

CSE 123b

Communications Software

Spring 2003

**Lecture 2: Internet architecture and
Internetworking**

Stefan Savage

Some history...

- 1968: DARPAnet (precursor to Internet)
 - ♦ Bob Taylor, Larry Roberts create program to build first wide-area packet-switched network
 - ♦ Why?
- 1978: new networks emerge
 - ♦ SATNet, Packet Radio, Ethernet
 - ♦ All islands
- Big question: how to connect these networks?

Plug: “Where Wizards Stay Up Late” by Hafner and Lyon is the best account of early Internet History I’ve seen.

Primary Goal: Connect Stuff

- “Effective technique for multiplexed utilization of existing interconnected networks” – David Clark
 - ♦ **Minimal** assumptions about underlying networks
 - › No support for broadcast, multicast, real-time, reliability
 - › Extra support could actually get in the way (X.25 example)
 - ♦ Packet switched, store and forward
 - › Matched application needs, nets already packet switched
 - › Enables **efficient resource sharing**/high utilization
 - ♦ “Gateways” interconnect networks
 - › Routers/Switches in today’s nomenclature

Why is this hard?

Heterogeneity

- ◆ Addressing
 - » Each network media has a different addressing scheme; routing protocol
- ◆ Bandwidth
 - » Modems to terabits
- ◆ Latency
 - » Seconds to nanoseconds
- ◆ Packet size
 - » Dozens to thousands of bytes
- ◆ Loss rates
 - » Differ by many orders of magnitude
- ◆ Service guarantees
 - » Send and pray vs reserved bandwidth

How to connect different networks?

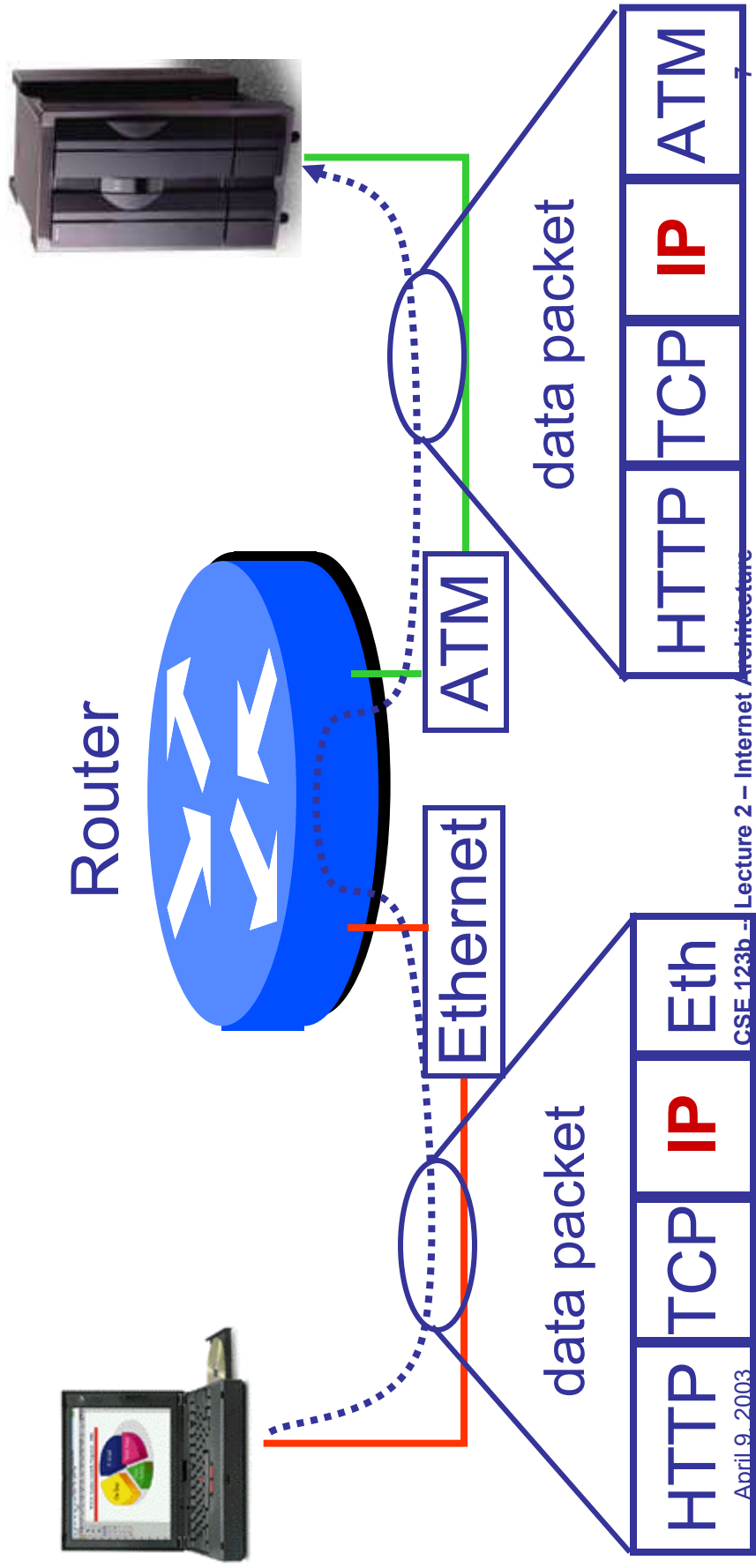
- **Monopoly**
 - ♦ Re-engineer network to use a single set of protocols everywhere
 - ♦ Economic cost
- **Translation Gateways**
 - ♦ Translates directly between different network formats
 - ♦ $O(n^2)$ complexity (n is # of protocols)
 - ♦ May not be able to translate perfectly (QoS)
- **Indirection Gateways**
 - ♦ Translates between local network format and universal “intermediate” format
 - ♦ $O(n)$ complexity
 - ♦ May not take advantage of features in underlying network
- Note impact of economics on decision. Engineering not science.

Internetworking

- Cerf & Kahn74, “A Protocol for Packet Network Intercommunication”
 - ◆ Foundation for Internetworking and hence, the Internet
 - ◆ We’ll talk about the reliability issues later
- All packets use a common Internet Protocol
 - ◆ Any underlying data link protocol
 - ◆ Any higher layer transport protocol

How IP works

Separate physical networks communicate to form a *single* logical network



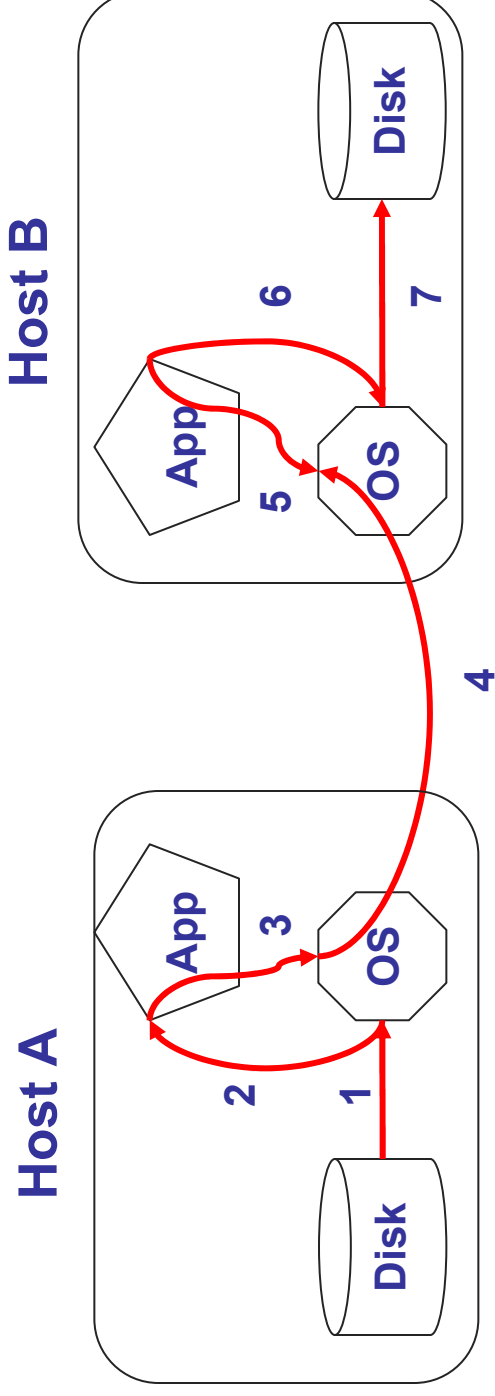
What should the Internet Protocol do?

- Packetization?
 - Addressing?
 - Error detection?
 - Reliable transmission?
 - Packet sequencing?
 - QoS?
 - Security?
-
- Decisions informed by the “End-to-End Principle”

Saltzer et al 84: End-to-End Principle

- **Key question:** Where should functionality be placed in a communications system?
- **End-to-end argument**
 - ◆ Functionality should be implemented at a lower layer iff it can be **correctly** and **completely** implemented there
 - ◆ Incomplete versions of a function can be used as a performance enhancement, but not for correctness
- **Early, and still relevant, example**
 - ◆ ARPAnet provided reliable link transfers between switches
 - ◆ Packets could still get corrupted on host-switch link, or inside of the switches
 - ◆ Hence, still need reliability at higher layers

Example: Reliable File Transfer



- Where can data be corrupted?
- How to tell if data has been corrupted?
- Is there any value in lower-layer reliability?

Example: Reliable File Transfer

- From server disk over network to client disk
- Many places where errors can be introduced
 - ◆ Disk can introduce bit errors
 - ◆ Host I/O bus can introduce bit errors
 - ◆ Packets can be corrupted, dropped, reordered at any node
- Conclusion
 - ◆ Still need integrity checks on entire file, at application level, not per packet or per hop
 - ◆ Impossible to design “perfect” layers because perfect requires support from higher layers

Internet architecture

- Impose few demands on network
 - ◆ Make few assumptions about what network can do
 - ◆ No QoS, no reliability, no ordering, no large packets
 - ◆ No persistent state about communications
- Manage heterogeneity at hosts
 - ◆ Adapt to underlying network heterogeneity
 - ◆ Re-order packets, detect errors, retransmit lost messages, etc.
 - ◆ Persistent network state only kept in hosts (fate-sharing)
- Service model: *send and pray*

So what does IP do?

- **Addressing**
 - ◆ How do I name the destination?
- **Fragmentation**
 - ◆ How do I handle packets that are larger than the next hop can accept (e.g. FDDI's maximum packet is 4500 bytes while Ethernet is 1500 bytes)
- **Error detection**
 - ◆ How do I know if a packet got corrupted?
- **Potpourri**
- **Routers forward packets to next hop**
- **They do not:**
 - ◆ Detect data corruption, packet loss, packet duplication
 - ◆ Reassemble or retransmit packets

Addressing

- Hierarchical addressing
 - ◆ Global inter-network address
 - ◆ Local network-specific address



Original ARPANET address format

- Why hierarchical?
- Assumptions about networks?

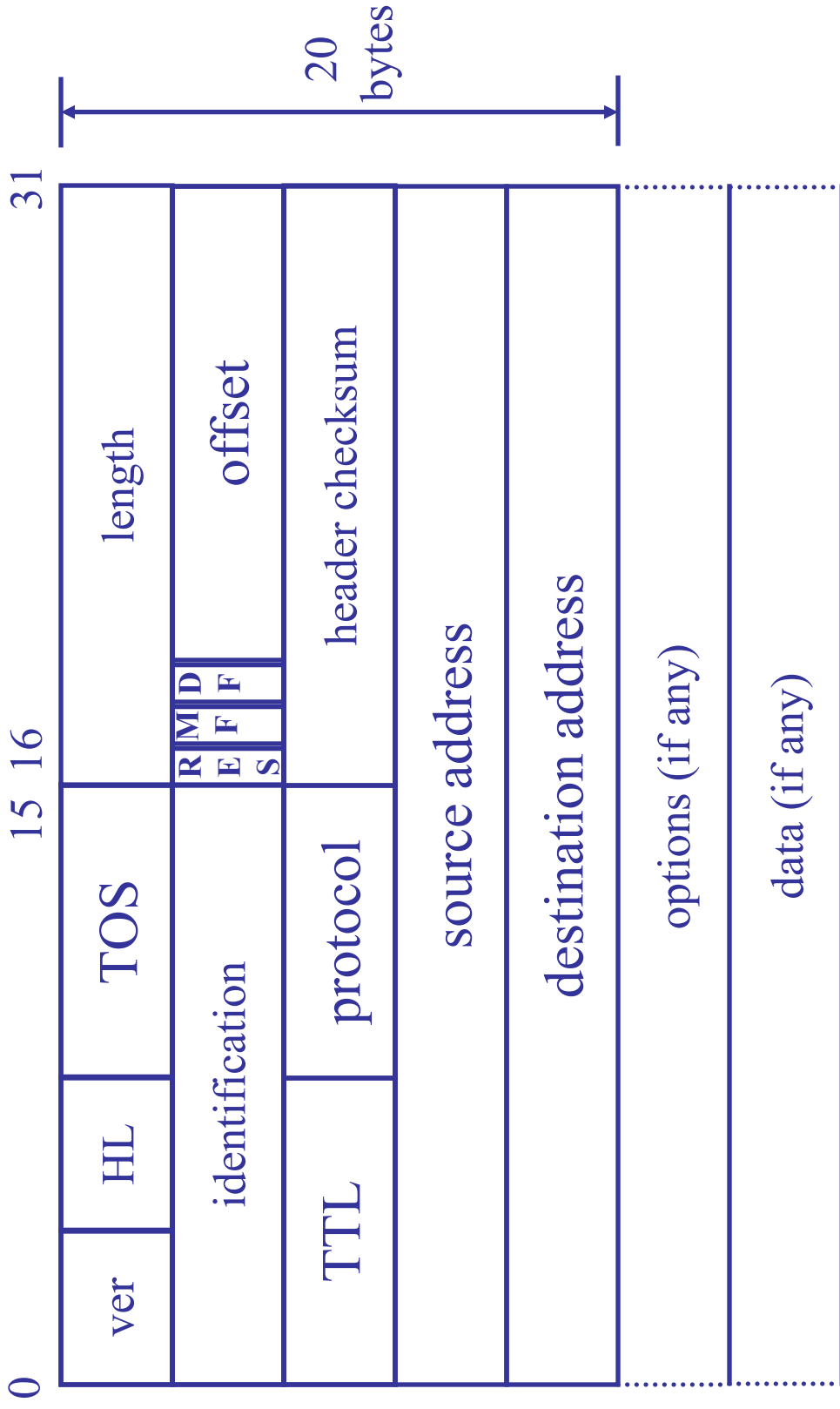
Fragmentation

- In a router each link may have a different *Maximum Transmission Unit (MTU)* – the largest packet it can transmit
 - ♦ Ethernet: ~1500 bytes
 - ♦ FDDI: ~4500 bytes
- Router needs to forward a packet that is too big for the next link it must cross
 - ♦ Router breaks up single IP packet into two or more smaller IP packets
 - ♦ Each fragment is labeled so it can be correctly reassembled
 - ♦ Those fragments can, in turn, be fragmented by later routers
- End host receives fragments and reassembles them into original packet

Error detection

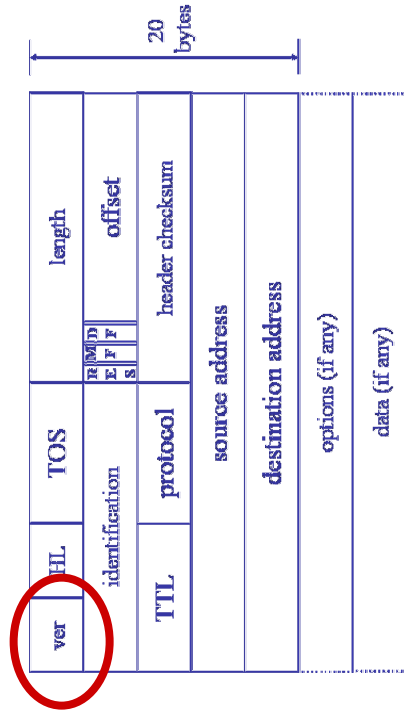
- **Bit errors**
 - ◆ Data-link layer (e.g. Ethernet) generates a Cyclic Redundancy Check (CRC) for each packet
 - » When packet is received by router or host, it checks packet against CRC for errors
 - » Why isn't this enough?
 - ◆ Network-layer (IP) checksum written by sender
 - ◆ Checked at each hop and by receiver
 - » Why not just check at the receiving host?
- **Packet losses**
 - ◆ Not part of IP, we'll deal with this next time

Today's IP Packet Header



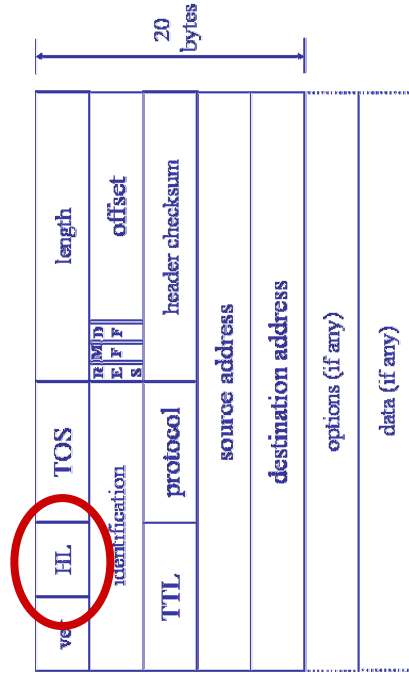
Version field

- Which version of IP is this?
 - ◆ Plan for change
 - ◆ Very important!
- Current versions
 - ◆ 4: most of Internet
 - ◆ 6: new protocol with larger addresses
 - ◆ What happened to 5? Standards body politics.



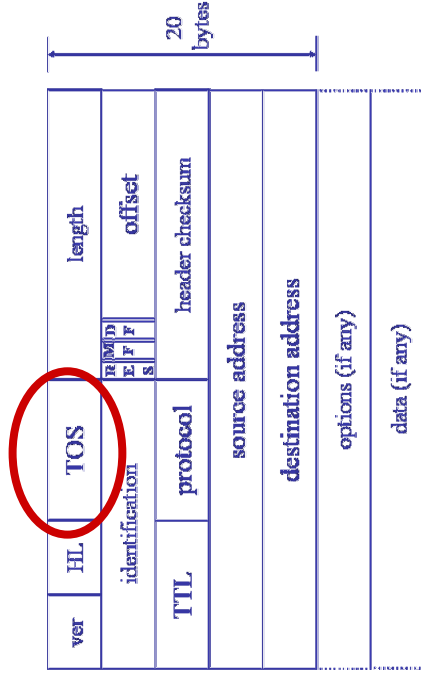
Header length

- How big is IP header?
 - ◆ In # of 32bit words
 - ◆ Variable length
 - » Options
 - ◆ Engineering consequences of variable length...
- Most IP packets are 20 bytes long



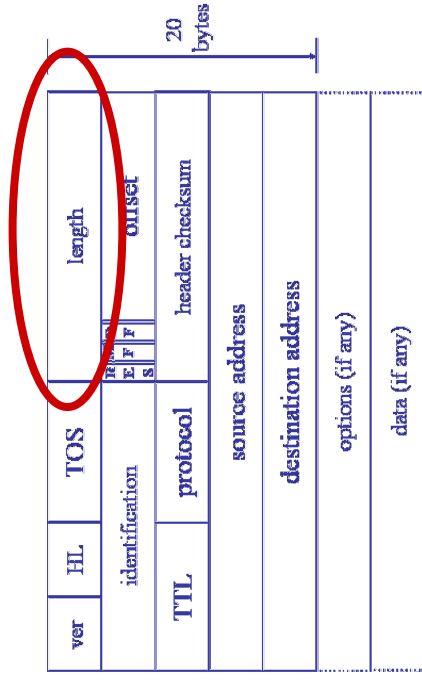
Type-of-Service

- How should this packet be treated?
 - ◆ Care/don't care for delay, throughput, reliability, cost
 - ◆ How to interpret, how to apply on underlying net?
 - ◆ Largely unused until 2000



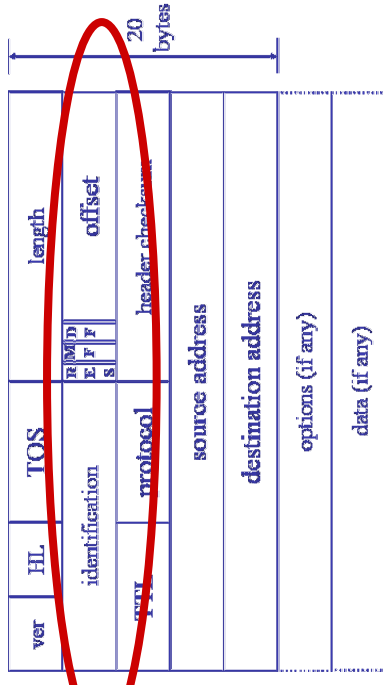
Length

- How long is whole packet in bytes/octets?
 - ◆ Includes header
 - ◆ Limits total packet to 64K
 - ◆ Redundant?



Fragmentation

- Sender writes unique value in identification field
- If router fragments packet it copies id into each fragment
- Offset field indicates position of fragment in bytes (offset 0 is first)
 - ◆ MoreFragments flag indicates that this isn't the last fragment
 - ◆ DontFragment flag tells gateway not to fragment
- All routers must support 576 byte packets

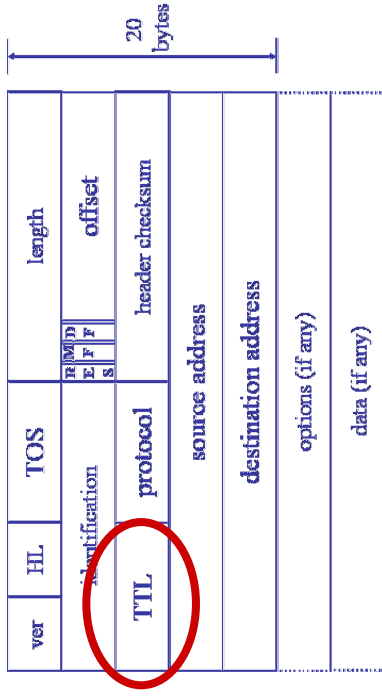


Aside: costs of fragmentation

- Interplay between fragmentation and retransmission
- Packet must be completely reassembled before it can be consumed on the receiving host
- What if a fragment gets lost?

TTL (Time-to-Live)

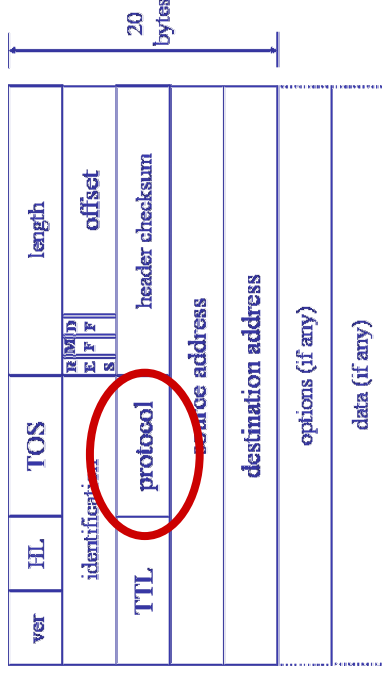
- How many more routers can this packet pass through?
 - ♦ Designed to limit packet from looping forever
- Each router decrements TTL field
- If TTL is 0 then router discards packet



Protocol

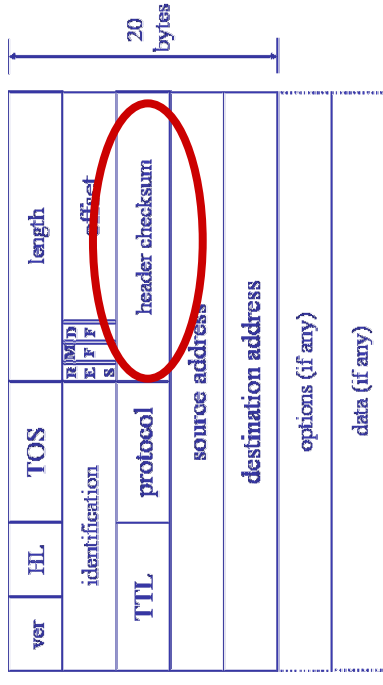
- Which transport protocol is the data using?
 - ♦ i.e. how should a host interpret the data
 - ♦ Called *demultiplexing*

- TCP = 6
- UDP = 17



Header checksum

- Detects errors in IP header
 - ◆ Calculated by sending host
 - ◆ Checked by receiving host
- Must be recalculated by router. Why?
- Only protects header, not data

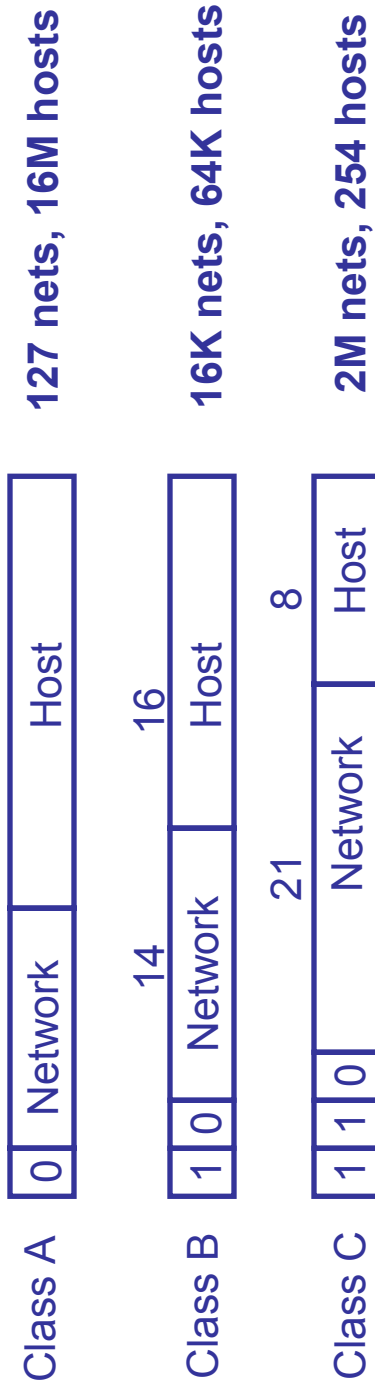


IP addressing

- 32-bits in an IPv4 address
 - ♦ Dotted decimal format a.b.c.d
 - ♦ Each represent 8 bits of address
- Network part and host part
 - ♦ E.g. IP address 132.239.15.3
 - ♦ 132.239 refers to the UCSD campus network
 - ♦ 15.3 refers to the host gremlin.ucsd.edu
- Which part is network vs host?

Class-based routing (<1993)

- Most significant bits determines “class” of address



- Pro: single lookup to find address
- Con
 - ◆ Fragmentation
 - ◆ Hard to aggregate

Classless addressing (1993)

- Classless Inter-Domain Routing (CIDR)
 - ♦ Routes represented by tuple (network prefix/mask)
 - ♦ Allows arbitrary allocation between network and host address



Prefix **Mask=# significant bits representing prefix**

- ♦ e.g. 10.95.1.2/8: 10 is network and remainder (95.1.2) is host
- Pro: Finer grained allocation; aggregation
- Con: More expensive lookup: *longest prefix match*

Options

- Special requests
 - ◆ Route Record
 - ◆ Timestamp
 - ◆ Source Route
 - ◆ Others...
- Variable length
- Interpreted by each router
 - ◆ Expensive design decision

ICMP

- Internet Control Message Protocol
 - ◆ Sister protocol to IP
- Management functions (in response to pkts)
 - ◆ Asynchronous response from routers
 - ◆ Destination Unreachable
 - ◆ Time Exceeded
- Testing functions (request/reply pairs)
 - ◆ Echo request/response
 - ◆ Timestamp request/response

How is IP changing?

- IPv6
 - ◆ 128bit addresses
 - ◆ No fragmentation (so no header length), no options per se
 - ◆ Flow label
 - ◆ 1500 MTU
 - ◆ Security and mobility built in
- IPSEC
 - ◆ Authentication and Encryption of packet
 - ◆ Generally implemented end-to-end (at hosts)
- Diffserv
 - ◆ Reuse ToS bits to indicate (roughly) a local QoS class

Meta-points...

- The Internet was designed
 - ♦ There is no natural law that says TCP/IP, network routing, etc.. had to look the way it does now
 - ♦ It could well have been done differently
- The Internet evolves
 - ♦ The Internet today is not the same Internet as 1988, 1973
 - ♦ TCP/IP have changed considerably over the years
 - ♦ We're using IPv4, with IPv6 (maybe) being deployed
- Many of these design issues are deep
 - ♦ Seemingly straightforward decisions can have very subtle correctness and performance implications
 - ♦ E.g. Implications of fragmentation

Stuff you should definitely remember

- End-to-end principle and how its applied
- Purpose of the Internet Protocol
 - ◆ What problems it solves
 - ◆ How it solves them

For Next Time...

- Reliable Transmission and Flow Control
 - ◆ Some TCP specifics
- Read 2.5 and Chap 5 up to (but not including) 5.3