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 2012
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

We’ve already run out of IPv4

http://www.potaroo.net/tools/ipv4

Global Internet penetration only 30% but no addresses left!
### 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
Essentials of IPv6

Major IPv6 Changes

No more NAT
- Previously organizations “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!

- 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
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?

Protocol Stack Comparison

- As shown in the stack, only the IP Layer changes but
  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 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

Example application support for IPv6

<table>
<thead>
<tr>
<th>Application/Server</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apache httpd</td>
<td>Web server</td>
</tr>
<tr>
<td>BIND</td>
<td>DNS server</td>
</tr>
<tr>
<td>FileZilla</td>
<td>FTP, FTPS and SFTP client</td>
</tr>
<tr>
<td>IIS</td>
<td>Web server</td>
</tr>
<tr>
<td>Internet Explorer</td>
<td>Web browser</td>
</tr>
<tr>
<td>Java</td>
<td>Programming language</td>
</tr>
<tr>
<td>Microsoft Outlook</td>
<td>E-mail client</td>
</tr>
<tr>
<td>Mozilla Firefox</td>
<td>Web browser</td>
</tr>
<tr>
<td>Pidgin</td>
<td>Instant messenger</td>
</tr>
<tr>
<td>PuTTY</td>
<td>SSH client</td>
</tr>
<tr>
<td>µTorrent</td>
<td>BitTorrent client</td>
</tr>
</tbody>
</table>

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

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

- 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

Example

- IPv6 address

```
Network Info II
Device IP: 192.168.0.100
External IP:
Ext. Hostname (L):
Ext. Hostname (R):
Method: cat /proc/net/if_inet6
eth0 IP: 2012:abcd::9a0c:82ff:fe19:7402
eth0 Prefix: /64
eth0 Scope: Global
eth0 IP: fe80:9a0c:82ff:fe19:7402
eth0 Prefix: /64
eth0 Scope: Link local
```
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

Out with the old, in with the new!

- The `gethostbyname()` for IPv4 and `gethostbyname2()` function for IPv6 was deprecated and replaced by `getaddrinfo()` function

- Node name-to-address translation is therefore done in a protocol-independent way

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

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

```c
#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**

Which one do I try first? 198.137.240.92 or 2012:34::00:c84:93a:c3cd:b343?

Note: the v6 & v4 lookups are two separate operations. So now should application try using IPv4 or IPv6? How much delay does this introduce to the service?

<table>
<thead>
<tr>
<th>Lookup table</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td><a href="http://www.amazon.com">www.amazon.com</a></td>
</tr>
<tr>
<td><a href="http://www.yahoo.com">www.yahoo.com</a></td>
</tr>
<tr>
<td><a href="http://www.google.com">www.google.com</a></td>
</tr>
<tr>
<td><a href="http://www.google.com">www.google.com</a></td>
</tr>
</tbody>
</table>
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:

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.:
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

Simple Example: ping

C:\Users\hp>ping 216.218.221.42
Ping 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
Ping 2001:470:35:5be::1 with 32 bytes of data:
Request timed out.
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
Simple Example: ping

• Unfortunately, Ubuntu doesn’t fare so well

![Image of ping output]

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**
  - 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

IPv6 Training Framework

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

IPv6 Roadmap

1D: Essentials of IPv6
2D: CIPA
3D: CIPP: Practical Networking with IPv6
4D: CIPE IPv4 routing
5D: IPv6 Application Development
6D: IPv6 Network Services (DNSv6, IPSec, DHCPv6)
7D: IPv6 Routing Implementation
8D: IPv4 Routing Implementation
Thank You
Terima Kasih

Many thanks for your time and attention

info@orbitage.com