Preparing Samba for Windows Server 2008 and IPv6

Dr David Holder CEng FIET MIEEE
david.holder@erion.co.uk

Preparing Samba for IPv6

- Status of IPv6 and Windows
- Review of IPv6 and Samba 3/4
- Coding for IPv6
- Porting Applications to IPv6
- Code Analysis of Samba 3
- Code Analysis of Samba 4
- Recommended Way Forward
- Discussion

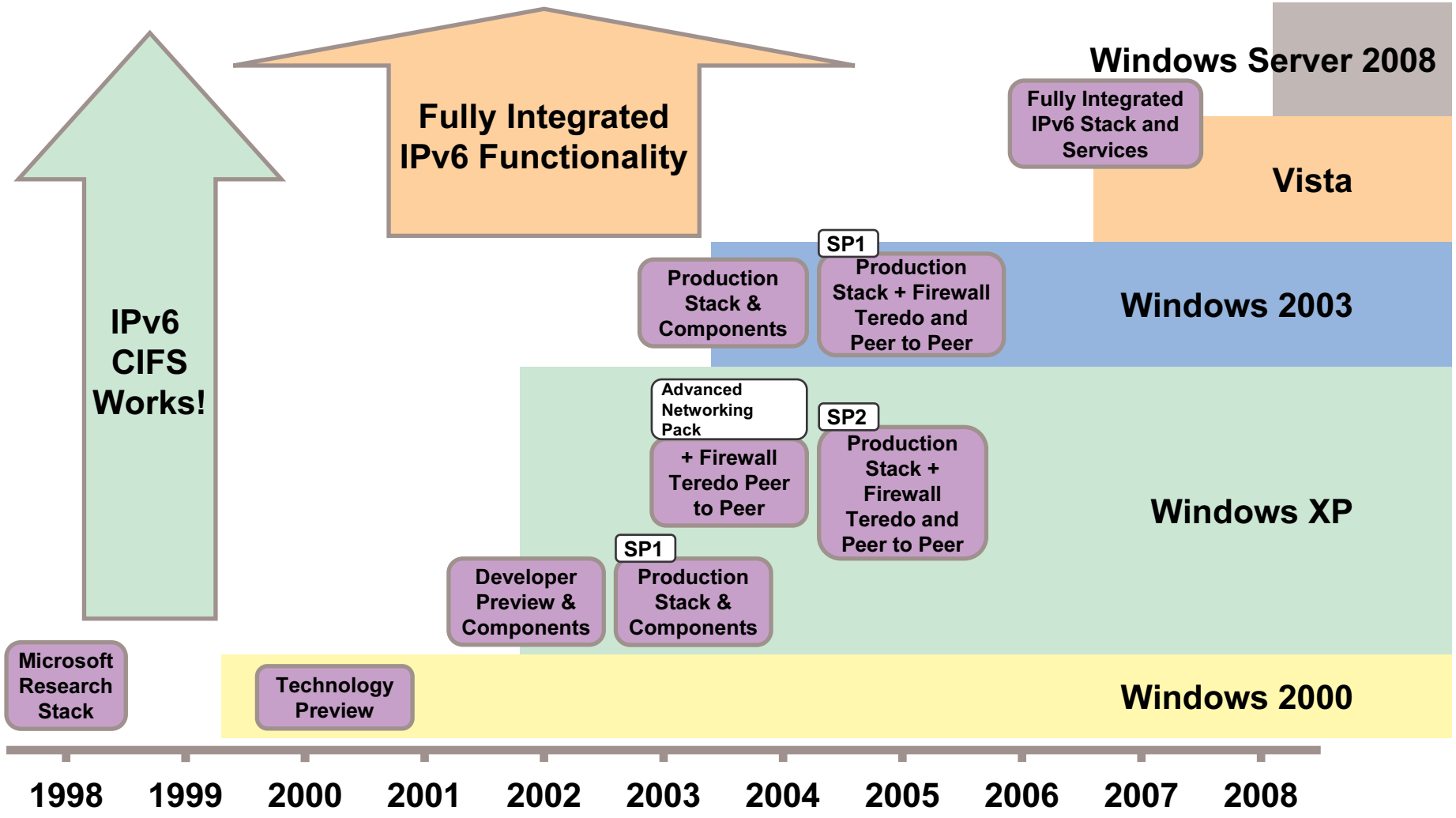
Erion

Quick Comparison IPv4 vs IPv6

	IPv4	IPv6
Address space	Scarce	Huge
End to end connectivity	Broken (NAT)	Yes
Stateless auto-configuration	No	Yes
IPSec standard	No	Yes
IP mobility	Impractical	Yes
Network renumbering	Difficult	Easy
Peer to peer applications	Difficult	Easy

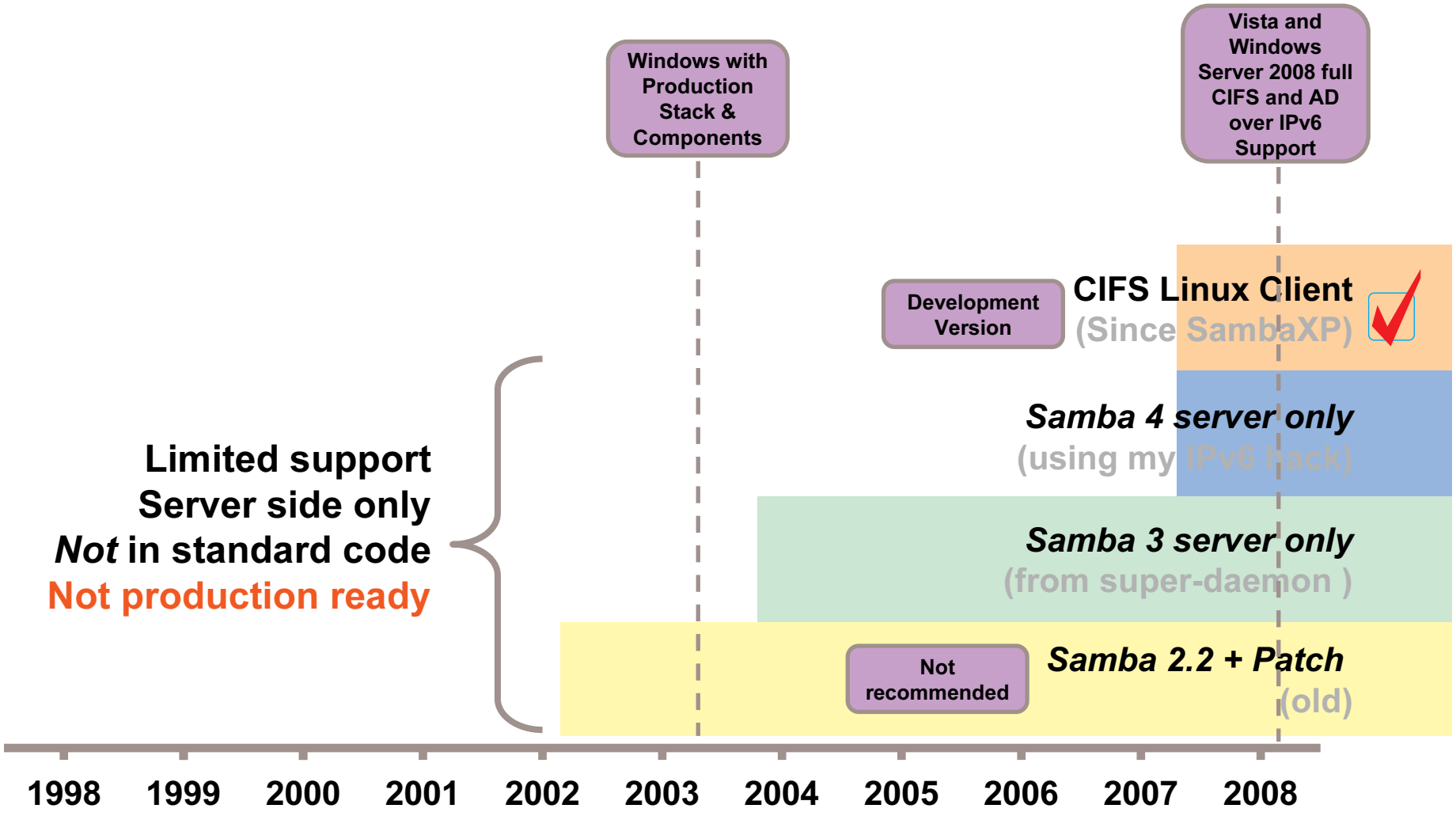
Erion

History of Windows and IPv6



Erion

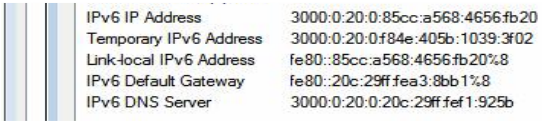
History of Samba and IPv6



Erion

IPv6 & Vista/Windows Server 2008

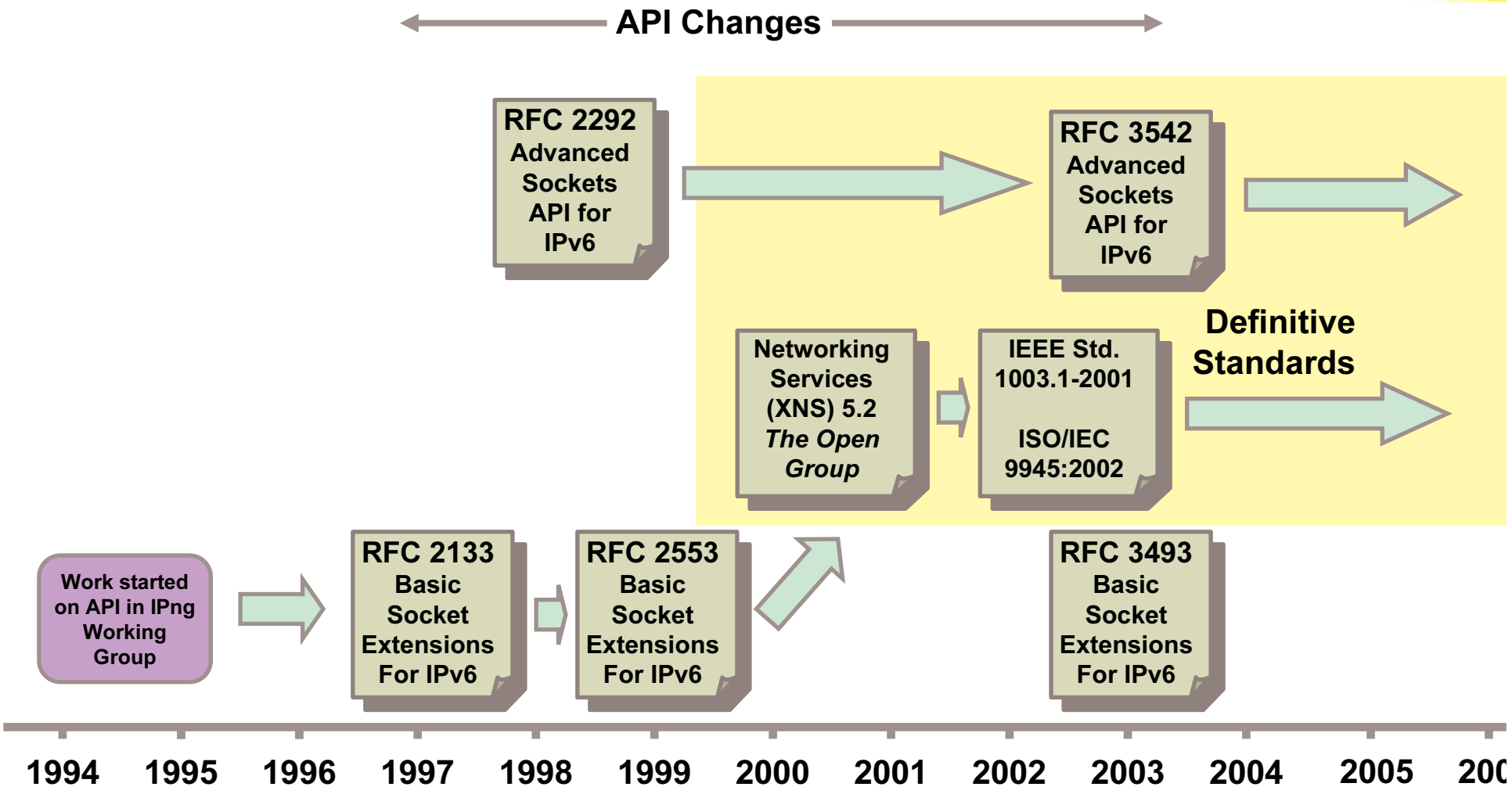
- Vista & Longhorn will introduce IPv6 into many networks
  - Enabled by default
  - Preferred protocol
  - Configured automatically



- IPv6 by stealth
  - Organisation implements Longhorn and/or Vista and uses IPv6 by default
  - Transition mechanisms enable IPv6 on IPv4 only networks
- IPv6 by design
  - Organisation implements Longhorn and/or Vista as a part of strategic plan to move to IPv6

Erion

History of IPv6 Socket API



Erion

Socket API IPv4 vs IPv6

	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 

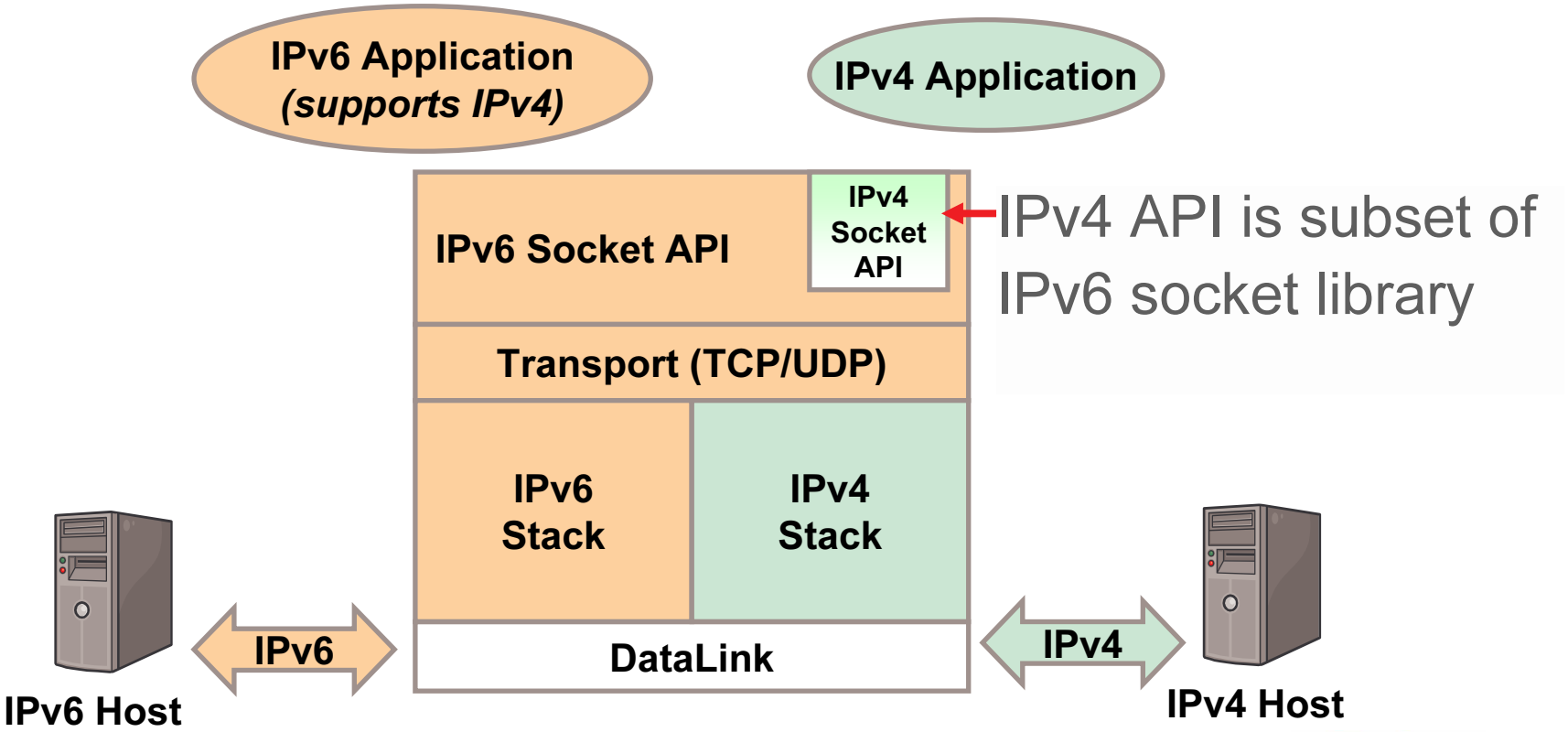
Erion

Changes to the Socket API

- **Core socket functions**
  - Transport independent – no change
  - Support for new address structure
- **Address data structures**
  - New structure for IPv6 addresses
  - New socket address structure for IPv6 sockets
  - New protocol independent socket address structure
- **Name to address translation functions**
  - New protocol independent functions
- **Address conversion functions**
  - New functions that support IPv4 and IPv6
- **New functions to support new IPv6 features**

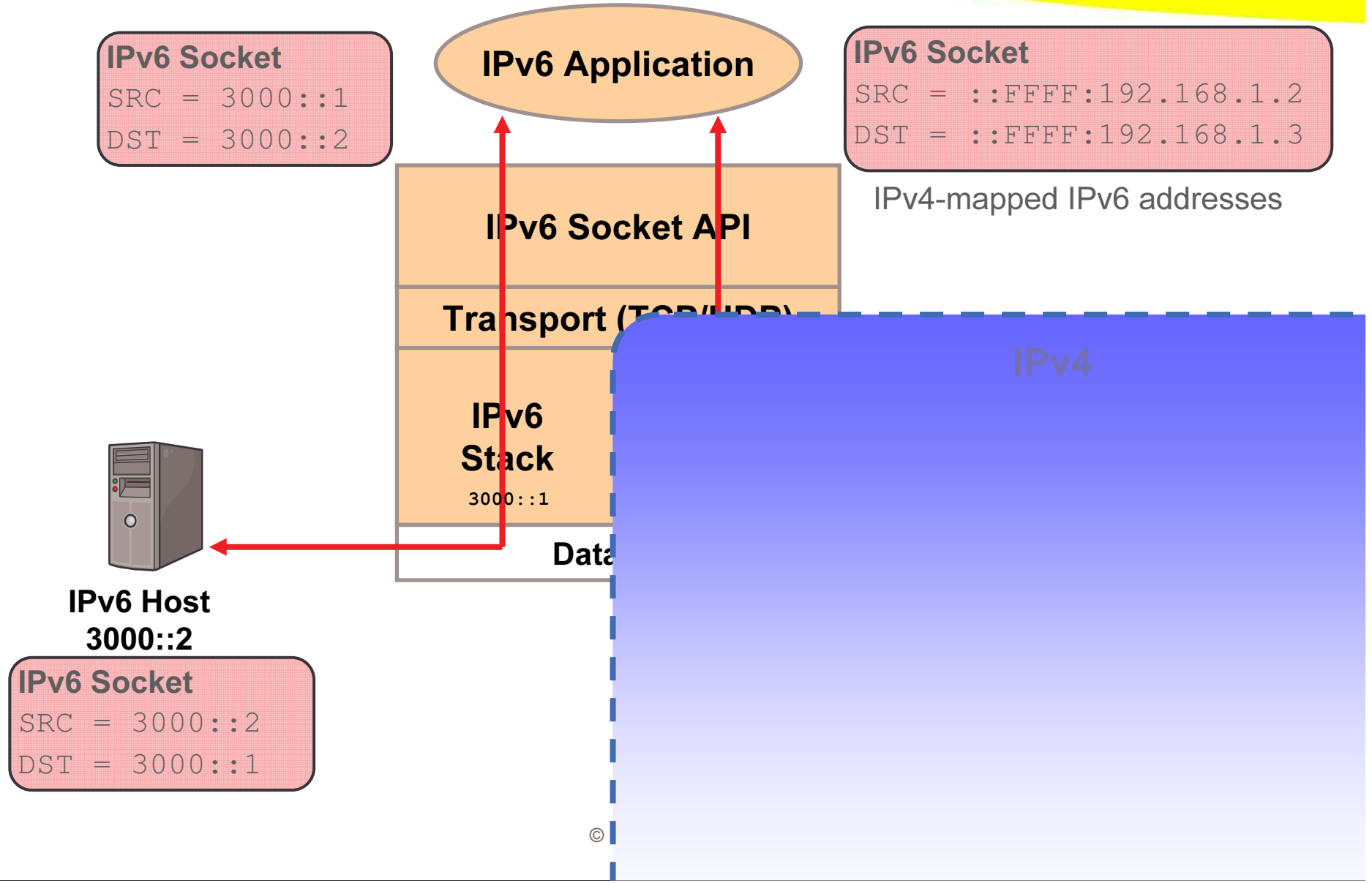
Dual Stack and IPv6 API

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



Erion

Dual Stack and IPv6 API



Socket API IPv4 vs IPv6

	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

Address Structures

- New `sockaddr` structure to store IPv6 addresses

```
struct in6_addr {
    union {
        uint8_t u6_addr8[16];
        uint16_t u6_addr16[8];
        uint32_t u6_addr32[4];
    } in6_u;
    #define s6_addr                 in6_u.u6_addr8
    #define s6_addr16               in6_u.u6_addr16
    #define s6_addr32               in6_u.u6_addr32
};

struct sockaddr_in6 {
    sa_family_t sin6_family;    /* AF_INET6 */
    in_port_t sin6_port;        /* Transport layer port # */
    uint32_t sin6_flowinfo;     /* IPv6 flow information */
    struct in6_addr sin6_addr;  /* IPv6 address */
    uint32_t sin6_scope_id;     /* IPv6 scope-id */
};
```

sockaddr_storage

- Eliminates protocol dependencies from source code

```
#if ULONG_MAX > 0xffffffff
# define __ss_aligntype __uint64_t
#else
# define __ss_aligntype __uint32_t
#endif
#define _SS_SIZE        128
#define _SS_PADSIZE     (_SS_SIZE - (2 * sizeof (__ss_aligntype)))

struct sockaddr_storage
{
    sa_family_t ss_family;      /* Address family */
    __ss_aligntype __ss_align;  /* Force desired alignment.  */
    char __ss_padding[_SS_PADSIZE];
};
```

- Portable applications should use `sockaddr_storage`

Erion

Socket Functions

- Socket API has not changed
  - The socket API handles the generic address structure
- How socket API is used *has* changed

1. Creating a socket
2. Passing socket address from application to API
3. Passing socket address from API to application

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 */
```

  - **Important note**: Even if you specify `PF_INET6` you may still be communicating using IPv4. On a dual stack if your socket call uses an IPv4-mapped IPv6 address it will use the IPv4 stack.

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);
```

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);
```


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);
```

Portable Code

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

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


Erion

Address Conversion

- Conversion from binary representation to textual and vice versa

IPv4 Code

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

/*    Convert IPv4 binary to text */
char        *inet_ntoa(struct in_addr in);
```

IPv6 Code

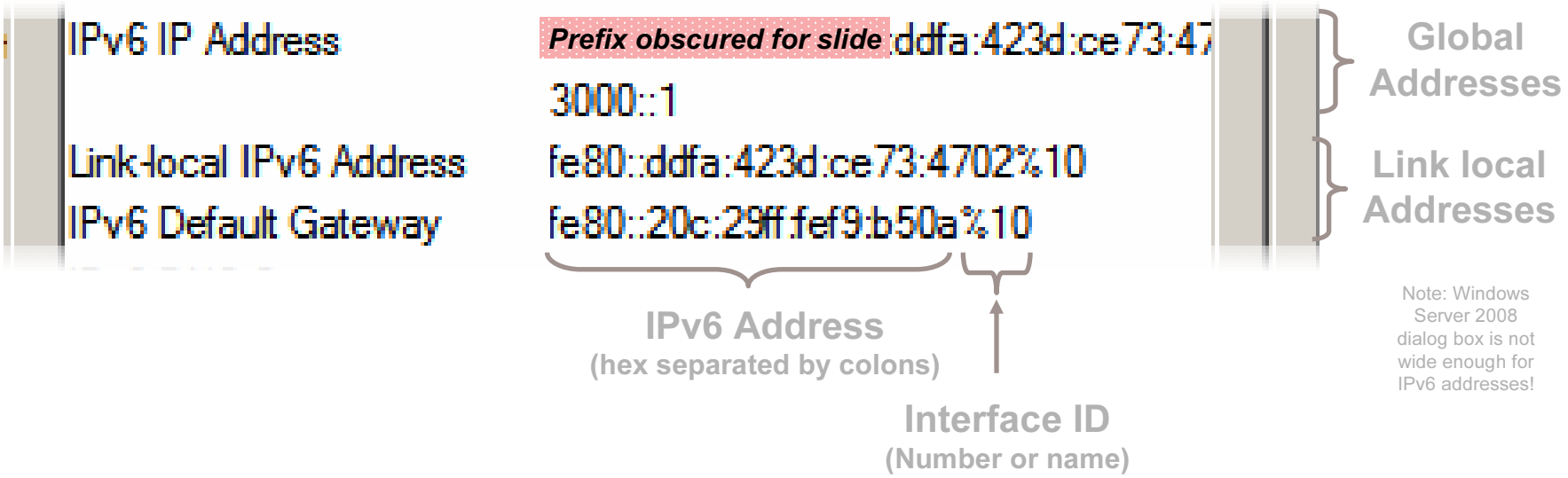
```
int inet_pton(int family, const char *src, void *dst);

/*    Convert IPv4 or IPv6 binary to text    */
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

- 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 domain names instead)

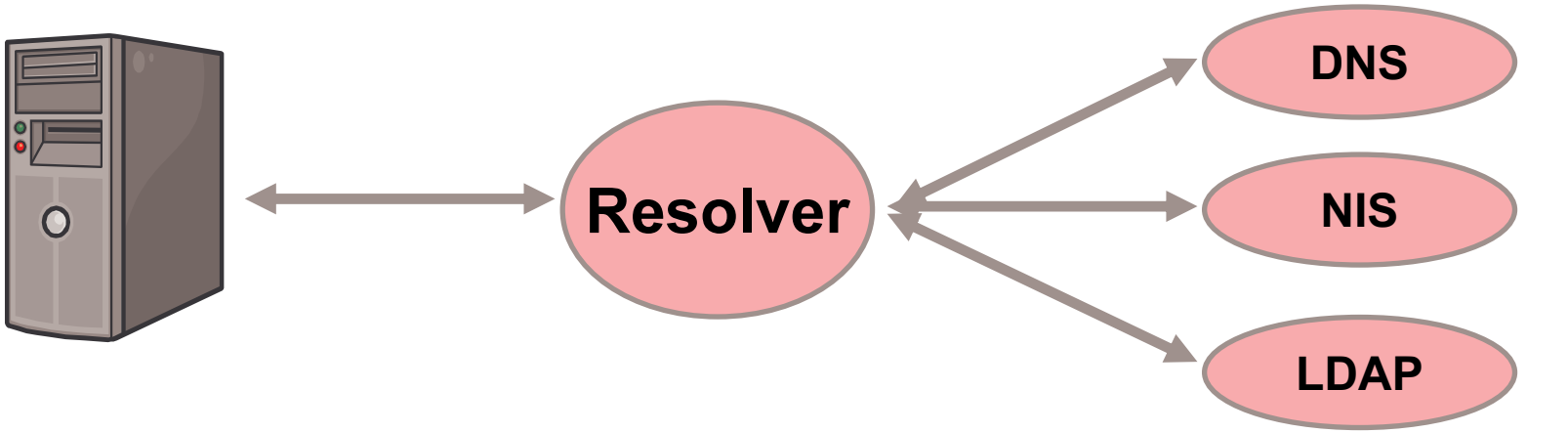
```
3000-0-20-0-3de2-17ca-d07d-5f10.ipv6-literal.net.
```

```
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()`**

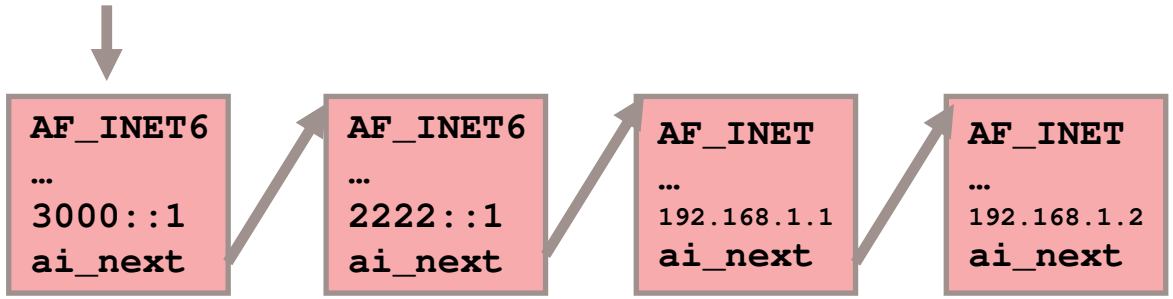


Erion

getaddrinfo()

- Returns linked list of `addrinfo` structures (IPv6 & IPv4)
- Allocates memory (must free using `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 character 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

Interface Identification

- Interfaces uniquely identified by small positive integer
- Interfaces also have unique textual names

`if_nametoindex()`

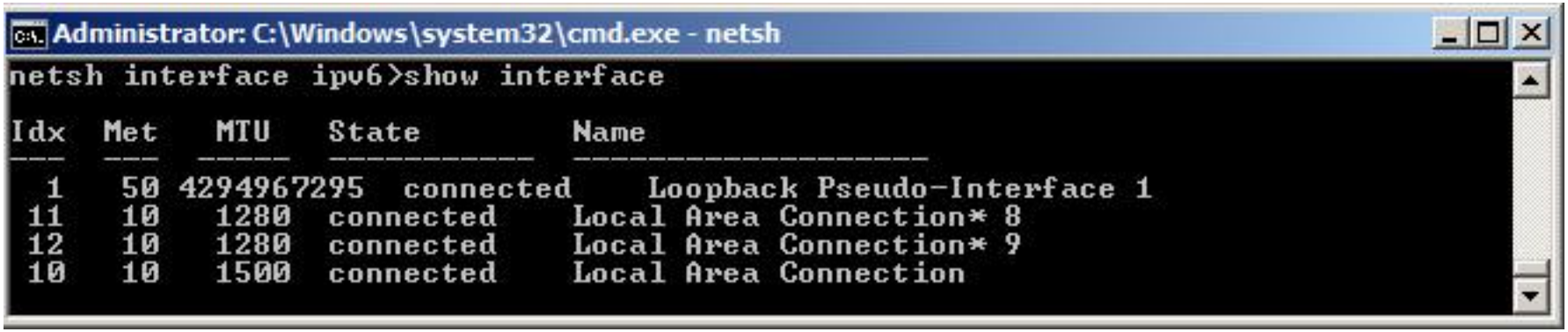
- Converts interface name into interface index

`if_indextoname()`

- Converts interface index to interface name

`if_nameindex()`

- Enumerates all interfaces



Erion

Specifying the Interface

- Use **`setsockopt`** and **`IPV6_PKTINFO`**

```
struct in6_pktinfo {
    struct in6_addr ipi6_addr;     /* src/dst IPv6 address */
    unsigned int    ipi6_ifindex; /* send/recv interface index */
     };
```

```
setsockopt(fd, IPPROTO_IPV6, IPV6_PKTINFO, &po, sizeof(po));
```

  - Necessary with link local addresses

    `fe80:::20:fec4:5d49%eth0`

  - Maybe used for IPv4 only sockets
    - E.g for IPv4 only protocols, WINS etc
  - Defaults are, `in6addr_any` and interface ID of 0

Other Things to be Aware Of

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

IPv6 Porting Tools

- Many tools for porting applications to IPv6

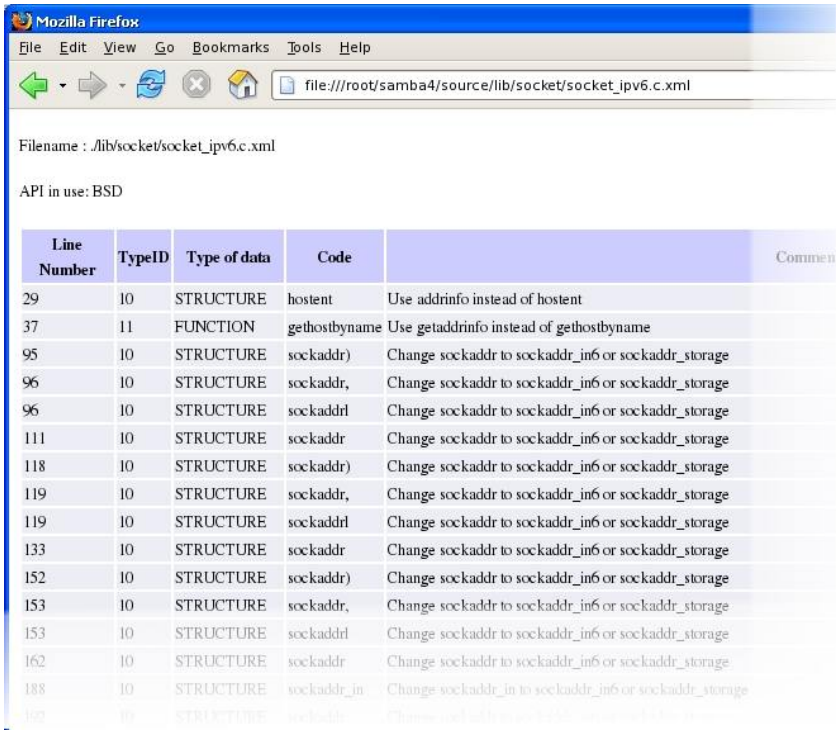
Microsoft	checkv4.exe
Sun	IPv6 Socket Scrubber
HP	IPv6 porting assistant
Open Source	PortToIPv6

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

Erion

Samba4 and 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

Samba4 IPv6 ToDo List (1)

1. Use IPv6 API for everything
   - Start by implementing IPv6 hack
   - Includes IPv4 (dual stack)
   - Only support platforms with IPv6 API
2. Add IPv6 support to Socket Wrapper
   - Current socket wrapper is IPv4 only

Continued…

Erion

Samba4 IPv6 ToDo List (2)

3. Change protocol & address families to **`PF_INET6`** & **`AF_INET6`**

3. Change all address structures to **`sockaddr_storage`**
   - If necessary use **`sockaddr_in6`**

4. Replace **`inet_aton`** & **`inet_ntoa`** with **`inet_pton`** & **`inet_ntop`**

5. Change all name resolution with **`getnameinfo`** and **`getaddrinfo`**
   - Use **`addrinfo`** instead of **`hostent`**
   - Use **`freeaddrinfo`**

6. Replace **all** IPv4 addresses with IPv4-mapped IPv6 addresses

Continued…

Erion

Samba4 IPv6 ToDo List (3)

8. Change all address strings to handle:
   - IPv4 addresses (as now) `56.10.100.1`
   - IPv4-mapped IPv6 addresses `::ffff:56.10.100.1`
   - IPv6 addresses `3000::20:fec4:5d49`
   - URL encoded `[3000::20:fec4:5d49]`
   - Link local with interface identifier `fe80:::20:fec4:5d49%eth0`

   - Ensure that IPv4 and IPv4-mapped address strings for the same address have the same effect (e.g. ACLs)

   - Increase length of address strings (**`INET6_ADDRSTRLEN`**) 46 bytes

Continued…

Erion

Samba4 IPv6 ToDo List (4)

9. Change all IPv4 socket options to IPv6 socket options
   - For example, IPPROTO_IP to IPROTO_IPV6

10. Replace all hard-coded IPv4 addresses (e.g. 127.0.0.1 & 0.0.0.0)
    - Where appropriate replace 0.0.0.0 with ::

11. Change IPv4 loopback to IPv6 loopback (::1) **`in6addr_loopback`**

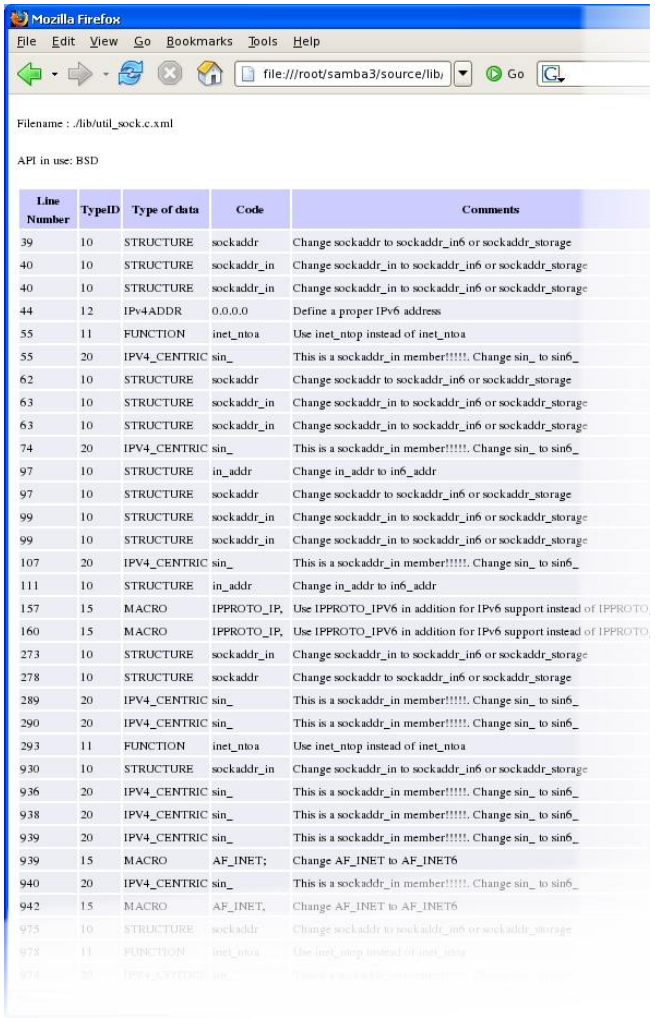
12. Add ipv6-literal.net & literal.net support to Samba resolver

13. Write IPv4 only functionality using IPv4-mapped IPv6 addresses
    - For example WINS
    - Bind to IPv4-mapped IPv6 addresses

Erion

Samba3 and PortToIPv6

- Reported changes needed in
  - 125 source files
  - 1138
  - Will be many more!
- Notes
  - Some false positives
  - Some false negatives
  - Many changes to addresses in strings not picked up
  - Many, many more changes required than for Samba4
  - Socket Wrapper



Line Number	TypeID	Type of data	Code	Comments
39	10	STRUCTURE	sockaddr	Change sockaddr to sockaddr_in6 or sockaddr_storage
40	10	STRUCTURE	sockaddr_in	Change sockaddr_in to sockaddr_in6 or sockaddr_storage
40	10	STRUCTURE	sockaddr_in	Change sockaddr_in to sockaddr_in6 or sockaddr_storage
44	12	IPv4ADDR	0.0.0.0	Define a proper IPv6 address
55	11	FUNCTION	inet_ntoa	Use inet_ntop instead of inet_ntoa
55	20	IPV4_CENTRIC	sin_	This is a sockaddr_in member!!!!!. Change sin_ to sin6_
62	10	STRUCTURE	sockaddr	Change sockaddr to sockaddr_in6 or sockaddr_storage
63	10	STRUCTURE	sockaddr_in	Change sockaddr_in to sockaddr_in6 or sockaddr_storage
63	10	STRUCTURE	sockaddr_in	Change sockaddr_in to sockaddr_in6 or sockaddr_storage
74	20	IPV4_CENTRIC	sin_	This is a sockaddr_in member!!!!!. Change sin_ to sin6_
97	10	STRUCTURE	in_addr	Change in_addr to in6_addr
97	10	STRUCTURE	sockaddr	Change sockaddr to sockaddr_in6 or sockaddr_storage
99	10	STRUCTURE	sockaddr_in	Change sockaddr_in to sockaddr_in6 or sockaddr_storage
99	10	STRUCTURE	sockaddr_in	Change sockaddr_in to sockaddr_in6 or sockaddr_storage
107	20	IPV4_CENTRIC	sin_	This is a sockaddr_in member!!!!!. Change sin_ to sin6_
111	10	STRUCTURE	in_addr	Change in_addr to in6_addr
157	15	MACRO	IPPROTO_IP,	Use IPPROTO_IPV6 in addition for IPv6 support instead of IPPROTO
160	15	MACRO	IPPROTO_IP,	Use IPPROTO_IPV6 in addition for IPv6 support instead of IPPROTO
273	10	STRUCTURE	sockaddr_in	Change sockaddr_in to sockaddr_in6 or sockaddr_storage
278	10	STRUCTURE	sockaddr	Change sockaddr to sockaddr_in6 or sockaddr_storage
289	20	IPV4_CENTRIC	sin_	This is a sockaddr_in member!!!!!. Change sin_ to sin6_
290	20	IPV4_CENTRIC	sin_	This is a sockaddr_in member!!!!!. Change sin_ to sin6_
293	11	FUNCTION	inet_ntoa	Use inet_ntop instead of inet_ntoa
930	10	STRUCTURE	sockaddr_in	Change sockaddr_in to sockaddr_in6 or sockaddr_storage
936	20	IPV4_CENTRIC	sin_	This is a sockaddr_in member!!!!!. Change sin_ to sin6_
938	20	IPV4_CENTRIC	sin_	This is a sockaddr_in member!!!!!. Change sin_ to sin6_
939	20	IPV4_CENTRIC	sin_	This is a sockaddr_in member!!!!!. Change sin_ to sin6_
939	15	MACRO	AF_INET;	Change AF_INET to AF_INET6
940	20	IPV4_CENTRIC	sin_	This is a sockaddr_in member!!!!!. Change sin_ to sin6_
942	15	MACRO	AF_INET,	Change AF_INET to AF_INET6

Erion

Samba3 IPv6 ToDo List

- Everything on Samba4 IPv6 ToDo list plus some!
  - IPv6 enabling Samba3 is a bigger job than for Samba4
  - Is it necessary to IPv6 enable Samba3?
    - Jeremy says it is
    - If Samba4 includes client as well as DC functionality then make Samba4 the IPv6 version of Samba and forget IPv6 in Samba3?

Erion

Questions?

- What about platforms with no IPv6 API?
  - lib/replace
- What about platforms with old IPv6 API?
  - lib/replace
- Should Samba4 be built only on platforms with recent IPv6 API?

Erion

References

- IPv6 Blog and Samba Patches
  - www.ipv6consultancy.com/ipv6blog
- Current RFCs
  - RFC4584 **Extension to Sockets API for Mobile IPv6**.
  - RFC4038 **Application Aspects of IPv6 Transition**.
  - RFC3542 **Advanced Sockets Application Program Interface (API) for IPv6**
  - RFC3678 **Socket Interface Extensions for Multicast Source Filters**
  - RFC3493 **Basic Socket Interface Extensions for IPv6**
  - RFC3338 **Dual Stack Hosts Using "Bump-in-the-API" (BIA)**
- Porting Guides
  - HP-UX IPv6 Porting Guide, HP
  - Porting Networking Applications to the IPv6 APIs, Sun
  - Many more…

Erion