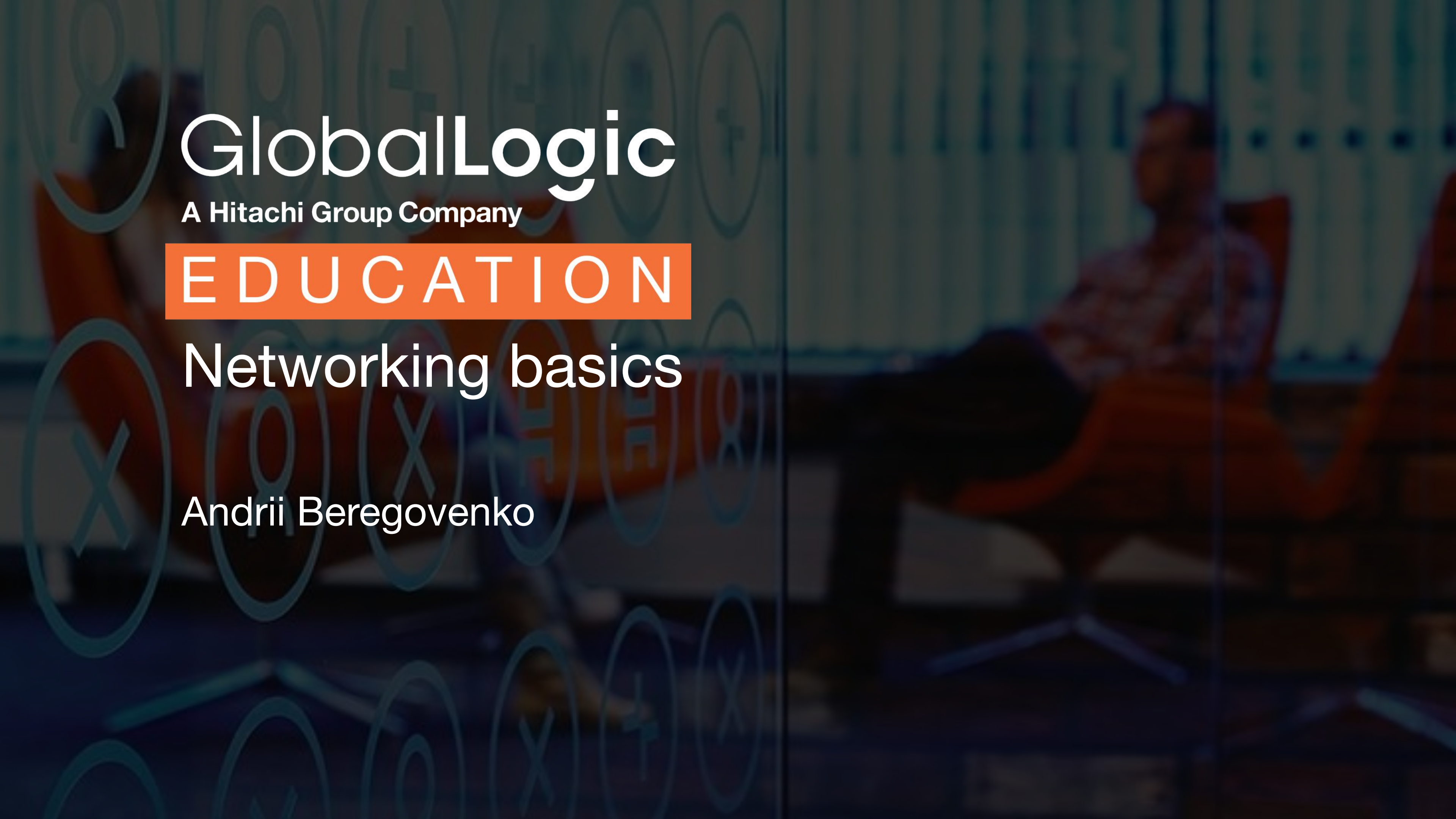


GlobalLogic

A Hitachi Group Company

EDUCATION

Networking basics

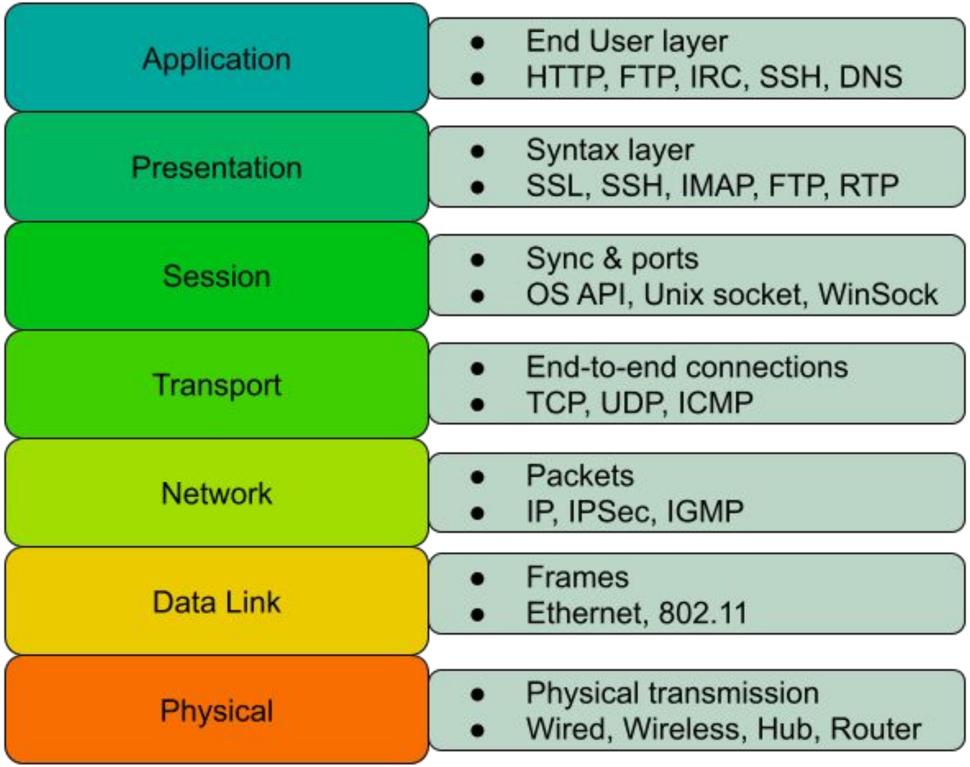
Andrii Beregovenko

Agenda:

- Model OSI
- Network subsystem in OS
- Routing in networks
- Traffic encapsulation
- Traffic manipulation: policing and filtering
- Programming API

Open systems interconnection model

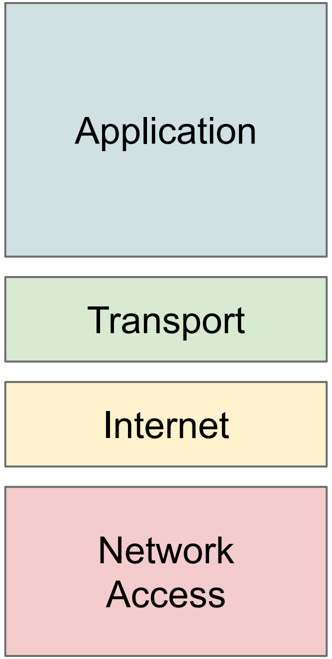
7 layers of the OSI Model



Application	• End User layer • HTTP, FTP, IRC, SSH, DNS
Presentation	• Syntax layer • SSL, SSH, IMAP, FTP, RTP
Session	• Sync & ports • OS API, Unix socket, WinSock
Transport	• End-to-end connections • TCP, UDP, ICMP
Network	• Packets • IP, IPSec, IGMP
Data Link	• Frames • Ethernet, 802.11
Physical	• Physical transmission • Wired, Wireless, Hub, Router

https://en.wikipedia.org/wiki/OSI_model#Layer_architecture

TCP/IP Model



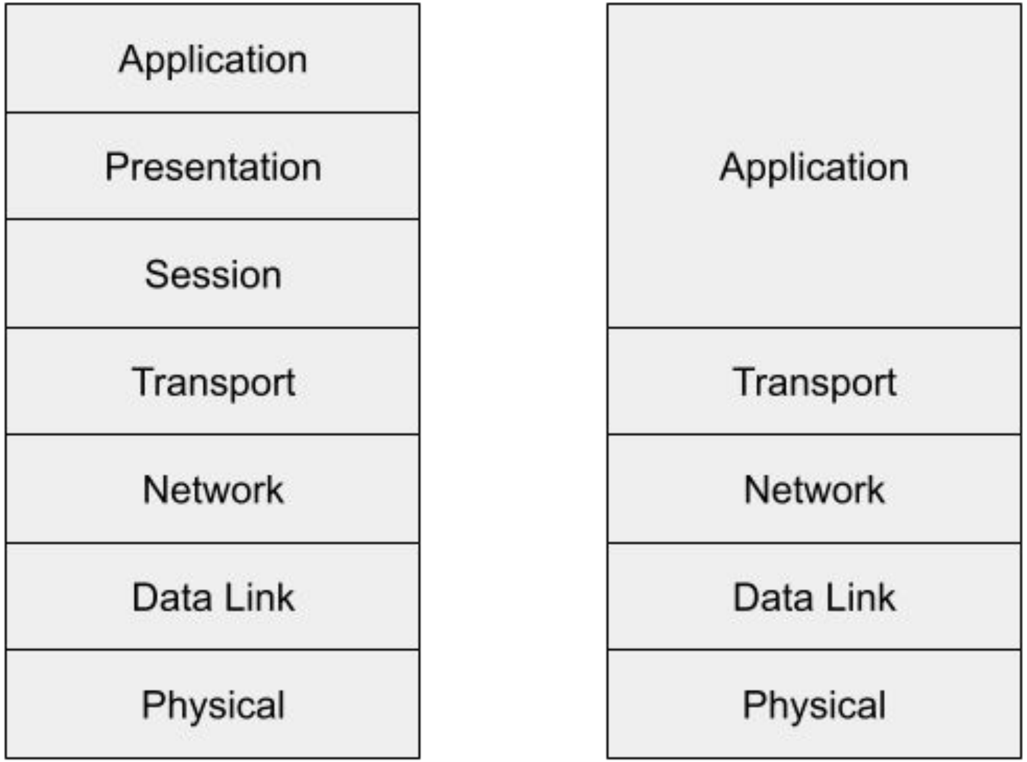
Application

Transport

Internet

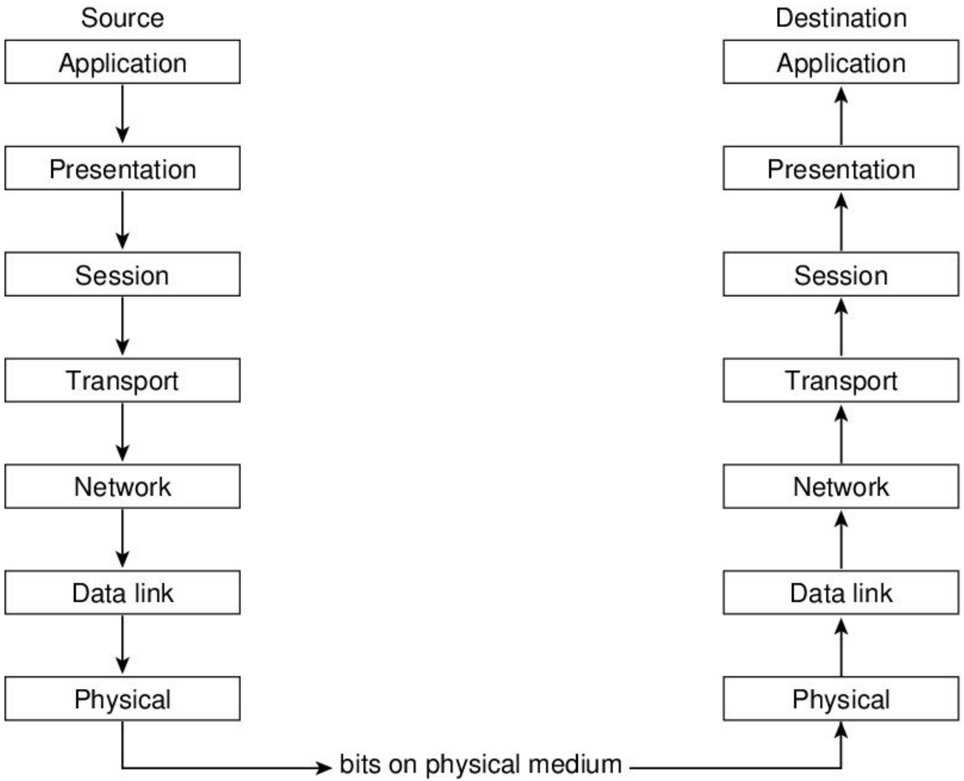
Network Access

OSI vs TCP/IP Model



Application	Application
Presentation	
Session	
Transport	Transport
Network	Network
Data Link	Data Link
Physical	Physical

Data encapsulation path



Source

Application

Presentation

Session

Transport

Network

Data link

Physical

Destination

Application

Presentation

Session

Transport

Network

Data link

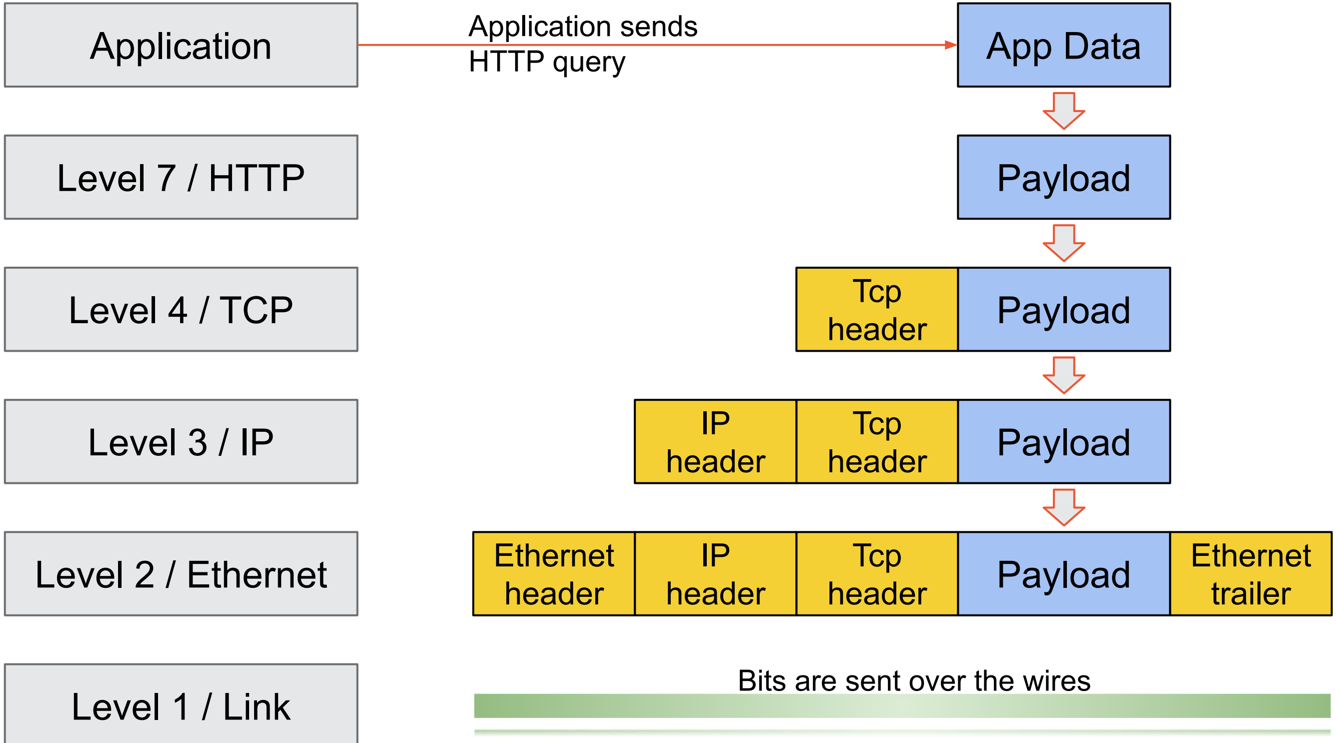
Physical

bits on physical medium

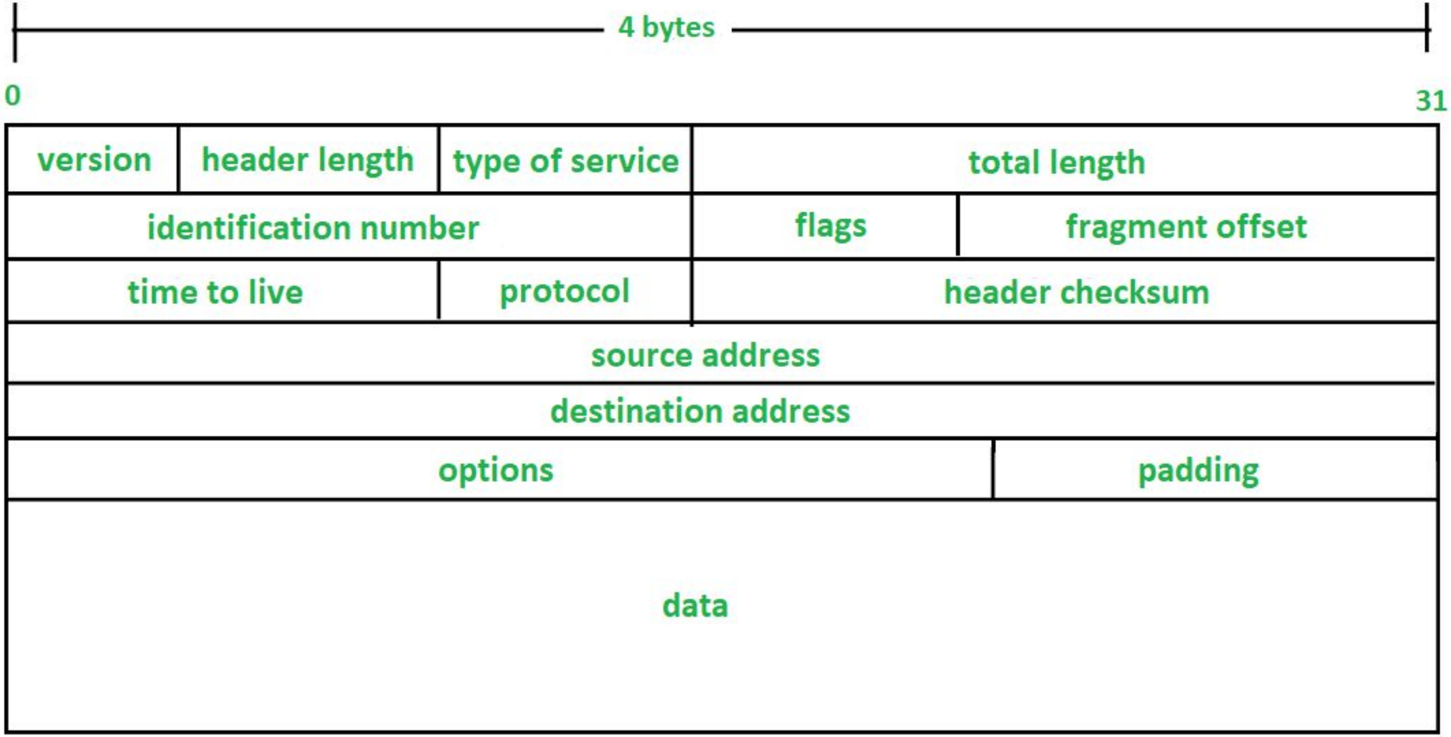
Key principles assumed in model

- Every next/lower level incorporates complete data from previous level
- The data from previous level are ALWAYS treated as a payload, thus:
    - the payload is never analized
    - the payload does not inflict or impact to how level logic operates
    - the only exception is traffic filtering, but this is done by extracting original and reinjecting back modified packets or frames
    - the only important thing about payload is its length
- Every level above/higher never knows what would be lower level
- The only physical level leads a data outside of the system
- The physical level is not really mandatory to make communication work, but obviously you cannot leave system in such case

Example of how the data travels through levels



IP header



4 bytes

0

31

version | header length | type of service | total length

identification number | flags | fragment offset

time to live | protocol | header checksum

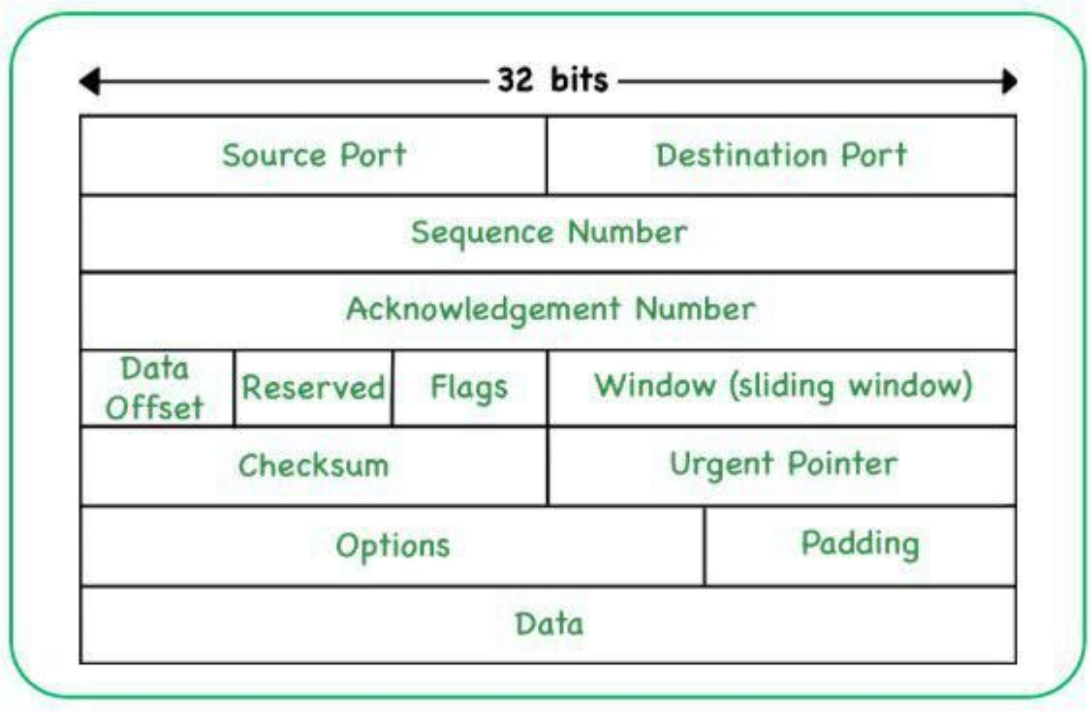
source address

destination address

options | padding

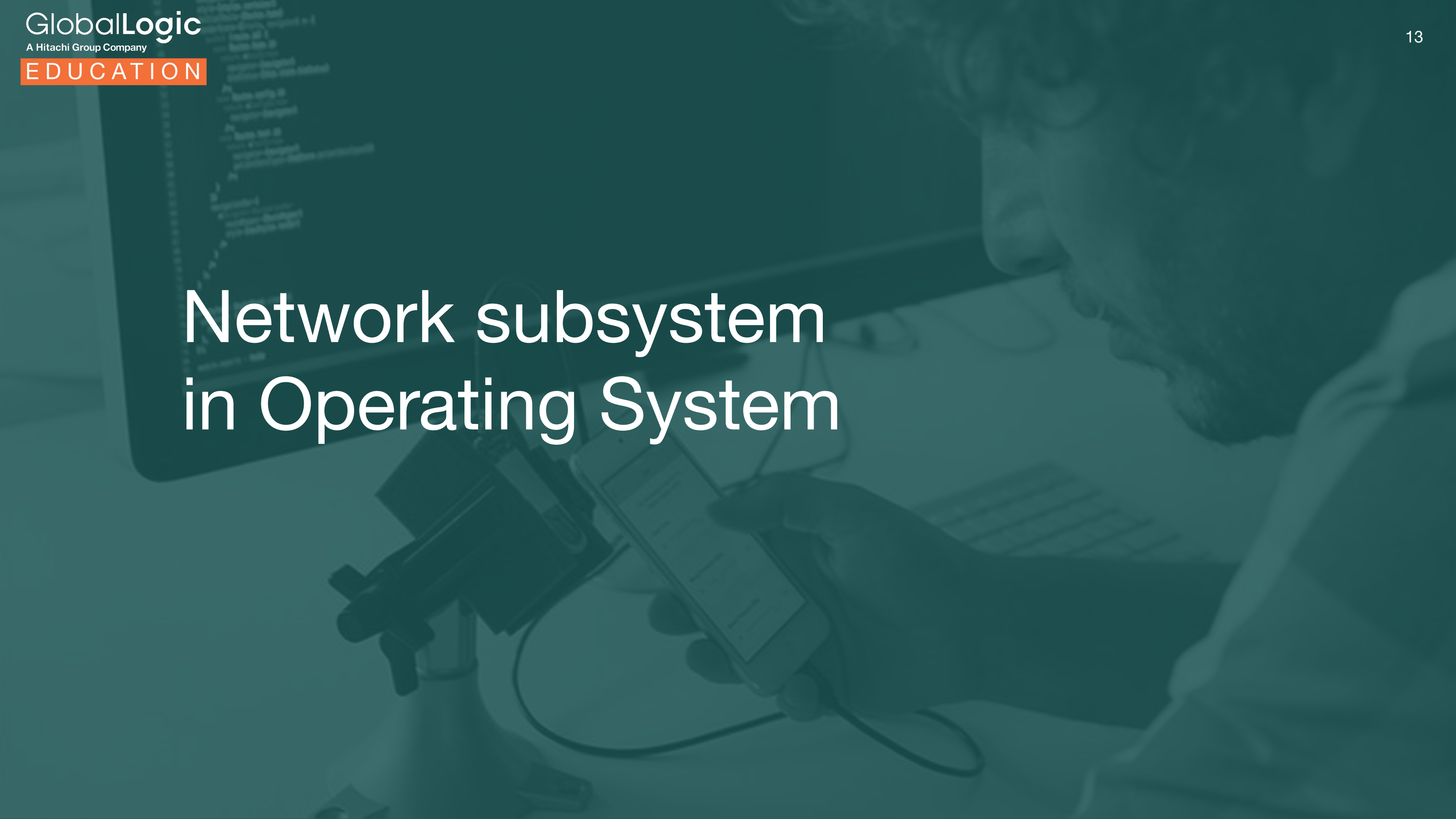
data

TCP header



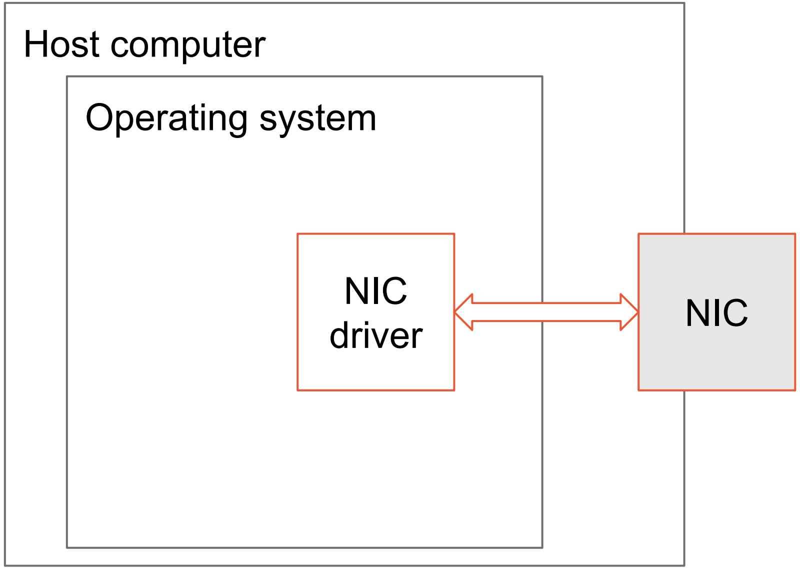
32 bits

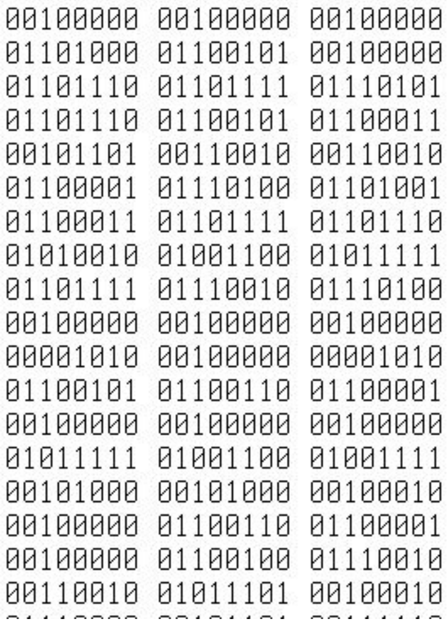
<table>
<tr><td colspan="3">Source Port</td><td colspan="2">Destination Port</td></tr>
<tr><td colspan="5">Sequence Number</td></tr>
<tr><td colspan="5">Acknowledgement Number</td></tr>
<tr><td>Data Offset</td><td>Reserved</td><td>Flags</td><td colspan="2">Window (sliding window)</td></tr>
<tr><td colspan="3">Checksum</td><td colspan="2">Urgent Pointer</td></tr>
<tr><td colspan="4">Options</td><td>Padding</td></tr>
<tr><td colspan="5">Data</td></tr>
</table>



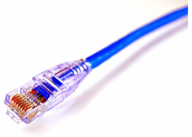
Network subsystem in Operating System

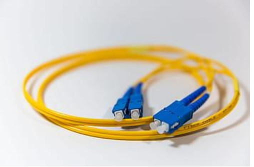
Receiving data from outside



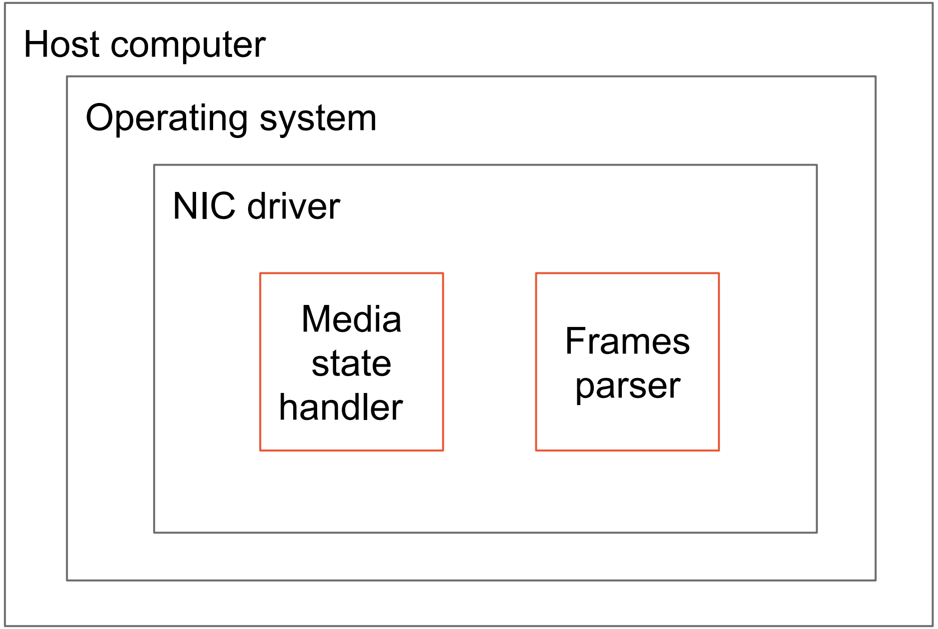


```
00100000 00100000 00100000
01101000 01100101 00100000
01101110 01101111 01110101
01101110 01100101 01100011
00101101 00110010 00110010
01100001 01110100 01101001
01100011 01101111 01101110
01010010 01001100 01011111
01101111 01110010 01110100
00100000 00100000 00100000
00001010 00100000 00001010
01100101 01100110 01100001
00100000 00100000 00100000
01011111 01001100 01001111
00101000 00101000 00100010
00100000 01100110 01100001
00100000 01100100 01110010
00110010 01011101 00100010
```



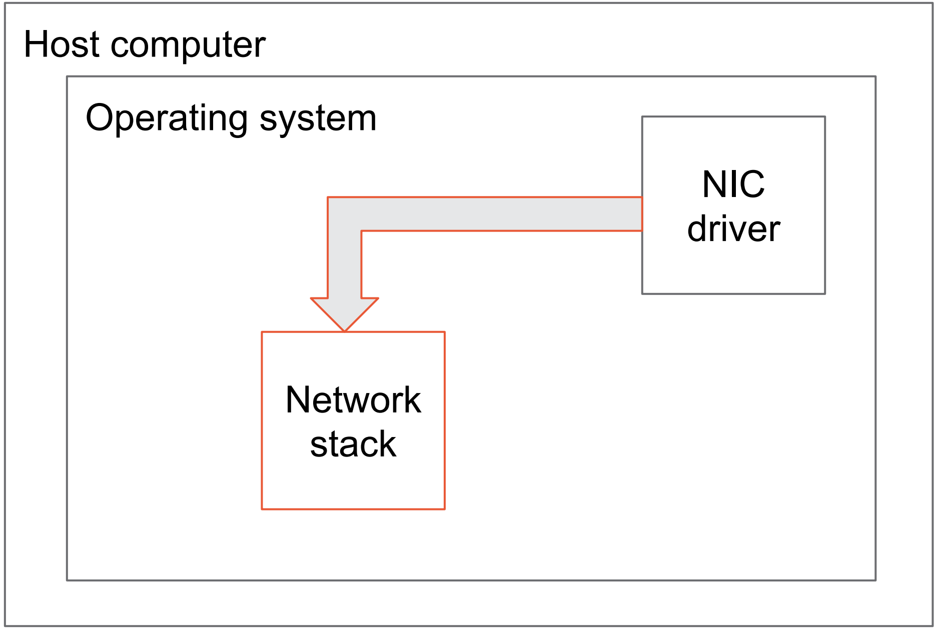


Raw bits are being processed on lowest level



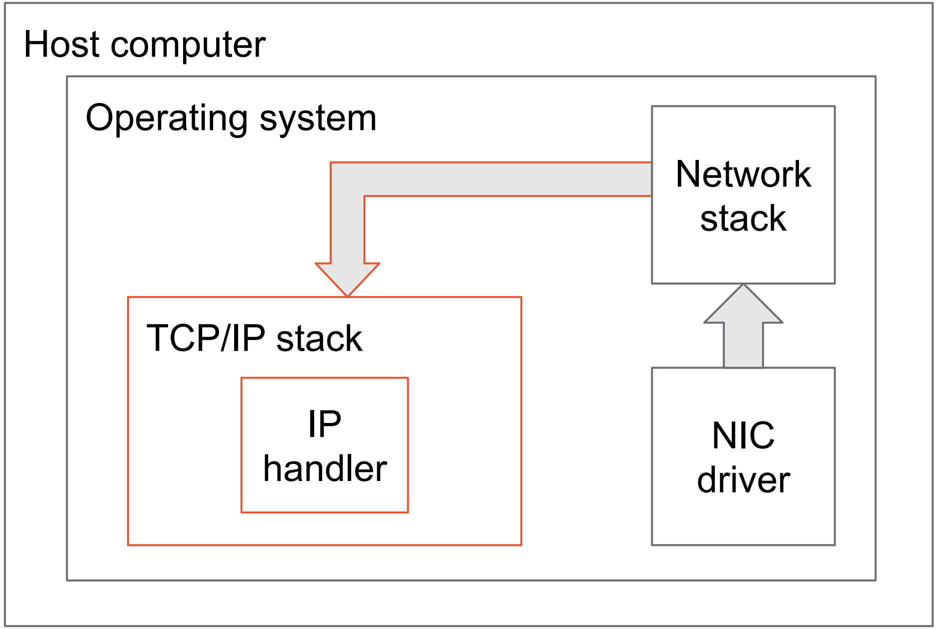
The Network Interface Driver handles a proper state of device, low level information transmission procedures and reads/analise received bit flow

Low level/HW pass network frames to the OS stack



The network stack determines type of frame and where pass the data for further processing base on information from header

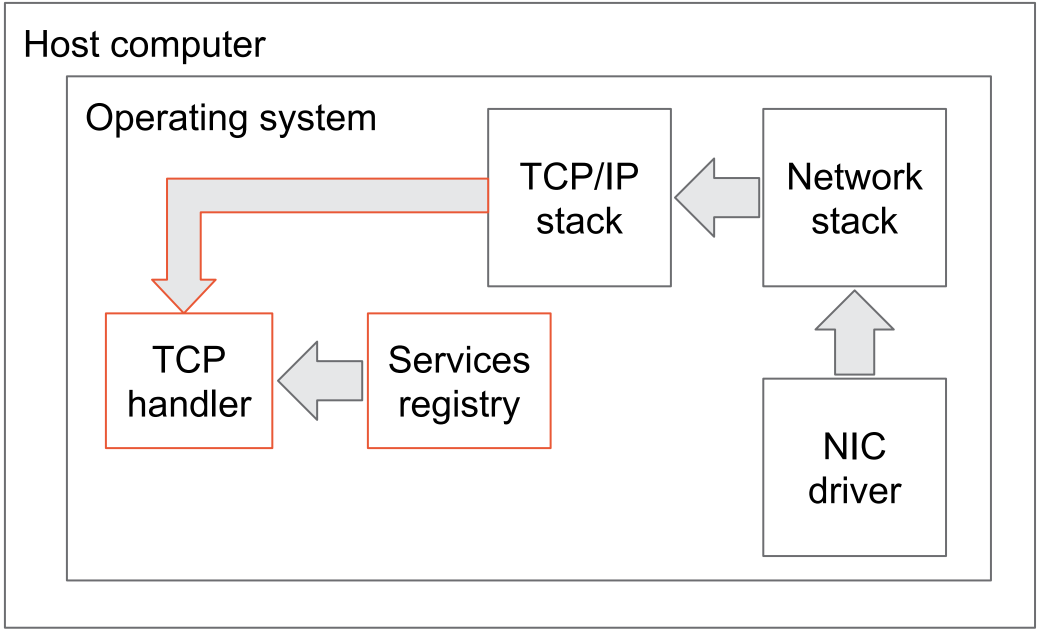
Frames are mangles and packets are formed



The TCP/IP stack, its IP handler, analysez what to do with ip packet:

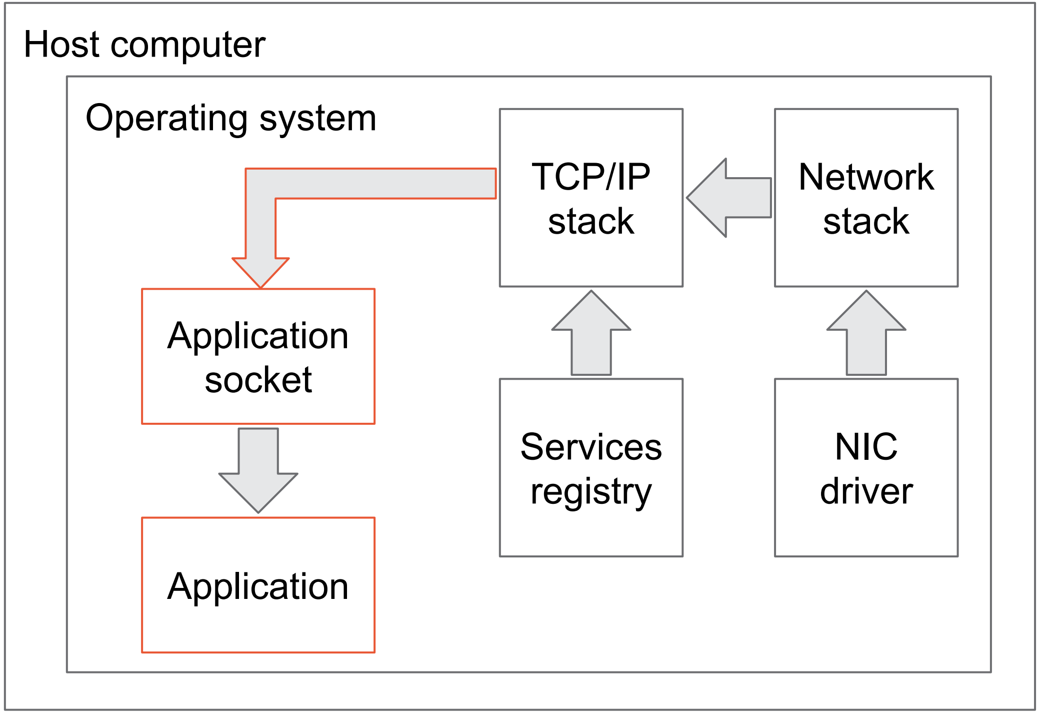
- Consume it on host
- Forward to the other host

Packets are assigned to ports/services



The TCP/IP stack reads from operating system registry of services if there is an application to handle TCP stream

App receives data/payload as a bitstream

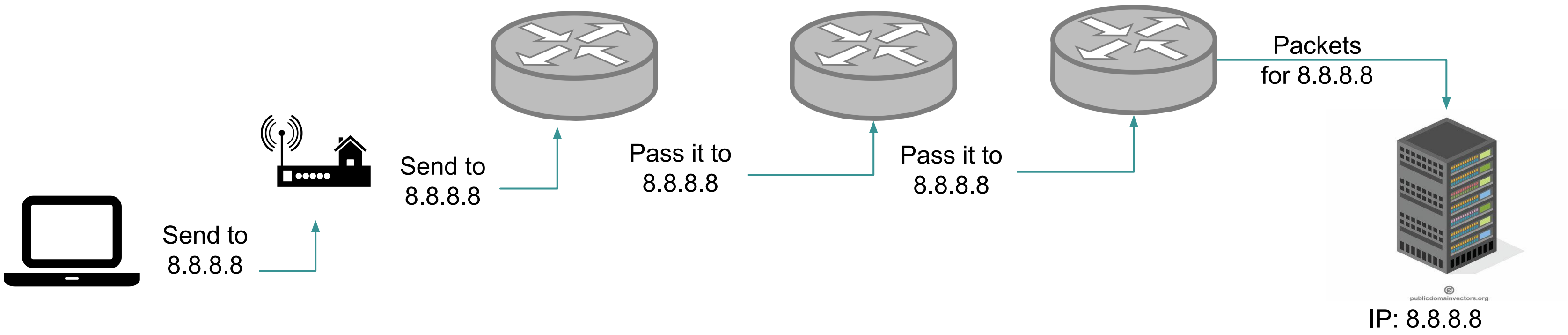


The TCP/IP stack passes the TCP data stream (payload) to associated with service application

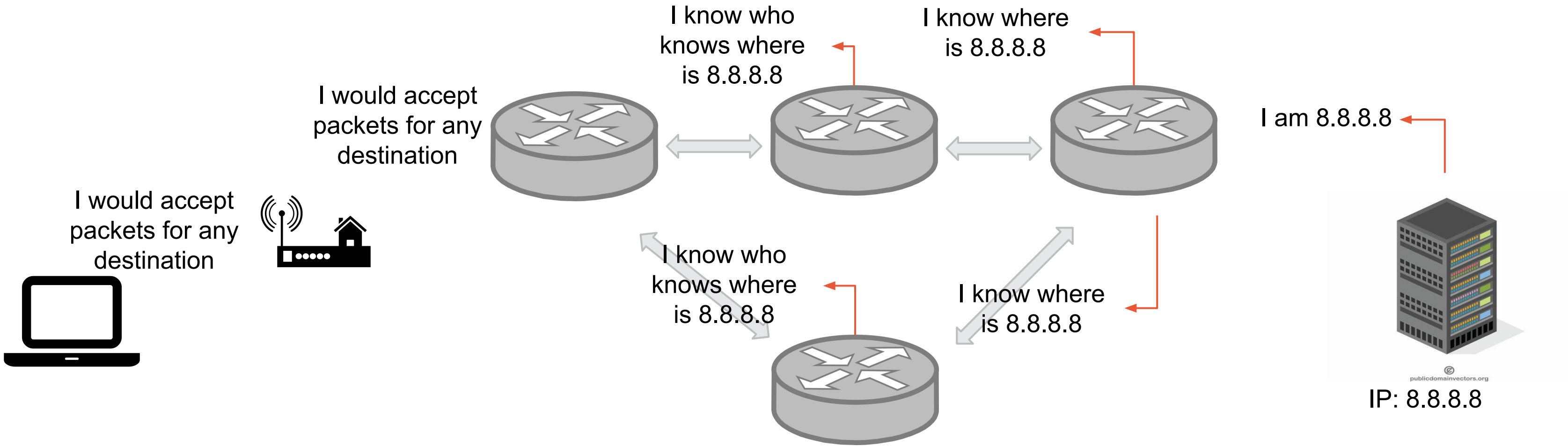
Routing in networks

- Routing of the traffic in the Internet is based mostly on IP protocol
- Hence IP protocol attributes are used to determine the source and destination of the traffic
- The IP protocol traffic is splitted into separate small entities called **IP packets**
- The typical side of the IP packet is 1500 bytes, on magistral network often multiplied size is used 9000 bytes or similar

The IP packets routing in Internet is a process of passing packets through multiple routers, where each of them passes a packet to the next router who is closer to destination.



The information how to reach every existing/reachable destination in IP networks is hardcoded (static routing) or (mostly) is propagated using dynamic routing protocols. This information is called **routing table**



I would accept packets for any destination

I would accept packets for any destination

I know who knows where is 8.8.8.8

I know where is 8.8.8.8

I am 8.8.8.8

I know who knows where is 8.8.8.8

I know where is 8.8.8.8

IP: 8.8.8.8

- The internet routers are exchanging information between each other, so that each participant knows how to reach every destination
- Usually there are multiple ways to reach one destination
- The global routing table is always subjective to the host it belong to
- The global routing table as it is usually is not used, the **views** are used instead
- The **view** is a compilation of the all received routing tables from other routers and processed into a single relative to itself routing table
- While compiling the **local routing table** or **view** other information is being considered, such as link speed, path cost, other factors

To reduce complexity of the Internet all networks has attribute Owner. The Owner is an organization or a person whom this network belongs to. Group of networks handled by a single Owner is called **Autonomous System**. Each autonomous system has following attributes:

- Owner - in 99.9% it is a company that uses a network for its own mostly business purposes
- Autonomous System numer - a short number prepended with prefix “AS”, for instance AS3320 is a Deutsche Telekom AG, AS6432 is Google

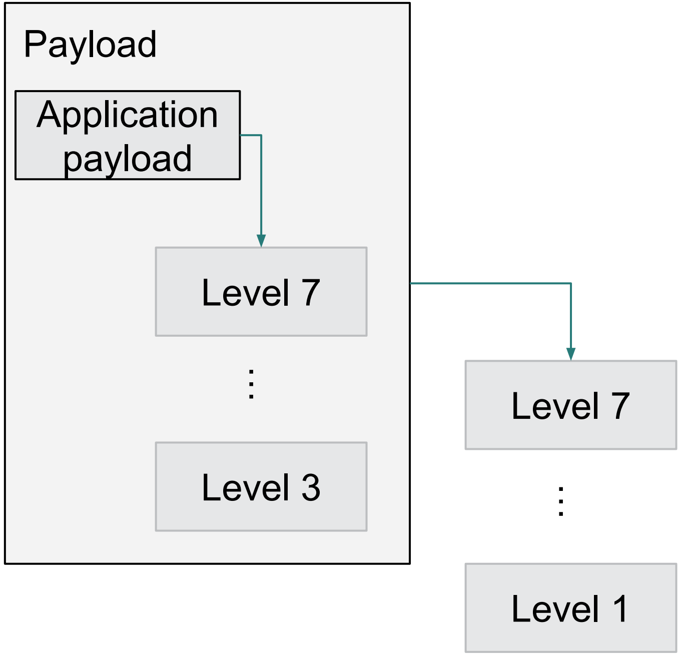
While doing a routing in the Internet on Operator level the packets are traveling through ASes, and IP information mostly is used while entering and leaving Operator backbones

Non standard traffic encapsulation

The non standard traffic encapsulation is a process when traffic is encapsulated with deviation from standard OSI or TCP/IP models.

Typical it is being used for:

- Tunneling (vpn)
- Proxying
- Deep Packet Inspection or Intrusion Prevention Systems
- Transmitting through complicated environment
- Traffic hiding - used in hacking of the computer networks



The typical example is:

- the application payload is processed through the network stack
- At level of IP, the raw packets are started to be treated as a payload for another application
- The second application takes this raw frames as own data
- The second application sends data through the network stack

Traffic manipulation

The standard manipulation operations consist of two procedures:

- Filtering traffic base on policies
- Traffic modification

The traffic manipulation procedures are usually happens on Level 3 (IP) or Level 4 (TCP, UDP, etc).

Thus the typical attributes which are participated are the protocol field values and overall protocol states.

In very rare situations the payload of the higher levels might be analyzed, but this is very resource intensive, inefficient and requires a special hardware to be involved(DPI technique).

Filtering of the traffic is based on the policies:

- Accept packets for further processing
- Reject packets as configured
- Drop packets with no further notification
- Forward packets to another host

The linux does have a special subsystem to do this job, it is managed by the tool **iptables**(it is not the only tool, there are another ones like bpf and netfilter). All packets are placed into a queue and processed one after another. Modern systems can handle tens of Gigabit of the IP traffic.

Specialized hardware manages hundreds of Gigabits of the IP traffic.

Traffic modification is a quite standard procedure in daily operations

- while doing Network Address Translation - going prime private/closed/corporate network to the Internet and vice versa
- while changing a transport media type - transmitting via WiFi and later on through high speed connections like Ethernet and FO
- while redirecting traffic base on policies

The traffic modification usually modify the only headers of some protocols on some levels. For instance, while redirecting an IP packet from one host to another, the host which received it originally changes the destination IP address and than proceeds it as a normal IP packet further, sending it to a desired host.

Programming API

Unix sockets: common API

Lifecycle operations

- socket() - creates a socket for the further use
- bind() - assign a socket to a specific service type and service level
- close() - close the socket

Runtime operations

- read(), recv(), recvmsg() - get the data from the active socket
- write(), send(), sendmsg() - put data to the active socket(to be sent)

System level manipulations

- ioctl() - manipulate unusual socket parameters

Unix socket: service side API

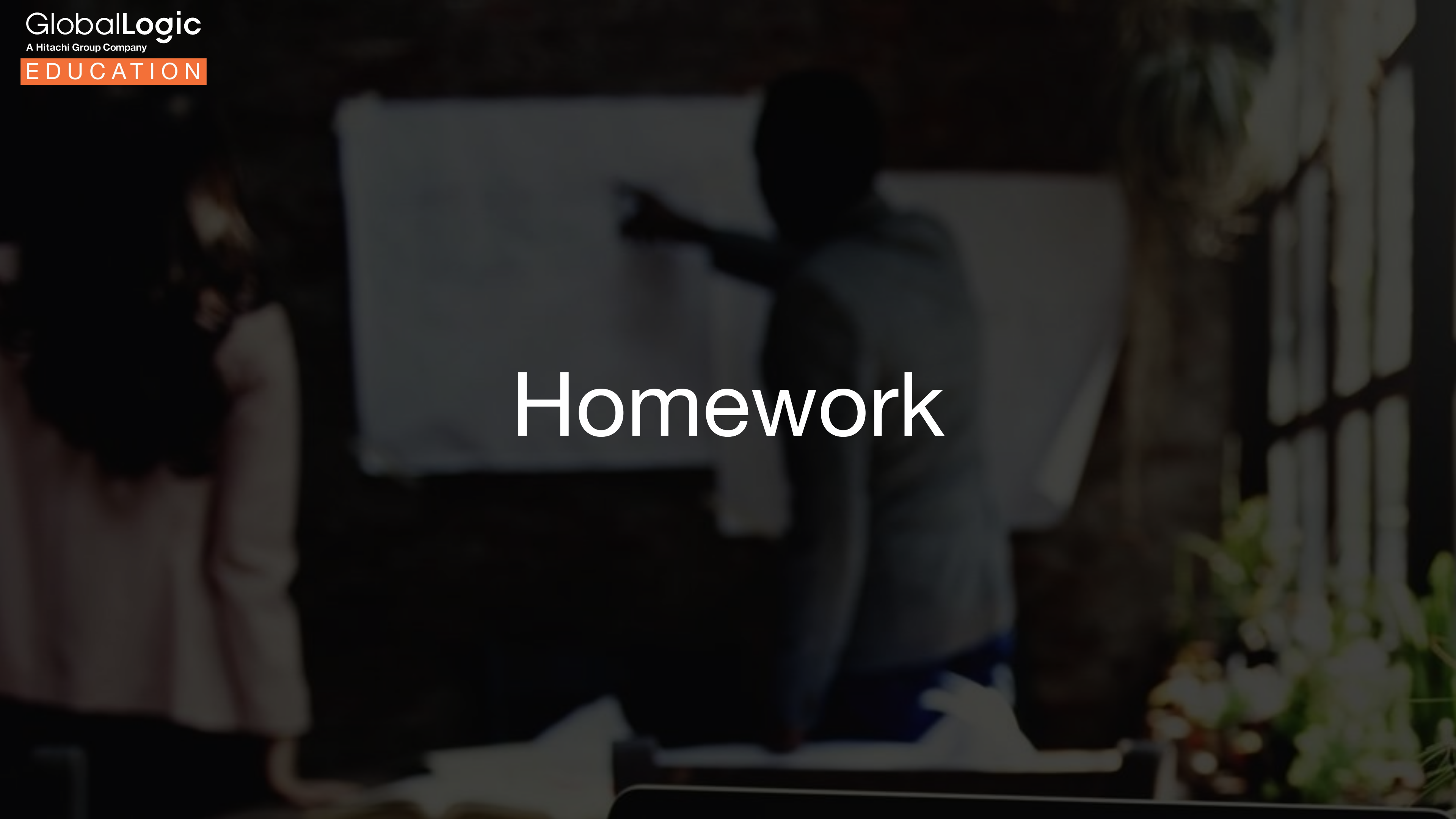
The active socket is required to properly function:

- listen() - start listening for the incoming connection/traffic
- accept() - accept incoming connection is there is a such request
- close() - closes the current connection socket (not listening socket)

Unix socket: client side API

The valid socket is required which is bound to the appropriate transport/network subsystem of the operating system:

- connect() - initiate connection to the remote service
- sendto(), sendmsg() - for stateless connection, send a message
- recvfrom(), recvmsg() - for stateless connection, read the data



Homework

Homework

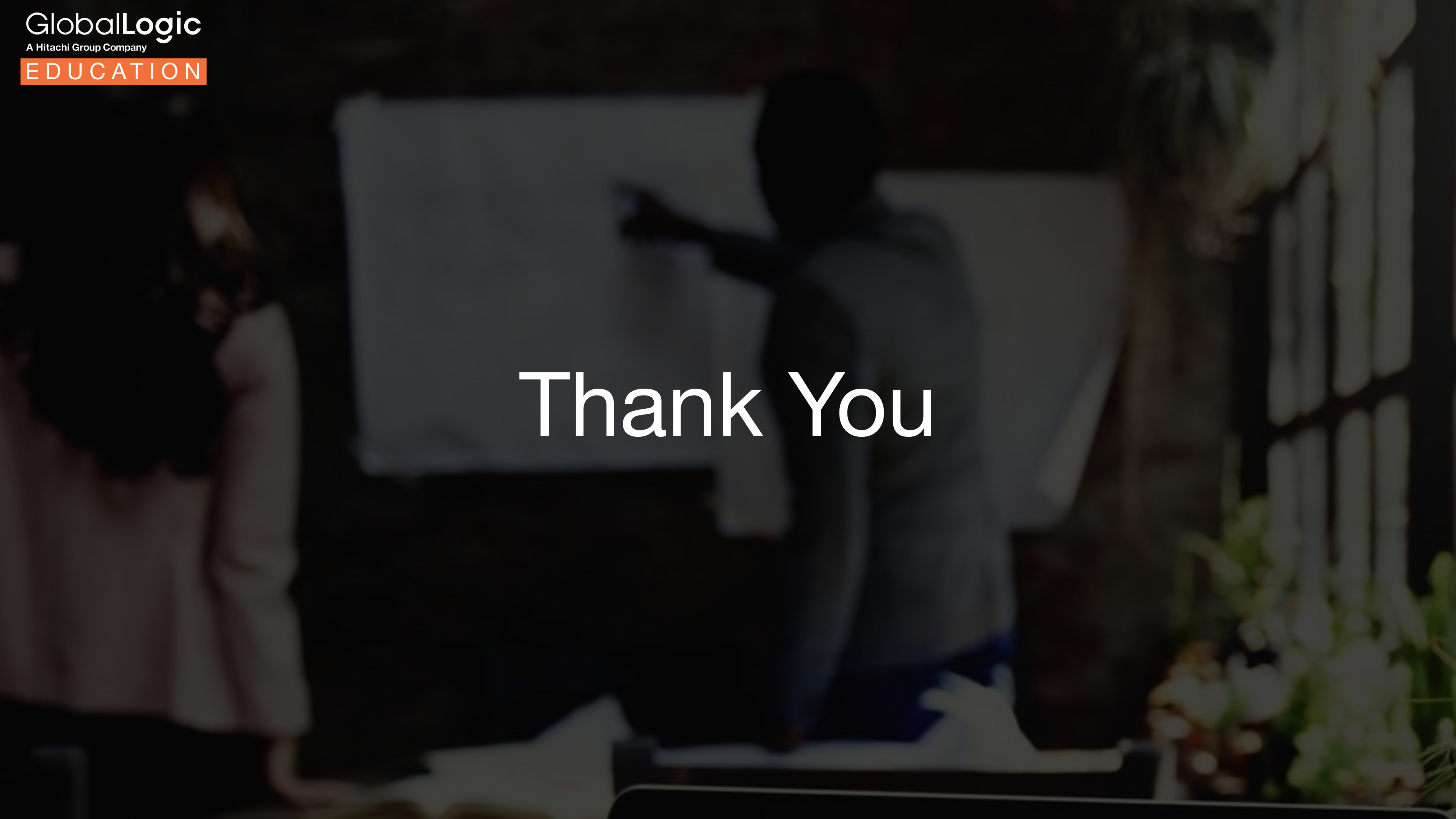
Analize provided packet(shaped as a hex stream) and provide following answers:

- If packet is and IP packet find source and destination addresses
- If the packet was used for TCP stream find source and destination ports
- If the packet was part of HTTP session find a web page address
- If the packet was Internet Control Message Protocol find sequence number

Describe as much as possible packets to get additional points.

Good luck!

Q&A



Thank You