Routing – review/intro

TCP/IP class

Jim Binkley

outline

- traceroute question
- review of a few IP fundamentals
  - IP addresses
  - routing table
  - supernet/VLSM
- cisco routing table evolution

traceroute across cisco routers

% traceroute big-baby.cs.pdx.edu

1. wintermute.seas.pdx.edu 0.858 ms 0.663 ms 0.631 ms
2. deepthot 0.799 ms 1.119 ms 0.834 ms
3. skynet 1.225 ms 0.729 ms 0.605 ms
4. dexter 1.814 ms 1.318 ms 1.264 ms
5. radia 3.086 ms 1.637 ms 1.675 ms
6. tony 3.454 ms 1.871 ms 1.840 ms
7. yakov 3.619 ms 2.148 ms 2.221 ms
8. big-baby 4.294 ms * 3.012

analysis?

- **1. what is the fundamental anomaly in the traceroute?**
- **2. what is the explanation for said anomaly?**
- 3. many of the routers in the traceroute are small Cisco 2621 routers (still there in the lab)
- 4. big-baby is fundamentally not relevant to this question – why?

another question:

- you can take a Linux or FreeBSD PC and turn it into a router
- or you can buy a Cisco
  - 3750 switch/router
  - 12008 gigabit switch router
- how might these things differ?
  - in packet flow for routing?
  - in system hw/os architecture?

a few issues to consider in terms of router architecture

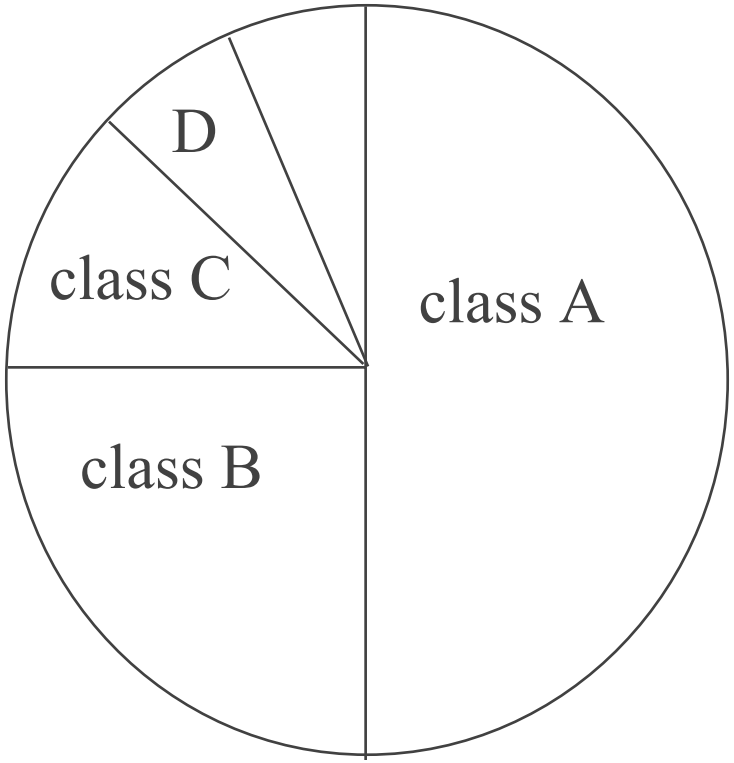
- does it have a hard disk
- what kinds of parallelism exist for packet flow from port to port
- how many ports, what kinds of interfaces are supported
- redundancy in terms of power, CPU, ports, etc – e.g., 24x8 always up supported?
- uplink bigger than downlinks?
- how are ACLs handled
- what is the routing algorithm anyway?

classful addressing vs classless addressing

- with IPv4 we have 2 kinds of addresses
- classful – acc. to old class A, B, C, D, E
- classless – use of CIDR/netmask to create group of contiguous block
  - /17 with class A or class C chunk
  - PSU class B /16 as example

the ip pie - not a picnic

ip pie slices
acc. to Class



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

classless/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)

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 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

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?)
- 131.252.215.64/26, 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

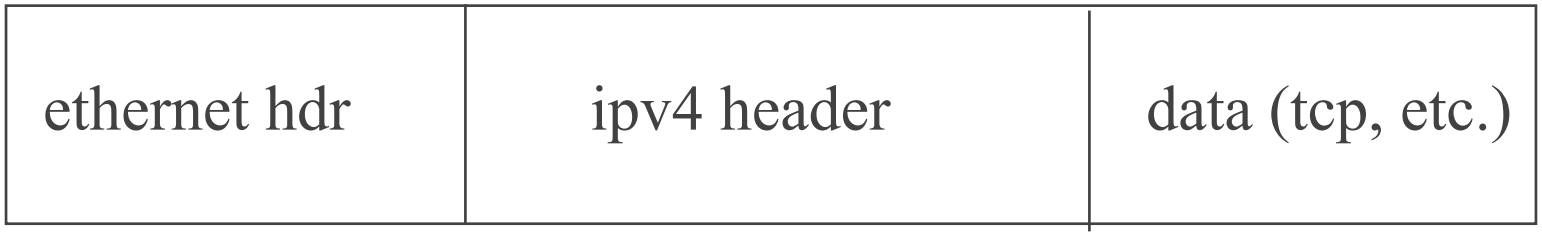
VLSM can be used

- to subnet subnets, ad infinitum
- e.g., given 131.252.215.0/24
- break it up into 4 IP subnet subranges
- slash what?

keep in mind: principle of longest prefix match

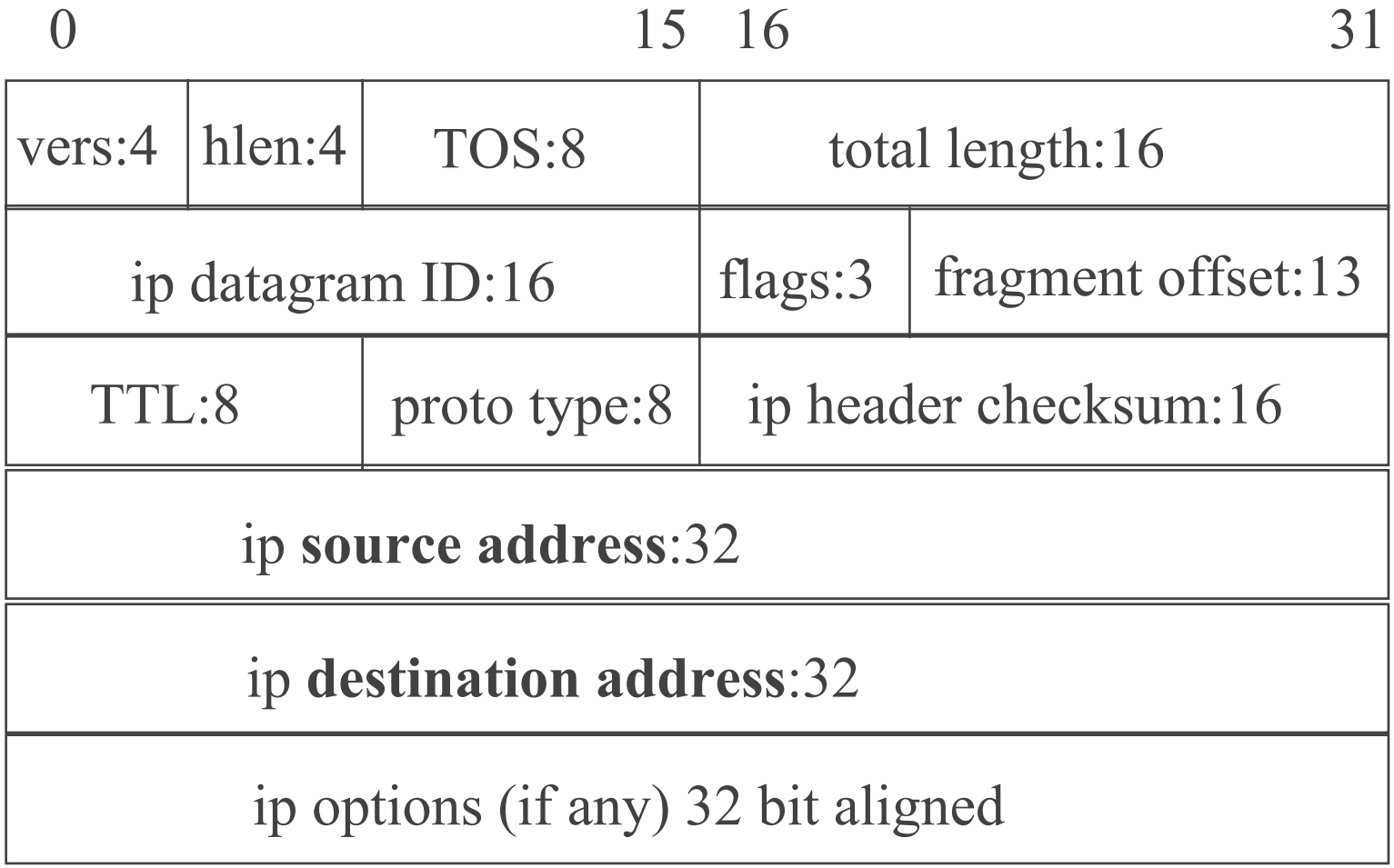
- **classful -> classless how**
  - **observation:**
- **routing algorithm must support longest prefix match**

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 features

- IP addresses are at fixed offset
- proto type available - TCP, UDP, ICMP
- checksum
  - over header
  - must be recalculated by routers
- options – best if none …
- routers may fragment – best if they don't

ip checksum

- sender
  - ip cksum 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**

routing and algorithms

- **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
- IOS-based cisco router
  - (router) show ip route

routing table

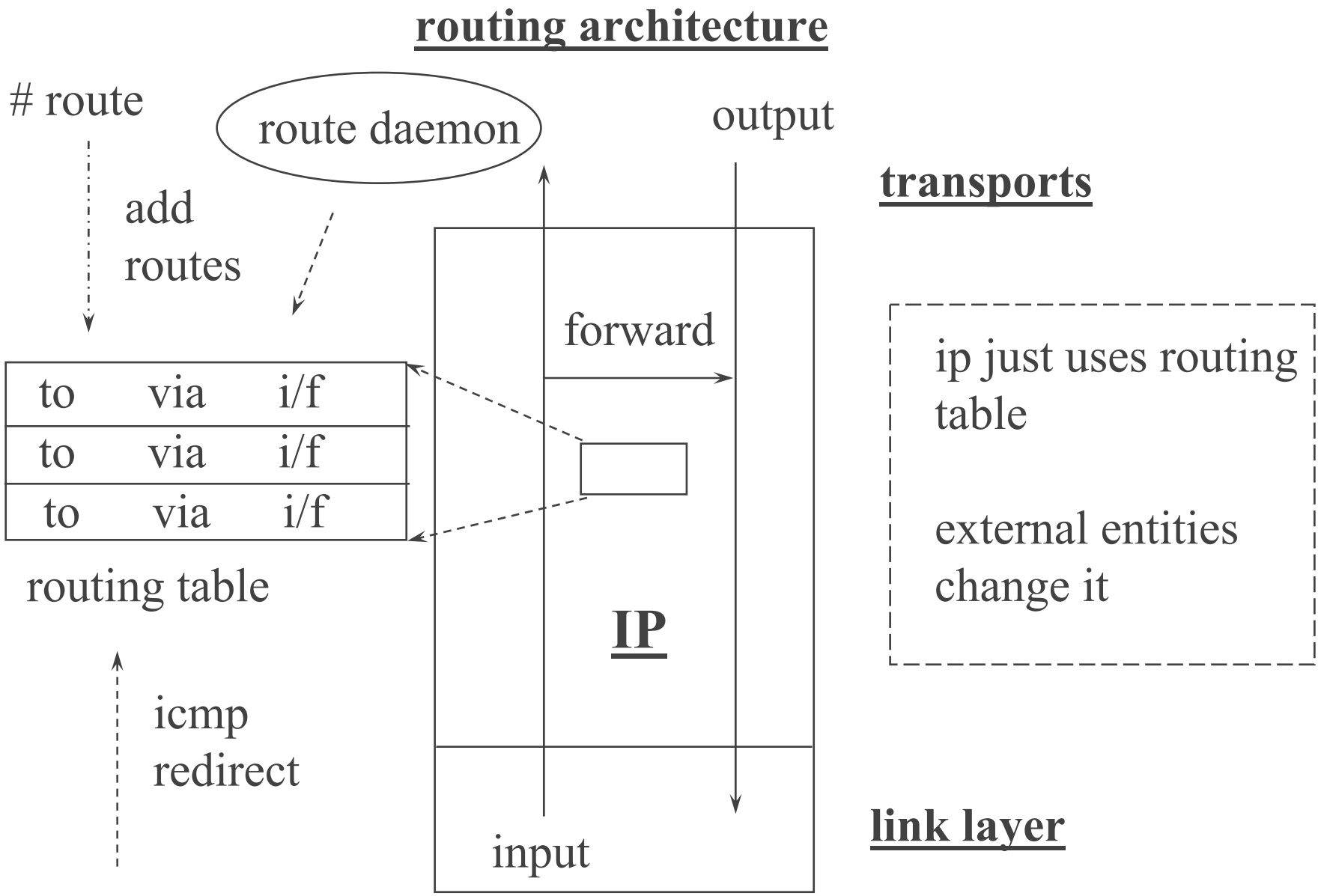
- entries logically
  (destination, mask, via gateway, metric/s)
- destination - network or host address
- mask - subnet mask for dst address
- via gateway - next hop (maybe router)
- metric/s - depends on routing table algorithm and dynamic routing protocols

manual adds to routing table

- on FreeBSD unix host:
  - # route add default 204.1.2.3 (default route)
  - # 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
route
route daemon
output
transports
add
routes
forward
to via i/f
to via i/f
to via i/f
routing table
IP
ip just uses routing table
external entities change it
icmp
redirect
input
link layer

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
- algorithms vary from (too) simple to hardware-assisted (commercial routers)

question/s:

- what features may routing tables have?
- if they have them, why?
  - note:
  - types of routes and types of routing tables
  - whether or not you have default route
  - can you have > 1 default route?
  - what about the path through the router?
  - what about the impact and or efficiencies/advantages of different kinds of routing algorithms?

routing algorithms in some depth

- dumb pc routing algorithm (in TCP class)
- 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

evolved classful routing algorithm

```
if host specific match
        send to next hop
else if IP dst subnet match and we have port on that subnet
        send to that subnet/if
else if network match
        send to that network
else no match
        if I have default route
                send to that
else
        drop packet
        send ICMP unreachable
```

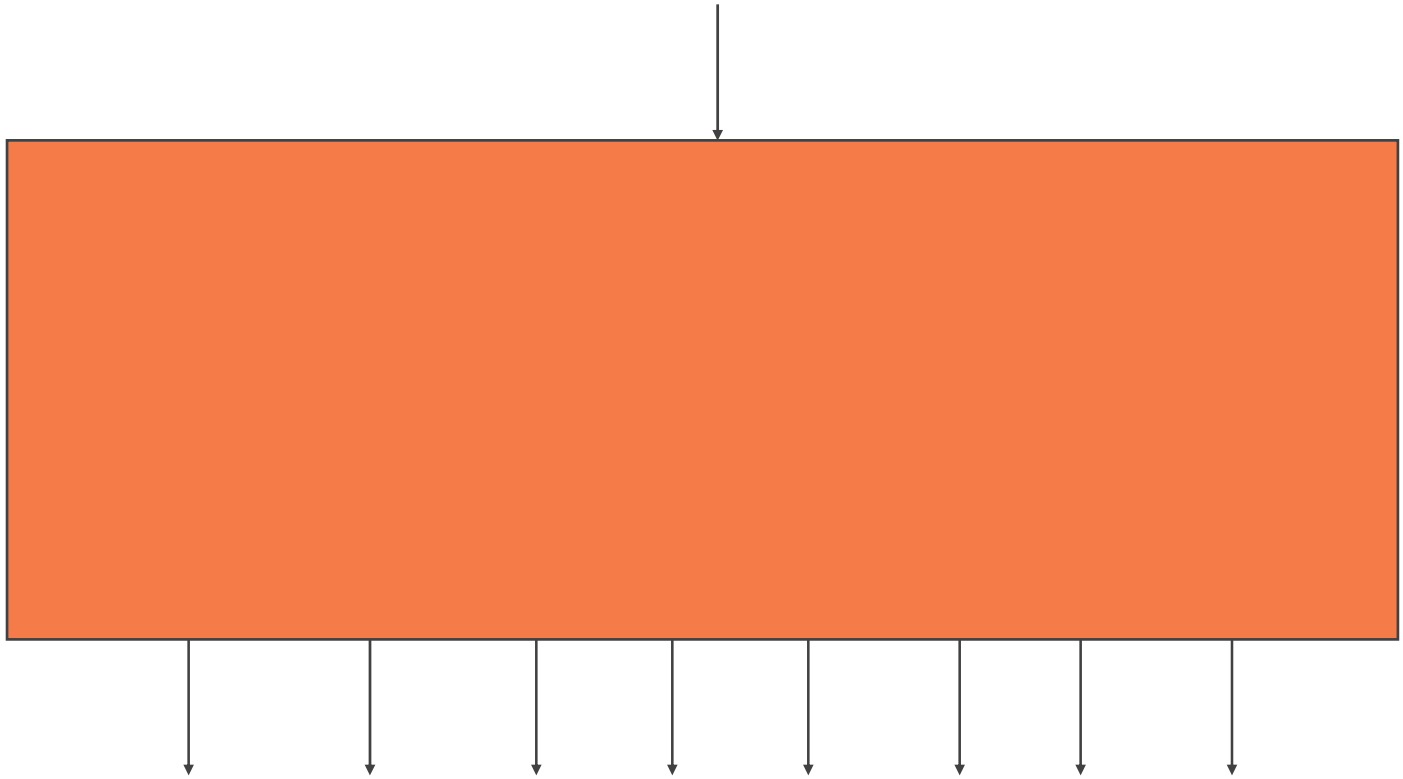
notes:

- typically IP dst, route entry cached and we check cache BEFORE we invoke this alg.
- the above may be classful BUT what does it imply in terms of the order of lookup
  - more bits to lesser bits
  - default is no bit match at all
- one real question: how do we efficiently cache the IP addresses?

flow can mean different things:

- but in a router or switch it means:
- we have many Ethernet ports
- an efficient way to match one input to many possible outputs
- ideally we map:
  - IP dst to a Layer 2 port
  - (IP dst, L2 switch port)
- be aware that this makes the L3 router/L2 switch distinction fuzzy with Ethernet ports

logically 1 input – many possible ouputs



and many repeats of same IP src/IP dst

classless routing algorithm might take this form:

- host entry has a mask with all 1's.
- default has mask with all 0s
- subnet (intermediate) has appropriate subnet mask
- routing tuple: IP dst, mask, gateway, metrics, next hop port
- now do longest-prefix match, caching
  - **in parallel in sense of N entries, find the best one**

BSD tree lookup algorithm

- Wright/Stevens, TCP/IP Illustrated, vol 2., pp 559-1995 Addison/Wesley ISBN
- or see the original USENIX paper:
- K. Sklower, “A Tree-Based Packet Routing Table for Berkeley UNIX”, USENIX, Dallas TX, 1991
- Patricia tree/trie/radix lookup – more or less the same

classful predecessor had 2 tables

- a host table
- a net table
- given IP address X
  - first lookup in host table
  - then lookup in net table
- so essentially this was evolving towards masks and longest prefix lookup
- hashed base lookup

trie/radix organization

- routing table (or cache) organized as binary tree
- each entry has netmask
- we match search key if search key anded with mask of entry matches the entry itself
- tree = nodes + leafs of course
- test bits and backtrack if host leaf found but does NOT match ip dst key
- try this for yourself with a 4 bit address

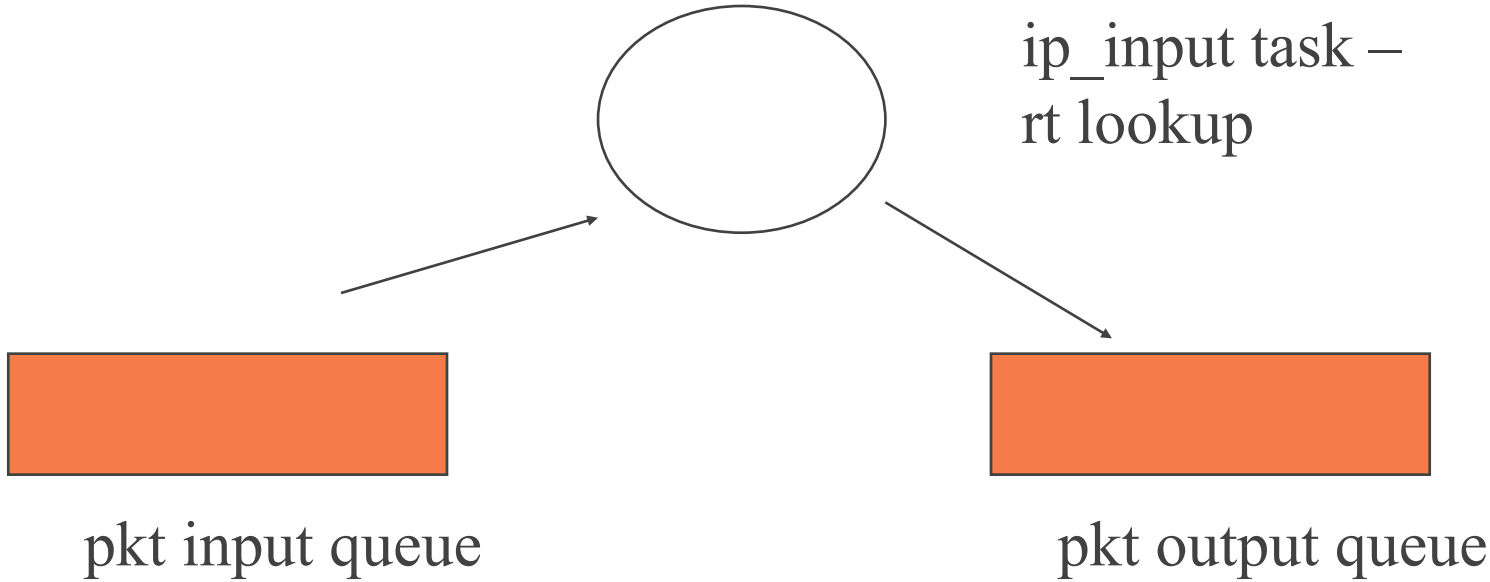
results:

- Sklower showed that radix tree was 4 times faster than hash
- this algorithm is more or less still used
- in BSD, route caches have existed but their form is based on
  - TCP (L4) control block
  - i.e., TCP routes packets towards the dst and caches the routing table lookup
  - so not fast path through L3 router

Cisco routing architecture intro

- from Inside Cisco IOS Software Architecture, Bollapragada, etc. Cisco Press 20000
- look at
  - process switching
  - fast switching
  - CEF – Cisco Express Forwarding
- note: Cisco IOS is a traditional embedded os with tasks, and no MM unit

process switching – input path



steps in the process

- 1. network i/f is interrupted and puts pkts in input Q memory
- 2. interrupts CPU
- 3. ip_input as task must run
  - must compete with other tasks
  - therefore scheduled
- 4. ip_input must make rt table lookup

ip input task

- must acquire
  - next hop i/f
  - mac address of next hop i/f (this is src mac)
  - mac address of next hop ITSELF (dst mac via arp)
  - rewrite MAC header in i/o memory
- assuming memory exists, queue pkt for interrupt-driven output

cons:

- 1. because i/o task must be scheduled this is slow
  - note: PC o.s. has advantage here in that all this work can be done without scheduling a task
  - task must switch in, previous task switch out
- question of performance as rt tab size increases (150k rt tab entries?)
- memory copies – how many, how costly?
- question of ECMP, > 1 rt table entry to same destination, how to handle?

1st order improvement: fast switching

- when can we cache and what do we cache?
- arp/MAC/next hop info …
- we want to AVOID ip_input when possible
- therefore Cisco developed: **fast switching**

fast switching algorithm:

- ip_input post lookup
  - caches needed info in fast cache
- interrupt software
  - searches fast cache first
  - if found
    - » rewrite MAC header
    - » hand pkt directly to output Q
- goal: route once, forward many times
- now can you explain the traceroute?

problems:

- arp table may change, must invalidate associated cache entries
- routing table may change, must etc.
- of course may see flows we haven't seen before
  - consider SQL slammer attack with rotating L4 IP addresses, changing L4 dst ports
- ios command: # show ip cache verbose

evolution in cache structure

- 1. first implemented as hash
- 2. 2nd implemented as trie
- 3. algorithms had problems supporting load balancing at L3
- 4. still can't deal with 1st packet though

optimum switching – 1 evolutionary step

- important idea:
- 256-way tree
- each parent node can have 1-256 subnodes
- thus one level for each part of A.B.C.D
- put another way: 1.2.3.4 -> 4 levels max

CEF

- note that hw must support – basically we have a marriage of hw + sw
- CEF table mirrors arp/routing tables
  - 256-way tree
  - view this as the routing table
- matched by adjacency table
  - contains info for fast next hop processing
- key idea: CEF tables built as side effect of changes to route tables, not necessarily at 1st pkt in flow time

solves a number of problems

- 1st pkt recv. is no longer problem
- multiple routes to same destination are supported by
  - replacing 1 adjacency with a list of adjacencies
- bottom-line assumption
  - hw support is useful
  - cache lookup/route table lookup merged

routing table semantics

- 1. a routing table is a LOGICAL idea, and implementations will differ
- 2. a routing daemon may keep a shadow routing table:
  - input route information
  - output route table changes to go into rttab
- 3. routing table entries are typed

types acc. to Cisco

- connected routes:
  - directly connected subnet out Ethernet 0 e.g.,
- static routes:
  - human being stuck them in via config
- dynamic routing – interior
  - interior routing protocol like RIP, OSPF, EIGRP
- dynamic routing – exterior
  - BGP routes

and we have weights supplied by admin

- e.g., you have two upstreams ports
  - slow modem
  - fast DSL
- both are DEFAULTS
  - but you choose the fast DSL 100%
  - UNTIL IT GOES DOWN

  - hw/sw recognizes it is down and chooses the less favored modem as the default

this is a redundancy technique