

欧
中
网
格

EUChinaGRID IPv6 Tutorial

Valentino R. Carcione - GARR
Roma, 26.10.2006

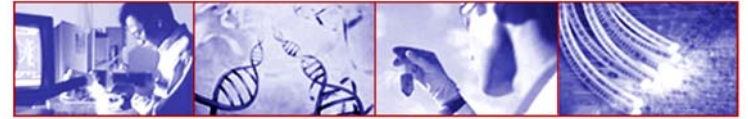




ICMPv6

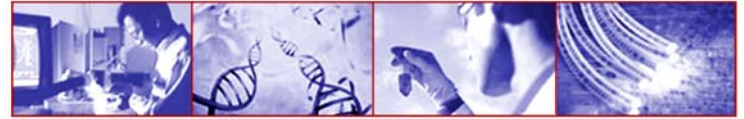
Neighbor Discovery

Addresses configuration



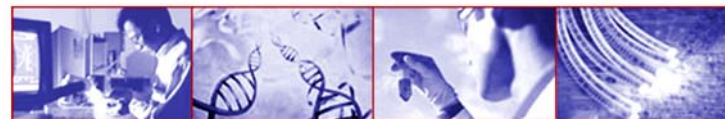
欧
中
网
格

ICMPv6



ICMPv6 protocol

- ▶ It is the IPv6 version of ICMP with the same basic features
 - Error discovery, controll, debugging
- ▶ Add new funtionalities
 - Neighbor discovery
 - Neighbor Solicitation, Unreachability, Autoconfiguration
 - Multicast group management
- ▶ has the same functionalities of ICMP, ARP, e IGMP protocols for IPv4.



ICMPv6: Header

▶ IPv6 Next Header = 58

▶ The field of the ICMPv6 header are:

- ICMPv6 Type
- ICMPv6 Code
- Header Checksum
- ICMPv6 Data

Ver	Class	Flow Label	
Length		Next Hdr	Hop Limit
Source Address			
Destination Address			
Type	Code	Checksum	
Data			

Type	Code	Checksum
ICMPv6 Data		



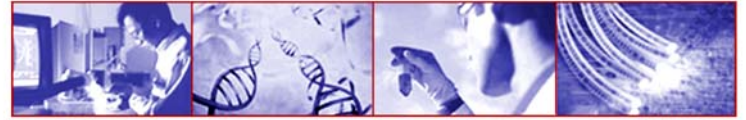
ICMPv6: Message type

▶ Two class of messaged:

- From type 0 to 127 Error Messages
- From type 128 to 255 Informational Messages

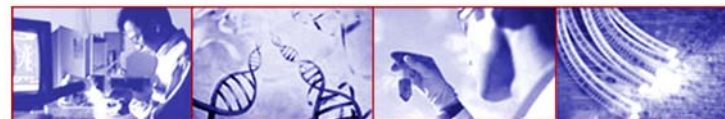
▶ The most common error messages are:

- Destination Unreachable (1)
- Packet Too Big (2)
- Time Exceeded (3)
- Parameter Problem (4)



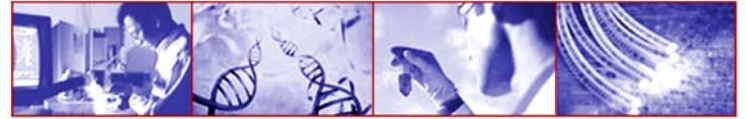
ICMPv6: Informational message

- ▶ Diagnostic and troubleshooting
 - Echo request/Echo reply (128/129)
- ▶ Control messages
 - Multicast group management
 - Multicast Listener Query/Report/Done (130/131/132)
 - Neighbor discovery
 - Router Solicitation/Advertisement (133/134)
 - Neighbor Solicitation/Advertisement (135/136)
 - Redirect (137)
 - Inverse Neighbor Discovery (141/142)
- ▶ Information request
 - Node Information Query/Response (139/140)



ICMPv6 Message type

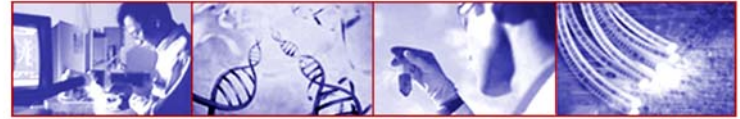
1	Destination Unreachable	133	Router Solicitation
2	Packet Too Big	134	Router Advertisement
3	Time Exceeded	135	Neighbor Solicitation
4	Parameter Problem	136	Neighbor Advertisement
		137	Redirect Message
128	Echo Request	138	Router Renumbering
129	Echo Reply	139	ICMP Node Information Query
130	Multicast Listener Query	140	ICMP Node Information Response
131	Multicast Listener Report	141	Inverse Neighbor Disc. Solicitation
132	Multicast Listener Done	142	Inverse Neighbor Disc. Advertisement



欧
中
网
格

ICMPv6

Path MTU Discovery



Path MTU Discovery

(1)

- ▶ IPv6 fragmentation management is end-to-end
 - Routers don't fragment packets
 - The fragmentation process is managed by host
- ▶ The host uses Path MTU Discovery to know the maximum MTU available on the link.
 - Based on ICMPv6 "Packet too big" messages
 - A router creates a packet too big message when the MTU used is too large for the path
 - Specifies the new MTU in the data field.



Path MTU Discovery

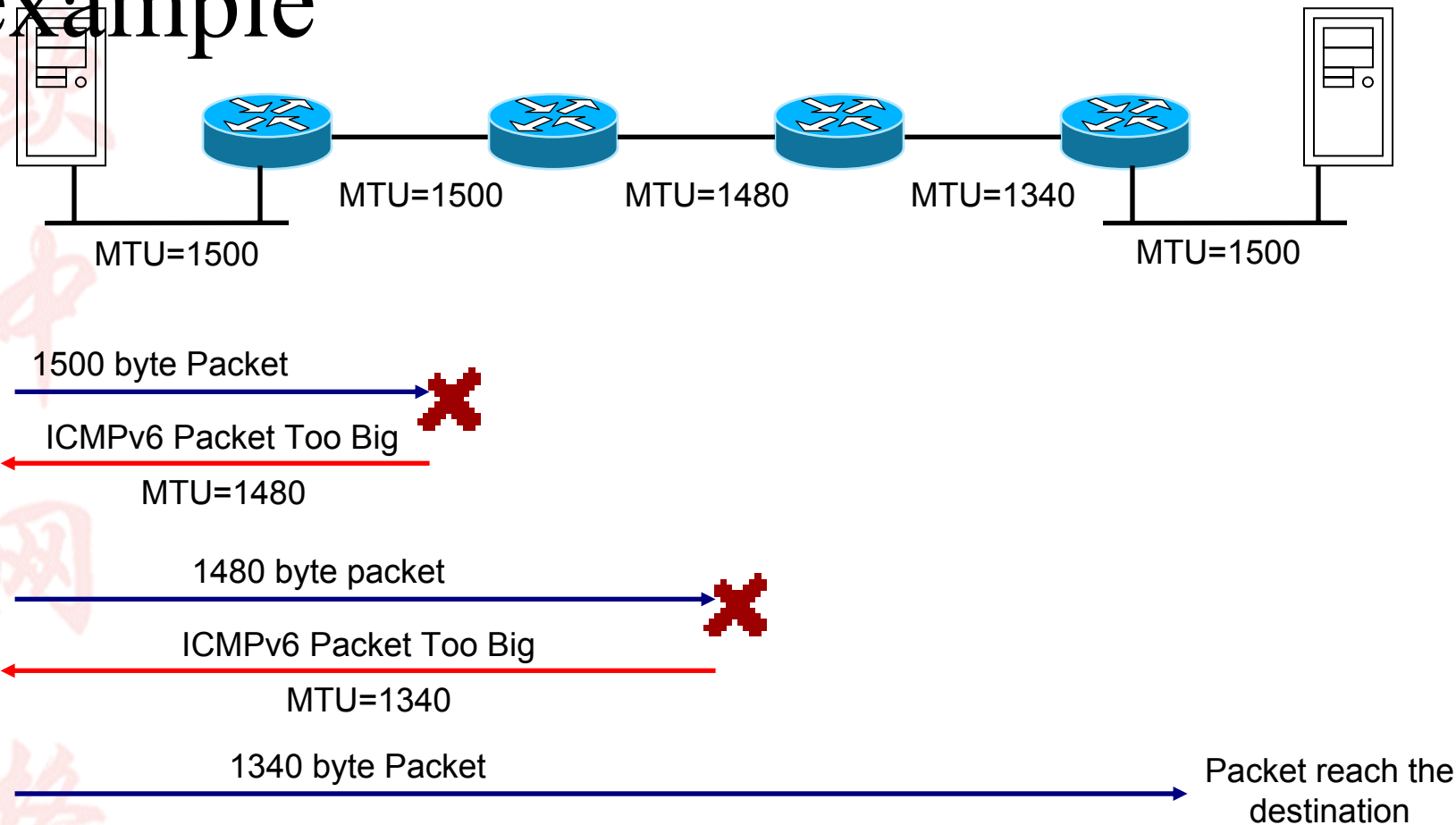
(2)

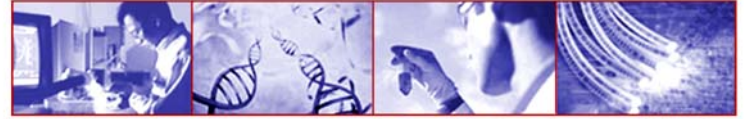
▶ How MTU path discovery works:

- The host sends the first packet with the same dimension as MTU of its link
- If a “Packet Too Big” is reached the host sends another message with the new MTU
- The host repeats the process until no error is found
- ▶ The host sends packets periodically, to check if the path has changed
- ▶ Minimum MTU for IPv6 is 1280 byte



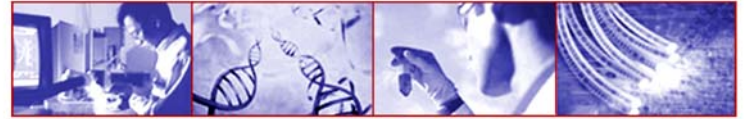
Path MTU Discovery: example





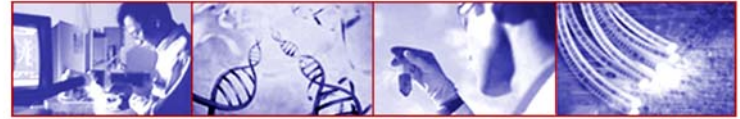
欧
中
网
格

Neighbor Discovery



Neighbor Discovery

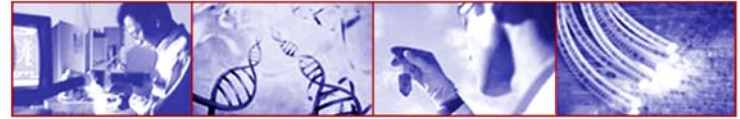
- ▶ Uses ICMPv6
- ▶ Manages the control information within a link
 - Address resolution
 - Neighbor Solicitation and Neighbor Advertisement
 - Neighbor Unreachability Detection
 - Autoconfiguration
 - Router Solicitation e Router Advertisement
 - Redirect
- ▶ Messages cannot be sent outside the link
 - Valid messages have Hop Limit = 255



Neighbor Solicitation

(1)

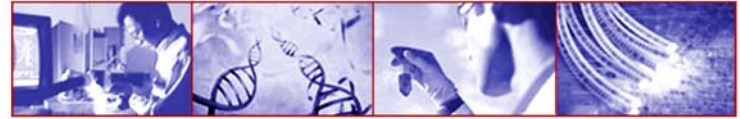
- ▶ Like IPv4 ARP
- ▶ Uses ICMPv6 packets
 - media-independent
 - Can use IPSEC authentication and encryption
 - Uses multicast addresses, unsolicited nodes can discard the solicitation at layer 3



Neighbor Solicitation

(2)

- ▶ To obtain a physical address from another host :
 - The host deduces the (multicast) Solicited-Node address that corresponds to receiver's IPv6 address
 - The host sends a Neighbor Solicitation to this address and specify the receiver's IPv6 address in the data field
- ▶ The receiver, if present, sends back a Neighbor Advertisement
 - Its physical address is specified in the data field
 - Adds to the Neighbor Cache the address of the sender



Solicited-Node

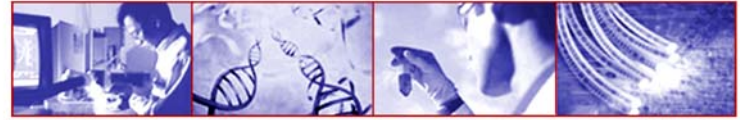
Multicast

- ▶ For each IPv6 unicast there is a multicast Solicited-Node

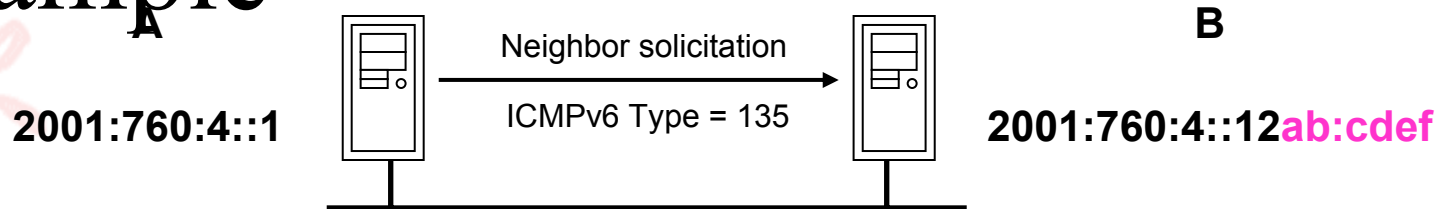


- ▶ Is composed by adding the last 24 bit of the unicast address to the prefix **ff02::1:ff00:0/104**
 - Avoids collisions in case of duplicated Interface ID

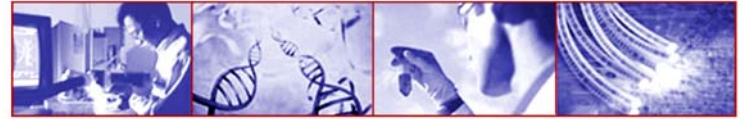




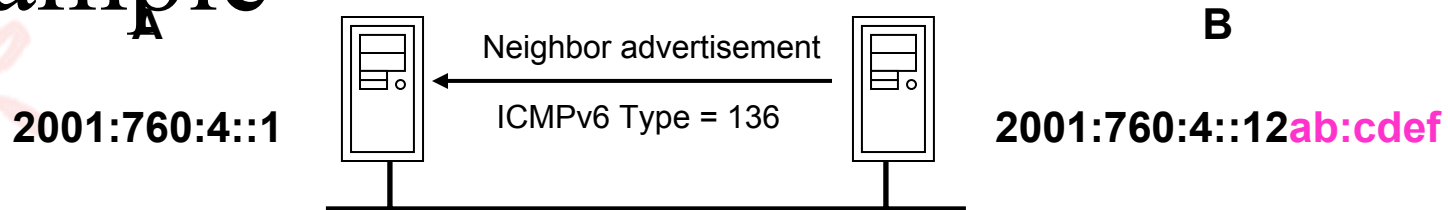
Neighbor Solicitation: example



- ▶ A wants to get the physical address of B
- ▶ A deduces the multicast Solicited-Node address of B:
ff02::1:ffab:cdef
- ▶ A sends a Neighbor Solicitation:
 - Source: A's IPv6 address
 - Destination: the deduced solicited-node
 - Data ICMPv6:
 - B's IPv6 address
 - A's physical address (shows B the address to answer)



Neighbor Solicitation: example



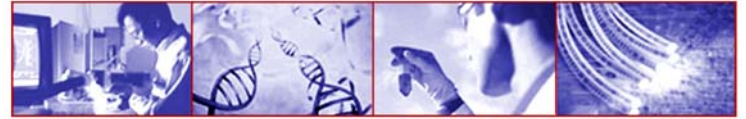
▶ B sends back a Neighbor Advertisement:

- Source: B's IPv6 address
- Destination: A's IPv6 address
- Data ICMP:
 - B's IPv6 address
 - B's physical address



Neighbor Unreachability Detection

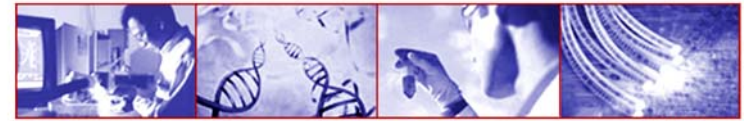
- ▶ Algorithm that allows to find quickly faults and addresses changes
 - More efficient than a simple timeout
 - Useful for mobile IPv6
- ▶ Each host traces the reachability of neighbour:
 - Using information from upper layers (eg. TCP ACK)
 - Closer hosts: tracing the running hosts
 - Remote hosts: tracing the running next-hop router
 - Sending Neighbor Solicitation to the remote host



Stateless

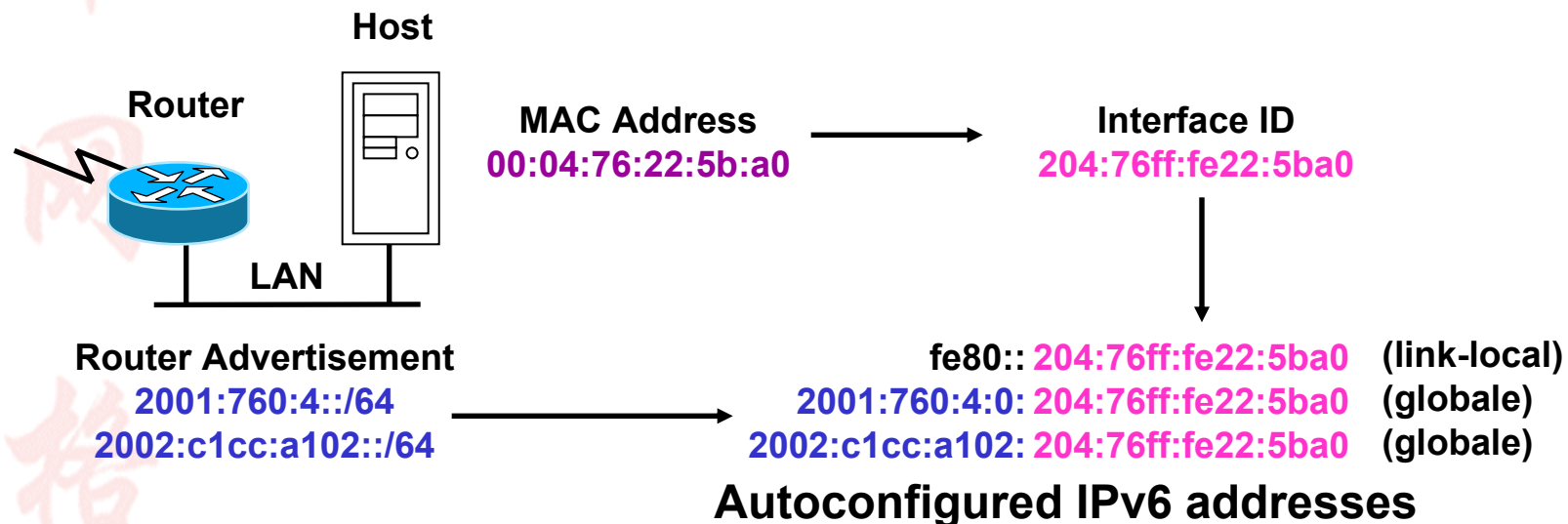
Autoconfiguration

- ▶ Allows the IPv6 hosts to connect to the network without manual configuration
 - ▶ No need to use DHCP
 - Uses specific multicast group
 - ▶ Addresses are based on Interface ID
 - ▶ On the link, hosts can communicate among them using link-local addresses
- ▶ Unlike DHCP, the DNS must be configured manually



Stateless Autoconfiguration

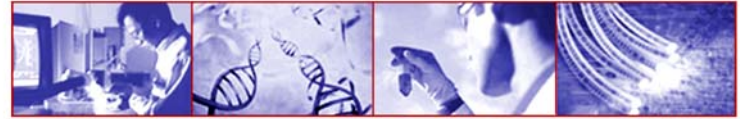
- ▶ Routers send the prefix of the LAN using a Router Advertisement (ICMPv6 Type 135)
- ▶ Hosts use the MAC address of the NIC to configure the IPv6 addresses, it is composed by adding the Interface ID (64 bit) to the prefix (or prefixes) advertised by the router





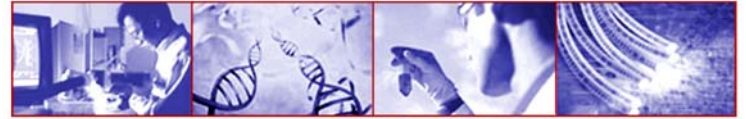
Stateful Configuration

- ▶ Addresses and other network parameters (eg. DNS) can be configured manually, too:
 - Entirely manual configuration
 - DHCPv6
 - Stateless autoconfiguration
 - Router Advertisement include two flags, which specify configuration modes:
 - > “Managed Address Configuration”: indicates if host can obtain addresses from DHCPv6
 - > “Other Stateful Configuration”: indicates if host can use DHCPv6 to obtain other configuration information (eg. server DNS, server NTP, ...). Mandatory if Managed Address Configuration is true.



DHCPv6

- ▶ Is used like DHCP in IPv4
- ▶ The addresses are added to the ones obtained by autoconfiguration
- ▶ Enables a better host configuration control
- ▶ How DHCP works:
 - Like IPv4 DHCP
 - Server maintains the information about the client's state
 - Enables IPv6 address configuration and/or provides other configuration information (DNS, NTP...)
 - Uses UDP datagram
 - Uses multicast groups ff02::1:2 (all DHCP agents, link-local scope) and ff05::1:3 (all DHCP servers, site-local scope)



欧
中
网
格

Configuration Basics

Linux



Configuration Basics - Linux

- ▶ IPv6 support is available since Linux kernel release 2.4.
- ▶ The current support does not implement all RFC features
- ▶ A patch (USAGI patch) is available to provide all extensions to the kernel.
- ▶ Further information:
- ▶ USAGI Project <http://www.linux-ipv6.org/>



Configuration Basics - Linux

- ▶ If the IPv6 support is available on our kernel, the file

`/proc/net/if_inet6`

- ▶ must be present. If not, we can try to load the IPv6 kernel module,

```
# modprobe ipv6
```

- ▶ and then test it again.
- ▶ If the module is not available we must rebuild our kernel.

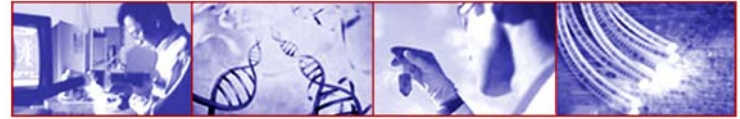


Configuration Basics - Linux

▶ Resources:

▶ Kernel documentation

▶ Linux Kernel HOWTO
(<http://www.linuxdoc.org/HOWTO/Kernel-HOWTO.html>).



Interface Configuration

- Linux

▶ The configuration syntax for IPv6 is similar to IPv4. In the following examples, we will use ifconfig.

▶ Add an IPv6 address to an interface

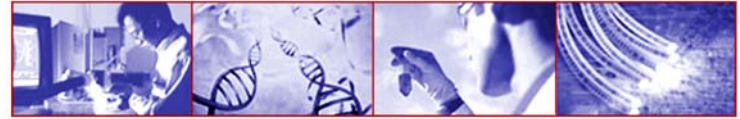
```
# /sbin/ifconfig <interface> inet6 add <ipv6address>/<prefixlength>
```

```
# /sbin/ifconfig eth0 inet6 add 2001:760:ffff::126/64
```

▶ Delete an IPv6 address from an interface

```
# /sbin/ifconfig <interface> inet6 del <ipv6address>/<prefixlength>
```

```
# /sbin/ifconfig eth0 inet6 del 2001:760:ffff::126/64
```



Interface Configuration

- Linux

Show the interface configuration

```
#ifconfig eth0
```

```
eth0  Link encap:Ethernet  HWaddr 00:10:B5:DA:59:B8
```

```
inet addr:193.206.158.126  Bcast:193.206.158.255
```

```
Mask:255.255.255.0
```

```
inet6 addr: 2001:760:ffff::126/64 Scope:Global
```

```
inet6 addr: fe80::210:b5ff:feda:59b8/10 Scope:Link
```

```
UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
```

```
RX packets:6262494 errors:0 dropped:0 overruns:0 frame:0
```

```
TX packets:7971062 errors:0 dropped:0 overruns:0 carrier:0
```

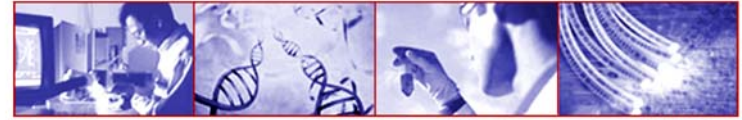
```
collisions:0 txqueuelen:100
```

```
Interrupt:5 Base address:0xc000
```

Global unicast address

Link local address

MAC ADDRESS EUI-64 format



Routing table - Linux

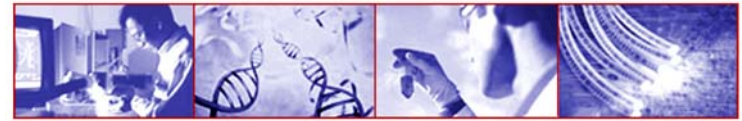
- ▶ As for IPv4 we can operate on the routing table.
- ▶ We will use for these functions the route command.

• Show the routing table

```
#route --inet6
```

Kernel IPv6 routing table

Destination	Next Hop	Flags	Metric	Ref	Use	Iface
::1/128	::	U	0	0	0	lo
2001:760:ffff::126/128	::	U	0	0	0	lo
2001:760:ffff::/64	::	UA	256	0	0	eth0 ;route for the global address
fe80::210:b5ff:feda:59b8/128	::	U	0	0	0	lo
fe80::250:56ff:fec0:1/128	::	U	0	0	0	lo
fe80::250:56ff:fec0:8/128	::	U	0	0	0	lo
fe80::/10	::	UA	256	0	0	eth0 ;route for the link-local
ff00::/8	::	UA	256	0	0	eth0 ;generic route for multicast
::/0	::	UDA	256	0	0	eth0 ; automatic default route



Routing Table - Linux

- ▶ Add or delete an entry on the routing table

```
#route --inet6 add|del <ipv6network>/<prefix> gw <ipv6addr> [dev <device>]
#route --inet6 add|del <ipv6network>/<prefix> [dev <device>]
```

```
#route --inet6 add default gw 2001:760:ffff::11
```

```
#route --inet6
```

Kernel IPv6 routing table

Destination	Next Hop	Flags	Metric	Ref	Use	Iface
.....						
::/0	2001:760:ffff::11	UG	1	0	0	eth0 ;default route
::/0	::	UDA	256	0	0	eth0 ; automatic default route



Host Configuration – Fedora Core

Linux

▶ Edit the file `/etc/sysconfig/network-scripts/ifcfg-<interfacename>`
(`ifcfg-eth0` for the first ethernet interface)

▶ Add the following lines

IPV6INIT=yes

IPV6ADDR=<ipv6address/prefixlen>

▶ Adding the following configuration to `/etc/sysconfig/network`

NETWORKING_IPV6=yes

IPV6_DEFAULTGW=<ipv6gatewayaddress>

▶ Restart the network

#service network restart



Host Configuration – Generic Linux

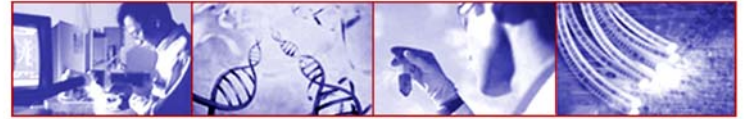
- ▶ The following configuration should be used if the IPv6 scripts are not available on the operating system
 - ▶ Add the following lines to */etc/rc.local* (it could be */etc/rc.d/rc.local* on many distributions)

```
IPV6_ADDRESS=<ipv6address/prefixlen>
```

```
IPV6_GW=<ipv6gatewayaddress>
```

```
/sbin/ifconfig eth0 inet6 add $IPV6_ADDRESS
```

```
/sbin/route --inet6 add default gw $IPV6_GW
```



欧
中
网
格

Configuration Basics

Windows



Configuration Basics – Microsoft

Windows

▶ IPv6 for Windows is currently available for:

- Windows 2003
- Windows XP
- Windows NT4 e Windows 2000 with the add-on provided by Microsoft
(<http://msdn.microsoft.com/downloads/sdks/platform/tcpipv6/download.asp>)
- The IPv6 support is available for Windows 95/98/NT by means of Trumpet Winsock 5.0 (payware)



Configuration Basics – Microsoft

Windows

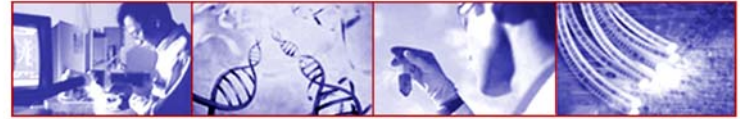
▶ Windows IPv6 implementation supports:

- Autoconfiguration
- Tunnel
- 6to4 e 6over4

▶ Some Windows software like

- Internet Explorer.
- Ping, traceroute e telnet.

are IPv6 ready:



Configuration Basics – Microsoft

Windows

- ▶ On Windows 2000 you need to install Service Pack-1, 2 or 3. The installation kit must be modified as follow:

Download the IPv6 kit from URL

<http://msdn.microsoft.com/downloads/sdks/platform/tpipv6/download.asp>

Extract the archive content to a temporary folder (e.g. C:\>ipv6kit);

From this folder, execute `setup.exe -x`,

A folder called files will be created;

Edit the file `Hotfix.inf` and modify the key `NTServicePackVersion`:

- For SP2 “NTServicePackVersion=512”
- For SP3 “NTServicePackVersion=768”

Run `Hotfix.exe` and restart the computer.



Configuration Basics – Microsoft

Windows

▶ References:

▶ <http://www.microsoft.com/ipv6>



Running IPv6 – Microsoft

Windows

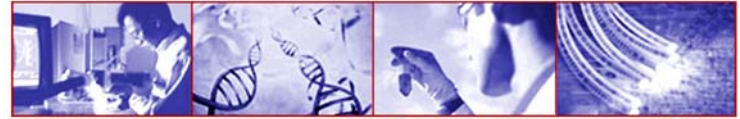
- ▶ We can activate or deactivate the IPv6 stack using the command net

```
net stop tcpip6
```

- ▶ Disable the IPv6 support and remove the related kernel module
- ▶ The net command cannot deactivate IPv6 if an IPv6 socket is in use.

```
net start tcpip6
```

- ▶ Load the IPv6 kernel module (tcpip6.sys) and activate the IPv6 support.



Running IPv6 – Microsoft Windows

- ▶ The `ipv6` command manage the windows IPv6 stack.
The following command show the interfaces configuration:

```
C:>ipv6 if 4
```

```
Interface 4 (site 1):
```

```
uses Neighbor Discovery
```

```
link-level address: 00-50-56-a3-00-01
```

```
preferred address 2001:760::196, infinite/infinite
```

```
preferred address fe80::250:56ff:fea3:1, infinite/infinite
```

```
multicast address ff02::1, 1 refs, not reportable
```

```
multicast address ff02::1:ffa3:1, 1 refs, last reporter
```

```
multicast address ff02::1:ff00:0, 1 refs, last reporter
```

```
link MTU 1500 (true link MTU 1500)
```

```
current hop limit 128
```

```
reachable time 36000ms (base 30000ms)
```

```
retransmission interval 1000ms
```

```
DAD transmits 1
```

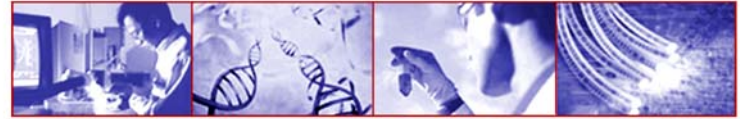


Running IPv6 – Microsoft

Windows

▶ The `ipv6.exe` command is also used to:

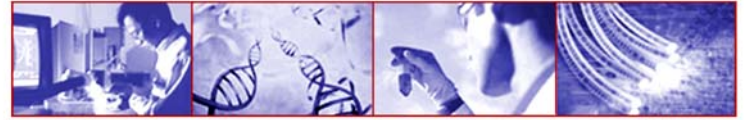
- Add and delete IPv6 addresses on the network interfaces.
 - View and modify the some protocol attributes (router advertisement, forward options etc.)
 - Add or delete an interface
 - Show and manage the routing table



Running IPv6 – Microsoft

Windows

C:\>ipv6
usage: ipv6 if [ifindex]
 ipv6 ifc ifindex [forwards] [-forwards] [advertises] [-advertises] [mtu #
bytes] [site site-identifier]
 ipv6 ifd ifindex
 ipv6 adu ifindex/address [lifetime validlifetime[/preflifetime]] [anycast
] [unicast]
 ipv6 nc [ifindex [address]]
 ipv6 ncf [ifindex [address]]
 ipv6 rc [ifindex address]
 ipv6 rcf [ifindex [address]]
 ipv6 bc
 ipv6 rt
 ipv6 rtu prefix ifindex[/address] [lifetime L] [preference P] [publish] [
age] [spl SitePrefixLength]
 ipv6 spt
 ipv6 spu prefix ifindex [lifetime L]



Running IPv6 – Microsoft

Windows

▶ In Windows XP the use of the *netsh* utility instead of *ipv6.exe* is suggested

▶ A complete reference to migrate *ipv6.exe* command to *netsh* is available at the URL

<http://www.microsoft.com/technet/itsolutions/network/ipv6/ipv62netshtable.msp>

▶ To install the IPv6 support type the following command in a command window

```
netsh interface ipv6 install
```



Running IPv6 – Microsoft Windows

▶ Show the interface table

C:>netsh interface ipv6 show interface

Idx	Met	MTU	State	Name
6	0	1500	Disconnected	Wireless Network Connection
5	0	1500	Connected	Local Area Connection
4	2	1280	Disconnected	Teredo Tunneling Pseudo-Interface
3	1	1280	Connected	6to4 Pseudo-Interface
2	1	1280	Connected	Automatic Tunneling Pseudo-Interface
1	0	1500	Connected	Loopback Pseudo-Interface

▶ Add the IPv6 address to the Local Area Connection Interface

C:>netsh interface ipv6 add address interface=5 address=<ipv6address/prefixlen>

▶ Add the default gateway route through the same interface

C:>netsh interface ipv6 add route ::/0 5 <ipv6gatewayaddress>



Running IPv6 – Microsoft

Windows

▶ Check the routing table

C:>netsh interface ipv6 show route

Querying active state...

Publish	Type	Met	Prefix	Idx	Gateway/Interface Name
no	Manual	0	::/0	5	2001:760::11



Running IPv6 – Microsoft Windows

▶ To check the IPv6 connectivity the *ping6* command can be used

```
C:>ping6 www.kame.net
```

```
Pinging www.kame.net [2001:200:0:8002:203:47ff:fea5:3085]
```

```
from 2001:760::73 with 32 bytes of data:
```

```
Reply from 2001:200:0:8002:203:47ff:fea5:3085: bytes=32 time=310ms
```

```
Reply from 2001:200:0:8002:203:47ff:fea5:3085: bytes=32 time=310ms
```

```
Reply from 2001:200:0:8002:203:47ff:fea5:3085: bytes=32 time=310ms
```

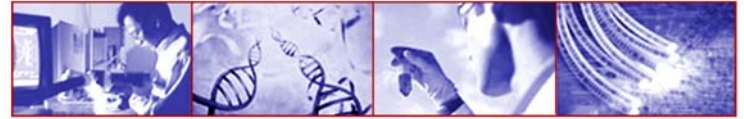
```
Reply from 2001:200:0:8002:203:47ff:fea5:3085: bytes=32 time=310ms
```

```
Ping statistics for 2001:200:0:8002:203:47ff:fea5:3085:
```

```
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
```

```
    Approximate round trip times in milli-seconds:
```

```
        Minimum = 310ms, Maximum = 310ms, Average = 310ms  
netsh interface ipv6 add address  
interface=5 address=<ipv6address/prefixlen>
```



欧
中
网
格

Configuration Basics

Cisco IOS



Configuration Basics – Cisco IOS

▶ Command reference:

- To activate the IPv6 traffic forwarding

```
Router(config)#ipv6 unicast-routing
```

- To activate IPv6 support for each interface:

```
Router(config)#interface <type num/num>
```

```
Router(config-if)# ipv6 address <ipv6addr>
```

```
[/<prefix-length>]
```

```
Router(config-if)#ipv6 enable
```



Configuration Basics – Cisco IOS

- ▶ We can use three different syntax to configure the IPv6 address on an interface:

`ipv6 address <ip6addr>[</prefix-length>]`

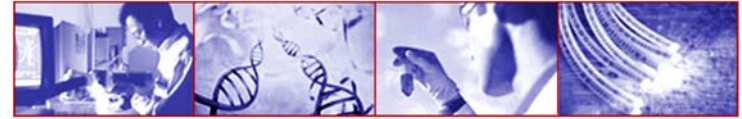
- Is used to configure manually the link-local address, the site-local address, or the global unicast address.

`ipv6 address <ip6addr> </prefix-length> eui-64`

- Is used for site-local, or global unicast addresses in EUI-64 format.

`ipv6 unnumbered <interface>`

- configure the interface as unnumbered like standard IPv4 configurations.



Configuration Basics – Cisco IOS

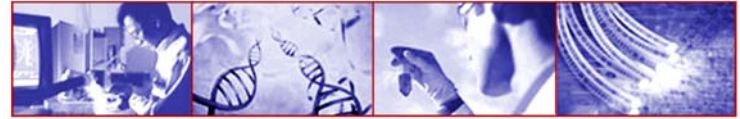
```
milano.6net>sh ipv6 int
ATM3/0.1 is up, line protocol is up
IPv6 is enabled, link-local address is FE80::208:E2FF:FE3B:6200
Description: Connection to 6Net core router it6.it.6net.org
Global unicast address(es):
  2001:798:20:200::2, subnet is 2001:798:20:200::/64
Joined group address(es):
  FF02::1
  FF02::2
  FF02::1:FF3B:6200
  FF02::1:FF00:2
MTU is 4470 bytes
ICMP error messages limited to one every 500 milliseconds
ICMP redirects are enabled
```

MAC ADDRESS (EUI-64 format)

All host on the link multicast address

All router on the link multicast address

Sollicited node multicast address



IPv6 Tools — Cisco IOS

- ▶ The following command are available for debugging operation.
- ▶ The syntax, and the functionalities are similar to the related IPv4 tools.

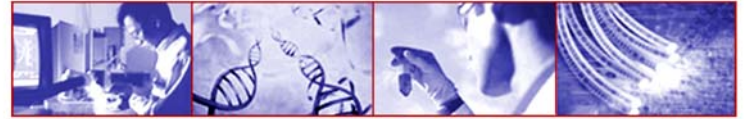
tracert `<ipv6addr>`

ping ipv6 `<ipv6add>`

telnet `<ipv6addr>`

TFTP (e.g. copy running-config tftp://2001:760::30/conf.txt)

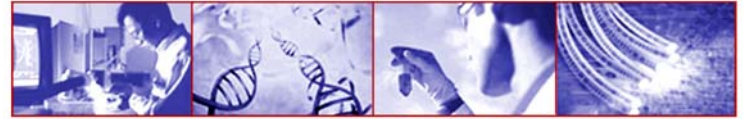
- ▶ (the same tools are available on the other operative system with similar functionalities)



欧
中
网
格

Autoconfiguration and Neighbor Discovery

Linux



Neighbor Discovery - Linux

- ▶ IPv6 Neighbor discovery is similar in functionalities to IPv4 ARP.
- ▶ Like in ARP the kernel tries to discover the neighbours over the network, but the user can view and manage the ND learn table.
- ▶ In the linux enviroment the users can use the ip command to do this.



Show the neighbour information - Linux

▶ Syntax:

- # ip -6 neigh show [dev <device>]
- The follow example show the information written on the learn table
- # ip -6 neigh show
- fe80::202:a5ff:fee3:af53 dev eth0 lladdr 00:02:a5:e3:af:53 nud reachable

Managing neighbour table - Linux

▶ Other feature of IP command:

▶ Add an entry to the neighbour table

- # ip -6 neigh add <ipv6addr> lladdr <macaddr> dev <device>
- # ip -6 neigh add fec0::1 lladdr 02:01:02:aa:ff:ee dev eth0

▶ Delete an entry to the neighbour table

- # ip -6 neigh del <ipv6addr> lladdr <macaddr> dev <device>
- # ip -6 neigh del fec0::1 lladdr 02:01:02:aa:ff:ee dev eth0

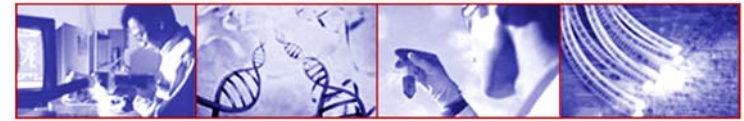
▶ Change and replace option are also available.



Autoconfiguration - Linux

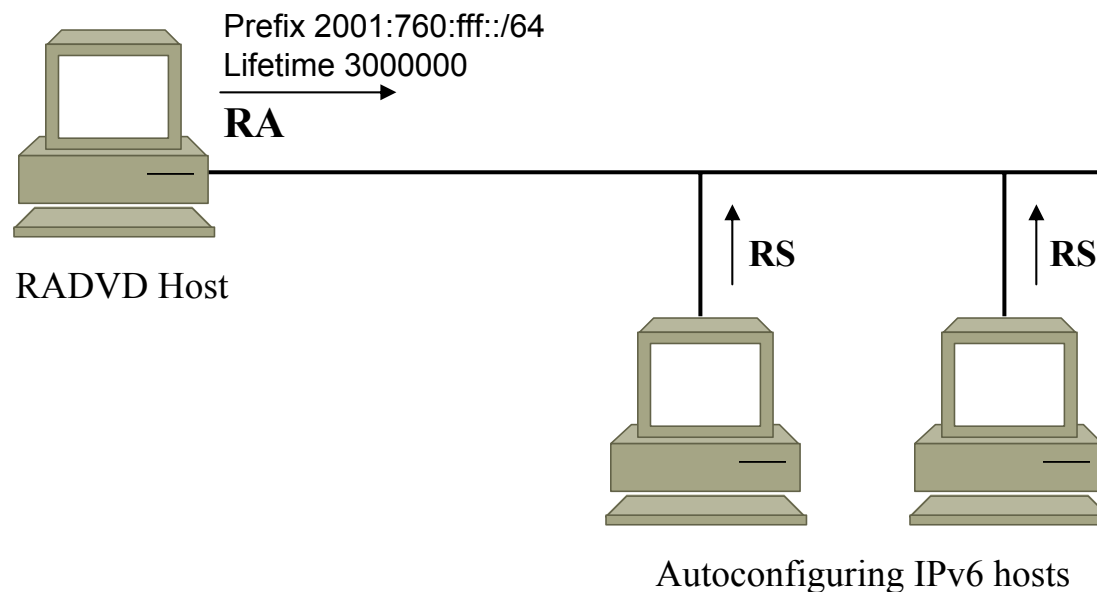
欧
中
网
格

- ▶ Stateless autoconfiguration (for link local and global unicast addresses) is enabled as default when the interface link bring up.
- ▶ If you use a *NIX host to provide the stateless autoconfiguration over the network segment, you can use a freely available daemon software called RADVD (Router Advertisement Daemon)
- ▶ In order to activate radvd you can edit the file /etc/radvd.conf and start the daemon in the bootstrap process.



Autoconfiguration - Linux

- ▶ Radvd answers with a router advertisement when a host sends a router solicitation packet.





Autoconfiguration - Linux

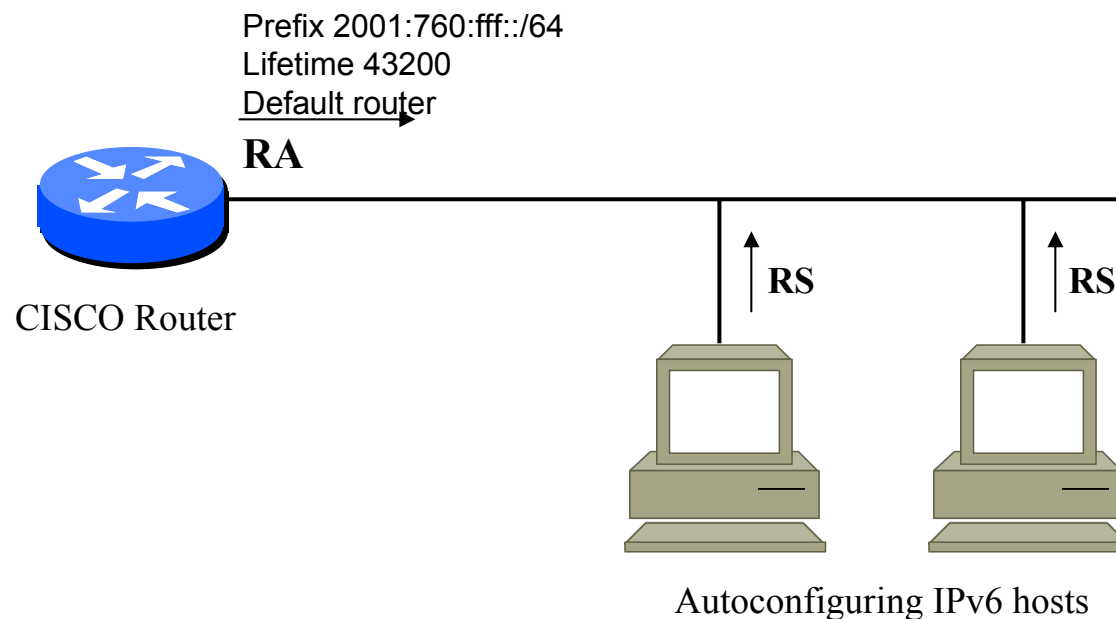
- ▶ Configuration example:

```
interface eth0 {  
    AdvSendAdvert on;  
    MinRtrAdvInterval 3; # default 0.33*MaxRtrAdvIntervall  
    MaxRtrAdvInterval 10; # Value optimized for mobile IPv6, default 600  
    prefix 2001:760:fff::/64 {  
        AdvOnLink on;  
        AdvAutonomous on;  
        AdvRouterAddr on;  
    };  
};
```

- ▶ If the lifetime is not configured is used the default value (3000000 sec)

Autoconfiguration – Cisco IOS

- ▶ Also CISCO routers can be configured to send router advertisement on the link.



Autoconfiguration – Cisco IOS

Command reference

- ▶ Command syntax:

```
ipv6 nd prefix-advertisement <prefix>/<length> <valid-lifetime> <preferred-lifetime> [onlink|autoconfig]
```

- ▶ Per interface configuration:
 - Edit the interface configuration

```
Router(config)# interface <type num/num>
```

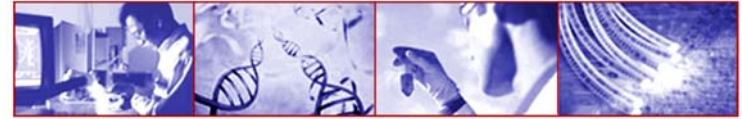
Set the parameters for RA

```
network: 2001:760:fff::/64
```

```
valid-lifetime: 43200 sec (12 hours)
```

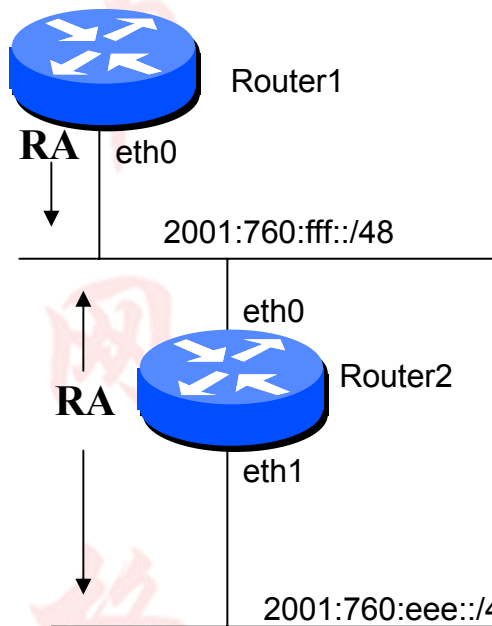
```
preferred-lifetime: 43200 sec (12 hours)
```

```
Router(config-if)#ipv6 nd prefix-advertisement 2001:760:fff::/64 43200  
43200 onlink autoconfig
```



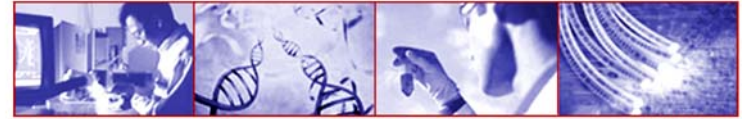
Managing lifetimes – Cisco IOS

- Lifetime values can be used for network renumbering and to change default gateway advertisement.



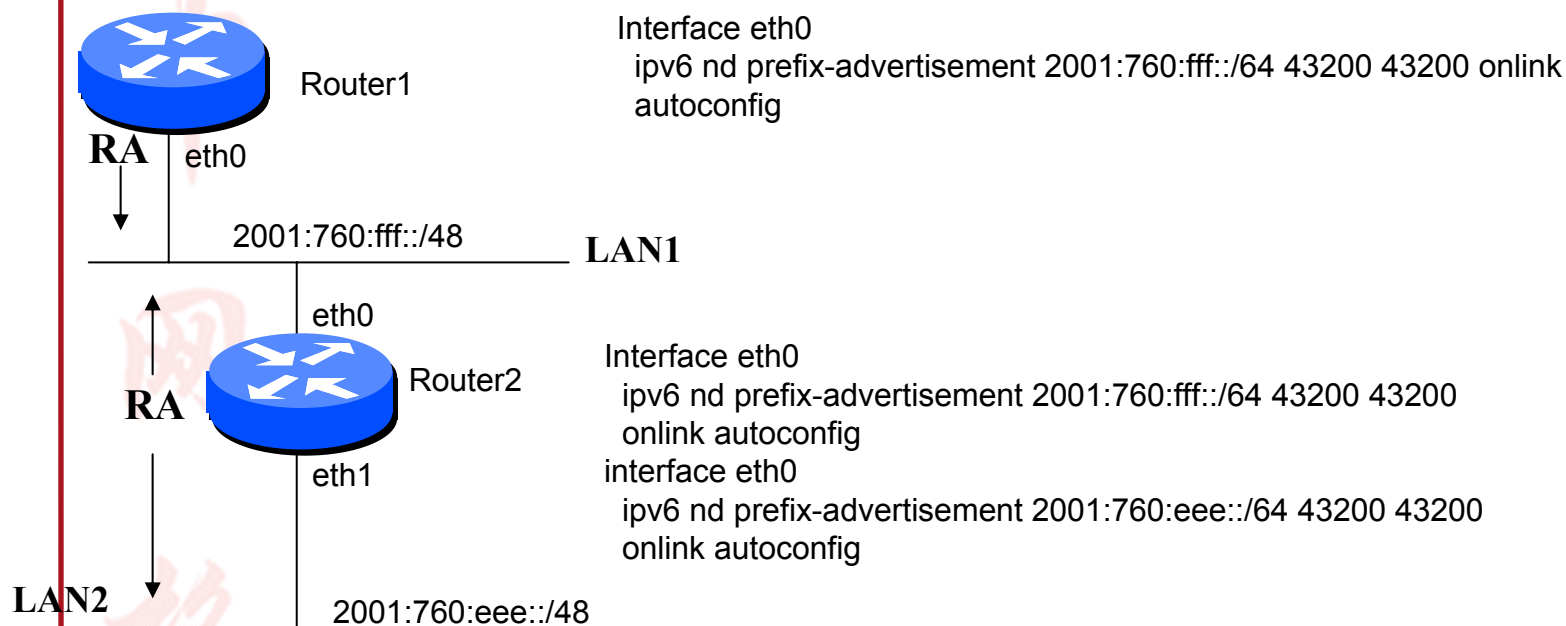
Interface eth0
 ipv6 nd prefix-advertisement 2001:760:fff::/64 43200 43200 onlink
 autoconfig

Interface eth0
 ipv6 nd prefix-advertisement 2001:760:fff::/64 43200 43200
 onlink autoconfig
 interface eth0
 ipv6 nd prefix-advertisement 2001:760:eee::/64 43200 43200
 onlink autoconfig



Managing lifetimes – Cisco IOS

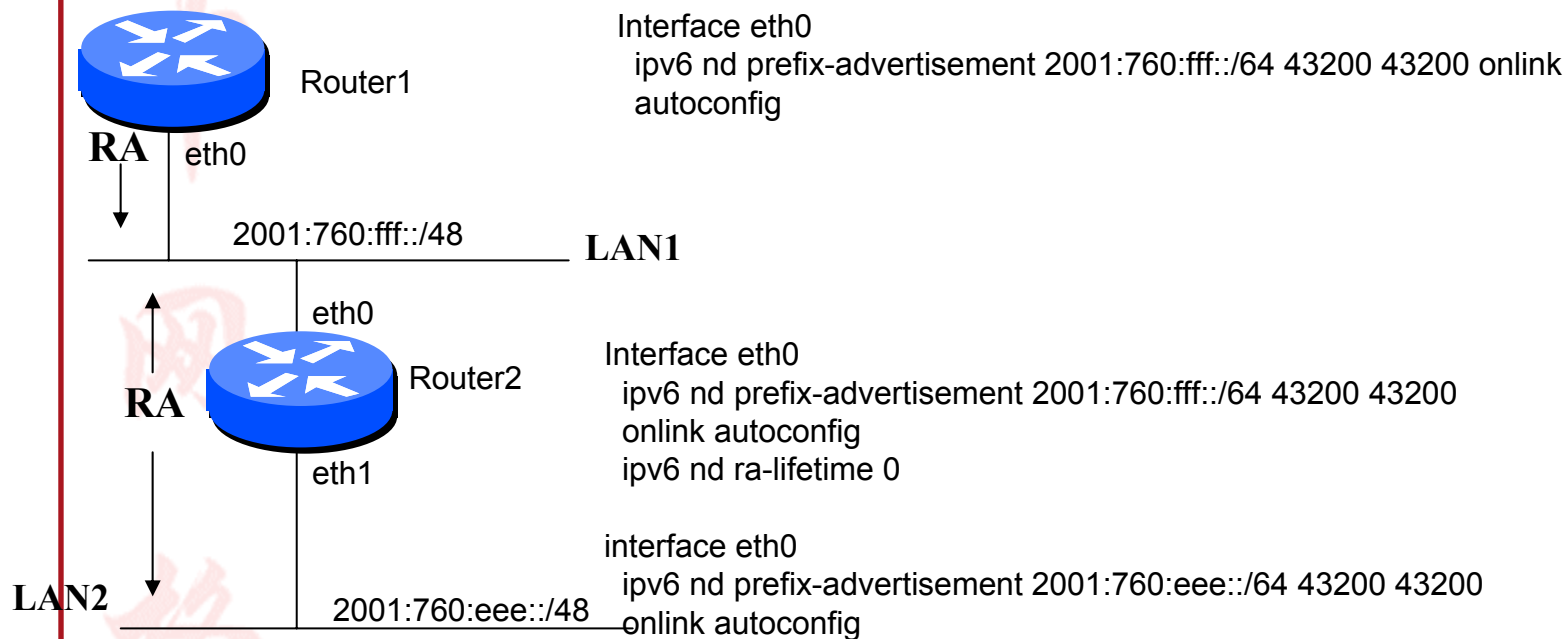
- ▶ In the example both router are configured as LAN1 default gateway.





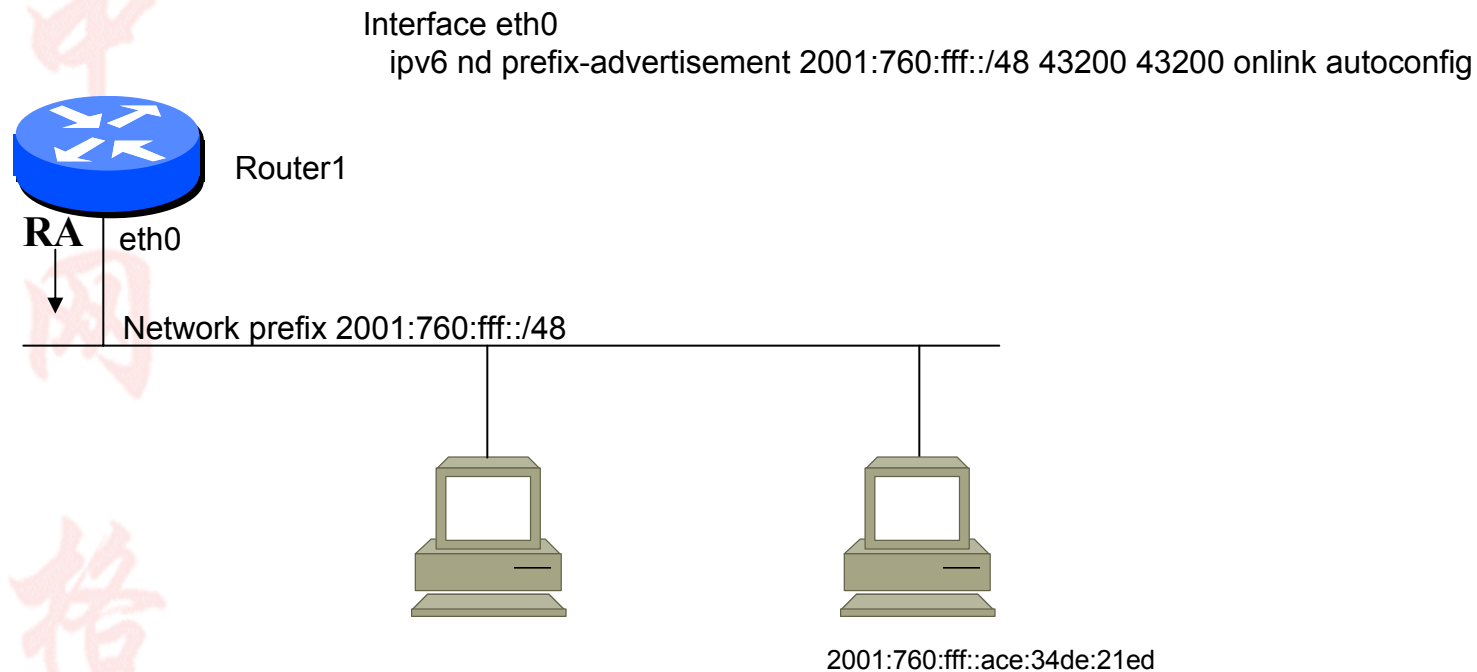
Managing lifetimes – Cisco IOS

- ▶ Using the command `ipv6 nd ra-lifetime 0` you can set ra-lifetime to zero.
- ▶ If ra-lifetime is zero the router stops the default gateway advertisements.



Renumbering – Cisco IOS

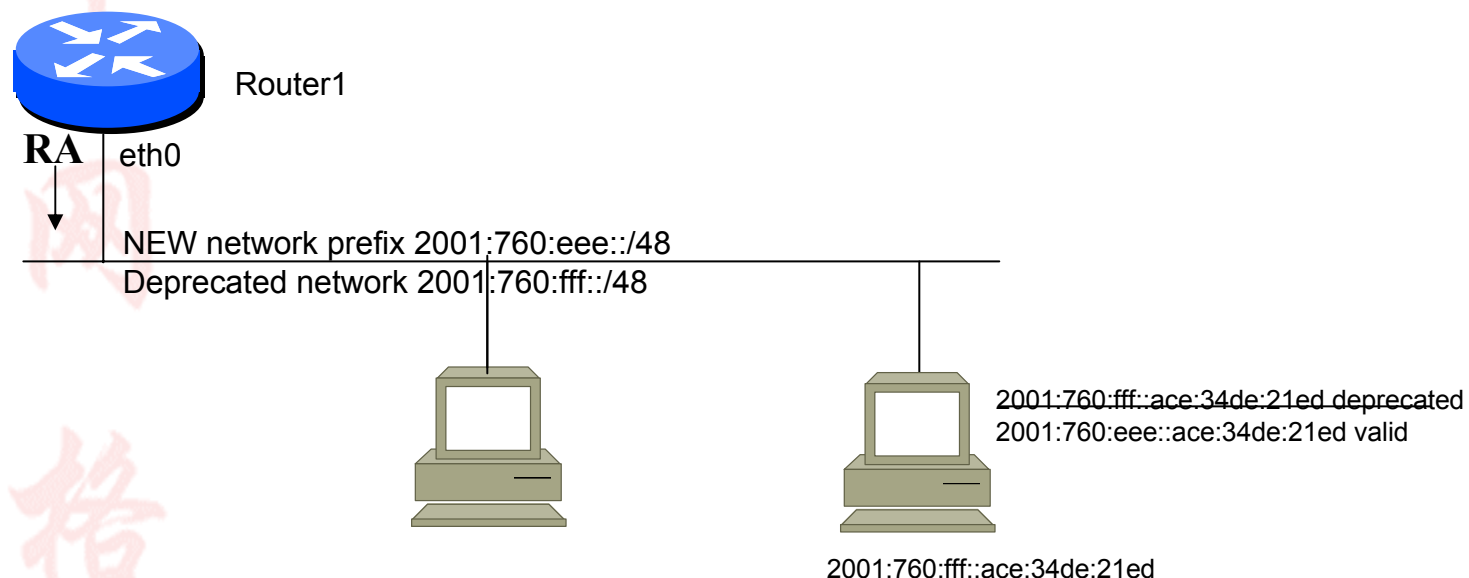
- ▶ Scenario:
- ▶ Router1 advertise to the host the network prefix 2001:760:fff::/48 with 12 hours lifetime.

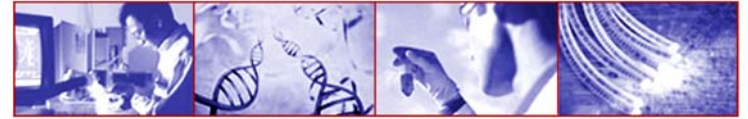




Renumbering – Cisco IOS

- ▶ If the preferred lifetime for a network is set to zero the hosts on the link sets the related network as deprecated.
- ▶ The network can be used until the valid lifetime reach zero.
- ▶ Now a new prefix advertisement can be used to renumbering the network.





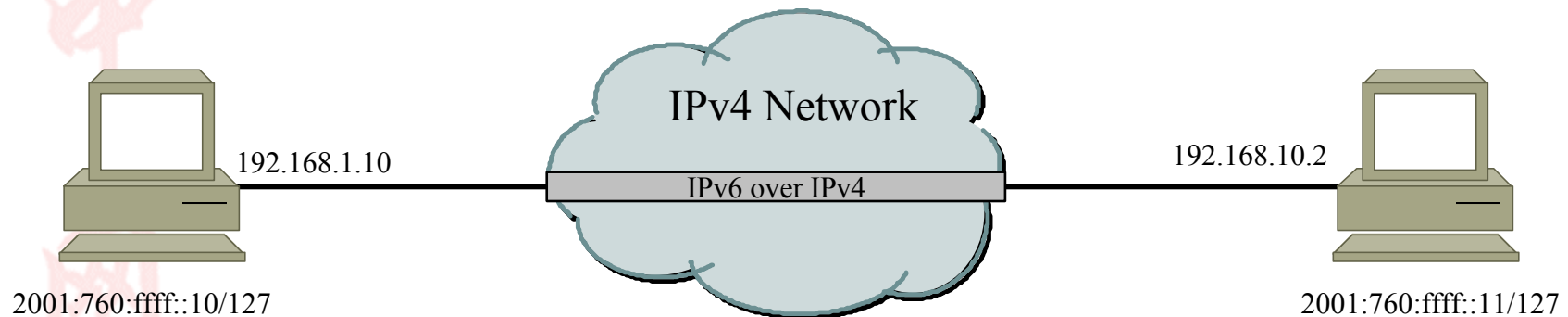
欧
中
网
格

Tunnel and Transition Tools

Configuration Examples

Manual Tunnel configuration – Linux scenario

- ▶ Creates a IPv6 connection between two hosts using IPv4 like a lower layer





Manual Tunnel Configuration – Linux scenario

▶ We could activate IPv6 transport over IPv4 network using tunnel connections.

▶ The linux utility iptunnel is used to create, delete, and modify ip-over-ip, gre, sit, and other kinds of tunnels

```
iptunnel {add|change|del|show} NAME mode {ipip|gre|sit} remote <endpointaddr> local  
<localaddr> [ ttl TTL ] [ tos TOS ] [ nopmtudisc ] [ dev PHYS_DEV ]
```



Manual Tunnel Configuration – Linux

scenario

▶ Step-by-step configuration:

Tunnel configuration and tunnel interface creation;

```
#iptunnel add sit1 remote 192.168.10.2 local 192.168.1.10 mode sit ttl  
64
```

IPv6 address configuration on the primary interface;

```
#ifconfig sit1 inet6 add 2001:760:ffff::10/127
```

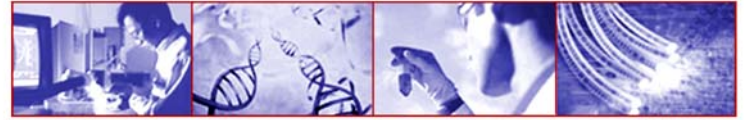
Tunnel interface activation;

```
#ifconfig sit1 up
```

Static route for the tunnel;

```
#route --inet6 add default gw 2001:760:ffff::11
```

▶ Using the static route explained in the example all IPv6 traffic flows to the tunnel.



Manual Tunnel Configuration – Linux scenario

```
#route --inet6
```

```
.....
```

```
::/0          2001:760:fff::11 UG  1    0    0 sit1
```

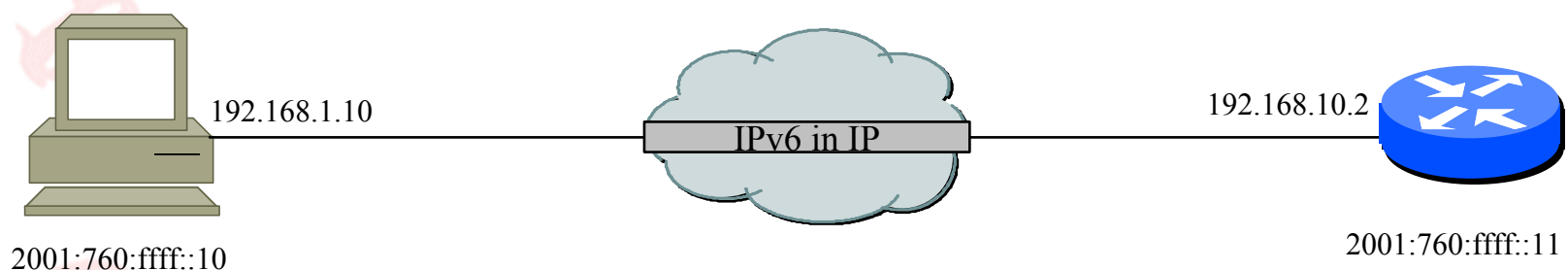
```
#iptunnel show
```

```
sit1: ipv6/ip remote 192.168.10.2 local any ttl 64
```

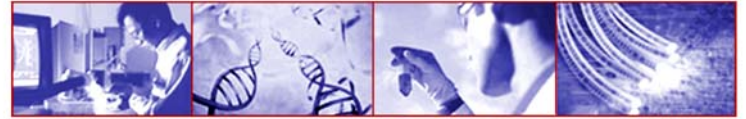


Manual Tunnel Configuration

```
C>ipv6 rtu ::0 2/::192.168.10.2 pub  
c>ipv6 adu 2/2001:760:ffff::10
```



```
interface tunnel0  
ipv6 address 2001:760:ffff::11  
tunnel source 192.168.10.2  
tunnel destination 192.168.1.10  
tunnel mode ipv6ip
```



欧
中
网
格

Introduction to IPv6 Programming



Introduction to IPv6 Programming in C

- ▶ The new generic interface for IPv6 socket programming is addressed in RFC-3493 “Basic socket interface extension for IPv6”
- ▶ The RFC defines new:
 - Core socket functions
 - Address data structures
 - Name-to-Address translation functions
 - Address conversion functions
- ▶ RFC-3493 aims to make easier the conversion of the existing IPv4 application in IPv6
 - ▶ Using this API, applications should not need to know which type of host they are communicating with.



Core Socket Function

- ▶ These functions are used to set up TCP connections or to send and receive data in UDP streaming.
- ▶ No changes are needed in socket functions to address IPv6 connection but a new set of data structures was defined.



Address Data Structure

- ▶ A new address family name, AF_INET6 was defined for IPv6; the related protocol family is PF_INET6, and names belonging to it are defined as follow:

```
#define      AF_INET6      10
#define      PF_INET6      AF_INET6
```



Address Data Structure

- ▶ Socket functions handle generic address structures. Like IPv4 `sockaddr_in` structure, a specific socket address structure is defined for IPv6 family.

```

struct sockaddr_in6 {
    sa_family_t  sin6_family;    /* AF_INET6 */
    in_port_t    sin6_port;     /* transport layer port # */
    int32_t      sin6_flowinfo; /* IPv6 traffic class & flow info
*/
    struct in6_addr sin6_addr;   /* IPv6 address */
    uint32_t     sin6_scope_id; /* set of interfaces for a scope */
};
    
```



Address Data Structure

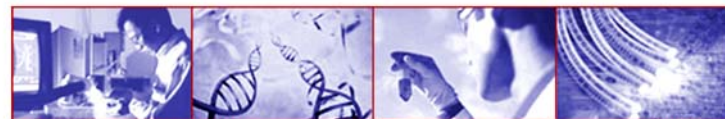
- ▶ Structure `in6_addr` is defined as follow:

```
struct in6_addr {  
    union {  
        uint8_t  _S6_u8[16];  
        uint32_t _S6_u32[4];  
        uint64_t _S6_u64[2];  
    } _S6_un;  
};  
  
#define s6_addr _S6_un._S6_u8
```



Address Data Structure

- ▶ In order to write portable and multiprotocol applications, another data structure is defined: the new *sockadd_storage*.
- ▶ This function is designed to store all protocol specific address structures with the right dimension and alignment.
- ▶ The implementation of the structure is platform dependent, a linux implementation is:



Address Data Structure

欧
中
网
格

```

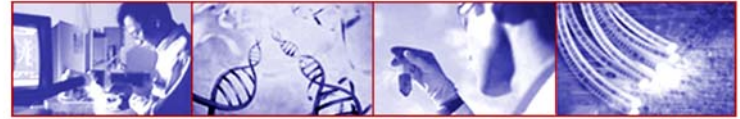
/* Desired design of maximum size and alignment (see RFC2553) */
#define    _K_SS_MAXSIZE          128
                /* Implementation specific max size */
#define    _K_SS_ALIGNSIZE    (__alignof__ (struct sockaddr *))
                /* Implementation specific desired alignment */
struct __kernel_sockaddr_storage {
    unsigned short    ss_family; /* address family */
    /* Following field(s) are implementation specific */
    char    __data[_K_SS_MAXSIZE - sizeof(unsigned short)];
                /* space to achieve desired size, */
                /* _SS_MAXSIZE value minus size of ss_family */
} __attribute__((aligned(_K_SS_ALIGNSIZE)));
                /* force desired alignment
#define sockaddr_storage    __kernel_sockaddr_storage
    
```



Name to Address Translation

Function

- ▶ The use of name instead of address in applications is advisable: in fact, usually the hostname remains the same, while the address may change more easily.
- ▶ Moreover, IPv6 addresses, and especially the EUI-64 automatic addresses, are more complex than IPv4 ones.



Name to Address Translation

Function

- ▶ The `gethostbyname2` function is used to convert the hostname in numerical address.
- ▶ The prototype of this function is

```
struct hostent *gethostbyname2(const char *name, int af);
```

- ▶ The use of this function is deprecated because this function is address-family dependent.
- ▶ A new generic function, *getaddrinfo*, and a new structure, *addrinfo*, are defined for these



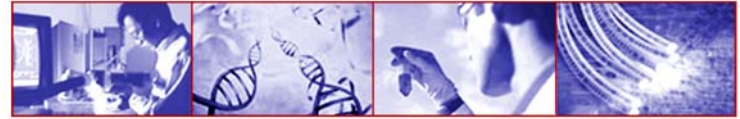
Name to Address Translation

Function

- ▶ The *getaddrinfo* function is very flexible and has several modes of operation.
- ▶ It returns a dynamically allocated linked list of *addrinfo* structures.
- ▶ The structure *addrinfo* is defined as follows:

```

struct addrinfo {
    int    ai_flags;           /* AI_PASSIVE, AI_CANONNAME, .. */
    int    ai_family;         /* AF_INET6, AF_INET, .. */
    int    ai_socktype;       /* SOCK_xxx */
    int    ai_protocol;       /* 0 or IPPROTO_xxx for IPv4 and IPv6 */
    socklen_t ai_addrlen;     /* length of ai_addr */
    char *ai_canonname;       /* canonical name for nodename */
    struct sockaddr *ai_addr; /* binary address */
    struct addrinfo *ai_next; /* next structure in linked list */
};
    
```



Name to Address Translation

Function

- ▶ The function to get socket address structures is

```
int getaddrinfo(const char *nodename, const char *service, const struct  
addrinfo *hints, struct addrinfo **res);
```

- ▶ Normally, in client applications both *nodename* and *service* are specified.
- ▶ In server applications *nodename* may be null
- ▶ It is used to allow server to accept client connections on any node address.



Name to Address Translation

Function

- ▶ To free the memory allocated for the addrinfo list, the freeaddrinfo function can be used.

```
void freeaddrinfo(struct addrinfo *res);
```



Address Conversion Function

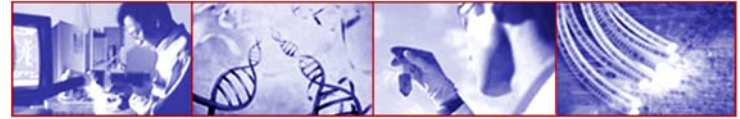
- ▶ The new address conversion functions for both IPv4-IPv6 are:

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

- ▶ converts the character string *src* into the network address structure *dst* in the *af* address family.

```
const char *inet_ntop(int af, const void *src, char *dst, socklen_t cnt);
```

- ▶ converts the network address structure *src* in the *af* address family into character string *dst* which is *cnt* bytes long



Porting application to IPv6₍₁₎

欧
中
网
格

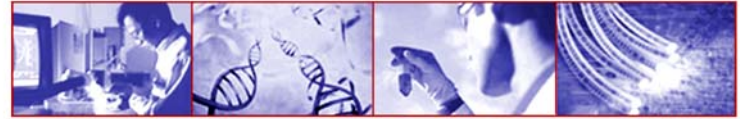
- ▶ To port IPv4 applications in a multi-protocol environment, developers should look out for these basic points
 - Use DNS names instead of numerical addresses
 - Replace incidental hard-coded addresses with other kinds
 - Sequences of zeros can be replaced by double colons sequence :: only one time per address, e.g. The previous address can be rewritten as 2001:760:40ec::12:3a
 - In the IPv6 RFCs and documentation, the minimum subnet mask is shown as /64, but in some cases, like point-to-point connections, a smaller subnet (such as /126) can be used.
 - In numerical addressing, RFC2732 specifies that squared brackets delimit IPv6 address to avoid mismatches with the port separator such as `http://[2001:760:40ec::12.3a]:8000`



Porting application to IPv6⁽²⁾

欧
中
网
格

- Applications in a dual-stack host prefer to use IPv6 address instead of IPv4
- In IPv6, it is normal to have multiple addresses associated to an interface. In IPv4, no address is associated to a network interface, while at least one (link local address) is in IPv6.
- All functions provided by broadcast in IPv4 are implemented on multicast in IPv6.
- The two protocols cannot communicate directly, even in dual-stack hosts. There are some different methods to implement such communication, but they are out of scope of this document.



Rewriting applications

- ▶ To rewrite an application with IPv6 compliant code, the first step is to find all IPv4-dependent functions.
- ▶ A simple way is to check the source and header file with UNIX grep utility.

```
$ grep sockaddr_in *.c *.h
```

```
$ grep in_addr *.c *.h
```

```
$ grep inet_aton *.c *.h
```

```
$ grep gethostbyname *.c *.h
```



Rewriting applications

欧
中
网
格

- ▶ Developers should pay attention to hard-coded numerical address, host names, and binary representation of addresses.
- ▶ It is recommended to made all network functions in a single file.
- ▶ It is also suggested to replace all *gethostbyname* with the *getaddrinfo* function, a simple switch can be used to implement protocol dependent part of the code.
- ▶ Server applications must be developed to handle multiple listen sockets, one per address family, using the *select* call.



IPv6 and Java

欧
中
网
格

- ▶ Java APIs are already IPv4/IPv6 compliant. IPv6 support in Java is available since 1.4.0 in Solaris and Linux machines and since 1.5.0 for Windows XP and 2003 server.
- ▶ IPv6 support in Java is automatic and transparent. Indeed no source code change and no bytecode changes are necessary. Every Java application is already IPv6 enabled if:
 - It does not use hard-coded addresses (no direct references to IPv4 literal addresses);
 - All the address or socket information is encapsulated in the Java Networking API;
 - Through setting system properties, address type and/or socket type preferences can be set;
 - It does not use non-specific functions in the address translation.