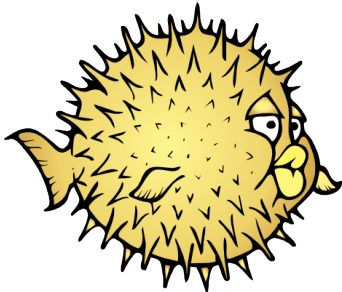


Taming OpenBSD Network Stack Dragons



Martin Pieuchot
mpi@openbsd.org

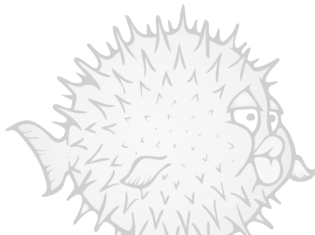
EuroBSDcon, Sofia

September 2014

Taming OpenBSD Network Stack Dragons

sys/net/radix_mpath.c

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



Agenda

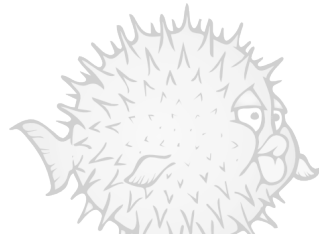
Motivation

Representing Addresses & Routes

Stack Metamorphosis

Where are we now?

Conclusion



Agenda

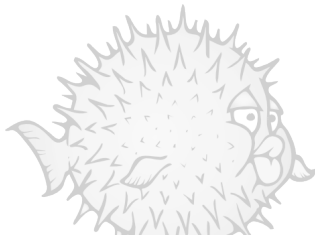
Motivation

Representing Addresses & Routes

Stack Metamorphosis

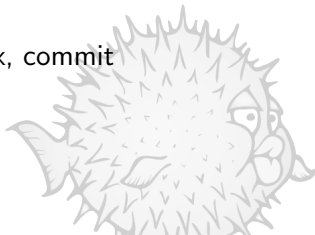
Where are we now?

Conclusion



Motivation

1. Give a talk at EuroBSDCon
2. Enjoy code from the 80's
3. Make it easier to run it in parallel
 - Execute (some parts of) the forwarding path on > 1 CPUs
 - Cleaning from the "top": ioctl and ipforward paths
4. Adapt it to a Plug & Play world
5. Development process: commit early, revert, fix, commit



Agenda

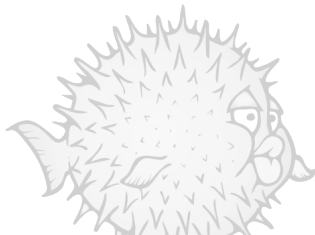
Motivation

Representing Addresses & Routes

Stack Metamorphosis

Where are we now?

Conclusion



What do we use addresses for?

Identify peers

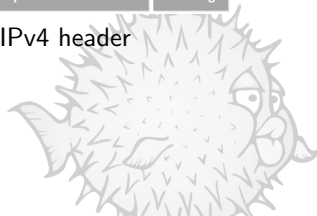
- Who is the receiver?
- Who is the sender?

Direct packets

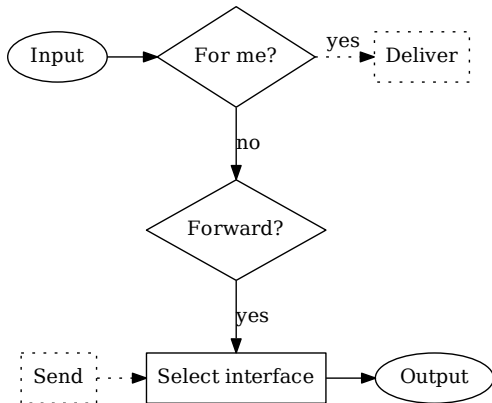
- Where is the destination?

Ver	IHL	TOS	Packet Length	
Identification			Flags	Fragment Offset
Time to Live		Protocol	Header Checksum	
Source Address				
Destination Address				
Options				Padding

IPv4 header

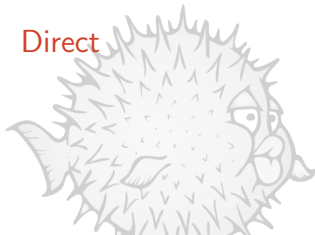


Journey of a packet



Identify

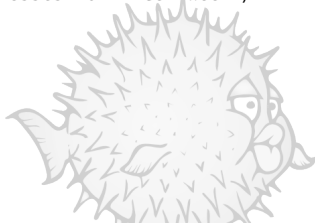
Direct



Representation of an address

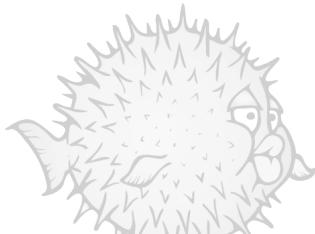
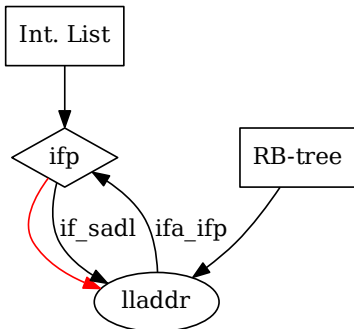
Interface address (*ifa*)

```
struct ifaddr {
    struct sockaddr *ifa_addr;    /* address of interface */
    struct sockaddr *ifa_dstaddr; /* other end of p-to-p link */
#define ifa_broadaddr ifa_dstaddr /* broadcast address interface */
    struct sockaddr *ifa_netmask; /* used to determine subnet */
    struct ifnet *ifa_ifp;       /* back-pointer to interface */
    TAILQ_ENTRY(ifaddr) ifa_list; /* list of addresses for interface */
    [...]
};
```



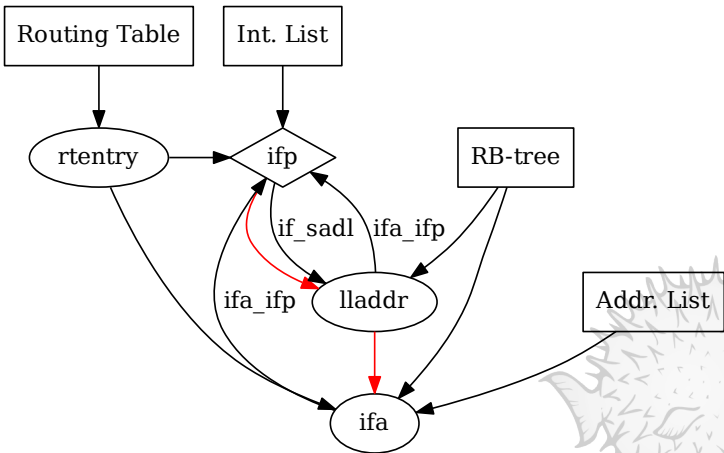
Global data structures

Interface without address

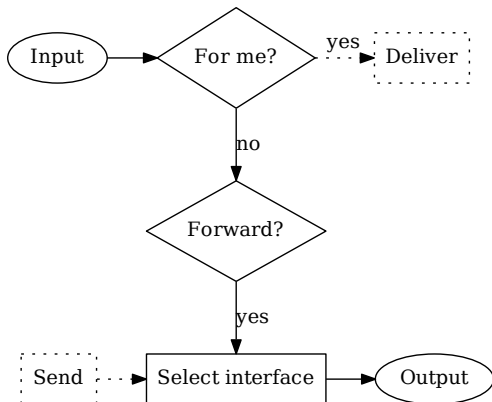


Global data structures

Interface with an address



When are they accessed?



ip_input

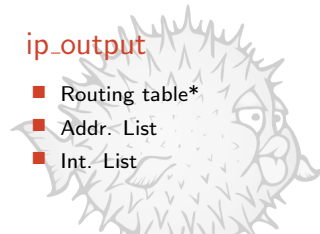
- RB-tree
- Addr. List
- Int. List

ip_forward

- Routing table*

ip_output

- Routing table*
- Addr. List
- Int. List



Agenda

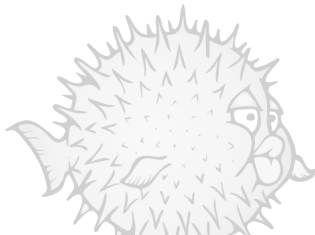
Motivation

Representing Addresses & Routes

Stack Metamorphosis

Where are we now?

Conclusion

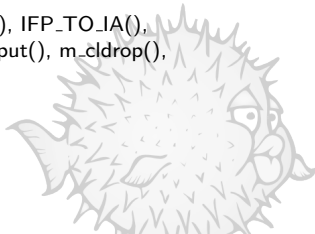


Global lists

1. Get rid of link-layer address lookups
2. Use local (per *ifp*) lists instead of global ones
3. Or simply rewrite the code without the lookup
4. Otherwise (in the process context) use the Interface List

Some modified functions

`carp_set_addr()`, `ether_output()`, `ifa_ifwithnet()`, `ifa_ifwithroute()`, `IFP_TO_IA()`,
`in_localaddr()`, `in_pcbbind()`, `in_selectsrc()`, `ipv4_input()`, `ip_output()`, `m_cldrop()`,
`rip_usrreq()`, `rt_getifa()`...



Interface list

ifa_ifwith**addr**(), ifa_ifwith**dst**addr() and ifa_ifwith**net**()

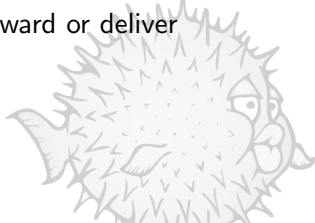
1981: One address per interface (struct *ifnet*)

1985: Per interface list of addresses (struct *ifaddr*)

2010: Global RB-Tree of addresses

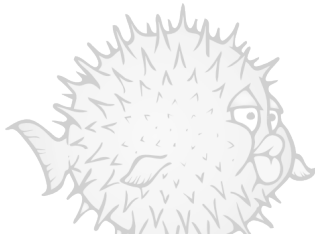
1985: Global list of addresses per protocol (i.e. struct *in_ifaddr*)

1999: KAME uses the routing table to forward or deliver

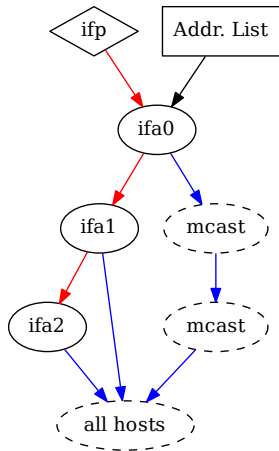


Routing table

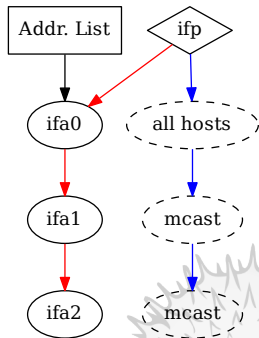
- Use the routing table for address lookups
 - Consolidate KAME's "loobpack" hack
 - RTF_LOCAL** For each configured address
 - RTF_BROADCAST** For every IPv4 subnet
- Only one global structure
 - Easier than maintaining coherency between various structures
 - Needs some love to be accessed in parallel
- Not slower/faster than the actual RB-tree



Protocol multicast addresses



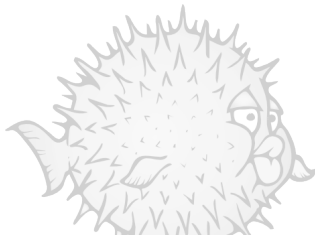
OpenBSD 5.4



OpenBSD 5.5

Related changes

- The link-layer address has been removed from all the lists
 - No need to move this information to the routing table
 - Many many dragons in this code
- `SO_DONTROUTE` is no longer supported
 - No option to bypass the routing table
- Interface indexes are now unique
 - Avoid dangling pointers
- `inet_ntop()` replaces `inet_ntoa()` in the kernel



Agenda

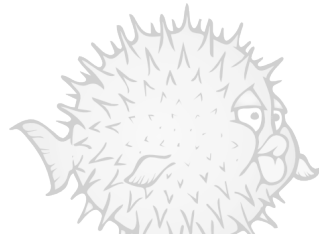
Motivation

Representing Addresses & Routes

Stack Metamorphosis

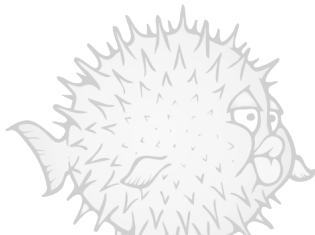
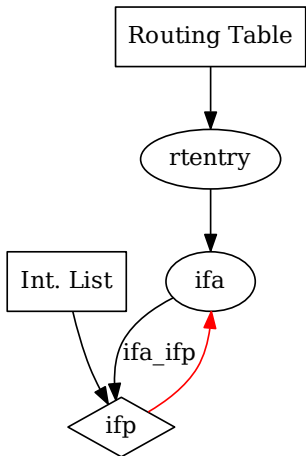
Where are we now?

Conclusion

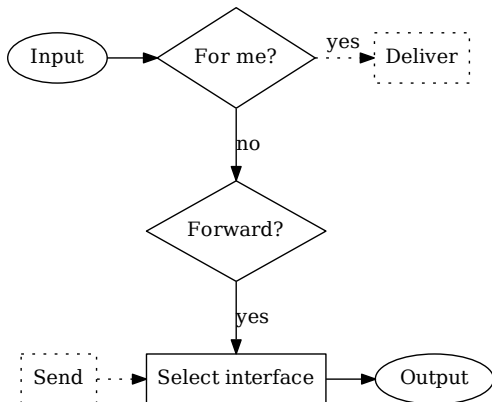


Global data structures

Interface with an address



When are they accessed?



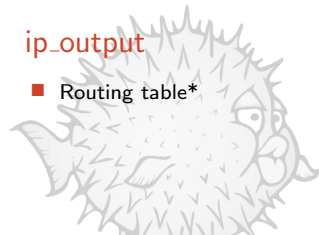
ip_input

- Routing table*

ip_forward

ip_output

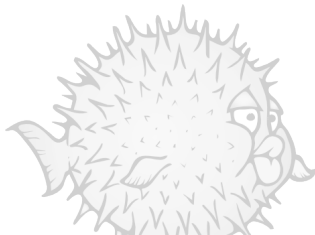
- Routing table*



Well, we're almost there

- Diff to kill the RB-tree is on [tech@](#)
- `RTF_LOCAL` routes still points to `lo0`
- Still doing 2 lookups in the forwarding case

Hopefully integrated for OpenBSD 5.7



Agenda

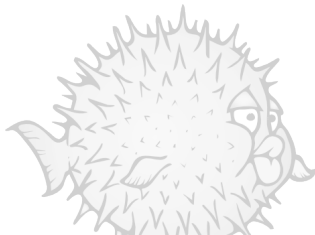
Motivation

Representing Addresses & Routes

Stack Metamorphosis

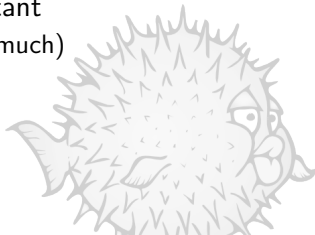
Where are we now?

Conclusion



Conclusion

- Refactoring 30 years old code is hard
 - But we have a pretty good history
- Very few people care because
 - It's not a "feature"
 - There's no visible speed gain
 - Changes always find some dragons
- Understanding what you're changing is important
 - Future developers won't hate you (or not that much)
- Still plenty of dragons



Questions?

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

