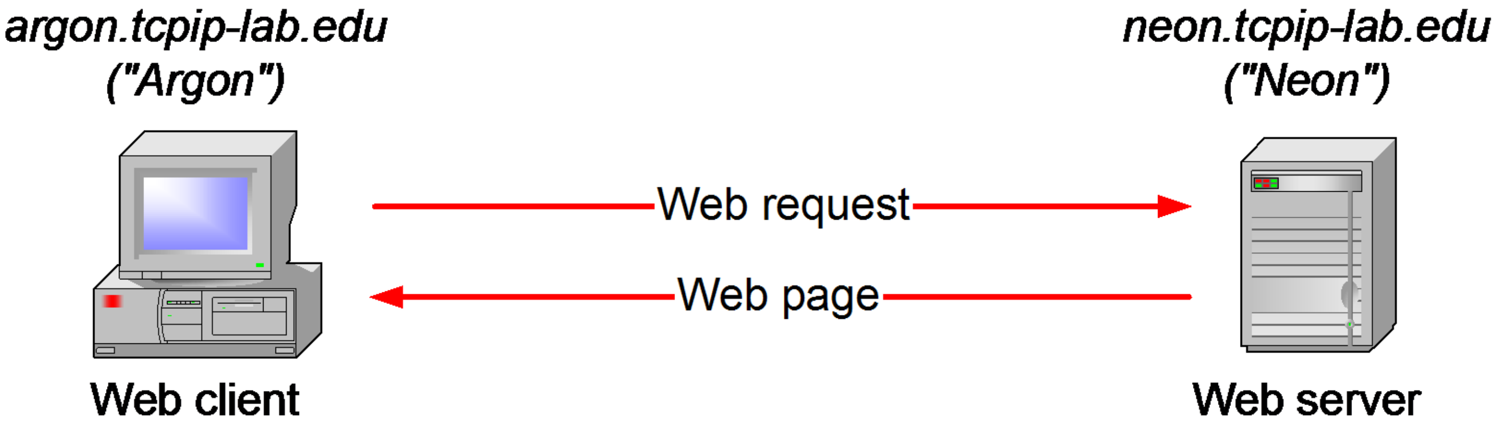
TCP/IP Networking Basics

A simple TCP/IP Example

- A user on host *argon.tcpip-lab.edu* (“*Argon*”) makes a web access to URL

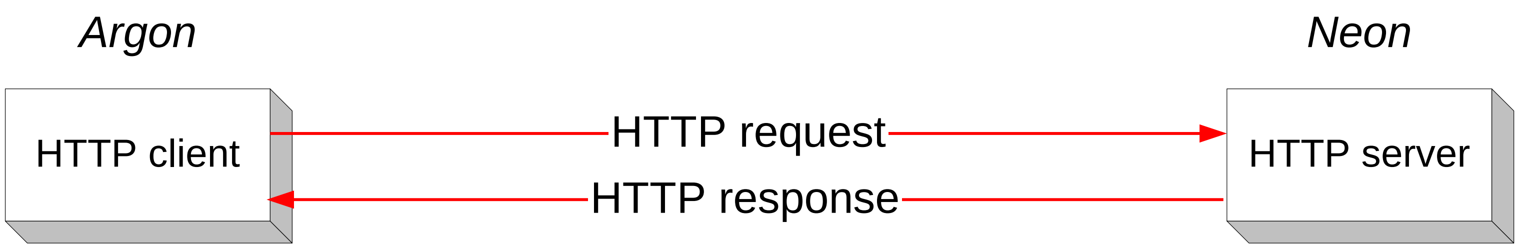
http://neon.tcpip-lab.edu/index.html.



- What actually happens in the network?

HTTP Request and HTTP response

- Web browser runs an HTTP client program
- Web server runs an HTTP server program
- HTTP client sends an HTTP request to HTTP server
- HTTP server responds with HTTP response



HTTP Request

```
GET /index.html HTTP/1.1
Accept: image/gif, */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0
Host: neon.tcpip-lab.edu
Connection: Keep-Alive
```

HTTP Response

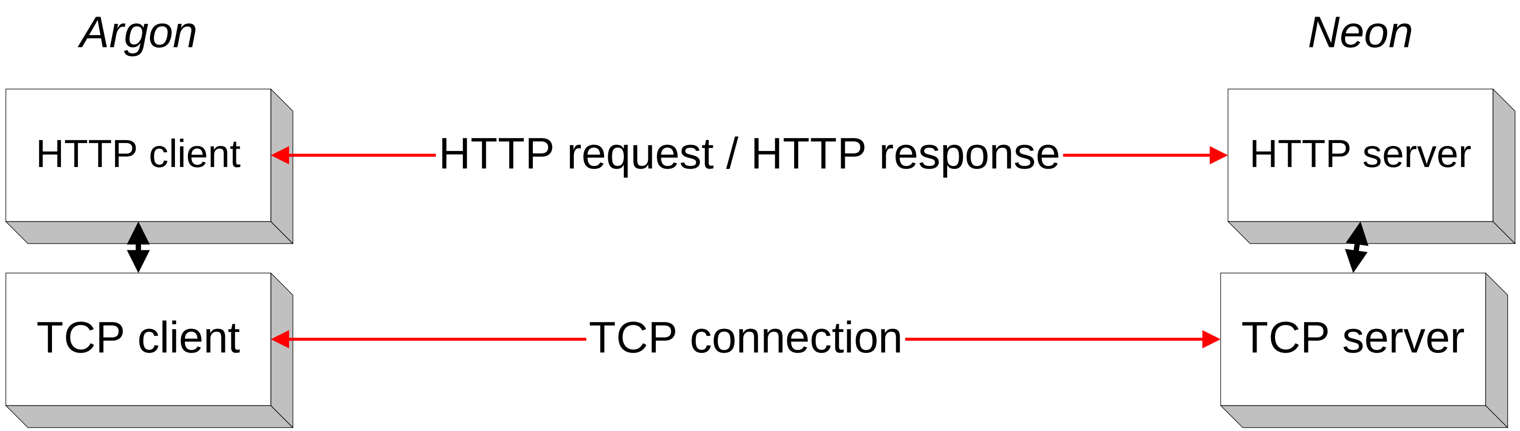
```
HTTP/1.1 200 OK
Date: Sat, 25 May 2002 21:10:32 GMT
Server: Apache/1.3.19 (Unix)
Last-Modified: Sat, 25 May 2002 20:51:33 GMT
ETag: "56497-51-3ceff955"
Accept-Ranges: bytes
Content-Length: 81
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Content-Type: text/html

<HTML>
<BODY>
<H1>Internet Lab</H1>
Click <a href="http://www.tcpip-lab.net/index.html">here</a> for the Internet Lab webpage.
</BODY>
</HTML>
```

- How does the HTTP request get from Argon to Neon ?

From HTTP to TCP

- To send request, HTTP client program establishes an TCP connection to the HTTP server Neon.
- The HTTP server at Neon has a TCP server running



Resolving hostnames and port numbers

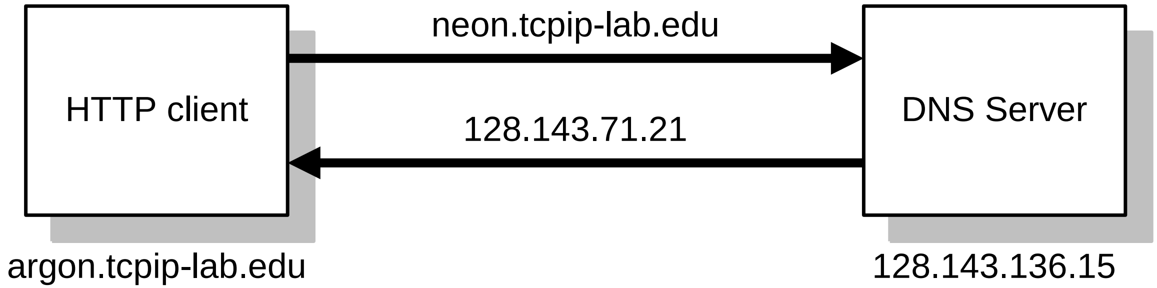
- Since TCP does not work with hostnames and also would not know how to find the HTTP server program at Neon, two things must happen:

1. The name “neon.tcpip-lab.edu” must be translated into a 32-bit **IP address.**

2. The HTTP server at Neon must be identified by a 16-bit **port number**.

Translating a hostname into an IP address

- The translation of the hostname *neon.tcpip-lab.edu* into an IP address is done via a database lookup



- The distributed database used is called the ***Domain Name System (DNS)***
- All machines on the Internet have an IP address:

argon.tcpip-lab.edu	*128.143.137.144*
neon.tcpip-lab.edu	*128.143.71.21*

Finding the port number

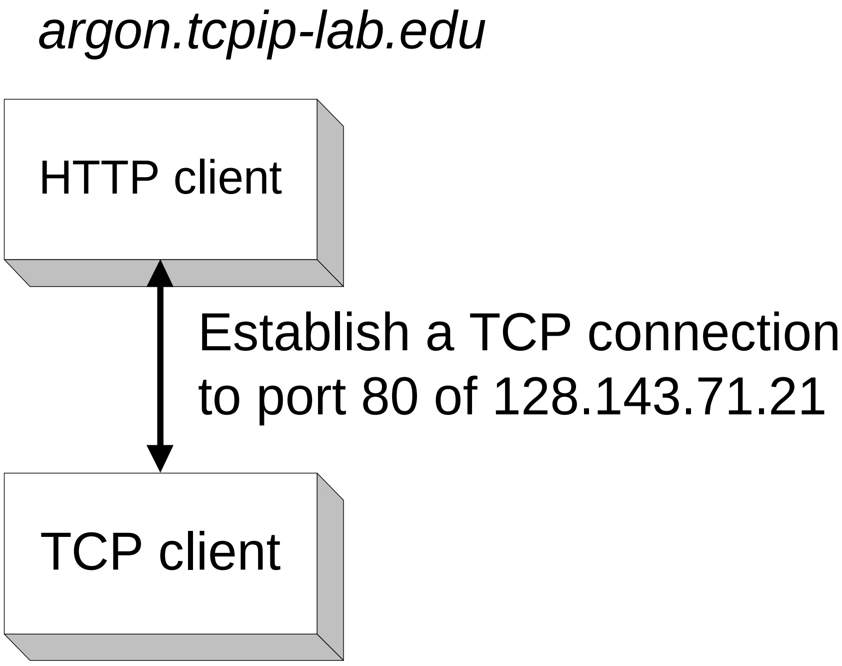
- **Note:** Most services on the Internet are reachable via well-known ports. E.g. All HTTP servers on the Internet can be reached at port number “80”.
- **So:** Argon simply knows the port number of the HTTP server at a remote machine.

- On most Unix systems, the well-known ports are listed in a file with name /etc/services. The well-known port numbers of some of the most popular services are:

ftp	21	finger	79
telnet	23	http	80
smtp	25	nntp	119

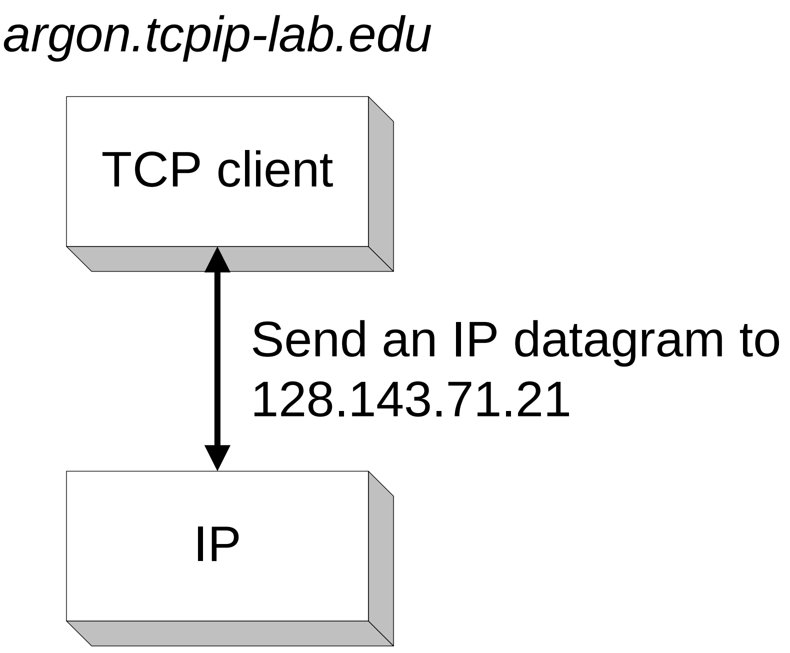
Requesting a TCP Connection

- The HTTP client at *argon.tcpip-lab.edu* requests the TCP client to establish a connection to port 80 of the machine with address 128.141.71.21



Invoking the IP Protocol

- The TCP client at *Argon* sends a request to establish a connection to port 80 at *Neon*
- This is done by asking its local IP module to send an IP datagram to *128.143.71.21*
- *(The data portion of the IP datagram contains the request to open a connection)*



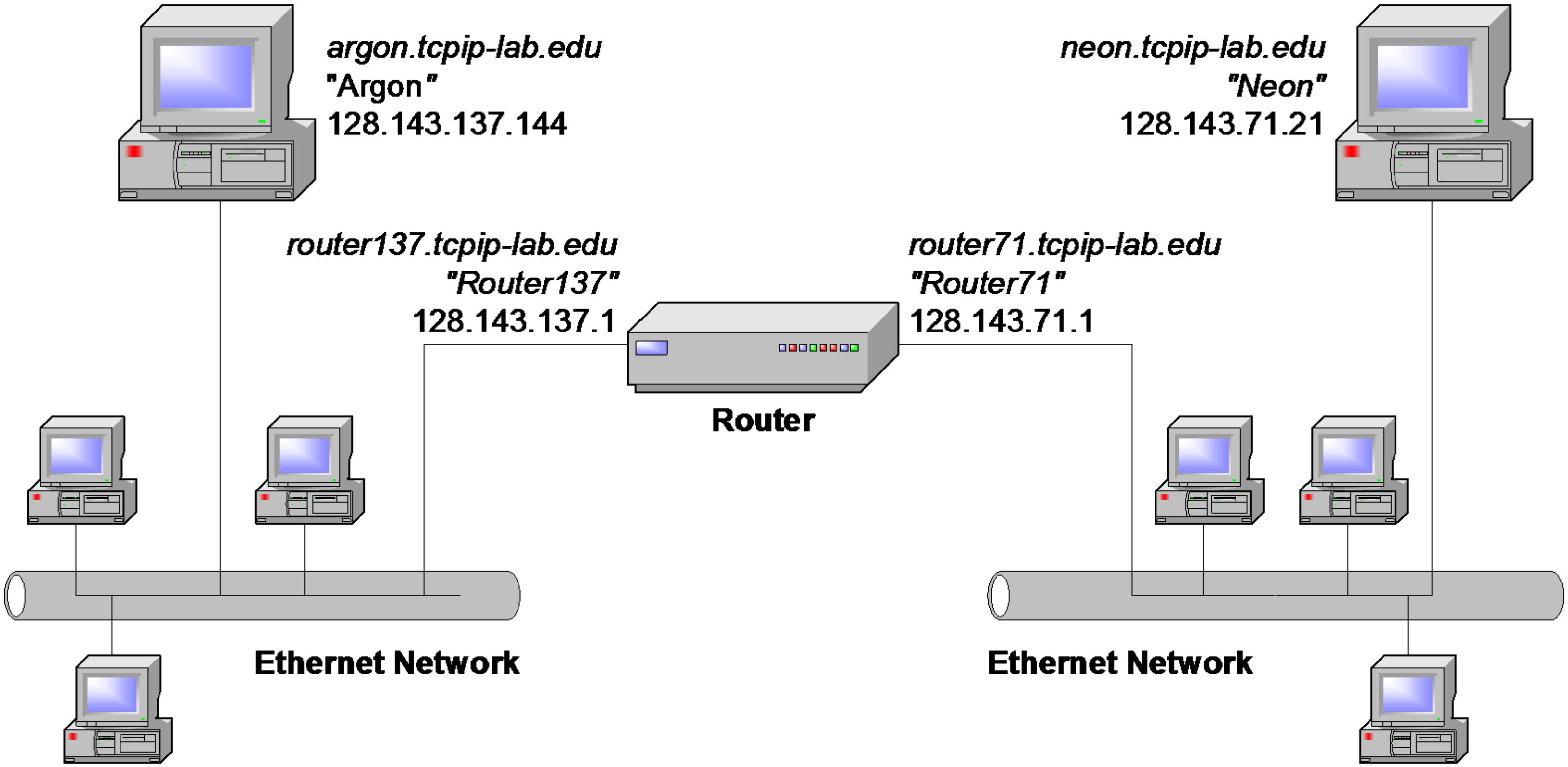
Sending the IP datagram to an IP router

- *Argon (128.143.137.144)* can deliver the IP datagram directly to *Neon (128.143.71.21)*, only if it is on the same local network ("subnet")

- But *Argon* and N*eon* are not on the same local network

- So, *Argon* sends the IP datagram to its default gateway
- The default gateway is an IP router
- The default gateway for *Argon* is *Router137.tcpip-lab.edu* (128.143.137.1).

The route from *Argon* to *Neon*

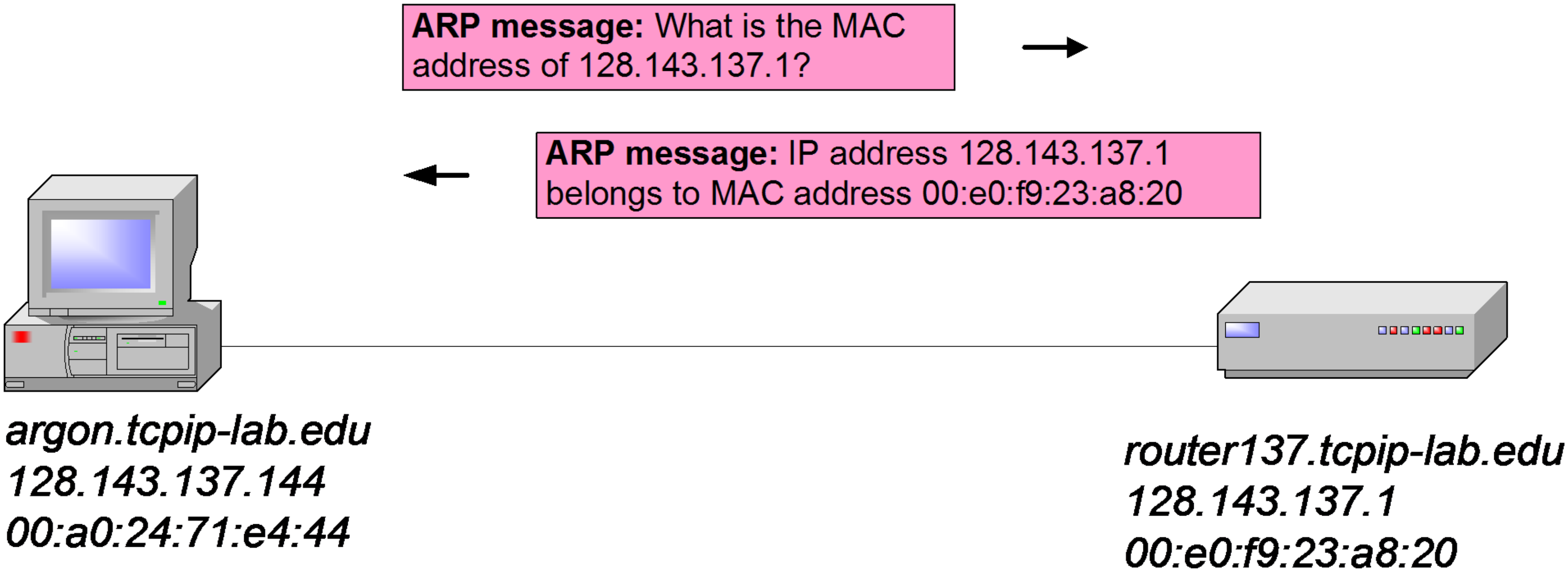


- Note that the gateway has a different name for each of its interfaces.

Finding the MAC address of the gateway

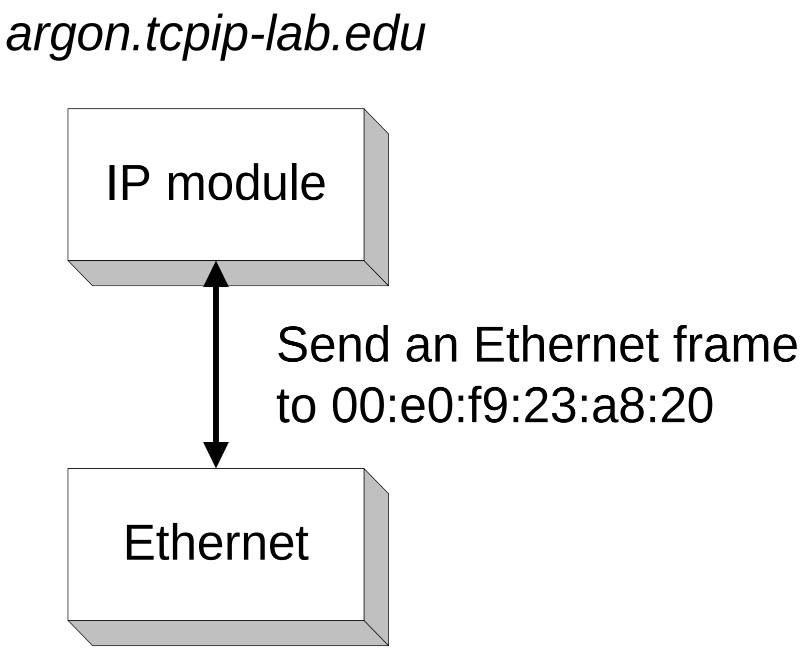
- To send an IP datagram to Router137, *Argon* puts the IP datagram in an Ethernet frame, and transmits the frame.
- However, Ethernet uses different addresses, so-called Media Access Control (MAC) addresses (also called: physical address, hardware address)
- Therefore, *Argon* must first translate the IP address 128.143.137.1 into a MAC address.
- The translation of addressed is performed via the Address Resolution Protocol (ARP)

Address resolution with ARP



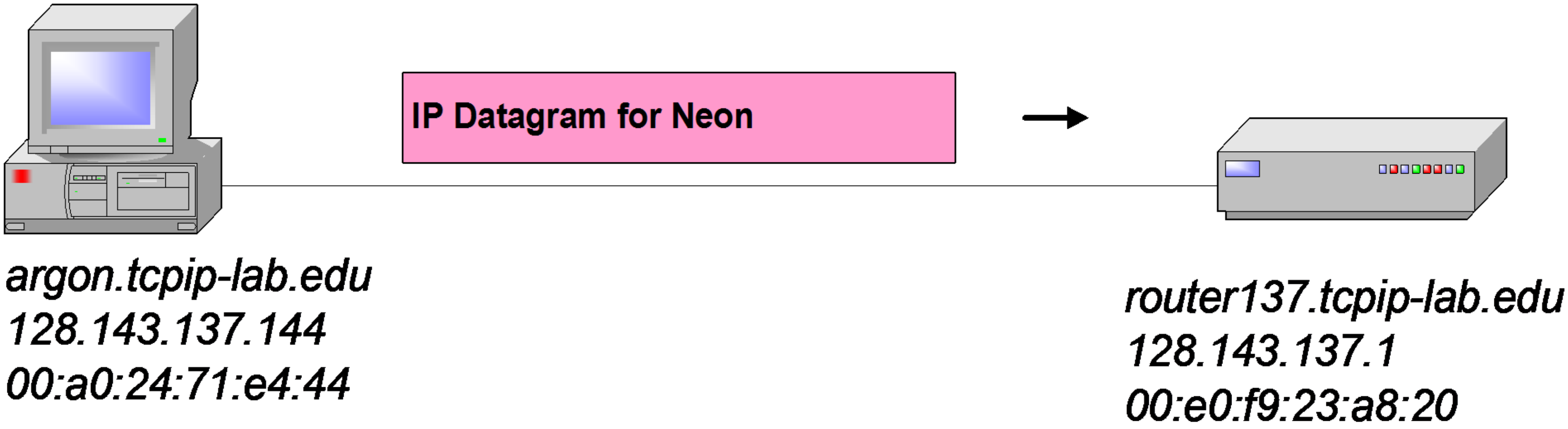
Invoking the device driver

- The IP module at *Argon*, tells its Ethernet device driver to send an Ethernet frame to address *00:e0:f9:23:a8:20*



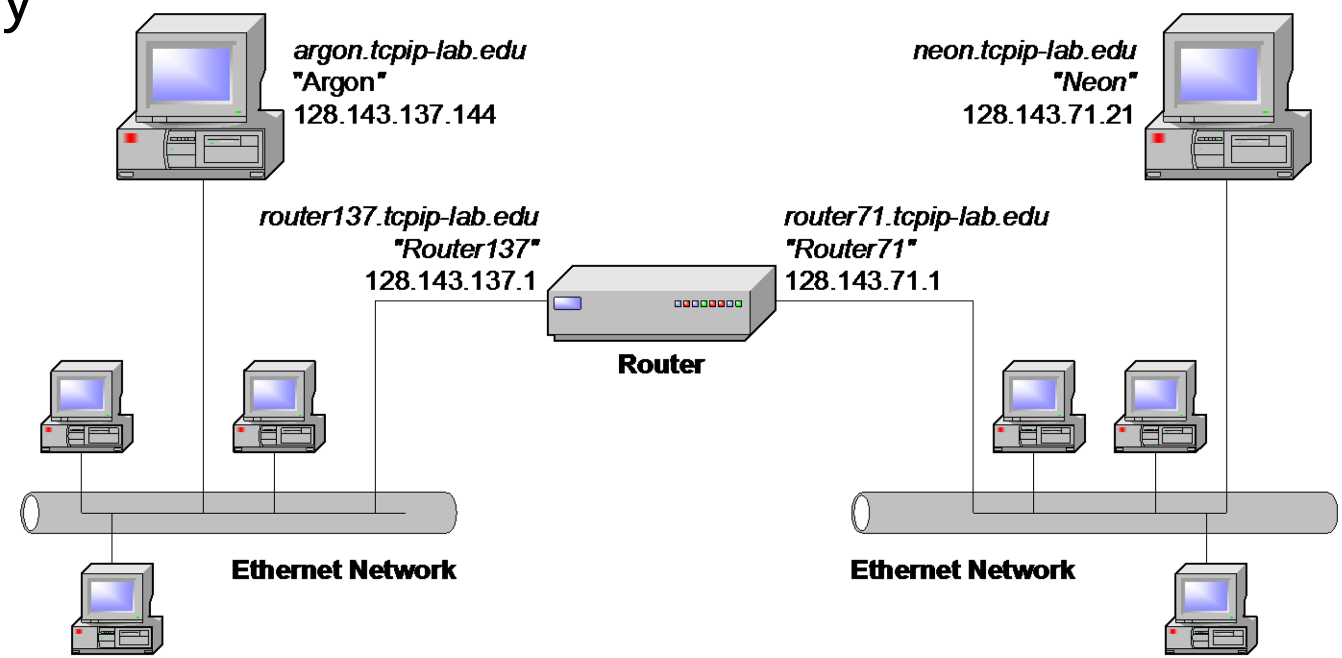
Sending an Ethernet frame

- The Ethernet device driver of *Argon* sends the Ethernet frame to the Ethernet network interface card (NIC)
- The NIC sends the frame onto the wire



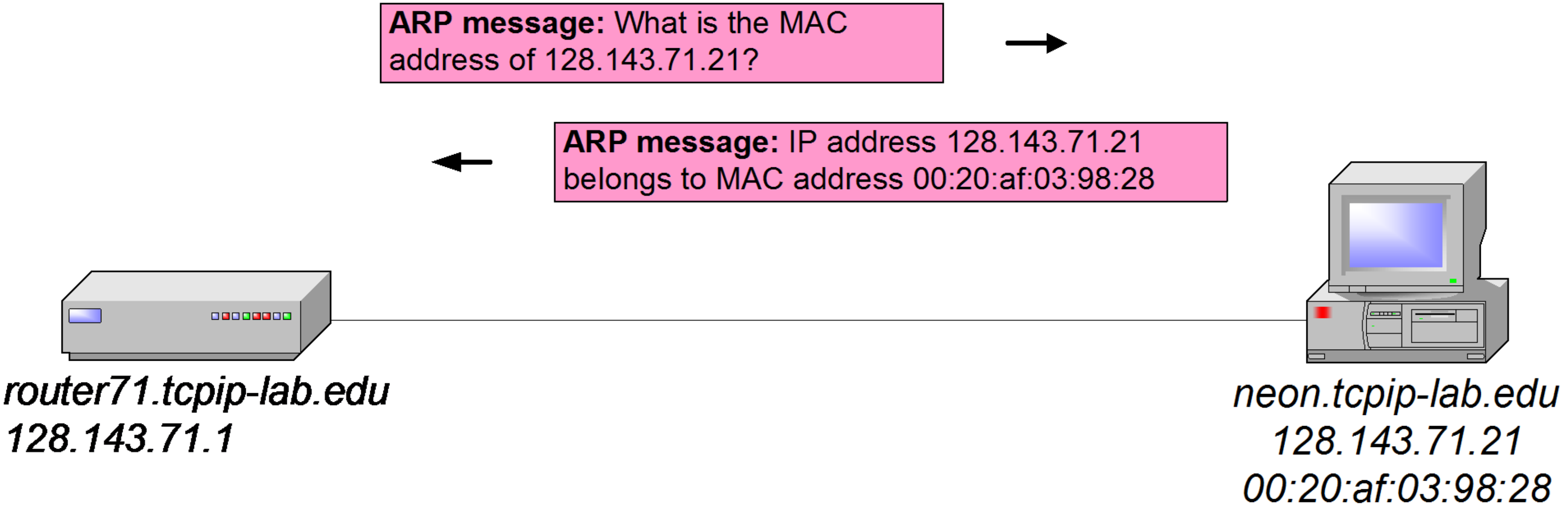
Forwarding the IP datagram

- The IP router receives the Ethernet frame at interface 128.143.137.1, recovers the IP datagram and determines that the IP datagram should be forwarded to the interface with name 128.143.71.1
- The IP router determines that it can deliver the IP datagram directly



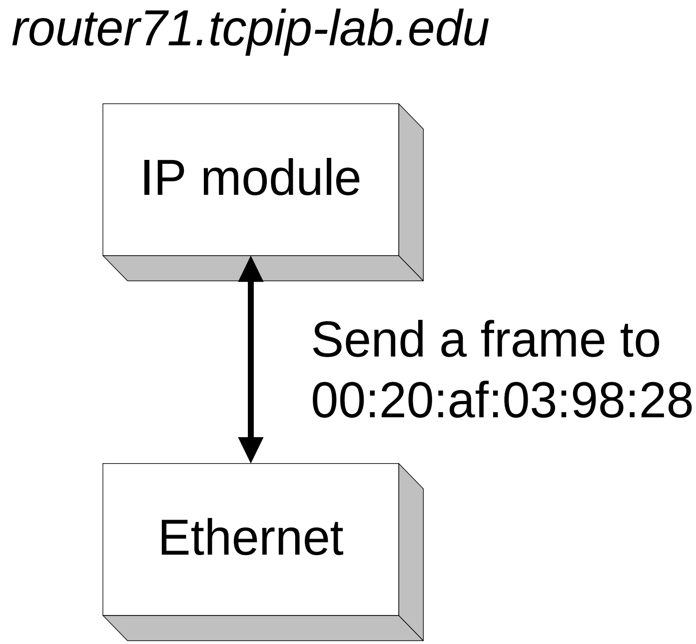
Another lookup of a MAC address

- The rouer needs to find the MAC address of *Neon*.
- Again, ARP is invoked, to translate the IP address of *Neon* (128.143.71.21) into the MAC address of neon (00:20:af:03:98:28).



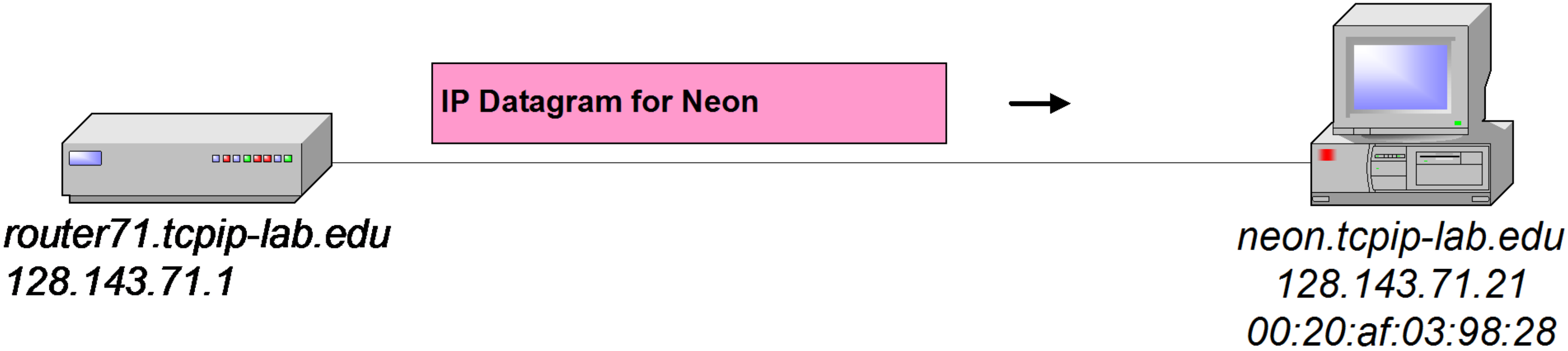
Invoking the device driver at the router

- The IP protocol at *Router71*, tells its Ethernet device driver to send an Ethernet frame to address *00:20:af:03:98:28*



Sending another Ethernet frame

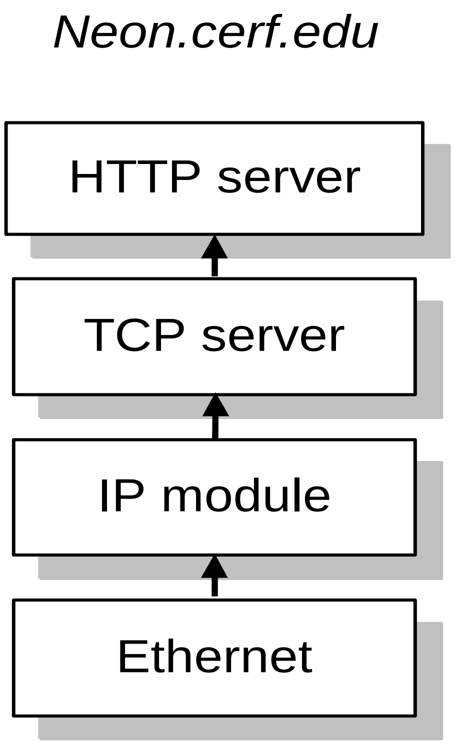
- The Ethernet device driver of *Router71* sends the Ethernet frame to the Ethernet NIC, which transmits the frame onto the wire.



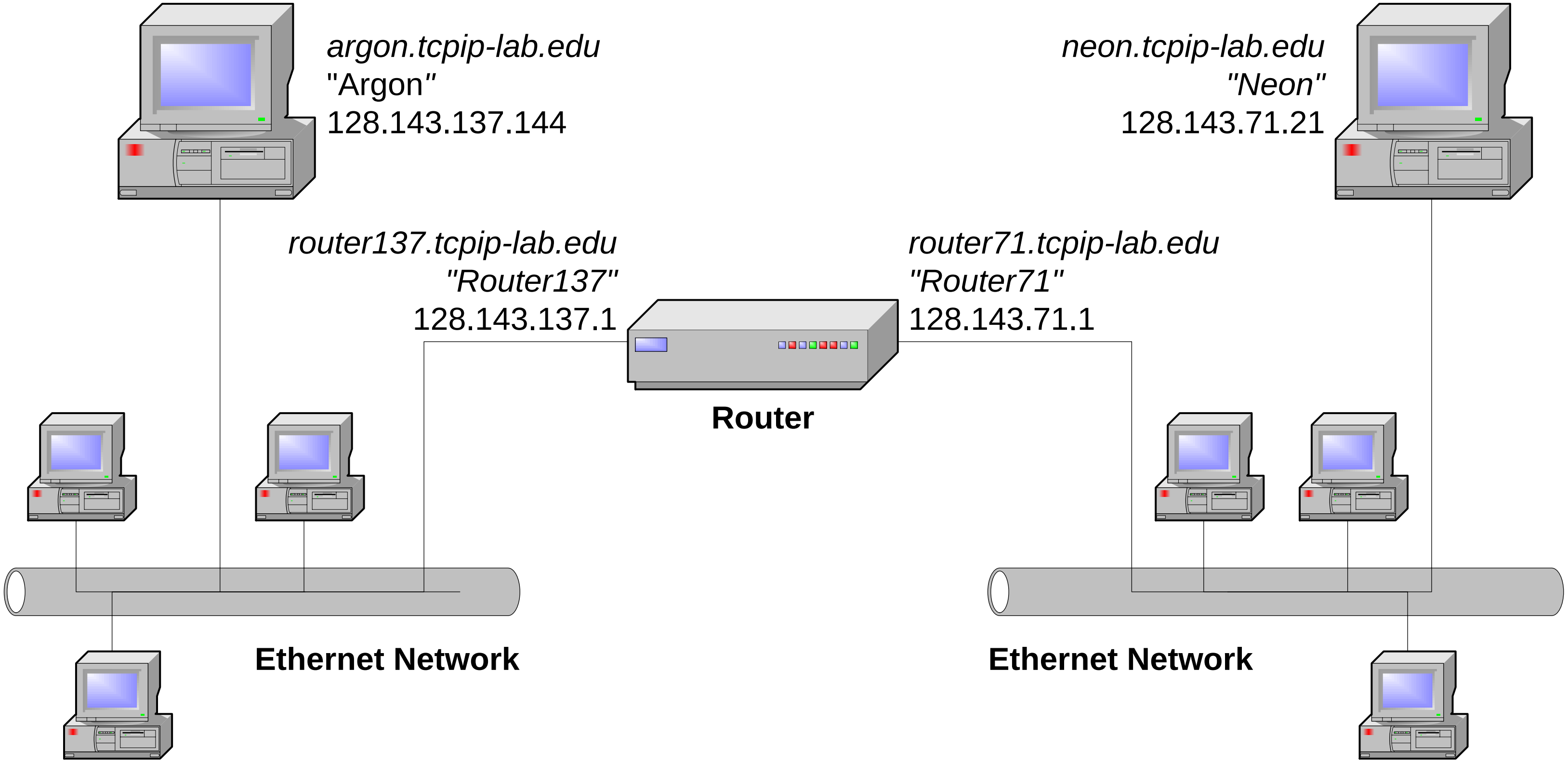
Data has arrived at Neon

- *Neon* receives the Ethernet frame
- The payload of the Ethernet frame is an IP datagram which is passed to the IP protocol.
- The payload of the IP datagram is a TCP segment, which is passed to the TCP server

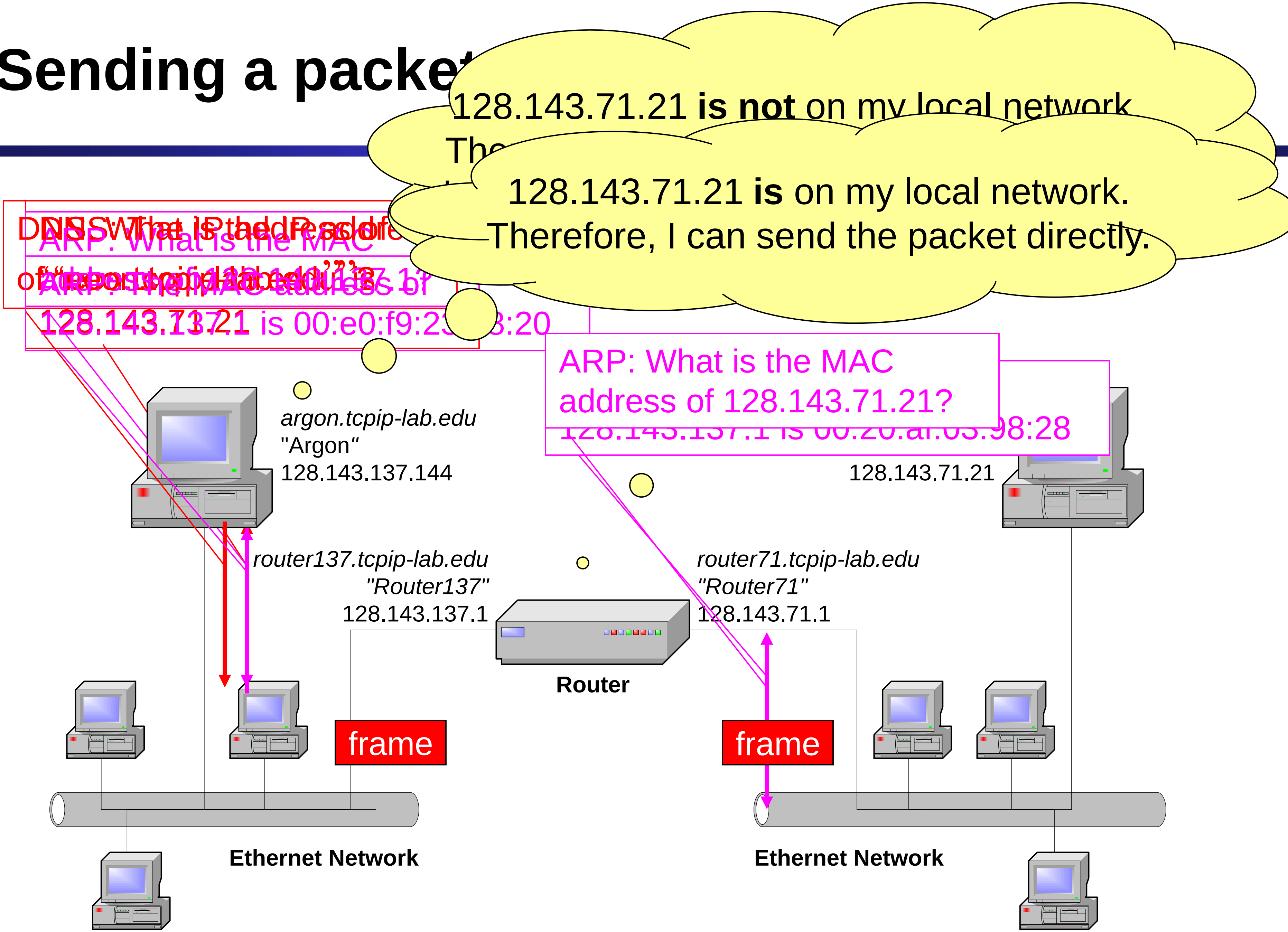
- **Note**: Since the TCP segment is a connection request (SYN), the TCP protocol does not pass data to the HTTP program for this packet. Instead, the TCP protocol at neon will respond with a SYN segment to *Argon*.



Sending a packet from Argon to Neon



Sending a packet



Wrapping-up the example

- So far, *Neon* has only obtained a single packet
- Much more work is required to establish an actual TCP connection and the transfer of the HTTP Request

- The example was simplified in several ways:
  - No transmission errors
  - The route between *Argon* and *Neon* is short (only one IP router)
  - *Argon* knew how to contact the DNS server (without routing or address resolution)
  - ….

Networking Concepts

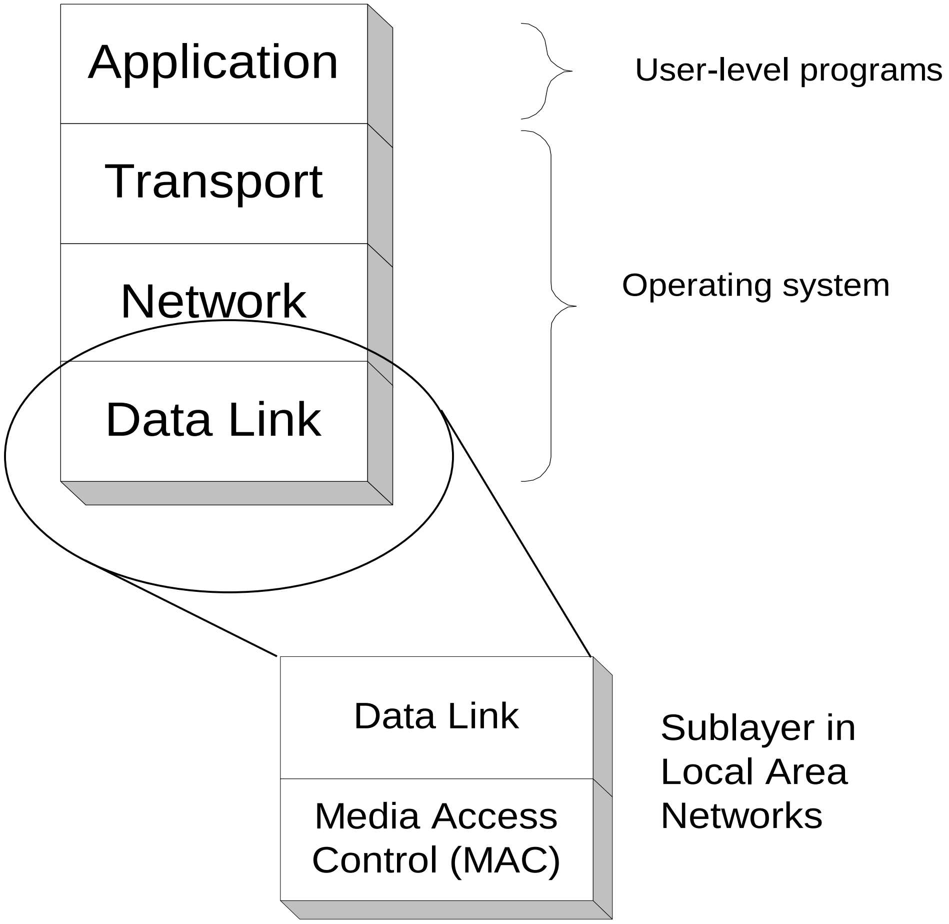
- Protocol Architecture
- Protocol Layers
- Encapsulation
- Network Abstractions

Communications Architecture

- The complexity of the communication task is reduced by using multiple protocol layers:
  - Each protocol is implemented independently
  - Each protocol is responsible for a specific subtask
  - Protocols are grouped in a hierarchy
- A structured set of protocols is called a communications architecture or protocol suite

TCP/IP Protocol Suite

- The TCP/IP protocol suite is the protocol architecture of the Internet
- The TCP/IP suite has four layers: Application, Transport, Network, and Data Link Layer
- End systems (hosts) implement all four layers. Gateways (Routers) only have the bottom two layers.

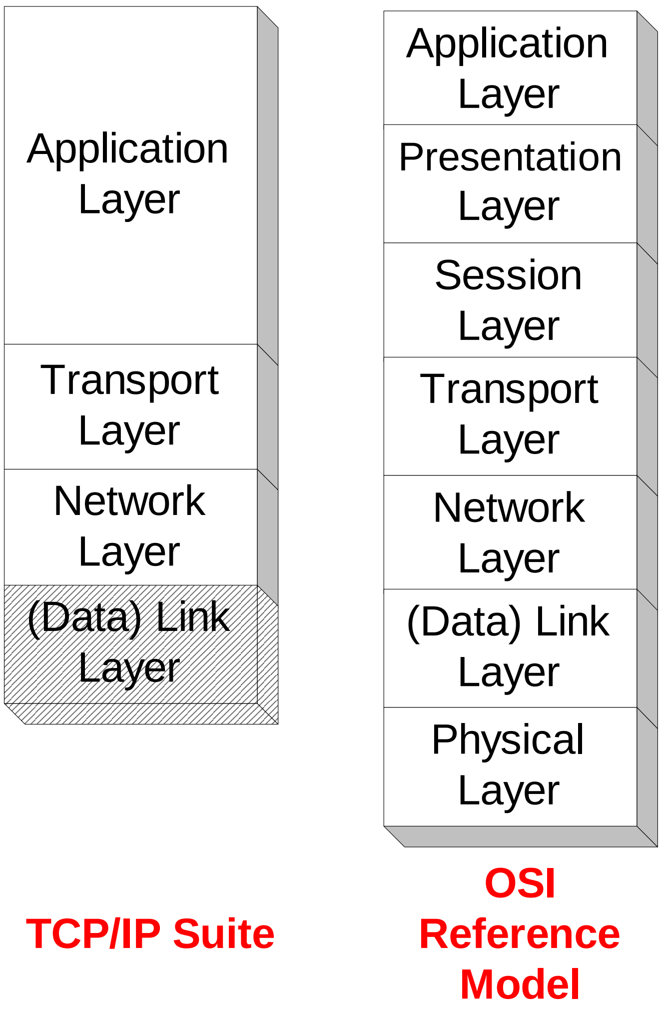


Functions of the Layers

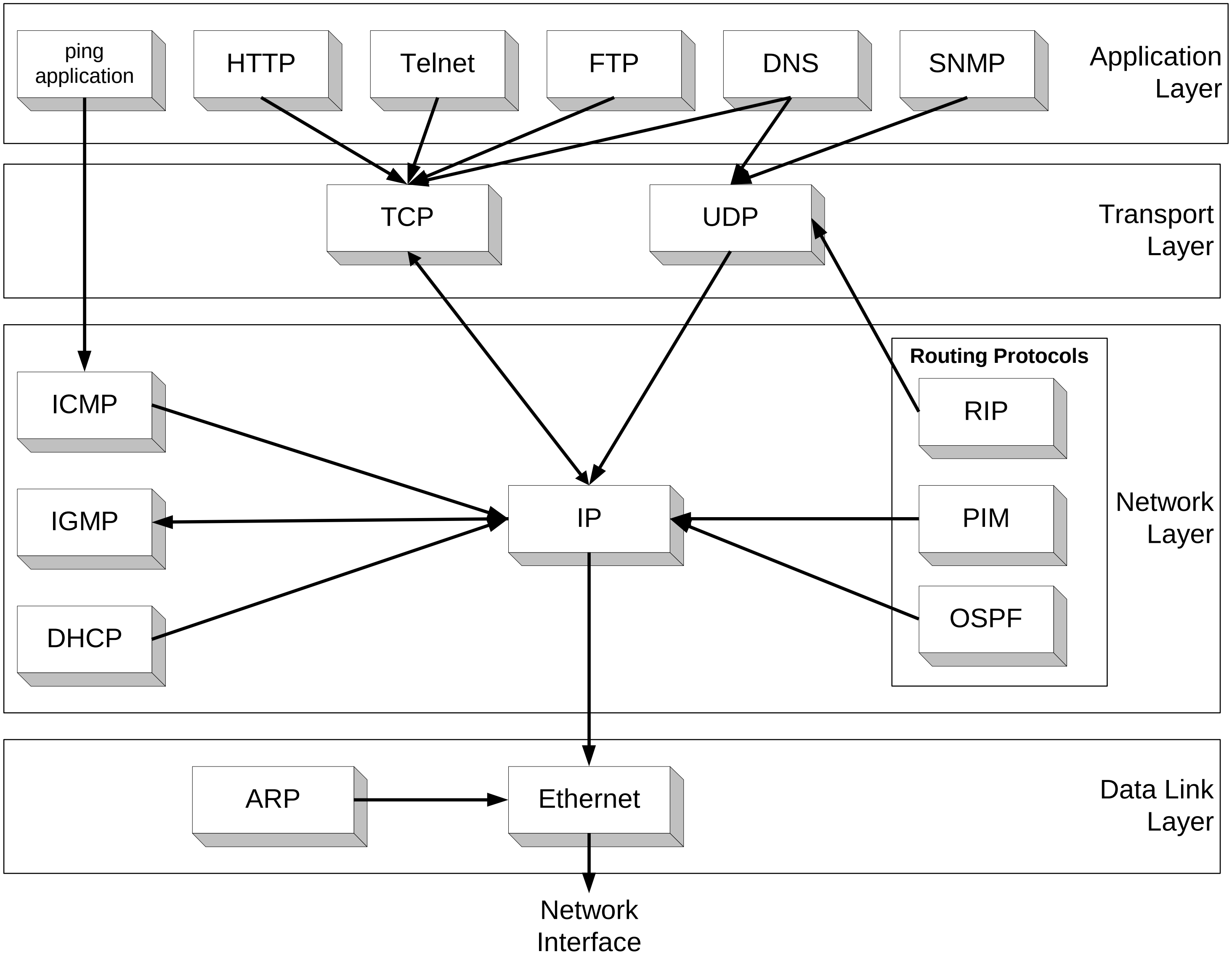
- Data Link Layer:
  - Service: Reliable transfer of frames over a link
    Media Access Control on a LAN
  - Functions: Framing, media access control, error checking
- Network Layer:
  - Service: Move packets from source host to destination host
  - Functions: Routing, addressing
- Transport Layer:
  - Service: Delivery of data between hosts
  - Functions: Connection establishment/termination, error control, flow control
- Application Layer:
  - Service: Application specific (delivery of email, retrieval of HTML documents, reliable transfer of file)
  - Functions: Application specific

TCP/IP Suite and OSI Reference Model

The TCP/IP protocol stack does not define the lower layers of a complete protocol stack

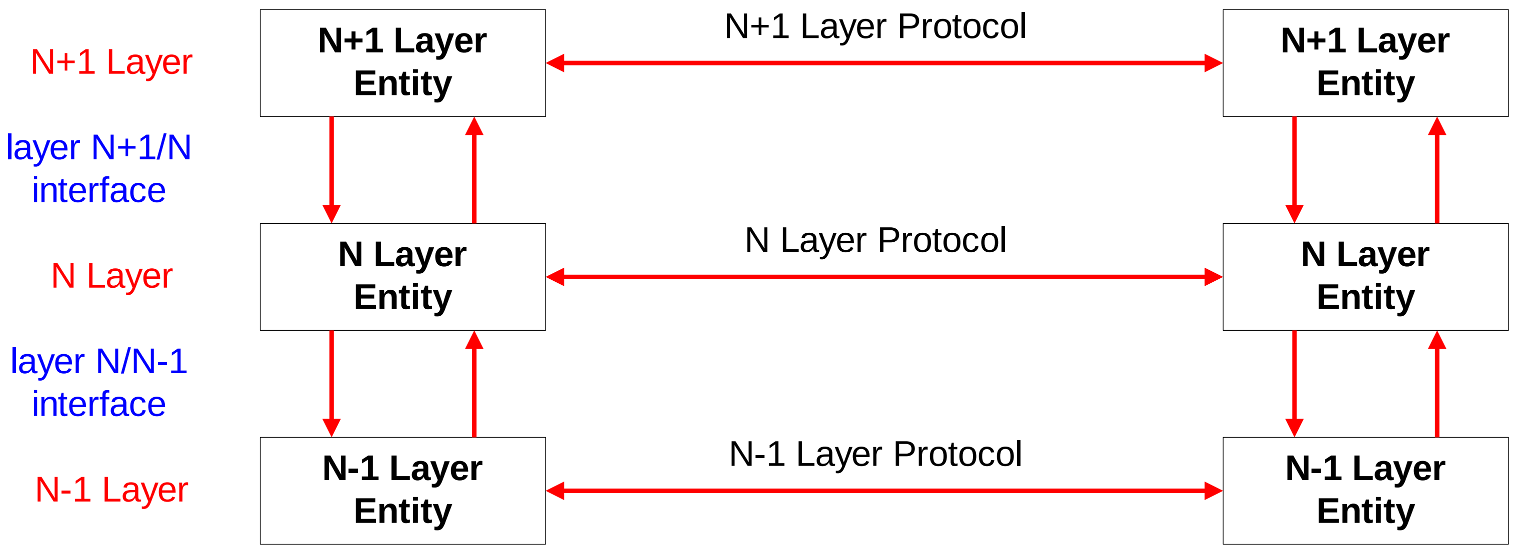


Assignment of Protocols to Layers



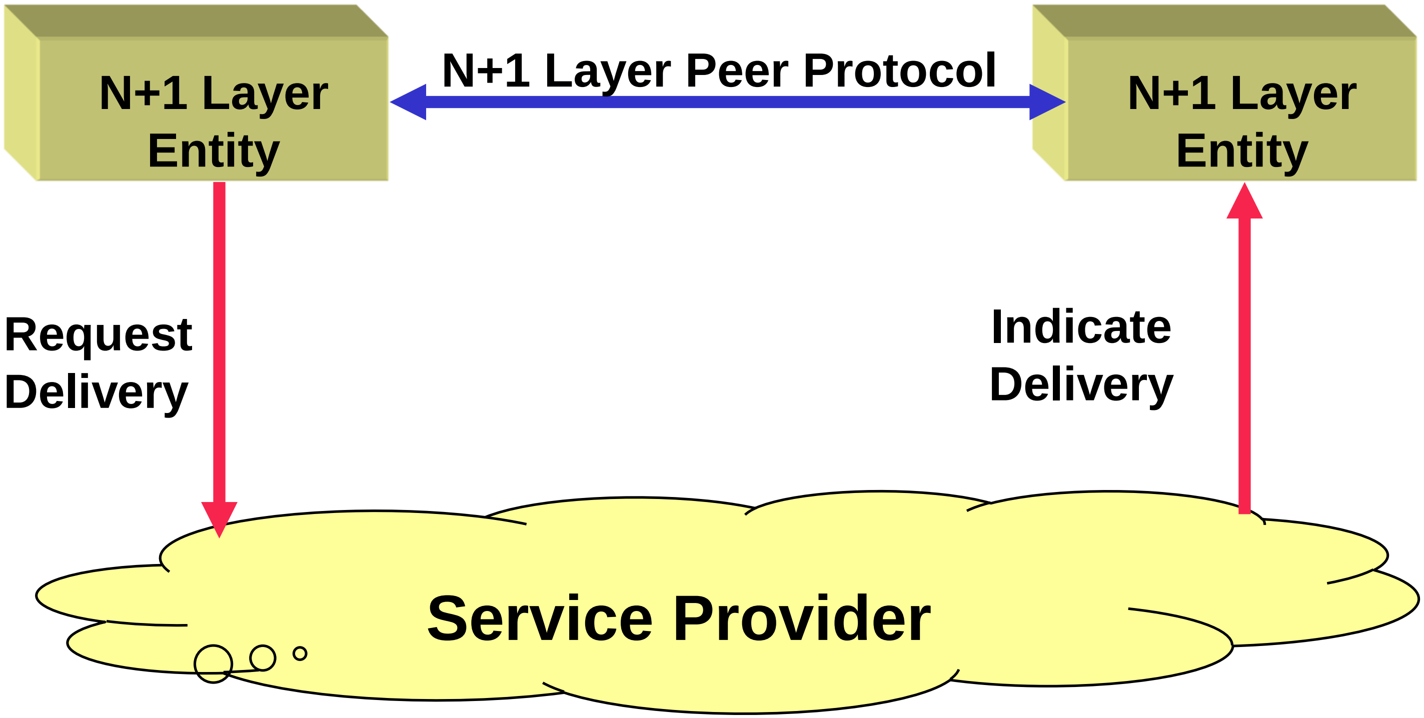
Layered Communications

- An entity of a particular layer can only communicate with:
  1. a peer layer entity using a common protocol (**Peer Protocol**)
  2. adjacent layers to provide services and to receive services



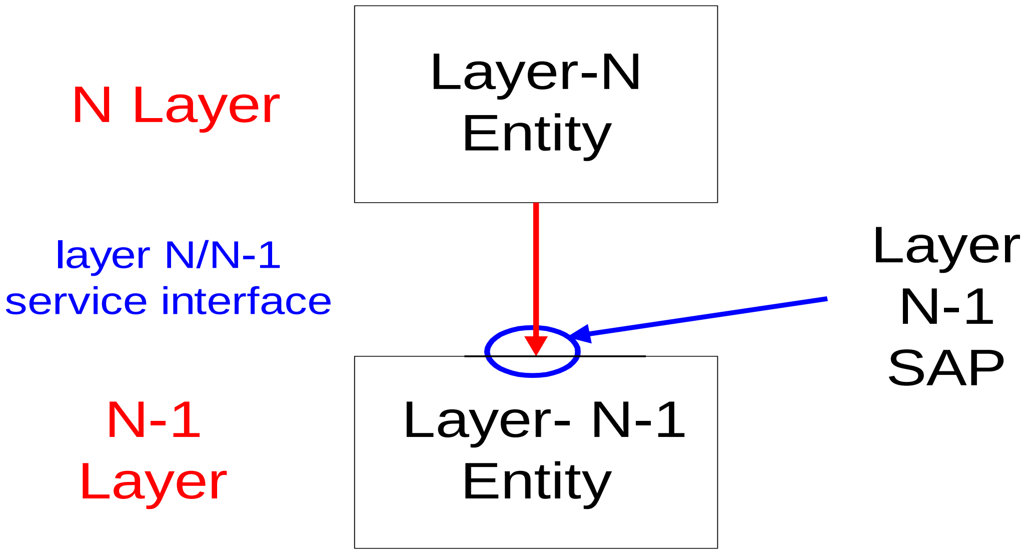
Layered Communications

A layer N+1 entity sees the lower layers only as a service provider



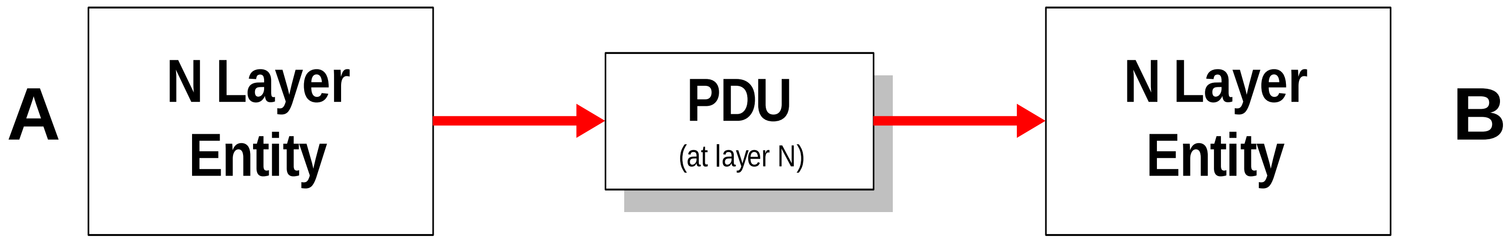
Service Access Points

- A service user accesses services of the service provider at Service **Access Points (SAPs)**
- A SAP has an address that uniquely identifies where the service can be accessed



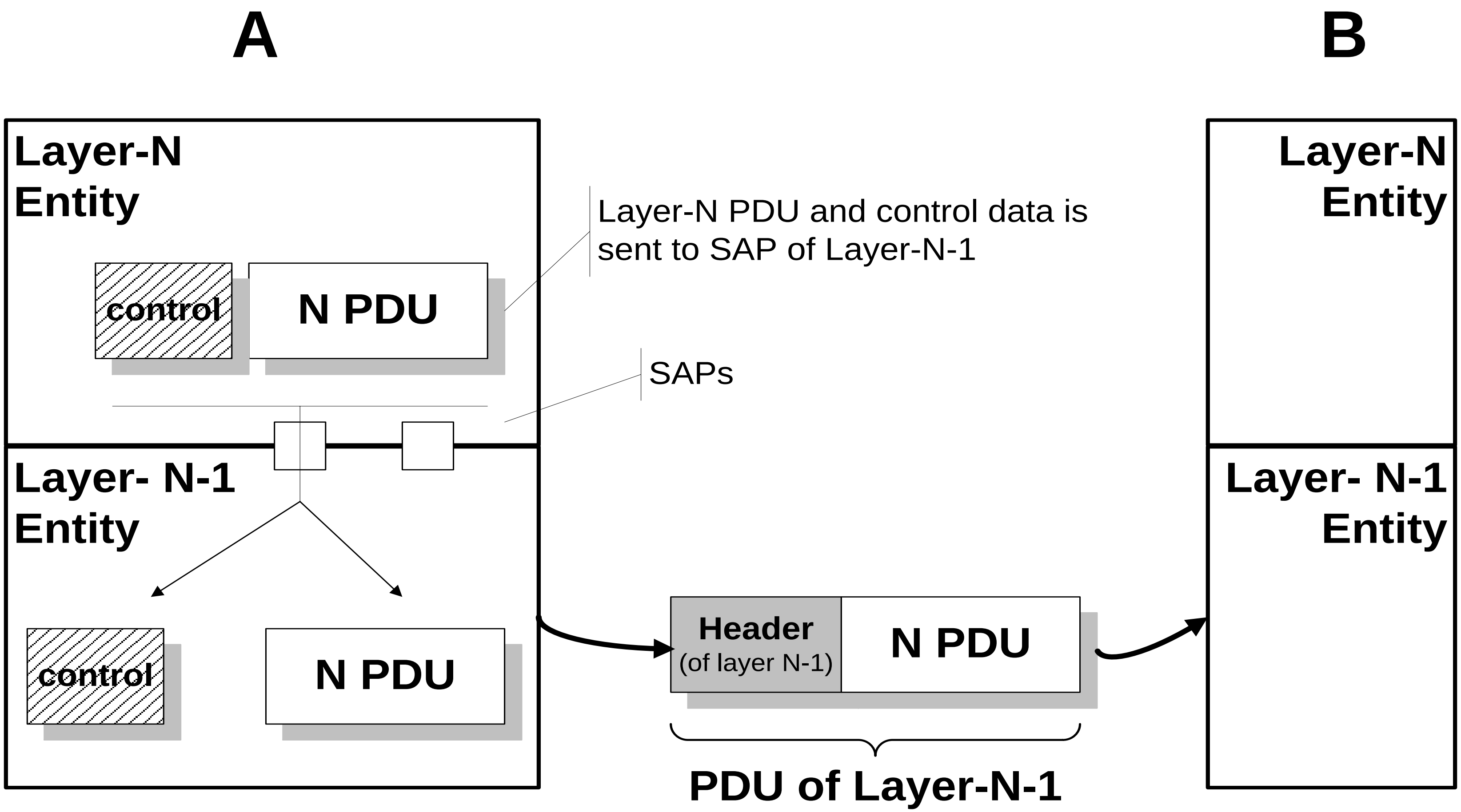
Exchange of Data

- The unit of data send between peer entities is called a **Protocol Data Unit (PDU)**
- For now, let us think of a PDU as a single packet

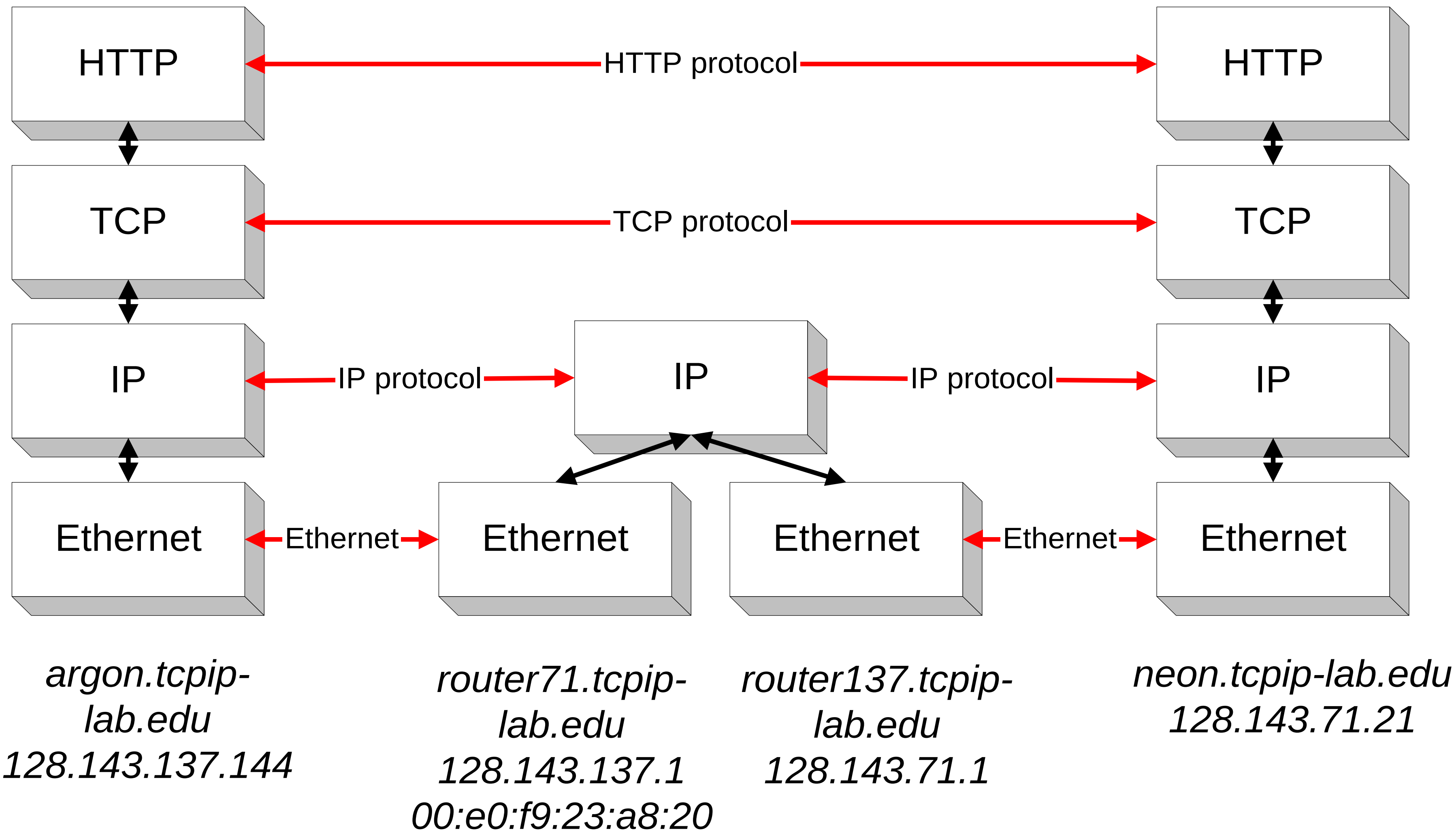


- **Scenario:** Layer-N at A sends a layer-N PDU to layer-N at B
- What actually happens:
  - A’s layer-N passes the PDU to one the SAPs at layer-N-1
  - Layer-N-1 entity at A constructs its own (layer-N-1) PDU which it sends to the layer-N-1 entity at B
  - PDU at layer-N-1 = layer-N-1 Header + layer –N PDU

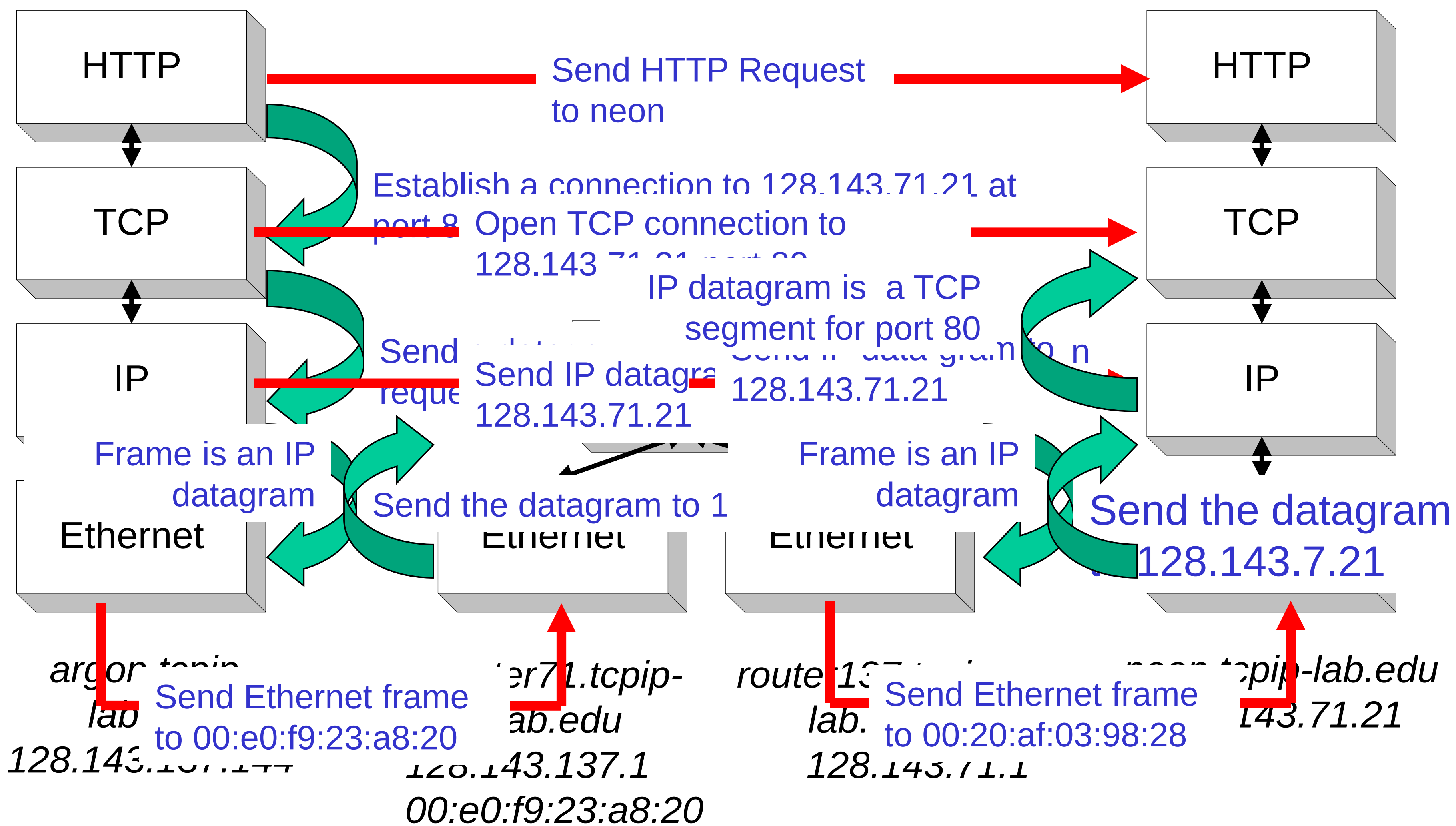
Exchange of Data



Layers in the Example



Layers in the Example



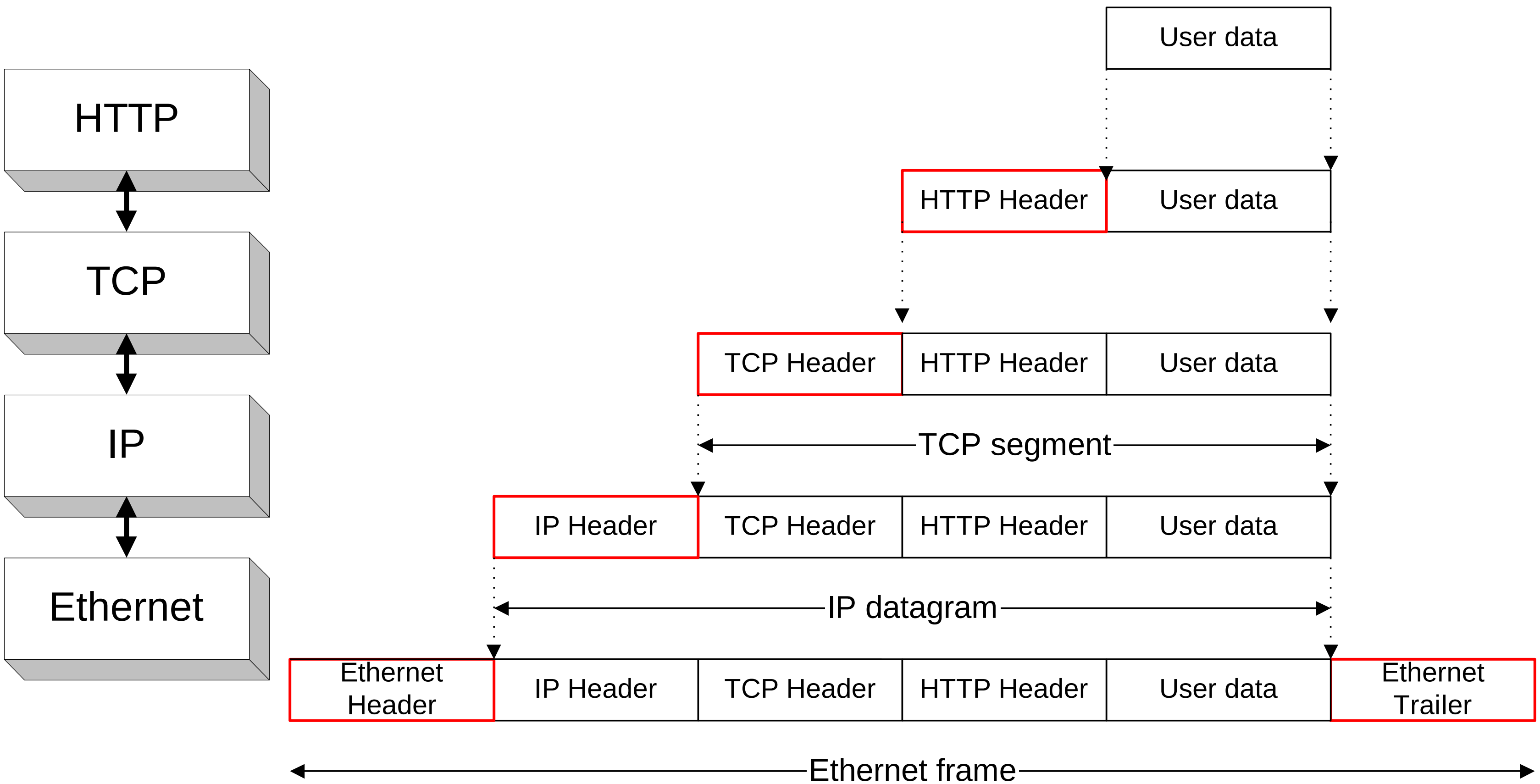
Layers and Services

- Service provided by TCP to HTTP:
  - reliable transmission of data over a logical connection
- Service provided by IP to TCP:
  - unreliable transmission of IP datagrams across an IP network
- Service provided by Ethernet to IP:
  - transmission of a frame across an Ethernet segment

- Other services:
  - DNS: translation between domain names and IP addresses
  - ARP: Translation between IP addresses and MAC addresses

Encapsulation and Demultiplexing

- As data is moving down the protocol stack, each protocol is adding layer-specific control information



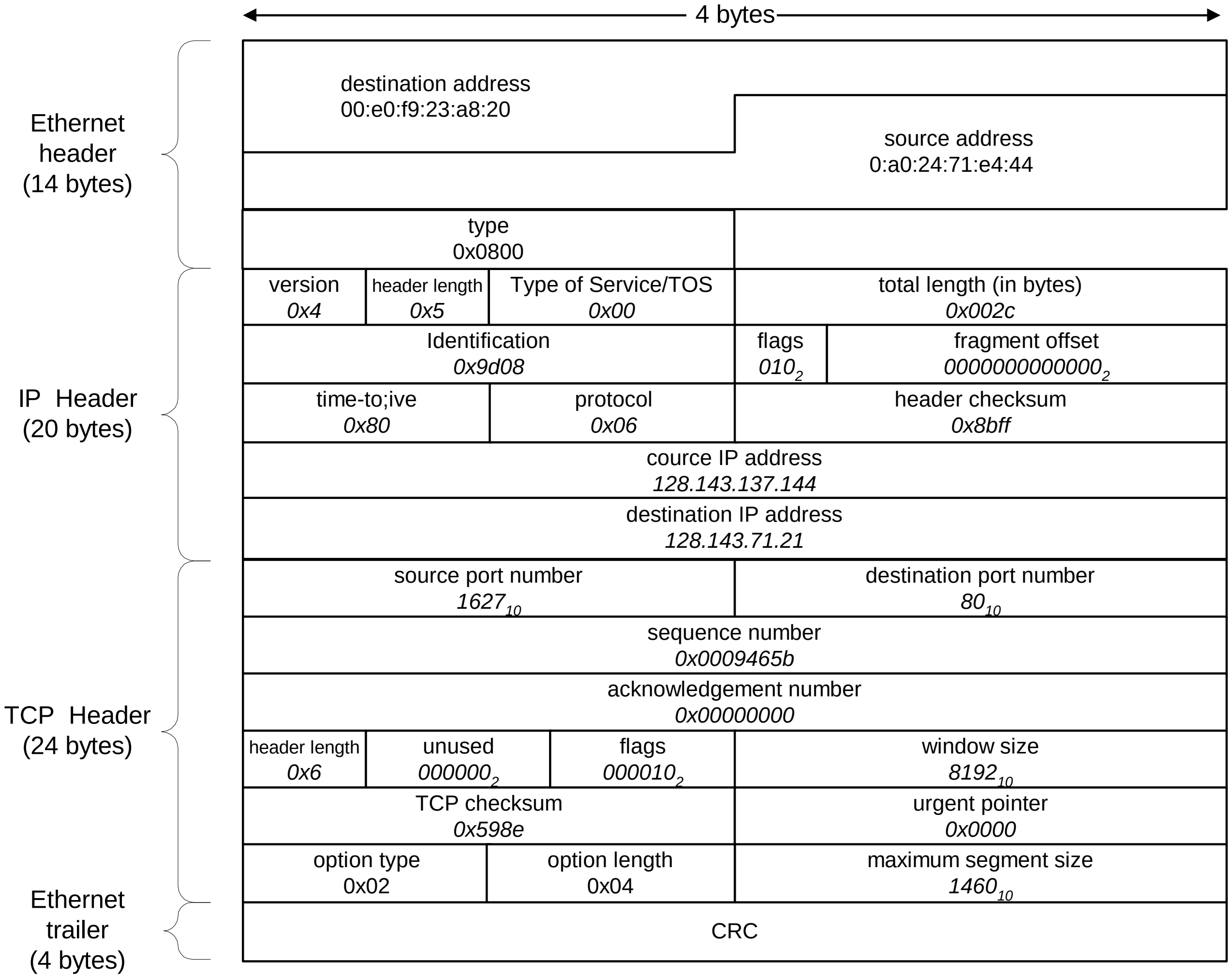
Encapsulation and Demultiplexing in our Example

- Let us look in detail at the Ethernet frame between Argon and the Router, which contains the TCP connection request to Neon.

- This is the frame in hexadecimal notation.

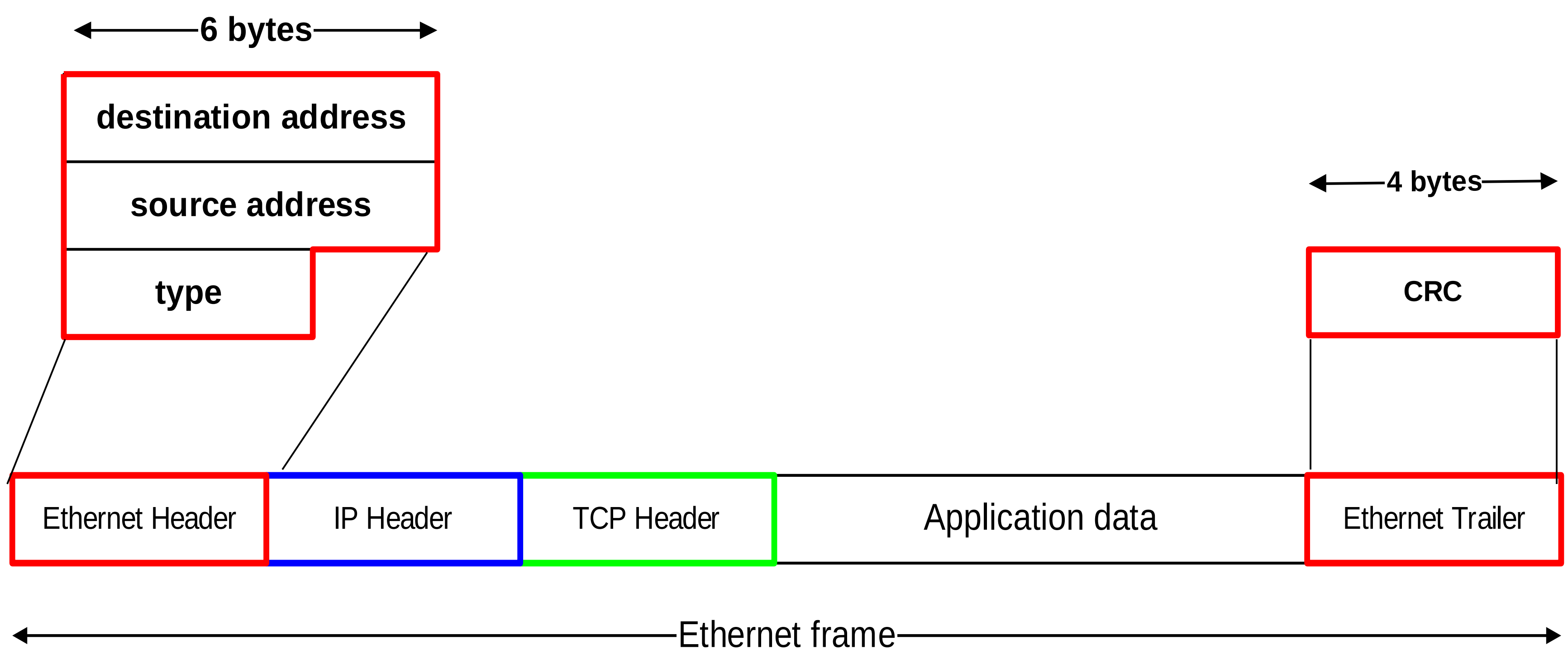
```
00e0 f923 a820 00a0 2471 e444 0800 4500 002c
9d08 4000 8006 8bff 808f 8990 808f 4715 065b
0050 0009 465b 0000 0000 6002 2000 598e 0000
0204 05b4
```

Parsing the information in the frame



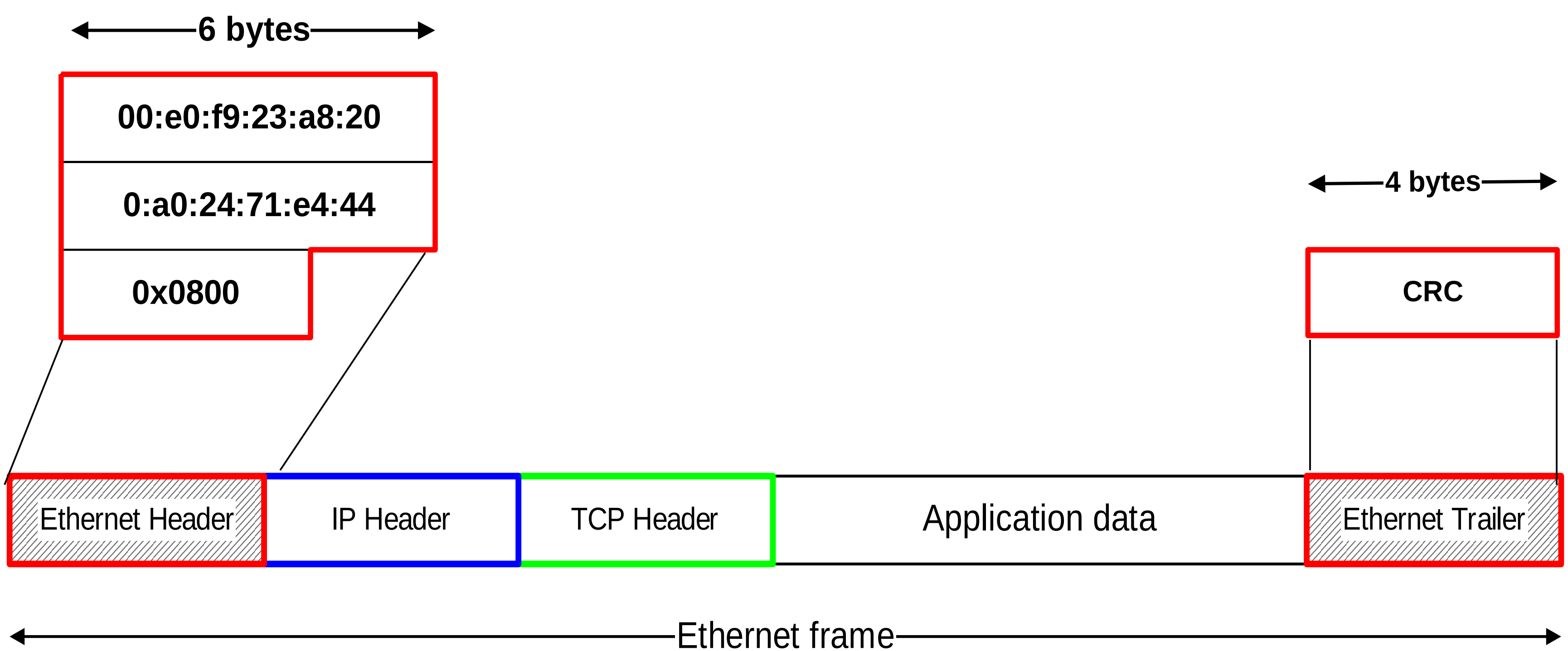
Encapsulation and Demultiplexing

```
00e0 f923 a820 00a0 2471 e444 0800 4500 002c 9d08
4000 8006 8bff 808f 8990 808f 4715 065b 0050 0009
465b 0000 0000 6002 2000 598e 0000 0204 05b4
```

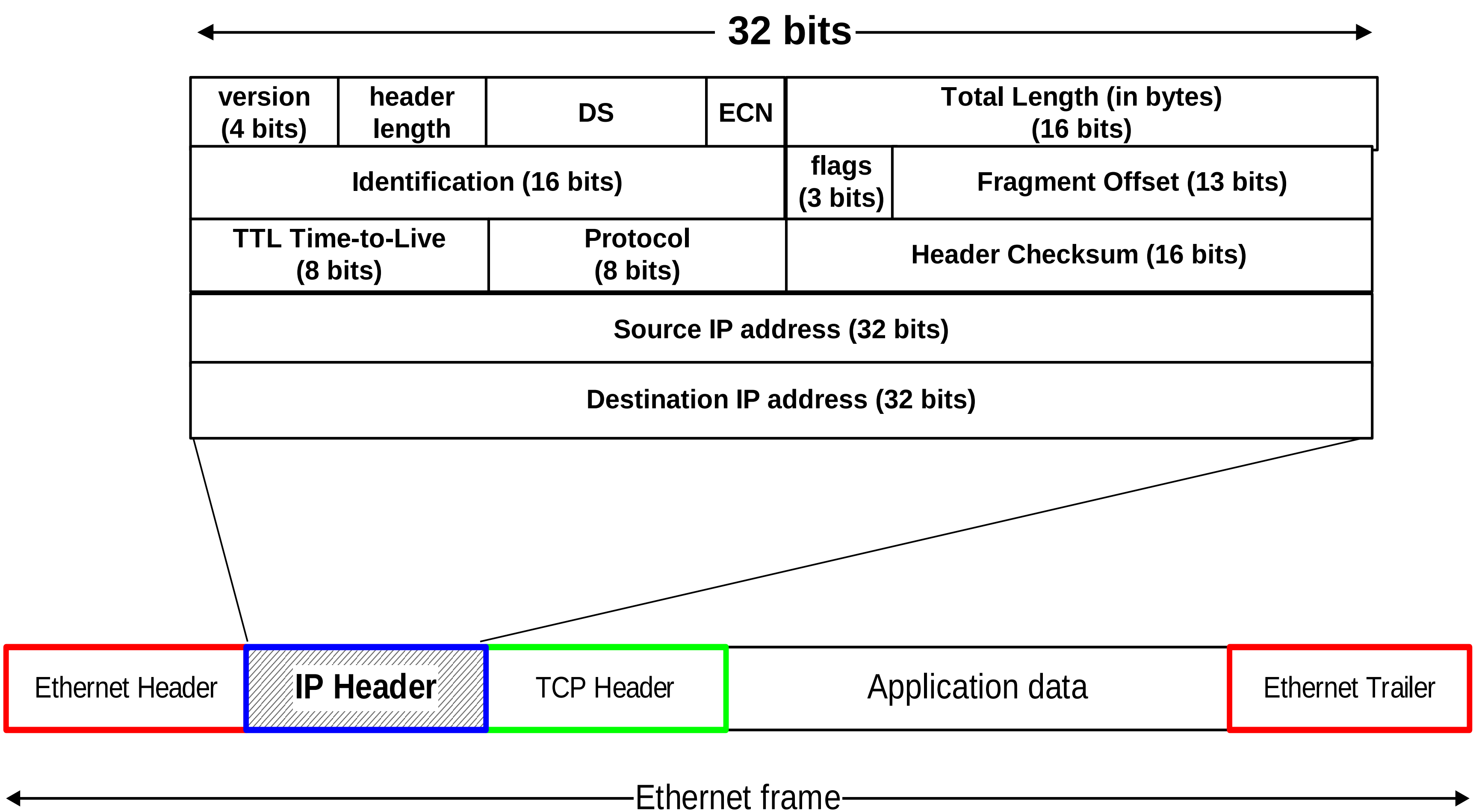


Encapsulation and Demultiplexing: Ethernet Header

```
00e0 f923 a820 00a0 2471 e444 0800 4500 002c 9d08
4000 8006 8bff 808f 8990 808f 4715 065b 0050 0009
465b 0000 0000 6002 2000 598e 0000 0204 05b4
```



Encapsulation and Demultiplexing: IP Header

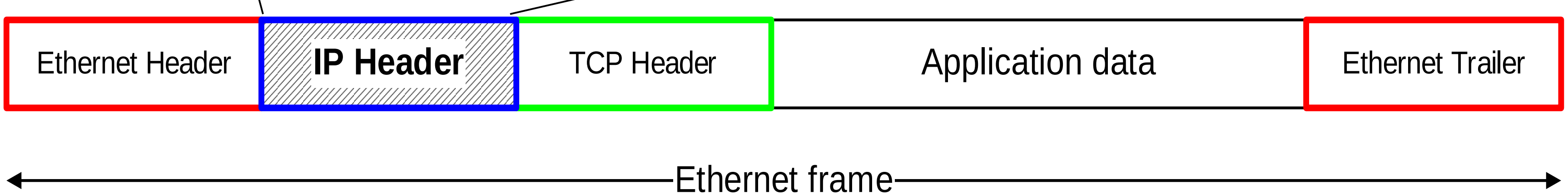


Encapsulation and Demultiplexing: IP Header

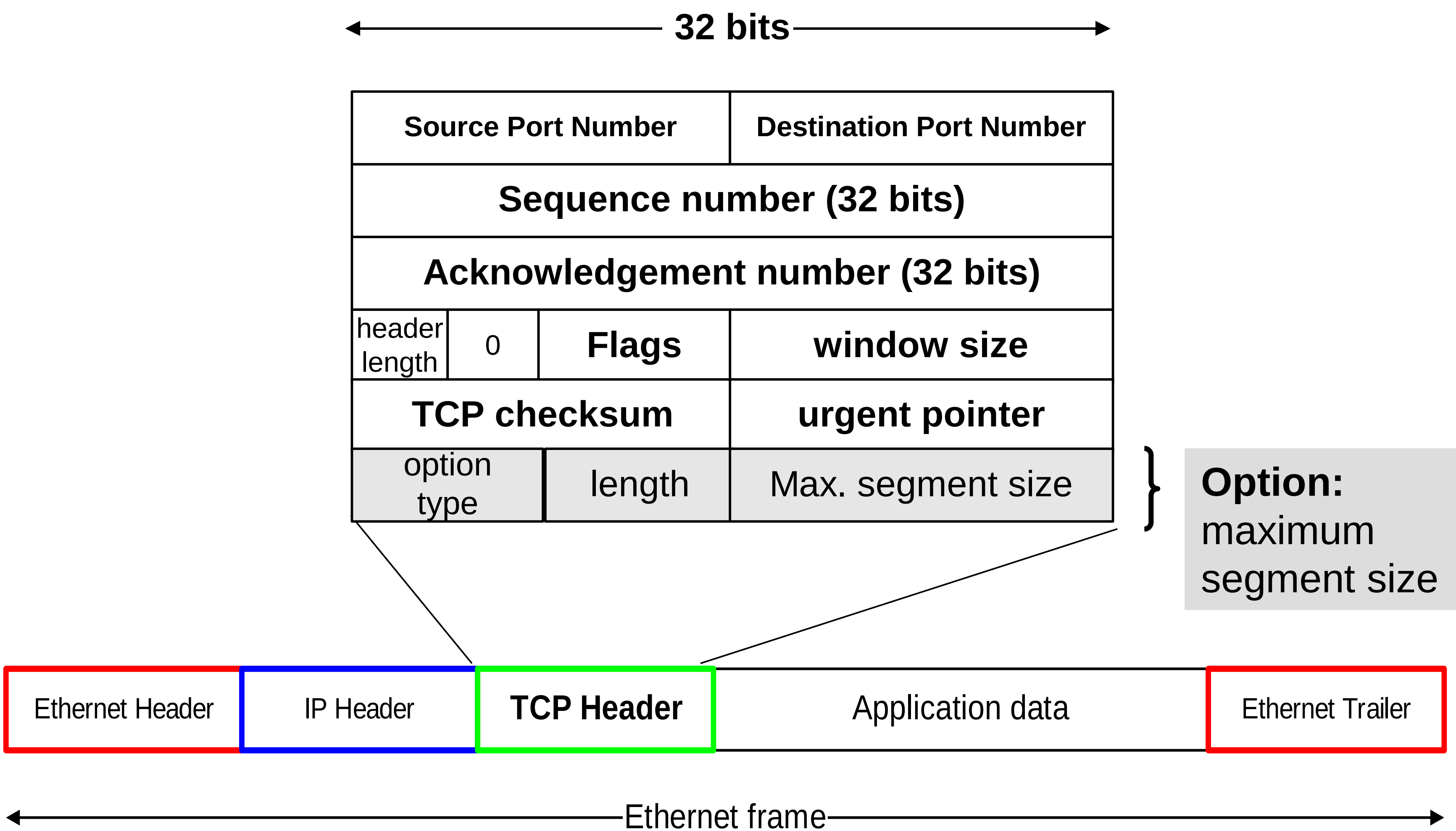
32 bits

<table>
<tr><td>0x4</td><td>0x5</td><td>0x0</td><td>0x0</td><td colspan="2">44_{10}</td></tr>
<tr><td colspan="4">9d08</td><td>010_{2}</td><td>0000000000000_{2}</td></tr>
<tr><td colspan="2">128_{10}</td><td colspan="2">0x06</td><td colspan="2">8bff</td></tr>
<tr><td colspan="6">128.143.137.144</td></tr>
<tr><td colspan="6">128.143.71.21</td></tr>
</table>

```
00e0 f923 a820 00a0 2471 e444 0800 4500 002c 9d08
4000 8006 8bff 808f 8990 808f 4715 065b 0050 0009
465b 0000 0000 6002 2000 598e 0000 0204 05b4
```

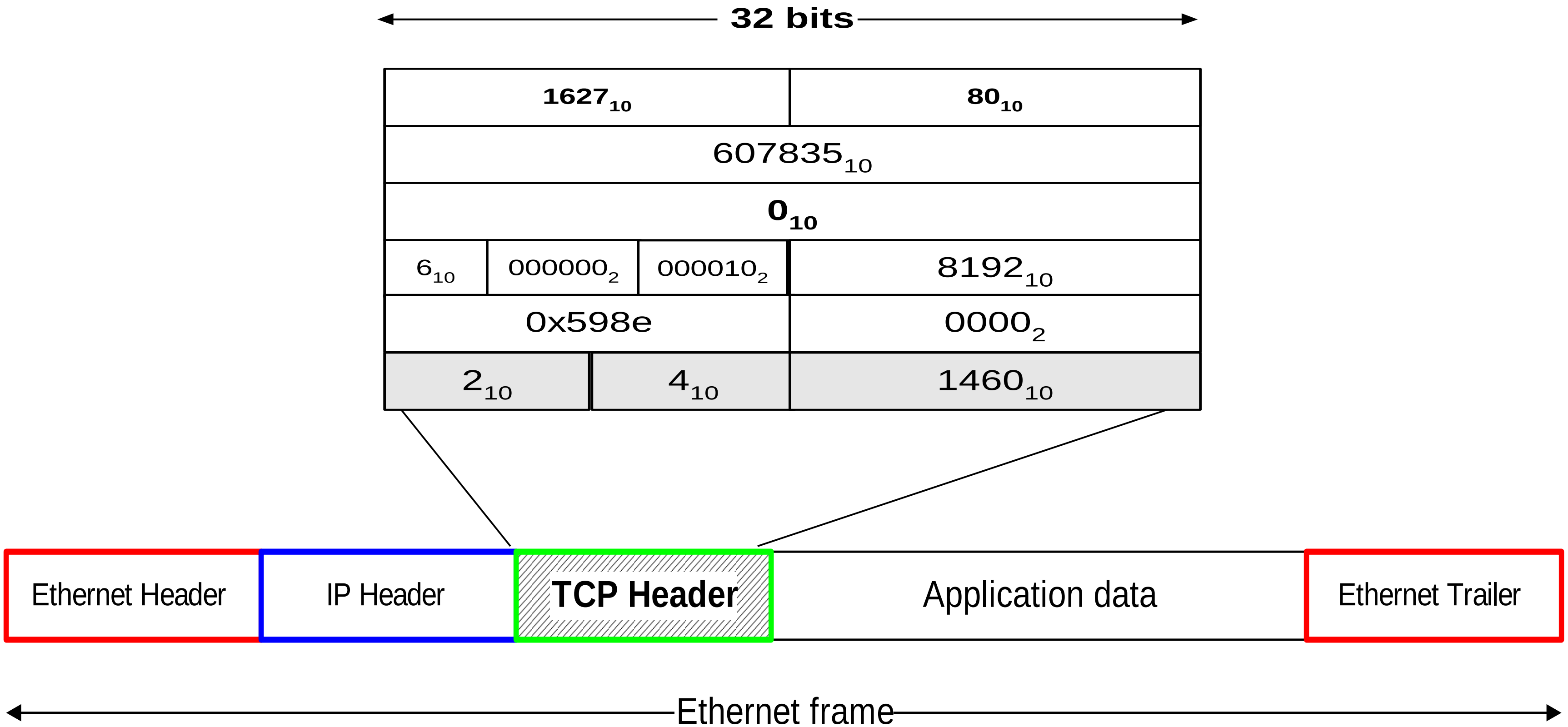


Encapsulation and Demultiplexing: TCP Header



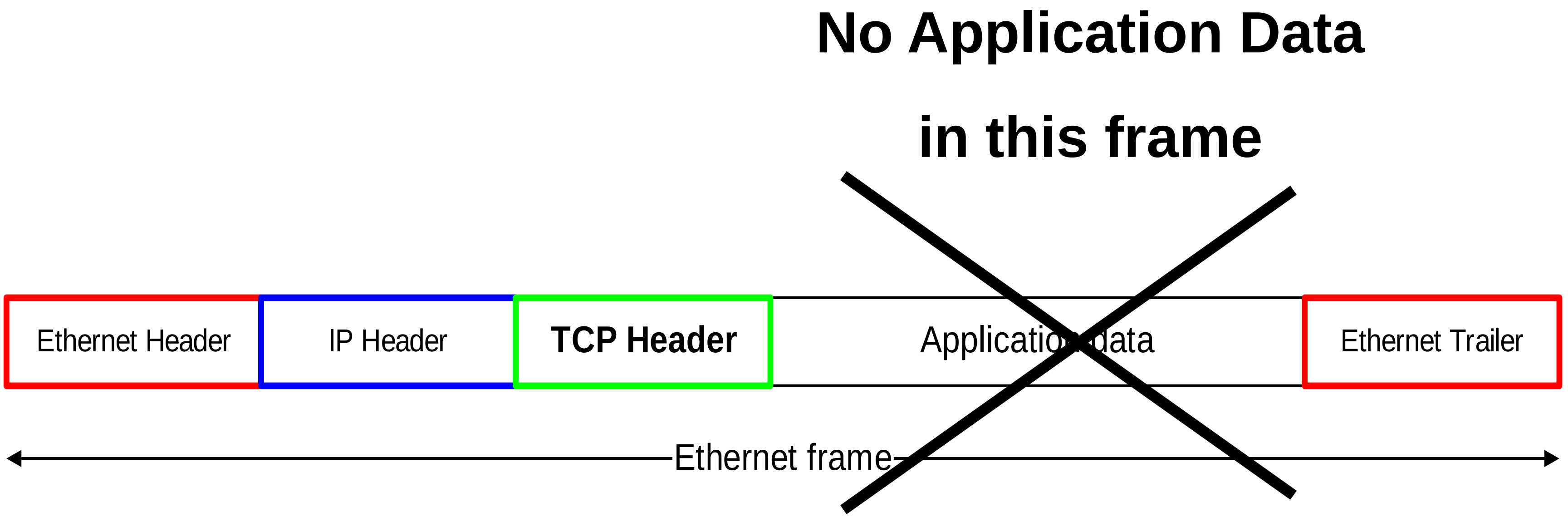
Encapsulation and Demultiplexing: TCP Header

```
00e0 f923 a820 00a0 2471 e444 0800 4500 002c 9d08
4000 8006 8bff 808f 8990 808f 4715 065b 0050 0009
465b 0000 0000 6002 2000 598e 0000 0204 05b4
```



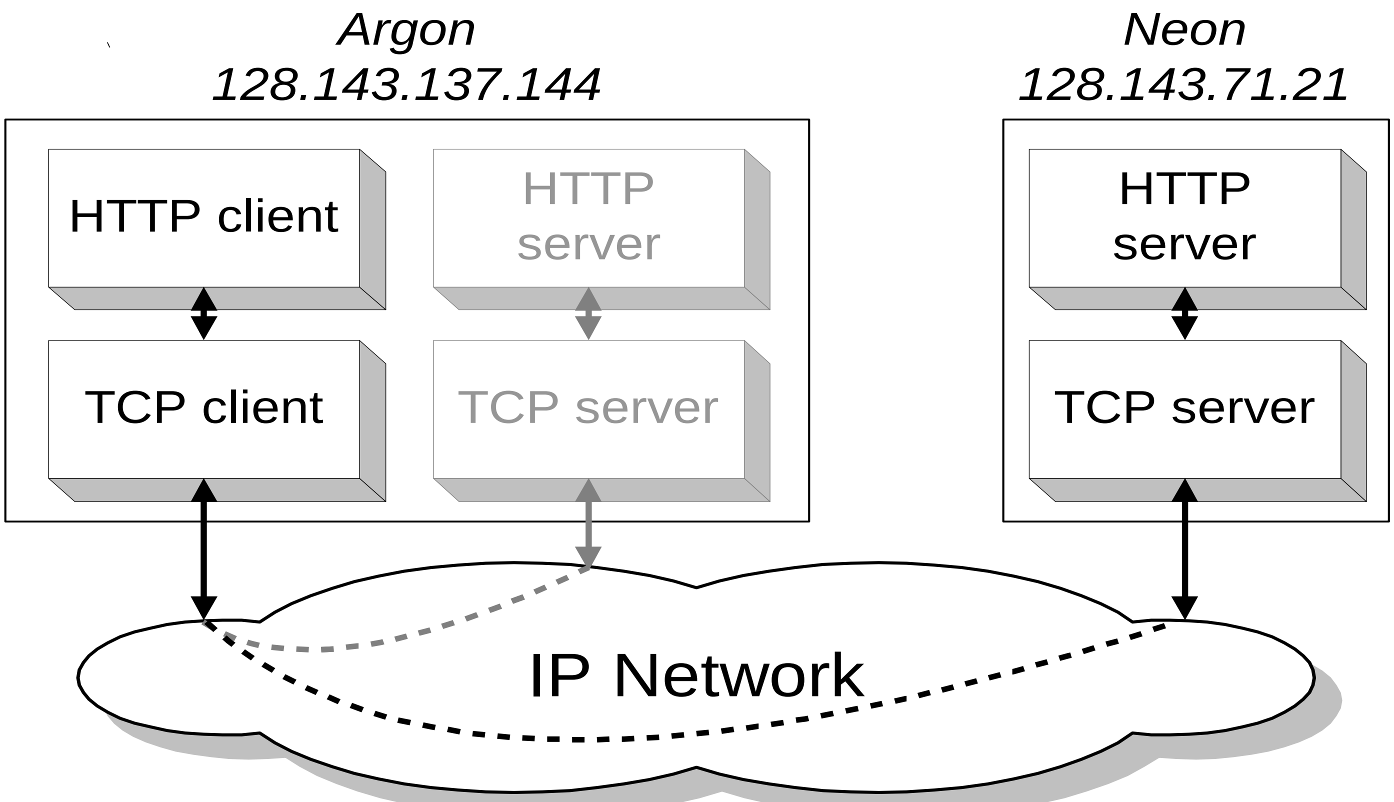
Encapsulation and Demultiplexing: Application data

```
00e0 f923 a820 00a0 2471 e444 0800 4500 002c 9d08
4000 8006 8bff 808f 8990 808f 4715 065b 0050 0009
465b 0000 0000 6002 2000 598e 0000 0204 05b4
```

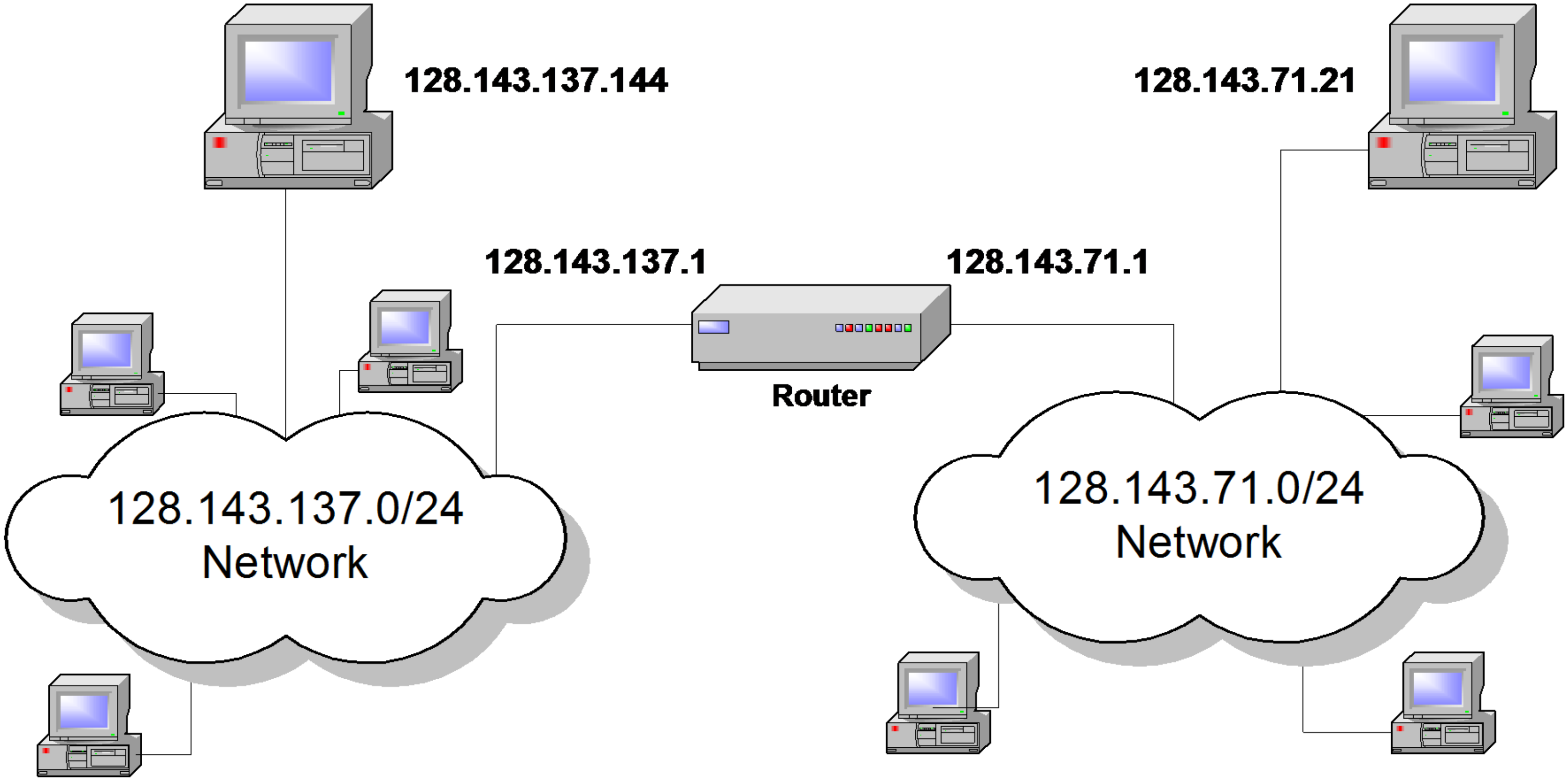


Different Views of Networking

- Different Layers of the protocol stack have a different view of the network. This is HTTP's and TCP's view of the network.



Network View of IP Protocol



Network View of Ethernet

- Ethernet's view of the network