IPv6 Enabling CIFS/SMB Applications

2010 Storage Developers Conference
Santa Clara

Dr David Holder  CEng FIET MIEEE

david.holder@erion.co.uk
http://www.erion.co.uk
Background – Erion

● David Holder
  ● Over twelve years in IPv6
  ● Over twenty years Windows networking
  ● Author
  ● Erion Director
  ● IPv6 enabling Samba

● Erion Ltd
  ● Over twelve years providing IPv6 training and consultancy
  ● World’s leading IPv6 training company
Why is IPv6 so Important Now?

- Ubiquitous support for IPv6
- IPv4 addresses *are* finally running out
- Widespread mandates for IPv6
- Default on Windows Server 2008 & Windows 7
- Default on Linux and Unix
- You may be using IPv6 and don’t know it!
Windows Networking

Pre Active Directory (NT etc)
- NetBIOS
- NBT
- WINS
- SMB

Active Directory
- DNS
- LDAP
- CLDAP
- Kerberos
- SMB/CIFS

IPv6 and Windows Networking

- NetBIOS cannot be IPv6 enabled
- Raw SMB over IPv6 works ✔

<table>
<thead>
<tr>
<th>Port</th>
<th>Protocol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>137</td>
<td>UDP</td>
<td>NBT Name Service</td>
</tr>
<tr>
<td>137</td>
<td>TCP</td>
<td>NBT Name Service</td>
</tr>
<tr>
<td>138</td>
<td>UDP</td>
<td>Datagram service</td>
</tr>
<tr>
<td>138</td>
<td>TCP</td>
<td>Unused</td>
</tr>
<tr>
<td>139</td>
<td>UDP</td>
<td>Unused</td>
</tr>
<tr>
<td>139</td>
<td>TCP</td>
<td>Session Service</td>
</tr>
<tr>
<td>445</td>
<td>TCP</td>
<td>Raw SMB over TCP/IP</td>
</tr>
</tbody>
</table>

Active Directory: DNS, LDAP, CLDAP, Kerberos, SMB/CIFS can operate over IPv6 ✔

NOTE: Active Directory is more than *the sum of the individual protocols*
History of Windows and IPv6

- **1998**: Microsoft Research Stack
- **1999**: IPv6, CIFS Works!
- **2000**: Developer Preview & Components
- **2001**: Production Stack + Components
- **2002**: Advanced Networking Pack
- **2003**: Production Stack + Firewall Teredo and Peer to Peer
- **2004**: Production Stack + Firewall Teredo and Peer to Peer
- **2005**: SP1
- **2006**: Production Stack + Firewall Teredo and Peer to Peer
- **2007**: SP2
- **2008**: Windows Server 2008
- **2009**: Windows Vista
- **2010**: Windows 7

IPv6 Active Directory

**Fully Integrated IPv6 Functionality**

- Windows 2000
- Windows XP
- Windows 2003
- Windows Vista
- Windows 7
- Windows Server 2008

© Erion Ltd 2010 - All rights reserved
## Windows Networking & IPv6

<table>
<thead>
<tr>
<th>Feature</th>
<th>IPv4</th>
<th>IPv6</th>
</tr>
</thead>
<tbody>
<tr>
<td>NBT/NetBIOS</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>WINS</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>NT Domains</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>SMB/CIFS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Windows XP</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Windows 2003</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Active Directory</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Windows Vista</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Windows 7</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Windows Server 2008</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

File Sharing

Including file sharing and *everything*…

© Erion Ltd 2010 - All rights reserved
History of Samba and IPv6

Early versions: IPv6 name resolution possible with Erion Patch

IPv6 support possible with Erion Patch

No IPv6 support (IPv6 support will Not be added)


Windows with Production IPv6 Stack & Components

CIFS Linux Client

IPv6 Support Works

Vista and Windows Server 2008 support for CIFS and AD over IPv6

IPv6 Support Added

Samba 3.2

IPv6 Support Not Recommended

IPv6 Patch

Early versions: IPv6 name resolution possible with Erion Patch

No IPv6 support (IPv6 support will Not be added)
IPv6 PROGRAMMING
History of IPv6 Socket API

Work started on API in IPng Working Group

RFC 2133
Basic Socket Extensions For IPv6

RFC 2292
Advanced Sockets API for IPv6

RFC 2553
Basic Socket Extensions For IPv6

Networking Services (XNS) 5.2 The Open Group

IEEE Std. 1003.1-2001

ISO/IEC 9945:2002

RFC 3493
Basic Socket Extensions For IPv6

RFC 3542
Advanced Sockets API for IPv6

Definitive Standards

API Changes

## Socket API IPv4 vs IPv6 (1)

<table>
<thead>
<tr>
<th>Feature</th>
<th>IPv4 Socket API</th>
<th>IPv6 Socket API</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protocol Independent Name Resolution</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Protocol Independent Address Structure</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Supports IPv4 &amp; IPv6 Protocols</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Supports IPv4 &amp; IPv6 Applications</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Source &amp; binary compatibility for IPv4 Apps</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Protocol Independent Interface Identification</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Thread safe</td>
<td>Depends…</td>
<td>Yes</td>
</tr>
</tbody>
</table>

© Erion Ltd 2010 - All rights reserved
# Socket API IPv4 vs IPv6 (2)

<table>
<thead>
<tr>
<th>Protocol Family</th>
<th>IPv4 only</th>
<th>Dual IPv6 &amp; IPv4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address Family</td>
<td>PF_INET</td>
<td>PF_INET6</td>
</tr>
<tr>
<td>Socket Address Structure</td>
<td>sockaddr_in</td>
<td>sockaddr_in6</td>
</tr>
<tr>
<td>Generic Address Structure</td>
<td>in_addr</td>
<td>in_addr6</td>
</tr>
<tr>
<td>IP Address Structure</td>
<td>gethostbyname</td>
<td>getaddrinfo</td>
</tr>
<tr>
<td>Resolve Name to Address</td>
<td>gethostbyaddr</td>
<td>getnameinfo</td>
</tr>
<tr>
<td>Resolve Address to Name</td>
<td>inet_aton</td>
<td>inet_ntop</td>
</tr>
<tr>
<td>Text to Binary Conversion</td>
<td>inet_ntoa</td>
<td></td>
</tr>
<tr>
<td>Binary to Text Conversion</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

© Erion Ltd 2010 - All rights reserved
Dual Stack and IPv6 API

- New socket API *explicitly* dual stack only
- New API supports *both* IPv6 and IPv4
- IPv6 applications can *also* be IPv4 applications

IPv4 API is subset of IPv6 socket library
Creating a Socket

● Creating an IPv4 socket

```c
socket(PF_INET, SOCK_STREAM, 0); /* TCP socket */
socket(PF_INET, SOCK_DGRAM, 0); /* UDP socket */
```

● Creating an IPv6 (or IPv4) socket

```c
socket(PF_INET6, SOCK_STREAM, 0); /* TCP socket */
socket(PF_INET6, SOCK_DGRAM, 0); /* UDP socket */
```
Passing Addresses to API

**IPv4 Code**

```c
struct sockaddr_in addr;
socklen_t addrlen = sizeof(addr);

/* Put an IPv4 address in addr structure */
bind(sockfd,(struct sockaddr *)&addr, addrlen);
```

**IPv6 Code**

```c
struct sockaddr_in6 addr;
socklen_t addrlen = sizeof(addr);

/* Put an IPv6 address in addr structure */
bind(sockfd,(struct sockaddr *)&addr, addrlen);
```

**Portable Code**

```c
struct sockaddr_storage addr;
socklen_t addrlen = sizeof(addr);

/* Put an IPv4 or IPv6 address in addr structure and set addrlen */
bind(sockfd,(struct sockaddr *)&addr, addrlen);
```
Passing Addresses to Application

IPv4 Code

```c
struct sockaddr_in addr;
socklen_t addrlen = sizeof(addr);
accept(sockfd,(struct sockaddr *)&addr, &addrlen);
```

IPv6 Code

```c
struct sockaddr_in6 addr;
socklen_t addrlen = sizeof(addr);
accept(sockfd,(struct sockaddr *)&addr, &addrlen);
```

Portable Code

```c
struct sockaddr_storage addr;
socklen_t addrlen = sizeof(addr);
accept(sockfd,(struct sockaddr *)&addr, &addrlen);
```
Dual Stack and IPv6 API (1)

IPv6 Application

IPv6 Socket API

Transport (TCP/UDP)

IPv6 Stack

Layer 3

IPv4 Host

192.168.1.3

IPv6 Host

3000::2

IPv6 Socket

SRC = 3000:::1
DST = 3000:::2

IPv6 Socket

SRC = ::FFFF:192.168.1.2
DST = ::FFFF:192.168.1.3

IPv4 Socket

SRC = 192.168.1.3
DST = 192.168.1.2

IPv6 Socket

SRC = 3000::1
DST = 3000::2

IPv6 Socket

SRC = 3000::2
DST = 3000::1

IPv4 Socket

SRC = 192.168.1.3
DST = 192.168.1.2

IPv4-mapped IPv6 addresses

sockaddr_in6 or sockaddr_storage

sockaddr_in6 or sockaddr_storage

sockaddr_in6 or sockaddr_storage

sockaddr_in6 or sockaddr_storage
Dual Stack and IPv6 API (2)

IPv6 Socket
SRC = 3000::1
DST = 3000::2

IPv6 Application

IPv6 Socket API

Transport (TCP/UDP)

DataLink

IPv4 Socket
SRC = 192.168.1.2
DST = 192.168.1.3

IPv4 Host
192.168.1.3

IPv6 Host
3000::2

IPv6 Socket
SRC = 3000::2
DST = 3000::1
Address Conversion

- Conversion from binary representation to textual and vice versa

**IPv4 Code**

To Binary
int inet_aton (const char *cp, struct in_addr *inp);
in_addr_t inet_addr( const char *cp);

To Text
char *inet_ntoa(struct in_addr in);

**IPv6 Code**

To Binary
int inet_pton(int family, const char *src, void *dst);

const char *inet_ntop(int family, const void *src,
                        char *dst, size_t cnt);

- These functions are **not** protocol independent and should be avoided – if possible! Use `getaddrinfo()` and `getnameinfo()`
Textual Address Formats (1)

● Global and Link Local addresses

IPv6 IP Address
Prefix obscured for slide
ddfa:423d:ce73:47
3000::1

Link-local IPv6 Address
fe80::ddfa:423d:ce73:4702%10
fe80::20c:29ff:fe9:b50a%10

IPv6 Default Gateway

Global Addresses
Link local Addresses

IPv6 interfaces have unique interface ID and name

# smbclient -L //fe80::9416:bd6b:8d9c:7490%eth0 -U Administrator

IPv4-mapped IPv6 addresses
::ffff:192.168.1.1
Textual Address Formats (2)

- URLs, URIs and UNCs
  - Use IPv6 in square brackets in URIs and URLs
    
    \[3000:0:20:0:3de2:17ca:d07d:5f10\]

- Not in UNCs (use ipv6-literal.net. names instead)
  
  3000-0-20-0-3de2-17ca-d07d-5f10.ipv6-literal.net.
Name Resolution

- API has evolved over time
- Use `getaddrinfo()` and `getnameinfo()`
- Protocol independent
- Thread safe (some other functions are not)
- Don’t use `getipnodebyname()` or `getipnodebyaddr()`
Windows IPv6 Name Resolution Options

- NetBIOS name resolution
  - IPv4 Only
- WINS
  - IPv4 Only
- Hosts file
  - IPv4 and IPv6
- Link-local Multicast Name Resolution (LLMNR)
  - IPv4 and IPv6
  - Note: Windows Only
- DNS
  - IPv4 and IPv6
- Literal Addresses
  - IPv4 and IPv6
  - Linux/Unix NSS module
Literal Addresses

- In UNCs can use ipv6-literal.net. names
  - 2045-5249-4f4e--1.ipv6-literal.net

- Literal DNS name converts to IPv6 addresses
- Hyphens replace colons in domain name
- s indicates interface (replaces %)
- NSS module nss-ipv6literal provides this on Linux/Unix
Link-local Multicast Name Resolution (LLMNR)

- Performs name resolution without DNS
- DNS over multicast (*not* mDNS)
- Works for IPv4 *and* IPv6 hosts
- Uses multicast addresses
  - IPv6: FF02::1:3
  - IPv4: 224.0.0.252
getaddrinfo() 

- Returns linked list of addrinfo structures
- Allocates memory, free with freeaddrinfo()
getnameinfo()

- Converts address and service into strings
- IPv6 and IPv4
- Argument is socket address structure

```c
error = getnameinfo((struct sockaddr *)&clientaddr,
                    addrlen,
                    clienthost,
                    sizeof(clienthost),
                    clientservice,
                    sizeof(clientservice),
                    NI_NUMERICHOST);

/* handle error here! */

printf("Received request from host=[%s] port=[%s]\n",
        clienthost, clientservice);
```
Coding Choices with IPv6 API

- In IPv4 there is one way to write socket applications
- IPv6 has multiple ways to write same applications
- Large number of options which do you use?

Examples:
- You can write application with *only* IPv6 sockets and addresses and it will usually also support IPv4!
- You can use `sockaddr_in`, `sockaddr_in6` or protocol independent `sockaddr_storage` for IPv4 sockets!
- You can use IPv4 addresses or IPv6 addresses to specify IPv4 end-points! (192.168.1.10 or ::ffff:192.168.1.10)
Learn New IPv6 Features (1)

- **IPv6 Interfaces**
  - Standard method of enumerating interfaces
  - Sometimes you **must** specify interface

- **IPv6 Wildcard Addresses** – two forms
  - `in6addr_any` Used in assignments
  - `IN6ADDR_ANY_INIT` Only used at declaration

- **Loopback Address** – two forms
  - `in6addr_loopback` Used in assignments
  - `IN6ADDR_LOOPBACK_INIT` Only used at declaration
Learn New IPv6 Features (2)

- New socket options
- Multicast
- Special sockets (*IPV6_V6ONLY*)
- Error handling
- New address testing macros
- New constants
IPv6 Code Migration

1. Learn IPv6
2. Modify data structures
3. Change function calls
4. Remove hardcoded addresses
5. Modify the user interface
6. Change some higher-layer protocols
7. Manage dual stack sockets
IPv6 Porting Tools

Many tools for porting applications to IPv6:

- Microsoft checkv4.exe
- Sun IPv6 Socket Scrubber
- HP IPv6 porting assistant
- Open Source PortToIPv6

- Some can automatically change source code
  - Not necessarily a good idea!
  - Don’t always give the correct advice!
Example Samba4 & PortToIPv6

- Reported changes needed in
  - 112 source files
  - 1034 lines
  - Will be many more!

- Notes
  - Some false positives
  - Some false negatives
  - Many changes to addresses in strings not picked up
  - Relatively straightforward changes
  - Socket Wrapper
EXAMPLE SAMBA AND IPV6
Samba 3.x and IPv6

- IPv6 enabled by default
  - Samba 3.2 onwards
- IPv6 transport works!
  - Client and server side functionality over IPv6
  - Join Windows Server 2008 AD domains over IPv6
  - Serve shares and printers over IPv6

IPv6 Samba 3.2 Join to Windows Server 2008 Domain

Wednesday, January 30th, 2008

Yesterday I carried out the first every join of a Samba 3.2 server to a Windows domain over IPv6.

(see: http://www.ipv6consultancy.com/ipv6blog/?p=25)
Linux CIFS and IPv6

- Kernel CIFS module is IPv6 enabled by default
  - Since SambaXP 2007

- `mount.cifs`

```bash
# mount -t cifs //W2008KENT/TESTSHARE /mnt/erion "
  user=Administrator,pass='Pa$$w0rd'
```
Samba 4 and IPv6

- IPv6 *not* enabled by default
  - IPv6 provisioning works with IPv6 address option
- Samba3 & 4 merge will bring IPv6 support
- IPv6 can be enabled with Erion patch
  - See [http://www.ipv6consultancy.com/ipv6blog](http://www.ipv6consultancy.com/ipv6blog)
- With patch IPv6 transport works!
  - IPv6 client and server side functionality
  - IPv6 domain controller functionality
- Join Samba4 domains over IPv6
IPv6 Addresses – Quick Test

- 2045:5249:4f4e:2054:5241:494e:494e:4720
- ::ffff:50.10.1.10
- fe80::1%1
- ff02::2%eth0
- 2001:0000:0102:0304::efff:f6ff:fffe
- 2002:0800:0001::1
- 3ffe:0302:0011:0020:0000:5EFE:0102:0304
- fe80::5EFE:0102:0304

You need to know what these are!
Key Lessons

- Training on IPv6 is very important
- IPv6 does not equal a new version of IPv4
- IPv6 & IPv4 addresses are very different
- Myriad options for migrating code to IPv6
- IPv6 has new features you need to understand
- IPv4 compatibility will complicate things
- Even so migrating code is designed to be easy!
IPv6 and Samba References

- **SambaXP 2008 Presentation**
  - [http://www.ipv6consultancy.com/ipv6blog/?p=34](http://www.ipv6consultancy.com/ipv6blog/?p=34)

- **Google IPv6 Conference 2008 (YouTube)**
  - [http://youtube.com/watch?v=iK0nzdtzjvM](http://youtube.com/watch?v=iK0nzdtzjvM)

- **Google CIFS Workshop Presentation**

- **SambaXP 2007 Presentation**

- **Linux CIFS Client**
  - [http://www.ipv6consultancy.com/ipv6blog/?p=9](http://www.ipv6consultancy.com/ipv6blog/?p=9)

- **Samba4 Hack (old version)**
Erion and IPv6 References

- IPv6 Services
  - http://www.erion.co.uk/ipv6.html
- IPv6 Blog
  - http://www.ipv6consultancy.com/ipv6blog
- IPv6 Training
  - http://www.ipv6training.com
- IPv6 Consultancy
  - http://www.ipv6consultancy.com

Contact david.holder@erion.co.uk
Questions

Thank you for listening