

IPV6

Linux Kernel Networking (3)- advanced topics



Rami Rosen
ramirose@gmail.com

Haifux, April 2008

www.haifux.org

Linux Kernel Networking (3)- advanced topics

- Note:
- This lecture is a sequel to the following two lectures I gave:
- **Linux Kernel Networking lecture**
 - <http://www.haifux.org/lectures/172/>
 - **slides**:<http://www.haifux.org/lectures/172/netLec.pdf>
- **Advanced Linux Kernel Networking - Neighboring Subsystem and IPSec lecture**
 - <http://www.haifux.org/lectures/180/>
 - **slides**:<http://www.haifux.org/lectures/180/netLec2.pdf>

Contents

- IPV6
 - General
 - ICMPV6
 - Radvd
 - Autoconfiguration
- Network Namespaces
- Bridging Subsystem
- Pktgen kernel module.
- Tips
- Links and more info

Scope

- We will not deal with wireless.
- The L3 network protocol we deal with is ipv4/ipv6, and the L2 Link Layer protocol is Ethernet.

IPV6 -General

- Discussions started at IETF in 1992 (IPng).
- First Specification: RFC1883 (1995).
- Was deprecated by RFC2460 (1998)
- Main reason for IPV6: shortage of IPv4 addresses.
 - The address space is enlarged in IPV6 from 2^{32} to 2^{128} . (which is by 2^{96}).
- Secondary reason: improvements over IPV4.
 - For example: using ICMPV6 as a neighbour protocol instead of ARP.
 - Fixed IP header (40 bytes) (in IPV4 it is 20-60 bytes).

IPV6 -General

- Usually in IPV4, Mobile devices are behind NAT.
- Using mobile IPV6 devices which are not behind a NAT can avoid the need to send Keep-Alive.
- Growing market of mobile devices
 - Some say number of mobile devices will exceed 4 billion in the end of 2008.
- IPSec is mandatory in IPV6 and optional in IPV4.
 - Though most operating systems implemented IPSec also in IPv4.

IPV6 - history

- In the end of 1997 **IBM's AIX 4.3** was the first commercial platform that supported IPv6
- **Sun Solaris** has IPv6 support since Solaris 8 in February 2000.
- 2007: **Microsoft Windows Vista** (2007) has IPv6 supported and enabled by default.
- February 2008: **IANA** added DNS records for the IPv6 addresses of **six of the thirteen root name servers** to enable Internet host to communicate by IPV6.

- Around the world
 - There was a big IPV6 experiment in China
 - USA, Japan, Korea, France: sites which operate with IPV6.
 - Israel: experiment in Intenet Zahav
 - <http://www.m6bone.net/>
 - Freenet6: <http://go6.net/4105/freenet.asp>

IPV6 in the Linux Kernel

- IPV6 Kernel part was started long ago - 1996 (by Pedro Roque), based on the BSD API; It was **Linux kernel 2.1.8**.
- When adding IPV6 support to the Linux kernel, almost only the Network layer (L3) is changed.
 - Almost no other layer needs to be changed because each layer operates independently.

IPV6 in the Linux Kernel-contd.

- **USAGI** Project was founded in 2000 in Japan.
 - “Universal Playground for Ipv6”.
 - Held by volunteers, mostly from Japan. The **USAGI** aimed at providing missing implementation according to the new IPV6 RFCs.
 - Awarded with IPV6 ready logo
- Yoshifuki Hideaki-member of USAGI Project; Keio University
- comaintainer of IPV6 in the Linux Kernel.(also maintainer of iputils)



IPV6 -General

- Yoshfuji git tree.
- From time to time, the main networking tree pulls this git tree.
- `git-clone git://git.linux-ipv6.org/gitroot/yoshfuji/linux-2.6-dev.git inet-2.6.26`
- This git tree supports IPV6 Multicast Routing. (with pim6sd daemon, ported from kame; PIM-SM stands for Protocol Independent Multicast—Sparse Mode).
 - *Based on Mickael Hoerdts IPv6 Multicast Forwarding Patch.*
 - <http://clarinet.u-strasbg.fr/~hoerdts/>
 - Hoerdts patch is partially based on mrouted.
- Many patches in 2.6.* kernel are from the USAGI project.

- There was also the **KAME** project in Japan, sponsored by six large companies.
- It aimed at providing a free stack of IPv6, IPsec, and Mobile IPv6 for BSD variants.
- <http://www.kame.net/>
- Sponsored by Hitachi and Toshiba and others.
- Mobile IPV6:
 - HUT - Helsinki University of Technology
 - <http://www.mobile-ipv6.org/>

IPV6 -General

- Many router vendors support IPV6:
 - Cisco supports IPv6 since IOS 12.2(2)T
 - Hitachi
 - Nortel Networks
 - Juniper Networks, others.
 - <http://www.ietf.org/IESG/Implementations/ipv6-implementations.txt>
- Drawbacks of IPV6
 - Currently LVS is not implemented in IPV6.
 - Takes effort to port existing, IPV4 applications.
 - Tunnels, transitions (IPv6 and IPv4)

IPv6 Addresses

- RFC 4291, IP Version 6 Addressing Architecture
- Format of IPv6 address:
- 8 blocks of 16 bits => 128 bits.
- `xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx`
- Where x is a hexadecimal digit.
- Leading zeroes can be replaced with "::<" , but only once.
- Localhost address:
 - `0000:0000:0000:0000:0000:0000:0000:0001`
 - Or, in compressed form:
 - `::1`

IPv6 Addresses - contd

- No broadcast address in IPv6 as in IPv4.
- Global addresses: 2001:....
 - There are more.
- Link Local : FE80:....

IPV6 -General

- Caveat:
 - Sometimes ipv6 is configured as a module.
 - You cannot rmmmod the ipv6 module.
- How can I know if my kernel has support for IPV6?
 - Run: *ls /proc/net/if_inet6*
- Managing IP address:
- *ifconfig eth0 inet6 add 2001:db8:0:1:240:95ff:fe30:b0a3/64*
- *ifconfig eth0 inet6 del 2001:db8:0:1:240:95ff:fe30:b0a3/64*

IPV6 -General

- Can be done also by “ip” command (IPROUTE2).
- `ip -6 addr`
- Using tcpdump to monitor ipv6 traffic:
 - `tcpdump ip6`
- or , for example:
 - `tcpdump ip6 and udp port 9999.`
- For wireshark fans:
 - `tethereal -R ipv6`

IPV6 -General

- To show the Kernel IPv6 routing table :
 - *route -A inet6*
 - *Also: ip -6 route show*
- ssh usage: *ssh -6 2001:db8:0:1:230:48ff:fe61:e5e0*
- *traceroute6 -i eth0 fe80::20d:60ff:fe9a:26d2*
- *netstat -A inet6*
- *ip6tables* solution exist in IPV6.

IPV6 -General

- *tracepath6* finds PMTU (path MTU).
 - This is done using IPV6_MTU_DISCOVER and IPV6_PMTUDISC_PROBE socket options.
 - using a UDP socket.

ICMPV6

- In IPV6, the neighboring subsystem uses ICMPV6 for Neighboring messages (instead of ARP in IPV4).
- There are 5 types of ICMP codes for neighbour discovery messages:

Message

ICMPV6 code

NEIGHBOUR SOLICITATION

(135) -parallel to ARP request

in IPV4

NEIGHBOUR ADVERTISEMENT

(136) -parallel to ARP reply in

IPV4

ROUTER SOLICITATION	(133)	
ROUTER ADVERTISEMENT	(134)	// see sniff below
REDIRECT	(137)	

- **ROUTER ADVERTISEMENT** can be periodic or on demand.
- When **ROUTER ADVERTISEMENT** is sent as a reply to a **ROUTER SOLICITATION**, the destination address is **unicast**.
When it is periodic, the destination address is a **multicast** (all hosts).

Statefull and Stateless config

- There are two ways to configure IPV6 addresses on hosts (except configuring it manually):
- **Statefull:** DHCPV6 on a server.
 - RFC3315, Dynamic Host Configuration Protocol for IPv6 (DHCPv6).
- **Stateless:** for example, RADVD or Quagga on a server.
 - RFC 4862 - IPv6 Stateless Address Autoconfiguration (SLAAC) from 2007 ; Obsoletes RFC 2462 (1998).

In RADVD, you declare a prefix that only hosts (not routers) use. You can define more than one prefix.

- **Special Addresses:**

- **All nodes (or : All hosts) address: FF02::1**

- *ipv6_addr_all_nodes()* sets address to FF02::1

- **All Routers address: FF02::2**

- *ipv6_addr_all_routers()* sets address to FF02::2

Both in *include/net/addrconf.h*

- IPV6: All addresses starting with FF are multicast address.
- IPV4: Addresses in the range 224.0.0.0 – 239.255.255.255 are multicast addresses (class D).
- see <http://www.iana.org/assignments/ipv6-address-space>

- `ping6 -I eth0 FF02::2` or `ping6 -I eth0 ff02:0:0:0:0:0:0:2`
will cause all the routers to reply.
- This means that all machines on which
/proc/sys/net/ipv6/conf/eth/forwarding* is 1 will reply.

RADVD

- RADVD stands for ROUTER ADVERTISEMENT daemon.
- Maintainer: Pekka Savola
- <http://www.litech.org/radvd/>
- Sends **ROUTER ADVERTISEMENT** messages.
- The handler for all neighboring messages is *ndisc_rcv()*.
(*net/ipv6/ndisc.c*)
- When **NDISC_ROUTER_ADVERTISEMENT** message arrives from radvd, it calls *ndisc_router_discovery()*.
(*net/ipv6/ndisc.c*)

RADVD - contd

- If the receiving machine **is a router** (or is configured **not to accept router advertisement**), the **Router Advertisement** is not handled: see this code fragment from *ndisc_router_discovery()* (*net/ipv6/ndisc.c*)

```
if (in6_dev->cnf.forwarding || !in6_dev->cnf.accept_ra) {  
    in6_dev_put(in6_dev);  
    return;
```

- *addrconf_prefix_rcv()* eventually tries to create an address using the prefix received from radvd and the mac address of the machine (*net/ipv6/addrconf.c*).

RADVD - contd

- Adding the IPV6 address is done in `ipv6_add_addr()`
 - How can we be sure that there is no same address on the LAN ?
 - We can't !
 - Therefore we set this address first to be tentative
 - In `ipv6_add_addr()`:
 - `ifa->flags = flags | IFA_F_TENTATIVE;`
 - *This means that initially this address cannot communicate with other hosts. (except for neighboring messages).*

RADVD - contd

- Then we start DAD (*by calling `addrconf_dad_start()`*)
- **DAD** is “Duplicate Address Detection”.
- Upon successful completion of DAD, the `IFA_F_TENTATIVE` flag is removed and the host can communicate with other hosts on the LAN. The flag is set to be `IFA_F_PERMANENT`.
- Upon failure of DAD, the address is deleted.
- You see a message like this in the kernel log:
 - **eth0: duplicate address detected!**

RADVD - contd

- Caveat:
- When using radvd official FC8 rpm, you will see, in /var/log/messages, the following message after starting the daemon:

...

```
radvd[2614]: version 1.0 started
```

```
radvd[2615]: failed to set CurHopLimit (64) for eth0
```

- You may ignore this message.
- This is due to that we run the daemon as user radvd.
- This was fixed in radvd 1.1 (still no Fedora official rpm).

RADVD - contd

- radvd: sending router advertisement

// from *radvd-1.0/send.c*

```
send_ra()
```

```
{
```

```
...
```

```
    addr.sin6_family = AF_INET6;
```

```
    addr.sin6_port = htons(IPPROTO_ICMPV6);
```

```
    memcpy(&addr.sin6_addr, dest, sizeof(struct in6_addr));
```

```
    memset(&buff, 0, sizeof(buff));
```

```
radvert = (struct nd_router_advert *) buff;  
radvert->nd_ra_type = ND_ROUTER_ADVERT;
```

...

- and we have in /usr/include/netinet/icmpv6.h:
 - #define ND_ROUTER_SOLICIT 133
 - #define ND_ROUTER_ADVERT 134
- This Router Advertisement is sent to all hosts address:
 - **FF02::1**

- `nd_router_advert` structure is declared in */usr/include/netinet/icmp6.h*.
- It includes `icmpv6` header.
- Is there a protection against sending malicious Router Advertisements ?
- No thing as rejecting “unsolicited arp replies” as in IPV4 (which is the default behaviour, in order to prevent ARP Cache poisoning)

radvd.conf - example:

```
interface eth0 #see man radvd.conf
```

```
{
```

```
    AdvSendAdvert on;
```

```
    MaxRtrAdvInterval 30;
```

```
    prefix 2002:db8:0:1::/64
```

```
    {
```

```
        AdvOnLink on;
```

```
        AdvAutonomous on;
```

```
    };
```

```
};
```

radvd.conf – example (contd)

- The prefix length **MUST** be 64.
- See RFC2464, **Transmission of IPv6 Packets over Ethernet Networks** and RFC4291- **IP Version 6 Addressing Architecture**.
- Caveat:
- If the prefix length will be different than 64 than the Router Advertisement will be rejected.
- Caveat: You will not notice it, unless your syslog prints KERN_DEBUG messages (see man syslog.conf)
- In case the syslog is configured for printing kernel debug messages, you will see this messages in the kernel log

IPv6 addrconf: prefix with wrong length

radvd.conf – example (contd)

- Caveat 2:
- You cannot start radvd service if there is no link local address configured on your machine. Trying to do so will result with:

```
radvd: no linklocal address configured for (null)  
radvd: error parsing or activating the config file:  
[FAILED]
```
- Even if it was possible, the kernel would reject a Router Advertisements originating from machines without link local IPV6 address.
- `radvd -d 5 -m stderr` (for starting with many debug messages)

Router Advertisement with Prefix Information option sniff

The image shows a Wireshark capture of an ICMPv6 Router Advertisement packet. The packet is identified as No. 1, Time 0.000000, Source fe80::215:58ff:fe95:5be6, Destination ff02::1, Protocol ICMPv6, and Info Router advertisement. The packet structure is as follows:

- Internet Protocol Version 6
 - 0110 = Version: 6
 - 0000 0000 = Traffic class: 0x00000000
 - 0000 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
 - Payload length: 56
 - Next header: ICMPv6 (0x3a)
 - Hop limit: 255
 - Source: fe80::215:58ff:fe95:5be6 (fe80::215:58ff:fe95:5be6)
 - Destination: ff02::1 (ff02::1)
- Internet Control Message Protocol v6
 - Type: 134 (Router advertisement)
 - Code: 0
 - Checksum: 0x97b5 [correct]
 - Cur hop limit: 64
 - Flags: 0x00
 - Router lifetime: 0
 - Reachable time: 0
 - Retrans timer: 0
 - ICMPv6 Option (Prefix information)
 - Type: Prefix information (3)
 - Length: 32
 - Prefix length: 64
 - Flags: 0xc0
 - Valid lifetime: 2592000
 - Preferred lifetime: 604800
 - Prefix: 2002:db8:0:1::

Type (icmpv6.type), 1 byte P: 3 D: 3 M: 0

- `valid_lft` - how long this prefix is valid, in seconds.
- `preferred_lft` - how long this address can be in preferred state, in seconds.
- `ip -6 addr` (note: `ifconfig` does not show these time values)

```
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu  
1500 qlen 1000
```

```
inet6 2004:db8:0:1:240:95ff:fe30:b0a3/64 scope global dynamic
```

```
    valid_lft 279sec preferred_lft 99sec
```

```
inet6 fe80::240:95ff:fe30:b0a3/64 scope link
```

```
    valid_lft forever preferred_lft forever
```

- When **preferred time** is finished, this IPv6 address will stop communicating. (will not answer ping6, etc).
- When the **valid time** is over, the IPV6 address is removed.
- *ipv6_del_addr() in net/ipv6/addrconf.c is responsible for deleting non valid addresses (called from **addrconf_verify()**)*
- This is useful for renumbering.
 - RFC 2894 - Router Renumbering for IPv6

- Radvd also send its mac address of itself as part of the options.
- This enable the receiving host to add/update it neighbour table accordingly with the mac address of the router:
- From *ndisc_router_discovery()*: (*net/ipv6/ndisc.c*)

...

```
lladdr=ndisc_opt_addr_data(ndopts.nd_opts_src_lladdr,skb->dev);
```

...

```
neigh_update(neigh, lladdr, ...)
```

...

- lladdr is the mac address of the router, passed in the options of Router Advertisement.

Router Advertisement with Link Layer option sniff

The image shows a Wireshark capture window titled "radvdEther6.eth - Wireshark". The main pane displays a list of captured packets with the following columns: Source, Destination, Protocol, and Info. The packets are:

Source	Destination	Protocol	Info
fe80::215:58ff:fe95:5	ff02::1	ICMPv6	Router advertisement
2001:db8:0:1:290:27ff	2001:7b8:3:1f:0:2:53	DNS	Standard query A 13
2001:db8:0:1:290:27ff	2001:7b8:3:1f:0:2:53	DNS	Standard query A 13
fe80::215:58ff:fe95:5	ff02::1	ICMPv6	Router advertisement
fe80::290:27ff:fe34:2	fe80::215:58ff:fe95::	ICMPv6	Neighbor solicitation
fe80::215:58ff:fe95:5	fe80::290:27ff:fe34::	ICMPv6	Neighbor advertisement

The bottom pane shows the detailed view of the selected ICMPv6 Router Advertisement packet. The options are:

- Cur hop limit: 64
- Flags: 0x00
- Router lifetime: 90
- Reachable time: 0
- Retrans timer: 0
- ICMPv6 Option (Prefix information)
- ICMPv6 Option (Source link-layer address)
 - Type: Source link-layer address (1)
 - Length: 8
 - Link-layer address: 00:15:58:95:5b:e6

Option (icmpv6.option), 32 bytes

radvd.conf and a default router

- Specifying AdvDefaultLifetime in radvd.conf will cause the host to add the radvd router as a default router.
 - Unless */proc/sys/net/ipv6/conf/eth0/accept_ra_defrtr* is 0.
- This default router has a limited lifetime. It will expire after the value specified for AdvDefaultLifetime.
- Maximum AdvDefaultLifetime value is 18.2 hours.

Example (after setting AdvDefaultLifetime to 8000)

- *ip -6 route show default*
- default via fe80::215:58ff:fe95:5be6 dev eth0 proto kernel metric 1024 **expires 7996sec** mtu 1420 advmss 1360 hoplimit 64

radvd.conf and default router-cont.

- When we stop the radvd daemon this will send a Neighbour Advertisement with Router Lifetime as 0.
- This will cause the hosts which receive this message to delete the default router.
- Implemented by: *ip6_del_rt()* called from *ndisc_router_discovery()* in *net/ipv6/ndisc.c*
- You can also set MTU in radvd.conf
- This is not the MTU you see in ifconfig, but you see it in */proc/sys/net/ipv6/conf/eth0/mtu*.
- Radvd builtin util: **radvdump** prints out the contents of incoming router advertisements sent by radvd.

IPV6 :Neighboring Solicitation sniff

The image shows a Wireshark capture window titled "ipv6_ping_and_neighbour.cap". The main display area shows a list of captured packets. The third packet is highlighted in blue and is an ICMPv6 Neighbor solicitation. The packet details pane is expanded to show the structure of the IPv6 header and the ICMPv6 payload.

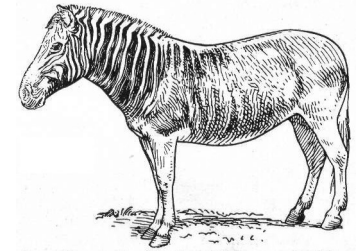
Destination	Protocol	Info
:215:58ff:fe95:5be6	fe80::240:95ff:fe30:b	ICMPv6 Echo request
:240:95ff:fe30:b0a3	fe80::215:58ff:fe95:5	ICMPv6 Echo reply
:240:95ff:fe30:b0a3	fe80::215:58ff:fe95:5	ICMPv6 Neighbor solicitation
:215:58ff:fe95:5be6	fe80::240:95ff:fe30:b	ICMPv6 Neighbor advertisement

Packet 3 Details:

- Ethernet II, Src: RPTinter_50:00:a5 (00:40:95:50:00:a5), Dst: FOXCOM
- Internet Protocol Version 6
 - 0110 = Version: 6
 - 0000 0000 = Traffic class: 0x00000000
 - 0000 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
 - Payload length: 32
 - Next header: ICMPv6 (0x3a)
 - Hop limit: 255
 - Source: fe80::240:95ff:fe30:b0a3 (fe80::240:95ff:fe30:b0a3)
 - Destination: fe80::215:58ff:fe95:5be6 (fe80::215:58ff:fe95:5be6)
- Internet Control Message Protocol v6
 - Type: 135 (Neighbor solicitation)
 - Code: 0
 - Checksum: 0x83d7 [correct]
 - Target: fe80::215:58ff:fe95:5be6 (fe80::215:58ff:fe95:5be6)
 - ICMPv6 Option (Source link-layer address)

Type (icmpv6.type), 1 byte | P: 19 D: 19 M: 0

Quagga



- Quagga replaces Zebra
- <http://www.quagga.net/>
- Many routing protocols (BGP, OSPF, RIP, others)
- Supports IPV6
- Supports sending Router Advertisements.

interface eth0

ipv6 nd send-ra

ipv6 nd prefix-advertisement 2001:0db8:0005:0006::/64

Privacy Extensions

- Since the address is build using a prefix and MAC address, the identity of the machine can be found.
- To avoid this, you can use Privacy Extensions.
 - This adds randomness to the IPV6 address creation process. (calling *get_random_bytes()* for example).
- RFC 3041 - Privacy Extensions for Stateless Address Autoconfiguration in IPv6.
- You need CONFIG_IPV6_PRIVACY to be set when building the kernel.

- Hosts can disable receiving Router Advertisements by setting */proc/sys/net/ipv6/conf/all/accept_ra* to 0.
- Hosts can request Router Advertisements by sending a **Router Solicitation** message.

Autoconfiguration

- When a host boots, (and its cable is connected) it first creates a Link Local Address.
 - A Link Local address starts with **FE80**.
 - This address is tentative (only works with ND messages).
- The host sends a **Neighbour Solicitation** message.
 - The target is its tentative address, the source is all zeros.
 - This is **DAD** (Double Address Detection).
- If there is no answer in due time, the state is changed to permanent. (IFA_F_PERMANENT)

Autoconfiguration - contd.

- Then the host send Router Solicitation.
 - The target address of the Router Solicitation message is the All Routers multicast address **FF02::2**
 - All the routers reply with a Router Advertisement message.
 - The host sets address/addresses according to the prefix/prefixes received and starts the DAD process as before.

Autoconfiguration - contd.

- At the end of the process, the host will have two (or more) IPv6 addresses:
 - Link Local IPV6 address.
 - The IPV6 address/addresses which was built using the prefix. (in case that there is one or more routers sending RAs).
- There are three trials by default for sending Router Solicitation.
 - It can be configured by:
 - */proc/sys/net/ipv6/conf/eth0/router_solicitations*

- If a host boots when its cable is disconnected it will not get an IPV6 address.
- Connecting the cable will trigger an event (NETDEV_CHANGE) in *addrconf_notify()* and will result in sending Router Solicits (calling `ndisc_send_rs`) and eventually autoconfiguration will set an IPV6 address to the host.

Optimistic DAD



- Do not wait till DAD is completed, and allow hosts to communicate with peers before DAD has finished successfully
- Target: to reduce latencies in the DAD process.
- The kernel should be build with: **CONFIG_IPV6_OPTIMISTIC_DAD**.
- Very few apps need Optimistic DAD ; Usually the DAD process of DAD takes **less than 2 seconds**.
- RFC 4429 , Optimistic Duplicate Address Detection (DAD) for IPv6.

IPV6 Fragmentation

- In IPV6, fragmentation is **not** done by routers (as in IPV4).
- The Minimum MTU is IPV6 is 1280.
- It is the responsibility of the host (**sender**) to fragment packets.
- Path MTU discovery is done by ICMPV6
 - ICMPV6_PKT_TOOBIG messages.
 - RFC 1981, Path MTU Discovery for IP version 6.

- Lookup in the IPV6 routing tables is done by *fib6_lookup()*
 - (*net/ipv6/ip6_fib.c*)
- The parameters for the lookup are the root of the table and the source and destination IPV6 address. (struct *in6_addr*)
- The result of the lookup is saved in *rt6_info*.
- *rt6_info* is the parallel of *rtable* in IPV4.

// from *include/net/ip6_fib.h*

```
struct rt6_info
{
union {
struct dst_entry dst;
} u;
```

IPV6 - contd

- Enable forwarding:
- `echo "1" > /proc/sys/net/ipv6/conf/all/forwarding`
- For Multicast Routing forwarding, there will be in the future:
 - *`/proc/sys/net/ipv6/conf/all/mc_forwarding`*

IPv6 header – 40 bytes

- `include/linux/ipv6.h`

Version (4)	Priority/Traffic Class (4)	Flow Label (24)
Payload Length (16)	Next Header (8)	Hop Limit (8)
Source Address (128 bits=>16 bytes)		
Destination Address (128 bits=>16 bytes)		

IPV6 header - contd.

- The IPV6 header length is **fixed**: 40 bytes.
- Therefore there is no header length field as in IPV4.
- In IPV4 the ip header is of **variable size**: 20 - 60 bytes; so we need the header length field. We can add to the base ip header by multiplications of 4 bytes up to 60 bytes.
 - Extension headers in ipv6.

IPV6 header – Hop Limit

- The hop limit is by default 64.
 - `IPV6_DEFAULT_HOPLIMIT` is 64.
- This is the parallel of ttl field in ip header.
- *ip6_forward()* checks the hop_limit ; when it reaches 0 ,
it sends an ICMP message:
 - (`ICMPV6_TIME_EXCEED`, `ICMPV6_EXC_HOPLIMIT...`)
- and the packet is dropped.
- **Note:** there is **NO checksum field** (as in IPV4).

Extension Headers

- Hop-by-hop options (IPPROTO_HOPOPTS)
- Routing packet header extension (IPPROTO_ROUTING)
- Fragment packet header extension (IPPROTO_FRAGMENT)
- ICMPV6 options (IPPROTO_ICMPV6)
- No next header (IPPROTO_NONE)
- Destination options (IPPROTO_DSTOPTS)
- Mobility options (IPPROTO_MH)
- Other Protocols (TCP,UDP,...)
 - See *include/linux/in6.h*
 - *There are some types of Next Headers which cannot have a Next Header field. For example, ICMPV6, TCP, UDP, no next header (IPPROTO_NONE).*

Extension Headers - contd.

- All these protocols are registered by *inet6_add_protocol()*
- If a host tries to parse an extension header which it does not recognize, then an ICMP error will be sent, notifying about a parameter problem. (type: *ICMPV6_PARAMPROB*, code: *ICMPV6_UNK_NEXTHDR*) **and the packet will be dropped.**
- For example, in *ip6_input_finish()* (in *net/ipv6/ip6_input.c*)
- *// next header does not specified a registered protocol*

...

```
icmpv6_send(skb, ICMPV6_PARAMPROB,  
            ICMPV6_UNK_NEXTHDR, nhoff, skb->dev);
```

Extension Headers - contd

- **Router Alert** is a subtype of Hop-by-hop option, and it tells the router to process the packet besides forwarding it. It is used in multicasting.

DHCPv6

- The DHCPv6 client runs on UDP port 546.
- The DHCPv6 server runs on UDP port 547.
- Projects:
- Dibbler:
 - <http://klub.com.pl/dhcpv6/>
 - Linux and windows
- <https://fedorahosted.org/dhcpv6/>
- Maintained by David Cantrell (Red Hat)
- No mailing list...

- WIDE-DHCPv6
 - originally developed in KAME project,
 - for BSD and Linux
- DHCPV6 clients send SOLICIT requests in order to find DHCPV6 servers.
- Hosts send DHCPV6 solicit messages are sent on to the all-DHCPv6 multicast address (FF02::1:2).
- DHCPv6 Servers reply with advertisements.

Socket API

- By default, the port space is not shared between IPV6 and IPV4.
- Simple example for creating TCP server with IPV6:

```
unsigned short port=9999;
```

```
struct sockaddr_in6 server;
```

```
struct sockaddr_in6 from;
```

```
sock = socket(AF_INET6, SOCK_STREAM, 0);
```

```
server.sin6_family = AF_INET6;
```

```
server.sin6_addr = in6addr_any;
```

Socket API - contd.

```
server.sin6_port = htons(port);  
bind(sock,(struct sockaddr*)&server, sizeof(server));  
fromlen = sizeof(from);  
if (listen(sock,5)<0)  
printf("error listening\n");  
while (1)  
{  
newsock = (int*)malloc(sizeof(int));  
*newsock = accept(sock, (struct sockaddr *)&from,&fromlen);
```

Socket API - contd.

- When trying to run a similar application in IPV4 on the same port simultaneously, you will get the following error:

binding socket error

bind

: Address already in use

- It will succeed if you set *IPV6_V6ONLY* option in IPV6 socket:

int on=1;

*if (setsockopt(sock, IPPROTO_IPV6, IPV6_V6ONLY,
(char *)&on, sizeof(on)) == -1)*

Receiving an IPv6 packet

- *ipv6_rcv()* is the handler for IPv6 packet (*net/ipv6/ip6_input.c*)
- Performs some sanity checks and then calls:
- *return NF_HOOK(PF_INET6, NF_IP6_PRE_ROUTING, skb, dev, NULL, ip6_rcv_finish);*
- *ip6_rcv_finish()* performs a lookup in the routing subsystem by calling *ip6_route_input(skb)* in order to construct *skb->dst*.

IPV6 – address types

- **Unicast**
 - The target is a single interface;
 - packet is delivered to a single interface.
- **Anycast** (new ! Does not exist in IPV4).
 - The target is a set of interfaces;
 - packet is delivered to a single interface.
- **Multicast**
 - The target is a set of interfaces;
 - packet is delivered to all the interfaces in this set.

- *ip6_mc_input()* currently only verifies that the packet is indeed for a multicast address of which the netdevice device is a member and calls *ip6_input()*
- The Multicast Routing patch (CONFIG_IPV6_MROUTE) adds a call to *ip6_mr_input()*.
 - net/ipv6/ip6_input.c
 - There is a user space daemon (pim6sd) which works in conjunction with multicast routing.
 - Configuration file: /usr/local/etc/pim6sd.conf
 - pim6sd is part of mcast-tools

MLD

- **MLD - Multicast Listener Discovery**
 - also known as **Multicast Group Management**.
- MLD is similar to IGMP in IPV4 but used ICMPv6 messages
- MLD messages are sent via ICMPV6.
- MLDV2: RFC 3810 (added filtering abilities).
- The MLD is used by routers to discover the presence of Multicast listeners.
- MLDV2 is based on IGMPv3.

MLD - contd.

- A host can belong to more than one multicast group.
- The Ethernet frame for a multicast address starts with 0x3333
- In IPV4, in multicast addresses, the first bit is 1 in the Ethernet frame (this is half of the MAC addresses!).
- The hop limit is always 1 in MLD messages so that a router will not forward them.
- `netstat -g -n` : show IPv6/IPv4 Group Memberships.
- `ip -6 maddr show`
- mcjoin: a util for joining an IPv6 Multicast Group
 - http://www.benedikt-stockebrand.net/hacks_e.html

MLD - contd.

- When a host boots, it first sends an MLDV2 message in ICMPV6. This is Type 143 message (ICMP code), and it is a Multicast Listener Discovery 2 **Report Message**. The report message tells routers and multicast-aware switches that the host wants to receive messages sent to the multicast address of the group it joined. This message has a hop limit of 1, so that it won't be forwarded outside. It is sent to **FF02::16** (A multicast address, which represents the all MLDv2-capable routers multicast group).

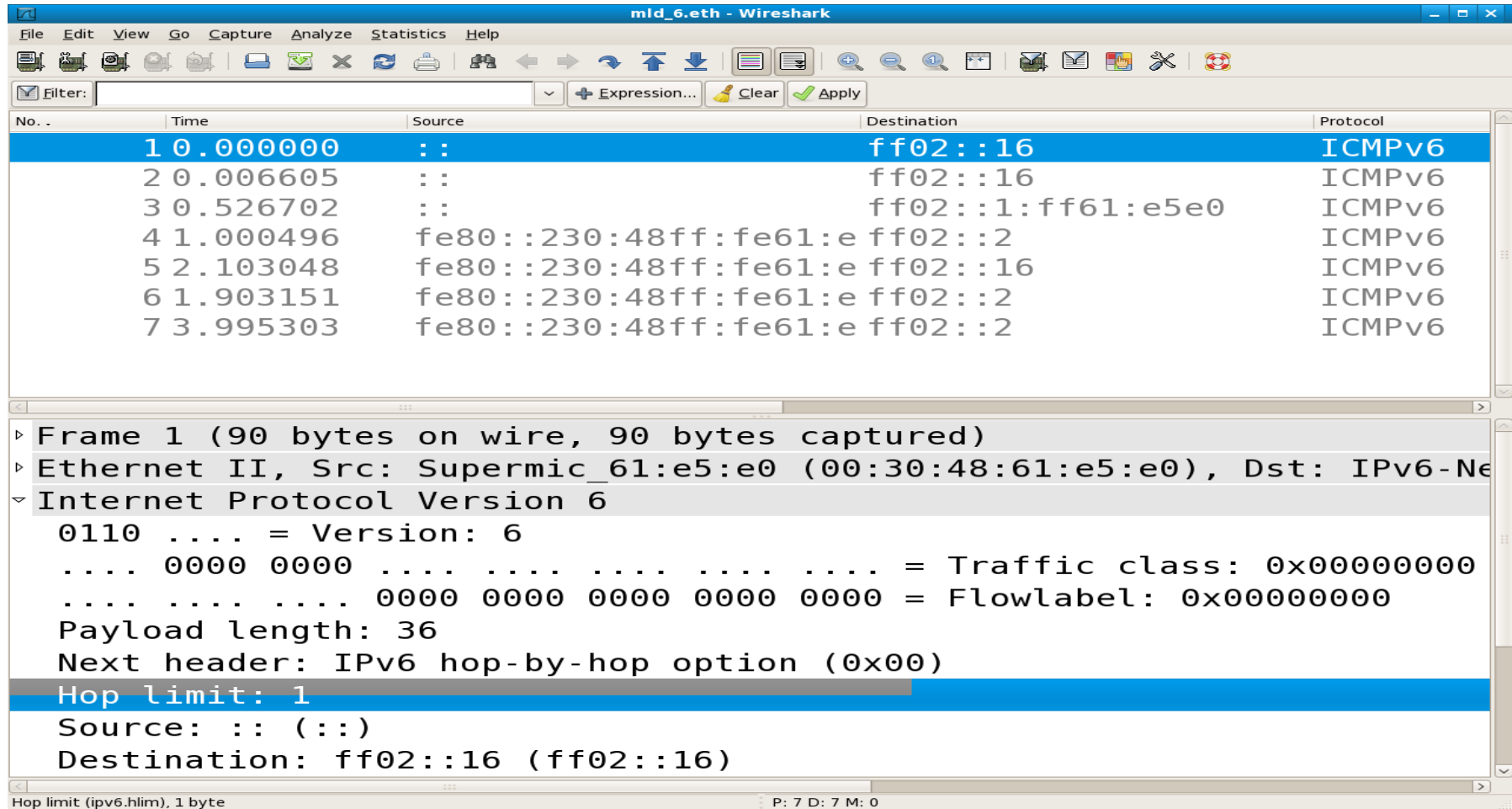
MLD - contd.

- *addrconf_add_linklocal()* calls *ipv6_add_addr()* which eventually calls *igmp6_group_added()*, *mld_newpack()* , setting type to **ICMPV6_MLD2_REPORT** and send an ICMP message.
- (*net/ipv6/mcast.c*)
- The source address of the MLD message can be a Link Local address or the unspecified address (::)
- When a host boots, it has a tentative address (until DAD is finished) and it sends an MLD report message to join the solicited node multicast group.
- *addrconf_join_solict()* calls *ipv6_dev_mc_inc()* *net/ipv6/addrconf.c*

MLD - contd.

- In this case , the source address of the MLD messages is the unspecified address (::)
- "Change to Exclude" in MLDV2 report.
- When a host leaves a group, it sends an MLDv2 ICMPV6_MGM_REDUCTION message
- *(igmp6_leave_group() in /net/ipv6/mcast.c)*
- This message is sent to the all routers multicast address (note the difference against REPORT, which is sent to FF02::16).

MLD sniff



The image shows a Wireshark capture window titled "mld_6.eth - Wireshark". The main display area shows a list of captured packets. The first packet is highlighted in blue. Below the list, the packet details pane is expanded to show the structure of the first packet, which is an ICMPv6 message. The details pane shows the following information:

- Frame 1 (90 bytes on wire, 90 bytes captured)
- Ethernet II, Src: Supermic_61:e5:e0 (00:30:48:61:e5:e0), Dst: IPv6-Ne
- Internet Protocol Version 6
 - 0110 = Version: 6
 - 0000 0000 = Traffic class: 0x00000000
 - 0000 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
 - Payload length: 36
 - Next header: IPv6 hop-by-hop option (0x00)
 - Hop limit: 1**
 - Source: :: (:::)
 - Destination: ff02::16 (ff02::16)

The status bar at the bottom of the window indicates "Hop limit (ipv6.hlim), 1 byte" and "P: 7 D: 7 M: 0".

No. -	Time	Source	Destination	Protocol
1	0.000000	::	ff02::16	ICMPv6
2	0.006605	::	ff02::16	ICMPv6
3	0.526702	::	ff02::1:ff61:e5e0	ICMPv6
4	1.000496	fe80::230:48ff:fe61:e	ff02::2	ICMPv6
5	2.103048	fe80::230:48ff:fe61:e	ff02::16	ICMPv6
6	1.903151	fe80::230:48ff:fe61:e	ff02::2	ICMPv6
7	3.995303	fe80::230:48ff:fe61:e	ff02::2	ICMPv6

MLD - contd.

- A router will recognize this MLD message by the hop-by-hop option in the extended header.
- *NEXTHDR_HOP* in *include/net/ipv6.h*
- *RFC 2711 - IPv6 Router Alert Option.*

MLD - contd.

- There are two types of messages in MLDV2:
 - Query (130) (ICMPV6_MGM_QUERY)
 - In icmp6 header, icmp6_type = 130
 - Report (143) ICMPV6_MLD2_REPORT
 - In icmp6 header, icmp6_type = 143
 - Reports are sent by MLDV2 with destination address of **FF02::16** (FF02:0:0:0:0:0:0:16)

Network Namespaces

- Two types of virtualization in the Linux Kernel
- OS virtualization.
- process/container virtualization. (Like solaris zones).
- OS virtualization:
 - Xen
 - Kvm (hardware virtualization)
 - Lguest (Only 32 bit; there is a RedHat trial to write 64 bit version).

Network Namespaces - contd.

- OpenVZ project (<http://openvz.org/>).
 - Currently only for Linux (the FreeBSD port was dropped).
 - There is a serious effort to integrate it into mainline Linux kernel.
- Many patches recently to netdev kernel mailing list.
- struct net (*include/net/net_namespace.h*)
- Adding support for namespaces in IPV4 is finished.
 - Currently work is being done on adding support for namespaces in IPV6.

Packet Generator

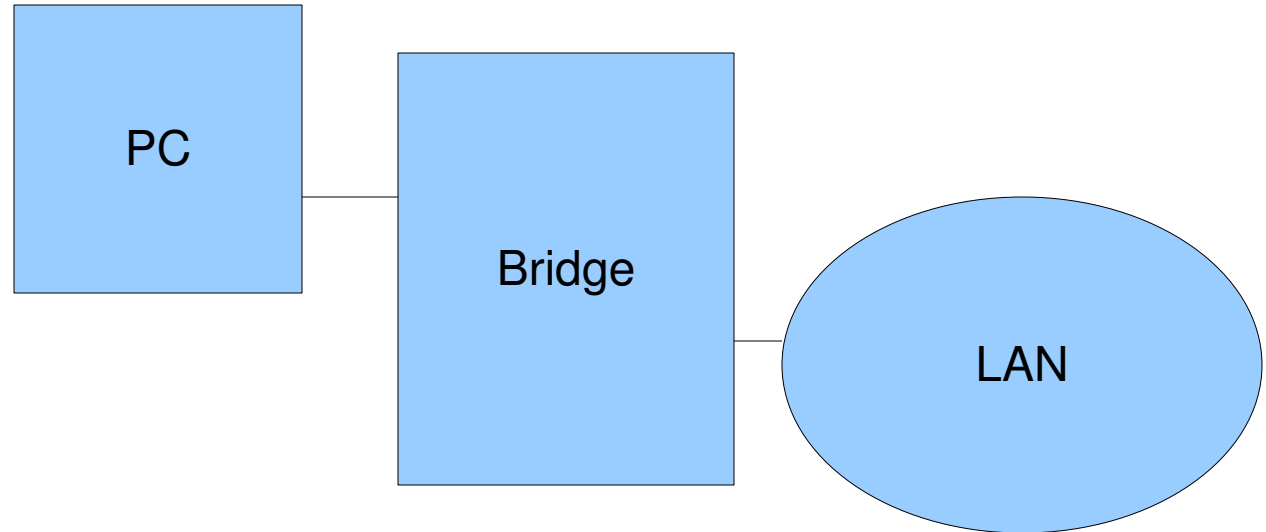
- Pktgen kernel module (Robert ollson)
 - Works also with IPV6.

Bridging Subsystem

- You can define a bridge and add NICs to it (“enslaving ports”) using `brctl` (from `bridge-utils`).
- You can have up to 1024 for every bridge device (`BR_MAX_PORTS`) .
- Example:
- `brctl addbr mybr`
- `brctl addif mybr eth0` #adding interface to a bridge
- `brctl show`

Simple example

- In this simple example, you can connect a PC to a bridge without any configuration on the PC.



Bridging Subsystem-contd

- There are devices which you cannot add to a bridge (by `addif`); like another bridge or a loopback device or a tunnel device or any other device which has no HW address.
- You can add a tap device (but not a tun device) (?)
 -
- When a NIC is configured as a bridge port, the `br_port` member of `net_device` is initialized.
- (`br_port` is an instance of struct `net_bridge_port`).
- When we receive a frame, `netif_receive_skb()` calls `handle_bridge()`.

Bridging Subsystem-contd

- *br_handle_frame()* is invoked (*net/bridge/br_input.c*)
- `NF_HOOK(PF_BRIDGE, NF_BR_PRE_ROUTING, skb, skb->dev, NULL, br_handle_frame_finish);`
- *br_handle_frame_finish()* checks the MAC destination of the packet.
 - If the packet is for the local machine, we do not forward the packet but call *br_pass_frame_up()*.
 - *br_pass_frame_up()* calls:
 - `NF_HOOK(PF_BRIDGE, NF_BR_LOCAL_IN, skb, indev, NULL, netif_receive_skb);`

Bridging Subsystem-contd

- If the packet is for the local machine we forward the packet:
 - by *br_forward()* if the address is in the forwarding DB.
 - *br_flood_forward()* if the address is in not the forwarding DB.

Bridging Subsystem-contd

- The bridging forwarding database is searched for the destination MAC address.
- In case of a hit, the frame is sent to the bridge port with *br_forward()* (net/bridge/br_forward.c).
- If there is a miss, the frame is flooded on all bridge ports using *br_flood()* (net/bridge/br_forward.c).
- Note: this is not a broadcast !
- The ebttables mechanism is the L2 parallel of L3 Netfilter.

Bridging Subsystem-contd

- Ebtables enable us to filter and mangle packets at the link layer (L2).

Tips for hacking

- Documentation/networking/ip-sysctl.txt: networking kernel tunables
- Example of reading a hex address:
- `iph->daddr == 0x0A00A8C0` or
means checking if the address is 192.168.0.10 (C0=192,A8=168, 00=0,0A=10).
- A BASH script for getting MAC address from IP address: (ipToHex.sh)

```
#!/bin/sh
```

```
IP_ADDR=$1
```

```
for I in $(echo ${IP_ADDR}| sed -e "s^./ /g"); do
```

```
    printf '%02X' $I
```

```
done
```

```
echo
```

```
usage example: ./ipToHex.sh 192.168.0.1 => C0A80001
```

Tips for hacking - Contd.

- Disable ping reply:
- `echo 1 >/proc/sys/net/ipv4/icmp_echo_ignore_all`
- Disable arp: ***ip link set eth0 arp off*** (the NOARP flag will be set)
- Also ***ifconfig eth0 -arp*** has the same effect.
- How can you get the Path MTU to a destination (PMTU)?
 - Use `tracert` (see `man tracert`).
 - `tracert` is from `iputils`.

Tips for hacking - Contd.

- **inet_addr_type()** method: returns the address type; the input to this method is the IP address. The return value can be RTN_LOCAL, RTN_UNICAST, RTN_BROADCAST, RTN_MULTICAST etc.
See: `net/ipv4/fib_frontend.c`

Tips for hacking - Contd.

- In case you want to send a packet from a user space application through a specified device without altering any routing tables:

```
struct ifreq interface;
```

```
strncpy(interface.ifr_ifrn.ifrn_name, "eth1",IFNAMSIZ);
```

```
if (setsockopt(s, SOL_SOCKET, SO_BINDTODEVICE, (char  
    *)&interface, sizeof(interface)) < 0)
```

```
{
```

```
    printf("error setting SO_BINDTODEVICE");
```

```
    exit(1);
```

```
}
```

Tips for hacking - Contd.

- Keep iphdr struct handy (printout): (from linux/ip.h)

```
struct iphdr {  
    __u8  ihl:4,  
    version:4;  
    __u8  tos;  
    __be16  tot_len;  
    __be16  id;  
    __be16  frag_off;  
    __u8  ttl;  
    __u8  protocol;  
    __sum16  check;  
    __be32  saddr;  
    __be32  daddr;  
    /*The options start here. */  
};
```

Tips for hacking - Contd.

- NIPQUAD() : macro for printing hex addresses
- Printing mac address (from net_device):

```
printk("sk_buff->dev =%02x:%02x:%02x:%02x:%02x:%02x\n",  
((skb)->dev)->dev_addr[0], ((skb)->dev)->dev_addr[1],  
((skb)->dev)->dev_addr[2],((skb)->dev)->dev_addr[3],  
((skb)->dev)->dev_addr[4], ((skb)->dev)->dev_addr[5]);
```

- Printing IP address (primary_key) of a neighbour (in hex format):

```
printk("neigh->primary_key =%02x.%02x.%02x.%02x\n",  
neigh->primary_key[0], neigh->primary_key[1],  
neigh->primary_key[2],neigh->primary_key[3]);
```

Tips for hacking - Contd.

- Or:

```
printk("***neigh->primary_key= %u.%u.%u.%u\n",  
        NIPQUAD(*(u32*)neigh->primary_key));
```

- CONFIG_NET_DMA is for TCP/IP offload.
- When you encounter: xfrm / CONFIG_XFRM this has to do with IPSEC. (transformers).

Tips for hacking - Contd.

- Showing arp statistics by:
- ***cat /proc/net/stat/arp_cache***

entries allocs destroys hash_grows lookups hits res_failed
rcv_probes_mcast rcv_probes_ucast periodic_gc_runs
forced_gc_runs

periodic_gc_runs: statistics of how many times the *neigh_periodic_timer()* is called.

Links and more info

- IPV6 howto (Peter Bieringer) :
<http://www.ibiblio.org/pub/Linux/docs/HOWTO/other-formats/pdf/Linux+IPv6-HOWTO.pdf>
- USAGI Project - Linux IPv6 Development Project
 - <http://www.linux-ipv6.org/>
- **Porting applications to IPv6 HowTo BY Eva M. Castro:**
 - <http://gsyc.es/~eva/IPv6-web/ipv6.html>
 -
- RFC 3493: Basic Socket Interface Extensions for IPv6.
- RFC 3542: Advanced Sockets Application Program Interface (API) for IPv6.

Links and more info

- **Books:**
- **IPv6 Essentials, Second Edition (OReilly)**
 - A book By Silvia Hagen
 - Second Edition May 2006
 - Pages: 436
 - ISBN 10: 0-596-10058-2 | ISBN 13: 9780596100582

Links and more info

- **IPv6 in Practice: A Unixer's Guide to the Next Generation Internet**
- by Benedikt Stockebrand (Author) ; Springer; 1 edition, 2006.
- Talks about implementation of IPv6 in Linux, Solaris, BSD.
- http://www.benedikt-stockebrand.net/books_e.html

Links and more info

- 1) **IPv6 Advanced Protocols Implementation (2007)**
- 2) **IPv6 Core Protocols Implementation (2006)**

Both books were written by Qing Li, Tatuya Jinmei and Keiichi Shima

- published by Morgan Kaufmann Series in Networking.
 - Both books discuss the **Kame** implementation of IPV6. (in BSD).

Links and more info

- IPv6 Information Page!
 - <http://www.ipv6.org/>
- What's up in the Linux IPv6 Stack
- Lecture slides by Hideaki YOSHIFUJI from Ica2008.
 - Keio University
 - USAGI/WIDE Project
- <http://mirror.linux.org.au/pub/linux.conf.au/2008/slides/131-200801-LCA2008-LinuxIPv6.pdf>
- Html: <http://www.linux-ipv6.org/materials/200801-LCA2008/>

Links and more info

Linux Network Stack Walkthrough (2.4.20):

http://gicl.cs.drexel.edu/people/sevy/network/Linux_network_stack_wa

Understanding the Linux Kernel, Second Edition

By Daniel P. Bovet, Marco Cesati

Second Edition December 2002

chapter 18: networking.

- Understanding Linux Network Internals, Christian benvenuti

Oreilly , First Edition.

Links and more info

Linux Device Driver, by Jonathan Corbet, Alessandro Rubini, Greg Kroah-Hartman

Third Edition February 2005.

- Chapter 17, Network Drivers

Linux networking: (a lot of docs about specific networking topics)

- http://www.linux-foundation.org/en/Net:Main_Page

–

netdev mailing list: <http://www.spinics.net/lists/netdev/>

Links and more info

Removal of multipath routing cache from kernel code:

<http://lists.openwall.net/netdev/2007/03/12/76>

<http://lwn.net/Articles/241465/>

Linux Advanced Routing & Traffic Control :

<http://lartc.org/>

ebtables – a filtering tool for a bridging:

<http://ebtables.sourceforge.net/>

Links and more info

Writing Network Device Driver for Linux: (article)

- <http://app.linux.org.mt/article/writing-netdrivers?locale=en>

Links and more info

Netconf – a yearly networking conference; first was in 2004.

- <http://vger.kernel.org/netconf2004.html>
- <http://vger.kernel.org/netconf2005.html>
- <http://vger.kernel.org/netconf2006.html>
- Next one: Linux Conf Australia, January 2008, Melbourne
- David S. Miller, James Morris , Rusty Russell , Jamal Hadi Salim , Stephen Hemminger , Harald Welte, Hideaki YOSHIFUJI, Herbert Xu , Thomas Graf , Robert Olsson , Arnaldo Carvalho de Melo and others

Links and more info

Policy Routing With Linux - Online Book Edition

- by Matthew G. Marsh (Sams).
- <http://www.policyrouting.org/PolicyRoutingBook/>

THRASH - A dynamic LC-trie and hash data structure:

Robert Olsson Stefan Nilsson, August 2006

<http://www.csc.kth.se/~snilsson/public/papers/trash/trash.pdf>

IPSec howto:

<http://www.ipsec-howto.org/t1.html>

Links and more info

Openswan: Building and Integrating Virtual Private Networks ,
by Paul Wouters, Ken Bantoft

<http://www.packtpub.com/book/openswan/mid/061205jqdnh2by>

publisher: Packt Publishing.

a book including chapters about LVS:

“The Linux Enterprise Cluster- Build a Highly Available Cluster
with Commodity Hardware and Free Software”, By Karl
Kopper.

<http://www.nostarch.com/frameset.php?startat=cluster>

<http://www.vyatta.com> - Open-Source Networking

Links and more info

Address Resolution Protocol (ARP)

– <http://linux-ip.net/html/ether-arp.html>

ARPWatch – a tool for monitor incoming ARP traffic.

Lawrence Berkeley National Laboratory -

<ftp://ftp.ee.lbl.gov/arpwatch.tar.gz>.

arptables:

<http://ebtables.sourceforge.net/download.html>

TCP/IP Illustrated, Volume 1: The Protocols

By W. Richard Stevens

<http://www.informit.com/store/product.aspx?isbn=0201633469>

Links and more info

Unix Network Programming, Volume 1: The Sockets Networking API (3rd Edition) (Addison-Wesley Professional Computing Series) (Hardcover)

by W. Richard Stevens (Author), Bill Fenner (Author), Andrew M. Rudoff (Author)

Linux Ethernet Bridging mailing list:

<http://www.spinics.net/lists/linux-ethernet-bridging/>

Questions

- Questions ?
- Thank You !