IPv6 Install-Fest at NovaLUG

2010-Aug-14

Table of Contents

Table of Contents (cont)

Acronyms

6in4	IPv6 packets encapsulated in IPv6 packets
AAAA	Quad-A DNS Record
AfriNIC	African Network Information Centre
APNIC	Asia-Pacific Network Information Centre
ARIN	American Registry for Internet Numbers
ARP	Address Resolution Protocol
ARPA	Advanced Research Programs Agency
ARPANET	ARPA Network
AS	Autonomous System
ASN	Autonomous System Number
AYIYA	Anything In Anything Tunneling Protocol
BBN	Bolt Beranek and Newman Inc.
BGP	Border Gateway Protocol
CIDR	Classless Inter-Domain Routing
DARPA	Defense Advanced Research Programs Agency

Acronyms (cont)

DHCP	Dynamic Host Control Protocol
DNS	Domain Name Server
EUI-48	48-bit Extended Unique Identifier
GiB	GibiByte ($2^3$0 Bytes)
HTTP	Hyper-Text Transport Protocol
IANA	Internet Assigned Numbers Authority
ICANN	Internet Corporation for Assigned Names and Numbers
ICMP	Internet Control Message Protocol
IEEE	Institute of Electrical and Electronic Engineers
IETF	Internet Engineering Task Force
IMP	Interface Message Processor
IP	Internet Protocol
IPng	IP: Next Generation
IPv4	IP version 4
IPv6	IP version 6

Acronyms (cont)

ISP	Internet Service Provider
LACNIC	Latin American and Caribbean Internet Addresses Registry
LAN	Local Area Network
LIR	Local IP Registry
MAC	Media Access Control
MIPv6	Mobile IPv6
MTA	Mail Transport Agent
MTU	Maximum Tranmission Unit
NAT	Network Address Translation
NCP	Network Control Program
NIR	National IP Registry
NS	Name Server
NTP	Network Time Protocol
PC	Personal Computer
PMTU	Path MTU

Acronyms (cont)

rDNS	Reverse DNS
RFC	Request For Comments
RIPE NCC	Réseaux IP Européens Network Coordination Centre
RIR	Regional IP Registry
SMTP	Simple Mail Transport Protocol
SSH	Secure Shell
TCP	Transmission Control Protocol
TLD	Top Level Domain
UDP	User Datagram Protocol
ULA	Unique Local Address
URL	Uniform Resource Locator
VLSM	Variable Length Subnet Mask
VPN	Virtual Private Network

Introduction

Early packet switched networks had very small address spaces, and had to upgrade several times. From 1979 to end of 1982, IPv4 coexisted with its predecessor NCP. Now IPv4 is approaching exhaustion, and coexisting with its successor IPv6.

This lecture presents some internet history, and the basics of IPv6.

Then follows a lab, where we try to get everyone's laptop IPv6 ready. We should then be able to access IPv6 ready web-sites, and see the dancing kame (Japanese turtle) at <http://www.kame.net/>, and perform our web searchs at <http://ipv6.google.com/>

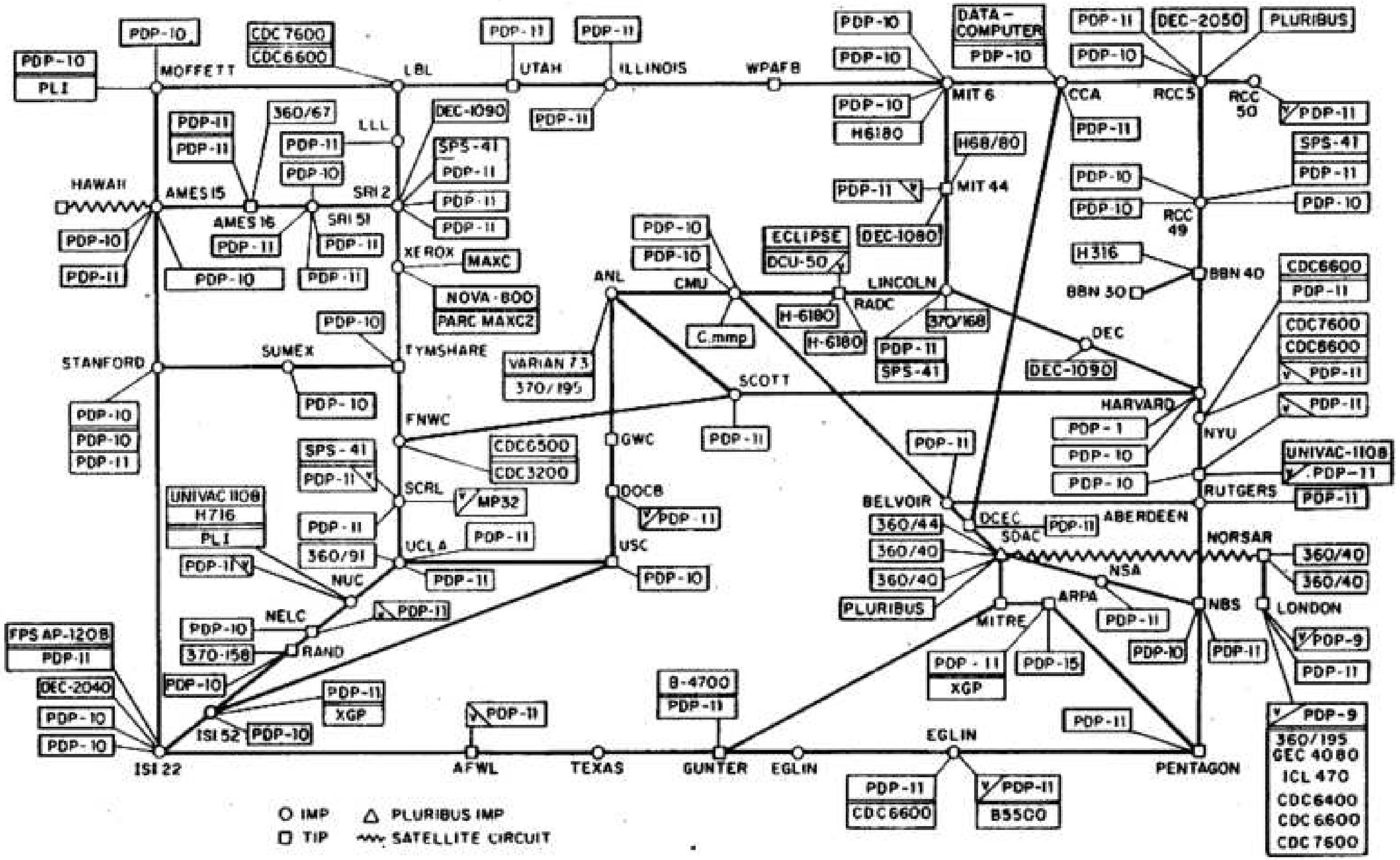
Finally, there will be info on how to get certification.

1. Lecture: History
1.1 The Road to IPv4

Early packet switched networks used small address spaces

1961	First mention of Packet Switched Network	RM3420
1969	ARPANET uses Network Control Program (NCP)	
	(6-bit IMP gateway, 2-bit host)	BBN 1822
1970	NCP standardized and deployed	RFC 33
	(16-bit IMP gateway, 8-bit host)	BBN 1822
1973	DARPA begins Internetting Project	
1974	Protocol for Packet Network Intercommunication	CERF74
	(8-bit network, 16-bit host)	
1977	Vinton Cerf picks 32-bit address for IP	
	(8-bit network, 24-bit host)	
	TCPv0, TCPv1, TCPv2	
1978	Split into TCPv3 and IPv3, and UDP defined	CERF93
	due to problems with voice transmission	
	where delay is worse than packet loss	

ARPANET LOGICAL MAP, MARCH 1977



(PLEASE NOTE THAT WHILE THIS MAP SHOWS THE HOST POPULATION OF THE NETWORK ACCORDING TO THE BEST INFORMATION OBTAINABLE, NO CLAIM CAN BE MADE FOR ITS ACCURACY)

NAMES SHOWN ARE IMP NAMES, NOT (NECESSARILY) HOST NAMES

1. Lecture: History
1.2 IPv4 Evolution

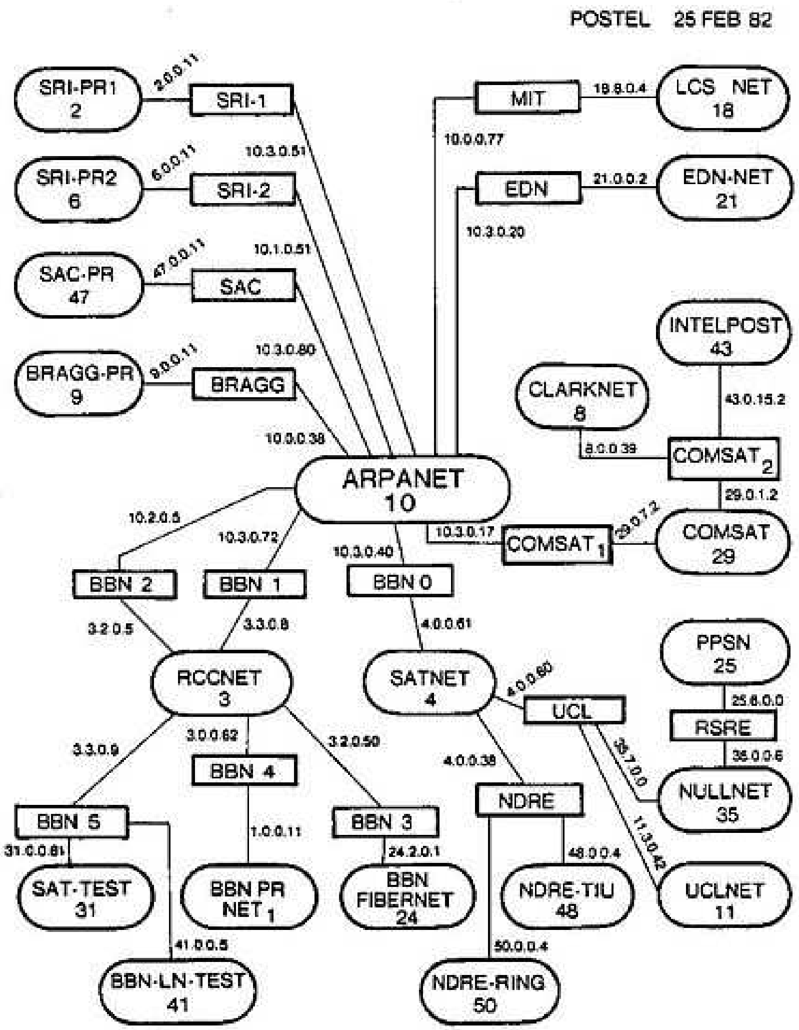
Rapid growth of the Internet led to: growing pains, especially, growth of routing tables and IPv4 address exhaustion.

Internet Engineering Task Force (IETF) bought time by adapting IPv4.

1979	TCPv4 and IPv4 stabilized	
1980	IPv4 address documented (8-bit network prefix)	RFC 760
1981	Classful network addressing defined (8,16,24-bit)	RFC 790
1981	IPv4 documented	RFC 791
1985	Subnetting defined aka Variable Length Subnet mask (VLSM) subnet	RFC 950
1993	Classless Inter-Domain Routing (CIDR) defined for super-netting to allow route aggregation	RFC 1518
1996	Private address spaces defined used with Network Address Translation (NAT)	RFC 1918

However, today IPv4 32-bit address space is nearly exhausted.

1. Lecture: History
1.1 IPv4 Evolution (cont)



1. Lecture: History
1.3 IPv6 Clean Design

IETF also began working on a clean design.

1992	IP: Next Generation whitepapers solicited	RFC 1550
1994	IPng adopted, working groups formed	
1995	IPv6 specification released	RFC 1883
1998	IPv6 standardized	RFC 2460

IPv6 offers: a large (128-bit) and redesigned address space, new services, and solves many old problems.

However, IPv6 deployment has been a challenge.

2. Lecture: IPv4 Address Exhaustion
2.1 IPv4 Allocation Process

IPv4 Addresses are allocated in blocks:

ICANN → IANA → RIRs → LIRs, NIRs, ISPs → End Users.

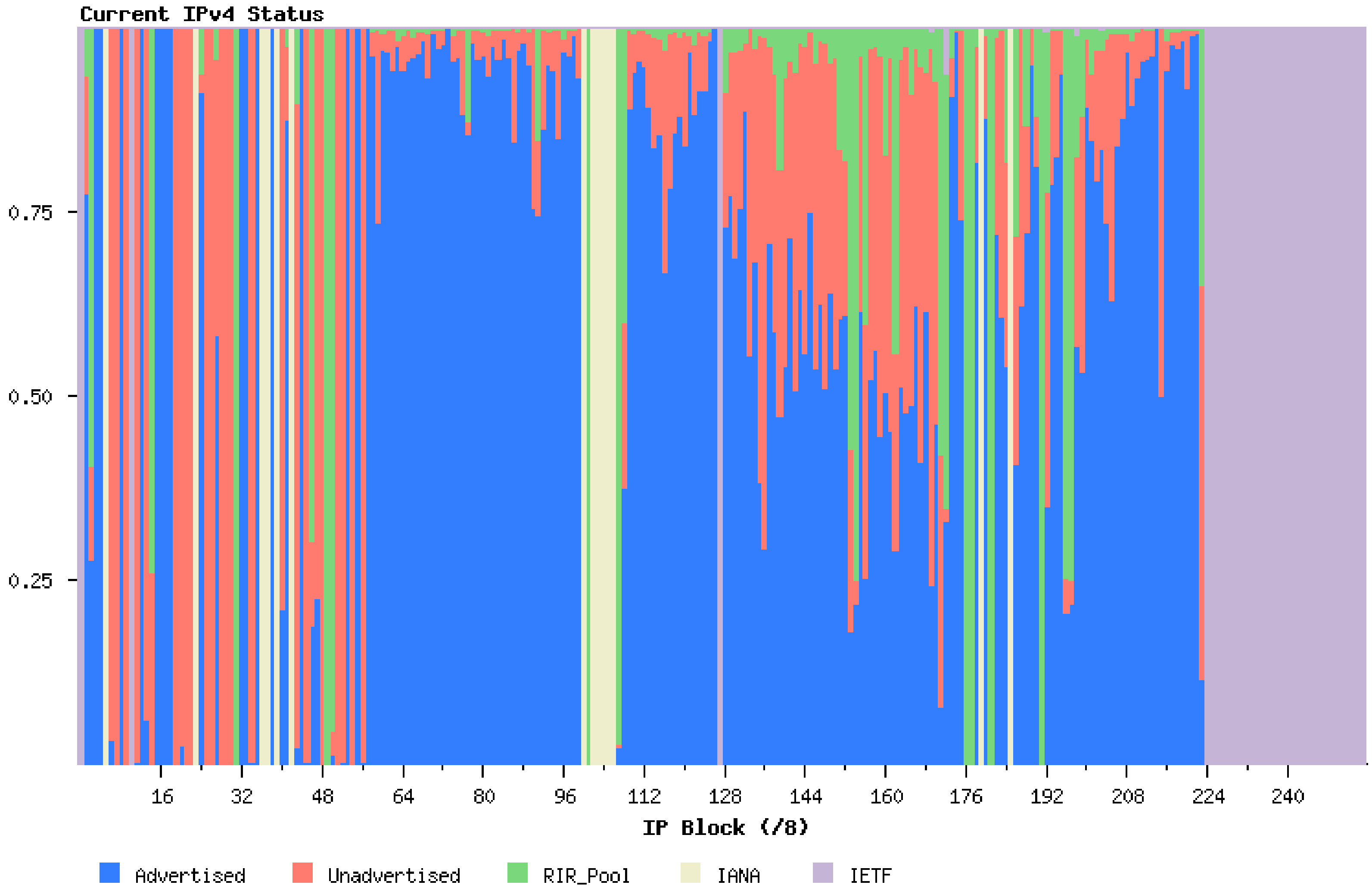
Internet Corporation for Assigned Names and Numbers (ICANN) allocates /8 blocks of over 16 million addresses to IANA.

Internet Assigned Numbers Authority (IANA) allocates /8 blocks to RIRs.

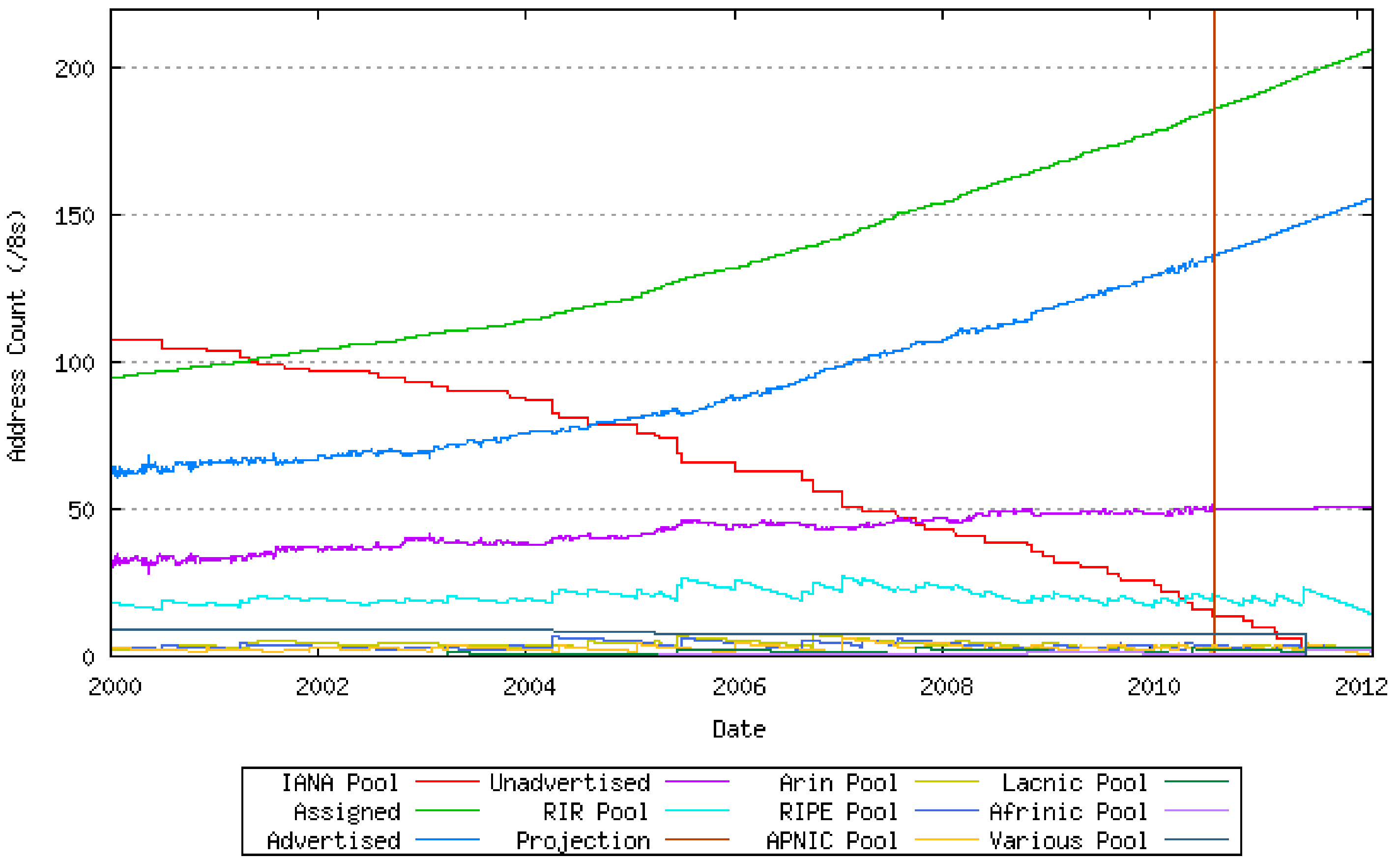
Regional IP Registries (RIRs): ARIN, RIPE NCC, APNIC, LACNIC, and AfriNIC; allocate smaller IPv4 address ranges to LIRs and ISPs.

Local IP Registries (LIRs) and Internet Service Providers (ISPs) allocate IPv4 addresses to End Users

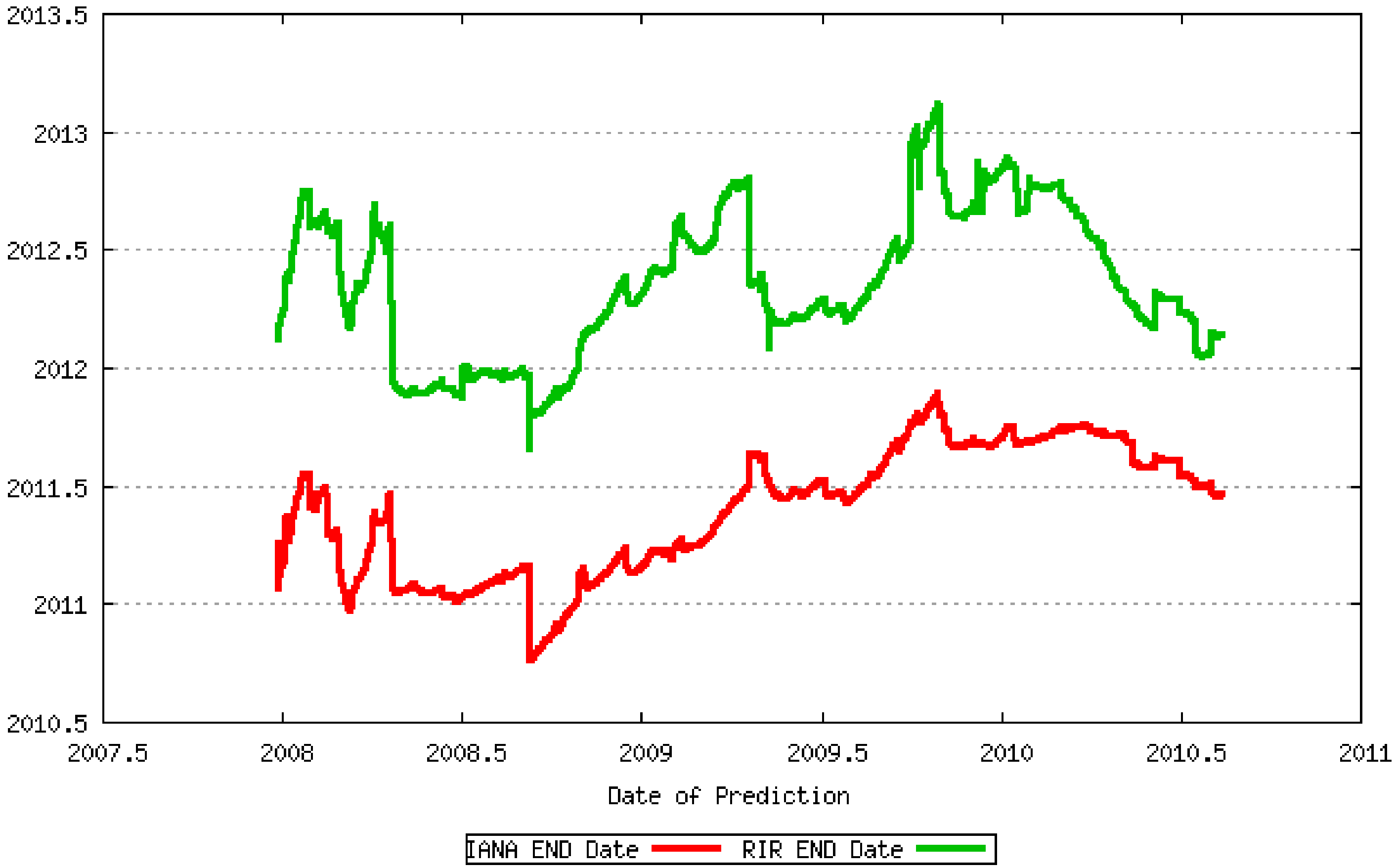
2. Lecture: IPv4 Address Exhaustion
2.2 IPv4 Status on 2010-08-14



2. Lecture: IPv4 Address Exhaustion
2.3 IPv4 Pool for IANA and RIRs over Time



2.4 IPv4 Predicted End Date for IANA and RIRs



3. Lecture: ASN Address Exhaustion
3.1 ASN Allocation Process

The Internet is comprised of networks connected by gateways.

Gateways communicate with each other using the Border Gateway Protocol (BGP). BGP determines the routing of IP traffic.

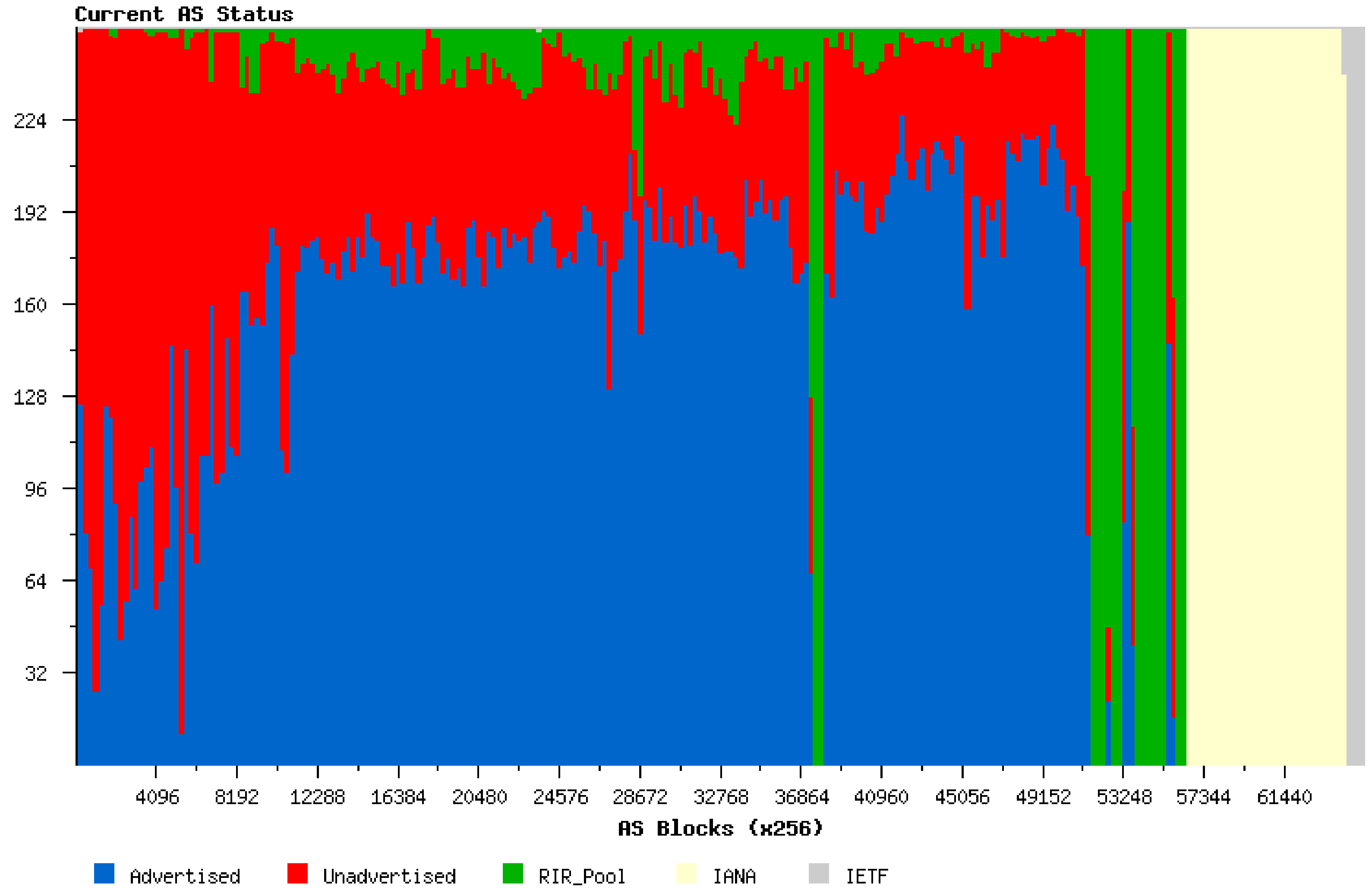
Each network, Autonomous System (AS), is assigned a unique Autonomous System Number (ASN) for use in BGP routing. For example, each ISP must have an officially registered ASN.

ASNs are allocated in blocks: IETF → IANA → RIR → AS operator.

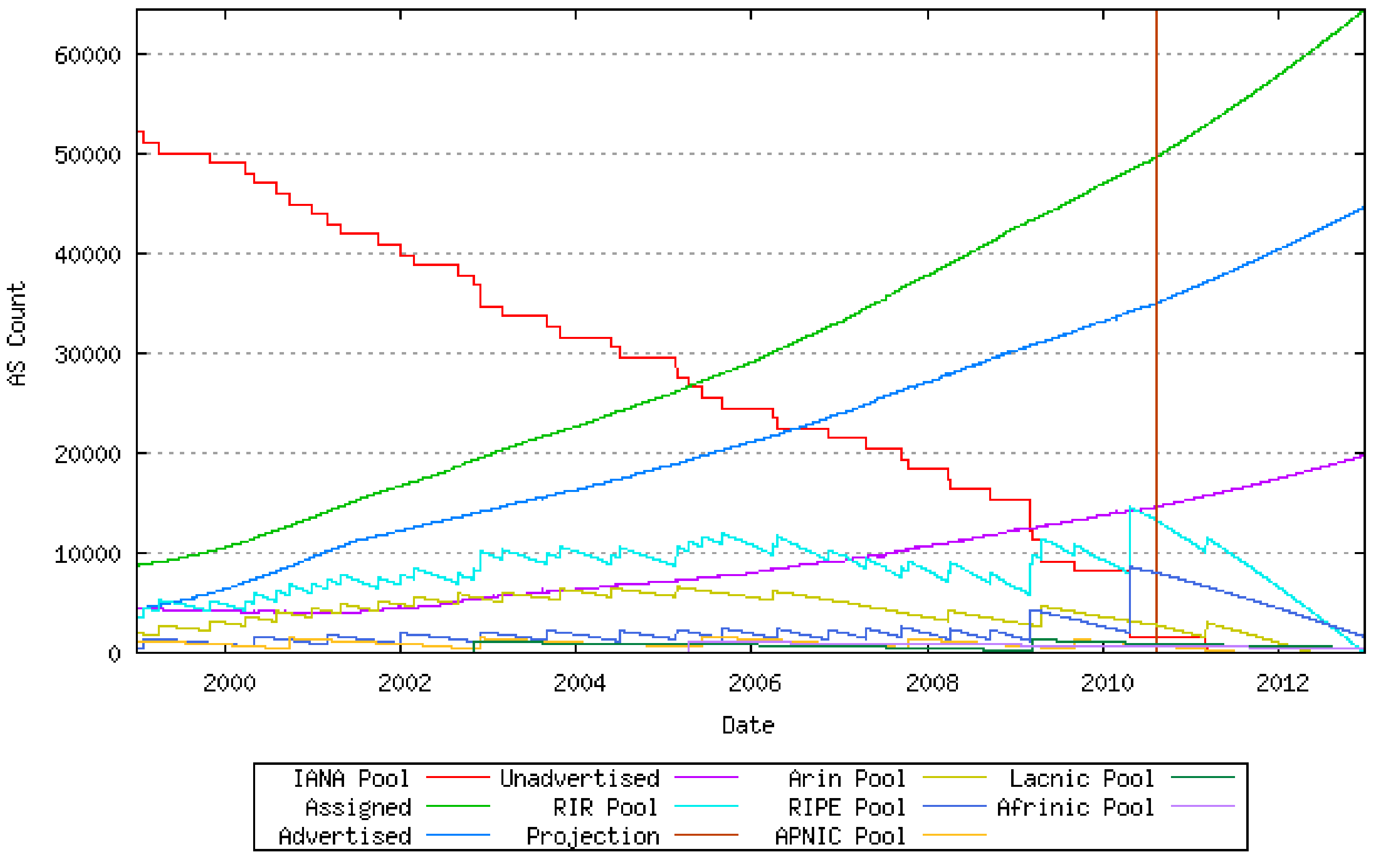
ASNs are allocated in sequence. 16-bit ASNs have run out. Allocation of 32 ASNs has begun. Conversion to 32-bit ASNs turned out to be easy.

Exponential growth of BGP routing tables is a problem.

3. Lecture: ASN Address Exhaustion
3.2 ASN Status on 2010-08-014



3. Lecture: ASN Address Exhaustion
3.3 ASN Pool for IANA and RIRs over Time



4. Lecture: IPv6 Advantages

4.1 Offers New Services and Solves Old Problems

Larger and redesigned address architecture (RFC 3513, RFC 3587):

- Address of all types are assigned to interfaces, not nodes.
- 128-bit (typically 48-bit network, 16-bit subnet, 64-bit interface).
- Avoids need for Network Address Translation (NAT).
- Avoids need for complex subnetting schemes.
- Improved routing.
- Special purpose address ranges.

Stateless address autoconfiguration:

- Increased role for ICMPv6
  - E.g. replaces Address Resolution Protocol (ARP).
- Hosts autoconfigured when connected to routed IPv6 network (ICMPv6)
  - Host sends a link-local multicast router solicitation packet, and routers respond with a router advertisement packet.
- Hosts may still use stateful configuration (DHCPv6) or manual.
- Routers need manual configuration.

4. Lecture: IPv6 Advantages
4.1 Offers New Services and Solves Old Problems (cont)

Multicast:

Multicast addresses (RFC 4291).

No broadcast addresses (function superceded by multicast).

Link-local addresses:

Interfaces (usually) have multiple IPv6 addresses.

Link-local addresses are generated from MAC addresses (never change).

Jumbograms

Jumbograms can be as large as 4GiB, for improved performance over high Maximum Transportation Unit (MTU) networks.

Hosts use Path MTU discovery (ICMPv6).

4. Lecture: IPv6 Advantages

4.1 Offers New Services and Solves Old Problems (cont)

Network-layer security

- IPsec for encryption and authentication is integral to IPv6 suite.

Mobility

- Mobile IPv6 (MIPv6) avoids triangular routing.
- Routing is as efficient as for IPv6.

Simpler processing by routers

- Simplified header. Seldom used fields moved to options header.
- Uses hop-limit instead of time-to-live.
- No error checking at IPv6 layer. Relies instead on link-layer and transport-layer for error checking. So router does not recompute checksum when hop-limit changes.
- Routers never fragment packets. Hosts use Path MTU discovery.

5. Lecture: IPv6 Address Notation and Architecture
5.1 Notation

IPv6 address are 128 bits long (RFC 4291):

- 64-bit network prefix (or 48-bit network, 16-bit sub-net),
- 64-bit interface ID (often auto-generated from interface MAC address).

Written as eight groups of hexadecimal digits separated by a colon (:).

- One contiguous group of 0000s may be replaced with two colons (::).
  - 2001:0db8:0000:0000:0000:0000:1234:5678
  - 2001:0db8::0000:0000:0000:1234:5678
  - 2001:0db8::0000:1234:5678
  - 2001:0db8::1234:5678
  - 2001:db8::1234:5678 (leading zeros may be omitted)
- For URLs, enclose with square brackets (RFC 2732, RFC 3986).
  - https://[2001:db8::1234:5678]:443/
- IPv4 addresses may use dot notation.
  - ::ffff:12.34.56.78 (or ::ffff:0c22:384e)

Networks written using CIDR notation (ipv6-address/prefix-length)

- 2001:0db8:1234::/48

5. Lecture: IPv6 Address Notation and Architecture
5.2 Address Architecture

```
Unicast Addresses
  1-to-1 network address to network endpoint (interface, not node).
    Global unicast addresses - unique globally routable address.
    Link-local addresses     - used for interfaces. never leaked by route
    Site-local addresses     - used for private networks and VPNs.
      aka Unique Local  Addresses
    Special Addresses        - see next slide.
Multicast Addresses
  1-to-many network address to network endpoint.
  Delivered to all endpoints.
  Useful for router and neighbor discovery, advertising of services.
Anycast Addresses
  1-to-many network address to network endpoint.
  Delivered to ``nearest'' endpoint.
  Useful for load balancing, failover, internationally distributed server
```

5. Lecture: IPv6 Address Notation and Architecture
5.3 Special Addresses

```
Link local
  ::/128       - the ``unspecified address'', only to be used as source
                 address before initializing host has learned its
                 address. Never used as a destination address.
  ::1/128      - loopback / localhost address.  Never sent outside host.
                 Never forwarded by router.  Always dropped if received.
  fe80::/64    - prefix for addresses valid only on local physical link.
                 Never forwarded by router (no leakage).
Site local
  fc00::/7     - prefix used for Unique Local Addresses (ULA),
                 not globally routable (RFC 4193).
                 Split into two ranges:
    fc00::/8   - to be managed by ``ULA-Central'' (never created!)
                 (SIXXS created a voluntary database for ULAs).
    fd00::/8   - allocated by appending a random 40-bit string
                 to derive a valid /48.
```

5. Lecture: IPv6 Address Notation and Architecture
5.3 Special Addresses (cont)

```
Multicast
  ff00::/8    - prefix used for multicast addresses.
                No broadcast, the function is superceded by multicast.
              - Example, the all-nodes ff02::1 multicast address,
                first and second hex - ff - means multicast,
                third hex            -  0 - means permanent from IANA,
                fourth hex           -  2 - means link-level scope.
Anycast
  <64-bit-sub-network-prefix>:feff:ffff:ffff:8000:/121
              - prefix used for anycast addresses (RFC 2526)
                (128 addresses per subnet)
IPv4
  ::ffff:0:0/96 - prefix used for IPv4 mapped addresses.
  2002::/16     - prefix used for 6to4 addressing (RFC 3056).
```

5. Lecture: IPv6 Address Notation and Architecture

5.3 Special Addresses (cont)

```
Teredo
  2001::/32     - prefix used for Teredo tunneling addresses (RFC 4380).
Documentation
  2001:db8::/32 - prefix used for documentation (RFC 3849).
                  Very important.  Beware the newbie, for
                  he will copy-and-paste your example.
Deprecated
  ::/96         - zero prefix used for IPv4 compatible addresses,
                  deprecated (RFC 4291).
  fec0::/10     - site-local prefix, deprecated (RFC 3879).
```

6. Lab: IPv6 Ready System Check
6.1 Kernel, Module, Utilities

Test the kernel:

```
# test -f /proc/net/if_inet6 && echo ‘‘Kernel is IPv6 ready.’’
```

Test the module (if not compiled into the kernel):

```
# lsmod | grep -w ’ipv6’ && echo ‘‘Module ’ipv6’ is loaded.’’
```

Test traditional utilities (net-tools)

```
# /sbin/ifconfig -? 2>& 1 | grep -qw ’inet6’ && \
> echo ‘‘Utility ’ifconfig’ is IPv6 ready.’’
# /sbin/route -? 2>& 1 | grep -qw ’inet6’ && \
> echo ‘‘Utility ’route’ is IPv6 ready.’’
```

Test preferred utilities (iproute aka iproute2)

```
# /sbin/ip 2>& 1 | grep -qw ’inet6’ && \
> echo ‘‘Utility ’ip’ is IPv6 ready.’’
```

6. Lab: IPv6 Ready System Check
6.1 ping6, traceroute6, tracepath6

```
Test ping6 (iputils-ping)
# ping6 -c2 ::1
Note: if you ping6 a link-local address, you must give the interface
# ifconfig | grep inet6
# ping6 -I eth0 -c2 fe80::xxxx:xxxx:xxxx:xxxx

Test traceroute6 (traceroute)
# traceroute6 ::1
Test tracepath6 (iputils-tracepath)
# tracepath6 ::1
```

6. Lab: IPv6 Ready System Check
6.1 tcpdump, netstat

Test tcpdump (tcpdump): in one terminal box enter

```
# tcpdump -t -n -i lo -s 512 -vv ip6 or proto ipv6
```

then in another terminal box enter

```
# ping6 -c2 ::1
```

Some services are probably already listening on IPv6 addresses:ports

```
# netstat -nltup
```

7. Lab: Configuring the Packet Filter
7.1 Netfilter Configuration

```
IPTABLES=‘‘/sbin/ip6tables/’’

# Reset
$IPTABLES -F     # Flush  all rules in all chains in the filter table
$IPTABLES -X     # Delete all user-defined chains in the filter table
$IPTABLES -Z     # Zero  all packet/byte counters in the filter table

# Establish policy
$IPTABLES -P INPUT   DROP      # Block all inbound traffic
$IPTABLES -P FORWARD DROP      # A laptop should never forward a packet
$IPTABLES -P OUTPUT  DROP      # Block all outbound traffic
$IPTABLES -N in-new
```

7. Lab: Configuring the Packet Filter
7.1 Netfilter Configuration (cont)

```
# INPUT chain
$IPTABLES -A INPUT -i lo -j ACCEPT
$IPTABLES -A INPUT -m rt --rt-type 0 -j REJECT \
  --reject-with icmp6-port-unreachable
$IPTABLES -A INPUT -p ipv6-icmp -j ACCEPT
$IPTABLES -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
$IPTABLES -A INPUT -m state --state INVALID -j DROP
$IPTABLES -A INPUT -m state --state NEW -j in-new
$IPTABLES -A in-new -p tcp -m tcp --dport 22 --syn -j ACCEPT  # SSH
$IPTABLES -A INPUT -m limit --limit 3/min --limit-burst 10 -j LOG \
  --log-prefix "IPv6 input packet died: : "
```

7. Lab: Configuring the Packet Filter
7.1 Netfilter Configuration (cont)

```
# OUTPUT chain
$IPTABLES -A OUTPUT -m rt --rt-type 0 -j REJECT \
  --reject-with icmp6-port-unreachable
$IPTABLES -A OUTPUT -j ACCEPT

# FORWARD chain
$IPTABLES -A FORWARD -m rt --rt-type 0 -j REJECT \
  --reject-with icmp6-port-unreachable
$IPTABLES -A FORWARD -j REJECT
```

8. Lab: Setting Kernel Parameters
8.1 Sysctl (cont)

1) Edit /etc/sysctl.conf to set kernel parameters
   a) For a PC with static address, you should block router advertisements and prevent forwarding, by adding lines:

```
net.ipv6.conf.default.autoconf = 0
net.ipv6.conf.default.accept_ra = 0
net.ipv6.conf.default.accept_ra_defrtr = 0
net.ipv6.conf.default.accept_ra_rtr_pref = 0
net.ipv6.conf.default.accept_ra_pinfo = 0
net.ipv6.conf.default.accept_source_route = 0
net.ipv6.conf.default.accept_redirects = 0
net.ipv6.conf.default.forwarding = 0
```

   and

8. Lab: Setting Kernel Parameters
8.1 Sysctl (cont)

1) Edit /etc/sysctl.conf to set kernel parameters (cont)

a) For a PC with static address (cont):

```
net.ipv6.conf.all.autoconf = 0
net.ipv6.conf.all.accept_ra = 0
net.ipv6.conf.all.accept_ra_defrtr = 0
net.ipv6.conf.all.accept_ra_rtr_pref = 0
net.ipv6.conf.all.accept_ra_pinfo = 0
net.ipv6.conf.all.accept_source_route = 0
net.ipv6.conf.all.accept_redirects = 0
net.ipv6.conf.all.forwarding = 0
```

Also uncomment the following lines:

```
net.ipv6.conf.all.accept_redirects=0
net.ipv6.conf.all.accept_source_route=0
```

2) Reload kernel parameters

```
# sysctl -p
```

8. Lab: Setting Kernel Parameters
8.1 Sysctl

1) Edit /etc/sysctl.conf to set kernel parameters

   b) For a laptop with dynamic address, (i.e. intended to accept router advertisements), the above *.ipaccept_ra_* parameters should be set to one (1).

   c) For a router, uncomment

```
net.ipv6.conf.all.forwarding=1
```

2) Reload kernel parameters

```
# sysctl -p
```

9. Lab: Configuring the Interface
9.1 /etc/network/interfaces

For interfaces with dynamic addresses, edit /etc/network/interfaces

```
# The loopback network interface
auto lo
iface lo inet loopback

iface eth0 inet6 static
     # Get address from Router Advertisement Daemon (radvd) on router.
     up   ip link set dev eth0 up
     down ip link set dev eth0 down
     netmask 64
     # Set this as high as you can without generating the error:
     #   # SIOCSIFMTU: Invalid argument
     mtu 1500
     # If DNS fails to autoconfigure (zeroconf), then uncomment
     #dns-nameservers 2001:db8::3
```

9. Lab: Configuring the Interface
9.1 /etc/network/interfaces (cont)

For interfaces with static addresses, edit /etc/network/interfaces

```
# The loopback network interface
auto lo
iface lo inet loopback

iface eth0 inet6 static
     # Get address from Router Advertisement Daemon (radvd) on router.
     address 2001:db8::1
     netmask 64
     # Set this as high as you can without generating the error:
     #   # SIOCSIFMTU: Invalid argument
     mtu 1500
     # If DNS fails to autoconfigure (zeroconf), then uncomment
     #dns-nameservers 2001:db8::3
```

9. Lab: Configuring the Interface
9.1 /etc/network/interfaces (cont)

For a router, add the following to /etc/network/interfaces

```
# Tunnel 6in4 (sixxs)
iface sixxs inet manual
        #endpoint 11.222.33.444   # tunnel-server
        #address  2001:db8::2     # tunnel-my-end
        #netmask  64
        #gateway  2001:db8::1     # tunnel-far-end
        #ttl 64
        # Note: replaced /etc/rc[2345].d/S20aiccu with K20aiccu
        #       to prevent automatic start during runlevel changes
        pre-up    /etc/init.d/aiccu start
        up        sleep 1
        # Path MTU discovery can fail in a tunnel. Use minimum (1280).
        up        ip link set mtu 1280 dev $IFACE
        down      ip link set mtu 1500 dev $IFACE
        post-down /etc/init.d/aiccu stop
```

10. Lab: Discovering Links, Addresses, and Neighbors
10.1 Discovering Links

Connect your PC to a LAN (e.g. via a switch or wireless access point).

Use “ip -6” as it is more useful than “ifconfig”.

Find the interfaces and their MAC addresses.

```
$ ip -6 help

$ ip -6 link help
$ ip -6 link show dev lo
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
$ ip -6 link show dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
state UP qlen 1000
    link/ether 00:cc:cc:aa:aa:aa brd ff:ff:ff:ff:ff:ff
```

10. Lab: Discovering Links, Addresses, and Neighbors
10.2 Discovering Addresses

Every interface is required to have one link-local unicast address.

Each interface may have more than one address of any type (unicast, multicast, anycast) or scope (host, link, site, global).

Find the “scope host” and “scope link” addresses.

```
$ ip -6 addr help
$ ip -6 addr show dev lo
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
$ ip -6 addr show dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qlen 1000
    inet6 fe80::2cc:ccff:feaa:aaaa/64 scope link
       valid_lft forever preferred_lft forever
```

10.3 How Link-Level Addresses Are Generated

The link-local address of an interface is generated automatically from its "MAC Address".

```
1) Start with the EUI-48 address, which is formatted (in bits)

|0       0 0       1|1       2 2       3|3       3 4       4|
|0       7 8       5|6       3 4       1|2       9 0       7|
+--------+--------+--------+--------+--------+--------+
|ccccccug|cccccccc|cccccccc|mmmmmmmm|mmmmmmmm|mmmmmmmm|
+--------+--------+--------+--------+--------+--------+

where ``c'' are the bits of the company ID,
      ``u'' is the universal/local bit,
      ``g'' is the individual/group bit, and
      ``m'' is the manufacturer selected extension.
```

10. Lab: Discovering Links, Addresses, and Neighbors
10.3 How Link-Level Addresses Are Generated (cont)

2) Flip the ‘‘u’’ bit, and insert ‘‘fffe’’ after the company ID.

```
|0      0 0      1|1      2 2      3|3      3 4      4|4      5 5      6|
|0      7 8      5|6      3 4      1|2      9 0      7|8      5 6      3|
+--------+--------+--------+--------+--------+--------+--------+--------+
|ccccccUg|cccccccc|cccccccc|11111111|11111110|mmmmmmmm|mmmmmmmm|mmmmmmmm|
+--------+--------+--------+--------+--------+--------+--------+--------+
```

3) Finally, prepend ‘‘fe80::’’, thus

```
EUI-48 address       00:11:22:33:44:55, becomes
link-local address   fe80::0211:22ff:fe33:4455
```

10. Lab: Discovering Links, Addresses, and Neighbors
10.4 Discovering Neighbors

Each PC on your LAN has an interface with a link-local address.

```
$ ip -6 neigh help
$ ip -6 neigh
$ ip -6 neigh show dev eth0
fe80::02cc:ccff:fecc:cccc lladdr 00:cc:cc:cc:cc:cc router STALE

$ ping6 -I eth0 -c2 ff02::1    # all-nodes link-local multicast address
Ping Ff02::1(Ff02::1) from fe80::20a:e4ff:fee2:d91e eth0: 56 data bytes
64 bytes from fe80::211:22ff:fe33:4455: icmp_seq=1 ttl=64 time=0.066 ms
64 bytes from fe80::2cc:ccff:fecc:cccc: icmp_seq=1 ttl=64 time=0.207 ms (
64 bytes from fe80::211:22ff:fe33:4455: icmp_seq=2 ttl=64 time=0.066 ms

--- ff02::1 ping statistics ---
2 packets transmitted, 2 received, +1 duplicates, 0% packet loss, time 99
rtt min/avg/max/mdev = 0.065/0.112/0.207/0.067 ms
```

11. Lab: Discovering Router and Autoconfiguration
11.1 Discovering Router

Connect your PC (or its LAN) to a router. Note the automatic router discovery (you should see a new "2001:db8::" address assigned to your interface).

```
$ ip -6 addr
$ ip -6 addr show dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qlen 1000
    inet6 2001:db0::211:22ff:fe33:4455/64 scope global dynamic
       valid_lft 2600000sec preferred_lft 600000sec
    inet6 fe80::211:22ff:fe33:4455/64 scope link
       valid_lft forever preferred_lft forever

$ ip -6 route
$ ip -6 route show dev eth0
fe80::/64  metric 256  mtu 7200 advmss 7140 hoplimit 4294967295
default via fe80::02cc:ccff:fecc:cccc  proto kernel  metric 1024 \
  expires 1600sec  mtu 7200 advmss 7140 hoplimit 64
```

11. Lab: Discovering Router and Autoconfiguration
11.2 Router Advertisement Daemon

Install the Router Advertisement Daemon (radvd) on the router.

```
1) Edit its configuration file /etc/radvd.conf
# The LAN side of the router.
interface eth0
{
   IgnoreIfMissing on;
   AdvSendAdvert on;
   prefix 2001:db8::/64
   {
      AdvOnLink on;
      AdvAutonomous on;
      AdvRouterAddr on;   # This supports Mobile IPv6 (mobile clients)
   };
};

2) Start the daemon
# /etc/init.d/radvd start
```

11. Lab: Discovering Router and Autoconfiguration
11.2 Router Advertisement Daemon

Edit /etc/network/interfaces so radvd starts when interface comes up.

```
# The LAN side
iface eth0 inet6 static
     address 2001:db8::4
     netmask 64
     # assign to eth0 the IPv6 address of tunnel server
     up   ip -6 addr  add 2001:db8::1/64 dev $IFACE
     down ip -6 addr  del 2001:db8::1/64 dev $IFACE
     # drop traffic from unused subnets
     up   ip -6 route add 2001:db8::/48  dev lo
     down ip -6 route del 2001:db8::/48  dev lo
     # enable packet forwarding (should be set in /etc/sysctl.conf)
     up   echo 1 > /proc/sys/net/ipv6/conf/all/forwarding
```

11. Lab: Discovering Router and Autoconfiguration
11.2 Router Advertisement Daemon (cont)

```
# use largest MTU that does not generate the error:
#    # SIOCSIFMTU: Invalid argument
mtu 1500
# router advertisement daemon (radvd)
# If you forget to connect eth0, you may get the error:
#    # RTNETLINK answers: File exists
up   sleep 1
up   /etc/init.d/radvd start
down /etc/init.d/radvd stop
```

12. Lab: Browsing the Internet With IPv6
12.1 Domain Name Service

Domain Name Servers use AAAA records for IPv6 addresses.

```
1) Forward DNS query
$ dig AAAA ipv6.google.com
;; ANSWER SECTION:
ipv6.google.com.        10643   IN      CNAME   ipv6.l.google.com.
ipv6.l.google.com.      143     IN      AAAA    2001:4860:800f::63
ipv6.l.google.com.      143     IN      AAAA    2001:4860:800f::68
ipv6.l.google.com.      143     IN      AAAA    2001:4860:800f::93
ipv6.l.google.com.      143     IN      AAAA    2001:4860:800f::67

2) Reverse DNS query
$ dig -x 2001:4860:800f::63 +trace
```

12.2 Browsing IPv6 Sites

All major browsers are IPv6 ready.

```
1) Test IPv6 web-sites
$ ping6 -nc2 ipv6.google.com
$ ping6 -nc2 2001:4860:800f::63
$ iceweasel http://ipv6.google.com/
$ iceweasel http://[2001:4860:800f::63]/
$ iceweasel https://www.sixxs.net/main/

2) Configure your browser.  In this example, Konqueror:
  a) Click menu ‘‘Settings’’, click ‘‘Configure Konqueror...’’
  b) Click ‘‘Web Shortcuts’’
  c) Scroll down to and click ‘‘Google   gg, google’’
  d) Click ‘‘Change...’’
  e) Replace Search URI ‘‘www.google.com’’ with ‘‘ipv6.google.com’’
```

13. Lab: Making Services IPv6 Ready
13.1 apache2

1) Apache2 is IPv6 ready out of the box

```
$ netstat -nltup | grep apache2
tcp6     0     0 :::80              :::*             LISTEN      5947/apache2
```

2) If you want to make changes, edit /etc/apache2/ports.conf.

If you want Apache to handle...

a) both IPv4 and IPv6 connections (this is the default)

```
Listen 80
```

b) both IPv4 and IPv6 connections, but on separate sockets

```
Listen [::]:80
Listen 0.0.0.0:80
```

c) IPv4 connections only from given networks

```
Listen 0.0.0.0:80
Listen 192.168.0.1:80
```

13. Lab: Making Services IPv6 Ready
13.1 apache2 (cont)

```
  If you want Apache to handle...
  d) IPv6 connections only
Listen [::]:80

  Then restart the server
# /etc/init.d/apache2 restart
```

13. Lab: Making Services IPv6 Ready
13.2 sshd daemon, ssh/scp/sftp client

1) SSHD might be ready out of the box

```
$ netstat -nltup | grep apache2
tcp6     0    0 :::22                    :::*             LISTEN      7039/sshd
```

2) To enable the server, /etc/ssh/sshd_config uncomment:

```
ListenAddress ::
```

For extra security, add a line to define authorized accounts:

```
AllowUsers root <username> ... <username>
```

Then restart the server

```
# /etc/init.d/sshd restart
```

3) To enable the client, in /etc/ssh/ssh_config uncomment:

```
AddressFamily any  # [any|inet|inet6]
```

13. Lab: Making Services IPv6 Ready
13.2 sshd daemon, ssh/scp/sftp client (cont)

4) Try connecting your client to your own daemon.

```
$ ssh -6 ::1
$ ssh ::1
```

5) Then try connecting to a IPv6 address

```
$ ssh my-site.com         # replace this with a site you control
$ ssh 2001:db8::1         # use dig AAAA to get its IPv6 address
$ scp -pr local-file \[2001:db8::1\]:remote-file  # note the brackets
```

13. Lab: Making Services IPv6 Ready
13.3 NTP, Other Services

```
1) NTPv4 is IPv6 ready out of the box.  (Do not use NTPv3)
   To limit access to your NTP daemon, read
$ iceweasel http://support.ntp.org/bin/view/Support/AccessRestrictions

2) SIXXS offers a pool of NTP servers with IPv6 addresses.

3) For many other IPv6 ready services see:
$ iceweasel http://ipv6.niif.hu/m/IPv6apps
```

14. Homework: Setup a Proto 41 Tunnel
14.1 Tunnel Brokers

1) If you have native IPv6 via your ISP (RFC 4241) start using it.

2) If you have a static IPv4 address, go to Hurricane Electric

`http://ipv6.he.net/`
`http://www.tunnelbroker.net/`

which offers a free "6in4" tunnel service, which uses Protocol 41 (RFC 1933). HE also offers FAQ, presentations, and a certification program.

3) If you are mobile or stand behind a NAT, go to SIXXS

`http://www.sixxs.net/`

which offers a free 6in4 tunnel service, which uses the AYIYA (anything in anything) tunneling protocol.

15. Homework: Setup an IPv6 Subnet
15.1 Tunnel Brokers

When you have gained experience using IPv6 on a single PC, go back to your tunnel-broker (e.g. SIXXS) and order a subnet.

1) Add the following lines to your ip6tables:

```
# FORWARD chain
$IP6TABLES -A FORWARD -m rt --rt-type 0              -j REJECT
$IP6TABLES -A FORWARD -i eth0 -o sixxs -s $SUBNET  -j ACCEPT
$IP6TABLES -A FORWARD -i sixxs -o eth0 -d $SUBNET  -j ACCEPT
$IP6TABLES -A FORWARD                                -j REJECT
```

2)Uncomment the following lines in /etc/sysctl.conf:

```
net.ipv6.conf.all.accept_redirects=0
net.ipv6.conf.all.accept_source_route=0
net.ipv6.conf.all.forwarding=1
```

3) Then reload the kernel parameters:

```
# /sbin/sysctl -p
```

16. Homework: IPv6 Certification

16.1 Hurricane Electric

Hurricane offers an automated testing and certification program. It has seven levels:

Level	Understand	Demonstrate
Newbie	Basic IPv6 concepts	
Explorer	Tunneling	IPv6 connection
Enthusiast	HTTP, Webserver	IPv6 webserver
Administrator	SMTP, MTA	IPv6 email address
Professional	Reverse DNS	MTA has working rDNS
Guru	Forward DNS	Authoritative NS and AAAA record
Sage	DNS Glue	TLD IPv6 glue configuration

That is all there is to it!

A. References
Historical Documents

1 Introduction to Distributed Communication Networks,
Paul Baran, RM-3420-PR, 1964
<http://www.rand.org/about/history/baran.list.html>

2 Report No. 1822, Interface Message Processor: Specifications
for the Interconnection of a Host and an IMP
Bolt Beranek and Newman Inc.
<http://www.bitsavers.org/pdf/bbn/imp/BBN1822_Jan1976.pdf>

3 A Protocol for Packet Network Interconnection, Vinton Cerf
IEEE Transactions of Communications Technology, May, 1974.

4 How the Internet Came to Be, Vinton Cert, as told to Bernard Aboba
<http://netvalley.com/archives/mirrors/cerf-how-inet.html>

5 <http://www.livinginternet.com/i/ii_cerf.htm>

A. References

IPv4 and 16-bit ASN Exhaustion

1 IPv4 Address Report
<http://www.potaroo.net/tools/ipv4/index.html>

2 The 16-bit AS Number Report
<http://www.potaroo.net/tools/asns/index.html>

A. References
IPv6 HOWTO

1 Linux+IPv6-HOWTO
<http://www.bieringer.de/linux/IPv6/>
<http://tldp.org/HOWTO/Linux+IPv6-HOWTO/>

2 IPv6 with Debian Linux
<http://ipv6.debian.net/>

2 IPv6 Tutorials
<http://ipv6.he.net/presentations.php>

3 IPv6 Tunnel Brokers
<http://www.sixxs.net/>
<http://www.tunnelbroker.net/>