36

TCP/SNA

update

December 1999

In this issue

3 Network address translation
19 An SMF termination exit for batch jobs – part 3
32 Clustering and load-balancing
43 A mailbox system for SMTP under MVS TCP/IP – 8
62 March 1997 – December 1999 index
64 TCP/SNA news

TCP/SNA Update

Published by
Xephon
27-35 London Road
Newbury
Berkshire RG14 1JL
England
Telephone: 01635 38342
From USA: 01144 1635 38342
E-mail: trevore@xephon.com

North American office
Xephon/QNA
1301 West Highway 407, Suite 201-405
Lewisville, TX 75077-2150
USA
Telephone: 940 455 7050

Contributions

Articles published in *TCP/SNA Update* are paid for at the rate of £170 ($250) per 1000 words and £90 ($140) per 100 lines of code for original material. To find out more about contributing an article, without any obligation, please contact us at any of the addresses above and we will send you a copy of our *Notes for Contributors*.

***TCP/SNA Update* on-line**

Code from *TCP/SNA Update* can be downloaded from our Web site at http://www.xephon.com/tcpsnaupdate.html; you will need the user-id shown on your address label.

Editor
Trevor Eddolls

Disclaimer

Readers are cautioned that, although the information in this journal is presented in good faith, neither Xephon nor the organizations or individuals that supplied information in this journal give any warranty or make any representations as to the accuracy of the material it contains. Neither Xephon nor the contributing organizations or individuals accept any liability of any kind howsoever arising out of the use of such material. Readers should satisfy themselves as to the correctness and relevance to their circumstances of all advice, information, code, JCL, EXECs, and other contents of this journal before making any use of it.

Subscriptions and back-issues

A year's subscription to *TCP/SNA Update*, comprising four quarterly issues, costs £130.00 in the UK; $190.00 in the USA and Canada; £136.00 in Europe; £142.00 in Australasia and Japan; and £140.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the March 1991 issue, are available separately to subscribers for £33.00 ($48.00) each including postage.

Printed in England.

Network address translation

INTRODUCTION

Ten years ago, corporate networks were dominated by the SNA architecture, which was designed to allow terminals to access mainframes. SNA networks had a simple routing architecture: the routing nodes were identified by subarea numbers, while the devices attached to these nodes had an element number. When the SNA architecture gained widespread acceptance, it became necessary to interconnect networks with overlapping addressing structures (eg identical subarea numbers). IBM extended the architecture by qualifying a network by a NetID and by creating SNI. The NetID allowed SNI-capable SNA routers to direct SNA traffic through an SNI gateway into another SNA network. As the number of SNA networks grew and organizations interconnected and connected to service providers, SNI became a component of many corporate networks. Although SNI was sometimes awkward to set up and configure, it was very powerful. The addressing of the interconnecting SNA networks did not need to change, and identical internal addressing schemes could be used on both sides of the SNI gateway.

Over the past few years, most SNA networks have been replaced by networks based on the Internet Protocol (IP). IP was originally conceived for a single network – the Internet – with a single addressing scheme. In this grand picture, all the devices attached to the network (‘hosts’) would receive a universal and unique address, avoiding the kind of addressing conflicts that used to plague SNA networks.

However, the Internet, now thirty years old, grew far more than had been anticipated. As the number of connected networks grew exponentially, the address space became exhausted and a shortage of official Internet addresses developed. In many new corporate IP networks, ‘unofficial’ addresses were implemented. These addresses were not assigned to the organization by the IANA. With the advent of these unofficial IP addresses, the dream of the single Internet address space without addressing conflicts crumbled. Duplicate addresses appeared, and it became necessary to translate IP addresses between different IP networks, in order to resolve the conflicts. The technique used to implement this is called network address translation. Its basic principles are laid down in RFC 1631.

THE NEED FOR NETWORK ADDRESS TRANSLATION

A first need for network address translation arises when a network using unofficial IP addresses needs to connect to the Internet. Today, most organizations connecting to the Internet are assigned only one or two official 'class C' Internet addresses, which in practice is sufficient to connect only a few tens of hosts to the Internet. In most cases, this number is only a fraction of the number of hosts in the organization's network.

Many corporate IP addresses therefore use unofficial IP addresses in their internal network, in order to have a sufficiently large address space to support all hosts. To avoid overlap of these address spaces with the Internet address space, the IANA defined three ranges of IP address, reserved for use in private networks without coordination with the IANA or any Internet registry (RFC 1918 – see Figure 1). These addresses never appear on the Internet and may only be used internally in IP networks that are not (directly) connected to the Internet. When a network using reserved IP addresses connects to the Internet, the internal addresses need to be translated into the (small) range of official Internet addresses that is assigned to the network.

With this addressing, the reserved IP address ranges are 're-used'

```
 10.0.0.0 to 10.255.255.255 (the 10/8 prefix)
172.16.0.0 to 172.31.255.255 (the 172.16/12 prefix)
192.168.0.0 to 192.168.255.255 (the 192.168/16 prefix)
```

These addresses should never appear on the Internet.
They are intended for use in private networks only.

Figure 1: IANA reserved Internet addresses

many times in private networks that are connected to the Internet. These networks are largely hidden from the Internet, and the IP addresses of their resources are translated into the small official class C network addresses at the boundary between the private network and the Internet. This approach solves the problem of the limited number of assigned official addresses, and also avoids a complete renumbering of the internal IP network when connecting to the Internet.

A second need for address translation arises when two IP networks, each using reserved addresses, need to be interconnected directly. This may occur when networks merge or when organizations connect their network to a service provider that uses reserved addresses. Such an interconnection may create an immediate addressing conflict: identical IP addresses may occur at both sides of the interconnection, making direct communication difficult, if not impossible. Communication between networks using identical IP addresses is made possible only by translating IP addresses in packets flowing between the networks.

For corporate networks, the need for network address translation is growing fast because of the accelerating pace of interconnection of these networks to the Internet and because of the growing need for direct interconnection of IP networks.

GENERAL PRINCIPLES

In an IP network, two addressing components are used in the communication between hosts. The first is the IP address. This is a 32-bit field containing an identifier of the interface of the host to the network. The IP address is structured in a Network ID, used for routing, and a Host ID, identifying the host within each network.

The second component is the UDP or TCP port number. This number identifies a higher-layer application, engaged in communication. The basic idea is that a packet, received at an interface with a certain IP address, is presented to the appropriate host application, indicated by the port number. Application servers on IP networks have pre-defined port numbers. For example, a Web-server usually has port number 80: a Web browser engaging in a communication with a Web server should direct its requests to port 80 on the Web server host. It is common practice that server applications (WWW, Telnet servers, FTP servers, mail servers, applications) use port numbers below 1024. Most of these port numbers are pre-defined, to allow clients to start the communication. Clients (Web browsers, news and mail readers, Telnet users, FTP clients, etc) commonly choose a port number beyond 1024 when they initiate communication with a server.

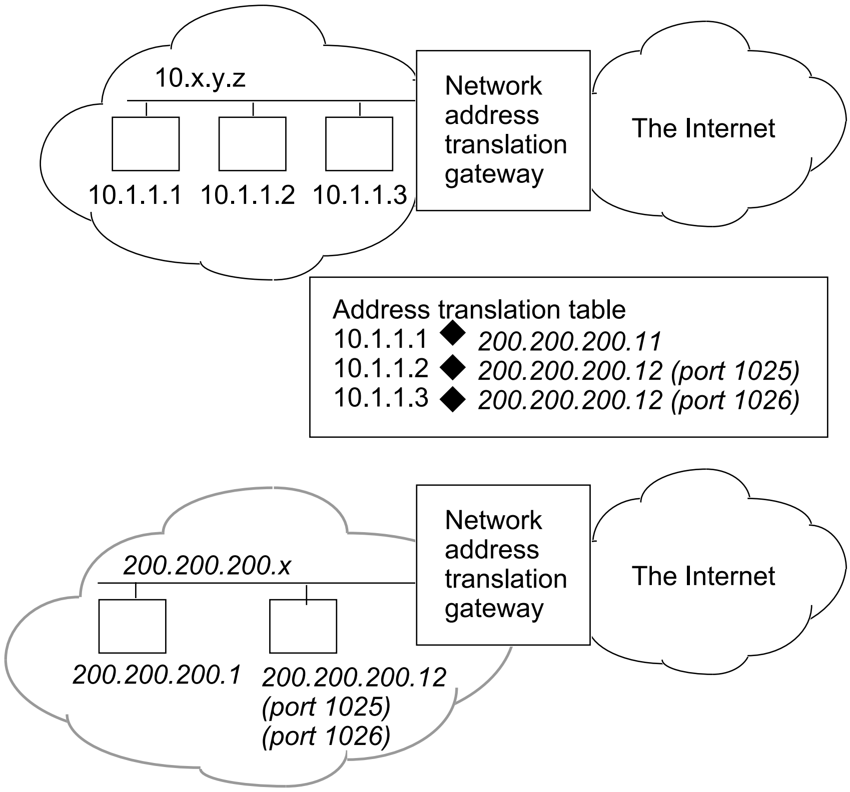
Network address translation basically converts the addressing of one IP network (the 'internal' network) into a virtual subnet in another network (the 'external' network). A user in the external network 'sees' the users in the internal network (of which the addresses are translated) as if they were located in a subnet of his/her own IP network, respecting the addressing scheme of the external network (see Figure 2). In this architecture, the qualification 'internal' indicates only that the IP addresses of this network are to be translated. The term 'external' then refers to the other network.

Network address translation is carried out by a gateway, to which both the internal and the external network are attached. The gateway inspects all IP packets passing between the networks, and compares the source and destination addresses and ports to the pre-defined network address translation rules. The gateway changes the necessary addresses and ports in the IP packet, reconstructs it, and delivers it to the other network.

There are two techniques for achieving this:

- *IP address mapping*. In this technique, the IP address in the internal network is mapped onto an IP address in the external network. This is the straightforward translation of addresses. Viewed from the external network, a host simply acquires a different IP address. Port numbers are not translated and remain the same (although it is in theory possible to translate the port numbers too).

  IP address mapping can be static or dynamic. In static mapping, one-to-one relations between IP addresses in both networks are pre-defined and remain fixed. In dynamic mapping, the network address translation gateway assigns translated addresses, chosen



The network to the left (the 'internal' network) uses reserved IP addresses (10.x.y.z) that must be translated into a small range of official Internet addresses (200.200.200.x). Addresses on the Internet (the 'external' network) are not translated. The upper view gives the local addressing. The lower half gives the view of the network from the Internet: a virtual subnet (grey lines), using official Internet addresses. The corresponding address translation table is shown in the middle. Both IP address mapping (first entry) and IP port mapping (last two entries) are shown. Italics indicate virtual IP addresses, generated by the network address translation gateway.

Figure 2: Single address translation

from a (pre-defined) pool of allowed addresses. Static mapping has the advantage that the addresses and their translations are known in advance. Its disadvantage is that the number of addresses (and translation definitions) needed equals the number of potential internal hosts that will communicate through the gateway. So, if 1,000 clients want to access the Internet, 1,000 entries in the translation gateway and 1,000 official Internet addresses are needed, which is practically impossible. Dynamic address mapping solves this problem by assigning official IP addresses on-the-fly, at the cost of losing the one-to-one relation between internal IP addresses and their translations.

IP address mapping is used mostly for translating server addresses between networks. In the external network, these servers are then known under their (permanent) translated address. The services on the host are accessible by using the same port numbers. Static IP address mapping may sometimes be unavoidable for some clients. This may occur when clients are to be identified explicitly in the external network, for example because a client is registered in an application by its IP address.

- *IP port mapping* (also called port address translation). In this technique, IP addresses in the internal network are converted into a single IP address in the external network, but each on a different UDP or TCP port number.

  IP port mapping is commonly used for translating client IP addresses to external networks. Viewed from the external network, all clients appear to be concentrated in a single ‘client host’ (ie a single IP address), but with different port numbers. This is completely natural, because clients are free to choose their port number when communicating on an IP network. Also, it is not unusual to have several clients at a single IP address (eg multiple users browsing or transferring files on a Unix system).

  IP port mapping is normally dynamic – the address translation gateway assigns the ports as connections are requested.

Network address translation is used in two basic architectures – single address translation, where IP addresses are translated in one direction, and double address translation, where IP addresses are translated in both directions (see Figure 3). These architectures are described below:

What happens to the:

	Source address	*Source port*	*Destination address*	*Destination port*
Single address translation				
IP address mapping				
Outgoing [1]	Translated	Unchanged	Unchanged	Unchanged
Incoming	Unchanged	Unchanged	Translated	Unchanged
IP port mapping				
Outgoing [1]	Translated	Changed	Unchanged	Unchanged
Incoming	Unchanged	Unchanged	Translated	Changed
Double address translation				
IP address mapping				
Either way	Translated	Unchanged	Translated	Unchanged
IP port mapping				
Outgoing [1]	Translated	Changed	Translated	Unchanged
Incoming	Translated	Unchanged	Translated	Changed

[1] From the internal network to the external network

Figure 3: Different forms of network address

- *Single address translation*. In this architecture, the private (internal) network normally uses reserved IP addresses that should not appear on the (external) Internet (see Figure 3). From the Internet, the private network looks like a subnet, using a small range of official Internet addresses. The conversion is done by the network address translation function, situated between the Internet and the private network. Seen from the private network, the Internet looks exactly the same (ie Internet addresses are not translated).

  This network address translation architecture is the common solution for connecting a private network to the Internet. It allows large private networks to connect to the Internet while using only a small range of official IP addresses. IP address translation is used for servers in the private network, which must be accessible from the Internet. IP port translation is used for clients from the private network, accessing resources on the Internet. The total number of required official IP addresses is equal to the sum of the number of Internet-accessible servers plus one IP address for all outgoing clients (plus a few for interfaces and the router on the Internet access link). A single official class C Internet network address is in most cases sufficient to support the connection.

In this architecture, no addressing conflicts occur, provided that reserved IP addresses are used in the private network. In this case, routers in the private network can discern traffic destined for the Internet from internal traffic: if the address is outside the reserved address range, used internally, the packet is destined for the Internet and is routed to the gateway. If the private network uses unofficial IP addresses outside the reserved range, the same address range on the Internet is inaccessible to users in the private network. For example, if an organization uses the IP class A network 20/8 for internal use, the 20/8 address range on the Internet (ie the domain csc.com) is inaccessible from within the private network. Indeed, if a host in the private network sends an IP packet to an address 20.x.y.z, the internal routers will assume that the destination of the packet is inside the private network, and it will never be forwarded to the Internet. This is why organizations should use reserved IP addresses in their private networks, and not implement a randomly chosen range of IP addresses, even if they have no immediate intention to connect to the Internet.

- *Double address translation*. Double address translation is used to interconnect two private networks that may be using overlapping addresses (see Figure 4).

  IP addresses of both networks are translated. The upper panel shows the local addresses, used in each of the networks. The middle panel shows the network configuration, as viewed from the 172.16 network (the network to the right). In this view, the 10.1 network is the 'internal' network and the 172.16 network is the 'external' network. The lower panel shows the network configuration, as viewed from the 10. network (the network to the

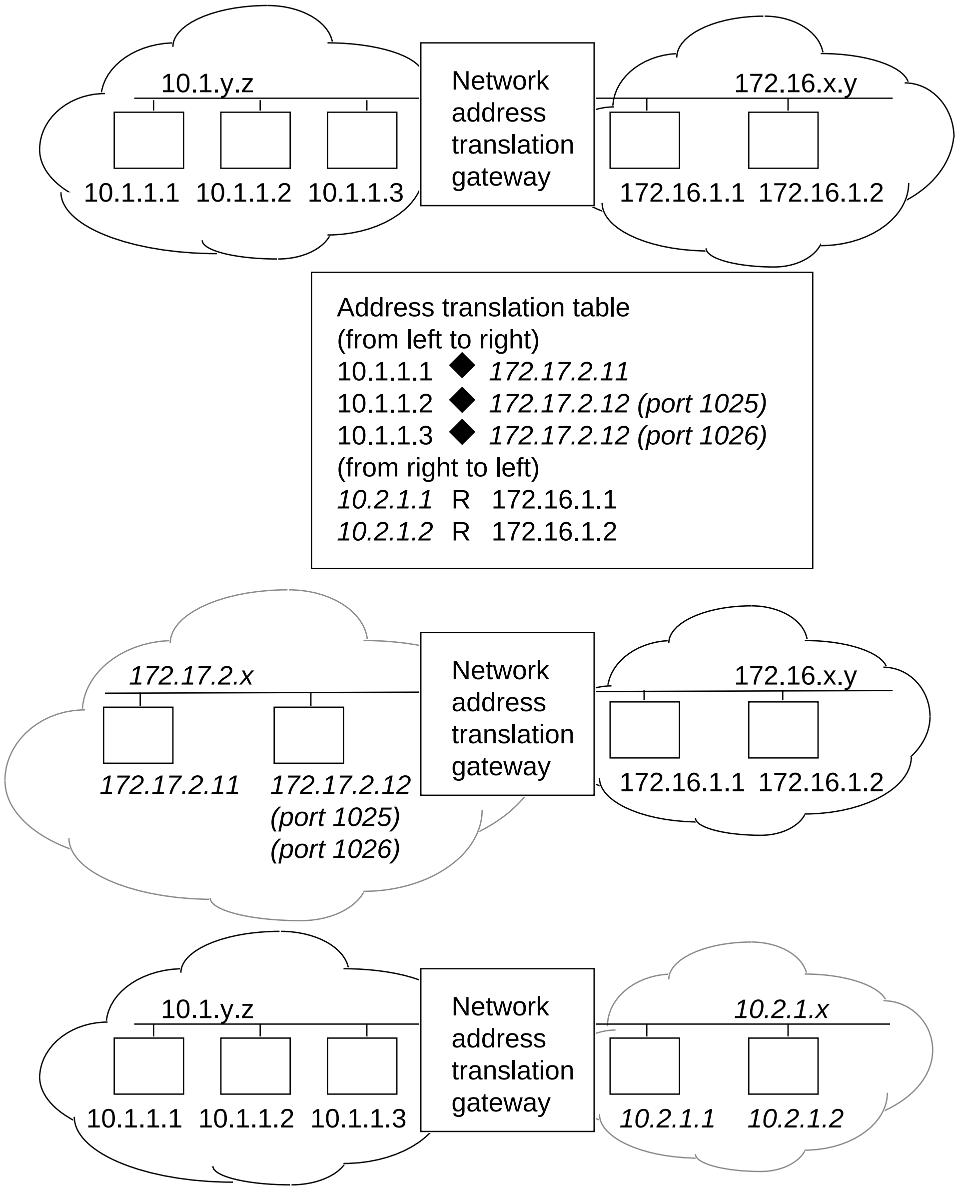


Figure 4: Double address translation

left). In this view, the 10.1 network is the ‘external’ network and the 172.16 network is the ‘internal’ network. Each of the networks is viewed from the other network as a virtual subnet (grey lines). Italics indicate virtual IP addresses, generated by the network address translation gateway.

In this architecture, the network address translation is symmetric

– addresses in both networks are translated when communication crosses the border between the two networks.

In each of the networks, a virtual representation of the other network is created. Each of the networks has the role of internal and of external network. Both IP address translation and IP port translation can be used in double address translation.

Double address translation may be necessary when two networks that use reserved IP addresses interconnect. In this case, both networks may (partly) use the same address range, and identical addresses may be assigned to hosts in both networks. Double address translation is the only way to solve the problem of overlapping address spaces in directly connected networks (ie by using a single gateway).

Double address translation is not a good solution for connecting a private network to the Internet – the Internet address space is vast, and cannot be compressed and translated into a limited virtual subnet of the private network. In practice, this means that you should not try to translate the addresses of hosts on the Internet into private IP addresses. Double address translation is therefore not appropriate for resolving address conflicts created by implementing unofficial IP addresses outside the reserved range. Instead, such private networks should be renumbered, implementing reserved IP addresses.

IMPLEMENTATION

Network address translation is more complicated than a simple address substitution in IP packets. Several higher-layer protocols, such as FTP, refer to IP addresses in their own headers, so the address translation gateway must understand and monitor these protocols and adjust any references to IP addresses and/or port numbers. Many Internet Control Messages Packets (ICMP) also contain IP addresses. Again, the gateway must understand these messages and change the IP addresses as needed.

When dynamic address translation is used, the gateway has to be aware of the connection between the client and the server. For example, when a client from a private network accesses a server on the Internet, the network address translating gateway maps the (private) IP address of the client onto a fixed (official) IP address and a port number chosen by the gateway from a pool of port numbers. The gateway needs to keep track of this relationship for the duration of the communication, in order to be able to deliver incoming packets (containing the port number, assigned by the gateway) to the correct IP address in the private network. The same is true for dynamic IP address translation, because only the address translation gateway knows the relationship between the original and the translated IP address.

Network address translation is implemented in routers or in firewall products. Both these implementations have advantages and disadvantages:

- In a router, network address translation is configured by setting up simple translation rules using the router command sequences. This set-up is not very user friendly, and may be hard to test. On the other hand, routers are inexpensive and efficient packet manipulators, so they are a cost-effective way to implement address translation.

  Routers do not always support all possible address translation architectures and techniques. There are large differences in implementation between different router vendors and between different models from one vendor. Implementations vary from the absence of address translation to sophisticated translation features, including double address translation and the full interpretation of higher-level protocols and control messages.

- In firewalls, network address translation is combined with protection against intruders. This means that firewalls offer a highly-secure and managed environment for address translation. Several products offer user-friendly and comprehensive set-up of the translation rules, and feature extensive monitoring, logging, and reporting. However, firewalls are expensive, and require a continuous management effort.

Some routers and firewall products can support simultaneous address translation between more than two networks. The actual number depends on the product and the number of network interfaces available on the hardware, but may amount to eight or even sixteen.

For simple network address translation problems – for example a simple static translation of server addresses between two parts of a private network – a router may be sufficient. A router may also be appropriate when connecting to a trusted network with an overlapping address scheme (eg the network of a services provider). For more complicated address translation problems, especially when security issues are also involved, the use of firewall software may be necessary.

Organizations connecting to the Internet need a firewall anyway (for reasons of security), so they should preferably implement the network address translation on this firewall. The network address translation capabilities of firewall products varies tremendously: some products offer excellent address translation capabilities, others have no address translation at all. Vendors of firewall products tend to emphasize the security features of their products and not the address translation features, so a detailed study of their offerings may be necessary when implementing an address translation firewall.

IMPLEMENTATION ISSUES

Implementing network address translation is not a simple matter, so a proper architecture and careful planning and documentation are absolutely essential. The architecture is in many cases complex, and the implementation details (eg router command sequences) are often obscure. Before the address translation is implemented, all hosts that need address translation should be identified. This includes servers that should be 'visible' from the external network and clients that need to access this network. Then, a translation technique should be chosen for each host. This will probably be IP address mapping for the servers and port mapping for the clients, although address mapping should be used for clients that are to be known explicitly by their IP address in the external network. Only then should the virtual 'translated' networks be defined: one virtual network for single address translation, and two virtual networks for double address translation. The final step is then to derive the address translation rules from the architecture.

The address translation implementation should be properly documented. When networking engineers investigate operational problems in networks with address translation, both the original and the translated IP addresses of hosts may appear. As IP addresses in packets change during their journey from source to destination, it is difficult to trace traffic and to isolate errors without proper documentation of the address translation implementation.

An important issue is the proper separation of the address spaces of both networks. To be good, the address translating gateway should be able to qualify each IP address with the network it belongs to, or at least with the appropriate network interface. Some address translating products do not handle this properly: packets that arrive at a gateway network interface are deposited in a central routing queue, losing vital information on their origin. This may create severe problems when duplicate addresses are present in interconnected networks (and double address translation is necessary). The central routing engine may not be able to determine the correct source and destination network of a packet, resulting in random routing behaviour. If the network address translation gateway cannot handle overlapping address spaces, this kind of addressing conflict can be solved only by implementing an intermediate network with unique addresses and two address translations in cascade. This requires two routers, or one router and a firewall.

Domain Name Service (DNS) relates host names to IP addresses. Because the IP addresses of external resources are different when translated, a DNS name resolution of a host name in a network with translated addresses obviously does not yield the correct IP address. There are two possible solutions to the DNS problem (see Figure 5):

- DNS name resolution for hosts in an internal ('virtual') network (with translated addresses) is implemented in a local DNS server in the external network. This DNS server knows all the (translated) IP addresses of the relevant resources in the internal network (mostly servers). This is consistent with the view of the internal network as a 'virtual subnet' of the external network. However,

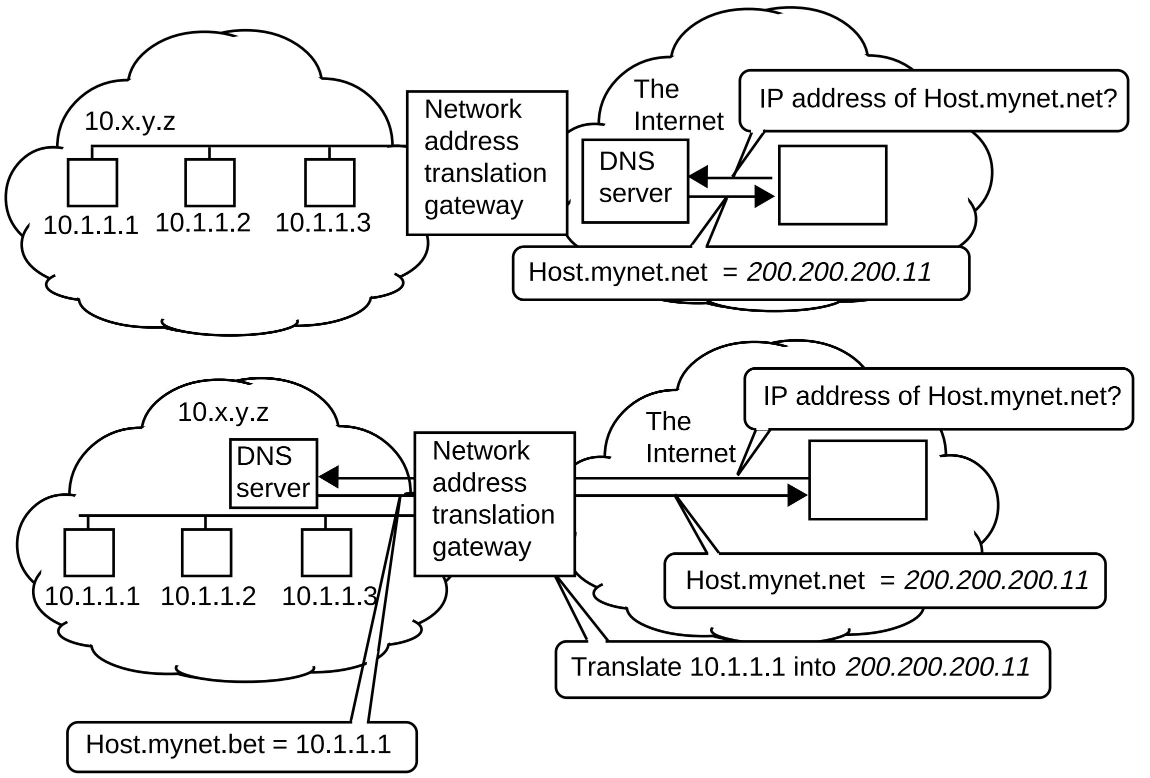


Figure 5: The DNS service, in combination with network address translation

it requires that the DNS server be synchronized with the address translation gateway: both have to use the same IP address for the same virtual resource. This is not a particular problem if static IP address translation is used and translated IP addresses are persistent. However, it is a problem when dynamic IP address translation is used for resources that need DNS name resolution.

- The DNS server is in the internal network and the address translation gateway converts the IP addresses into DNS replies. From the architectural viewpoint, this is the ‘cleanest’ solution – each network has its own DNS server and replies to DNS requests with local IP addresses. However, the network address translation gateway must be able to recognize and convert the IP addresses of hosts in the internal network, contained in the DNS replies. The situation is even more complicated if dynamic IP address translation is implemented. In this case, the gateway has to trace the further usage of the (dynamically assigned) IP address after inserting it into the DNS reply. This requires careful management of the IP traffic and the timers used to re-assign idle addresses.

Two solutions are shown in Figure 5 – address resolution by a local server (top) and by a server in the 'internal' network. The configuration is identical to that in Figure 1. In this example, a host in the external network queries the DNS server for the IP address of the host Host.mynet.be.

When translating IP addresses or port numbers dynamically, the gateway assigns addresses and/or port numbers on a temporary basis from a pool of addresses and/or port numbers. This may inadvertently expose traffic, creating operational problems and even security exposures. When a connection is considered to be terminated (for instance, after a certain idle time), the IP address and/or port number may be reassigned to another connection, ie between two other hosts. After the re-assignment, one of the hosts may send unexpected IP packets to the gateway, destined for the 'old' destination. The address translation gateway may forward these to the host now assigned to the IP address and/or port number – and thus to an unwanted destination.

Traffic exposure may occur when hosts in the external network cache virtual IP addresses they retain from a previous connection with a host in the internal network. When a new communication is set up, the host may directly use the destination IP address (which it thinks to be still existing), although it does not exist any more. Some address translation gateways (especially the firewall implementations) can effectively handle this problem. They will monitor all conversations between hosts, and will drop any IP packet that does not fit into a conversation. When a stray IP packet arrives, it will be removed from the traffic stream.

There is an obvious conflict between network address translation and encryption. If the payload of an IP packet is encrypted, the address translation gateway cannot adjust any references to the IP addresses contained in the payload. IP packets should therefore be decrypted before address translation is done. This may introduce a security exposure in networks carrying highly-confidential traffic.

Another issue is the definition of the routes to the internal network. Usually, static routes are defined from the external network through the gateway into the ‘virtual’ network. Route advertisement into the external network should be carefully managed, to avoid routes being advertised, based on addresses that are not appropriately translated.

CONCLUSION

Network address translation is emerging as an important component of enterprise networks. In many enterprise networks, an address translation gateway is becoming a vital component of the network, implementing critical gateways to the Internet or to partner- or provider-networks. Although the basic architecture behind network address translation is relatively simple, the implementation can be difficult because of the many subtle technical details related to address assignments, DNS, and routing issues.

The currently available address translation products are often limited or expensive to implement and manage. However, in the future, router, LAN switch, and firewall products should offer more complete and better architected network address translation features.

Claude Doom
CSC Computer Sciences (UK)

An SMF termination exit for batch jobs – part 3

This month we conclude the article looking at an SMF termination exit.

```
MNOJCTMG WTO   'ISSDØ36A JOB CONTROL INFORMATION UNAVAILABLE FOR R88JOHN
               NG',DESC=2,ROUTCDE=(1,15),MF=L
MLNGTHNJ EQU   *-MNOJCTMG
         SPACE 2
MONITORL WTO   'ISSDØ36I R88JOHNG PROCSTEP STEPNAME NO. PROGNAME - ABEND
               DED SXXX',DESC=6,ROUTCDE=15,MF=L
MLNGTHMN EQU   *-MONITORL
         SPACE 2
JCPUINFO DC    CL36'TCB CRU SECONDS.....................'
         DC    XL1Ø'4Ø2Ø2Ø6B2Ø2Ø214B2Ø2Ø'
         DC    CL2' '
         DC    CL18'COST..............'
         DC    XL1Ø'4Ø2Ø2Ø6B2Ø2Ø214B2Ø2Ø'  COST MASK
         SPACE 2
JXCPINFO DC    CL36'PROCESS EXCPS.......................'
         DC    XL1Ø'4Ø2Ø6B2Ø2Ø2Ø6B2Ø212Ø'
         DC    CL2' '
         DC    CL18'COST..............'
         DC    XL1Ø'4Ø2Ø2Ø6B2Ø2Ø214B2Ø2Ø'  COST MASK
         SPACE 2
JCRUINFO DC    CL36'ESTIMATED COST OF COMPUTER RESOURCES'
         DC    CL1Ø' CONSUMED '
         DC    CL2Ø'BY THIS JOB.......'
         DC    XL1Ø'4Ø2Ø2Ø6B2Ø2Ø214B2Ø2Ø'  COST MASK
         EJECT
JIPTINFO DC    CL36'TOTAL CARDS READ....................'
         DC    XL1Ø'4Ø2Ø6B2Ø2Ø2Ø6B2Ø212Ø'
         DC    CL2' '
         DC    CL18'COST..............'
         DC    XL1Ø'4Ø2Ø2Ø6B2Ø2Ø214B2Ø2Ø'  COST MASK
         SPACE 2
JPUNINFO DC    CL36'TOTAL CARDS GENERATED...............'
         DC    XL1Ø'4Ø2Ø6B2Ø2Ø2Ø6B2Ø212Ø'
         DC    CL2' '
         DC    CL18'COST IF PUNCHED...'
         DC    XL1Ø'4Ø2Ø2Ø6B2Ø2Ø214B2Ø2Ø'  COST MASK
         SPACE 2
JPRTINFO DC    CL36'TOTAL LINES GENERATED...............'
         DC    XL1Ø'4Ø2Ø6B2Ø2Ø2Ø6B2Ø212Ø'
         DC    CL2' '
         DC    CL18'COST IF PRINTED...'
         DC    XL1Ø'4Ø2Ø2Ø6B2Ø2Ø214B2Ø2Ø'  COST MASK
```



```
         SPACE 2
JTAPETPR DC    CL36'TOTAL SPECIFIC TAPE MOUNTS..........'
         DC    XL1Ø'4Ø2Ø6B2Ø2Ø2Ø6B2Ø212Ø'
         DC    CL2' '
         DC    CL18'COST..............'
         DC    XL1Ø'4Ø2Ø2Ø6B2Ø2Ø214B2Ø2Ø'  COST MASK
         SPACE 2
JTAPEPTM DC    CL36'TOTAL NON-SPECIFIC TAPE MOUNTS......'
         DC    XL1Ø'4Ø2Ø6B2Ø2Ø2Ø6B2Ø212Ø'
         DC    CL2' '
         DC    CL18'COST..............'
         DC    XL1Ø'4Ø2Ø2Ø6B2Ø2Ø214B2Ø2Ø'  COST MASK
         SPACE 2
JTAPEUSE DC    CL36'TOTAL NUMBER OF TAPE UNITS USED.....'
         DC    XL1Ø'4Ø2Ø6B2Ø2Ø2Ø6B2Ø212Ø'
         DC    CL2' '
         DC    CL18'COST..............'
         DC    XL1Ø'4Ø2Ø2Ø6B2Ø2Ø214B2Ø2Ø'  COST MASK
         EJECT
ISSØØ5I  DS    ØCL7Ø
         DC   C'ISSØØ5I PROGRAM NAME SPECIFIED IS '
         DC   C'RESTRICTED TO ISSD INTERNAL USE ONLY'
         SPACE 1
INFOJOB  DS    ØCL76
         DC    CL4'JOB '       JOB
         DC    CL5' '          HASP NUMBER
         DC    CL8' '
         DC    CL9' ACCOUNT '
         DC    CL8' '
         DC    CL7' CPUID '
         DC    CL4' '
         DC    CL6' DATE '
         DC    XL7'4Ø212Ø4B2Ø2Ø2Ø'
         DC    CL6' TIME '
         DC    XL1Ø'4Ø212Ø2Ø7A2Ø2Ø7A2Ø2Ø'
         DC    CL2' '
         EJECT
********************
*  MISCELLANEOUS   *
********************
         SPACE 1
ATABLE   DC    5CL2'  '
         ORG   ATABLE
         DC    CL2'JJ'      EXCLUDE DOT JOBS
         ORG
ATABENT  EQU   (*-ATABLE)/2
         SPACE 1
         DS    ØH
RTABLE   DC    6XL2'FFFF'
         ORG   RTABLE
```

```
*                   DOT RMTS 7-11
         DC    H'Ø7',H'Ø8',H'Ø9',XL2'ØØØA',XL2'ØØØB'
         ORG
RTABENT  EQU   (*-RTABLE)/2
         SPACE 2
RESTABLE EQU   *-193
         DC    X'Ø1Ø2Ø3Ø4Ø5Ø6Ø7Ø8Ø9'
         DC    XL7'ØØ'
         DC    X'ØAØBØCØDØEØF1Ø1112'
         DC    XL8'ØØ'
         DC    X'131415161718191A'
CMRTRANS EQU   *-24Ø
         DC    C'Ø123456789ABCDEF'
         EJECT
********************
*  DEVICE TABLES   *
********************
         SPACE 1
DASDTABL DC    X'ØF',CL4'339Ø'
CMRUSIZE EQU   *-DASDTABL
         DC    X'ØE',CL4'338Ø'
         DC    X'ØB',CL4'335Ø'
         DC    X'Ø7',CL4'23Ø5'
         DC    X'Ø9',CL4'333Ø'
         DC    X'ØD',CL4'3331'
         DC    X'ØC',CL4'3375'
         DC    X'ØA',CL4'334Ø'
DASDCNT  EQU   (*-DASDTABL)/5
         SPACE 2
TAPETABL DC    AL1(UCB349Ø),CL4'349Ø'
         DC    AL1(UCB348Ø),CL4'348Ø'
         DC    AL1(UCB34ØØ),CL4'342Ø'
TAPECNT  EQU   (*-TAPETABL)/5
         SPACE 2
         LTORG
         TITLE 'HASP CONTROL BLOCKS'
***********************************************************************
*        GENERATE HASP CONTROL BLOCKS                                 *
***********************************************************************
        SPACE
        PRINT NOGEN
        SPACE
        $HASPEQU
        SPACE
        $BUFFER
        SPACE
        $CAT
        SPACE
        $TQE
        SPACE
```



```
        $JCT
        SPACE
        $SJB
        SPACE
        $SCAT
        SPACE
        $MIT
        SPACE
        $XECB
         TITLE 'OS CONTROL BLOCKS'
***********************************************************************
*                                                                     *
*        GENERATE OS CONTROL BLOCKS                                   *
*                                                                     *
***********************************************************************
         SPACE
         CVT   DSECT=YES
         SPACE 1
         IEZDEB
         SPACE 1
         IEFUCBOB PREFIX=YES
         SPACE 1
         IEFJSSIB
         SPACE 1
         IEZJSCB
         SPACE 1
         IHAPSA
         SPACE 1
         IFGRPL
         SPACE 1
         IKJTCB
         SPACE 1
         IEFASIOT
         SPACE 1
TIOT     DSECT
         IEFTIOT1
         SPACE
SCTDSECT DSECT
         IEFASCTB
         SPACE
         IEFJMR
         SPACE
         IECDIOCM
         SPACE 1
         IFASMFR 3Ø
         TITLE 'OS INITIATOR/TERMINATOR LINKAGE CONTROL TABLE'
***********************************************************************
*        OS LINKAGE CONTROL TABLE.  THIS TABLE IS POINTED AT BY       *
*        R12 AT ENTRY FROM THE OS INITIATOR.  THIS IS NOT DOCU-       *
*        MENTED AS SUCH, AND THE ASSUMPTION THAT R12 WILL ALWAYS      *
```



```
*        POINT TO THE LCT MAY NOT BE VALID.  IT IS POSSIBLE          *
*        IN FUTURE RELEASES THAT THE LCT MAY NOT BE AVAILABLE OR     *
*        WILL BE POINTED AT BY SOME OTHER REGISTER.                  *
**********************************************************************
         SPACE 5
LCTDSECT DSECT
         SPACE 1
         IEFALLCT
         TITLE 'SMF EXIT DATA TABLE DSECT'
**********************************************************************
*                                                                    *
*        THIS TABLE IS POINTED AT BY R1 AT ENTRY TO IEFACTRT.        *
*        EACH ENTRY IN THE TABLE IS A POINTER TO THE RESPECTIVE      *
*        ITEM.  THE ENTRY EXDRDW POINTS TO THE RECORD DESCRIPTOR     *
*        WORD OF THE SMF RECORD, WHICH IS OFFSET -4 FROM THE         *
*        RECORD ITSELF.                                              *
*                                                                    *
**********************************************************************
         SPACE 5
EXDDSECT DSECT
         SPACE 2
EXDCOMTB DS    A                   COMMON EXIT TABLE
EXDSTPNM DS    A                   STEP NAME
EXDPGRNM DS    A                   PROGRAMMER NAME
EXDJRT   DS    A                   JOB RUN TIME
EXDJAD   DS    A                   JOB ACCOUNTING DATA
EXDSRT   DS    A                   STEP RUN TIME
EXDSAD   DS    A                   STEP ACCOUNTING DATA
EXDFLAGS DS    A                   FLAGS AND STEP NUMBER
EXDCOMPL DS    A                   COMPLETION CODE
EXDRDW   DS    A                   SMF RECORD DESCRIPTOR WORD
         TITLE 'DSECTS FOR VOLATILE STORAGE'
**********************************************************************
*        THE FOLLOWING DSECT DESCRIBES STORAGE WHICH IS OBTAINED     *
*        FOR EACH ENTRY OF IEFACTRT.  ALL OF THE WORK AREAS ARE      *
*        INITIALIZED TO ZERO (BINARY).                               *
**********************************************************************
         SPACE 2
WORKAREA DSECT
         SPACE 2
         DS    61F                 REG SAVE AREA FOR CALLED RTNS
CLAMLOVE DS    F                   LOCATOR
CLAMSTEP DS    F                   STEP SMF TYPE 30 RECORD
CLAMTYPE DS    C                   LAST SMF TYPE 30 RECORD
CLAMJOB  DS    AL3                 JOB SMF TYPE 30 RECORD - THIS FIELD
*                                  SHOULD BE EXPANDED TO ACCOMMODATE
*                                  31-BIT ADDRESSES.
         SPACE 2
SAVE1    DS    46F                 SAVE AND WORK AREAS
SAVELAST DS    F                   ADDRESS OF SAVE AREA ABOVE US
```



```
TERMTIME DS    F
TERMDATE DS    F
CLAMWORK DS    2D
CLAMHOLD DS    CL1Ø
MSGLEN   DC    AL2(L'MSG)
MSGADDR  DC    A(MSG)
TEMPD1   DS    D
         ORG   TEMPD1              SET UP FIELDS FOR DEVICE PROCESSING
TMPDEVC  DS    B
TMPDEVT  DS    B
TMPDEVAD DS    H
TMPCOUNT DS    F
TEMPD2   DS    D
DOUBLE   DS    D
WORKTIME DS    F
WORKDATE DS    PL4
RUNTIME  DS    F
ADDRLCT  DS    A                   HOLDS ADDRESS OF LCT
ADDREXD  DS    A                   HOLDS ADDRESS OF EXD
MSG      DS    CL8Ø                BUFFER FOR PRINTING MESSAGES
*                                  THIS FIELD IS REDEFINED AS FOLLOWS:
         EJECT
***********************************************************************
*        DEFINITIONS OF OVERLAYS WITHIN MSG BUFFER                    *
***********************************************************************
         SPACE
         ORG   MSG+1
BLANK1   DC    C' '
BLANK2   DC    CL(L'MSG-3)' '
         SPACE 1
*        DEFINE JOB TITLE LINE
         SPACE 1
         ORG   MSG+2
HGJOBNAM DC    CL8'JOB NAME'
         DC    CL1' '
HGPRIO   DC    CL1' '
         DC    CL1' '
HGCLASS  DC    CL1' '
         DC    CL2' '
HGJOBNO  DC    CL4'JOB#'
         DC    CL2' '
HGPD     DC    CL4'P/D'
         DC    CL2' '
HGSYSTEM DC    CL6'SYSTEM'
HGSYSID  EQU   *-5,4
         DC    CL2' '
HGACOUNT DC    CL17'BILLING CODE'
         DC    CL2' '
HGPGMR   DC    CL23'PROGRAMMER''S NAME FIELD'
         SPACE 2
```

```
*        DEFINE JOB TIME LINE
         SPACE 1
         ORG   MSG+2
         DC    C' '
HJSSTART DC    CL2Ø'     JOB START    '
HJSDATE  EQU   HJSSTART-1,1Ø
HJSTIME  EQU   HJSSTART+1Ø,11
         DC    CL6' '
HJEND    DC    CL2Ø'       JOB END     '
HJEDATE  EQU   HJEND-1,1Ø
HJETIME  EQU   HJEND+1Ø,11
         DC    CL4' '
HJELAPSD DC    CL18' JOB ELAPSED TIME '
HJEPTIME EQU   *-14,11
         EJECT
*        DEFINE STEP TITLE LINE
         SPACE 1
         ORG   MSG+2
HJOBNAME DC    CL8'  JOB'
         DC    CL2' '
HSTEPNAM DC    CL8'  STEP  '
         DC    CL2' '
HSTEPNUM DC    CL3'NUM'
         DC    CL2' '
HPGMNAME DC    CL8'PGM NAME'
         DC    CL3' '
HSTART   DC    CL18'    STEP START    '
HSDATE   EQU   HSTART-1,1Ø
HSTIME   EQU   HSTART+1Ø,8
         DC    CL4' '
HEND     DC    CL18'     STEP END     '
HEDATE   EQU   HEND-1,1Ø
HETIME   EQU   HEND+1Ø,8
         SPACE 1
*        DEFINE TASK TIME HEADER
         SPACE 2
         ORG   MSG+2
HELAPSED DC    CL13'ELAPSED TIME:'
         DC    CL1' '
HTIMELAP DC    CL11' '
         DC    CL2' '
HCPUTIME DC    CL15'CPU TIME: TCB ='
         DC    CL1' '
HTIMECPU DS    CL11' '
HSRBTIME DS    CL1Ø'     SRB ='
         DC    CL1' '
HTIMESRB DS    CL11' '
         SPACE 2
*        DEFINE SERVICE UNITS HEADER
         SPACE 1
```

```
         ORG   MSG+2
HSERVICE DS    CL2Ø'SERVICE UNITS: CPU ='
HSCPU    EQU   *-2,11
         DS    CL11' '
HSRBSERV DS    CL5'SRB ='
HSSRB    EQU   *-2,11
         DS    CL11' '
HIOSERV  DS    CL5'I/O ='
HSIO     EQU   *-2,11
         DS    CL11' '
HMSOSERV DS    CL5'MSO ='
HSMSO    EQU   *-3,11
         EJECT
*        DEFINE PAGING HEADER
         SPACE 2
         ORG   MSG+2
HCPI     DS    CL2'PI'
HPPI     EQU   *,7
         DS    CL9' '
HCPO     DS    CL2'PO'
HPPO     EQU   *,7
         DS    CL9' '
HCPR     DS    CL2'  '                 DROP PR RECLAIMS
HPPR     EQU   *,7
         DS    CL9' '
         DS    CL5' '
HCVI     DS    CL2'VI'
HPVI     EQU   *,7
         DS    CL9' '
HCVO     DS    CL2'VO'
HPVO     EQU   *,7
         DS    CL9' '
HCVR     DS    CL2'VR'
HPVR     EQU   *,7
         SPACE 1
*        DEFINE COMMON PAGING HEADER
         SPACE 2
         ORG   MSG+2
HCPAGE   DS    CL12'CSA: PAGE-IN'
HCCIN    EQU   *,7
         DS    CL9' '
HCRECLAM DS    CL8'HYPER-PI'
HCCRCLAM EQU   *,7
         DS    CL1Ø' '
HLPAIN   DS    CL12'LPA: PAGE-IN'
HCLIN    EQU   *,7
         DS    CL9' '
HLRECLAM DS    CL8'HYPER-PO'
HCLRCLAM EQU   *,7
         SPACE 2
```

```
*        DEFINE SWAPPING HEADER
         SPACE 1
         ORG   MSG+2
HSWAPING DS    CL19'SWAPPING: SEQUENCES'
HSSS     EQU   *-2,7
         DS    CL7' '
HCSIN    DS    CL3'IN'
HSSIN    EQU   *-2,7
         DS    CL7' '
HCSOUT   DS    CL4'OUT'
HSSOUT   EQU   *-2,7
         DS    CL11' '
HCSTOLEN DS    CL13'PAGES STOLEN:'
HSSTOLEN DS    CL7' '
         EJECT
*        DEFINE STORAGE ALLOCATION HEADERS
         SPACE 1
         ORG   MSG+2
HREGION  DS    ØCL77
HRTCONA  DS    CL24'REGION(VIRT) SIZE      :'
HRTYPE2  EQU   HREGION+7,4
         DS    CL3' '
HRREQ2   DS    CL7' '
HCK      DS    CL2'K'
HRLOC2   DS    CL25' '
         SPACE 2
*        DEFINE I/O SECTION HEADER
         SPACE 1
         ORG   MSG+2
HIOSEC   DS    ØCL77
HIODDNAM DS    CL8'DDNAME'
         DS    CL1' '
HIOUNIT  DS    CL4'UNIT'
         DS    CL1' '
HIOADDR  DS    CL4'ADDR'
         DS    CL1' '
HIOBLKSZ DS    CL6'BLKSIZ'
         DS    CL1' '
HIOEXCP  DS    CL11'-- EXCPS --'
         DS    CL2' '
H2ODDNAM DS    CL8'DDNAME'
         DS    CL1' '
H2OUNIT  DS    CL4'UNIT'
         DS    CL1' '
H2OADDR  DS    CL4'ADDR'
         DS    CL1' '
H2OBLKSZ DS    CL6'BLKSIZ'
         DS    CL1' '
H2OEXCP  DS    CL11'-- EXCPS --'
         SPACE 2
```

```
*        DEFINE TOTAL I/O HEADER
         SPACE 1
         ORG   MSG+2
HTIO     DS    ØCL77
HCTEXCP  DS    CL12'DISK EXCPS ='
HTIOEXCP EQU   *-2,12
         DS    CL13' '
HCTAPE   DS    CL12'TAPE EXCPS ='
HTIOTAPE EQU   *-2,12
         DS    CL13' '
HCJESV   DS    CL11'JES + VIO ='
HTIOJV   EQU   *-2,12
         EJECT
*        DEFINE TOTAL TAPE MOUNTS HEADER
         SPACE 1
         ORG   MSG+2
HTTMOUNT DS    ØCL77
HCTMOUNT DS    CL23'TAPE MOUNTS: SPECIFIC ='
HTTMSPEC EQU   *-1,4
         DS    CL5' '
HCTNM    DS    CL14'NON-SPECIFIC ='
HTTMNSPC EQU   *-1,4
         DS    CL7' '
HCTUSED  DS    CL16'TAPE UNITS USED:'
HTTTUSED EQU   *-1,4
         SPACE 2
         SPACE 2
*        DEFINE TASK COMPLETION CODE HEADER
         SPACE 1
         ORG   MSG+2
HCOMP    DC    CL21'STEP COMPLETION CODE:'
         DS    C' '
HCACODE  DS    CL7' '
HCSA     EQU   HCACODE+1,3
HCUA     EQU   HCACODE+1,4
HCCC     EQU   HCACODE,3
HCCCODE  EQU   HCACODE+3,2
         DS    CL3' '
HABERC   DC    CL18'ABEND REASON CODE:'
         DS    C' '
HCRCODE  DS    CL8' '
         EJECT
*        DEFINE STEP/JOB CHARGE LINES
         SPACE
         ORG   MSG+2
COSTLINE DS    ØCL76
COSTLIN  DS    CL36
COSTCNT  DS    XL1Ø
         DS    CL2
COSTINFO DS    CL18
```

```
COSTMASK DS    XL10'4020206B2020214B2020'
         SPACE 2
*        DEFINE JOB MONITOR INFORMATION LINE
         SPACE
         ORG   MSG+2
MONITOR  WTO   'ISSD036A R88JOHNG PROCSTEP STEPNAME NO. PROGNAME - ABEND
               DED SXXX',DESC=2,ROUTCDE=(1,15),MF=L
MCC      EQU   *-4-4,4
MTYPE    EQU   *-4-4-8,5
MPROGRAM EQU   *-4-4-8-11,8
MSTEPNO  EQU   *-4-4-8-11-4,3
MSTEPNAM EQU   *-4-4-8-11-4-9,8
MPROCSTP EQU   *-4-4-8-11-4-9-9,8
MJOBNAME EQU   *-4-4-8-11-4-9-9-9,8
         SPACE 2
*        DEFINE MONITOR ERROR MESSAGE
         SPACE
         ORG   MSG+2
MNOJCTML WTO   'ISSD036A JOB CONTROL INFORMATION UNAVAILABLE FOR R88JOHN
               NG',DESC=2,ROUTCDE=15,MF=L
MNOJCTJN EQU   *-4-8,8
         EJECT
*        DEFINE ARGUMENT LISTS FOR ISDACTRT
         SPACE 1
         ORG
STEPARGS DS    0D                    START OF ARGUMENT LIST FOR STEP CALL
CPUTIME  DS    F                     CPU TIME FOR THE STEP
VIOEXCPS DS    F                     SUMMATION OF JES AND VIO EXCPS
DISKEXCP DS    F                     TOTAL OF EXCPS TO DISK DEVICES
DISKUSCT DS    H                     TOTAL OF MOUNTABLE DISK UNITS USED
DISKMONT DS    H                     TOTAL OF DISKS ACTUALLY MOUNTED-
TAPEEXCP DS    F                     TOTAL OF EXCPS TO TAPE DEVICES
TAPEUSCT DS    F                     TOTAL OF TAPE UNITS USED
URECEXCP DS    F                     TOTAL OF EXCP'S TO UNIT REC DEVICES
         ORG   STEPARGS              GO BACK TO BEGINNING OF ARGS
JOBARGS  DS    0D                    START OF ARGUMENT LIST FOR JOB CALL
CRDSREAD DS    F                     NUMBER OF CARDS READ BY JES2
PUNCHCRD DS    F                     NUMBER OF CARDS GENERATED BY JES2
PRNTLNES DS    F                     NUMBER OF LINES GENERATED BY JES2
PRNTCOPY DS    X                     NUMBER OF PRINT COPIES REQUESTED
         ORG   ,                     GET BACK TO NEXT AVAILABLE SLOT
         SPACE 2
*        DEFINE LIST OF ARGUMENTS RETURNED FROM ISDACTRT
         SPACE 1
RETRNARG DS    0F                    BEGINNING OF LIST RETURNED
RETCOST  DS    F                     CRU   COST
RETOCOST DS    F                     CPU COST
RETXCOST DS    F                     EXCP COST
RETBCOST DS    F                     BMP COST
RETICOST DS    F                     COST OF CARDS READ
```



```
RETLCOST DS    F                   COST OF PRINTED LINES
RETCCOST DS    F                   COST OF PUNCHED CARDS
RETSCOST DS    F                   COST OF A SPECIFIC TAPE MOUNT
RETNCOST DS    F                   COST OF NON-SPECIFIC TAPE MOUNT
         SPACE 2
*        DEFINE WORK AREA FOR ISDACTRT (MUST REMAIN IN GIVEN ORDER)
         SPACE 1
CALIOTIM DS    F                   IO EXCPS  *  (CRU/EXCP)
CALBPTIM DS    F                   BMP CALLS  *  (CRU/BMP CALLS)
CALFACPU DS    F                   CPU TIME  *  (CRU/CPU)
CALFACRU DS    F                   TOTAL CRU TIME 1/1ØØ SEC
         ORG   ,                   GET TO LAST AVAILABLE SLOT
         SPACE 2
*        DEFINE LENGTH OF DYNAMIC STORAGE AREA
         SPACE 1
         DS    ØD                  FORCE DOUBLEWORD BDRY FOR LENGTH
WORKLEN  EQU   *-WORKAREA          COMPUTE LENGTH FOR GET-, FREEMAIN
CLEARLEN EQU   *-TEMPD1            AREA TO BE ZEROED AFTER GETMAIN
         EJECT
***********************************************************************
*        THE FOLLOWING DSECT DESCRIBES STORAGE WHICH IS ACQUIRED      *
*        DURING THE FIRST STEP OF A JOB AND IS RELEASED WHEN THE      *
*        JOB ENDS.  THE ADDRESS OF THIS AREA IS KEPT IN THE COMMON    *
*        EXIT USER DATA FIELD OF THE COMMON EXIT TABLE.               *
***********************************************************************
         SPACE 1
KEEPSECT DSECT
KEEPJCT  DS    A                   ADDRESS OF JOB'S JCT IF HASP IS UP
KEEPSPAR DS    F                   SPARE
KEEPEXCP DS    F                   SUM OF ALL EXCPS(DA,TP,UR)
KEEPCPU  DS    F                   SUM OF CPU FOR ALL STEPS
KEEPBMP  DS    F                   BMP
KEEPINRT DS    H                   SAVE HASP INPUT ORIGIN
KEEPPRRT DS    H                   SAVE HASP PRINT ROUTE
KEEPPURT DS    H                   SAVE HASP PUNCH ROUTE
KEEPUSI  DS    X                   SAVE USI FLAGS
KEEPSMBF DS    X                   SMB PRINT FLAG
KEEPXXXX DS    X
KEEPYYYY DS    X
KEEPZZZZ DS    X
KEEPTMS  DS    F                    TMS ET AL WORK AREA
         SPACE
KEEPUTL  DS    F                    IEFUTL ET AL WORK AREA
         ORG   KEEPUTL
         SPACE
KEEPWAIT DS    H                    CONTIGUOUS WAIT COUNT
KEEPXTRA DS    H
         ORG
         SPACE
KEEPTPR  DS    H                    TALLY AREA FOR SPECIFIC TAPE MOUNTS
```

```
KEEPPTM  DS    H                    TALLY AREA - NON-SPECIFIC TAPE MNTS
KEEPUSCT DS    H                    TALLY AREA FOR TAPE DRIVES USED
KEEPRSVD DS    H                    AVAILABLE
         SPACE
KEEPTARY DS    F                    HOLD AREA FOR ASCBEWST
KEEPCIAO DS    F                    HOLD AREA FOR PREVIOUS ASCBEWST
KEEPCONV DS    CL16                 WORK AREA FOR CONVERT OF WAIT-BEGIN
         SPACE
KEEPLEN  EQU   ((*-KEEPSECT+7)/8)*8 COMPUTE LENGTH FOR GET- & FREEMAIN
         EJECT
***********************************************************************
*    TITLES USED FOR INFORMATION CONTAINED WITHIN THE EXCP SECTION    *
***********************************************************************
         SPACE
CMRDSECT DSECT
         DS    ØCL77
CMRDDNAM DS    CL8'DDNAME'
         DS    CL1' '
CMRUNIT  DS    CL4' '
         DS    CL1' '
CMRADDR  DS    CL4'ADDR'
         DS    CL1' '
         DS    CL6'BLKSIZ'
CMRBLKSZ EQU   *-7,7
         DS    CL1' '
         DS    CL11'-- EXCPS --'
CMREXCP  EQU   *-12,12
         DS    CL2' '
         EJECT
*        LOCAL EQUATES
         SPACE 1
JCTSDKAD EQU   32                  OBTAIN THIS VALUE FROM IEFAJCTB(JCT)
         SPACE
STEPTERM EQU   12
JOBTERM  EQU   16
PLNK     EQU   R8
KEEP     EQU   R9
SMF      EQU   R1Ø
BASE     EQU   R11
RAT      EQU   R12
WORK     EQU   R13
WORKSP   EQU   253                 SUBPOOL FOR WORKAREA
KEEPSP   EQU   239                 SUBPOOL FOR KEEPSECT
         SPACE 2
IEFACTRT CSECT ,
         SPACE
         END   IEFACTRT
```


Clustering and load-balancing

INTRODUCTION

As the millennium closes, there is a feature of distributed computer systems that has become widespread, particularly in the financial services industries – the need for 24 hours by 7 days a week availability. By itself, this presents a considerable challenge, but there is another feature: that of service. Not only do users want availability at any time of the day or night, but they also want a service level. They want a response time that is sufficiently short and that does not detract from the particular task that they are carrying out. For the system developer, this is where the problem starts – in fulfilling these requirements regardless of the number of users accessing the service. If the processing capability is increased on a single server, then this will not do anything for availability. So, how can we increase availability and processing capabilities whilst maintaining a reasonable response time? This is really the subject matter of this article. The topics covered are load-balancing and clustering.

LOAD-BALANCING

In its simplest form, load-balancing is a means of distributing load across two or more servers. Not only does it provide resilience in case a processor becomes unavailable, but it also improves on response time, by distributing the processing across the available processors.

With a load-balancing scheme in place configuration is likely to be scalable by simply adding processors or servers to the server farm. For Internet and intranet configurations, this is exactly what is required because it is very difficult to predict the likely peak loads. This unpredictability of Web-based traffic needs to be addressed by technologies that allow for scaling. One of the most effective is load-balancing amongst a group of servers that are all capable of providing the same set of services.

DNS and round robin

Perhaps the simplest form is to use a round robin scheme. In a simple configuration, a user requests the IP address of a computer system by sending the name to a Domain Name Server (DNS). The DNS then matches the name with an IP address and returns the address to the user. There may be several candidate servers and their associated IP addresses. The DNS returns a list of available IP addresses where the order has been arranged using a round robin algorithm. The user then selects the first IP address in the list and initiates the TCP connection to the associated server. In the event that the first address should fail, the second will be available because the user will hold the address in its cache. When the next user requests an IP address to the same server name, the DNS server responds with a list, as before, but with a different IP address first in the list.

Though simple and comparatively cheap, this scheme is not very effective because there may be instances when, although a server is unavailable, the DNS server has not been made aware of the change of status and is still listing the server's address as being available. In addition, the DNS server has no information relating to the relative loading on the servers. So, placing a server's IP address at the start of the list could result in a user connecting to the most heavily loaded server in the server farm – not a desirable state.

Front-end load-balancing

So, what exactly is load-balancing using a front-end processor? It is a technique for presenting a user workstation with a single IP address as an access route to two or more processors. As with the previous system, a user workstation requests the IP address of a computer system by sending the name to a Domain Name Server. The DNS then matches the name with an IP address and returns the address to the user. The user then sends the connection request to that IP address.

That IP address is not the target server that will service the request. Instead, it is the IP address of a load-balancer, whose task in life is to determine which physical server will service the request. The load-balancer is located as a form of front-end to the servers that are configured behind.

This form of load-balancer actually has two roles, the first of which is to determine which server processor is to process the next request and the second is to map the IP addresses seen by the user to the IP addresses seen by the chosen server processor. As the load-balancer is configured in front of the target server processors, the user is unaware of the identity of the IP address of the server that actually processes the request. Equally, the server has no awareness of the IP address of the user.

When the TCP connection is established, the load-balancer provides Network Address Translation for all subsequent exchanges for as long as that connection remains active, with the logical relationship between the user and the server remaining unchanged.

Many vendors provide this form of load-balancing and each has its own proprietary algorithm for determining which server will take the next connection. The range of algorithms includes:

- Round robin – where each new connection request is passed to the next server in the round robin list of server IP addresses.
- Number of concurrent connections with each server – where the load-balancer attempts to keep the number of connections balanced.
- Processor utilization and memory available – using software agents loaded on each server. These agents are interrogated by the load-balancer to obtain a real-time view of the processing resources available on each server. For the next connection request, the load-balancer will then select a server that has the lowest apparent processing load and thus a probability of providing the best service.
- Response time of each server – where the load-balancer monitors response time and will select the server giving the lowest response time. That response time might be the time it takes to receive the response to a ping issued from the load balancer and is thus an indication of the availability of each server’s Network Interface Card. This may not truly represent the response time to an application available on that server.

As a front-end, the load-balancer is itself a contributor to the overall performance viewed by a user. Thus high throughput capability of the load-balancer is an essential parameter when trying to determine performance.

Some load-balancers, such as Cisco's Cisco Local Director and Alteon's AceDirector, use router technology and have a potential throughput that is capable of meeting the requirements of the most demanding of configurations. However, for the most dynamic load-balancers, where the decision algorithms exploit information such as server processor utilization and memory usage, NT and Unix boxes are used. These boxes are configured and tuned for throughput and need to be sized very carefully if they are not to be the subject of performance degradation during periods of high load.

Single point of failure

As described, a load-balancer configured as a front end to the servers has an undesirable characteristic: it suffers from being a single point of failure. If the load-balancer fails, then all connections between users and servers are broken and none can be established.

The solution is to provide a standby or redundant load-balancer, which can be immediately switched in to take over the load-balancer tasks. Unfortunately, it is likely that all current connections will be lost and the users will have to attempt to re-establish their TCP connections.

Various schemes are available for configuring the standby load-balancer, but all involve doubling up the hardware used for load-balancing. There is definitely a cost penalty here, but it does provide resilience against a total system failure. The standby load-balancer monitors the activity of the live load-balancer that emits a heartbeat. If the heartbeat should fail within a set time-out period, then the standby load-balancer will take over and will also assume the network IP address – so no changes are required at the user workstation.

CLUSTERING

One of the major requirements in computer system configurations is scalability – defined here as the ability to provide more processor power without changing the underlying architecture. It might be adding one or more engines to a System/390 sysplex configuration. In the mainframe environment the technology is well understood and widely practised.

SMP clustering

Clustering is the term used to describe the relationship between two or more processors that share some common functions. Often used for NT and Unix configurations, the processors share memory and I/O, run under the same image of an operating system, and are typically described as Symmetric Multi-Processor (SMP) configurations. Though widely implemented as servers, the SMP technology provides scalability in terms of processor power, but does not contribute to any increase in availability. Scalability is also limited to a comparatively small number of processors. Though more processors can often be added, the relative increase in available processor power is often so small as to make the additional power not worth the cost. An example is NT running on a 4-processor SMP cluster. This is widely regarded as the optimum configuration with today's releases of NT. When Windows 2000 is delivered, it is expected that the maximum cost-effective number of processors in an SMP configuration will rise to at least eight.

Notice that the capability for SMP clustering is dependent on the operating system. There are Unix servers that can scale up to much larger numbers of processors, such as those from Sequent, IBM, Unisys, and Sun, each running their own variant of Unix.

Adding more power by adding more processors to an SMP cluster, along with additional memory, might be a way of servicing a greater number of concurrent users, but it does nothing for increasing the available up-time. If a processor should fail in an SMP cluster, that is likely to cause the operating system to fail and all users will suffer. This represents a single point of failure and means that we will not be able to achieve the very high availability required from today's computer systems.

So, SMP clustering provides us with extra processing power but does not provide adequate resilience, or protection against system failure.

Fail-over

Resilience implies some form of protection against failure. One solution is to provide a standby configuration that can be activated in the event of a failure of the live processor. This second processor, often referred to as the fail-over processor, is monitoring the live processor. At regular intervals the live processor is emitting a heartbeat and it is this heartbeat that is being continuously monitored by the standby processor. If the live processor fails to emit a heartbeat within a given time-out period, then a failure condition will be assumed and the second processor will instruct the live processor to halt, if it hasn't already, and then assume the live processor role. It is at this point during the recovery process that the 'new' live processor must recover the log and journal files and reconstruct the last stable state of databases and other resources for which integrity is a major concern.

When the recovery process has been completed, with any partially committed transactions rolled-back using information from the log and journal files, the fail-over processor can assume the full live state and can accept transaction requests.

Whilst on the subject of availability, during the recovery process it will not be possible to accept any transaction requests as the system is unavailable. The length of time during which the system is unavailable will depend on what the server cluster is actually being used for. If it is an information server providing read-only services for its users, the recovery time should be very short. Alternatively, if the server cluster has a population of users with read and write access to applications, then the time for the recovery process will be dependent on the number of current users and the activities they were carrying out at the time of the failure.

During the switch-over, the TCP connections with the users may be lost. However, there are features available that allow for IP address migration onto the fail-over machine. After fail-over the IP address of the now-dead machine can be assumed by the fail-over machine. It may be possible for the user connections to be maintained, though this will really be dependent on the application and the time taken to effect the switch-over and recovery.

The principles of fail-over have been widely implemented with each vendor taking a proprietary approach to the monitoring of the live processor by the fail-over processor. Examples include IBM with its HACMP (High Availability Clustering Multi-Processor system), and HP with ServiceGuard. These examples are all using Unix operating systems.

A more complex configuration using fail-over has the fail-over processor providing resilience for *n* live processors, where *n* might be 3, 4, or 5 processors. In this 1-for-*n* configuration, the number will depend on the availability characteristics of the processors and the stability of the applications running on them.

Recovery and fail-over

There are two scenarios in which fail-over is an essential feature of a high-availability configuration. The first is an unplanned outage, when the live processor fails and is unable to complete any work in progress or accept any new work. The second is the planned outage for maintenance purposes, perhaps as part of a systems software or hardware upgrade or even for loading new applications.

It might sound a fairly simple condition where the fail-over processor is set to take over the load that was previously assigned to the now-failed processor. Unfortunately this is far from the case. The only exception to this condition is when all work on the live processor can be gracefully terminated before the fail-over is invoked. This is never possible for the unplanned outage!

There are many circumstances that need addressing, some of which are:

- A fail-over is invoked in preparation for an upgrade of some software. After completion of the upgrade, a restore process takes place. It is possible that there could be some incompatibilities between the applications on the fail-over processor (that was live) and the applications on the upgraded processor. A user might find that an application they were accessing behaves differently when they are re-connected.

- The IP address of the Network Interface Card on the fail-over processor is different from that on the live processor. Unless there is a mechanism for migrating the IP address from the live processor to the fail-over processor, all connections with any remote systems and terminals will fail and users will have to initiate a new login.
- Print queues on the live processor may not transfer to the fail-over processor, in which case printing will cease until restoration has taken place or access enabled to the original print queues.
- There may be locks held by applications on the live processor that the fail-over system is unable to release.
- Ownership and configuration of disks – if disks are accessible from the fail-over system, then the relative physical and logical disks must be identical if problems are to be avoided.

Fail-over of the processors is but one of the major challenges to an automated fail-over scenario. The other major challenge relates to data fail-over, where the fail-over processor can take over all data whilst maintaining a state of high integrity.

Assuming that the live processor and the fail-over processor have their own disk drives and their own files, how does the fail-over processor maintain a copy of the data on the live processor? One solution exploits replication, where any change made by an application on the live processor is replicated onto the disk files of the fail-over processor. This might sound a satisfactory solution but there will be those occasions when the data being written to the live processor files are incorrect because of some process failure, resulting in incorrect or inconsistent data being replicated to the fail-over processor files. This is the so-called ‘toxic’ data condition and it is this condition that must be guarded against.

So, what are the options here? With replication, the log and journal files created by the live processor could be copied over to the fail-over processor each time there are changes. The fail-over processor will then search the log and journal files and apply or roll-back as appropriate.

Another solution is to configure the system with an interconnect that is shared by both processors. Then, at the time of switch-over, the fail-over processor can assume ownership of the data because it is able to access the same data files over the shared interconnect. The interconnect might take the form of a string of SCSI disks that are accessible by both processors, or access to files held on a RAID device.

Perhaps the most effective technique for guarding against inconsistencies between the live system files and the fail-over system files is to make all changes to a file or table within a logical unit of work. Using the principles defined by the ACID properties, all changes to the live files and the copy written to the fail-over files will be subject to a two-phase commit. Either both systems are changed or neither. This is the technique used by teleprocessing monitors that manage multiple resource managers such as DBMSs.

What are the ACID properties? In brief they are:

- Atomicity – relating to parts of a unit of work, either all parts are committed or all parts are rolled-back. There is no intermediate state.
- Consistency – any time that a transaction is invoked and successfully completes, the result is always the same, regardless of any other conditions that might have changed.
- Isolation – during the processing of a unit of work, any pending changes that have not been committed or rolled-back are not visible by any other transaction.
- Durability – after a transaction has completed, the result is lasting, even after a complete restart.

Some software vendors have exploited the fail-over technology by developing additional features that enable a fail-over condition to occur but within a distributed environment. All processors have live transactions in a distributed environment. In the event of a failure condition, the remaining live processor will take on the full load. An example is Oracle's Parallel Server (OPS), where an additional interconnect occurs between caches in each processor memory. This takes care of locks and provides a means of one processor requesting the lock held by another processor.

Where does NT fit?

The fail-over schemes as described are proprietary to each vendor, where the live processor and fail-over processor must come from the same vendor and be running the same operating system. A much better scheme would be to standardize on this process and eliminate any hardware and software dependencies. This is the objective behind the Microsoft approach to high availability, originally code-named Wolfpack. The intention has been to provide a mechanism that can be exploited by applications, so, instead of being limited to database recovery as exploited by many of the fail-over configurations, it should be possible to make business applications aware of the fail-over processor through some APIs. Then, in the event of a fail-over, it would be possible for an application to recover state and perhaps present the user with a screen as if nothing has actually happened. The user would be unaware of the fail-over condition.

The Microsoft Clustering Service provides developers with an API that can be used to control specific resources under conditions of fail-over. Under normal fail-over conditions, as described, all applications running on a server would be failed-over to another server. With the Microsoft Clustering Service, each application could be failed-over or distributed to other server nodes within the cluster. This eliminates the need to have a dedicated fail-over processor, so that all server nodes within the cluster are live and all have live applications.

Microsoft's Windows NT Load Balancing Service (WLBS) provides load balancing amongst cluster servers where each cluster server might be configured with Microsoft Clustering Service. This combination provides high availability and is highly scalable. Note that the capability of a complete WLBS and Microsoft Clustering Service is slowly being delivered and there are many features yet to appear in software releases.

Another technology, which is not being addressed in this article, concerns load balancing amongst servers supporting the Component Object Model (COM). This is a technology that will be realized with COM+, where objects can be instantiated on multiple servers on a dynamic basis. This is specific only to COM configurations and would not apply to browsers accessing a Web server, for example.

CLOSING THOUGHTS

This brief canter through the world of clustering and load balancing is by no means exhaustive. There are many other technologies that provide similar services, perhaps the most notable being those of Tandem and Digital, now both owned by Compaq. It is interesting that some of the technologies associated with both these vendors are appearing in the Microsoft clustering and load-balancing technologies.

To construct a site having clustering and load-balancing requires many design decisions that will all have a major impact on the overall cost. This has to be balanced against what exactly the business requirements are and whether the total cost of ownership is something that the business is willing to bear. Whatever those decisions are, the most important of all is scalability; being able to scale up by simply adding more processors is an essential feature of any architecture that is to be deployed to support Internet and intranet users.

Richard Lavender
Logica (UK)

A mailbox system for SMTP under MVS TCP/IP – 8

This issue we continue the code for the implementation of a mailbox system for SMTP, based on ISPF functions.

CLIST COMPLIB to compress PDS:

```
/*
/*   COMPLIB
/*    LIBRARY COMPRESS SHARED OR EXCLUSIVE CONTROL
/*
/*     RETURNS RC 4Ø95 IF ALLOCATION CONTROL CANNOT BE ACHIEVED
/*
PROC 2 LIB DISP DEBUG(NONE)
CONTROL NOMSG NOFLUSH NOLIST NOCONLIST NOSYMLIST
ERROR DO
 SET &RET = &LASTCC
 RETURN
END
SET &PRE =
SET &PREUID = &PRE&SYSUID
IF &STR(&DEBUG) = DEBUG THEN DO
 CONTROL MSG NOFLUSH LIST CONLIST SYMLIST
END
SET &PFX = &STR(&PREUID)
SET &LDSN = &LENGTH(&LIB)
IF &SUBSTR(1:1,&LIB) = &STR(') THEN DO
 SET &LIB = &SUBSTR(2:&LDSN,&LIB)
END
SET &LDSN = &LENGTH(&LIB)
IF &SUBSTR(&LDSN:&LDSN,&LIB) = &STR(') THEN DO
 SET &LIB = &SUBSTR(1:&LDSN - 1,&LIB)
END
IF &SYSENV = BACK THEN DO
 WRITE COMPRESS OF &LIB WITH DISP=&DISP
 ALLOC F(SYSPRINT) SYSOUT(X) /* DON'T REUSE, IT COULD BE PREALLOCATED
 SET &MAXCC = Ø              /* RESET IF NOT REALLOCATED
 IF &PREUID.. = . THEN DO
  SET &PFX = &STR(INSTPREF)
 END
END
IF &SYSENV NE BACK THEN DO
 ALLOC F(SYSPRINT) DUMMY REUSE
END
IF &SYSISPF = ACTIVE THEN DO
 ISPEXEC VGET (ZSCREEN)
 SET &QL = &STR(S&ZSCREEN..)
END
```

```
ELSE DO
 SET &QL =
END
SET &TS = +
&STR(T)&SUBSTR(1:2,&SYSTIME)&SUBSTR(4:5,&SYSTIME)&SUBSTR(7:8,&SYSTIME)
ALLOC DA('&PFX..&QL&TS..TEMP.LIST') FI(SYSIN) NEW SPACE(1) TRACKS +
RECFM(F) BLKSIZE(80) REUSE
OPENFILE SYSIN OUTPUT
SET &SYSIN = &STR( C I=LIB,O=LIB    )
PUTFILE SYSIN
CLOSFILE SYSIN
SET &RET = 0
ALLOC DA('&LIB') FI(LIB) &DISP REUSE
SET &ALLOCRET = &RET
IF &RET NE 0 THEN DO
 IF &SYSISPF = ACTIVE THEN DO
  ISPEXEC CONTROL DISPLAY LINE START(20)
 END
 WRITE ALLOCATION OF DISP=&DISP ON &LIB NOT ACHIEVED; NO COMPRESS DONE.
 CONTROL MSG
 SET &RET = 0
 ALLOC DA('&LIB') FI(LIB) &DISP REUSE
 CONTROL NOMSG
 IF &RET NE 0 THEN DO
  WRITE ALLOCATION OF DATASET &LIB NOT POSSIBLE
  IF &SYSISPF = ACTIVE THEN DO
   ISPEXEC CONTROL DISPLAY LINE START(1)
  END
  DEL '&PFX..&QL&TS..TEMP.LIST'
  EXIT CODE(4095)
 END
 WRITE
 WRITE IF CARE IS TAKEN AND YOU ARE 100% SURE THAT NOBODY ELSE IS +
 TRYING TO UPDATE
 WRITE '&LIB', YOU CAN ENTER: COMPRESS SHR
 WRITE AND COMPRESS WILL BE PERFORMED WITHOUT EXCLUSIVE CONTROL.
 IF &SYSISPF = ACTIVE THEN DO
  ISPEXEC CONTROL DISPLAY LINE START(1)
 END
 DEL '&PFX..&QL&TS..TEMP.LIST'
 EXIT CODE(4095)
END
ELSE DO
 IF &SYSISPF = ACTIVE AND &DISP = SHR THEN DO
  ISPEXEC LMINIT DATAID(DID) DATASET('&LIB') ENQ(SHRW)
  SET &N = 0
  SET &RET = 999
  DO WHILE &RET NE 0 AND &N < 30
   SET &N = &N + 1
   SET &RET = 0
   ISPEXEC LMOPEN DATAID(&DID) OPTION(OUTPUT)
```



```
    SLEEP 1
   END
   IF &RET NE Ø THEN DO
    ISPEXEC CONTROL DISPLAY LINE START(2Ø)
    WRITE SHARED ALLOCATION OF DATASET &LIB NOT POSSIBLE; TRY AGAIN
    ISPEXEC CONTROL DISPLAY LINE START(1)
    DEL '&PFX..&QL&TS..TEMP.LIST'
    EXIT CODE(4Ø95)
   END
   ISPEXEC LMCLOSE DATAID(&DID)
   ISPEXEC LMFREE DATAID(&DID)
    /* WHEN OPEN OUTPUT IS USED,THE DSCB IS REWRITTEN WITH THE OLD    */
    /* USED SIZE AT CLOSE TIME, SO IF LMCLOSE IS ISSUED AFTER THE     */
    /* CALL TO IEBCOPY COMPRESS,THE DSCB IS REWRITTEN WITH THE OLD    */
    /* SIZE NOT REFLECTING THE RELEASED SPACE FROM COMPRESS.          */
    /* THEREFORE WE CANNOT FULLY PROTECT A SHARED COMPRESS WITH THE   */
    /* ISPF ENQ OF SHRW.                                              */
  END
  /* INHIBIT ATTENTION BEFORE COMPRESS
  NOBREAK
  TSOEXEC CALL 'SYS1.LINKLIB(IEBCOPY)' ''
  /* ALLOW ATTENTION AGAIN
  BREAK
 END
 DEL '&PFX..&QL&TS..TEMP.LIST'
 ALLOC DA('&LIB') FI(LIB) SHR REUSE    /* ALLOC SHR INSTEAD OF OLD */
 FREE FI(LIB)
 IF &SYSENV NE BACK THEN DO
  ALLOC F(SYSPRINT) DA(*) REUSE
  ALLOC F(SYSIN) DA(*) REUSE
 END
 EXIT CODE(&MAXCC)
```

This CLIST can be used to create the ISPF primary command GETMAIL that can receive mail from SMTP:

```
   /*
   /* This CLIST will add GETMAIL as an ISPF command
   /*
PROC Ø
CONTROL   MSG NOFLUSH   LIST   CONLIST   SYMLIST
ERROR DO
 SET &RET = &LASTCC
 RETURN
END
SET &RET = Ø
 /*           COPY TO OWN DATASET OF ISPCMDS AS XXXCMDS
ALLOC FI(SYSPRINT) DA(*) REUSE
ALLOC FI(SYSIN) DUMMY REUSE
ALLOC FI(SYSUT1) DA('SYS1.SISPTENU(ISPCMDS)') SHR REUSE
```



```
ALLOC FI(SYSUT2) DA('&SYSUID..ISPF.ISPPROF(XXXCMDS)') SHR REUSE
CALL 'SYS1.LINKLIB(IEBGENER)'
FREE FI(SYSUT1,SYSUT2)
ALLOC FI(SYSIN) DA(*) REUSE
 /*           TAKE BACKUP OF ISPCMDS
ALLOC FI(SYSPRINT) DA(*) REUSE
ALLOC FI(SYSIN) DUMMY REUSE
ALLOC FI(SYSUT1) DA('SYS1.SISPTENU(ISPCMDS)') SHR REUSE
ALLOC FI(SYSUT2) DA('SYS1.SISPTENU(#ISPCMDS)') SHR REUSE
CALL 'SYS1.LINKLIB(IEBGENER)'
FREE FI(SYSUT1,SYSUT2)
ALLOC FI(SYSIN) DA(*) REUSE
  /*          MODIFY XXXCMDS IN OWN DATASET
ALLOC FI(ISPTABL) DA('&SYSUID..ISPF.ISPPROF') SHR REUS
ISPEXEC TBOPEN XXXCMDS
ISPEXEC TBBOTTOM XXXCMDS NOREAD
  /*
SET &ZCTVERB  = GETMAIL
SET &ZCTTRUNC = Ø
SET &ZCTACT   = SELECT CMD(%MAILRECV) NEWAPPL(U81) PASSLIB
SET &ZCTDESC  = ALLOW CLIST MAILRECV AS COMMAND GETMAIL
ISPEXEC VPUT (ZCTVERB ZCTTRUNC ZCTACT ZCTDESC)
ISPEXEC TBADD XXXCMDS
  /*
ISPEXEC TBCLOSE XXXCMDS
  /* COPY BACK XXXCMDS TO SYS1.SISPTENU AS ISPCMDS AND DELETE XXXCMDS
SET &RET = Ø
ALLOC FI(SYSPRINT) DA(*) REUSE
ALLOC FI(SYSIN) DUMMY REUSE
ALLOC FI(SYSUT1) DA('&SYSUID..ISPF.ISPPROF(XXXCMDS)') SHR REUSE
ALLOC FI(SYSUT2) DA('SYS1.SISPTENU(ISPCMDS)') SHR REUSE
CALL 'SYS1.LINKLIB(IEBGENER)'
FREE FI(SYSUT1,SYSUT2)
ALLOC FI(SYSIN) DA(*) REUSE
DEL '&SYSUID..ISPF.ISPPROF(XXXCMDS)'
EXIT
```

CLIST SDSFBACK to run SDSF in batch; this CLIST has several functions – among these are testing for pending mail to receive and call to MAILRECV to receive the mail:

```
 /*
 /* SDSFBACK:
 /* RUN SDSF IN BACKGROUND.
 /*
 /* PARAMETERS:
 /*   FUNC      : SDSF FUNCTION, EXAMPLES H, DA OSTC, O ...
 /*   PFX       : ADDRESS SPACE NAME PREFIX
 /*   RUNTYPE   : CHECK: SIMULATE FUNCTION; UPDATE: ISSUE COMMAND
 /*   CMD       : SDSF COMMAND TO ISSUE
```



```
 /*   TYPE      : RELATIVE POSITION IN SDSF OUTPUT OF ADDR SPACE TYPE
 /*   JNUM      : RELATIVE POSITION IN SDSF OUTPUT OF JOBNUMBER
 /*   JOBPOS    : RELATIVE POSITION IN SDSF OUTPUT OF JOBNAME
 /*   ADT       : PREFIX OF ADDRESS SPACE TYPE: J, S OR T
 /*   CICSLGN   : NO => NOT A CICS COMMAND;  YES => ISSUE CICS COMMAND
 /*   CONSID    : CONSOLE NO FOR USE IN CICS LOGON/CICS COMMAND
 /*   HASPCMD   : JES2 COMMAND PREFIX CHARACTER
 /*   CU        : USERID OF USER TO SIGN ON TO CICS
 /*   CP        : PASSWORD OF USER TO SIGN ON TO CICS
 /*   DEST      : DESTINATION; DEFAULT LOCAL
 /*   DEBUG     : CLIST TRACING, DEBUG: TRACING ACTIVE
 /*
PROC Ø DEBUG(NEBUG) FUNC('H') PFX('%%%C%%') +
RUNTYPE(UPDATE) CMD($TO) TYPE(4) JNUM(5) ADT(S) CICSLGN(NO) +
JOBPOS(1) CONSID(5) HASPCMD($) CU(CICSOPR) CP(CICSOPR) DEST(LOCAL)
CONTROL   MSG NOFLUSH NOLIST NOCONLIST NOSYMLIST
ATTN DO
 SET &FLUSH = FLUSH         /* NEXT STATEMENT MUST BE NULL LINE     */

END
ERROR DO
 SET &RET = &LASTCC
 RETURN
END
IF &STR(&DEBUG) = DEBUG THEN DO
 CONTROL MSG NOFLUSH LIST CONLIST SYMLIST
END
IF &FLUSH = FLUSH THEN DO
 CLOSFILE ISFIN
 FREE FI(ISFIN)
 CLOSFILE ISFOUT
 FREE FI(ISFOUT)
 ISPEXEC SETMSG MSG(INST34Ø)
 SET &ZISPFRC = Ø
 ISPEXEC VPUT (ZISPFRC) SHARED
 EXIT CODE(&ZISPFRC)
END
WRITE RUNTYPE &STR(====> &RUNTYPE)
WRITE FUNC &STR(====> &FUNC)
WRITE PFX &STR(====> &PFX)
WRITE CMD &STR(====> &CMD)
WRITE TYPE &STR(====> &TYPE)
WRITE JNUM &STR(====> &JNUM)
SET &RET = Ø
ALLOC FI(ISFIN) UNIT(VIO) SPACE(1 1) TRACKS NEW DELETE RECFM(F B) +
LRECL(8Ø) BLKSIZE(2792Ø) REUSE
OPENFILE ISFIN OUTPUT
IF &RET ¬= Ø THEN DO
 SET &ZISPFRC = &RET
 ISPEXEC VPUT (ZISPFRC) SHARED
 EXIT CODE(&ZISPFRC)
```

```
END
SET &ISFIN = &STR(PREFIX &PFX)
PUTFILE ISFIN
IF &STR(&CMD) = $TO AND &SUBSTR(1:1,&STR(&FUNC)) = &STR(O) THEN DO
 SET &ISFIN = &STR(DEST &DEST)
 PUTFILE ISFIN
END
SET &ISFIN = &STR(&FUNC)
PUTFILE ISFIN
CLOSFILE ISFIN
SET &DSPREF = &STR(INSTPREF)      /* HARD-CODED GENERAL PREFIX      */
IF &SYSPREF = &STR() THEN DO
 PROFILE PREFIX(&DSPREF)
END
ELSE DO
 IF &SYSPREF ¬= &DSPREF THEN DO
  PROFILE PREFIX(&SYSUID)
  SET &RET = Ø
  LISTC ENT('&SYSUID')
  IF &RET ¬= Ø THEN DO
   PROFILE PREFIX(&DSPREF)
  END
 END
END
SET &TSTAMP = +
&STR(T)&SUBSTR(1:2,&SYSTIME)&SUBSTR(4:5,&SYSTIME)+
&SUBSTR(7:8,&SYSTIME)
SET &SYSOUTTRAP = 999999
DEL '&SYSPREF..&TSTAMP..TEMP.LIST'
SET &SYSOUTTRAP = Ø
SET &CNT = Ø
SET &RET = Ø
ALLOC FI(ISFOUT) DA('&SYSPREF..&TSTAMP..TEMP.LIST') +
NEW SPACE(1 1) CYLINDERS +
UNIT(WORK) RECFM(F B A) LRECL(241) REUSE
DO WHILE &RET ¬= Ø AND &CNT < 3ØØ THEN DO
 SET &TSTAMP = +
 &STR(T)&SUBSTR(1:2,&SYSTIME)&SUBSTR(4:5,&SYSTIME)+
 &SUBSTR(7:8,&SYSTIME)
 SET &SYSOUTTRAP = 999999
 DEL '&SYSPREF..&TSTAMP..TEMP.LIST'
 SET &SYSOUTTRAP = Ø
 SET &RET = Ø
 ALLOC FI(ISFOUT) DA('&SYSPREF..&TSTAMP..TEMP.LIST') +
 NEW SPACE(1 1) CYLINDERS +
 UNIT(WORK) RECFM(F B A) LRECL(241) REUSE
 IF &RET NE Ø THEN DO
  FREE DA('&SYSPREF..&TSTAMP..TEMP.LIST')
 END
 SET &CNT = &CNT + 1
 SLEEP 5
```



```
END
SET &RET = Ø
ALLOC FI(ISFOUT) DA('&SYSPREF..&TSTAMP..TEMP.LIST') SHR REUSE
IF &RET ¬= Ø THEN DO
 SET &ZISPFRC = &RET
 ISPEXEC VPUT (ZISPFRC) SHARED
 EXIT CODE(&ZISPFRC)
END
SET &RET = Ø
SDSF ++24Ø,24Ø
IF &RET ¬= Ø THEN DO
 SET &ZISPFRC = &RET
 ISPEXEC VPUT (ZISPFRC) SHARED
 EXIT CODE(&ZISPFRC)
END
SET &RET = Ø
OPENFILE ISFOUT INPUT
IF &RET ¬= Ø THEN DO
 SET &ZISPFRC = &RET
 ISPEXEC VPUT (ZISPFRC) SHARED
 EXIT CODE(&ZISPFRC)
END
SET &SCANJOB = NO
SET &CNT = Ø
SET &RET = Ø
DO WHILE &RET ¬= 4ØØ AND &CNT < 512
 SET &RET = Ø
 GETFILE ISFOUT
 IF &RET = Ø THEN DO
  SET &SYSDVAL = &STR(&SYSNSUB(1,&ISFOUT))
  SET &SYSDVAL = &STR(&SYSNSUB(1,&SYSDVAL))
  READDVAL &A1 &A2 &A3 &A4 &A5 &A6 &A7 &A8 &A9 &A1Ø &A11 &A12 +
  &A13 &A14 &A15 &A16 &A17 &A18 &A19 &A2Ø &A21 &A22 &A23 &A24 +
  &A25 &A26 &A27 &A28 &A29 &A3Ø &A31 &A32 &A33 &A34 &A35 &A36 +
  &A37 &A38 &A39 &A4Ø &A41 &A42 &A43 &A44 &A45 &A46 &A47 &A48
  IF &SCANJOB = YES AND &STR(&A1) ¬= &STR() THEN DO
   WRITE &STR(&A1 &A2 &A3 &A4 &A5 &A6 &A7 &A8 &A9 &A1Ø &A11 &A12 +
   &A13 &A14 &A15 &A16 &A17 &A18 &A19 &A2Ø &A21 &A22 &A23 &A24 +
   &A25 &A26 &A27 &A28 &A29 &A3Ø &A31 &A32 &A33 &A34 &A35 &A36 +
   &A37 &A38 &A39 &A4Ø &A41 &A42 &A43 &A44 &A45 &A46 &A47 &A48)
   SET &D = &STR()
   SET &C = &&A&TYPE
   SET &J = &&A&JOBPOS
   IF &SUBSTR(1:1,&STR(&CMD)) = &HASPCMD THEN DO
    SET &D = &&A&JNUM
   END
   IF &STR(&C) ¬= &STR() THEN DO
    SET &C = &SUBSTR(1:1,&STR(&C))
   END
   SET &CMDSUF = &STR()
   IF &STR(&CMD) = $TO AND &STR(&C) = &STR(&ADT) AND +
```

```
&STR(&FUNC) = &STR(H) THEN DO
 IF &STR(&ADT) = &STR(S) THEN DO
  SET &CMDSUF = &STR(,ALL,ODISP=HOLD,NDISP=WRITE,Q=M)
 END
 IF &STR(&ADT) = &STR(T) THEN DO
  SET &CMDSUF = &STR(,ALL,ODISP=HOLD,NDISP=WRITE,Q=Z)
 END
 IF &STR(&ADT) = &STR(J) THEN DO
  SET &CMDSUF = &STR(,ALL,ODISP=HOLD,NDISP=WRITE,Q=W)
 END
END
IF &STR(&CMD) = S AND &STR(&C) = &STR(&ADT) AND +
&STR(&FUNC) = &STR(O) AND &STR(&PFX) = SMTP THEN DO
 WRITE MAIL TO ===> &STR(&A5)
 IF &RUNTYPE ¬= UPDATE THEN DO
  SE 'MAIL FOR YOU WAITING IN SMTP, ISSUE P.81.12 OR GETMAIL.' +
  USER(&STR(&A5)) NOW NOWAIT
 END
 ELSE DO
  %MAILRECV MAILUSER(&STR(&A5)) TOUSER(&STR(&A5)) +
  DEBUG(&STR(&DEBUG))
 END
END
IF &STR(&CMD) = $TO AND &STR(&C) = &STR(&ADT) AND +
&SUBSTR(1:1,&STR(&FUNC)) = &STR(O) THEN DO
 IF &STR(&A5) = &STR(&DEST) THEN DO
  SET &A = A
  SET &MAXVAR = 48
  SET &N = Ø
  DO WHILE &N < &MAXVAR
   SET &N = &N + 1
   SET &E = &STR(&&A&N)
   SET &LOC = &SYSINDEX(&STR(:),&STR(&E),Ø)
   IF &LOC > Ø THEN DO
    SET &N = &N + 1
    SET &OUTGRP = &STR(&&A&N)
    SET &N = &N + 1
    SET &OUTGRP = &STR(&OUTGRP)&STR(.)&STR(&&A&N)
    SET &N = &N + 1
    SET &OUTGRP = &STR(&OUTGRP)&STR(.)&STR(&&A&N)
    WRITE &STR(&OUTGRP)
    SET &N = &MAXVAR
   END
  END
  SET &CMDSUF = &STR(,OUTGRP=&OUTGRP,Q=H,D=U1)
 END
END
IF &STR(&C) = &STR(&ADT) THEN DO
 IF &RUNTYPE ¬= UPDATE AND &STR(&CICSLGN) ¬= YES THEN DO
  IF &STR(&CMDSUF) ¬= &STR() THEN DO
```

```
      WRITE &STR(&CMD&C&D&CMDSUF)
     END
    END
    IF &RUNTYPE = UPDATE AND &STR(&CICSLGN) ¬= YES THEN DO
     IF &STR(&CMDSUF) ¬= &STR() THEN DO
      COMMANDN &STR(&CMD&C&D&CMDSUF)
     END
    END
    IF &RUNTYPE ¬= UPDATE AND &STR(&CICSLGN) = YES THEN DO
     WRITE &STR(F &J,&CMD)
    END
    IF &RUNTYPE = UPDATE AND &STR(&CICSLGN) = YES THEN DO
     %CICSLGN &STR(&J) CONS(&CONSID) USERID(&CU) PW(&CP) +
     DEBUG(&STR(&DEBUG))
     COMMAND&CONSID &STR(F &J,&CMD)
     IF &STR(&CMD) = &STR(CEMT P SHUT) THEN DO
      COMMAND V NET,TERM,PLU=&STR(&J)
     END
    END
   END
  END
  IF &STR(&A1) = NP AND &STR(&A2) = JOBNAME THEN DO
   SET &SCANJOB = YES
   WRITE ====> SCANJOB
   WRITE &STR(&A1 &A2 &A3 &A4 &A5 &A6 &A7 &A8 &A9 &A1Ø &A11 &A12 +
   &A13 &A14 &A15 &A16 &A17 &A18 &A19 &A2Ø &A21 &A22 &A23 &A24 +
   &A25 &A26 &A27 &A28 &A29 &A3Ø &A31 &A32 &A33 &A34 &A35 &A36 +
   &A37 &A38 &A39 &A4Ø &A41 &A42 &A43 &A44 &A45 &A46 &A47 &A48)
  END
 END
END
CLOSFILE ISFOUT
FREE FI(ISFIN,ISFOUT)
SET &ISFRET = Ø
SET &ZISPFRC = &ISFRET
ISPEXEC VPUT (ZISPFRC) SHARED
EXIT CODE(&ISFRET)
```

```
// EXEC ASMCL,MEMBER=ADSPNM
*
*    CREATE ADDRESS SPACE NAME IN CLIST VARIABLE &ADSPNM
*
*     TSO COMMAND
*
ADSPNM   INITR
         JOBNAME                        GET JOBNAME
         MVC   ID,Ø(R15)                GET ADSPNM
         LOAD  EP=INSØ7Ø,ERRET=EXITRC8  GET CLIST VAR SUBR
         LR    R15,RØ                   GET ADDR OF SUBR
         CALLXA (15),(LENGTH,ID,VARLEN,VAR) CALL SUBRUTINE
         DELETE EP=INSØ7Ø               DELETE SUBR AGAIN
```

```
EXIT     EQU   *
         EXITR                          RETURN
EXITRC8  EQU   *
         EXITR RC=(8)                   RETURN WITH ERROR
ID       DC    CL8' '                   ADDRESS SPACE NAME
LENGTH   DC    AL2(L'ID)                LENGTH FOR SUBROUTINE
VAR      DC    C'ADSPNM'                CLIST VARIABLE
VARLEN   DC    AL2(L'VAR)               LENGTH OF VARIABLE
         LTORG
         END
```

```
// EXEC ASMCL,MEMBER=INSØ25M
*
*   FAST SEQUENTIAL DATASET COPY USING QSAM (IEBGENER REPLACEMENT)
*   ______________________________________________________________
*
*        DDNAMES:  SYSUT1   : INPUT DATASET
*                  SYSUT2   : OUTPUT DATASET
*                  SYSPRINT : SYSOUT
*
*   IF NO DCB INFO IS SPECIFIED ON SYSUT2 AND NEW DATASET OR SYSOUT,
*   THEN THE DCB-INFO WILL BE COPIED FROM SYSUT1.
*   IN CASE OUTPUT HAS GRATER LRECL THAN INPUT, OUTPUT RECORD WILL BE
*   PADDED WITH BINARY ZEROS.
*
         PRINT NOGEN
         DCBD  DSORG=PS .               DEFINE DCB
         IEFJFCBN .                     DEFINE JFCB
INSØ25   INITR AMODE=24,RMODE=24        BECAUSE OF IO
         OPEN  (SYSUT1,(INPUT))         OPEN INPUT
         OPEN  (SYSPRINT,(OUTPUT))      OPEN SYSPRINT
         LA    R14,SYSUT1               ADDRESS INPUT DCB
         USING IHADCB,R14               ADDRESS INPUT DCB
         ICM   R15,3,DCBBLKSI           GET INPUT BLKSIZE
         STH   R15,INPUTBLK             SAVE INPUT BLKSIZE
         ICM   R15,3,DCBLRECL           GET INPUT RECORD LENGTH
         STH   R15,INPUTLRL             SAVE INPUT RECORD LENGTH
         XR    R15,R15                  CLEAR FOR INSERT
         ICM   R15,1,DCBRECFM           GET INPUT RECORD FORMAT
         STC   R15,INPUTRFM             SAVE INPUT RECORD FORMAT
         RDJFCB SYSUT2                  GET JFCB FOR SYSUT2
         LTR   R15,R15                  TEST FOR GOOD RESPONSE
         BNZ   NOJFCB                   IF BAD DON'T TAKE DD-STATEMENT
         LA    R1Ø,JFCBWORK             GET ADDR OF JFCB
         USING INFMJFCB,R1Ø             BASE FOR JFCB
         OI    JFCBTSDM,JFCNWRIT        DON'T REWRITE JFCB
* JFCBDSCB CONTAINS 3-BYTES SVA OFFSET TO DSCB, CAN BE RETRIEVED
* FOR INFORMATION ABOUT EXITING DATASET.
NOJFCB   EQU   *
         LA    R14,SYSUT2               ADDRESS OUTPUT DCB
         USING IHADCB,R14               ADDRESS OUTPUT DCB
```

```
         ICM   R15,3,DCBBLKSI           GET OUTPUT BLKSIZE
         BNZ   NOSETBLK                 DON'T SET BLKSIZE
         ICM   R15,3,JFCBLKSI           GET JFCB BLKSIZE
         BNZ   NOSETBLK                 DON'T SET BLKSIZE
* JFCNEW CONSISTS OF TWO BITS, EACH OF WHICH ARE JFCMOD AND JFCOLD,
* THEREFORE THE TEST BNO.
         TM    JFCBTSDM,JFCSDS          TEST FOR SYSOUT
         BO    SETBLK                   GO SET BLKSIZE
         TM    JFCBIND2,JFCNEW          TEST FOR NEW DATASET
         BNO   NOSETBLK                 EXISTING DATASET
SETBLK   EQU   *
         MVC   DCBBLKSI,INPUTBLK        SET OUTPUT BLKSIZE
NOSETBLK EQU   *
         ICM   R15,3,DCBLRECL           GET OUTPUT RECORD LENGTH
         BNZ   NOSETLRL                 DON'T SET RECORD LENGTH
         ICM   R15,3,JFCLRECL           GET JFCB RECORD LENGTH
         BNZ   NOSETLRL                 DON'T SET RECORD LENGTH
         TM    JFCBTSDM,JFCSDS          TEST FOR SYSOUT
         BO    SETLRL                   GO SET LRECL
         TM    JFCBIND2,JFCNEW          TEST FOR NEW DATASET
         BNO   NOSETLRL                 EXISTING DATASET
SETLRL   EQU   *
         MVC   DCBLRECL,INPUTLRL        SET OUTPUT RECORD LENGTH
NOSETLRL EQU   *
         XR    R15,R15                  CLEAR FOR INSERT
         ICM   R15,1,DCBRECFM           GET OUTPUT RECORD FORMAT
         BNZ   NOSETRFM                 DON'T SET RECORD FORMAT
         ICM   R15,3,JFCRECFM           GET JFCB RECORD FORMAT
         BNZ   NOSETRFM                 DON'T SET RECORD FORMAT
         TM    JFCBTSDM,JFCSDS          TEST FOR SYSOUT
         BO    SETRFM                   GO SET RECFM
         TM    JFCBIND2,JFCNEW          TEST FOR NEW DATASET
         BNO   NOSETRFM                 EXISTING DATASET
SETRFM   EQU   *
         MVC   DCBRECFM,INPUTRFM        SET OUTPUT RECORD FORMAT
NOSETRFM EQU   *
* DON'T USE OPEN TYPE=J SINCE JFCB IS NOT MODIFIED AND CERTAIN JCL
* INFO CAN BE LOST LIKE VOL=REF ETC
         OPEN  (SYSUT2,(OUTPUT))        OPEN OUTPUT DATASET
LOOP     EQU   *
         XR    R14,R14                  CLEAR WORK REGISTER
         XR    R15,R15                  CLEAR WORK REGISTER
         XR    R7,R7                    CLEAR WORK REG
         ICM   R7,7,=AL3(L'IOAREA)      GET LENGTH TO BE CLEARED
         LA    R6,IOAREA                GET ADDRESS OF RECORD
         MVCL  R6,R14                   CLEAR INFO AREA WITH BIN ZEROS
         GET   SYSUT1,IOAREA            GET NEXT MEMBER REC
         L     R14,RECNO                GET RECORD NO
         LA    R14,1(R14)               UPDATE RECORD NO
         ST    R14,RECNO                SAVE RECORD NO
```

```
         LA    R1,IOAREA                GET ADDRESS OF RECORD
         LR    RØ,R1                    ADDRESS FOR PUT
         PUT   SYSUT2,(RØ)              WRITE AGAIN
         B     LOOP                     RECYCLE
EOF      EQU   *
         L     R14,RECNO                GET NO OF RECORDS
         CVD   R14,D1                   CONVERT IT TO DECIMAL
         UNPK  D2,D1                    UNPACK IT
         OI    D2+L'D2-1,X'FØ'          SET SIGN
         PUT   SYSPRINT,TEXT            INFORM USER
         CLOSE (SYSUT1,,SYSUT2,,SYSPRINT) CLOSE FILES
         EXITR
RECNO    DC    F'Ø'                     NO OF RECORDS PROCESSED
INPUTBLK DS    H                        INPUT BLOCK SIZE
INPUTLRL DS    H                        INPUT RECORD LENGTH
INPUTRFM DS    C                        INPUT RECORD FORMAT
TEXT     DC    CL32'NUMBER OF RECORDS PROCESSED: '
D2       DS    D
DUMMY    DC    CL1ØØ' '                 DUMMY
D1       DS    D
SYSUT1   DCB   DDNAME=SYSUT1,DSORG=PS,MACRF=(GM),EODAD=EOF
SYSUT2   DCB   DSORG=PS,DDNAME=SYSUT2,MACRF=(PM),                      *
               EXLST=EXLST              OUTPUT DCB
EXLST    DC    ØF'Ø',X'87',AL3(JFCBWORK) EXITLIST
JFCBWORK DC    ØD'Ø',176X'ØØ'           JFCB AREA
SYSPRINT DCB   DDNAME=SYSPRINT,DSORG=PS,MACRF=PM,LRECL=12Ø,RECFM=FB
         LTORG
IOAREA   DS    28CL1ØØ                  328ØØ BYTES
         END

//*
//*    TSO COMMAND, EXEC PGM, CALL OR SUBROUTINE
//*      SET THE TSO USER IN WAIT THE NUMBER OF SECS INDICATED
//*      EG. SLEEP 1Ø                   WAIT FOR 1Ø SECS WAIT
//*      EXEC PGM=SLEEP,PARM='1Ø'
//*      CALL 'YOUR.LOAD.LIBRARY(SLEEP) '1Ø'
//*
// EXEC ASMCL,MEMBER=SLEEP,
// PARM.ASM='RENT',
// PARM.LKED='XREF,LET,LIST,RENT,REFR,REUS'
         GBLC  &ID
         GBLA  &IDLEN
SLEEP    INITR AMODE=31,RMODE=ANY,GENCODE=YES,SIZE=GETSIZE
WORKAREA DSECT
         ORG   USERWORK
DW       DS    D                        WORK FOR CONVERT
INTVL    DS    F                        WAIT INTERVAL
GETSIZE  EQU   *-WORKAREA
*
&ID      CSECT
*
```

```
         LTR   R15,R15                   TEST FOR ZERO DATA LENGTH
         BNP   EXITRC16                  IF NO DATA
         LR    R8,R14                    GET ADDR OF 1ST DATAADDR
SCAN     EQU   *
         TM    OPTIONS,ATTN              IS ATTN FLAG SET
         BO    EXIT                      RETURN IF ATTN
         CLI   Ø(R8),C'Ø'                TAKE AWAY LEADING NONDIGITS/Ø
         BH    FIRSTDIG                  GOT A DIGIT
         LA    R8,1(R8)                  POINT TO NEXT
         BCT   R15,SCAN                  RECYCLE
         B     EXITRC8                   NO VALUE, THEN EXIT WITH ERROR
FIRSTDIG EQU   *
         LR    R9,R8                     SAVE ADDR OF FIRST DIGIT
SCAN2    EQU   *
         LA    R9,1(R9)                  GET NEXT BYTE
         BCTR  R15,Ø                     COUNT DOWN RESIDUAL COUNT
         LTR   R15,R15                   SOMETHING LEFT
         BZ    ENDSCAN                   NO MORE INPUT
         CLI   Ø(R9),C'Ø'                LOOK FOR NONDIGITS
         BNL   SCAN2                     IF DIGIT RESCAN
ENDSCAN  EQU   *
         SR    R9,R8                     COMPUTE LENGTH
         CH    R9,=H'9'                  TEST FOR TOO LONG
         BNH   LENOK                     LENGTH OK
         LA    R9,9                      ASSUME LENGTH OF 9
LENOK    EQU   *
         BCTR  R9,RØ                     REDUCE FOR EXECUTE
         LA    R1Ø,7                     GET LENGTH OF DOUBLE WORD
         SLL   R1Ø,R4                    SHIFT TO HIGH ORDER
         OR    R1Ø,R9                    SET UP FOR EXECUTE
         EX    R1Ø,PACK                  PACK THE NUMBER
         CVB   R11,DW                    CONVERT TO BINARY
         LA    R6,1ØØ                    GET IN HUNDREDS
         MR    R1Ø,R6                    GET IN HUNDREDS
         ST    R11,INTVL                 SAVE WAIT TIME
         STIMER WAIT,BINTVL=INTVL        WAIT
         EXITR                           RETURN
PACK     PACK  DW(Ø),Ø(Ø,R8)             EXECUTED PACK
         LTORG
         END
```

```
// EXEC ASMCL,MEMBER=HALT
*
*    TSO COMMAND, EXEC PGM, CALL OR SUBROUTINE
*    OPTIONAL PARAMETER: MAX WAIT TIME IN SECONDS
*    SET THE TSO USER IN WAIT THE NUMBER OF SECS INDICATED OR IF NO
*     PARAMETER AND ENDLESS WAIT.
*        EG HALT 1Ø                     WAIT FOR 1Ø SECS WAIT
*        EG HALT                        WAIT UNTILL TERMINATED
*        EG HALT Ø     TERMINATE DIRECTLY (SIMULATE TIME EXP, SET RC4)
```

```
*            EXEC PGM=HALT,PARM='10'
*            EXEC PGM=HALT
*            CALL 'YOUR.LOAD.LIBRARY(HALT)' '10'
*
*
*  WAIT UNTIL COMMUNICATIONS ECB IS POSTED, AND IF WAIT TIME IS
*   SUPPLIED, LIMIT WAITTIME TO SPECIFIED TIME.
*  CONTENTS OF MODIFY WILL BE SHOWN ON FILE SYSPRINT AND IN CLIST
*  VARIABLE &HALT.
*  IF STOP ISSSUE &HALT WILL CONTAIN:  STOP
*  IF TIME EXPIRED &HALT WILL CONTAIN: TIME EXPIRATION
*
*  RC: 0 FOR NORMAL RETURN FROM MODIFY
*  RC: 4 FOR NORMAL RETURN IF STOP IS ISSUED
*  RC: 4 FOR NORMAL RETURN IF SLEEP TIME EXPIRED
*  RC: 12 FOR ERROR.
*
         PRINT NOGEN
         CVT   DSECT=YES,PREFIX=YES,LIST=NO
         PRINT NOGEN
         IHAASCB
         IHAASXB
         IHAPSA
         USING PSA,R0
         IHARB .                         RB
         IKJTCB
         IHAACEE
         IEZJSCB
         IKJPSCB
         IEFAJCTB
         IEFTCT
         IKJTSB
         IEESMCA
         IEFUCBOB PREFIX=YES
UCBPFLEN EQU   UCBCMSEG-UCB
         IEFJESCT .                      JESCT
         IEFJSCVT .                      JSCVT (SSCT)
         DCBD  DSORG=PS                  DCB
         DSECT
         IEZCOM                          COMMUNICATIONS MAPPING
         DSECT
         IEZCIB                          COMMUNICATIONS INPUT BUFFER
         IKJCPPL                         CPPL
         IHAECB                          ECB
         IHASDWA DSECT=YES               SDWA FOR ESTAE/SETRP MACRO
         PRINT GEN
*
WORKAREA DSECT                           GETMAINED WORKAREA
SAVEAREA DS    CL72                      SAVE AREA
STAXD    STAX  STAXEXIT,MF=L             STAX LIST FORM
```

```
ESTAEW   DS    XL(LESTAEL)              ESTAE PARM LIST AREA
ESTAPARM DS    4F                       PARM LIST TO RETRY ROUTINE:
ERROR    DS    H                        ERROR CODE
PARMADDR DS    A                        ADDR OF PARMLIST
DATAADDR DS    A                        ADDR OF PARAMETER DATA
PARMLEN  DS    H                        LENGTH OF PARAMETER DATA
DW       DS    D                        WORK FOR CONVERT
INTVL    DC    F'1ØØ'                   WAIT INTERVAL
CIBADDR  DS    A                        ADDR OF CIB
OPTIONS  DS    X                        EXECUTION OPTIONS
ATTN     EQU   X'8Ø'                    ATTN FLAG SET
TSOCMD   EQU   X'4Ø'                    INDICATE CALLED AS TSO COMMAND
EXECCALL EQU   X'2Ø'                    INDICATE JCL-EXEC OR TSO-CALL
SUBRUTIN EQU   X'1Ø'                    INDICATE CALLED AS SUBROUTINE
SLEEP    EQU   X'Ø8'                    CALL SLEEP
COMMPOST EQU   X'Ø4'                    COMM ECB POSTED
TIMEPOST EQU   X'Ø2'                    TIME ECB POSTED
LEADNULL EQU   X'Ø1'                    LEADING ZERO FOUND IN PARM
WORKLEN  EQU   *-WORKAREA               LENGTH TO GETMAIN
*
&ID      SETC  'HALT'
&IDLEN   SETA  K'&ID
&ID      INITR SIZE=WORKLEN,AMODE=24,RMODE=24,CLEAR=YES   MUST AMODE 24
         USING WORKAREA,R13             ADDRESS WORKAREA
         ST    R1,PARMADDR              SAVE ADDR OF PARMLIST
         LA    RØ,RTRYRTN1              RETRY ROUTINE - NO SDWA
         ST    RØ,ESTAPARM              STORE IN PARAMETER LIST
         LA    RØ,RTRYRTN2              RETRY ROUTINE WITH SDWA
         ST    RØ,ESTAPARM+4            STORE IN PARAMETER LIST
         STM   R12,R13,ESTAPARM+8       STORE BASE & DATA REG IN PARM
         MVC   ESTAEW(LESTAEL),ESTAEL   MOVE IN ESTAE PARAMETER LIST
         ESTAE RECOVERY,CT,PARAM=ESTAPARM,MF=(E,ESTAEW) SETUP RCVRY
         MVC   STAXD(STAXLEN),STAXL     MOVE IN STAX LIST TO GETMAINED
         STAX  STAXEXIT,USADDR=WORKAREA,MF=(E,STAXD) SET ATTN EXIT
         L     R6,PARMADDR              GET ADDR TO INPUT PARM
         L     R8,Ø(R6)                 GET PARM ADDR
         XR    R15,R15                  CLEAR BEFORE INSERT
         ICM   R15,3,Ø(R8)              GET PARM LENGTH
         STH   R15,PARMLEN              SAVE LENGTH OF INPUT
         LA    R14,4+&IDLEN             GET LENGTH OF PGM NAME + HDR
         LA    R1,4(R8)                 POINT TO EVENTUAL CMD-NAME
CMDSCAN  EQU   *
         CR    R15,R14                  ANY ROOM FOR LEN + CMDNAME
         BL    SUBROUTINE               IF NOT, TRY SUBROUTINE
         CLC   Ø(&IDLEN,R1),=C'&ID'     TSO COMMAND
         BE    CMDFOUND                 FOUND CMD-NAME
         CLI   Ø(R1),C' '               BLANK BEFORE CMD-NAME
         BNE   SUBROUTINE               TRY SUBROUTINE
         LA    R1,1(R1)                 POINT TO NEXT IN INPUT
         LA    R14,1(R14)               COUNT UP LENGTH OF PREFIX
```

```
         B     CMDSCAN                  RECYCLE
CMDFOUND EQU   *
         XR    R1,R1                    CLEAR WORK REGISTER
         ICM   R1,3,2(R8)               GET OFFSET TO DATA
         LA    R1,4(R1)                 ACCOUNT FOR LENGTH FIELDS
         SR    R15,R1                   REDUCE BY LENGTH OF HEADER
         STH   R15,PARMLEN              SAVE LENGTH OF INPUT
         BZ    NOSLEEP                  IF EQUAL, PROCEED WITH NO DATA
         LA    R14,Ø(R1,R8)             GET ADDR OF DATA
         ST    R14,DATAADDR             SAVE ADDR OF DATA
         OI    OPTIONS,TSOCMD           INDICATE CALLED AS TSOCOMMAND
         B     TSOCOMMAND               PROCEED
SUBROUTINE EQU *
         OI    OPTIONS,SUBRUTIN         INDICATE CALLED AS SUBROUTINE
         LA    R14,2(R8)                GET ADDR OF DATA
         XR    R1,R1                    CLEAR WORK REGISTER
         ICM   R1,3,PARMLEN             GET MAX PARM LENGTH
         BNP   ENDPARMSCAN              NO DATA AT ALL
PARMSCAN EQU   *
         CLI   Ø(R14),C' '              LEADING BLANK
         BNE   ENDPARMSCAN              END SCAN FOR LEADING BLANK
         LA    R14,1(R14)               POINT TO NEXT IN PARAMETER
         BCT   R1,PARMSCAN              RECYCLE SCAN
ENDPARMSCAN EQU *
         STH   R1,PARMLEN               SAVE REAL LENGTH OF INPUT
         ST    R14,DATAADDR             SAVE ADDR OF DATA
         L     R1,PARMADDR              GET ADDR TO INPUT PARM
         ICM   R1,15,Ø(R1)              GET FIRST PARM ADDR
         BM    SETONEPARM               IT WAS LAST PARM
*
*  DECIDE IF PROGRAM IS DIRECTLY EXECUTED VIA JCL-EXEC OR TSO-CALL;
*  AND SET OPTIONS ACCORDINGLY:
         L     R1,PSATOLD               GET TCB ADDR
         USING TCB,R1                   ADDRESS TCB
         L     R1,TCBRBP                GET RB POINTER
         USING RBBASIC,R1               ADDRESS REQUEST BLOCK
         CLC   =C'&ID',RBEXSAVE         DIRECT EXECUTE OF EXIT-NAME
         BNE   TSOCOMMAND               NOT EXEC PGM= OR TSO CALL
SETONEPARM EQU *
         OI    OPTIONS,EXECCALL         INDICATE JCL-EXEC OR TSO-CALL
         NI    OPTIONS,255-SUBRUTIN     TURN OFF SUBROUTINE
TSOCOMMAND EQU *
*
*  NORMAL PROCESSING
*
         XR    R15,R15                  CLEAR WORK REGISTER
         ICM   R15,3,PARMLEN            ANY DATA
         BNP   NOSLEEP                  IF NO DATA, PROCEED
         L     R8,DATAADDR              GET ADDR OF 1ST DATAADDR
SCAN     EQU   *
```



```
         TM    OPTIONS,ATTN             IS ATTN FLAG SET
         BO    EXIT                     RETURN IF ATTN
         CLI   Ø(R8),C'Ø'               TAKE AWAY LEADING NONDIGITS/Ø
         BH    FIRSTDIG                 GOT A DIGIT
         BL    NOTADIG                  NOT A DIGIT
         OI    OPTIONS,LEADNULL         SET NULL FOUND
NOTADIG  EQU   *
         LA    R8,1(R8)                 POINT TO NEXT
         BCT   R15,SCAN                 RECYCLE
         TM    OPTIONS,LEADNULL         WAS ONLY XERO FOUND
         BO    TIMEEXPD                 PARM WAS NULL, SIMULATE EXPIRED
         B     NOSLEEP                  NO VALUE, THEN PROCEED
FIRSTDIG EQU   *
         OI    OPTIONS,SLEEP            WE WANT TO SLEEP AS WELL
         LR    R9,R8                    SAVE ADDR OF FIRST DIGIT
SCAN2    EQU   *
         LA    R9,1(R9)                 GET NEXT BYTE
         BCTR  R15,Ø                    COUNT DOWN RESIDUAL COUNT
         LTR   R15,R15                  SOMETHING LEFT
         BZ    ENDSCAN                  NO MORE INPUT
         CLI   Ø(R9),C'Ø'               LOOK FOR NONDIGITS
         BNL   SCAN2                    IF DIGIT RESCAN
ENDSCAN  EQU   *
         SR    R9,R8                    COMPUTE LENGTH
         CH    R9,=H'9'                 TEST FOR TOO LONG
         BNH   LENOK                    LENGTH OK
         LA    R9,9                     ASSUME LENGTH OF 9
LENOK    EQU   *
         BCTR  R9,RØ                    REDUCE FOR EXECUTE
         LA    R1Ø,7                    GET LENGTH OF DOUBLE WORD
         SLL   R1Ø,R4                   SHIFT TO HIGH ORDER
         OR    R1Ø,R9                   SET UP FOR EXECUTE
         EX    R1Ø,PACK                 PACK THE NUMBER
         CVB   R11,DW                   CONVERT TO BINARY
         LA    R6,1ØØ                   GET IN HUNDREDS
         MR    R1Ø,R6                   GET IN HUNDREDS
         ST    R11,INTVL                SAVE WAIT TIME
NOSLEEP  EQU   *
         OPEN  (SYSPRINT,(OUTPUT))      OPEN SYSPRINT
         LA    R6,SYSPRINT              ADDRESS DCB
         USING IHADCB,R6                ADDRESS DCB
         EXTRACT COMM,'S',FIELDS=(COMM) GET COMM POINTERS
         L     R1Ø,COMM                 GET ADDRESS OF FIRST ANSWER PTR
         USING COMLIST,R1Ø              ADDRESS POINTERS
         L     R15,COMECBPT             GET COMMUNICATIONS ECB ADDRESS
         ST    R15,COMMECBA             SAVE ECB ADDRESS IN WAIT LIST
         XC    TIMEECB,TIMEECB          CLEAR TIMER ECB
         L     RØ,TIMEECBA              GET COMMUNICATIONS ECB ADDRESS
         LA    R1,1                     GET A BIT
         SLL   R1,31                    PUT BIT IN HIGH ORDER
```



```
         OR    R1,RØ                    COMPUTE WAIT ECB AS LAST ECB
         ST    R1,TIMEECBA              SAVE ECB ADDRESS IN WAIT LIST
         ICM   R11,15,COMCIBPT          GET ADDR OF START CIB
         ST    R11,CIBADDR              SAVE ADDR OF CIB
         USING CIBNEXT,R11              ADDRESS CIB
         BZ    NOCIB                    NO START CIB
         CLI   CIBVERB,CIBSTART         IS IT START CIB
         BNE   NOCIB                    DON'T FREE IF NOT START
         QEDIT ORIGIN=COMCIBPT,BLOCK=(R11) FREE START CIB
         LTR   R15,R15                  TEST FOR GOOD RC
         BNZ   EXITRC12                 RETURN WITH ERROR
NOCIB    EQU   *
         QEDIT ORIGIN=COMCIBPT,CIBCTR=1 SET UP FOR NO OF MODIFIES
         TM    OPTIONS,SLEEP            DO WE WANT TO SLEEP AS WELL
         BZ    NOATTACH                 DON'T ISSUE SLEEP
         STIMER REAL,TIMEEXIT,BINTVL=INTVL   SET TIMER AND EXIT
NOATTACH EQU   *
         WAIT  1,ECBLIST=ECBLIST,LONG=YES  WAIT UNTIL AN ECB IS POSTED
         TM    OPTIONS,ATTN             IS ATTN FLAG SET
         BO    EXIT                     RETURN IF ATTN
         QEDIT ORIGIN=COMCIBPT,CIBCTR=Ø DON'T ALLOW MODIFIES
         L     R9,COMMECBA              GET ADDR OF COMM ECB
         USING ECB,R9                   ADDRESS ECB
         TM    ECBCC,ECBPOST            ECB POSTED
         BZ    NOTCOMME                 WAS NOT A COMM ECB
         OI    OPTIONS,COMMPOST         COMMUNICATIONS ECB POSTED
         B     BYPCOMME                 PROCEED
NOTCOMME EQU   *
         OI    OPTIONS,TIMEPOST         TIMER ECB POSTED
         QEDIT ORIGIN=COMCIBPT,BLOCK=(R11) FREE CIB
* DON'T TEST FOR ERROR IN R15
BYPCOMME EQU   *
         TM    OPTIONS,SLEEP            DO WE WANT TO SLEEP AS WELL
         BZ    NODETACH                 DON'T DETACH
         TM    OPTIONS,TIMEPOST         WAS TIMER ECB POSTED
         BZ    NOTTIMEE                 WAS NOT A TIME ECB POST
         LH    R15,TIMEECB+2            GET RETCODE
         STH   R15,ERROR                SAVE RETCODE
NOTTIMEE EQU   *
NODETACH EQU   *
         MVI   MODDATA,C' '             CLEAR RECEIVING DATA
         MVC   MODDATA+1(L'MODDATA-1),MODDATA CLEAR RECEIVING DATA
         TM    OPTIONS,TIMEPOST         WAS TIMER ECB POSTED
         BO    TIMEEXPD                 WAS A TIME ECB POST
         ICM   R11,15,COMCIBPT          GET ADDR OF CIB
         CLI   CIBVERB,CIBSTOP          IS IT STOP CIB
         BE    STOP                     STOP
         CLI   CIBVERB,CIBMODFY         IS IT MODIFY
         BE    MODIFY                   STOP
         BNZ   EXITRC12                 ELSE RETURN WITH ERROR
```



```
MODIFY   EQU   *
         ICM   R15,3,CIBDATLN           GET LENGTH OF MODIFY
         BZ    NODATA                   NO DATA IN MODIFY
         BCTR  R15,Ø                    REDUCE FOR EXECUTE
         EX    R15,*+4                  MOVE MODIFY DATA
         MVC   MODDATA(Ø),CIBDATA       MOVE MODIFY DATA
NODATA   EQU   *
         TM    DCBOFLGS,DCBOFOPN        IS SYSPRINT OPEN
         BZ    NOPUTCM                  BYPASS SYSPRINT
         PUT   SYSPRINT,MODIFYCM        SHOW MODIFY
         PUT   SYSPRINT,MODDATA         SHOW MODIFY DATA
NOPUTCM  EQU   *
         ICM   R15,3,CIBDATLN           GET LENGTH OF MODIFY
         STH   R15,VARLEN               BUILD LENGTH OF VARIABLE
         QEDIT ORIGIN=COMCIBPT,BLOCK=(R11) FREE CIB
* DON'T TEST FOR ERROR IN R15
         LA    R1,PARMLIST              POINT TO PARMLIST
         LINK  EP=INSØ7Ø,ERRET=BYPVAR   CALL CLIST VAR CREATION
BYPVAR   EQU   *
         B     EXIT                     RETURN INDICATING MODIFY
**DON'T|  B     RECYCLE                  PROCESS NEXT CIB IN CHAIN
STOP     EQU   *
         LA    R15,L'STOPVARC           BUILD LENGTH OF VARIABLE
         STH   R15,VARLEN               BUILD LENGTH OF VARIABLE
         MVC   MODDATA(L'STOPVARC),STOPVARC BUILD CLIST VAR
         QEDIT ORIGIN=COMCIBPT,BLOCK=(R11) FREE CIB
* DON'T TEST FOR ERROR IN R15
         LA    R1,PARMLIST              POINT TO PARMLIST
         LINK  EP=INSØ7Ø,ERRET=BYPVAR1  CALL CLIST VAR CREATION
BYPVAR1  EQU   *
         TM    DCBOFLGS,DCBOFOPN        IS SYSPRINT OPEN
         BZ    NOCLSCM                  BYPASS SYSPRINT
         PUT   SYSPRINT,STOPCM          SHOW STOP
         CLOSE SYSPRINT                 CLOSE SYSPRINT
NOCLSCM  EQU   *
         MVC   ERROR,=H'4'              RETURN WITH STOP
         B     EXIT                     RETURN INDICATING MODIFY
TIMEEXPD EQU   *
         LA    R15,L'TIMEVARC           BUILD LENGTH OF VARIABLE
         STH   R15,VARLEN               BUILD LENGTH OF VARIABLE
         MVC   MODDATA(L'TIMEVARC),TIMEVARC BUILD CLIST VAR
         LA    R1,PARMLIST              POINT TO PARMLIST
         LINK  EP=INSØ7Ø,ERRET=BYPVAR2  CALL CLIST VAR CREATION
BYPVAR2  EQU   *
         TM    DCBOFLGS,DCBOFOPN        IS SYSPRINT OPEN
         BZ    NOCLSTM                  BYPASS SYSPRINT
         PUT   SYSPRINT,TIMECM          SHOW TIME EXPIRE
         CLOSE SYSPRINT                 CLOSE SYSPRINT
NOCLSTM  EQU   *
         MVC   ERROR,=H'4'              RETURN WITH TIME EXPIRE
```

```
         B     EXIT                      RETURN INDICATING MODIFY
EXITRC12 EQU   *
         MVC   ERROR,=H'12'              RETURN WITH ERROR
         B     EXIT                      RETURN INDICATING MODIFY
EXIT     EQU   *
         ESTAE Ø                         CANCEL ESTAE EXIT
QUICKOUT EQU   *
         LH    R15,ERROR                 GET RC
         EXITR RC=(R15)                  RETURN WITH RC
* TIMER EXPIRATION EXIT ROUTINE
TIMEEXIT EQU   *
         USING *,R15                     ADDRESS TEMPORARILY
         SAVE  (14,12)                   SAVE REGS
         BALR  R12,Ø                     SET UP BASE
TIMEBASE EQU   *
         L     R15,TIMEOFFS              SET UP BASE OFFSET
         SR    R12,R15                   SET UP REAL BASE
         DROP  R15                       USE STANDARD BASE
         POST  TIMEECB                   POST WAIT COMPLETED
         RETURN (14,12),RC=Ø             RETURN
```

Editor's note: this article will be concluded in the next issue.

Nils Plum
Systems Programmer (Denmark)

March 1997 – December 1999 index

Items below are references to articles that have appeared in *TCP/SNA Update* since March 1996. References show the issue number followed by the page number(s). All these back-issues of *TCP/SNA Update* can be ordered from Xephon. See page 2 for details.

3174	24.28-30, 27.22, 27.48
3720	25.24
3745	25.24-27
Address translation	36.3-18
AnyNet	34.6
APPN	26.27, 32.16
ATM	32.17, 34.61-63
Auditing	28.6-12
Automation	33.3-11
Bind	26.23-26
Buffer pool statistics	31.3-8
CICS	27.30
Clustering	36.32-42
CMIP alerts	26.3-17

DLSw 29.13-15, 32.19, 34.5
Dynamic line updates 27.51-60
Dynamic reconfiguration 26.17-22
Encapsulation 34.4
Enterprise printing 35.29-37
File transfer 24.6-28
FRAD 32.17
Frame Relay 34.3
FTP 28.14-53, 30.11, 33.45-47
Generalized Trace Facility (GTF) 21.30
Half-Session Control Block (HSCB) 21.3-29
High Performance Routing (HPR) 27.10-14
HPR 34.4-5
IBM enterprises 29.8-20
IMS 27.31
Independent logical units 26.23-38
Integration 34.3-7
Internet 30.3-8
IP Version 6 33.21-25
ISTCOSDF 28.12-14
ISTRACON 25.51-59
JES nodes 25.35-38, 27.31
LAN 29.8-20
Load balancing 36.32-42
Logon Mode table 32.27-39, 33.25-44
LOSTERM 27.60-61, 28.57, 29.21
LUGROUP 24.3-6
LUSEED 24.3-6
MAC 29.11
Maintenance 33.3-11
Management 29.22-26
MVS system symbols 29.3-7
NCP 25.10-35, 26.31, 27.14-50, 29.46-59, 30.34.52, 31.16-18
Net/Master 25.34
NetView 25.35-38, 27.3-10, 28.14-53, 32.5-16
NetView Distribution Manager 26.38-59
NetView Session Monitor 25.34
Network console 32.3-16, 33.11-21, 34.7-22
Network management 23.17-56
NMVT 24.30-45
OMEGAMON for CICS 27.16
OS/2 32.22-26
OSI 35.9-13
OSI BER 26.4-8
OSPF 29.9
Pacing 25.21-24, 26.27, 27.33
Performance 27.14-50, 31.10-16
QLLC 32.40-41
Response times 27.22
RIP 29.9
S-HTTP 29.18-19
SAS/CPE 28.6-12
SCO Unix 28.54
Security 30.3-8
SMF 34.23-37, 35.13-28, 36.19-31
SMTP 29.27-46, 30.12-31, 31.19-48, 32.42-57, 33.48-63, 34.38-60, 35.38-67, 36.43-62
SNA 32.16-21, 32.39-41, 34.3-7
Sockets 31.8-9
SOLVE:Netmaster NTS 26.17-23
TCP/IP 28.53, 30.9-10, 32.16-21, 34.3-7
TN3270 34.6-7
Token Ring Network 25.13, 25.27-29
TSO 32.22-26
Tuning 31.10-16
USS tables 28.53, 32.27-39
VBR 30.53-59
VLAN 29.10-13
VTAM 33.3-11
VTAM applications 28.3-6
VTAM configuration restart 25.3-10
VTAM constants 25.51-59
VTAM exits 24.45-55
VTAM monitor 35.3-8
VTAM session termination 27.61-63
WAN 30.31-33
Web server 31.48-63
X.25 28.54-57

TCP/SNA news

Tivoli has announced Version 1.3 of its NetView for OS/390, along wth NetView Performance Monitor for measuring network response time, network utilization, and traffic statistics. It's also started shipping its previously-announced Tivoli Service Desk for OS/390 1.2.

Using a new NetView Management Console, Version 1.3 manages both TCP/IP and SNA networks from a single console.

It reports both TCP/IP and SNA network to the service desk for problem tracking and resolution. Version 1.3 includes an SNMP Management Information Base (MIB) compiler, said to manage any vendor's networking hardware while reducing problem detection time.

Performance Monitor 2.5 combines performance tracking and reporting for both SNA and TCP/IP networks. It has a new GUI and claimed faster installation and depicts performance in real-time graphically, identifying potential problem areas before they can impact business.

When response time or utilization thresholds are exceeded, it sends notification to NetView for corrective action.

For further information contact:
Tivoli Systems, 9442 Capital of Texas Highway North, Arboretum, Austin, TX 78759, USA.
Tel: (512) 436 8000.
URL: http://www.tivoli.com.

* * *

Candle has announced immediate support for OS/390 Version 2.8 in all relevant products. It also announced expanded participation in the IBM SystemPac programme.

New is OMEGAMON II Version 500 for VTAM with a range of new features including TCP/IP analysis, and new flexible user profile controls.

OMEGAVIEW II for the Enterprise Version 200 and OMEGAVIEW for 3270 Version 300 get a simplified architecture and claimed higher performance.

SystemPac is available now for OMEGAVIEW II for the Enterprise, the integration component to link information from CCC on and off the OS/390 platform, and OMEGAVIEW for 3270, the VTAM-based component for integrating alerts and other information from underlying Candle and other systems management tools.

For further information contact:
Candle, 2425 Olympic Blvd, Santa Monica, CA 90404, USA.
Tel: (310) 829 5800.
Candle Services, 1 Archipelago, Lyon Way, Frimley, Camberley, Surrey, GU16 5ER, UK.
Tel: (01276) 414700.
URL: http://www.candle.com.

* * *

IBM has announced Version 3.1 of its DCE for both AIX and Solaris, providing TCP/IP remote commands on the AIX version.

For further information contact your local IBM representative.