Python Socket Programming

Daniel Zappala

CS 360 Internet Programming
Brigham Young University
Addresses
Address Families

- **AF_UNIX**
  - communication between two processes on the same machine
  - represented as a string

- **AF_INET**
  - communication over the Internet, with IP version 4
  - represented as a tuple of \((host, port)\), \(host\) is a string host name, \(port\) is an integer port number
  - \(host\) can be a Internet host name (www.cnn.com) or an Ip address (64.236.24.20)

- **AF_INET6**
  - communication over the Internet, with IP version 6
  - represented using a tuple of \((host, port, flow_info, scope_id)\)
    - \(flow_info\) is a flow identifier used for Quality of Service (e.g. low delay or guaranteed bandwidth)
    - \(scope_id\) is a scope identifier, which can limit packet delivery to various administrative boundaries
Server
## Create a Socket

1. \( \texttt{socket ( | family | , | type | [ , | protocol | ] ) } \)

- returns a socket identifier
- \( \texttt{family} \) is \( \texttt{AF\_UNIX} \), \( \texttt{AF\_INET} \), or \( \texttt{AF\_INET6} \)
- \( \texttt{type} \) is usually \( \texttt{SOCK\_STREAM} \) for TCP, or \( \texttt{SOCK\_DGRAM} \) for UDP
- \( \texttt{protocol} \) is ignored in most cases

```python
1 from socket import *
2 s = socket(AF_INET, SOCK_STREAM)
```
Bind the Socket

```
1       bind ( | address | )
```

- `address` is a tuple defined by the address family

```
1       host = ''
2       port = 50000
3       s.bind((host, port))
```

- AF_INET is a (host, port) tuple
- setting host to the empty string tells the OS to use any address associated with the host
- port number must not be currently used, or else an exception is raised
Listen

1  listen(|backlog|)

- tells the server to listen for incoming connections
- `backlog` is an integer specifying the maximum number of connections the server will hold in a queue
- use a minimum of one, OS maximum is usually 5
- use threads to service the queue of connections quickly if service time for a connection is large

1  backlog = 5
2  s.listen(backlog)
Accept a Client

1. `accept()`

- returns a tuple `(socket, address)`
- `socket` is a new socket identifier for the client
- `address` is the client address, a tuple defined by the address family (host, port for AF_INET)

1. `client, address = s.accept()`
Client
Connect to the Server

1. `connect(|address|)

- **address** is a tuple defined by the address family

1. `host = 'localhost'
2. `port = 50000
3. `s.connect((host, port))

- use a (host, port) tuple just like bind
- must use the address and port of the server, not the client
- using localhost means the server is running on the local machine – use an Internet host name or an IP address for a remote machine
- server must be listening for clients, or else an exception is raised
Sending and Receiving
Sending Data

1. `send([string [, flags]])`
   - returns the number of bytes sent
   - `string` is the data to be sent
   - see Linux send man page for flags
   - possible that some of the data is not sent – must check return value and resend if necessary

```python
1  data = "Hello World"
2  client.send(data)
```
Receiving Data

1 \texttt{recv(|buffersize|[,|flags|])}

- returns a string representing the data received
- \textit{buffersize} is the maximum size of the data to be received
- possible that less data is received than the maximum
- use \texttt{client.setblocking(0)} for non-blocking I/O on a client socket

1 \texttt{size = 1024}
2 \texttt{data = client.recv(size)}
Example Code

- Echo Client and Server