How does the Internet work?

By IEEE-NITK

INTERNET... WHAT IS IT?

It is simply a network of networks!

Well, What is a network?

It is a collection of machines which can talk to one another!

A machine can be your laptop, computer, phone etc.,

How do 2 machines talk to each other?

Say your phone and facebook server..

To understand that,

We will design the internet!!

Let us take the example of postal system.

How does a letter reach from your home to your friend's home?

How does a letter reach from your home to your friend's home?

1. First you put the letter in an envelope.

How does a letter reach from your home to your friend's home?

1. First you put the letter in an envelope.
2. You drop that envelope in the nearest postbox.

How does a letter reach from your home to your friend's home?

1. First you put the letter in an envelope.
2. You drop that envelope in the nearest postbox.
3. That envelope is taken to the nearest post-office.

What is that number you put on an envelope?

It is the ZIP CODE!

How does a letter reach from your home to your friend's home?

1. First you put the letter in an envelope.
2. You drop that envelope in the nearest postbox.
3. That envelope is taken to the nearest post-office.
4. Inside the post-office, several such envelopes are segregated according to the ZIP-Code.

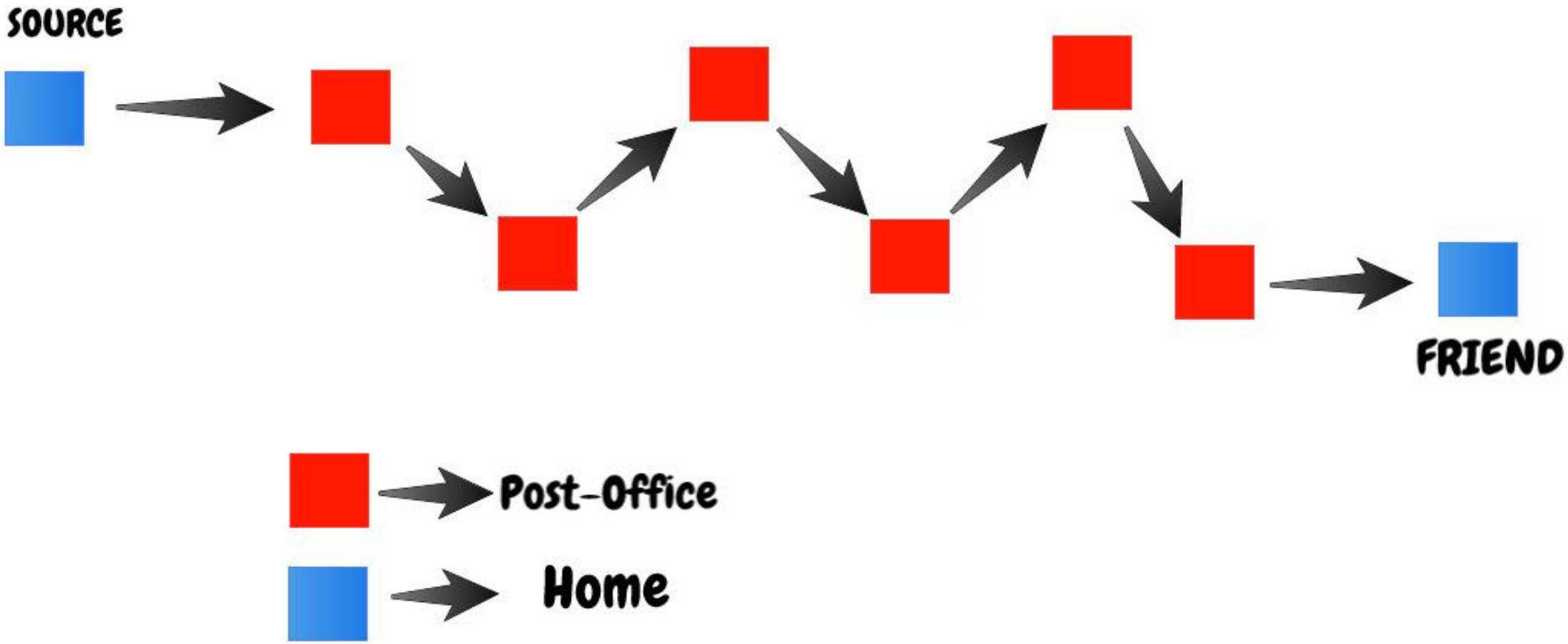
How does a letter reach from your home to your friend's home?

1. First you put the letter in an envelope.
2. You drop that envelope in the nearest postbox.
3. That envelope is taken to the nearest post-office.
4. Inside the post-office, several such envelopes are segregated according to the ZIP-Code.
5. It goes to several such post-offices.

How does a letter reach from your home to your friend's home?

1. First you put the letter in an envelope.
2. You drop that envelope in the nearest postbox.
3. That envelope is taken to the nearest post-office.
4. Inside the post-office, several such envelopes are segregated according to the ZIP-Code.
5. It goes to several such post-offices.
6. Finally, it reaches the post-office in your friend's location.

Look at this image



Now what?

How will the letter reach your friend's home??

You will need an address right?

SUMMARY

Specify Destination Address on the envelope.

What if your friend wants to write a letter back to you?

What if your friend wants to write a letter back to you?

Specify Source Address on the envelope.

Summary

The Envelope now has both Source and Destination Addresses

Once your friend gets the letter, the envelope is removed.

And your friend reads the letter!

Let us go back to our question…

How do 2 machines talk to each other?

Can we draw ideas from the letter example?

If machines have to talk to each other, every one of those machines should have an 'address'

How can machine 'A' send a file to machine 'B'?

Put the file in an 'electronic' envelope with source and destination addresses on it.

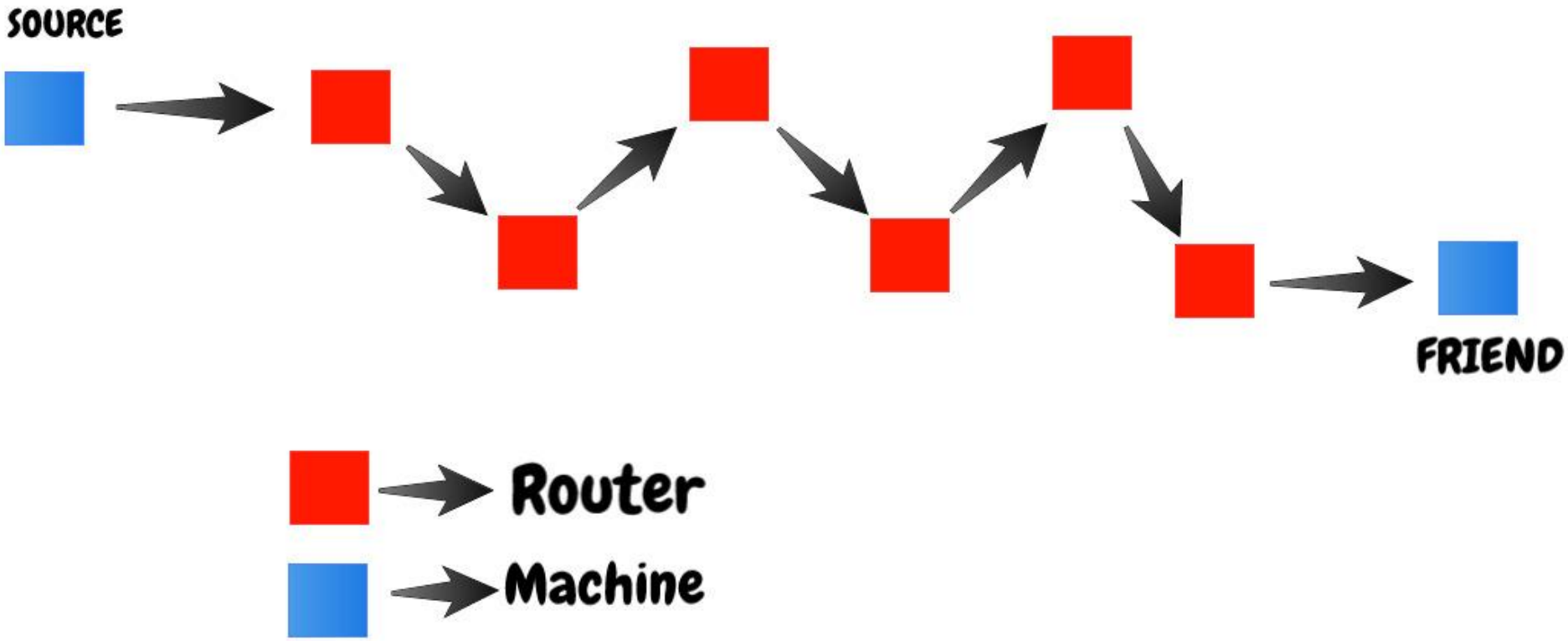
Then send it!

Instead of post-offices, let us have electronic devices which do the same job as post-offices.

Instead of post-offices, let us have electronic devices which do the same job as post-offices.

Such devices are called Routers.

Look at this image:



Once the electronic envelope reaches machine 'B', 'B' removes the envelope and saves the file.

In technical terms...

PACKET = DATA + ELECTRONIC ENVELOPE

Packet!

In the actual internet,

IP Addresses are used.

IP Addresses

1. In Windows, open the Command prompt and type the command “ipconfig” .
2. In Linux and Mac, open the Terminal and type the command “ifconfig”

EG: 10.12.89.43 , 192.168.1.122 ,
172.16.34.68 ETC.,

It is a 4-byte number separated by dots.

Imp: no 2 machines are given the same IP Address.

It is a unique address - like your home address.

Now, 2 machines can talk to each other.

Is this the internet?

Let us see...

Client-Server Architecture

Anyone heard the terms client and server?

Client - requests for data

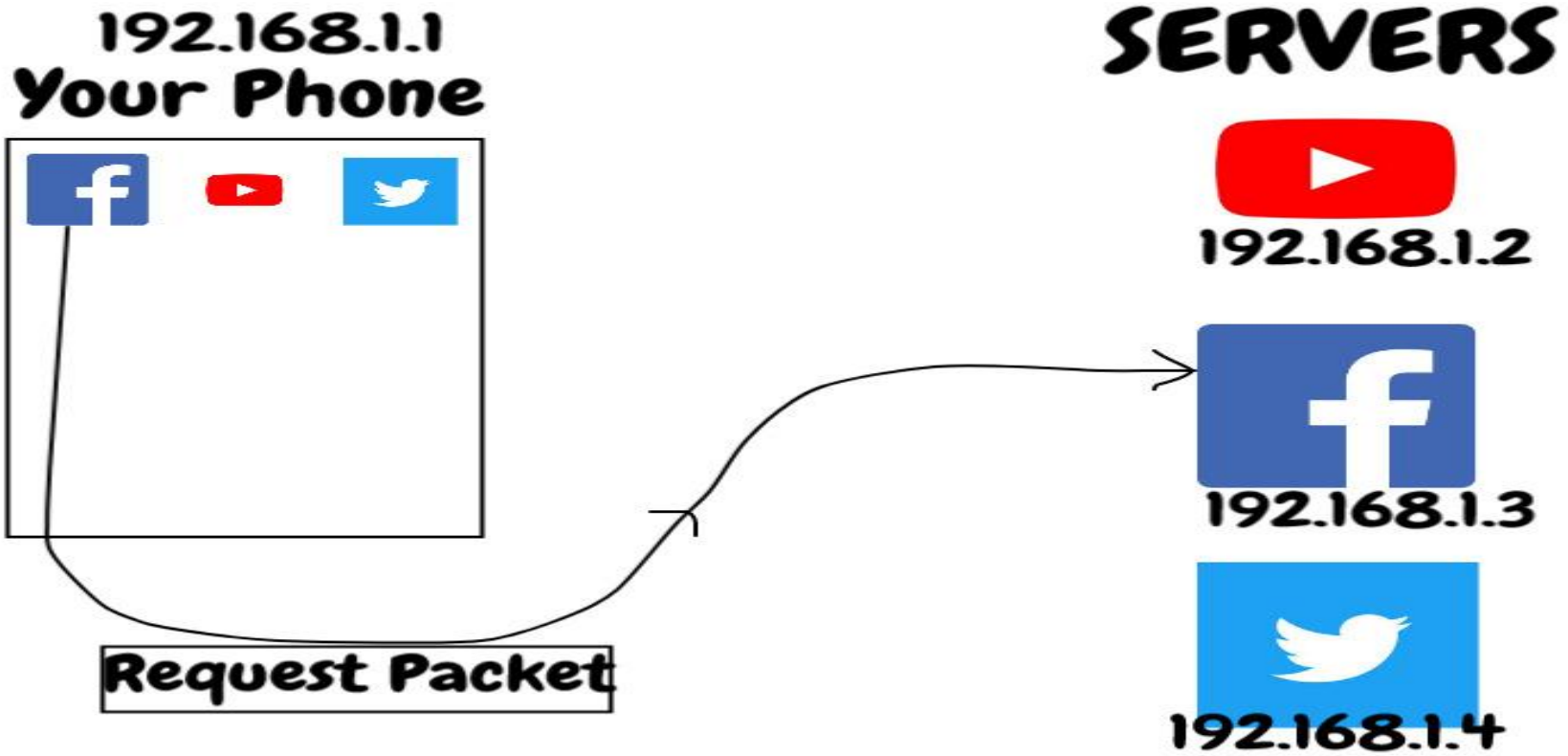
Server - 'serves' that data to client

IMP: Client always starts first!

Consider this scenario...

Qn: Facebook app requests it's server for an image. Facebook server sends back the image. Will the image reach the application?

Request

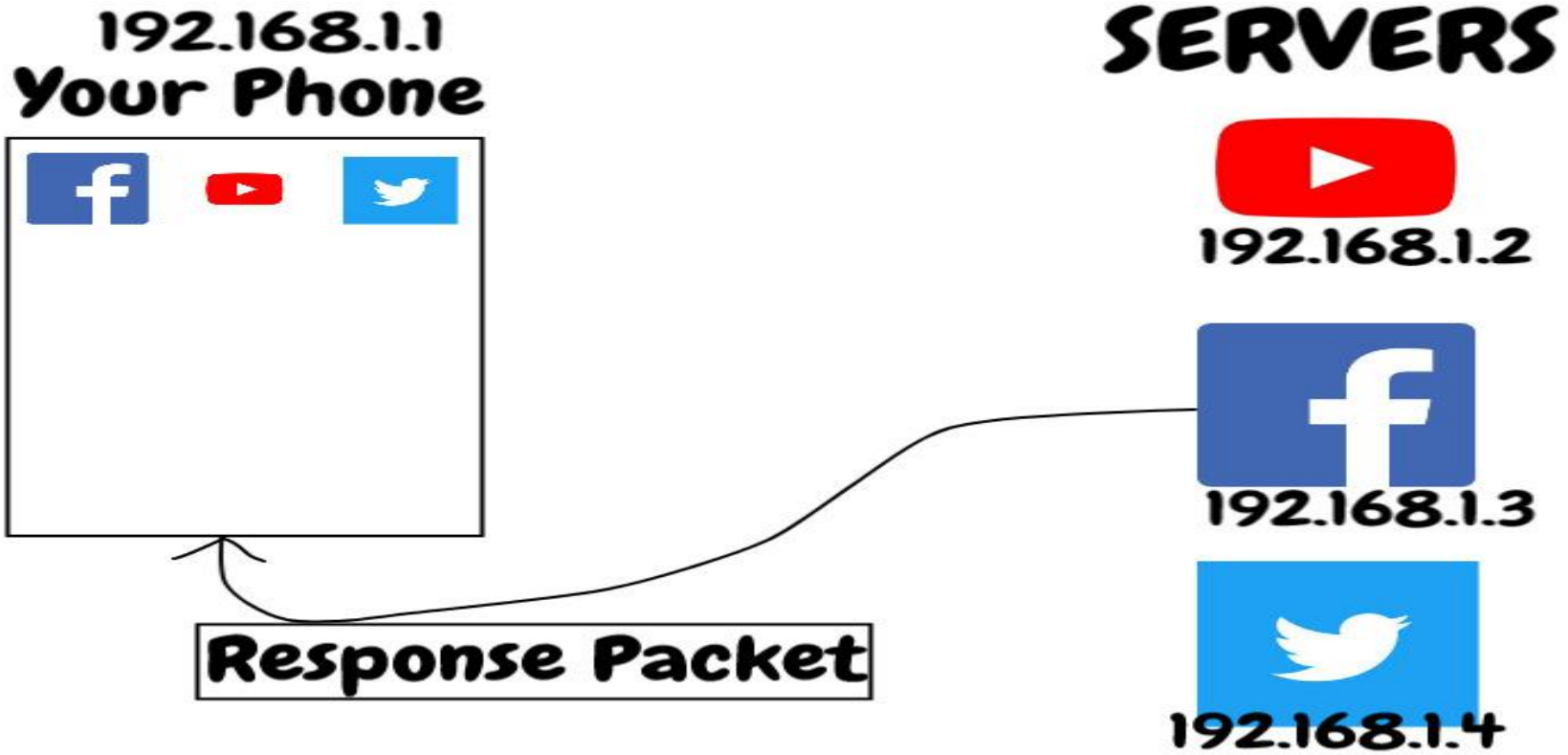

192.168.1.1
Your Phone
SERVERS
192.168.1.2
192.168.1.3
192.168.1.4
Request Packet

Request Packet

Give me "image.png"	192.168.1.1	192.168.1.3

The Request packet goes to Facebook Server. Server sends a response - a packet containing the image 'image.png'

Response


192.168.1.1
Your Phone
SERVERS
192.168.1.2
192.168.1.3
192.168.1.4
Response Packet

Response Packet

file – image.png	192.168.1.3	192.168.1.1

Now what??

Can we just send the Response packet to Facebook app?

How does your phone know to which application to send it to?

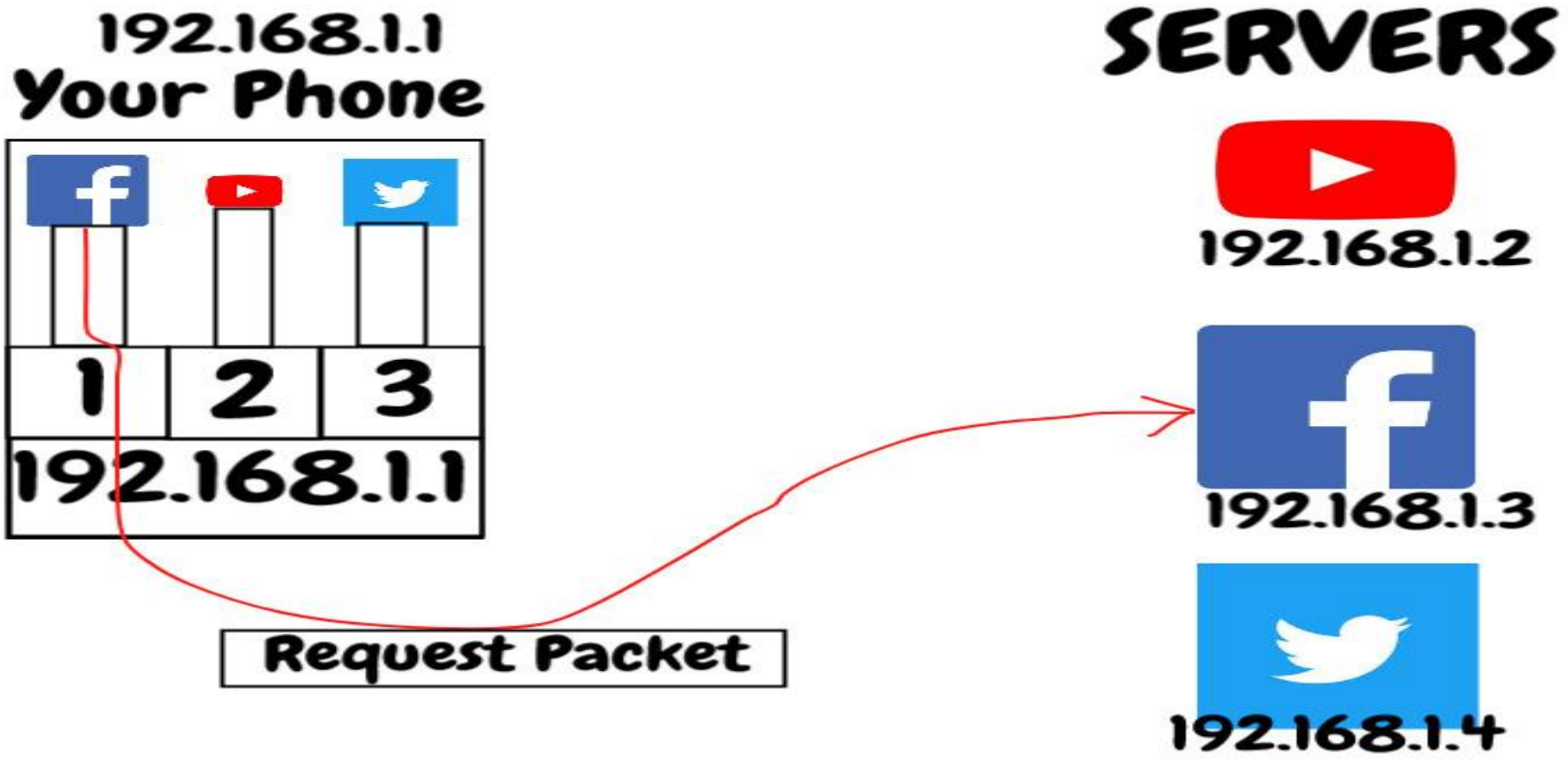
There are 3 applications - Facebook, YouTube, Twitter.

How can we solve this problem?

Solution: Assign unique numbers to these applications

Let us take a look at the solution

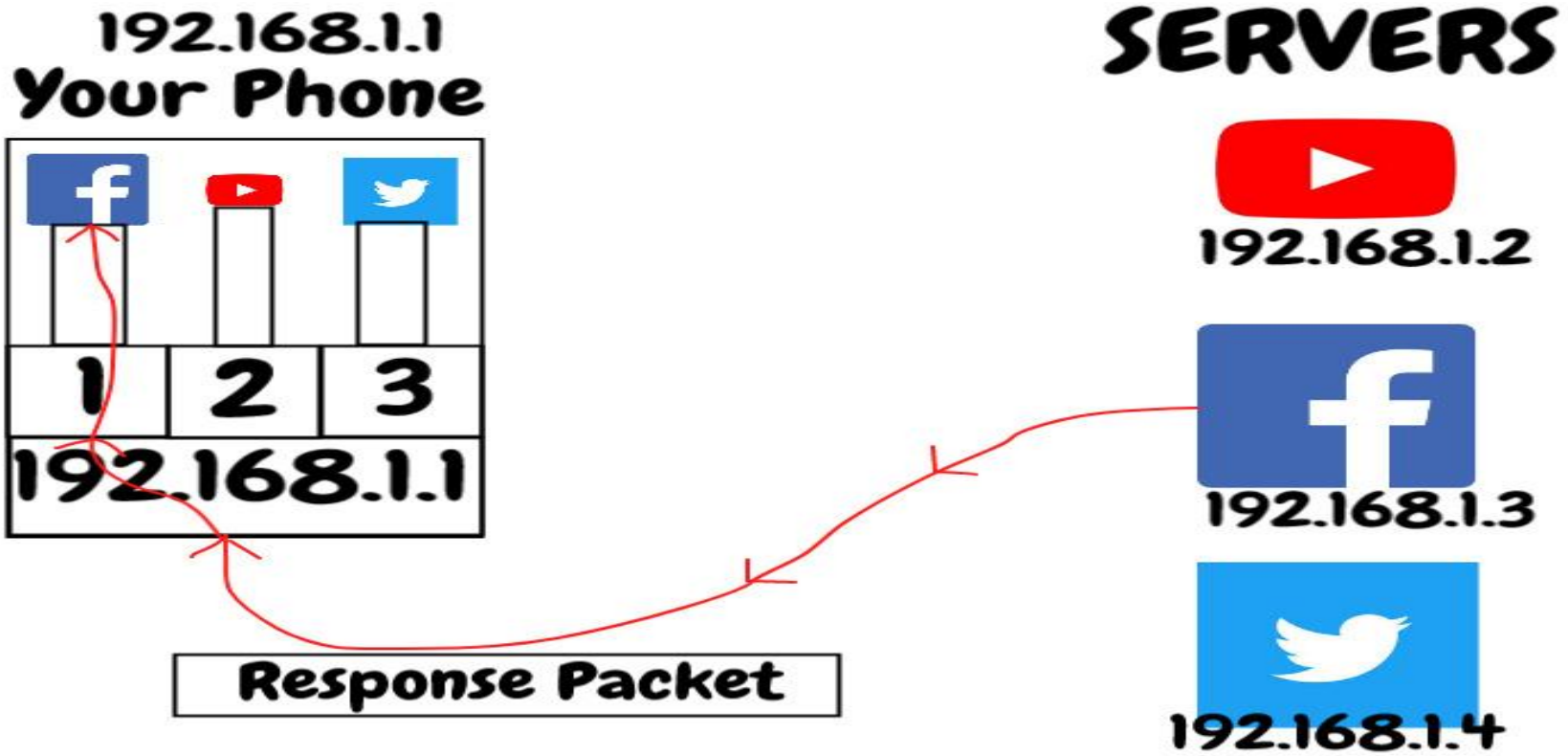
Request


192.168.1.1
Your Phone
1
2
3
192.168.1.1
Request Packet
SERVERS
192.168.1.2
192.168.1.3
192.168.1.4

New request packet

Give me "image.png"	1	?	192.168.1.1	192.168.1.3

Response


192.168.1.1
Your Phone
1
2
3
192.168.1.1
SERVERS
192.168.1.2
192.168.1.3
192.168.1.4
Response Packet

Response Packet

file - "image.png"	?	1	192.168.1.3	192.168.1.1

Scenario: Suppose there are 2 tabs open in your browser. Both are connected to youtube.

Is our solution capable of handling it?

Those numbers that we assigned to each application are called PORT NUMBERS.

In every packet, source and destination port numbers are also added.

Structure of our packet

Coming back to our question…

How do 2 machines talk to each other?

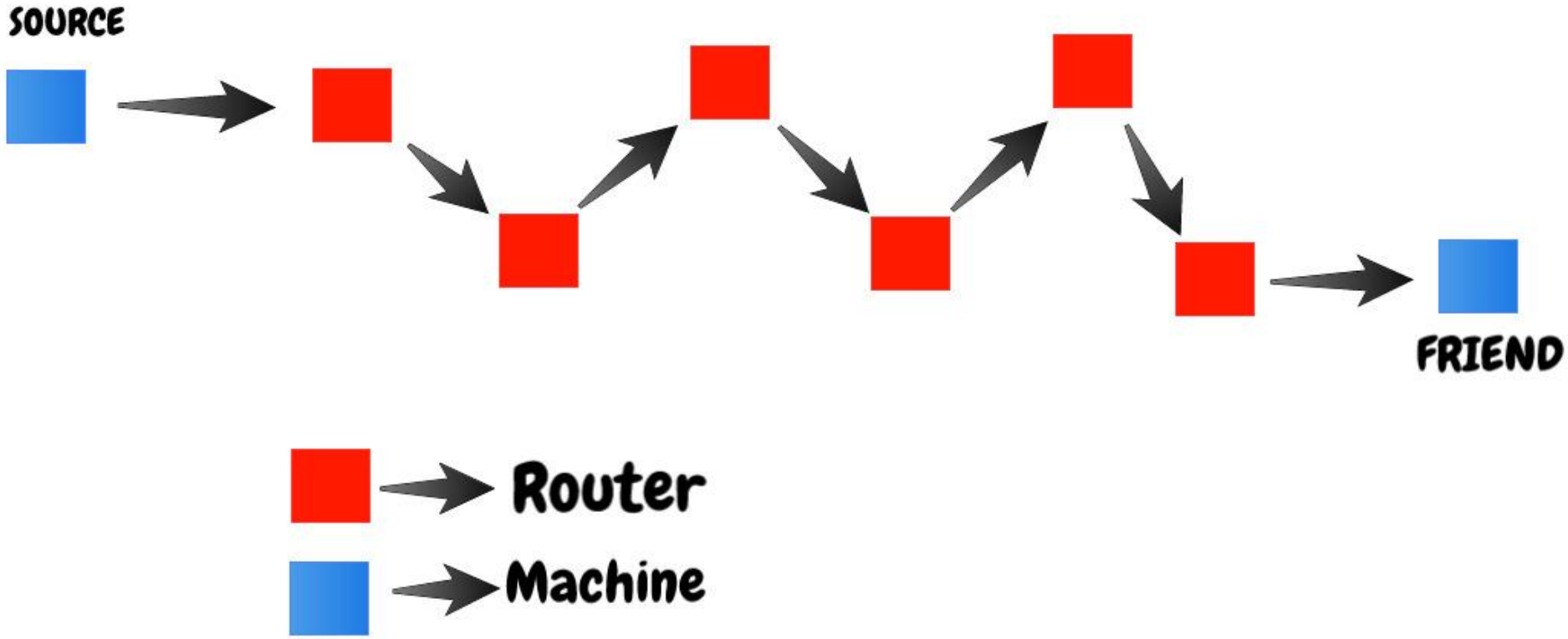
Do you now have an answer for that?

Let us write our own client and server programs and see 2 machines talk to each other live!

Just keep in mind that the pair (IP Address, Port Number) is important.

Let us get Started!

DO YOU HAVE TO TAKE CARE OF ALL THESE DETAILS?



Nope :)

This is similar to postal system. Once you put the post into the postbox, you don't have to worry about anything else.

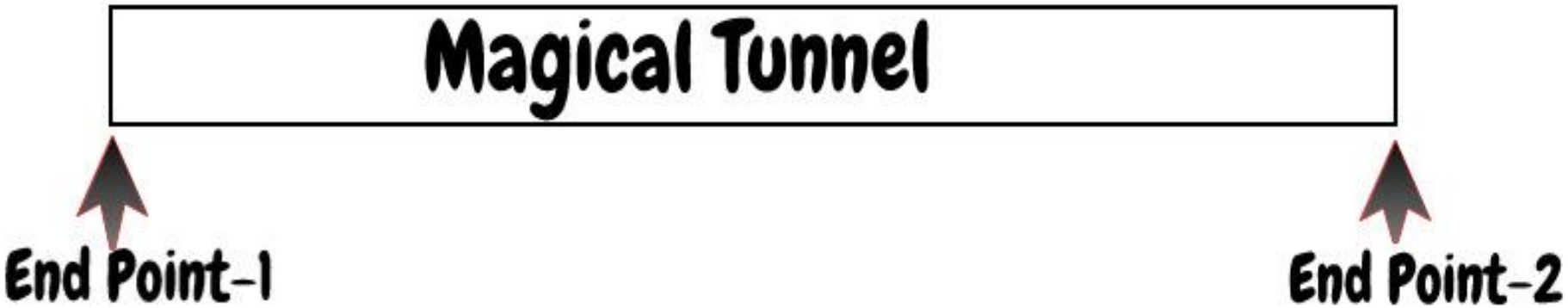
What is a Socket?

What if Internet looked like this and you just have to send and receive data to and from that tunnel?

Researchers and developers have put in efforts to make that possible.

It is in fact a magical tunnel which will do everything for us!!!

End Points of a tunnel



Those End-Points are called 'sockets'

Like an Electric Socket

What work do you have to do?

1. Create those sockets / endpoints.
2. Specify IPAddress and Port Number for each socket.
3. Write code for data transfer to happen.

PRACTICALS!

Let us first write our own Server!

What all should a server do?

What all should a server do?

1. Listen to incoming connections from various clients.

What all should a server do?

1. Listen to incoming connections from various clients.
2. If there is enough resources, accept those connections. Else, reject them.

What all should a server do?

1. Listen to incoming connections from various clients.
2. If there is enough resources, accept those connections. Else, reject them.
3. Once connection accepted, start serving the client’s requests.

Creation of a socket!

```
ListenSocket = socket.socket()
```

Find your machine's IP Address - ServerIPAddress

Use “ifconfig” in Linux and Mac,
“ipconfig” in Windows

ServerPair = (ServerIPAddress, 1234)

```
ServerPair = (ServerIPAddress, 1234)
```

Bind that socket to ServerPair

```
ListenSocket.bind(ServerPair)
```

Start LISTENING for incoming connection requests!

```
RequestQueueSize = 1
ListenSocket.listen(RequestQueueSize)
```

If you have enough resources,

Accept a request!

```
(ClientHandleSocket, ClientPair) = ListenSocket.accept()
```

Use the New socket to talk to the client!

Use send() and recv() functions

Receive the data sent by client.

Wait till client sends something

```
BufferSize = 10000
ClientData = ClientHandleSocket.recv(BufferSize)
```

Process the Client data and SEND back a response

```
ClientHandleSocket.send(SendData)
```

Don't forget to CLOSE the socket once done!

```
ClientHandleSocket.close()   # Connection terminated
```

If you want, close the server by closing the listening socket

```
ListenSocket.close()    # Server program terminated
```

Our tiny Server is done!

Open up “server.py” provided, read it and run it!

Now, the client!

Creation of a socket!

```
ClientSocket = socket.socket()
```

Remember the ServerPair

```
ServerPair = (ServerIPAddress, 1234)
```

Go ahead and Connect to server!

(Make sure server is running :P)

```
ClientSocket.connect(ServerPair)
```

Assuming connection is established, SEND data to server.

```
Prepare SendData
ClientSocket.send(SendData)
```

Wait for Server to send data.

Receive it!

```
BufferSize = 10000
ServerData = ClientSocket.recv(BufferSize)
```

Close the socket once done.

```
ClientSocket.close()    # Connection terminated
```

Our tiny client is done :)

Open up “client.py”, read it and run it!

Play around with the server and client and understand what is happening

We are now done with all the basics :)

Improvise?

Can we make our server better?

Tasks!

Task1: Server should be able to handle multiple connection requests.

Task2(final one :P): Can we write a small chat program?

Questions?

Suggestions, feedback?

Thank you :-)

Contact us

Adwaith Gautham

1. Email ID: adwait.gautham@gmail.com
2. Phone: +91 9663572932

Download the presentation and code from here:

Link:https://github.com/adwait1-G/How-does-the-Internet-work