Sentinel

Networking - Sockets

DEFENDING OUR DIGITAL WAY OF LIFE

What is a socket?

- Software interface for network communication
- A (virtual) connection between 2 applications (client and server)
- Identifies a specific network connection
    - Source/Destination ports, Source/Destination IPs
- You can read and write from sockets in Python, like files

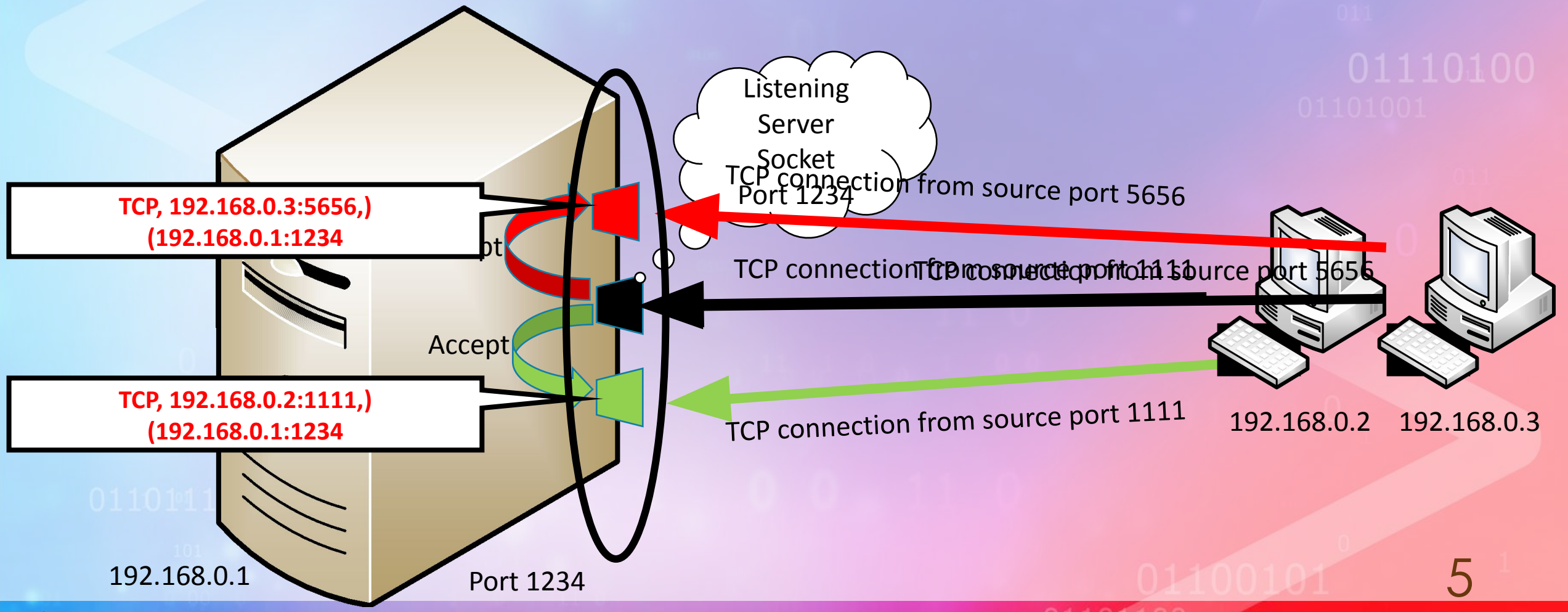
Types

- UDP socket
  - Send and receive UDP packets
- TCP socket
  - Manage TCP connections (listen, connect, close connection)
  - Send data over a TCP connection

TCP sockets

- Server
  - Listening socket - waiting for incoming connections (SYN)
  - After connection (SYN, SYN-ACK, ACK) - creates a new communication socket for this specific connection
  - The original socket continues listening for new connection requests
- Client
  - Connects to a server (IP + port)
  - After connecting - the socket is used for this specific connection

TCP sockets


Listening
Server
Socket
Port 1234
TCP, 192.168.0.3:5656,)
(192.168.0.1:1234
Accept
TCP, 192.168.0.2:1111,)
(192.168.0.1:1234
TCP connection from source port 5656
TCP connection from source port 1111
TCP connection from source port 5656
TCP connection from source port 1111
192.168.0.2
192.168.0.3
192.168.0.1
Port 1234

TCP client - connect

```
import socket

SERVER_IP = '192.168.1.1'
SERVER_PORT = 1234

sock = socket.socket()  # default is TCP
sock.connect((SERVER_IP, SERVER_PORT))
```

TCP server - bind and listen

- bind - connect the socket locally to a port and IP (0.0.0.0 - wait for connections on all available interfaces)
- listen - start waiting for incoming connections

```
import socket

SERVER_IP = '0.0.0.0'
SERVER_PORT = 1234

sock = socket.socket()  # default is TCP
sock.bind((SERVER_IP, SERVER_PORT))
sock.listen()
```

TCP server - accept

- accept - wait for incoming connection (blocking).
  When there's a connection, returns a new socket for this connection, and the client's address (IP, port)

```
con, addr = sock.accept()
print('Incoming connection from', addr)

# con is the new socket for this connection
```

TCP client/server - send data

- send - sends binary data (bits and bytes).

  Before we send text, we have to encode it to binary with *encode()*

```
data = 'Hello!'
sock.send(data.encode())
```

TCP client/server - receive data

- receive - receives binary data (bits and bytes).

  We have to specify the buffer size (maximum bytes to receive).

  Before we print/use it, we have to decode from binary to text with *decode()*

```
data = sock.recv(1024) # buffer size
print(data.decode())
```

TCP client example

- close - closes the connection (sends FIN).

```
import socket

sock = socket.socket()
sock.connect(("127.0.0.1", 1337))
sock.send("Hello world!".encode())
data = sock.recv(1024)
print(data.decode())
sock.close()
```

TCP server example

```
import socket

sock = socket.socket()
sock.bind(("0.0.0.0", 1337))
sock.listen()

con, addr = sock.accept()
con.send("Hello world!".encode())
data = con.recv(1024)
print(data.decode())
sock.close()
```

UDP sockets

- No connection - just send and receive data
- We have to specify the socket type

  socket.AF_INET - means that it's a connection over IP

  socket.SOCK_DGRAM - means that it's UDP (TCP is: socket.SOCK_STREAM)

```
import socket

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

UDP client example

- sendto - send data to an arbitrary IP and port.
- recvfrom - returns the received data and the address of the sender.

```
import socket

sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock.sendto("Hello world!".encode(), ("127.0.0.1", 1337))
data, addr = sock.recvfrom(1024)
print("Message from:", addr)
print(data.decode())
sock.close()
```

UDP server example

- Same as client, except we must bind the socket to an IP and port before we can receive data.

```
import socket

sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock.bind(("0.0.0.0", 1337))
data, addr = sock.recvfrom(1024)
print("Message from:", addr)
print(data.decode())
sock.sendto("Hello world!".encode(), addr)
sock.close()
```

What did we learn?

- What is a socket
- TCP sockets
  - Client - connect
  - Server - bind, listen, accept
  - Send and receive
- UDP sockets
  - sendto and recvfrom