = recv_all(s, 3)
print 'Server:', repr(reply)
s.close
```

```python
import socket, sys
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

HOST = '127.0.0.1' #localhost
PORT = 8888

def recv_all(sock, length):
     data = ''
     while len(data) < length:
          more = sock.recv(length - len(data))
          if not more:
               raise EOFError('socket closed %d bytes into a %d-byte
message'
                                                   % (len(data), length))
          data += more
     return data

s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
s.bind((HOST, PORT))
s.listen(1) #listen 1 client only
while True:
     print 'Listening at', s.getsockname()
     sc, sockname = s.accept() #wait here until there is a request

     #get the length of the message first
     msg_length = int(recv_all(sc, 3))

     #get the 'real' message with proper length
     message = recv_all(sc, msg_length)
     print 'Client said: ', repr(message)
     sc.sendall('Bye')
     sc.close()
```

参考资料

- Learn Python

  https://www.codecademy.com/learn/python

- Socket in Python

  https://docs.python.org/2/howto/sockets.html

  https://docs.python.org/2/library/socket.html

参考题目1

- 设计一个类http协议
- 服务器端保存一份学生名单，包括学号、照片、姓名等。名单的存放方式随意
- 客户端针对学生名单进行各类请求，如增加，删除，查看等，每种请求通过头部字段进行具体的要求。

参考题目2

- 设计一个简单的文件传输协议
- 实现客户与服务器之间简单的文件传递，如get/put等
- 客户可以查询服务器存放文件的目录，自定义文件存放的目录等

参考题目3

- 简单的小说阅读器的设计
- 服务器端保存小说文本（txt格式的即可）
- 客户可以打开对应的文本，翻页，翻章，跳页，书签，下载，关闭等
- 建议最好有图形界面，因为是txt格式，所谓的“页”可以通过规定每次内容包含的字节来规定

参考题目4

- 简单的聊天系统
- 建议最好有图形界面

项目提交要求

- 按照规定要求自行设计
- 至多两人一组
- 上机课演示　（提前向助教报名，先到先选）
- 建议最好有图形界面
- 提交内容：
  - 代码
  - 设计文档