UDP/IP

udp_send.py

```
import socket
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
addr = ("127.0.0.1", 5005)

sock.sendto('hello'.encode('utf8'), addr)
```

```
import socket
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
addr = ("127.0.0.1", 5005)

import sys
data = sys.argv[1]

sock.sendto(data.encode('utf8'), addr)
```

```
import socket
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
addr = ("127.0.0.1", 5005)

while data := input()
    sock.sendto(data.encode('utf8'), addr)
```

bash commandline equivalent

```
echo "hello" > /dev/udp/127.0.0.1/5005
# or
nc -u 127.0.0.1 5005
```

udp_recv.py

```
import socket

sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

sock.bind(('0.0.0.0', 5005))
while True:
    data, addr = sock.recvfrom(1024)
    print(f"received bytes: {data} from {addr}")
```

bash commandline equivalent

```
nc -u -l 5005
```

udp_recv_tk.py

```
import socket
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock.bind(('0.0.0.0', 5005))

import tkinter
root = tkinter.Tk()
canvas = tkinter.Canvas(root, width=512, height=512)
canvas.pack()

def recv_msg(x1,y1,x2,y2):
    #canvas.delete(tkinter.ALL)
    canvas.create_rectangle(x1,y1,x2,y2,outline="#F00", fill="#0F0")
    canvas.update()

def recv():
    while True:
        data, addr = sock.recvfrom(1024)
        print(f"received bytes: {data} from {addr}")
        recv_msg(*data.decode('utf8').split(','))

import threading
thread = threading.Thread(target=recv)
thread.daemon=True
thread.start()

root.mainloop()
```

`udp_recv_tk2.py`

```
GRID_SIZE=80
WIDTH=5
HEIGHT=4

import socket
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock.bind(('0.0.0.0', 5005))

import tkinter
root = tkinter.Tk()
root.resizable(False, False)
canvas = tkinter.Canvas(root, width=GRID_SIZE*WIDTH, height=GRID_SIZE*HEIGHT)
canvas.pack()

for x in range(WIDTH):
    canvas.create_line(x*GRID_SIZE, 0, x*GRID_SIZE, HEIGHT*GRID_SIZE, fill="black",
for y in range(HEIGHT):
    canvas.create_line(0, y*GRID_SIZE, WIDTH*GRID_SIZE, y*GRID_SIZE, fill="black", w


def recv_msg(*args):
    #canvas.delete(tkinter.ALL)
    canvas.create_rectangle(*args, outline="#F00", fill="#0F0")
    canvas.update()

def recv():
    while True:
        data, addr = sock.recvfrom(1024)
        print(f"received bytes: {data} from {addr}")
        recv_msg(*data.decode('utf8').split(','))

import threading
thread = threading.Thread(target=recv)
thread.daemon=True
thread.start()

root.mainloop()
```

```
python3 udp_recv_tk.py
python3 udp_send.py 100,100,200,200
```

Resources

- https://tkdocs.com/tutorial/canvas.html
- http://zetcode.com/gui/tkinter/drawing/

Mental Model

- User Datagram Protocol UDP/IP
  - connectionless
  - No akc (acknowledgement)
  - No guaranteed order
- Transmission Control Protocol TCP/IP
  - ack (will auto retry)
  - Reorders
- UDP/IP is subset (sort of misconception) TCP/IP. TCP is built on top of IP (not UDP)
- When sending UDP you send from a port (like a server), but a random port one is `bind`ed for you. 5 digit? port.
  - This allows sender to listen to responses
- If `listen`ing, conventionally 4 digits port

Video?

Mad lad ...

https://www.ffmpeg.org/doxygen/0.6/pixfmt_8h.html
GRAY8
https://github.com/superLimitBreak/mediaTimelineRenderer/blob/e99b8f421094efeaba2619dfc6c01a6ca0599d89/mediaTimelineRendererLib/renderer_video.py#L20