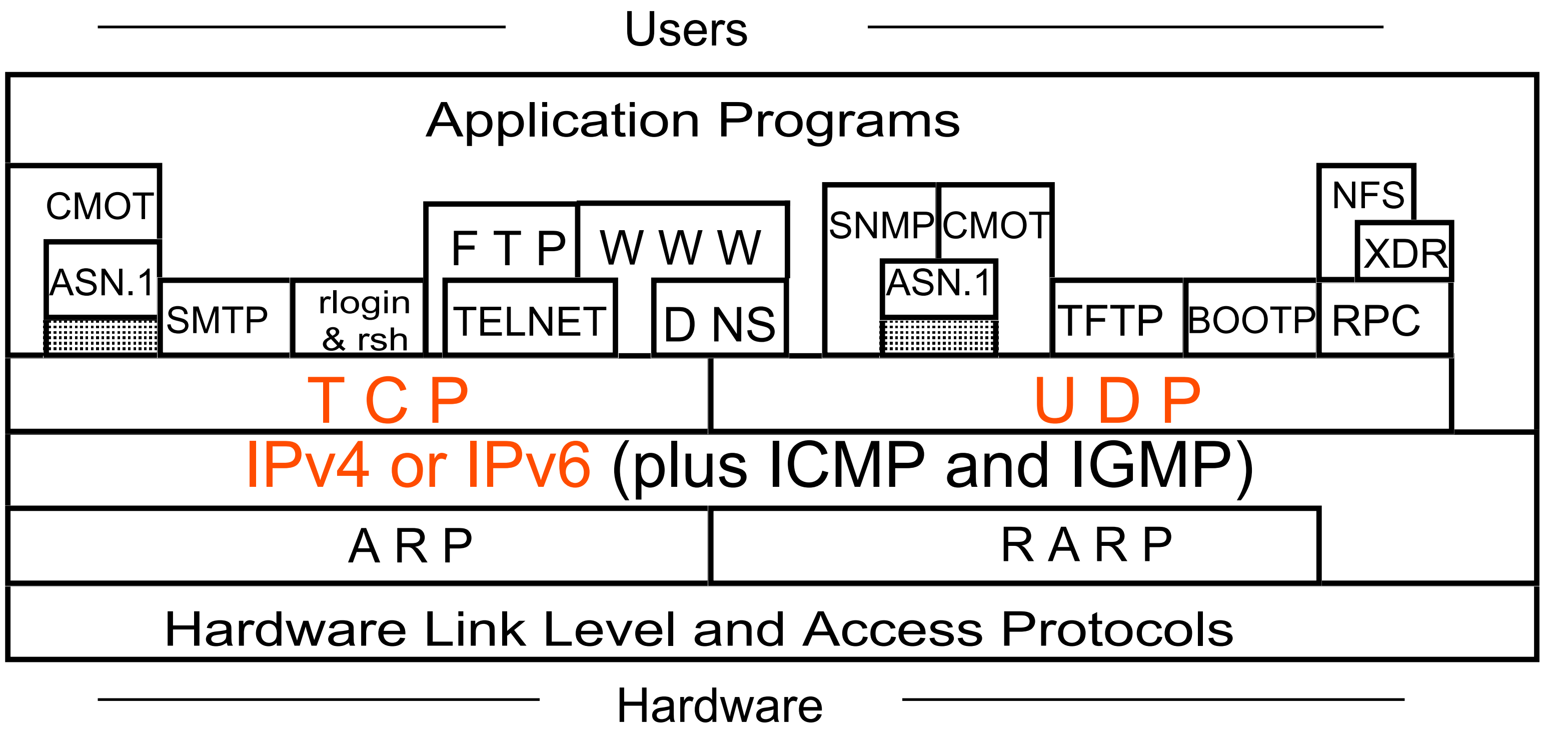
The TCP/IP Protocol suite



TELNET	Terminal emulator	FTP	File Transfer Protocol
SMTP	Simple Mail Transfer Protocol	CMOT	CMIP over TCP
DNS	Domain Name Server	SNMP	Simple Network Maintenance Protocol
TFTP	Trivial File Transfer Protocol	BOOTP	Boot Protocol
RPC	Remote Procedure Call	NFS	Network File System

Address Resolution Protocol (ARP)

Stations can come and go on an Ethernet or similar network; a problem is to find out the *physical address* corresponding the *protocol (or IP) address* of a previously unknown station (a message is received with an IP address and must be sent to a Physical MAC address) or, sometimes, to find the protocol address corresponding to a known physical address.

This is done by the host (probably a router) maintaining a cache of IP/MAC correspondences, with entries deleted if there is no traffic to the station for a while.

If there is traffic to an unknown station (unknown physical address, but known protocol address), the host must broadcast an *ARP request* with the IP address, to which the correct station replies, also with a broadcast. All stations can then update their local caches as appropriate.

The ARP/RARP message is very general, with variable-length hardware and protocol addresses.

Ethernet — ARP type = 0806; RARP type = 8035

Hardware Type (1 for Ethernet)		Protocol Type (0800 for IP)	
H'ware Addr Len	Protocol Addr Len	OPERATION	
Sender Hardware Address (octets 0–3)			
Sender Hardware Addr (octets 4–5)		Sender IP addr (octets (0–1)	
Sender IP addr (octets (2–3)		Target Hardware Addr (octets 0–1)	
	Target Hardware Addr (octets 2–5)		
Target IP Address (octets 0–3)			

Operation 1=ARP request; 2=ARP reply, 3=RARP request; 4=RARP reply

The sender's IP-to-physical address binding is included in every ARP broadcast; receivers update the IP-to-physical address binding information in their cache before processing an IP packet.

The ARP protocol is used to find the hardware address for a given protocol address.

1. The “host” broadcasts a request (opn = 1) with the desired IP address (and empty Target Hardware Address)
2. The station with that IP address replies (opn = 2) with its hardware address (the sender & target are swapped from the request).
3. All stations on the network may update their own caches according to the sender addresses.

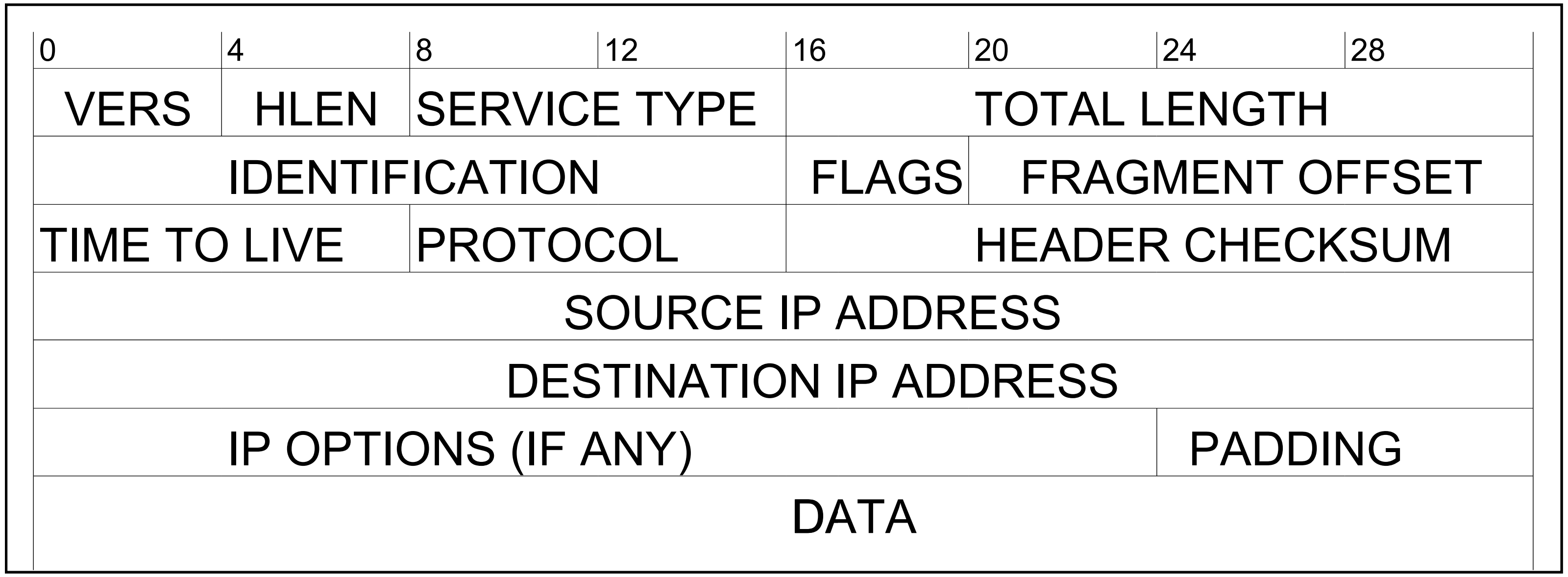
The RARP protocol (Reverse Address Recognition Protocol) is used when a station wishes to find its own IP address (the central server already knowing the hardware–IP address correspondence.

The protocol is very similar to ARP, using the same message format, but different message types.

1. The station broadcasts an RARP message (opn=3) with its own MAC (hardware) address.
2. One or more servers reply with the appropriate “address binding” in an RARP (opn=4) message.

Internet Protocol (IPv4)

The IPv4 datagram is used for all data transfers over the Internet (except those using the newer IP version 6)



• VERS	IP version (=4)
• HLEN	Header length, in 32-bit words
• TOTAL LENGTH	Packet length, in octets (max 65,535)

• PROTOCOL	protocol type 6=TCP, 17=UDP, 1=ICMP, etc.
• TYPE OF SERVICE	Allows priority, low delay, high throughput, high reliability, if possible (and if routers recognise it!)
• CHECKSUM	– (16-bit 1s complement sum of IP header)
• TIME TO LIVE	Initially set to maximum packet life in seconds, it is decremented by 1 at each hop, and by 1 for each full second the packet is held in a node
• ADDRESSES	The IP addresses (32 bits each) of the source and destination nodes.
• IDENTIFICATION, FLAGS, FRAGMENT OFFSET	These are used in message fragmentation.
• IP OPTIONS	° record route; routers put their addresses in packet ° timestamp; routers add timestamp to addresses ° source route; source can supply router address list

IP Fragmentation (IPv4 only)

- All communication links can handle messages up to some maximum transmission length (Maximum Transmission Unit, or MTU).
- IP packets are sent with no knowledge of the links within the path, or of their MTUs (although there are ways of finding the MTU along a path).
- Sometimes (often?) packets must be split into two or more units or fragments to fit within the MTU of a link.
- Any fragments *always* travel independently until they are reassembled at the final destination.
- When a packet is fragmented, its original IDENTIFICATION field is copied into all of the generated fragments. (This allows all related fragments to be identified for reassembly.)

Fragmentation – principles

- A packet of total length 4000 octets is to be sent over an Ethernet link with MTU of 1500 octets.
- With an IP header of 20 octets, the MTU has 1480 octets of data, which is a multiple of 8 octets.
- To support fragmentation, each packet has the fields —

LENGTH	the length of this fragment
OFFSET	the position of this fragment from start of packet (in units of 8 octets, so most lengths are a multiple of 8)
IDENTIFICATION	an identifier to identify the original packet
more fragments bit	= 1 if not last fragment; = 0 if last of packet
do not fragment bit	if = 1, then this packet will not be fragmented

- Note that "Length" is data length + 20 octets (for the IP header).

Handling of “More Fragments” (MF) bit

- Every packet is initially generated with MF = 0
- When a packet is fragmented (or a fragment is further fragmented) —
  - its *last fragment* retains the *incoming* MF bit
  - all *earlier fragments* of the packet have MF set to 1

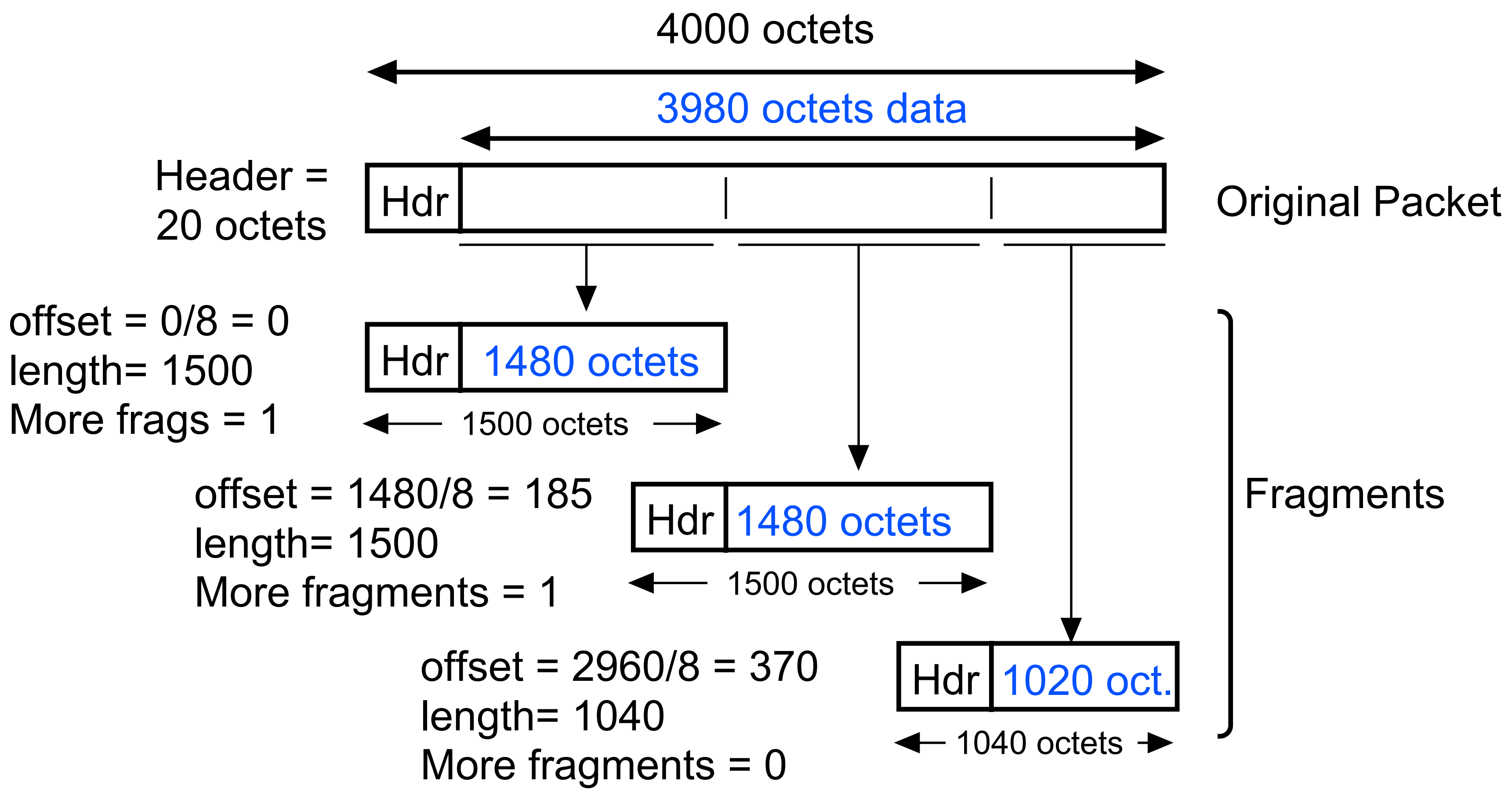
The result is that the very last fragment of the original packet retains MF=0, and all preceding ones get MF=1

Identification field

This is a unique number given to each IP packet when it is first generated. When the IP packet is fragmented, the IDENTIFICATION is is copied into all fragments so that they can be identified for reassembly

Example of Fragmentation

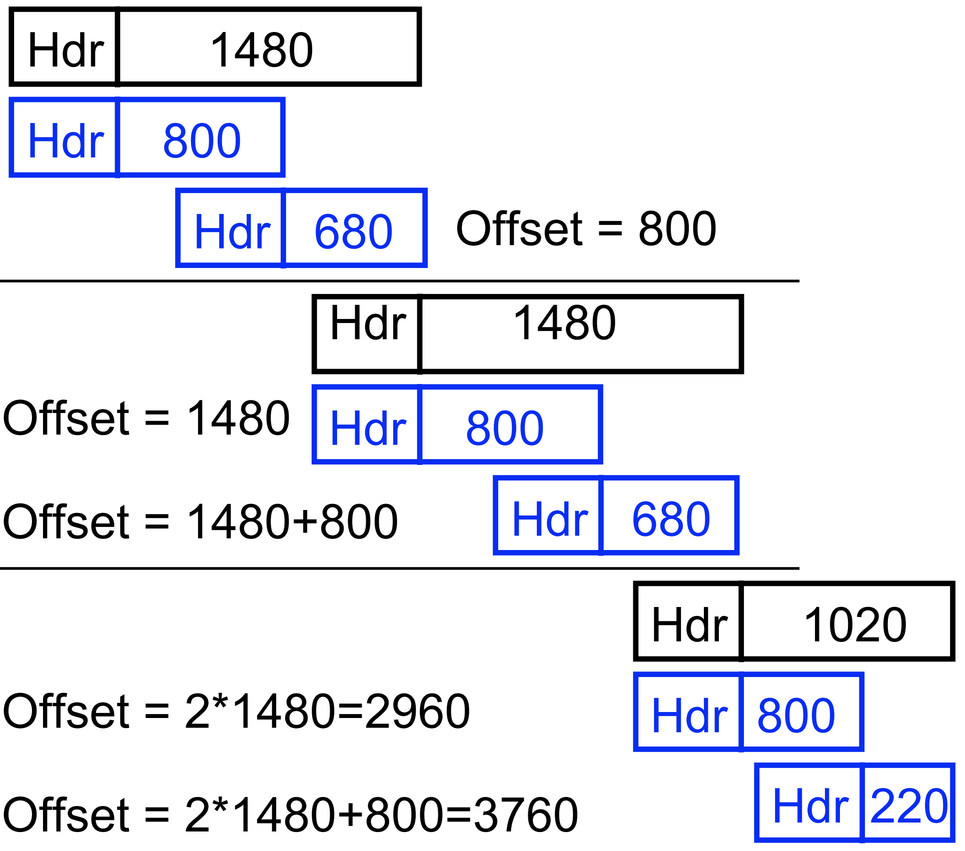
MTU = 1500 (1480 octets of data + 20 octets IP Header).



Multiple Fragmentation

These fragments are sent over a link MTU=820 (20 header+800 data)

offset	length	mfb
0/8 = 0	1500	1
0/8 = 0	820	1
800/8 = 100	700	1
1480/8 = 185	1500	1
1480/8 = 185	820	1
2280/8 = 285	700	1
2960/8 = 370	1040	0
2960/8 = 370	820	1
3760/8 = 470	240	0



Fragmentation Control

- The "*do not fragment*" bit (DF) in the IP header prevents fragmentation as the packet traverses the network.
- If a packet with DF=1 would require fragmentation, it is discarded and an ICMP message "UNREACHABLE DESTINATION – FRAGMENTATION NEEDED" is returned to the sender.
- This allows the sender to probe a route to find the minimum MTU by sending messages with DF=1 and finding which ones give errors.
- Some very simple protocols such as TFTP (Trivial File Transfer Protocol) cannot reassemble fragmented IP packets.
- As the offset is in units of 8 bytes, all fragments except the last have a payload length which is a multiple of 8.
  (Any odd lengths are accommodated the last fragment; its *start* or offset is a multiple of 8, its *length* is not restricted.)

IP Reassembly (“defragmentation”)

These actions are performed for each received packet

- check the packet *identification* to select the *assembly buffer*
- use the *offset**8 to place the fragment correctly in the assembly buffer
- get the *data_length* = *total_length* – 4**header_length*
- if this is the first copy of this fragment, add the fragment *data_length* into the total *received_length* for the packet
- if *mfb=0*, calculate *original_length* from *offset**8+*data_length*
- if *original_length* = *received_length*, then packet is complete

(The offset must be multiplied by 8 because it is in units of 8 octets; similarly the header length is in units of 4 octets and is multiplied by 4.)

ICMP

ICMP (Internet Control Message Protocol) is used to send reports from one node to another.

- ICMP messages are included in IP datagrams, with Protocol=1.

Type	ICMP Message		Type	ICMP Message
0	Echo Reply		12	Datagram Parameter Error
3	Destination Unreachable		13	Timestamp Request
4	Source Quench		14	Timestamp Reply
5	Redirect			
8	Echo Request		17	Address Mask Request
11	Datagram Time exceeded		18	Address Mask Reply

- As IP checksums only the header, the ICMP message includes a checksum.

ICMP examples

- Echo Request or Reply (PING to remote station)

0	4	8	12	16	20	24	28
TYPE (8 or 0)		CODE (0)		CHECKSUM			
IDENTIFICATION				SEQUENCE NUMBER			
OPTIONAL DATA							

- Unreachable Destination (routing failure)

0	4	8	12	16	20	24	28
TYPE (3)		CODE (0–5)		CHECKSUM			
UNUSED – MUST BE ZERO							
INTERNET HEADER + FIRST 64 BITS OF DATAGRAM							

This case includes “fragmentation needed”

- Source Quench (packet discarded, or congestion imminent)

0	4	8	12	16	20	24	28
TYPE (4)		CODE (0)		CHECKSUM			
UNUSED – MUST BE ZERO							
INTERNET HEADER + FIRST 64 BITS OF DATAGRAM							

Network Byte Order

Computers have many interesting and different conventions for the ordering of bits in bytes, bytes in words, and words in messages.

To give a uniform interpretation within the network, the Internet Standards specify a *Network Byte Order* for all fields interpreted by the network.

- Integers within the network headers must be sent with the bytes in decreasing numerical significance, most-significant byte first. (The most-significant byte is nearest the start of the packet.)

This means that bytes or octets are transmitted in “raster-scan” order, left to right, top to bottom, as we look at the diagrams.

User data may be converted to Network Byte Order at the choice of the user.

IPv6 (IP Version 6)

IP Version 6 addresses several problems with the older IP (version 4), especially —

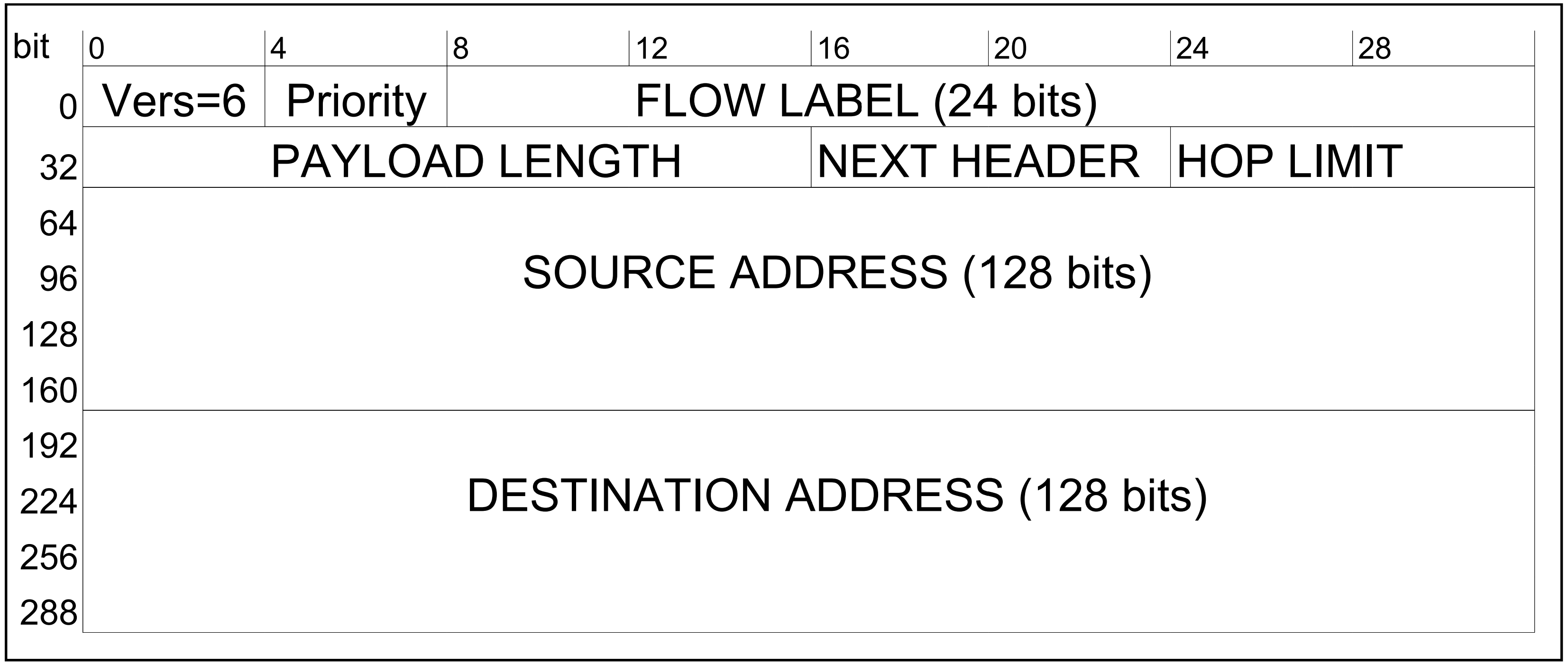
- Overflow of the 32-bit address field
- Fragmentation is difficult and expensive for routers
- The IPv4 header is too complex — many aspects can be removed to specialised headers

Special term

IPv6 defines a *flow* as a sequence of related packets from a source station to a destination, something like a virtual circuit.

Do not confuse an IPv6 flow with "flow control".

The IPv6 Header



- The introduction is 64 bits long, and each address is two lots of 64 bits for better efficiency in fast processors.
- The NEXT HEADER identifies either the header OR (on the last header) the packet protocol (e.g. TCP).

Priority

Priorities 0 – 7 have the source providing congestion control (backoff)

0	no priority
1	background traffic, such as netnews
2	unattended transfer (email , WWW)
3	— reserved
4	bulk transfer (FTP)
5	— reserved
6	interactive (telnet)
7	control traffic (routing protocols, SNMP)

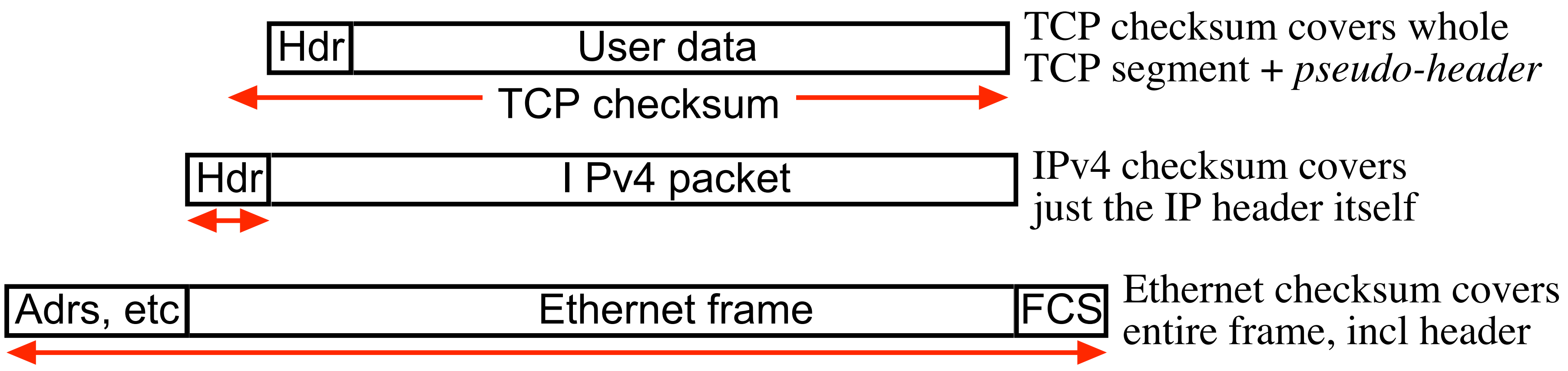
Real-time traffic which does not backoff for congestion

8	most willing to be discarded
15	least willing to be discarded

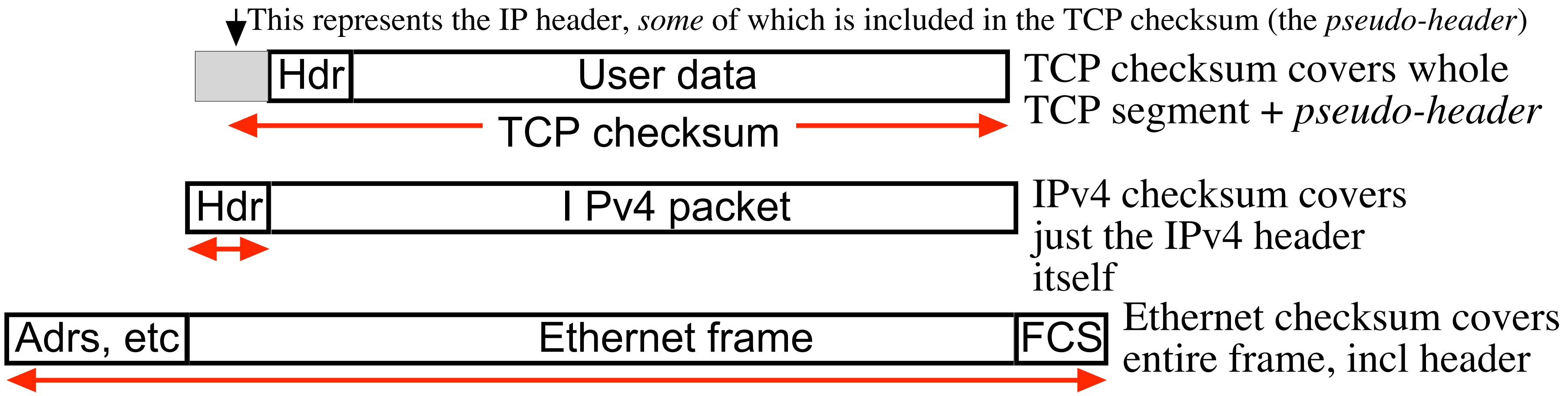
- Hop Limit is set to the maximum number of hops and is decremented by each switch. The packet is discarded if its hop limit becomes zero.
  It is like the IPv4 "TIME TO LIVE" (as it eventually developed, NOT as originally defined), but is not decremented by time.
- Flow label is a unique identifier allocated by the source to a series of related packets. It allows packets to be treated similarly by routers, such as following the same paths. (The flow label and source address may be used to retrieve cached route information.)
  The flow label usually expires if not used for a while.
- Header labels extend the idea of *protocol labels*. Each header (including the IPv6 header) indicates the nature of the next header, or the payload protocol if there are no more optional headers.
- Checksums are assumed to be handled by the underlying datalink layer and are omitted from the IPv6 header.

Notes on Checksums

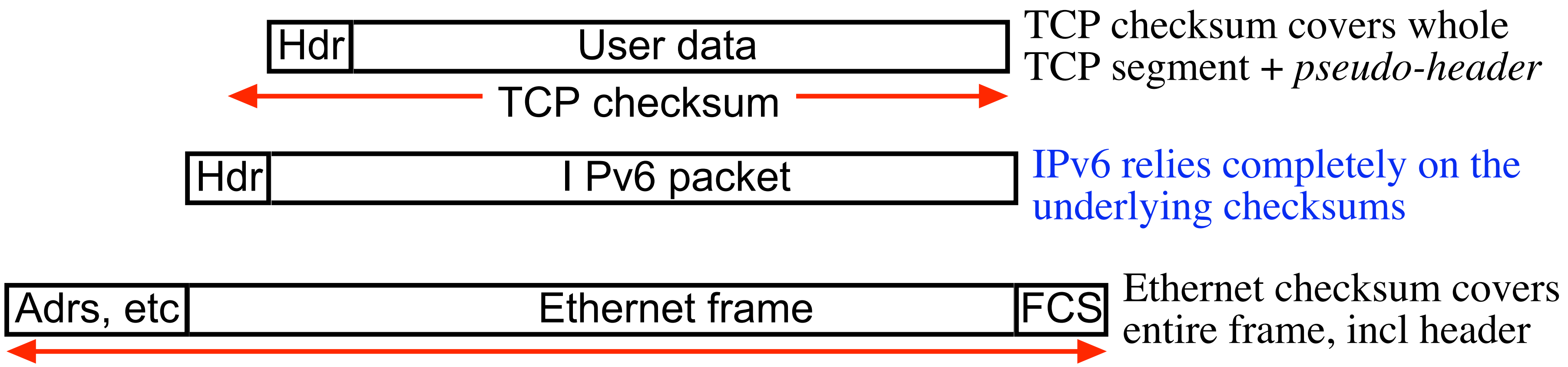
TCP and IP were originally designed for unreliable links with poor error checking. They therefore have multiple levels of overlapping checksums. With Ethernet we get that


Hdr
User data
TCP checksum
TCP checksum covers whole TCP segment + *pseudo-header*
Hdr
I Pv4 packet
IPv4 checksum covers just the IP header itself
Adrs, etc
Ethernet frame
FCS
Ethernet checksum covers entire frame, incl header

- TCP check covers whole of user data + TCP header for a complete end-to-end check, including source and destination addresses.
- IPv4 check covers just the IP header to minimise switching overheads.
- Ethernet check is very strong and covers all of each IP packet.


This represents the IP header, *some* of which is included in the TCP checksum (the *pseudo-header*)
Hdr
User data
TCP checksum
TCP checksum covers whole TCP segment + *pseudo-header*
Hdr
I Pv4 packet
IPv4 checksum covers just the IPv4 header itself
Adrs, etc
Ethernet frame
FCS
Ethernet checksum covers entire frame, incl header

- As the underlying Ethernet (or similar) checking is so good, it is most unlikely that the IP checksum will ever be presented with an error. Accordingly, IPv6 has no IP header checksum.


Hdr
User data
TCP checksum
TCP checksum covers whole TCP segment + *pseudo-header*
Hdr
I Pv6 packet
IPv6 relies completely on the underlying checksums
Adrs, etc
Ethernet frame
FCS
Ethernet checksum covers entire frame, incl header

Extension Headers

- IPv6 extracts some information from the IPv4 header into "extension headers" which follow the main IPv6 header and have defined formats.
- The IPv6 header has a "next header" field which gives the type of the first header, and each header itself has a "next header" field.

Header types are —

- Authentication
- Destination options
- Fragmentation
- Hop-by-hop
- Routing
- Security

IPv6 Fragmentation

- IPv6 routers and switches do not fragment packets *en route*; the sender is supposed to ensure that the packet fits within the MTU for the path.
- If a user packet is too long, the *sender* must fragment it and create appropriate smaller packets, each with an appropriate fragment header to allow reassembly of the user packet.
- Thus fragmentation is entirely the responsibility of the sender

More Information on IPv6

The “Resources” section in the 314 Web page has some foils from a seminar given here in 2000 by one of the leaders in designing IPv6. Some of it may interest you, but most is far beyond what you will need.

It includes information on “tunnelling” IPv6 over IPv4 (and vice versa) so that where an IPv6 link does not exist, an IPv6 packet can be encapsulated in an IPv4 packet and then extracted and sent on as IPv6.

It also includes the address assignments, which is quite complex. We certainly will not use all $2^{128} = 3.4\times10^{38}$ addresses, which is 6.68×10^{23} per square metre of the Earth’s surface, or about 1,000 per surface atom!

The full URL is

http://www.cs.auckland.ac.nz/compsci314s1c/resources/BobFink-IPv6.pdf

RFCs (Requests for Comment)

The Internet is administered by the Internet Activities Board (IAB) with most technical work handled by the Internet Engineering Task Force (IETF).

Technical specifications by IETF are in contained in RFCs. They start as "Internet Drafts" and are progressively refined and tested until they stabilise as full standards. The initial RFC number is retained, but the RFC status is held in the document header. All RFCs are plain-text documents, but may be compressed (GZIP format), or sometimes in PostScript.

See Shay pages 568–569 for RFC information. Many RFCs are held locally at <`ftp://ftp.auckland.ac.nz/pub/rfc`>.
There are now over 2700 RFCs and even finding one if you don't know the number can be difficult.

Addressing in Internet Protocol (IP)

- The Internet sends all packets using the Internet Protocol, or as “IP”.
- The Internet uses two forms of addressing —

1. IP routing uses a *network* 32 bit or 128 bit address, such as 130.216.34.63.
2. Users use a *logical* “name” address – `ruru@cs.auckland.ac.nz`

IPv4 Network Addressing

- Traditional IP, “version 4”, or “IPv4” uses a 32-bit network address. (IPv6 uses 128 bit addresses.)
- The network IP address is normally written in the form 130.216.34.63, called “dotted decimal”. Each decimal number is the decimal value of 8 bits of the 32-bit address. Thus

  `130.216.34.63 = 1000 0010.1101 1000.0010 0010.0011 1111`
- The 32 bits of an IPv4 network address are divided into
  - a network number (allocated by a central authority), and
  - a local identifier (allocated by a local administrator)
  - the local identifier is often further divided, see later
- There are 5 defined classes of IPv4 address

IP address classes

Class A	0*nnnnnnn*	*xxxxxxxx*	*xxxxxxxx*	*xxxxxxxx*	$2^7 = 128$	$2^{24}=16,777,216$
Class B	10*nnnnnn*	*nnnnnnnn*	*xxxxxxxx*	*xxxxxxxx*	$2^{14} = 16,384$	$2^{16}=65,536$
Class C	110*nnnnn*	*nnnnnnnn*	*nnnnnnnn*	*xxxxxxxx*	$2^{21} = 2,097,152$	$2^8=256$
Class D	1110	a 28 bit multicast address, for groups of stations				
Class E	1111	reserved for future use				

Most networks are now class B;
Auckland University is 130.216 = 1000 0010 1101 1000

Subnets

- As the 65,536 local identifiers are too many for most organisations to easily administer, the identifier is split into two components, a sub-net and a user.
- Often use an 8-bit subnet and an 8-bit user. For example Computer Science uses subnet numbers like 32, 33, 34, 35, 36.
- As the division between subnet and user is local and arbitrary, each site also defines a *subnet mask*, such as 255.255.255.0. ANDing the mask against the address yields the net and subnet number; ANDing with the complement of the mask gives the user.
- Some users may choose a mask of 255.255.254.0 (for 128 subnets and 512 users each) or 255.255.255.192 (for 1,024 subnets and 64 users each), or any other division of lengths. (OR write addresses as *x.x.x*/24 or *x.x.x*/23, with the /*y* giving the length of the network part.)
- A subnet mask such as 255.255.7.0 is quite legal but strongly discouraged.

Routing

Internet physical routing uses a hierarchical system. We discuss it here in concept, but real networks may be rather different

- Each subnet has a router which learns, or is told, the physical addresses of all stations on its subnet. (With IP most nodes will use ARP (*Address Resolution Protocol*) to find say the physical 802.3 address corresponding to an IP address.) Usually the router handles all traffic between the subnet and the outside world.
- The local network itself has a router to handle connections into the larger network. (It will have to know routes to about 65,000 networks, which is probably not that many with today's computers.)
- The routers themselves use special protocols to exchange routing information, not discussed at this level.
- Just to confuse things, long-distance routing uses different addressing again.

Name Resolution.

- Few users are interested in the IP or similar addresses, but use a logical name structure, such as `ruru@cs.auckland.ac.nz`.
- Here, the `cs`, `auckland`, `ac` and `nz` are successively higher domain names, allocated by successively higher administration authorities.
- Thus New Zealand administrators (domain `nz`) allocated an “academic” domain (`ac`); its administrators allocated a domain (`auckland`) to the University of Auckland and it allocated the name `cs` to the Department of Computer Science.
- These “user-friendly” names must be converted into the network IP addresses for communication.

Name Server Hierarchy

The Name Servers are in a hierarchy —

- Local Name Servers, close to each user or subnet, this resolves local addresses immediately, or forwards requests to a higher level.
- Authoritative Name Servers contain the mappings for administering authorities or domains.
- Root Name Servers, about a dozen all told (world-wide), know how to reach the authoritative servers.

Addresses for a typical computer

Ethernet address	`08 00 07 7E 1C 84`
IP address	`130.216.33.12`
	`130.216.33.12/24`
subnet mask	`255.255.255.0`
router	`130.216.33.253`
name server	`130.216.35.100`