

IPv4: Internet Protocol version 4

- Concept
- Addressing
- Packet format
- Fragmentation
- Control messages (ICMP)
- Getting an address (DHCP)
- Finding neighbours (ARP)
- Naming things (DNS)

Concept of a connectionless datagram (1)

- The idea goes back to 1962, and the current version of IP was defined in the late 1970s
- Share expensive links by mixing variable-length packets sent between logical addresses
 - Much more dynamic than hardware multiplexing or circuit switching
 - As we've seen, allows a variety of routing mechanisms

Concept of a connectionless datagram (2)

- *Share expensive links by mixing variable-length packets sent between logical addresses*
 - Advantages: sharing costs, universal connectivity, great flexibility
 - Disadvantages: variable response time, risk of congestion or packet loss
- The success of the Internet shows that the advantages far outweigh the disadvantages

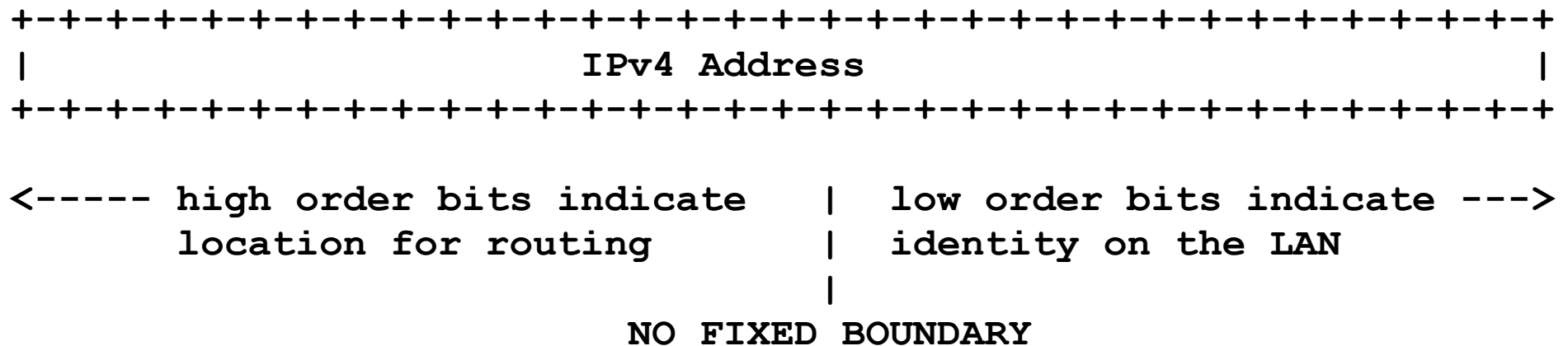
Logical addressing

- The source and destination addresses of IP packets are **logical**, not physical
 - Assigned by software
 - and can be changed
 - Assigned to **interfaces** (not whole computers)
 - Must be unique, for routing to be possible
 - Must be related to topology, for routing to *scale*
 - Are also used as unique identifiers, as we'll see later
 - One interface can have multiple addresses (rare in IPv4)

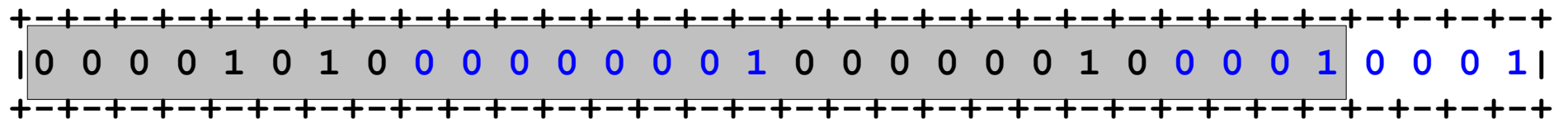
IPv4 Address Format

- In the abstract, it's just a 32 bit binary number:
`01010011 11001010 10010110 00000010`
- Conventionally written in “dotted decimal:”
`83.202.150.2`
- Upper layers of software have no business treating addresses as anything but meaningless bit strings
- But to the routing system, addresses have some real meaning

Location versus Identity



- For example, in 10.1.2.17, you cannot assume that the network is 10.1.2.0/24
 - i.e. a subnet with 256 addresses
- It might equally well be, e.g., 10.1.2.16/28
 - i.e. a subnet with 16 addresses



Old-fashioned IPv4 addressing (1)

- In the early years of IPv4 (up to about 1993), addresses were divided into three *classes*
 - Class A, user site was given a /8 prefix and had 24 bits free to assign locally (16M addresses)
 - Class B, /16 prefix with 16 local bits (65k addresses)
 - Class C, /24 prefix with 8 local bits (256 addresses)
- This was scrapped because it led to inefficient use of address space and to sparse routing tables

Old-fashioned IPv4 addressing (2)

- Addresses are now assigned in very large blocks to ISPs and sub-divided among their customers
 - CIDR (classless inter-domain routing) was in fact brought in together with BGP4
 - Because of CIDR, you can't tell how long the prefix is by looking at the address
 - Instead (e.g. in RIPv2 packets) you specify the complete prefix, e.g. 130.216.32.0/24

Special types of IPv4 address (1)

- So far we have discussed *unicast* addresses
 - That means an address used to send a packet to exactly one interface
- IP also supports *multicast* addressing and routing
 - That means an address used to send a packet to a large set of interfaces in parallel
 - Multicast IPv4 addresses are under prefix 224/4:

[illegible]

- The *broadcast* address is 255.255.255.255 but it only works locally (it's blocked by routers)

Special types of IPv4 address (2)

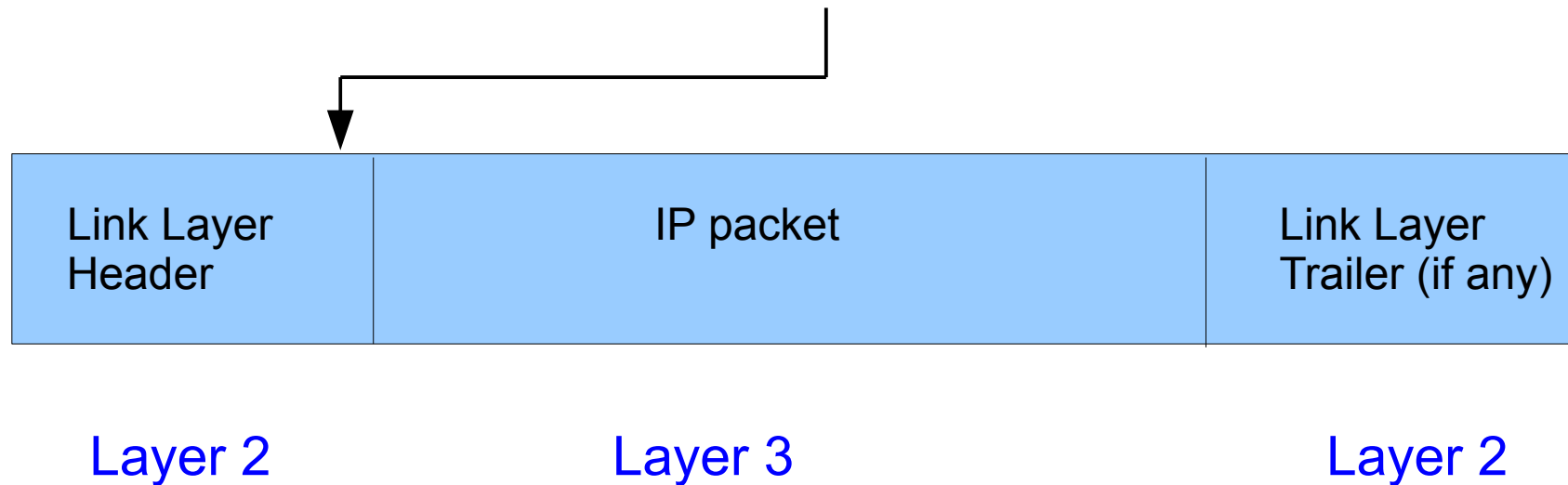
- Sometimes a unicast address is used as an *anycast* address
 - Used to send a packet to a group of interfaces, but only one should respond, normally to provide redundant servers
 - There is no way to tell an anycast address by looking at it; they have to be manually coded into the routing system
- 0.0.0.0 means “this host”
 - “host” is internet jargon for “computer”
 - 0.0.0.0/0 is also the way a default route is identified
- 127.0.0.1 is the loopback address (send packets to yourself)
- 169.254.0.0/16 is “link local” space for isolated networks (RFC 3927)

Special types of IPv4 address (3)

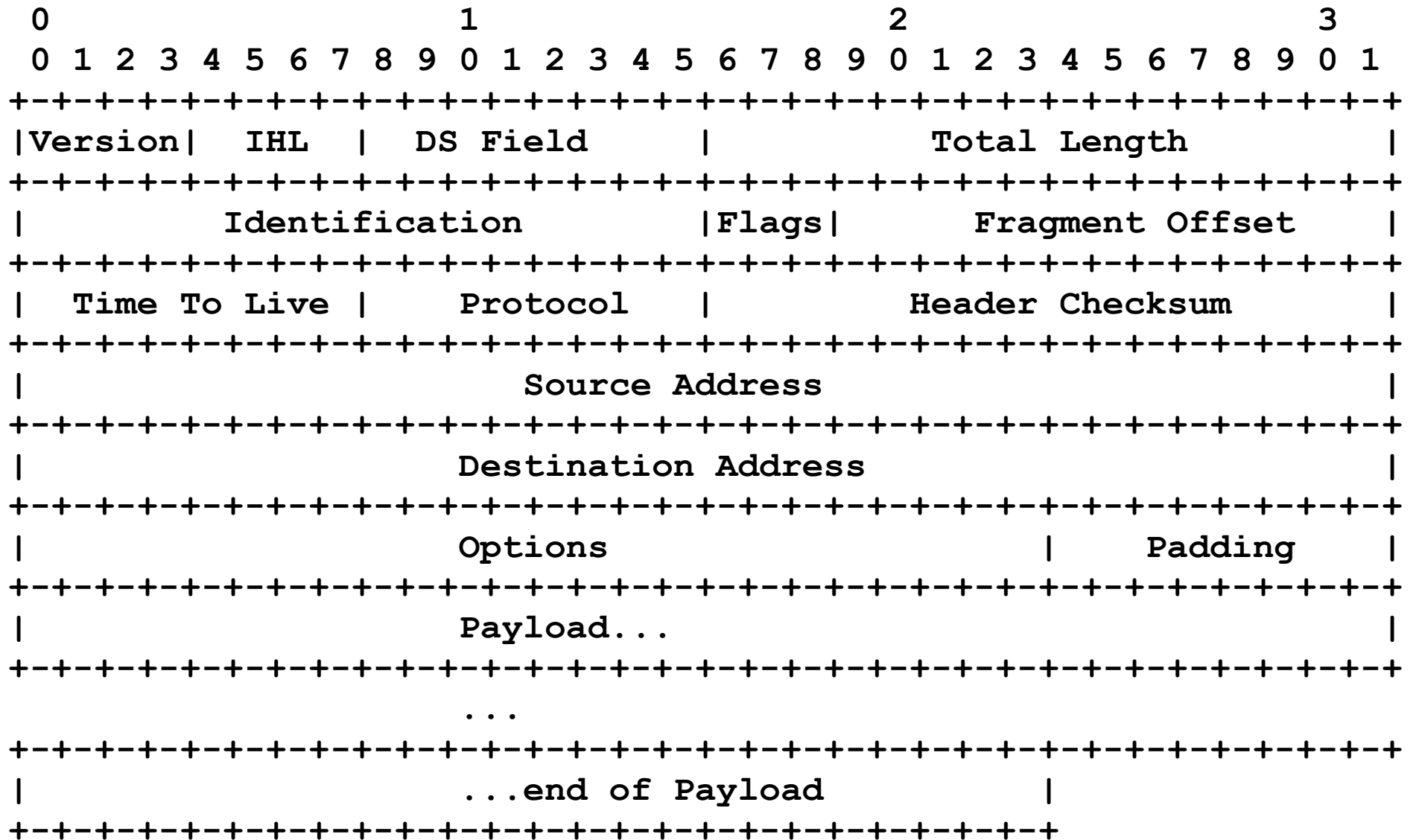
- Three address ranges are reserved for private use *within* a site
 - 10.0.0.0/8
 - 172.16.0.0/12
 - 192.168.0.0/16
- Since *anybody* can use these addresses, they are ambiguous and must *never* be routed off-site
- (This is not a complete list of special addresses. For a complete list, see RFC 3330 at www.rfc-editor.org)

Mapping to Layer 2

- The IP packet has to be sent inside a Layer 2 frame, such as an Ethernet frame
- The exact way this is done depends on the type of Layer 2 link
 - e.g. using Ethertype 0x0800 on Ethernet



IPv4 Packet Format



Explanation of IPv4 header (1)

- Version: 4
- IHL (IP header length)
 - header length (bytes/4, i.e. 32 bit words)
- DS (differentiated services) Field, previously known as TOS (type of service) Field
 - 8 bits used to manage quality of service
- Total Length
 - length of IP header plus IP payload (bytes)
- Identification, Flags and Fragment Offset
 - used for packet fragmentation, see later

Explanation of IPv4 header (2)

- Time To Live (often called TTL)
 - actually a hop count, decreased by 1 at each router. The packet is discarded if TTL=0, to prevent loops
- Protocol
 - a value that defines the type of payload (TCP, UDP, etc.)
- Header checksum
 - 16 bit 1's complement of 16 bit 1's complement sum of all other header fields
 - recalculated by each router, since TTL changes
- Source and Destination addresses
 - as defined previously

IPv4 Header Options

- Most packets don't have them
 - New options are hard to deploy since old routers don't like them
- All options start with an option type byte

```
+-----+  
| CxxNNNN |  
+-----+
```

- C = 1 copied into each fragment, in case of fragmentation
 - C = 0 not copied
 - xx = option class (control or debugging)
 - NNNNN = option number
- Most options have more bytes

```
+-----+-----+-----+      +-+  
| CxxNNNN | size   | data... |    ...  |  
+-----+-----+-----+      +-+
```


Example IPv4 Header Options

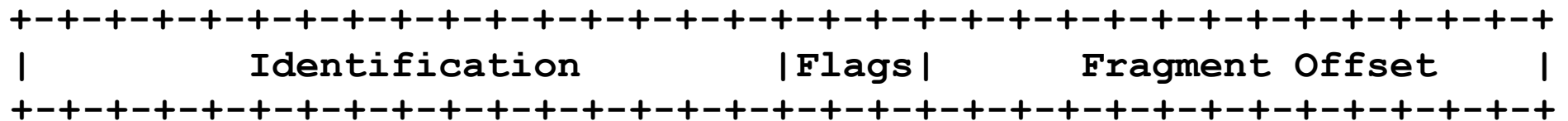
- Record Route
 - each router inserts its address in the option
 - generally blocked due to security worries
- Loose Source Route
 - allows the sender to specify the route
 - also performs 'record route'
 - generally blocked due to security worries
- Router Alert
 - tells each router to check further into the packet instead of just forwarding it
 - a good way to slow your packet down
- Generally speaking, header options were not a big success in the IPv4 design

Fragmentation

- An IPv4 host is required to handle datagrams of at least 576 bytes including the IPv4 header
- A given network path has a Maximum Transmission Unit (MTU) size, normally more than 576
 - Somewhat less than Ethernet size is common, 1400-1500 bytes
 - Fragmentation is designed to work for link MTUs down to 68 bytes
- Two problems to send a packet $>$ link MTU size
 1. How to know what the MTU size is?
 2. How to split the large packet up?
- For the moment, assume we know the MTU size
- The hard part isn't fragmentation; it's re-assembly

The fragment header

- The sender splits up the packet; each fragment has a fragment header:



- Identification: all fragments of the same packet have the same value
- Flags
 - one unused bit
 - DF bit - if set, Don't Fragment this packet
 - MF bit - if set, More Fragments will follow
- Fragment offset: how far into the packet this fragment begins, in units of 8 bytes
- If a sender (usually a router) knows that the next hop MTU is too small, it splits the packet into fragments

Reassembling fragments

- Routers don't reassemble fragments; that's left to the final receiver
- If you receive a packet with an unknown non-zero Identification value, you must
 - reserve a reassembly buffer
 - tag the buffer with the Identification value
 - store the fragment in the buffer at the given offset (remembering that the first fragment may not arrive first and the last fragment may not arrive last)
 - as further fragments with the same Identification arrive, store them in the buffer
 - when all fragments have arrived, act as if the whole packet had just arrived
 - if not all fragments arrive after a timeout, discard the buffer

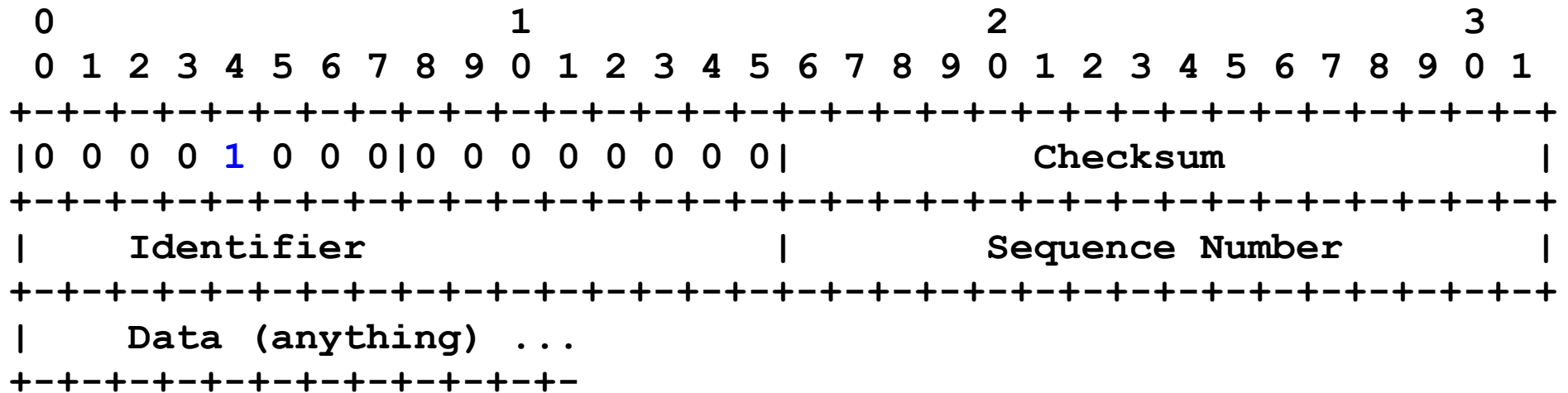
Problems with fragmentation

- Double fragmentation
 - if MTU reduces twice along the path, fragmentation could happen twice
- Silly fragmentation
 - if the actual MTU is just a bit shorter than each packet we'll keep sending one long fragment and one very small one
- Reassembly is a slow process
- Interferes with TCP flow control
- On gigabit networks, the 16 bit ID field can wrap around ('recycle') in less time than the reassembly timeout
 - disastrous, as fragments of a new packet may be mistaken for lost fragments of an old one

ICMP: Internet Control Message Protocol

- Used for low-level management functions in an IP network
- Sent as IP packets with Protocol = 1
- First byte of payload is an ICMP Message Type
- ICMP packets typically report errors in the processing of IP packets
 - To avoid recursion of messages about messages, no ICMP messages are sent about ICMP messages
- Now three example ICMP messages ...

ICMP “Echo” and “Echo Reply”



Type

8 = Echo, 0 = Echo Reply

Identifier

A random value used to match echo requests and replies

Sequence Number

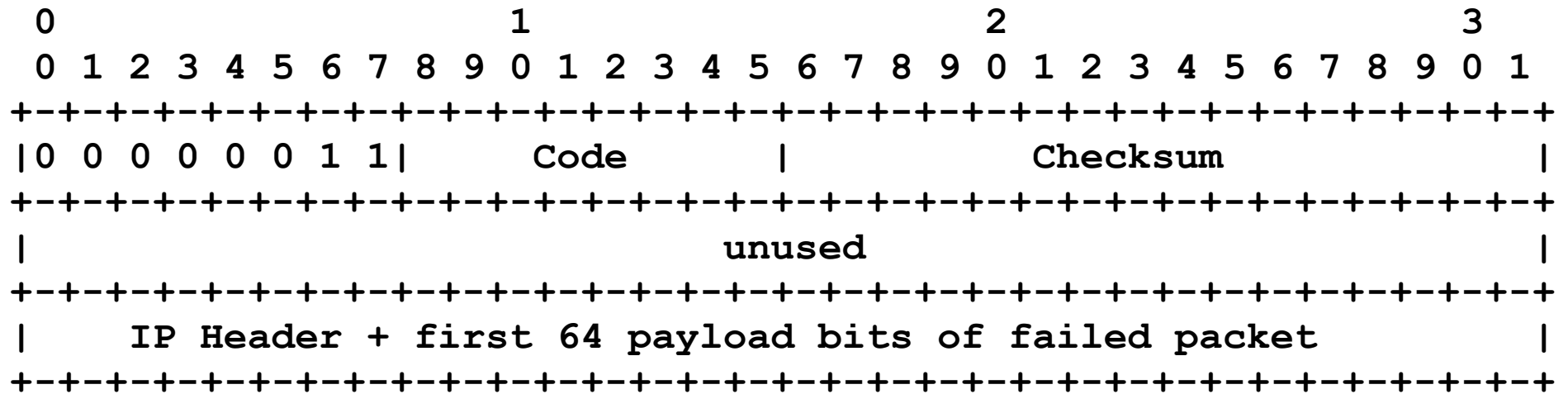
Counts up, to match requests and replies in series

Data

Should be sent back without change

Note: This is what *ping* uses.

ICMP “Destination Unreachable”



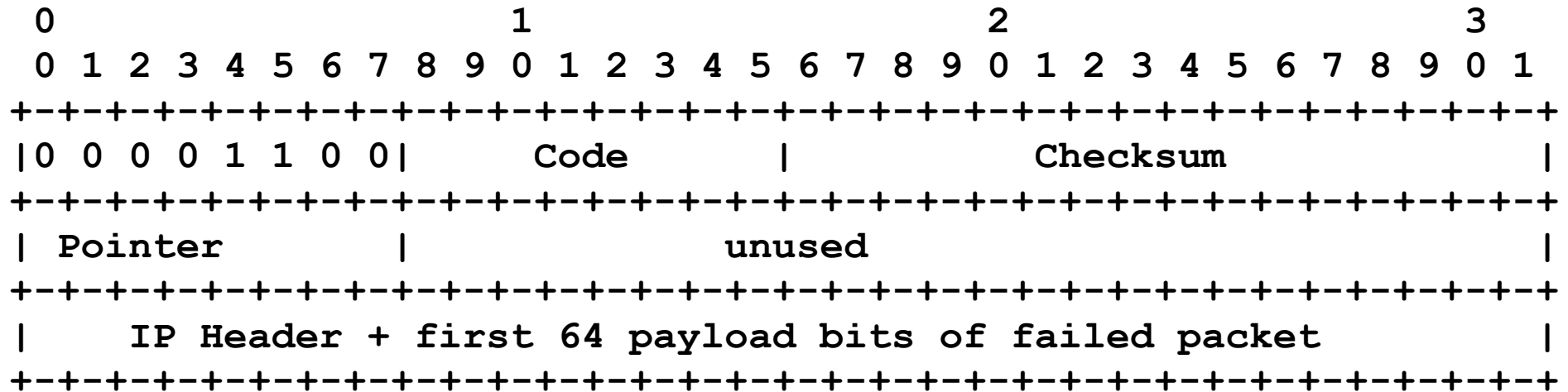
Code

- 0 = net unreachable
- 1 = host unreachable
- 2 = protocol unreachable
- 3 = port unreachable
- 4 = fragmentation needed but DF set
- 5 = source route failed

Checksum

16 bit 1's complement checksum of ICMP message

ICMP “Parameter Problem”



Code

0 = pointer points to error

No other values defined

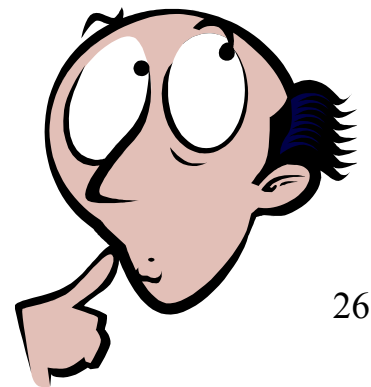
Pointer

Byte number in failed packet where problem was found

Dynamic Host Configuration Protocol

- For many years, addresses had to be assigned by hand and configured by hand
 - Obviously impractical once PCs appeared by the million
 - DHCP appeared by 1993
- DHCP allows a machine to ask a central server for an address (and other info) when it reboots
 - May be a different address each time, which is OK for clients but inconvenient for servers
- First step is to send a request to the DHCP server
 - But after a reboot, you don't know the address of the DHCP server and you don't have an IP source address to send from. A bit of a puzzle

Oops! DHCP is important, but not detailed in Shay.



Bootstrapping DHCP

- Client starts by *broadcasting* a DHCP DISCOVER message on its LAN
 - Source IP address is 0.0.0.0
 - Destination IP address is 255.255.255.255
 - Destination hardware address is LAN broadcast
 - DISCOVER message includes client's LAN hardware address
- DHCP server will catch the broadcast and reply with a DHCP OFFER message
 - An OFFER message includes a fresh IP address for the client
 - Source address is the DHCP server's own IP address
 - Destination IP address is the offered IP address
 - Destination hardware address is the one supplied by the client

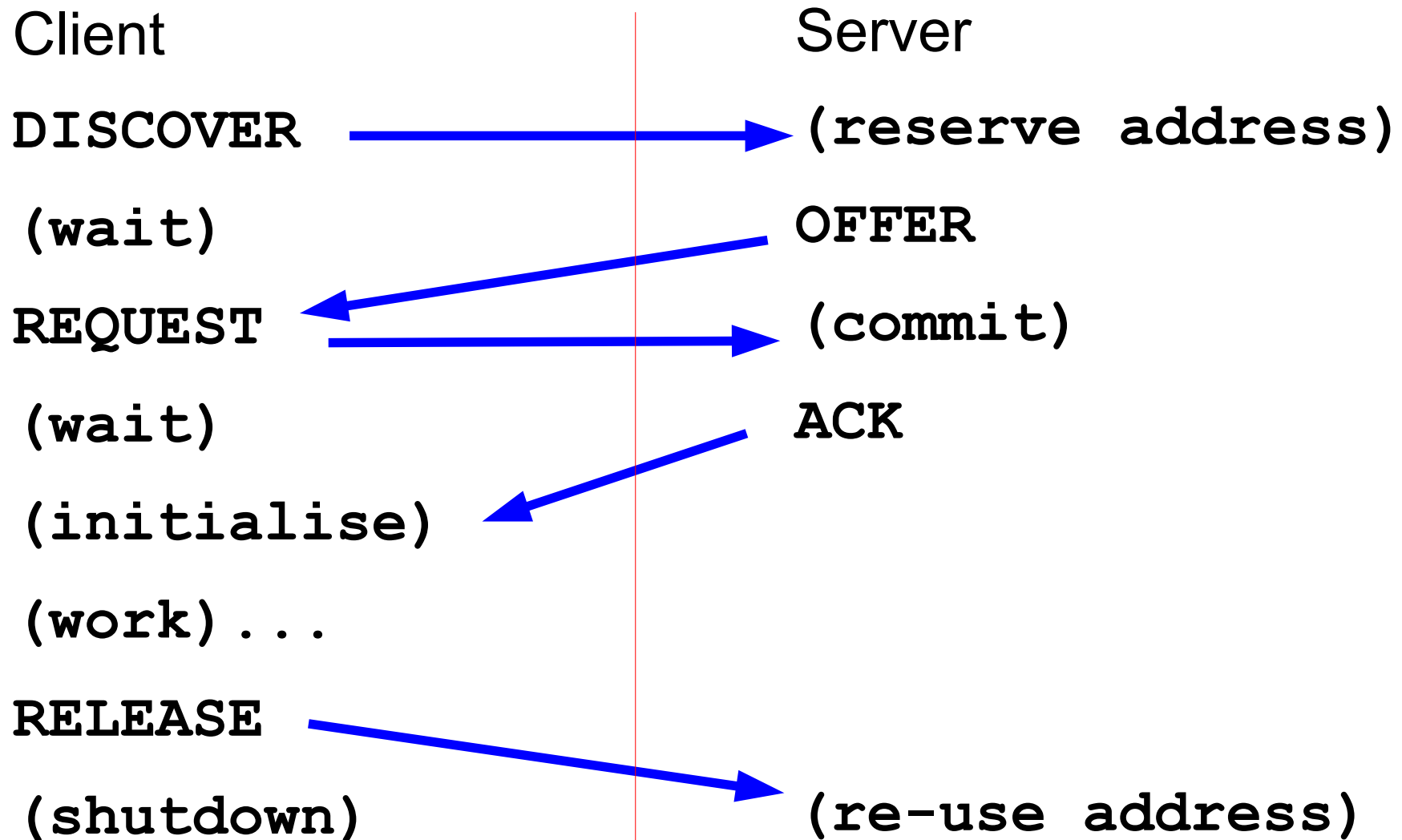
Some DHCP details

- DHCP is built up from an older bootstrap protocol called BOOTP
 - BOOTP and DHCP messages are sent over UDP (to be discussed later), not raw IP
- Either the DHCP server is on the LAN, or a 'DHCP relay' (built into a router) will catch the DHCP DISCOVER and send it on
- There can be several DHCP servers and several DHCP OFFER messages
 - The client must choose one of them
- DHCP addresses have a lifetime (known as a *lease*)
 - The client must renew after that lifetime expires

DHCP message types

- DISCOVER, OFFER - as above
- REQUEST - client requests to accept OFFER or extend lease
- ACK - server accepts REQUEST
- NAK - server denies REQUEST or expires lease
- DECLINE - client rejects OFFER
- RELEASE - client has finished with address
- INFORM - client has address, but requests other parameters

Normal DHCP sequence



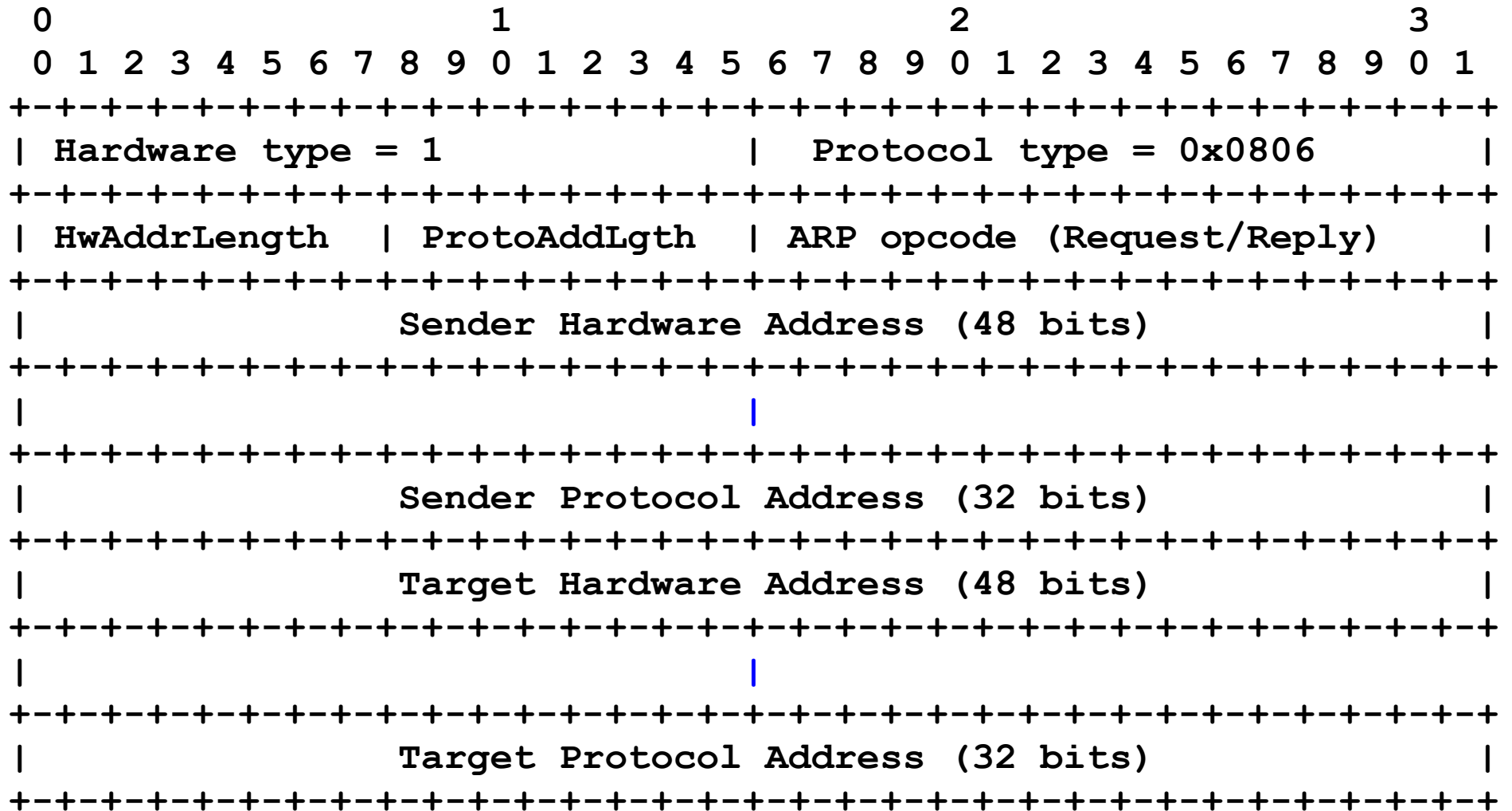
Other information (optionally) supplied by DHCP

- DHCP now has >100 optional parameters
 - Default router address(es) ('default gateway')
 - Static routes
 - Local net mask
 - DNS server address
 - Parameters for MTU discovery
 - Parameters for router discovery
 - Type of Ethernet encapsulation
 - ...
 - Mail server addresses
 - Timezone information
 - Physical location data (street address etc.)

Finding Neighbours: Address Resolution Protocol

- Suppose you have an IP address from DHCP as well as the IP address of the default router
 - You: 130.216.1.17
 - Router: 130.216.1.1
- By definition, the default router is on your LAN, but how do you know its Ethernet address?
 - That is the problem ARP solves
- Concept
 - Broadcast an ARP Request asking for 130.216.1.1
 - That host unicasts an ARP Reply
 - Cache the Ethernet address found in the Reply

ARP message format (on Ethernet)



ARP message format notes

- ARP is carried directly over Layer 2, not over IP, using Ethertype 0x0806
- Hardware type, etc., allow for use over other LAN types than Ethernet and other protocols than IPv4
- Opcodes: Request = 1, Reply =2
- Target Hardware Address is blank in Request and filled in in the Reply
 - Target and Sender are swapped between Request and Reply

ARP in practice

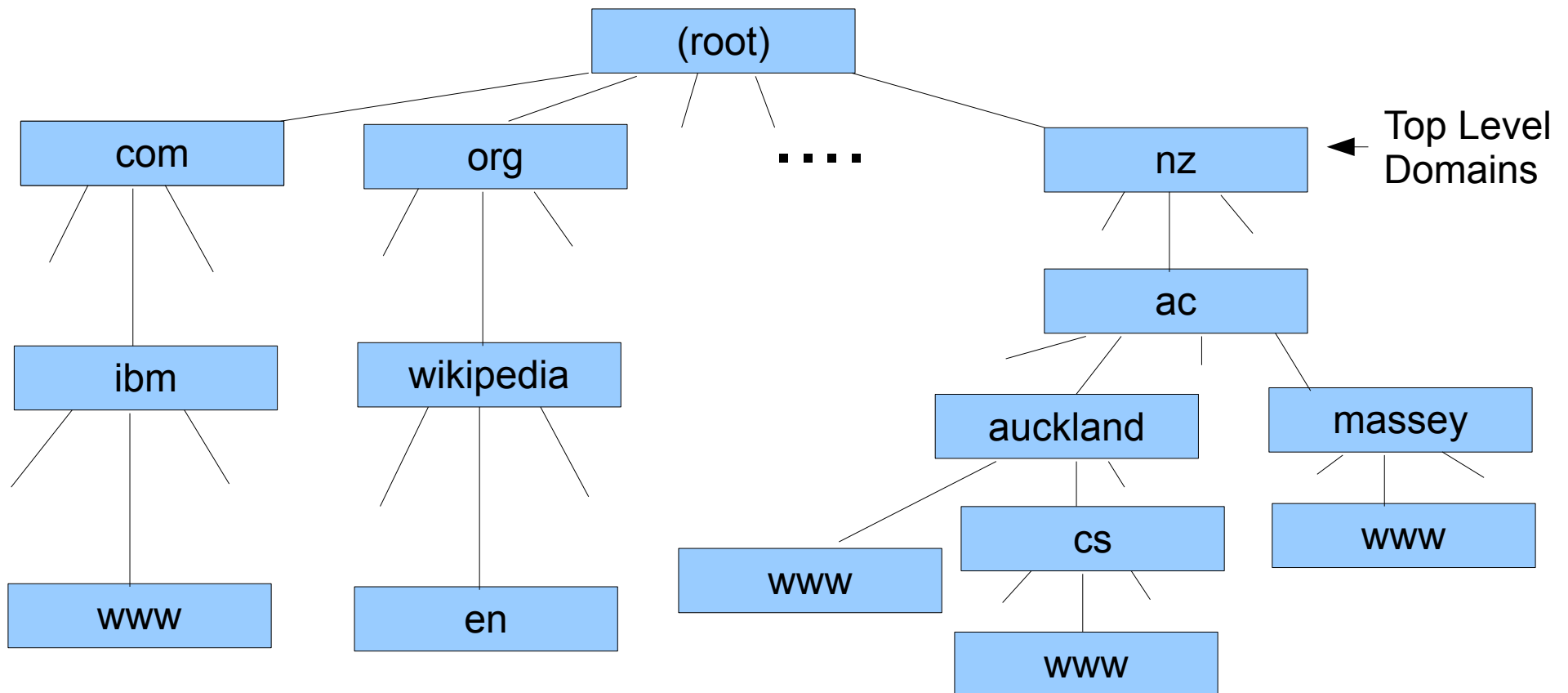
- Clear ARP cache on restart to avoid stale data
- Two Replies to one Request - disaster!
 - Somehow, two hosts believe they have the same address
 - Should not trust either of the replies
- When a host disconnects, DHCP might give its address to someone else - but it's still in your ARP cache - disaster!
 - ARP cache timeout must be short compared to DHCP hold time
 - Unsolicited ARP with **Sender=Target** refreshes the cache
- ARP Reply may come from a proxy (e.g. a bridge)

Let's see where we are ...

- We know what an IPv4 packet looks like
- We know how to get an IPv4 address, default router address, etc. (DHCP)
- We know how to find a neighbour's LAN hardware address, given its IPv4 address (ARP)
- We know how to send a packet, fragment and reassemble packets, and handle packet level errors (ICMP)
- We know how to route off the LAN (RIP, OSPF, BGP4)
- *Missing:* how do we find the IPv4 address of another system from its name?

Naming Things: DNS (Domain Name System)

- Basic concept: unique names in a structured tree
 - Tree is string-based, n-ary (not binary)



DNS names

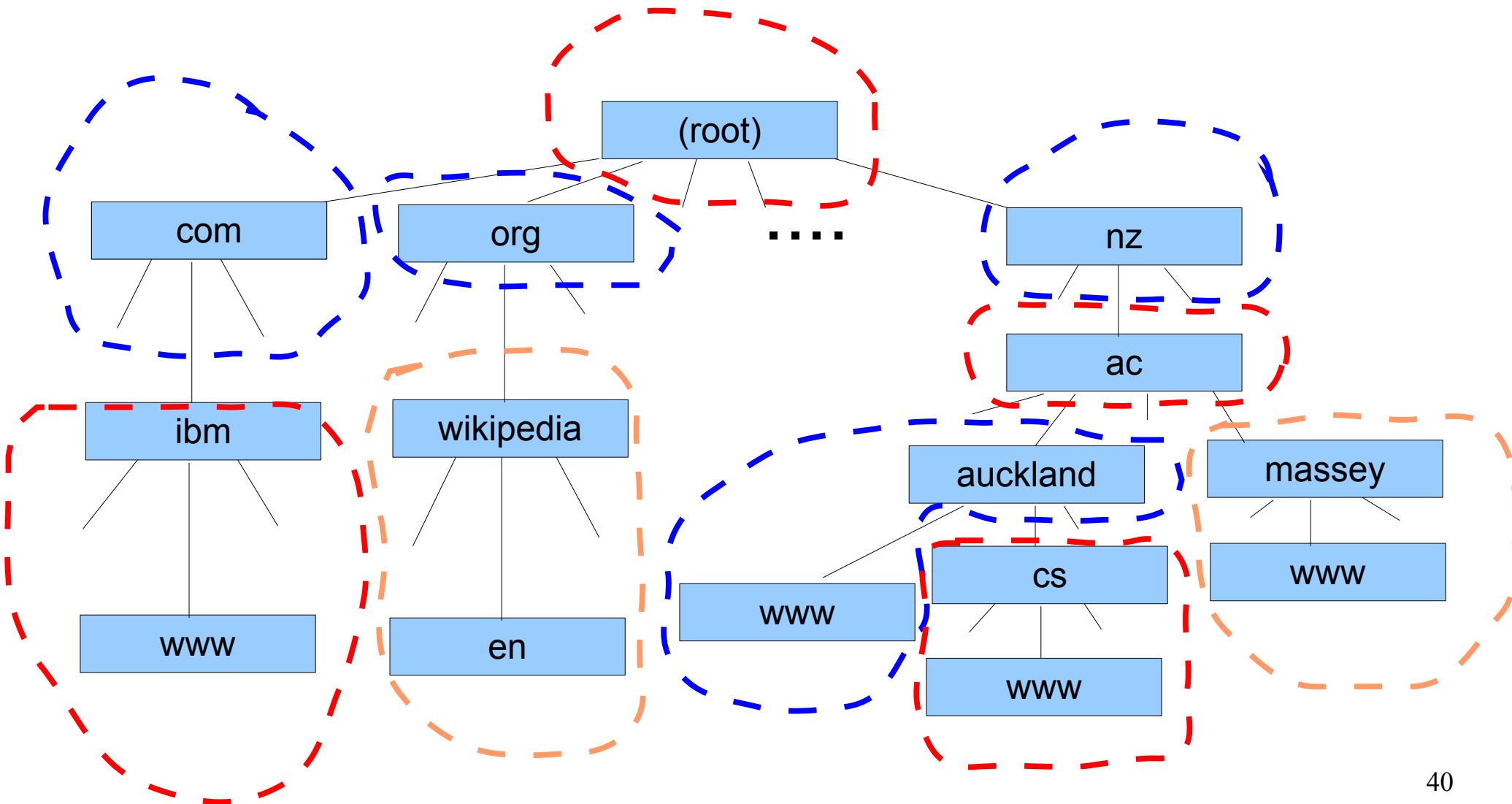
- *www.auckland.ac.nz* and *www.cs.auckland.ac.nz* are FQDNs - Fully Qualified Domain Names
- They are unique (i.e. represent different leaves on the DNS tree)
 - The DNS must have a unique root
 - Names must be registered to guarantee uniqueness
- TLD (Top Level Domain) names are registered worldwide by IANA (Internet Assigned Numbers Authority)
- Each TLD such as *com* or *nz* has its own registry
- Subdomains such as *ac.nz* and *ibm.com* manage their own registries

DNS is a massive Distributed Database

- The database contains hundreds of millions of entries of several types, called RRs (resource records)
- The most important RR type today is an A record
 - The A record for *www.cs.auckland.ac.nz* contains 130.216.33.106
- When a client machine in Switzerland asks its local DNS server for that A record, how does it get there from Auckland?
 - Obviously, it is impractical for every one of the millions of DNS servers in the world to be pre-loaded with hundreds of millions of RRs
 - Obviously, it would be horribly slow if every lookup of every FQDN had to be sent back to the original registry that registered it

Divide and Conquer: DNS Zones

- The namespace is divided into hierarchical zones



Authoritative name servers

- Each zone contains NS records for the authoritative name servers for its child zones
 - The root has an NS record for *nz*
 - *nz* has an NS record for *ac.nz*
 - *ac.nz* has an NS record for *auckland.ac.nz*
 - *auckland.ac.nz* has an NS record for *cs.auckland.ac.nz*
 - *cs.auckland.ac.nz* has no NS records - it is a leaf zone
- The authoritative name servers are configured with all RRs for all FQDNs in their zone
 - But not for FQDNs in child zones; those are delegated
 - Configuration is often done from an equipment database, and requires careful clerical work

Finding the RRs for a given FQDN

- Our problem is reduced to finding the address of the authoritative server of the domain containing those RRs
- Every host includes code called a *resolver* which takes an FQDN and returns an RR
 - A full resolver interacts with multiple DNS servers in sequence
 - A simple resolver interacts with one “recursive” DNS server
 - In both cases, the lookup process is the same
 - Resolver, or recursive server, sends DNS Request messages
 - Servers send DNS Response messages

Illustrative full DNS lookup

- Resolver is pre-configured with well-known IP addresses of the *root servers* and knows nothing else
- DNS Request to a root server for NS record of *nz*
 - DNS Response with *nz* servers including
ns4.dns.net.nz = 203.97.40.200
- DNS Request to ns4.dns.net.nz for NS record of *ac.nz*
 - DNS Response with *ac.nz* servers including
ns6.dns.net.nz = 204.74.113.253
- DNS Request to ns6.dns.net.nz for NS record of *auckland.ac.nz*
 - DNS Response with *auckland.ac.nz* servers including
dns1.auckland.ac.nz = 130.216.1.2
- DNS Request to dns1.auckland.ac.nz for A record of *www.auckland.ac.nz*
 - DNS Response 130.216.11.202

Making DNS scale to trillions of requests per day

- That means avoiding full lookup in most cases
- Principle: all zones have a defined TTL (time to live). All DNS servers and resolvers may cache any RR found in a DNS Response until its zone TTL expires
 - You really shouldn't be looking up *.com* or *.nz* all the time!
 - Since TTL may be long (days), DNS updates sometimes lag unless somebody flushes the resolver cache
 - For example, *cs.auckland.ac.nz* has TTL=1 day. A resolver that has cached it will not see any change until tomorrow
- Practice: load sharing within a zone
 - Most zones of any size operate multiple parallel DNS servers to provide load sharing and backup
 - Zone files must be kept identical between them

Many other aspects of DNS

- This was an overview. We don't have time for:
 - DNS message formats (sent over UDP)
 - Reverse lookup (getting from an IP address to an FQDN)
 - Dynamic DNS updates (to avoid clerical work)
 - DNS Security (to prevent DNS spoofing)
 - Creative uses of DNS
 - DNS operational pitfalls
- DNS is the only example of a successful distributed database that is deployed worldwide on hundreds of millions of systems. Its designer (Paul Mockapetris) deserves great credit

Summing up on IPv4 ...

- We know what an IPv4 packet looks like
- We know how to get an IPv4 address, default router address, etc. (DHCP)
- We know how to find a neighbour's LAN hardware address, given its IPv4 address (ARP)
- We know how to send a packet, fragment and reassemble packets, and handle packet level errors (ICMP)
- We know how to route off the LAN (RIP, OSPF, BGP4)
- We know how to find the IPv4 address of another system from its FQDN (DNS)

References

- Shay 11.1, 11.2
 - Bug:
 - Talks about Class A, B, C addresses in present tense
- Amazon will find you good books on TCP/IP by
 - Douglas E. Comer and David L. Stevens
 - W. Richard Stevens
 - Pete Loshin
- Many RFCs, but the older ones are hard to understand. Try RFC 1122, but today the only true definition is the running code in Linux, Windows, etc.