Communications Server z/OS® V1R5 and V1R6 Technical Update

OMPROUTE for z/OS® V1R5

Topics

z/OS V1R5

- Ability to redirect OMPROUTE debug trace to CTRACE
- Increased number of multipath routes from 4 to 16
- Ability to ignore interfaces not defined to OMPROUTE
- Cisco-compatible MD5 authentication key definition
- Increase number of local interfaces supported
- Ability to display generic interfaces
- RIPng - RIP for IPv6

OMPROUTE - IPv4 enhancements in z/OS V1R5

IPv4-based enhancements to OMPROUTE

- Ability to redirect OMPROUTE debug trace to CTRACE
  - Existing trace capabilities of OMPROUTE have not been adequate in dealing with the amount of trace data that is being generated by OMPROUTE in complex network topologies
- Increased number of multipath routes from 4 to 16
  - 4 was increasingly becoming a bottleneck
- Ability to ignore interfaces not defined to OMPROUTE
  - Previously, OMPROUTE would install some default attributes for all interfaces that were not defined to OMPROUTE - causing more confusion than good
- Cisco-compatible MD5 authentication key definition
  - Just different ways of entering an MD5 key
- Increase number of local interfaces supported
  - Was 255 in total (real plus VIPAs)
- Ability to display generic interfaces
  - Generic interfaces are interfaces that are not configured as OSPF or RIP

OMPROUTE tracing to CTRACE

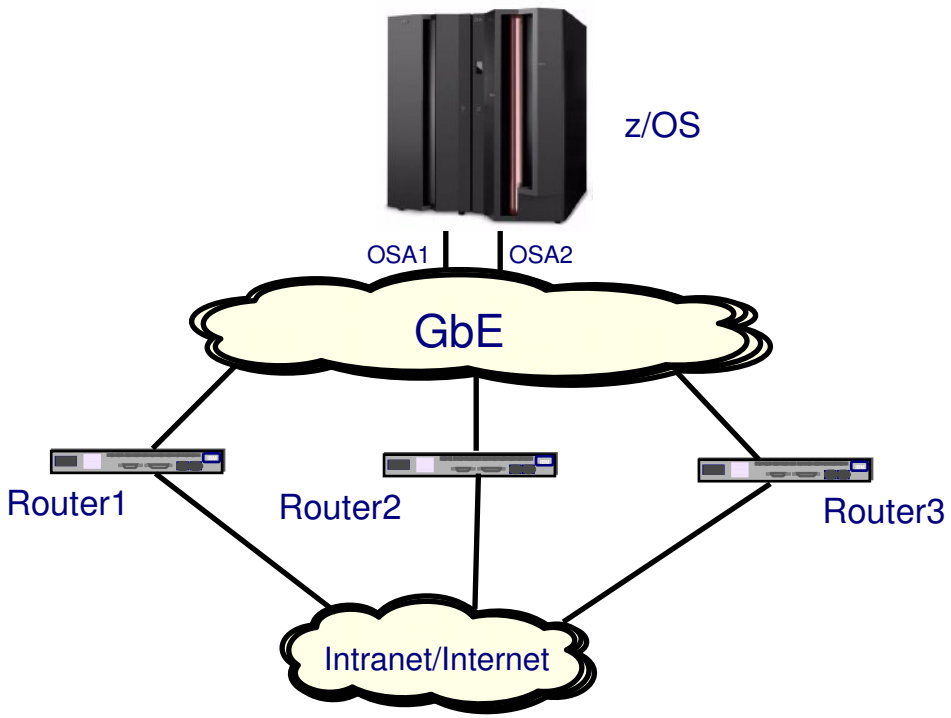
- OMPROUTE debug trace can be optionally written to OMPROUTE CTRACE
  - OMPROUTE already had a limited CTRACE capability
  - This facility has been enhanced
- New OMPROUTE CTRACE option triggers this
  - also included in ALL option
- As with all CTRACE functions, CTRACE is recorded internally and, optionally, an external writer may be used.
  - If internal CTRACE is used, the CTRACE buffer will reside in the OMPROUTE address space so a dump of OMPROUTE will provide all trace information
- You specify trace and debug levels in OMPROUTE normally
  - use -tn -dn (for IPv4) or -6tn -6dn (for IPv6) command line parameters, and/or
  - use modify commands to change trace and debug levels
- However, if new OMPROUTE CTRACE option DEBUGTRC is set, these traces will now be written to OMPROUTE CTRACE
  - no other debug or trace files will be created or written to, regardless of values of OMPROUTE_DEBUG_FILE environment variable

```
TRACEOPTS ON
 BUFSIZE(50M)
        OPTIONS('OPACKET '
                ,'RPACKET '
                ,'IPACKET '
                ,'SPACKET '
                ,'DEBUGTRC'
                )
```

PTFed back to OS/390 V2R10 - APAR number PQ76172

MULTIPATH support increased from 4 to 16 equal-cost routes

- OMPROUTE could only calculate four equal-cost dynamic routes to the same destination
  - if more routes to a destination were needed, static routes (which have no multipath limits) had to be used
- For most installations this was sufficient, but the example below shows a case in which it was a problem
- OMPROUTE in z/OS V1R5 has raised the number of equal-cost routes to 16



Undefined interfaces in the OMPROUTE configuration file

- For IPv4, OMPROUTE sets the BSDROUTINGPARMS MTU value and subnet mask on every interface
- If an interface exists on the stack but is not defined to OMPROUTE, OMPROUTE sets default values for the MTU and subnet mask, overriding any conflicting BSDROUTINGPARMS definitions
  - Default MTU size is 576
  - Default subnet mask is the class mask of the interface's home address
    - This could cause undesirable results. For example, an undefined interface with home address 9.67.101.12 would cause OMPROUTE to assign a subnet mask of 255.0.0.0. This in turn would cause OMPROUTE to add a route to the stack's routing table to 9.0.0.0/8 over that interface -- which is probably not what was intended.
    - If OMPROUTE is an Autonomous System Boundary Router and is importing direct routes, it would also advertise that 9.0.0.0/8 route to the rest of the OSPF and/or RIP routers, causing all traffic to unknown destinations in the 9 network to be sent to the router and then forwarded over that interface!

- Because of these problems, IBM has long recommended that every stack interface be defined to OMPROUTE, regardless of whether or not dynamic routing will be used over it.
    - INTERFACE configuration statement used for interfaces over which no dynamic routing will be done
- OMPROUTE can now be configured to ignore stack interfaces which are not defined to it
  - If a stack interface matches any wildcard interface definitions in the OMPROUTE configuration file, it is considered to be defined and is not eligible to be ignored.

- If an interface is being ignored by OMPROUTE:
  - No BSDROUTINGPARMS values will be updated for it by OMPROUTE
  - OMPROUTE will not add a direct route to its subnet to the TCP/IP routing table
  - Neither its home address nor its subnet will be advertised in RIP or OSPF
  - Static routes which use this interface will not be accepted from TCP/IP, and therefore can't be advertised by OMPROUTE
  - In short, OMPROUTE will behave as if the interface simply does not exist

- This may be a good way to "hide" certain interfaces from the rest of the network
  - For example, dynamic XCF if it is not being used to route user data, or an interface attached to a test network

Controlling undefined interfaces in the OMPROUTE configuration file

New configuration statement for the OMPROUTE configuration file:

```
                                +--IGNORE_UNDEFINED_INTERFACES=NO-----+
                                |                                     |
>>----GLOBAL_OPTIONS------+-----------------------------------------------><
                                |                                     |
                                +--IGNORE_UNDEFINED_INTERFACES=value--+
```

Possible values are YES or NO. Default is NO.

- When an interface is ignored, the following messages are seen in UNIX System Services syslog and/or OMPROUTE debug trace:

```
EZZ7871I No matching interface statements for 9.67.101.3 (CTC3TO4)
EZZ7975I Ignoring undefined interface CTC3TO4
```

- Also, you can display ignored interfaces using the new GENERIC display commands (see later topic "Ability to display generic interfaces")
- This configuration statement affects both IPv4 and IPv6
- You should carefully consider whether you want to implement this function
- As far as OMPROUTE is concerned, ignored interfaces do not exist but you can still reach the interface from other hosts by using static routes
- Also remember that an interface may be undefined because of typos in the OMPROUTE configuration file for example, misspelling the interface name or mistyping the home address this possibility always existed, but if undefined interfaces are being ignored it may not be as easy to notice -- the interface simply "disappears" from OMPROUTE

OSPF MD5 incompatibilities with Cisco routers

- For OSPF MD5 authentication, OMPROUTE in prior releases requires that the 16-byte MD5 key be specified in hexadecimal on the OSPF_INTERFACE statement
  - How you arrive at that key is of no concern to OMPROUTE
  - IBM provides the `pwtokey` utility to use standard methods to generate MD5 keys from passwords
- Cisco routers require the MD5 key to be defined as a password
- But Cisco routers do not use standard methods to convert that password into a 16-byte hexadecimal MD5 key
  - Other vendor routers, such as Extreme Networks, have also adopted the Cisco method
- So MD5 keys generated using `pwtokey` are not compatible with MD5 keys generated by Cisco from the same password
- To make OMPROUTE MD5 keys compatible with Cisco MD5 keys, it was necessary for customers to hand-convert the Cisco key into hexadecimal
  - Cisco simply uses the hexadecimal ASCII values of the specified password, padded to 16 bytes with hexadecimal zeroes if necessary
- You would have had to convert the Cisco key into ASCII hex and input those hex values to OMPROUTE, for example to be compatible with this Cisco definition:

```
ip ospf message-digest-key 4 md5 ABCDEFGHIJKLMNOP
```

- You would have had to code this on the OSPF_INTERFACE statement in OMPROUTE:

```
Authentication_Type      = Md5
Authentication_Key_ID    = 4
Authentication_Key       = 0x41424344454647484 94A4B4C4D4E4F50
```

- Where 0x41424344454647484 94A4B4C4D4E4F50 is the ASCII hexadecimal value for ABCDEFGHIJKLMNOP

A more compatible way of specifying MD5 keys

- OMPROUTE is enhanced to allow specification of MD5 keys using the same method as Cisco, Extreme Networks, and other vendor routers that use the Cisco method.

- If the MD5 key is specified to OMPROUTE as a Cisco-compatible key, OMPROUTE converts it to hexadecimal using the same nonstandard method that these routers use
  - even though the router method is not an RFC standard method like **`pwtokey`**, it has become a defacto standard because of widespread implementation in routers

- The Authentication_Key parameter on the OSPF_INTERFACE statement is modified:

  - For authentication type MD5, the value can now be coded in one of two ways:

    1 The previously existing method is with a 16-byte hexadecimal string beginning with 0x (0x plus 32 hex characters).

    2 An additional method has been added to provide compatibility with Cisco, Extreme, and other vendor routers that use a Cisco-compatible CLI interface. With this method, the MD5 key is coded as an ASCII string, specified in double quotes, prefixed with A, for example:

```
AUTHENTICATION_KEY = A"ABCDEFGHIJKLMNOP"
```

More than 255 interfaces in OMRPOUTE

- OMPROUTE was only able to handle a limit of 255 local interfaces
- This limit included physical interfaces and VIPAs.
  - so for example if a TCP/IP stack had 250 VIPAs, OMPROUTE could only support 5 additional physical interfaces
- This limit comes from the days before dynamic VIPA, dynamic XCF, Hipersockets, and other z/OS features which can result in a large number of local interfaces
  - this limit was never thought to be a factor for anyone
    - few routers had more than a few interfaces
  - But with widespread adoption of these features on z/OS, this limit is starting to be felt
- The limit on how many interfaces OMPROUTE can support has been significantly relaxed
- For IPv4 OMPROUTE can now support
  - 255 physical interfaces plus
  - an unlimited number of VIPAs
    - (within protocol limits, see next slide)
- For IPv6 there is no limit on the number of physical or VIPA interfaces that OMPROUTE can support
- There are no externals to code or monitor for this enhancement

Migration considerations for over 255 interfaces in OMPROUTE

- Certain limits imposed by the OSPF protocol and by other routers have not been repealed
- For IPv4, the OSPF protocol requires that a router has to be able to advertise all of its local interfaces in one packet.
- The maximum packet size allowed in OSPF is 65535 bytes
- Some routers have smaller limits
  - some routers may not allow OSPF packets larger than the MTU size on the link that they will be sent or received on -- packets that are "too big" are dropped
    - since an OSPF router advertisement gets passed all around the OSPF area, this can get to be a severe limitation if there are small MTU sizes in the area
  - OMPROUTE's OSPF packet size limit is the largest MTU size of all its interfaces
    - coding a VIPA with MTU size of 65535 ensures OMPROUTE's limit is as large as possible
- The IP Configuration guide gives the following guidelines for determining how many bytes are needed to advertise certain OSPF interface types, in addition to the 52 byte router LSA header:
  - VIPA: 12 bytes for each VIPA host route, plus 12 bytes for each VIPA subnet route
  - Point-to-point: 24 bytes
  - Point-to-multipoint: 12 bytes, plus 12 bytes for each neighbor on the interface
  - All other types: 12 bytes
- From this, you can see that it will take a lot of VIPAs to exceed 64K in a router advertisement

Display generic interfaces in OMPROUTE

- It is difficult to get information from OMPROUTE about generic interfaces
  - A generic interface is an interface over which no routing protocol is being run
  - defined to OMPROUTE using INTERFACE (for IPv4) or IPV6_INTERFACE (for IPv6) statements
  - Interfaces which are not defined to OMPROUTE and are not being ignored are also generic interfaces
- A new family of display commands has been introduced to OMPROUTE
  - d tcpip,tcpipjobname,omproute,generic,.....
    - for IPv4 generic interfaces
  - d tcpip,tcpipjobname,omproute,generic6,.....
    - for IPv6 generic interfaces
      - not described here, see the IPv6 Support for OMPROUTE section
- These commands display information and interfaces not related to any particular routing protocol

New OMPROUTE display command

`Display TCPIP,`*tcpipjobname*`,OMPROUTE,GENERIC,INTERFACES`

Function:

- Lists information for each IPv4 generic interface that is installed in TCP/IP
- Note that interfaces being ignored by OMPROUTE are also listed here
  - recall that these are interfaces not configured to OMPROUTE when `GLOBAL_OPTIONS IGNORE_UNDEFINED_INTERFACES` is on

Sample Output:

```
EZZ8060I IPv4 GENERIC INTERFACES
IFC NAME          IFC ADDRESS     SUBNET MASK       MTU  CFG  IGN
VIPL0943786F      9.67.120.111    255.255.255.0     576  YES   NO
CTC3TO1           130.200.1.3     N/A               N/A   NO  YES
CTC3TO7           9.67.102.3      N/A               N/A   NO  YES
```

- CFG = YES|NO indicates if the interface was configured in the OMPROUTE configuration file
- IGN = YES|NO indicates whether or not the interface is being ignored by OMPROUTE.

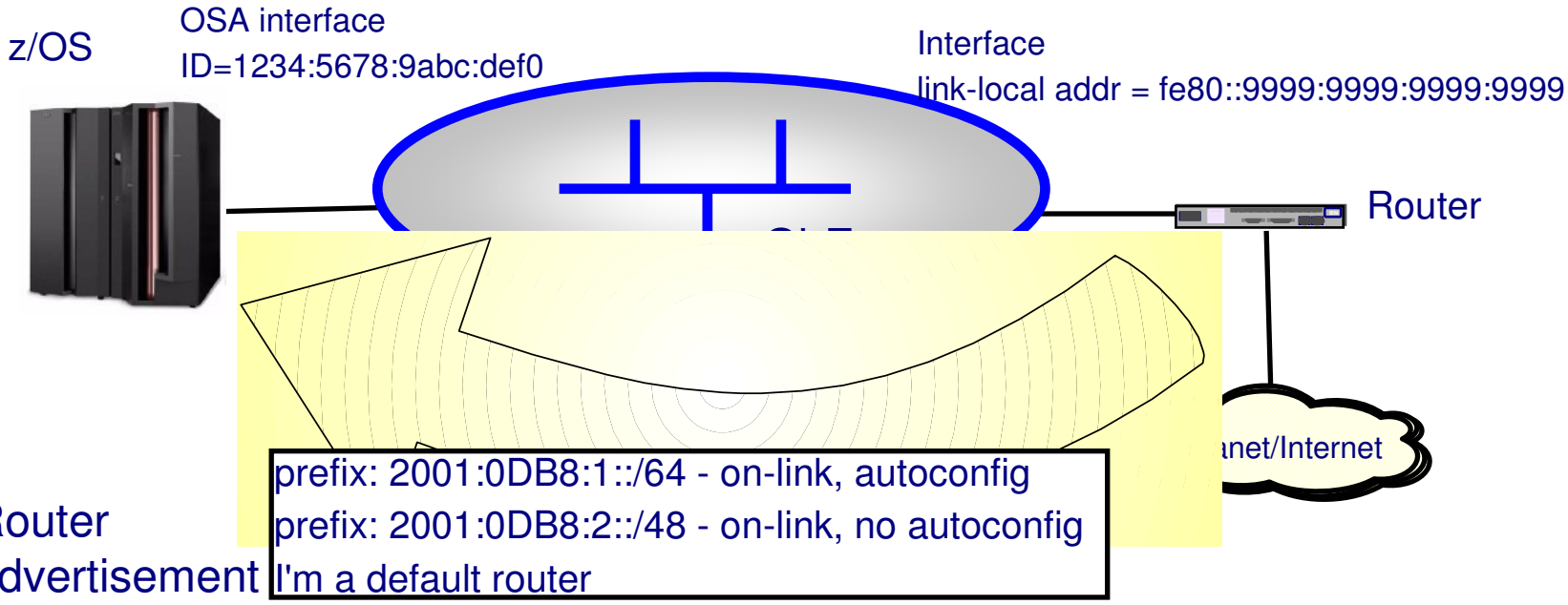
OMPROUTE Support for RIPng (RIP for IPv6) - z/OS V1R5

Basics

- Basic IPv6 does have some rudimentary built-in dynamic routing
  - Router Discovery
  - Stateless Address Autoconfiguration
  - Neighbor Unreachability Detection
  - ICMPv6 redirect processing

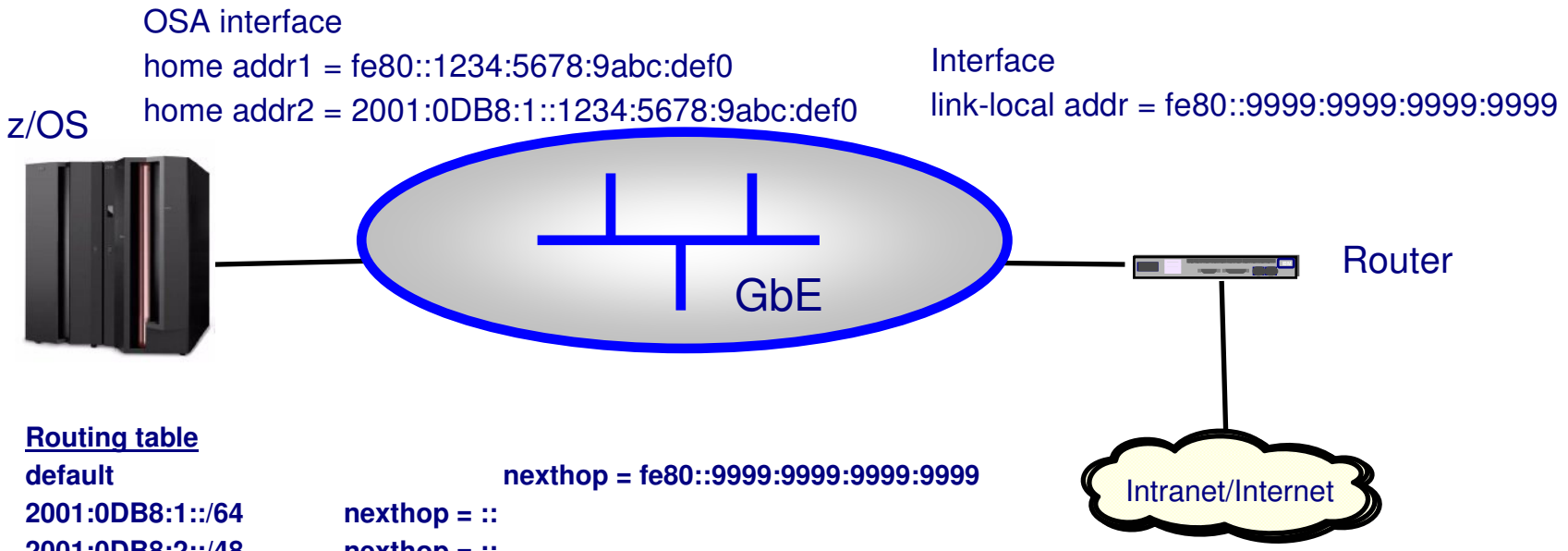
Router Discovery

- With Router Discovery, a router informs a host of what prefixes are assigned to the medium
  - a prefix is a concept that's analogous to an IPv4 subnet, but with some key differences that will be discussed later
- A router can also use Router Discovery to advertise itself as a default router
- CS for z/OS supports receiving, but not sending, router advertisements
  - this support is in the TCP/IP stack
  - no routing daemon necessary to exploit



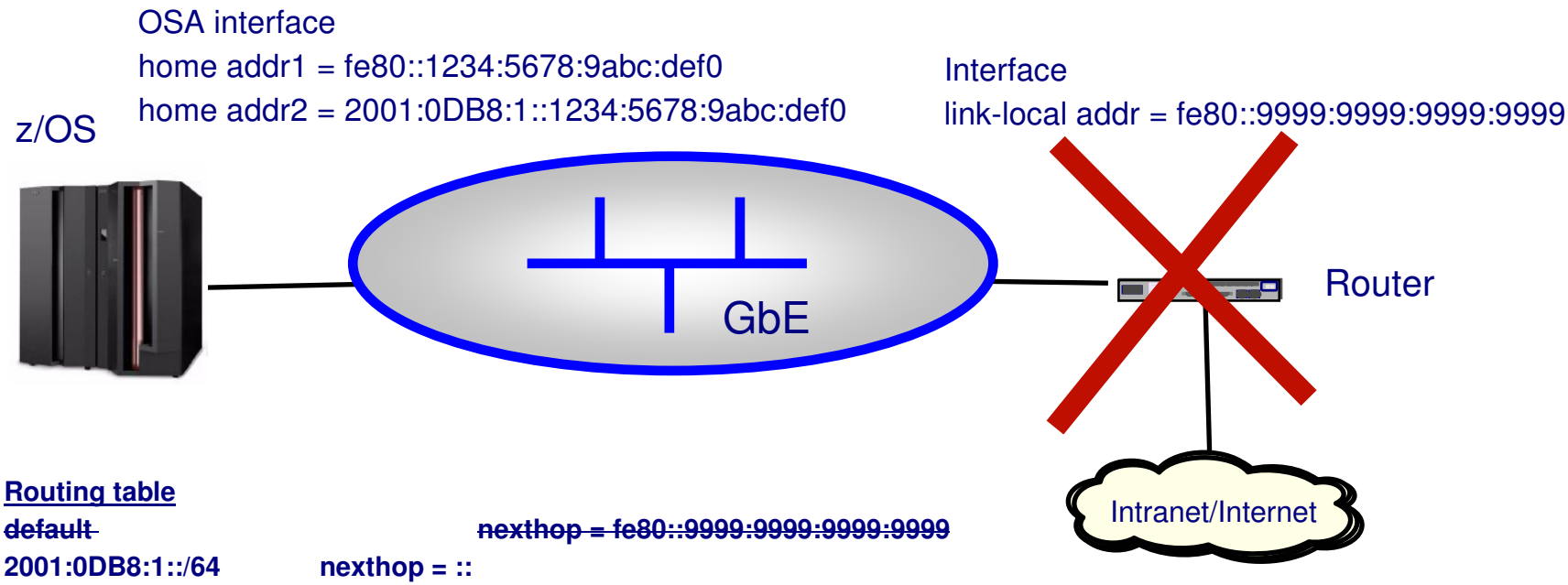
Stateless Address Autoconfiguration

- Stateless Address Autoconfiguration builds on the Router Discovery function described on the previous chart
  - A link-local address is always configured by adding the link-local prefix (FE80::/64) to the 64-bit interface ID
  - If the autoconfig flag in the Router Advertisement is set, other IP addresses are configured by adding an advertised prefix to the 64-bit interface ID
  - Routes are added to the routing table for prefixes that have the on-link flag set
  - On-link and autoconfiguration can both be set, or be independent
- The picture below shows the result of autoconfiguration and contents of the routing table, after the router advertisement in the previous slide is processed.



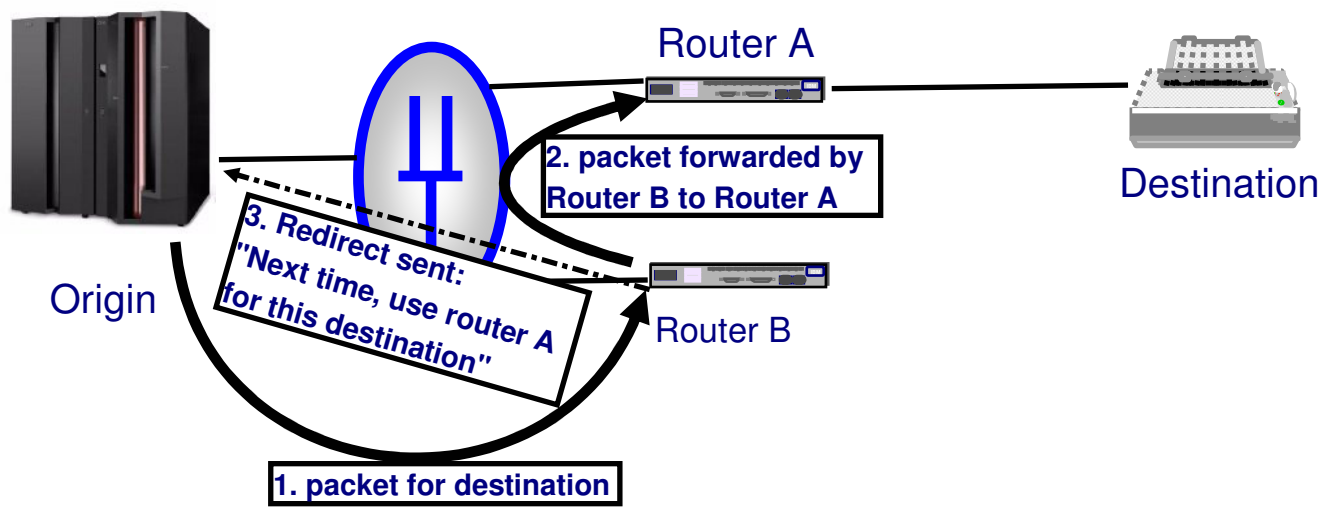
Neighbor Unreachability Detection

- Among other functions, Neighbor Discovery replaces IPv4's ARP
  - For purposes of this discussion, the important concept is that when Neighbor Unreachability Detection discovers that a router is down, the default route that it advertised is removed from the routing table
  - However, the routes to the on-link prefixes advertised in the Router Discovery packet sent by that router are not removed from the routing table, nor are the autoconfigured addresses removed from interfaces on the link.
- The picture below shows what happens in the configuration of the previous slide, when the router goes down and Neighbor Unreachability Detection learns this fact



ICMPv6 Redirect processing

- ICMPv6 Redirect processing works just like IPv4 ICMP Redirect processing
- A Redirect is sent by a next-hop router when that router is forwarding a packet toward a destination and it determines that a better next hop for the destination exists *on the same link*
  - It does not help if there is a better next hop on a different link
  - It only reacts to a packet sent on the correct link but to the wrong router. It does not help determine which link to send the packet on in the first place
- The Redirect packet causes the routing table to be updated for the destination, so that subsequent sends will use the better next hop.



Basic capabilities - summary

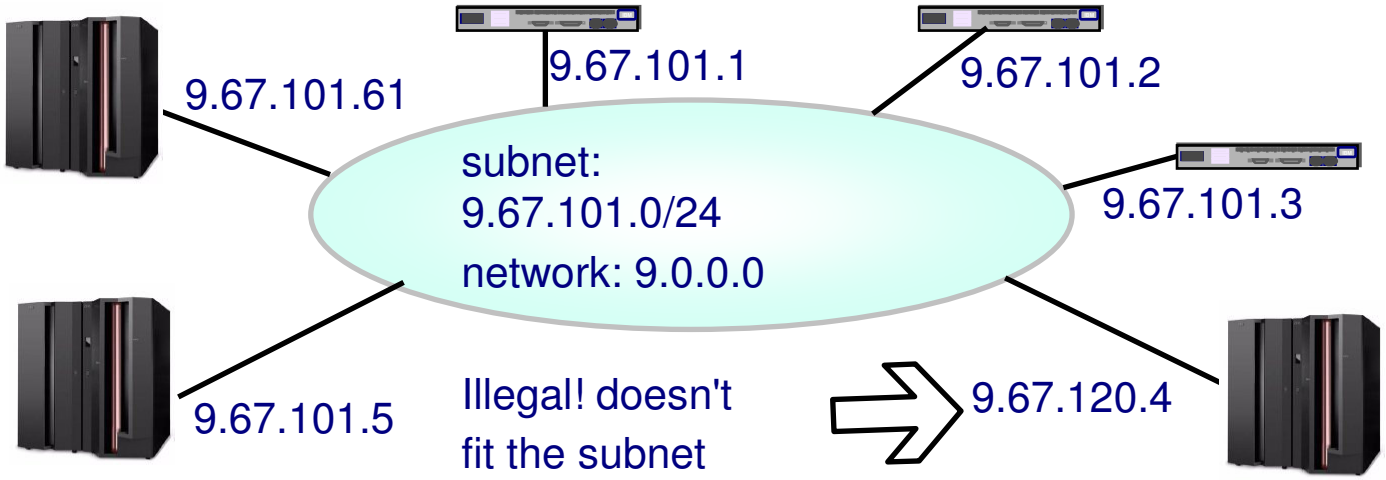
- As you can see from the foregoing, base IPv6 function provides some rudimentary dynamic routing
- However it is very limited
  - CS for z/OS cannot advertise routes and destinations
    - this particularly limits the usefulness of VIPA
  - The only routes that can be dynamically learned are the default route, direct prefix routes, and ICMP redirects
    - this limits flexibility in network design or requires extensive static routes
- Dynamic routing is needed to overcome these limitations

IPv6 Dynamic Routing, general info

- IPv6 has significant differences from IPv4, and some of them impact how dynamic routing works
- It is important that system administrators who are familiar with IPv4 dynamic routing understand these differences
  - An interface can have multiple home IP addresses
  - Prefixes instead of subnets and networks
  - Scopes, and use of link-local addresses
- In IPv4, an interface has a single home address
- In IPv6, an interface can theoretically have an unlimited number of home addresses
  - Every interface has an interface ID
    - the default value is derived from the MAC address but on most platforms, including CS for z/OS, this interface ID can be specified in TCP/IP profile
  - Every interface has at least one link-local address
    - CS for z/OS and all known implementations limit an interface to one link-local address, which is generated by prepending the link-local prefix to the interface ID
  - Additional home addresses can be added and deleted using:
    - stateless address autoconfiguration
      - if a new prefix is learned from a router, a new home address could be added by prepending that prefix to the interface ID
    - stateful address autoconfiguration
      - not supported by CS for z/OS, so not discussed here
    - customer definition
      - TCP/IP profile, vary obey

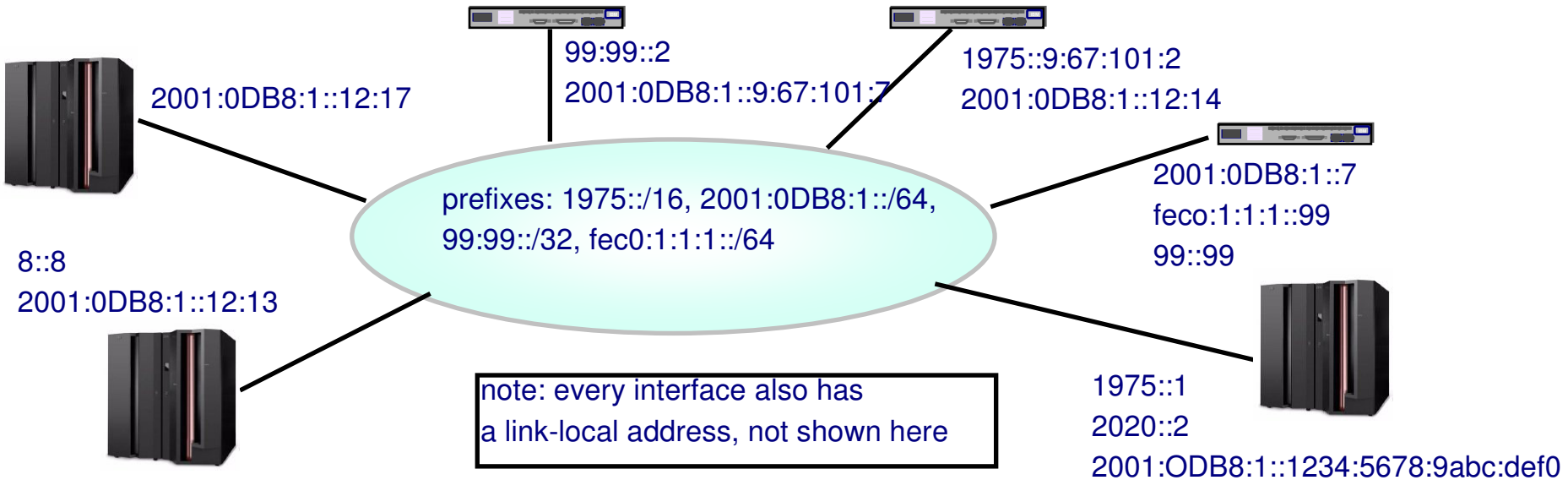
Prefixes instead of subnets and networks

- In IPv4, an IP address has a natural network
  - for example, class A, class B, class C, etc.
- In IPv4 a medium has exactly one subnet address
  - an example of a medium is an ethernet LAN, or a dynamic XCF group
  - the subnet address of a medium is mathematically related to the home address of each interface attached to the medium
    - interface_address & subnet_mask = subnet
    - all interfaces on a medium have to be in the subnet
- On the IPv4 medium depicted below, the subnet number is 9.67.101.0. The subnet mask is 255.255.255.0. The AND of the subnet mask and every interface on the medium should equal 9.67.101.0



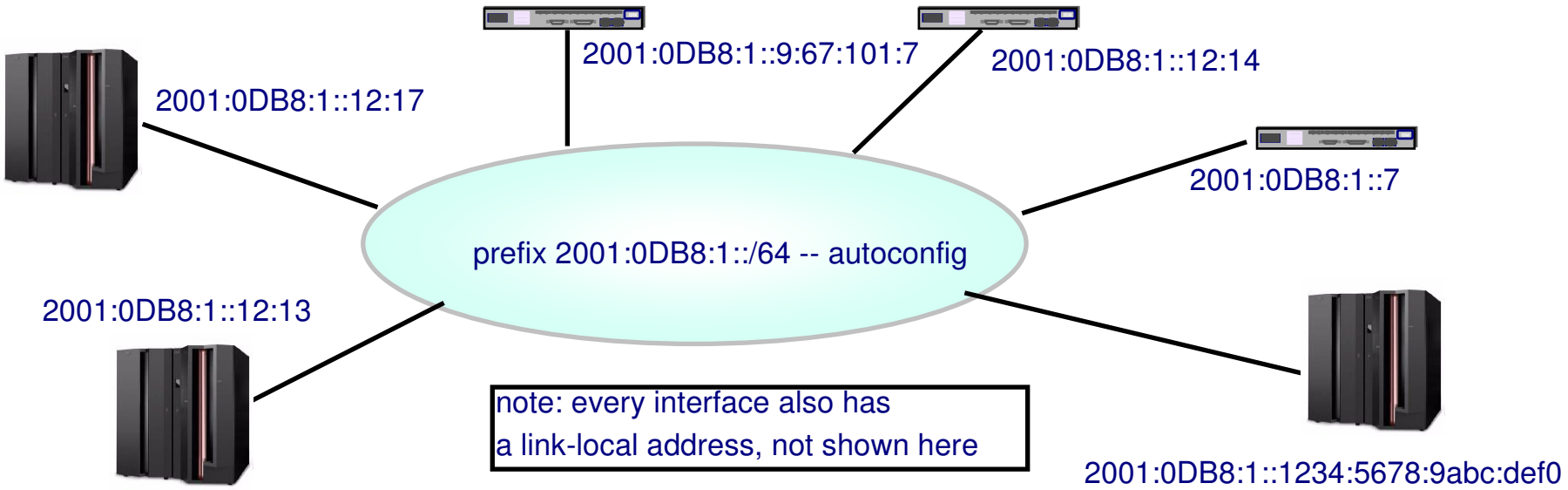
Prefixes instead of subnets and networks, continued

- The restrictions on the previous page do not apply to IPv6
- Instead of subnets and networks, the concept is ***prefix***
  - a medium can have multiple prefixes
  - the home addresses of attached interfaces can be in all, some, or none of the medium's prefixes
  - there are no "natural" prefixes
  - there is no required mathematical relationship between a medium's prefixes and the home addresses of its attached interfaces
- Address and prefix assignment is jungle rules! Anything is allowed anywhere.
- This complicates routing algorithms that in IPv4 make heavy use of subnets to simplify routing tables



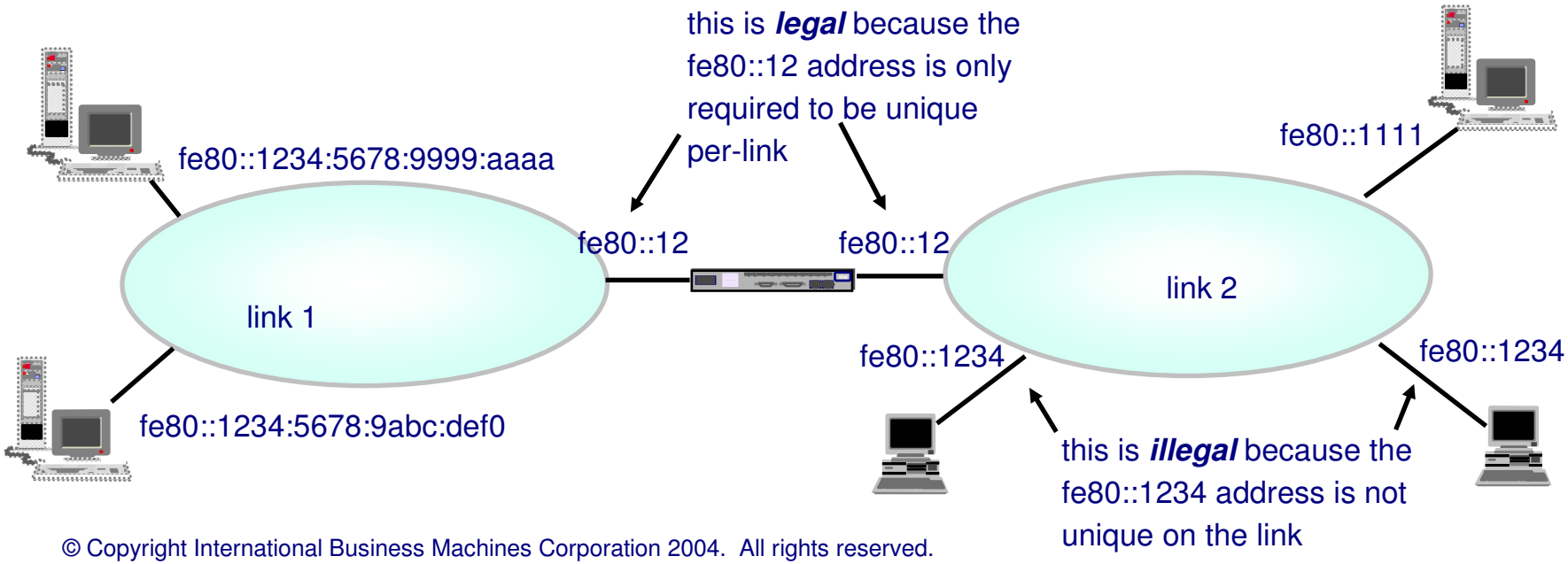
Prefixes instead of subnets and networks, continued

- Hopefully the chaos on the previous page is a worst-case scenario
- IPv6 depends on the system programmer to enforce order in a network like this
- For example, you may require all hosts on the network to use autoconfiguration
  - don't permit them to manually configure addresses
  - have your router advertise a prefix to be used by autoconfiguration
- This kind of order, while desirable, is not required by IPv6 and its routing protocols have to be able to handle the worst case



Scopes, and link-local

- In IPv6, addresses have scope that determines the domain in which they have to be unique
  - `fe80::/10`: link-local -- address only has to be unique on a link
  - `fec0::/10`: site-local -- address only has to be unique within a site
    - *note: this scope is being phased out. Do not use!*
  - all other non-multicast addresses have global scope
- Since site-local scope is being phased out, we are mainly concerned with link-local addresses



Some routing implications of link-local addresses

- Because link-local addresses do not have uniqueness beyond a link, they are not advertised or kept in routing tables
  - every route to a link-local address is a direct route onto the link where it resides
- Because every non-VIPA interface has a link-local address, dynamic route next-hops are always link-local
  - the combination of outgoing interface name and link-local address makes up a next hop
    - interface name tells which interface to forward the packet over
    - link-local address tells which node on the link will receive the packet
- A link-local gateway address is always considered to be reachable
  - all remote link-local addresses should be learned from the source address field of received RIPng packets
    - exception: static routes defined in TCP/IP profile
  - therefore if we know of a link-local address, it must be reachable
- Non-link-local next hops are only possible when the nexthops are defined to TCP/IP as direct static routes
  - static routes are defined to TCP/IP profile with BEGINROUTES statement
  - only static routes and default routes defined to OMPROUTE using the IPv6_Default_Route configuration statement can have non-link-local next hops
- The only type of interface that is not required to have a link-local address is VIPA
  - this is allowed because one can never route directly to a VIPA anyway -- always have to come in on a real interface first

Notes

NOTES

- For example, with the following TCP/IP configuration:

```
BEGINROUTES
ROUTE 1975::1/128 = NSQDIO1L6 MTU 3000
ENDROUTES
```

- It would be possible to define default routes to OMPROUTE using the IPv6_Default_Route configuration statement, which have 1975::1 as the next hop, for example:

```
IPv6_Default_Route
NEXT_HOP=1975::1
NAME=NSQDIO1L6;
```

- It would also be possible to use the statically defined direct route as a nexthop in a static route, for example:

```
BEGINROUTES
ROUTE 1975::1/128 = NSQDIO1L6 MTU 3000
ROUTE 1999::1/128 1975::1 NSQDIO1L6 MTU 3000
ENDROUTES
```

- However given the second BEGINROUTES above, this would not be a valid OMPROUTE configuration because the nexthop is not directly attached:

```
IPv6_Default_Route
NEXT_HOP=1999::1
NAME=NSQDIO1L6;
```

Note that OMPROUTE does not know at configuration time which nexthops will turn out to be directly attached and which ones won't, so there is no validity checking done on this at configuration time -- the default route simply wouldn't get installed since the nexthop will never become known to OMPROUTE as a directly attached gateway.

RIPng Basics

- Like IPv4 RIP, RIPng is a distance vector protocol
  - routers do not advertise topology to each other
  - instead they advertise routes and metrics
    - at a most basic level: one router receives a route advertisement from another router, adds its own metric, and passes it on
- RIPng is based on the IPv4 RIP protocols and has many things in common with them
  - maximum reachable metric is 15
  - routers advertise their full routing table to each other every 30 seconds
  - routes time out if not refreshed in 3 minutes
  - concepts like split horizon and poison reverse routes are implemented to minimize routing loops
  - extensive use of filters
- Because of its similarity to IPv4 RIP, RIPng shares many of its advantages and disadvantages
- Advantages:
  - simplicity
  - low CPU demand
- Disadvantages
  - limited network size
    - because of maximum metric of 15
  - slow convergence time
    - since updates sent only every 30 seconds
    - OMPROUTE does support triggered updates, which improve this
  - potential for high link usage
    - since sending full routing table every 30 seconds

OMPROUTE RIPng support basics

- OMPROUTE has been enhanced to support IPv6 and RIPng
  - Like IPv4, there is support for the routing protocol, plus support for basic IPv6 routing concepts
    - generic interfaces
      - no dynamic routing over interface
    - static routes
      - defined as usual via BEGINROUTES in TCP/IP's profile
    - direct routes
      - Routes defined via direct connection to a network
    - prefix and router advertisement routes
      - Information learned via base IPv6 router functions
- This new support has been added to OMPROUTE alongside its existing IPv4 dynamic routing support
- You use new sets of IPv6 configuration statements and display commands to activate and monitor this new support

New Configuration Statements

IPv6 Protocol-Neutral Configuration Statements:

- IPv6_Default_Route
- IPv6_Interface

IPv6 RIP Configuration Statements:

- IPv6_RIP_Interface
- IPv6_Accept_RIP_Route
- IPv6_RIP_Filter
- IPv6_Ignore_RIP_Neighbor
- IPv6_Originate_RIP_Default
- IPv6_RIP_Send_Only

New Display RT6TABLE command

Display TCPIP,tcpipjobname,OMPROUTE,RT6TABLE

```
EZZ7979I IPv6 ROUTING TABLE
DESTINATION: ::/0
  NEXT HOP: FE80::7CB6:C5D5:6593:C076
  TYPE: DFLT           COST:  0         AGE: 39
DESTINATION: 4:4:4::/48
  NEXT HOP: FE80::4000:E0E:3634:300
  TYPE:  RIP           COST:  2         AGE: 10
DESTINATION: 7:7:7:7:7:7:7/128
  NEXT HOP: ::
  TYPE:  DIR*          COST:  1         AGE: 39
DESTINATION: 1946::/16
  NEXT HOP: FE80::4000:E0E:3634:300
  TYPE:  RIP           COST:  2         AGE: 10
DESTINATION: 1946::4/128
  NEXT HOP: FE80::4000:E0E:3634:300
  TYPE:  RIP           COST:  2         AGE: 10

DEFAULT GATEWAY IN USE.

TYPE COST     AGE        NEXT HOP
DFLT 0        39         FE80::7CB6:C5D5:6593:C076
                          0 NETS DELETED, 0 NETS INACTIVE
```

New Display IPV6RIP,ALL command

Display TCPIP,tcpipjobname,OMPROUTE,IPV6RIP,ALL

```
EZZ8030I IPv6 RIP CONFIGURATION
   TRACE6: 2, DEBUG6: 3
   STACK AFFINITY: TCPCS3
   IPv6 RIP: ENABLED
   IPv6 RIP DEFAULT ORIGINATION: ALWAYS, COST = 1

   EZZ8027I IPv6 RIP INTERFACES
                                        ---------SEND-----------  --RCV--
   NAME               MTU STATE IN OUT PRF HST STA DEF RADV PSN  PRF HST
   NSQDIO3L6         9000    UP  1   0  NO YES YES YES   NO  NO  YES YES
   LOSAFE3           4000   N/A  1   0 YES  NO YES  NO  YES YES  YES  NO


   EZZ8031I IPv6 RIP ROUTE ACCEPTANCE
   ACCEPT IPv6 RIP UPDATES ALWAYS FOR:
     2001:0DB8::1:9:67:115:66/128
     2001:0DB8:0:A1B::/64

   EZZ8029I GLOBAL IPv6 RIP FILTERS
   SEND ONLY: VIRTUAL, DEFAULT

   IGNORE IPv6 RIP UPDATES FROM:
     FE80::1:2:3:4

   FILTERS: NOSEND     2001:0DB8::1:8:E2:43:28/128
            NORECEIVE  2001:0DB8:0:A1E::/64
```

Trademarks, Copyrights, and Disclaimers

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

IBM	CICS	IMS	MQSeries	Tivoli
IBM(logo)	Cloudscape	Informix	OS/390	WebSphere
e(logo)business	DB2	iSeries	OS/400	xSeries
AIX	DB2 Universal Database	Lotus	pSeries	zSeries

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are registered trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel, ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a registered trademark of Linus Torvalds.

Other company, product and service names may be trademarks or service marks of others.

Product data has been reviewed for accuracy as of the date of initial publication. Product data is subject to change without notice. This document could include technical inaccuracies or typographical errors. IBM may make improvements and/or changes in the product(s) and/or program(s) described herein at any time without notice. Any statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business. Any reference to an IBM Program Product in this document is not intended to state or imply that only that program product may be used. Any functionally equivalent program, that does not infringe IBM's intellectual property rights, may be used instead.

Information is provided "AS IS" without warranty of any kind. THE INFORMATION PROVIDED IN THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IBM EXPRESSLY DISCLAIMS ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IBM shall have no responsibility to update this information. IBM products are warranted, if at all, according to the terms and conditions of the agreements (e.g., IBM Customer Agreement, Statement of Limited Warranty, International Program License Agreement, etc.) under which they are provided. Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. IBM makes no representations or warranties, express or implied, regarding non-IBM products and services.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents or copyrights. Inquiries regarding patent or copyright licenses should be made, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.

Note to U.S. Government Users - Documentation related to restricted rights-Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract and IBM Corp.