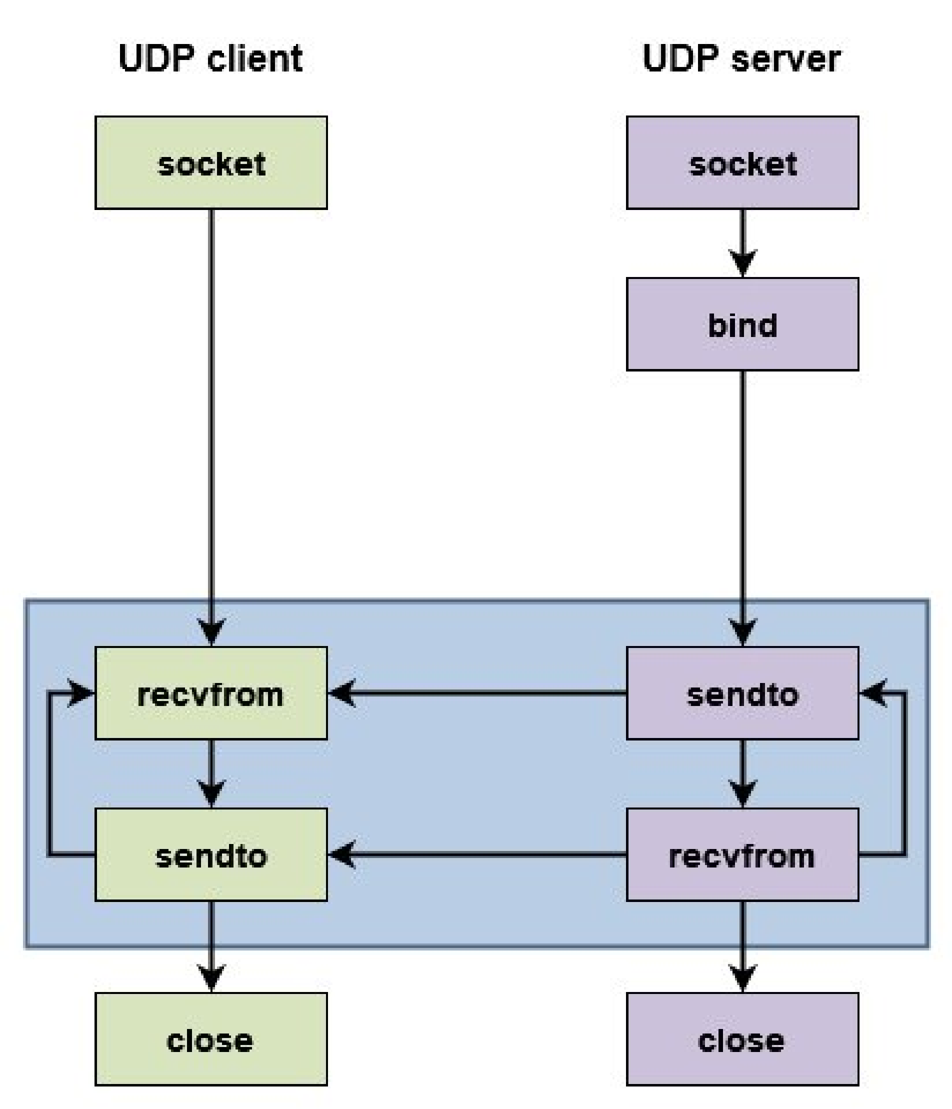
Telecommunications Network

Practice 5

UDP


UDP client
UDP server
socket
socket
bind
recvfrom
sendto
sendto
recvfrom
close
close

UDP

- socket

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

- recvfrom()

```
data, address = sock.recvfrom(4096)
```

- sendto()

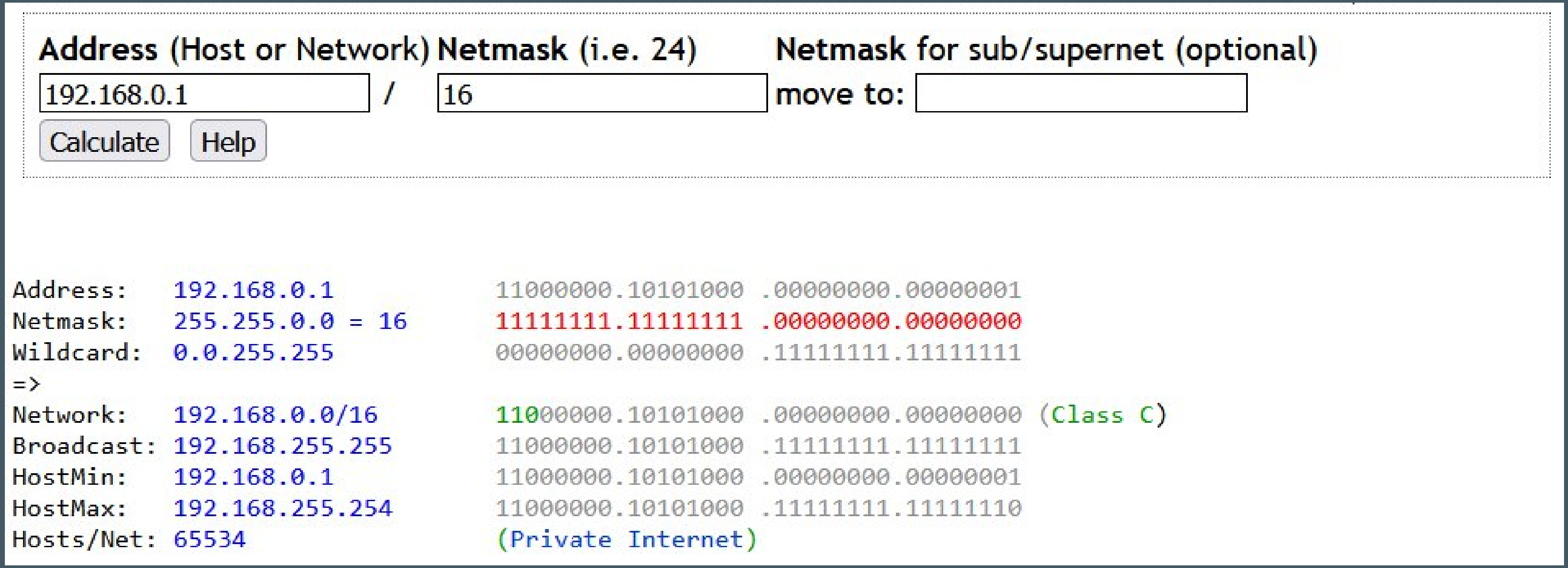
```
sent = sock.sendto(data, address)
```

Exercise I.

- Write a server-client application that uses UDP.
- The client should send a b”Hello Server” bytestring to the server and in return the server should respond with b”Hello Client”.
- Is our server capable of handling multiple clients? Why or why not?

Netmask

- Description of addresses in a subnet.



Address (Host or Network) 192.168.0.1 / Netmask (i.e. 24) 16

Netmask for sub/supernet (optional) move to:

```
Address:    192.168.0.1          11000000.10101000 .00000000.00000001
Netmask:    255.255.0.0 = 16     11111111.11111111 .00000000.00000000
Wildcard:   0.0.255.255          00000000.00000000 .11111111.11111111
=>
Network:    192.168.0.0/16       11000000.10101000 .00000000.00000000 (Class C)
Broadcast:  192.168.255.255      11000000.10101000 .11111111.11111111
HostMin:    192.168.0.1          11000000.10101000 .00000000.00000001
HostMax:    192.168.255.254      11000000.10101000 .11111111.11111110
Hosts/Net:  65534                (Private Internet)
```

Netmask RFC

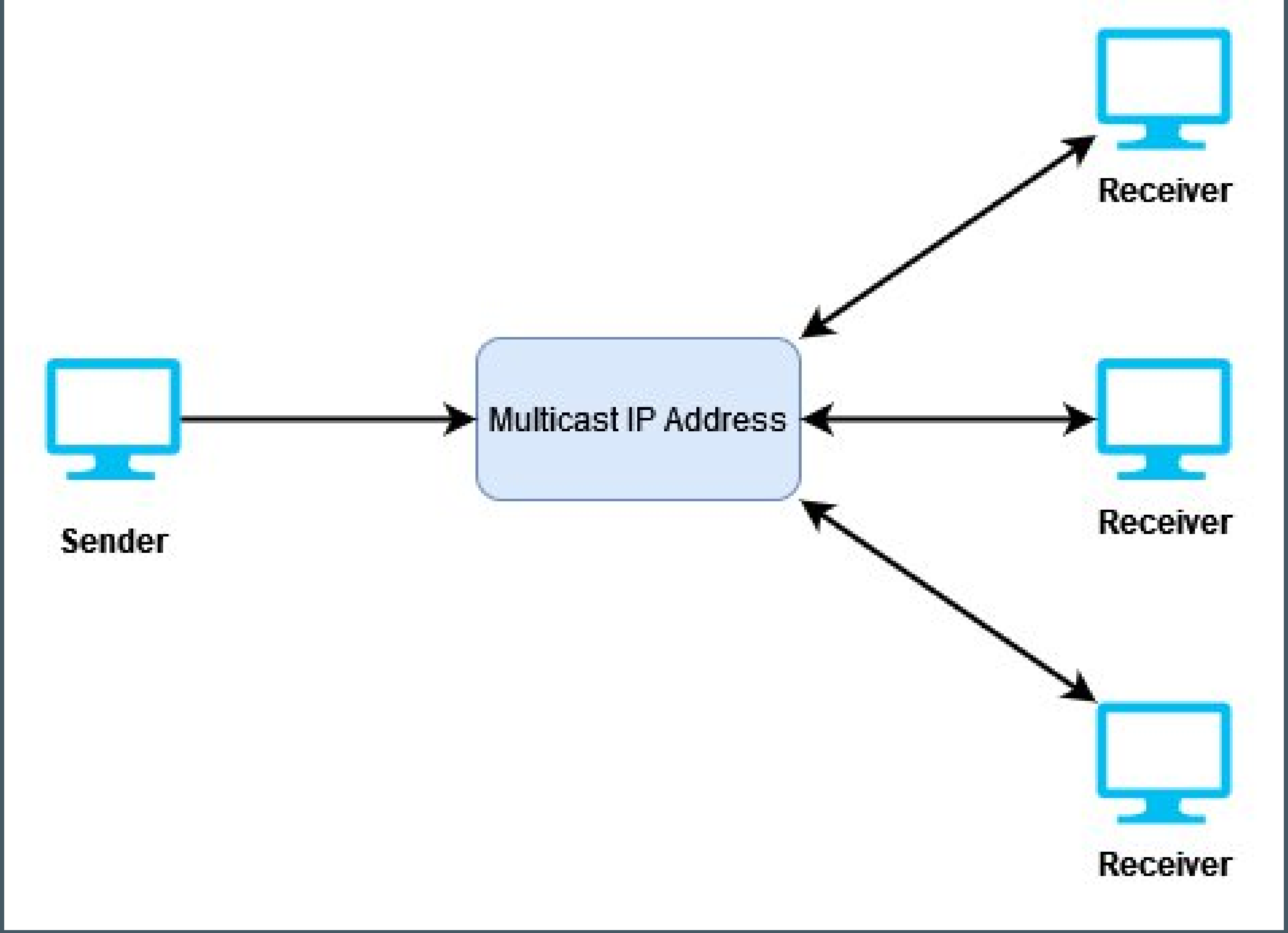
CIDR RFC

- How many addresses do we get with the following masks?
- Calculate the min and max address for each option.
  - 188.100.22.12/32
  - 188.100.22.12/20
  - 188.100.22.12/10

Multicast

Class	Range	Description
A	0.0.0.0 - 127.255.255.255	Unicast
B	128.0.0.0 - 191.255.255.255	Unicast
C	192.0.0.0 - 223.255.255.255	Unicast
D	224.0.0.0 - 239.255.255.255	Multicast
E	240.0.0.0 - 255.255.255.255	Reserved

Multicast



Multicast

- setsockopt() (sender)

```
ttl = struct.pack("b", 1)
sock.setsockopt(socket.IPPROTO_IP, socket.IP_MULTICAST_TTL, ttl)
```

- Adding a socket to the multicast group (recv)

```
multicast_group = "224.3.29.71"
group  =  socket.inet_aton(multicast_group)
mreq = struct.pack("4sL", group, socket.INADDR_ANY)
sock.setsockopt(socket.IPPROTO_IP, socket.IP_ADD_MEMBERSHIP, mreq)
```

Exercise II

Imagine a news agency (the sender) needs to send an urgent bulletin to all its subscribed news terminals (the receivers) at once. Instead of sending an individual copy to each terminal, it will send one single message to the 224.1.1.1 multicast group. Any terminal that has joined this group will instantly receive the bulletin.

Exercise II

Key Steps:

Create a UDP Socket: Same as the receiver.

Set the Time-To-Live (TTL): Multicast packets have a TTL that determines how many network hops (routers) they can cross. A TTL of 1 is standard practice to ensure the packets do not leave the local network segment. This is configured using setsockopt with IP_MULTICAST_TTL.

Send Data: Use sendto() to send the message directly to the multicast group address and port.

Wait for Acknowledgement: The sender will then wait for a brief period to receive the unicast ack from any receivers that heard the message.

Udp stream example

- Example code on the website.
- Install OpenCV

```
python3 –m pip install --user opencv-python
```

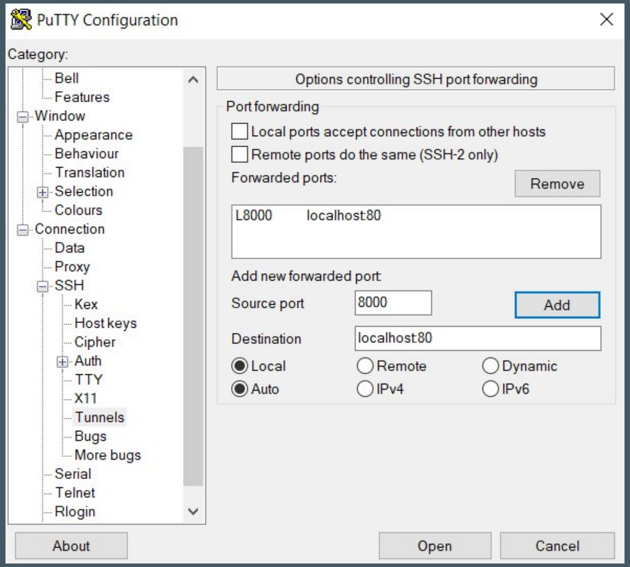
How video streaming works

SSH Tunnel

- In the terminal (Works on Windows as well!):

```
ssh -L 8000:localhost:80 user@hostname
```

- Another option can be the use of a friendlier (one with a GUI) ssh client e.g. Putty.



PuTTY Configuration

Options controlling SSH port forwarding

Port forwarding

□ Local ports accept connections from other hosts

□ Remote ports do the same (SSH-2 only)

Forwarded ports:

```
L8000       localhost:80
```

Add new forwarded port:

Source port: 8000

Destination: localhost:80

● Local ○ Remote ○ Dynamic

● Auto ○ IPv4 ○ IPv6

Exercise III.

- Modify the calculator application so that it uses UDP instead of TCP.

Exercise IV.

- Write a server-client application, where the client send a picture to the server over UDP.
  - The client should send the file in 200 byte chunks.
  - If it gets to the end of the file, the client should send an empty string.
  - The server should acknowledge every chunk it receives with the b”OK” bytestring.