IP layer - Internet Protocol

TCP/IP class

Jim Binkley

IP - Internet Protocol

- intro
- IP addresses
- subnetting
- header
  - fragmentation, ttl, options
- routing/algorithms/architecture
- CIDR/IPng

intro - IP version 4

- fundamental TCP/IP protocol
- RFC 791, other related RFCS
  - Inet checksum, rfc 1071, 1141, 1624
  - path mtu, rfc 1191
  - ip datagram reassembly 815
  - 1122, communications

ip layer - fundamental idea

- ip implements a **ip virtual network** on top of different kinds of hw where ip address is endpoint
- hw is hidden by network layer (except for a few things like MTU)

what does ip do (and not do?)

- sends and recvs packets to/from ip addresses - **ip datagrams**
- no retries, doesn't promise reliable delivery
- packets due to various reasons may be *lost*, *duplicated*, *delayed*, *delivered out of order*, *or corrupted* -
- **best effort** - don't lose them on purpose but only when nets busy - resources unavailable

ip functions

- route packets
  - routing: process of determining path for data
  - ip routes packets when they come to it from
    - » transport layer (down stack)
    - » link layer (up stack) - we are router and forward pkts
- fragmentation acc. to link-layer MTU
- handle ip options
- send/recv ICMP error and control messages

ip address

- 32 bits, “dotted-decimal” notation
  - 1.2.3.4, big-endian byte order, 0..255 is range
- associated with interface, not machine
- if machine > 1 i/f, then **multi-homed**
- if multi-homed, not necessarily router
- ip address in UNIX assigned to i/f with
  - #ifconfig ed0 inet 131.253.1.2 netmask 255.255.255.0

ip address structure

- each address has structure in it: (network, subnet, host)
- classically address consists of (net, host) portions
- **subnet mask** used to determine subnet part
  - taken from host bits
  - ipaddress & subnet mask

ip address table (net/host)

type	**prefix**	**bytes**	**range**
class A	0	1 net:3 host	1-126.h.h.h
class B	10	2:2	128-191.n.h.h
class C	110	3:1	192-223.n.n.h
class D	1110	flat	224..239
class E	11110	-	240..254

class D: multicast

class E: experimental (unused at present), note 255 used for broadcast

ip addresses - examples

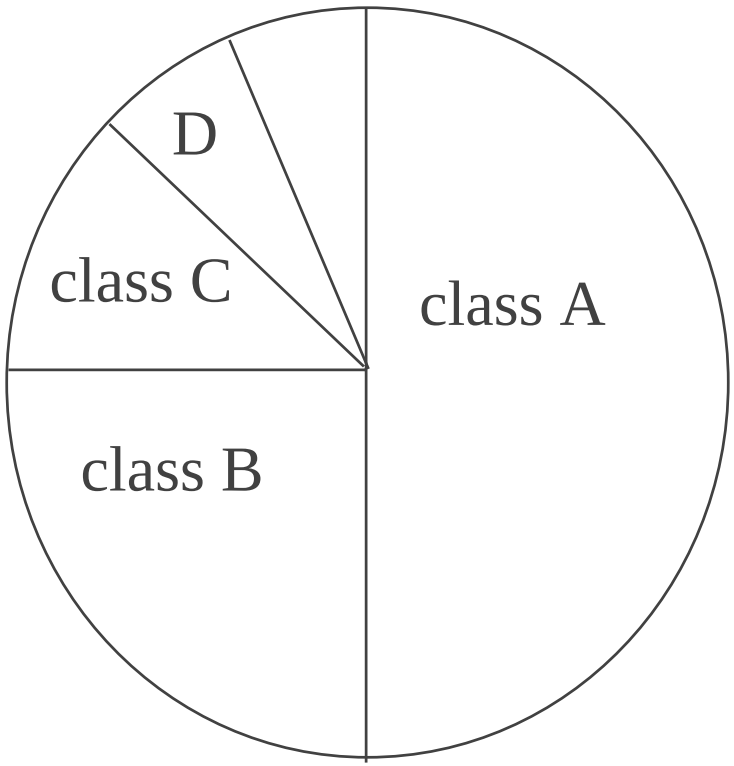
- 0.0.0.0 - if src, then boot == “this net, this host”
  if dest, old 4.2 BSD broadcast address
- 127.0.0.0 - localhost (loopback)
- 1.2.3.4 - class A
- 143.1.2.3 - class B
- 201.1.2.3 - class C
- 224.0.0.1 - multicast
- 255.255.255.255 - **limited broadcast**
- 200.0.1.255 - **directed broadcast** (assume subnet == class C part)

ip address - problems

- assigning class by bit means class A takes 1/2 of range, class B 1/4, class C 1/8, etc.
- problems with current setup
  - class assignment is wasteful
  - ip host addresses not necessarily utilized well
  - too many networks in core routers
  - running out of ip addresses ??

the ip pie - not a picnic

ip pie slices
acc. to Class



why was this a mistake?

Jim Binkley

subnetting

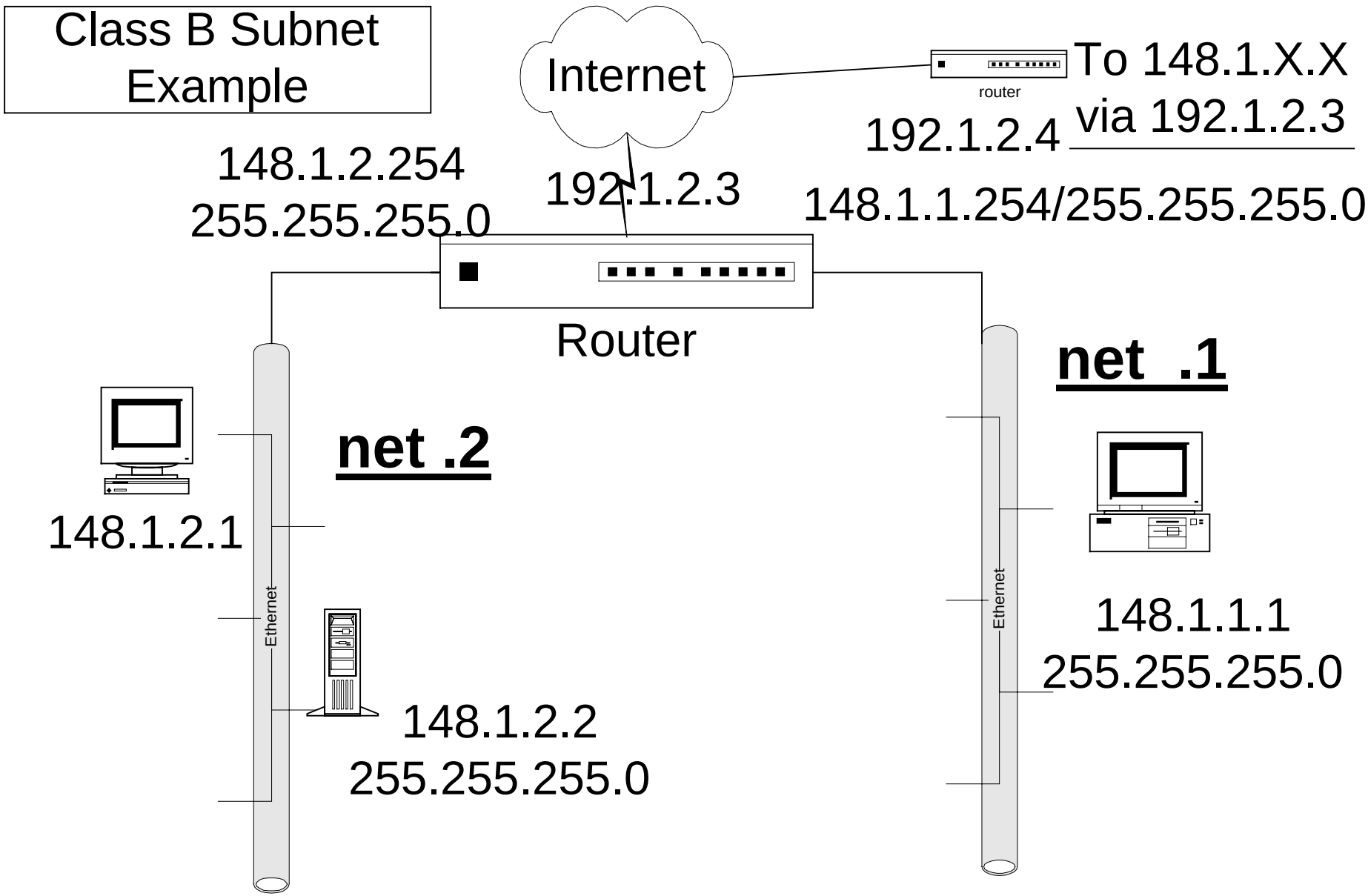
- subnet - use single IP network address to hide multiple physical nets
- subnet notion converts (net, host) into slightly more hierarchical (net, subnet, host)
- associate subnet mask with i/f ip address
- Example, class B, one byte of subnet:
  ip = 148.1.1.1 subnet=255.255.255.0

subnetting

- subnetting functions:
  1. you can subnet an ip address and split it up on separate networks across routers (conserve address space)
  2. you hide your routing structure from remote routers, thus reducing routes in their routing tables
- *if dest ip addr & subnet mask == my ip addr and subnet mask*

  dest is on same subnet

  else on different subnet (send pkt to router)


Class B Subnet Example
Internet
router
To 148.1.X.X
via 192.1.2.3
192.1.2.4
148.1.1.254/255.255.255.0
148.1.2.254
255.255.255.0
192.1.2.3
Router
net .1
net .2
148.1.2.1
Ethernet
148.1.2.2
255.255.255.0
Ethernet
148.1.1.1
255.255.255.0

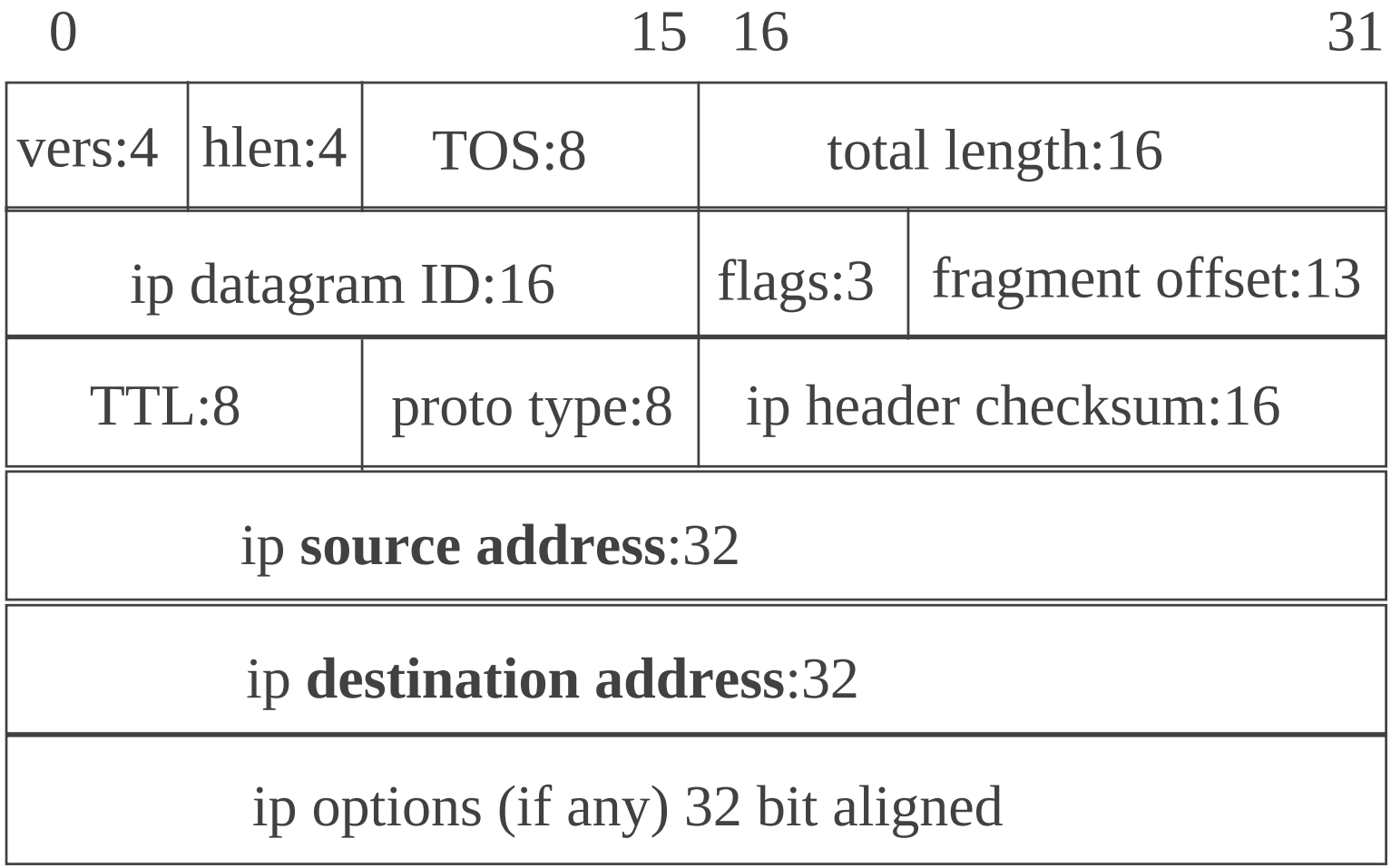
subnetting homework assignment

- assume your ISP gives you 1 class C network address, 200.0.1.X
- assume your router's ip address to outside world (Internet) is 134.3.1.2
- design a subnetting scheme that will allow approximately 30 hosts per subnet
- draw a picture, show ip addresses and subnet masks for all router ports. Assume the router ports == #subnets + 1
- show one host on one of the subnets with ip address, subnet mask, and broadcast address

ip encapsulation

20 bytes (no options)

ip header


0
15
16
31
vers:4
hlen:4
TOS:8
total length:16
ip datagram ID:16
flags:3
fragment offset:13
TTL:8
proto type:8
ip header checksum:16
ip **source address**:32
ip **destination address**:32
ip options (if any) 32 bit aligned

ip header

- ip version == 4
- header length in 32-bit words, h == 5 with no options (20 bytes)
- type of service and precedence
  - not used much in past but starting to be used
  - bits 0-2, precedence
  - bits 3-5, TOS, hint to routing about how to queue
    - » D (bit 3) - low delay (telnet),
    - » T (4) - high thruput (FTP), R (5) - reliability

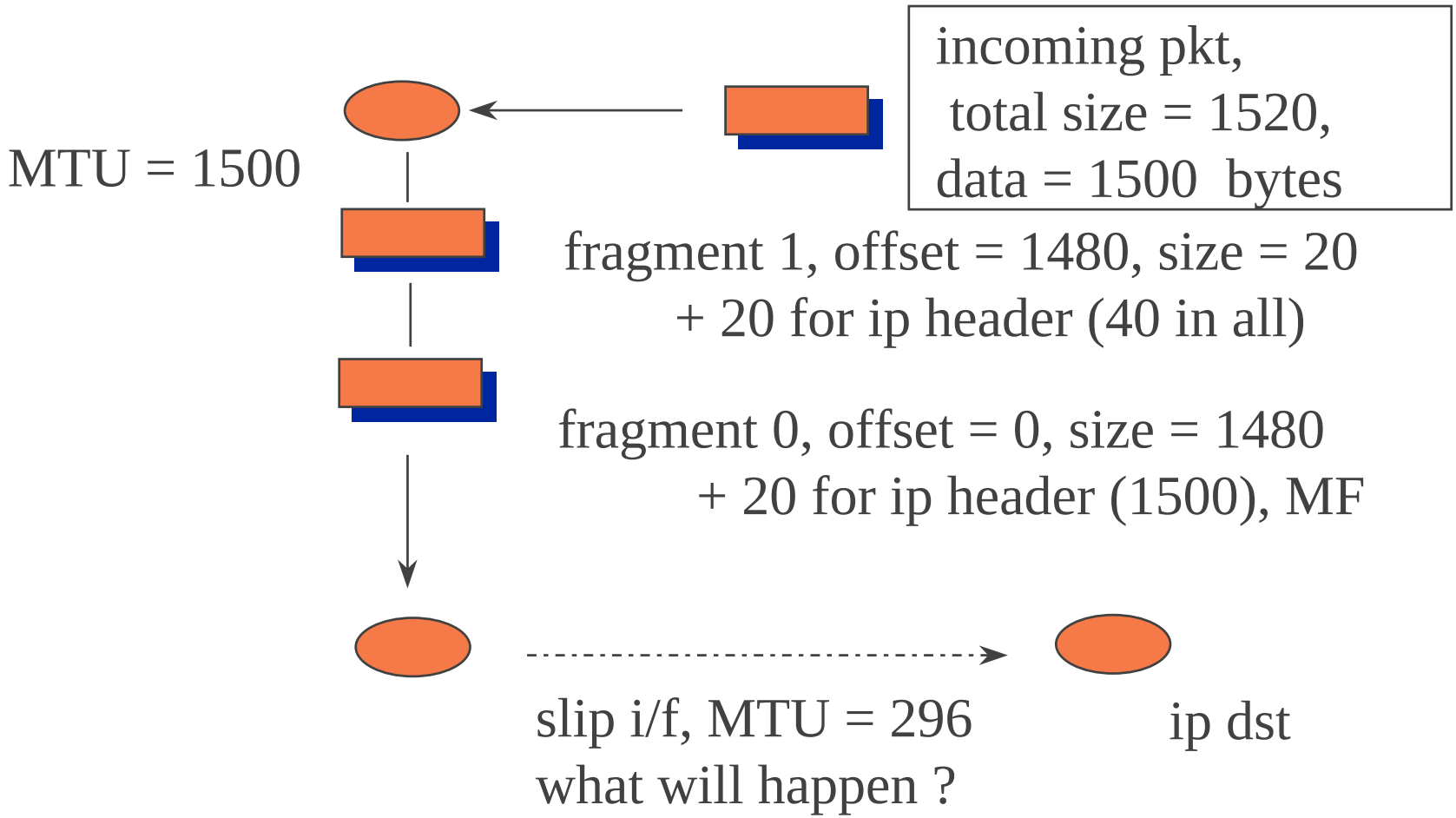
ip header

- total length - max ip datagram is 64k
- fragmentation
  - **fragment ip_id stays the same for all fragments**
    - » ip_id ids the logical IP datagram and all its parts
  - flags (DONT_FRAGMENT, MORE_FRAGMENTS)
  - fragment offset from 0 start of packet, e.g.,
  - 0, 0x400, 0x800
  - ip length is length of fragment, not total datagram

fragmentation - how it works

- ip fragments because outgoing packet is too big for MTU of i/f
- fragments must be reassembled at final ip destination and can be fragmented again on way
- if any fragment lost, all of datagram must be resent (not by IP)
- IP uses best effort even to allocate internal buffers
- TCP tries to avoid, UDP not smart enough
- IP fragmentation not a STRONG mechanism

ip fragmentation



ip_id, ip_src retained in all (new) fragments

even more fragmentation

- reassembly done at ultimate destination
  - pros:
    - » simplicity - fragments can be routed independently
    - » simplicity - intermediate routers don’t have to store
  - cons:
    - » any fragment lost, entire datagram lost
    - » philosophical - is this really hiding the link layer ?
- path MTU is a way around
- note: routers may not see all fragments

fragments and the MSS

- MSS - **maximum segment size** == 576
- IP fragmentation up to this size guaranteed to work
- UDP apps often do not exceed this size
- TCP often uses PATH MTU, ignores this size - TCP will try local MTU (often 1500) and try to make it work end/end

ip header

- proto type - TCP, UDP, ICMP
- checksum
  - over header only, useful?
  - same algorithm used by tcp/udp
  - with ip itself, only over header
    - » deemed not useful in IPv6
  - routers must redo IP checksum since ttl changes

ip checksum

- sender
  - ip checksum field = 0
  - add together by 16-bit words and take one's compliment (1's compliment arithmetic)
  - store in ip cksum field
- recv
  - adds N sections, complements result, should get 0, else error

ip header - ttl

- TTL - time to live, actually hop count, not time
- when packet crosses router
  - ttl--
  - if ttl == 0
    - » discard and send ICMP ttl exceeded to ip src
- **important guarantee that datagrams will be discarded even if network loops**

ip options

- not very used and possibly not very useable
- variable length encoding mechanism
- form of TLV or tag/length/value (data)
- options come in multiples of 32 bits
- ignore packet format details
- pro: extensible format
- con: not as easy to parse as fixed format

option types

- end of option list
- noop - used for alignment
- military ?
- loose source routing: specify inexact path
- strict source routing exact path (with ip addresses)
- record route - possibly useful
- gather timestamps

sassing options

- encoding is not efficient for routers
- length is limited by IP header length - not big enough for size of Inet diameter
  - 60 bytes max, if 4 byte IPs == 15 ip addr total
  - e.g., record route could not work > 15 hops
- source routing not secure -- someone could stick in a intermediate route and spy on your packets

routing

- **routing - the process of choosing a path over which to send datagrams**
- hosts and routers route
- input: ip destination address
- output: next hop ip address
  and internally an interface to send it out
- routing does not change ip dest address

how do routes get into routing table?

- static routes - by hand, on unix with
  % route to_dest via_next_hop
- dynamically via routing daemon, routed or gated on UNIX, protocols=RIP/OSPF/BGP
- via ICMP redirect

show routing table

- unix host
  - % netstat -rn
    - » n is for NO dns, else you may cause DNS queries
- cisco router
  - (router) show ip route

routing table

- entries logically
  (destination, mask, via gateway, metric/s)
- destination - network or host address
- mask - subnet mask for dst address
- via gateway - next hop (maybe router)
- metric/s - depends on routing table algorithm and dynamic routing protocols

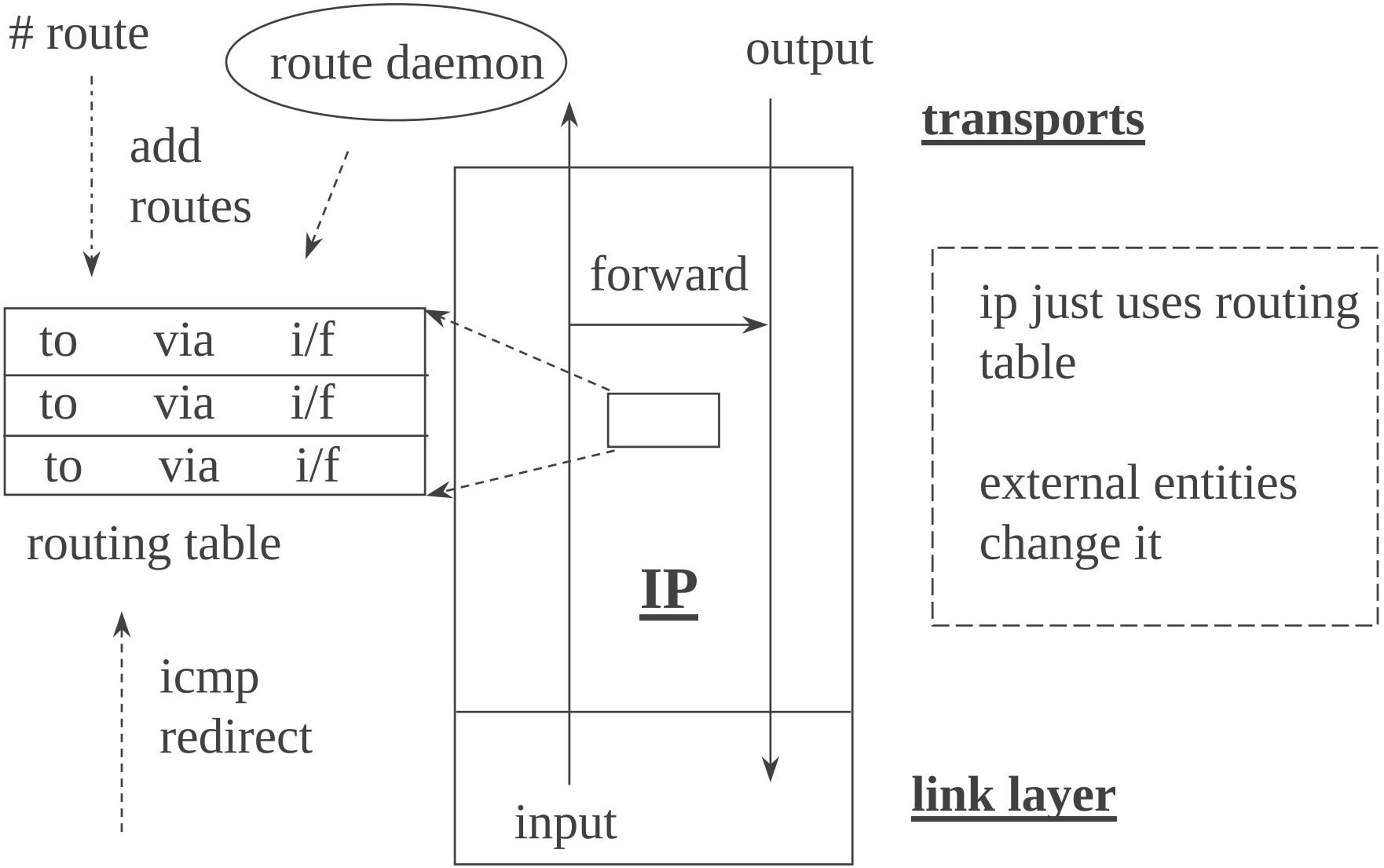
manual adds to routing table

- on FreeBSD unix host:
  - # route add default 204.1.2.3 (default route)
    - » internally default has value 0.0.0.0 (all dsts)
  - # route add 1.1.1.1 2.2.2.2
    - » 2.2.2.2 is the next-hop router for 1.1.1.1
    - » we must have direct connection to 2.2.2.2 (i/f must be on same subnet and must exist)
    - » # ifconfig ed0 2.2.2.1 (our i/f must exist)

SOME possible kinds of routes

- host, 210.1.3.21/32 (to specific host)
- subnet, 131.253.1.2/24 (to specific subnet)
- network, 131.253.0.0/16 (to specific net)
- default route - normally the router on a net, send it here when nothing else matches
  - expressed internally as 0.0.0.0
- note: default route to host route - least specific to most specific (natural ordering)

routing architecture



netstat -rn - UNIX example

Destination	Gateway	Flags	Rfct	Use	if
127.0.0.1	127.0.0.1	UH	4	541053	lo0
131.252.20.0	131.252.20.183	U	148	225888	le0
131.252.21.0	131.252.21.183	U	92	182083	le1
default	131.252.20.1	UG	1	12	le0
131.252.10.17	131.252.20.2	UGHD	0	10089	le0
192.220.224.0	131.252.20.1	UG	0	0	le0
192.147.168.0	131.252.20.1	UG	0	0	le0
158.104.0.0	131.252.20.1	UG	0	0	le0
192.147.160.0	131.252.20.1	UG	0	0	le0

note: and more, RIP in use

routing algorithm/s

- there is no *one* routing algorithm
- over time, notion that match should go with longest prefix has come into being
- default < net < subnet < host < broadcast
  - host is longer prefix than net than default
- algorithms vary from (too) simple to hardware-assisted (commercial routers)

some examples

- dumb pc routing algorithm (example)
- linux (1.2.x/1.3.x) - linear search
- old BSD style, 2 tables, host > net and routes to i/fs in table, hashing for speed
- new BSD style, Patricia tree algorithm, supports longest matching prefix
- commercial routers - proprietary, hw cache support

dumb pc algorithm

assumptions:

consider “to” part, the KEY for every entry

Assume one i/f and we have its subnet mask

we have one default router ip address

algorithm:

if (ip dst & subnet mask) == (KEY ip & subnet mask)

do “direct” routing; i.e., deliver to i/f

else

do “indirect” routing, send to default router

Note: this algorithm implies you can’t talk to a different subnet on the same link

supernetting/CIDR

- early 90’s -- too many class B addresses given out - running out of ip network addresses?
- decision made to allocate blocks of class C instead - many nets at one set; hence **supernetting**
- downside would be more routes in routing tables

Classless Internet Domain Routing

- pronounce “cider” -> do away with classes
- allocate contiguous power of 2 blocks
- represent in route table by (net, mask) where the mask indicates a range of addresses
- think of this in simplified form as {base+ offset}; e.g.., net 4, +4 ([4..7] inclusive)

CIDR continued

- **routing algorithm must support longest prefix match for destination**
- more specific better than less specific router (host route better than subnet)
- put another way, /32 better than /24
- caused rewrite of most os routing algorithms and dynamic routing protocols
  - e.g., BGP-3 became BGP-4

thus at least two CIDR tricks

- **1. supernetting -** aggregation “up”; i.e.steal bits from ***network*** portion
- e.g., multiples of class C nets
- **2. subnetting** (aka **Variable Length Subnet Masks**); i.e., steal bits from the ***host*** portion
- VLSM can be used to **subnet a subnet**
  - e.g., class C assumes 24 bits of mask
  - divide that up into half (/25) and half again (/26)

network range notation

- express CIDR block in **slash notation**
- ip prefix/length of net mask
  - assume contiguous bits, never discontiguous
- class A, thus 1.0.0.0/8
  - netmask is 255.0.0.0
- class B, thus 128.1.0.0/16
  - 255.255.0.0
- class C, 192.1.2.0/24 (255.255.255.0)

CIDR conversion table

/15	255.254.0.0	2 class B
/17	255.255.128.0	128 class C
/18	255.255.192.0	64 class C
/24	**255.255.255.0**	**1 class C**
/25	255.255.255.128	1/2 class C
/26	255.255.255.192	1/4 class C
/31	255.255.255.254	1/128 class C

VLSM can be used

- to subnet subnets, ad infinitum
- e.g., given 192.1.2.0/24
- can break up class C subnet into two parts
- 192.1.2.0/25 and 192.1.2.128/25
- student exercise: take 192.1.2.0/25 and break it up into two /26 subnets
- this works as long as all boxes speak CIDR

examples: explain and think about

- 131.252/16 (PSU class B)
- **131.252.208.0/20** would mean what? range is?
  - supernet or VLSM? how much of class B is this?
- 131.252.215.3/32 (host route)
- 131.252.215.0/24 (VLSM or supernet?)
- 131.252.215.32/28, range is?
- 207.98.0.0/17 - how many class C addr?
  - range on latter is 0.0 - 207.98.127.255
- 0.0.0.0/0

assume previous slide is routing table with IP destinations

- which entry is chosen for each dst?:
- 1. 131.252.215.3
- 2. 131.252.215.4
- 3. 131.252.215.34
- 4. 131.252.215.0
- 5. 131.252.216.1
- 6. 131.252.1.1
- 7. 207.98.102.128, or 8. 207.98.240.1

NFS/UDP dump - decode IP

etherfind -x -v -between nfsserver nfsclient
1. UDP from server.2049 to client.1022 8300 bytes (only 1480 bytes long) (more fragments)
08 00 20 02 20 f2 00 00 3c 00 19 56 08 00 <- ethernet
45 00 05 dc ba 65 20 00 1e 11 91 2e 8f b9 06 01 8f b9 06 0a
(udp, etc...)
find ip tos, len, id, flags, offset (0), ttl, proto, src/dest ip

2. UDP fragment offset = 1480, length = 1480, MF set (ignore ethernet, look only at IP)
45 00 05 dc ba 65 20 b9 1e 11 90 75 8f b9 06 01 8f b9 06 0a

NFS/UDP - IP fragmentation

3.4.5. 3 fragments, length = 1480, MF set

last fragment:
6. UDP fragment offset = 7400, length = 900
45 00 03 98 ba 65 03 9d 1e 11 af d5 8f b9 06 01 8f b9 06 0a

note: MF is off, id still == ba65

study questions

- decode previous NFS/UDP dump at ip layer
- construct class C subnet as asked previously
- ISP gives you two class C subnets but you have ONE wire. How do you make all hosts talk to each other?
- take previous class C/VLSM example and divide /25 into /26. what happens if you run RIPv1 on that net?
- refer to the previous class B subnet picture. If the subnet mask is wrong and == 255.255.0.0 and a host on one subnet tries to talk to another, what would happen (assume dumb pc routing algorithm)?

more study questions

- what would 0.0.0.0/7 mean?
- why is 1.0.0.0/7 wrong? (think about it in terms of base 2 and draw a picture of the bit/s)
- say you have network 131.252.215.0/24
  break it up into 8 subnets of equal size
  - what are ranges and directed broadcast addresses?
- take previous 8 subnets and make it into 7 subnets, 1 of which has twice the host IPs
- make up an example where you use supernetting