CMSC417

Computer Networks

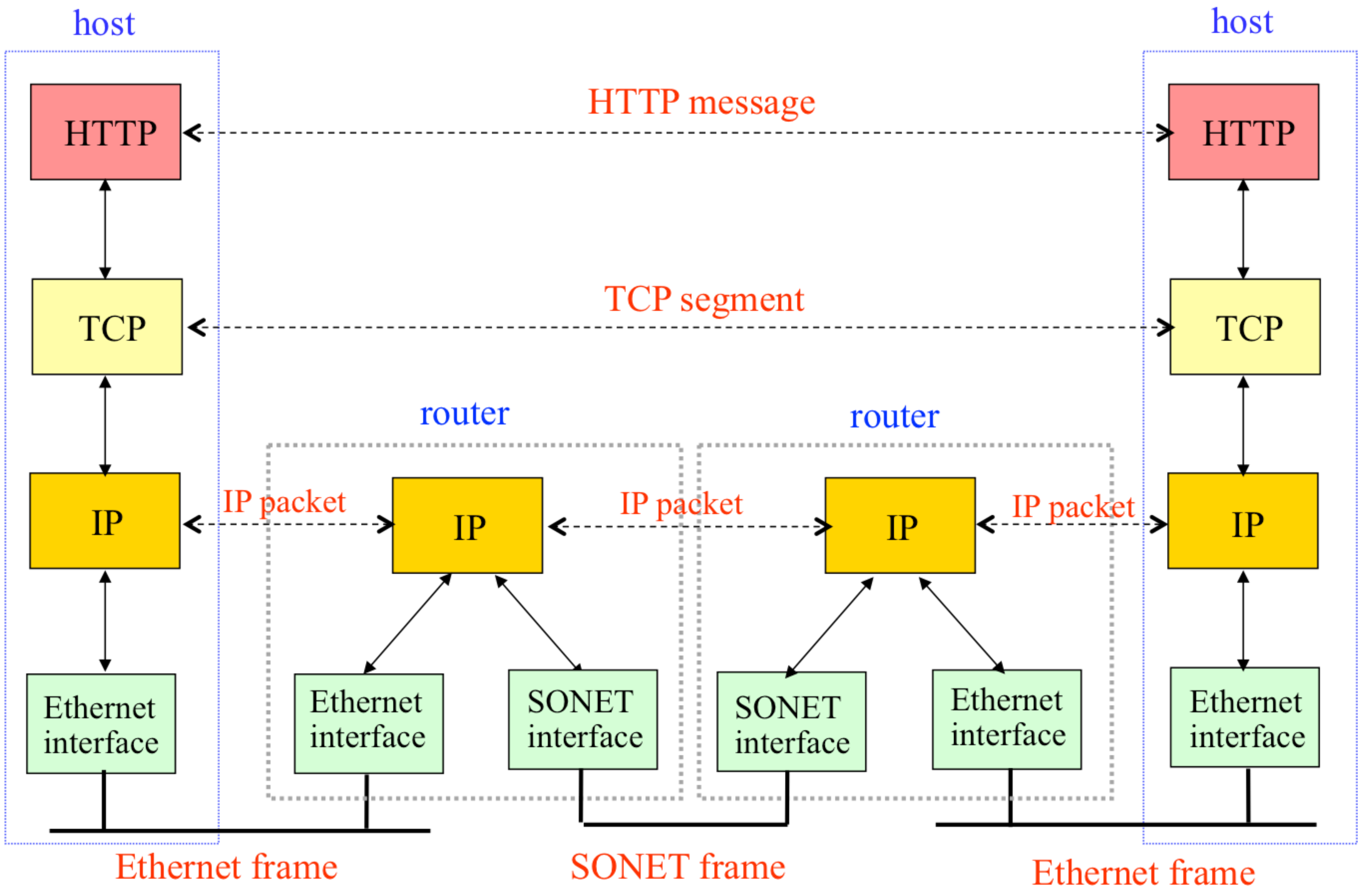
Prof. Ashok K. Agrawala

© 2017 Ashok Agrawala

September 11, 2018

The Network Layer

Message, Segment, Packet, and Frame

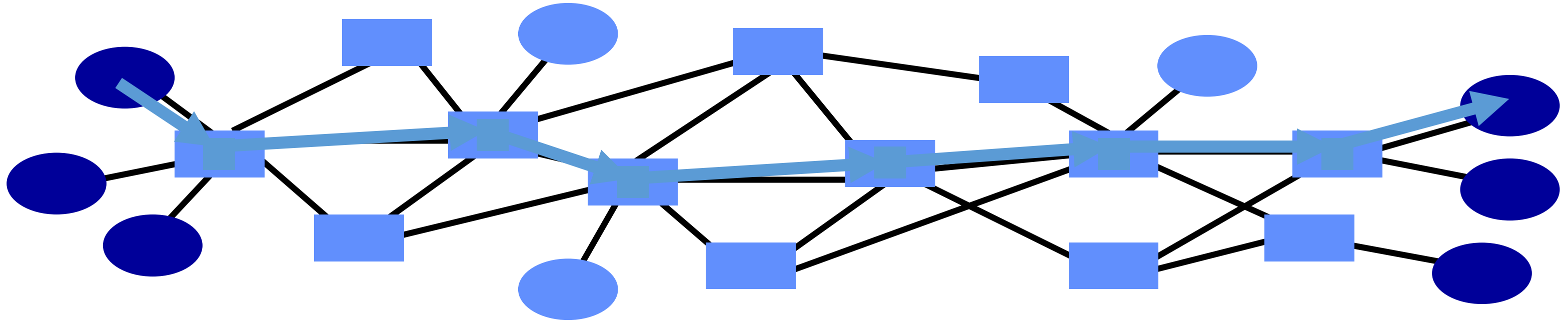


Network Layer Design Isues

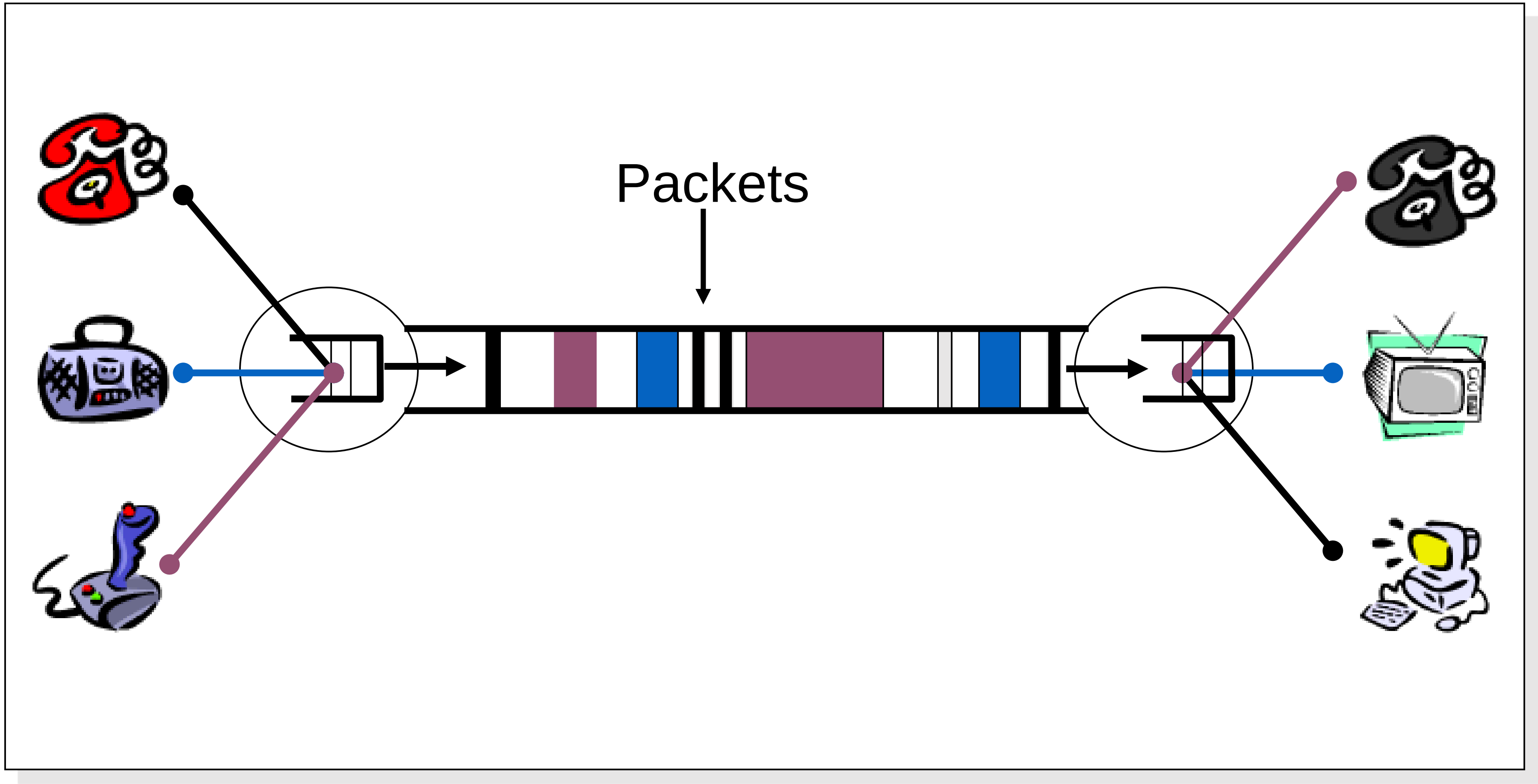
- Store-and-Forward Packet Switching
- Services Provided to the Transport Layer
- Implementation of Connectionless Service
- Implementation of Connection-Oriented Service
- Comparison of Virtual-Circuit and Datagram Subnets

Packet Switching (e.g., Internet)

- Data traffic divided into packets; each packet contains a header (with address)
- Packets travel separately through network
  - Packet forwarding based on the header
  - Network nodes may store packets temporarily
- Destination reconstructs the message

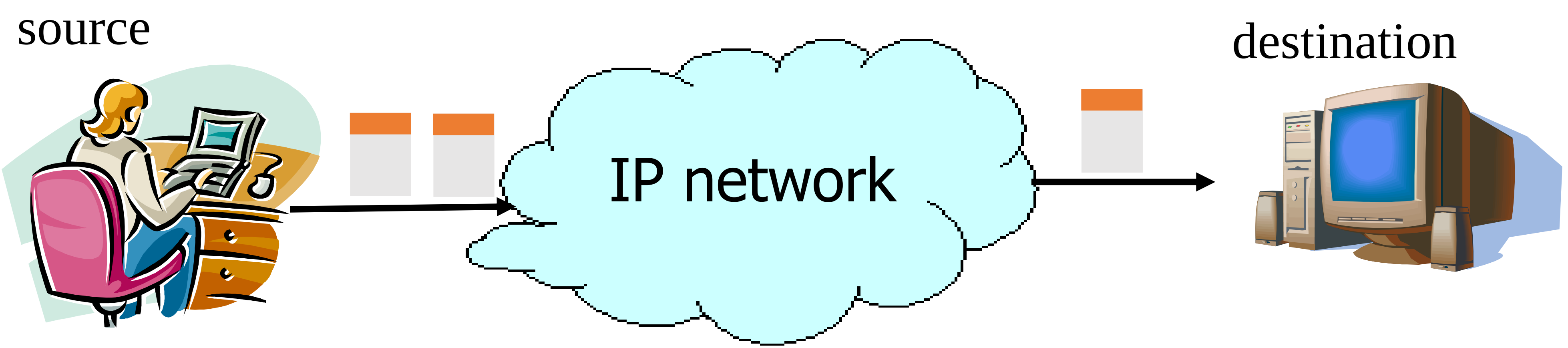


Packet Switching: Statistical Multiplexing



IP Service: Best-Effort Packet Delivery

- Packet switching
  - Divide messages into a sequence of packets
  - Headers with source and destination address
- Best-effort delivery
  - Packets may be lost
  - Packets may be corrupted
  - Packets may be delivered out of order



IP Service Model: Why Packets?

- Data traffic is bursty
  - Logging in to remote machines
  - Exchanging e-mail messages
- Don't want to waste reserved bandwidth as no traffic is exchanged during idle periods
- Better to allow multiplexing (different transfers share access to the same links)
- Packets can be delivered by most anything (RFC 2549: IP over Avian Carriers—a. k. a. birds!)

IP Service Model: Why Packets?

STILL, packet switching can be inefficient, as it requires extra header bits on every packet.

IP Service Model: Why Best-Effort?

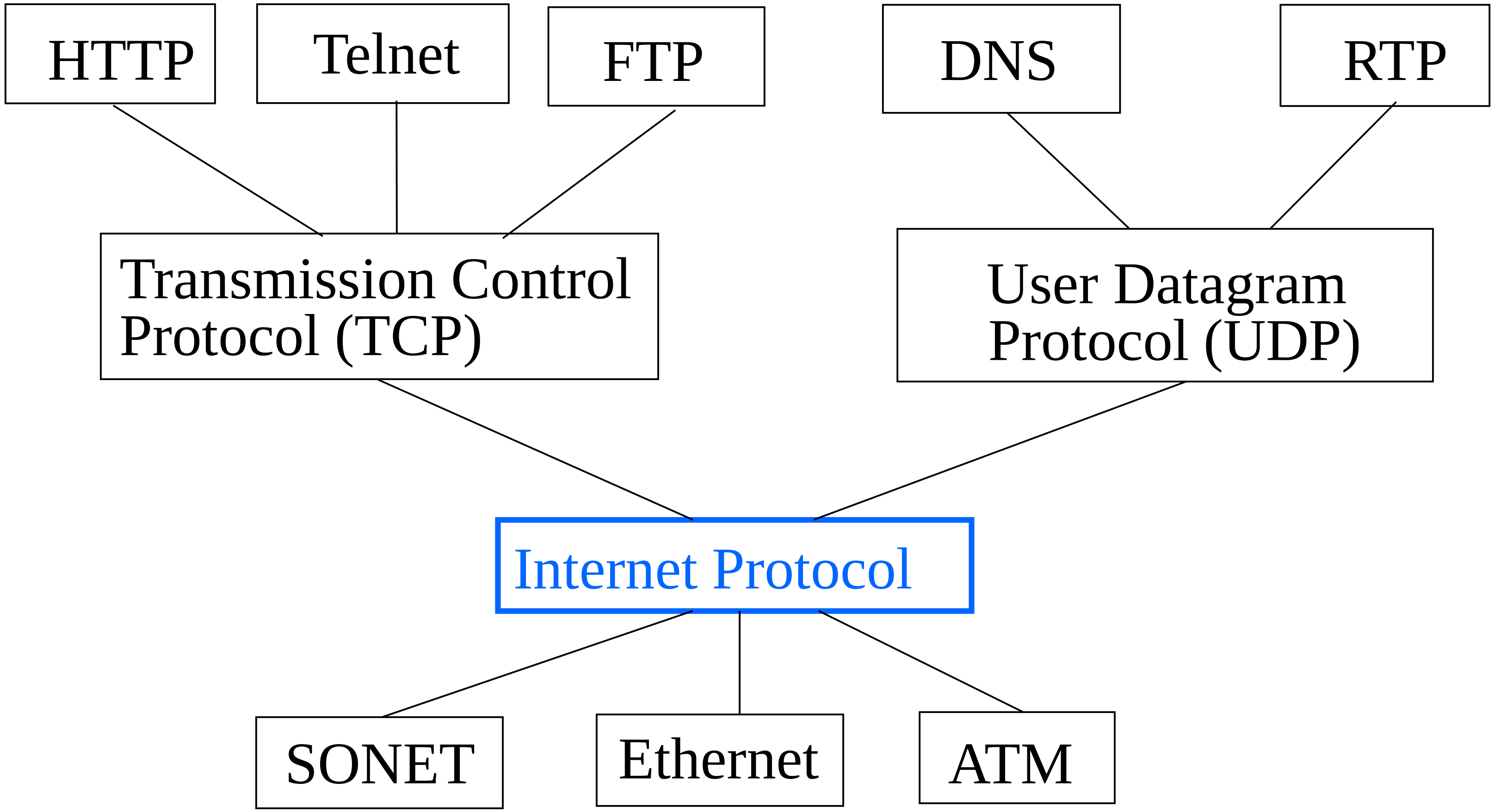
- IP means never having to say you're sorry
  - Don't need to reserve bandwidth and memory
  - Don't need to do error detection & correction
  - Don't need to remember from one packet to next
- Easier to survive failures—transient disruptions are okay during failover

Applications *do* want efficient, accurate transfer of data in order, in a timely fashion.

IP Service: Best-Effort is Enough

Apparent Problem	Justification
No error detection or correction	Higher-level protocol can provide error checking
Successive packets may not follow the same path	Not a problem as long as packets reach the destination
Packets can be delivered out-of-order	Receiver can put packets back in order (if necessary)
Packets may be lost or arbitrarily delayed	Sender can send the packets again (if desired)
No network congestion control (beyond "drop")	Sender can slow down in response to loss or delay

Layering in the IP Protocols



History: Why IP Packets?

- IP proposed in the early 1970s (Defense Advanced Research Project Agency (DARPA))
- Goal: connect existing networks
  - To develop an effective technique for multiplexed utilization of existing interconnected networks
  - e. g. connect packet radio networks to the ARPAnet

History: Why IP Packets?

- Motivating applications
  - Remote login to server machines
  - Inherently bursty traffic with long silent periods
- Prior ARPAnet experience with packet switching
  - Previous DARPA project
  - Demonstrated store-and-forward packet switching

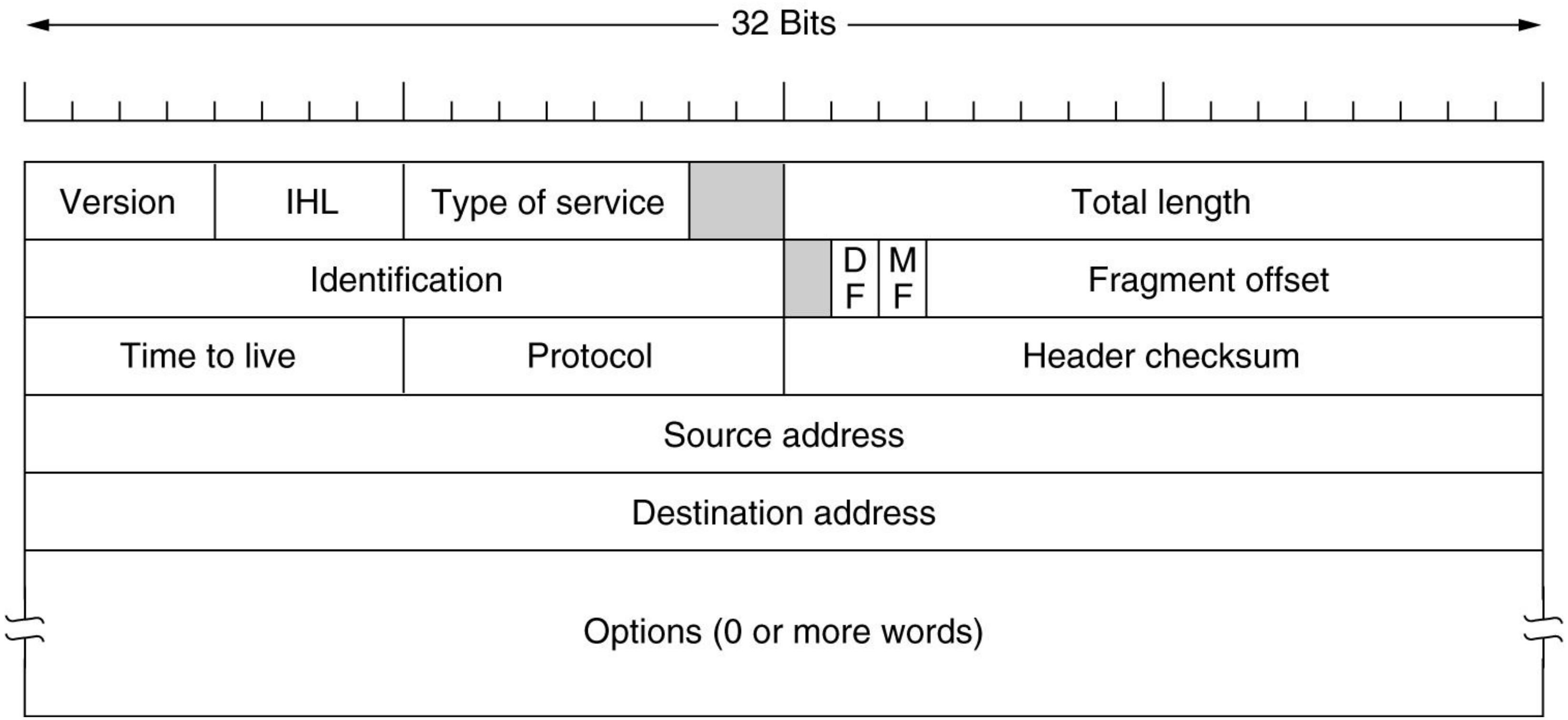
Other Main Driving Goals (In Order)

- Communication should continue despite failures
  - Survive equipment failure or physical attack
  - Traffic between two hosts continue on another path
- Support multiple types of communication services
  - Differing requirements for speed, latency, & reliability
  - Bidirectional reliable delivery vs. message service
- Accommodate a variety of networks
  - Both military and commercial facilities
  - Minimize assumptions about the underlying network

Other Driving Goals, Somewhat Met

Goal	Met via…	BUT…
Permit distributed management of resources	Nodes managed by different institutions	This is still rather challenging
Cost-effectiveness	Statistical multiplexing through packet switching	Packet headers and retransmissions wasteful, though!
Ease of attaching new hosts	Standard implementations of end-host protocols	However, you still need a fair amount of end-host software
Accountability for use of resources	Monitoring functions in the nodes	But this is still fairly limited and immature

The IP Protocol



The IP Datagram

(from Wikipedia):

Datagram: a basic transfer unit associated with a packet-switched network.

- typically structured in **header** and **payload** sections
- provide a **connectionless** service across a packet-switched network

The IP Datagram

<table>
<tr><td></td><td colspan="2">Bit 0</td><td></td><td></td><td>Bit 31</td></tr>
<tr><td rowspan="6">Header</td><td>Version (4)</td><td>Hdr Len (4)</td><td>TOS (8)</td><td colspan="2">Total Length in bytes (16)</td></tr>
<tr><td colspan="3">Identification (16 bits)</td><td>Flags (3)</td><td>Fragment Offset (13)</td></tr>
<tr><td colspan="2">Time to Live (8)</td><td>Protocol (8)</td><td colspan="2">Header Checksum (16)</td></tr>
<tr><td colspan="5">Source IP Address</td></tr>
<tr><td colspan="5">Destination IP Address</td></tr>
<tr><td colspan="5">Options (if any)</td></tr>
<tr><td>Data</td><td colspan="5">Data (variable length)</td></tr>
</table>

IP Packet Header

Version

- Version number of IP protocol
- Current version is Version 4
- Version 6 has different header format

Bit 0 … Bit 31

Version (4)	Hdr Len (4)	TOS (8)	Total Length in bytes (16)	
Identification (16 bits)			Flags (3)	Fragment Offset (13)
Time to Live (8)		Protocol (8)	Header Checksum (16)	
Source IP Address				
Destination IP Address				
Options (if any)				

IP Packet Header

Header Length (in 32 bit words)

- Indicates end of header and beginning of payload
- If no options, Header length = 5

Bit 0 | Bit 31

Version (4)	Hdr Len (4)	TOS (8)	Total Length in bytes (16)		
Identification (16 bits)			Flags (3)	Fragment Offset (13)	
Time to Live (8)		Protocol (8)	Header Checksum (16)		
Source IP Address					
Destination IP Address					
Options (if any)					

IP Packet Header

Type of Service (TOS)

- Allows different types of service to be requested
- Initially, meaning was not well defined
- Currently being defined (diffserv)

Bit 0 Bit 31

<table>
<tr><td>Version (4)</td><td>Hdr Len (4)</td><td>TOS (8)</td><td colspan="2">Total Length in bytes (16)</td></tr>
<tr><td colspan="3">Identification (16 bits)</td><td>Flags (3)</td><td>Fragment Offset (13)</td></tr>
<tr><td colspan="2">Time to Live (8)</td><td>Protocol (8)</td><td colspan="2">Header Checksum (16)</td></tr>
<tr><td colspan="5">Source IP Address</td></tr>
<tr><td colspan="5">Destination IP Address</td></tr>
<tr><td colspan="5">Options (if any)</td></tr>
</table>

IP Packet Header

Packet Length (in Bytes)

- Unambiguously specify end of packet
- Max packet size = 2^{16} = 65,535 Bytes

Bit 0 … Bit 31

Version (4)	Hdr Len (4)	TOS (8)	Total Length in bytes (16)	
Identification (16 bits)			Flags (3)	Fragment Offset (13)
Time to Live (8)		Protocol (8)	Header Checksum (16)	
Source IP Address				
Destination IP Address				
Options (if any)				

IP Packet Header

These three fields for Fragmentation Control

(will come back to them later)

Bit 0 … Bit 31

<table>
<tr><td>Version (4)</td><td>Hdr Len (4)</td><td colspan="2">TOS (8)</td><td colspan="2">Total Length in bytes (16)</td></tr>
<tr><td colspan="4">Identification (16 bits)</td><td>Flags (3)</td><td>Fragment Offset (13)</td></tr>
<tr><td colspan="2">Time to Live (8)</td><td colspan="2">Protocol (8)</td><td colspan="2">Header Checksum (16)</td></tr>
<tr><td colspan="6">Source IP Address</td></tr>
<tr><td colspan="6">Destination IP Address</td></tr>
<tr><td colspan="6">Options (if any)</td></tr>
</table>

IP Packet Header

Time to Live

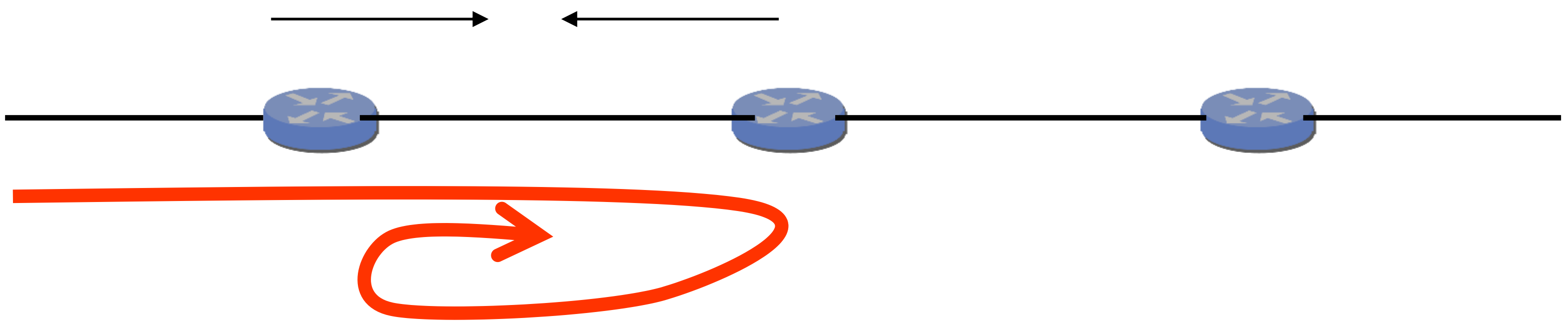
- Initially set by sender (up to 255)
- Decremented by each router
- Discard when TTL = 0 to avoid infinite routing loops

Bit 0 … Bit 31

<table>
<tr><td>Version (4)</td><td>Hdr Len (4)</td><td colspan="2">TOS (8)</td><td colspan="4">Total Length in bytes (16)</td></tr>
<tr><td colspan="4">Identification (16 bits)</td><td>Flags (3)</td><td colspan="3">Fragment Offset (13)</td></tr>
<tr><td colspan="2">Time to Live (8)</td><td colspan="2">Protocol (8)</td><td colspan="4">Header Checksum (16)</td></tr>
<tr><td colspan="8">Source IP Address</td></tr>
<tr><td colspan="8">Destination IP Address</td></tr>
<tr><td colspan="8">Options (if any)</td></tr>
</table>

Time-to-Live (TTL) Field

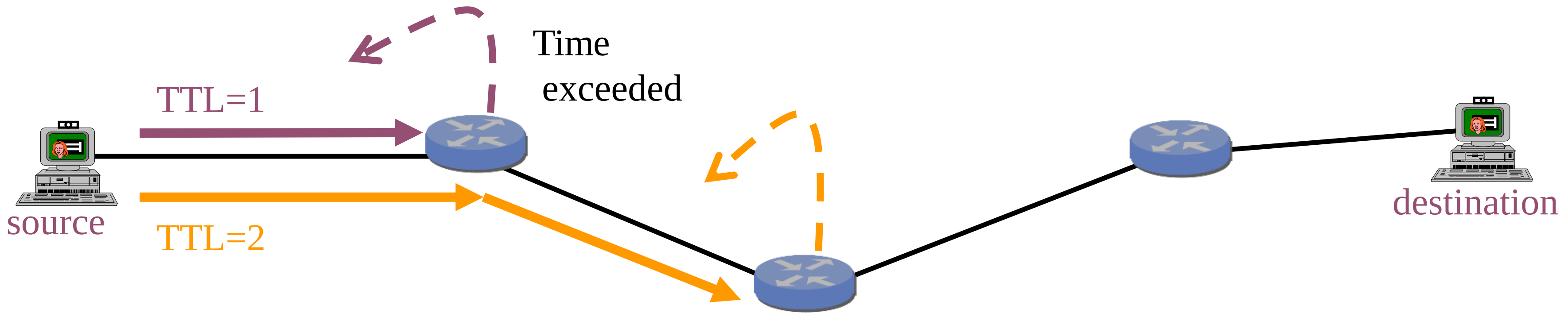
- Potential robustness problem
  - Forwarding loops can cause packets to cycle forever
  - Confusing if the packet arrives much later



- Time-to-live field in packet header
  - TTL field decremented by each router on the path
  - Packet is discarded when TTL field reaches 0…
  - …and “time exceeded” message is sent to the source

Application of TTL in Traceroute

- Time-To-Live field in IP packet header
  - Source sends a packet with a TTL of *n*
  - Each router along the path decrements the TTL
  - "TTL exceeded" sent when TTL reaches *0*
- Traceroute tool exploits this TTL behavior



Send packets with TTL=1, 2, … and record source of "time exceeded" message

Example Traceroute: Berkeley to CNN

No response from router

No name resolution

Hop number, IP address, DNS name		
1	169.229.62.1	inr-daedalus-0.CS.Berkeley.EDU
2	169.229.59.225	soda-cr-1-1-soda-br-6-2
3	128.32.255.169	vlan242.inr-202-doecev.Berkeley.EDU
4	128.32.0.249	gigE6-0-0.inr-666-doecev.Berkeley.EDU
5	128.32.0.66	qsv-juniper--ucb-gw.calren2.net
6	209.247.159.109	POS1-0.hsipaccess1.SanJose1.Level3.net
7	*	?
8	64.159.1.46	?
9	209.247.9.170	pos8-0.hsa2.Atlanta2.Level3.net
10	66.185.138.33	pop2-atm-P0-2.atdn.net
11	*	?
12	66.185.136.17	pop1-atl-P4-0.atdn.net
13	64.236.16.52	www4.cnn.com

Try Running Traceroute Yourself

- On UNIX machine
  - Traceroute
  - E.g., “traceroute www.cnn.com” or “traceroute 12.1.1.1”
- On Windows machine
  - Tracert
  - E.g., “tracert www.cnn.com” or “tracert 12.1.1.1”
- Common uses of traceroute
  - Discover the topology of the Internet
  - Debug performance and reachability problems

IP Packet Header

Protocol

- Value indicates what is in the data field
- Example: TCP or UDP

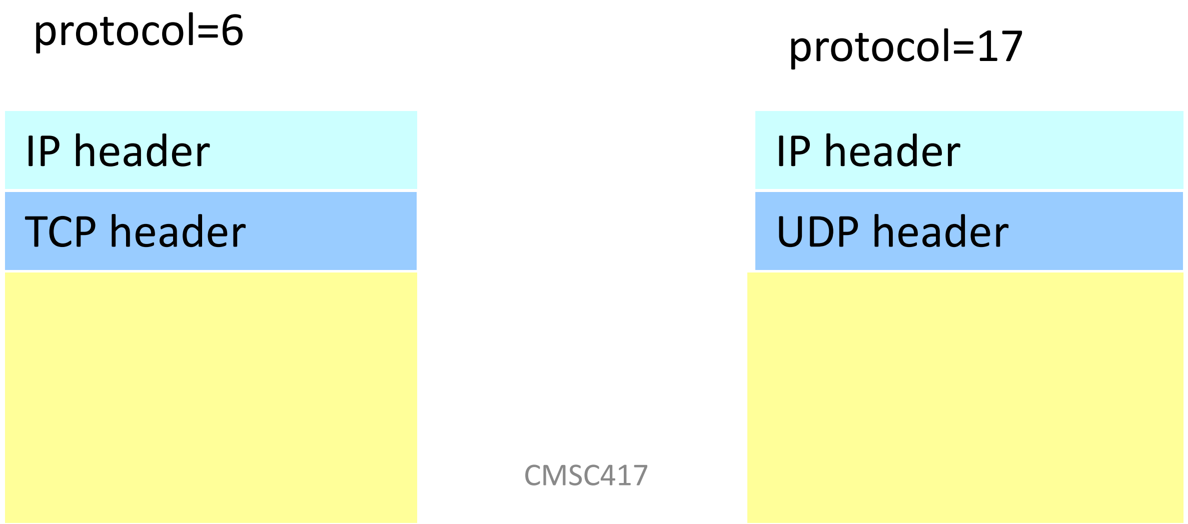
Bit 0 ... Bit 31

Version (4)	Hdr Len (4)	TOS (8)	Total Length in bytes (16)	
Identification (16 bits)			Flags (3)	Fragment Offset (13)
Time to Live (8)		Protocol (8)	Header Checksum (16)	
Source IP Address				
Destination IP Address				
Options (if any)				

IP Packet Header Fields (Continued)

Protocol (8 bits)

- Identifies the higher-level protocol
  - E.g., "6" for the Transmission Control Protocol (TCP)
  - E.g., "17" for the User Datagram Protocol (UDP)
- Important for demultiplexing at receiving host (indicates what kind of header to expect next)



IP Packet Header

Header Checksum

- Checks for error in the header only
- Bad headers can harm the network
- If error found, packet is simply discarded

Bit 0 Bit 31

<table>
<tr><td>Version (4)</td><td>Hdr Len (4)</td><td>TOS (8)</td><td colspan="2">Total Length in bytes (16)</td></tr>
<tr><td colspan="3">Identification (16 bits)</td><td>Flags (3)</td><td>Fragment Offset (13)</td></tr>
<tr><td colspan="2">Time to Live (8)</td><td>Protocol (8)</td><td colspan="2">Header Checksum (16)</td></tr>
<tr><td colspan="5">Source IP Address</td></tr>
<tr><td colspan="5">Destination IP Address</td></tr>
<tr><td colspan="5">Options (if any)</td></tr>
</table>

IP Packet Header Fields (Continued)

Checksum (16 bits)

- Sum of all 16-bit words in the IP packet header
- If any bits of the header are corrupted in transit, the checksum won't match at receiving host
- Receiving host discards corrupted packets—the sending host will retransmit the packet if needed

134	134
+ 212	+ 216
= 346	= 350
	Mismatch!

IP Packet Header

Source and Destination IP Addresses

Strings of 32 ones and zeros

Bit 0 … Bit 31

Version (4)	Hdr Len (4)	TOS (8)	Total Length in bytes (16)	
Identification (16 bits)			Flags (3)	Fragment Offset (13)
Time to Live (8)		Protocol (8)	Header Checksum (16)	
Source IP Address				
Destination IP Address				
Options (if any)				

IP Packet Header

Options

These can include:

- timestamp
- record route
- source route

Bit 0 … Bit 31

<table>
<tr><td>Version (4)</td><td>Hdr Len (4)</td><td colspan="2">TOS (8)</td><td colspan="4">Total Length in bytes (16)</td></tr>
<tr><td colspan="4">Identification (16 bits)</td><td>Flags (3)</td><td colspan="3">Fragment Offset (13)</td></tr>
<tr><td colspan="2">Time to Live (8)</td><td colspan="2">Protocol (8)</td><td colspan="4">Header Checksum (16)</td></tr>
<tr><td colspan="8">Source IP Address</td></tr>
<tr><td colspan="8">Destination IP Address</td></tr>
<tr><td colspan="8">Options (if any)</td></tr>
</table>

IP Fragmentation and Reassembly

- Maximum Transmission Unit (MTU)
  - Largest IP packet a network will accept
  - Arriving IP packet may be larger (max IP packet size = 65,535 bytes)
- Sender or router will split the packet into multiple fragments
- Destination will reassemble the packet
- IP header fields used to identify and order related fragments

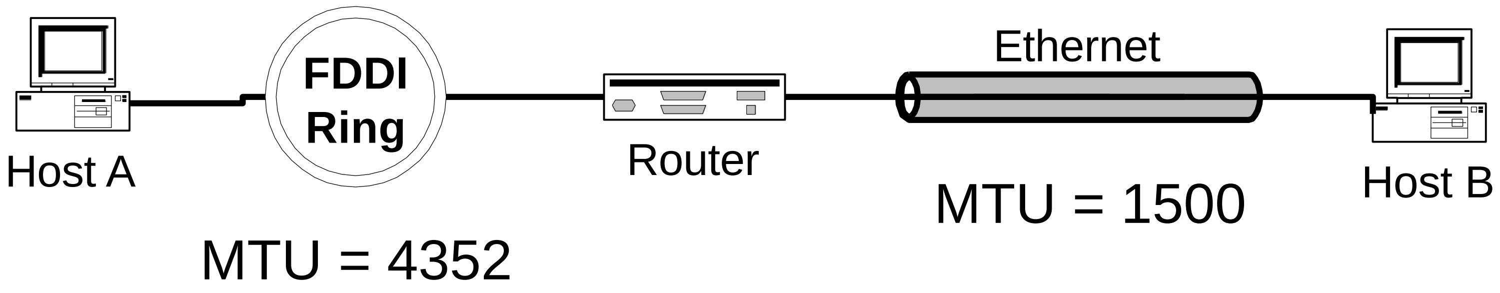
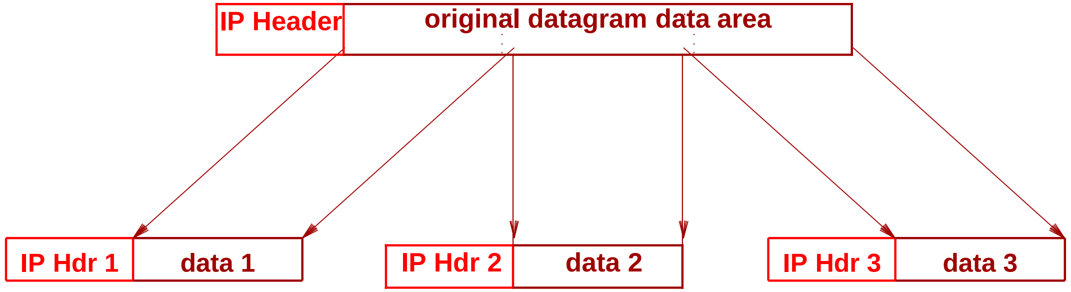


Illustration of Datagram Fragmentation

Each fragment has IP

datagram header fields

- Identify original datagram
- Indicate where fragment fits



IP Packet Header

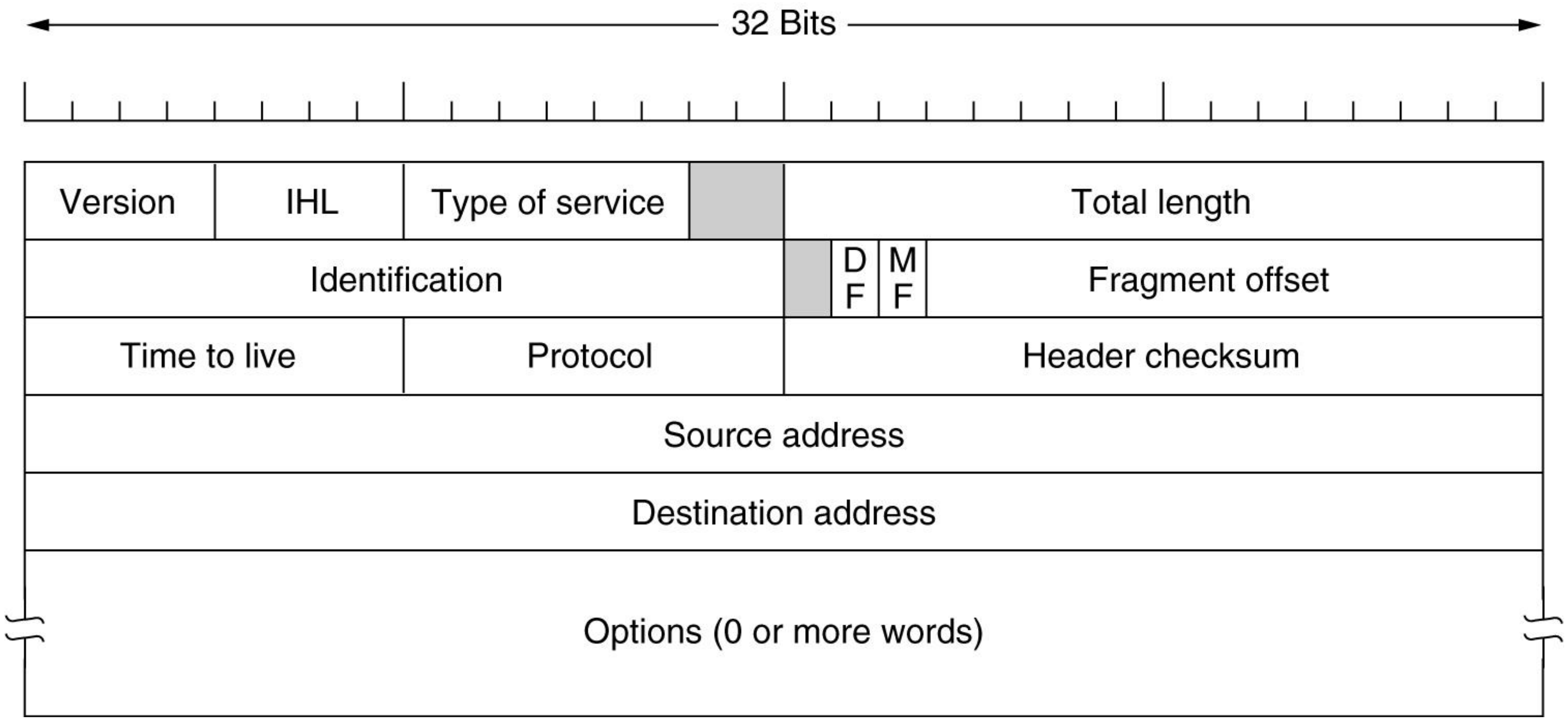
Identification

All fragments of a single datagram have the same identification number

Bit 0 … Bit 31

Version (4)	Hdr Len (4)	TOS (8)	Total Length in bytes (16)	
Identification (16 bits)			Flags (3)	Fragment Offset (13)
Time to Live (8)		Protocol (8)	Header Checksum (16)	
Source IP Address				
Destination IP Address				
Options (if any)				

The IP Protocol



IP Packet Header

Flags:

- *1st bit:* reserved, must be zero
- *2nd bit:* **DF**—Do Not Fragment
- *3rd bit:* **MF**—More Fragments

Bit 0 … Bit 31

<table>
<tr><td>Version (4)</td><td>Hdr Len (4)</td><td>TOS (8)</td><td colspan="2">Total Length in bytes (16)</td></tr>
<tr><td colspan="3">Identification (16 bits)</td><td>Flags (3)</td><td>Fragment Offset (13)</td></tr>
<tr><td colspan="2">Time to Live (8)</td><td>Protocol (8)</td><td colspan="2">Header Checksum (16)</td></tr>
<tr><td colspan="5">Source IP Address</td></tr>
<tr><td colspan="5">Destination IP Address</td></tr>
<tr><td colspan="5">Options (if any)</td></tr>
</table>

IP Packet Header

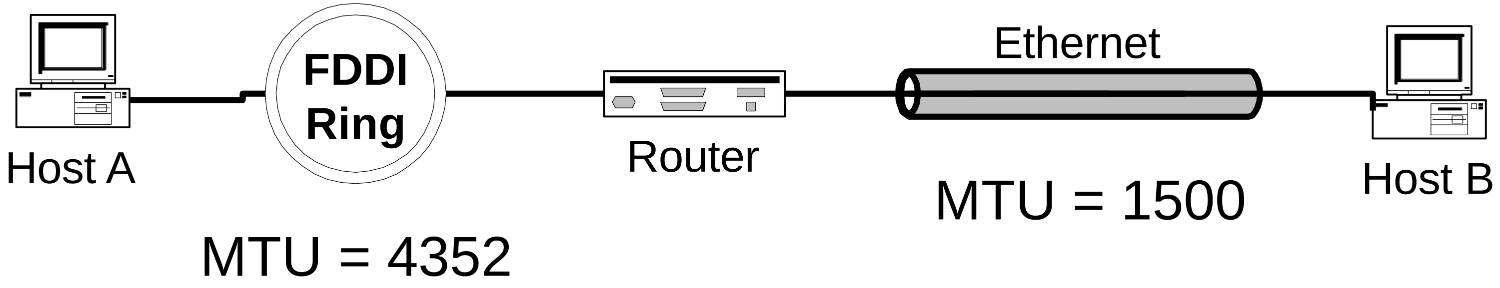
Fragment Offset (in units of 8 bytes)

- Used for reassembly of packet
- 1st fragment has offset = 0

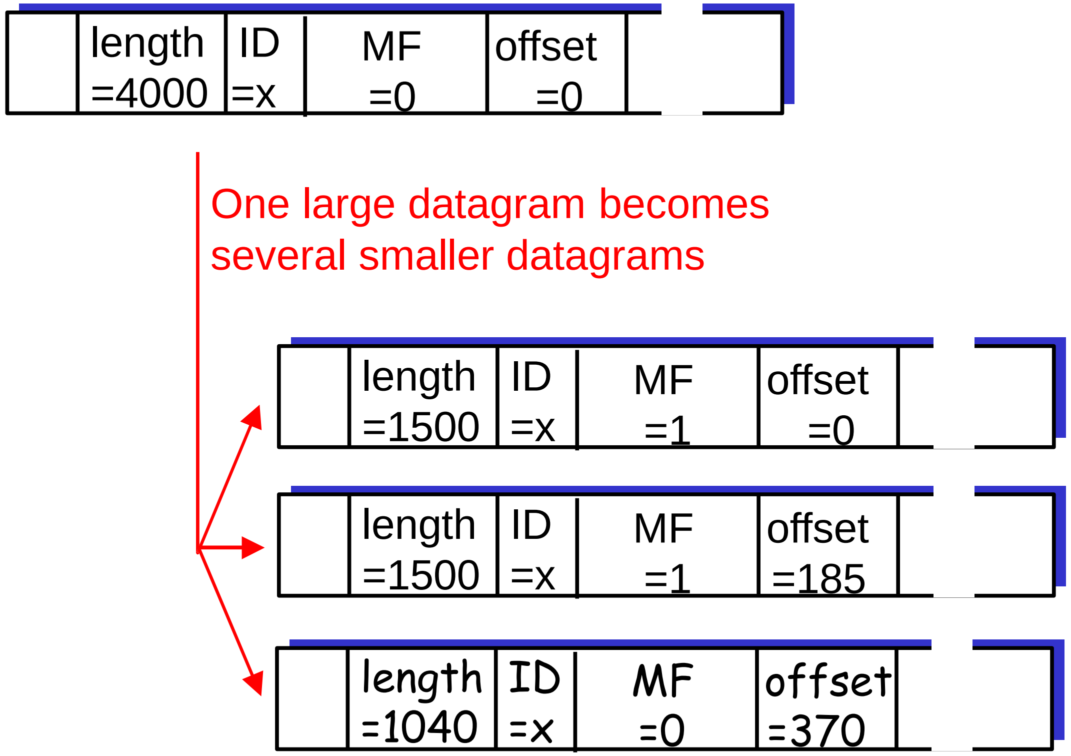
<table>
<tr><td colspan="3">Bit 0</td><td colspan="2">Bit 31</td></tr>
<tr><td>Version (4)</td><td>Hdr Len (4)</td><td>TOS (8)</td><td colspan="2">Total Length in bytes (16)</td></tr>
<tr><td colspan="3">Identification (16 bits)</td><td>Flags (3)</td><td>Fragment Offset (13)</td></tr>
<tr><td colspan="2">Time to Live (8)</td><td>Protocol (8)</td><td colspan="2">Header Checksum (16)</td></tr>
<tr><td colspan="5">Source IP Address</td></tr>
<tr><td colspan="5">Destination IP Address</td></tr>
<tr><td colspan="5">Options (if any)</td></tr>
</table>

IP Fragmentation Example

Host A wants to send to Host B an IP datagram of size = 4000 Bytes



IP Fragmentation Example



Multiple Fragmenting Points

Let MTUs along internet path be

- 1500
- 1500
- 1500
- 576
- 1500

Result: fragmentation can occur twice

Fragmentation Example

- Transport layer Segment size 4500 bytes
- IP Packet Size 4520 bytes – 20 byte header – no options
- MTU – 2500 bytes

Fragment	Total bytes	Header bytes	Data bytes	"More fragments" flag	Fragment offset (8-byte blocks)
1	2500	20	2480	1	0
2	2040	20	2020	0	310

- Total data Size = 2480+ 2020 = 4500
- Offset – 0
  - 0+2480/8 =310

Fragmentation Example- Cont’d

Fragment	Total bytes	Header bytes	Data bytes	"More fragments" flag	Fragment offset (8-byte blocks)
1	2500	20	2480	1	0
2	2040	20	2020	0	310

- Assume a link with MTU of 1500 bytes follows

Fragment	Total bytes	Header bytes	Data bytes	"More fragments" flag	Fragment offset (8-byte blocks)
1	1500	20	1480	1	0
2	1020	20	1000	1	185
3	1500	20	1480	1	310
4	560	20	540	0	495

Fragmenting a Fragment

- Needed when fragment too large for network MTU
- Arbitrary subfragmentation possible
- Router divides fragments into smaller pieces All

  fragments at same “level”
- Offset given with respect to original datagram
- Destination cannot distinguish subfragments

Fragment Lost

Receiver

- Collects incoming fragments
- Reassembles when all fragments arrive
- Does not know identity of router that did fragmentation
- Cannot request missing pieces

Consequence: Loss of one fragment means entire datagram lost

IP Packet Header (Continued)

- Two IP addresses
  - Source IP address (32 bits)
  - Destination IP address (32 bits)
- Destination address
  - Unique identifier for the receiving host
  - Allows each node to make forwarding decisions
- Source address
  - Unique identifier for the sending host
  - Recipient can decide whether to accept packet
  - Enables recipient to send a reply back to source

The IP Protocol

Some of the IP options

Option	Description
Security	Specifies how secret the datagram is
Strict source routing	Gives the complete path to be followed
Loose source routing	Gives a list of routers not to be missed
Record route	Makes each router append its IP address
Timestamp	Makes each router append its address and timestamp

What if the Source Lies?

- Source address should be the sending host
  - But, who's checking, anyway?
  - You could send packets with any source you want
- Why would someone want to do this?
  - Launch a denial-of-service attack
    - Send excessive packets to the destination
    - This overloads the node, or the links leading to the node
  - Evade detection by "spoofing"
    - But, the victim could identify you by the source address
    - So, you can put someone else's source address in the packets
  - Also, an attack against the spoofed host
    - Spoofed host is wrongly blamed
    - Spoofed host may receive return traffic from the receiver