IPv4 Fragmentation and Reassembly, ICMPv4, NAT

IP Fragmentation & Reassembly

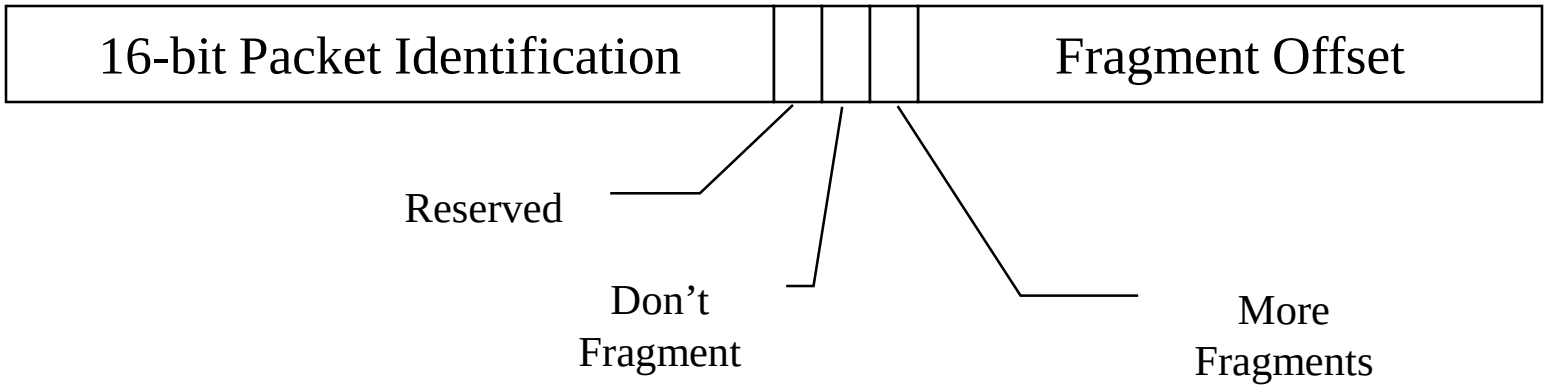
IP Fragmentation

- Maximum Transfer Unit (MTU) – maximum size of a physical frame
  - 1518 for Ethernet (including header), so 1500 bytes for the data
- When a router transmits a packet that is too large for the MTU of the outgoing link, the packet is fragmented
  - Otherwise the link layer will not be able to carry it
- Fragments may also be fragmented
- Fragmented packets are not reassembled until they reach their final destination
- Typically, if any fragment is lost, a router will discard all fragments. Routers usually only discover fragment loss if they drop the fragment themselves.
- The endpoint assumes fragments are lost after 30-60 seconds

Packets *vs.* Datagrams

- An IP datagram is the unit of end-to-end transmission at the IP layer (before fragmentation & after reassembly)
- A packet is the unit of data passed between the IP layer and the data link layer.
- A packet can be a complete IP datagram or a fragment

IP Header Fields Used for Fragmentation



Identifying and Ordering Fragments

- Fragments are identified using the datagram identification field in the header
  - All fragments of a datagram has the same identification field value
- Sequencing of fragments done using the Fragment Offset field
  - Contains offset from beginning of data carried by this fragment (in units of 8 bytes)
- Last fragment detected by "More Fragment" field
  - 1 bit of the Flags field in the header
  - Set to 1 on all but the last fragment of a datagram; set to 0 for the last fragment

Example

- IP datagram of size 2000 bytes (without header) with identification field = 12345 (say)
- MTU = 1500 bytes (without header)
  - So IP header + data in each packet must be ≤ 1500 bytes
  - Assume IP header size = 20 bytes
  - So maximum data that can be packed into one fragment = 1480
- Will break into 2 fragments of size 1480 and 520 bytes
- Put an IP header on each fragment (20 bytes header)
- Header fields
  - Fragment 1
    - Identification = 12345, Offset = 0, MF = 1, Total length = 1500
  - Fragment 2
    - Identification = 12345, Offset = 185, MF = 0, Total length = 540
  - Rest of the fields same (except checksum).
- Send the two fragments as independent IP packets

- Reassembly at receiver
  - Can happen only after all fragments are received
  - If any one fragment is lost/corrupted, the datagram cannot be reassembled
    - All fragments received of that datagram dropped after a timeout
  - How to know all fragments received?
    - Offset at each fragment says how many bytes present in other datagrams before this.
    - Gaps can be easily detected

- If a fragment is fragmented again at an intermediate router
  - Say at router A, the next link MTU is 900
  - Fragment 2 will pass fine
  - Fragment 1 will be fragmented again into 2 fragments
    - Fragment 1a and 1b with size 880 bytes (= 900 – size of IP header) and 600 (1480 – 880) bytes
    - Fragment 1a
      - Identification = 12345, Offset = 0, MF = 1, Total length = 900
    - Fragment 1b
      - Identification = 12345, Offset = 110, MF = 1, Total length = 620
  - What if the MTU was 400, so both fragments will get fragmented again?
    - Work out yourself. Think how the reassembly will happen

- What if we do not want a packet to be further fragmented in the middle by any router?
  - Set the DF (Don't Fragment) flag
  - If it cannot be sent by a router because MTU is smaller, it will be dropped
  - So why will we ever want to use it? Will see a scenario.

Internet Control Message Protocol (ICMP)

ICMP

- Required protocol with IP
- Used for reporting errors back to the source of an IP packet or for monitoring/measurement/feedback
- When a node drops an IP packet, an ICMP packet is sent back to the source
- Sent in some other cases without any error also
- Only error reporting, no error correction. Correction is left to the source node.
- RFC 792 and RFC 1122 (lists updated types)

ICMP Transmission

- ICMP packet contains ICMP header and may contain other information depending on type of message
- ICMP packet carried in data portion of an IP packet
- IP packet is routed normally back to the source
- Only difference: No ICMP packet is generated on error in IP packet carrying ICMP
- ICMP is not a transport layer protocol, it is a required support protocol at IP layer

ICMP Header Format

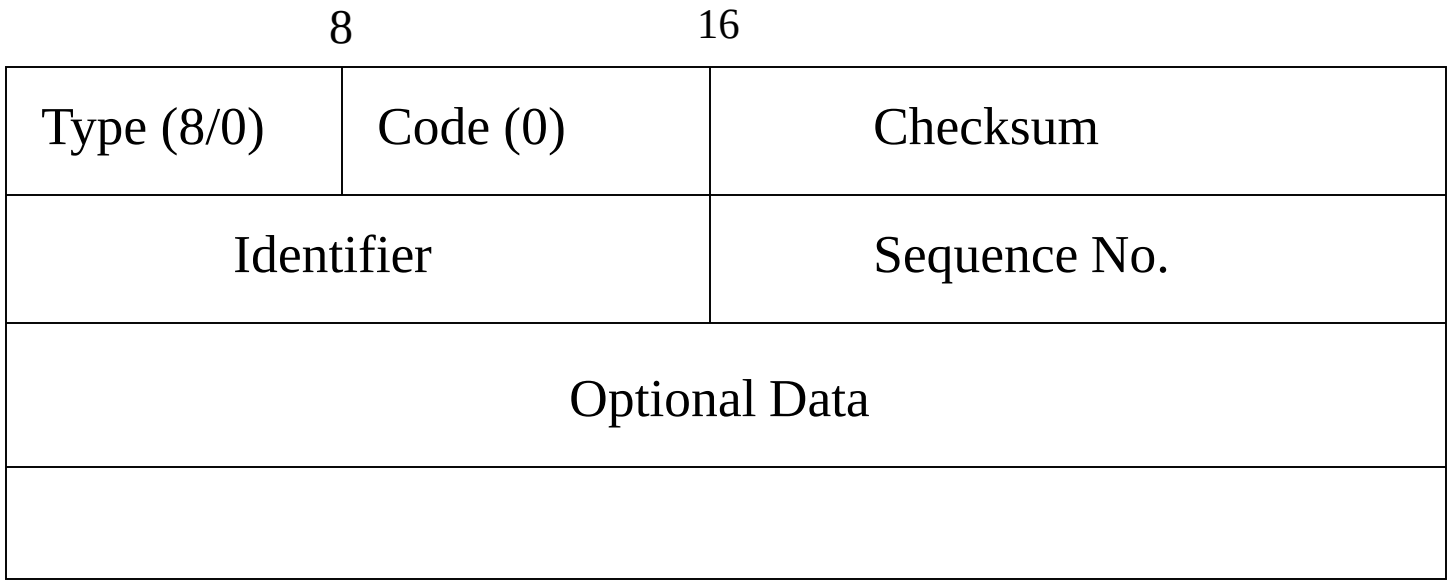
- 8-bit Type field
- 8-bit Code field
- 16-bit Checksum

ICMP Message Types

ICMP Message	Type
Echo reply	0
Destination unreachable	3
Source quench	4
Route redirect	5
Echo request	8
Time exceeded	11
Parameter problem	12
Timestamp request	13
Timestamp reply	14
Address mask request	17
Address mask reply	18

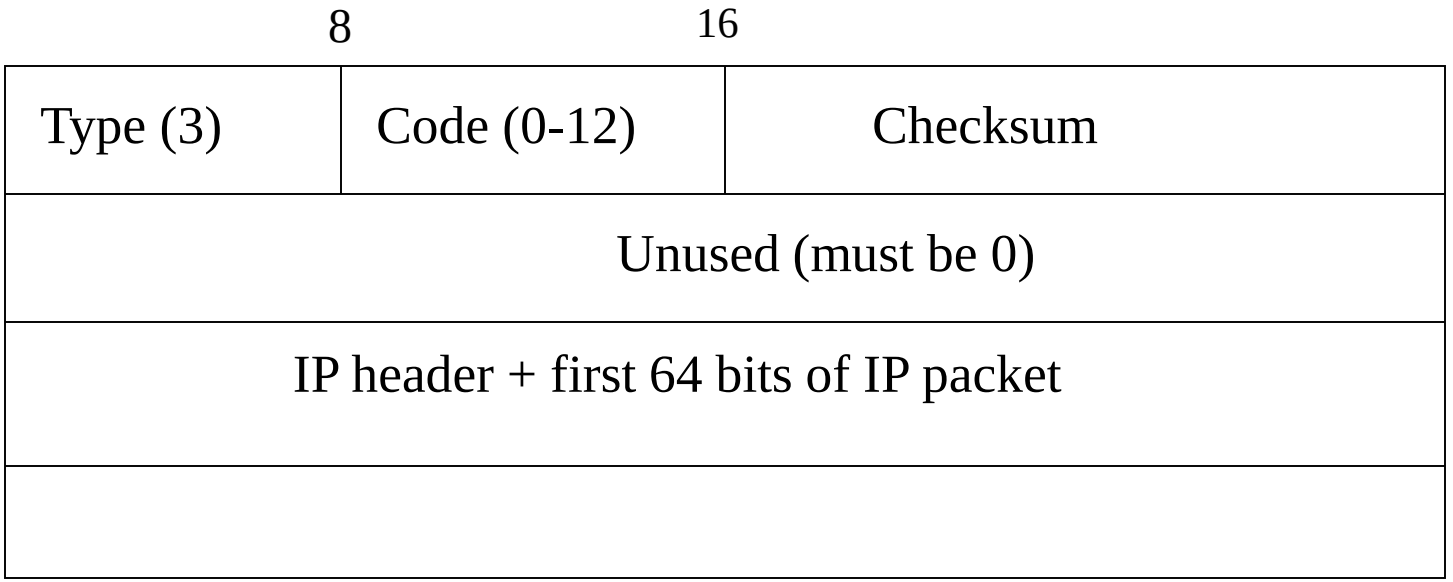
Echo Request/Reply

- To see if a destination is up and reachable
- Identifier/Sequence no. used to match request with reply
- Data may be sent in request, same data returned in reply, matched to see if destination is up
- Basis of *ping* tool



Destination Unreachable

- Sent by router when unable to forward or deliver a packet

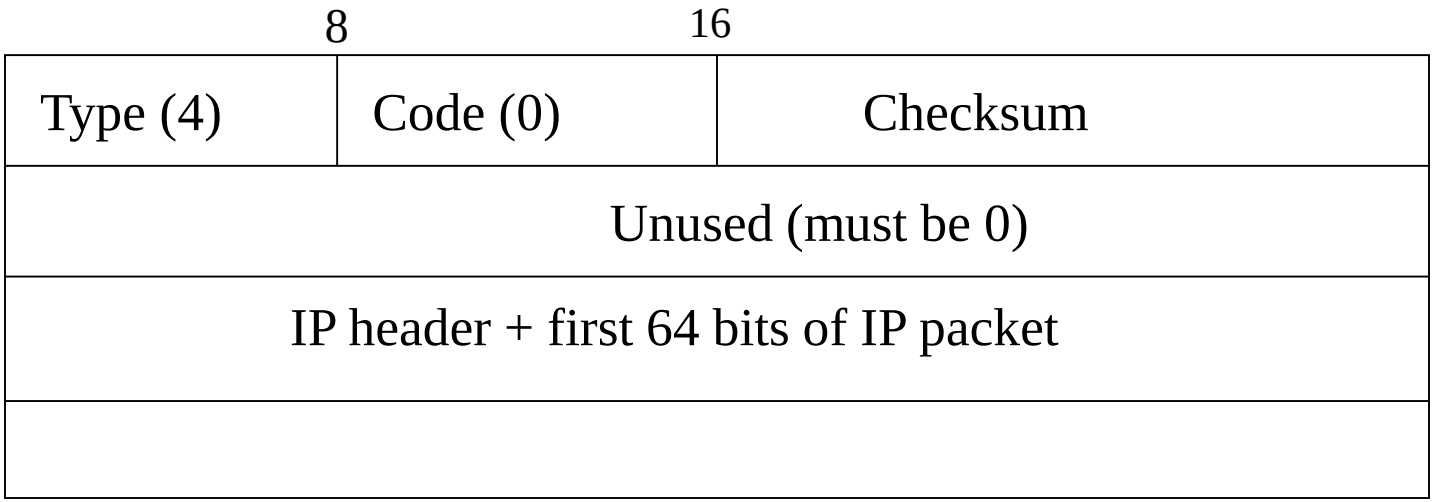


Example Code Values

Code	Meaning
0	network unreachable
1	host unreachable
2	protocol unreachable
3	port unreachable
4	fragmentation needed and DF set
5	source route failed
6	destination network unknown
7	destination host unknown

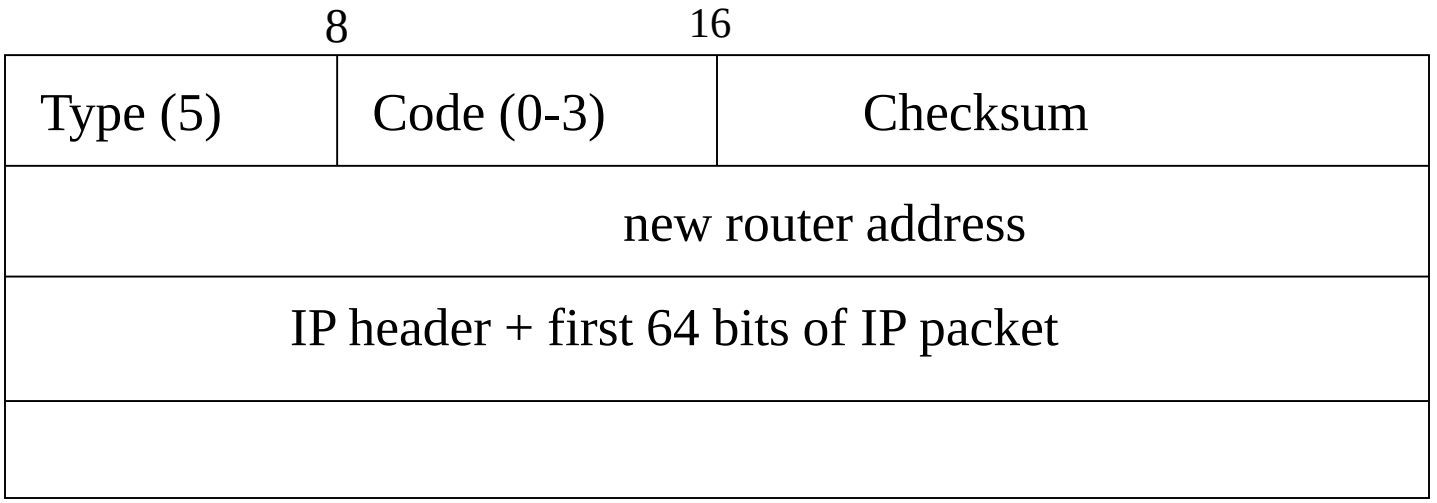
Source Quench

- Report congestion to source
- Sent when packet is dropped due to buffer overflow
- Source should reduce flow



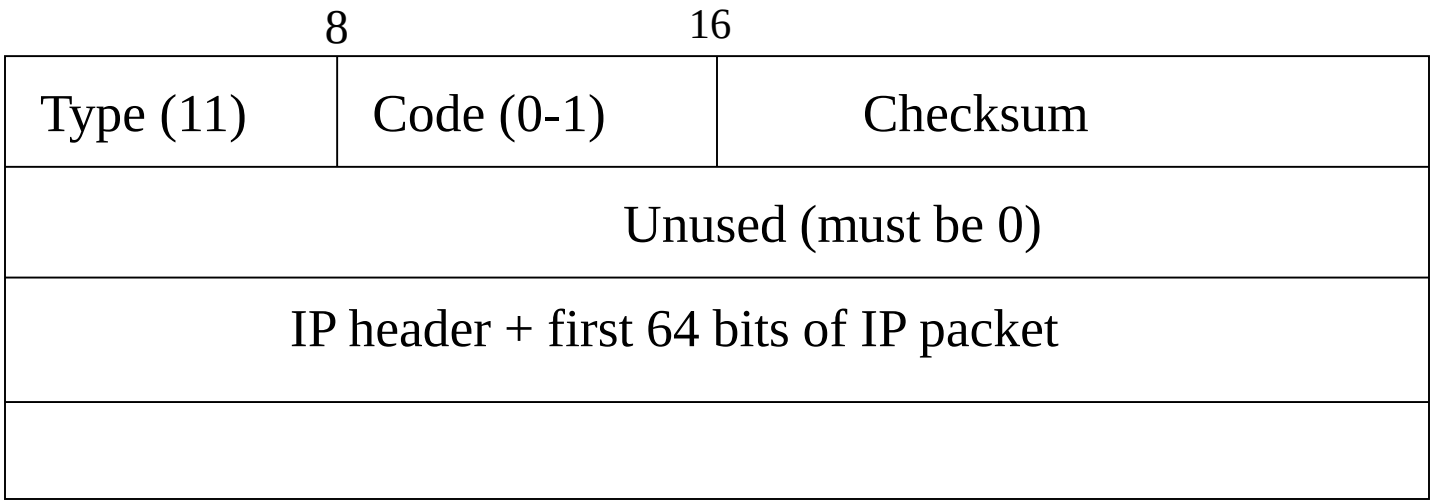
Route Redirect

- Used to let source know a better route to use for the destination address
- Packet is still forwarded



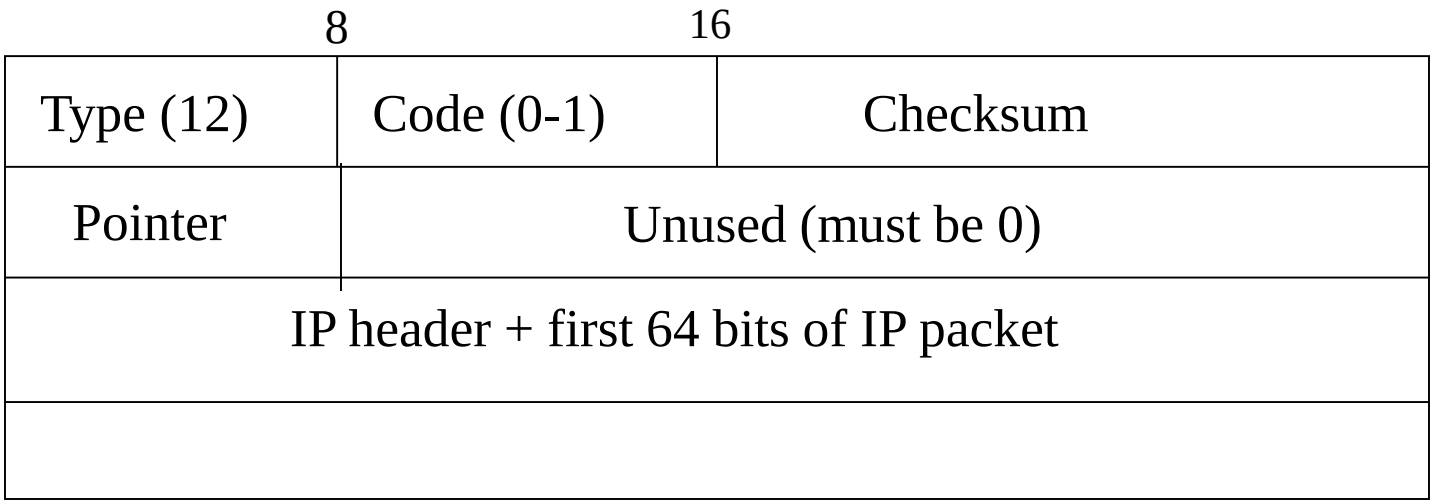
Time Exceeded for Datagram

- Used to detect circular or very long routes
- Sent when router discard packet because TTL = 0 (code 0) or fragment reassembly timer expires (code 1)
- Basis of *traceroute* tool



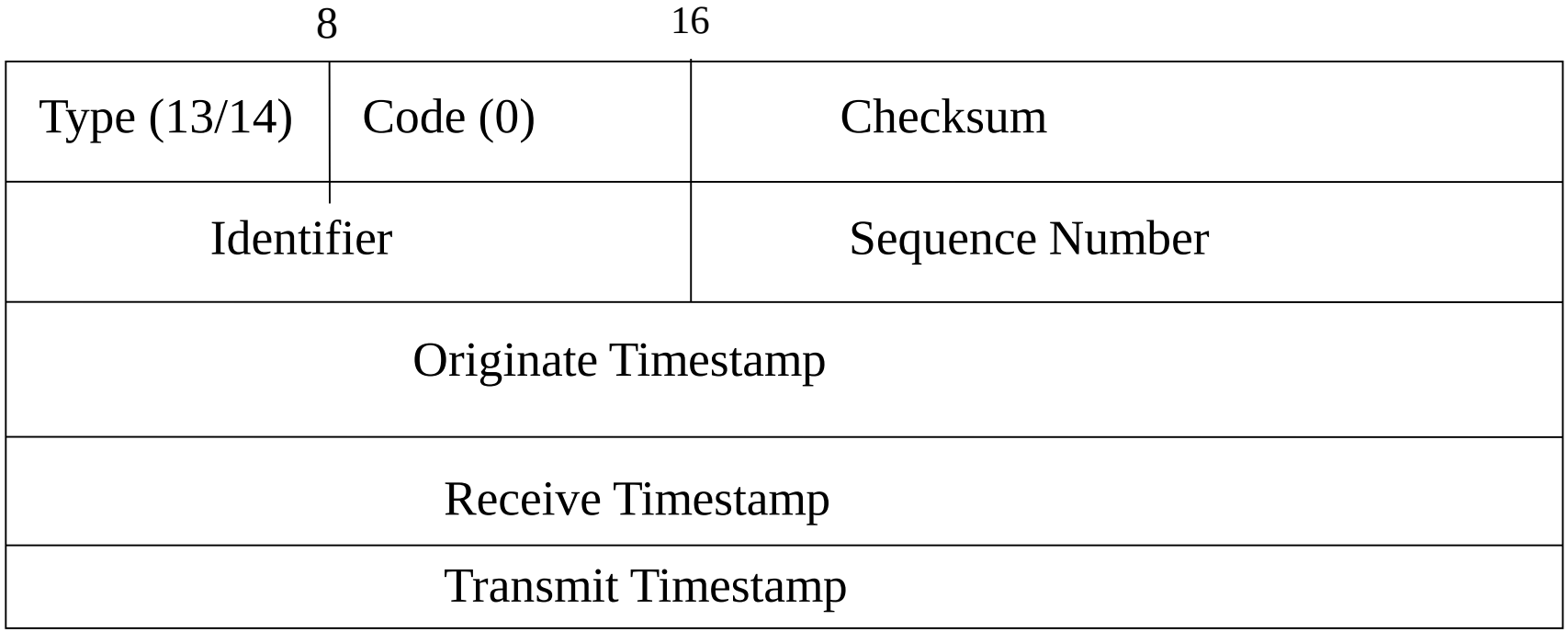
Parameter Problem

- Sent for any other errors that cause packet to be discarded (ex., incorrect option field)



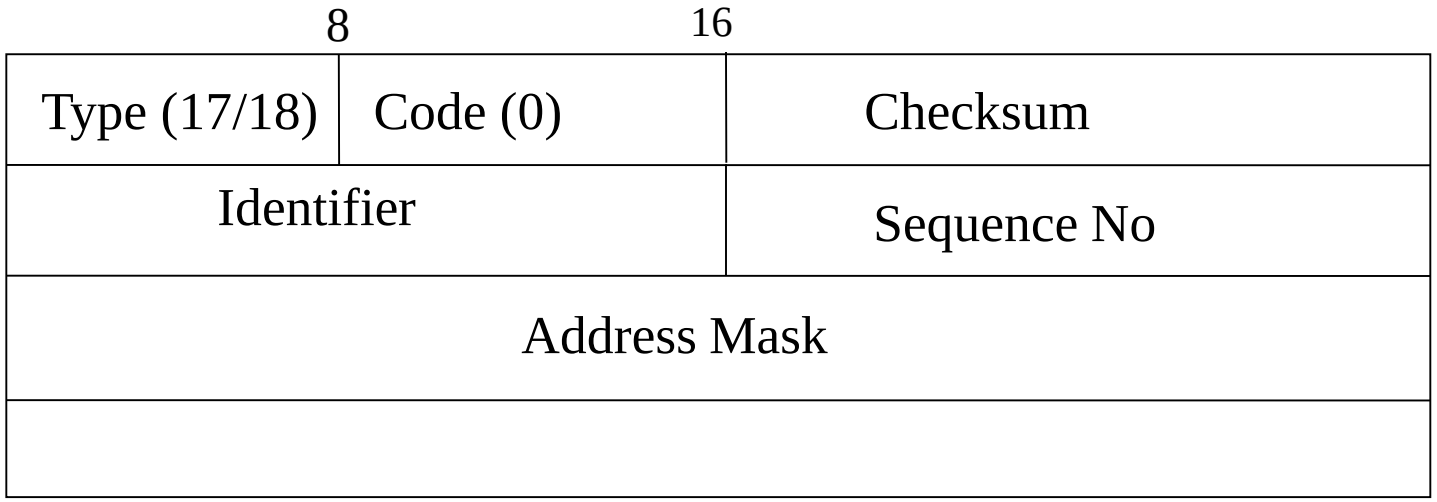
Timestamp Request/Reply

- May be used for clock synchronization



Address Mask Request/Reply

- May be used to obtain subnet mask of local network at boot



Network Address Translation (NAT)

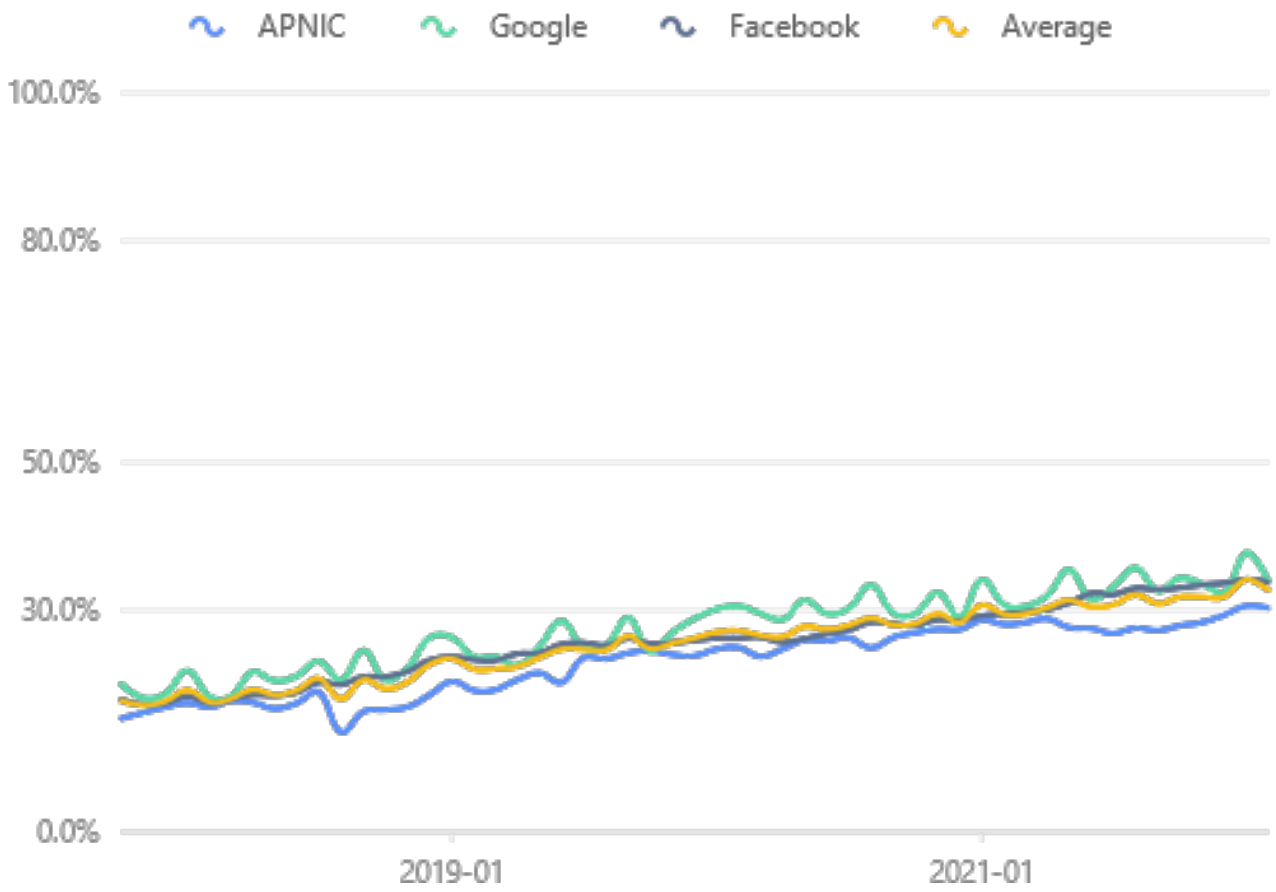
The Status of IPv4 Address

- IPv4 address space is limited
- No. of devices connected to the internet has exploded
  - Tens of billions of devices, and growing fast
- It was feared from early 2000's that we will run out of IPv4 addresses
- And we should have, even if each device needed one IP address
- Why haven't we?
  - Better allocation policies
  - IPv6
  - Use of private IPs

IPv6

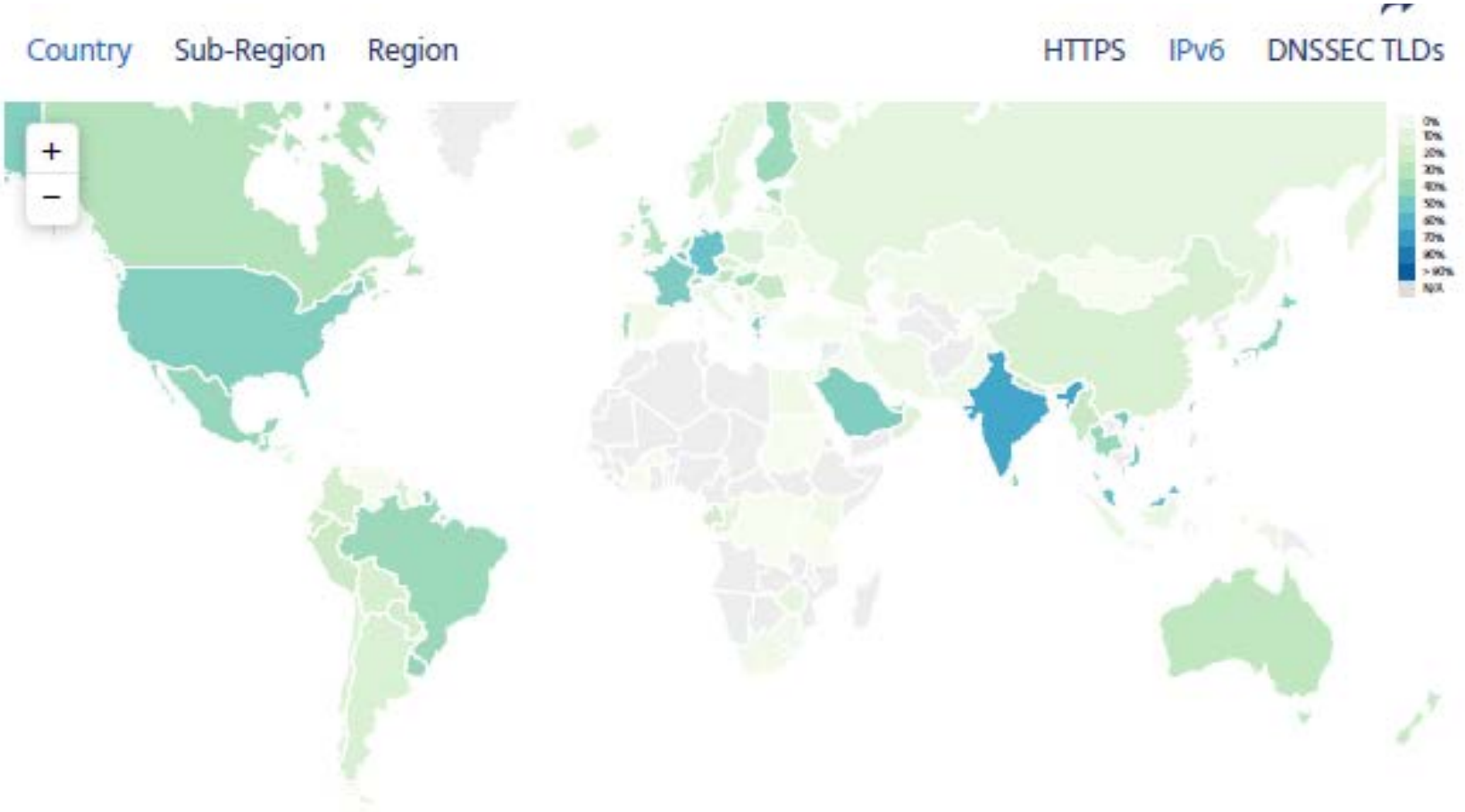
- Next generation IP protocol
- Allows for 128 bit addresses
  - Large enough for any foreseeable need
  - Other features also, but we will not cover here
- In early 2000's it was believed that IPv4 addresses will run out and the entire internet will transition to IPv6
- Hasn't happened, though there is different level of IPv6 penetration in different regions
  - Only about 34% of the top 1000 websites world-wise support IPv6 (Source: https://pulse.internetsociety.org/)

Percentage of IPv6 Traffic in Google, Facebook, APNIC



Source: https://pulse.internetsociety.org/

IPv6 Adoption by Region



Source: https://pulse.internetsociety.org/

- So IPv4 is still very much here, and will be
- But allocation policies are stricter, cannot just get a large block of IP addresses from the RIRs or from ISPs
- So how do we connect a very large number of devices with IPv4 addresses?
  - Use Private IP addresses

Using Private IP Addresses

- Think of your institute
- Internally, every machine gets a 10.*.*.* address from DHCP, which is a private IP
- The 10.*.*.* addresses never go out of IIT Kharagpur
  - No IP packet leaving or entering IIT Kharagpur has 10.*.*.* as either source address or destination address in the IP header
- The end network device (NAT-capable router for example) connecting IIT Kharagpur to the external world replaces the 10.*.*.* address with a public IP by a technique called NAT (Network Address Translation)
- Lets take an example

- Suppose your laptop in hostel has IP address 10.40.30.5.
  - Initially assume TCP connections only
- You connect to google with IP address 142.250.189.206, port 443
- Your IP packet goes with source IP = 10.40.30.5, source port = 20000 (say, assigned by system), destination IP = 142.250.189.206, destination port = 443
- The NAT-capable device intercepts the packet
- Suppose public IP used by IIT Kharagpur is 203.110.245.242

- NAT (Network Address Translation)
  - The device replaces the source IP 10.40.30.5 with 203.110.245.242 in the IP packet before sending it
  - Remembers the mapping (<10.40.30.5>, <142.250.189.206>)
  - When a packet is received with destination IP 203.110.245.242 and source IP = 142.250.189.206, the mapping is looked up, the destination IP is changed to 10.40.30.5, and routed inside the IIT network to your laptop
  - So one network address (private) is replaced by another network address (public)
  - There can be a pool of public IPs instead of one

- Problem: Allows only one internal host to make one connection to the internet at one time with a single IP
  - If 5000 people want to connect at the same time, NAT will need 5000 public IPs (anything > than the number of public IPs in the pool)
  - Getting so many public IPs is not feasible, does not scale
- Solution: Use port addresses also in NAT
  - Idea: Public IP addresses are scarce, but port addresses are plenty
    - Reuse the same public IP but with a different port
  - Also called NAPT (Network Address and Port Translation)

An Example of Using Ports in NAT

- In the earlier example, suppose another user connects from his/her laptop to the same site from 10.100.20.5 with port 30000
- Network device does the following:
  - When packet from 10.40.30.5 is received
    - Assigns a new port, say 34780
    - Creates the mapping (<10.40.30.5, 20000>, <142.250.189.206, 443>, <34780>)
    - Changes source IP to 203.110.245.242, source port to 34780 in the packet
    - Sends the packet

- When packet from 10.100.20.5 is received
  - Assigns a new port, say 34781
  - Creates the mapping (<10.100.20.5, 30000>, <142.250.189.206, 443>, <34781>)
  - Changes source IP to 203.110.245.242, source port to 34781 in the packet
  - Sends the packet

Internal IP	Internal Port	External IP	External Port	NAT Port Assigned
10.40.30.5	20000	142.250.189.206	443	34780
10.100.20.5	30000	142.250.189.206	443	34781

- When a packet is received with say source IP = 142.250.189.206, source port = 443, destination IP = 203.110.245.242, destination port = 34781
  - Look up the table see that the fields match the entry for 10.100.20.5
  - Change destination IP to 10.100.20.5
  - Change destination port to 30000
  - Send the packet to the internal host
- So the laptops do not see the public IP and the ports 34780/34781
- The external network does not see the 10.*.*.* addresses
- A single public IP can support multiple connections by changing port numbers

- We have assumed only TCP connections so far. What if there are other protocol packets?
  - One more field, protocol, added to the table
- What if two connections from internal machine uses the same client port
  - NAT port assigned will be different anyway, so packet will go with distinct source port
- How is the mapping table built?
  - Primarily by connections initiated from inside to outside
  - Other methods are there, including static mapping of some private IPs to fixed public IP
- Other uses of NAT
  - Organization has more machines than public IP, but only a small number of them access the internet at one time
  - Hiding internal addresses from the external world