Transport,Middleware & Applications

- UDP (Header, Services)
- Demultiplexing (Protocol, Port)
- Checksum
- TCP (Header, Services)
- TCP Connections
- Applications

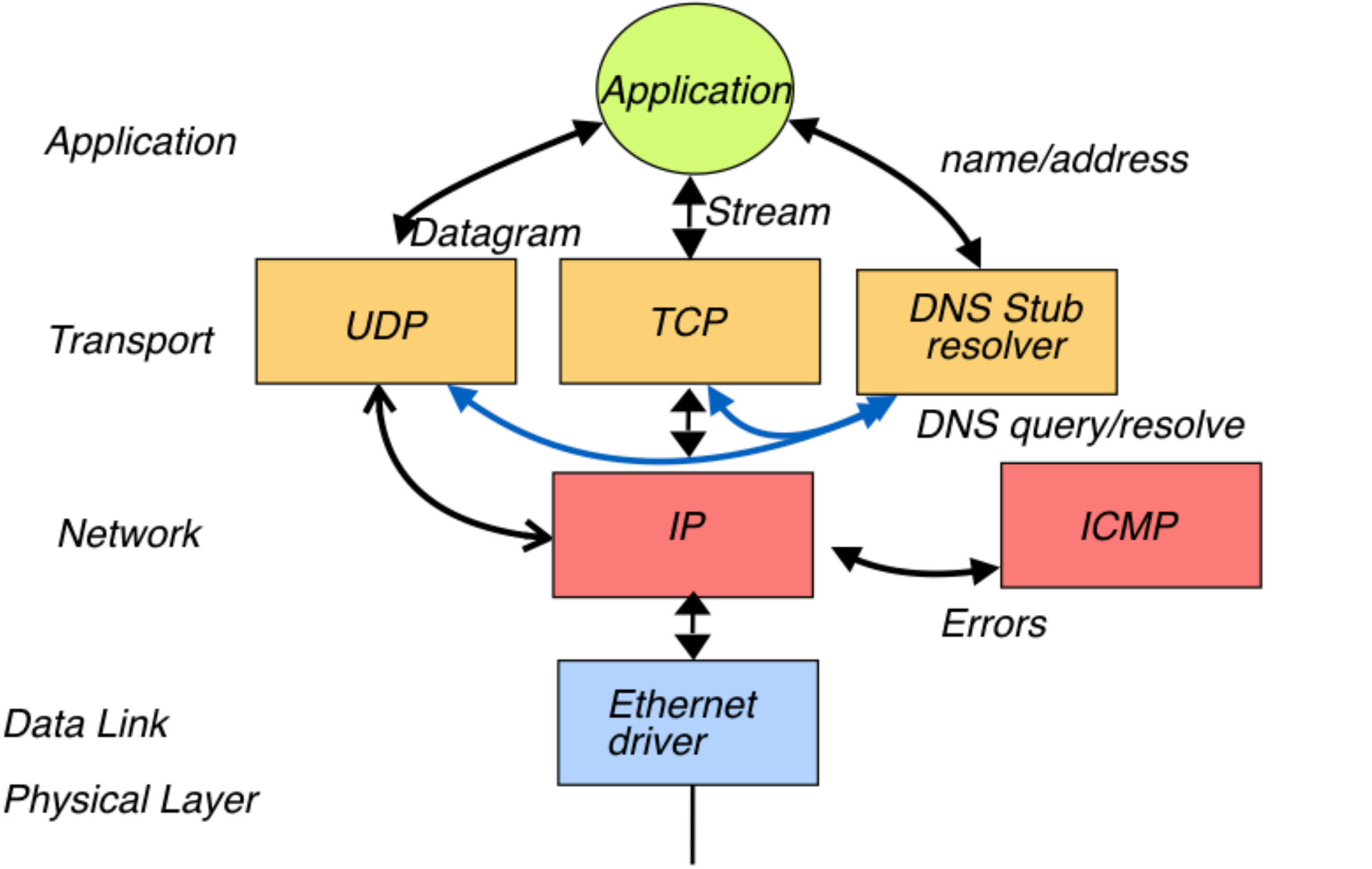
G Fairhurst, http://www.erg.abdn.ac.uk

UDP User Datagram Protocol

- Best Effort Service
- Multiplexing (service access points)
- Integrity Check

Protocol Layers

G Fairhurst, http://www.erg.abdn.ac.uk

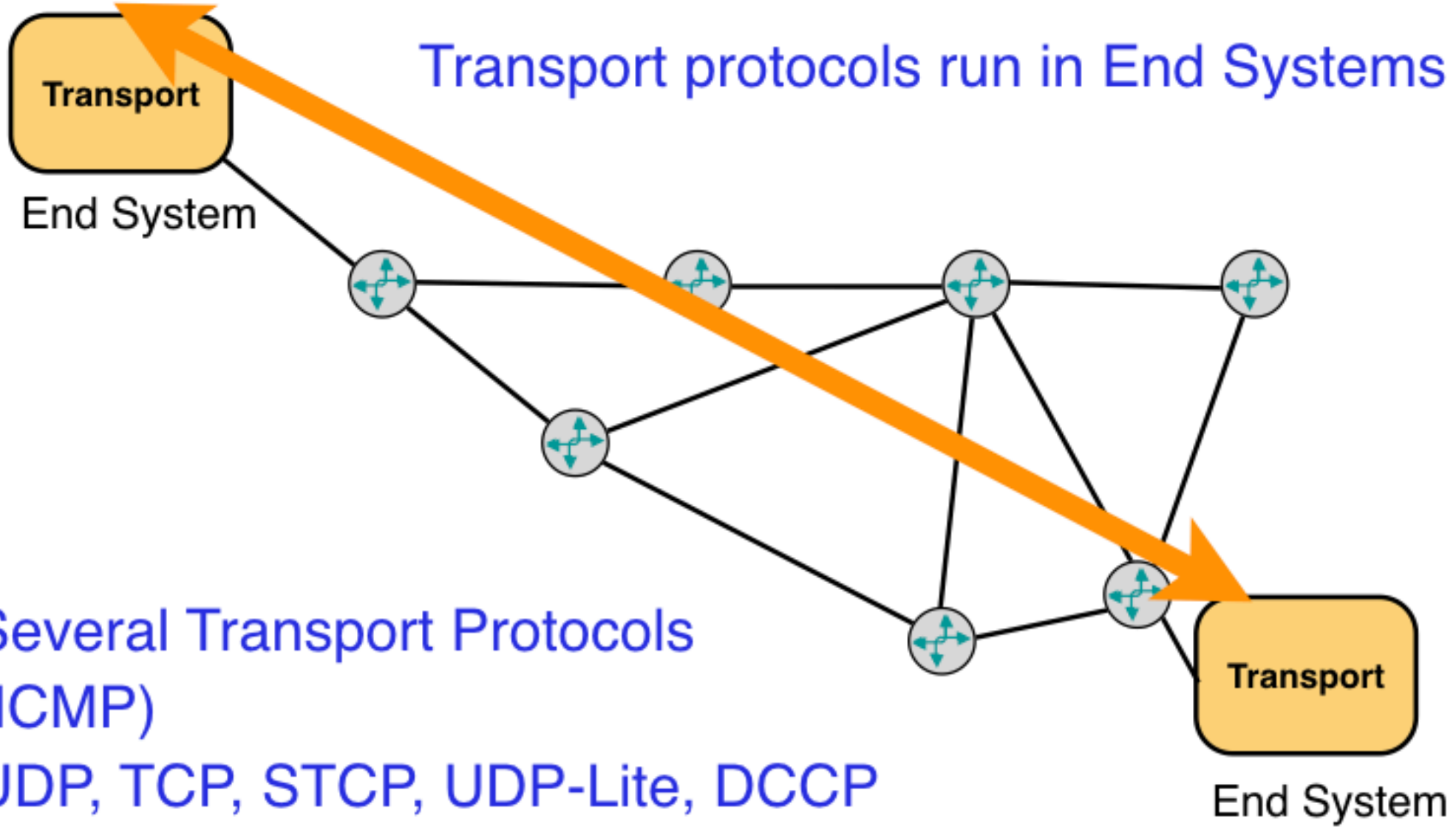


Transport Layer Service

G Fairhurst, http://www.erg.abdn.ac.uk

Logical link between applications

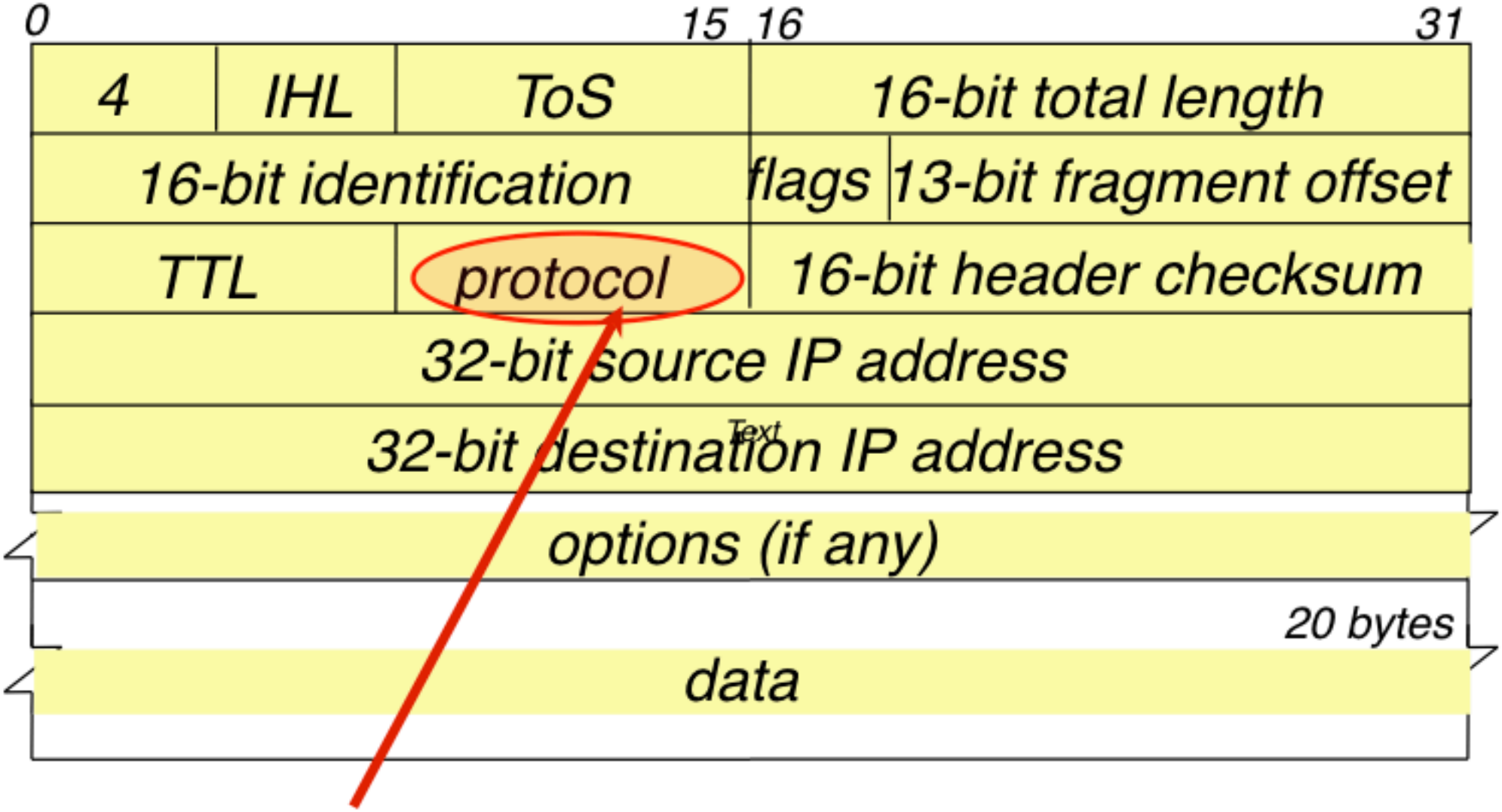
Transport protocols run in End Systems



Several Transport Protocols
(ICMP)
UDP, TCP, STCP, UDP-Lite, DCCP

IP Header

G Fairhurst, http://www.erg.abdn.ac.uk



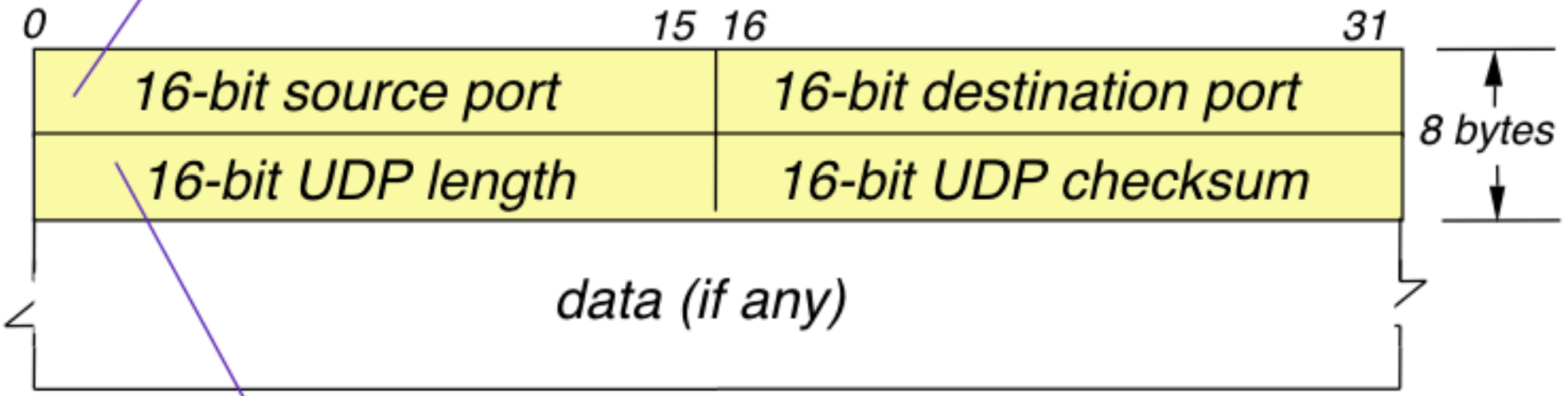
Layer 3 SAP
1 -> ICMP; 6 -> TCP; 17 -> UDP;

RFC 791

UDP Header

G Fairhurst, http://www.erg.abdn.ac.uk

Each connection identified by:
(dest IP, dest port))



Integrity checked by:
Verifying the length of PDU (incl. header)
Executing a checksum algorithm

RFC 768

Loss in Networks

G Fairhurst, http://www.erg.abdn.ac.uk

Why are datagram networks unreliable?

Transmission Frame Corruption (link CRC fails)

Router Header Corruption (IP Checksum fails)

Router Congestion (packet discarded by router)

Receiver Busy (packet discarded by end system)

No route to destination (packet discarded by router)

Equipment failure (packet discarded by router)

Each means packet does NOT reach the destination

Corruption in Networks

G Fairhurst, http://www.erg.abdn.ac.uk

Why are datagram networks unreliable? (II)

Corruption of packet:
- *inside* bridges
- *inside* routers
- *inside* end systems

Causes:
- Software errors (copy wrong data)
- DMA hardware faults

Errors in IP header detected by IP checksum
- Routers discard packets with header errors

Errors in IP payload undetected

A corrupted packet can reach the destination

.... YOU NEED A TRANSPORT CHECKSUM

16-Bit UDP Checksum

G Fairhurst, http://www.erg.abdn.ac.uk

Sender treats segment contents as sequence of 16-bit integers
Add (1's complement sum) of segment contents
Put checksum value into UDP checksum field

Receiver computes checksum of received segment
Is computed checksum equals checksum field value:
NO - error was detected.
YES - no error detected. But, may be errors nonetheless?

Some other things receivers check:
Addresses, Length, Protocol
These are also added into checksum

UDP Services

G Fairhurst, http://www.erg.abdn.ac.uk

Simple Services

- Startup / Bootstrap (DHCP, tftp)

Query / Response

- Disk Sharing (nfs)
- Address Query (dns)
- Time Query (ntp)
- Network Management (snmp)

Stream Services

- Audio, Video (Multimedia)
- Internet Telephony (Voice over IP)

Multicast

- Internet TV (Multicast Multimedia)
- File Distribution (Multicast File Transfer)

C2

Transport Layer

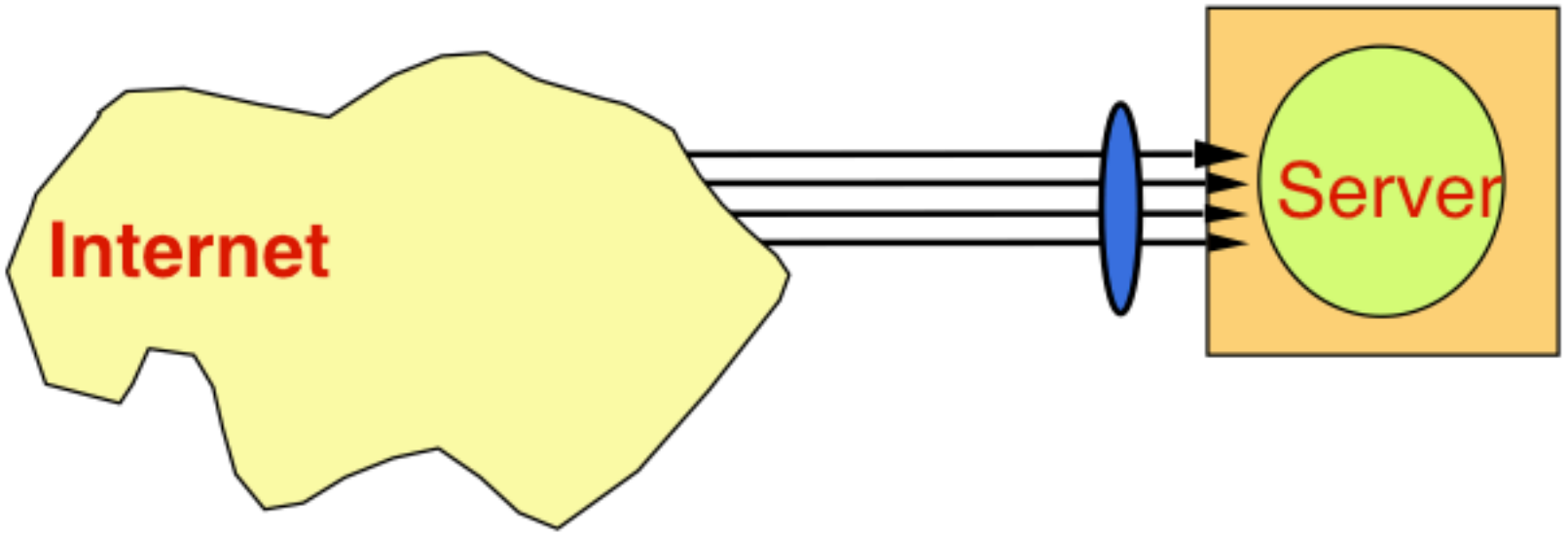
- Ports

Receiving IP packets

G Fairhurst, http://www.erg.abdn.ac.uk

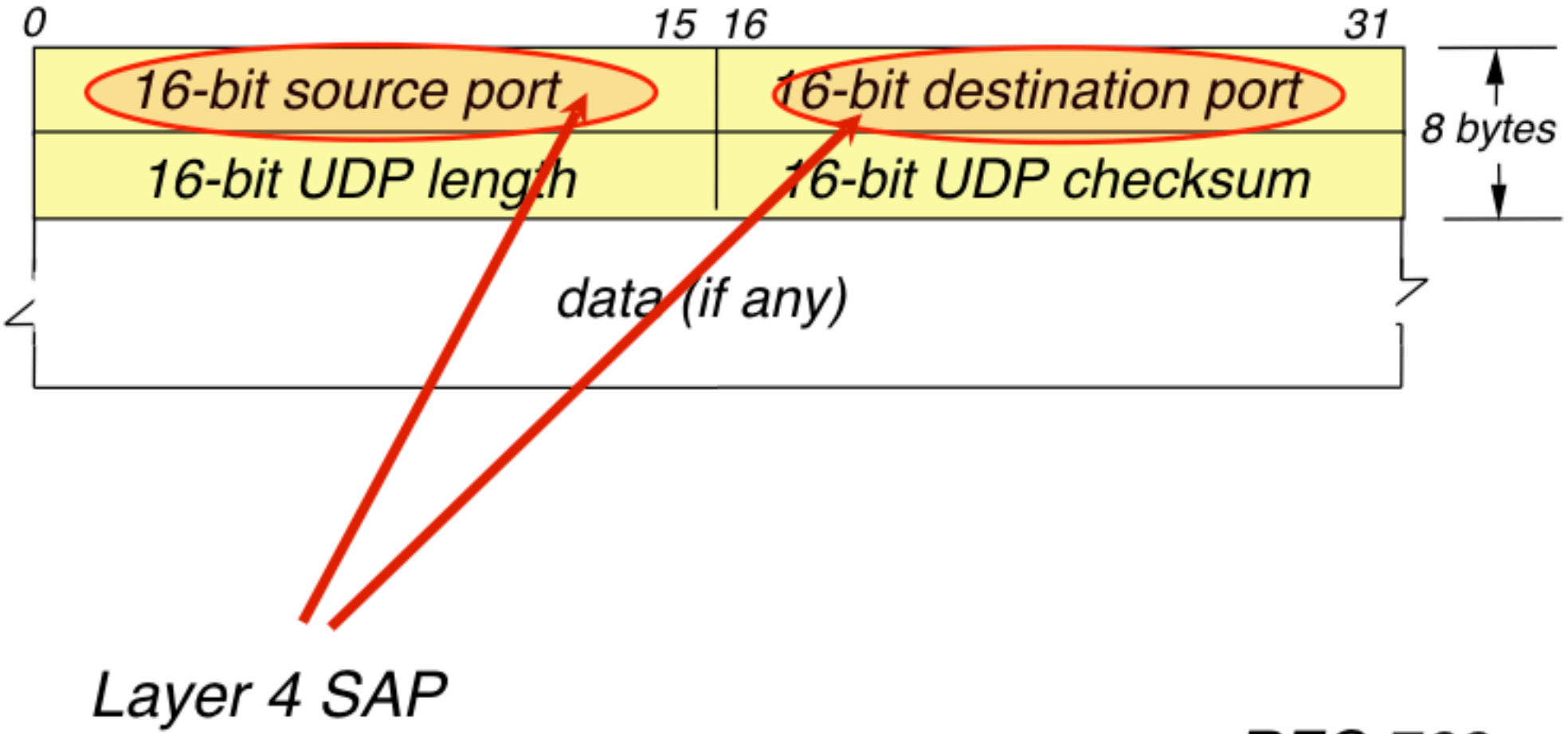
All packets with an IP destination address arrive at the same server

- The network does not care which “conversation” they belong to, it simply routes the packets



UDP Header

G Fairhurst, http://www.erg.abdn.ac.uk



RFC 768

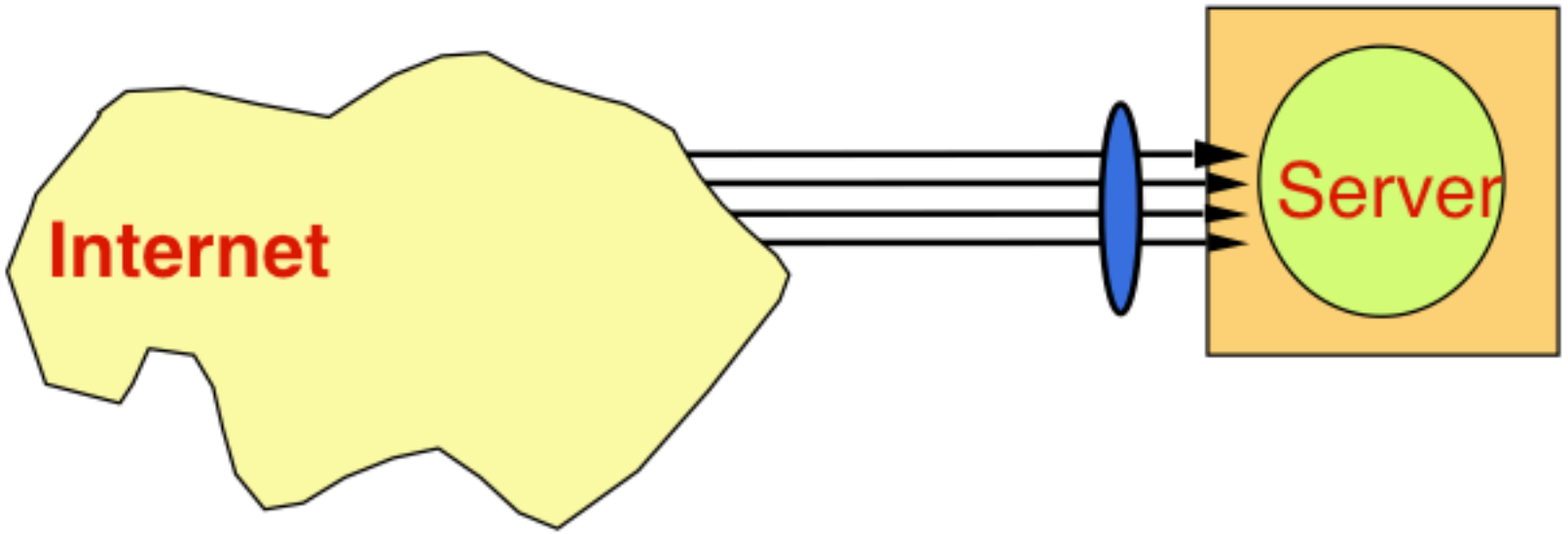
Port Numbers

G Fairhurst, http://www.erg.abdn.ac.uk

Clients and servers can connect to multiple systems

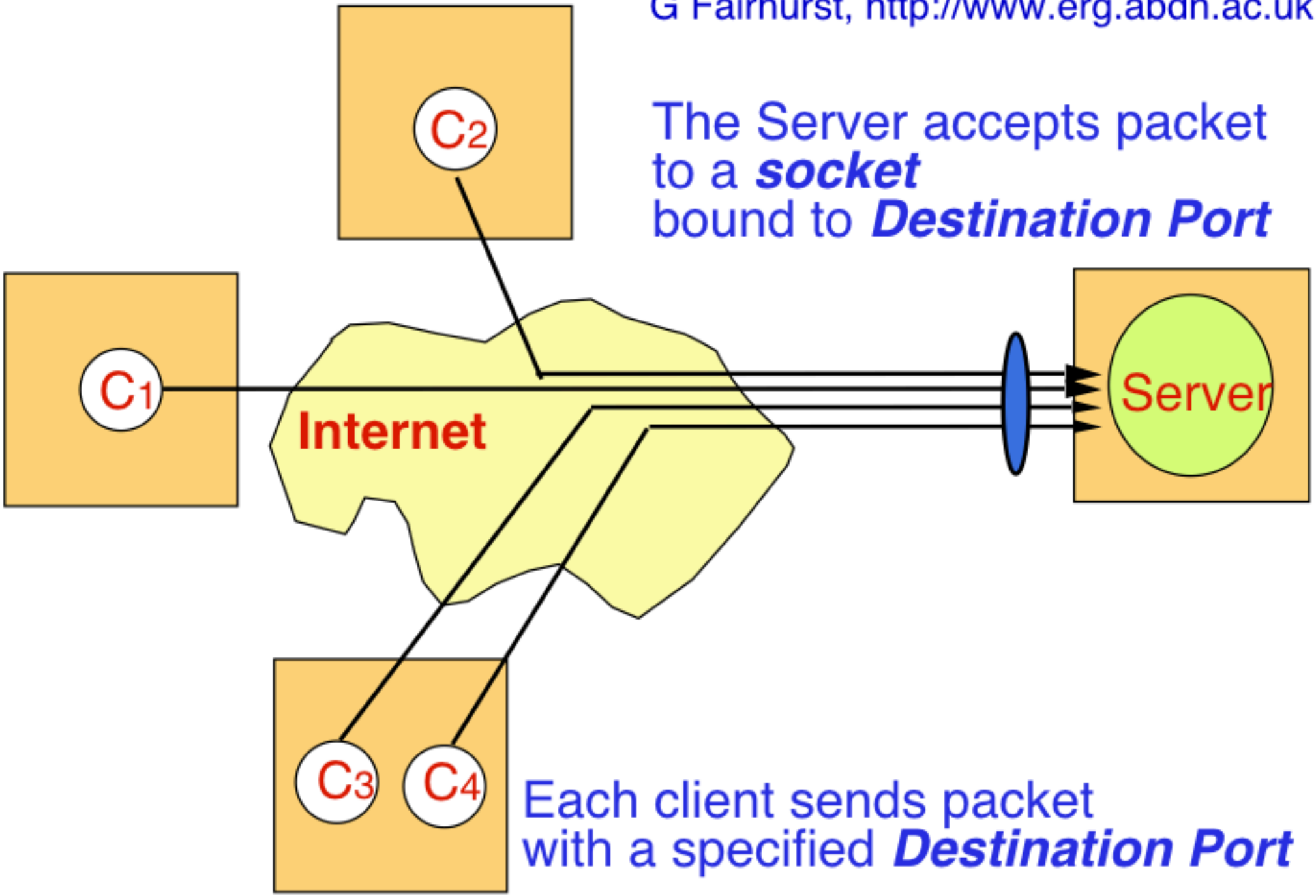
- How do they work out which packets belong to which conversation?

The answer is they use ports (e.g. UDP)



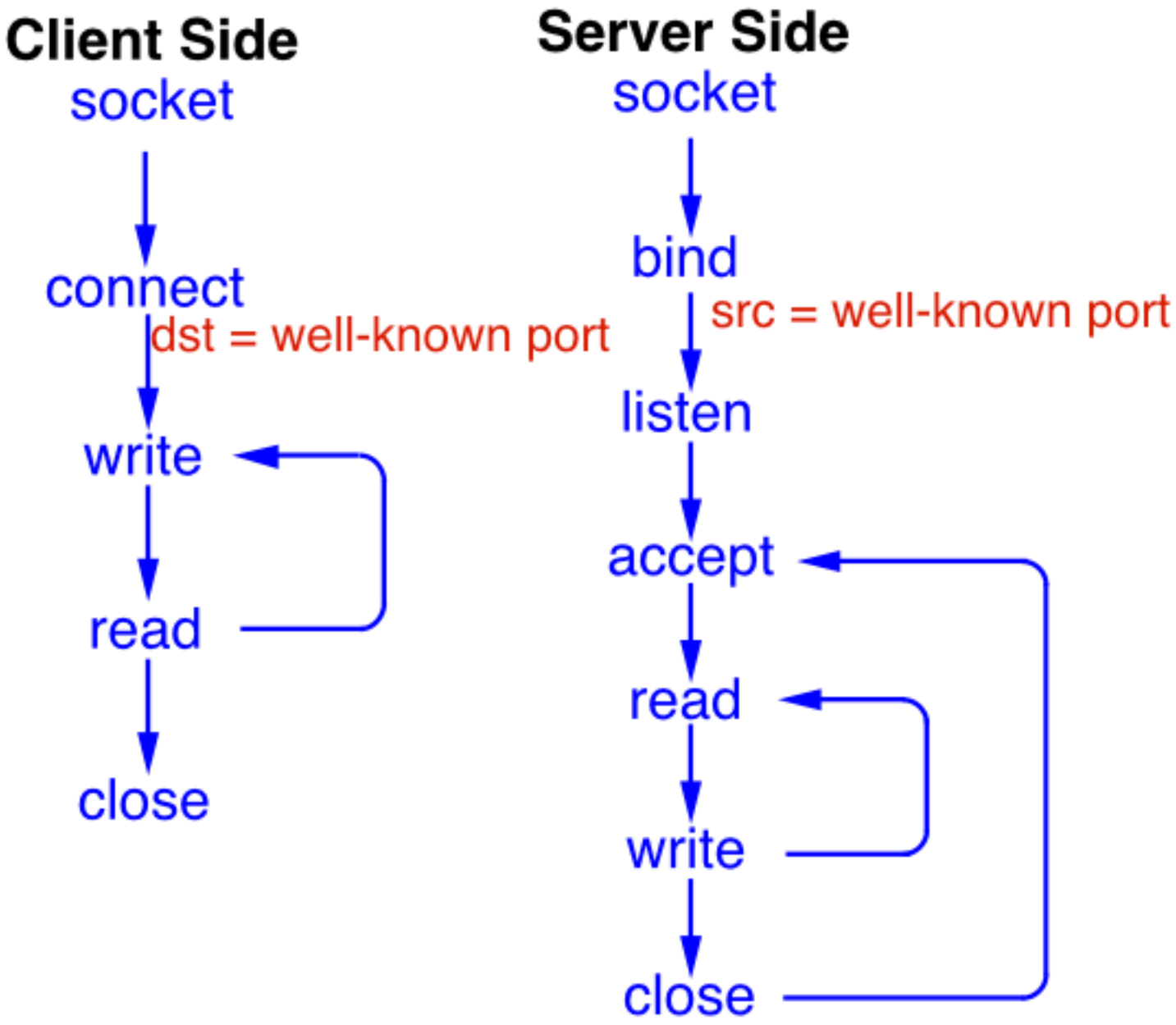
Port Numbers

G Fairhurst, http://www.erg.abdn.ac.uk



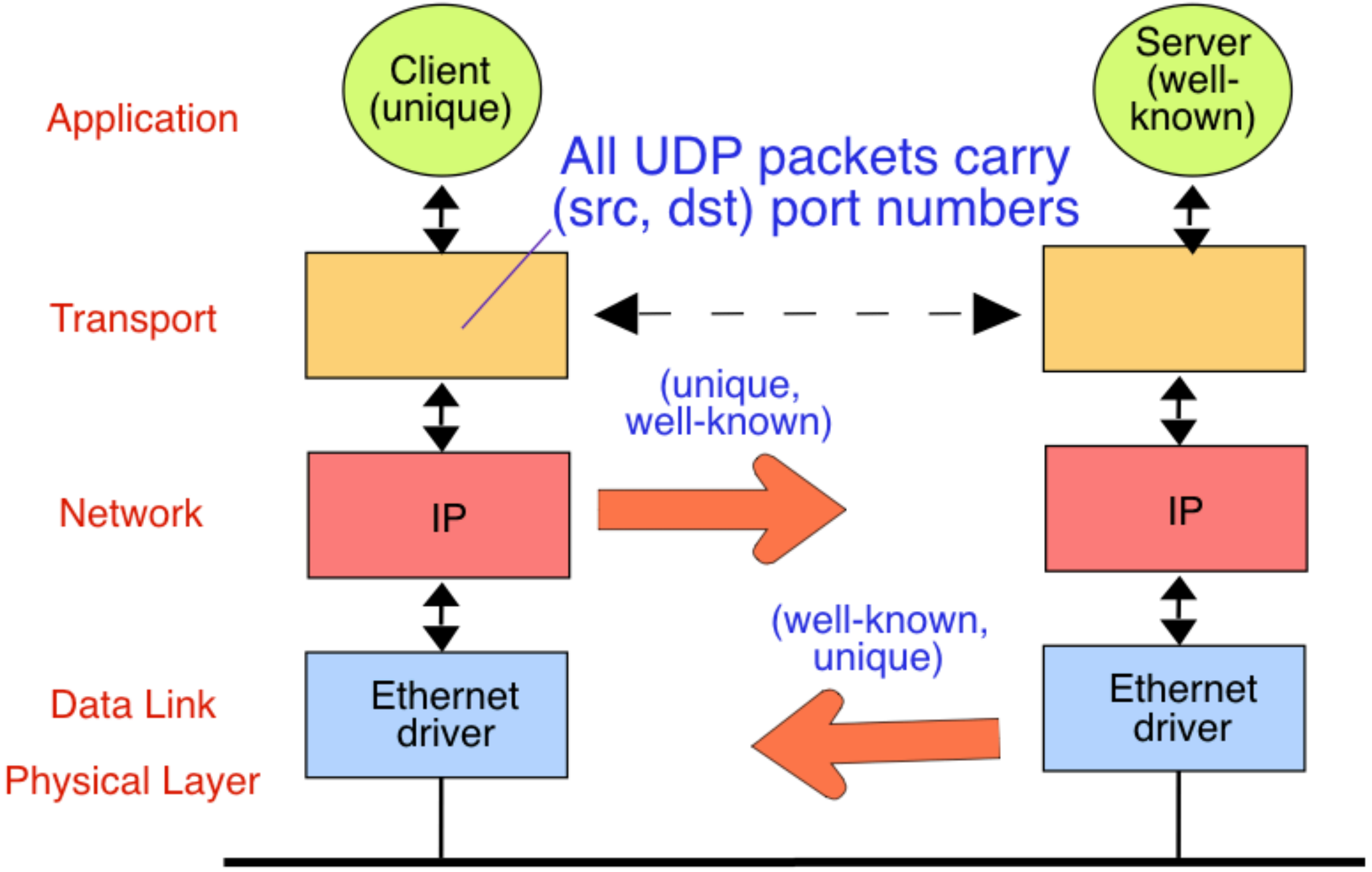
Client/Server Socket Interface

G Fairhurst, http://www.erg.abdn.ac.uk



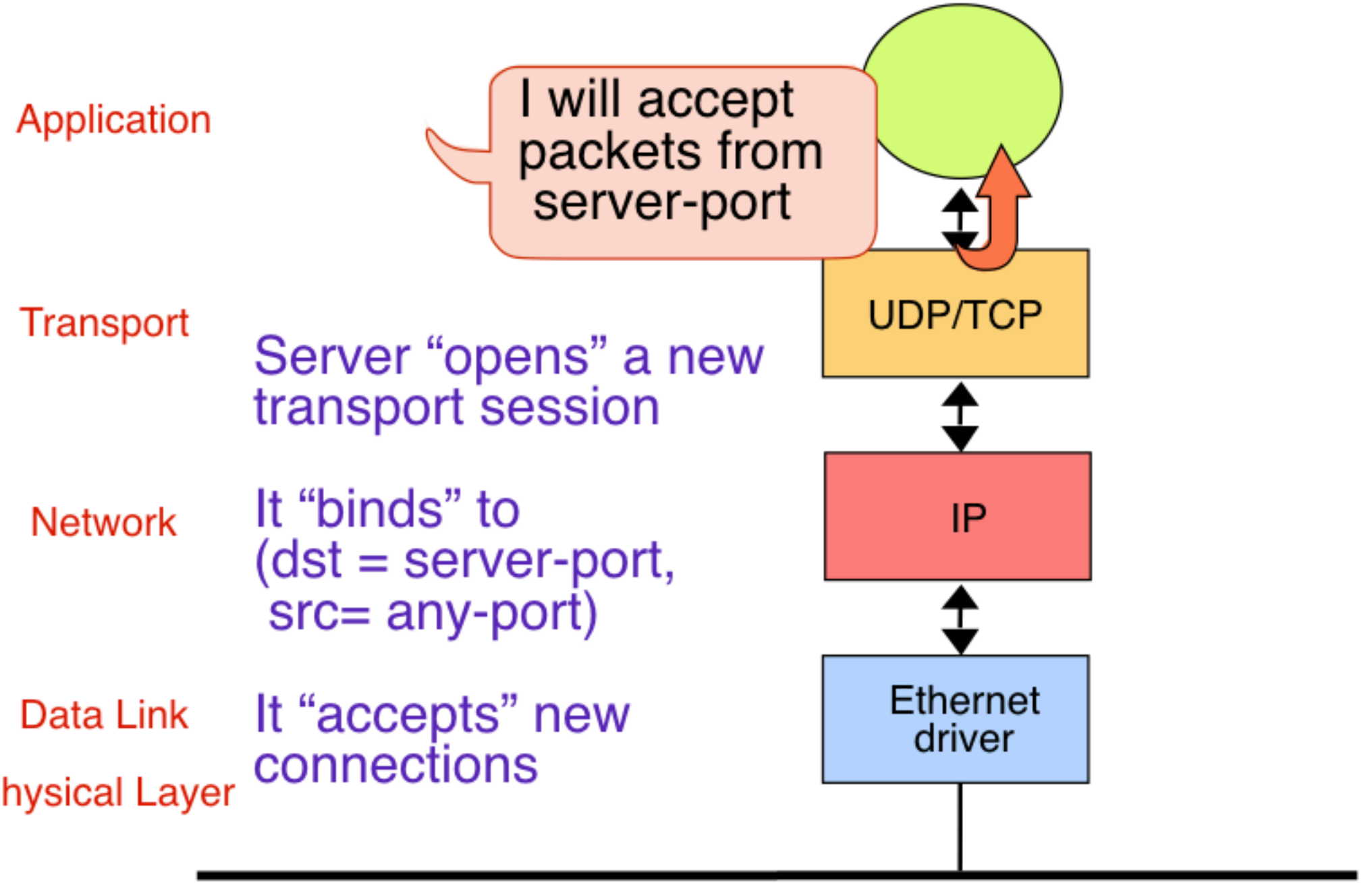
Src &Dest Port Numbers

G Fairhurst, http://www.erg.abdn.ac.uk



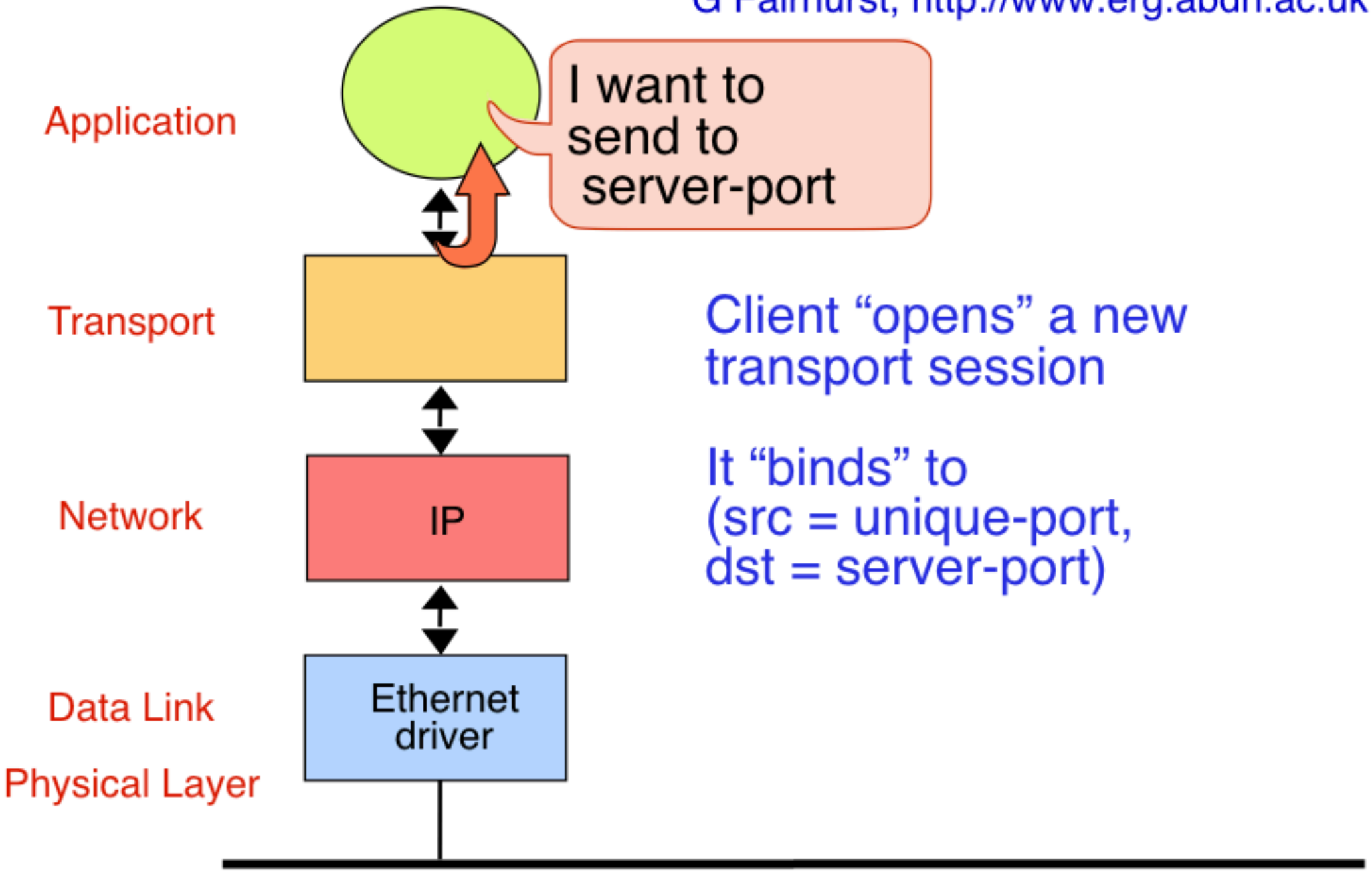
Well-Known Server Port Number

G Fairhurst, http://www.erg.abdn.ac.uk



Unique Client Port Number

G Fairhurst, http://www.erg.abdn.ac.uk



Client “opens” a new transport session

It “binds” to (src = unique-port, dst = server-port)

Well Known Port Numbers

G Fairhurst, http://www.erg.abdn.ac.uk

The Internet has agreed a set of well -known ports

There are lots of these - one for each service

Some examples are:

```
37          Network time Server (nntp)
53          Domain Name Server (dns)
```

*e.g. packets sent from a client **to** a dns server (53)*

IP header (client's IP addr, dns server IP addr)	UDP header (client port,53)

*e.g. packets (responses) **from** a dns server(3) to a client*

IP header (dns server IP addr, client's IP addr)	UDP header (53, client port)

G Fairhurst, http://www.erg.abdn.ac.uk

Multicast IP packet carrying UDP with RTP payload.

```
0:     0100 5e02 dc3e 00d0 bbf7 c6c0 0800 4500
16:    00cc e206 0000 7111 a1a9 84b9 8476 e002
32:    dc3e 7982 7982 00b8 08a0 8005 dbc6 d721
48:    69c0 0752 bb5f fe39 3600 8808 b120 8933
64:    6219 9118 5128 ffc8 1321 bc10 933e aa23
80:    3233 ba00 e892 a00c 1a3c 0a28 37ab 012d
96:    aca5 4819 9088 0b39 64ba 43a0 b9a8 04b3
112:   88b8 4bf8 3940 d024 0a98 8b0b 1703 0a3a
128:   8820 a381 a21f 3bc0 9298 e893 90bd 042a
144:   0a88 3287 59ab e980 1211 4002 2208 98b1
160:   7039 0b26 e898 99ab b118 a1aa a702 9ac4
176:   9128 ca21 7822 2971 090a 2194 98d0 27bb
192:   0958 8092 993f b3b0 2922 337a 0f88 8810
208:   8a29 0183 fb15 b888 0d4c
```

(a) How do we know this is a multicast packet?
    (i) at link layer?
    (ii) at network layer?
(b) How do we know this is an RTP packet?

C3

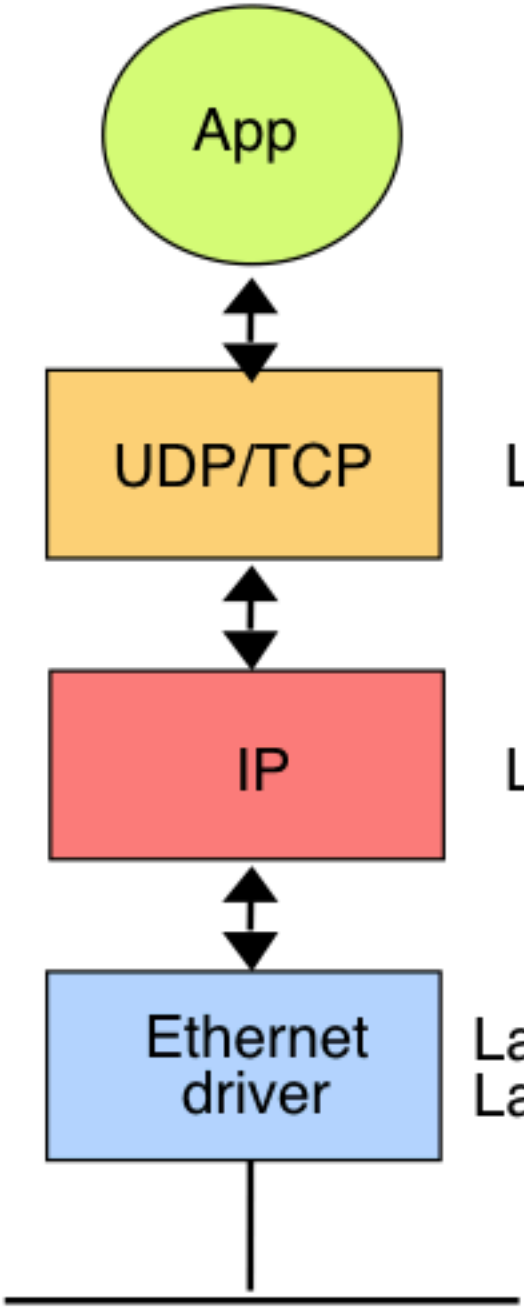
Packet Decodes

- Ethernet Header
- PDU Header Chart
- Hexadecimal Packet dump

```
  0: 0100 5e02 dc3e 00d0 bbf7 c6c0 0800 4500
 16: 00cc e206 0000 7111 a1a9 84b9 8476 e002
 32: dc3e 7982 7982 00b8 08a0 8005 dbc6 d721
 48: 69c0 0752 bb5f fe39 3600 8808 b120 8933
 64: 6219 9118 5128 ffc8 1321 bc10 933e aa23
 80: 3233 ba00 e892 a00c 1a3c 0a28 37ab 012d
 96: aca5 4819 9088 0b39 64ba 43a0 b9a8 04b3
112: 88b8 4bf8 3940 d024 0a98 8b0b 1703 0a3a
128: 8820 a381 a21f 3bc0 9298 e893 90bd 042a
```

Protocol Layers

G Fairhurst, http://www.erg.abdn.ac.uk



As user data is transmitted,
each stack layer adds protocol headers
the Protocol Data Units get bigger

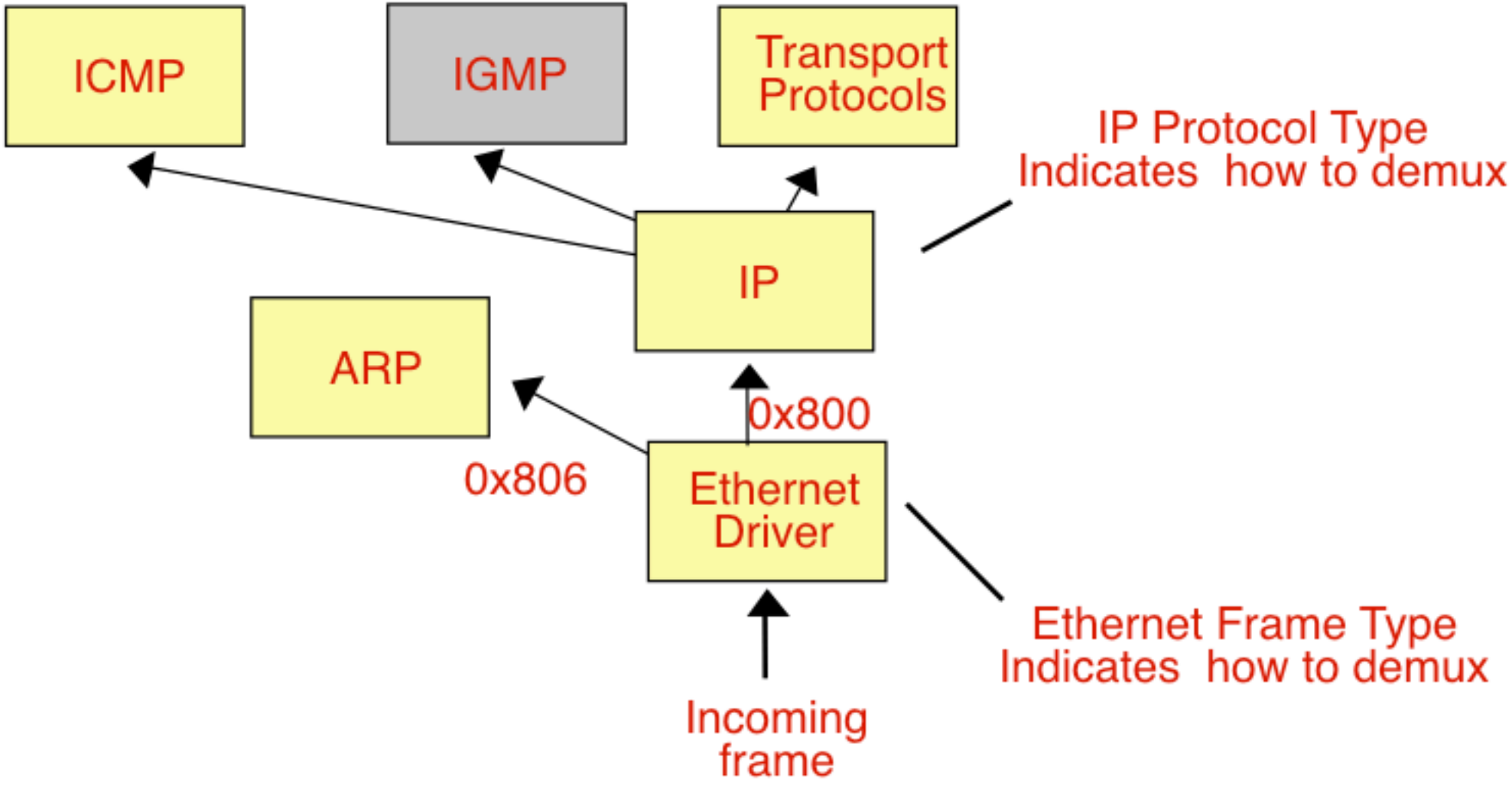
Layer 4:Transport Layer

Layer 3:Network Layer

Layer 2: Link Layer (MAC)
Layer 1:Physical Layer

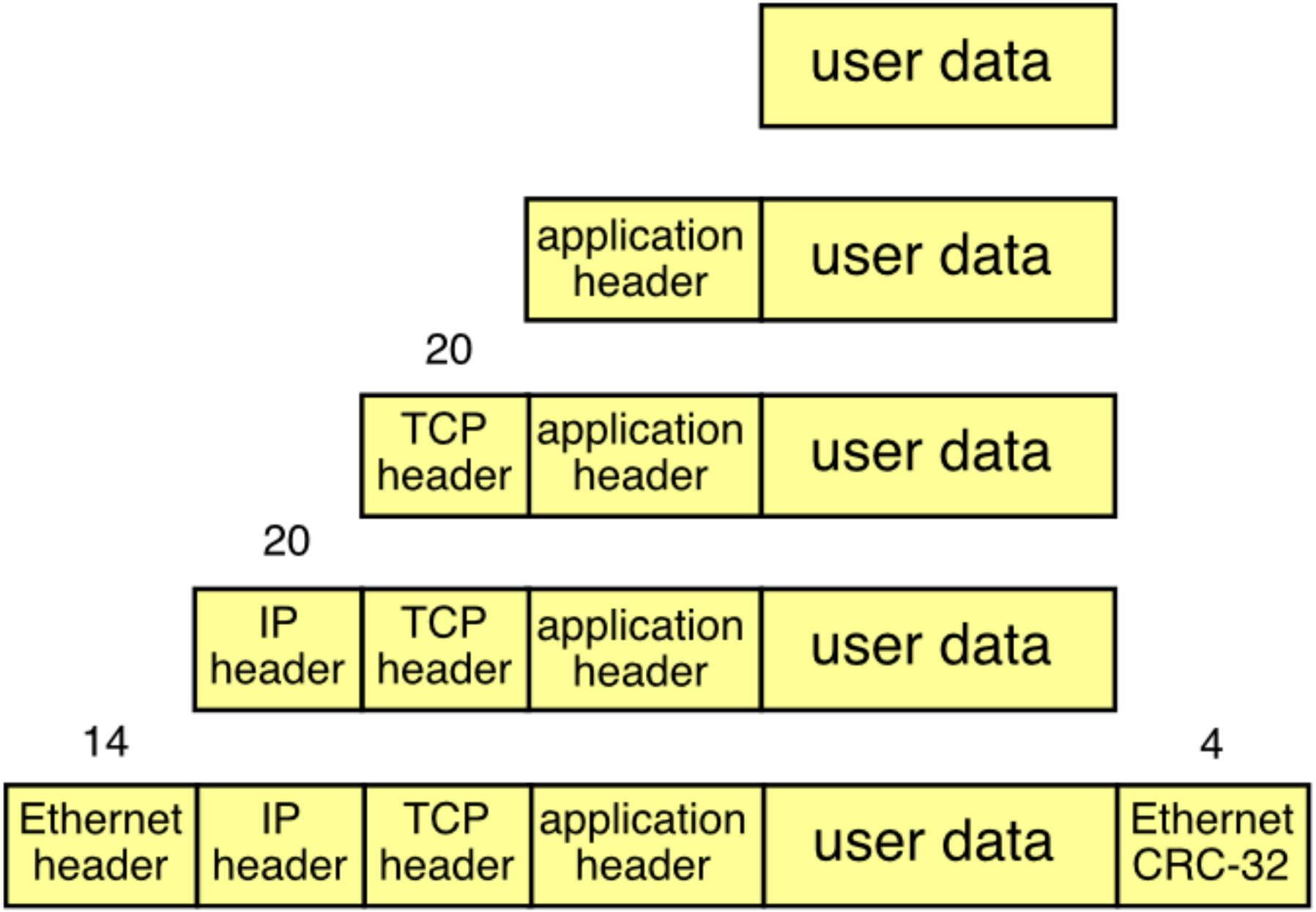
Protocol Demultiplexing

G Fairhurst, http://www.erg.abdn.ac.uk



Encapsulation

G Fairhurst, http://www.erg.abdn.ac.uk



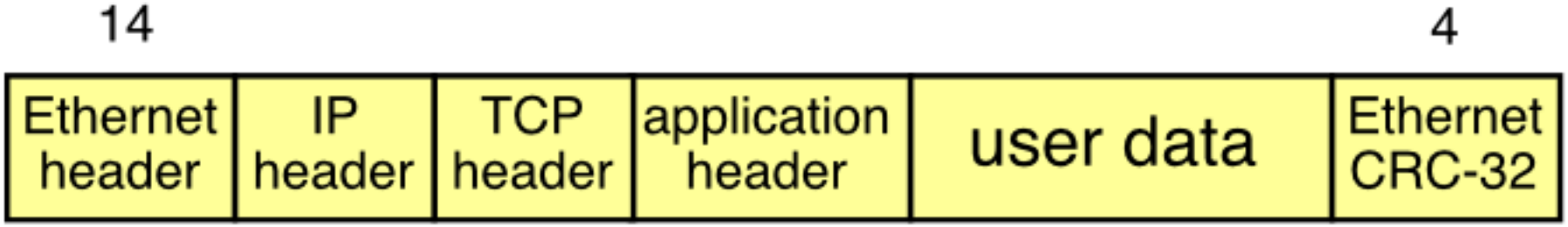
Decoding Packets

G Fairhurst, http://www.erg.abdn.ac.uk

Packets can be captured using an Ethernet Interface in *promiscuous mode*

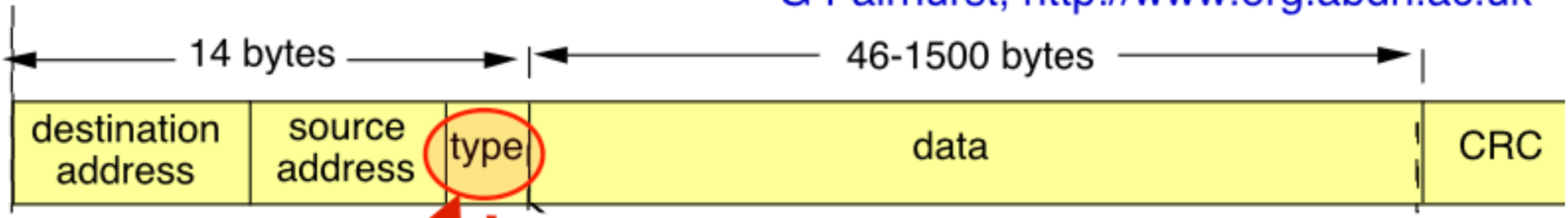
These need to be decoded by looking at each header in turn, starting at the outer and working inwards.

Each header provides a Service Access Point (SAP) that identifies the type of data carried within



Ethernet Header

G Fairhurst, http://www.erg.abdn.ac.uk

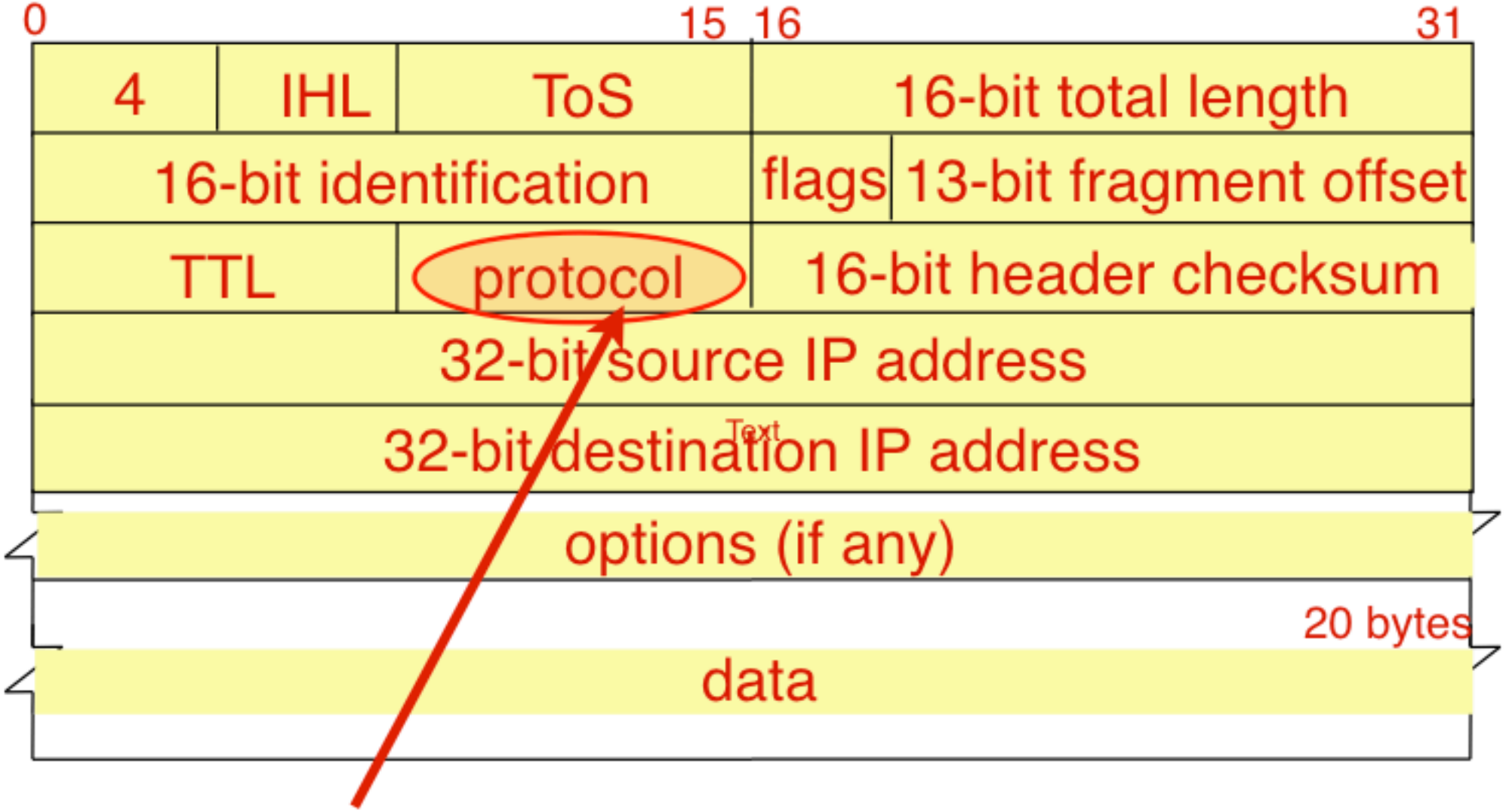


0x800 -> indicates an IP packet
0x806 -> indicates an ARP frame
0x86DD -> indicates an IPv6 packet

Layer 2 SAP

IP Header

G Fairhurst, http://www.erg.abdn.ac.uk

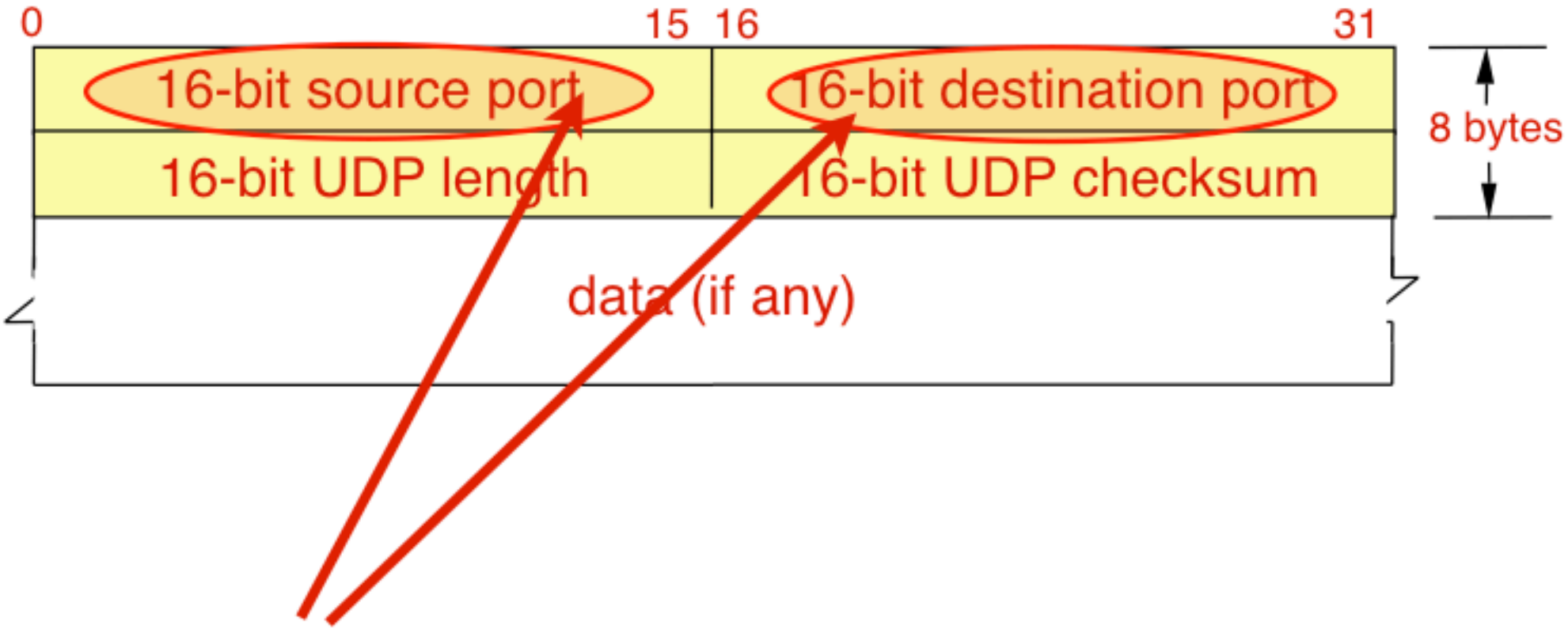


Layer 3 SAP
1 -> ICMP; 6 -> TCP; 17 -> UDP;

RFC 791

UDP Header

G Fairhurst, http://www.erg.abdn.ac.uk



RFC 768

Decoding Packets

G Fairhurst, http://www.erg.abdn.ac.uk

Each protocol header has a service access point
Each shows which header follows next
0x0800 in the EtherType field indicates an IP header
The IP header has a protocol type field of 11, 0x17...

```
  0: 0100 5e02 dc3e 00d0 bbf7 c6c0 0800 4500
 16: 00cc e206 0000 7111 a1a9 84b9 8476 e002
 32: dc3e 7982 7982 00b8 08a0 8005 dbc6 d721
 48: 69c0 0752 bb5f fe39 3600 8808 b120 8933
 64: 6219 9118 5128 ffc8 1321 bc10 933e aa23
 80: 3233 ba00 e892 a00c 1a3c 0a28 37ab 012d
 96: aca5 4819 9088 0b39 64ba 43a0 b9a8 04b3
112: 88b8 4bf8 3940 d024 0a98 8b0b 1703 0a3a
128: 8820 a381 a21f 3bc0 9298 e893 90bd 042a
144: 0a88 3287 59ab e980 1211 4002 2208 98b1
160: 7039 0b26 e898 99ab b118 a1aa a702 9ac4
176: 9128 ca21 7822 2971 090a 2194 98d0 27bb
192: 0958 8092 993f b3b0 2922 337a 0f88 8810
208: 8a29 0183 fb15 b888 0d4c
```

Decoding Packets

G Fairhurst, http://www.erg.abdn.ac.uk

```
  0: 0100 5e02 dc3e 00d0 bbf7 c6c0 0800 4500
 16: 00cc e206 0000 7111 a1a9 84b9 8476 e002
 32: dc3e 7982 7982 00b8 08a0 8005 dbc6 d721
 48: 69c0 0752 bb5f fe39 3600 8808 b120 8933
 64: 6219 9118 5128 ffc8 1321 bc10 933e aa23
 80: 3233 ba00 e892 a00c 1a3c 0a28 37ab 012d
 96: aca5 4819 9088 0b39 64ba 43a0 b9a8 04b3
112: 88b8 4bf8 3940 d024 0a98 8b0b 1703 0a3a
128: 8820 a381 a21f 3bc0 9298 e893 90bd 042a
144: 0a88 3287 59ab e980 1211 4002 2208 98b1
160: 7039 0b26 e898 99ab b118 a1aa a702 9ac4
176: 9128 ca21 7822 2971 090a 2194 98d0 27bb
192: 0958 8092 993f b3b0 2922 337a 0f88 8810
208: 8a29 0183 fb15 b888 0d4c
```

Decoding Packets

G Fairhurst, http://www.erg.abdn.ac.uk

ETHER
Packet size = 218 bytes
Destination = 1:0:5e:2:dc:3e, (multicast) (01-00-5e-02-dc-3e)
Source = 0:d0:bb:f7:c6:c0,
Ethertype = 0800 (IPv4)
IP
Version = 4, Header length = 20 bytes
Type of service = 0x00
Total length = 204 bytes (00cc)
ID = 57862, Flags = 0x00, Frags = 0
Time To Live = 113 seconds/hops
Protocol = 17 (UDP)
Header checksum = a1a9
Source address = 132.185.132.118
Destination address = 224.2.220.62
No options
UDP
Source port = 31106 (7982)
Destination port = 31106 (7982)
Length = 184 (00b8)
Checksum = 08a0
RTP
180B of Data

```
  0: 0100 5e02 dc3e 00d0 bbf7 c6c0 0800 4500
 16: 00cc e206 0000 7111 a1a9 84b9 8476 e002
 32: dc3e 7982 7982 00b8 08a0 8005 dbc6 d721
 48: 69c0 0752 bb5f fe39 3600 8808 b120 8933
 64: 6219 9118 5128 ffc8 1321 bc10 933e aa23
 80: 3233 ba00 e892 a00c 1a3c 0a28 37ab 012d
 96: aca5 4819 9088 0b39 64ba 43a0 b9a8 04b3
112: 88b8 4bf8 3940 d024 0a98 8b0b 1703 0a3a
128: 8820 a381 a21f 3bc0 9298 e893 90bd 042a
144: 0a88 3287 59ab e980 1211 4002 2208 98b1
160: 7039 0b26 e898 99ab b118 a1aa a702 9ac4
176: 9128 ca21 7822 2971 090a 2194 98d0 27bb
192: 0958 8092 993f b3b0 2922 337a 0f88 8810
208: 8a29 0183 fb15 b888 0d4c
```

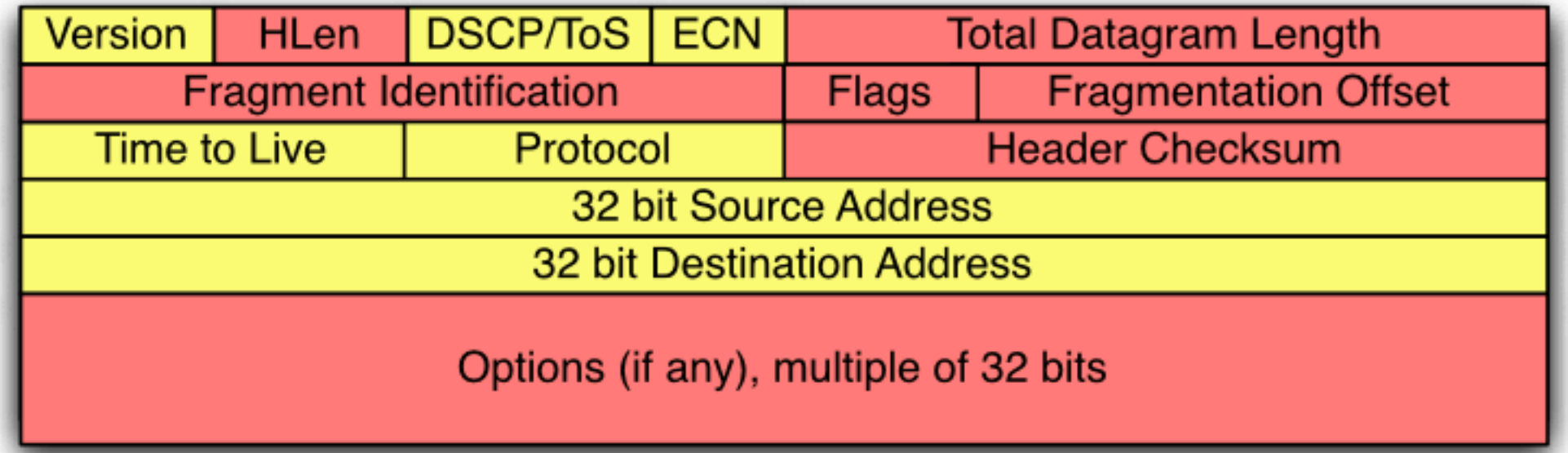
C4
IPv6

G Fairhurst, http://www.erg.abdn.ac.uk

```
0000:  47 5c 8f 15 00 80 6c 86  dd 60 00 00 00 00 40 3a
0010:  40 20 10 0d b8 85 a3 08  d3 13 19 8a 2e 03 70 73
0020:  35 20 10 0d b8 85 a3 08  d3 13 19 8a 2e 03 70 73
0030:  35 80 00 e9 6b 77 3d 00  04 9b 56 d9 47 00 00 00
0040:  00 3e 0f 0d 00 00 00 00  00 10 11 12 13 14 15 16
0050:  17 18 19 1a 1b 1c 1d 1e  1f 20 21 22 23 24 25 26
0060:  27 28 29 2a 2b 2c 2d 2e  2f 30 31 32 33 34 35 36
0070:  37 8f 05 4a 29 ff ff ff  ff ff ff ff ff ff ff ff
0080:  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff
0090:  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff
00a0:  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff
00b0:  ff ff ff ff ff ff ff ff  ff ff ff ff
```

Fields in IPv4 Header use by IPv6

G Fairhurst, http://www.erg.abdn.ac.uk



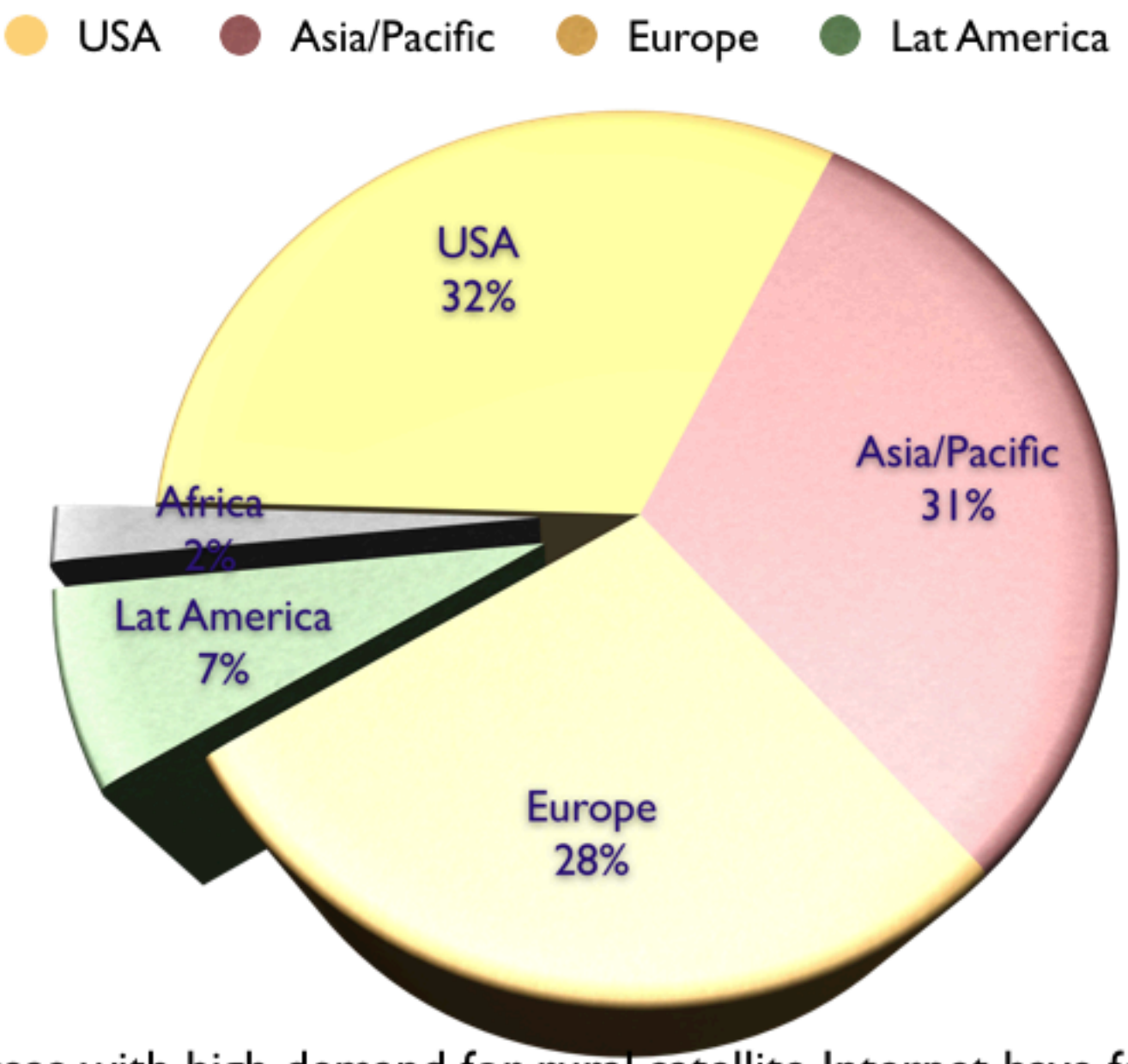
32-bit address in IPv4 limits number of addressable systems

Monolithic header (complex)

- Options not widely implemented
- Difficult to provide extensions for new applications

Distribution of allocated IPv4 Addresses

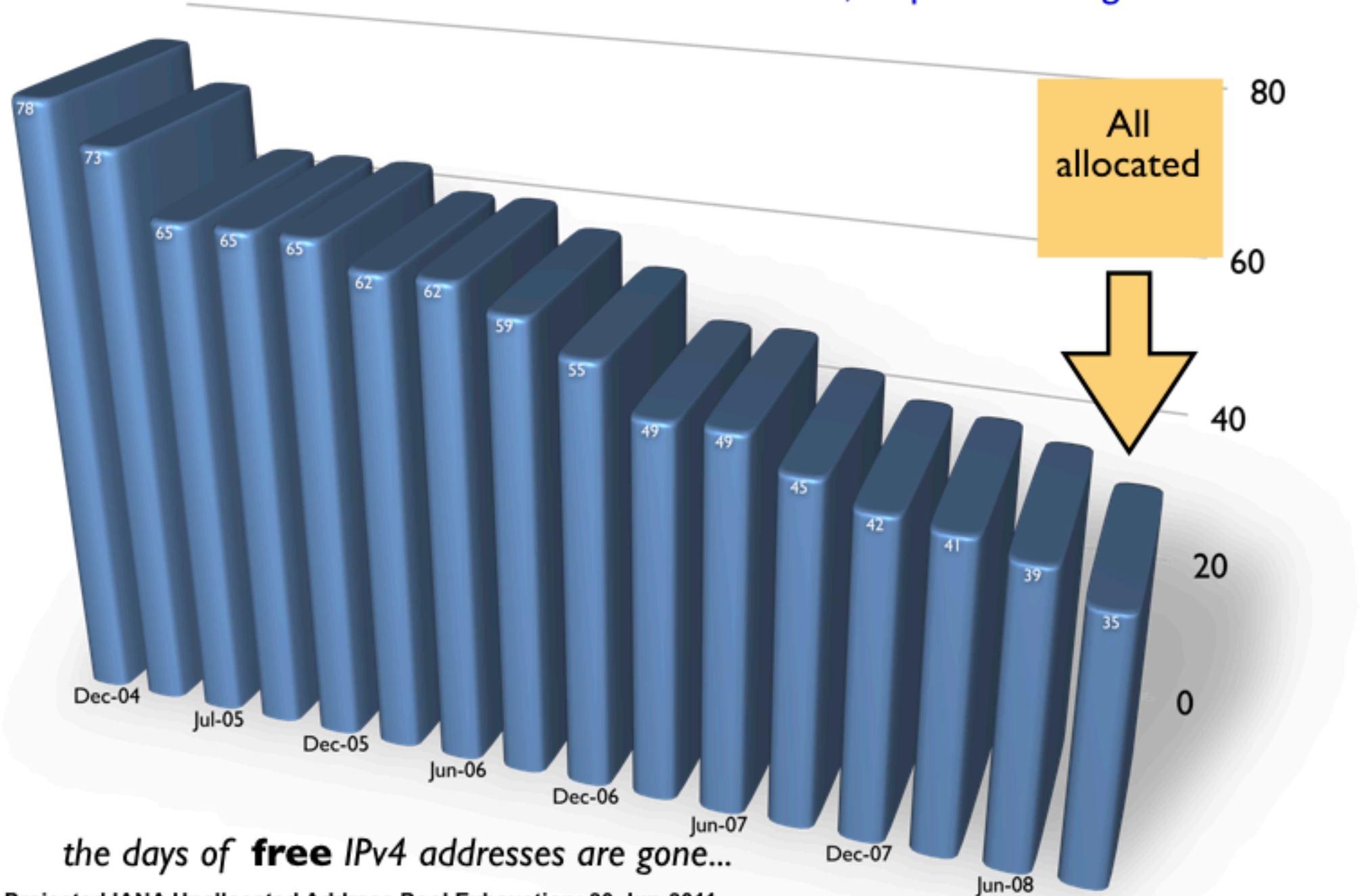
G Fairhurst, http://www.erg.abdn.ac.uk



Areas with high demand for rural satellite Internet have few IPv4 addresses

Unallocated IANA IPv4 /8 Addresses

G Fairhurst, http://www.erg.abdn.ac.uk



the days of **free** *IPv4 addresses are gone...*

Projected IANA Unallocated Address Pool Exhaustion: 20-Jun-2011

http://www.potaroo.net/tools/ipv4/index.html

Transitioning to a new Internet Protocol

G Fairhurst, http://www.erg.abdn.ac.uk

Current Internet

(1981)

Next Gen. Internet (1994)

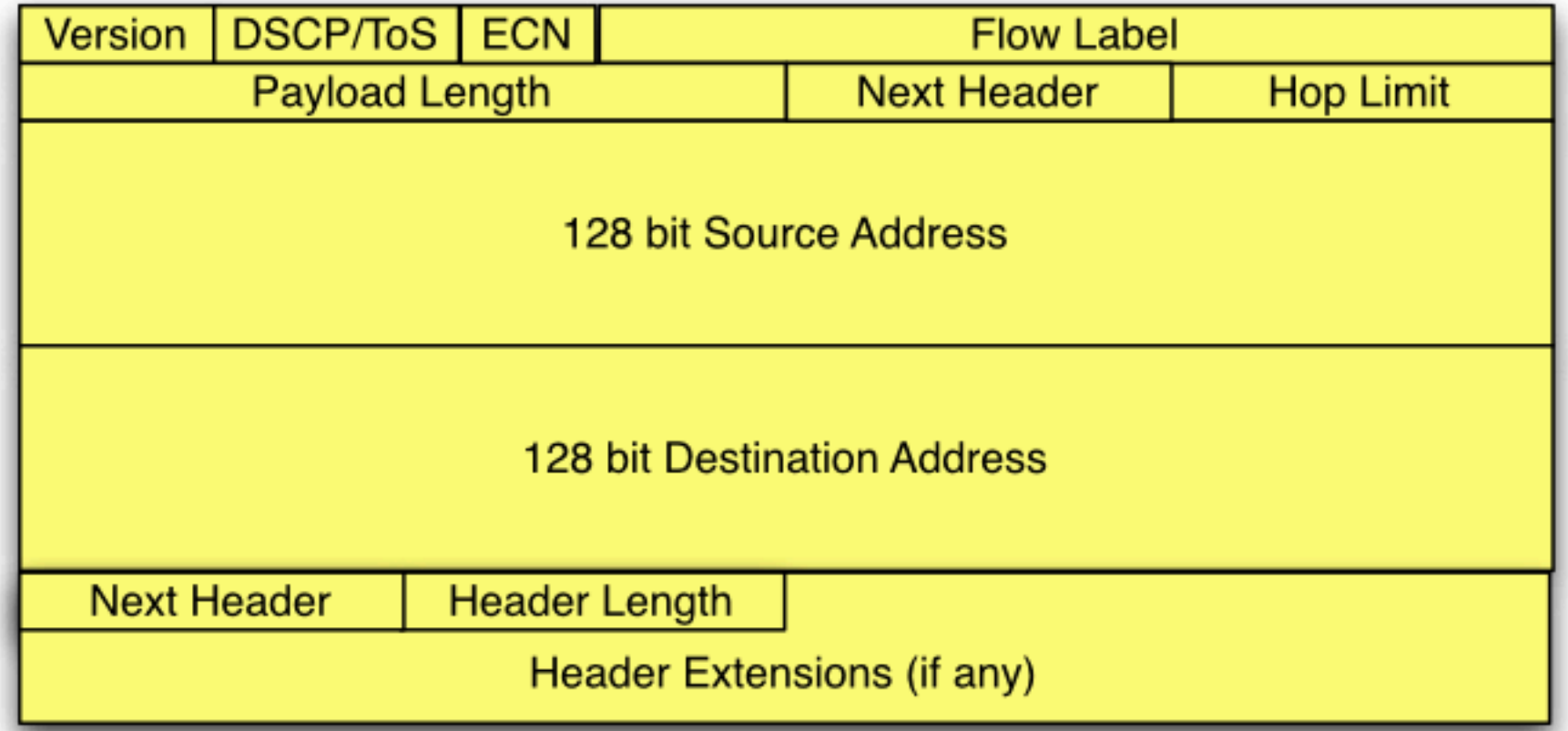
IPv6 launch (2012)

Protocol now widely deployed

Some networks (e.g. IoT) now only support IPv6 systems

"New" IPv6 Functions

G Fairhurst, http://www.erg.abdn.ac.uk



Simplified header format

- Good for router speed
- No network broadcast, multicast by default

Expands addressing 128-bit

- Enough for everyone - no NAT!

Improved support for Extensions (e.g. security)

IPv6 Decode

G Fairhurst, http://www.erg.abdn.ac.uk

```
0000:  47 5c 8f 15 00 80 6c 86  dd 60 00 00 00 00 40 3a
0010:  40 20 10 0d b8 85 a3 08  d3 13 19 8a 2e 03 70 73
0020:  35 20 10 0d b8 85 a3 08  d3 13 19 8a 2e 03 70 73
0030:  35 80 00 e9 6b 77 3d 00  04 9b 56 d9 47 00 00 00
0040:  00 3e 0f 0d 00 00 00 00  00 10 11 12 13 14 15 16
0050:  17 18 19 1a 1b 1c 1d 1e  1f 20 21 22 23 24 25 26
0060:  27 28 29 2a 2b 2c 2d 2e  2f 30 31 32 33 34 35 36
0070:  37 8f 05 4a 29 ff ff ff  ff ff ff ff ff ff ff ff
0080:  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff
0090:  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff
00a0:  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff
00b0:  ff ff ff ff ff ff ff ff  ff ff ff ff
```

Colours show the bytes for different headers.
Purple part is the IPv6 packet header

IPv6 Status

G Fairhurst, http://www.erg.abdn.ac.uk

Standard on all router platforms

Common on high-end switches

Standard in modern host operating systems

•

http://www.ipv6forum.com/

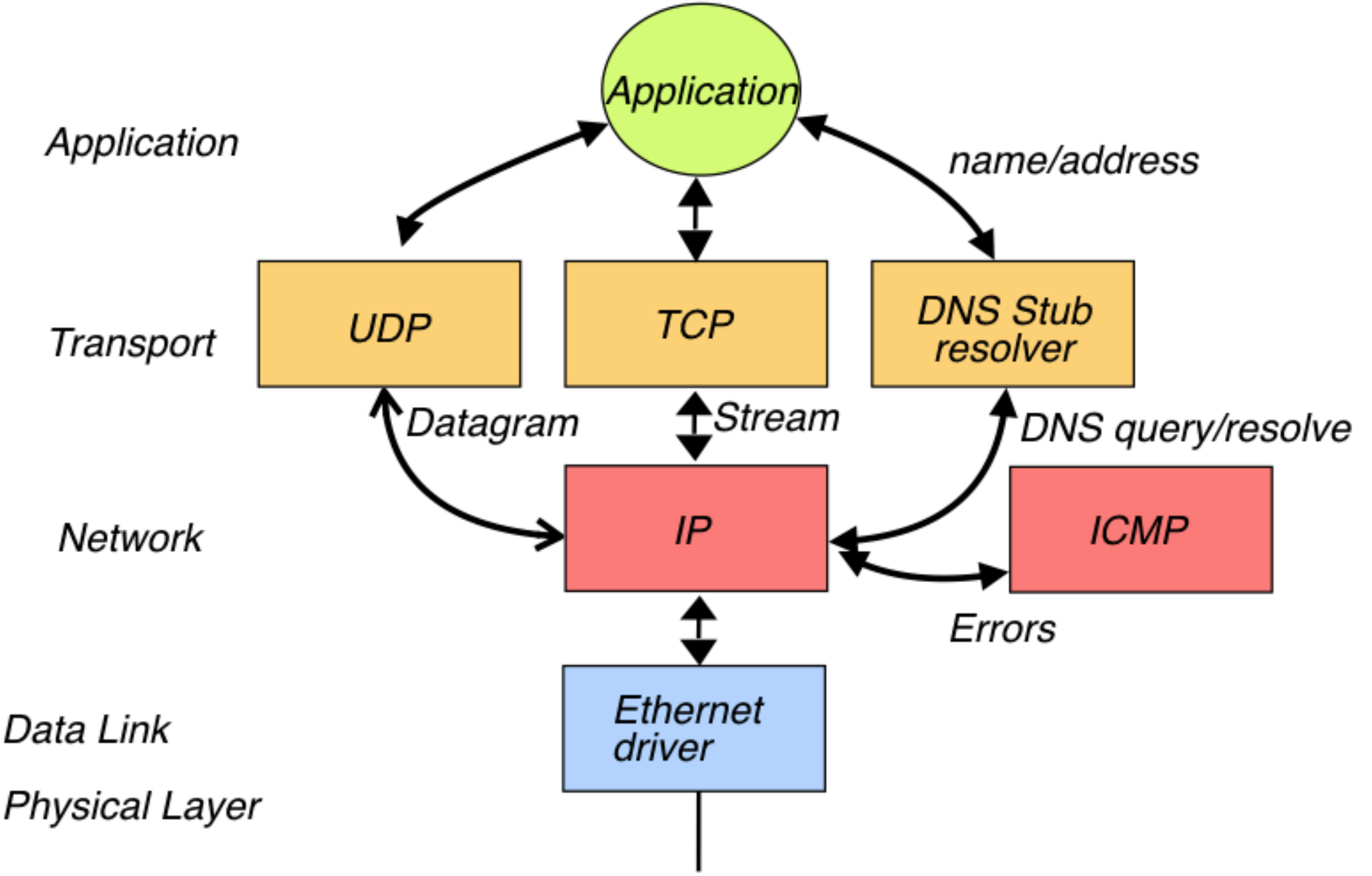
G Fairhurst, http://www.erg.abdn.ac.uk

TCP Transmission Control Protocol

- Integrity Check (as UDP)
- Multiplexing (similar to UDP)
- Reliable In-Order Delivery (retransmits)
- Stream-oriented Transport
- Flow Control (receiver slows-down sender)
- Congestion Avoidance (network slows-down sender)
- Out-Of-Band Data (little used)

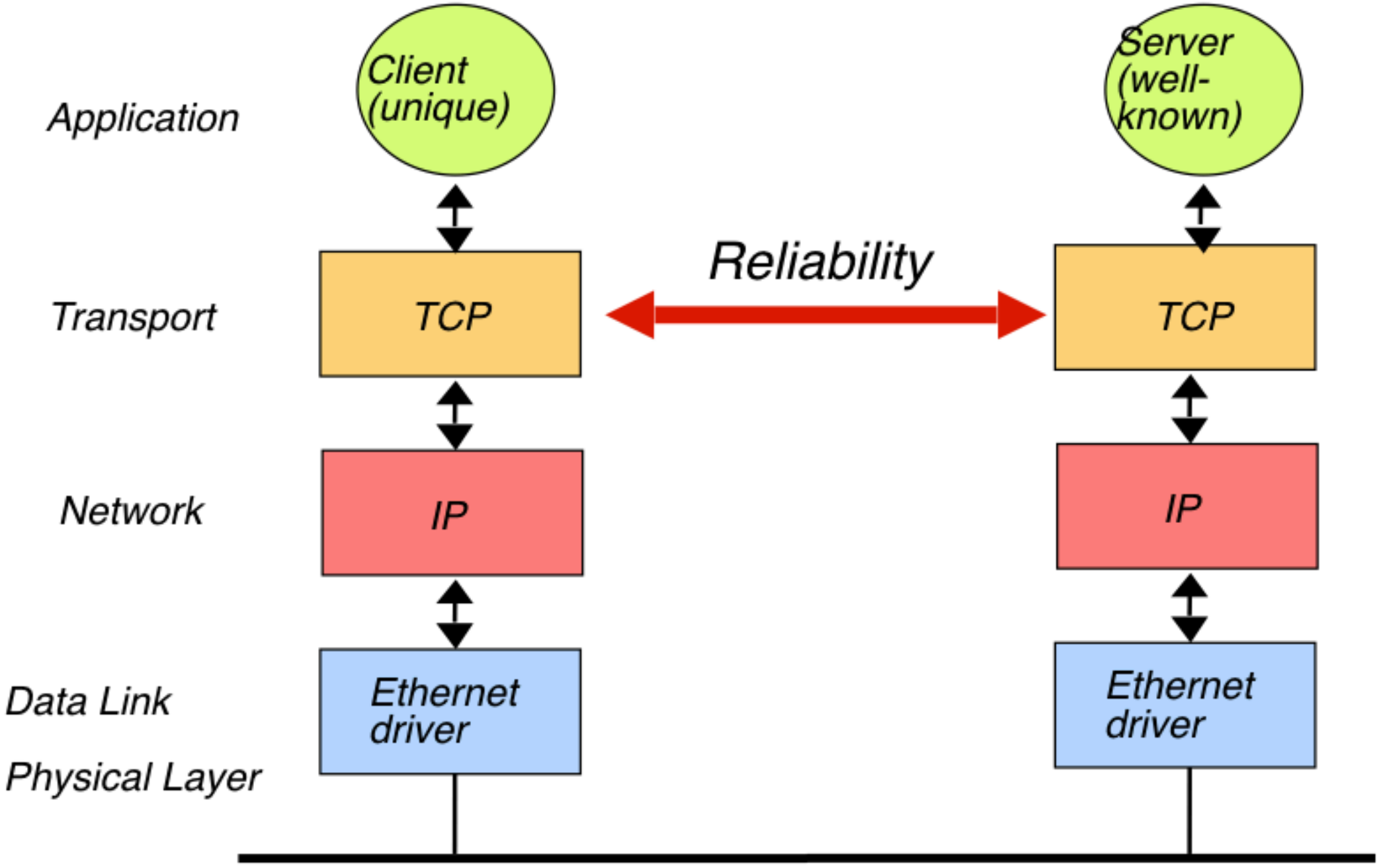
Protocol Layers

G Fairhurst, http://www.erg.abdn.ac.uk



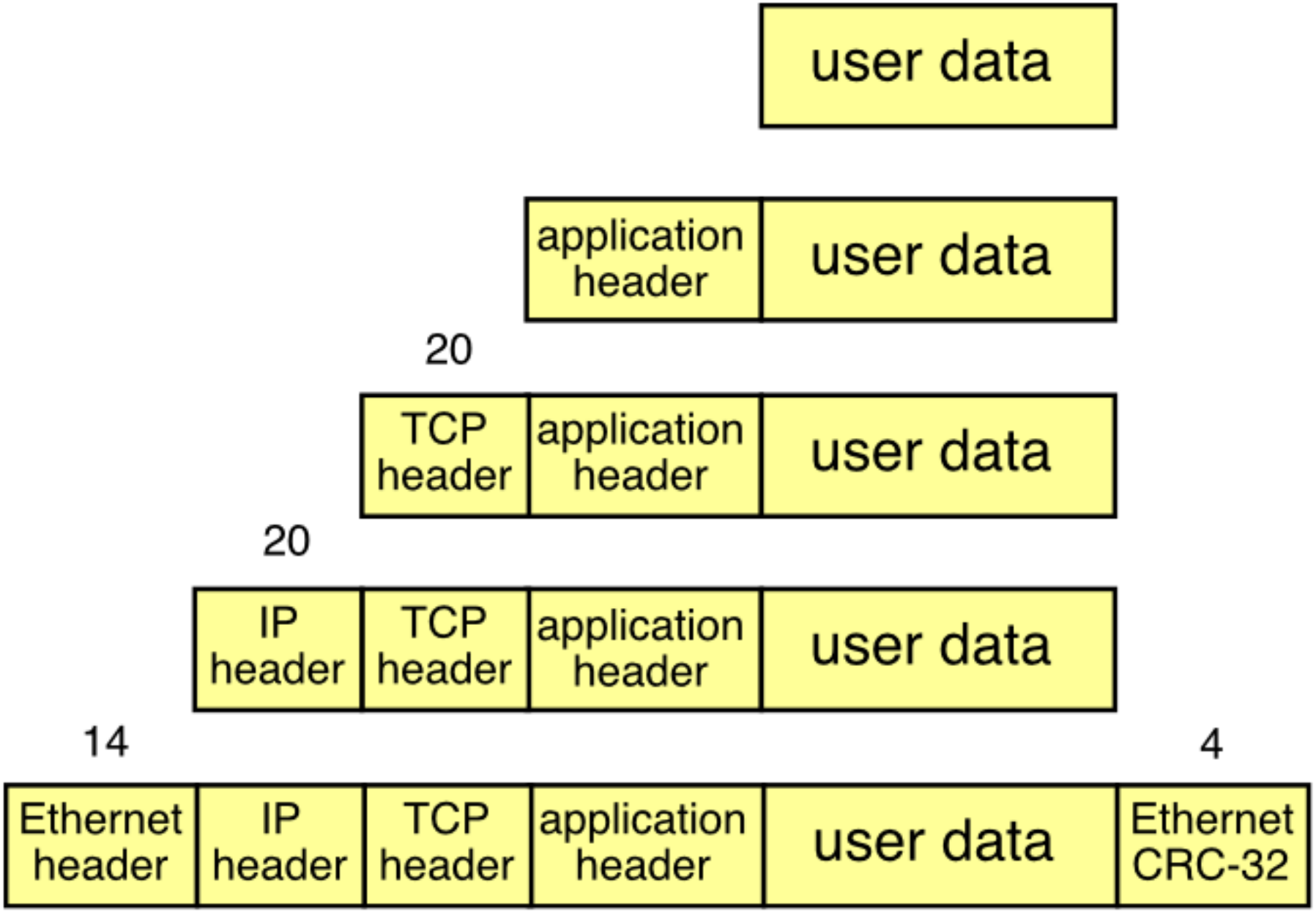
TCP between local hosts

G Fairhurst, http://www.erg.abdn.ac.uk



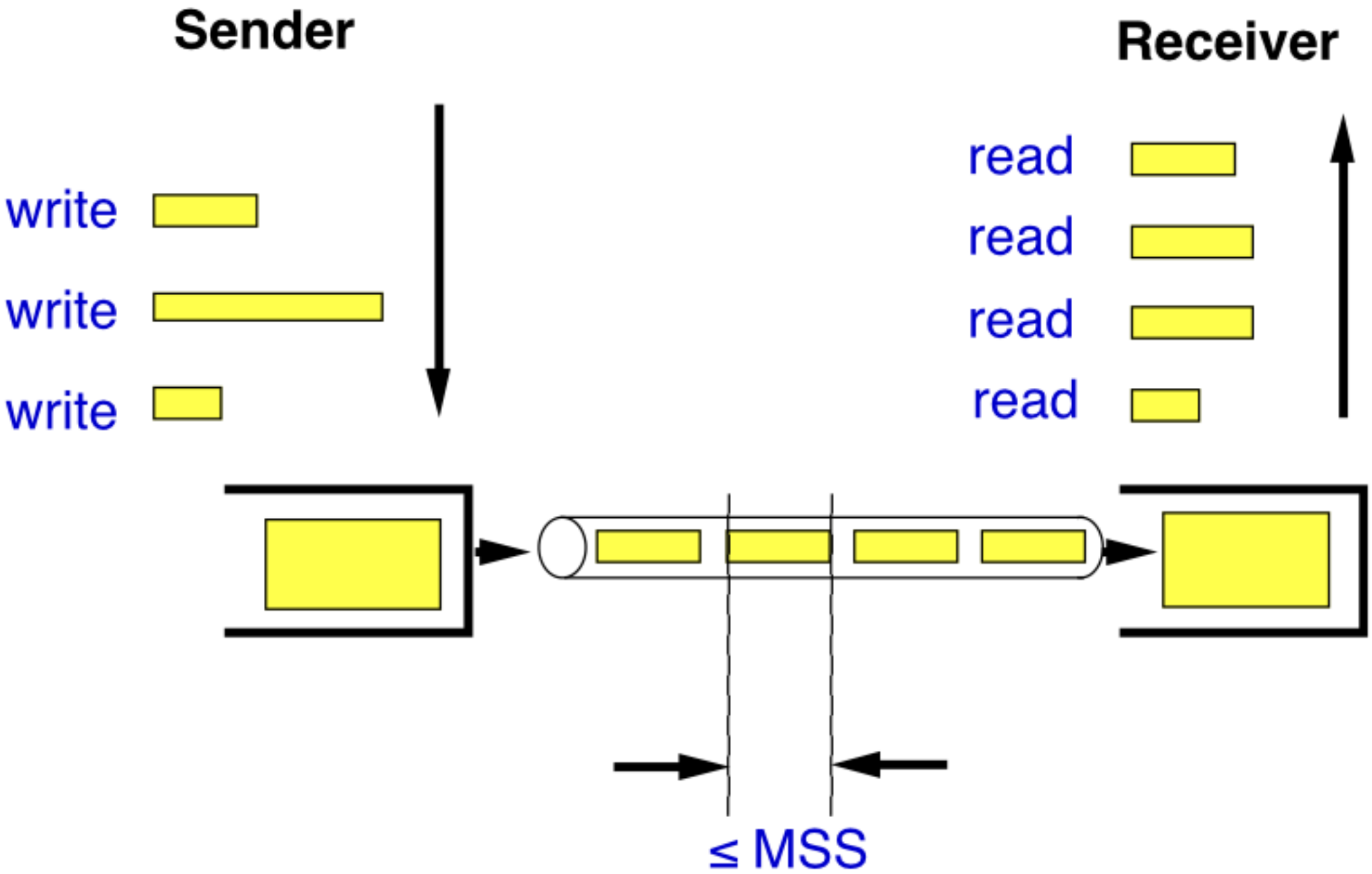
Encapsulation

G Fairhurst, http://www.erg.abdn.ac.uk



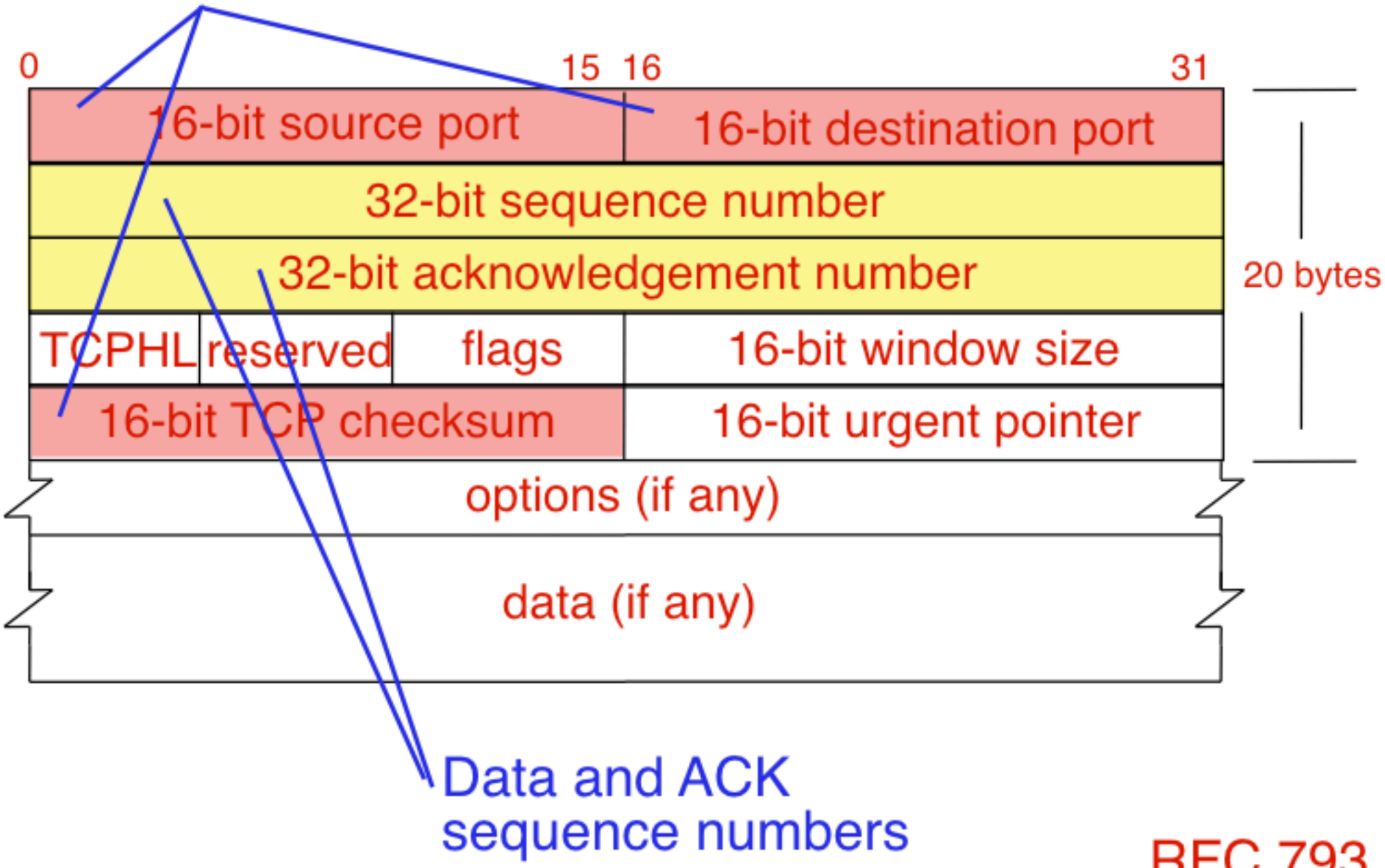
TCP Streams

G Fairhurst, http://www.erg.abdn.ac.uk



TCP Header

G Fairhurst, http://www.erg.abdn.ac.uk



RFC 793

Well Known Port Numbers

G Fairhurst, http://www.erg.abdn.ac.uk

The Internet has agreed a set of well -known ports

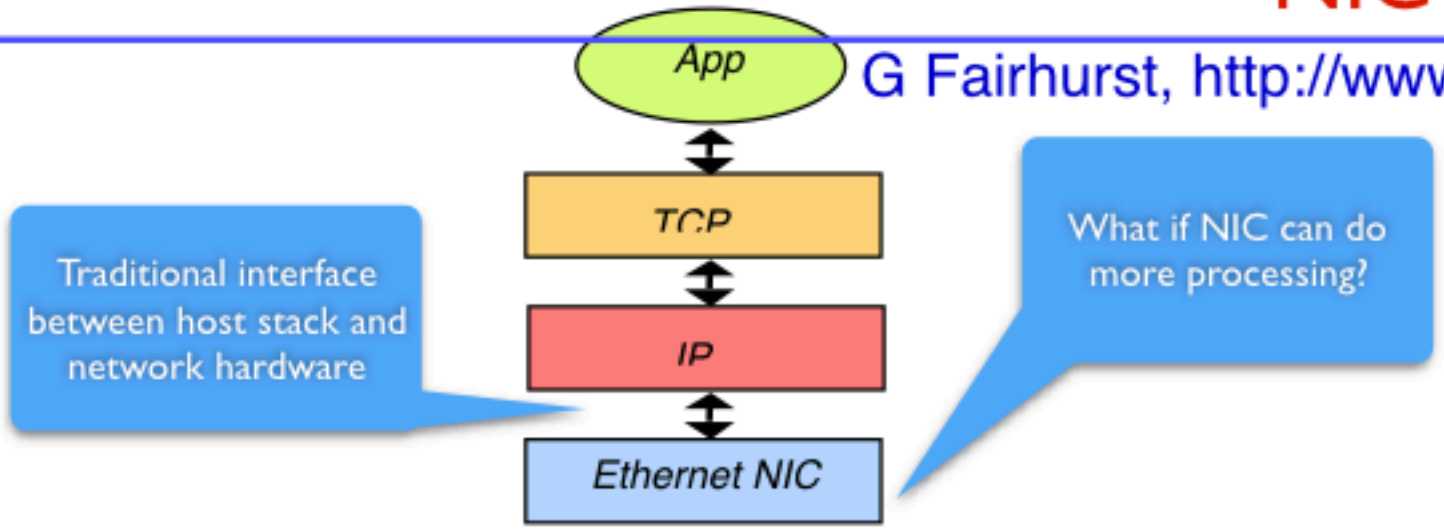
There are lots of these, see:
/etc/services file in UNIX
or RFC 1060 (Assigned Numbers)

Some examples are:

20	FTP-DATA	File Transfer [Default Data]
21	FTP	File Transfer [Control]
23	TELNET	Telnet
25	SMTP	Simple Mail Transfer
37	TIME	Time
69	TFTP	Trivial File Transfer
79	FINGER	Finger
110	POP3	Post Office Protocol v 3
123	NTP	Network Time Protocol
143	IMAP2	Interim Mail Access Prot. v2
161	SNMP	Simple Network Man. Prot.

NIC Off-loading

G Fairhurst, http://www.erg.abdn.ac.uk



In the mid 2000s hosts got smarter and NICs became able to process bursts of packet

Off-load can:

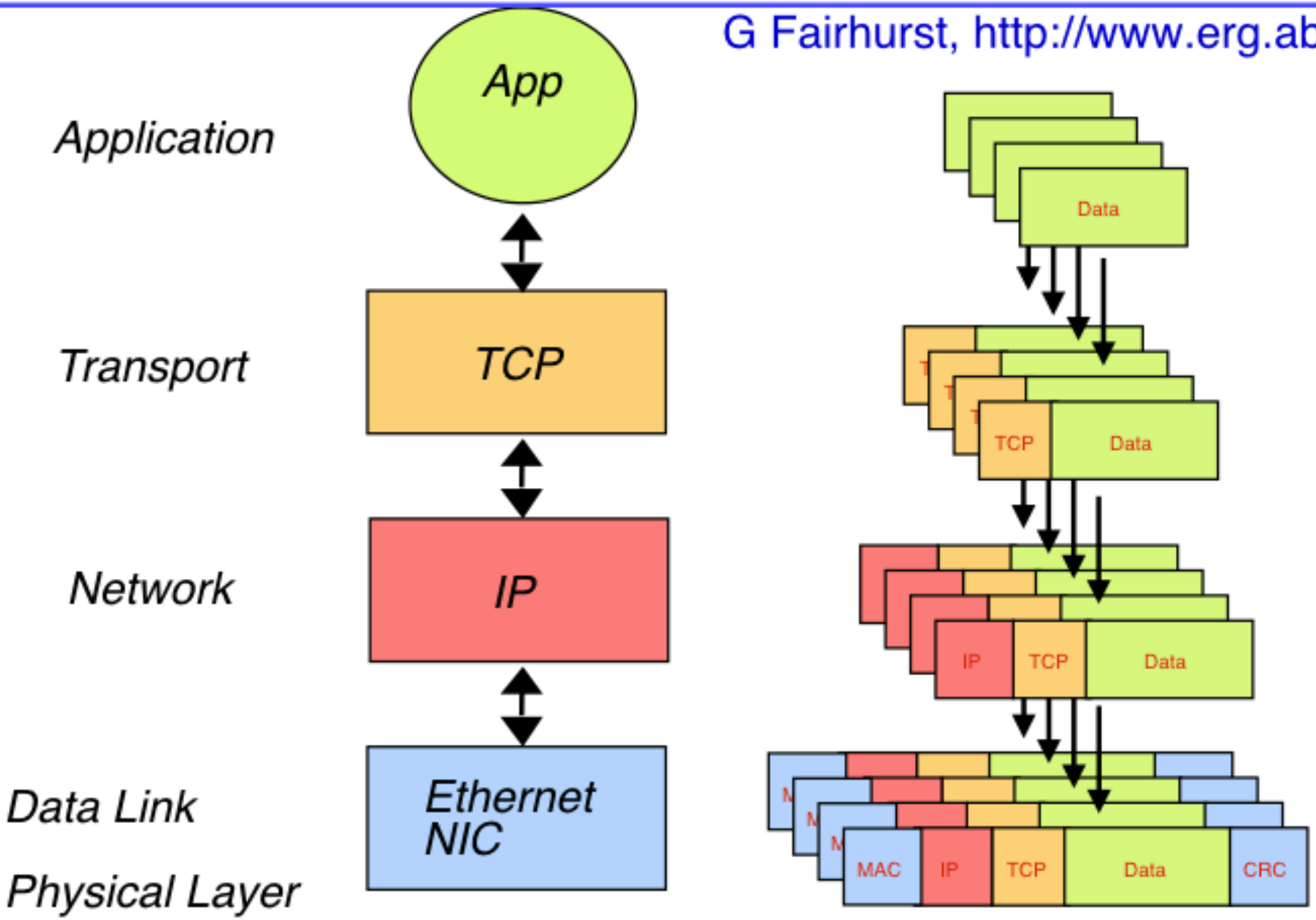
- Free the host CPU of some processing by supporting particular features of protocols, etc.
- Can enable power-saving (e.g., mobile and wake on LAN).

First forms of offload were tied to particular vendors, where new features are only available on one vendor's NIC.

Modern offload are designed to avoid vendor lock-in.

Transmission of Segments by the Host Stack

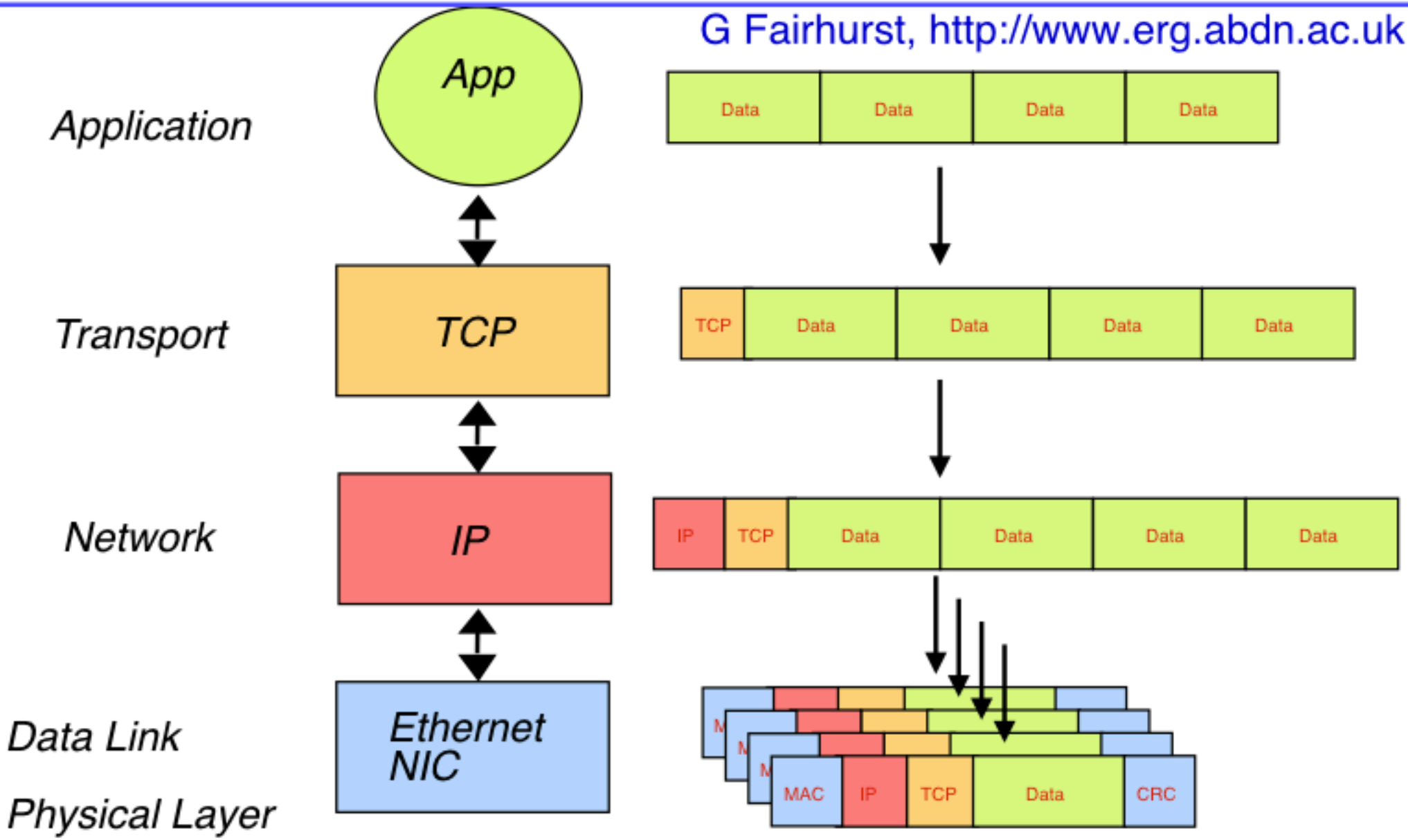
G Fairhurst, http://www.erg.abdn.ac.uk



Per-packet processing cost can be significant

Transmit Segment Offload to the NIC

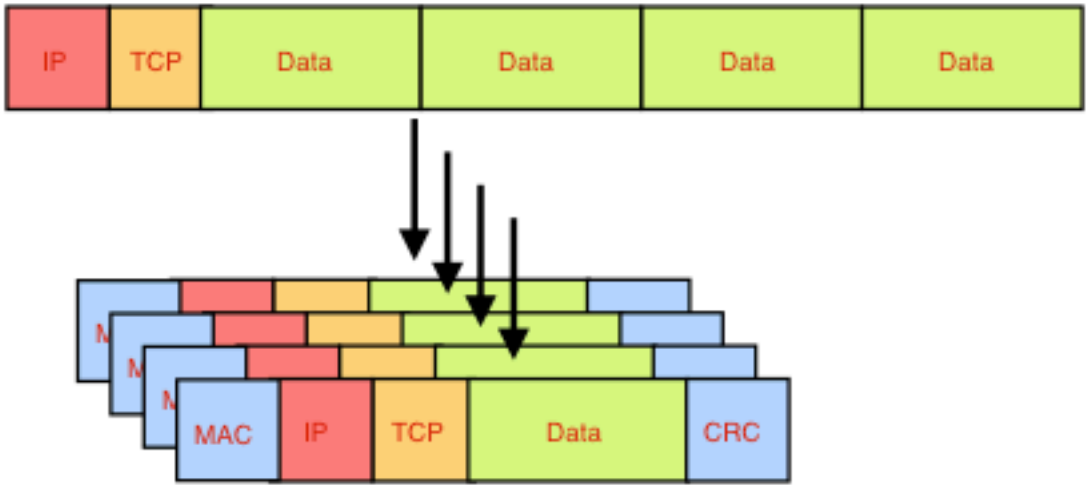
G Fairhurst, http://www.erg.abdn.ac.uk



Some functions can better be provided in hardware

Segmentation Off-loading

G Fairhurst, http://www.erg.abdn.ac.uk



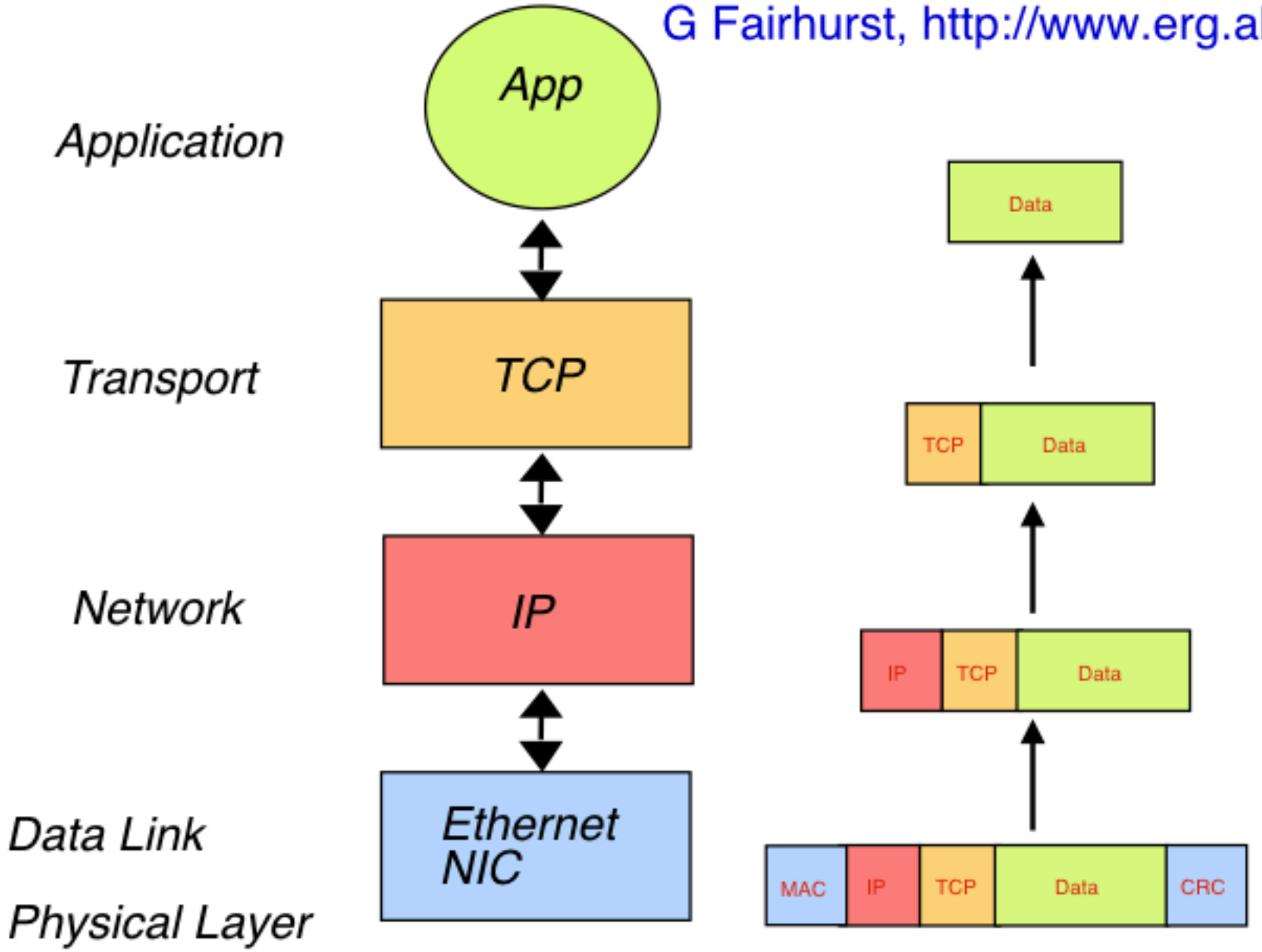
The host stack makes a large packet (e.g., 64 KB segment)

The NIC uses the MTU of the link, and segments the packet into a sequence of packets of the same size (except the last).

- Large Send Offload (LSO) uses hardware offload ***in the NIC*** (or TCP Segment Offload, TSO).
- Generic Segment Offload (GSO) provides software offload in the ***NIC driver***, but below the TCP/IP stack.

Reception of Segments by the Host Stack

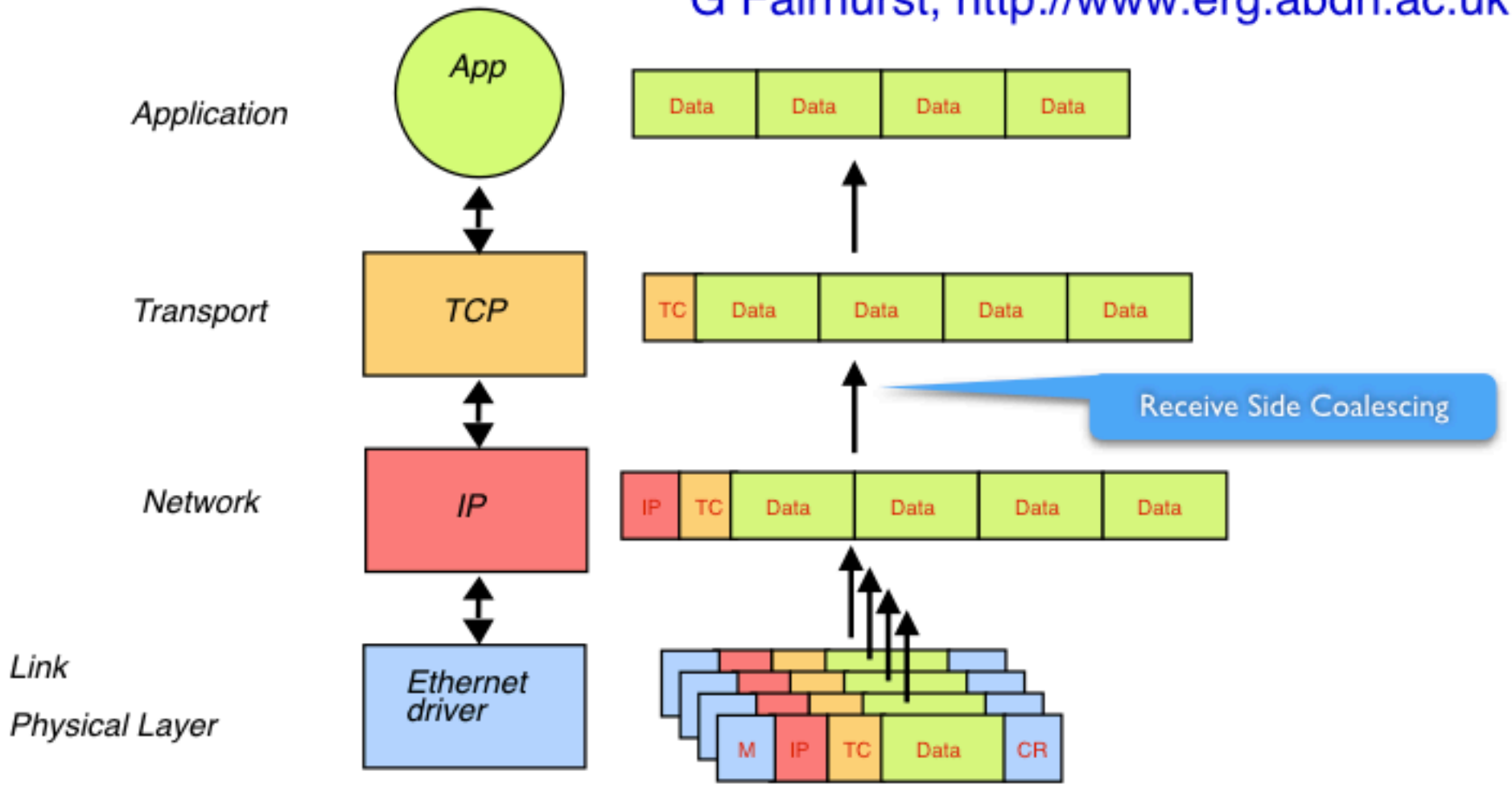
G Fairhurst, http://www.erg.abdn.ac.uk



Host stack processes each packet (interrupt for each packet/group of packets)

Receive Segment Offload by the NIC

G Fairhurst, http://www.erg.abdn.ac.uk

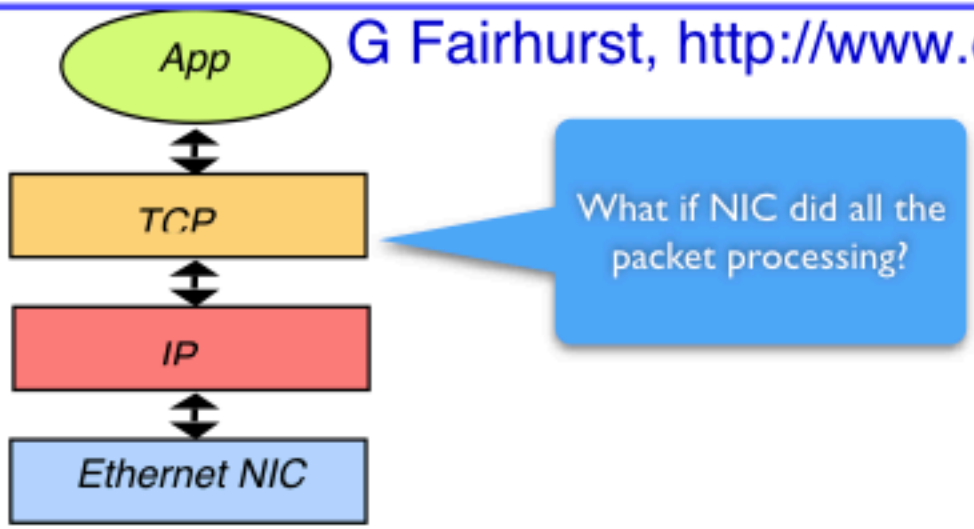


Receiver offload is where small packets of data are coalesced within the NIC before interrupting the host to process all data.

Receiver processing is much more complex, because it needs to understand how the protocol works.

Data Plane in Hardware

G Fairhurst, http://www.erg.abdn.ac.uk



Various types of reprogrammable hardware:

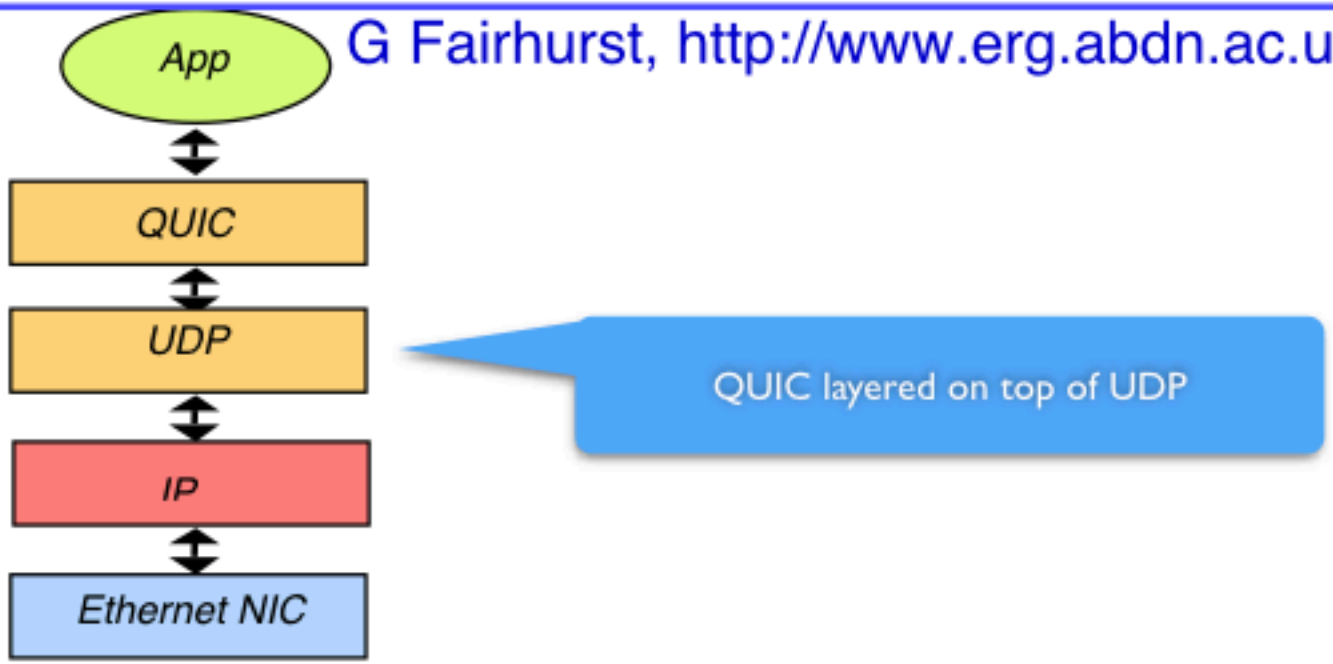
- Hardware can be configured using ASIC with TCAM tables.
- Network flow processor or network processing unit.
- FPGA, gate-level programming.
- General purpose processor, e.g. CPU on the NIC.

The host can give some of the "routine" tasks to the NIC.

More processing done in hardware reduces the cost.

QUIC

G Fairhurst, http://www.erg.abdn.ac.uk



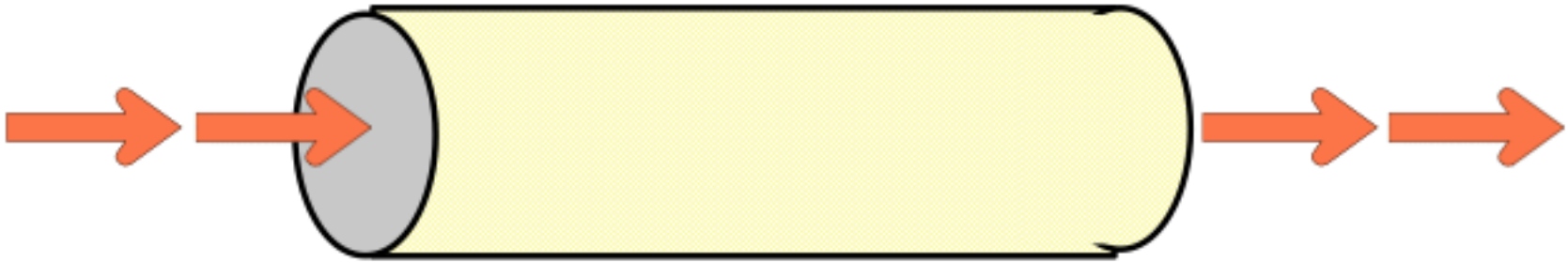
A new transport to be standardised in 2021
Works over UDP
Features like TCP - reliable, congestion-controlled, flow control
Secure and flexible
Increasingly used for web traffic and YouTube!

Throughput

G Fairhurst, http://www.erg.abdn.ac.uk

Throughput

G Fairhurst, http://www.erg.abdn.ac.uk



Defined as “the number of bits transferred per second from a given layer to the upper layer as a result of a conversation between two users of the layer”

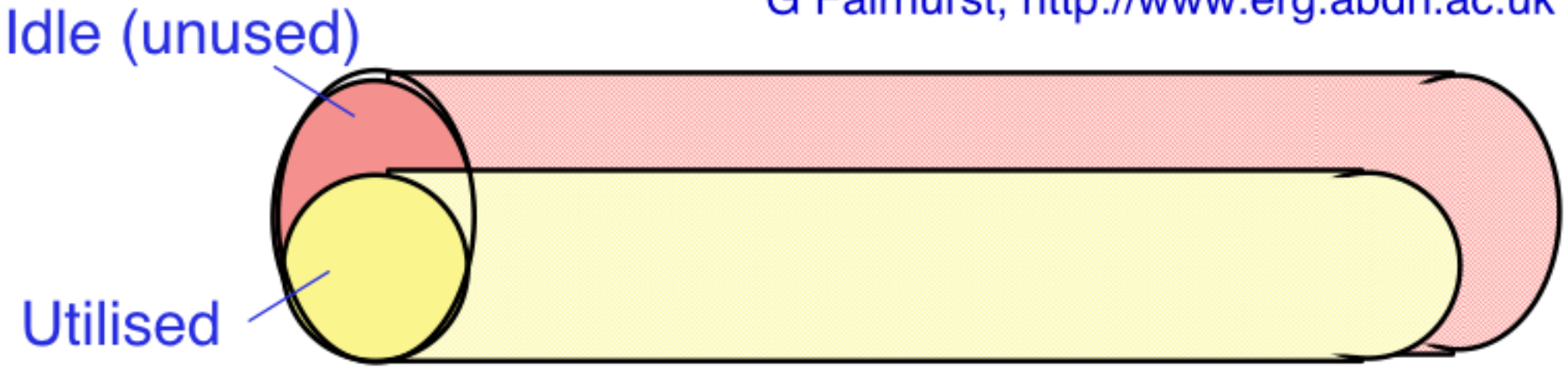
Considers only data forwarded to the OSI layer above (i.e. not layer ovehead)

Expressed in bits per second

Measures performance of a layer

Utilisation

G Fairhurst, http://www.erg.abdn.ac.uk



Defined as *"the total number of bits transferred at the physical layer to communicate a certain amount of data divided by the time taken to communicate the data."*

Includes all bits in all types of frame irrespective of whether they are corrupted or correctly received.

Expressed as a percentage of physical layer rate.

Measures link capacity used

.

And finally....

G Fairhurst, http://www.erg.abdn.ac.uk

Topics to be examined
- Everything on the syllabus

Topics excluded
- Calculation of link CRC (but know what it does!)
- Algorithm for DPLL (but know what it does!)
- Calculation of packet checksum (but know what it does!)

Topics not covered this year
- IP router fragmentation
- Path MTU Discovery
- TFTP