Windows Socket Programming & IPv6 Translation Middleware

Dr. Whai-En Chen

VoIP and IPv6 Laboratory
Research Assistant Professor
Dept. of Computer Science and Information Engineering
National Chiao Tung University
Email: wechen@mail.nctu.edu.tw
TEL: +886-3-5731924

Outline

- Introduction to Socket/WinSock Programming
- IPv4 WinSock Programming
- IPv6 WinSock Programming
- IPv6 Translation Middleware- Socket-layer Translator
- Conclusions

Introduction

- What is Windows Sockets?

  - An Open Interface for Network Programming under Microsoft Windows

- What are its Benefits?

  - an open standard

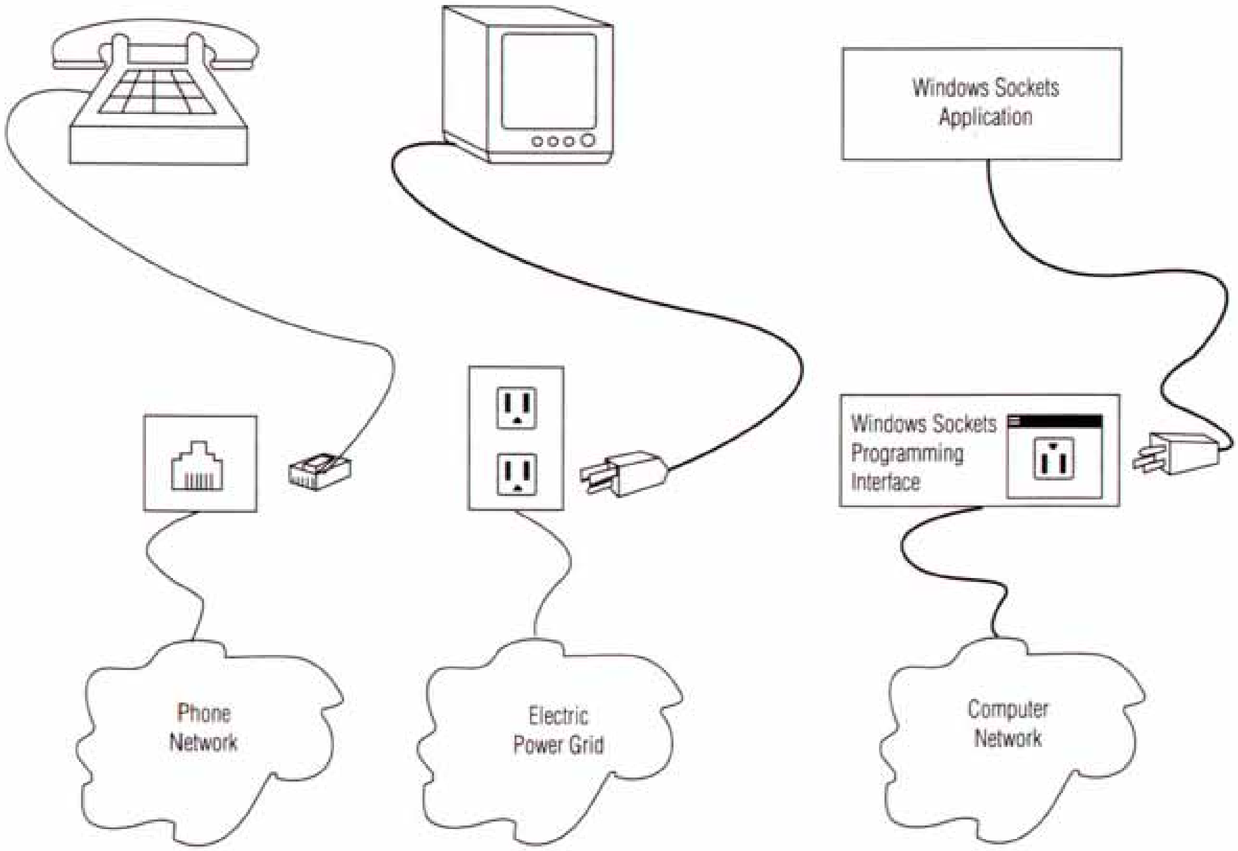
  - source code portability

  - support dynamic linking

- What is its Future?

  - WinSock 2

Windows Sockets


Windows Sockets Application
Windows Sockets Programming Interface
Phone Network
Electric Power Grid
Computer Network

Standard applications using standard interfaces to access standard services.

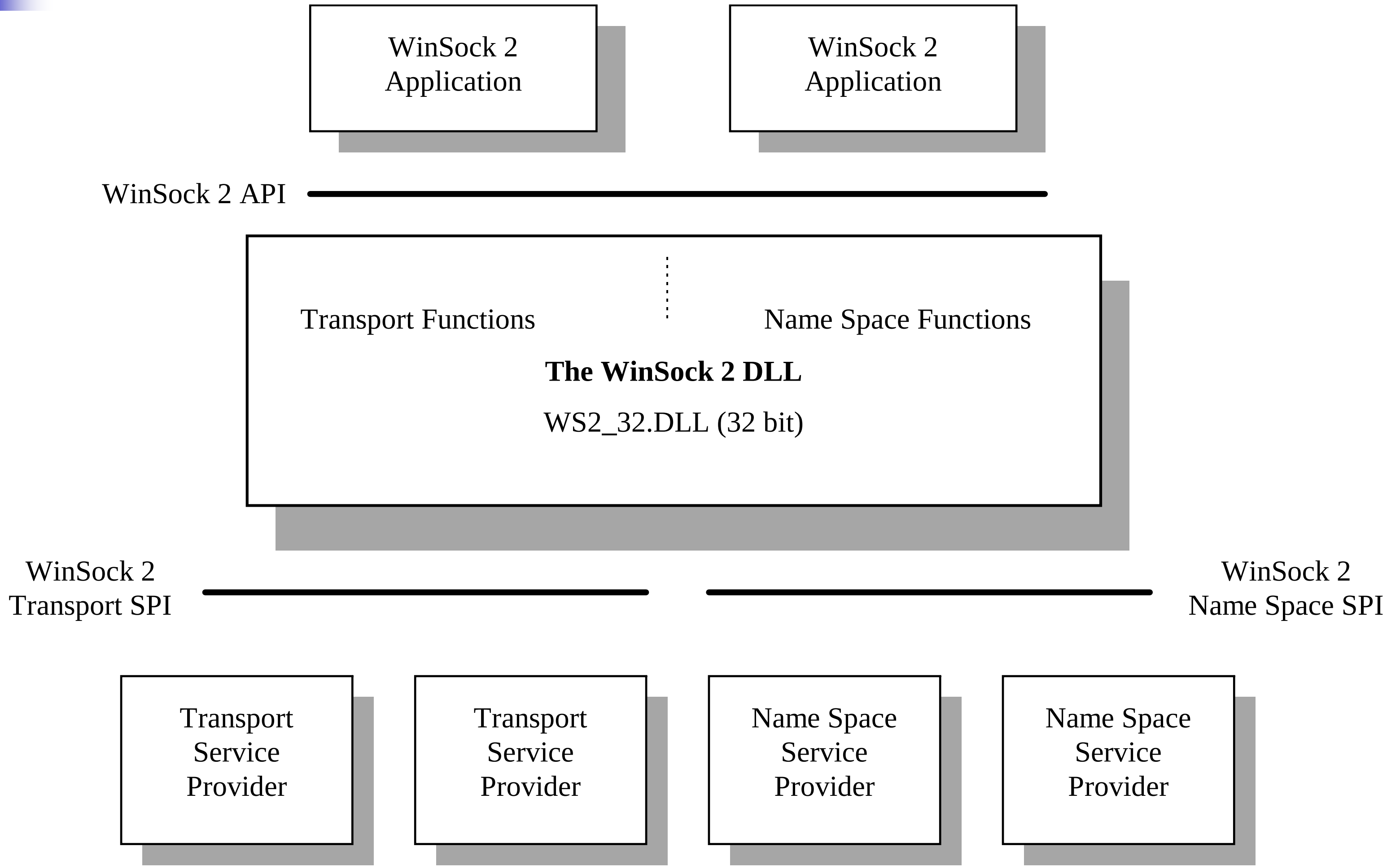
BSD Socket APIs

accept() bind() closesocket() connect()
getpeername() getsockname() getsockopt() htonl()
htons() inet_addr() inet_ntoa() ioctlsocket()
listen() ntohl() ntohs() recv()
recvfrom() select() send() sendto()
setsockopt() shutdown() socket()
gethostname()
gethostbyaddr() gethostbyname()
getprotobyname() getprotobynumber()
getservbyname() getservbyport()

Winsock APIs

WSAAsyncGetHostByAddr()
WSAAsyncGetHostByName()
WSAAsyncGetProtoByName()
WSAAsyncGetProtoByNumber()
WSAAsyncGetServByName()
WSAAsyncGetServByPort()
WSAAsyncSelect()
WSACancelAsyncRequest()
WSACancelBlockingCall()
WSACleanup()
WSAGetLastError()
WSAIsBlocking()
WSASetBlockingHook()
WSASetLastError()
WSAStartup()
WSAUnhookBlockingHook()

Windows Sockets 2.0 Architecture



WinSock 2 Application

WinSock 2 Application

WinSock 2 API

Transport Functions

Name Space Functions

The WinSock 2 DLL

WS2_32.DLL (32 bit)

WinSock 2 Transport SPI

WinSock 2 Name Space SPI

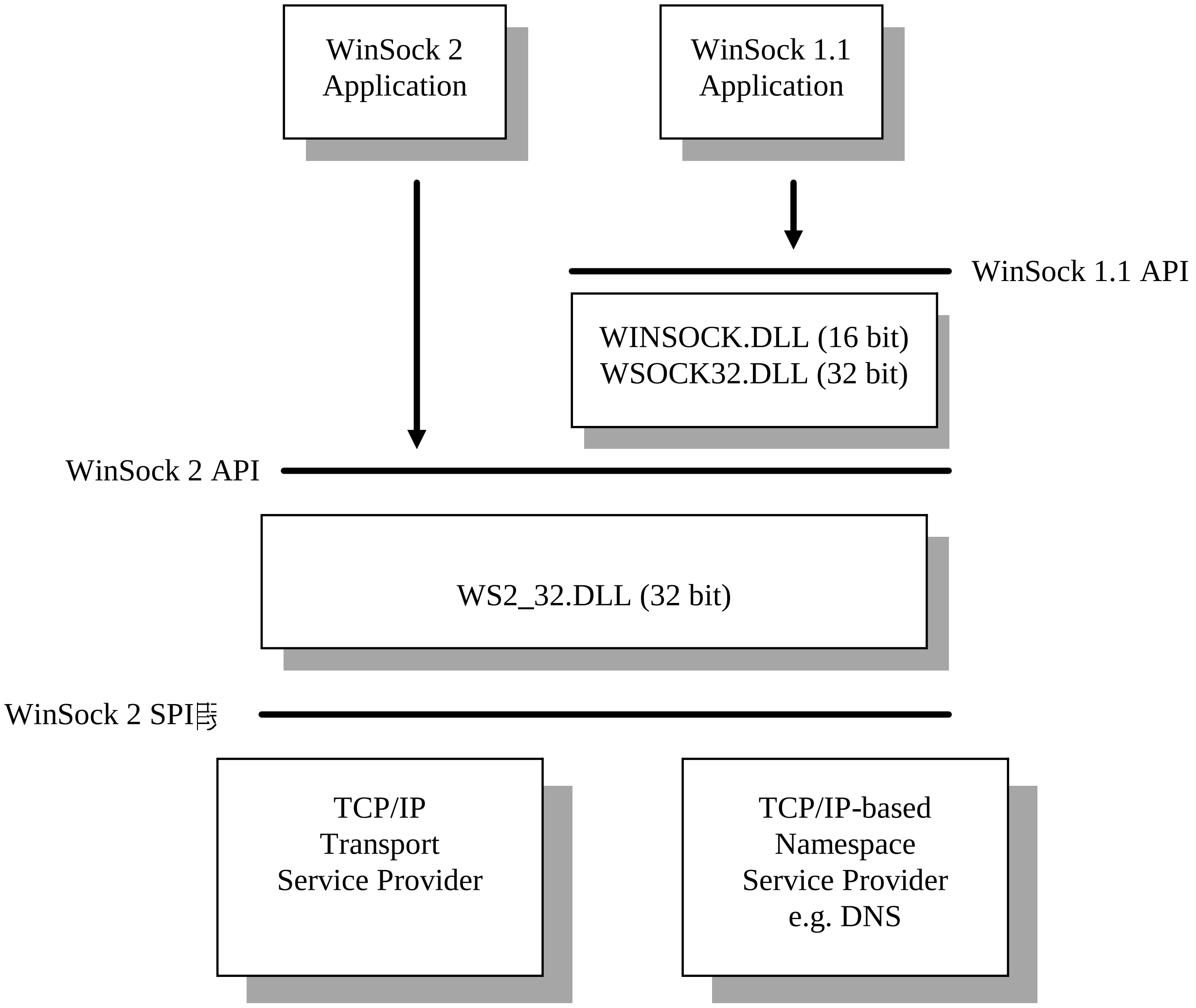
Transport Service Provider

Transport Service Provider

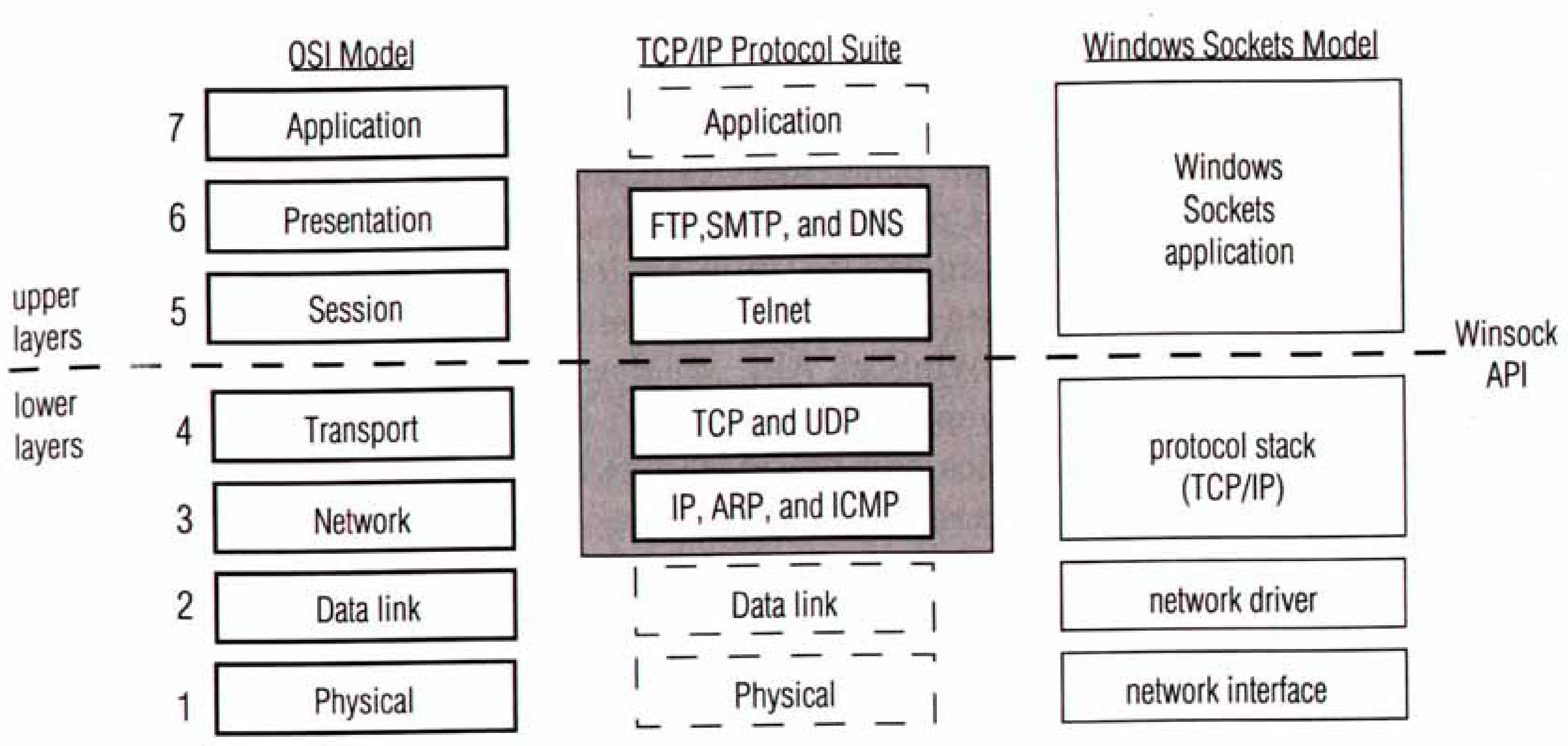
Name Space Service Provider

Name Space Service Provider

Compatibility of Winsock



Winsock and OSI Model



The TCP/IP protocol suite compared to the OSI model and Windows Sockets model.

Client/Server Model

- Client-Server Model
- Client and Server Association

  - protocol (same for both Clint and server sockets)

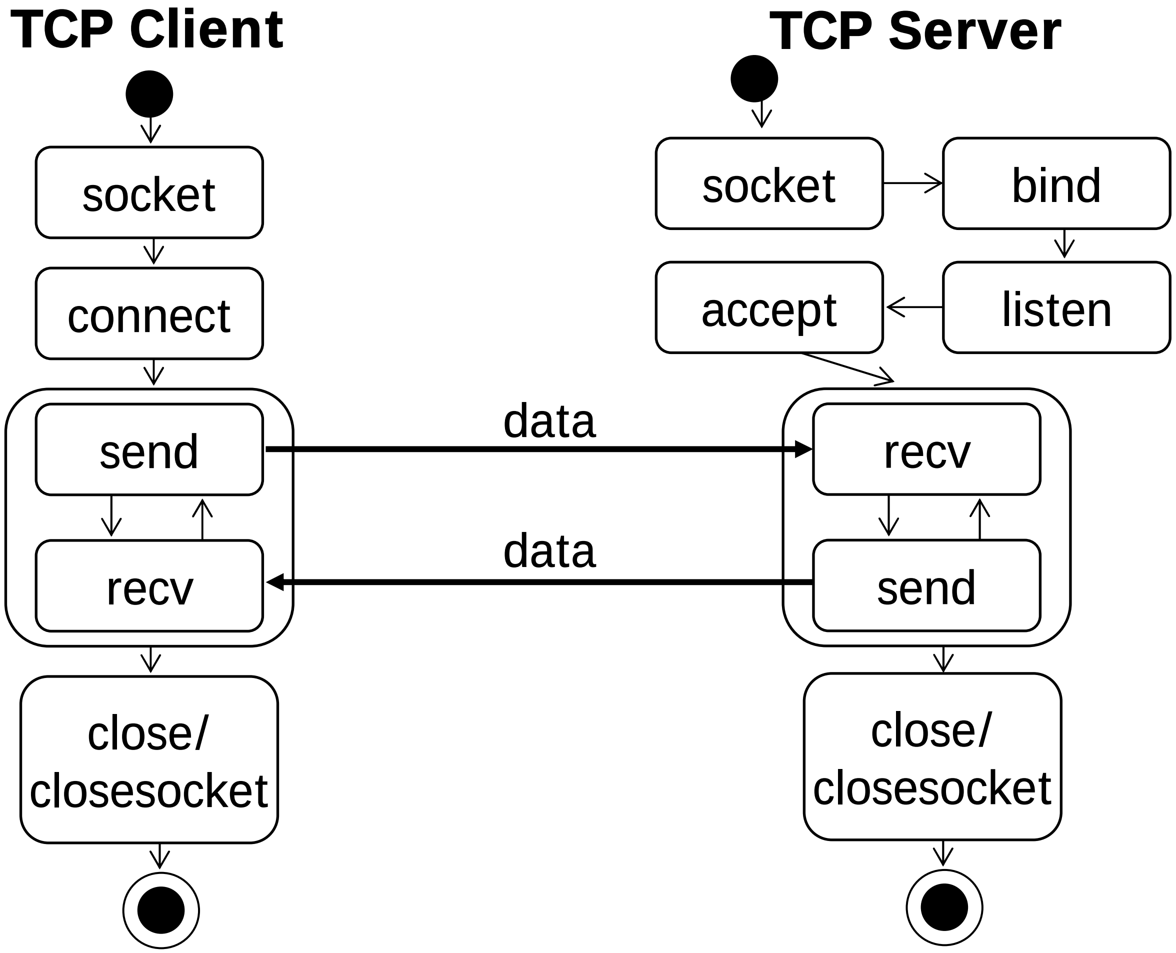
  - client IP address

  - client port number

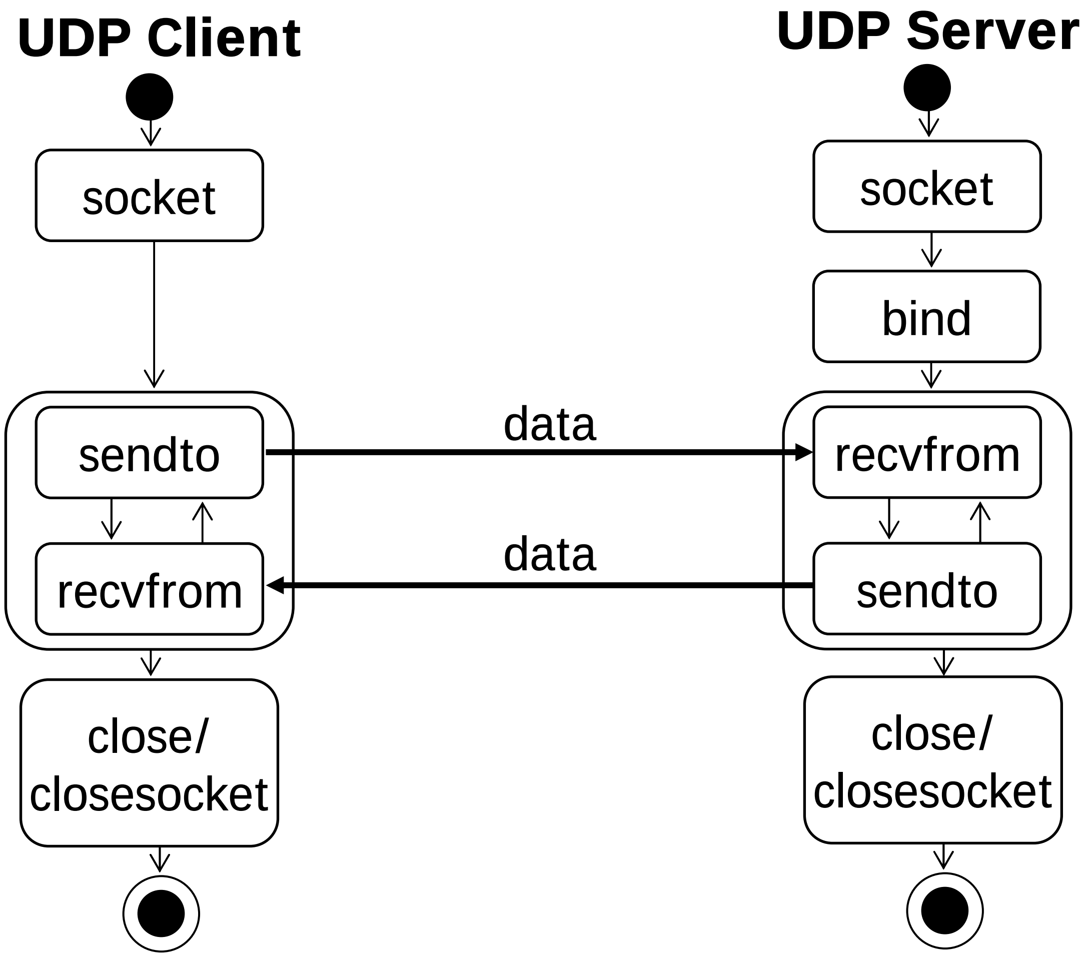
  - server IP address

  - server port number

Client/Server Programming(1)



Client/Server Programming(2)



UDP Client

socket

sendto

recvfrom

close/ closesocket

UDP Server

socket

bind

recvfrom

sendto

close/ closesocket

data

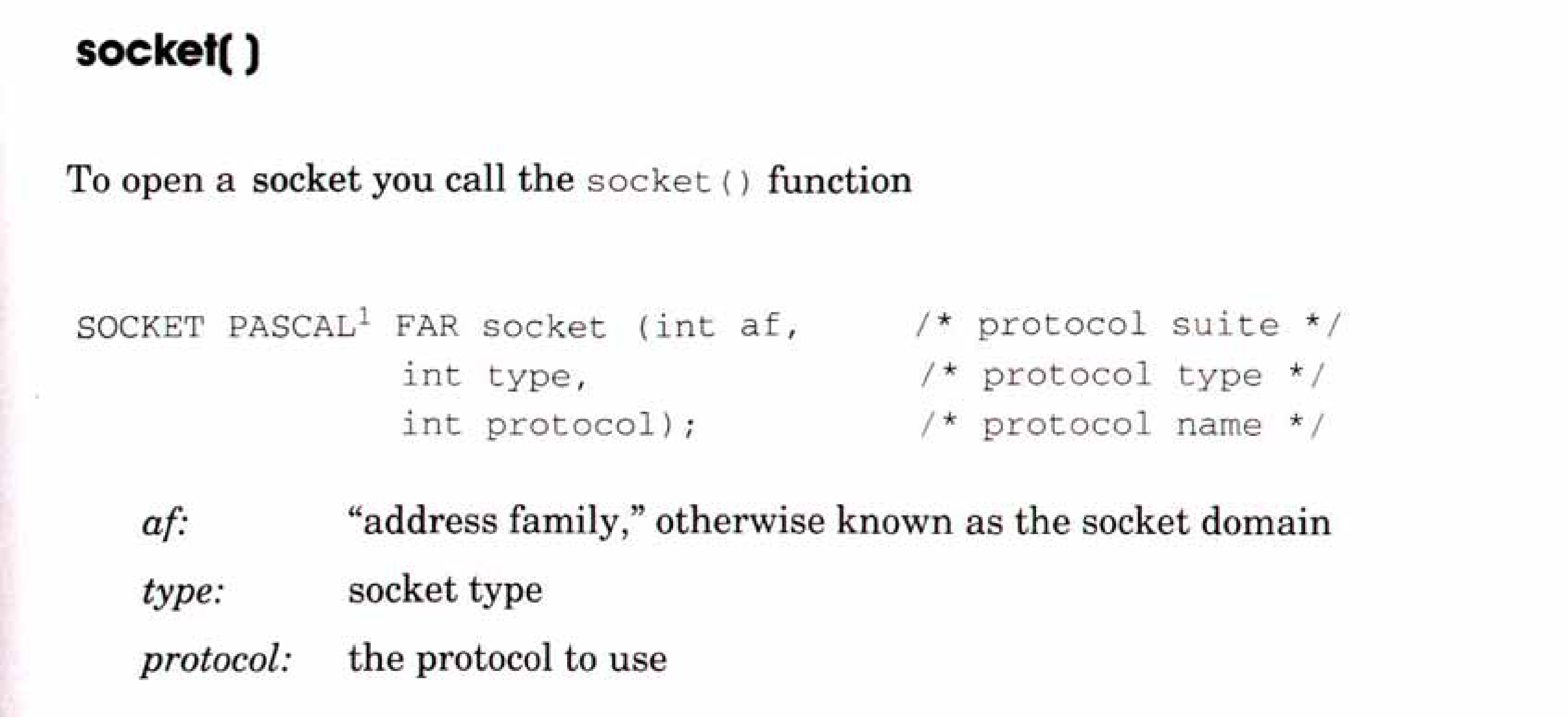
data

IPv4 Socket Programming

Network Program Sketch

- Open a socket
- Name the socket
- Associate with another socket
- Send and receive between sockets
- Close the socket

Open a Socket



socket( )

To open a socket you call the `socket()` function

```
SOCKET PASCAL¹ FAR socket (int af,       /* protocol suite */
                int type,                /* protocol type */
                int protocol);           /* protocol name */
```

af: “address family,” otherwise known as the socket domain

type: socket type

protocol: the protocol to use

Name the Socket

- What’s in a Socket Name?

  - protocol, port number and IP address

- bind( )

```
int PASCAL FAR bind ( SOCKET s,    /*an unbound socket */
 struct sockaddr  FAR *addr,          /*local port and IP addr */
 int namelen);                              /*addr structure length*/
```

S : *socket handle*

addr : ***pointer to a socket address structure (always a*** *sockaddr_in* ***data structure for TCP/IP)***

namelen: ***length of socket structure pointed to by*** *addr* ***(always 4 for TCP/IP)***

Name the Socket

- sockaddr Structure

```
struct stockaddr {
  u_short     sa_family;        /*address family*/
  char        sa_data[14];      /*undefined*/
 };
```

sa_family : ***address family***

sa_data: ***address structure data area defined according to address family value***

Name the Socket

- sockaddr_in Structure

```
structure sockaddr_in  {
   short        sin_family;   /* address family (PF_INET)  */
   u_short     sin_port;       /* port (service) number   */
   struct      in_addr sin_addr;   /* IP address (32-bit) */
   char         sin_zero[8];     /*<unused filler>*/
};
```

sin_family : address family

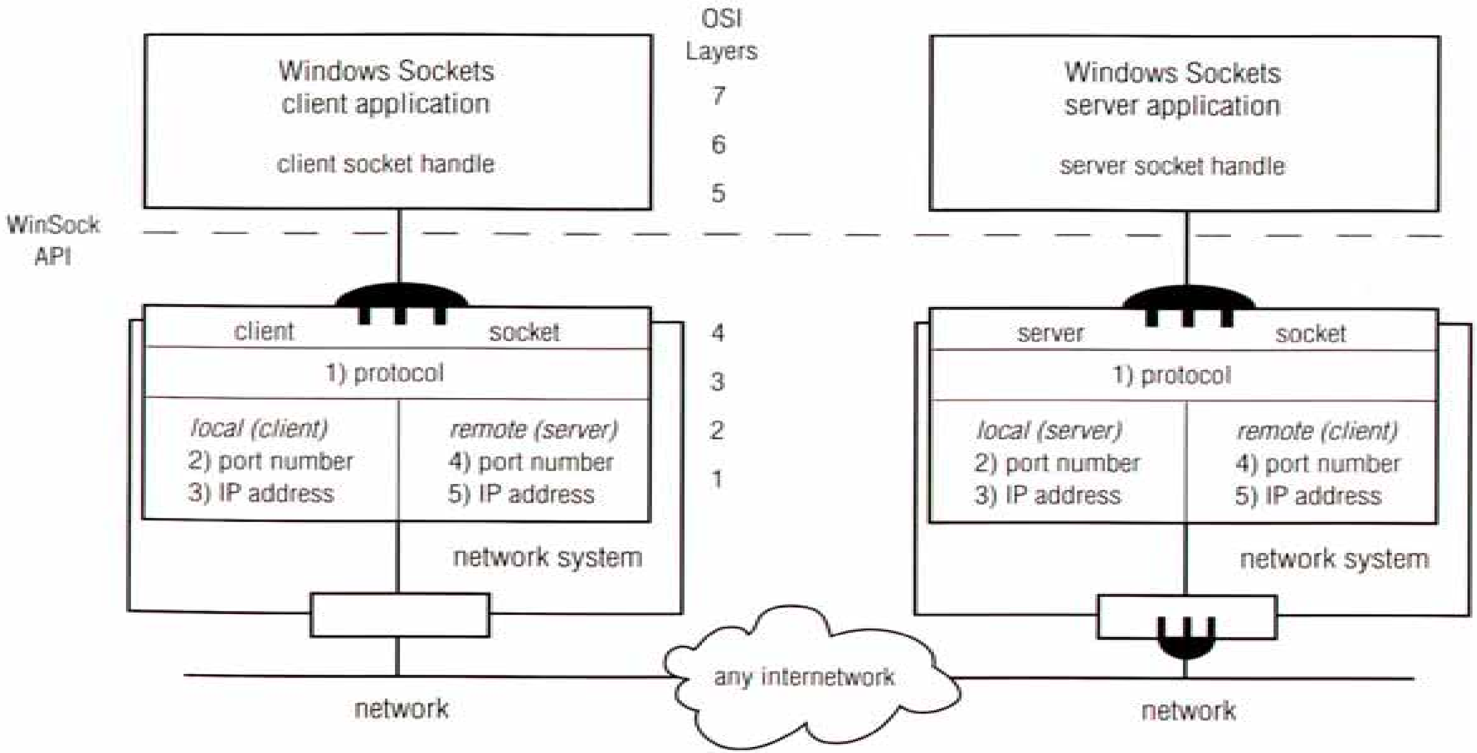
sin_port :   16-bit port number in network order

sine_addr :   32-bit Internet address in network order

Associate with Another Socket

- Protocol (same for both client and server sockets)
- client IP address
- client port number
- server IP address
- server port number

Associate with Another Socket



After the association is completed, the client and server know the socket name of their peer. The combination of the two socket names defines the association.

Associate with Another Socket

- How a Server Prepares for an Association

listen()

```
int PASCAL FAR listen ( SOCKET s,   /* a named, unconnected
                                       socket */
                        int  backlog) ;   /* pending connect queue
                                       length  */
```

s: ***socket handle to a named socket (bind() called),***
but not yet connected

backlog: ***length of the pending connection queue (not the***
same as the number of accepted connections)

Associate with Another Socket

- How a Client Initiate an Association

connect()

```
 int PASCAL FAR connect (SOCKET s,  /*an unconnected socket */
struct sockaddr FAR *addr,             /*remote port and IP addr */
int namelen );                          /* addr structure length */
```

s: ***socket handle***

addr: ***pointer to socket address structure (always a*** *sockaddr_in* ***structure for TCP/IP)***

namelen: ***length of structure pointed to by*** *addr* ***(always 4 for TCP/IP)***

Associate with Another Socket

- How a Server Completes an Association

accept()

```
SOCKET PASCAL FAR accept (SOCKET s,    /*a listening socket*/
                struct  sockaddr FAR *addr,      /*name of incoming
                                                     socket*/
                int FAR *addrlen);
```

s: ***socket handle***

addr: ***pointer to socket address structure (always a*** *sockaddr_in* ***structure for TCP/IP)***

addrlen: ***length of socket structure that*** *addr* ***points to (always 4 for TCP/IP)***

Send and Receiver between Sockets

- Sending Data on a “Connected” Socket

send()

```
int PASCAL FAR send (SOCKET s,            /*associated socket*/
        const char FAR *buf,       /*buffer with outgoing data*/
        int len,                   /*bytes to send*/
        int flags );               /*option flags*/
```

s: ***socket handle***

buf: ***pointer to a buffer that contains application data to send***

len: ***length of data (in bytes) to send***

flags: ***flags to affect the send (MSG_OOB, MSG_DONTROUTE)***

Send and Receiver between Sockets

- Sending Data on an “Unconnected” Socket

sendto()

```
int PASCAL FAR sendto  (SOCKET s,        /*a valid socket */
      const char FAR *buf,                /*buffer with outgoing data */
      int len,                            /*bytes to send */
      int flags,                          /*option flags */
      struct sockaddr FAR *to,            /*remote socket name */
      int tolen );                        /*length of sockaddr */
```

to: ***pointer to socket structure (always a*** *sockaddr_in* ***for TCP/IP) that contains destination address and port number (socket name)***

tolen: ***length of socket structure pointed to by to (always 4 for TCP/IP)***

Send and Receiver between Sockets

- Receiving Data

recv()

```
int PASCAL FAR recv (SOCKET s,             /*associated socket*/
        char FAR *buf,                     /*buffer with outgoing data*/
        int len,                           /*bytes to send */
        int flags );                       /*option flags */
```

recvform()

```
int PASCAL FAR recvform (SOCKET s,             /*a valid socket*/
        char FAR *buf,                     /*buffer with outgoing data*/
        int len,                           /*bytes to send */
        int flags );                       /*option flags */
        struct sockaddr FAR *from,         /*remote socket name */
        int fromlen );                     /*length of sockaddr */
```

Send and Receiver between Sockets

s: ***socket handle***

buf: ***pointer to a buffer that contains application data to send***

len: ***length of data (in bytes) to send***

flags: ***flags to affect the send (MSG_OOB, MSG_DONTROUTE)***

from: ***pointer to socket structure (always a*** *sockaddr_in* ***for TCP/IP) that contains source address and port number (socket name)***

fromlen: length of socket structure pointed to by *from* ***(always 4 for TCP/IP)***

Other Useful Socket Functions

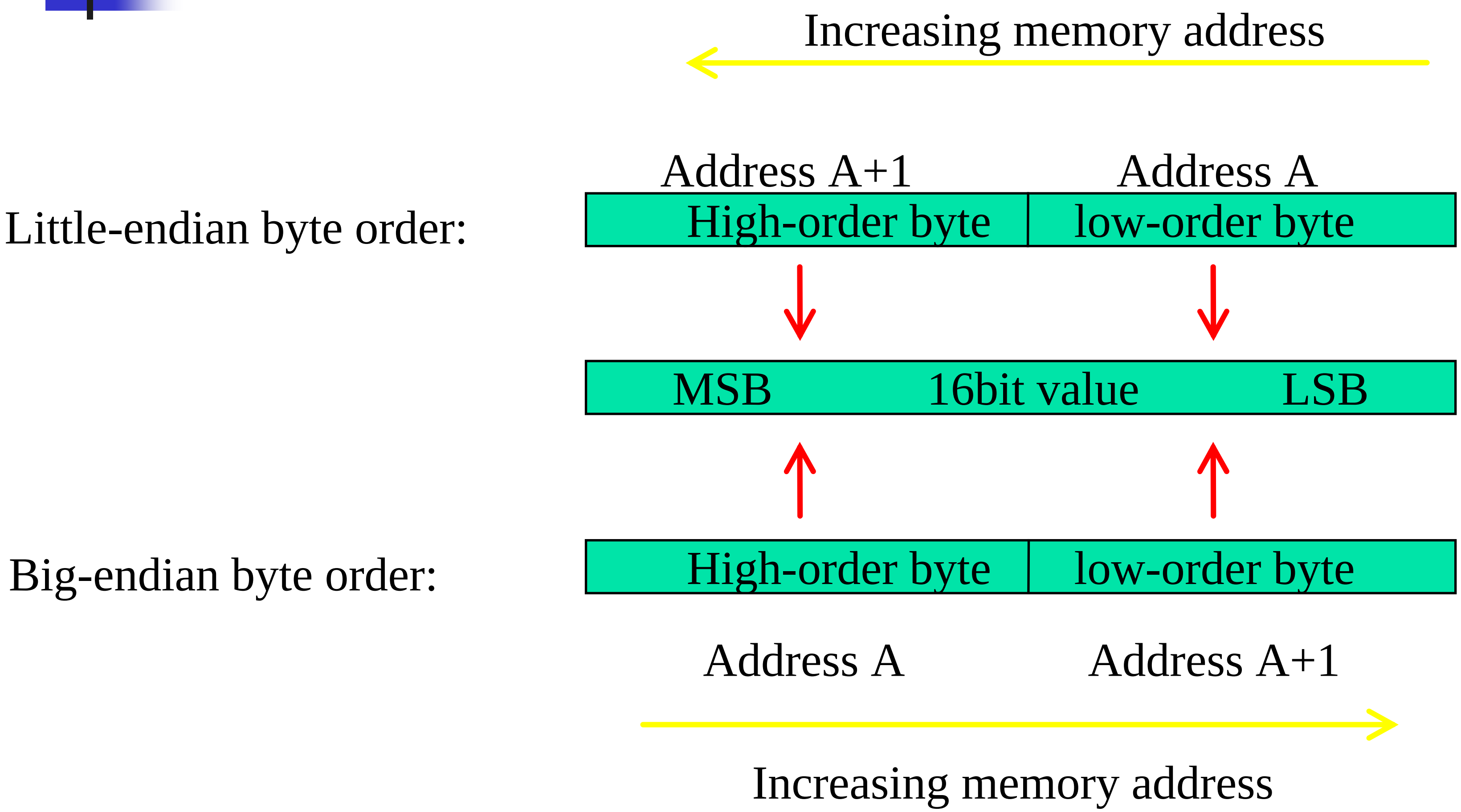
- Byte Ordering Functions
  - ntohs(), ntohl()
  - htons(), htonl()
- Address Translation Functions
  - inet_addr()- 將字串轉成32位元的IP位址
  - inet_nota()- 將32位元的IP位址轉成字串
- Name Resolution
  - gethostbyaddr()-利用 host 的位址來獲取該 host 的資料
  - gethostbyname()-利用 host 的名稱來獲取該 host 的資料
  - 傳回hostent的資料結構
- WSAStartup() and WSACleanup()

hostent 資料結構

```
struct hostent {
        char FAR *                h_name;
        char FAR * FAR *          h_aliases;
        short                     h_addrtype;
        short                     h_length;
        char FAR * FAR *          h_addr_list;
      }
```

- 是一個linked-list

Byte Ordering Function



Increasing memory address

Little-endian byte order:

Address A+1 | Address A

High-order byte | low-order byte

MSB | 16bit value | LSB

Big-endian byte order:

High-order byte | low-order byte

Address A | Address A+1

Increasing memory address

IPv4 Example for Daytime Server (Connection-oriented)

```
int main(int argc, char **argv)
{
      int listenfd, connfd;
      struct sockaddr_in servaddr;
      char buff[MAXLINE];
      time_t ticks;
      listenfd =
     socket(AF_INET, SOCK_STREAM, 0);
      bzero(&servaddr, sizeof(servaddr));
      servaddr.sin_family      = AF_INET;
      servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
      servaddr.sin_port        = htons(13);
     /* daytime server */

      bind(listenfd, (SA *) &servaddr, sizeof(servaddr));
```

IPv4 Example for Daytime Server (Connection-oriented)

```
    listen(listenfd, LISTENQ);

    for ( ; ; ) {
          connfd = accept(listenfd, (SA *) NULL, NULL);

     ticks = time(NULL);
     snprintf(buff, sizeof(buff), "%.24s\r\n", ctime(&ticks));
     write(connfd, buff, strlen(buff));

          Close(connfd);
    }
}
```

IPv4 Example for Daytime Client (Connection-oriented)

```
int main(int argc, char **argv)
{
      int sockfd, n;
      char recvline[MAXLINE + 1];
      struct sockaddr_in servaddr;
      if (argc != 2)
                err_quit("usage: a.out <IPaddress>");
      if ( (sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0)
                err_sys("socket error");
      bzero(&servaddr, sizeof(servaddr));
      servaddr.sin_family = AF_INET;
      servaddr.sin_port   = htons(13);
     /* daytime server */
      if (inet_pton(AF_INET, argv[1], &servaddr.sin_addr) <= 0)
                err_quit("inet_pton error for %s", argv[1]);
```

IPv4 Example for Daytime Client (Connection-oriented)

```
    if (connect(sockfd, (SA *) &servaddr, sizeof(servaddr)) < 0)
            err_sys("connect error");

    while ( (n = read(sockfd, recvline, MAXLINE)) > 0) {
            recvline[n] = 0;
           /* null terminate */
            if (fputs(recvline, stdout) == EOF)
                    err_sys("fputs error");
    }
    if (n < 0)
            err_sys("read error");
    exit(0);
}
```

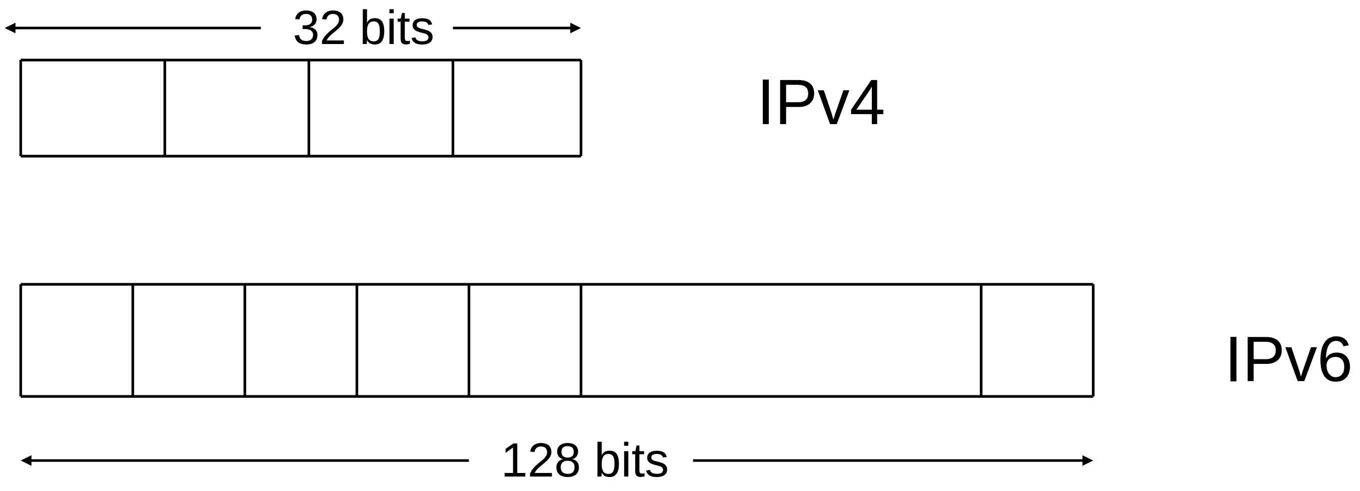
IPv6 Socket Programming

提供轉換IPv4程式到IPv6之方法

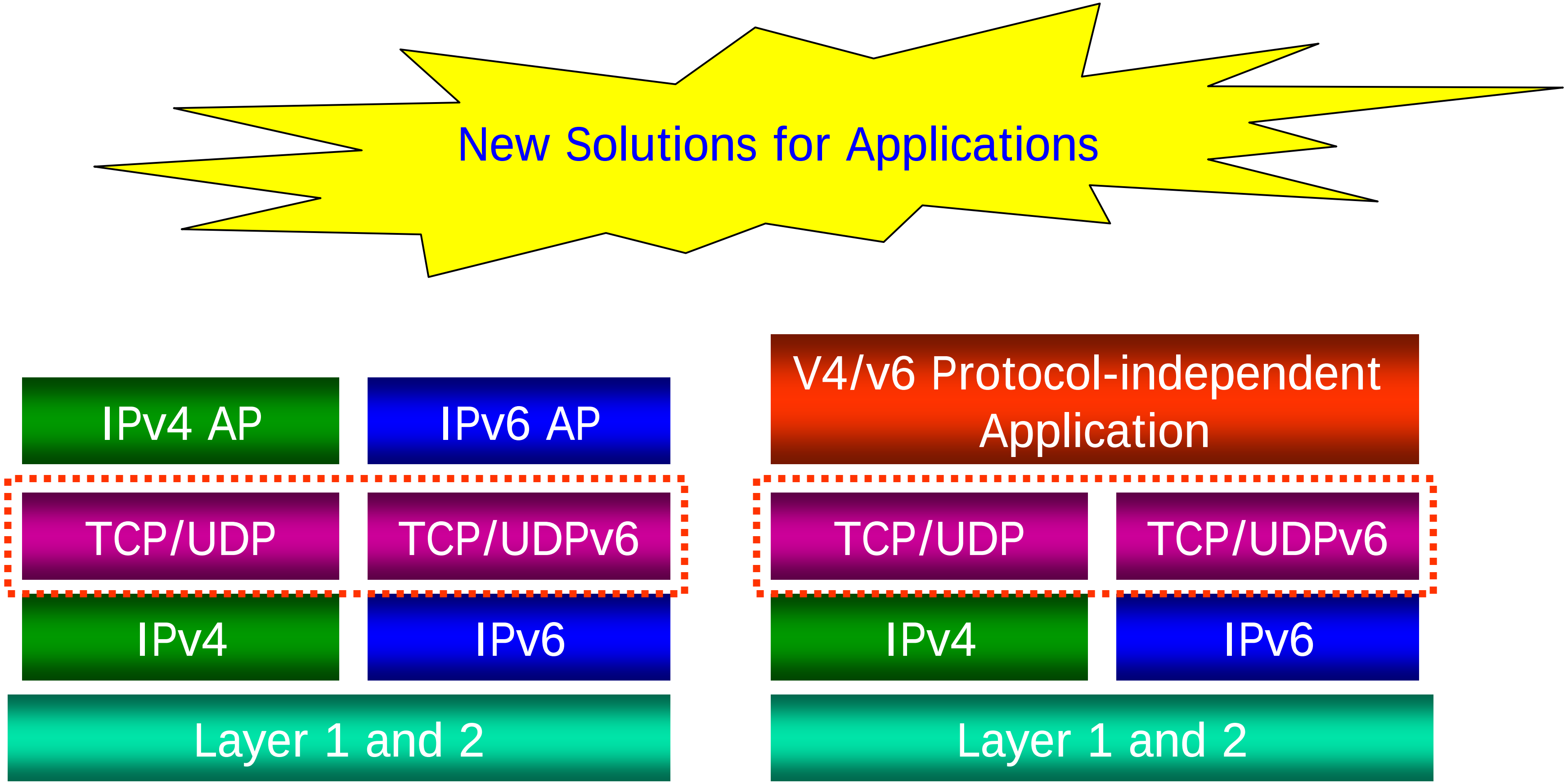
- 介紹IPv4與IPv6之長度不同
- 介紹為何需要改變應用程式
- 介紹不用轉換的Socket API
- 介紹需要轉換的Socket API
- 介紹需要轉換的資料結構

IPv4/IPv6位址長度不同

- 數字位址
  - IPv4, 32位元位址長度
  - IPv6, 128位元位址長度



為何需要轉換應用程式


New Solutions for Applications
IPv4 AP
IPv6 AP
TCP/UDP
TCP/UDPv6
IPv4
IPv6
Layer 1 and 2
V4/v6 Protocol-independent Application
TCP/UDP
TCP/UDPv6
IPv4
IPv6
Layer 1 and 2

不需要轉換的Socket API (依序)

- Server端的程式碼

socket	open a socket
bind	bind local address to the socket
listen	listen on a port
accept	wait for the connection
read/write	if TCP
recvfrom/sendto	if UDP

- Client端的程式碼

socket	open a socket
connect	connect to a server
read/write	if TCP
recvfrom/sendto	if UDP

轉換需要改變的部分

- 與IP位址相關的Socket API與參數需要修改
- 程式部分有運用到IP位址的部分
  - 位址轉換函式
  - 位址複製函式
  - 位址比較函式
  - 位址相關之記憶體指派與變數宣告

IPv4程式設計者的自訂的函式與變數也需要修改

API與資料結構的轉換

- Socket參數名稱轉換

IPv4	IPv6
AF_INET	AF_INET6
PF_INET	PF_INET6
IN_ADDR_ANY	inaddr6_any

API與資料結構的轉換

- 資料結構轉換

IPv4	IPv6
in_addr	in6_addr
sockaddr	sockaddr_in6
sockaddr_in	sockaddr_in6

IPv4 Socket Address Structure

```
Struct in_addr{
    in_addr_t    s_addr;                    /*32bit IPv4 address*/
  };                                        /*network byte ordered*/

struct sockaddr_in {
  uint8_t          sin_len;                /* length of structure(16) */
  sa_family_t      sin_family;             /* AF_INET */
  in_port_t        sin_port;               /* 16bit TCP or UDP port number */
                                           /*network byte ordered*/
  struct    in_addr  sin_addr;             /* 32bit IPv4 address */
                                           /*network byte ordered*/
  char      sin_zero[8];                   /* unused */
};  /* included in <netinet/in.h> */
```

IPv6 Socket Address Structure

```
Struct in6_addr{
        uint8_t      s6_addr[16];                              /*128bit  IPv6 address*/
    };                                                       /*network byte ordered*/
#define  SIN6_LEN                      /* required for compile-time tests */
struct sockaddr_in6 {
    uint8_t                  sin6_len;                        /* length of structure(24) */
    sa_family_t              sin6_family;                     /* AF_INET6*/
    in_port_t                sin6_port;                       /* Transport layer port# */
                                                              /*network byte ordered*/
    uint32_t                 sin6_flowinfo;                    /* priority & flow label */
                                                              /*network byte ordered*/
    struct     in6_addr    sin6_addr;                           /* IPv6 address */
                                                              /*network byte ordered*/
}; /* included in <netinet/in.h> */
```

API與資料結構的轉換

- 資料結構參數轉換

IPv4	IPv6
sin_len	sin6_len
sin_family	sin6_family
sin_port	sin6_port
sin_addr	sin6_addr
s_addr	s6_addr

API與資料結構的轉換

- 函式轉換

	IPv4	IPv6

Name-to_address Functions	inet_aton()
	inet_addr()
		inet_pton()

	inet_ntoa()	inet_ntop()

Address conversion Functions	gethostbyname()
	gethostbyaddr()
		getipnodebyname()
		getipnodebyaddr()
		getnameinfo()
		getaddrinfo()

Data Structure Comparison

- AF independent
  - struct sockaddr
- IPv4 dependent
  - struct in_addr
  - struct sockaddr_in
- Name resolving
  - struct hostent

IPv4

- AF independent
  - struct sockaddr_storage
- IPv6 dependent
  - struct in6_addr
  - struct sockaddr_in6
- Name resolving
  - struct addrinfo

IPv6

Definitions and Function Calls

- Address Family&Protocol Family
  - AF_INET6 & PF_INET6 for IPv6
- No changes to transport socket APIs
  - socket(), connect(), bind()……
- Name resolving
  - AF dependent functions are obsolete
  - New AF independent functions
  - gethostbyname() and gethostbyaddr()- IPv4-only
  - getaddrinfo() and getnameinfo()- IPv4 & IPv6

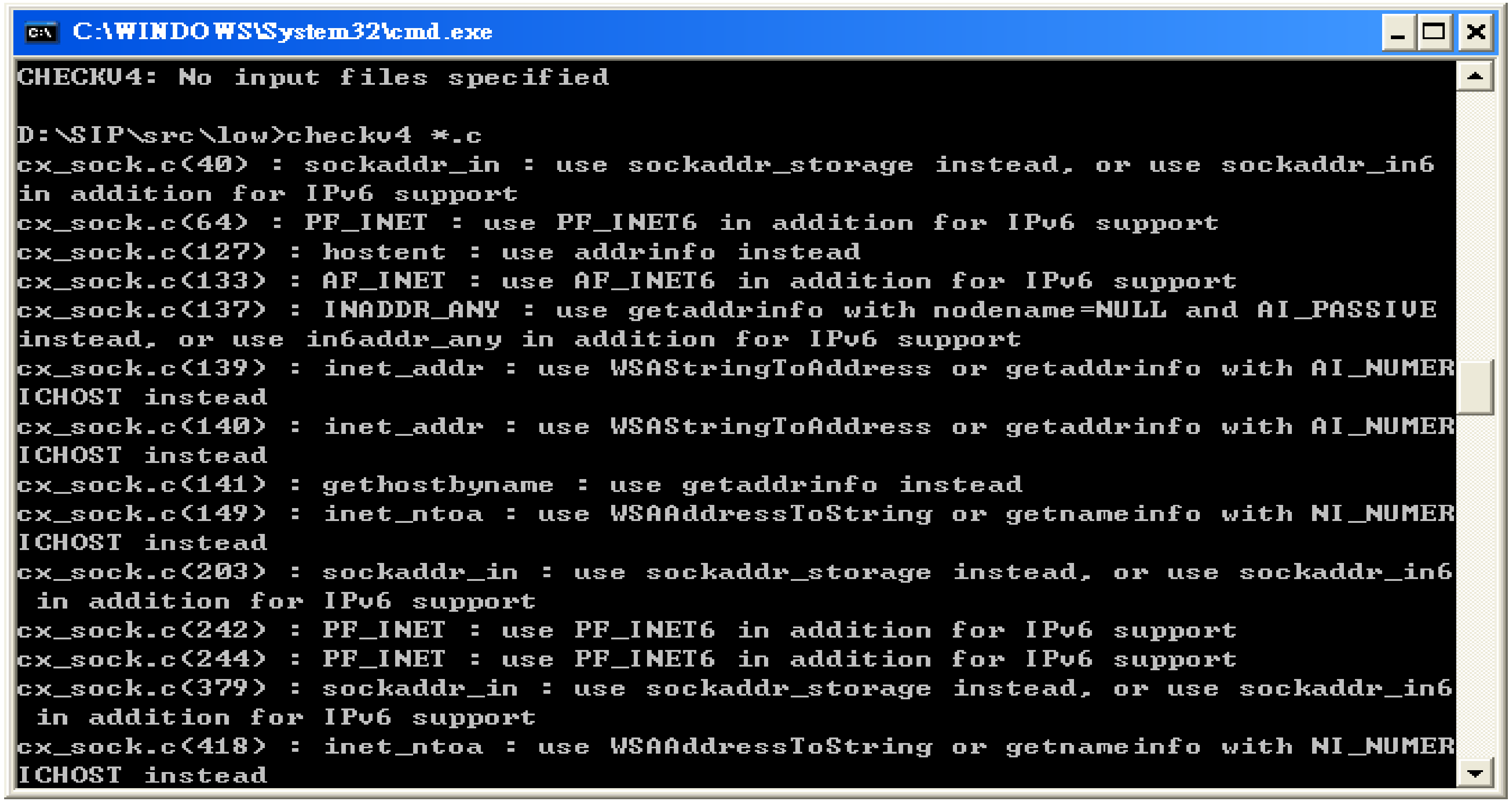
getaddrinfo() & getnameinfo()

- Convert strings storing address and service into sockaddr structure
  - getaddrinfo(“www.kame.net”,”www”,&hint,&res);
- Options are specified in hint
  - hint is an addrinfo structure
- Results are returned as a linked-list, each list node contains a sockaddr structure
- freeaddrinfo() to free returned linked-list
  - freeaddrinfo(res);
- getnameinfo() converts from sockaddr into strings storing address and service
  - getnameinfo(sa,name,sizeof(name),srv,sizeof(srv),0);

Introduction to Checkv4.exe

- Provided by Microsoft
- Identifies potential problems in codes and makes recommendations
- Identifies most trivial problems
  - Successfully checks presence of IPv4 specified code. e.g. gethostbyname(), struct sockaddr_in, and so on.
- Gives some false alert
  - Identifies parameters in comment
- Results from Checkv4.exe
  - About 200 lines for CCL/ITRI SkinUA

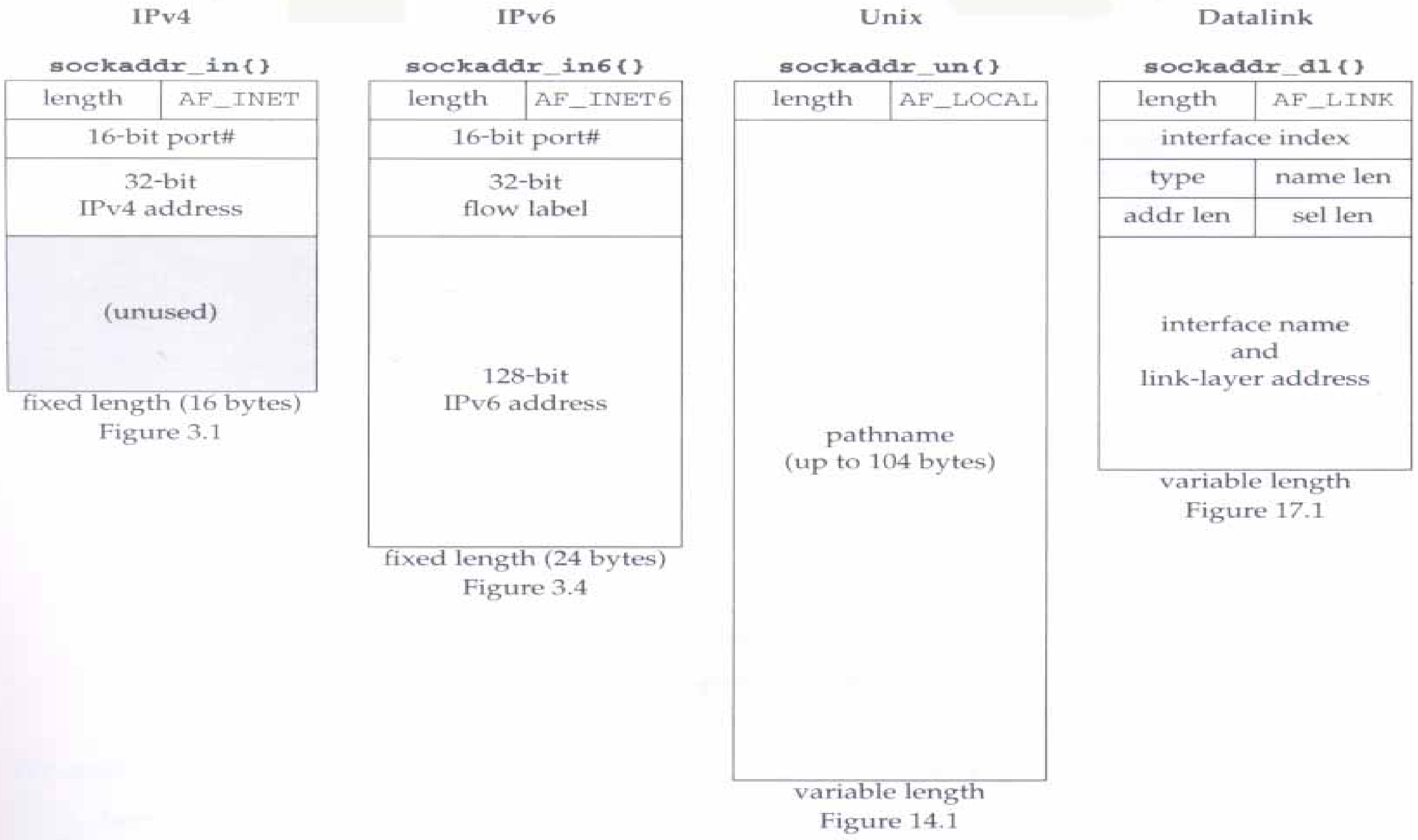
Checkv4.exe (Partial Results)



```
CHECKV4: No input files specified

D:\SIP\src\low>checkv4 *.c
cx_sock.c(40) : sockaddr_in : use sockaddr_storage instead, or use sockaddr_in6
in addition for IPv6 support
cx_sock.c(64) : PF_INET : use PF_INET6 in addition for IPv6 support
cx_sock.c(127) : hostent : use addrinfo instead
cx_sock.c(133) : AF_INET : use AF_INET6 in addition for IPv6 support
cx_sock.c(137) : INADDR_ANY : use getaddrinfo with nodename=NULL and AI_PASSIVE
instead, or use in6addr_any in addition for IPv6 support
cx_sock.c(139) : inet_addr : use WSAStringToAddress or getaddrinfo with AI_NUMER
ICHOST instead
cx_sock.c(140) : inet_addr : use WSAStringToAddress or getaddrinfo with AI_NUMER
ICHOST instead
cx_sock.c(141) : gethostbyname : use getaddrinfo instead
cx_sock.c(149) : inet_ntoa : use WSAAddressToString or getnameinfo with NI_NUMER
ICHOST instead
cx_sock.c(203) : sockaddr_in : use sockaddr_storage instead, or use sockaddr_in6
 in addition for IPv6 support
cx_sock.c(242) : PF_INET : use PF_INET6 in addition for IPv6 support
cx_sock.c(244) : PF_INET : use PF_INET6 in addition for IPv6 support
cx_sock.c(379) : sockaddr_in : use sockaddr_storage instead, or use sockaddr_in6
 in addition for IPv6 support
cx_sock.c(418) : inet_ntoa : use WSAAddressToString or getnameinfo with NI_NUMER
ICHOST instead
```

Comparison of socket address structure



IPv4

sockaddr_in{}

length | AF_INET

16-bit port#

32-bit IPv4 address

(unused)

fixed length (16 bytes)
Figure 3.1

IPv6

sockaddr_in6{}

length | AF_INET6

16-bit port#

32-bit flow label

128-bit IPv6 address

fixed length (24 bytes)
Figure 3.4

Unix

sockaddr_un{}

length | AF_LOCAL

pathname (up to 104 bytes)

variable length
Figure 14.1

Datalink

sockaddr_dl{}

length | AF_LINK

interface index

type | name len

addr len | sel len

interface name and link-layer address

variable length
Figure 17.1

Figure 3.5 Comparison of various socket address structures.

Socket address structure pass.

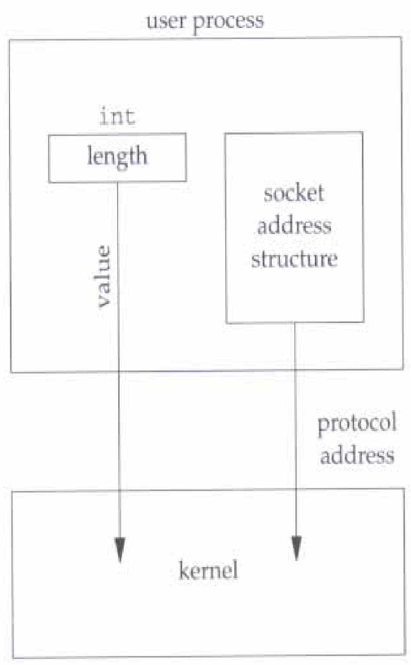


Figure 3.6 Socket address structure passed from process to kernel.

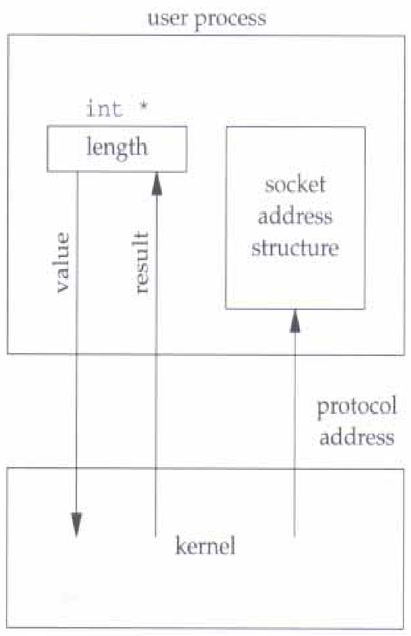


Figure 3.7 Socket address structure passed from kernel to process.

bind, connect, sendto

accept, recvfrom, getsockname, getpeername

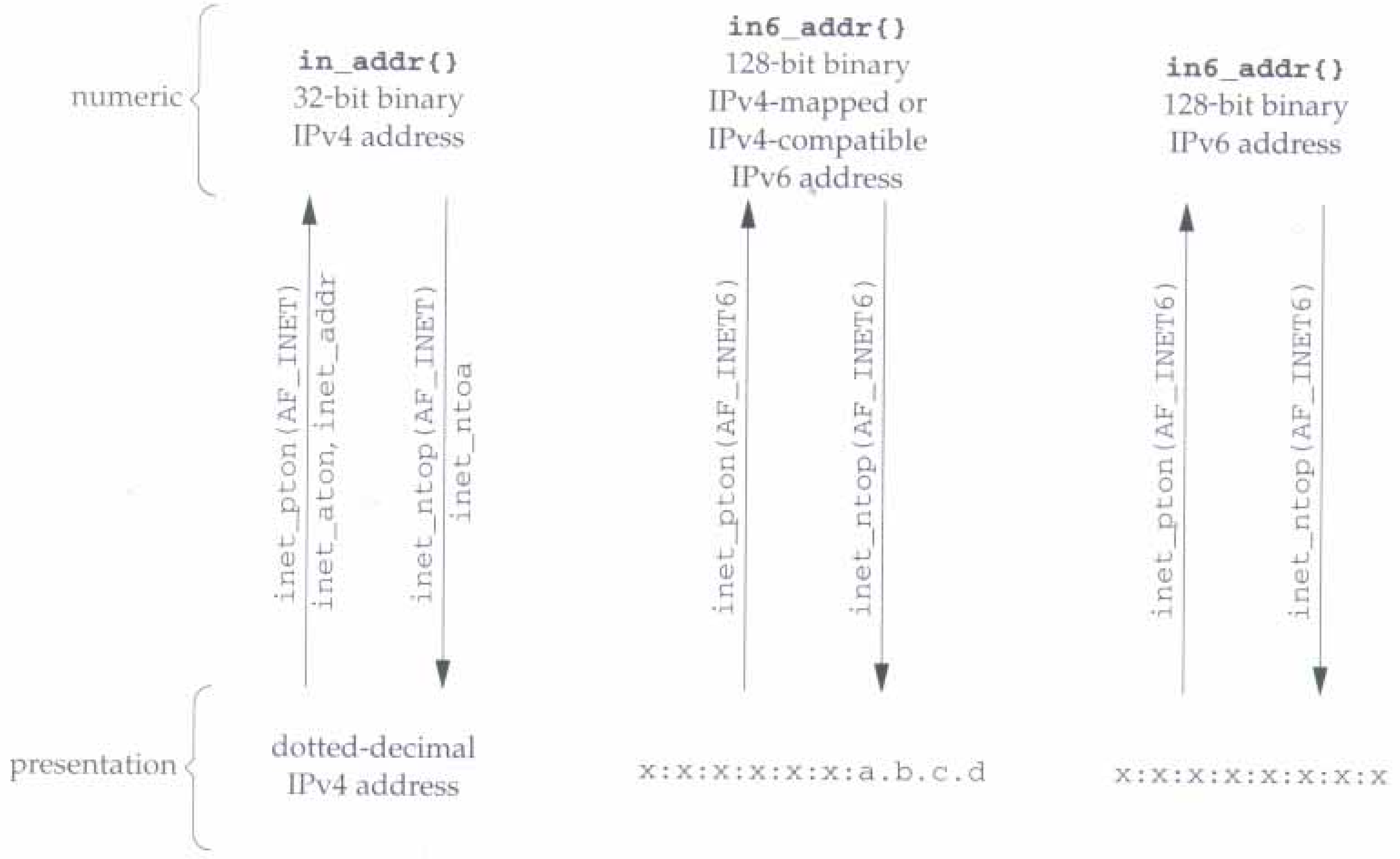
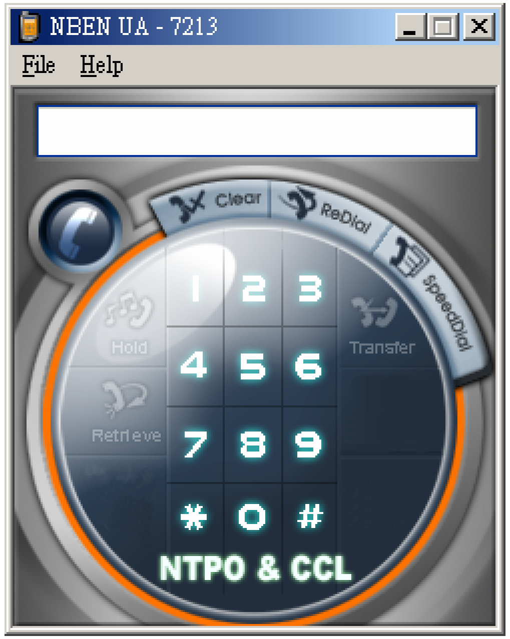


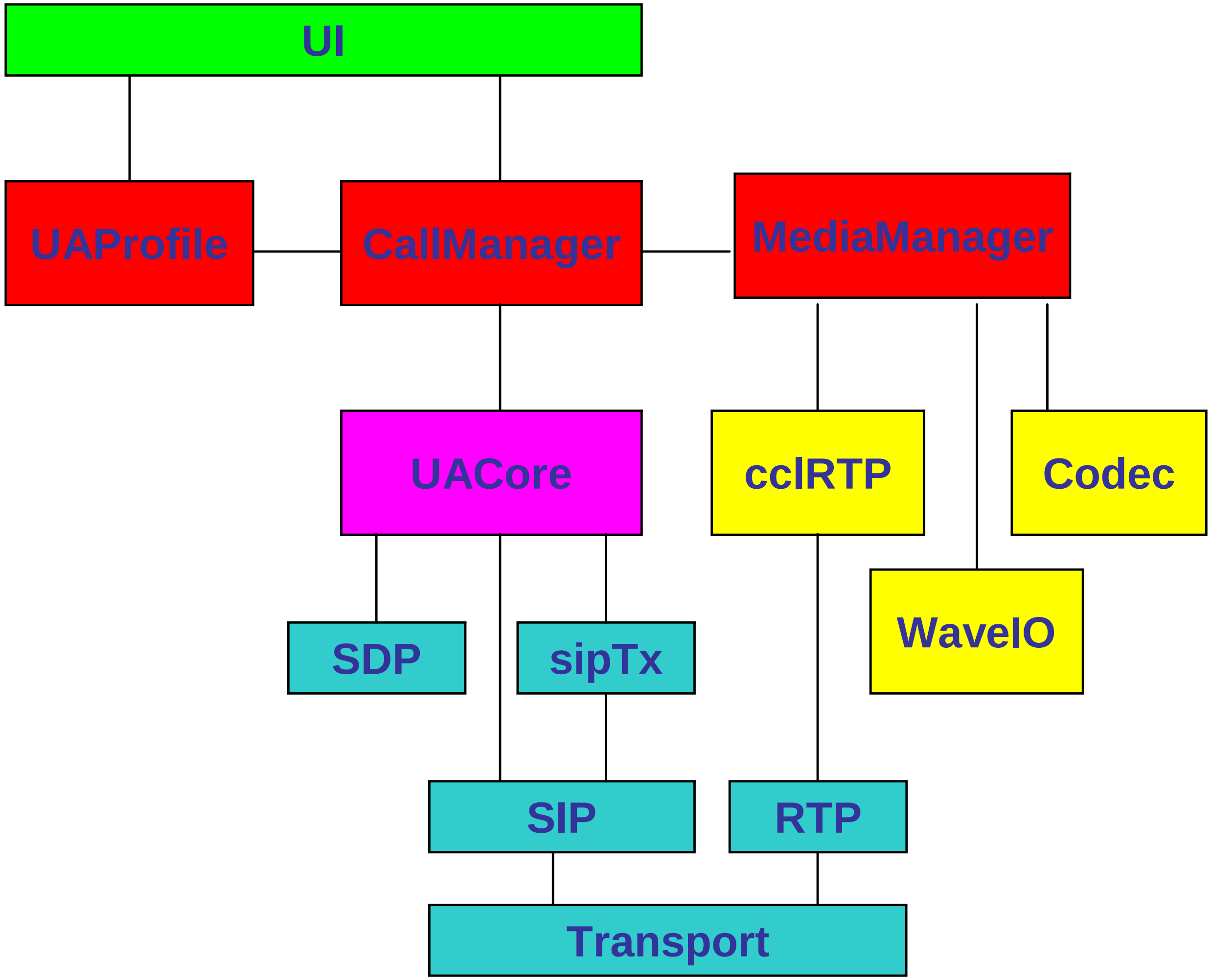
Figure 3.10 Summary of address conversion functions.

NTPO&CCL SIP User Agent (UA)

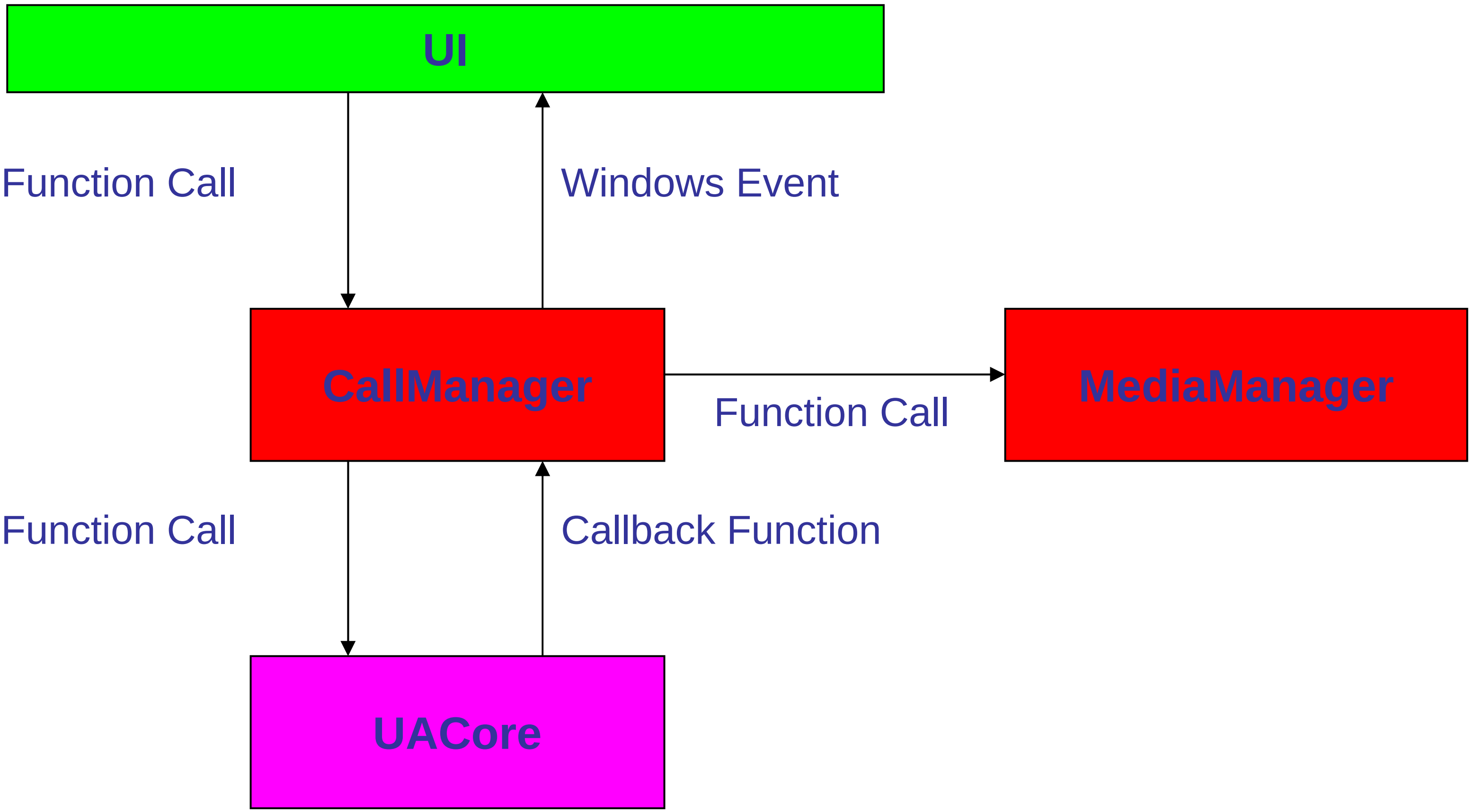
- SIP-based VoIP phone running on Windows
- Support H.263 Video codec
- Support G.711u/G.711a/G.723/G.729 Audio codec
- Support registration
- Support authentication



Structure of SIP UA

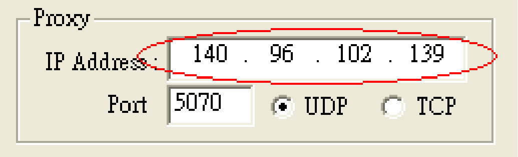


Component Relationship of CallManager



GUI Problem

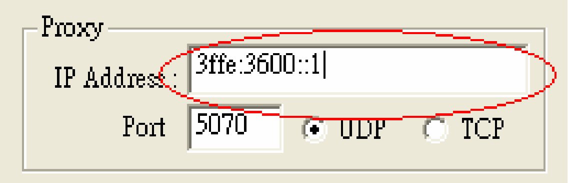
- IP Address control
  - Is IPv4 specified
  - Do not accept domain name
- Use Edit control instead



Proxy

IP Address : 140 . 96 . 102 . 139

Port 5070 ● UDP ○ TCP



Proxy

IP Address : 3ffe:3600::1

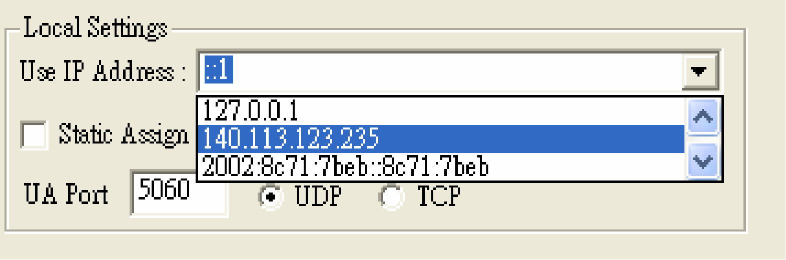
Port 5070 ● UDP ○ TCP

Get Local Address (1/2)

- Old method: gethostbyname()
  - Gethostbyname() on local hostname
- Does getaddrinfo() on local hostname works?
  - Not works on Windows XP
  - Works on Windows 2003

Get Local Address (2 of 2)

- Make use of IPHelper functions
  - Presented in Windows from Windows 98
  - A Windows-only solution
  - Works on both windows XP and 2003
- Function name: GetAdaptersAddresses()



Local Settings

Use IP Address : ::1

127.0.0.1
140.113.123.235
2002:8c71:7beb::8c71:7beb

□ Static Assign

UA Port 5060 ● UDP ○ TCP

Parsing URI with IPv6

- IPv6 address in URI
  - sip:wechen@[3ffe:1345:5643::3]:5060
- Some parser assume semicolon will be used only to separate IP and Port
- Modify parsing algorithm to deal with IPv6 address.
- URI in SIP header may contains IPv6 address
  - INVITE sip:wechen@[2001:238:f82:66::33]:5060
- **IP6** addrtype & IPv6 address in SDP
  - c=IN IP6 FE80:60::2

Goal of Porting SIP UA to IPv6

- Provide IPv6 communication to Users

  (a long-term solution)
- SIP UA should accept SIP URI that contains IPv6 literal address (specified in RFC 3261)
- SIP UA should correctly handle IPv6 addresses in SIP/SDP header fields
- SIP UA should operate with other IPv6 SIP UAs (KPhone and LinPhone) and SIP servers (IPtel and Partysip).

Modifications for SIP User Agent

- Auto IPv4/IPv6 negotiation requires modification in listening thread part and rewrite working flow of calling
    - The IP version is the same as the IP address that user choose
    - SIP UA will use either IPv4 or IPv6 at the same time.
    - Lower part in protocol stack should check an extra parameter that specifies address family

Modifications for SIP User Agent (cont.)

- IPv6 address Literal format has scope-id
  - E.g. fe80::201:2ff:fe85:37ed%3
  - Used by linked-local address
  - Identify the same address on different interface
- Scope-id must be specified when connecting to sites using link-local address
  - An extra parameter in data structure to keep this

Modifications for SIP User Agent (cont.)

- SIP URI may contain IPv6 address
  - E.g. sip:wechen@[2001:238:f82:6::2]:5060
  - Rewrite parser to ensure correctly dealing with colon
- Since IPv6 address are longer than IPv4 address, GUI components related to address should be modified
- Avoid using *IPAddressControl* that supports IPv4 address only

Results

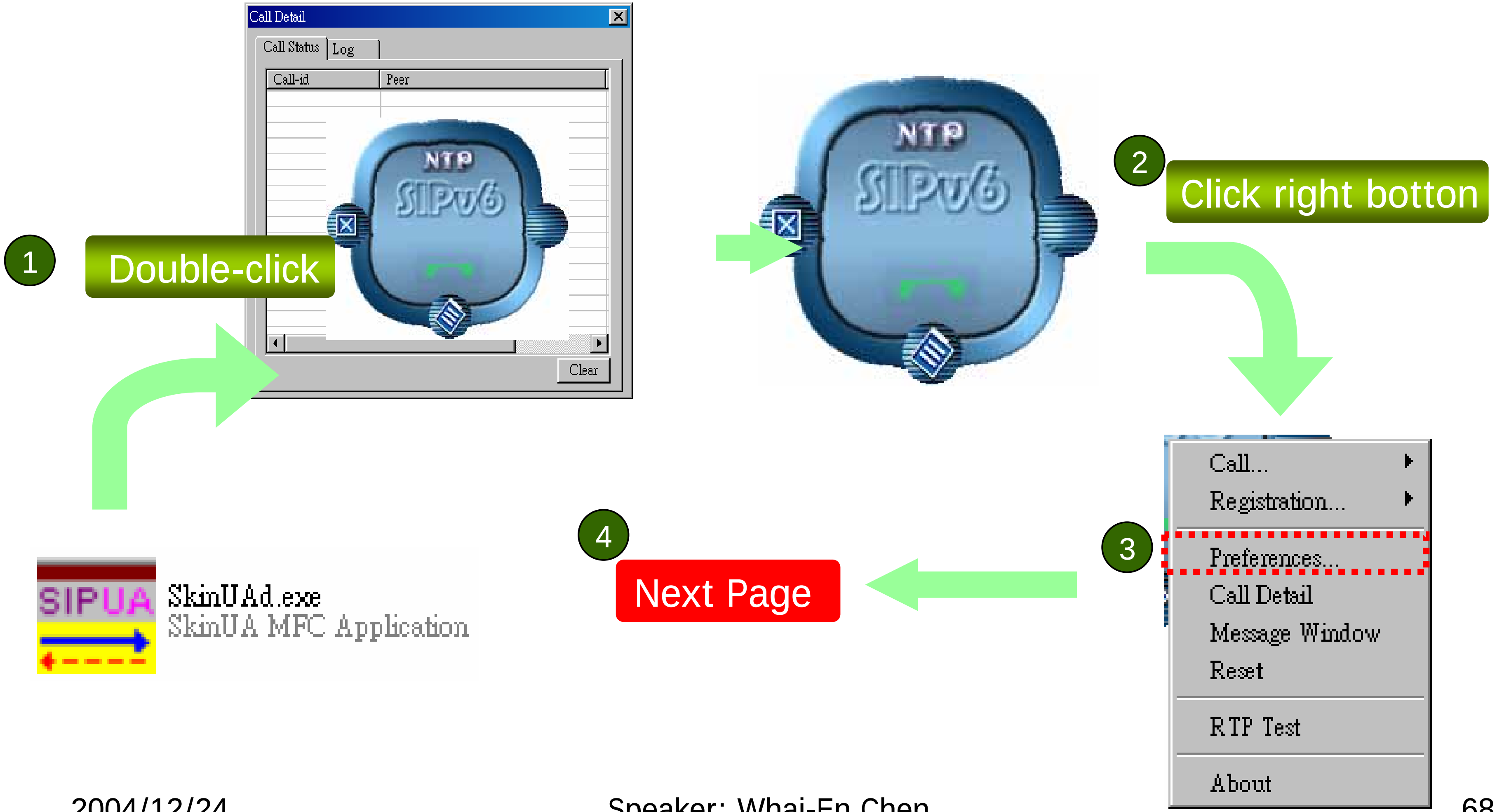
- Changes 500+ out of 100,000+ lines in 150 files
- About 300 lines are not identified by checkv4.exe
- SIP UA supports
  - IPv4 or IPv6 communication
  - IPv6 address in SIP URI
  - IPv6 address in GUI and form
- Modifications in SIP UA
  - Transport – handle different IP versions
  - GUI – handle IPv6 address
  - CallManager – URI parsing/generating

Modification Summary

Module name	Modified files
UACore	5
sipTX	4
sip	5
sdp	1
rtp	6
transport	6
cclRTP	2
MediaManager	4
UI	4
UAProfile	2

Total: 39 files

啟動SIPv6 User Agent



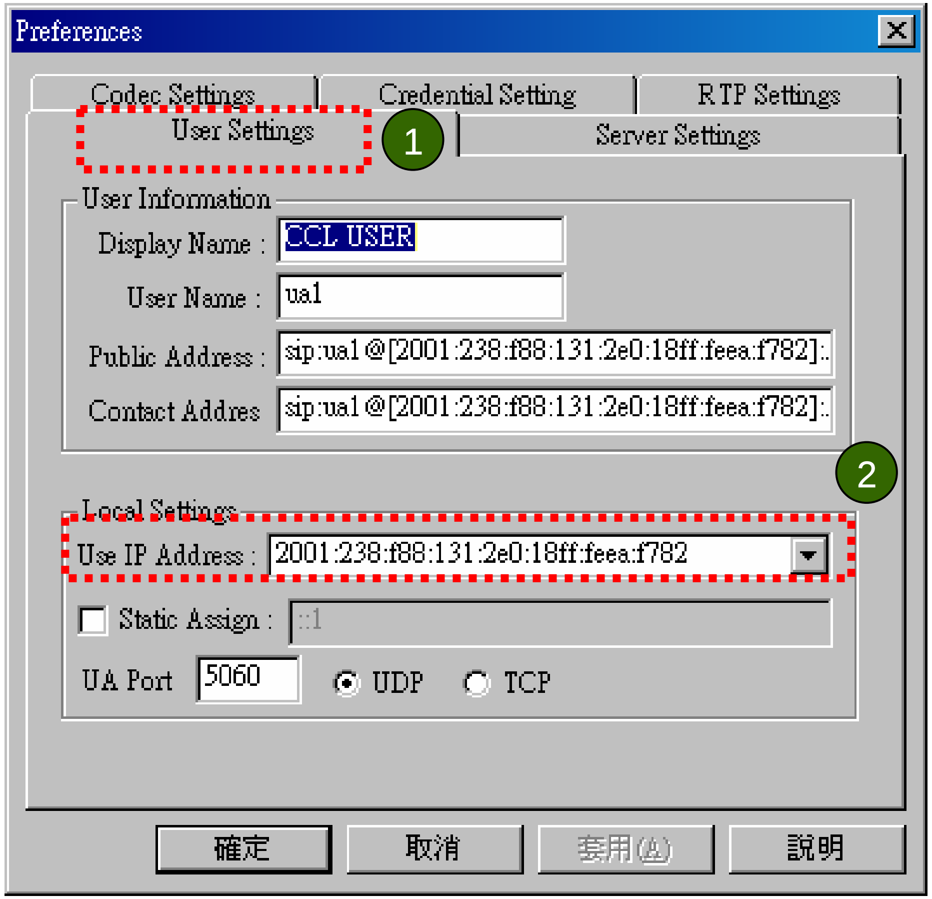
1 Double-click

2 Click right botton

3

4 Next Page

設定SIPv6 User Agent的IPv6位址



User Information

Display Name : CCL USER

User Name : ua1

Public Address : sip:ua1@[2001:238:f88:131:2e0:18ff:feea:f782]:.

Contact Addres : sip:ua1@[2001:238:f88:131:2e0:18ff:feea:f782]:.

Local Settings

Use IP Address : 2001:238:f88:131:2e0:18ff:feea:f782

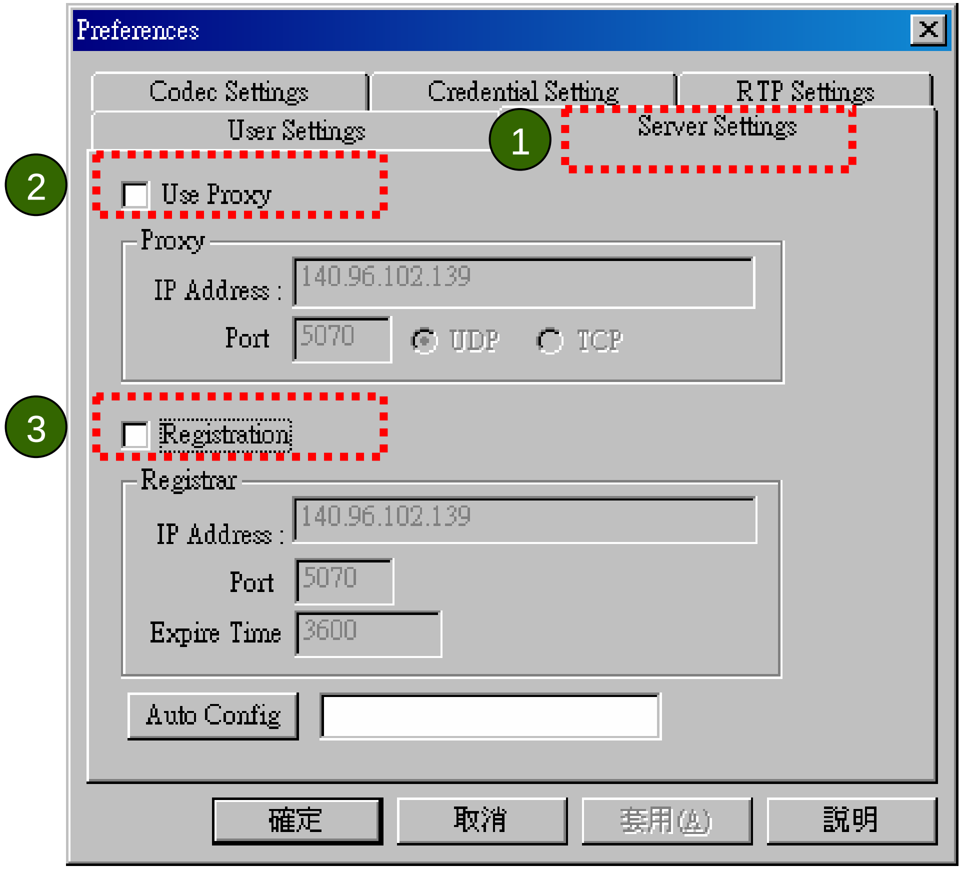
□ Static Assign : ::1

UA Port 5060 ● UDP ○ TCP

1. 選擇「User Settings」分頁
2. 在「User IP Address」選項中，選擇Global Unicast IPv6 Address（如：2001:238:f88:131:2e0:18ff:feea:f782）
3. 如果要跨越IPv4網路，則需要選擇6to4位址（Prefix是2002::/16）

3 Next Page

設定SIPv6 User Agent的伺服器



□ Use Proxy

Proxy

IP Address : 140.96.102.139

Port 5070 ● UDP ○ TCP

□ Registration

Registrar

IP Address : 140.96.102.139

Port 5070

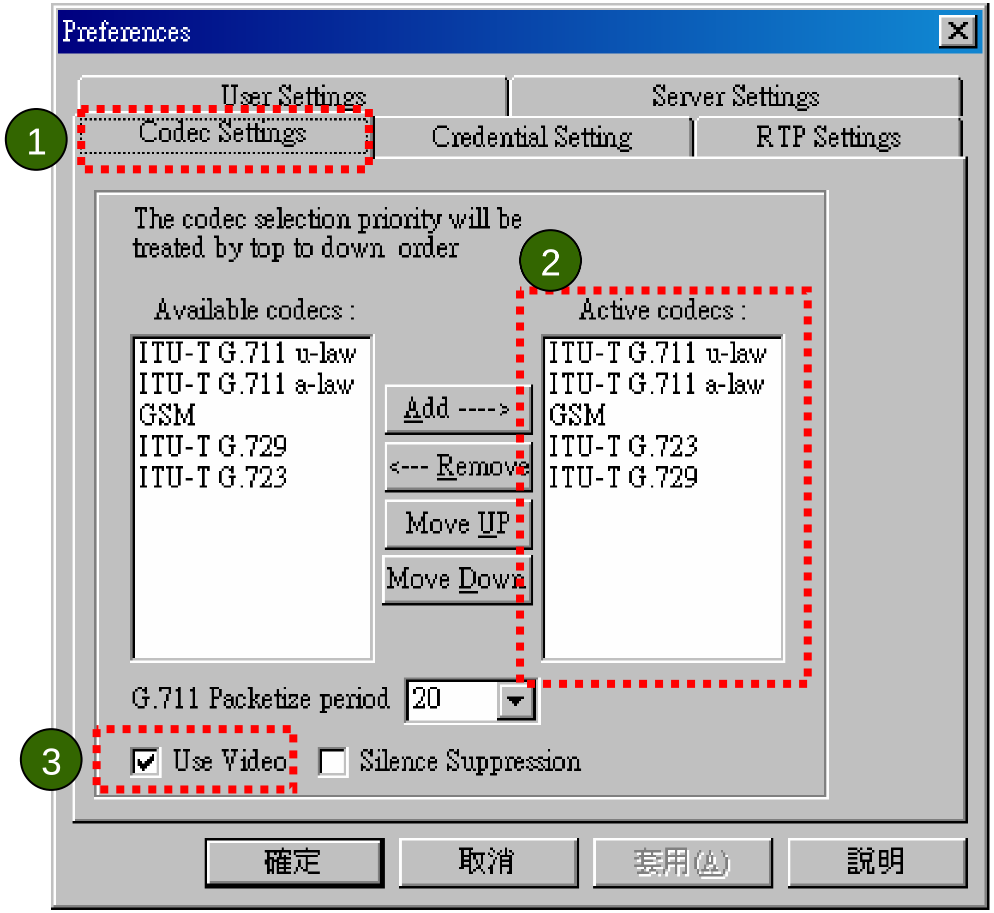
Expire Time 3600

1. 選擇「Server Settings」分頁
2. 取消「Use Proxy」選項
3. 取消「Registration」；若是有IPv6 SIP伺服器，則可以選取選項，並填入伺服器的IPv6位址

4

Next Page

設定SIPv6 User Agent的Codec參數



Preferences

The codec selection priority will be treated by top to down order

Available codecs :	Active codecs :
ITU-T G.711 u-law	ITU-T G.711 u-law
ITU-T G.711 a-law	ITU-T G.711 a-law
GSM	GSM
ITU-T G.729	ITU-T G.723
ITU-T G.723	ITU-T G.729

G.711 Packetize period: 20

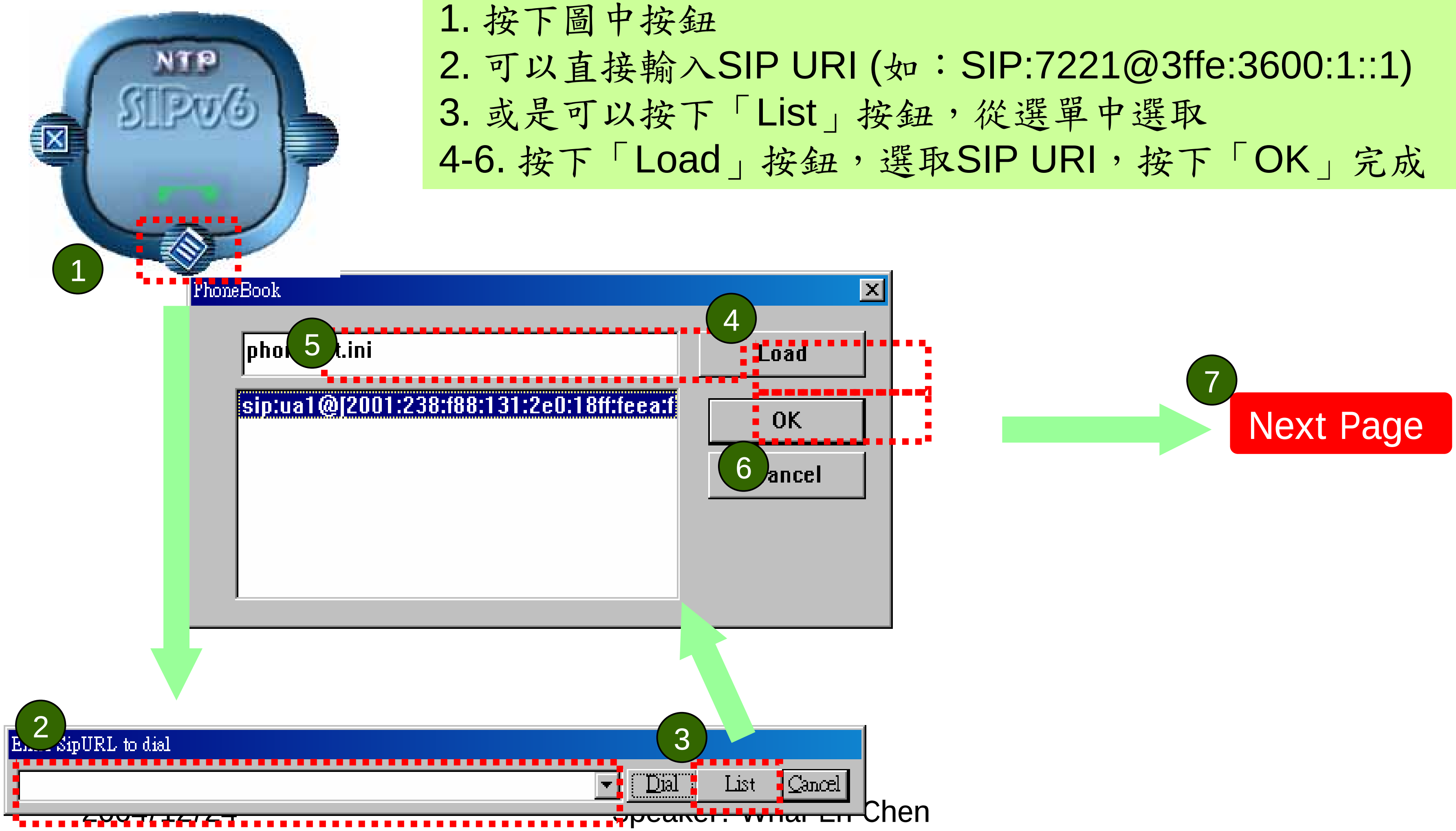
☑ Use Video ☐ Silence Suppression

1. 選擇「Codec Settings」分頁
2. 將要用的Codec放入「Active Codecs」選項中
3. 選取「Use Video」，若不需要影像則可以取消此選項
4. 按下「確定」按鈕，完成設定

4 Next Page

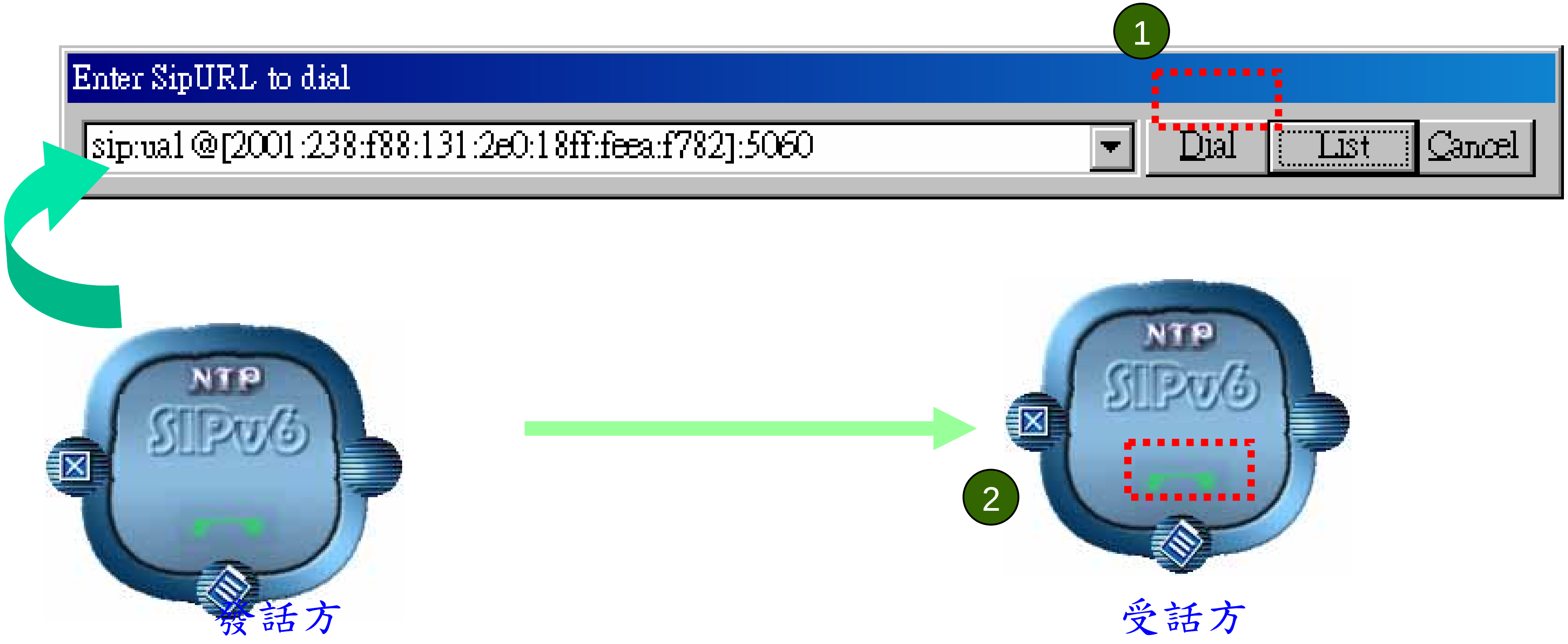
開始撥號 (輸入SIP URI)

1. 按下圖中按鈕
2. 可以直接輸入SIP URI (如：SIP:7221@3ffe:3600:1::1)
3. 或是可以按下「List」按鈕，從選單中選取
4-6. 按下「Load」按鈕，選取SIP URI，按下「OK」完成



sip:ua1@[2001:238:f88:131:2e0:18ff:feea:f

撥號與接聽

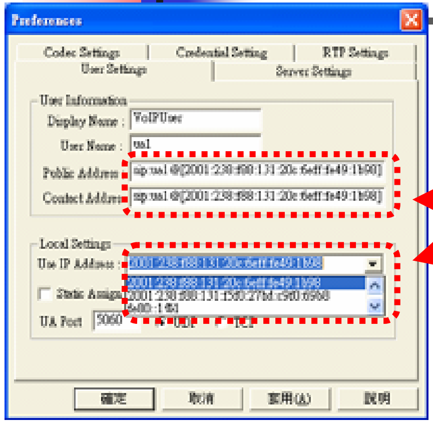


Enter SipURL to dial

sip:ua1@[2001:238:f88:131:2e0:18ff:feea:f782]:5060

1. 按下「Dial」按鈕，開始撥號
2. 受話方案下圖中電話筒圖案即可接聽

展示項目 - SIPv6 User Agent (UA) 移植成果


Preferences

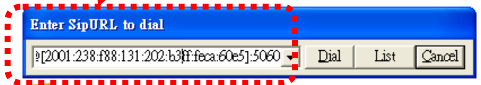
Codec Settings | Credential Setting | RTP Settings | User Settings | Server Settings

User Information
Display Name : VoIPUser
User Name : ua1
Public Address : sip:ua1@[2001:238:f88:131:20c:6eff:fe49:1b98]
Contact Address : sip:ua1@[2001:238:f88:131:20c:6eff:fe49:1b98]

Local Settings
Use IP Address : 2001:238:f88:131:20c:6eff:fe49:1b98
2001:238:f88:131:20c:6eff:fe49:1b98
2001:238:f88:131:1580:27bd:c980:6988
fe80::1%1
□ Static Assign
UA Port 5060 ○ UDP ○ TCP

確定 | 取消 | 套用(A) | 說明

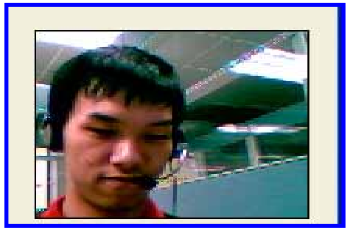
Using IPv6 Addresses

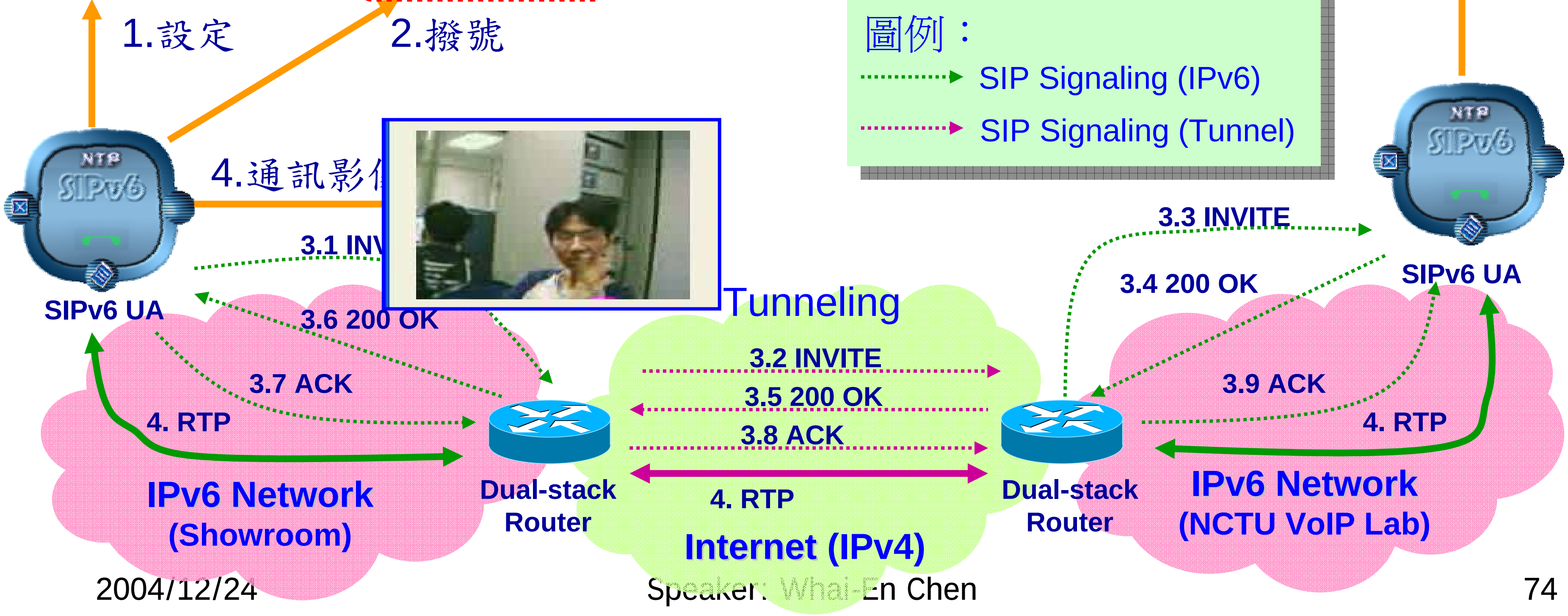

Enter SipURL to dial

p[2001:238:f88:131:202:b3ff:feca:60e5]:5060

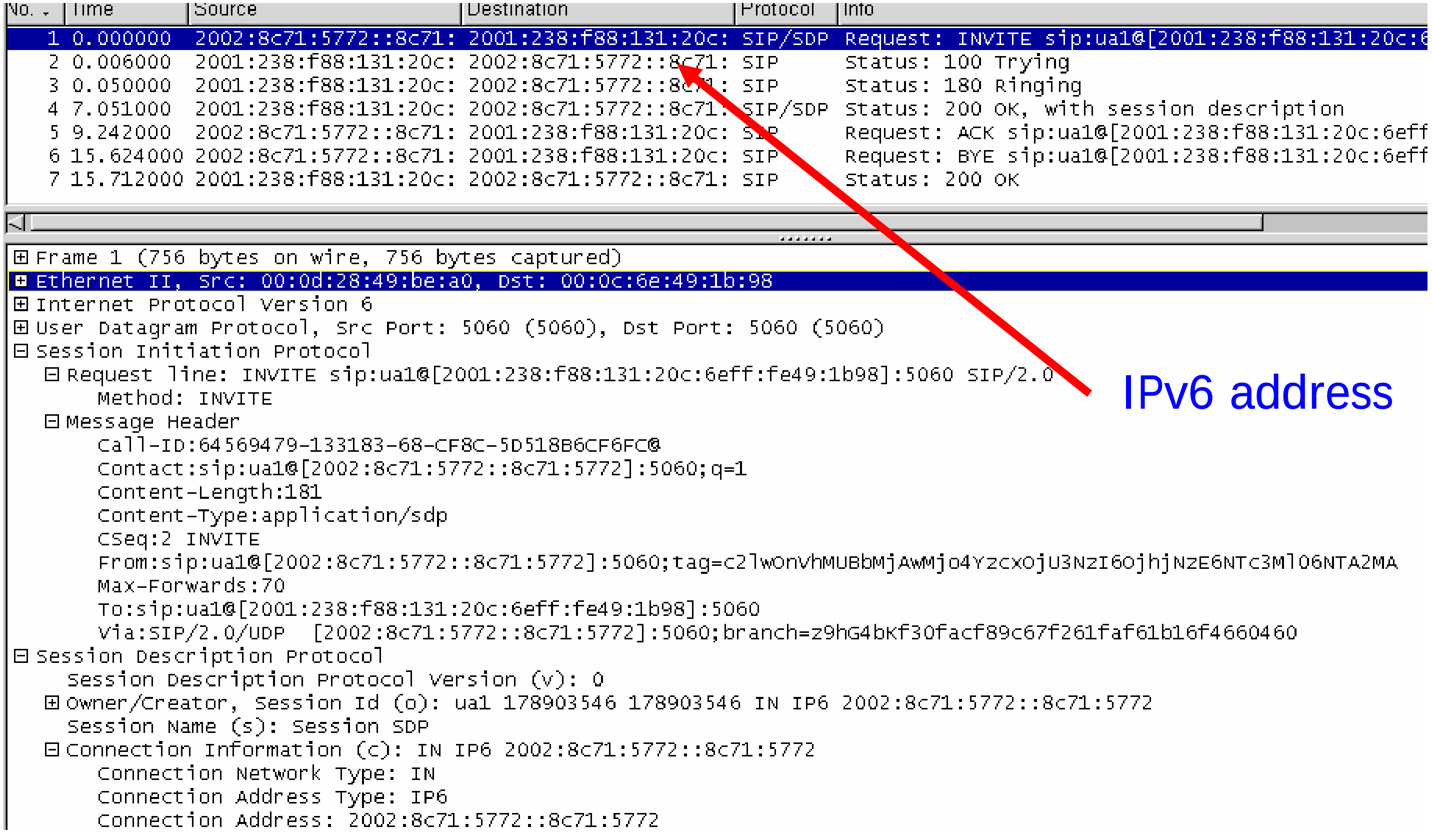
Dial | List | Cancel

4.通訊影像





SIP Messages over IPv6

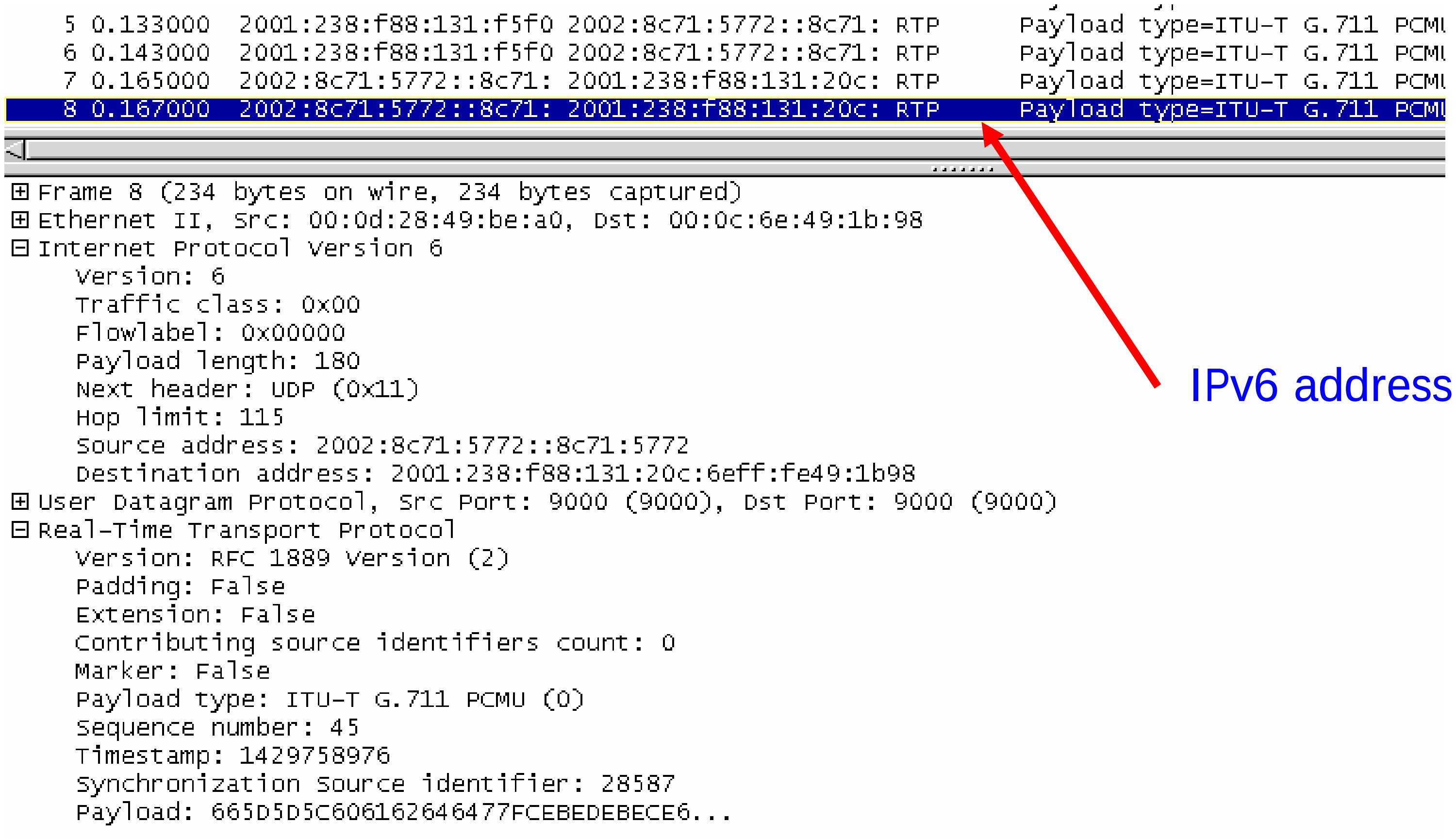


```
No. . | Time      | Source               | Destination          | Protocol | Info
    1 0.000000  2002:8c71:5772::8c71: 2001:238:f88:131:20c: SIP/SDP  Request: INVITE sip:ua1@[2001:238:f88:131:20c:6
    2 0.006000  2001:238:f88:131:20c: 2002:8c71:5772::8c71: SIP      Status: 100 Trying
    3 0.050000  2001:238:f88:131:20c: 2002:8c71:5772::8c71: SIP      Status: 180 Ringing
    4 7.051000  2001:238:f88:131:20c: 2002:8c71:5772::8c71: SIP/SDP  Status: 200 OK, with session description
    5 9.242000  2002:8c71:5772::8c71: 2001:238:f88:131:20c: SIP      Request: ACK sip:ua1@[2001:238:f88:131:20c:6eff
    6 15.624000 2002:8c71:5772::8c71: 2001:238:f88:131:20c: SIP      Request: BYE sip:ua1@[2001:238:f88:131:20c:6eff
    7 15.712000 2001:238:f88:131:20c: 2002:8c71:5772::8c71: SIP      Status: 200 OK

⊞ Frame 1 (756 bytes on wire, 756 bytes captured)
⊞ Ethernet II, Src: 00:0d:28:49:be:a0, Dst: 00:0c:6e:49:1b:98
⊞ Internet Protocol Version 6
⊞ User Datagram Protocol, Src Port: 5060 (5060), Dst Port: 5060 (5060)
⊟ Session Initiation Protocol
  ⊟ Request line: INVITE sip:ua1@[2001:238:f88:131:20c:6eff:fe49:1b98]:5060 SIP/2.0
      Method: INVITE
  ⊟ Message Header
      Call-ID:64569479-133183-68-CF8C-5D518B6CF6FC@
      Contact:sip:ua1@[2002:8c71:5772::8c71:5772]:5060;q=1
      Content-Length:181
      Content-Type:application/sdp
      CSeq:2 INVITE
      From:sip:ua1@[2002:8c71:5772::8c71:5772]:5060;tag=c2lwOnVhMUBbMjAwMjo4YzcxOjU3NzI6OjhjNzE6NTc3Ml06NTA2MA
      Max-Forwards:70
      To:sip:ua1@[2001:238:f88:131:20c:6eff:fe49:1b98]:5060
      Via:SIP/2.0/UDP  [2002:8c71:5772::8c71:5772]:5060;branch=z9hG4bKf30facf89c67f261faf61b16f4660460
⊟ Session Description Protocol
    Session Description Protocol Version (v): 0
  ⊞ Owner/Creator, Session Id (o): ua1 178903546 178903546 IN IP6 2002:8c71:5772::8c71:5772
    Session Name (s): Session SDP
  ⊟ Connection Information (c): IN IP6 2002:8c71:5772::8c71:5772
      Connection Network Type: IN
      Connection Address Type: IP6
      Connection Address: 2002:8c71:5772::8c71:5772
```

IPv6 address

RTP Stream over IPv6



```
  5 0.133000  2001:238:f88:131:f5f0 2002:8c71:5772::8c71: RTP     Payload type=ITU-T G.711 PCMU
  6 0.143000  2001:238:f88:131:f5f0 2002:8c71:5772::8c71: RTP     Payload type=ITU-T G.711 PCMU
  7 0.165000  2002:8c71:5772::8c71: 2001:238:f88:131:20c: RTP     Payload type=ITU-T G.711 PCMU
  8 0.167000  2002:8c71:5772::8c71: 2001:238:f88:131:20c: RTP     Payload type=ITU-T G.711 PCMU

⊞ Frame 8 (234 bytes on wire, 234 bytes captured)
⊞ Ethernet II, Src: 00:0d:28:49:be:a0, Dst: 00:0c:6e:49:1b:98
⊟ Internet Protocol Version 6
    Version: 6
    Traffic class: 0x00
    Flowlabel: 0x00000
    Payload length: 180
    Next header: UDP (0x11)
    Hop limit: 115
    Source address: 2002:8c71:5772::8c71:5772
    Destination address: 2001:238:f88:131:20c:6eff:fe49:1b98
⊞ User Datagram Protocol, Src Port: 9000 (9000), Dst Port: 9000 (9000)
⊟ Real-Time Transport Protocol
    Version: RFC 1889 Version (2)
    Padding: False
    Extension: False
    Contributing source identifiers count: 0
    Marker: False
    Payload type: ITU-T G.711 PCMU (0)
    Sequence number: 45
    Timestamp: 1429758976
    Synchronization Source identifier: 28587
    Payload: 665D5D5C606162646477FCEBEDEBECE6...
```

IPv6 address

Interoperability Testing

- Testing with 2 Linux SIP-based phone
  - Kphone 3.2 with IPv6 (patched by iptel)
  - Linphone 0.11.3 (claimed as IPv6 enabled)
- Environment
  - Windows XP SP1
  - Redhat linux 9.0
  - Partysip IPv6 SIP proxy
  - Iptel IPv6-enabled SIP server

Interoperability Testing Results

- To IPv6 SIP proxy

Item	Result
Register on iptel	Succeed
Register on partysip	Succeed
Call UA through partysip proxy server	Succeed

Interoperability Testing Results

- To IPv6 SIP UA

To / From	Kphone	Linphone	SkinUA
KPhone	OK	SIP ok	SIP ok
Linphone	SIP ok	OK	SIP ok
SkinUA	OK	SIP ok	OK

- Linphone & KPhone can not accept URI containing IPv6 Literal address in URI.

IPv6 Translation Mechanism-Bump-In-the-API

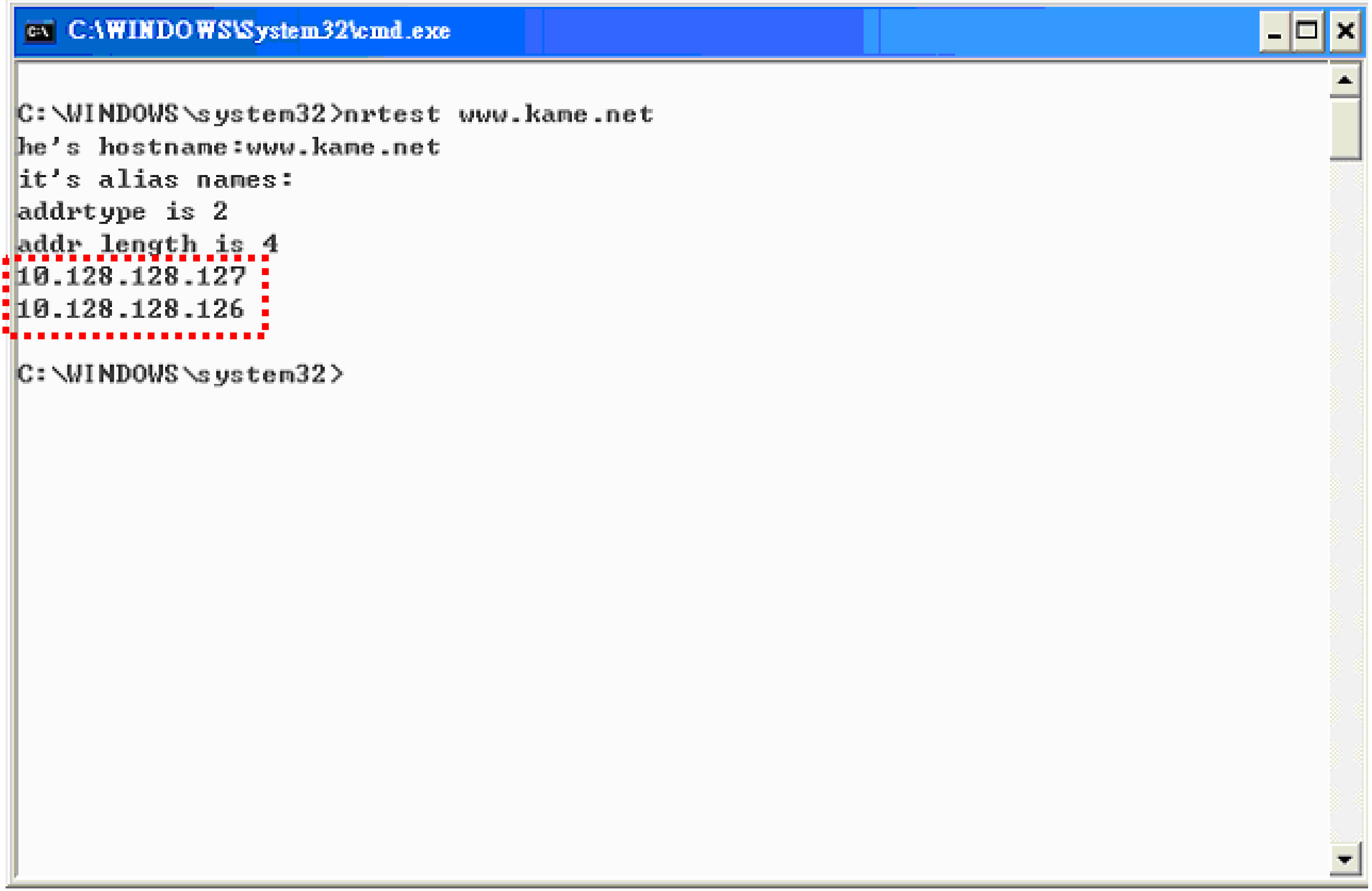
設計主機端轉換之中介軟體

- 可是要將應用程式升級成IPv6會有以下問題
  - 需要改用新的 API
  - 需要改用新的 Data structure
- 以SIP-based VoIP User Agent為例
  - 約有200行Socket API、資料結構需要轉換
  - 共約有600行位址相關函式、變數、記憶體指派需要修改
- 短期內將程式升級IPv6不容易
  - 需要改的函式、變數需要追蹤修訂
  - 程式版本升級時，亦需隨之修訂
- 提出一個轉換v4/v6的中介軟體，以 BIA為基礎，設計應用層轉換機制

軟硬體來源與執行平台

- BIA轉換器元件
  - Function Mapper
  - Name Resolver
  - Address Mapper
  - ALG Manager
  - FTP-ALG
- BIA轉換器的開發平台如下
  - 作業系統: Windows XP SP1
  - 中央處理器: Intel Celeron 2GHz
  - 記憶體: 128 MB
  - 硬碟: 20GB
  - 編譯程式: Microsoft Visual C++ 6.0
  - 開發函式庫: Microsoft Platform SDK February 2003
- BIA可以執行於微軟Windows XP/2003之上

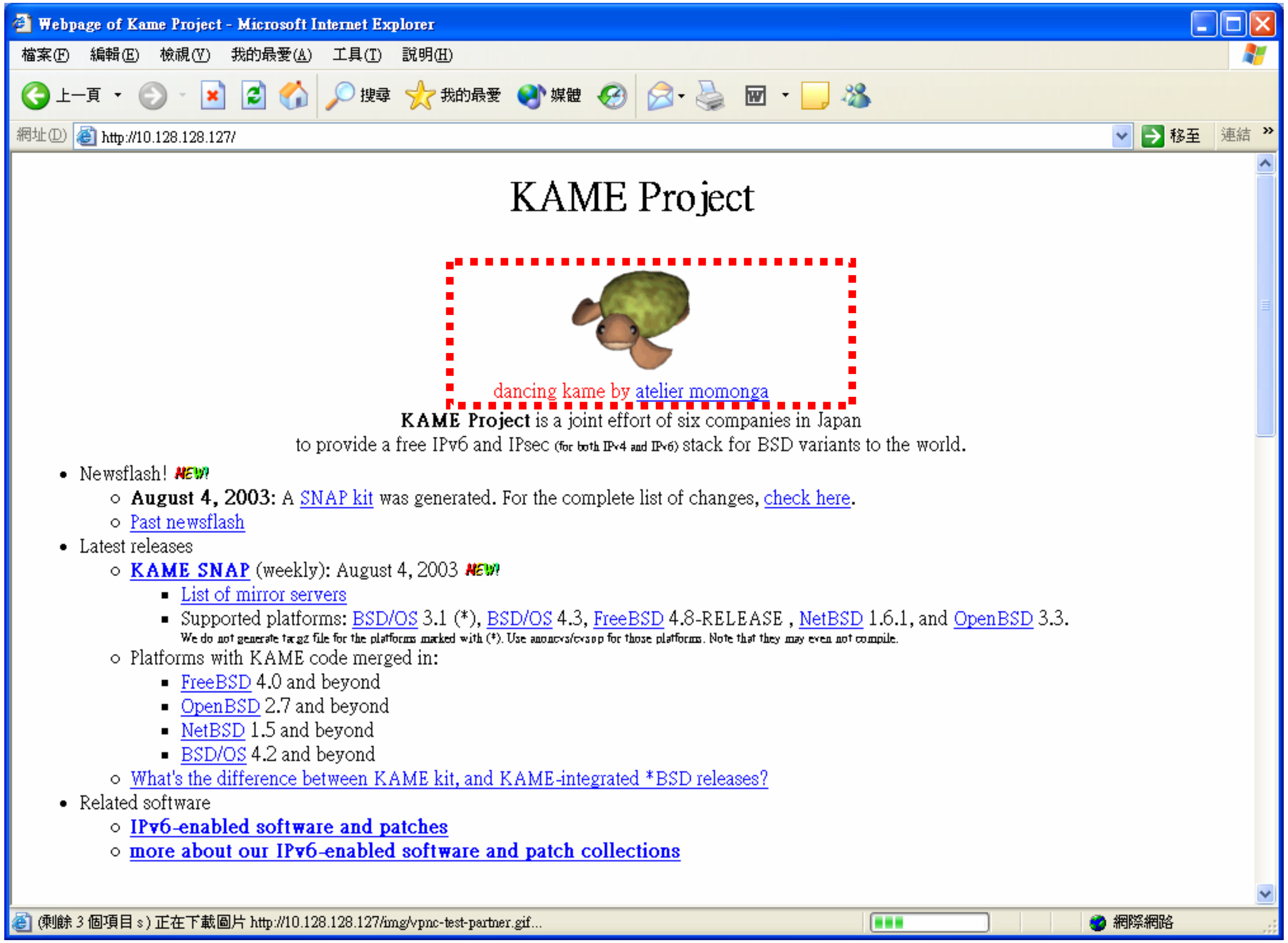
Name Resolving:
Translate IPv6 address to IPv4 address



```
C:\WINDOWS\system32>nrtest www.kame.net
he's hostname:www.kame.net
it's alias names:
addrtype is 2
addr length is 4
10.128.128.127
10.128.128.126

C:\WINDOWS\system32>
```

Socket-layer Translator Result


Webpage of Kame Project - Microsoft Internet Explorer
檔案(F) 編輯(E) 檢視(V) 我的最愛(A) 工具(T) 說明(H)
上一頁
搜尋
我的最愛
媒體
網址(D)
移至
連結

http://10.128.128.127/

KAME Project

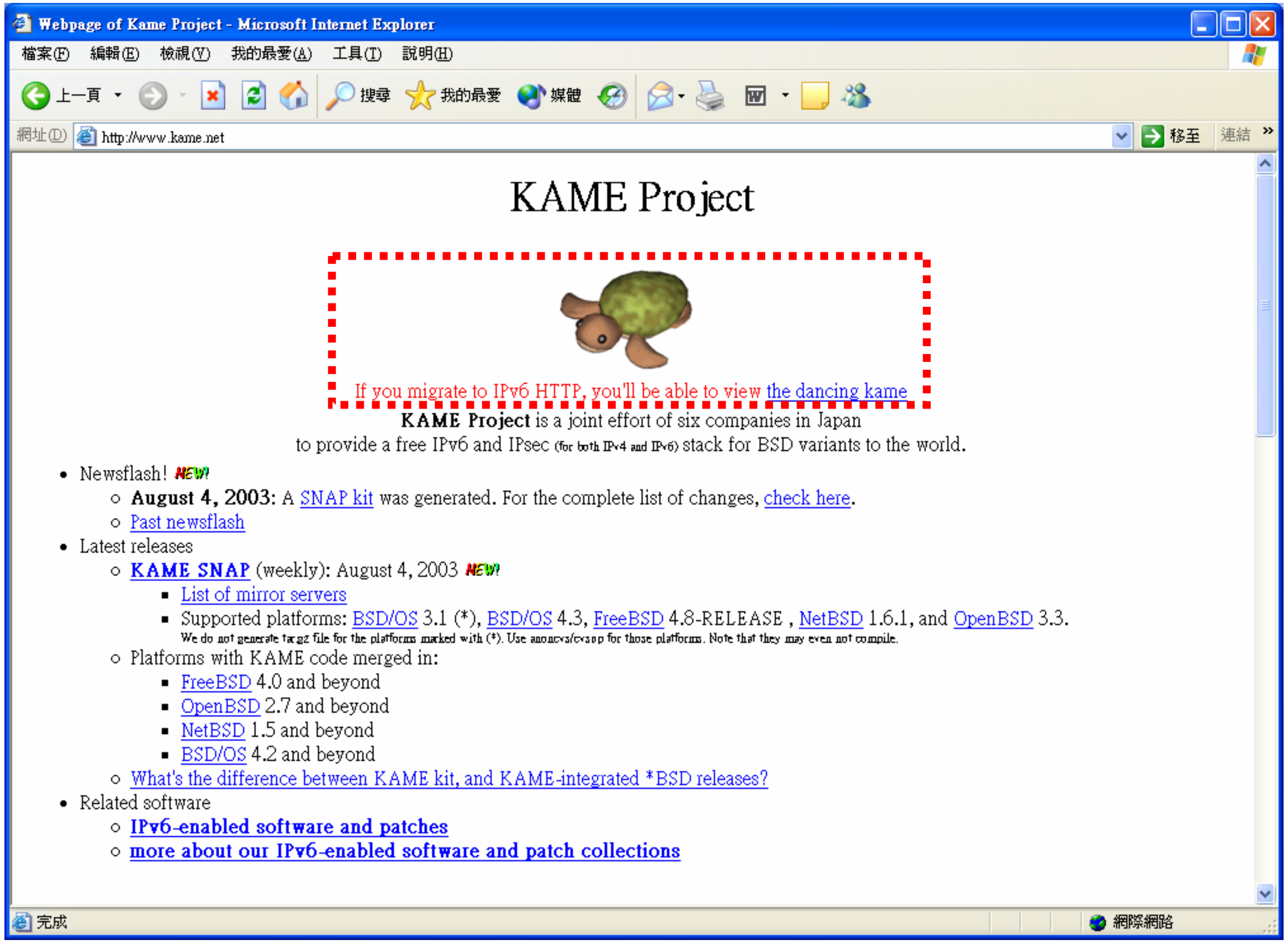
dancing kame by atelier momonga

KAME Project is a joint effort of six companies in Japan
to provide a free IPv6 and IPsec (for both IPv4 and IPv6) stack for BSD variants to the world.

- Newsflash! NEW!
  - **August 4, 2003:** A SNAP kit was generated. For the complete list of changes, check here.
  - Past newsflash
- Latest releases
  - **KAME SNAP** (weekly): August 4, 2003 NEW!
    - List of mirror servers
    - Supported platforms: BSD/OS 3.1 (*), BSD/OS 4.3, FreeBSD 4.8-RELEASE , NetBSD 1.6.1, and OpenBSD 3.3.
      We do not generate targz file for the platforms marked with (*). Use anoncvs/cvsup for those platforms. Note that they may even not compile.
  - Platforms with KAME code merged in:
    - FreeBSD 4.0 and beyond
    - OpenBSD 2.7 and beyond
    - NetBSD 1.5 and beyond
    - BSD/OS 4.2 and beyond
  - What's the difference between KAME kit, and KAME-integrated *BSD releases?
- Related software
  - **IPv6-enabled software and patches**
  - **more about our IPv6-enabled software and patch collections**

(剩餘 3 個項目 s) 正在下載圖片 http://10.128.128.127/img/vpnc-test-partner.gif...

Using IPv4 to Browse Without Socket-layer Translator


Webpage of Kame Project - Microsoft Internet Explorer
檔案(F) 編輯(E) 檢視(V) 我的最愛(A) 工具(T) 說明(H)
上一頁
搜尋
我的最愛
媒體
網址(D)
http://www.kame.net
移至
連結
完成
網際網路

KAME Project

If you migrate to IPv6 HTTP, you'll be able to view the dancing kame

KAME Project is a joint effort of six companies in Japan
to provide a free IPv6 and IPsec (for both IPv4 and IPv6) stack for BSD variants to the world.

- Newsflash! NEW!
  - **August 4, 2003:** A SNAP kit was generated. For the complete list of changes, check here.
  - Past newsflash
- Latest releases
  - **KAME SNAP** (weekly): August 4, 2003 NEW!
    - List of mirror servers
    - Supported platforms: BSD/OS 3.1 (*), BSD/OS 4.3, FreeBSD 4.8-RELEASE , NetBSD 1.6.1, and OpenBSD 3.3.
      We do not generate tar.gz file for the platforms marked with (*). Use anoncvs/cvsup for those platforms. Note that they may even not compile.
  - Platforms with KAME code merged in:
    - FreeBSD 4.0 and beyond
    - OpenBSD 2.7 and beyond
    - NetBSD 1.5 and beyond
    - BSD/OS 4.2 and beyond
  - What's the difference between KAME kit, and KAME-integrated *BSD releases?
- Related software
  - **IPv6-enabled software and patches**
  - **more about our IPv6-enabled software and patch collections**

Conclusions

- In this course, you can learn the following techniques
  - IPv4 Windows Socket Programming
  - IPv6 Windows Socket Programming
  - IPv4/IPv6 Domain Name Resolution
- You can try to do following advanced topics.
  - Writing IPv4/IPv6 compatible programs
  - Porting IPv4 applications to IPv6 version
  - Writing ALG on Socket-layer Translator
  - Writing IPv6 Test tools on SIPv6 Analyzer

References

- Microsoft Platform SDK
- MSDN Library
- VC++ 6.0

References

[1] RFC- 2766 Network Address Translation - Protocol Translation，*G. Tsirtsis*、*P. Srisuresh*，2000/2

[2] RFC-2765 Stateless IP/ICMP Translator (SIIT)，Nordmark, E.，2000/2

[3] RFC-2767 Bump-In-the-Stack，*K. Tsuchiya*、*H. Higuchi*、*Y. Atarashi*，2000/2， [4] RFC-3338 Bump-In-the-API，*S. Lee*、*M-K. Shin*、*Y-J. Kim*、*E. Nordmark*、*A. Durand*，2002/10

[5] IPv6 Guide for Windows Sockets Applications，*MSDN Library*，2003/2， http://msdn.microsoft.com/library/en-us/winsock/winsock/ipv6_guide_for_windows_sockets_applications_2.asp?frame=true

[6] How to upgrade WinSock application to support IPv6，*Makoto Ookawa*，2003/7，http://www.ipv6style.jp/en/apps/20030711/20030711_p.shtml

[7] RFC-2893 Transition Mechanisms for IPv6 Hosts and Routers，*R. Gilligan*、 *E. Nordmark*，2000/8

[8] Hitachi Toolnet6, http://www.hitachi.co.jp/Prod/comp/network/pexv6-e.htm

[9] IPv6解析，謝佳男、朱永正、陳懷恩譯，台灣歐萊禮，ISBN：986-7794-11-7

Appendix

IPv4 Header

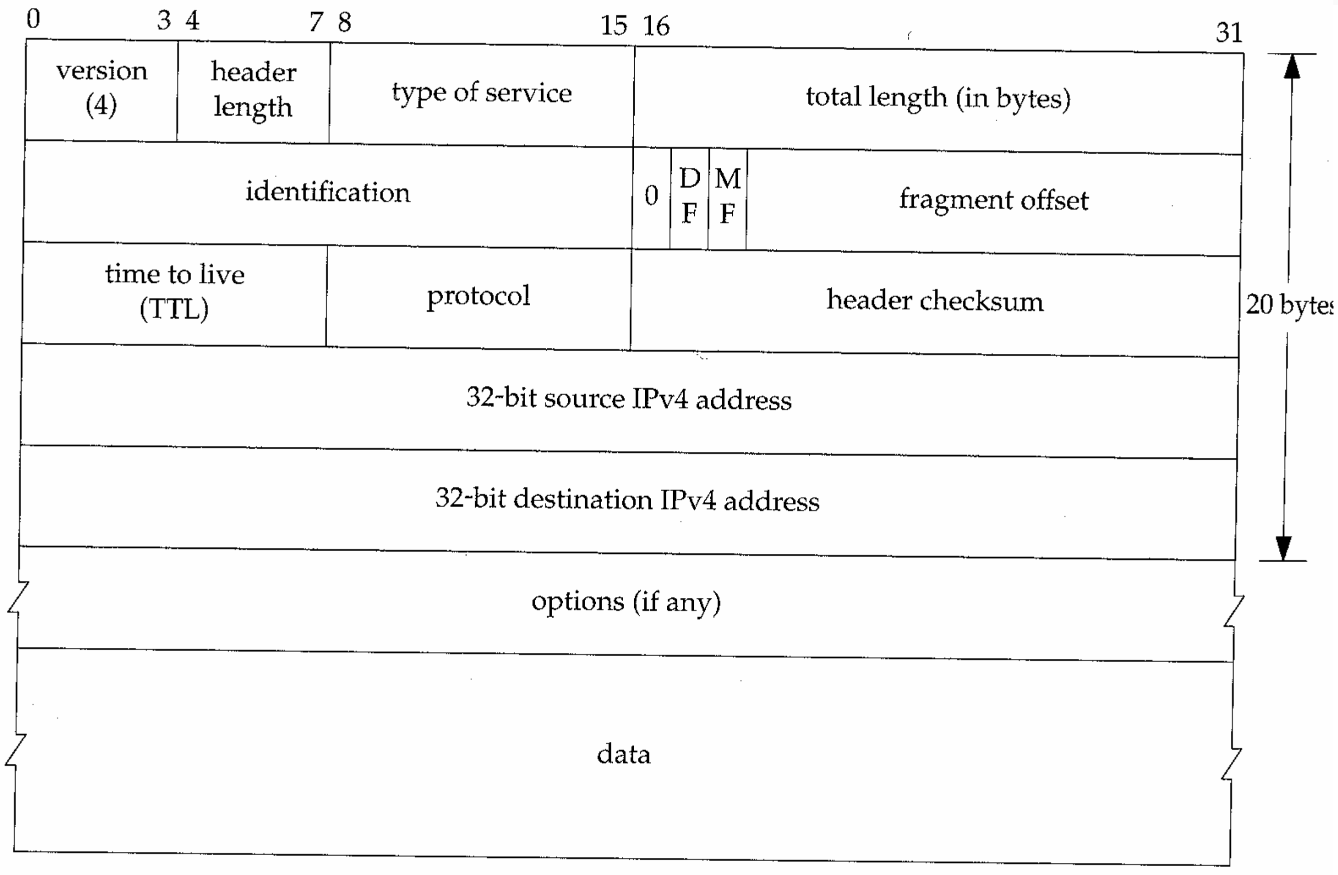


Figure A.1 Format of the IPv4 header.

IPv6 Header

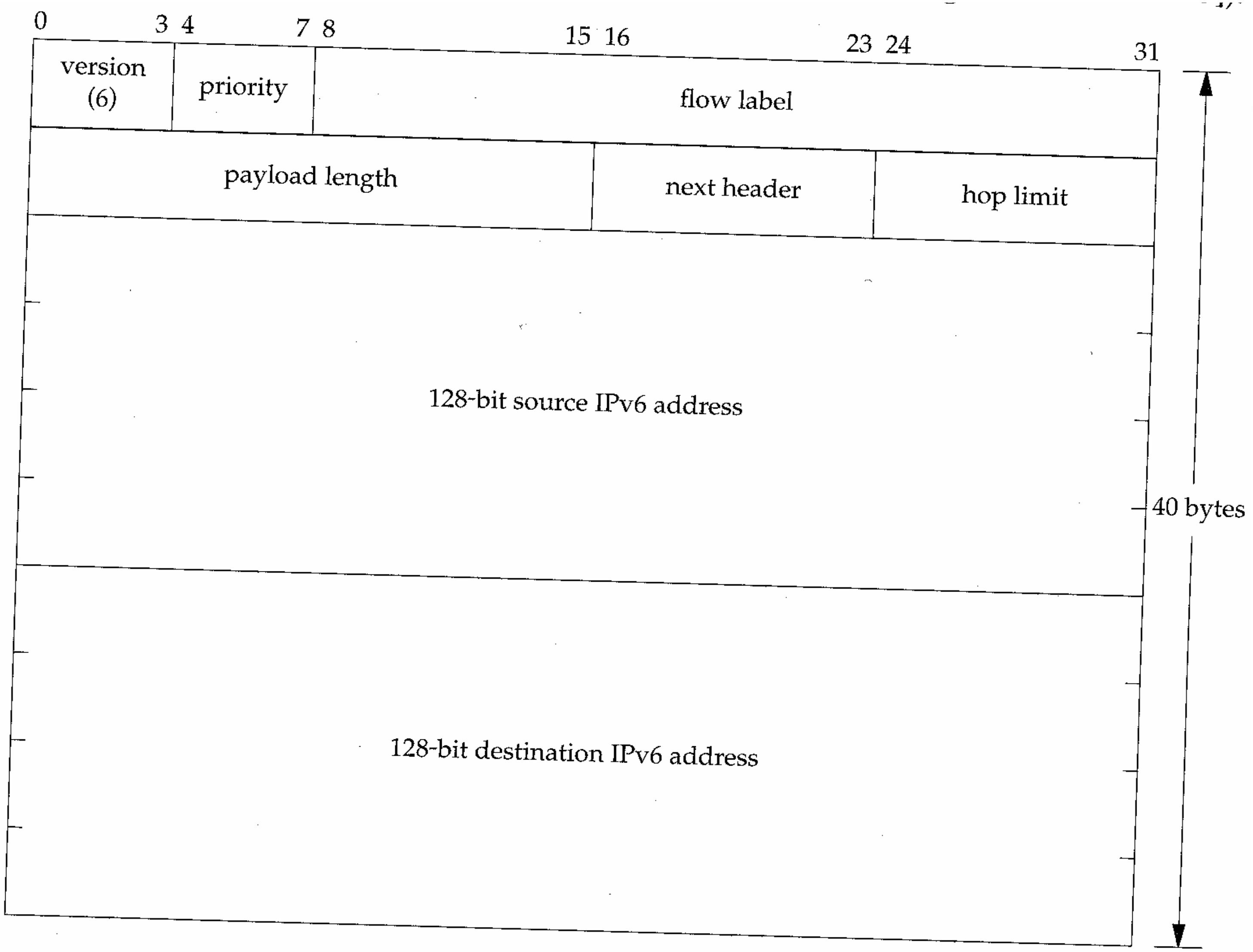


Figure A.2 Format of the IPv6 header.

IPv4 Address

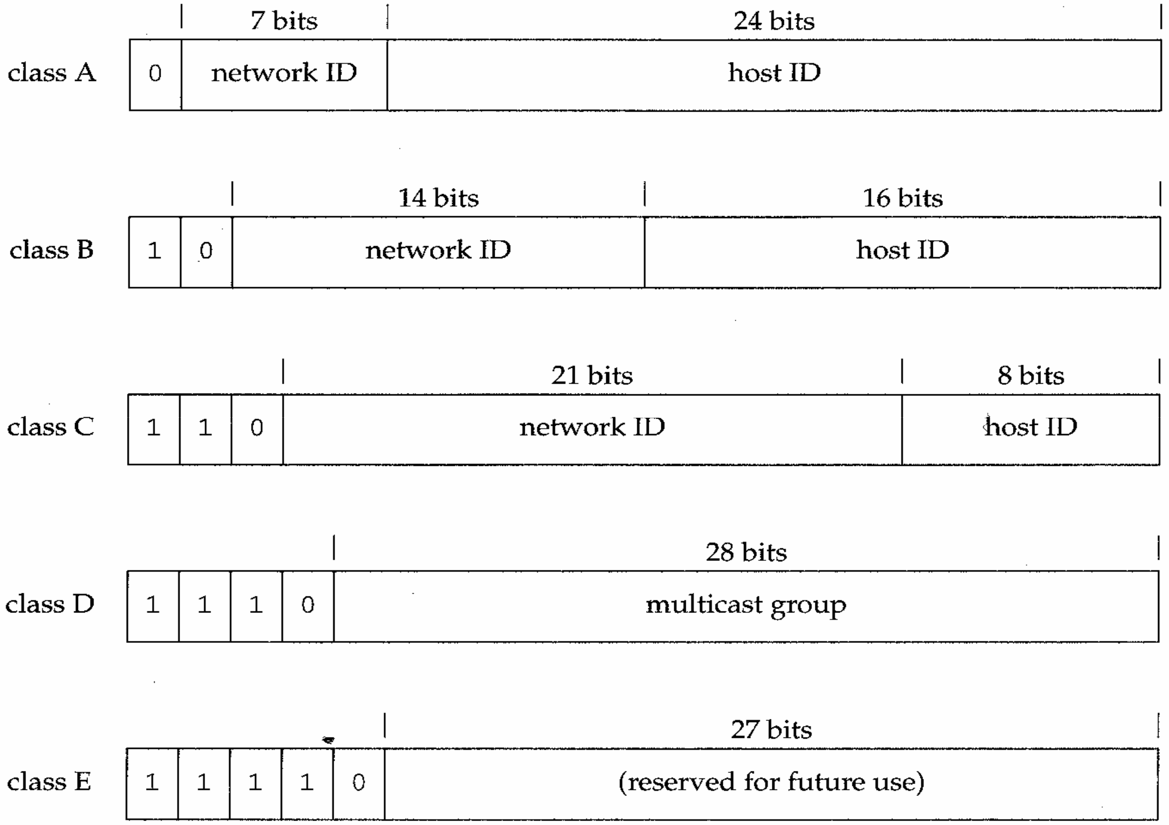
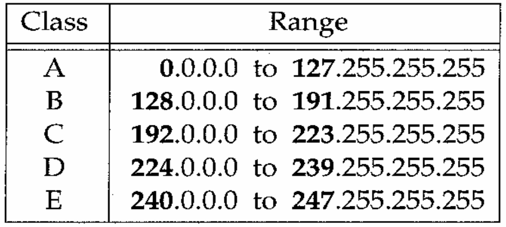


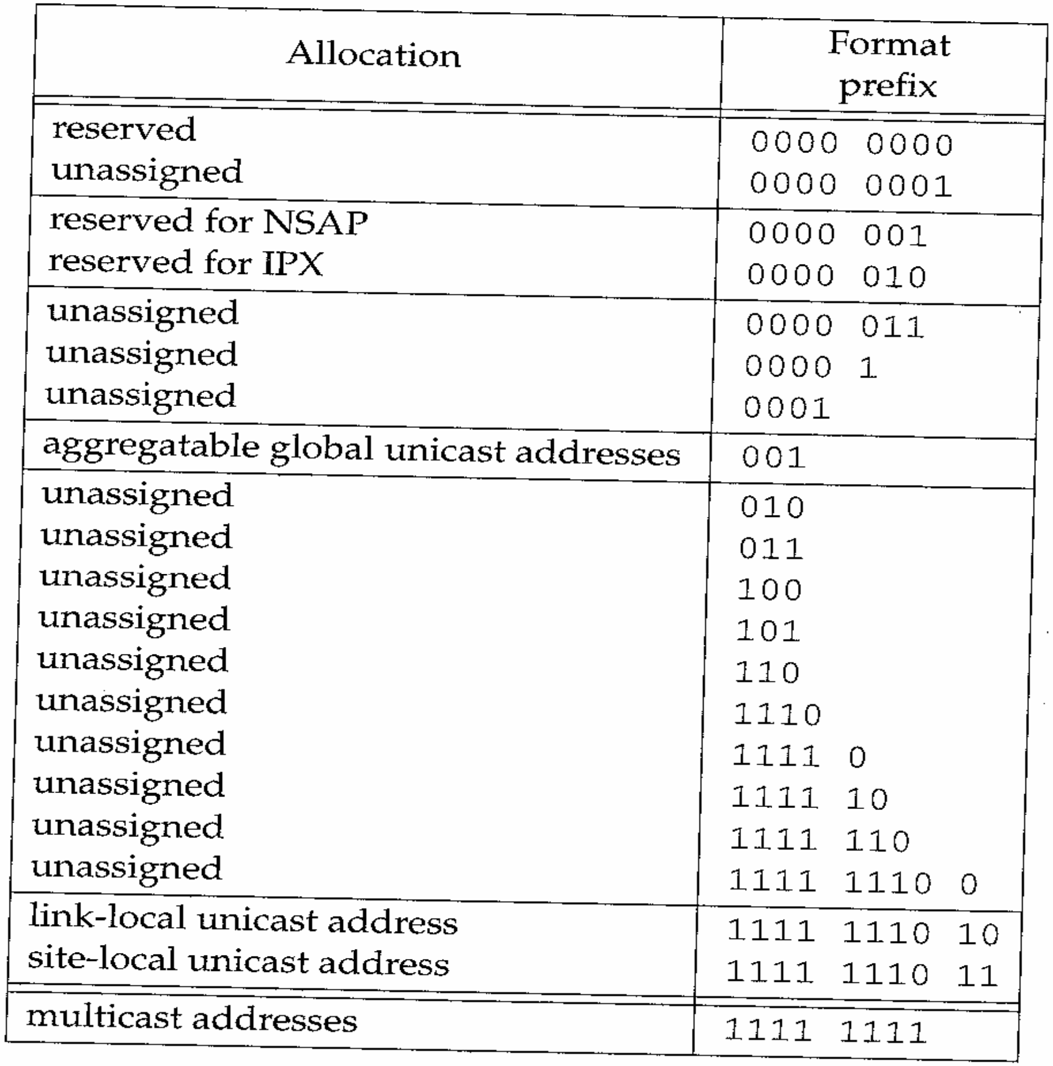
Figure A.3 IPv4 address formats.



Class	Range
A	**0**.0.0.0 to **127**.255.255.255
B	**128**.0.0.0 to **191**.255.255.255
C	**192**.0.0.0 to **223**.255.255.255
D	**224**.0.0.0 to **239**.255.255.255
E	**240**.0.0.0 to **247**.255.255.255

Figure A.4 Ranges for the five different classes of IPv4 addresses.

IPv6 Address



Allocation	Format prefix
reserved	0000 0000
unassigned	0000 0001
reserved for NSAP	0000 001
reserved for IPX	0000 010
unassigned	0000 011
unassigned	0000 1
unassigned	0001
aggregatable global unicast addresses	001
unassigned	010
unassigned	011
unassigned	100
unassigned	101
unassigned	110
unassigned	1110
unassigned	1111 0
unassigned	1111 10
unassigned	1111 110
unassigned	1111 1110 0
link-local unicast address	1111 1110 10
site-local unicast address	1111 1110 11
multicast addresses	1111 1111

Figure A.7 Meaning of high-order bits of IPv6 addresses.