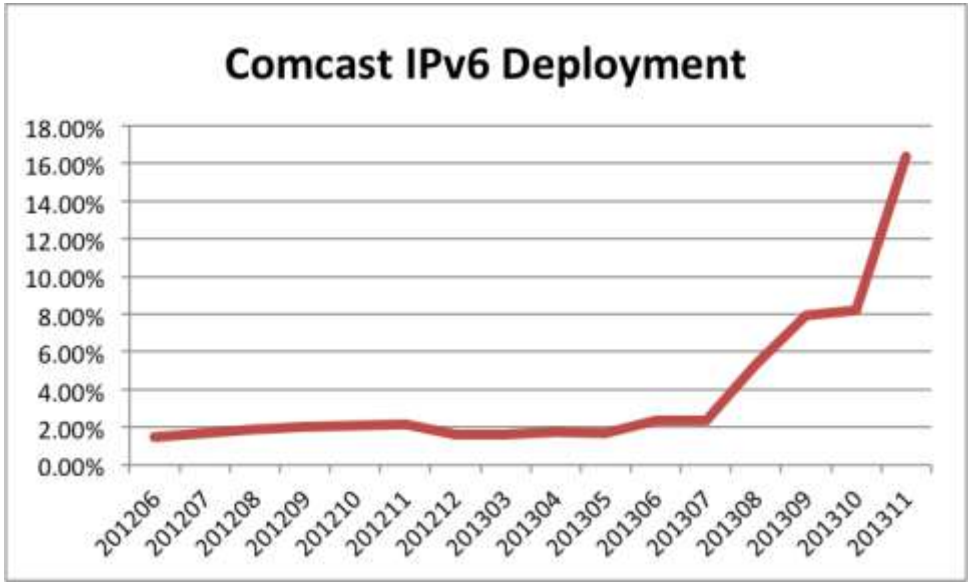


-or-

IPV6 LIVE

Current State and Demonstration of IPv6

James Small, Sr. Consultant at AT&T

Michigan!/usr/group

mug.org – A Free and Open Source Michigan Community

OVERVIEW OBJECTIVES

- Brief Why IPv6 and Current Landscape
- Getting IPv6 Up and Running
- IPv6 Reference and Parting Thoughts

Q&A throughout, I may postpone questions until the end depending on time

WHY IPV6?

Optimal Gaming Experience

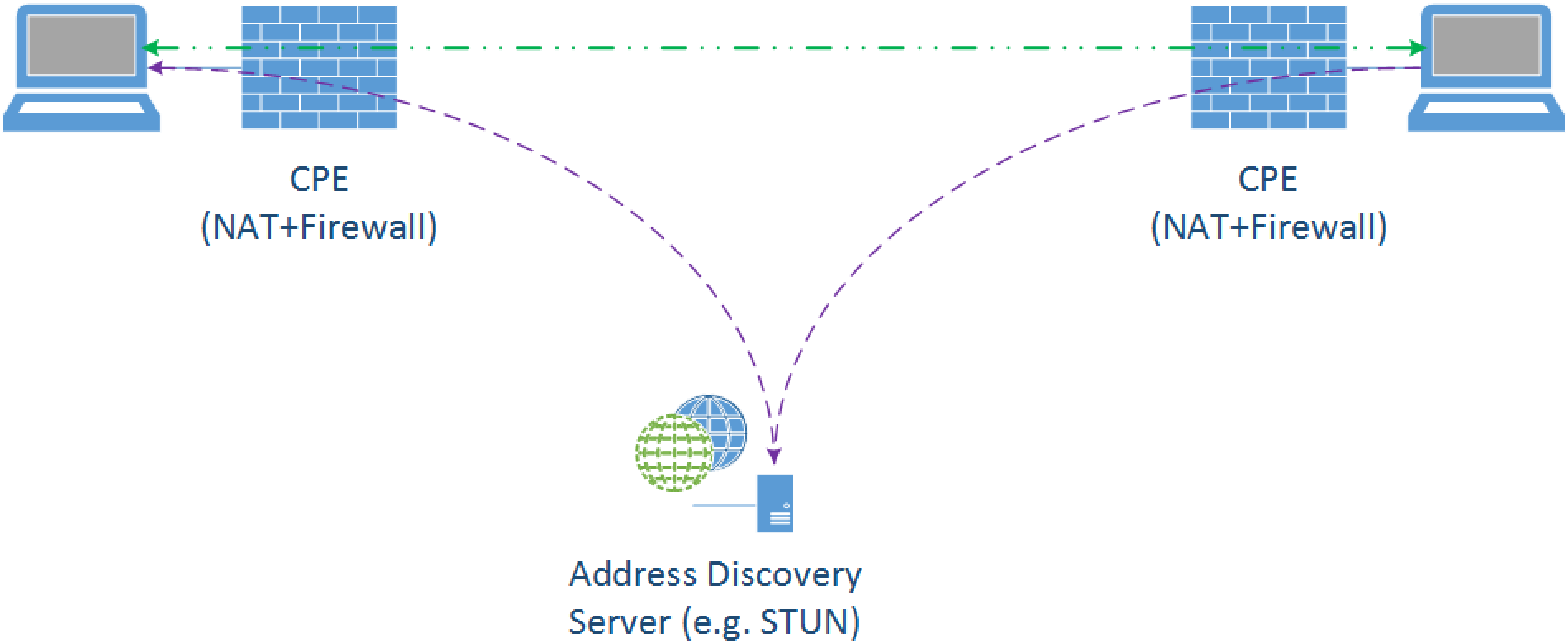
For the best possible user experience, Xbox users should use IPv6 connectivity.

– Christopher Palmer, Program Manager, Networking Core/Operating System Group

WHY IS IPV6 BETTER FOR GAMING?

Simple IPv4 Example

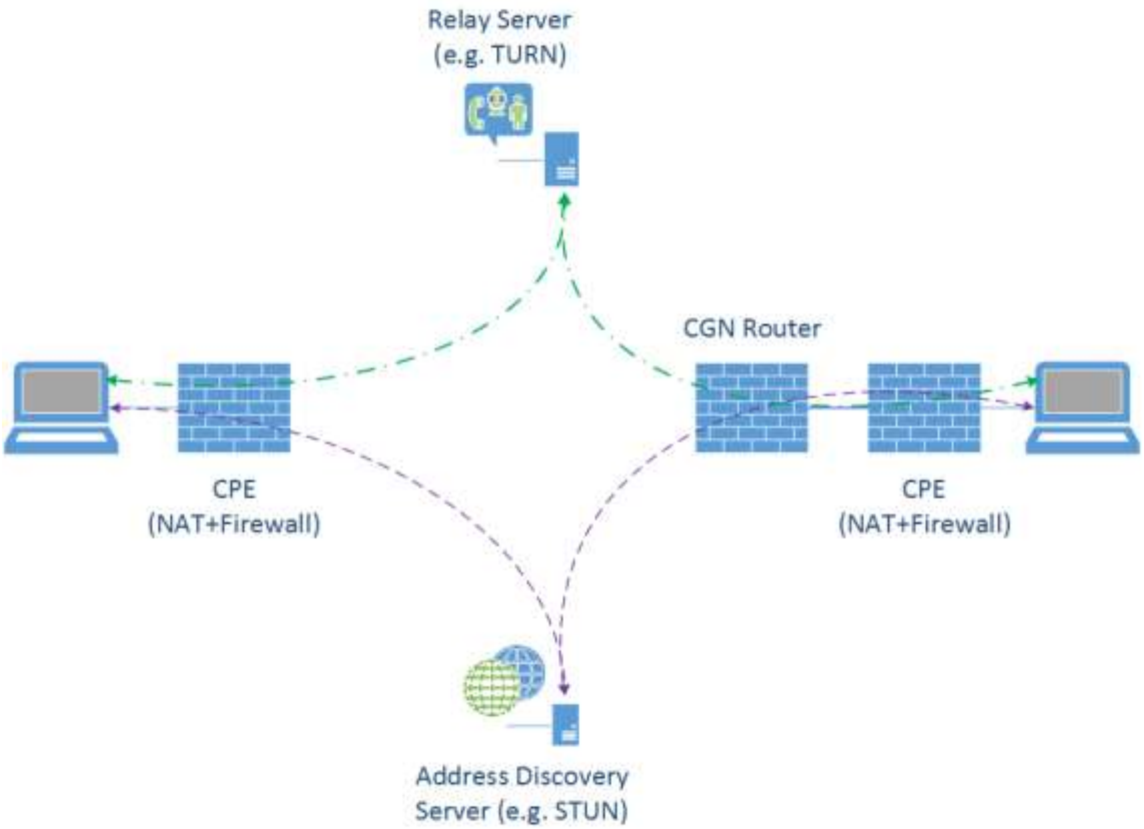
- Internet Server required for public address resolution



WHY IS IPV6 BETTER FOR GAMING?

More Typical IPv4 Example

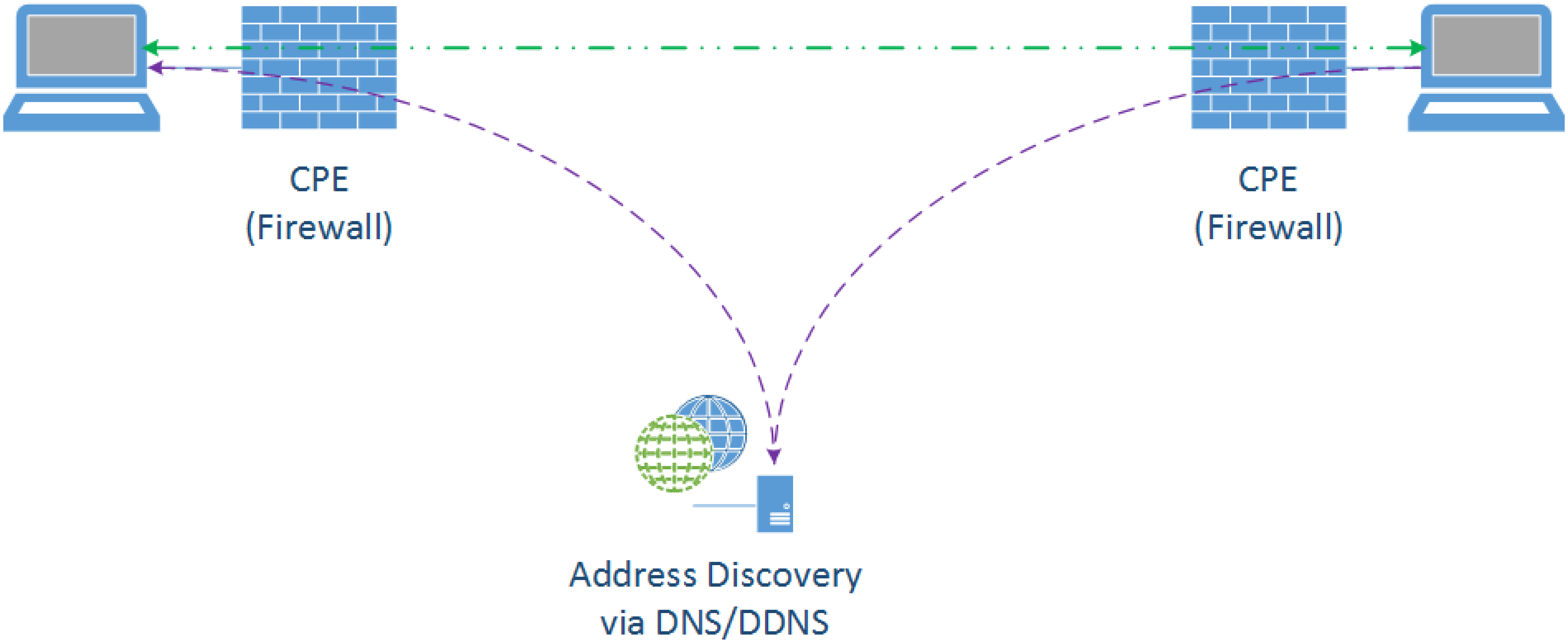
- Internet Server required for public address resolution
- STUN fails because of unsupported NAT setup
- Relay Server must be used
- Relaying adds latency
- Scale/speed/reliability dependent on relay service



WHY IS IPV6 BETTER FOR GAMING?

Typical IPv6 Example

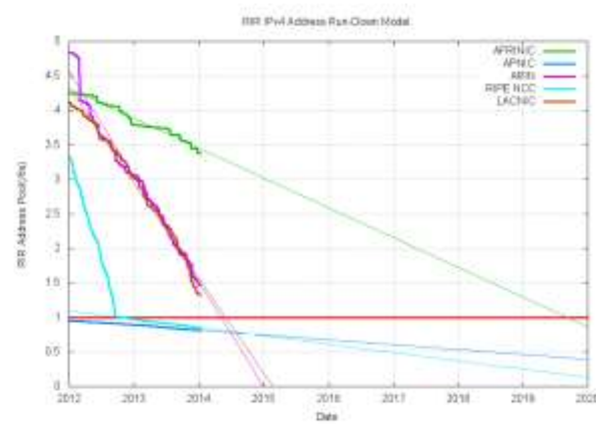
- DNS used for address resolution!
- At most, only a Dynamic DNS solution is required – many public ones readily available for free.



CURRENT IPV4 STATUS – DEPLETION

- IANA Free Pool Depletion – February, 2011
- APNIC (Asia/Pacific) Depletion – April, 2011
- RIPE (Europe/Middle East) Depletion – September, 2012
- ARIN (US/Canada) Depletion – ETA of Q1 2015
  - » Less than 25 million addresses remaining
  - » In 2012, ARIN handed out 45 million addresses
  - » Address allocation slowing because of CGN and addressing markets

- Massive growth coming from:
  - » Population penetration
  - » Mobile device explosion
  - » Internet of Things (all electronic devices – light bulbs, sensors, tags)



CURRENT IPV6 STATUS – DEPLOYMENT

Major Carrier Deployment Status

• Google Fiber	70.22%
• Virginia Tech	61.69%
• SPAWAR (US DOD)	51.12%
• Verizon Wireless	40.40%
• Comcast	20.15%
• AT&T	14.82%
• T-Mobile USA	6.49%

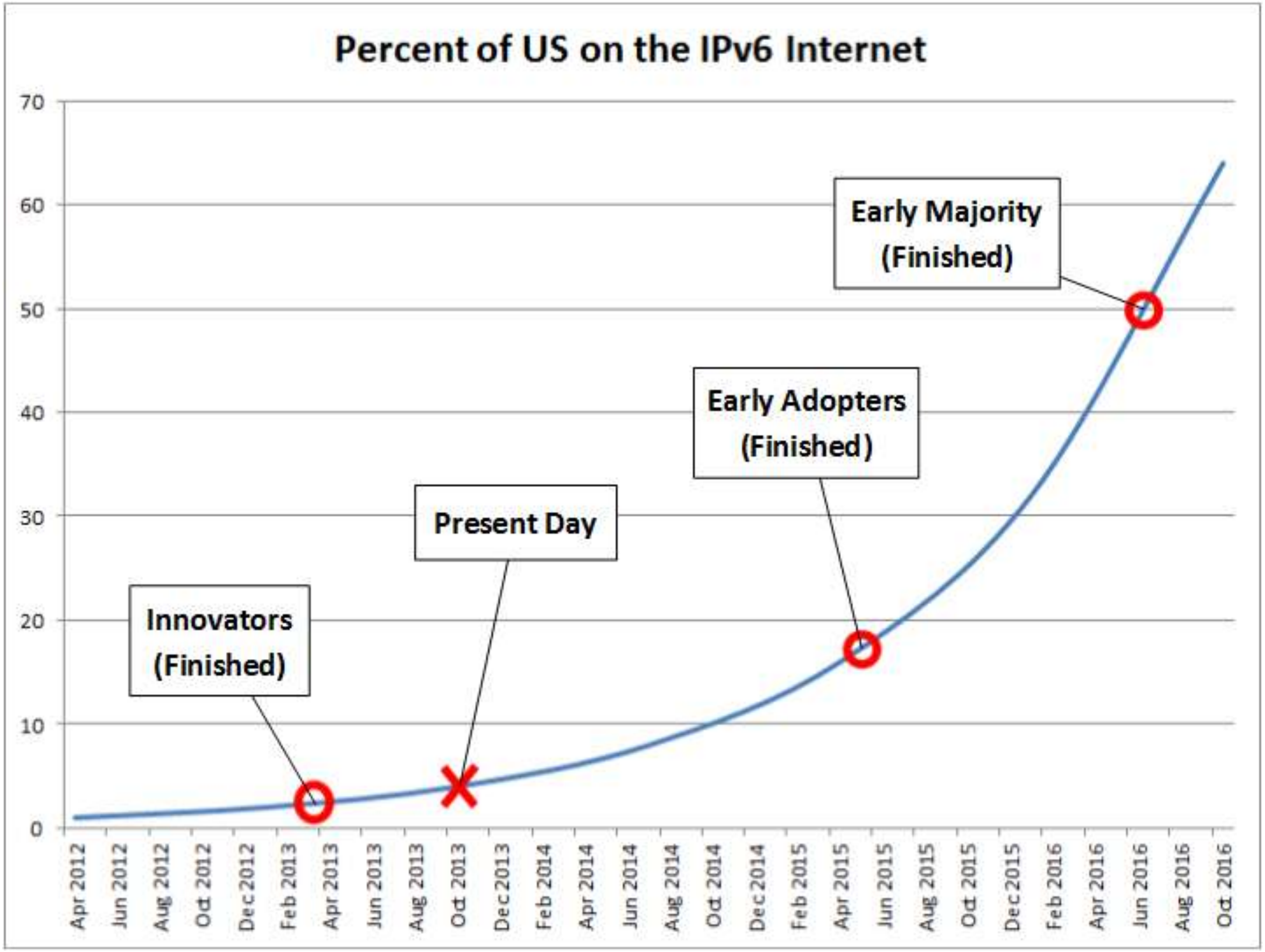
T-Mobile Goes IPv6 Only on Android 4.4 Devices using 464XLAT

US Overall IPv6 Penetration:

5.74% (fluctuates up to 6.25%)

US Weighted Web Content available via IPv6: 44.91%

CURRENT IPV6 PENETRATION PROJECTION

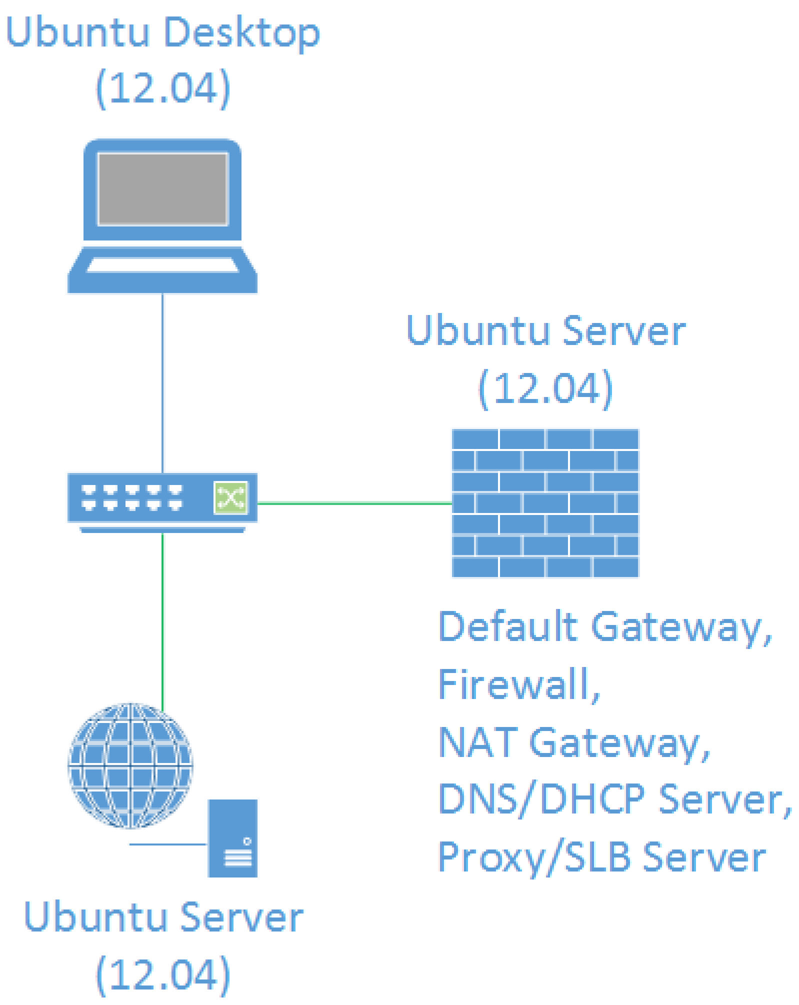


ROADMAP

- Brief Why IPv6 and Current Landscape
- ***Getting IPv6 Up and Running***
- IPv6 Reference and Parting Thoughts

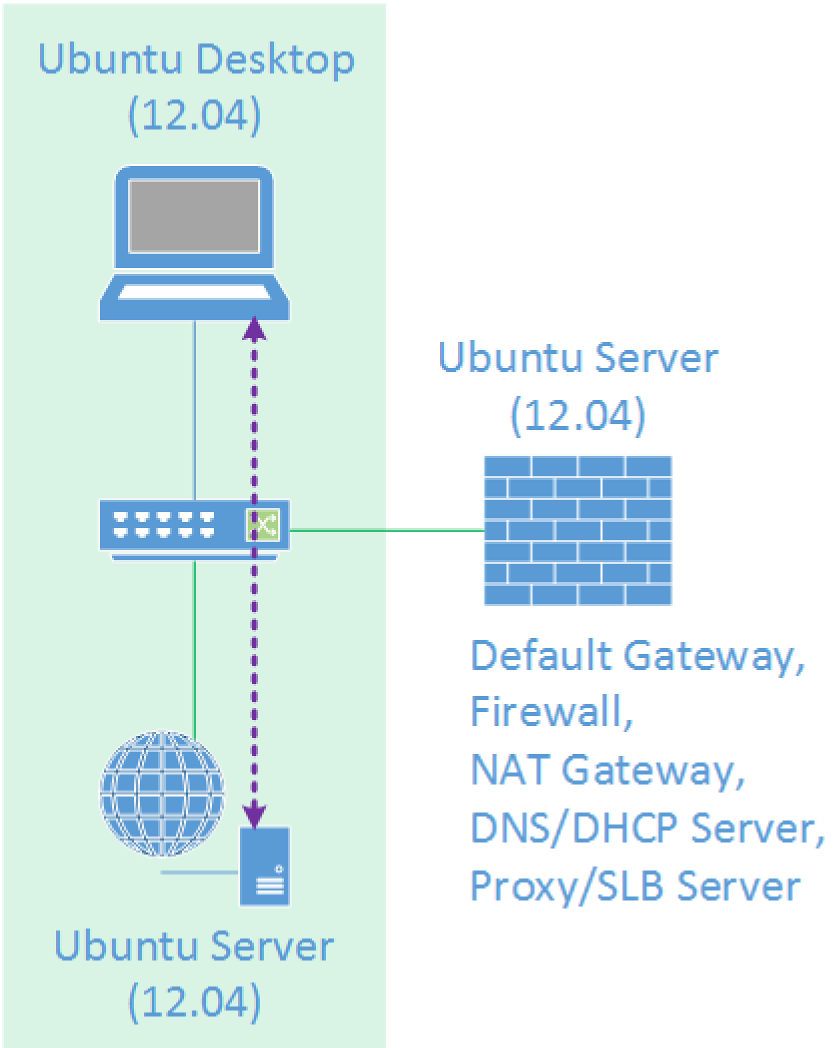
IPV6 TEST SETUP

- Using Linux
- Using the Ubuntu Distro
- Using the LTS Build
- Used all VMs
- Hypervisor ESX 5.1

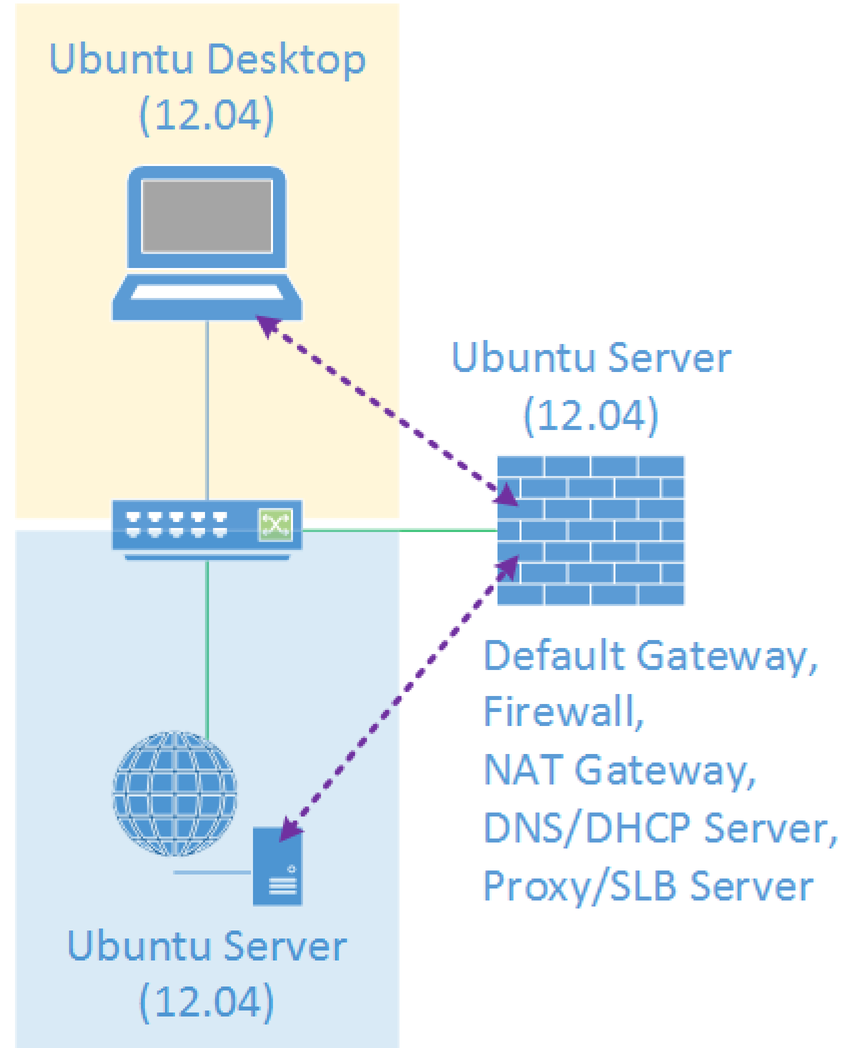


IPV6 TEST TOPOLOGIES

Desktop/Server on Shared VLAN

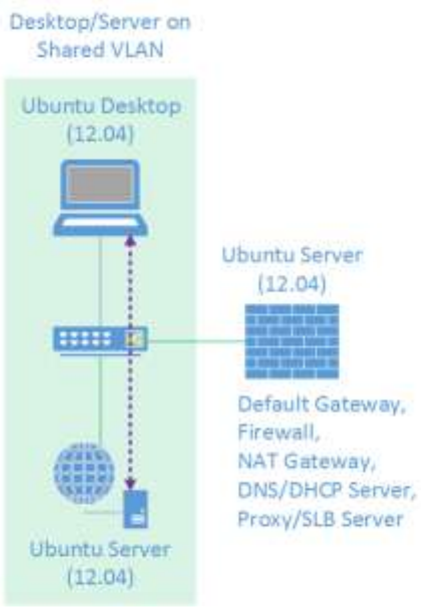


Desktop/Server on Separate VLANs



INITIAL IPV6 SETUP

- Default setup
- No IPv6 explicitly configured
- Let's look at the defaults:

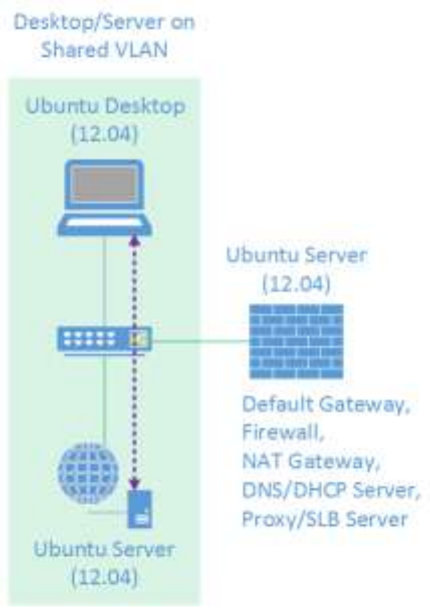



```
root@v6client:~# ifconfig -a
lo          Link encap:Local Loopback
            inet addr:127.0.0.1  Mask:255.0.0.0
            inet6 addr: ::1/128 Scope:Host
            UP LOOPBACK RUNNING  MTU:65536  Metric:1
           (…)

eth0        Link encap:Ethernet  HWaddr 00:50:56:8f:2a:54
            inet addr:192.168.231.11  Bcast:192.168.231.255  Mask:255.255.255.0
            inet6 addr: fe80::250:56ff:fe8f:2a54/64 Scope:Link
            UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
           (…)
```

INITIAL IPV6 SETUP

- Default setup
- No IPv6 explicitly configured
- Let's look at the defaults:




```
root@v6client:~# ifconfig -a
lo          Link encap:Local Loopback
            inet addr:127.0.0.1  Mask:255.0.0.0
            inet6 addr: ::1/128 Scope:Host
            UP LOOPBACK RUNNING  MTU:65536  Metric:1
          (…)

eth0        Link encap:Ethernet  HWaddr 00:50:56:8f:2a:54
            inet addr:192.168.231.11  Bcast:192.168.231.255  Mask:255.255.255.0
            inet6 addr: fe80::250:56ff:fe8f:2a54/64 Scope:Link
            UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          (…)
```

A QUICK STEP BACK

IPV4 VERSUS IPV6 ADDRESSING

Type	IPv4	IPv6
Unspecified	0.0.0.0	::
Loopback	127.0.0.1	::1
“Link-Local”	169.254.251.1	fe80::584c:7cf5:4a0e:3ce9%eth0
“Private”	10.152.16.87	fd8d:8b76:0494:659b::4ae:3ce9
Public	12.203.95.104	2001:46e:cae8::a7:1afe:e91d
Multicast	224.0.0.18	ff02::1:ff94:e774
“Broadcast”	255.255.255.255	ff02::1

INITIAL IPV6 SETUP

- Configure IPv6 address:

```
root@v6client:~# ifconfig eth0 inet6 add 2001:db8:1::1011/64

root@v6client:~# ifconfig eth0
         (…)
          inet6 addr: 2001:db8:1::1011/64 Scope:Global
         (…)
                          -or-

root@v6client:~# ip addr add 2001:db8:1::1011/64 dev eth0

root@v6client:~# ip addr show dev eth0
         (…)
    inet6 2001:db8:1::1011/64 scope global
       valid_lft forever preferred_lft forever
         (…)
```

IPV6 ADDRESS STRUCTURE

- Hexadecimal digits (0-9, a-f)
- 8 groups of 16 bit (4 digit) numbers
- Groups separated by colons (:)
- Digits not case sensitive, but lower case preferred
- Abbreviations are possible
  - » Can omit leading zeroes
  - » Can substitute longest string of zeroes with double colon (::)

- Examples:
  - » 2001:db8:cafe:f0a2:a8b0:2:ffe1:a90b
  - » fd92:e075:819c:7a2::fc9a:105
  - » fe80::f413:9b1e:9f3:feb6
  - » ff05::fb

IPV6 ADDRESSING BASICS

- IPv6 prefixes/addresses always use CIDR notation:
  - » IPv4 – 192.0.2.0/24
  - » IPv6 – 2001:db8:101::/48

- IPv6 addresses can omit leading zeroes, but not trailing ones:
  - » 2001:0db8:0101:00a0:0000:0000:0d20:9ce5
    - Becomes:
      - 2001:db8:101:a0:0:0:d20:9ce5
    - But Not (Just like 010 = 10, but 010 ≠ 1, 010 ≠ 01)
      - ~~2001:db8:101:a:0:0:d2:9ce5~~
      - ~~2001:db8:101:00a:0:0:0d2:9ce5~~

- IPv6 addresses can substitute longest string of zeroes with a double colon:
  - » 2001:db8:101:a0::d20:9ce5

IPV6 ADDRESSING

- Link Local – fe80::/10 (link only significance)
- Unique Local – fc00::/7 (not routable on Internet)
- Global – 2000::/3 (currently IANA allocated address space)

Currently:

- All global IPv6 addresses are currently allocated from 2000::/3
  - » In other words – 2000-3fff::
  - » Out of that, only 2000-2c0f:: has been assigned
  - » Bottom line – for now you will only see global IPv6 addresses starting with "2"
- Documentation prefix
  - » 2001:db8::/32 is reserved for documentation/examples, not routed on the Internet

IPV6 CHANGES FROM IPV4

- ~~***Broadcasts***~~
- ARP replaced by NDP, a subset of ICMPv6
- Nodes can auto-configure address with SLAAC (use RAs)
- DHCP replaced by RAs (subset of NDP) + DHCPv6
- Blocking ICMPv6 will completely break IPv6!!!
  - » Careful with firewall/route/switch/operating system ACLs
- Minimum MTU changes from 68 to 1280
- Fragmentation only done by end nodes, not by routers

NO MORE BROADCASTS

Why not?

- Most common link layer – Ethernet
- Broadcast sent to all on-link nodes regardless of need
  - » Node doesn't use protocol? Too bad…
  - » Node isn't interested in traffic? Too bad…
  - » Inefficient
- The source of many problems
  - » Broadcast storms
  - » Scalability issues

Replaced with Multicast

NO MORE BROADCASTS

Broadcast replacement is link-local multicast:

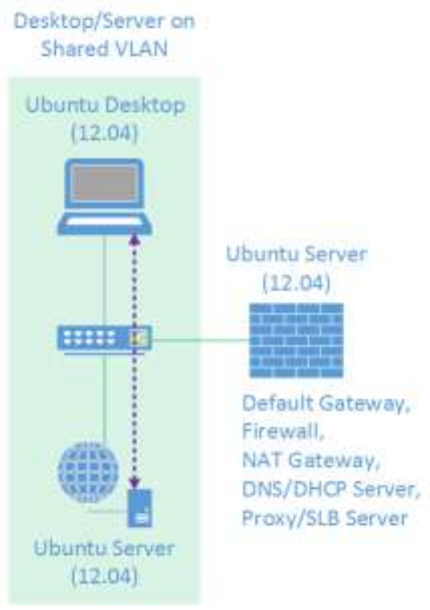
- ff02::1 – All nodes on the link
- Each application has its own group address
  - ff02::2 – All routers
  - ff02::fb – mDNSv6
  - ff02::1:2 – All DHCPv6 agents
  - Only interested nodes join the group
- IPv6 brings scoping
  - No more choosing between network security/stability and handy broadcast features
  - Directed broadcasts for features like Wake on LAN will be replaced with scoped multicast

INITIAL IPV6 MULTICAST GROUPS?

- Default setup
- No IPv6 explicitly configured
- Let’s look at the defaults:

```
root@v6client:~# ip -6 maddr show
1:      lo
        inet6 ff02::1

2:      eth0
        inet6 ff02::1
        inet6 ff02::fb
        inet6 ff02::1:ff8f:2a54
```



IPV6 CHANGES FROM IPV4

- ~~Broadcasts~~
- ***ARP replaced by NDP, a subset of ICMPv6***
- Nodes can auto-configure address with SLAAC (use RAs)
- DHCP replaced by RAs (subset of NDP) + DHCPv6
- Blocking ICMPv6 will completely break IPv6!!!
  - Careful with firewall/route/switch/operating system ACLs
- Minimum MTU changes from 68 to 1280
- Fragmentation only done by end nodes, not by routers

IP L2 ADDRESS RESOLUTION

IPv4 L2 Cache:

```
root@v6client:~# arp -a
v6client.local (192.168.231.11) at 00:0c:29:e9:bc:50 [ether] on eth0
v6server.local (192.168.231.12) at 00:26:2d:fc:05:9b [ether] on eth0
v6gateway.local (192.168.231.13) at 00:00:0c:9f:f0:65 [ether] on eth0
```

IPv6 L2 Cache:

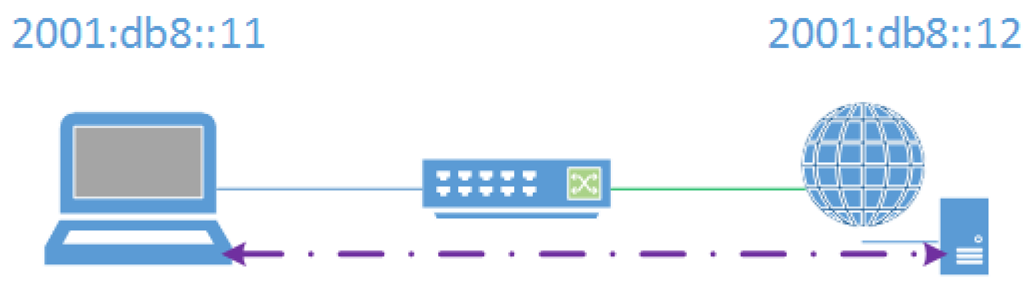
```
root@v6client:~# ip -6 neigh show
fe80::301:1 dev eth0 lladdr 00:05:73:a0:00:66 router STALE
2001:db8:c3:f1:3fd:6182:d9b6:b027 dev eth0 lladdr 00:26:2d:fc:05:9b REACHABLE
2001:db8:c3:f1::101:a861 dev eth0 lladdr 00:0c:29:80:a8:61 STALE
```

IPV4 L2 ADDRESS RESOLUTION (REVIEW)

- Client (192.168.231.11) needs to talk to server (192.168.231.12)
- Client needs MAC Address (L2) for server
- Client uses ARP to resolve IPv4 Address (L3) to MAC Address (L2)
- ARP uses broadcast – who owns the IPv4 Address 192.168.231.12?

- Note: ARP is a L2 protocol, does not use IPv4

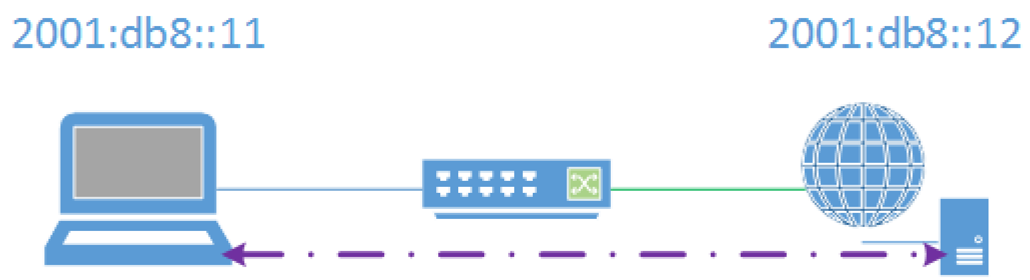
IPV6 L2 ADDRESS RESOLUTION



- Client (2001:db8::11) needs to talk to server (2001:db8::12)
- Client needs MAC Address (L2) for server
- Client uses NDP to resolve IPv6 Address (L3) to MAC Address (L2)
- NDP uses ICMPv6 which uses a special multicast group – who owns the IPv6 Address 2001:db8::12?

- Note: NDP/ICMPv6 is a L3 protocol, uses IPv6

IPV6 L2 ADDRESS RESOLUTION



NDP uses a special multicast group for L3 to L2 address resolution – the solicited node multicast address:

- ff02:0:0:0:0:1:ff/104
- NDP takes the last 24 bits of the address it needs L2 info for and appends it:
  - » Server Address: 2001:db8::12 (2001:db8::00:0012)
  - » Solicited Node Address: ff02::1:ff00:12

Much more scalable that ARP!

IPV6 L2 ADDRESS MECHANICS

- An IPv6 node will join the solicited node multicast address group for each interface address:

```
root@v6client:~# ip -6 addr show dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qlen 1000
    inet6 2001:db8:1::1011/64 scope global
       valid_lft forever preferred_lft forever
    inet6 fe80::250:56ff:fe8f:2a54/64 scope link
       valid_lft forever preferred_lft forever

root@v6client:~# ip -6 maddr show dev eth0
2:      eth0
        inet6 ff02::fb
        inet6 ff02::1:ff00:1011
        inet6 ff02::1:ff8f:2a54
        inet6 ff02::1
```

IPV6 CHANGES FROM IPV4

- ~~Broadcasts~~
- ARP replaced by NDP, a subset of ICMPv6
- ***Nodes can auto-configure address with SLAAC (use RAs)***
- DHCP replaced by RAs (subset of NDP) + DHCPv6
- Blocking ICMPv6 will completely break IPv6!!!
  - » Careful with firewall/route/switch/operating system ACLs
- Minimum MTU changes from 68 to 1280
- Fragmentation only done by end nodes, not by routers

ENABLE IPV6 ROUTER

- Enable IPv6 routing on gateway host – preparation:

```
Note:  radvd not present
root@v6gateway:~# apt-get install radvd

# Enable IPv6 routing
root@v6gateway:~# sysctl net.ipv6.conf.all.forwarding
net.ipv6.conf.all.forwarding = 0

root@v6gateway:~# sysctl -w net.ipv6.conf.all.forwarding=1
net.ipv6.conf.all.forwarding = 1
```

ENABLE IPV6 ROUTER

- Enable IPv6 routing on gateway host – preparation:

```
# Setup radvd.conf:
interface eth0 {
        AdvSendAdvert on;
        AdvManagedFlag off;      # Enable for stateful DHCPv6
        AdvOtherConfigFlag off; # Enable for stateless DHCPv6
#       AdvLinkMTU 1480;         # If needed (e.g. tunnels)

        prefix 2001:db8:1::/64 {
                AdvOnLink on;
                AdvAutonomous on;
};
        RDNSS 2001:4860:4860::8888 2001:4860:4860::8844 {
        };
        DNSSL test.local {
        };
};
```

ENABLE IPV6 ROUTER

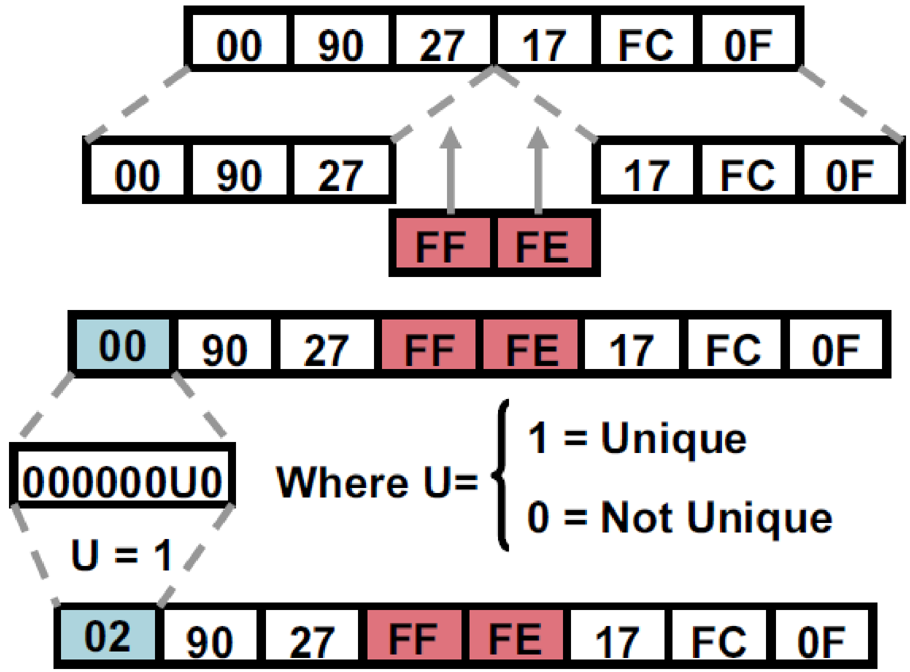
- Enable IPv6 routing on gateway host – preparation:

```
# Start radvd
service radvd start

# Let's look at a previously unconfigured client
# on the same link:
root@v6client:~# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:50:56:8f:2a:54
         (…)
          inet6 addr: 2001:db8:1:0:19ed:b935:f346:30e2/64 Scope:Global
          inet6 addr: fe80::250:56ff:fe8f:2a54/64 Scope:Link
          inet6 addr: 2001:db8:1:0:250:56ff:fe8f:2a54/64 Scope:Global
         (…)
```

IPV6 INTERFACE ADDRESSING

- Always 64 bits – Why?
  - IEEE hands out 48 bit (EUI-48) and 64 bit (EUI-64) MACs
  - Original idea was to use MAC to generate IID like in IPX
  - For 48 bit MACs, you insert 0xFFFE in the middle to generate a 64 bit address (IEEE Rule)
  - The "U" bit is also flipped (Modified EUI-64)
    This is so if you create a local address you can use 2001:db8::1 instead of 2001:db8::0200:0:0:1
    Wasn't that nice of them?



IPV6 ADDRESSES – SLAAC

- When we enabled the Router Advertisement Daemon (radvd), we enabled SLAAC for the link
- SLAAC – StateLess Address Auto-Configuration
- Idea is to make IPv6 more plug-in-play like IPX
- Compare IPv6 Addresses to MAC Address:
- 00:50:56:8f:2a:54 should be 0**2**50:56***ff:fe***8f:2a54

```
root@v6client:~# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:50:56:8f:2a:54
          (…)
           inet6 addr: 2001:db8:1:0:19ed:b935:f346:30e2/64 Scope:Global
           inet6 addr: fe80::250:56ff:fe8f:2a54/64 Scope:Link
           inet6 addr: 2001:db8:1:0:250:56ff:fe8f:2a54/64 Scope:Global
          (…)
```

?

√

IPV6 ADDRESSES – SLAAC

- The ip command is more useful for examing IPv6 Addresses:

```
root@v6client:~# ip -6 addr show dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qlen 1000
    inet6 2001:db8:1:0:19ed:b935:f346:30e2/64 scope global temporary dynamic
       valid_lft 85910sec preferred_lft 13910sec
    inet6 2001:db8:1:0:250:56ff:fe8f:2a54/64 scope global dynamic
       valid_lft 85910sec preferred_lft 13910sec
    inet6 fe80::250:56ff:fe8f:2a54/64 scope link
       valid_lft forever preferred_lft forever
```

- What’s a temporary address?

IPV6 INTERFACE ADDRESSING

- Modified EUI-64 was originally used for Stateless Auto Address Configuration (SLAAC)
- Unfortunately M-EUI-64 based addresses can be used as a super cookie
- To address this, privacy extensions were created (RFC 4941)

Privacy Extensions

- Nodes use a rotating, temporary address for outgoing communication
  - » The address changes periodically – typically once/day by default
- When creating the IID, a random number is used instead of the M-EUI-64 process
  - » This solves the “cookie” problem

IPV6 PRIVACY ADDRESSING

Privacy Addressing in Linux:

```
sysctl net.ipv6.conf.all.use_tempaddr

    use_tempaddr - Privacy Extensions Configuration

        <= 0 : disable Privacy Extensions

        == 1 : enable Privacy Extensions, but prefer public
               addresses over temporary addresses

         > 1 : enable Privacy Extensions and prefer temporary
               addresses over public addresses


Note:  Default in Ubuntu 12.04 is 2
```

- Privacy addresses are great for consumers but problematic for the enterprise
  - » DHCPv6 relays don't include client MAC address
  - » Privacy addresses make accountability/security difficult because addresses periodically rotate/change

IPV6 CHANGES FROM IPV4

- ~~Broadcasts~~
- ARP replaced by NDP, a subset of ICMPv6
- Nodes can auto-configure address with SLAAC (use RAs)
- ***DHCP replaced by RAs (subset of NDP) + DHCPv6***
- Blocking ICMPv6 will completely break IPv6!!!
  - » Careful with firewall/route/switch/operating system ACLs
- Minimum MTU changes from 68 to 1280
- Fragmentation only done by end nodes, not by routers

DHCPV6

DHCPv6 setup requires several components

- Local router advertising to use DHCPv6
  - Stateless DHCP

```
# radvd.conf:
interface eth0 {
    AdvOtherConfigFlag on; # Enable for stateless DHCPv6
```

  - Stateful DHCP

```
    AdvManagedFlag on;     # Enable for stateful DHCPv6
```

- DHCPv6 server
- Optionally DHCPv6 relays

DHCPV6 SERVER

DHCPv6 server setup - stateless

```
# DHCPv6 Parameters:
subnet6 2001:db8:1::/64 {
        # Options
        option dhcp6.name-servers 2001:4860:4860::8888;
        option dhcp6.domain-search "test.local";
}

Start:
dhcpd -6 -cf <PATH-to-configfile> -lf <PATH-to-leasefile>
```

DHCPV6 SERVER

DHCPv6 server setup - stateful

```
# DHCPv6 Parameters:
subnet6 2001:db8:1::/64 {
        # Range for clients
        range6 2001:db8:1::2000 2001:db8:1::2fff;
}

Start:
dhcpd -6 -cf <PATH-to-configfile> -lf <PATH-to-leasefile>
```

IPV6 CHANGES FROM IPV4

- ~~Broadcasts~~
- ARP replaced by NDP, a subset of ICMPv6
- Nodes can auto-configure address with SLAAC (use RAs)
- DHCP replaced by RAs (subset of NDP) + DHCPv6
- ***Blocking ICMPv6 will completely break IPv6!!!***
  - ***Careful with firewall/route/switch/operating system ACLs***
- Minimum MTU changes from 68 to 1280
- Fragmentation only done by end nodes, not by routers

IPV6 ACCESS CONTROL

- Reference NIST SP 800-119 (Section 3.5, Table 3-7)

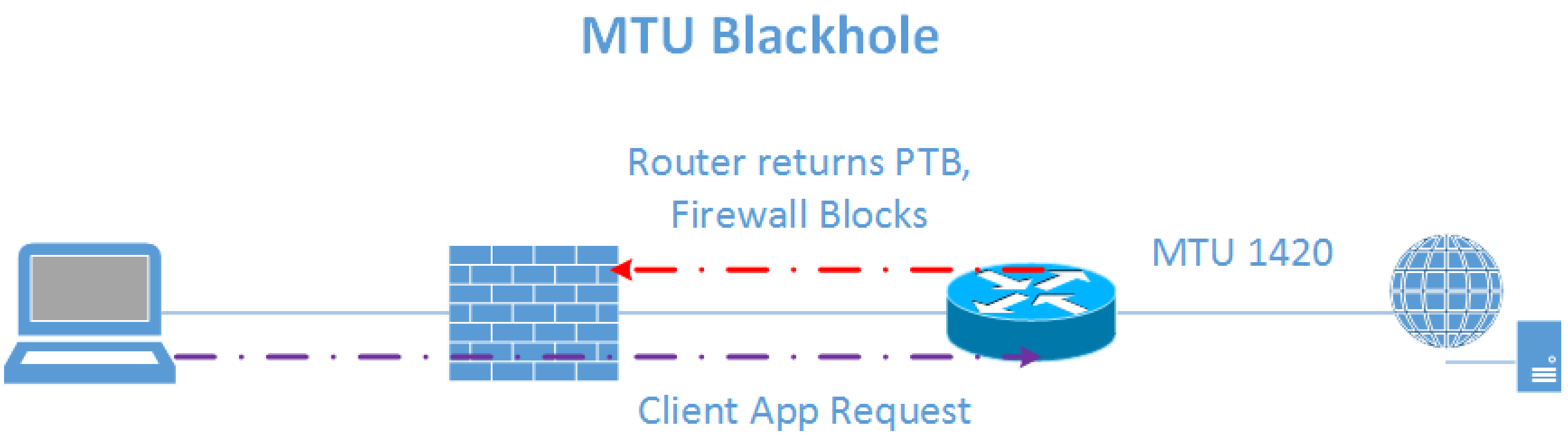
	Must Not Drop		Should Not Drop	
Message (Type)	Transit	Local	Transit	Local
Maintenance of Communication: Allow non-local when associated with allowed connections				
Destination Unreachable (1) – All codes	•	•		
Packet Too Big (2)	•	•		
Time Exceeded (3) – Code 0 only	•	•		
Parameter Problem (4) – Codes 1 and 2 only	•	•		
Connectivity Checking: Allow/disallow non-local based on topology/information concealment policy				
Echo Request (128)	•	•		
Echo Response (129)	•	•		
Address Configuration and Router Selection: Allow in link-local traffic only				
Router Solicitation (133)		•		
Router Advertisement (134)		•		
Neighbor Solicitation (135)		•		
Neighbor Advertisement (136)		•		
Inverse Neighbor Discovery Solicitation (141)		•		
Inverse Neighbor Discovery Advertisement (142)		•		

- Reference RFC 4890 (Recommendations for Filtering ICMPv6 Messages in Firewalls)

IPV6 CHANGES FROM IPV4

- ~~Broadcasts~~
- ARP replaced by NDP, a subset of ICMPv6
- Nodes can auto-configure address with SLAAC (use RAs)
- DHCP replaced by RAs (subset of NDP) + DHCPv6
- Blocking ICMPv6 will completely break IPv6!!!
  - Careful with firewall/route/switch/operating system ACLs
- ***Minimum MTU changes from 68 to 1280***
- ***Fragmentation only done by end nodes, not by routers***

IPV6 PATH MTU BLACKHOLE



- Client starts application (e.g. browser) which talks to server
- The server link has a lower MTU
- The intermediate router sends back an ICMPv6 Packet Too Big reply
- An intermediate firewall blocks all ICMPv6 traffic creating a Path MTU Blackhole

ROADMAP

- Brief Why IPv6 and Current Landscape
- Getting IPv6 Up and Running
- ***IPv6 Reference and Parting Thoughts***

IPV6 AND DNS

DNS has supported IPv6 for a long time – only a new resource record for the address and alternate pointer name space:

- IPv4 – A Record:
  - arin.net. IN A 192.149.252.75
- IPv6 – AAAA Record:
  - arin.net. IN AAAA 2001:500:4:13::80

- IPv4 – PTR Record:
  - 75.252.149.192.***in-addr.arpa.*** IN PTR www.arin.net.
- IPv6 – PTR Record:
  - 0.8.0.0.0.0.0.0.0.0.0.0.0.0.3.1.0.0.4.0.0.0.0.0.5.0.1.0.0.2.***ip6.arpa.*** IN PTR www.arin.net.

Transport agnostic - works equally well over IPv4 or IPv6, careful!

MULTI-PROTOCOL REALITIES

IPv4 and IPv6 are ships in the night!

IPv4 Firewall

```
# iptables -nL
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     icmp --  0.0.0.0/0            0.0.0.0/0
ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0            state RELATED,ESTABLISHED
ACCEPT     tcp  --  0.0.0.0/0            0.0.0.0/0            tcp dpt:22
```

IPv6 Firewall

```
root@ubuntu12:~# ip6tables -nL
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     icmpv6    ::/0                 ::/0
ACCEPT     all       ::/0                 ::/0                 state RELATED,ESTABLISHED
ACCEPT     tcp       ::/0                 ::/0                 tcp dpt:80
```

LINUX IPV6 TOOLS

General:

- ping → ping6
- traceroute → traceroute6
- tracepath → tracepath6

More:

- host/nslookup/dig – same tool, may have to specify IPv6 records (e.g. AAAA)
- telnet (still useful for raw connection to service) - same
- ssh - same

Network Analysis:

- tcpdump – only filtering options change
- wireshark – only filtering options change

LINUX IPV6 TOOL QUIRKS

IPv6 literals (RFC 3986)

- Generally means enclose the IPv6 address in brackets:
  - 2001:db8:fb::1a → [2001:db8:fb::1a]
- Necessary or many programs will interpret colons as port number delimeter
- Much more "interesting" if you use link local or multicast as you must specify the interface with a zone identifier

Examples:

- wget http://[2001:500:4:13::80] – works fine
- curl http://[2001:500:4:13::80] – doesn't work:
  - curl: (3) [globbing] error: bad range specification after pos 9
  - Known issue, must use "-g":
  - curl -g http://[2001:500:4:13::80]

LINUX IPV6 GUIDANCE

Start with the Linux IPv6 How To:

http://www.tldp.org/HOWTO/Linux+IPv6-HOWTO/index.html

Many things depend on the distro – check out wikis/documentation:

Ubuntu: https://wiki.ubuntu.com/IPv6

Debian: http://madduck.net/docs/ipv6/

A WORD OF CAUTION ON NAT

- NAT use cases:
  - Address Conservation
  - Topology Hiding
  - Path Symmetry
  - Provide some independence from ISP
  - Simple/Limited Multihoming
  - Restricts inbound connections

- New
  - Address Family translation

A WORD OF CAUTION ON NAT

- NAT Challenges
  - Adds complexity/operational overhead
  - Many applications use embedded addresses which is broken by NAT
  - Many applications require ALG to work through NAT
  - As applications are upgraded, ALG must be too
  - Loss of end to end connectivity/visibility
  - Make troubleshooting/auditing/attribution much harder

IPV6 ADDRESS PROVISIONING THOUGHTS

Address Options and Applicable Systems:

- Pure Static (Must disable SLAAC)
- Static with Options
- SLAAC, no DHCPv6
  - » Basic
  - » RDNSS
  - » Dynamic VLAN Assignment
- SLAAC with (Stateless) DHCPv6
- DHCPv6 (Stateful DHCPv6)
  - » Still requires SLAAC for default gateway

BUILD YOUR IPV6 DMZ

In order of preference:

- Option 1 – Dual Stack
- Option 2 – Load balanced (SLB64)
- Option 3 – Dual Stack Reverse Proxy
- Option 4 (Discouraged) – Use NAT64

BUILD YOUR IPV6 INTERNAL NETWORK

- Connect Internal IPv6 Network to IPv6 Internet
  - » Option 1 (Preferred) – Dual Stack
  - » Option 2 – Forward Proxy
  - » Option 3 – (Legacy) Tunneling
  - » Option 4 – Stateful NAT64 (IPv6 Only)

IPV6 CONNECTIVITY OPTIONS

In order of preference:

- Native dual stack (e.g. Comcast XFinity)
- You have a direct public IPv4 address:
  - 6rd – Must be supported by your ISP
  - Tunnelbroker (6in4 tunnel) – Hurricane Electric
  - Unmanaged 6to4 tunnel – Works better if your ISP supports, but will work without too
- Behind a NAT gateway/CGN/LSN or can’t terminate ISP connection:
  - AYIYA to Tunnelbroker (SiXXS)
  - TSP with Gogonet (Freenet6)
  - VPN or Tunnel Connection to someplace with IPv6 support
  - Use public Teredo/Miredo servers (but performance isn’t great)

OBTAIN AN IPV6 NETWORK ADDRESS

- Sign up for free IPv6 Internet access from Hurricane Electric (http://tunnelbroker.net)
- With your account, request a /48 prefix

- Q: Why start with Hurricane Electric?
- A: It works great, service is available from anywhere on the Internet, and you get a /48 all for free.

- Most important aspect of starting with HE:
  - » You need practice creating an addressing plan and deploying IPv6. It will take you at least 3 times to get your addressing plan right so let's get started…

IPV6 PREFIX POLICIES

- When multiple transport protocols are used (IPv4 and IPv6), a method must exist to choose which one is used including:
  - » Use IPv4 or IPv6?
  - » Where multiple addresses exist:
    - Which destination address should be chosen?
    - Which source address should be chosen?

RFC 6724 - Default Address Selection handles this

Note: Replaces RFC 3484

Prefix policies (RFC 6724 implementation) may be viewed and changed:

ip addrlabel show

IPV6 SPECIAL ADDRESSES

IPv4 Compatible - ::<IPv4> (deprecated)

IPv4 Mapped - ::FFFF:<IPv4>

Discard Prefix - 0100::/64 (Implement RTBH)

Well Known Prefix - 64:ff9b::/96 (NAT64)

Teredo - 2001:0000::/32 (Tunnel through NAT)

Documentation - 2001:db8::/32

6to4 - 2002::/16 - 6to4 (Tunnel through IPv4)

- 2002:<Public IPv4 Address>::/48 (Private IPv4 Address undefined)

ISATAP (IntraSite Tunnel through IPv4)

- 64 bit Unicast Prefix:0:5efe:<Private IPv4 Address>
- 64 bit Unicast Prefix:200:5efe:<Public IPv4 Address>

Solicited-Node Multicast

- ff02::1::ff00:0/104 + last 24 bits of IPv6 Address

IPV6 ADDRESS SPACE

Address Range	Description
0000-00FF::/8	Reserved and Special Purpose
0100-1FFF::/4	Reserved
2000-3FFF::/3	Global Unicast Addresses
2000-2CFF	Allocated
2D00-3FFF	Unallocated
4000-FBFF	Reserved (*5 more /3s*, a /4, /5, & /6)
FC00-FDFF::/7	Unique Local Unicast Addresses
FC00-FCFF::/8	Reserved for centralized allocations
FD00-FDFF::/8	Unrestricted – no registry/registration
FE00-FE7F::/9	Reserved
FE80-FEBF::/10	Link-local Addresses
FEC0-FEFF::/10	Reserved (Formerly Site-Local)
FF00-FFFF::/8	Multicast

MONITORING AND CONTROLLING IPV6

Service	Number	Description
IPv6 Encapsulation	IPv4/41	Tunnel IPv6 over IPv4
Generic Tunnel	IPv4/47	Tunnel anything over GRE
Teredo/Miredo	UDP/3544	Tunnel IPv6 over UDP (NAT Traversal)
Teredo/Miredo	Non-Standard	IPv6 destination starting with 2001:0000::/32 over UDP over IPv4
TSP	TCP|UDP/3653	IPv6 Tunnel Broker using the Tunnel Setup Protocol (RFC 5572)
AYIYA	TCP|UDP/5072	IPv6 Tunnel Broker using Anything in Anything (www.sixxs.net/tools/ayiya/)
Public 6to4 Anycast Relay	IPv4:192.88.99.1	Starting with IPv6 source address of 2002::/16 (6to4 is IPv6 over IPv4/41) Destined to 192.88.99.0/24 for IPv4
IPv6 Encapsulation	TCP/443	IPv6 over IPv4 SSL Tunnel, many variants
IPv6 Ethertype	0x86DD	Distinct from IPv4 Ethertype (0x0800)
DNS IPv6 Records	Several	AAAA, updated PTR records - can be transported over IPv4 or IPv6

Image source: gfi.com

QUESTIONS