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

BACK GROUND AND HISTORY

Why is IPv6 so Important Now?

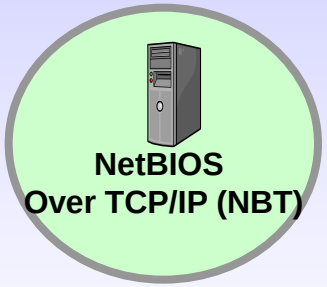
- Ubiquitous support for IPv6
- IPv4 addresses ***are*** finally running out
- Widespread mandates for IPv6
- Default on Windows Server 2008 & Windows 7
- Default on Linux and Unix
- You may be using IPv6 and don't know it!

Erion

IPv6 and Windows Networking

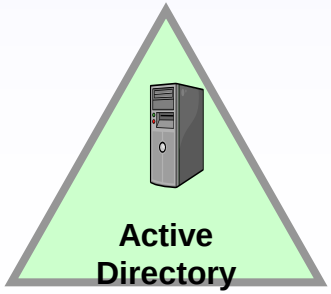
Windows Networking

Pre Active Directory (NT etc)



NetBIOS
NBT
WINS
SMB ✓

Active Directory



DNS ✓
LDAP ✓
CLDAP ✓
Kerberos ✓
SMB/CIFS ✓

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

Port	Protocol	Description
137	UDP	NBT Name Service
137	TCP	NBT Name Service
138	UDP	Datagram service
138	TCP	Unused
139	UDP	Unused
139	TCP	Session Service
445	TCP	Raw SMB over TCP/IP ✓

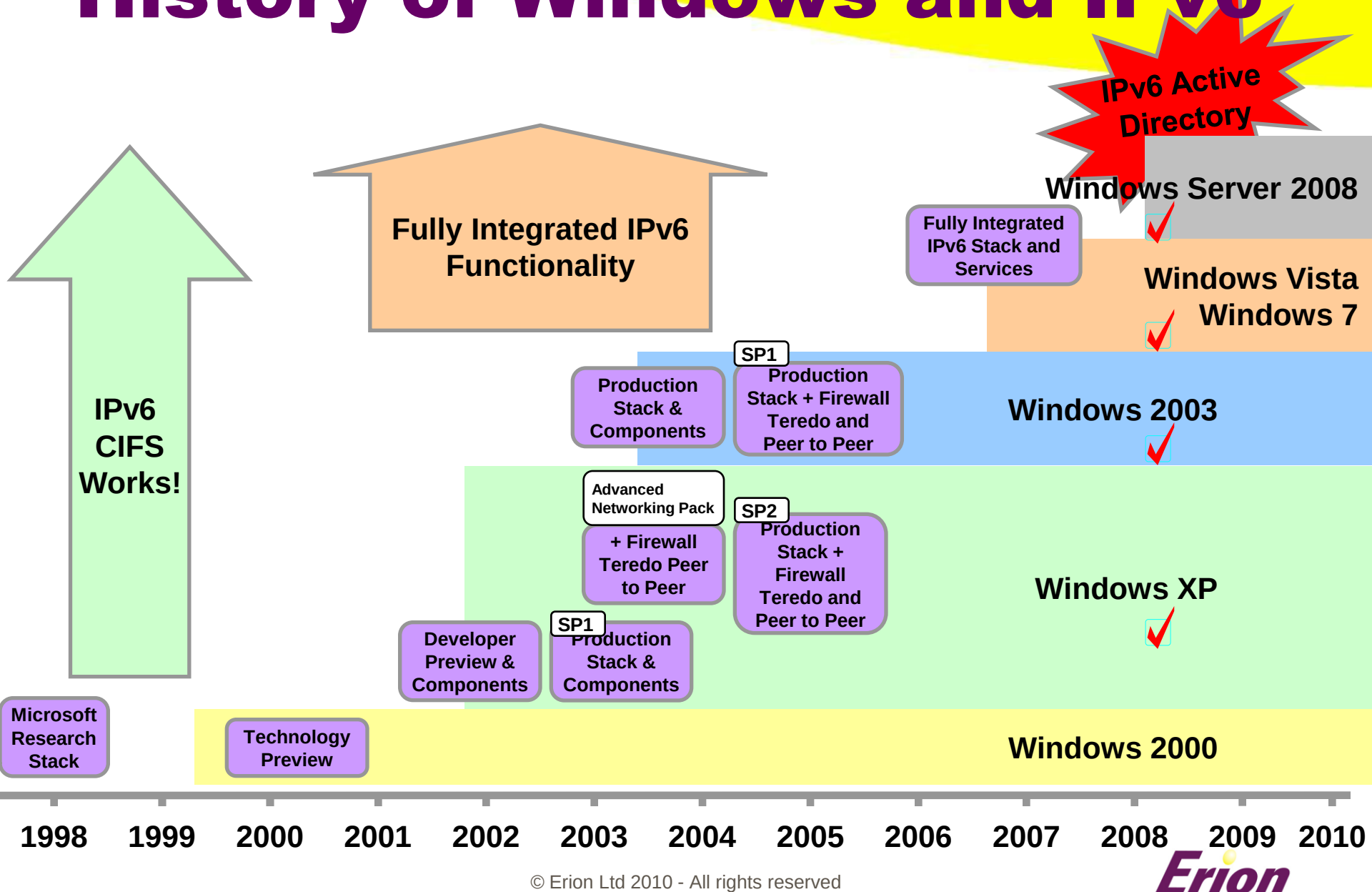
IPv4 Specific

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

NOTE: Active Directory is more than *the sum of the individual protocols*

Erion

History of Windows and IPv6



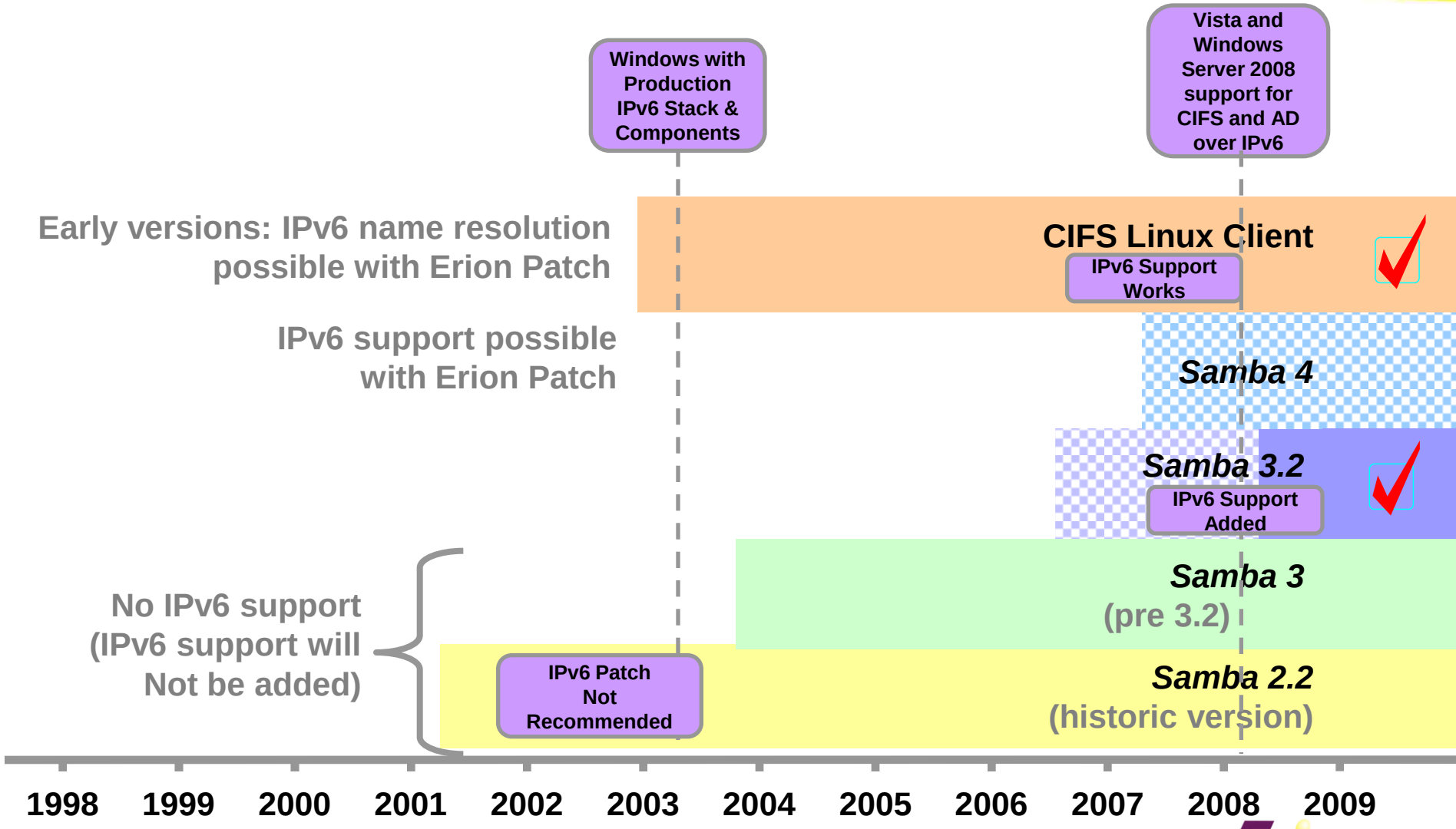
Erion

Windows Networking & IPv6

		IPv4	IPv6
	NBT/NetBIOS	Yes	No
	WINS	Yes	No
	NT Domains	Yes	No
SMB/CIFS File Sharing	Windows XP	Yes	Yes ✓
	Windows 2003	Yes	Yes ✓
Active Directory Including file sharing and *everything…*	Windows Vista	Yes	Yes ✓
	Windows 7	Yes	Yes ✓
	Windows Server 2008	Yes	Yes ✓

Erion

History of Samba and IPv6

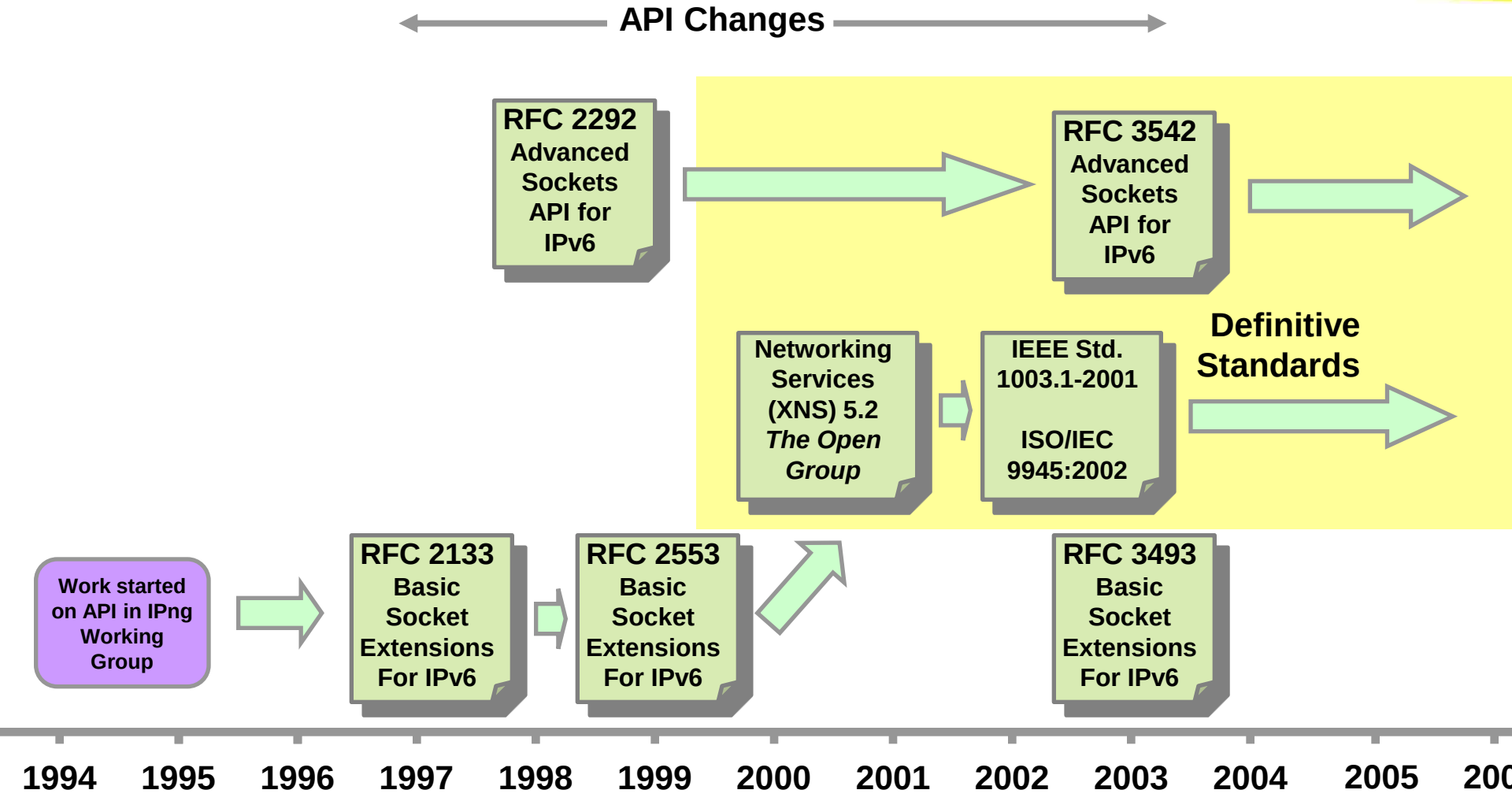


Erion

IPv6 PROGRAMMING

Erion

History of IPv6 Socket API



Erion

Socket API IPv4 vs IPv6 (1)

	IPv4 Socket API	IPv6 Socket API
Protocol Independent Name Resolution	No	Yes
Protocol Independent Address Structure	No	Yes
Supports IPv4 & IPv6 Protocols	No	Yes
Supports IPv4 & IPv6 Applications	No	Yes
Source & binary compatibility for IPv4 Apps	Yes it is IPv4!	Yes
Protocol Independent Interface Identification	No	Yes
Thread safe	Depends…	Yes

Erion

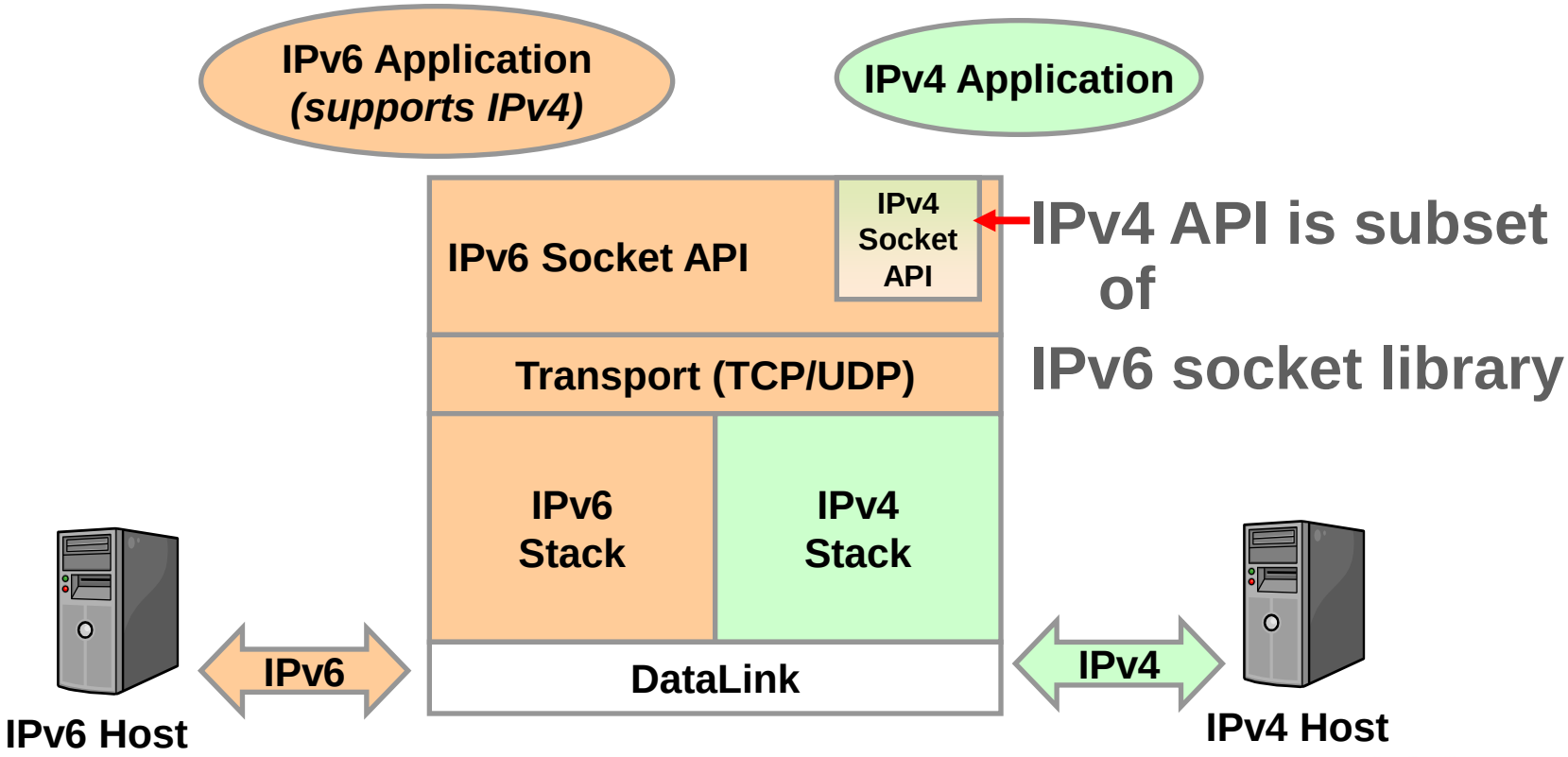
Socket API IPv4 vs IPv6 (2)

	IPv4 only	Dual IPv6 & IPv4
Protocol Family	`PF_INET`	`PF_INET6`
Address Family	`AF_INET`	`AF_INET6`
Socket Address Structure	`sockaddr_in`	`sockaddr_in6`
Generic Address Structure		`sockaddr_storage`
IP Address Structure	`in_addr`	`in_addr6`
Resolve Name to Address	`gethostbyname`	`getaddrinfo`
Resolve Address to Name	`gethostbyaddr`	`getnameinfo`
Text to Binary Conversion	`inet_aton`	`inet_pton`
Binary to Text Conversion	`inet_ntoa`	`inet_ntop`

Erion

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



Erion

Creating a Socket

- Creating an IPv4 socket

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

- Creating an IPv6 (*or IPv4*) socket

```
socket(PF_INET6, SOCK_STREAM, 0); /* TCP socket */
socket(PF_INET6, SOCK_DGRAM,  0); /* UDP socket */
```


Erion

Passing Addresses to API

IPv4 Code

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

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

IPv6 Code

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

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

IPv6 & IPv4

Portable Code

```
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);
```

IPv6 & IPv4

Erion

Passing Addresses to Application

IPv4 Code

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

accept(sockfd,(struct sockaddr *)&addr, &addrlen);
```

IPv6 Code

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

accept(sockfd,(struct sockaddr *)&addr, &addrlen);
```

IPv6 & IPv4

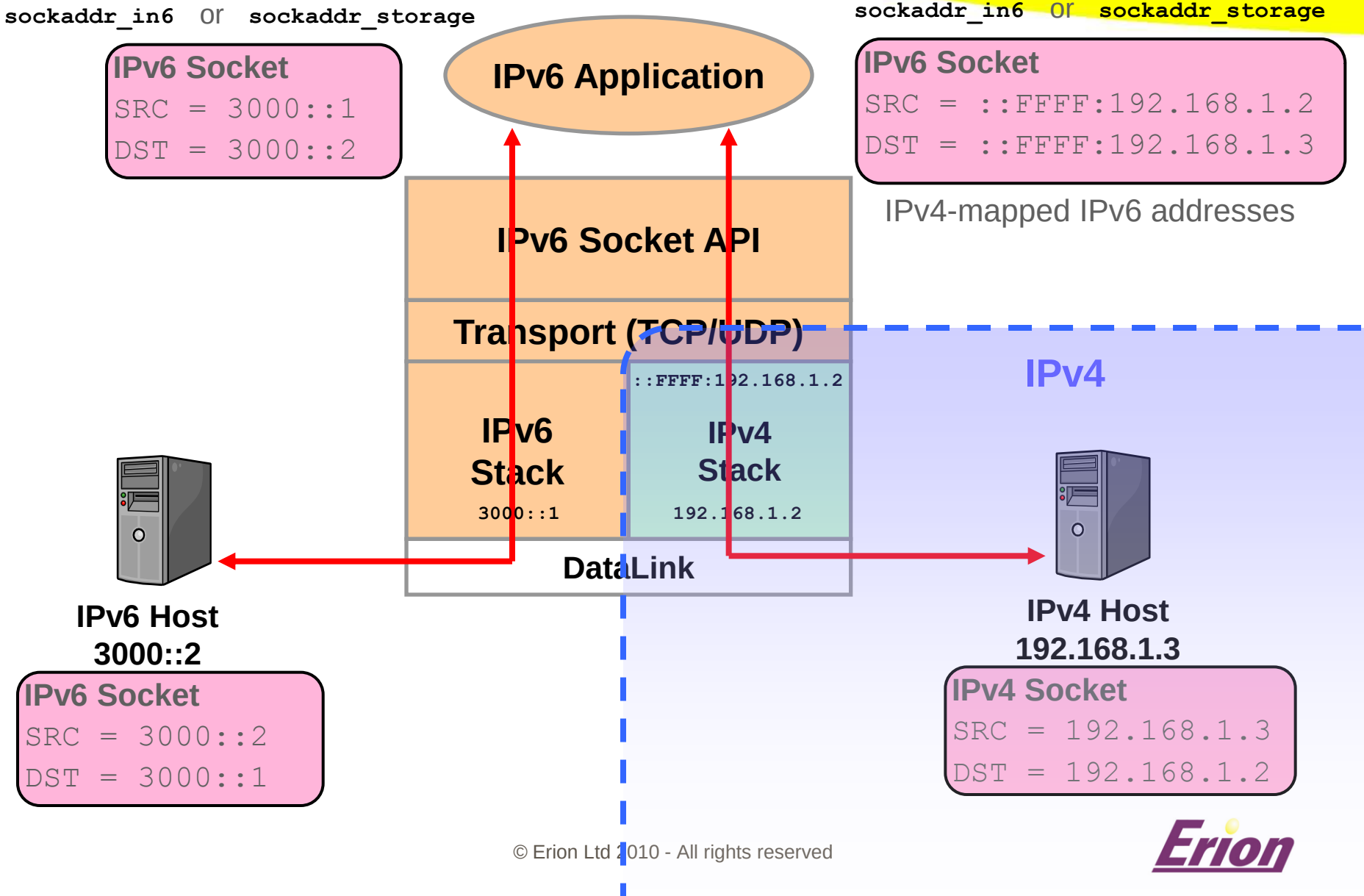
Portable Code

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

accept(sockfd,(struct sockaddr *)&addr, &addrlen);
```

IPv6 & IPv4

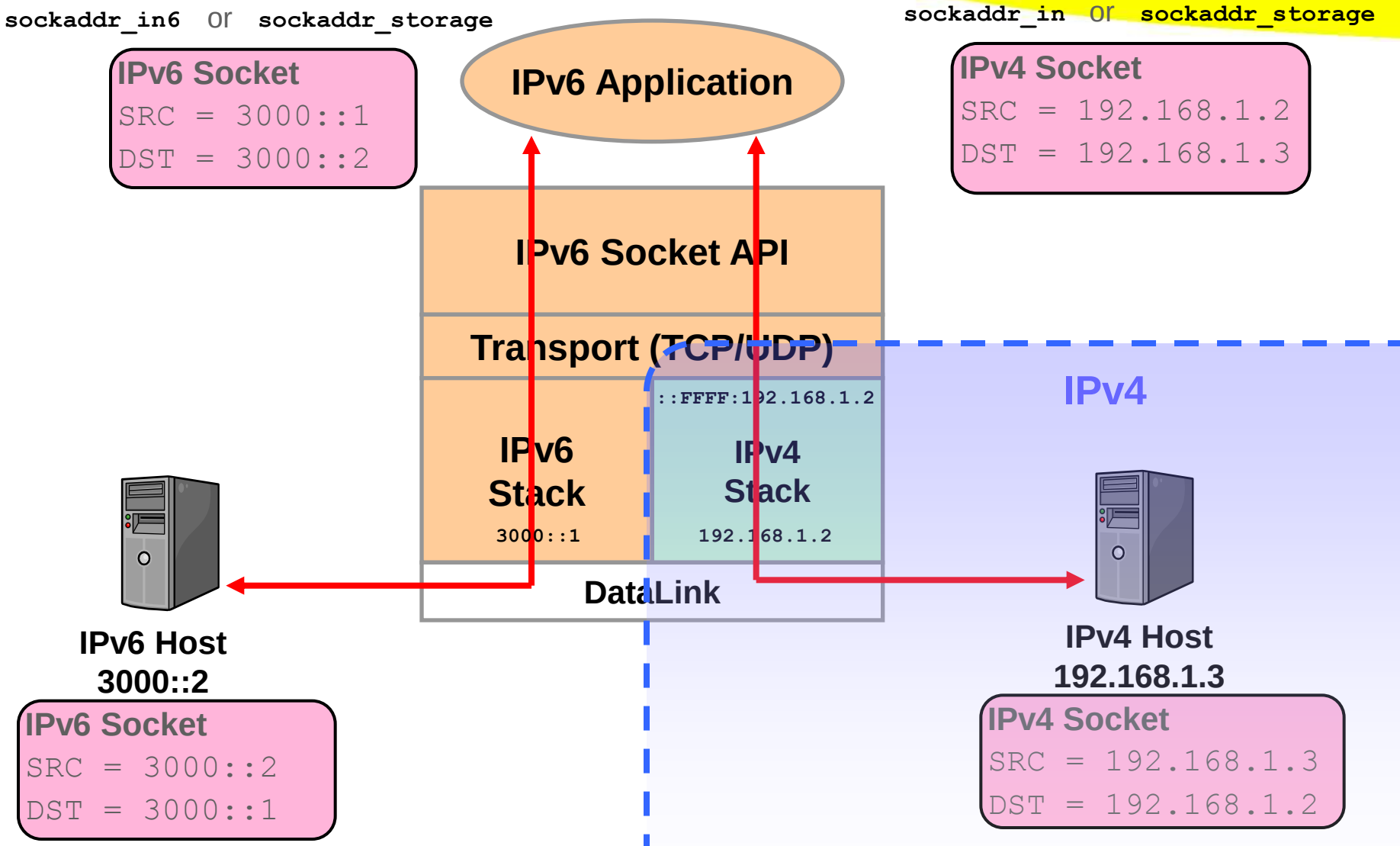
Erion

Dual Stack and IPv6 API (1)



Erion

Dual Stack and IPv6 API (2)



Erion

Address Conversion

- Conversion from binary representation to textual and vice versa

IPv4 Code

To Binary

To Text

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

char        *inet_ntoa(struct in_addr in);
```

IPv6 Code

To Binary

To Text

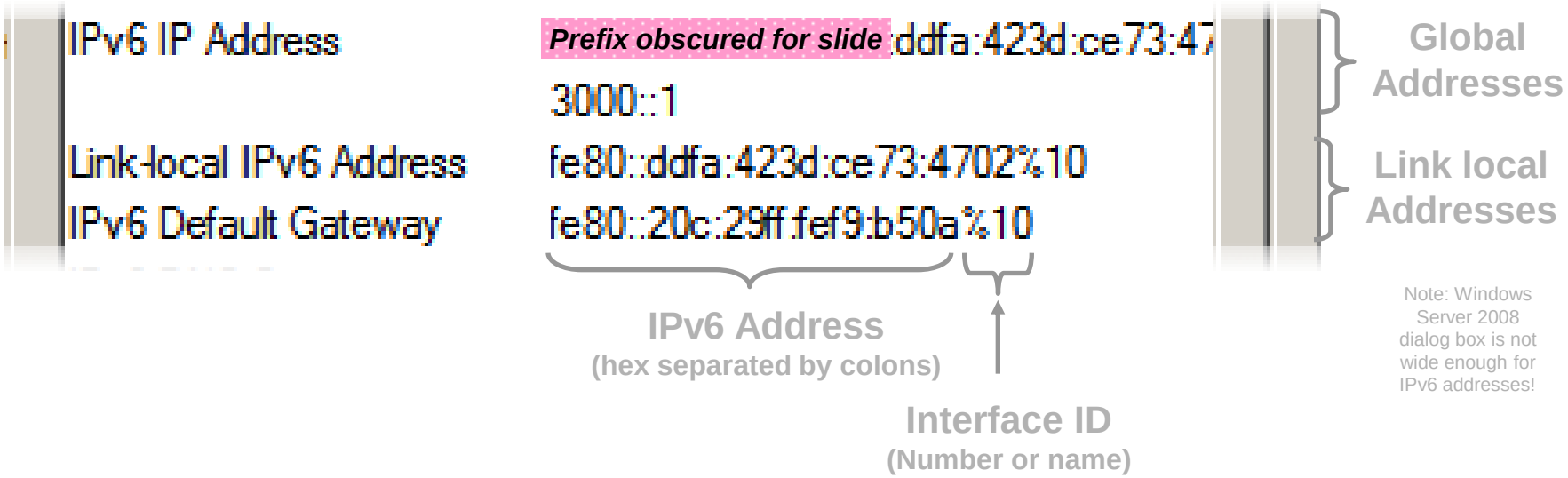
```
int inet_pton(int family, const char *src, void *dst);

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()`**

Erion

Textual Address Formats (1)

- Global and 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

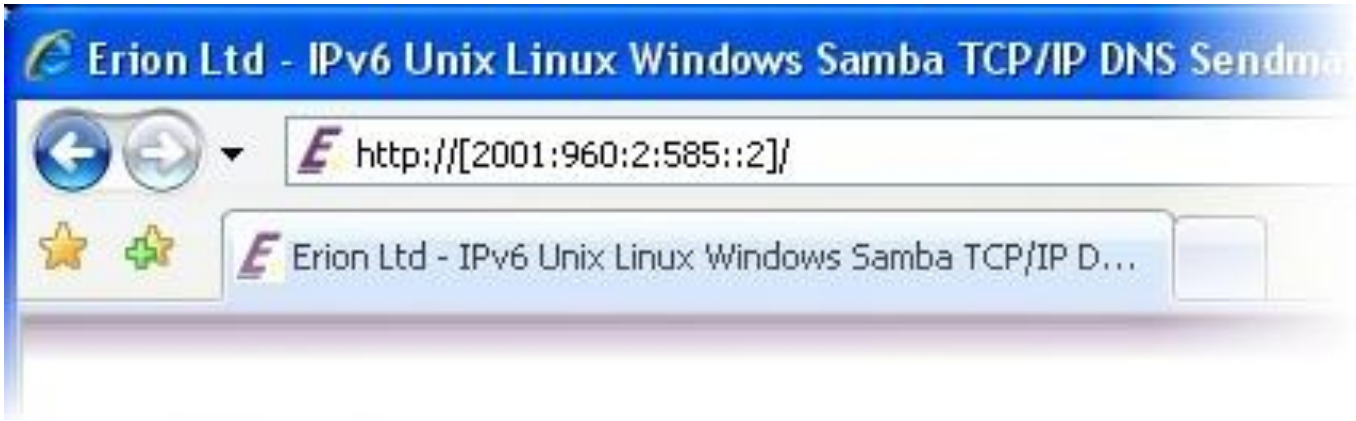
  `::ffff:192.168.1.1`

Erion

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)

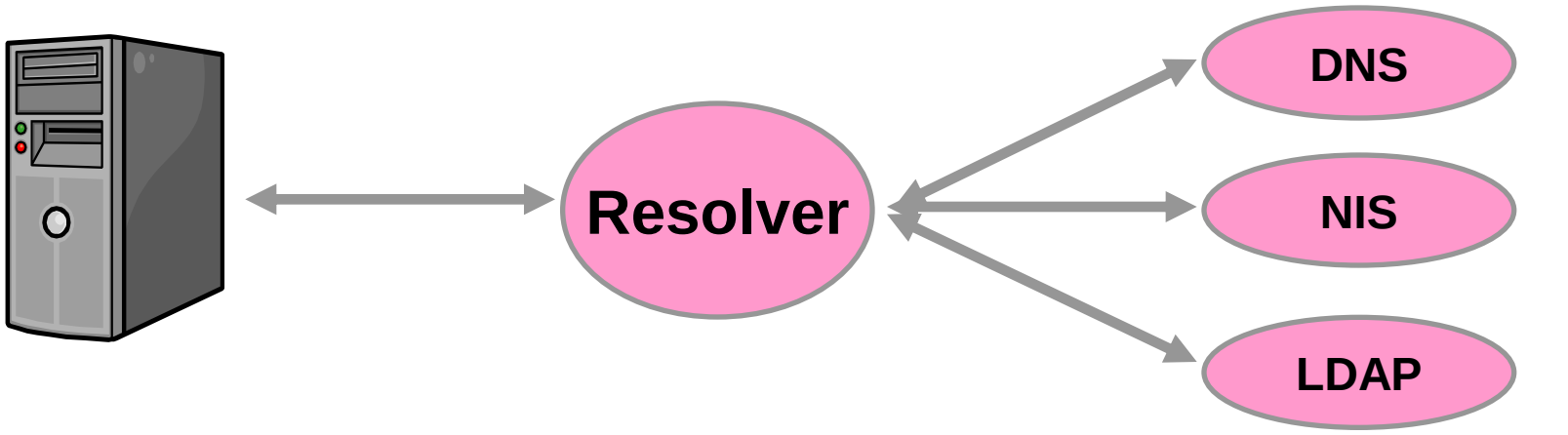
3000-0-20-0-3de2-17ca-d07d-5f10.ipv6-literal.net.

```
TCP    [3000:0:20:0:3de2:17ca:d07d:5f10]:49165  [3000:0:20:0:20c:29ff:fef1:925b]:445  ESTABLISHED
```


Erion

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()`**



Erion

Windows IPv6 Name Resolution Options

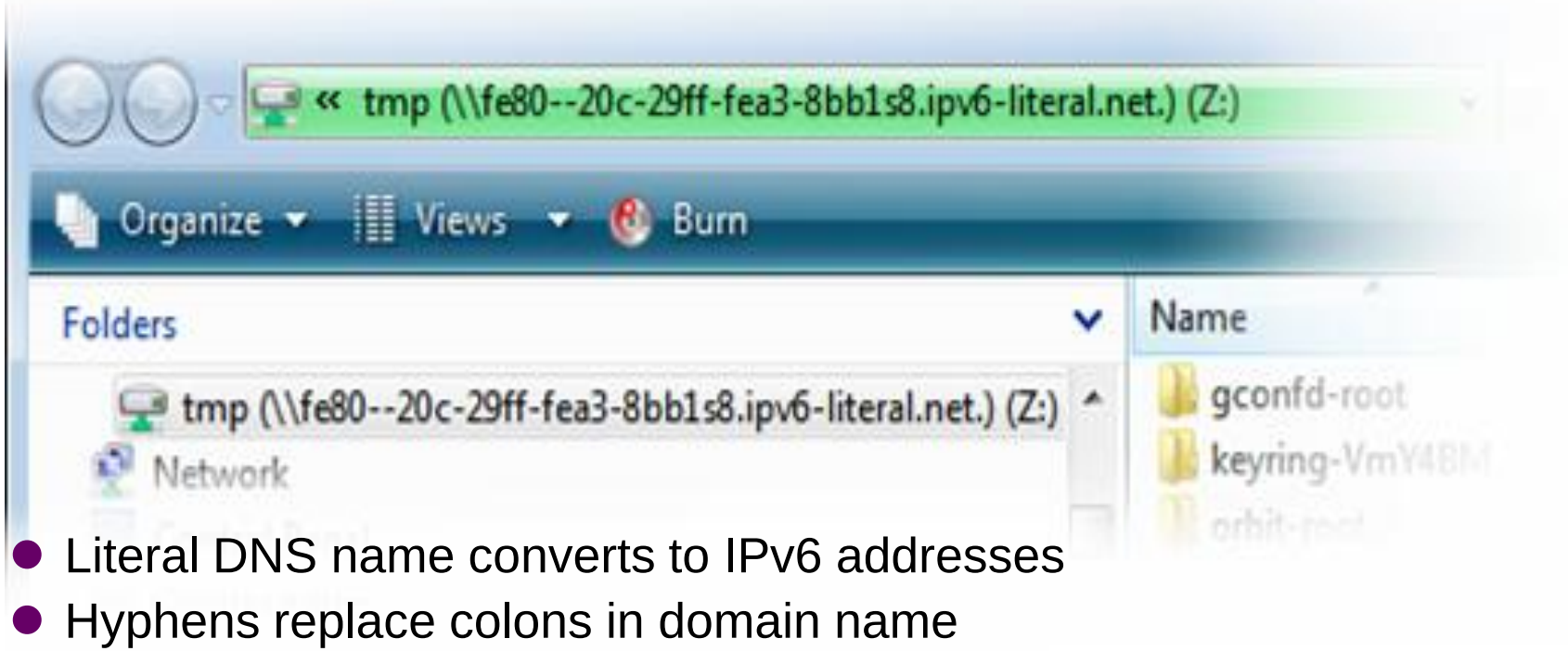
- NetBIOS name resolution — **IPv4 Only**
- WINS — **IPv4 Only**
- Hosts file — **IPv4 and IPv6** 
- Link-local Multicast Name Resolution (LLMNR) — **IPv4 and IPv6** 
  Note: Windows Only
- DNS — **IPv4 and IPv6** 
- Literal Addresses — **IPv4 and IPv6** 
  Linux/Unix NSS module

Erion

Literal Addresses

- In UNCs can use ipv6-literal.net. names

  `2045-5249-4f4e--1.ipv6-literal.net`

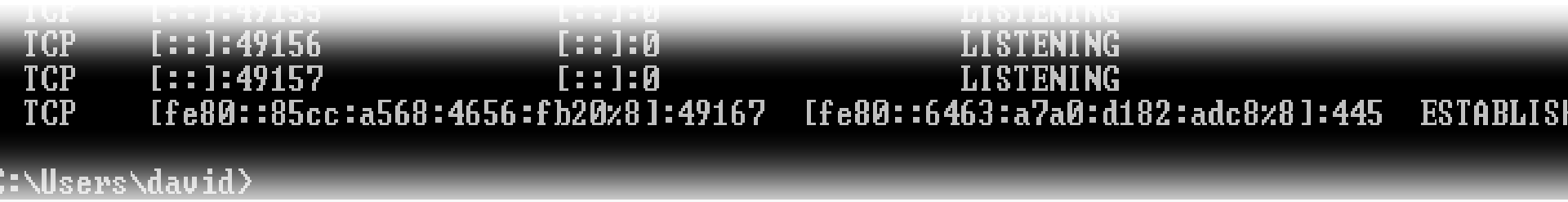


  - Literal DNS name converts to IPv6 addresses
  - Hyphens replace colons in domain name
  - **s** indicates interface (replaces %)
- NSS module nss-ipv6literal provides this on Linux/Unix

Erion

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**

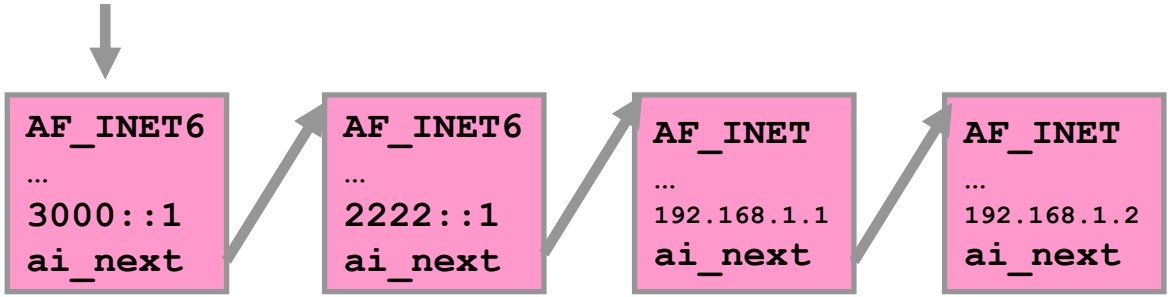


Erion

getaddrinfo()

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

getaddrinfo()



freeaddrinfo()

```
struct addrinfo {
    int     ai_flags;           /* AI_PASSIVE, AI_CANONNAME */
    int     ai_family;          /* AF_UNSPEC, AF_INET, AF_INET6 */
    int     ai_socktype;        /* SOCK_STREAM, SOCK_DGRAM ... */
    int     ai_protocol;        /* IPPROTO_IP, IPPROTO_IPV6 */
    size_t  ai_addrlen;         /* length of ai_addr */
    struct  *sockaddr ai_addr;  /* socket address structure */
    char    *ai_canonname;      /* cannonical name */
    struct  *addrinfo ai_next;  /* next addrinfo structure */
};
```


Erion

getnameinfo()

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

```
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);
```


Erion

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)

Erion

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

Erion

Learn New IPv6 Features (2)

- New socket options
- Multicast
- Special sockets (`IPV6_V6ONLY`)
- Error handling
- New address testing macros
- New constants

Erion

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

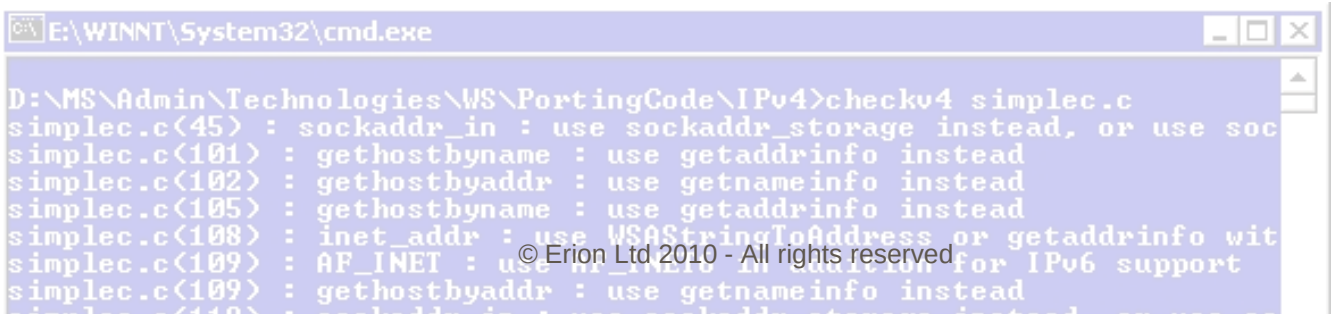
Erion

IPv6 Porting Tools

Many tools for porting applications to IPv6:

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

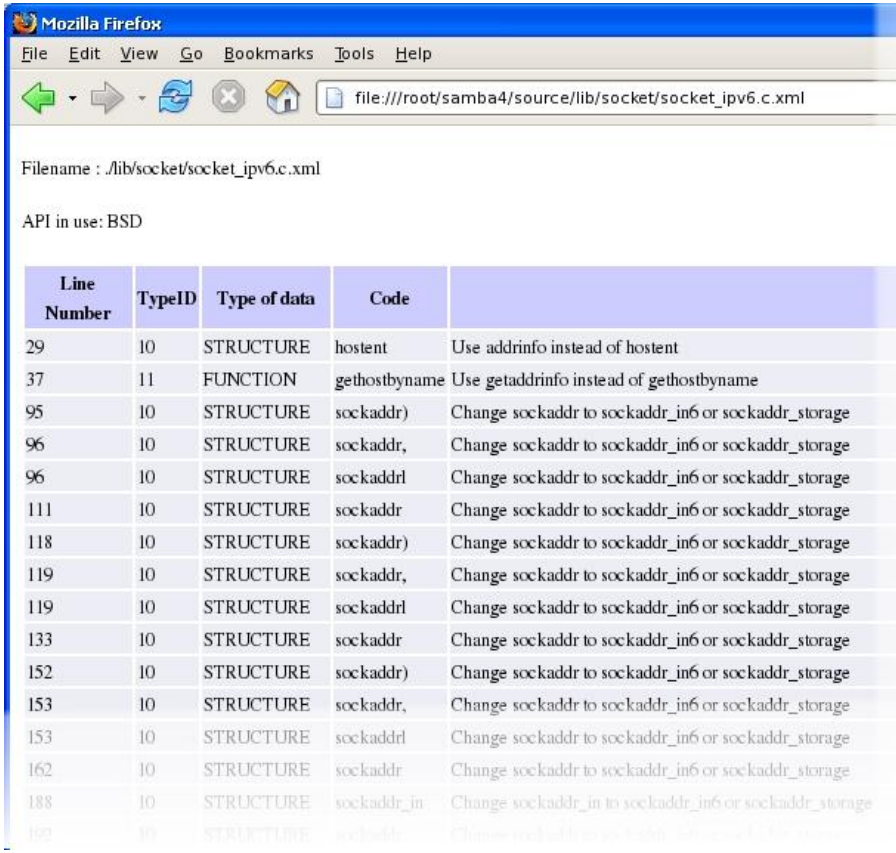
- Some can automatically change source code
  - Not necessarily a good idea!
  - Don't always give the correct advice!



Erion

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



Line Number	TypeID	Type of data	Code	
29	10	STRUCTURE	hostent	Use addrinfo instead of hostent
37	11	FUNCTION	gethostbyname	Use getaddrinfo instead of gethostbyname
95	10	STRUCTURE	sockaddr)	Change sockaddr to sockaddr_in6 or sockaddr_storage
96	10	STRUCTURE	sockaddr,	Change sockaddr to sockaddr_in6 or sockaddr_storage
96	10	STRUCTURE	sockaddrl	Change sockaddr to sockaddr_in6 or sockaddr_storage
111	10	STRUCTURE	sockaddr	Change sockaddr to sockaddr_in6 or sockaddr_storage
118	10	STRUCTURE	sockaddr)	Change sockaddr to sockaddr_in6 or sockaddr_storage
119	10	STRUCTURE	sockaddr,	Change sockaddr to sockaddr_in6 or sockaddr_storage
119	10	STRUCTURE	sockaddrl	Change sockaddr to sockaddr_in6 or sockaddr_storage
133	10	STRUCTURE	sockaddr	Change sockaddr to sockaddr_in6 or sockaddr_storage
152	10	STRUCTURE	sockaddr)	Change sockaddr to sockaddr_in6 or sockaddr_storage
153	10	STRUCTURE	sockaddr,	Change sockaddr to sockaddr_in6 or sockaddr_storage
153	10	STRUCTURE	sockaddrl	Change sockaddr to sockaddr_in6 or sockaddr_storage
162	10	STRUCTURE	sockaddr	Change sockaddr to sockaddr_in6 or sockaddr_storage
188	10	STRUCTURE	sockaddr_in	Change sockaddr_in to sockaddr_in6 or sockaddr_storage

Erion

EXAMPLE SAMBA AND IPV6

Erion

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

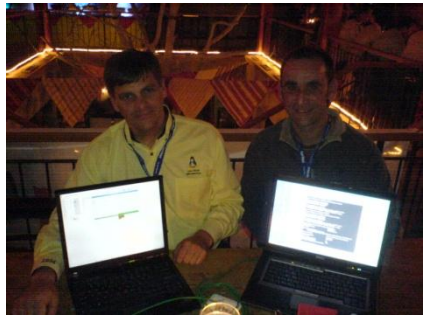
Yesterday I carried out the first every join of a Samba 3.2 server to a Windows domain over IPv6.

(see: http://www.ipv6consultancy.com/ipv6blog/?p=25)

Erion

Linux CIFS and IPv6

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



Steve French (IBM) and David Holder (Erion) The first ever CIFS client connection over IPv6

- **mount.cifs**

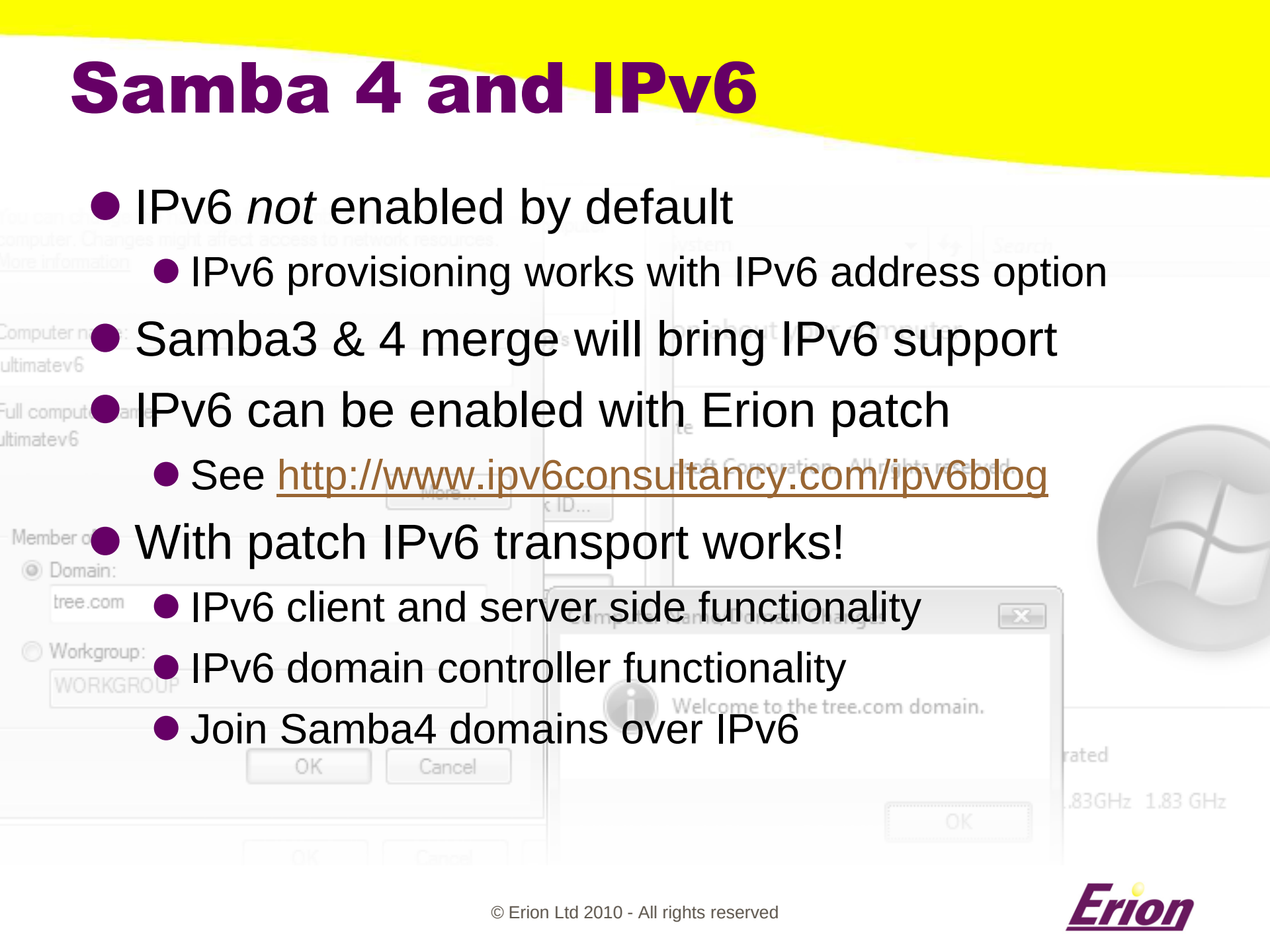
```
# mount -t cifs //W2008KENT/TESTSHARE /mnt/erion \
      user=Administrator,pass='Pa$$w0rd'
```

```
Local Address          Foreign Address        State
[2a01:348:13e:0:fc6f:d88f:6507:4ad]:445  [2a01:348:13e:0:20c:29ff:fea0:3883]:35906  ESTABLISHED

\Administrator>_
```

Erion

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
- With patch IPv6 transport works!
  - IPv6 client and server side functionality
  - IPv6 domain controller functionality
  - Join Samba4 domains over IPv6



Erion

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!

Erion

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!

Erion

IPv6 and Samba References

- SambaXP 2008 Presentation
  - http://www.ipv6consultancy.com/ipv6blog/?p=34
- Google IPv6 Conference 2008 (YouTube)
  - http://youtube.com/watch?v=iK0nzdtzjvM
- Google CIFS Workshop Presentation
  - http://www.ipv6consultancy.com/ipv6blog/?p=21
- SambaXP 2007 Presentation
  - http://www.sambaxp.org/files/SambaXP2007-PDF/Holder-SambaVistawithIPv6V2.pdf
  - http://www.ipv6consultancy.com/ipv6blog/?p=8
- Linux CIFS Client
  - http://www.ipv6consultancy.com/ipv6blog/?p=9
- Samba4 Hack (*old version*)
  - http://www.ipv6consultancy.com/ipv6blog/?p=12

Erion

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

Erion

Questions

Thank you for listening