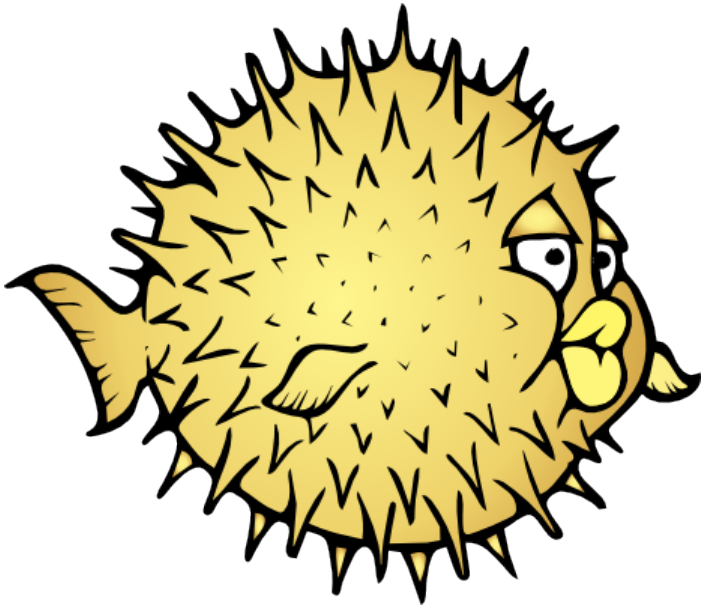
Embracing the BSD Routing Table



Martin Pieuchot
mpi@openbsd.org

EuroBSDcon, Belgrade

September 2016

Embracing the BSD Routing Table

How many *global data structures* do you need?

Agenda

BSD Routing Table

Refined Interface

New data structures

Conclusion

Agenda

BSD Routing Table

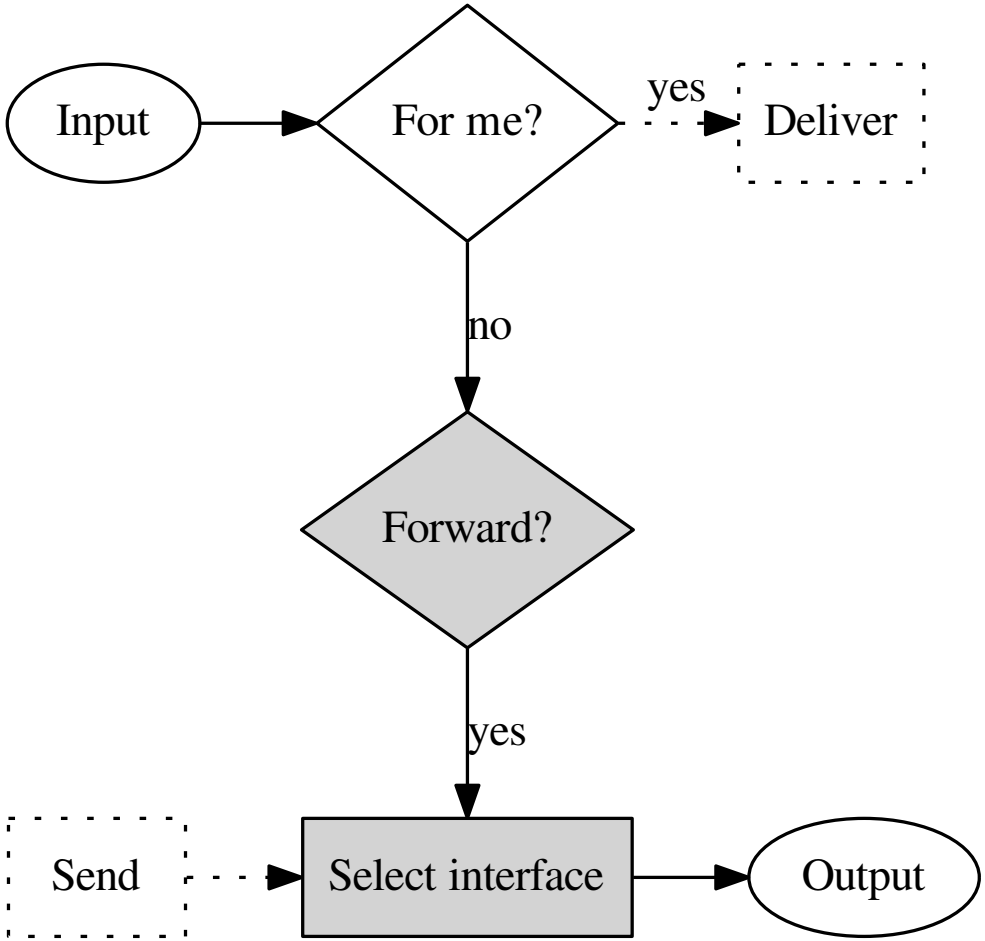
Refined Interface

New data structures

Conclusion

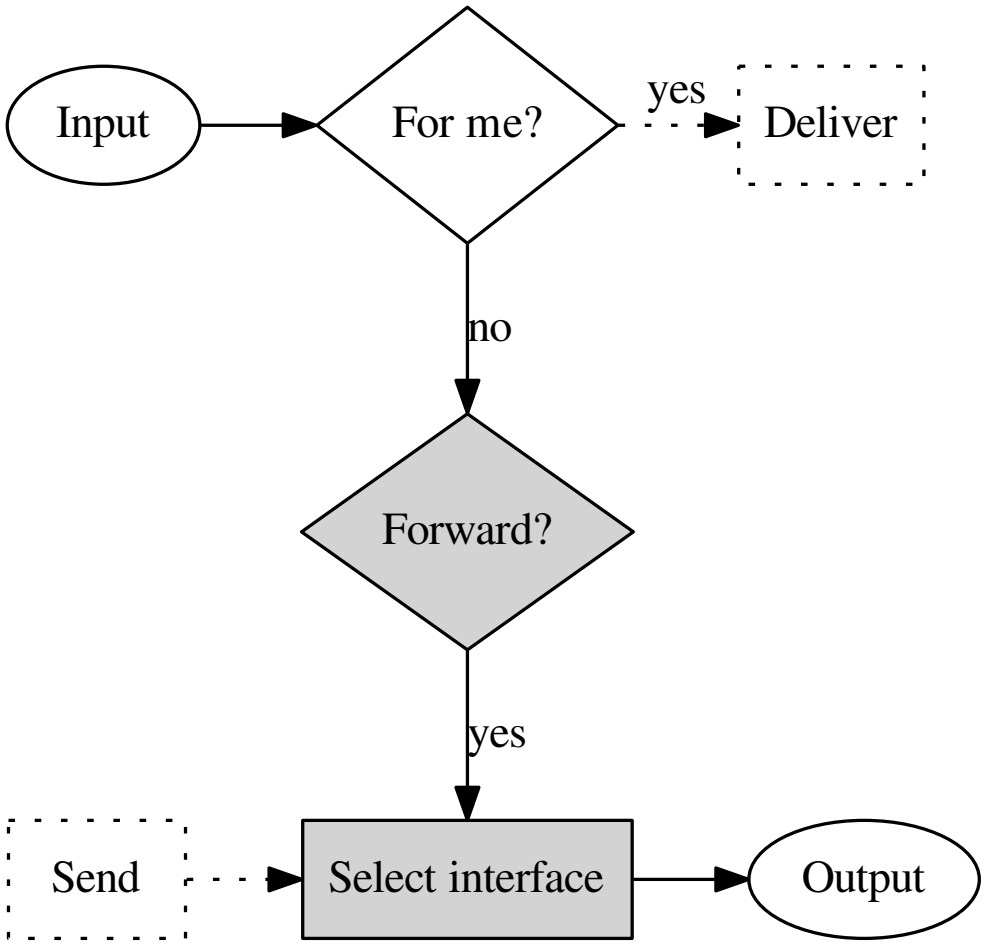
Forwarding table

sys/net/radix.c



Forwarding table

sys/net/radix.c



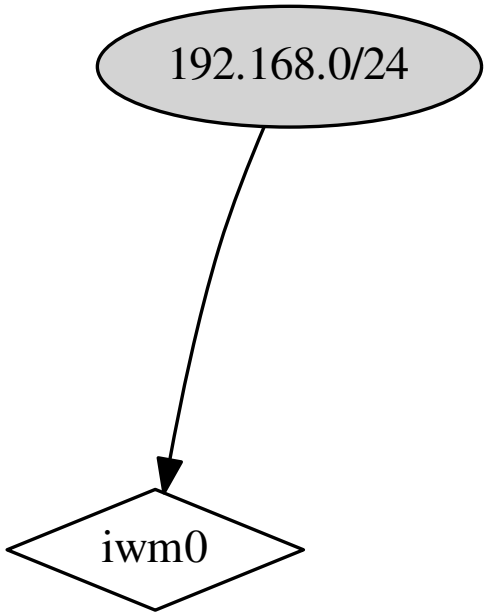
Since 4.3 Reno

- replace hash-based lookup
- PATRICIA trie
  - radix tree with $r = 2$

Link layer address translation

sys/net/if_ethersubr.c

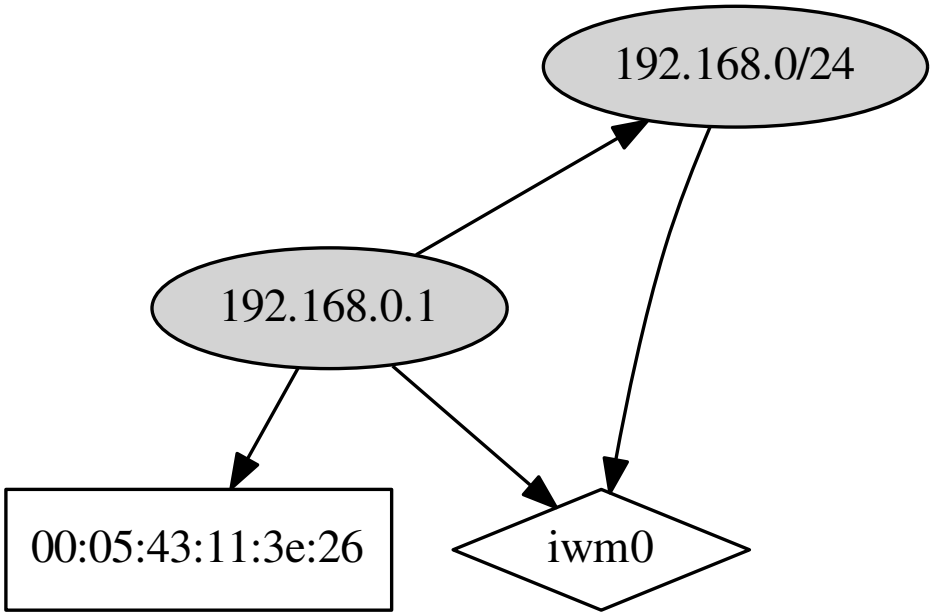
RTF_CLONING: For each connected route



Link layer address translation

sys/net/if_ethersubr.c

RTF_CLONING: For each connected route

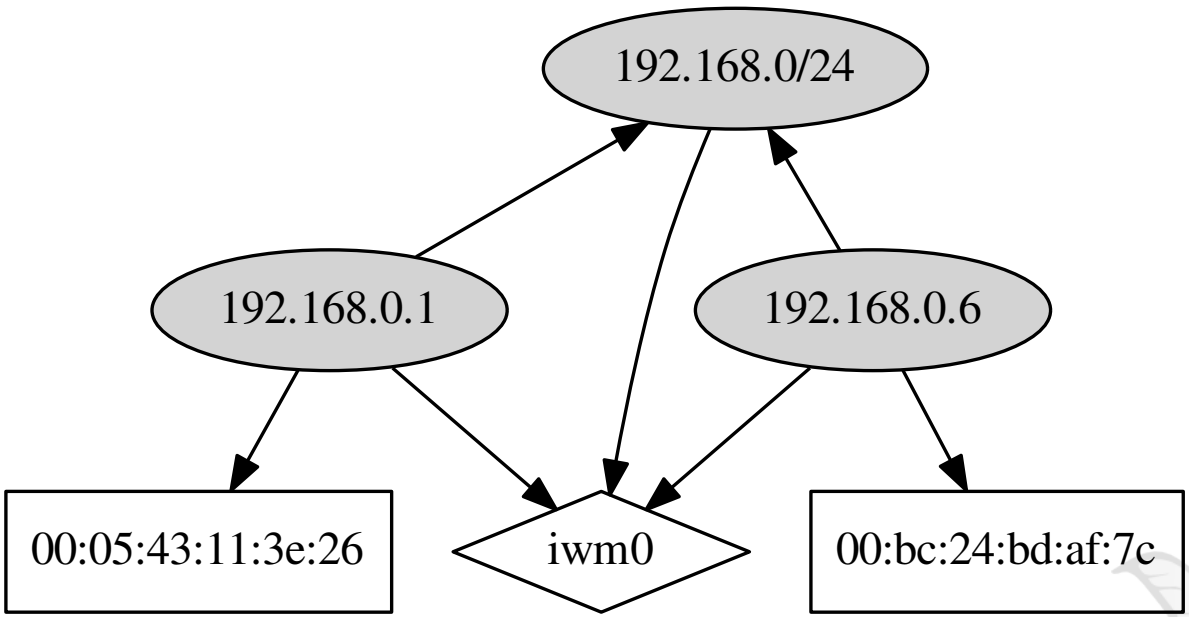


Link layer address translation

sys/net/if_ethersubr.c

RTF_CLONING: For each connected route

RTF_CLONED: For every host in the subnet

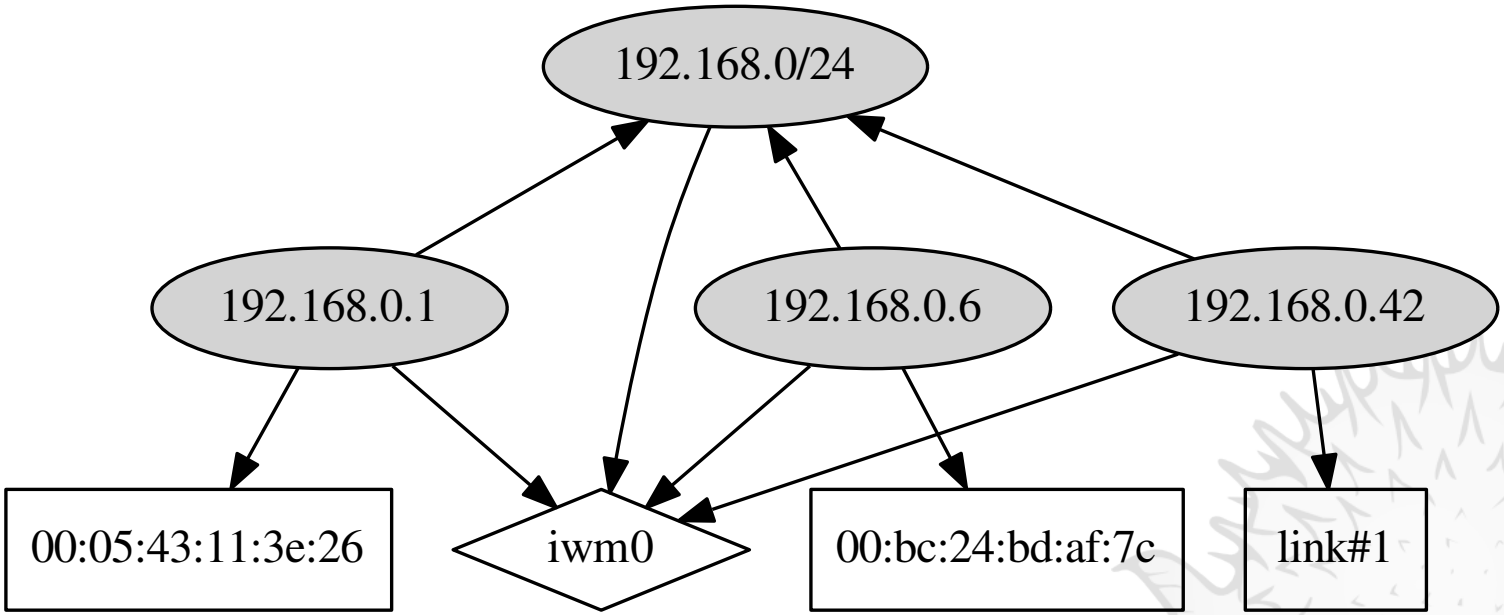


Link layer address translation

sys/net/if_ethersubr.c

RTF_CLONING: For each connected route

RTF_CLONED: For every host in the subnet



Message oriented IPC

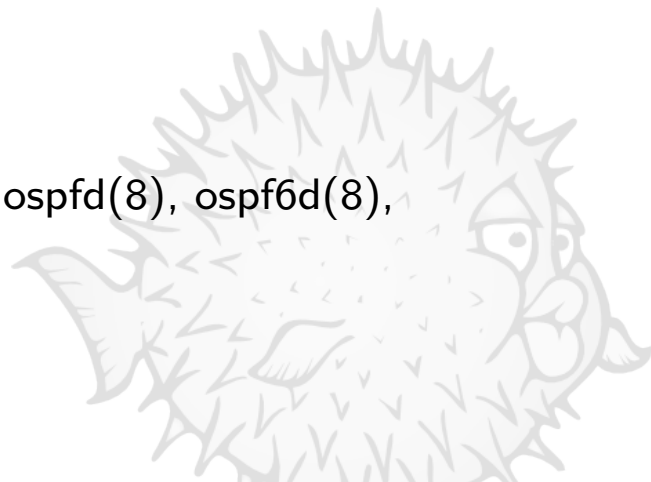
sys/net/rtsock.c

Routing messages

- RTM_ADD
- RTM_DELETE
- RTM_CHANGE
- RTM_GET
- RTM_NEWADDR
- RTM_DELADDR
- RTM_IFINFO
- ...

Native speakers

route(8), dhclient(8), bgpd(8), dvmrpd(8), eigrpd(8), ldpd(8), ospfd(8), ospf6d(8), ripd(8), snmpd(8), ...



Agenda

BSD Routing Table

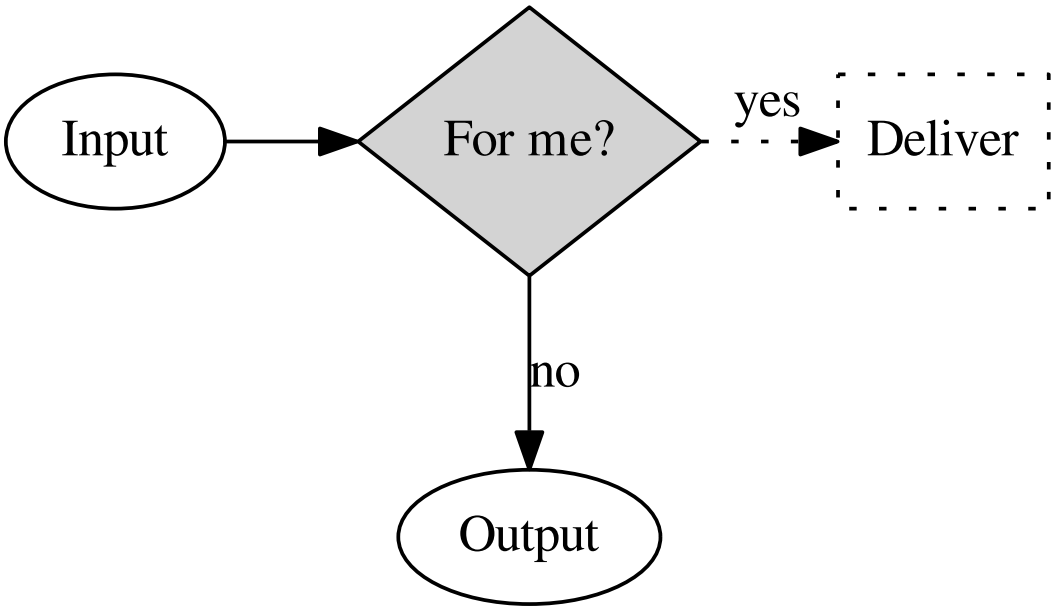
Refined Interface

New data structures

Conclusion

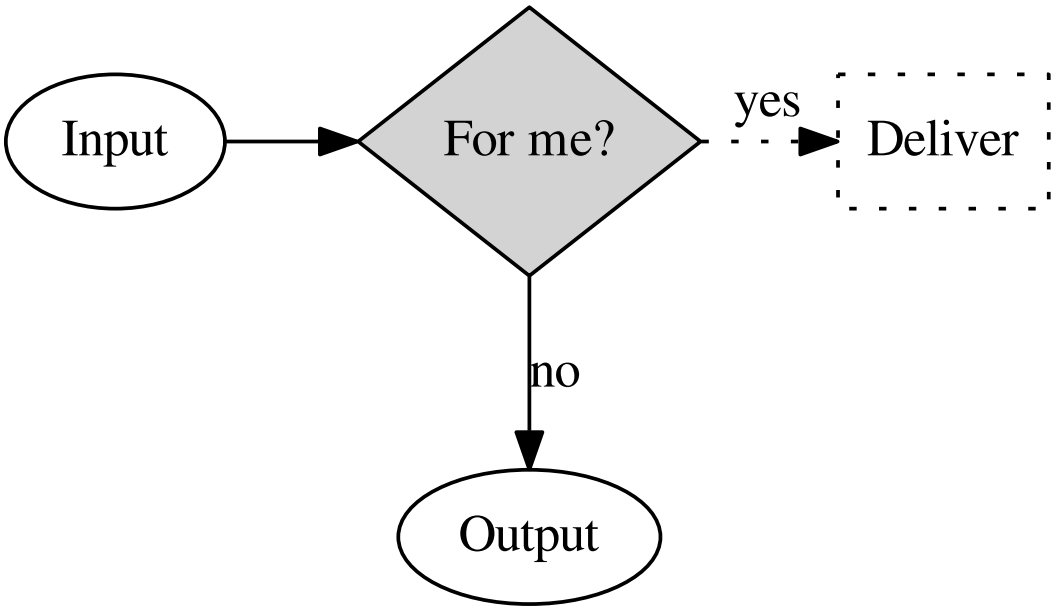
Single lookup

sys/netinet/ip_input.c



Single lookup

sys/netinet/ip_input.c



Forwarding?

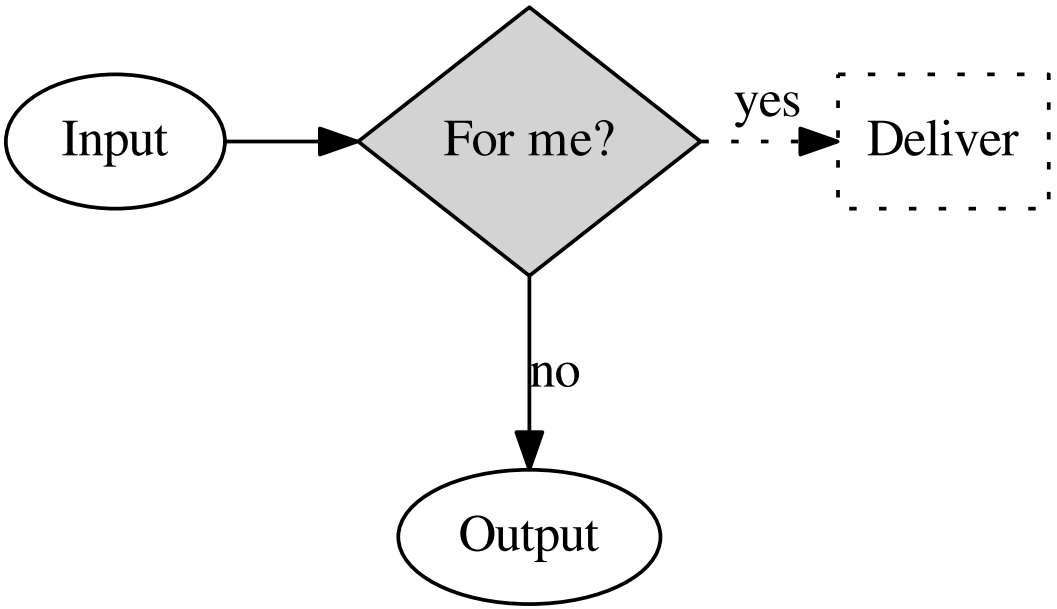
Where?

Which Source?

Link layer address?

Single lookup

sys/netinet/ip_input.c



Forwarding?

- RTF_LOCAL
- RTF_BROADCAST

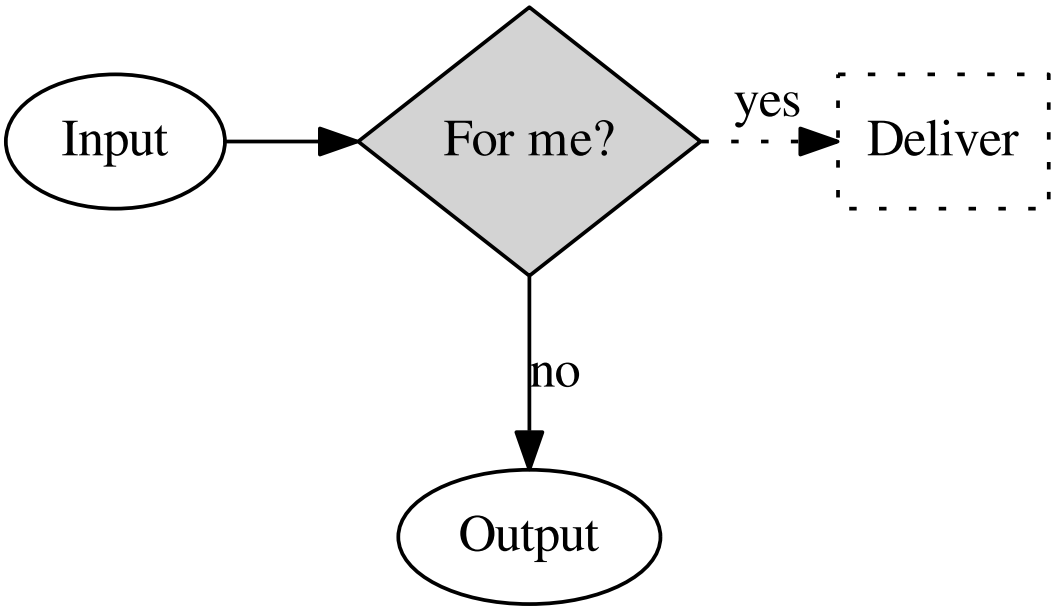
Where?

Which Source?

Link layer address?

Single lookup

sys/netinet/ip_input.c



Forwarding?

- RTF_LOCAL
- RTF_BROADCAST

Where?

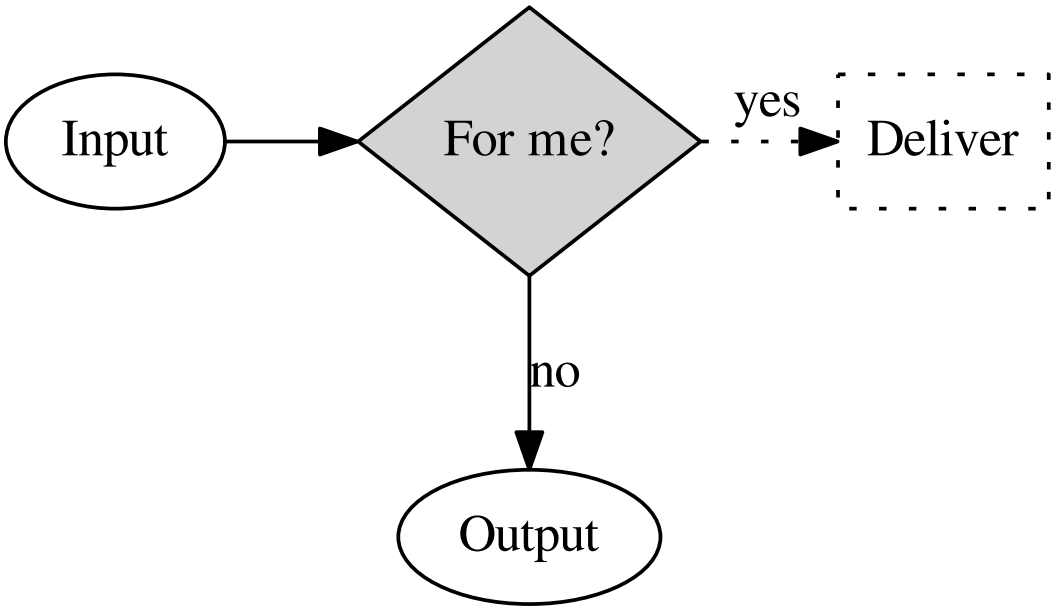
- rt_ifidx

Which Source?

Link layer address?

Single lookup

sys/netinet/ip_input.c



Forwarding?

- RTF_LOCAL
- RTF_BROADCAST

Where?

- rt_ifidx

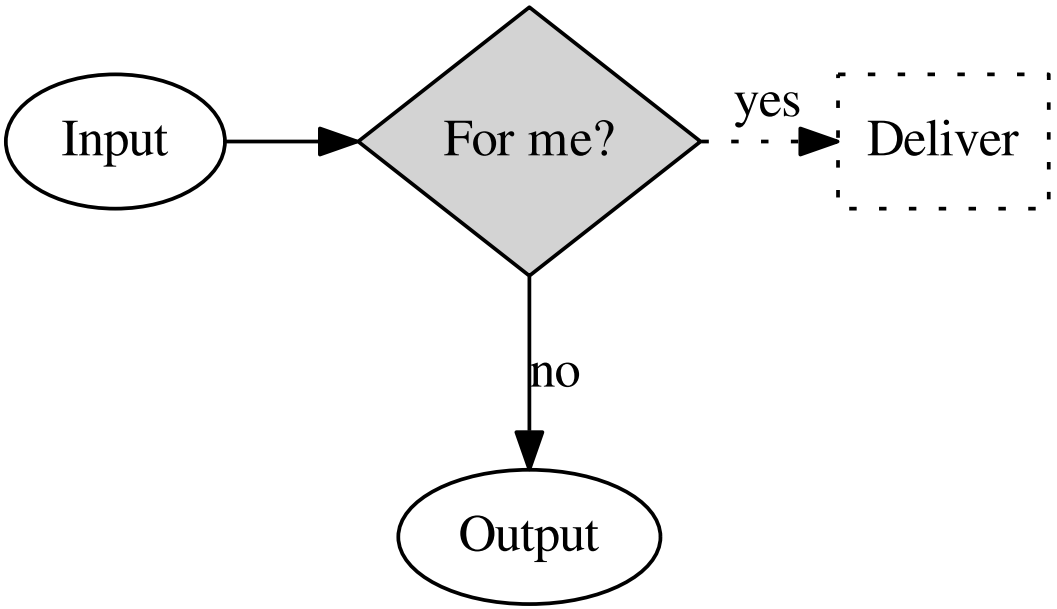
Which Source?

- rt_ifa

Link layer address?

Single lookup

sys/netinet/ip_input.c



Forwarding?

- RTF_LOCAL
- RTF_BROADCAST

Where?

- rt_ifidx

Which Source?

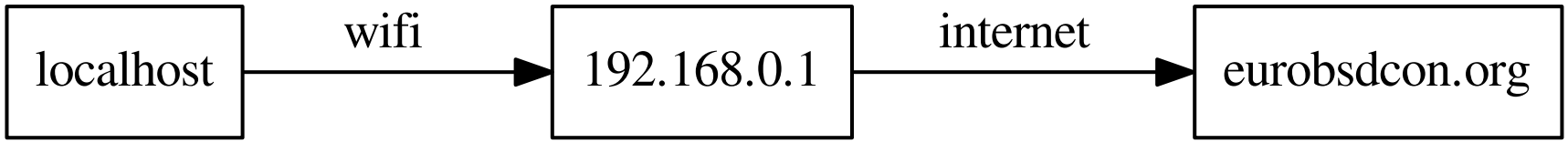
- rt_ifa

Link layer address?

- rt_gateway

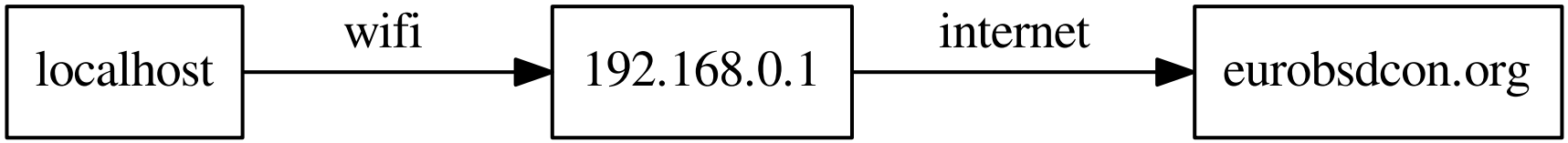
Gateway route

sys/net/route.c



Gateway route

sys/net/route.c

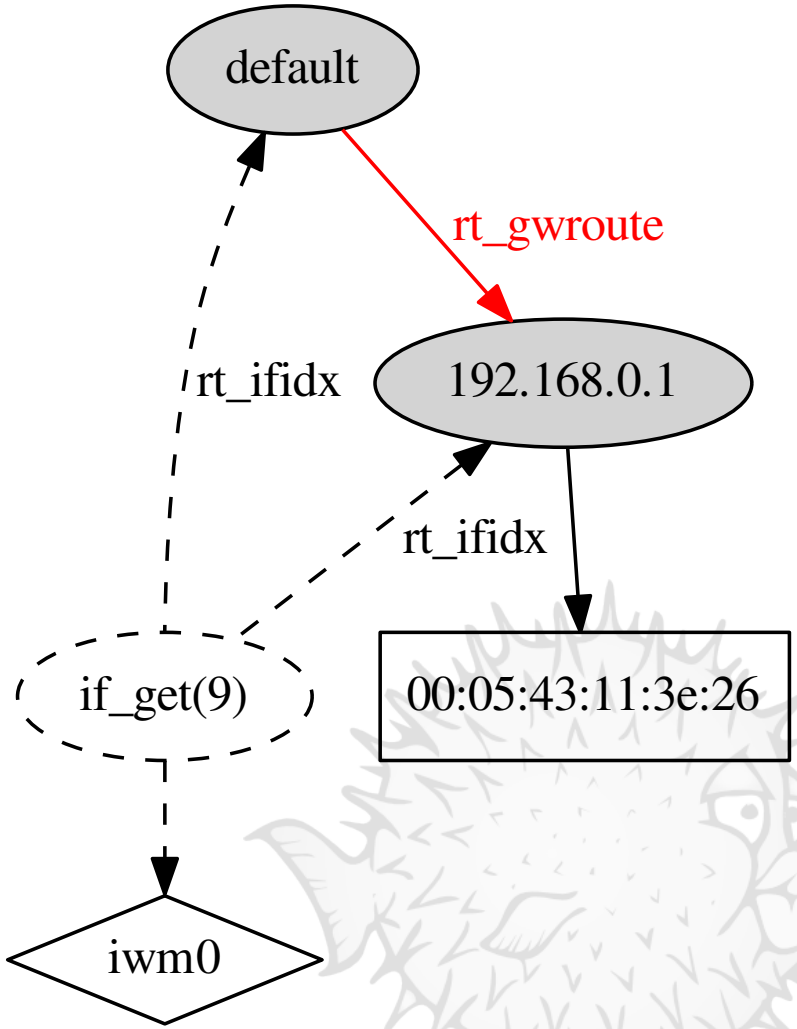



```
$ netstat -rnf inet

Routing tables

Internet:
Destination        Gateway            Flags    Refs      Use   Mtu  Prio Iface
default            192.168.0.1        UGS        20      420     -     8 iwm0
192.168.0/24       192.168.0.6        UC          2       10     -     4 iwm0
192.168.0.1        00:05:43:11:3e:26  UHLch       1      241     -     4 iwm0
192.168.0.6        00:bc:24:bd:af:7c  UHLl        1        4     -     4 iwm0
```

Link layer address of the gateway

sys/net/if_ethersubr.c

Single shared cache

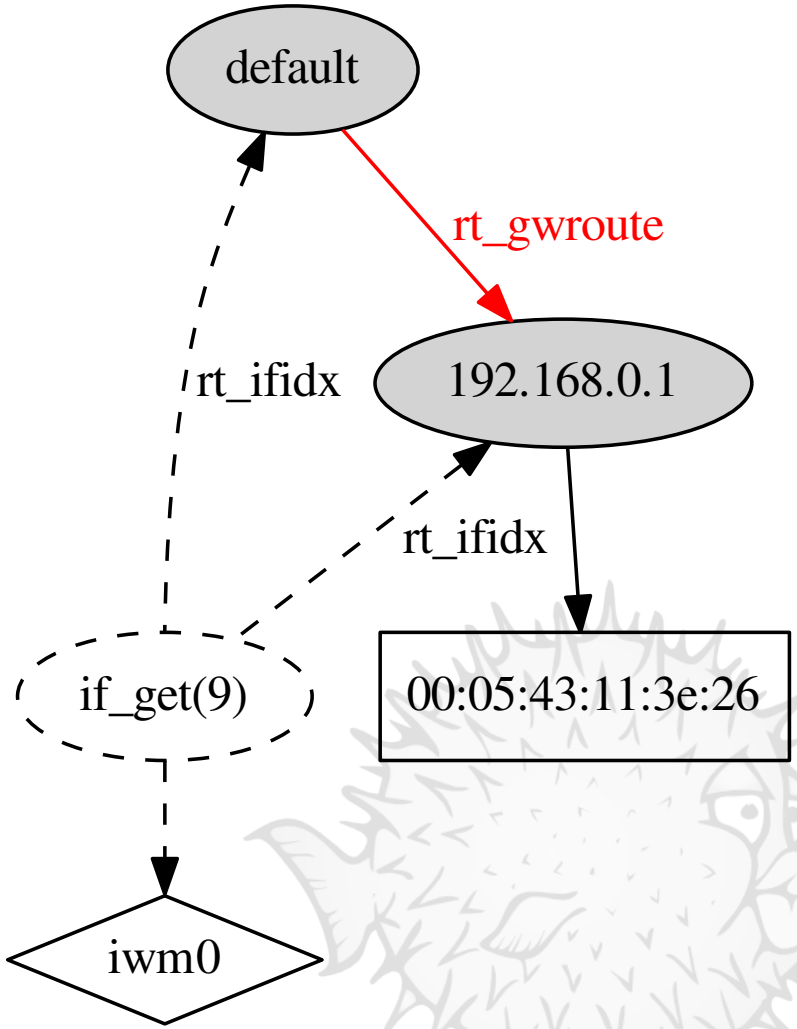


Link layer address of the gateway

sys/net/if_ethersubr.c

Single shared cache

- Proxy reference count

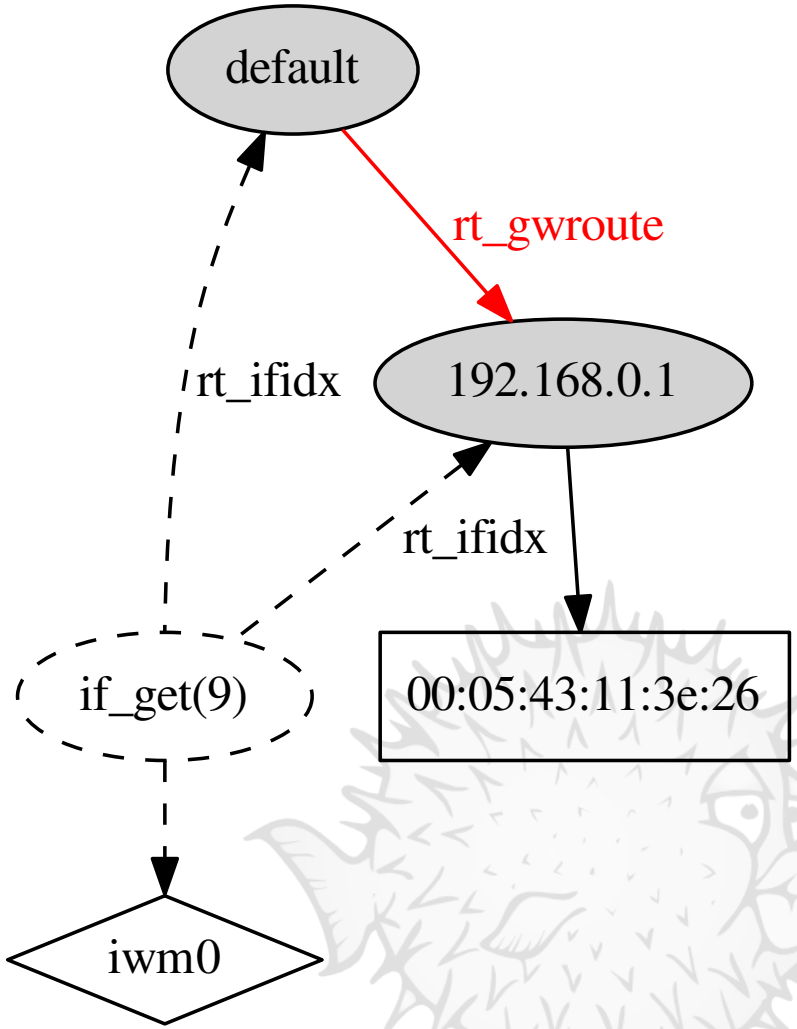


Link layer address of the gateway

sys/net/if_ethersubr.c

Single shared cache

- Proxy reference count
- Immutable pointer

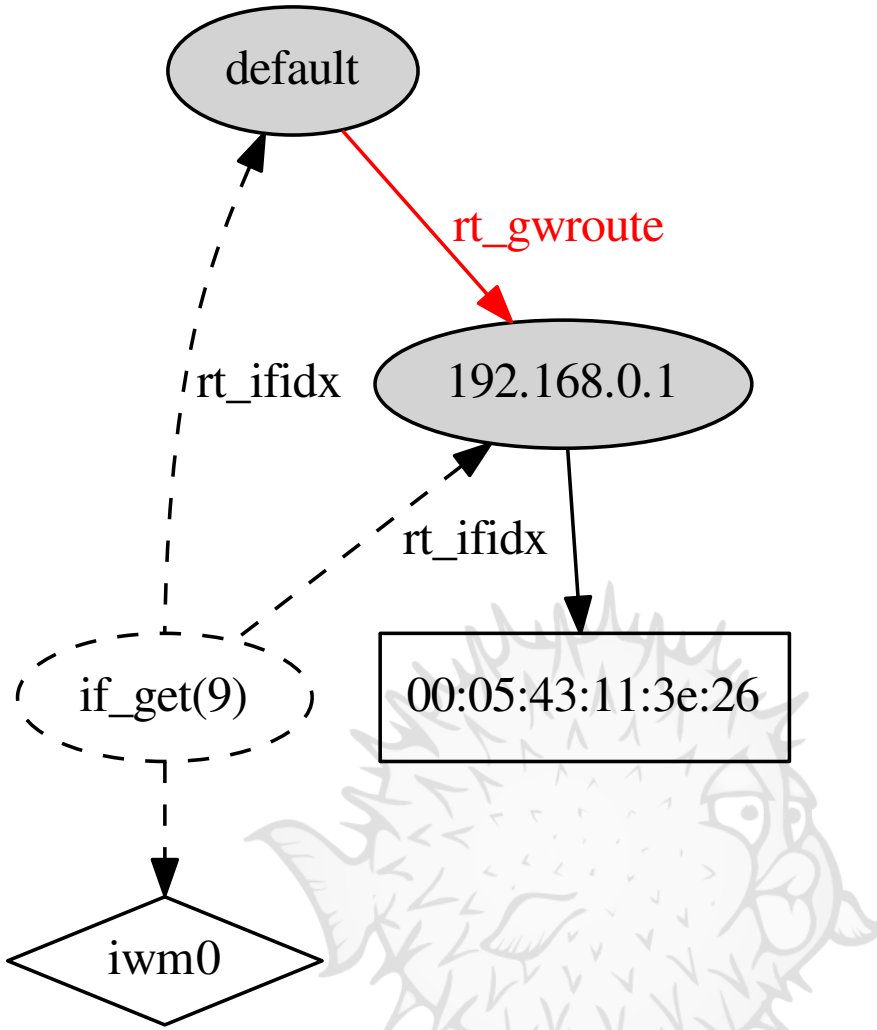


Link layer address of the gateway

sys/net/if_ethersubr.c

Single shared cache

- Proxy reference count
- Immutable pointer
- Flag it RTF_CACHED

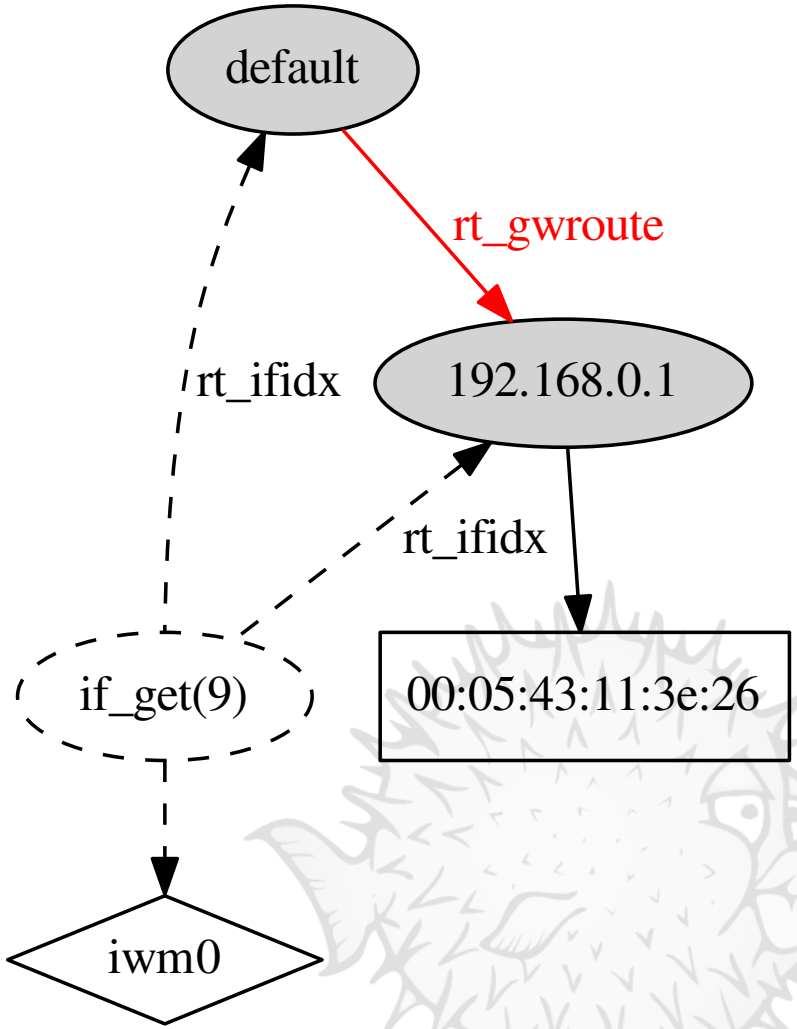


Link layer address of the gateway

sys/net/if_ethersubr.c

Single shared cache

- Proxy reference count
- Immutable pointer
- Flag it RTF_CACHED
- Checks during insertion
  - No second route lookup

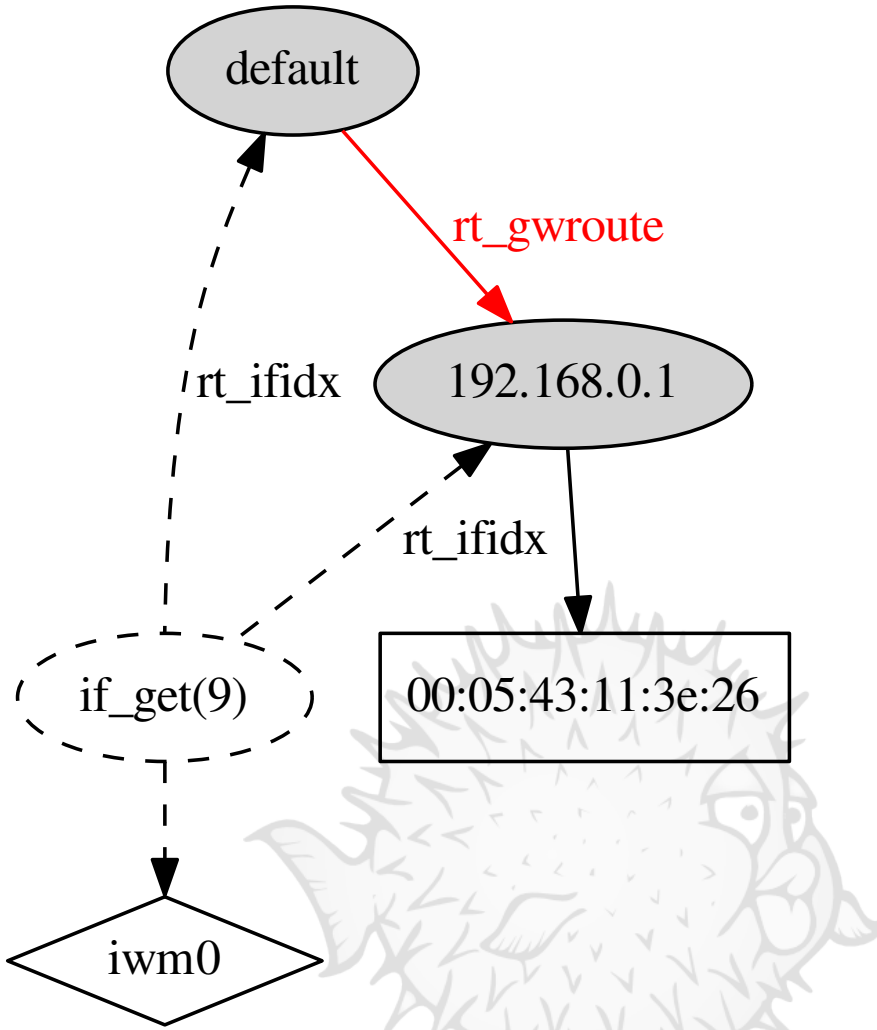


Link layer address of the gateway

sys/net/if_ethersubr.c

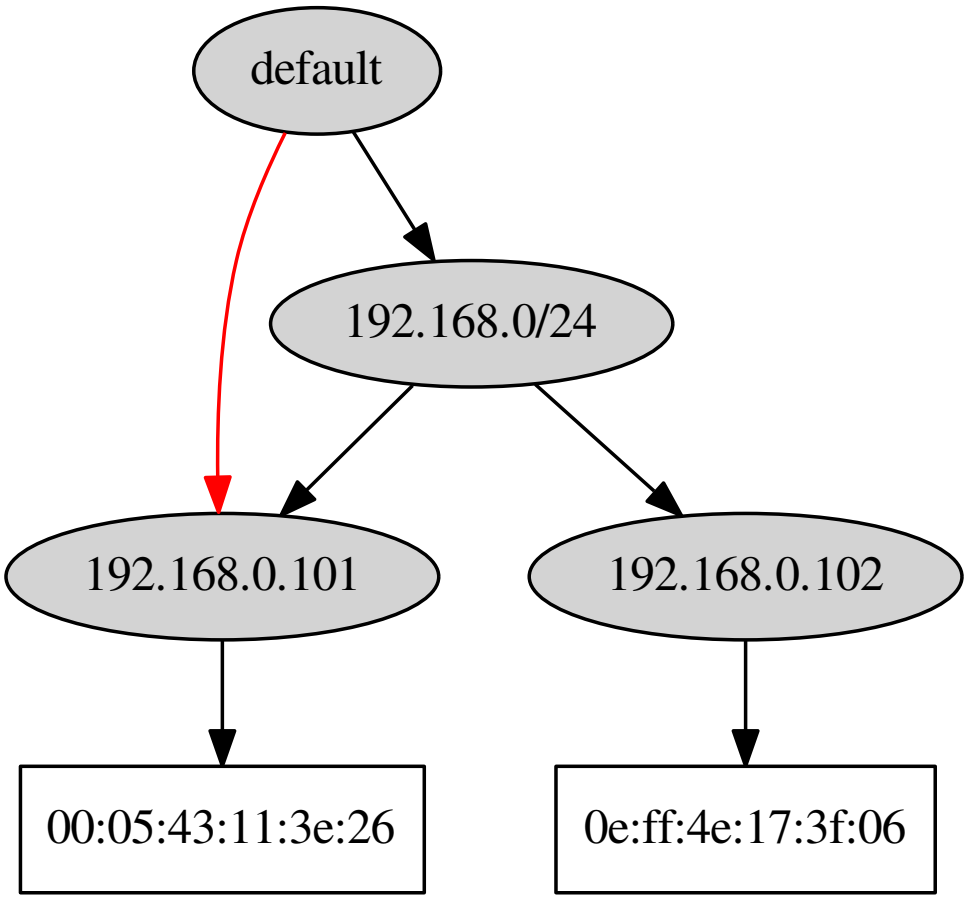
Single shared cache

- Proxy reference count
- Immutable pointer
- Flag it RTF_CACHED
- Checks during insertion
  - No second route lookup
- No atomic operations



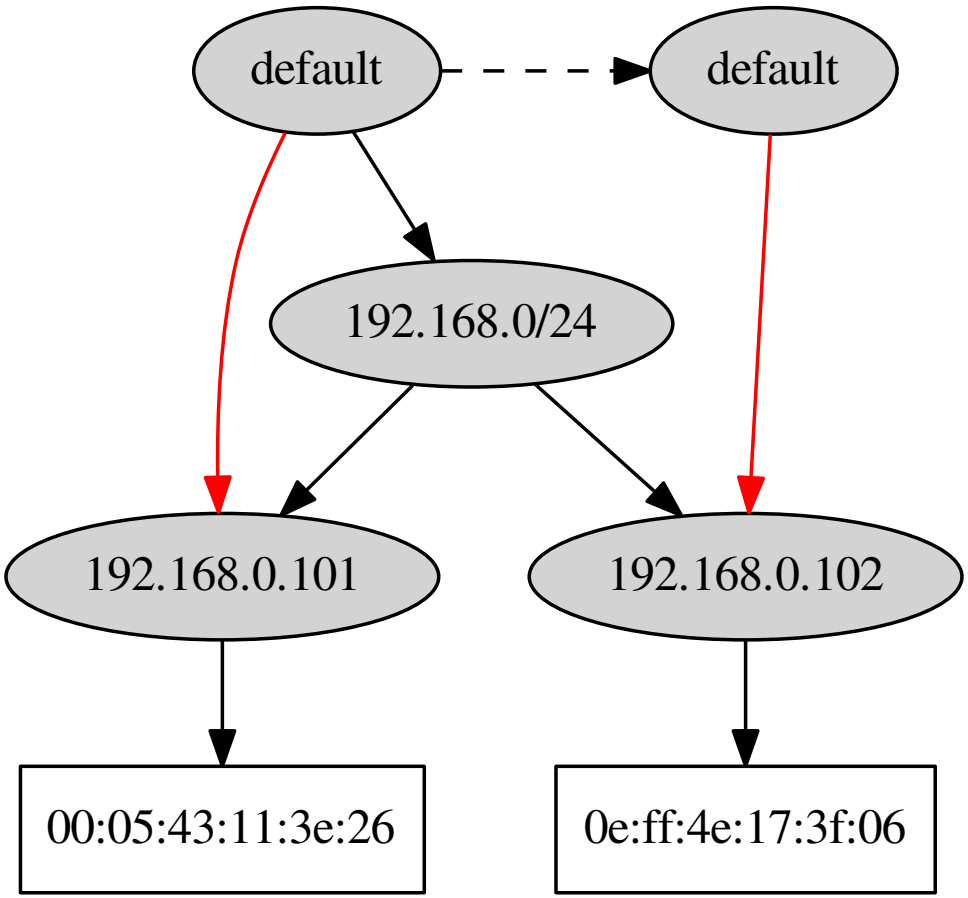
Multipath

sys/net/radix_mpath.c



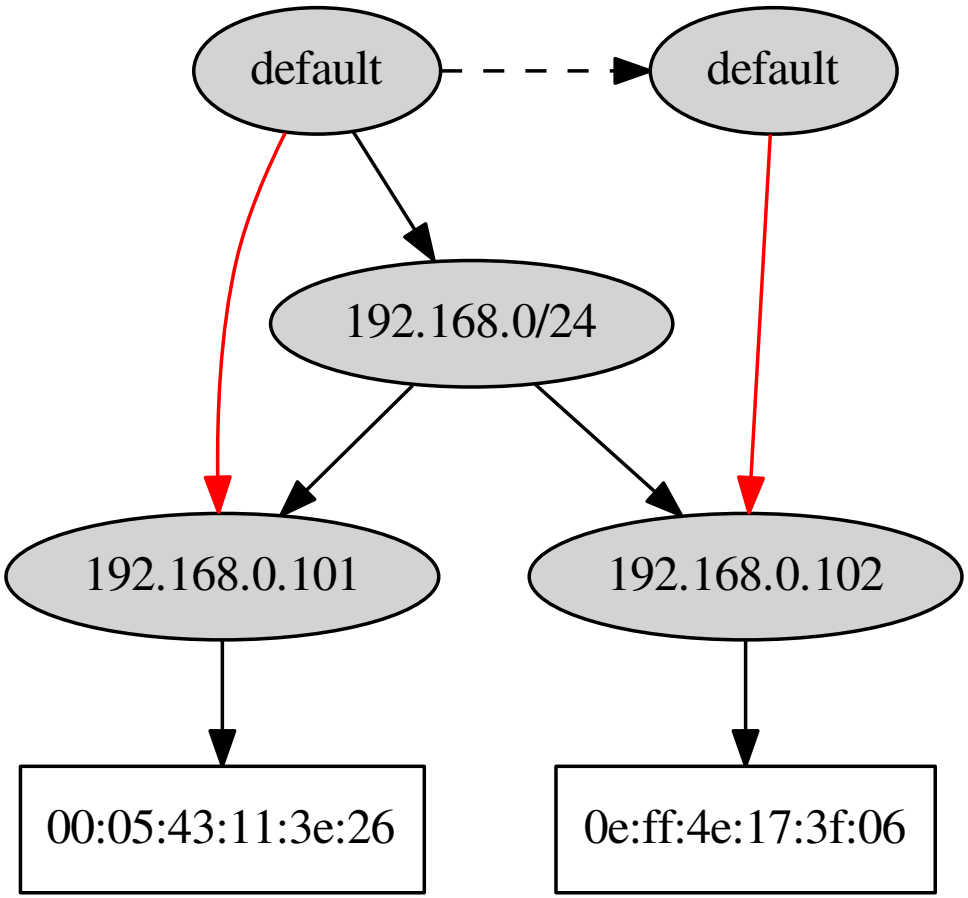
Multipath

sys/net/radix_mpath.c



Multipath

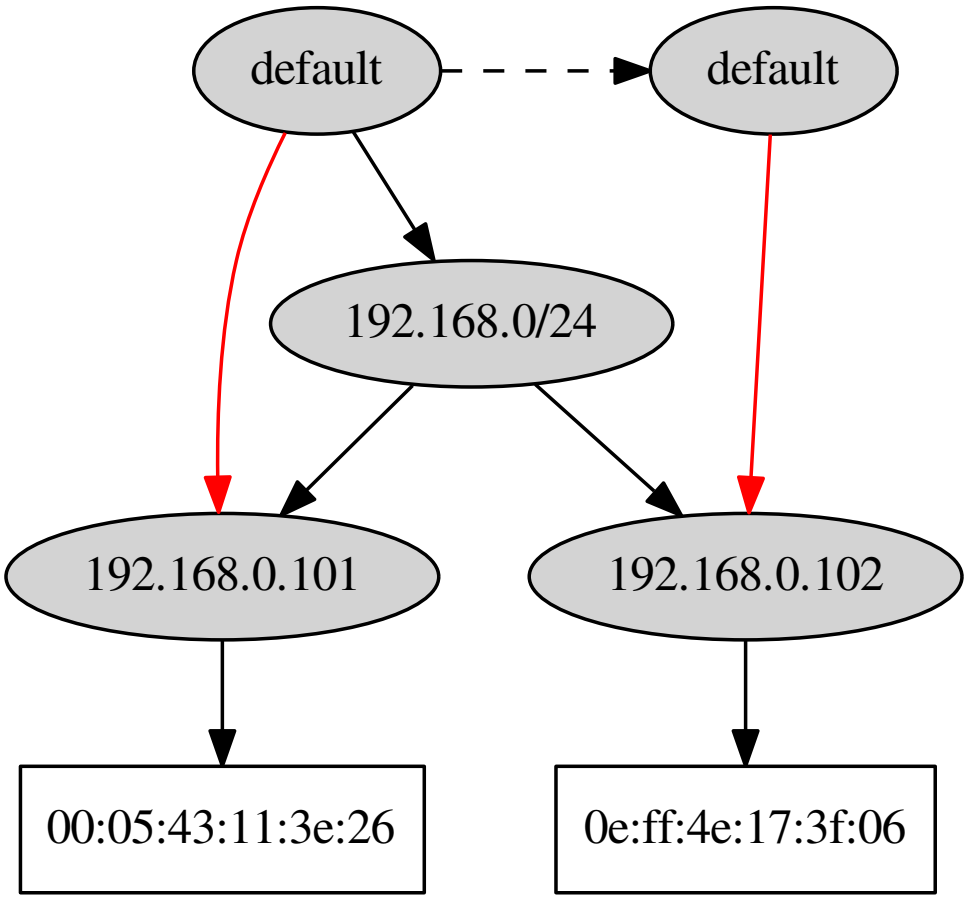
sys/net/radix_mpath.c



- Introduced by KAME
  - for sending/forwarding

Multipath

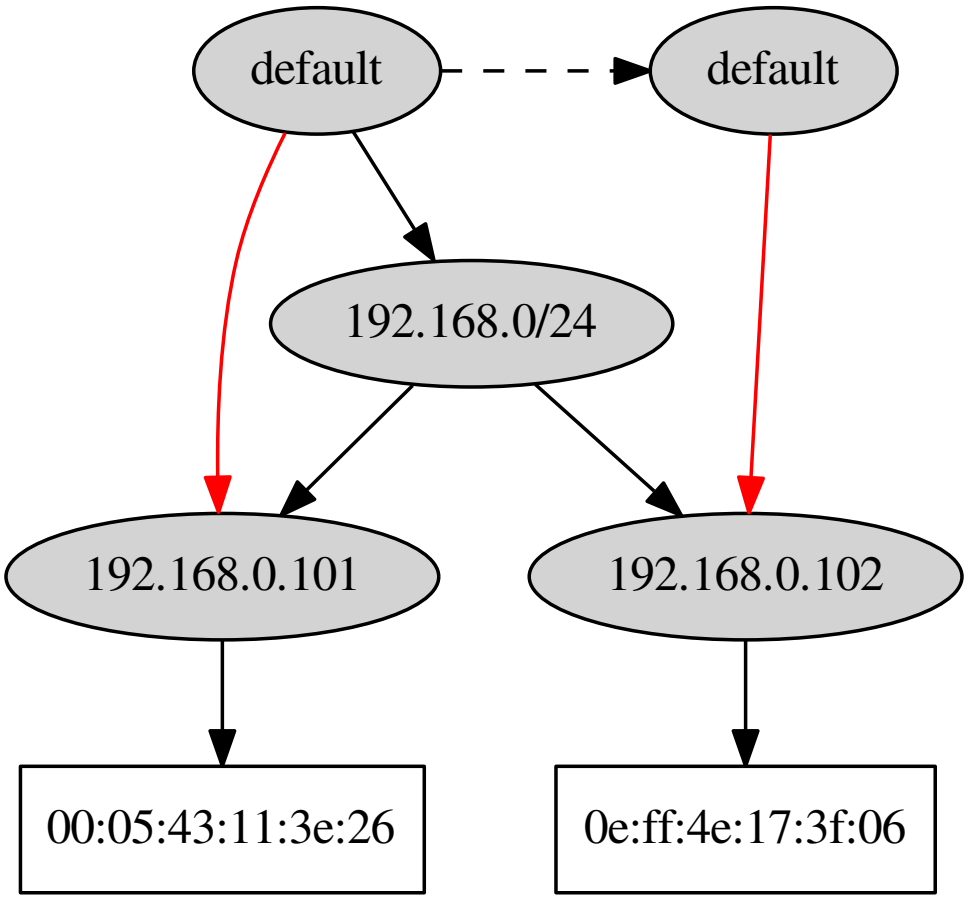
sys/net/radix_mpath.c



- Introduced by KAME
  - for sending/forwarding
- Identical *keys* in the tree
  - different priority, or
  - different gateway

Multipath

sys/net/radix_mpath.c



- Introduced by KAME
  - for sending/forwarding
- Identical *keys* in the tree
  - different priority, or
  - different gateway
- Extended to
  - Connected routes
  - ARP proxy entries
  - (Multicast groups)

Agenda

BSD Routing Table

Refined Interface

New data structures

Conclusion

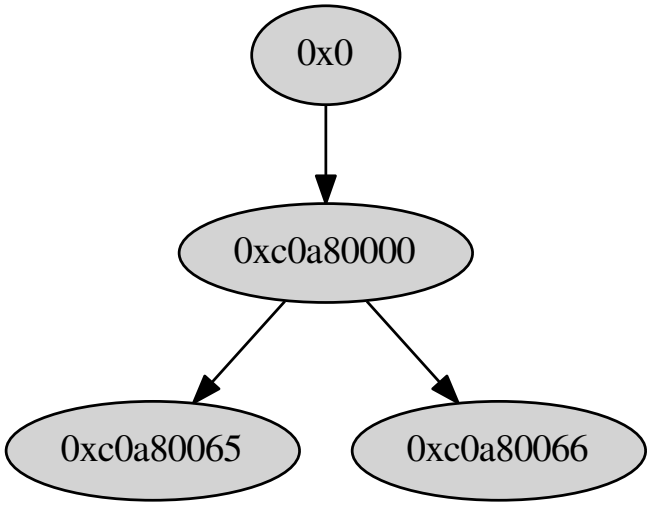
Why?

sys/net/radix_mpath.c

```
/*
 * Stolen from radix.c rn_addroute().
 * This is nasty code with a certain amount of magic and dragons.
 [...]
 */
```

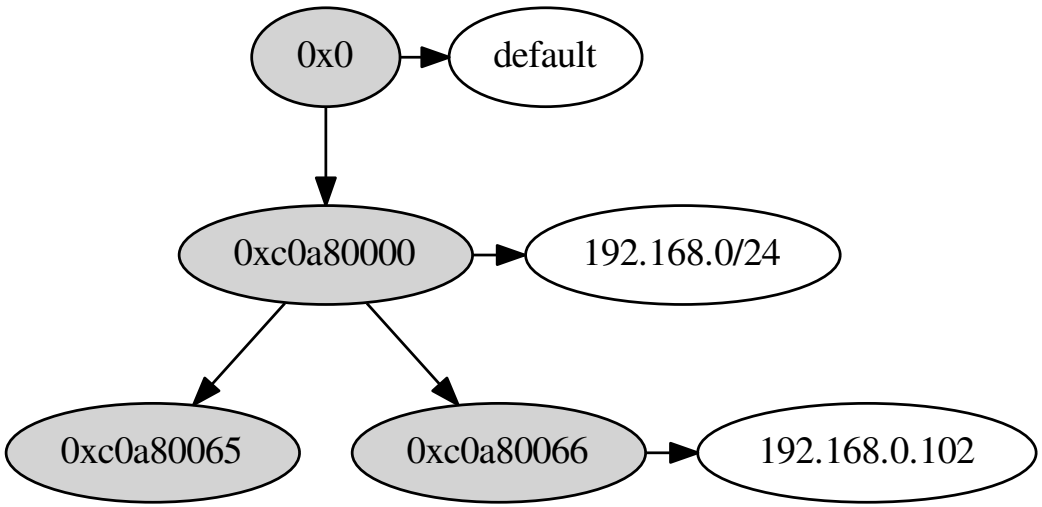
Everything is multipath

sys/net/rtable.c



Everything is multipath

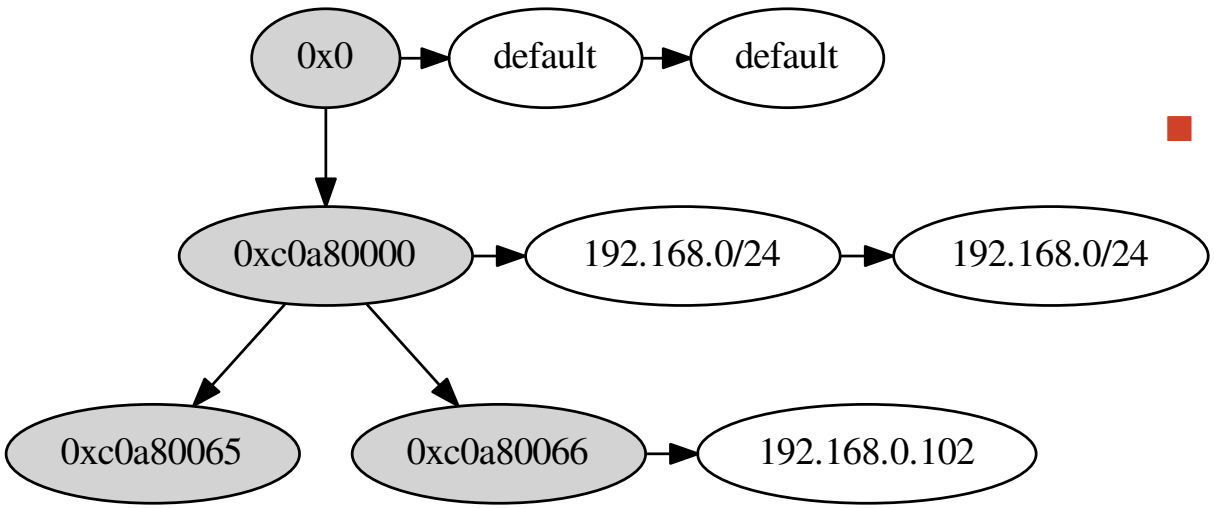
sys/net/rtable.c



- Data structure separation
  - network agnostic
  - *value* is a pointer

Everything is multipath

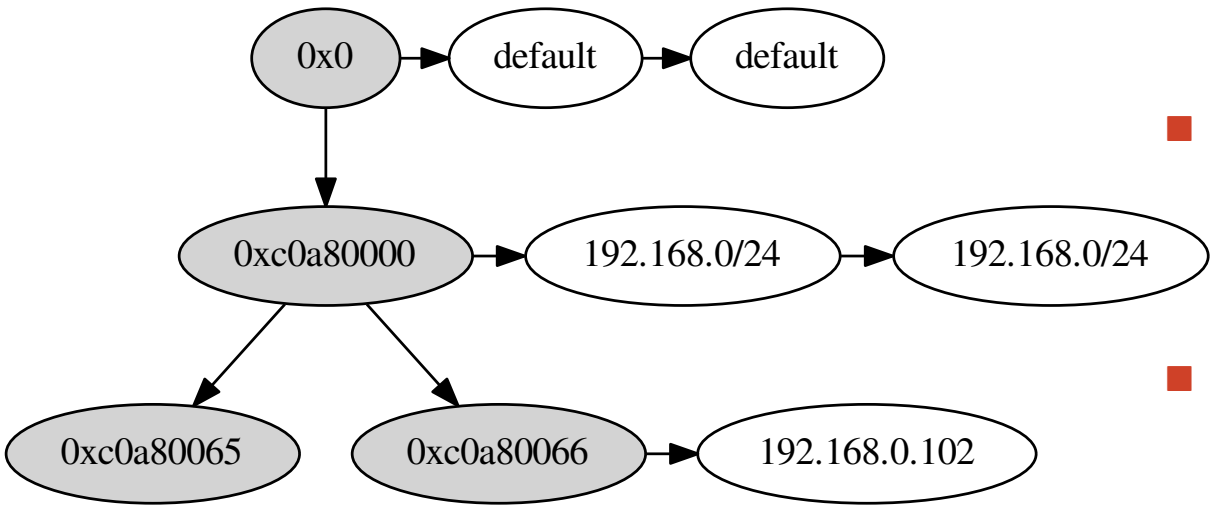
sys/net/rtable.c



- Data structure separation
  - network agnostic
  - *value* is a pointer
- List of entries
  - *value* points to a list
  - ordered by priority
  - generic multipath

Everything is multipath

sys/net/rtable.c



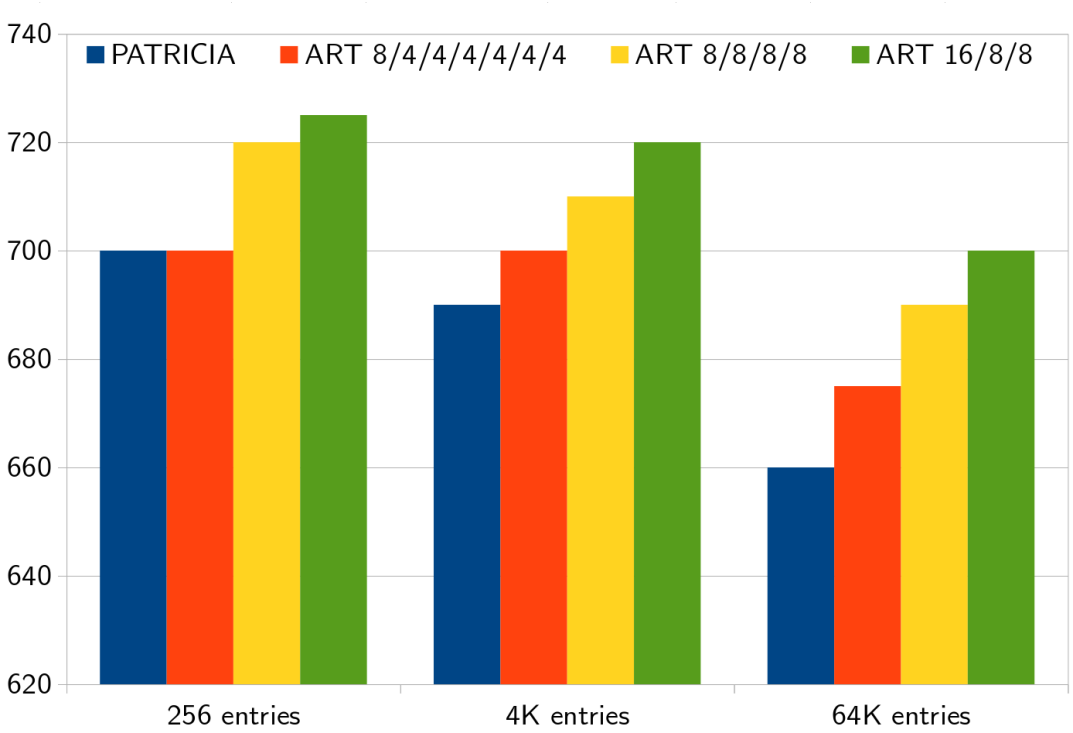
- Data structure separation
  - network agnostic
  - *value* is a pointer
- List of entries
  - *value* points to a list
  - ordered by priority
  - generic multipath
- MP ready
  - different lifetimes
  - separated refcount
  - no backpointer

Allotment Routing Table

sys/net/art.c

Number of packets received

while sending 800Kpps

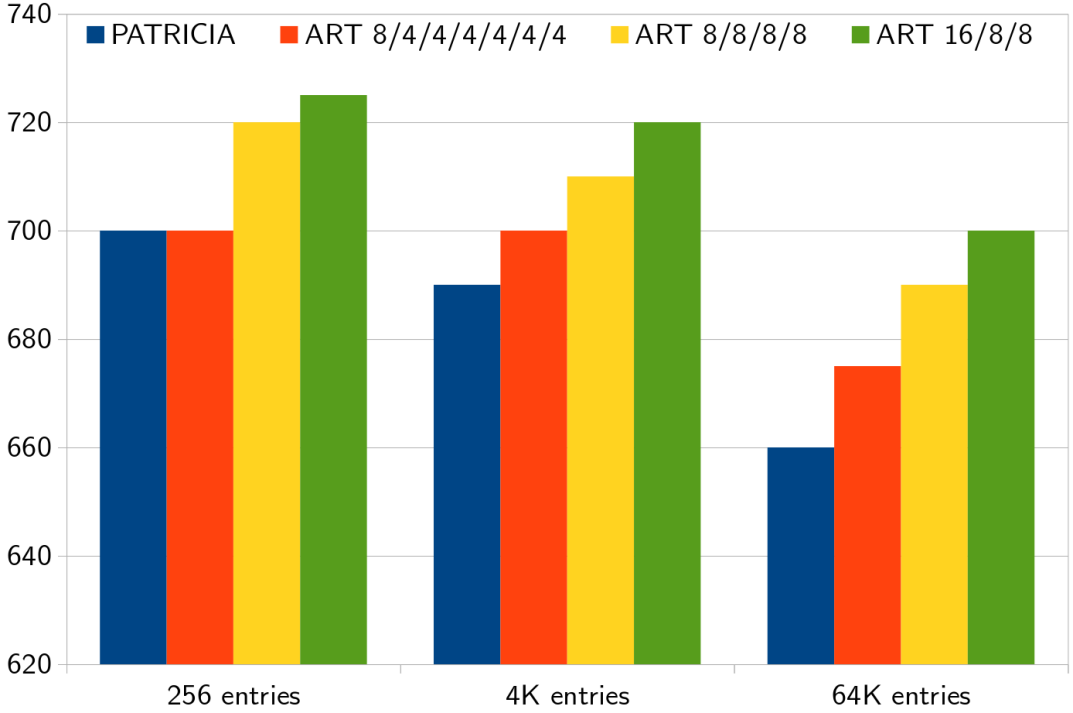


Allotment Routing Table

sys/net/art.c

Number of packets received

while sending 800Kpps



Shared code & knowledge

Beautiful free software story

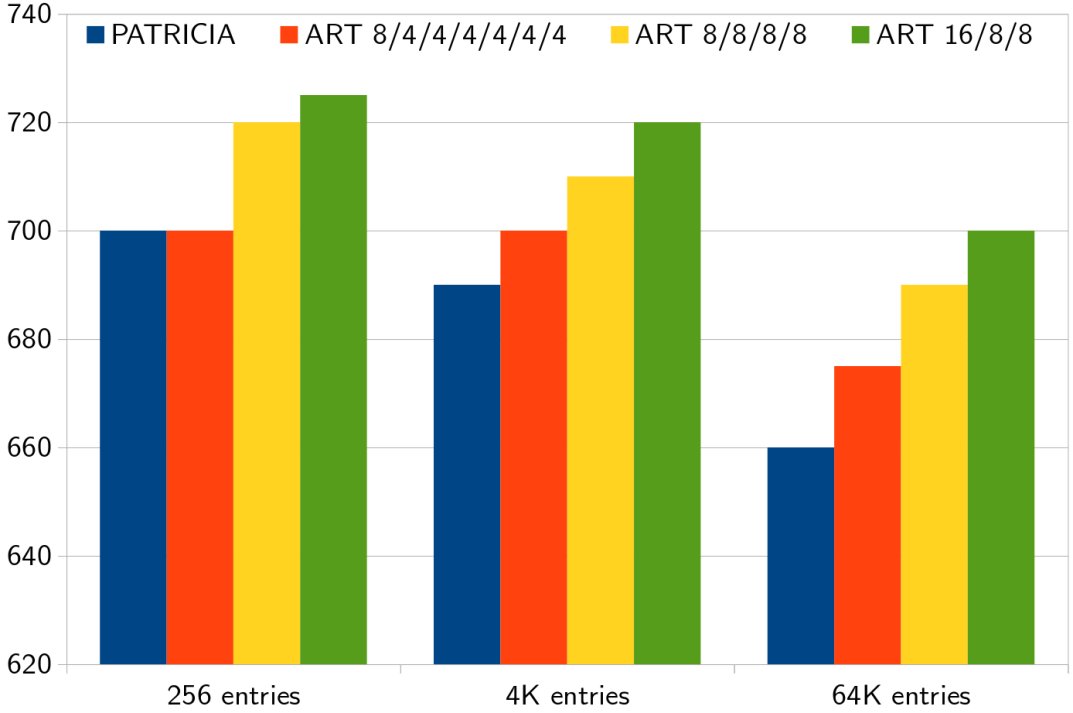
- Algorithm from **Donald Knuth**
  - patent free

Allotment Routing Table

sys/net/art.c

Number of packets received

while sending 800Kpps



Shared code & knowledge

Beautiful free software story

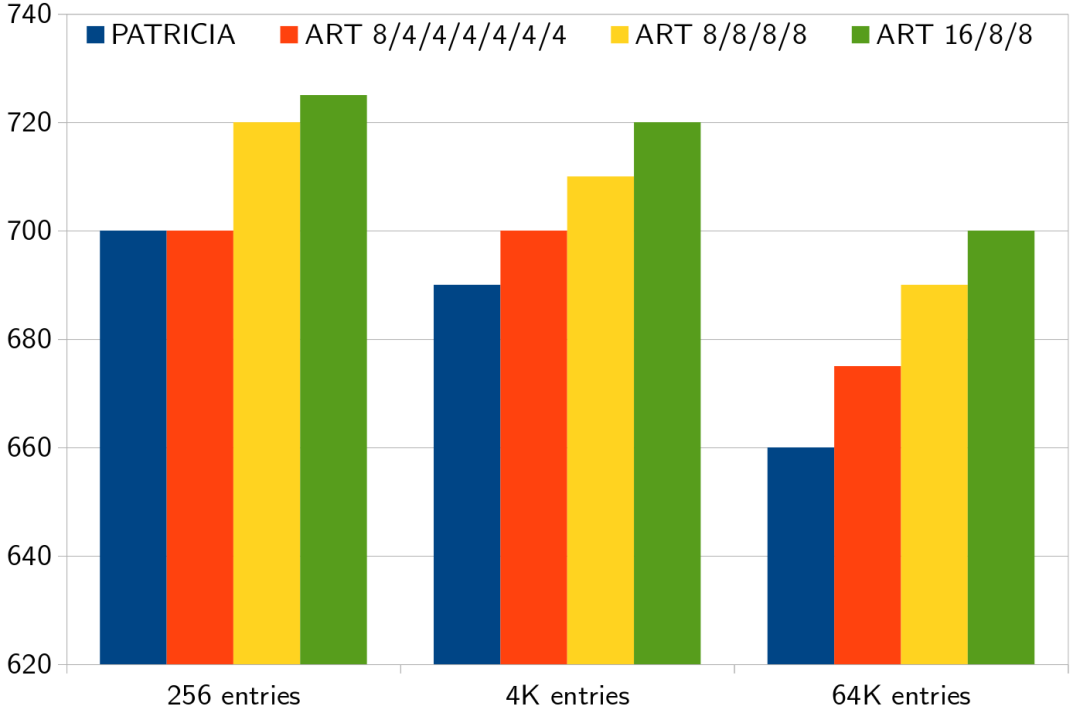
- Algorithm from **Donald Knuth**
  - patent free
- C version by **Yoichi Hariguchi**
  - documented in a paper
  - variable stride length
  - BSD licensed

Allotment Routing Table

sys/net/art.c

Number of packets received

while sending 800Kpps



Shared code & knowledge

Beautiful free software story

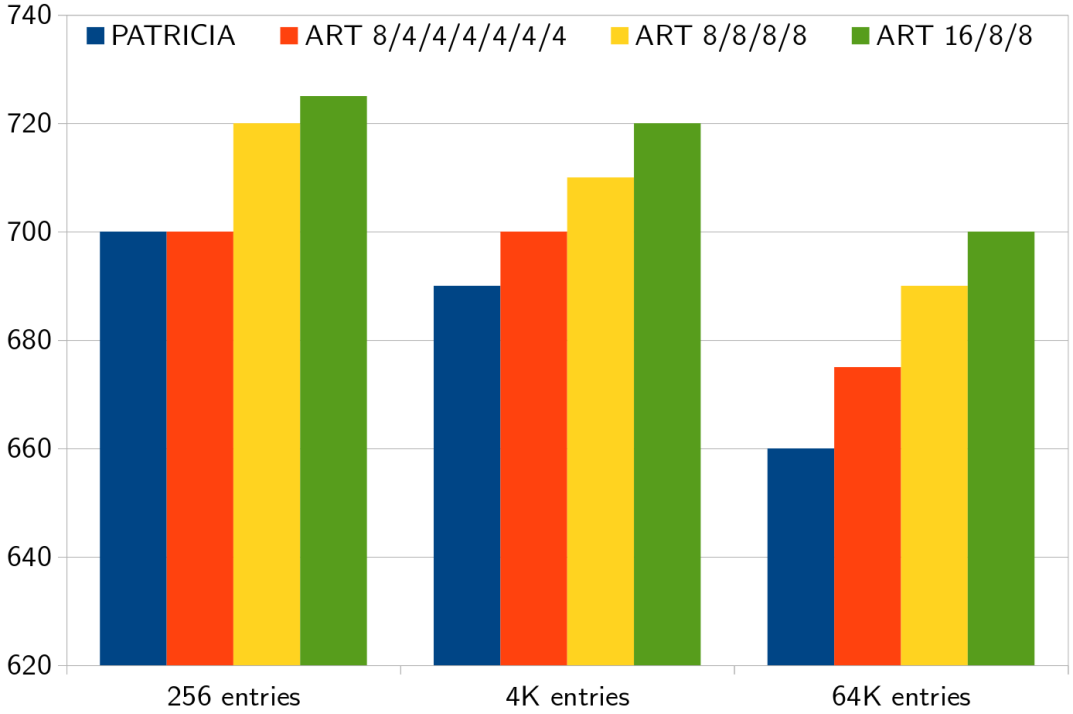
- Algorithm from **Donald Knuth**
  - patent free
- C version by **Yoichi Hariguchi**
  - documented in a paper
  - variable stride length
  - BSD licensed
- Integrated by **Martin Pieuchot**

Allotment Routing Table

sys/net/art.c

Number of packets received

while sending 800Kpps



Shared code & knowledge

Beautiful free software story

- Algorithm from **Donald Knuth**
  - patent free
- C version by **Yoichi Hariguchi**
  - documented in a paper
  - variable stride length
  - BSD licensed
- Integrated by **Martin Pieuchot**
- Lock free lookup by **Jonathan Matthew** & **David Gwynne**

Agenda

BSD Routing Table

Refined Interface

New data structures

Conclusion

Conclusion

sys/net/rtable.c

- Routing table as **single** *gobal data structure*

Conclusion

sys/net/rtable.c

- Routing table as **single** *gobal data structure*
  - Used for *forwarding*, *sending* and *receiving*

Conclusion

sys/net/rtable.c

- Routing table as **single** *gobal data structure*
  - Used for *forwarding*, *sending* and *receiving*
  - Consulted **once** per packet

Conclusion

sys/net/rtable.c

- Routing table as **single** *gobal data structure*
  - Used for *forwarding*, *sending* and *receiving*
  - Consulted **once** per packet
  - **Lock free** lookup

Conclusion

sys/net/rtable.c

- Routing table as **single** *gobal data structure*
  - Used for *forwarding*, *sending* and *receiving*
  - Consulted **once** per packet
  - **Lock free** lookup
- No secondary lookup for *link layer address* translation

Conclusion

sys/net/rtable.c

- Routing table as **single** *gobal data structure*
  - Used for *forwarding*, *sending* and *receiving*
  - Consulted **once** per packet
  - **Lock free** lookup
- No secondary lookup for *link layer address* translation
- No atomic primitive to get the *gateway* link layer address

Conclusion

sys/net/rtable.c

- Routing table as **single** *gobal data structure*
  - Used for *forwarding*, *sending* and *receiving*
  - Consulted **once** per packet
  - **Lock free** lookup
- No secondary lookup for *link layer address* translation
- No atomic primitive to get the *gateway* link layer address
- Generic, multi-use *multipath* implementation

Conclusion

sys/net/rtable.c

- Routing table as **single** *gobal data structure*
  - Used for *forwarding*, *sending* and *receiving*
  - Consulted **once** per packet
  - **Lock free** lookup
- No secondary lookup for *link layer address* translation
- No atomic primitive to get the *gateway* link layer address
- Generic, multi-use *multipath* implementation
- **Faster** route lookup via *ART*

Conclusion

sys/net/rtable.c

- Routing table as **single** *gobal data structure*
  - Used for *forwarding*, *sending* and *receiving*
  - Consulted **once** per packet
  - **Lock free** lookup
- No secondary lookup for *link layer address* translation
- No atomic primitive to get the *gateway* link layer address
- Generic, multi-use *multipath* implementation
- **Faster** route lookup via *ART*
- Interface didn't change

Questions?

Slides on http://www.openbsd.org/papers/

More stories on http://www.grenadille.net

Coming soon!

sys/net/pf.c