Week 9

Slides & Exercises

- **last week’s assignments review**
- Questions about midterm assignment? (due today/very soon)

ATTENTION/CAUTION

Unlike the previous class materials provided, this material does not contain too much details and is to be used in conjunction with 2 other (external) documents (where you can find more details).Please access these URLs!

https://drive.google.com/file/d/0B1az-X5BagKZTFZTWExrOXk2aUE/view?usp=sharing

https://drive.google.com/file/d/0B1az-X5BagKZcnJ4c2V3S2hXclk/view?usp=sharing

Network Communication at different level

- Socket level (lower level)
- Protocol level: HTTP, FTP, SMTP, IMAP4,

POP3,etc

★ you all know HTTP!(protocol used between web browser and web server)

“Protocol”? (example: IMAP4)/email

run this on your terminal:

```
$telnet 133.27.5.100 imap
```

run these commands once connected

```
a1 LOGIN MyUsername MyPassword
a2 LIST "" "*"
a3 EXAMINE INBOX
a4 FETCH 1 BODY [ ]
a5 LOGOUT
```

Socket level

- connection oriented vs connectionless
- UDP vs TCP! (Fundamentals of IT #2?)

“Server” and “Client”

- passive
- active

- multicast
- multiple connection

Socket programming in Python

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

extension for IPV6

http://tools.ietf.org/html/rfc3493.html

Socket Terminology/Methods

Primitive	**Meaning**
Socket	Create a new communication endpoint
Bind	Attach a local address to a socket
Listen	Announce willingness to accept connections
Accept	Block caller until a connection request arrives
Connect	Actively attempt to establish a connection
Send	Send some data over the connection
Receive	Receive some data over the connection
Close	Release the connection

blocking vs non-blocking

2 packages that we'll be covering next

A. `socket` package (This will be **our focus today**)
B. `twisted` package (no hands-on)

A. socket package

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

bind(*address*)

Bind the socket to *address*. The socket must not alreadv be bound. (The format of *address* depends on the address family — see above.)

listen(*backlog*)

Listen for connections made to the socket. The *backlog* argument specifies the maximum number of queued connections and should be at least 0; the maximum value is system-dependent (usually 5), the minimum value is forced to 0.

A. socket package(cont)-TCP

accept()

Accept a connection. The socket must be bound to an address and listening for connections. The return value is a pair (conn, address)where *conn* is a *new* socket object usable to send and receive data on the connection, and *address* is the address bound to the socket on the other end of the connection

connect(*address***)**

Connect to a remote socket at *address*. (The format of *address* depends on the address family)-see next page

A.socket package(cont)-Address

Socket addresses are represented as follows: A single string is used for the **AF_UNIX** address family. A pair (host, port) is used for the **AF_INET** address family, where *host* is a string representing either a hostname in Internet domain notation like 'daring.cwi.nl' or an IPv4 address like '100.50.200.5', and *port* is an integer. For **AF_INET6** address family, a four-tuple (host, port, flowinfo, scopeid) is used, where*flowinfo* and *scopeid* represents sin6_flowinfo and sin6_scope_id member in **struct sockaddr_in6** in C. The address format required by a particular socket object is automatically selected based on the address family specified when the socket object was created.

For IPv4 addresses, two special forms are accepted instead of a host address: the empty string represents **INADDR_ANY**, and the string'<broadcast>' represents **INADDR_BROADCAST**. The behavior is not available for IPv6 for backward compatibility, therefore, you may want to avoid these if you intend to support IPv6 with your Python programs.

`A.socket` package(cont)-TCP

recv(*bufsize*[, *flags*])

Receive data from the socket. The return value is a string representing the data received. The maximum amount of data to be received at once is specified by *bufsize*. See the Unix manual page *recv(2)* for the meaning of the optional argument *flags*; it defaults to zero.

sendall(*string*[, *flags*])

Send data to the socket. The socket must be connected to a remote socket. The optional *flags* argument has the same meaning as for**recv()** above. Unlike **send()**, this method continues to send data from *string* until either all data has been sent or an error occurs. None is returned on success. On error, an exception is raised, and there is no way to determine how much data, if any, was successfully sent.

(there is also “send”-but sendall in general is more convenient)

A.socket package(cont)-UDP

recvfrom(*bufsize*[, *flags*])

Receive data from the socket. The return value is a pair (string, address) where *string* is a string representing the data received and*address* is the address of the socket sending the data. See the Unix manual page *recv(2)* for the meaning of the optional argument*flags*; it defaults to zero.

sendto(*string*, *flags*, *address*)

Send data to the socket. The socket should not be connected to a remote socket, since the destination socket is specified by *address*. The optional *flags* argument has the same meaning as for **recv()**

A. `socket` package(cont)

close()

Close the socket. All future operations on the socket object will fail. The remote end will receive no more data (after queued data is flushed). Sockets are automatically closed when they are garbage-collected

B. `twisted` package

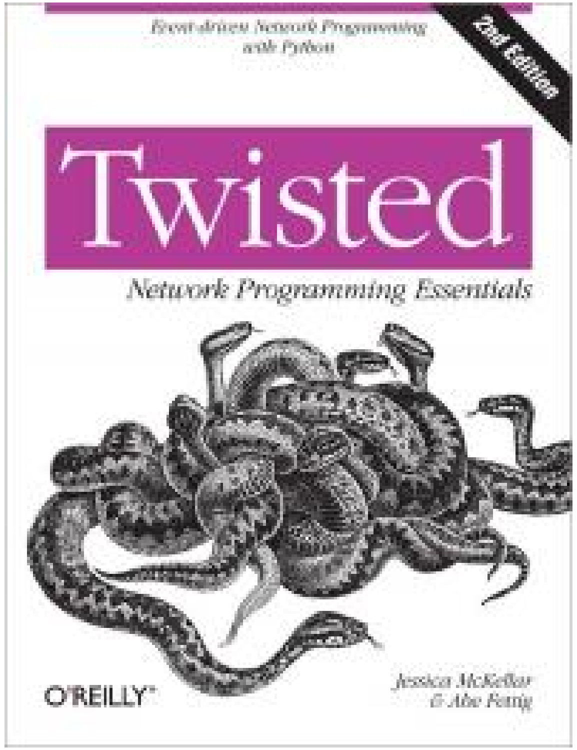
Ref https://pypi.python.org/pypi/Twisted

Twisted is an event-driven networking engine for Python.

Twisted implements a variety of networking and communication protocols and exposes them all as method-calls on your Python objects. Client and server implementations are provided for various standard protocols, including:

- HTTP (twisted.web)
- IMAP, POP, SMTP (twisted.mail)
- DNS (twisted.names)
- TLS (core)
- SSH, Telnet (twisted.conch)
- IRC, XMPP, OSCAR (twisted.words)
- Ethernet, IP, TUN/TAP (twisted.pair)
- NMEA (twisted.positioning)

One whole book for `twisted`!


Event-driven Network Programming with Python
2nd Edition
Twisted
Network Programming Essentials
O'REILLY
Jessica McKellar & Abe Fettig

For “real world” network programming twisted is commonly used (in Python)!

ETC TOPICS (required for exercise): try, finally, (except)

Making sure something is executed in the end.

try is typically used to catch exception for example:

```
while True:
...     try:
...         x = int(raw_input("Please enter a number: "))
...         break
...     except ValueError:
...         print "Oops!  That was no valid number.  Try again..."
...     finally:
...          print 'Goodbye, world!'
```

important point: A ***finally*** *clause* is always executed before leaving the **try** statement, whether an exception has occurred or not. When an exception has occurred in the **try** clause and has not been handled by an **except** clause (or it has occurred in a **except** or **else** clause), it is re-raised after the **finally** clause has been executed. The **finally** clause is also executed "on the way out" when any other clause of the **try** statement is left via a **break**, **continue** or **return** statement. **While(True)** will create an indefinite loop (the program will run"forever")

ETC TOPICS(required for exercise): etc system call

Getting current time using python time package

```
import time
now = time.strftime("%c")
```

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

Additional info only :Python system call (to execute something on the shell)

```
import os
os.system("your command here")
```

For example: os.system(“ls”), os.system(“date”),

```
import commands
tets=commands.getoutput("env")
```

ETC TOPICS(required for exercise):

Type this on the terminal prompt to find a machine’s name

```
$hostname
```

protocol/packet analyzer(install/use any of this-the red one(Wireshark)?)

“tcpdump” (shell)-available on your shell

http://www.macupdate.com/app/mac/24867/cocoa-packet-analyzer

http://download.cnet.com/Wireshark/3000-2092_4-10668286.html (may require XQuartz to be installed)

http://download.cnet.com/Eavesdrop/3000-18508_4-52176.html

http://download.cnet.com/NetDumper/3000-18508_4-189437.html

Install Wireshark Protocol Analyzer

- Install Xquartz on your mac first
- Install Wireshark

- Set Capture filter to the server’s IP address

(the server used in your exercise/example)

- Start

Example codes

Let’s have a look at these examples: (download them from SFS):

`<ex91S.py>,<ex91C.py>,<ex92S.py>,<ex92C.py>, <ex95S.py>(*),<ex95C.py>(*)`

Run them and observe them using protocol analyzer(20 min max)

(*)For ex95 you need to install twisted: $pip install twisted

Example codes(cont)

`<ex91S.py>` and `<ex91C.py>`

Demonstrate a server and client connection using **TCP**.The client send a message, the server simply send it back.Note that in this example server and client are on the same host. You will be required to modify this “base code” for the exercise

Example codes(cont)

`<ex92S.py> and <ex92C.py>`

Demonstrate a server and client connection using **UDP**.The client send a message, the server simply send it back.Note that in this example server and client are on the same host.Also note that this is “connectionless”

Example codes(cont)

`<ex95S.py> and <ex95C.py>`

Demonstrate twisted implementation of ex91.

We will not cover this in details.

https://twistedmatrix.com/documents/current/core/examples/

http://radar.oreilly.com/2013/04/twisted-python-the-engine-of-your-internet.html

Exercises

A. Implementing a simple “time server”
B. Use 2 machines. Get it to work and try to observe network activities using protocol analyzer.

Exercise 9-1 (A)

- Modify `<ex91C.py>` and `<ex91S.py>` so that :
  - client will send a message “reqtime”
  - server will send current time at the server
  - client will display the time sent from the server. Optional:It would be a bonus if you can set the client’s current time (need a sudo to change time) Hints:
  - you can try to use protocol analyzer to check the traffic and probaby time lag(delay)

- SUBMIT files

Exercise 9-2 (B)

- modify `<ex91C.py>, <ex91S.py>, <ex92C.py>, <ex92S.py>` so that the server runs on the CNS desktop machine (zmac?) in the class, while the client runs on your laptop
- observe using protocol analyzer. Point which line number(s) in the analyzer capture where the message is acually transmitted?Can you see 2 of these of just one? .submit your answer along with codes

SUBMIT files and answer(answer in a text file, zip together with code)