Orbitage
it's about communicating

Application & API aspects of IPv6

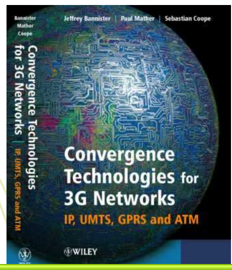
Mr Bjarte Olsen
bjarte@orbitage.com

info@orbitage.com

Orbitage
it's about communicating

About Orbitage

- Leading provider of consultancy and training
- Working with high tech industry in Malaysia including Telecoms, IT, Banking, Oil & Gas, etc.
- Our services:
  - Competence Management
  - Assessment & Training
  - Software Development
  - Consultancy Services
- Highlights in Malaysia:
  - Working with Telecoms & ICT to define a competence development framework based on job requirements
  - Ongoing structured "practical" skills development and assessment across Telecoms & ICT
  - Developed national competence standards in collaboration with Telcos and other industry players



Wrote a worldwide best selling book covering 3G & IP

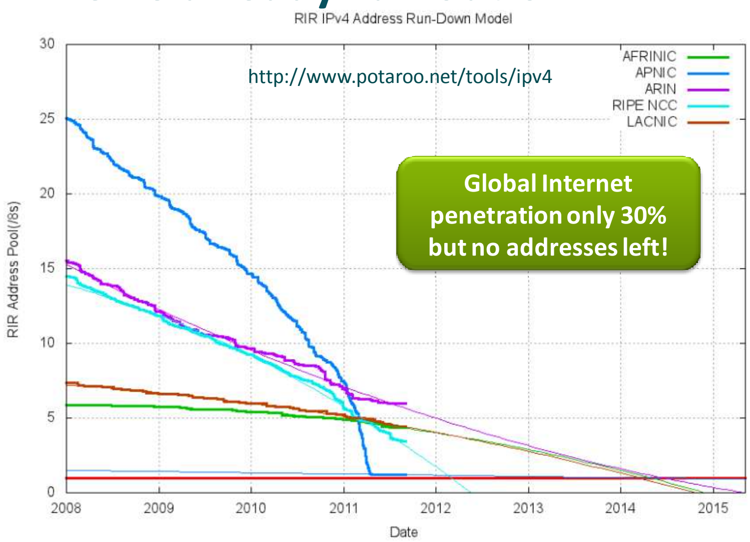
Orbitage
it's about communicating

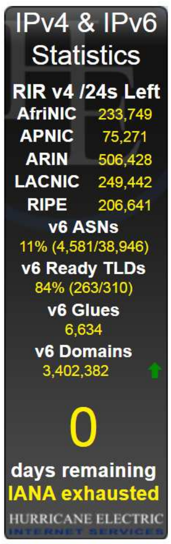
Topics we will address

- Why do programmers need to understand IPv6?
- The IPv6 Protocol Architecture and Operation
- Device/OS support for IPv6
- IPv6 Programming Considerations
- IPv6 socket APIs
- IPv6 Tools and Testing
- Developing IP “Version–Independent” Applications

Orbitage
it's about communicating

We’ve already run out of IPv4





Limitations of IPv4

- Lack of available address space
- No support for mobile hosts
- Non-hierarchical addresses leading to clutter in backbone routers
- Limited support for real-time
- No inherent security mechanisms
- Legacy components in header

IPv6 Specification

- Standardised in 1998
- Design flaws are that it was assumed that systems would gradually migrate to IPv6 while the IPv4 network was still small
  - Did not anticipate the huge growth of IPv4 networks in the mid-2000s
- Main advantages over IPv4 are:

Much larger address pool	Header is simpler	Designed to be extensible	Security is built-in

Major IPv6 Changes

No more NAT

- Previously organisations "hide" behind platforms that perform Network Address Translation (NAT)
- In IPv6, every device is expected to have a public, globally routable IPv6 address, removing the need for NAT

Only public addresses

- With IPv6, private addressing is not supported (officially!)
- Applications/services expected to work with public addresses – should make programming easier (i.e. no need to work with NAT)

Autoconfiguration

- Typically, all addresses will be configured automatically by the network/provider in IPv6
- Simpler configuration mechanisms opens up the market to simpler devices

IPv6 as an Application Enabler

- IPv6 in itself does not enable any new/unavailable applications
- However, since users will now have a globally routable IP address, this opens up possibilities for services
  - There is no NAT to mess things up - removal of NAT makes configuration different
  - Currently, most networks "break" end-to-end connections by converting IP addresses from private to public (often twice!)
  - Applications can be end-to-end – good for applications like voice, gaming, etc.
- Existing and emerging applications will still drive network traffic
  - Challenge for operators is that competition is coming more from Google, Facebook, etc. rather than traditional players
  - Telco/ISP can offer quality & security plus a relationship with customers

Offering seamless service support and smooth transition to IPv6 will offer a major competitive advantage
If you are not pushing IPv6 – be sure your competitors will be!

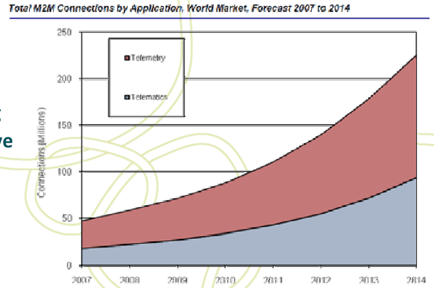
Applications & Services

- Example service areas enhanced by IPv6 include:
  - Applications needing mobility support
    - More of the applications we access are via mobile/wireless devices
    - More applications rely on either the content or the service being on the network – i.e. cloud computing
    - Seamless service access is required and IPv6 has solid support for this
  - Video/audio streaming & interactive services
    - IPv6 includes "multicasting" which allows a server, e.g. IPTV box, to send one traffic stream to multiple users
    - This opens up possibilities for a whole new range of applications based on this one-to-many communication, e.g. social networking, advertising, etc.
  - Applications needing to support large numbers of devices
    - In IPv4, there is a realistic maximum of 16 million addresses that can be supported in a single network
    - IPv6 has a virtually limitless supply of addresses
    - Example: Comcast need to be able to address 20 million customers for IPTV services. Each STB needs 2 addresses, and many homes have more than one STB -> this requires 100 million addresses!

Applications & API Aspects of IPv6 slide 9

Applications & Services

- Machine to machine/monitoring/sensor applications
  - This is arguably one of the biggest growth drivers
    - Devices can easily configure themselves into an IPv6 network
    - Devices will have a globally reachable address
    - Since addresses are plentiful, no restriction on numbers of devices supported
  - Ideal for monitoring anything: e.g. cars, utility meters, household appliances, etc.
  - Examples:
    - Japan using IPv6 for cattle monitoring with a v6 enabled sensor on each cow
    - BMW looking at using IPv6 monitoring for their cars – currently a car may have up to 70 embedded controllers



Source: ABI Research

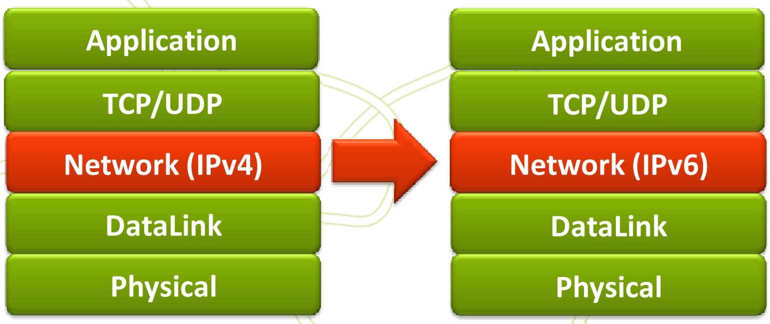
Applications & API Aspects of IPv6 slide 10

Customer Impact

- In general, movement to IPv6 is seen as a network issue rather than a customer issue
  - However, IPv6 is not nearly as widely tested so inevitably there will be issues related to access problems, software malfunctions & security
  - More and more software now requires persistent network connectivity
- In theory, layers above and below IP should be unaffected by migration
- However, any application/service that uses networks needs to be validated that it works with IPv6
- In addition, we need to look at efficiency of how applications work
  - Are there increased access delays?
  - Are the chances of a timeout increased?
  - Can the application recover if one connection doesn't work?

Applications & API Aspects of IPv6 © Orbitage 2012 slide 11

Protocol Stack Comparison



As shown in the stack, only the IP Layer changes <u>but</u>

1. The application may need to modified to support IPv6
2. The TCP/UDP layer may also need some modification

See RFC4038: Application Aspects of IPv6 Transition

Applications & API Aspects of IPv6 © Orbitage 2012 slide 12

Orbitage
it's about communicating

Applications must handle IPv4 &IPv6

- Reality check: Even though IPv4 has run out, we will be running IPv6 in parallel with IPv4 for many years to come
  - We will see progressive spread of IPv6 deployment and increased demand from customers for v6 support
- Need to decide if an application should be IPv4, IPv6 or dual stack aware
  - With wider v6 deployment, applications need to be "independent" i.e. work well with both
- The level of changes required is different for different programming languages/environments
- We need to minimize the impact to the end-user
  - Major issue is introducing latency in network access

Applications & API Aspects of IPv6 © Orbitage 2012 slide 13

Orbitage
it's about communicating

Example application support for IPv6

Application/Server	Category
Apache httpd	Web server
BIND	DNS server
FileZilla	FTP, FTPS and SFTP client
IIS	Web server
Internet Explorer	Web browser
Java	Programming language
Microsoft Outlook	e-mail client
Mozilla Firefox	Web browser
Pidgin	Instant messenger
PuTTY	SSH client
µTorrent	BitTorrent client

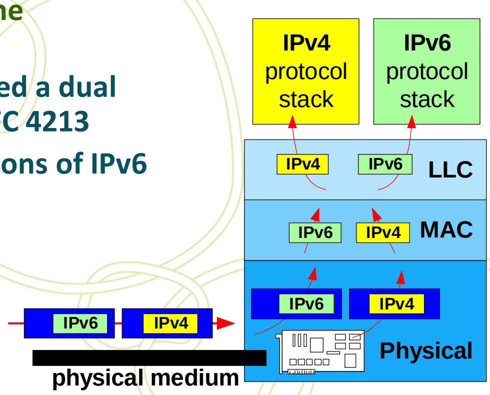
See http://en.wikipedia.org/wiki/Comparison_of_IPv6_application_support

Applications & API Aspects of IPv6 © Orbitage 2012 slide 14

The Dual Stack Environment

- It is relatively easy to write a network stack that supports both IPv4 and IPv6
  - This can share most of the programming code
- This implementation is called a dual stack and is described in RFC 4213
- Most current implementations of IPv6 use a dual stack
- Still required IPv4 address

Note that it does not help with IPv4 address depletion



Current OS Implementations

- All current versions of mainstream operating Systems support IPv6
- Linux has supported it fully for many years
- Windows added optional support in Windows XP SP2
  - However DNS queries use IPv4 only
- Windows Vista & Windows 7 enables IPv6 by default (and provide no easy way to switch it off)
- Some exceptions: VoIP phones, CPE devices

No Assessment of how well they support it

Mobile Device Support

- **Key Mobile Device Platform Support**

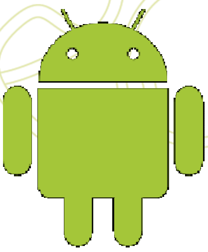
Device Platform	Support
Apple iPhone iOS	Since version 4.0
Google Android	Since version 2.1
Nokia Symbian	Supported since S60 3.2 – early adoption in Nokia 6630 and Nokia 9500 Communicator
Windows Phone	Supposed to support – reports of issues with version 7

- **However, need to be careful of OS issues**
  - **E.g. Apple iOS is not able able to fall back from IPv6 to IPv4 if the initial IPv6 connection attempt fails**

Applications & API Aspects of IPv6 | © Orbitage 2012 | slide 17

Example

- **IPv6 address**



Applications & API Aspects of IPv6 | © Orbitage 2012 | slide 18

Orbitage
it's about communicating

New IPv6 socket APIs

- **IETF standardized two sets of extensions:**

RFC3493 Basic Socket Interface Extensions for IPv6

- Provides standard definitions for Core socket functions, address data structures, Name-to-address translation & address conversion functions

RFC3542 Advanced Sockets Applications Program Interface (API) for IPv6

- Defines interfaces for accessing special IPv6 packet information such as IPv6 header & extension headers
- Advanced APIs are also used to extend the Basic Socket capabilities

Orbitage
it's about communicating

Out with the old, in with the new!

- **The *gethostbyname()* for IPv4 and *gethostbyname2()* function for IPv6 was deprecated and replaced by getaddrinfo() function**
- **Nodename-to-address translation is therefore done in a protocol-independent way**

```
#include <netdb.h>
struct hostent *gethostbyname(const char *name)
```

✗

```
#include <netdb.h>
#include <sys/socket.h>
struct hostent *gethostbyname2(const char *name, int af)
```

✗

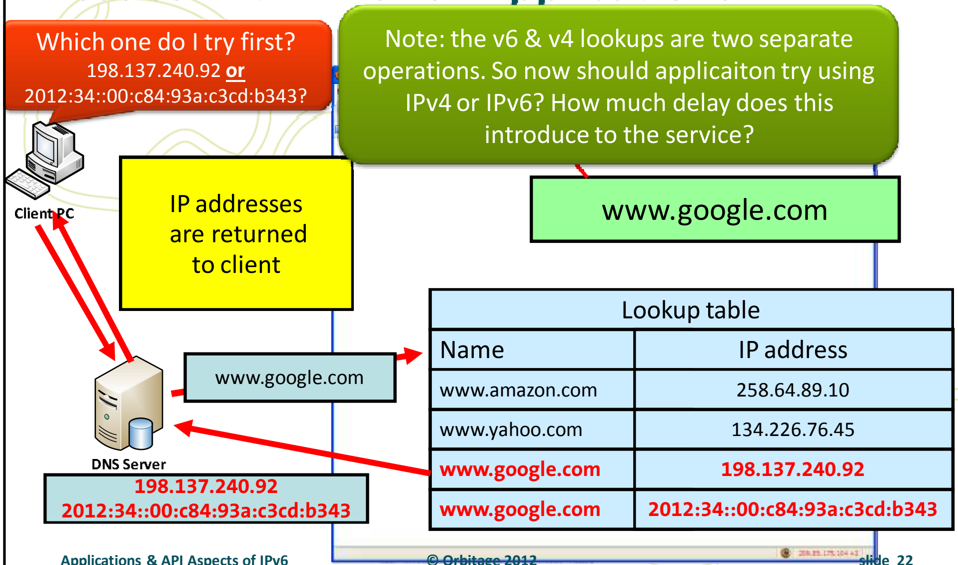
```
#include <netdb.h>
#include <sys/socket.h>
int getaddrinfo(const char *nodename, const char *servname,
const struct addrinfo *hints, struct addrinfo **res);
```

✓

IPv6 and Java

- Good news! Java APIs are already IPv6/v6 compliant
  - IPv6 support in Java is available since 1.4.0 in Solaris/Linux & since 1.5.0 for Windows XP/2003 server
- IPv6 support in Java is automatic and transparent
  - No source code change and no bytecode changes are necessary
- Every Java application is already IPv6 enabled if:
  - It does not use hard-coded addresses
  - All the address or socket information is encapsulated in the Java Networking API
  - It does not use non-specific functions in the address translation

Issues with DNS for Applications



Lookup table	
Name	IP address
www.amazon.com	258.64.89.10
www.yahoo.com	134.226.76.45
www.google.com	198.137.240.92
www.google.com	2012:34::00:c84:93a:c3cd:b343

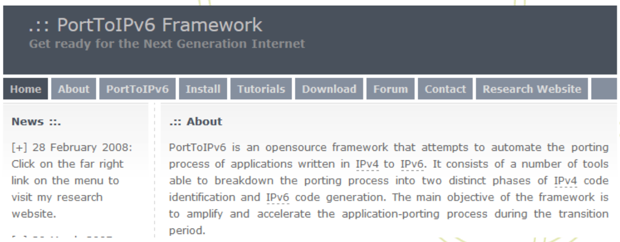
Issues with DNS for Applications

- Asking for an IPv6 and an IPv4 address normally happens serially
- This introduces an extra delay in connecting to a service
  - Many sites have embedded content from other sites, which extends the problem
  - Surveys have shown AAAA/IPv6 lookups taking quite a bit longer than A/IPv4 lookups
- In some cases, the connection may fail/timeout if the DNS doesn't handle the request properly
- From Mozilla Knowledgebase:

> IPv6 was designed in part to solve the problem IPv4 will soon be facing: the exhaustion of all possible IP addresses. Mozilla implemented IPv6 support in early 2000, but that support did not receive widespread testing until recently as IPv6-capable OSs and network software/equipment became more common.
>
> One particular bug that has appeared exists not in Mozilla, but in IPv6-capable DNS servers: an IPv4 address may be returned when an IPv6 address is requested. It is possible for Mozilla to recover from this misinformation, but a significant delay is introduced.
>
> Under certain versions of Mac OS X (those prior to 10.4), this bug is compounded by another bug wherein the OS still makes IPv6 DNS requests even if IPv6 support is disabled. A significant delay is introduced in all connections requiring DNS lookups while the OS and the DNS server exchange unnecessary (or redundant) queries and responses to resolve the address.

Application Development Guidelines

- Use DNS names instead of numerical addresses
  - Also replace hard-coded addresses in applications
- Audit code for IPv6 Address Issues
  - Look for data structures that have only 32 bits instead of 128 bits for an address.
  - Make sure storage/parsing of URI format as strings isn't looking for explicit '.' separators
  - There are several open-source tools to assist with this, e.g.:



Orbitage
it's about communicating

Application Development Guidelines

- Make code independent of address family
  - Applications need to be backward compatible so they can work with both IPv4 and IPv6
  - It should not be the user's responsibility to know which to use.
  - The application should work equally well with v4 only, v4/v6 and v6 only
  - Example, "ping" in Windows 7 will work with either a v4 or v6 address..
- Handle DNS properly
  - According to the IETF, IPv6-capable nodes should perform an AAAA query first and then an A query.
  - Also many DNS servers have IPv4 only connectivity however will still resolve IPv6 addresses, since "transport" and "returned address type" are two different issues.
- Socket connection preferences
  - Applications on devices with dual-stack should prefer v6 if it is possible, i.e. they should try v6 then fall back to v4.
  - Applications should be able to run on a network that uses either v4 or v6 but equally be able to operate on an IPv4-only network

Orbitage
it's about communicating

Simple Example: ping

```
C:\Users\hp>ping 216.218.221.42

Pinging 216.218.221.42 with 32 bytes of data:
Reply from 216.218.221.42: bytes=32 time=79ms TTL=61
Reply from 216.218.221.42: bytes=32 time=79ms TTL=61
Reply from 216.218.221.42: bytes=32 time=84ms TTL=61
Reply from 216.218.221.42: bytes=32 time=78ms TTL=61

Ping statistics for 216.218.221.42:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 78ms, Maximum = 84ms, Average = 80ms

C:\Users\hp>ping 2001:470:35:5be::1

Pinging 2001:470:35:5be::1 with 32 bytes of data:
Request timed out.
Reply from 2001:470:35:5be::1: time=79ms
Reply from 2001:470:35:5be::1: time=82ms
Reply from 2001:470:35:5be::1: time=80ms

Ping statistics for 2001:470:35:5be::1:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
    Minimum = 79ms, Maximum = 82ms, Average = 80ms
```

In Windows 7, works regardless of IP address requested

Simple Example: ping

- Unfortunately, Ubuntu doesn't fare so well

```
jeff@jeff-VirtualBox: ~
File  Edit  View  Search  Terminal  Help
jeff@jeff-VirtualBox:~$ ping 216.218.221.42
PING 216.218.221.42 (216.218.221.42) 56(84) bytes of data.
64 bytes from 216.218.221.42: icmp_req=1 ttl=61 time=81.3 ms
64 bytes from 216.218.221.42: icmp_req=2 ttl=61 time=81.4 ms
64 bytes from 216.218.221.42: icmp_req=3 ttl=61 time=82.0 ms
64 bytes from 216.218.221.42: icmp_req=4 ttl=61 time=81.7 ms
64 bytes from 216.218.221.42: icmp_req=5 ttl=61 time=82.2 ms
^C
--- 216.218.221.42 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4002ms
rtt min/avg/max/mdev = 81.327/81.774/82.207/0.380 ms
jeff@jeff-VirtualBox:~$ ping 2001:470:35:5be::1
ping: unknown host 2001:470:35:5be::1
jeff@jeff-VirtualBox:~$
```


IPv6 Tools and Testing

- Establish an IPv6 test environment for applications
- Some example applications to assist with testing

Netcat
- Simulate network traffic
- Spoofing HTTP headers

Wireshark
- Capture and analyze details of the IPv6 transmission

Virtualization
- Test applications in a sandbox environment before going live

Tunneling, getting access to an IPv6 network
- Try tunneling if you are not able to get native IPv6 connectivity

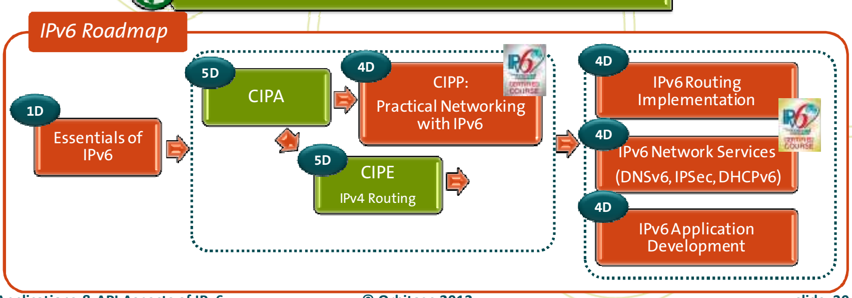
Conclusions

- Ensure you understand IPv6 issues for application development
- Review & audit existing applications for issues
- Make code independent of address families
- Ensure the application works equally well in a v4, v4v6 and v6 environment
- Ensure there is minimal end-user impact
- Test, test, test

Applications & API Aspects of IPv6 © Orbitage 2012 slide 29

IPv6 Training Framework

- Structured Training Pathway
- Hands-on competence based programs
- Comprehensive hands-on assessment
- Endorsed by IPv6 Forum and PSMB



Applications & API Aspects of IPv6 © Orbitage 2012 slide 30


Orbitage
it's about communicating
Thank You
Terima Kasih
谢谢
நன்றி
Many thanks for your time and attention
info@orbitage.com