Socket RAW programming

Client

- must handle every field of the datagram
- must send a datagram to the network

Server

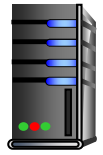
- must handle every field of the datagram
- Must receive a datagram from the network

- when using socket raw
  - the programmer must handle the *protocols headers* and payload, and any every type of protocol control
  - create and receive link level datagrams or you can filter any protocol level!

Application viewpoint

It must be transparent!

Client/server socket interaction: RAW



server (running on serverIP)

create socket RAW
serverSocket =
socket(AF_INET,SOCK_RAW)

↓

read datagram from
serverSocket

↓

write reply to
serverSocket
(specifying
client address,
port number
int the
headers)

client 

create socket:
clientSocket =
socket(AF_INET,SOCK_RAW)

↓

Create datagram with serverIP address
And port=x; send datagram via
clientSocket

↓

read datagram from
clientSocket

↓

close
clientSocket

Example app: Python RAW client

create RAW socket for client

define mac address

```
#!/usr/bin/env python
from socket import socket, AF_PACKET, SOCK_RAW
s = socket(AF_PACKET, SOCK_RAW)
s.bind(("eth1", 0))
 # We're putting together an ethernet frame here,
# but you could have anything you want instead
# Have a look at the 'struct' module for more
# flexible packing/unpacking of binary data
# and 'binascii' for 32 bit CRC
src_addr = "\x01\x02\x03\x04\x05\x06"
dst_addr = "\x01\x02\x03\x04\x05\x06"
payload = ("["*30)+"PAYLOAD"+("]"*30)
ethertype = "\x08\x01"
 s.send(dst_addr+src_addr+ethertype+payload)
```