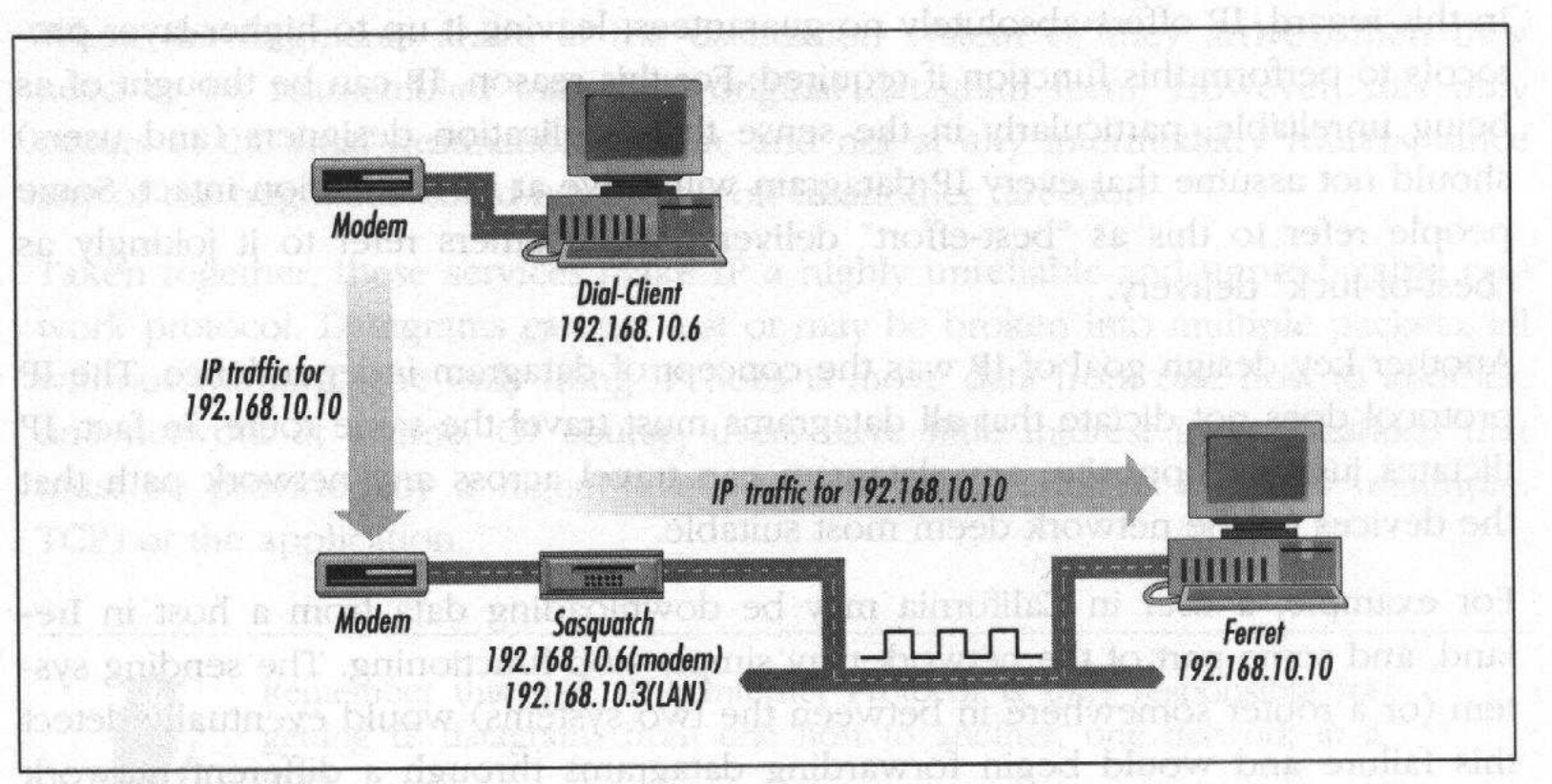
Internet Protocol Motivation

cross local network boundaries



Problem

Hoare's Law of Large Problems:

Inside every large problem is a small problem struggling to get out.

Which hardware address does 134.96.217.50 have?

Solution: ARP

ARP=address resolution protocol

must be implemented within layer 2

implements the relation

$$R \subset (B \times B \times B \times B) \times (B \times B \times B \times B \times B \times B)$$

for IP and Ethernet

where $B = \{0, 1, 2, \ldots, 255\}$ i.e. a byte

term byte non-standard $\rightsquigarrow$use octet

is this a map?

Protocol

A protocol is a finite sequence of instructions describing the interaction between two or more entities. This compounds the data format of messages between the entities and the handling of error cases.

Properties:

- Each entity must know the protocol steps.
- The steps must be deterministic for every situation.

Example

Customer buys a tomato from a merchant.

1. The customer asks the merchant for the tomato.
2. The merchant gives the tomato to the customer.
3. The customer gives the merchant money.
4. The merchant gives the customer change.

Problems:

- Does the merchant understand the word "tomato"?
- Is the tomato genuine?
- Is the money of the right money?
- Does the merchant recognize the money?
- Is the money genuine?

- Does the customer run away after step 2?

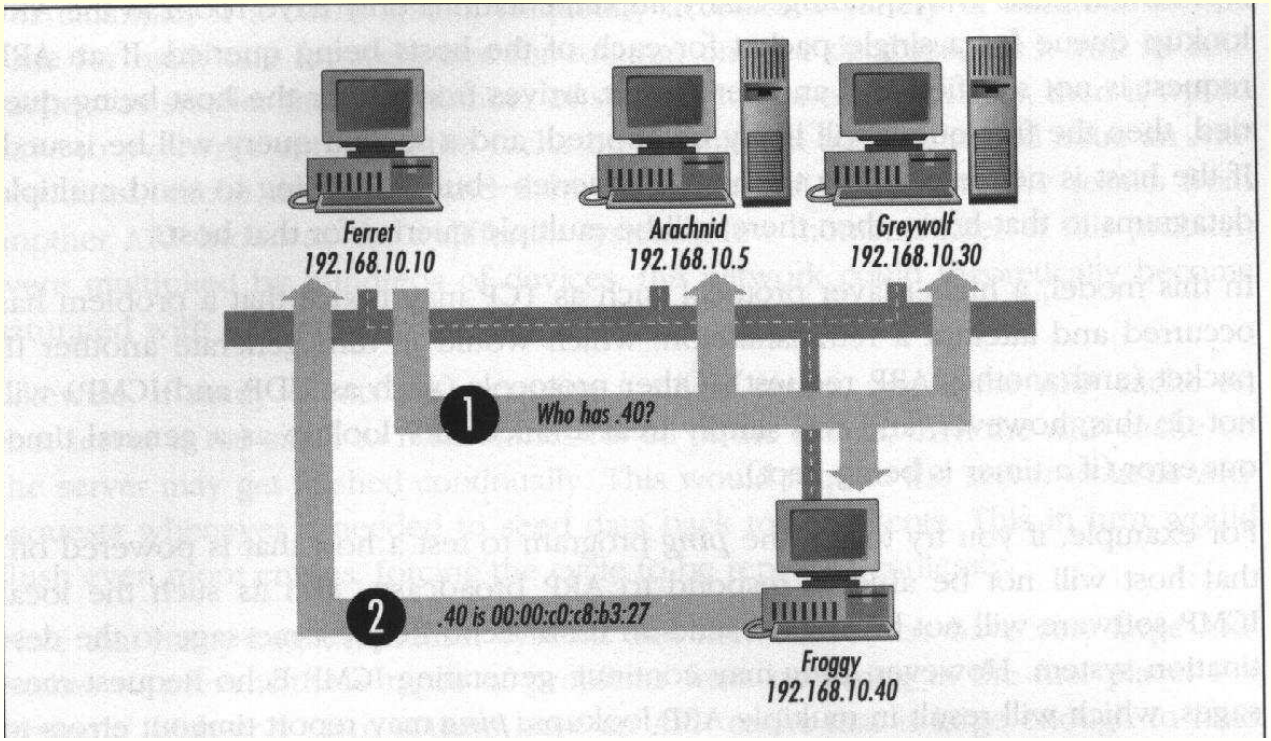
Protocol Destination

- unicast
- broadcast
- multicast

ARP Relation

$$R \subset \underbrace{(B \times B \times \ldots \times B)}_{\text{protocol address}} \times \underbrace{(B \times B \times \ldots \times B)}_{\text{hardware address}}$$

Protocol standard: RFC 826 (ARP)



ARP Message Format

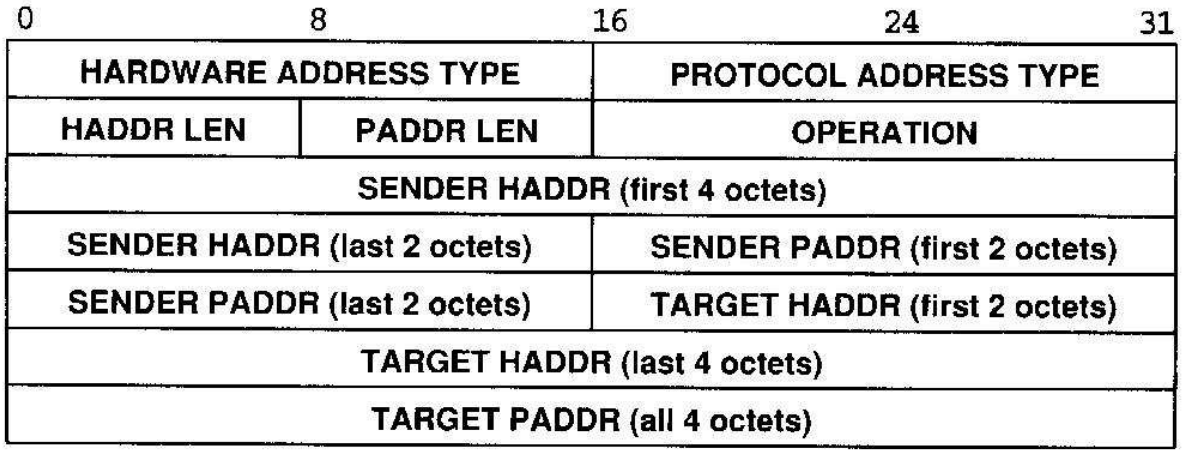


Figure 17.6 The format for an ARP message when used to bind Internet protocol addresses to Ethernet hardware addresses.

sender fills three of the addresses

the unknown entry is set to zero

recipient matches target PADDR $\rightsquigarrow$sends answer

ARP Message Entries

- 16-bit hardware type (Ethernet=6, ATM=19,. . .), see RFC 1700
- 16-bit protocol type (IP=0x0800)
- 8-bit hw-address-length (Ethernet=6)
- 8-bit protocol-address-length (IP=4)
- 16-bit message type
  - ARP Request (1) / Response (2)
  - RARP Request (3) / Response (4)
  - IARP Request (8) / Response (9)
- two source addresses, two dest addresses

Protocol Design

request A to all

response $B \longrightarrow A$

a request–response protocol

- simple (broadcast, wait for answer)
- no headers
- no timers
- no negotiation
- no identification

ARP implementation required (RFC 1122 Host Requirements)

Problems

- No response: host down or busy? Typical timeout after 20 sec.
- Bandwidth: too many ARP requests $\rightsquigarrow$caching
- How long is a cache entry allowed to live (normally 20 min.)

$\rightsquigarrow$ Exercise: find out the cache timeout for Linux/FreeBSD/Win2K/WinXP!

- How big should the ARP cache be?
- Burst data transmission $\rightsquigarrow$burst ARP requests
- Proxy ARP $\rightsquigarrow$ARP responses for disconnected machines
- There are entries which should never expire: static caching

ARP Tool

UNIX Tool for arp cache manipulations: `arp`

```
isl-c-01:~$ arp -a -n
? (134.96.216.1) at 00:11:bc:4c:38:00 on rl0 [ethernet]
? (134.96.216.92) at 00:20:ed:5f:03:3b on rl0 [ethernet]
? (134.96.216.115) at 00:0e:0c:5e:50:6c on rl0 [ethernet]
? (134.96.216.204) at 08:00:20:c4:92:2a on rl0 [ethernet]
? (134.96.216.217) at 00:00:cb:68:0b:81 on rl0 [ethernet]
? (134.96.216.218) at 00:0e:0c:5a:68:c6 on rl0 [ethernet]
? (134.96.216.225) at 00:03:ba:65:41:b7 on rl0 [ethernet]
? (134.96.216.235) at 00:03:ba:65:39:97 on rl0 [ethernet]
? (134.96.216.255) at ff:ff:ff:ff:ff:ff on rl0 permanent [et
```

ARP Extension: Gratuitous ARP

addresses problem of lifetime of ARP cache entry

assume server s gets new network card

every client x: stale ARP cache entry for s from old card

$\rightsquigarrow s$ can't be contacted from x (as long s doesn't send)

solution: send (unrequested) ARP response packet

protocol address is s for source and destination

source hardware address = hardware address of s

dest. hardware address = `FF:FF:FF:FF:FF:FF` (broadcast)

send this on boot–up

consequence: cache update but **not** cache insert

exercise: does your system (Linux, Win2K, WinXP) use Gratuitous ARP?

ARP Extension: UnARP

unsolicited ARP Reply

addresses problem of lifetime of ARP cache entry

assume DHCP server reassigns same IP address directly after DHCP release

ARP caches hold still previous hardware address

solution: send (unrequested) ARP response packet

HW address length = 0

RFC 1868 (experimental)

$\rightsquigarrow$address gets removed from all ARP caches

ARP reversed: RARP

reverse ARP: given hardware address, find protocol address

typical application: diskless workstation gets IP from server

today mostly replaced by DHCP because

- IP addresses may be reused
- no further info
  (hostname, subnet mask, routers, lease length. . .)

source protocol address = destination protocol address = 0

source hardware address = destination hardware address

The Internet Protocol (Layer 3)

analogy: sending a postcard

IP data entity is called packet

sender is oblivious to the routing and the delivery mechanisms

each local delivery agent does its best

every packet unrelated to each other (maybe different paths)

inherently unreliable

reliability in higher protocols (layer 4)

IP Standard

RFC 791 (IP)

RFC 1122 (Host Requirements)

Local Delivery

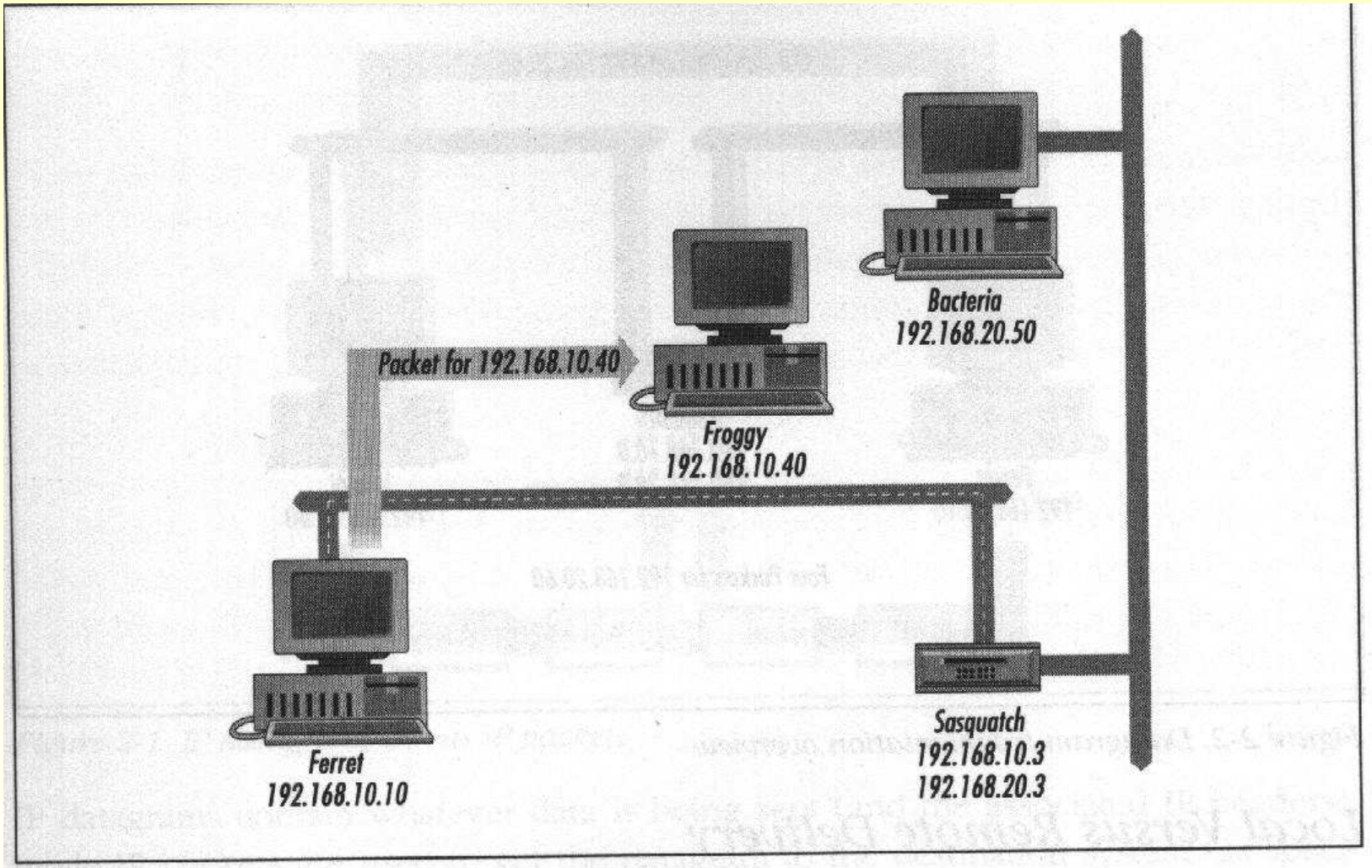
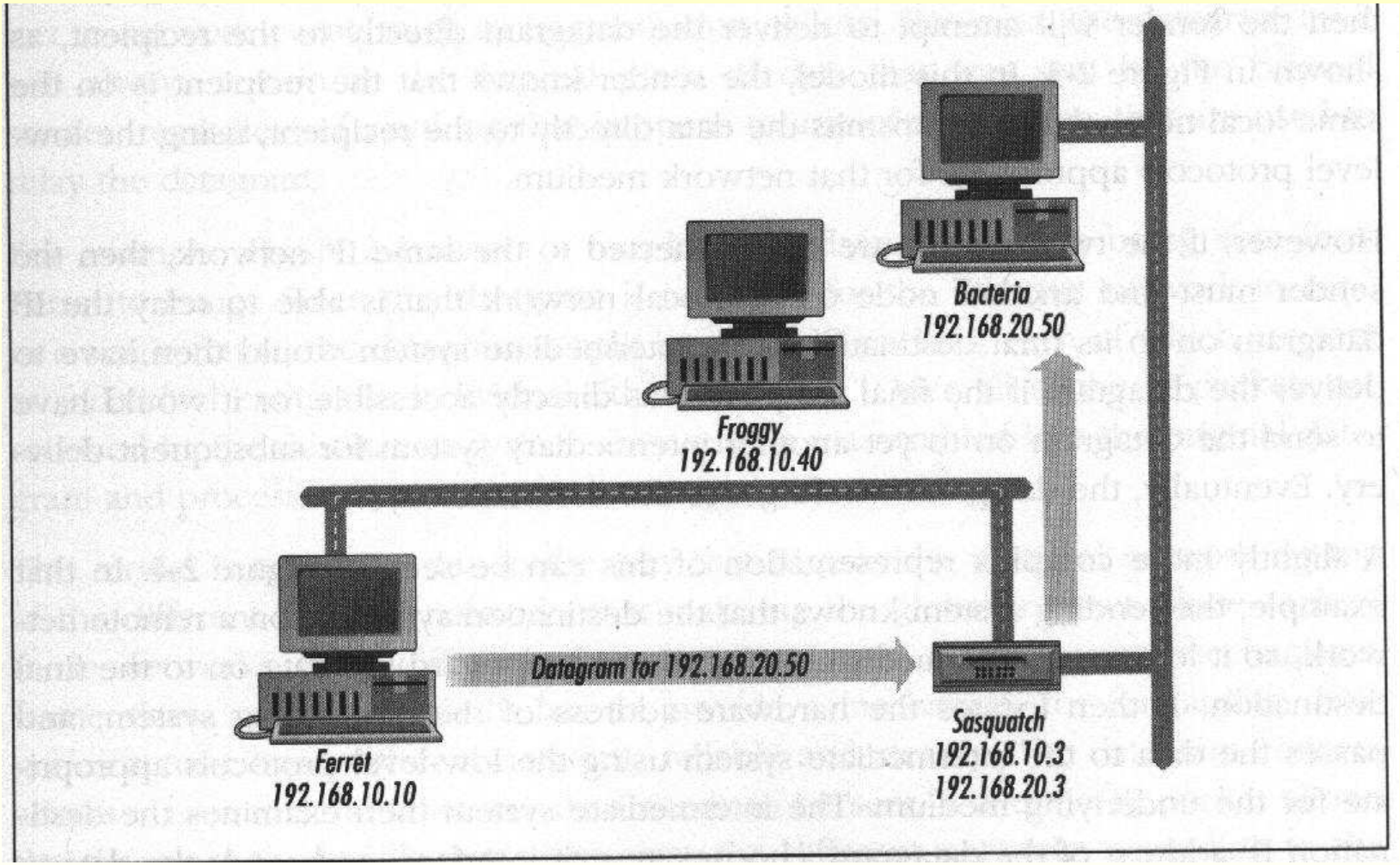
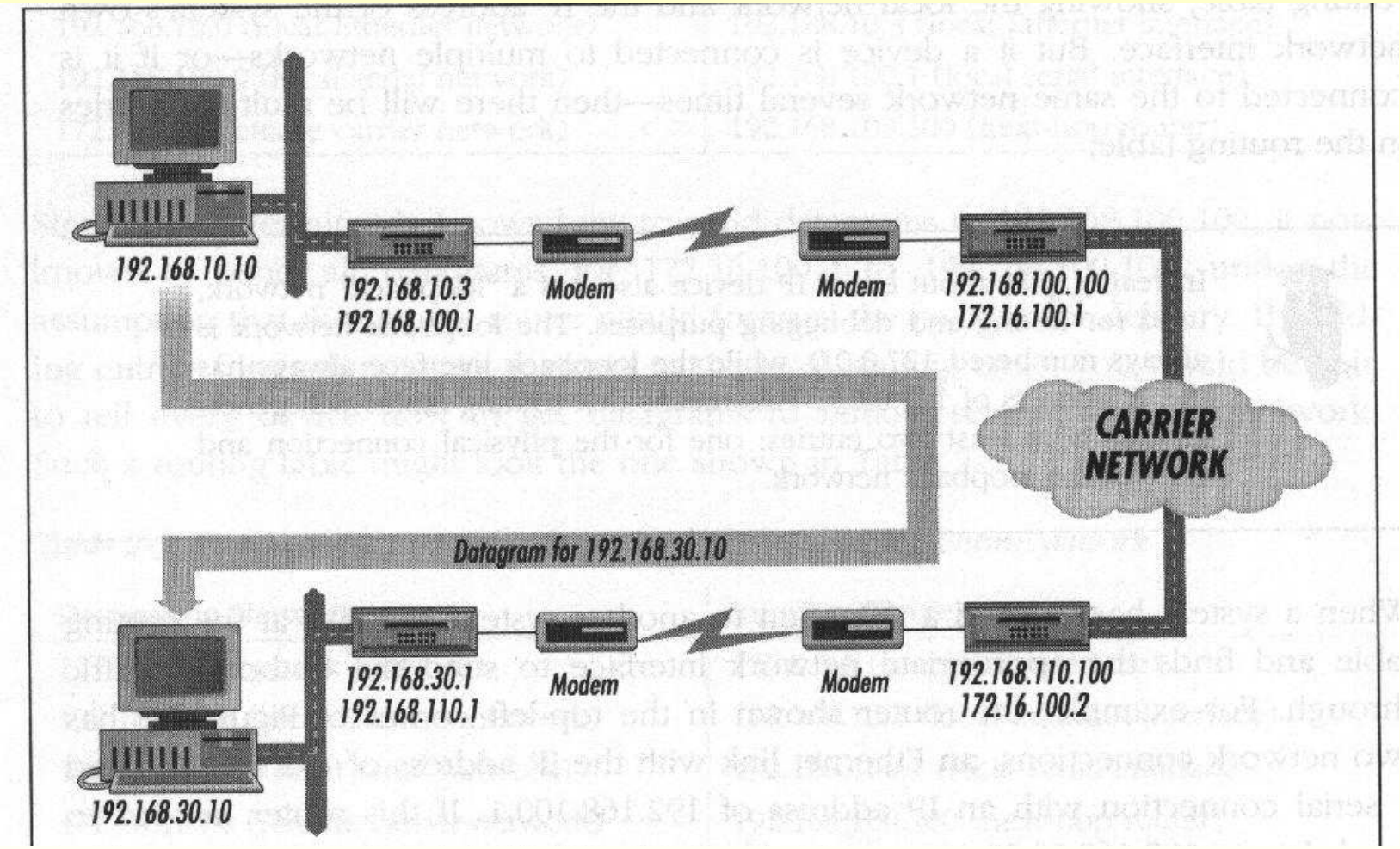


Figure 2-3. An example of local delivery

Routed Delivery



Multi-Hop Delivery



Adressing and Routing

one IP address per network interface

more than one $\rightsquigarrow$multihomed

internal routing table: which destination to which networks

loopback interface 127.0.0.1

default route

command: route, very different among UNIX systems

Simplification: Address Hierarchy

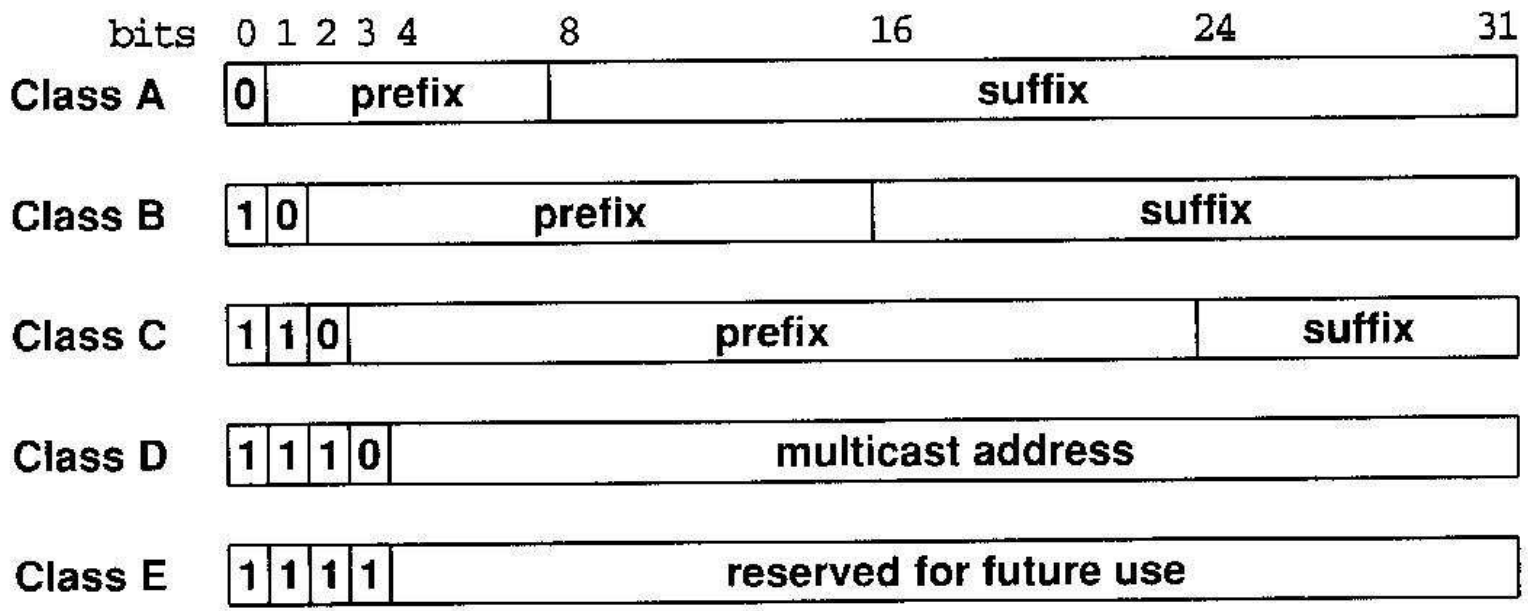


Figure 16.1 The five classes of IP addresses, where addresses assigned to hosts are either class *A*, *B*, or *C*. The *prefix* identifies a network, while the suffix is unique to a host on that network.

Simplification: Address Hierarchy

first byte decides on whether class A, class B, class C

First Four Bits Of Address	Table Index (in decimal)	Class of Address
0000	0	A
0001	1	A
0010	2	A
0011	3	A
0100	4	A
0101	5	A
0110	6	A
0111	7	A
1000	8	B
1001	9	B
1010	10	B
1011	11	B
1100	12	C
1101	13	C
1110	14	D
1111	15	E

Figure 16.2 A table that can be used to compute the class of an address. The first four bits of an address are extracted and used as an index into the table.

Simplification: Address Hierarchy

class	# networks	# hosts
A	128	16777216
B	16384	65536
C	2097152	256

exercise: write code to decide the class for a given IP address

Special Addresses

16.13 Summary Of Special IP Addresses

The table in Figure 16.7 summarizes the special IP address forms.

Prefix	Suffix	Type Of Address	Purpose
all-0s	all-0s	this computer	used during bootstrap
network	all-0s	network	identifies a network
network	all-1s	directed broadcast	broadcast on specified net
all-1s	all-1s	limited broadcast	broadcast on local net
127	any	loopback	testing

Figure 16.7 Summary of the special IP address forms.

IP Error Handling

very simple: destroy the packet

transient error: not caused by sender, nothing else happens

semi–permanent error: maybe fault of sender, send an ICMP error message

ICMP=internet control message protocol, sent over IP

IP Packet Format

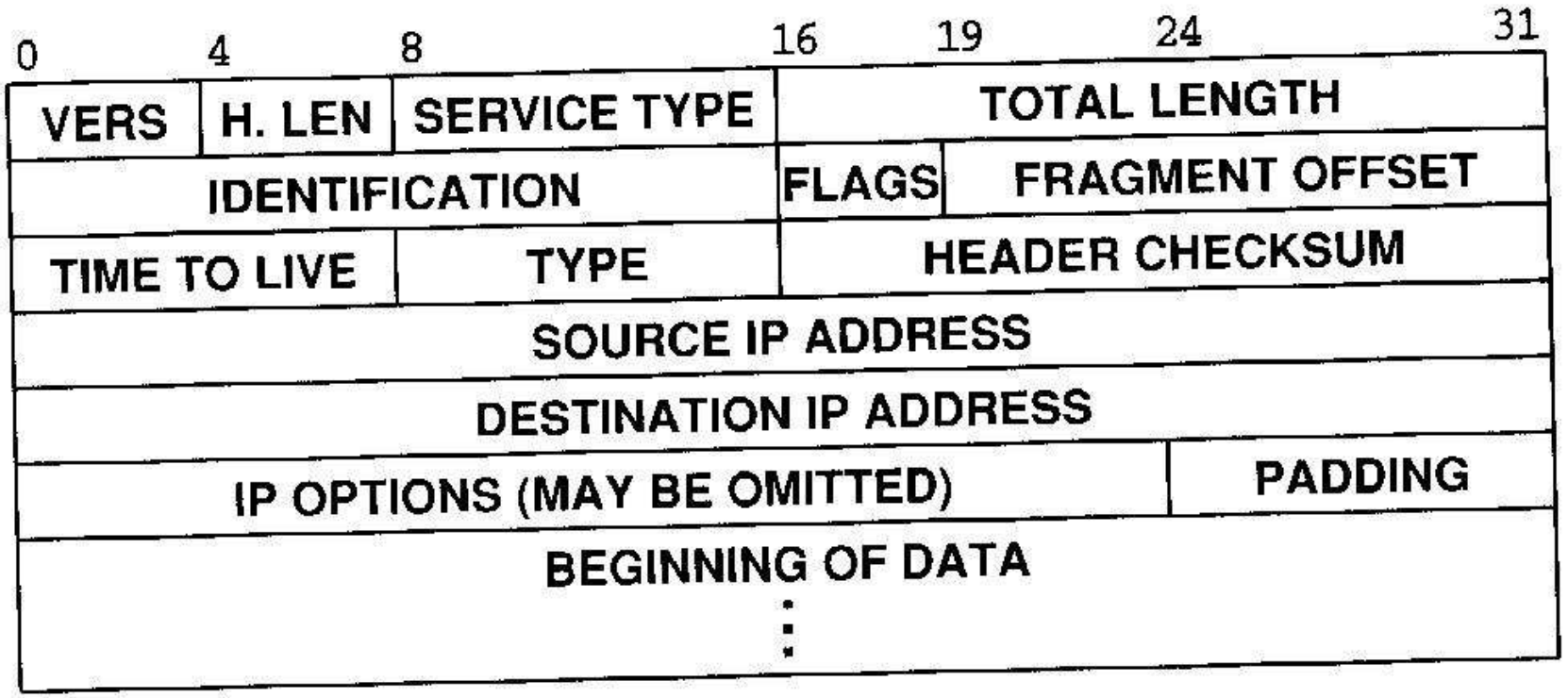


Figure 18.4 Fields in the IP datagram header. Both the source and destination addresses are Internet addresses.

Header Checksum

$$\overline{\sum u_i}, \quad u_i \text{ 16–bit words}$$

header most important part from IP's view

wrong checksum $\rightsquigarrow$no ICMP reply

how to implement the check?

why must the the checksum be recomputed at every router?

why no data checksum?

- computational overhead
- higher layer data (TCP/UDP) contained
- some application protocols can handle corrupt data

Time–to–Live (TTL)

on each forwarding device TTL gets decreased

destroy when TTL=0

$\rightsquigarrow$no loops possible

exercise: which TTL does your system use?

IANA suggestion is 64 (www.iana.org)

Fragmentation

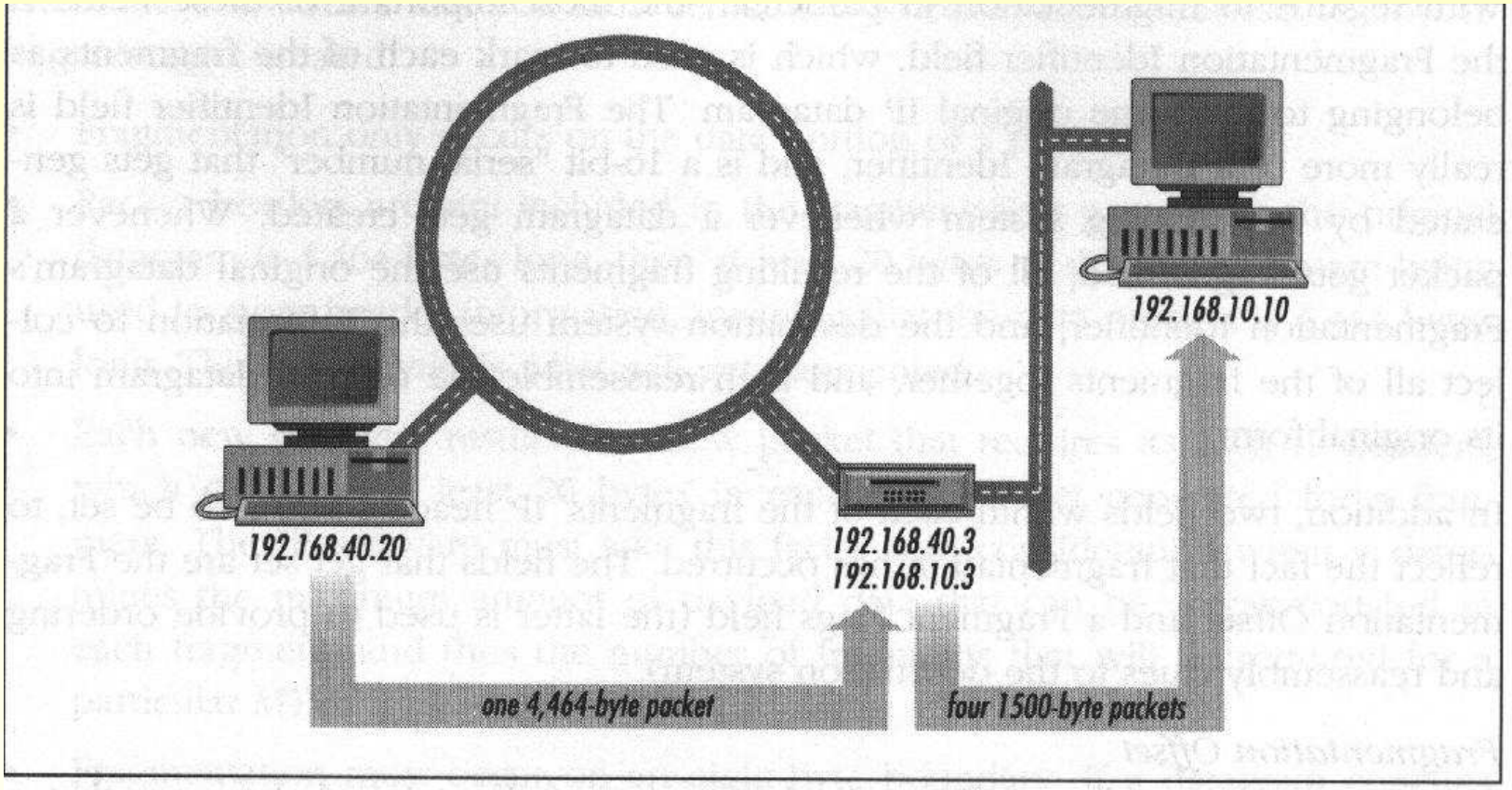
max IP packet length: $2^{16} = 65536$

max. transmission units (MTU) for layer 2

- Hyperchannel : 65536
- Ethernet and PPP: 1500
- IBM Token Ring: 17914
- X.25 and ISDN: 576

$\rightsquigarrow$IP packet must be splitted

Fragmentation Illustrated



Fragmentation Detailed

fragment flags

- the DF (=DON'T FRAGMENT) bit
- the MF (=MORE FRAGMENTS) bit

  decides whether current fragment is last fragment

⇝sequence important, need . . .

fragment offset = starting position in 8–byte blocks

every fragment is then an independent IP packet

(total packet length is now fragment length)

Fragmentation Rules

- the recipient assembles all fragments
- headers are not fragmented but newly generated
- multiples of 8 bytes
- each fragment contains same IDENTIFICATION field