Python Socket Programming

CME451 Tutorial 3

Hao Zhang
(Graduate Teaching Fellow)

Department of Electrical & Computer Engineering
University of Saskatchewan

Jan 20, 2017

Lab 1 Exercise Hints

- Exercise 2.2: Extract an arbitrary substring
  - Use `str.find("\"")` to find the location of the double quotes
- Exercise 2.3: Text processing and sorting
  - Use `line.split(',')` to separate each component
  - Use `eval()` to convert sting with double quotes to numerical values
  - Use `list.sort()` to sort the list
  - Use lambda statement to define key function
  - Can also be realized by `operator.itemgetter()`

Review of Last Tutorial

Application-Layer Programming

- Introduce the `urllib.request` modules
- Use `urlopen('http address')` to open a website
  - Convert `HTTPResponse` to `string` using `read()` method
  - Apply decoding to make it readable
  - HTML contents in a large string
- Retrieve contents from web server

```
urlretrieve('server_file_location', 'local_file_name')
```

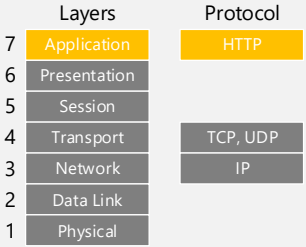
Lab 2 Objectives

Part 2: Socket Programming

- Learn the basics of socket programming
- Create a simple socket in server side
- Create a simple socket in client side
- Create web server and web client

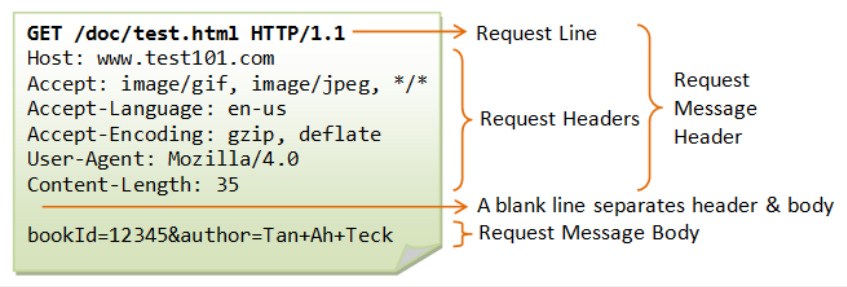
Socket Programming

Basic Concept - HTTP Protocol



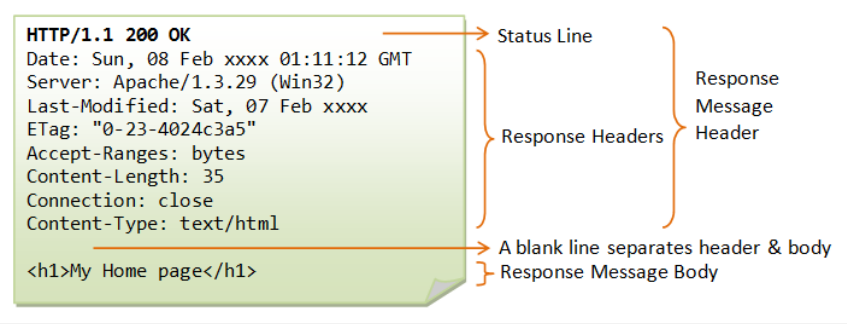
- In application-layer, we use `urllib.request` to deal with `url` website which usually uses `HTTP` protocol.
- `HTTP` (Hypertext Transfer Protocol) is a request-response protocol:
  - Client submits `request` to server.
  - Server returns `response` to client.
- `HTTP` encapsulates `request` and `response`.

- HTTP `request` format of `HTTP/1.1`
  - `request-method + request-URL + HTTP-version`



(*Figure source: `https://www.ntu.edu.sg/home/ehchua/programming/webprogramming/HTTP_Basics.html`)

- HTTP `response` format of `HTTP/1.1`
  - `HTTP-version + status code + status`
  - Note the blank line between header and body

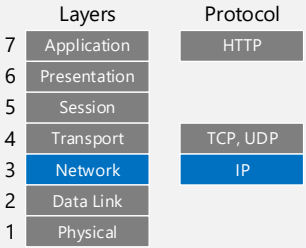


(*Figure source: `https://www.ntu.edu.sg/home/ehchua/programming/webprogramming/HTTP_Basics.html`)

- For more information of the Header and Status code, refer to:
  - `http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html`
  - `http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html`
- `HTTP` only encapsulate the messages.
- Need other protocols to transmit these message.

Socket Programming

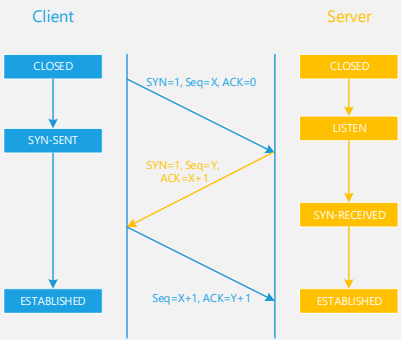
Basic Concept - TCP/IP



- `TCP/IP` is a set of transport and network layer protocols to transmit messages over network.
- `IP` (Internet Protocol) is network-layer protocol, deals with network addressing and routing.
  - In IP network, each machine is assigned a unique IP address.
  - IP software: routing a message from source IP to destination IP.
  - IPv4, e.g. `192.168.0.10`
  - IPv6, e.g. `2001:0db8:85a3:08d3:1319:8a2e:0370:7344`
  - `127.0.0.1` always refer to your own machine.
  - `localhost` can be used for local loopback testing.

Socket Programming

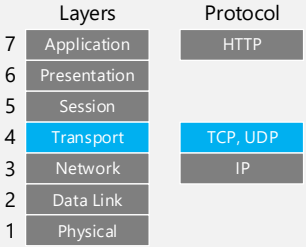
Basic Concept - TCP



- `TCP` (Transmission Control Protocol) is transport-layer protocol, responsible for establishing connection between two machines.
- TCP 3-Way Handshake:
  1. Client sends a `SYNchronize` packet to Server; Server receives client's `SYN`.
  2. Server sends a `SYN-ACKnowledgement`; Client receives server's `SYN-ACK`.
  3. Client sends `ACK`; Server receives client's `ACK`; TCP connection established.

Socket Programming

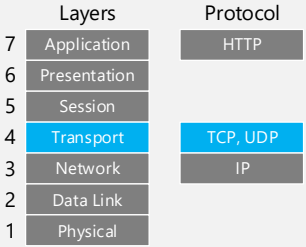
Basic Concept - TCP and UDP



- `TCP` provides a **reliable** transmission.
  - Each packet has a sequence number, and `ACK` is always expected.
  - If not received, packet will be re-transmitted.
- `UDP` (User Datagram Package) is **not reliable**.
  - Connectionless protocol.
  - No guaranteed delivery.
  - Faster and less network overhead.

Socket Programming

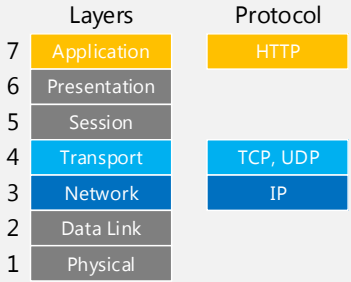
Basic Concept - TCP



- `TCP` supports up to 65536 ports (0 - 65535).
- Port 0 - 1023 are allocated to popular protocols:
  - e.g. `HTTP` at 80, `FTP` at 21, `DNS` at 53, ...
- Port 1024 and above are available to use.
  - For lab, use this range to assign port to socket.
- Although 80 is default `HTTP`, you can run a HTTP server using user available port (1024 - 65535).

Socket Programming

Basic Concept - Socket



- The APIs for us to use `TCP/IP` protocols are provided by `socket`.
- Conceptually network `socket` is an internal endpoint for sending or receiving data at a single node in a computer network.
  - It encapsulates TCP/IP protocols.
  - Socket is not a protocol.
- HTTP and Socket:
  - HTTP encapsulates messages.
  - Socket provides the connection and transmission.

- `domain`
  - The network protocol used, such as `AF_INET`, `AF_INET6`.
  - `AF_INET` for IPv4, `AF_INET6` for IPv6.
- `type`
  - Type of communication (transport protocol), such as `SOCK_STREAM`, `SOCK_DGRAM`.
  - `SOCK_STREAM` for connection-oriented protocol, such as TCP.
  - `SOCK_DGRAM` for connectionless protocol, such as UDP.
- `protocol`
  - Typically zero. Used to identify a protocol within a domain and type.

- `hostname`
  - Used as an identifier of a network interface.
  - A host machine name or host machine IP address.
- `port`
  - Another identifier.
  - Each server listens for clients calling on assigned ports.

Socket Programming

Create a Socket

```
>>> import socket
>>> s = socket.socket(family, type, protocol)
```

- `family`: `AF_INET` (IPv4, default), `AF_INET6`, `AF_UNIX`, ...
- `type`: `SOCK_STREAM` (TCP, default), `SOCK_DGRAM` (UDP), `SOCK_RAW`, ...
- `protocol`: can be omitted. System can automatically generate according to `family` and `type`.
  - e.g., if `family=AF_INET` and `type=SOCK_STREAM`, then `protocol` can only be TCP (`IPPROTO_TCP`).

Socket Programming

General Socket Method

- `s.recv(bufsize)`
  - Receive TCP message.
  - Return value is a byte object.
  - `bufsize` specify maximum amount of data to be received.
- `s.send(bytes)`
  - Transmit TCP message.
  - The socket must be connected to a remote socket.
  - Return the number of byte sent.
- `s.recvfrom(bufsize)`
  - Receive UDP message.
  - Return a pair (byte, address).
  - `bufsize` specify maximum amount of data to be received.
- `s.sendto(byte, address)`
  - Transmit UDP message.
  - The socket should not be connected to a remote socket.
  - The destination socket is specified by `address`.

- `s.close()`
  - Close the socket.
  - All future operations on the socket object will fail.
  - The remote end will receive no more data.
- `s.gethostname()`
  - Return a string containing the hostname of the machine.
- `s.gethostbyname(hostname)`
  - Translate the hostname to IPv4 address format.

Socket Programming

Server Socket Method

- `s.bind(address)`
  - Bind address (hostname and port pair) to the socket.
  - The socket **must not** already be bound.
- `s.listen()`
  - Server set up and wait for client connection.
- `s.accept()`
  - Establish client connection.
  - Return a pair `(conn, address)`:
  - `conn`: new socket object used to send and receive data on this connection;
  - `address`: address bound to the socket on the other end of connection.

Socket Programming

Client Socket Method

- `s.connect(address)`
  - Initiates server connection.
  - Connect to a remote socket at `address`.

Socket Programming

Server Socket - Sample

```
>>> import socket
# create socket
>>> s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
# get hostname and assign port number
>>> host, port = socket.gethostname(), 1234
# bind the socket to the network address
>>> s.bind((host, port))
# wait for client connection
>>> s.listen()
# Establish and close connection with client.
>>> while True:
>>>     c, addr = s.accept()
>>>     print('Got connection from', addr)
>>>     c.send(b'Thank you for connecting')
>>>     c.close()
```

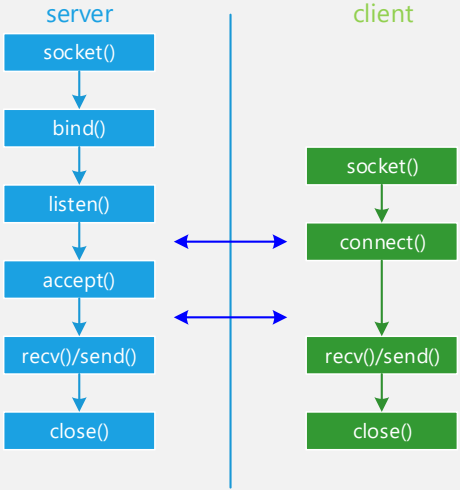
Socket Programming

Client Socket - Sample

```
>>> import socket
>>> s = socket.socket()
# only for test on same machine
>>> host, port = socket.gethostname(), 1234
# connect to server
>>> s.connect((host, port))
>>> print(s.recv(1024))
>>> s.close()
```

Socket Programming

Server Client Communication



Lab 2 Task
(continue)

- Try the simple server socket and client socket according to the lab manual.
- Create an echo server.
- Create a web server and web client.