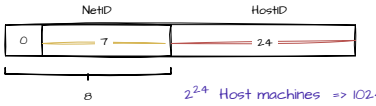
Class A

First bit zero

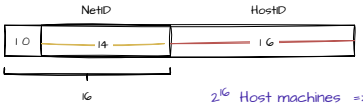


2^{24} Host machines => 1024 * 1024 * 16

2^{7} Networks => 128

Class B

Second bit zero

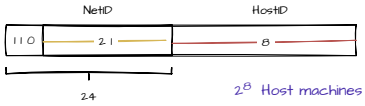


2^{16} Host machines => 1024 * 64

2^{14} Networks => 1024 * 16

Class C

Third bit zero

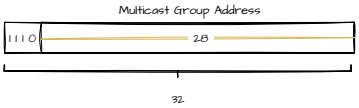


2^{8} Host machines => 256

2^{21} Networks => 1024 * 1024 * 2

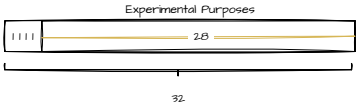
Class D

Fourth bit zero



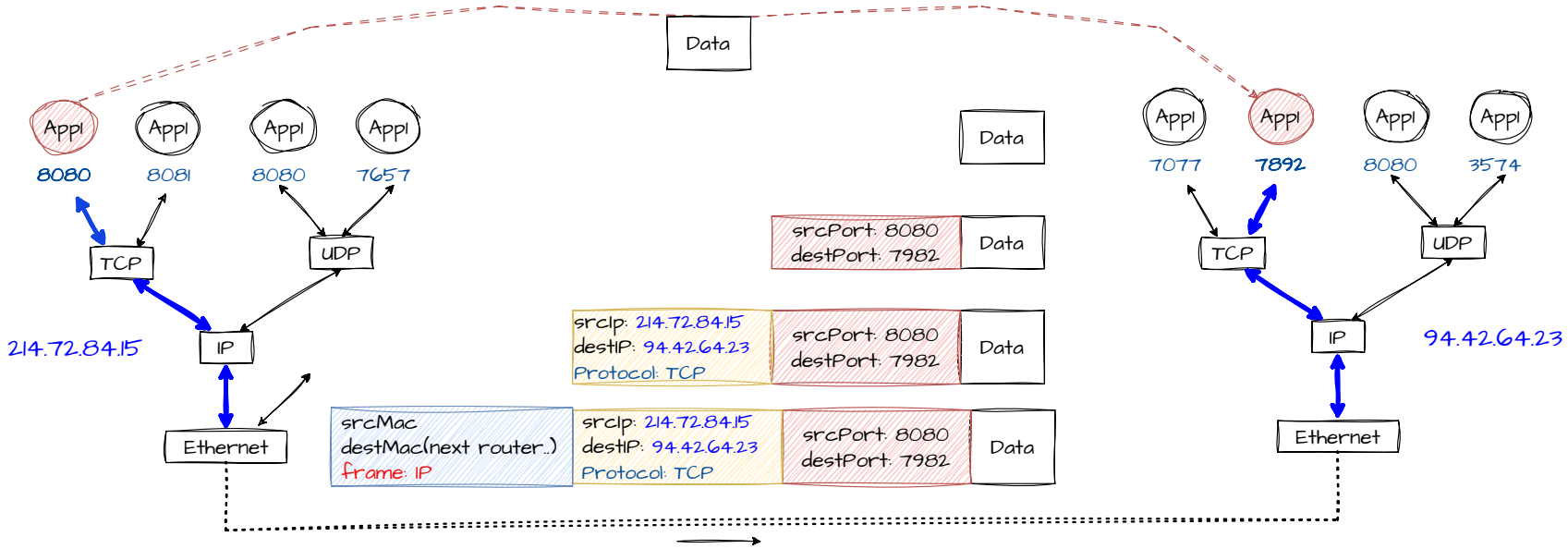
Class E

All four ones



IP Address Ranges

Class	Leading Bits	Address Range	Network Bits	Host Bits	Intended Use
A	0	0.0.0.0 – 127.255.255.255	8	24	Large networks
B	10	128.0.0.0 – 191.255.255.255	16	16	Medium networks
C	110	192.0.0.0 – 223.255.255.255	24	8	Small networks
D	1110	224.0.0.0 – 239.255.255.255	Multicast	N/A	Multicast groups
E	1111	240.0.0.0 – 255.255.255.255	Reserved	N/A	Research/Experimental


Data
Appl
Appl
Appl
Appl
8080
8081
8080
7657
TCP
UDP
IP
214.72.84.15
Ethernet
srcMac
destMac(next router..)
frame: IP
Data
srcPort: 8080
destPort: 7982
Data
srcIp: 214.72.84.15
destIP: 94.42.64.23
Protocol: TCP
srcPort: 8080
destPort: 7982
Data
srcIp: 214.72.84.15
destIP: 94.42.64.23
Protocol: TCP
srcPort: 8080
destPort: 7982
Data
Appl
Appl
Appl
Appl
7077
7892
8080
3574
TCP
UDP
IP
94.42.64.23
Ethernet

Basic TCP Interaction Overview

Server (say IP : 214.72.16.12)

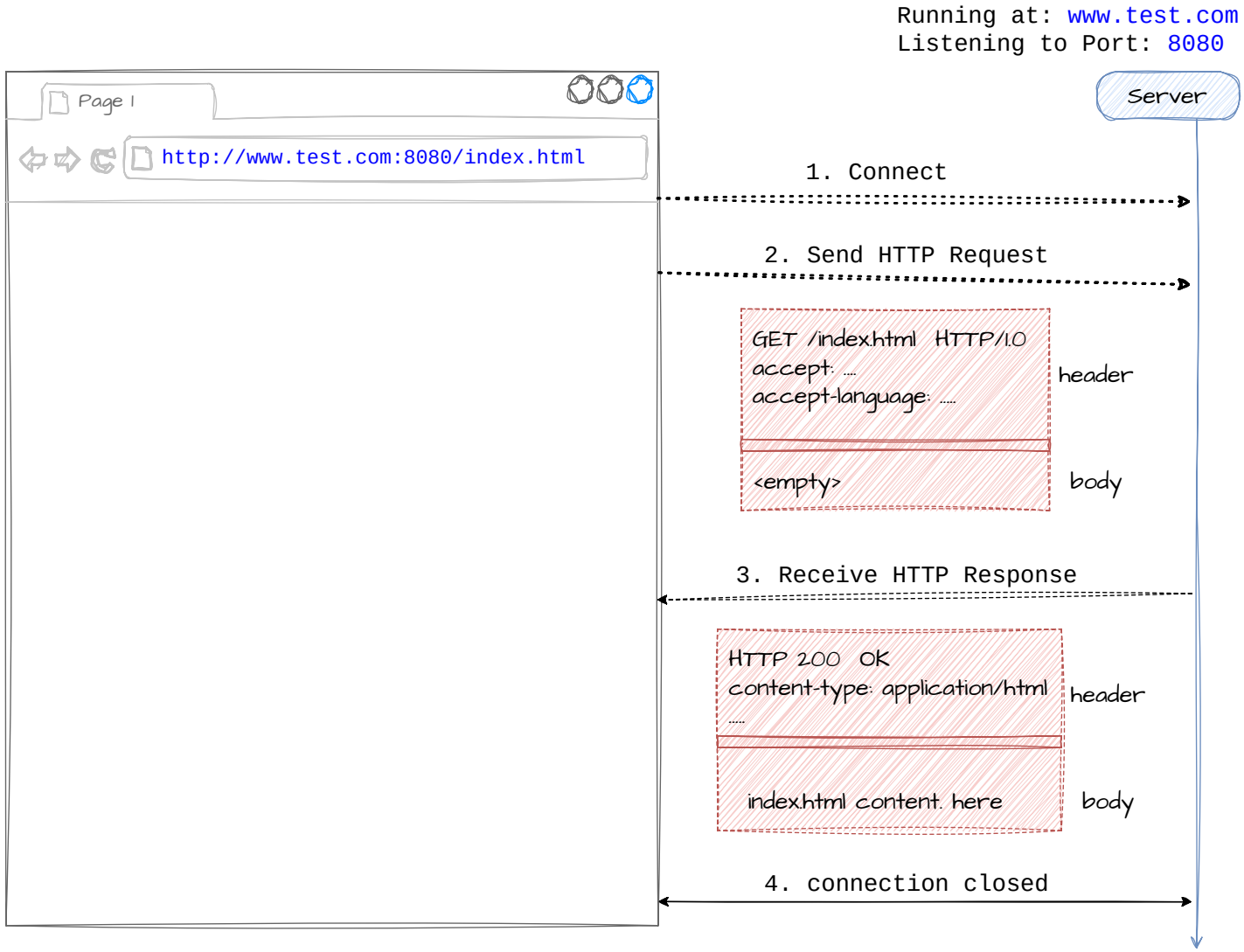
1. Request Port e.g. 8080
   => if success => step 2.
2. Listen to port 8080
   a. Client connected
   b. service the client (sync/async)
   c. close connection

repeat (2)

Client

1. Send connect request to <ServerIP, Port>
   e.g. (ServerIP: 214.72.16.12, Port: 8080)

2. If connected, communicate and once done close the connection.


Running at: www.test.com
Listening to Port: 8080
Page 1
http://www.test.com:8080/index.html
Server
1. Connect
2. Send HTTP Request
GET /index.html HTTP/1.0
accept: ....
accept-language: .....
header
<empty>
body
3. Receive HTTP Response
HTTP 200 OK
content-type: application/html
.....
header
index.html content here
body
4. connection closed