Networking from the Bottom Up: Routing and Forwarding

George Neville-Neil

gnn@neville-neil.com

May 11, 2011

Overview

- History and Terminology
- Traditional Router Design
- Forwarding
- Routing
- Interacting with Routing and Forwarding Systems
- Packet Filtering

Design Goals

- Move packets from port A to port B in least time
- Low impact on CPU and other system components
- Good Control and Management of the system

Challenges

- Packet Formats
- Variable Sized Packet Options
- Competing Uses

Packet Format Issues: IPv4

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version|  IHL  |Type of Service|          Total Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Identification        |Flags|      Fragment Offset    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Time to Live |    Protocol   |         Header Checksum       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Source Address                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Destination Address                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Options                    |    Padding    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Routing History

- Internet Message Processor
- Gateway
- Routers
- Switches

Terminology

routing
: Choosing an outgoing interface based on IP destination

forwarding
: Choosing an outgoing interface based on Layer 2 destination

switch
: Forwards packets to the next hop of a LAN

router
: Routes packets between networks, of any type

route
: A piece of state that describes a next hop destination

mask
: Used to disambiguate a route

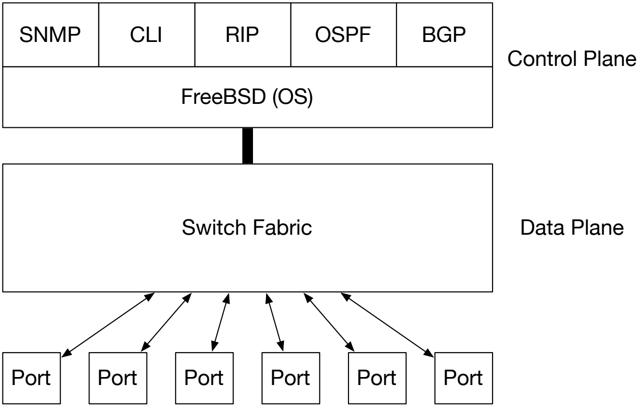
FIB
: Forwarding Information Base

RIB
: Routing Information Base

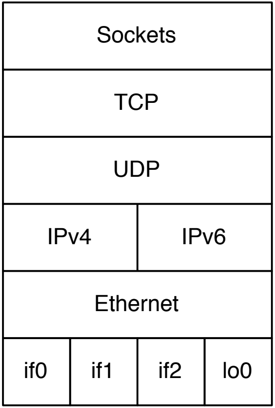
Traditional Router Design

- Control Plane
- Data Plane
- Routing Information Base (RIB)
- Forwarding Information Base (FIB)

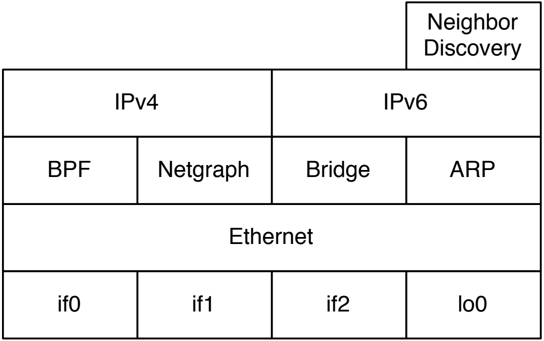
Traditional Router Block Diagram



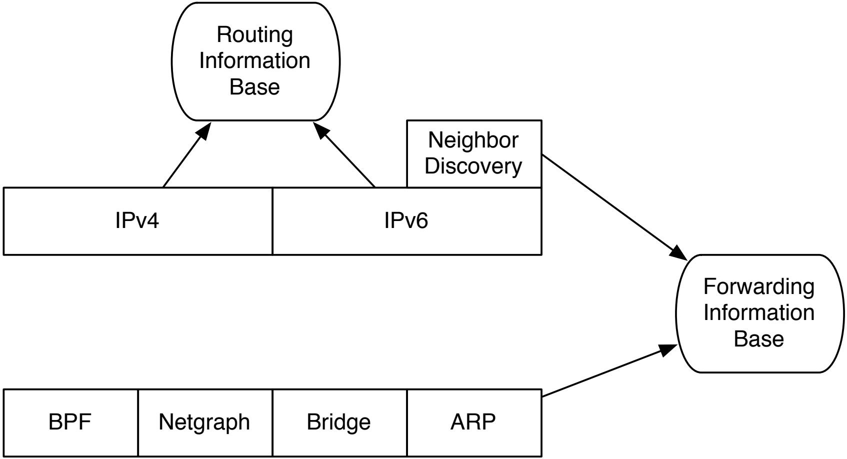
FreeBSD Networking Block Diagram



Forwarding Components



RIB and FIB Connections



Forwarding vs. Routing

- *The most significant difference between routing and forwarding is what part of the packet is inspected.*

Interfering with Packets

- Many components in the kernel are meant to divert or change packets

BPF	Berkeley Packet Filter used for debugging
Dummynet	Testing and debugging
Bridging	A software packet switch
Firewall	Security
PFil	Security
ALTQ	Traffic Shaping and QoS

- Different components grab packets at different locations in the packet processing pipeline

Three Basic Questions

Where	is the packet grabbed?
How	is the packet modified?
When	is the packet buffer freed?

Berkeley Packet Filter

- Captures packets as near the interface as possible
- Matches packet data vs. a filter
- Copies packet without modification
- Requires root privileges

BPF and Drivers

- BPF$_M TAPMUSTbecalledfromthedrivers$

BPF Callgraph

1. bfp_mtap
2. bpf_filter
3. catch_packet

Netgraph

- A framework for arbitrary packet processing
- Nodes implement packet transformation
- Edges allow nodes to be arranged into an arbitrary graph
- Currently used to implement Bluetooth and ATM network stacks

Bridging

- Spans packets at layer 2
- Packets are copied, in software, to one or more interfaces
- Need to protect against looping packets backwards
- Maintains its own forwarding table

Callgraph

1. bridge_input
2. bridge_span
3. GRAB_OUR_PACKETS
4. bridge_forward

Packet Demultiplexing and netisr

- The netisr implements a software interrupt handler
- A holdover from before interrupt threads
- Packets are normally carried through from the driver's interrupt thread.
- Packets can be queued for protocols but this is inefficient
- Protocols register a handler with the netisr system

Ethernet Packet Demultiplexing

- ether_input

Fast Forwarding

- Shortcut in IPv4 forwarding
- Allows well formed packets to be forwarded more quickly
- ip_fastforward()
- Any diversion from the norm results in normal forwarding

Fast Forwarding Code

- ip_fastforward

Forwarding Information Base

- Stores Layer 2 Address Information
- Allows lookup of next hop hardware address
- This is not routing
- Used by ARP and Neighbor Discovery

FIB Entries

```
struct llentry {
        LIST_ENTRY(llentry)      lle_next;
        struct rwlock            lle_lock;
        struct lltable           *lle_tbl;
        struct llentries         *lle_head;
        struct mbuf              *la_hold;
        int                      la_numheld;  /* # of packets currently held */
        time_t                   la_expire;
        uint16_t                 la_flags;
        uint16_t                 la_asked;
        uint16_t                 la_preempt;
        uint16_t                 ln_byhint;
        int16_t                  ln_state;     /* IPv6 has ND6_LLINFO_NOSTATE == -2 */
        uint16_t                 ln_router;
        time_t                   ln_ntick;
        int                      lle_refcnt;

        union {
                uint64_t        mac_aligned;
                uint16_t        mac16[3];
#ifdef OFED
                uint8_t         mac8[20];     /* IB needs 20 bytes. */
#endif
        } ll_addr;

        /* XXX af-private? */
        union {
                struct callout  ln_timer_ch;
                struct callout  la_timer;
        } lle_timer;
```

FIB Tables

```
struct lltable {
        SLIST_ENTRY(lltable)      llt_link;
        struct llentries          lle_head[LLTBL_HASHTBL_SIZE];
        int                       llt_af;
        struct ifnet              *llt_ifp;

        struct llentry *          (*llt_new)(const struct sockaddr *, u_int);
        void                      (*llt_free)(struct lltable *, struct llentry *);
        void                      (*llt_prefix_free)(struct lltable *,
                                      const struct sockaddr *prefix,
                                      const struct sockaddr *mask);
        struct llentry *          (*llt_lookup)(struct lltable *, u_int flags,
                                      const struct sockaddr *l3addr);
        int                       (*llt_rtcheck)(struct ifnet *, u_int flags,
                                      const struct sockaddr *);
        int                       (*llt_dump)(struct lltable *,
                                       struct sysctl_req *);
};
MALLOC_DECLARE(M_LLTABLE);
```

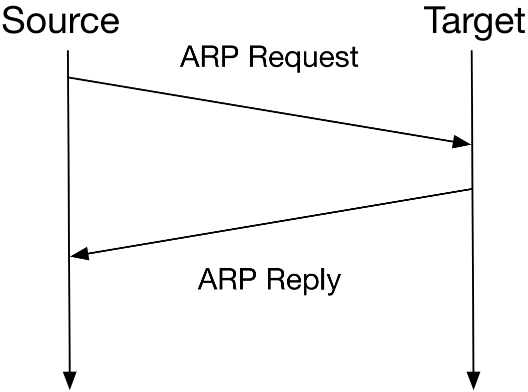
FIB APIs

new	create a new entry, with associated locks and timers
free	destroy an entry, handle locks and timers
prefix_free	destroy all entries with an associated prefix and mask
rtcheck	
lookup	find a matching entry in the FIB
dump	dump the entire table

ARP

- Address Resolution Protocol
- RFC 826
- Map an IPv4 address to a hardware address
- Runs directly on Ethernet or other layer 2 packets

Protocol Diagram



ARP Use of FIB

- FIB is not a routing table
- Implemented as a hash table
- Each protocol has its own Link Layer Table
- Every Link Layer Table has its own methods

ARP Lookup

- in_lltable_lookup

Neigbor Discovery

- RFC 4861
- Maps an IPv6 Address to a hardware address
- Runs on top of ICMPv6 packets
- Layer 2 agnostic
- Stores address mappings in the FIB

Neighbor Lookup

- in6_lltable_lookup

Data Structures

- Routing Table
- Routing Entry

Route Structure

```
struct route {
        struct  rtentry *ro_rt;
        struct  llentry *ro_lle;
        struct  sockaddr ro_dst;
};
```

Route Entry

```
struct rtentry {
        struct  radix_node rt_nodes[2]; /* tree glue, and other values */
        /*
         * XXX struct rtentry must begin with a struct radix_node (or two!)
         * because the code does some casts of a 'struct radix_node *'
         * to a 'struct rtentry *'
         */
#define rt_key(r)       (*((struct sockaddr **)(&(r)->rt_nodes->rn_key)))
#define rt_mask(r)      (*((struct sockaddr **)(&(r)->rt_nodes->rn_mask)))
        struct  sockaddr *rt_gateway;   /* value */
        int     rt_flags;               /* up/down?, host/net */
        int     rt_refcnt;              /* # held references */
        struct  ifnet *rt_ifp;          /* the answer: interface to use */
        struct  ifaddr *rt_ifa;         /* the answer: interface address to use */
        struct  rt_metrics_lite rt_rmx; /* metrics used by rx'ing protocols */
        u_int   rt_fibnum;              /* which FIB */
#ifdef _KERNEL
        /* XXX ugly, user apps use this definition but don't have a mtx def */
        struct  mtx rt_mtx;             /* mutex for routing entry */
#endif
};
```

Reference Counting

- Many subsystems can place a hold on a route
- Each time a hold is placed on a route its reference count is increased
- Routes with reference counts greater than 0 *cannot* be freed

Locking

- Routes are accessed from various subsystems
- Updating a route requires holding a lock on the route
- Changing the reference count requires use of the lock

The Patricia Trie

- Practical Algorithm to Retrieve Information Coded in Alphanumeric
- The data structure that holds all routes
- Each protocol has its own tree structure, rooted in a global variable
- Allows efficient storage and lookup of routes
- Never used in high end, production router

Route Lookup

- Necessary pieces of information

  Key Address we're seeking

  Mask Network mask, used in backtracking

Route Lookup Algorithm

1. Start at high order bit
2. Compare bit in Key with bit in the current Node
3. If bits AND to 1 take left path
4. else take right path
5. When leaf is reached and Key with leaf node's Address
6. If Key AND Node Address is equal to Key we have a match
7. else backtrack

Backtracking

- Can't find a host route? Backtrack.
- Go up from failed leaf to immediate parent
- Take the other path
- Mask off Key's host specific bits
- If masked Key AND Node Address equal masked Key we have a network match

Adding a Route

- in_addroute

Matching a Route

- in_matroute

Deleting a Route

- Follows a somewhat more tortuous route
- Learned routes are deleted via a timeout
- User set route as deleted by the route command
- rtrequest1_fib

Equal Cost Multipath Routing

- Used for load balancing across routes
- enabled with option RADIX_MPATH
- Nodes have lists of valid addresses
- Uses a hash to select from the list of routes
- Can play havoc with TCP

Processing Model

- Message passing and Event driven model
- Uncommon in Unix like systems
- Userland programs wait for events from the kernel
- Events arrive as routing messages

Routing Messages

```
struct rt_msghdr {
        u_short rtm_msglen;      /* to skip over non-understood messages */
        u_char  rtm_version;     /* future binary compatibility */
        u_char  rtm_type;        /* message type */
        u_short rtm_index;       /* index for associated ifp */
        int     rtm_flags;       /* flags, incl. kern & message, e.g. DONE */
        int     rtm_addrs;       /* bitmask identifying sockaddrs in msg */
        pid_t   rtm_pid;         /* identify sender */
        int     rtm_seq;         /* for sender to identify action */
        int     rtm_errno;       /* why failed */
        int     rtm_fmask;       /* bitmask used in RTM_CHANGE message */
        u_long  rtm_inits;       /* which metrics we are initializing */
        struct  rt_metrics rtm_rmx; /* metrics themselves */
};
```

User Level API

Add	a route
Delete	a route
Get	
Change	
Lock	

Events

New Address
: A new address was added to the table

New Multicast Address
: Same as above for multicast

Miss
: A routing lookup failed

Interface Change
: An interface was modified

Interface Announce
: An interface was added or removed

IEEE 802.11 message
: Wireless specific message

Overview

- Packet filtering interferes with packet forwarding
- Has many several:
  - Firewalls
  - Traffic Shaping
  - Protocol Testing

PFIL

- Packet Filtering System
- Allows relatively arbitrary hooks in packet flow
- Three choices:
  - Drop
  - Modify
  - Continue

PFIL locations

- Exists only in the lower layers
  - Bridge
  - Fast Forwarding
  - IP Input
  - IP Output
  - IP6 Forward
  - IP6 Input
  - IP6 Output

IPFW and Dummynet

- Is a consumer of PFIL
- Uses rules to decide what to do with packets

Wrap Up

- Forwarding and Routing are Different but Related
- Forwarding Information Base
- Routing Information Base
- Equal Cost Multipath Routing
- Routing Sockets
- Packet Filtering