

Python Socket Programming

- sockets are often used to establish a connection between machines
 - Python has a socket module for this purpose
- the Python socket module maps onto the standard C socket library
- sockets provide a ISO OSI-7 transport level layer interface
 - either UDP or TCP transports can be used
 - both use the Port level of addressing within UDP or TCP

Simple server

- ```
#!/usr/bin/python
from socket import *
myHost = ''
myPort = 2000

s = socket(AF_INET, SOCK_STREAM) # create a TCP socket
s.bind((myHost, myPort)) # bind it to the serv
s.listen(5) # allow 5 simultaneou
 # pending connections

while 1:
 # wait for next client to connect
 connection, address = s.accept() # connection is a ne
 while 1:
 data = connection.recv(1024) # receive up to 1K b
 if data:
 connection.send('echo -> ' + data)
 else:
 break
 connection.close() # close socket
```

## Simple client

- ```
#!/usr/bin/python
import sys
from socket import *
serverHost = 'localhost' # servername is local
serverPort = 2000        # use arbitrary port :

s = socket(AF_INET, SOCK_STREAM) # create a TCP socket
s.connect((serverHost, serverPort)) # connect to server o
s.send('Hello world') # send the data
data = s.recv(1024) # receive up to 1K by
print data
```

Streaming an MP3 file though stdout and stdin

- `stdin` is the default input file descriptor
- `stdout` is the default output file descriptor
- try this:


```
cat 01.mp3 | madplay -
```
- `cat` is a program which reads a file and writes it to `stdout`
- the `|` symbol is the pipe which connects the `stdout` of the left program to the `stdin` of the right program

Streaming an MP3 file though stdout and stdin

- madplay is a mp3 player which has been told to read its file from -
 - where - means stdin
- you should aim to replace the program cat with your own Python client which streams the mp3 file from your Python server
- the client only needs to write the mp3 file to stdout
 - where madplay will read it and play it!

SSH component of your Python client

```
#!/usr/bin/python
#
# secure shell pipe module
#

import os
import sys
from socket import *

localPortNo=8000
maxTries=10
blockSize=65536*16
```

SSH component of your Python client

```
def createTCPSocketSSH (remoteHostname, remotePort=22, localPort=-1)
global localPortNo
if localPort == -1:
    localPort = localPortNo
    localPortNo = localPortNo+1
tryNo = 1
while 1:
    command = "ssh -f -g -A -X -N -T -L%d:localhost:%d %s\n" \
              % (localPort, remotePort, remoteHostname)
    result = os.system(command)
    if result == 0:
        break
    localPort = localPort+1
    tryNo = tryNo + 1
    if tryNo == maxTries:
        os.exit(1)

# create a TCP socket which connects to our ssh pipe
s = socket(AF_INET, SOCK_STREAM)
s.connect(("localhost", localPort))
return s
```