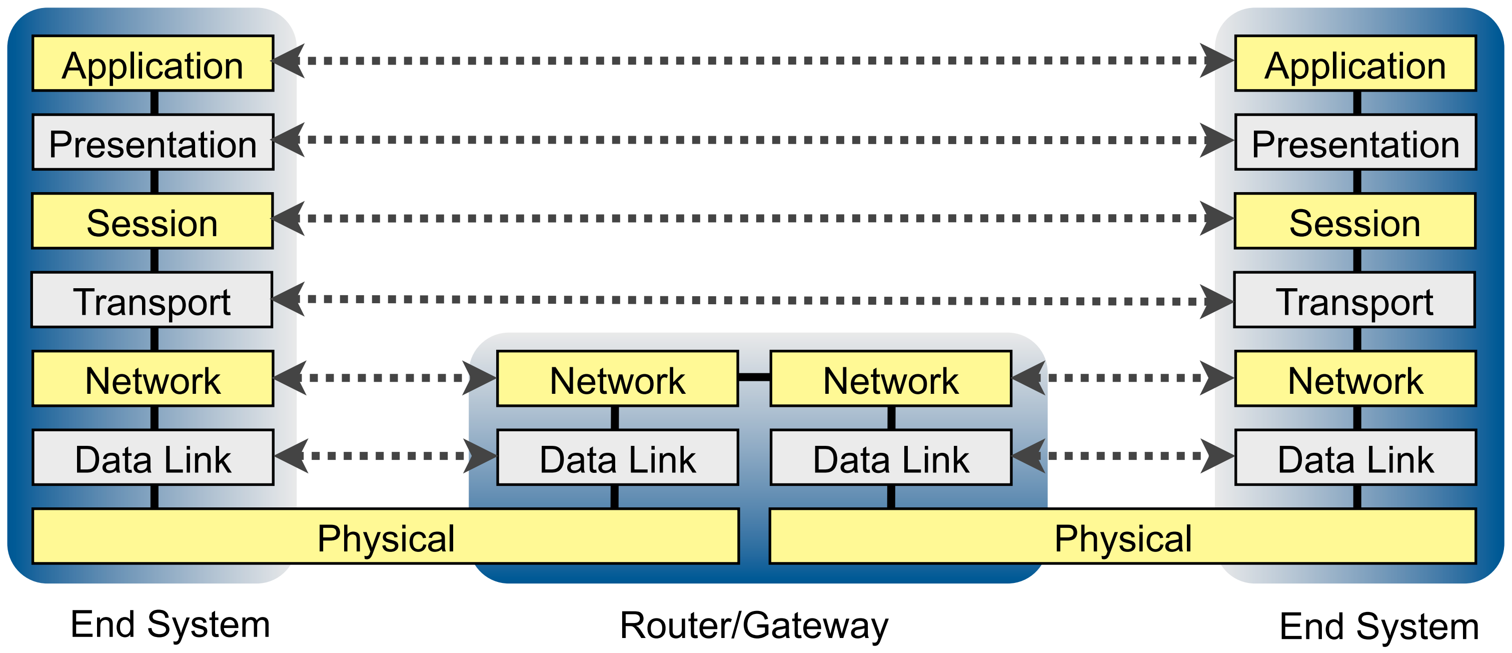
University of Glasgow

Network Layer (1)

Networked Systems 3
Lecture 8

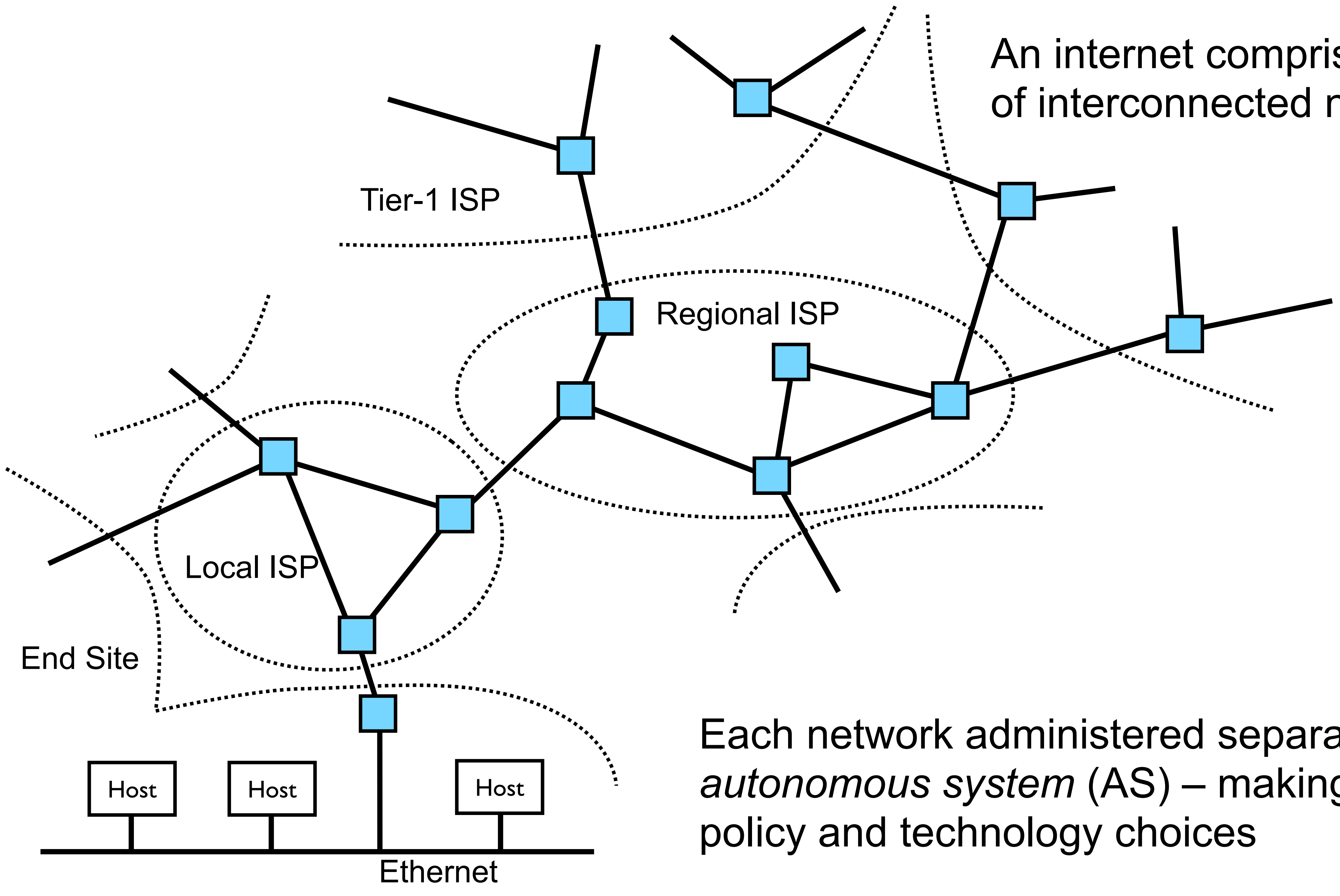
Role of the Network Layer

The network layer is the first end-to-end layer in the OSI reference model



- Responsible for end-to-end delivery of data:
    - Across multiple link-layer hops and technologies
    - Across multiple *autonomous systems*
    - Building an *Internet:* a set of <u>inter</u>connected <u>net</u>works

Interconnecting Networks



An internet comprises a set of interconnected networks

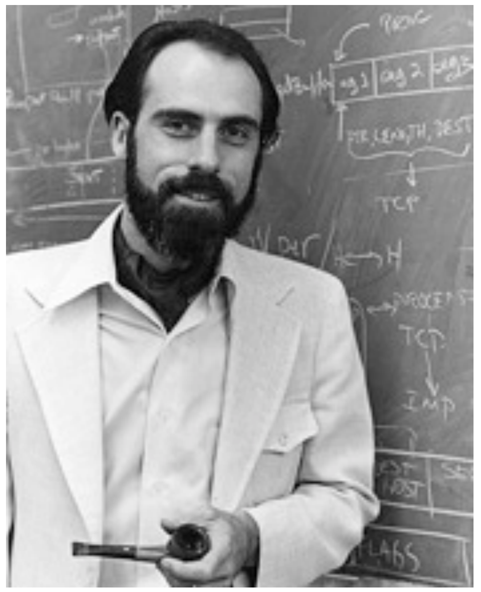
Each network administered separately – an *autonomous system* (AS) – making independent policy and technology choices

Components of *an* Internet

- A common end-to-end network protocol
  - Provide a single seamless service to transport layer
    - Delivery of data packets/provisioning of circuits
    - Addressing of end systems
- A set of *gateway* devices (a.k.a. *routers*)
  - Implement the common network protocol
  - Hide differences in link layer technologies
    - Framing, addressing, flow control, error detection and correction
    - Desire to perform the least amount of translation necessary

The Internet

- The globally interconnected networks running the *Internet Protocol* (IP)
  - Initial design by Vint Cerf and Robert Kahn, 1974
- IP provides an abstraction layer
  - Transport protocols and applications above
  - Assorted data link technologies and physical links below
  - A simple, best effort, connectionless, packet delivery service
  - Addressing, routing, fragmentation and reassembly

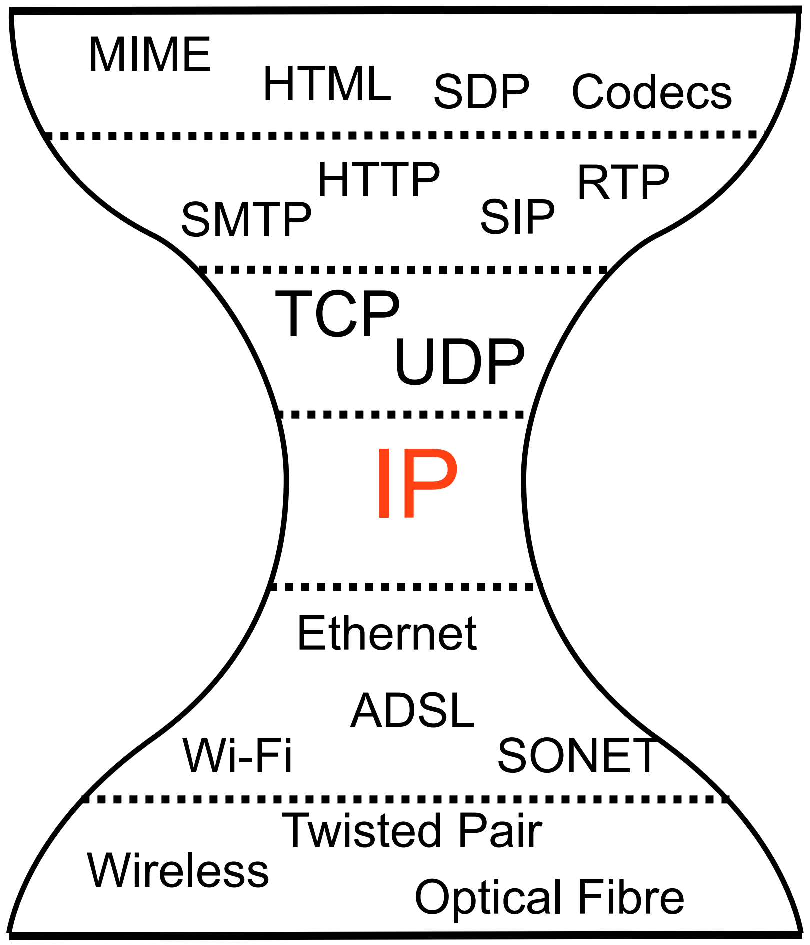

Source: IEEE

Vint Cerf

Source: IEEE

Robert Kahn

Internet Protocol

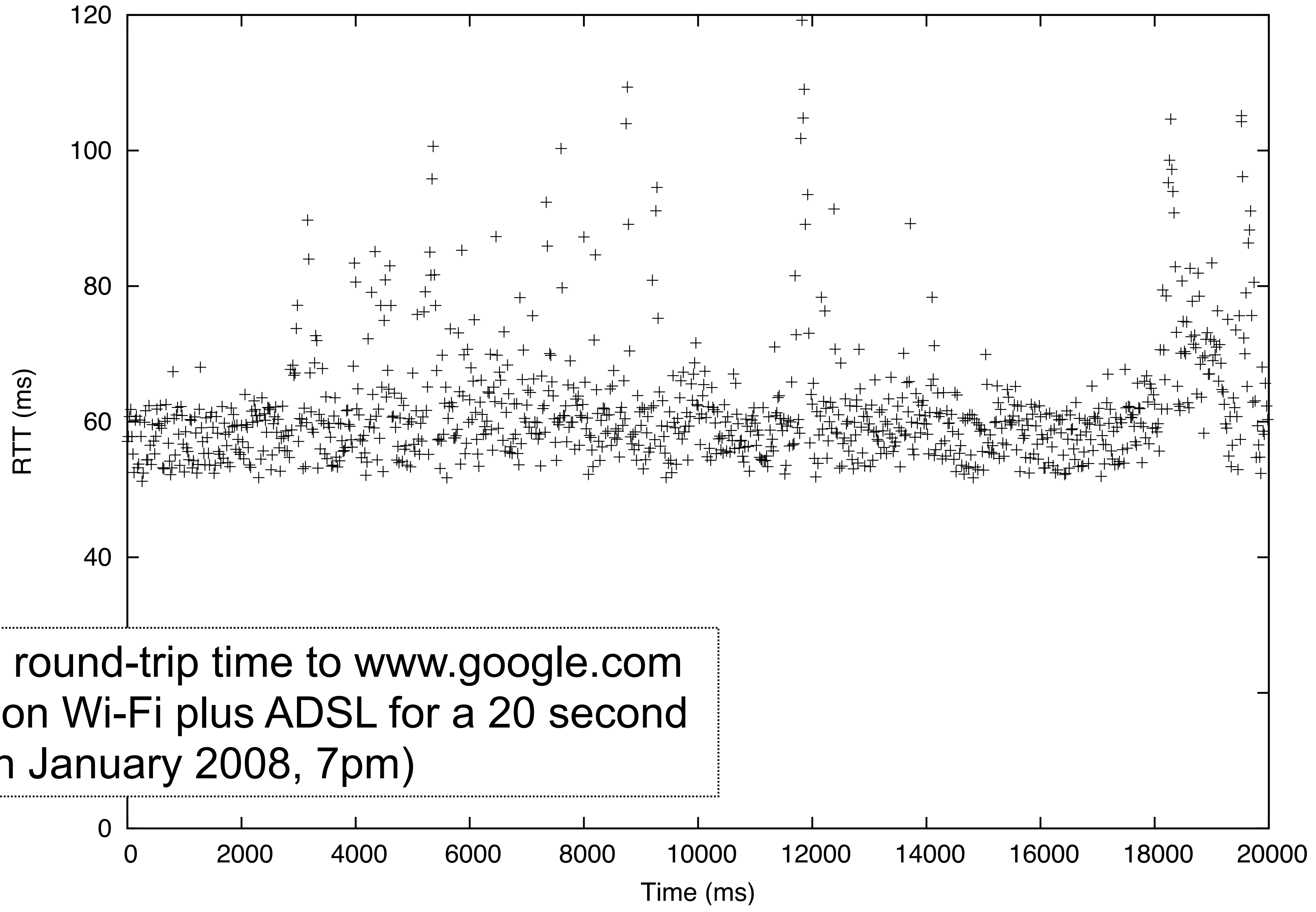


Hour glass Internet model: the IP *network layer* provides a common infrastructure component

IP Service Model

- Best effort, connectionless, packet delivery
  - Just send – no need to setup a connection first
  - Network makes its *best effort* to deliver packets, but provides no guarantees
    - Time taken to transit the network may vary
    - Packets may be lost, delayed, reordered, duplicated or corrupted
    - The network discards packets it can’t deliver
  - Easy to run over any type of link layer
  - Fundamental service: can easily simulate a circuit over packets, but simulating packets over a circuit difficult

Best Effort Packet Delivery



Measured round-trip time to www.google.com from host on Wi-Fi plus ADSL for a 20 second period (4th January 2008, 7pm)

Internet Protocol

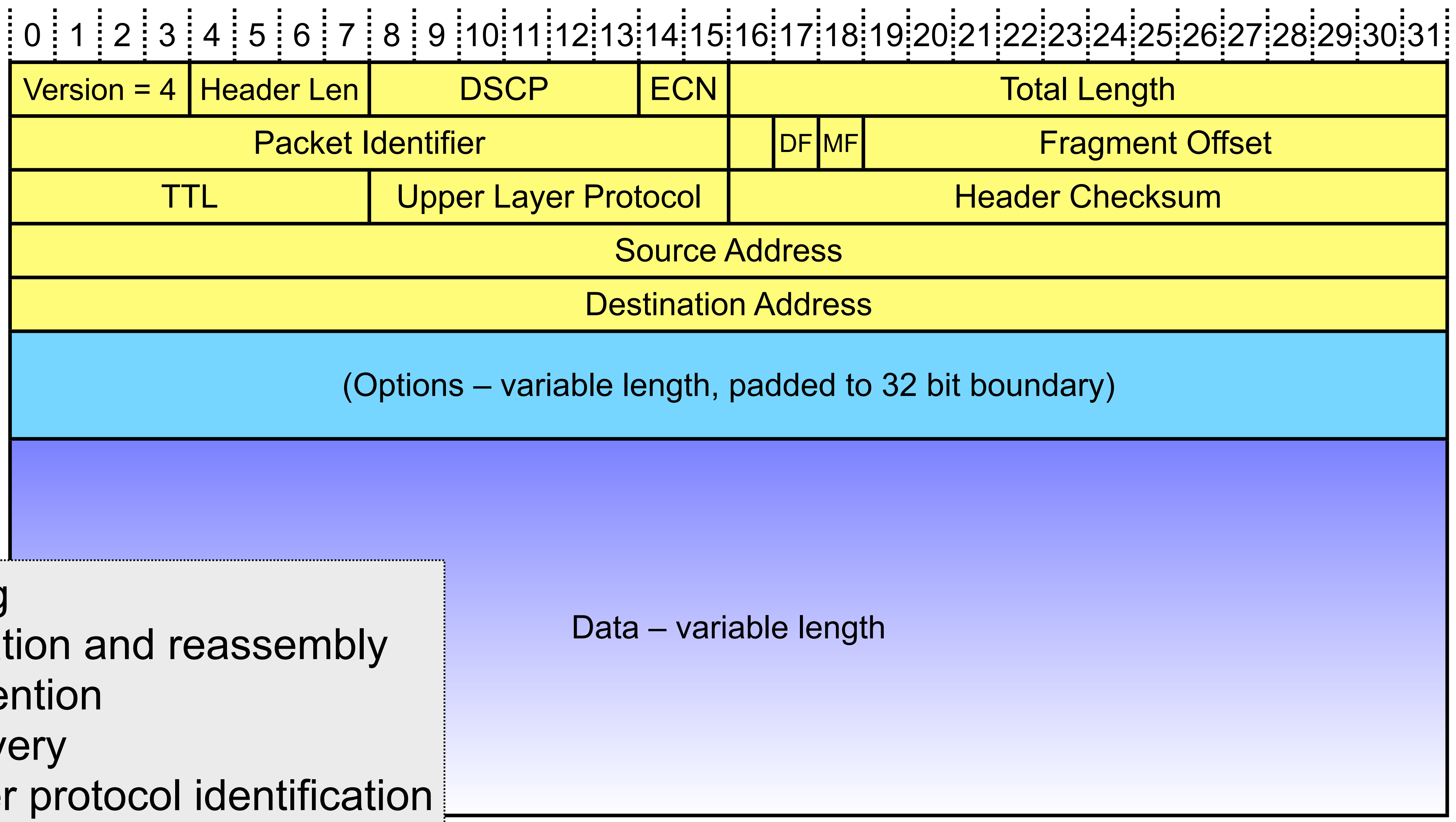
- Two versions of IP in use:
  - IPv4 – the current production Internet
  - IPv6 – the next generation Internet

  - IPv5 was assigned to the Internet Stream Protocol
    - An experimental multimedia streaming protocol developed between 1979 and 1995 [http://www.ietf.org/rfc/rfc1819.txt], but no longer used

IPv4

- The Internet is the global network running IPv4
  - Introduced in 1983
  - Initial implementation on BSD Unix, but now runs on *everything* from mainframes to PCs, phones, games, TV set-top boxes, etc…

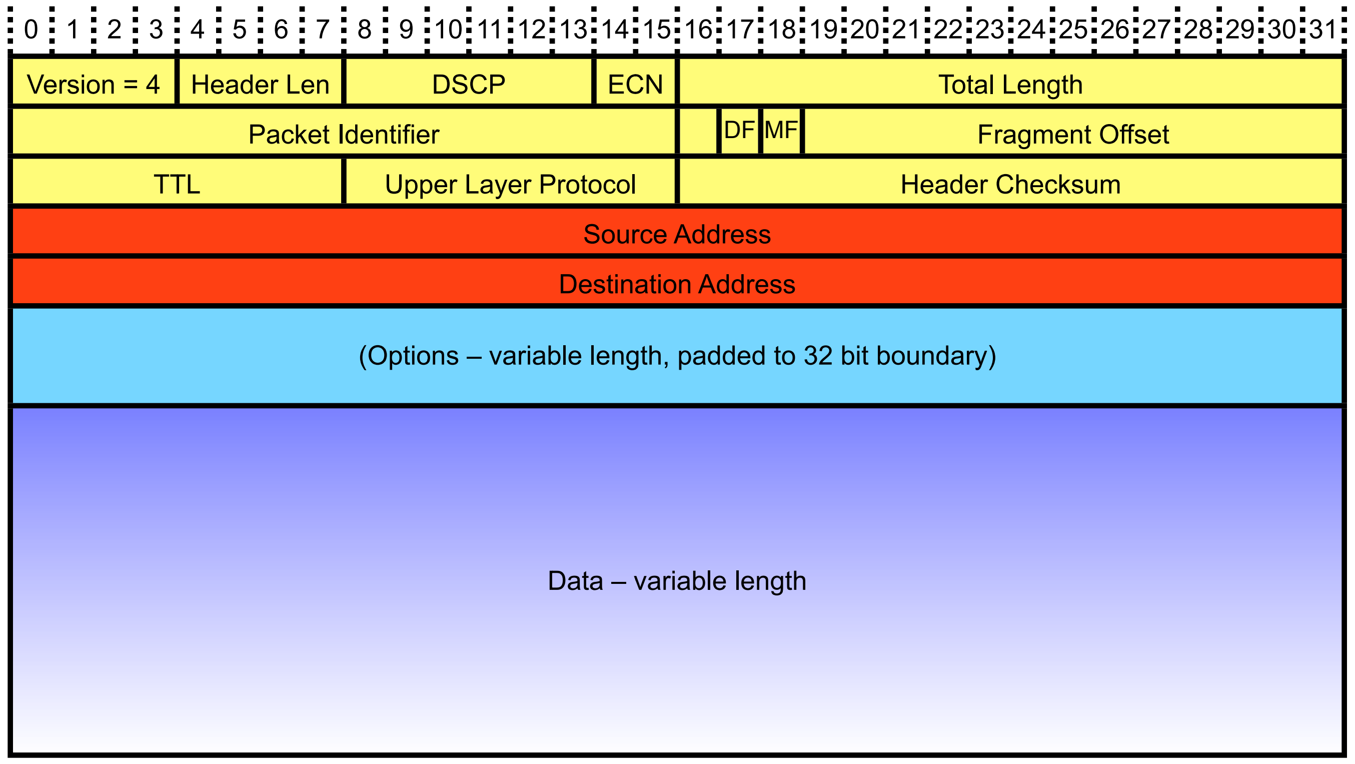
IPv4 Packet Format



Addressing
Fragmentation and reassembly
Loop prevention
Error recovery
Upper layer protocol identification
Differentiated services
Explicit congestion notification

IPv4 Addresses

- 32 bit destination and source addresses sent in every packet
  - Example: 130.209.247.112
- Intended to be globally unique, not enough addresses available
  - Details in next lecture...

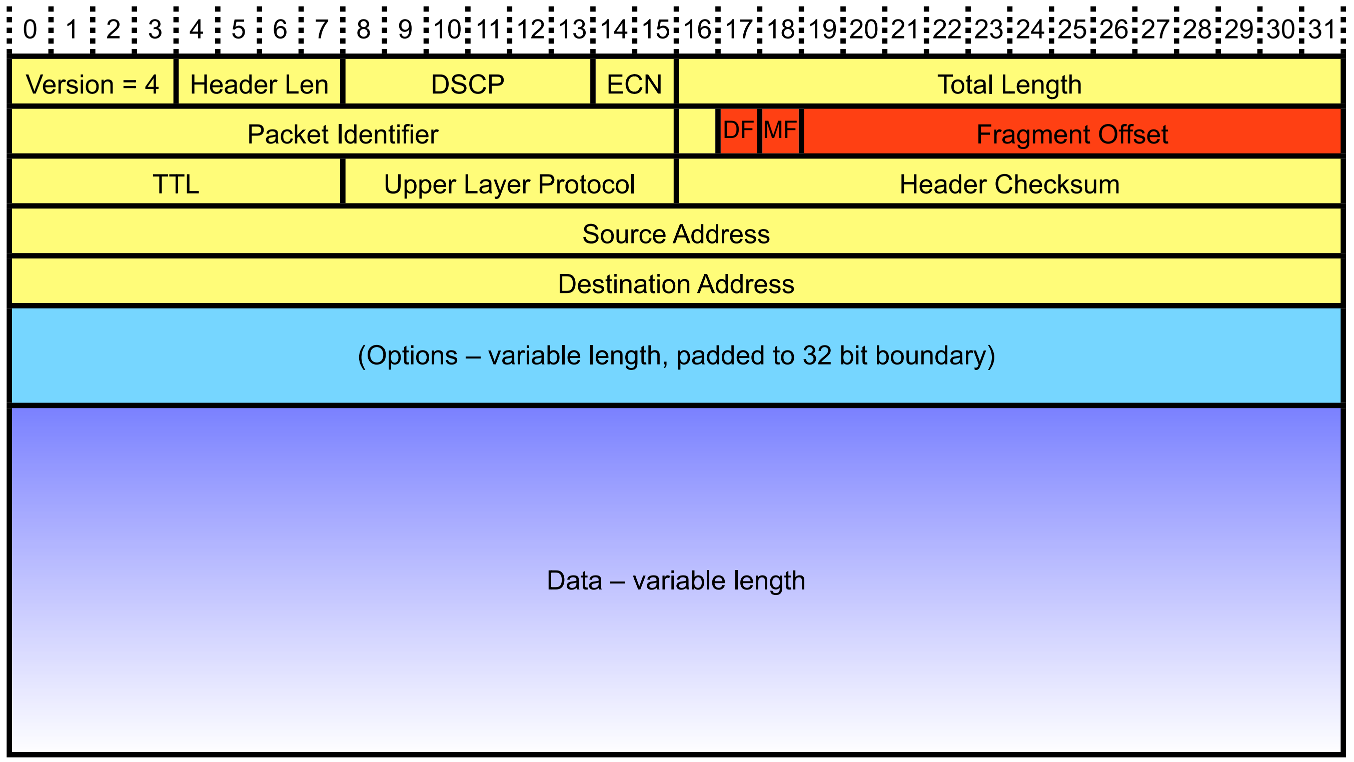


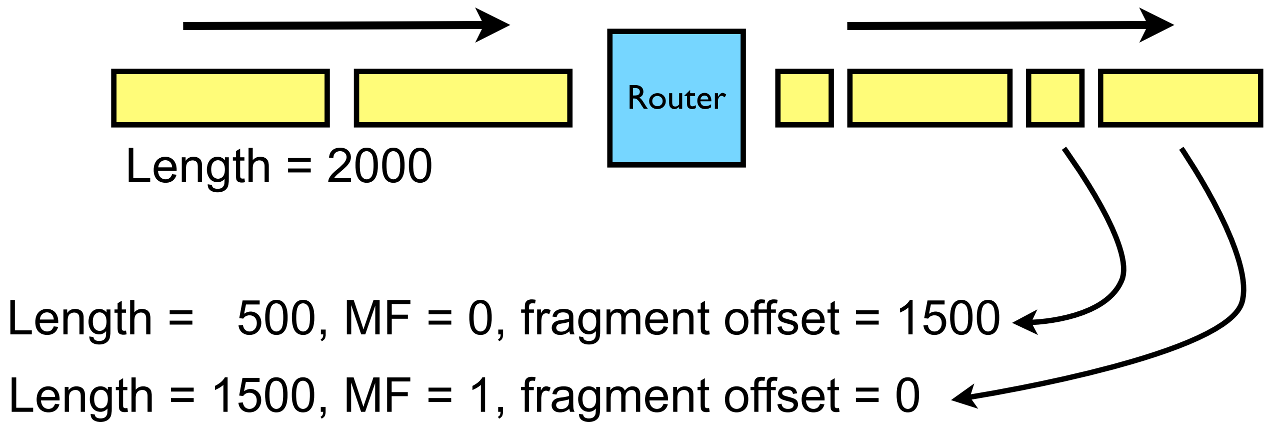
Aside: Address Resolution

- Network layer handles wide area routing, but how to communicate across a local link?
  - Know the network layer address you wish to talk to, but need to find the corresponding link layer address
  - Use link layer broadcast for queries:
    - Request: "I have network address N_s, link address L_s, and want talk to host with network address N_d" gets response "I have network address N_d and link address L_d".
    - Address Resolution Protocol (ARP) – a general protocol, not IP specific

IPv4 Fragmentation

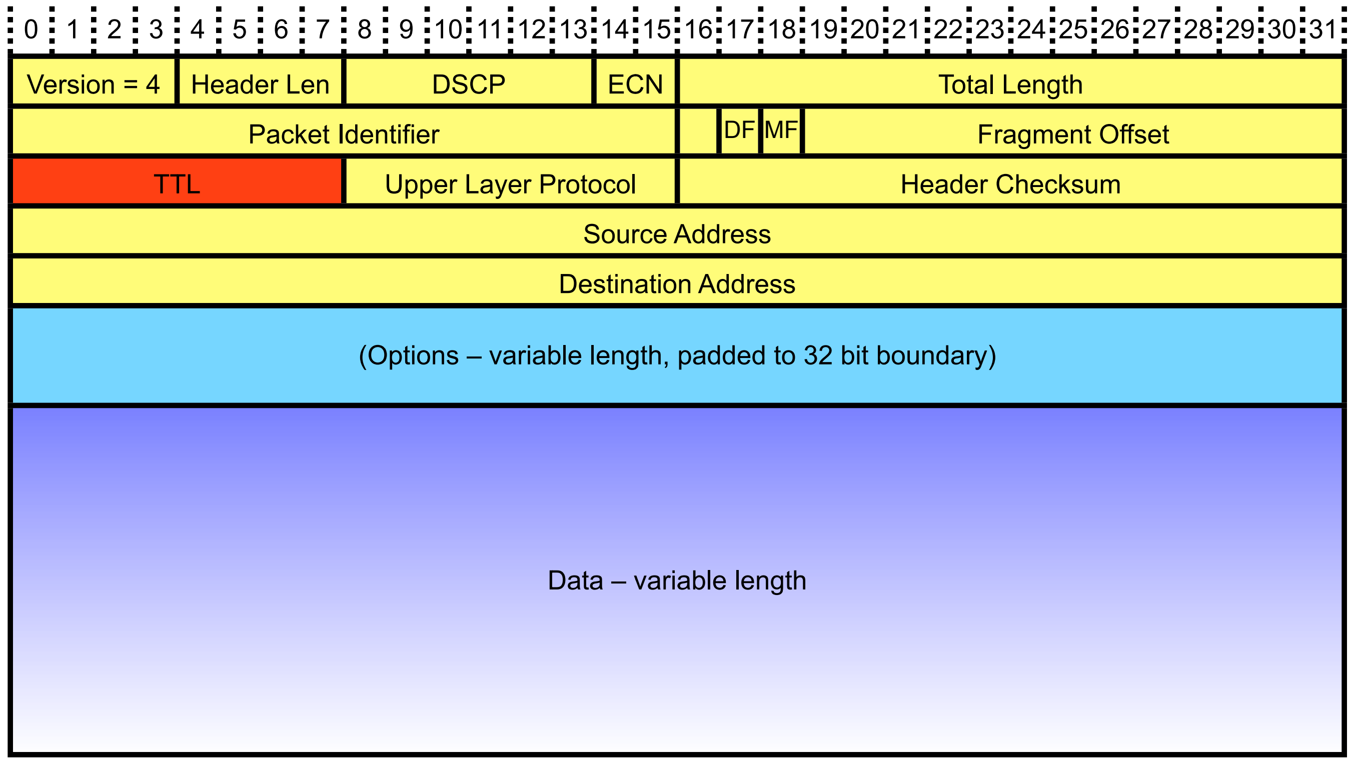
- Hosts may send packets with length up to 65535 bytes
- Routers will fragment if link MTU too small
    - MF set if more fragments follow; reconstruct using fragment offset
    - Unless DF bit is set, in which case packet discarded and an error is returned




Router
Length = 2000
Length = 500, MF = 0, fragment offset = 1500
Length = 1500, MF = 1, fragment offset = 0

IPv4 Loop Prevention

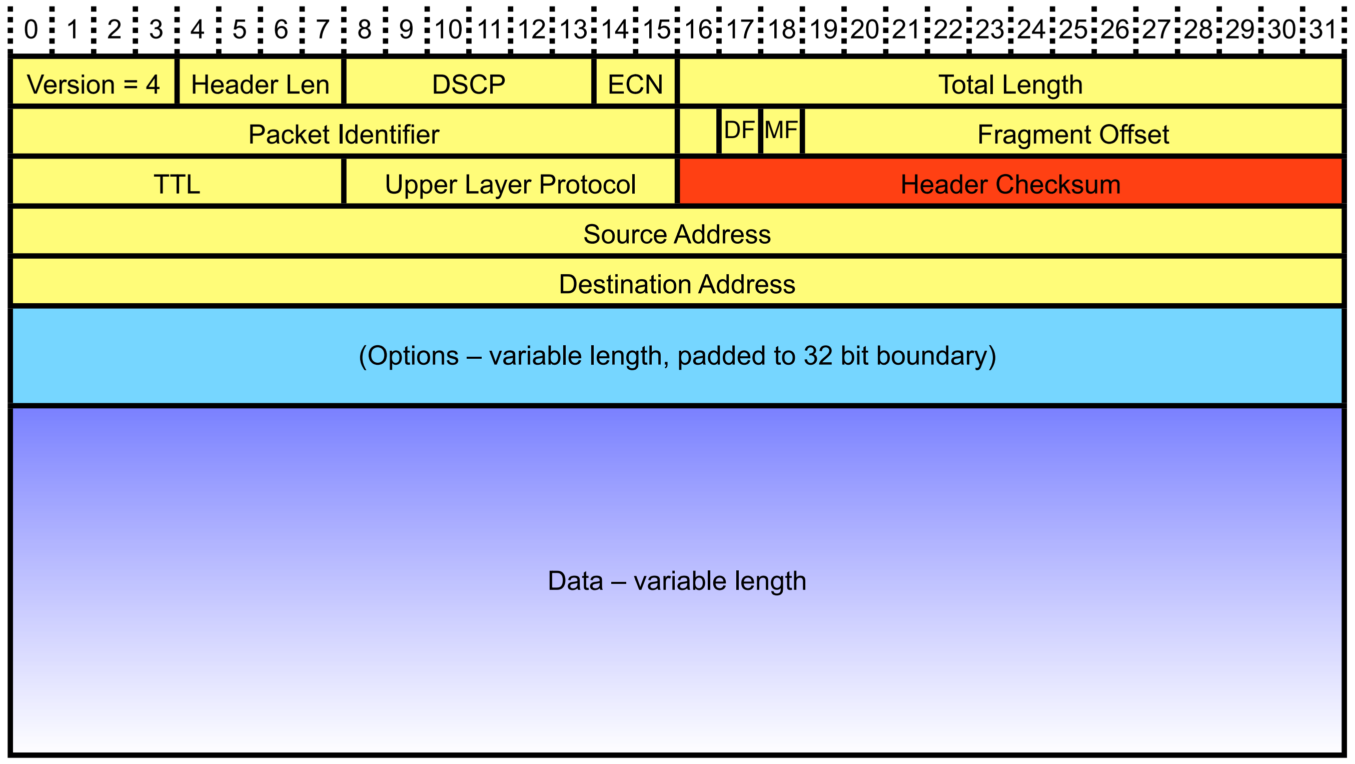
- Packets have *time to live* (TTL) field
    - Set to some value (typically 64) when the packet is sent
    - Decrease by one at each router traversed, discard packet if TTL reaches zero



- Prevents packets from looping forever if a routing problem causes a loop

IPv4 Error Recovery

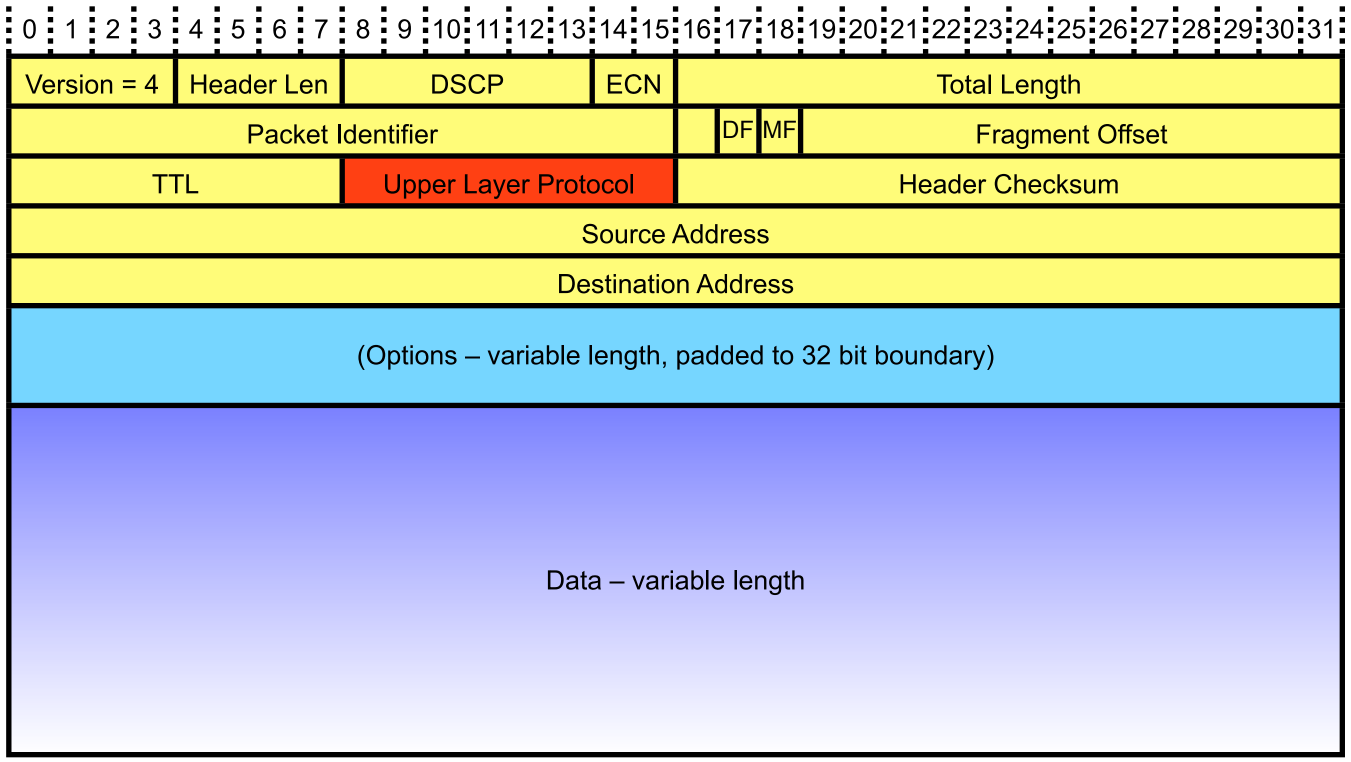
- Checksum to detect transmission errors
  - In the IP header only – data not protected (must be protected by upper layer protocol, if needed)
  - Generally redundant, since link layers often include a checksum over the entire packet



IPv4 Upper Layer Protocol ID

- Indicates what content follows the header

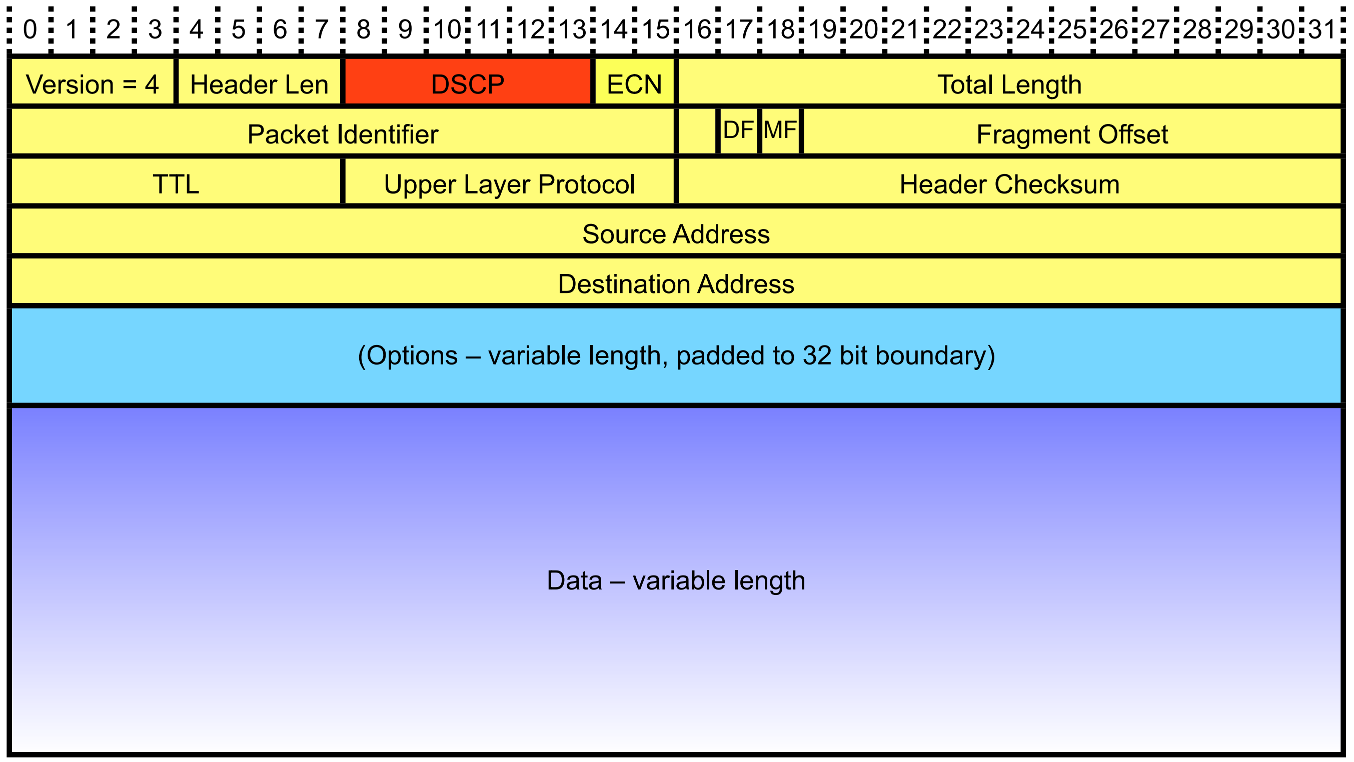
6	TCP
17	UDP
33	DCCP
…	…



- Identify transport layer protocol/format of the data
  - Used by end systems to pass packets to the correct upper layer protocol

IPv4 Differentiated Service

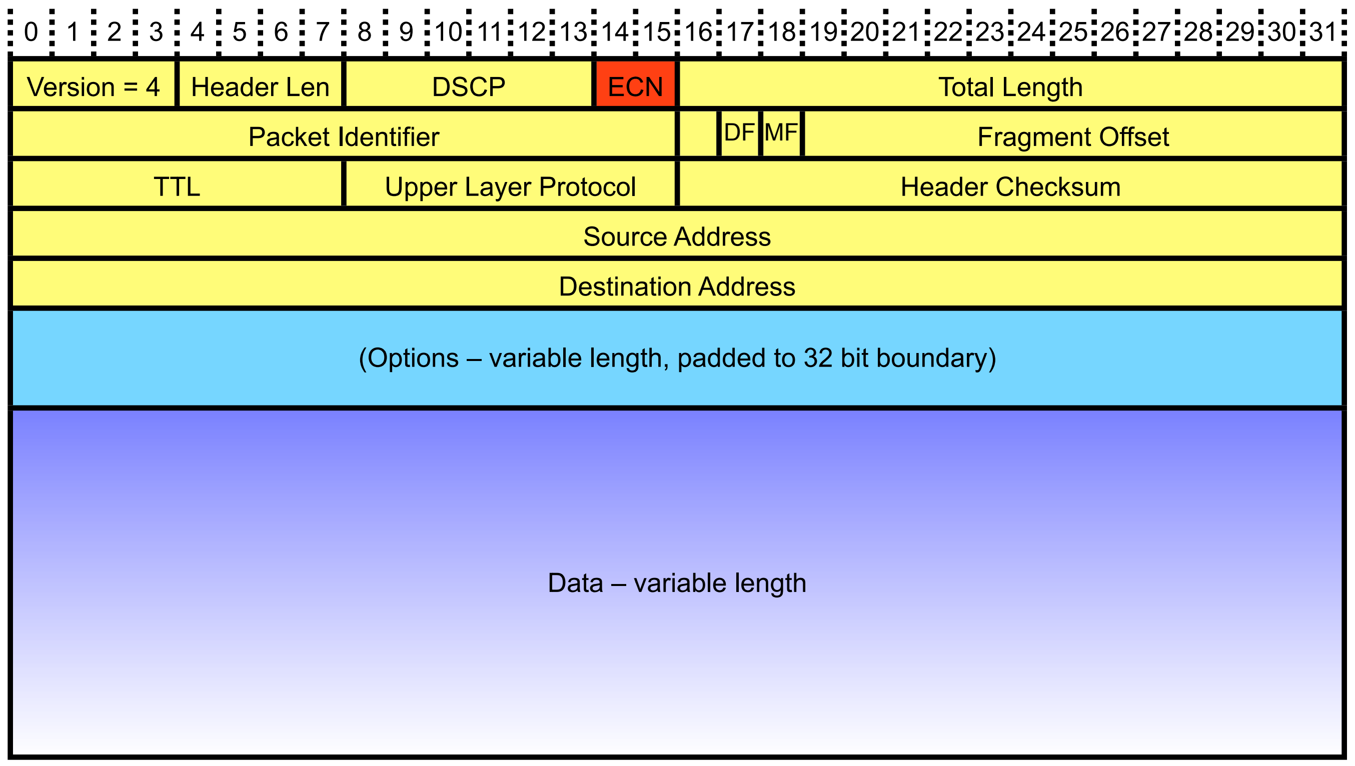
- End systems can request *differentiated service* from the network
  - E.g. request low latency for VoIP
  - Accounting issues: who is allowed to request differentiated service? do they have to pay extra for it?



- A hint to the network, not a guarantee
  - Police at the edges, to avoid state in the core
  - Scalability, compared to explicit reservation protocols, and why it's important

IPv4 ECN

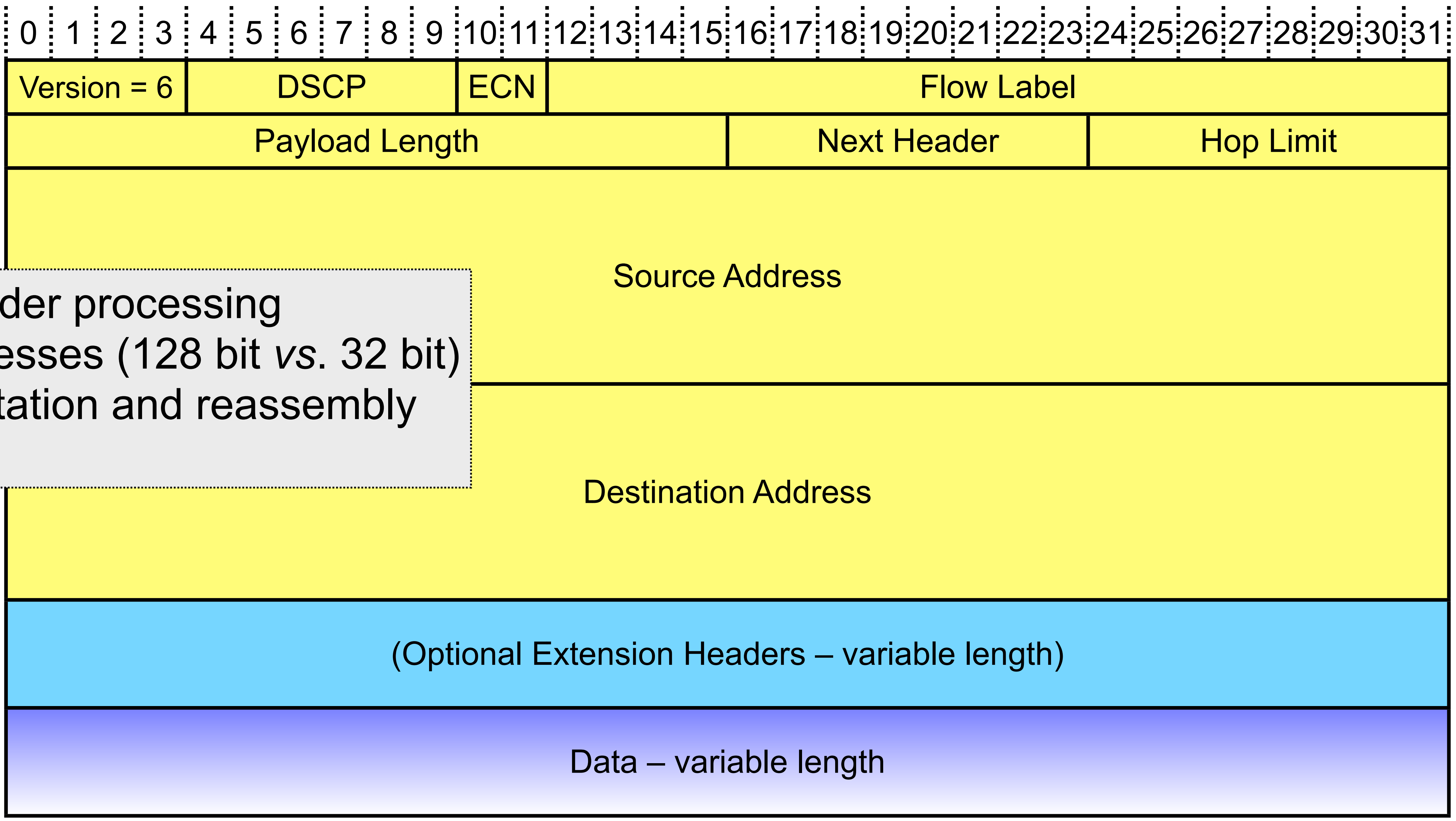
- Routers can set *explicit congestion notification* bits to signal that congestion is occurring
    - End systems use this as a hint to reduce their sending rate
    - If congestion doesn't ease, the routers start to discard packets



IPv6

- A revision to IP, intended to be the long term replacement for IPv4
  - Primary goal: increase the size of the address space, to allow more hosts on the network
  - Beginning to see wide implementation
    - Current version of specification published in 1998
    - Implemented in Windows XP, Windows Vista, MacOS X, Linux, FreeBSD, Symbian, …
    - Router implementations becoming more widespread (present in Cisco and Juniper backbone routers, but yet not common Linksys and D-Link home routers...)

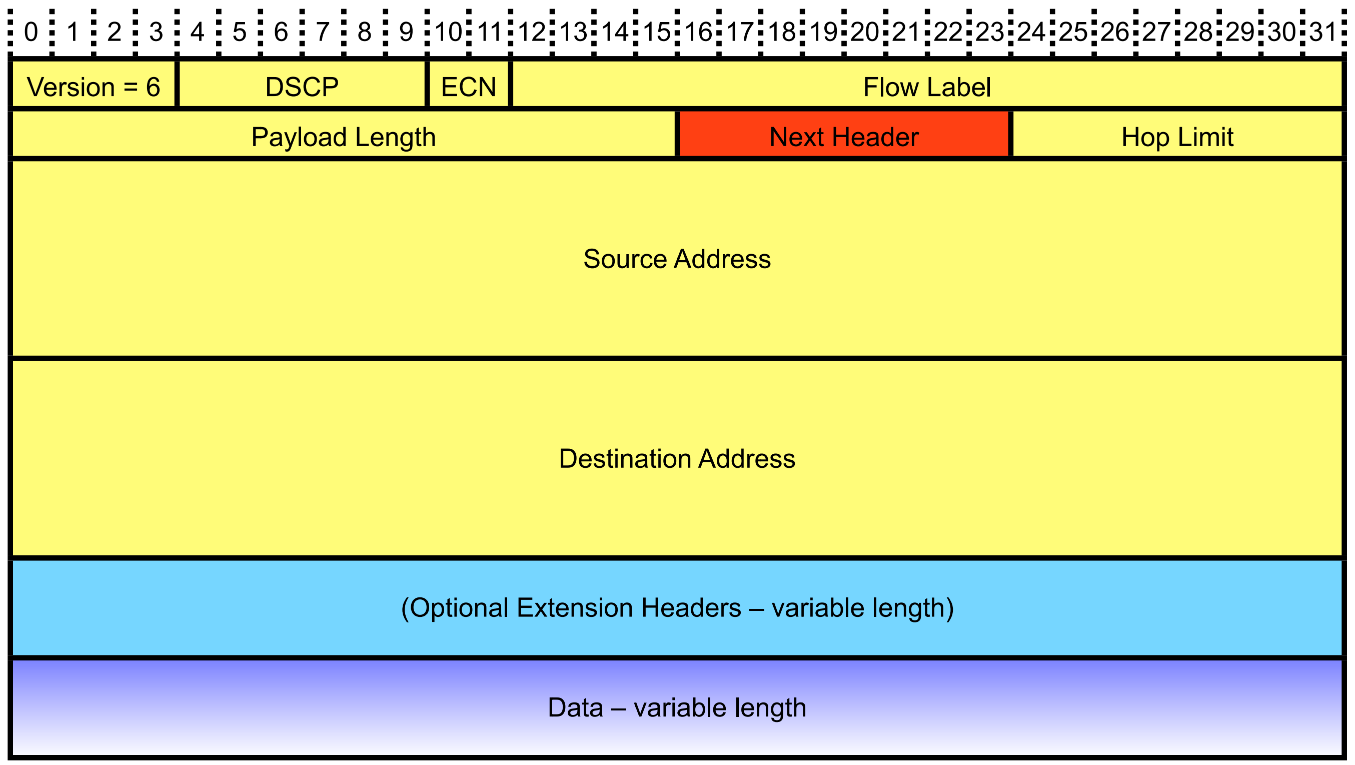
IPv6 Packet Format



Simpler header processing
Larger addresses (128 bit *vs*. 32 bit)
No fragmentation and reassembly
Flow labels

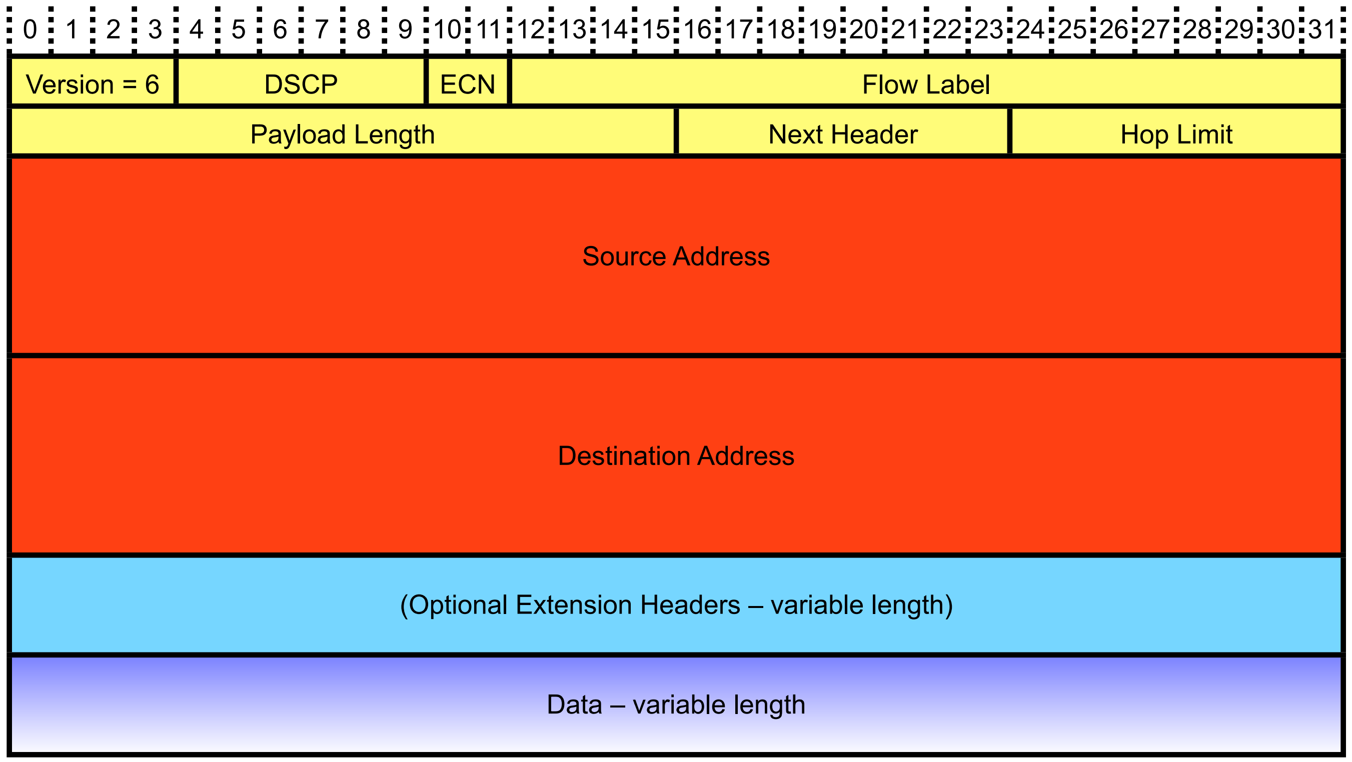
IPv6 Simplified Header

- Header has fewer fields, a more regular structure
  - Much easier to process at very high speeds
  - Now supports extension headers chained after IPv6 header – the *next header* field identifies
  - Each extension indicates if router must process it, or can ignore if not understood
  - Eventually the *next header* will be a transport protocol header



IPv6 Addresses

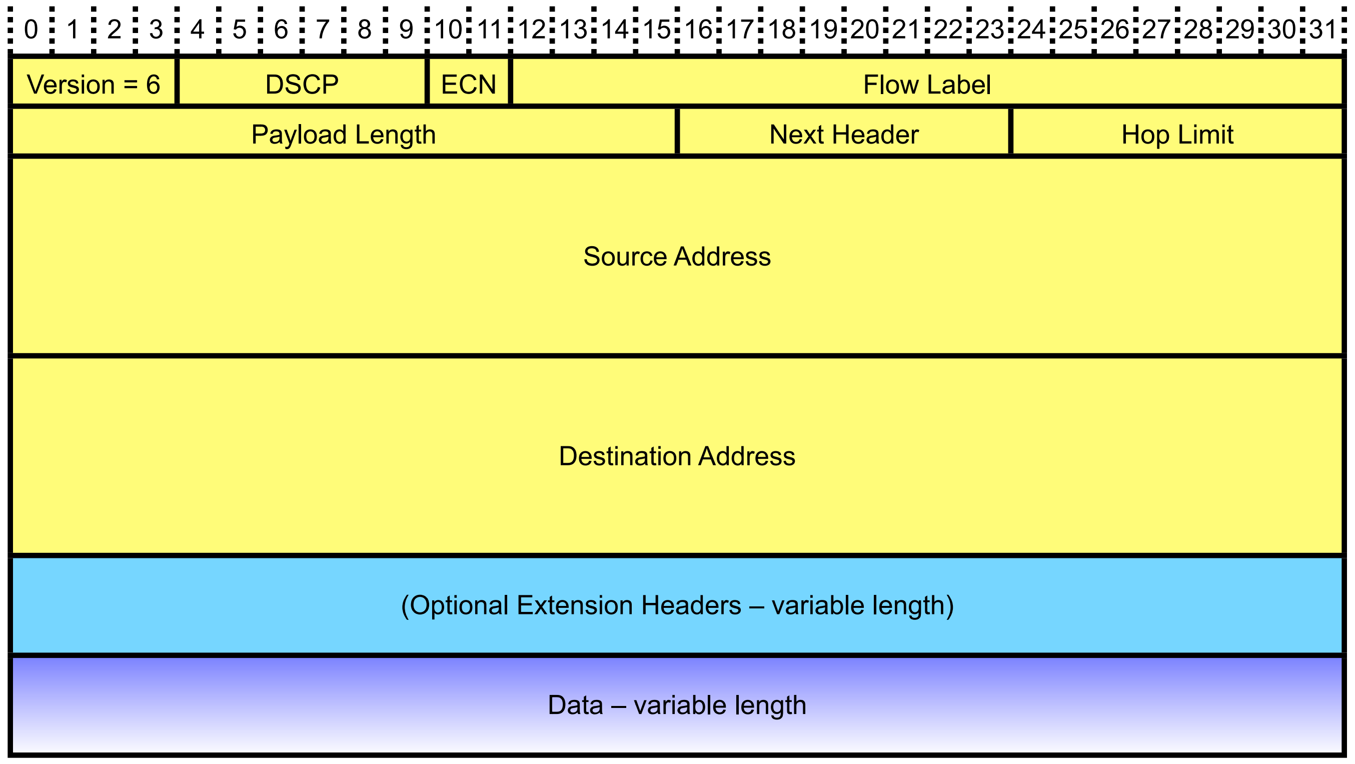
- Source and destination addresses are 128 bits



- Can give every host a globally unique address
  - Greatly simplifies applications
  - Addressable does not necessarily mean reachable (due to firewalls)
  - Privacy features lets hosts use random addresses, to give the same degree of privacy as IPv4 (i.e. very little, since your ISP is required by law to record the address that is assigned to you at all times, and keep connection logs)

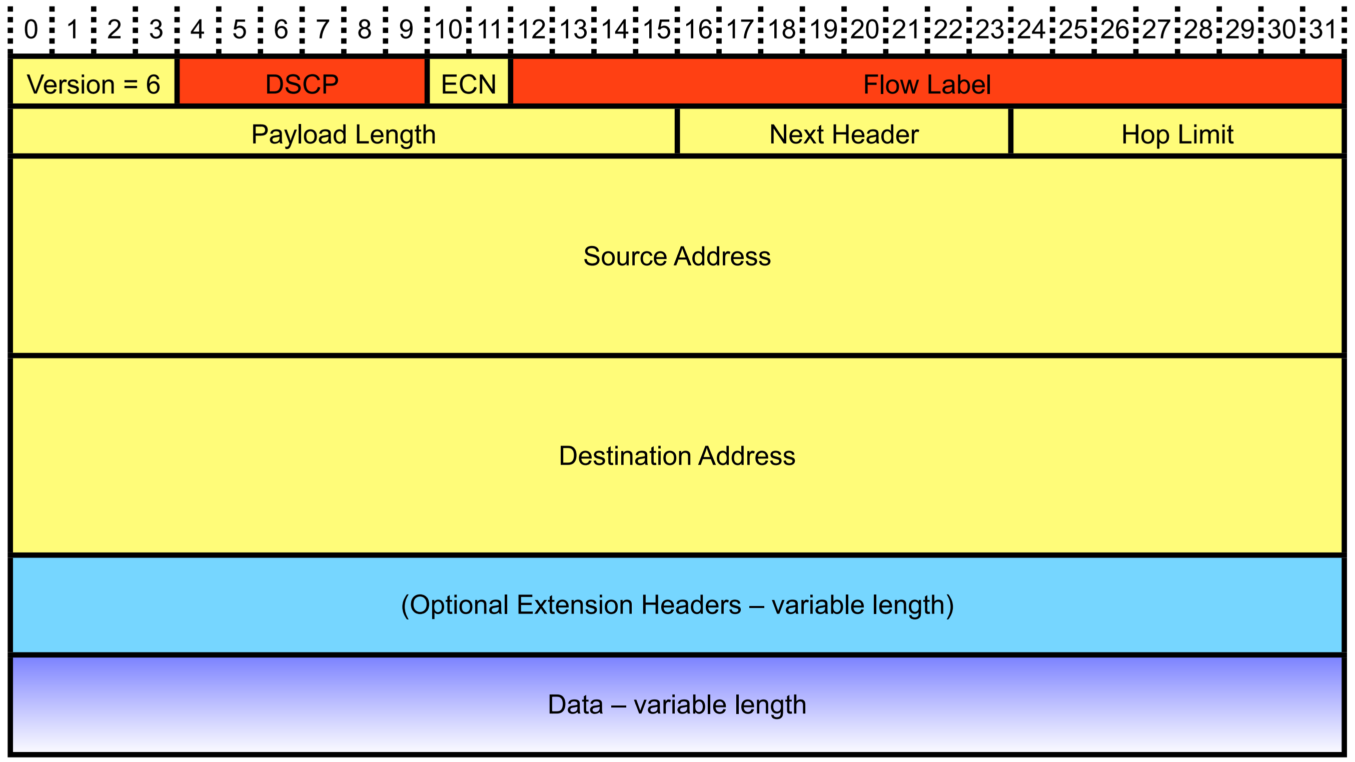
IPv6 Fragmentation

- Unlike IPv4, IPv6 routers don't fragment packets
  - Return a ICMP error and discard the large packet instead
  - Hosts required to recover from this, generating smaller packets
  - The end-to-end principle, used to simplify router implementations



IPv6 Quality of Service

- Differentiated services work exactly the same as IPv4
- *Flow Label* field identifies data flow independently of transport protocol



  - Routers no longer have to parse the header of TCP, UDP, DCCP, etc. to group packets into a flow; simplifies differentiated services for high-speed operation
  - End systems can group several transport layer connections into a single flow

IPv4 or IPv6?

- Not yet clear if IPv6 will be widely deployed
- But, simple to build applications that work with both versions of IP
  - DNS query using `getaddrinfo()` will return IPv6 address if it exists, else IPv4 address; all other socket calls use the returned value
  - Write new code to support both IPv6 and IPv4

Questions?