The importance of systems management for a Parallel Sysplex

by P. Johnson

IBM S/390® Parallel Sysplex™, a multisystem parallel processing environment, provides benefits in terms of reliability, availability, and the total cost of computing. These benefits, however, bring about a systems management challenge because of the increased number of address spaces that need to be managed. This paper describes how CICSPlex® System Manager (SM) for MVS/ESA™ provides simplified systems management of multiple CICS® regions within a sysplex environment. CICSPlex SM provides workload-sensitive balancing of CICS transactions, a single-system image for CICS operations and monitoring, and general-purpose thresholds for resource conditions within CICS. It also describes how CICSPlex SM is integrated with the MVS workload manager, the MVS automatic restart manager, and automation products such as NetView® and the NetView resource object data model (RODM).

This paper traces the evolution of CICS* (Customer Information Control System) from its beginning to its present-day configuration within the IBM System/390* (S/390*) Parallel Sysplex*. After a brief introduction to the systems management tools provided by CICS and the limitations of their use in the Parallel Sysplex environment, the functions of the CICSPlex* System Manager (SM) for MVS/ESA* (Multiple Virtual Storage/Enterprise Systems Architecture) are described, with examples of how they are integrated with other components of the Parallel Sysplex solution.

Systems management is defined by the IBM Open Blueprint* as a high-level set of applications required by an enterprise running a computer complex. These applications are: business management, change management, configuration management, operations management, performance management, and problem management.

These applications cover all aspects of running an enterprise-wide computer complex with multiple platforms from multiple vendors: the integration of new versions of system, vendor, and application code, the redistribution of resources, the day-to-day operation of the systems, the measurement of system performance, the projection of trends and bottlenecks, the detection and resolution of problems, and the execution of many other tasks that allow computing systems to process the applications that really matter—the business applications themselves. All of this must be achieved, whether manually, with some automation, or with complete automation.

Evolution of CICS

When CICS was originally introduced, transaction processing needs were significantly different from those of today. Then the transaction processing needs of a business were provided by a single CPU running a single CICS region. Networks of terminal devices were in their infancy, consisting of hundreds rather than tens of thousands of resources. The CICS system was normally started "cold" in the morning and

then shut down in the evening for batch processing or software maintenance.

As the evolution of CICS progressed, and the demands of businesses increased, the limitations of the single CICS address space began to be reached. Increasing numbers of attached terminals and increasing numbers and sophistication of applications all contributed to the consumption of storage within the CICS region, giving rise to virtual storage constraints.

At the same time, CICS could not fully utilize the power provided by the introduction of multiprocessor machines. It ran all its tasks under a single task control block (TCB); thus CICS, although it provided a much faster dispatching mechanism than MVS and was more appropriate for transaction processing, was not able to dispatch its work on multiple TCBs.

The requirements for availability of transaction systems were also evolving. No longer just the tool of the "back office," transaction processing was now being placed in the hands of the customer (for example, a bank ATM). The availability of CICS could therefore be critical to the survival of the business. Transaction processing systems were also in extensive use in businesses where a large amount of the company's assets were at risk every day. A typical example is a stockbroker system. Loss of the transaction processing system could mean the loss of millions of dollars. The failure of a CICS region, causing complete loss of transaction processing ability for even a few minutes, was no longer acceptable.

Birth of the CICSplex. In response to these growing needs, CICS introduced the concepts of distributed transaction processing, function shipping,[1] and transaction routing. These were built upon the facilities of multiregion operation (MRO) and intersystem communication (ISC), and the ability to use individual CICS systems for a specific functional need. Terminal-owning regions (TORs), application-owning regions (AORs), and file-owning regions (FORs) were introduced and the CICSplex was born (see Figure 1).

Although the introduction of the CICSplex has solved the problems of virtual storage and the utilization of multiprocessor power, the availability issue still remains. The failure of a TOR causes the loss of system availability, because its connected terminals are lost. Static routing to application-specific AORs prevents one application from affecting the availability of all applications. However, failure of the AOR causes the loss of availability of its application to all users. If the FOR fails, then access to its data is lost to all applications.

In response, users partition their data among several FORs and partition their network among several TORs in an attempt to minimize these effects. However, new applications that require access to other applications or data lead to the "spaghetti-like" network that has become familiar today.

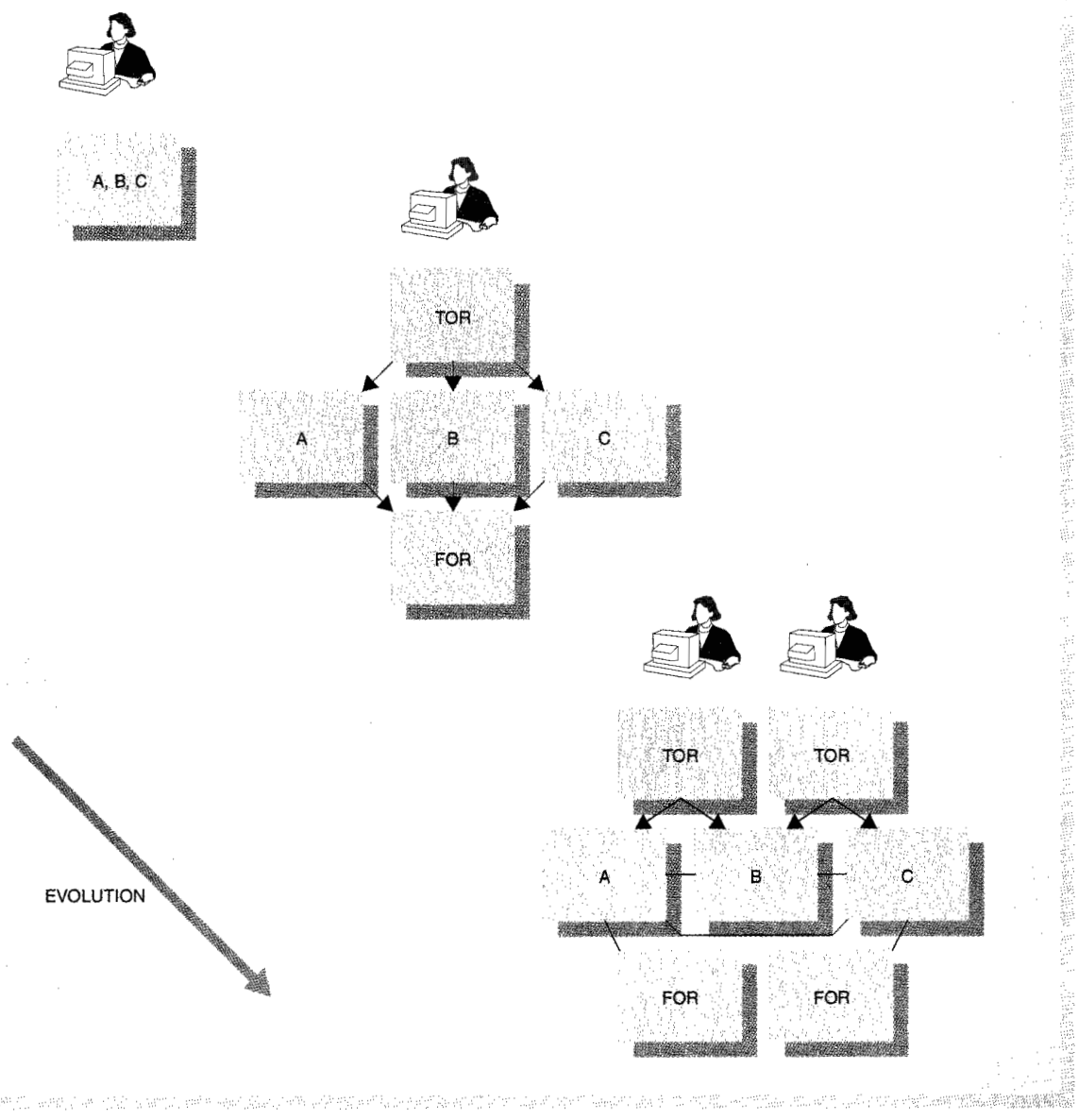
To compensate for the problems caused by single failures, and to cope with peak demands on a given processor, companies over-configured their systems. The cost of computing on S/390-based hardware was also becoming a critical issue, and many companies considered cheaper technology.

CICSplex in a Parallel Sysplex. It was in this environment that the Parallel Sysplex was conceived. The problem of failure at a single point preventing the access of data is eliminated by sharing data using the coupling facility. The VTAM* (Virtual Telecommunications Access Method) generic resource eliminates the single point of failure problem for network access to TORs and provides a single logical point of "log on" for the end user. Application availability is guaranteed by the ability to dynamically route transactions to multiple AORs. The dynamic allocation of TORs and AORs, along with data sharing, also solves the over-configuration problem, as work can be dynamically balanced across the sysplex in real time. The introduction of CMOS (complementary metal-oxide semiconductor) technology has totally changed the cost of computing.

The commonly proposed configuration within a Parallel Sysplex is displayed in Figure 2. Looking across the figure, we see why this configuration is called "horizontal striping." We have three applications available (A, B, and C). Identical copies of TORs and AORs, with the same resources and access to data, run in each MVS image. Any TOR can be accessed through VTAM generic resources, providing session balancing. Any AOR can run any of the applications available in the CICSplex. Dynamic routing from TORs to AORs is on an any-to-any basis. Each MVS image is a copy of the other (that is, the address spaces and MVS images are clones of each other).

The cloning model and associated hardware and software allows catering for peaks in demand (such as at Christmastime) by adding processors, rapidly cloning address spaces, and dynamically reconfiguring

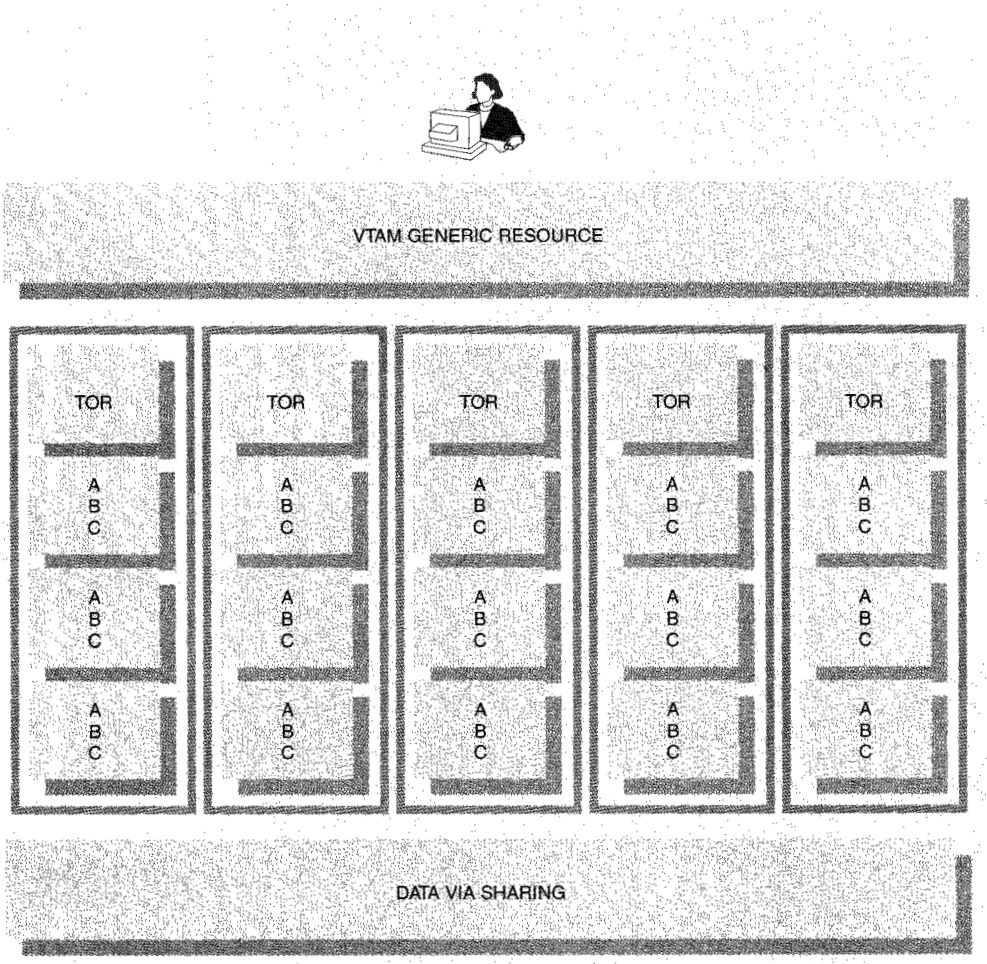
Figure 1 Evolution of the CICSplex. In the figure A, B, and C represent user applications.



the workload without interruption to existing service. The additional processors could be leased for the peak period, rather than purchased, eliminating the long-term investment of capital in hardware previously required. The recent introduction of software packages such as CICS Transaction Server for OS/390* (which integrates CICS and CICSPlex SM) has further reduced the software costs of managing such an environment.

A more often-used configuration has clusters of CICS regions (Figure 3). Clustering could be done to partition systems for development, quality assurance, and production. Figure 3 shows the same three ap-

Figure 2 Horizontal striping of applications

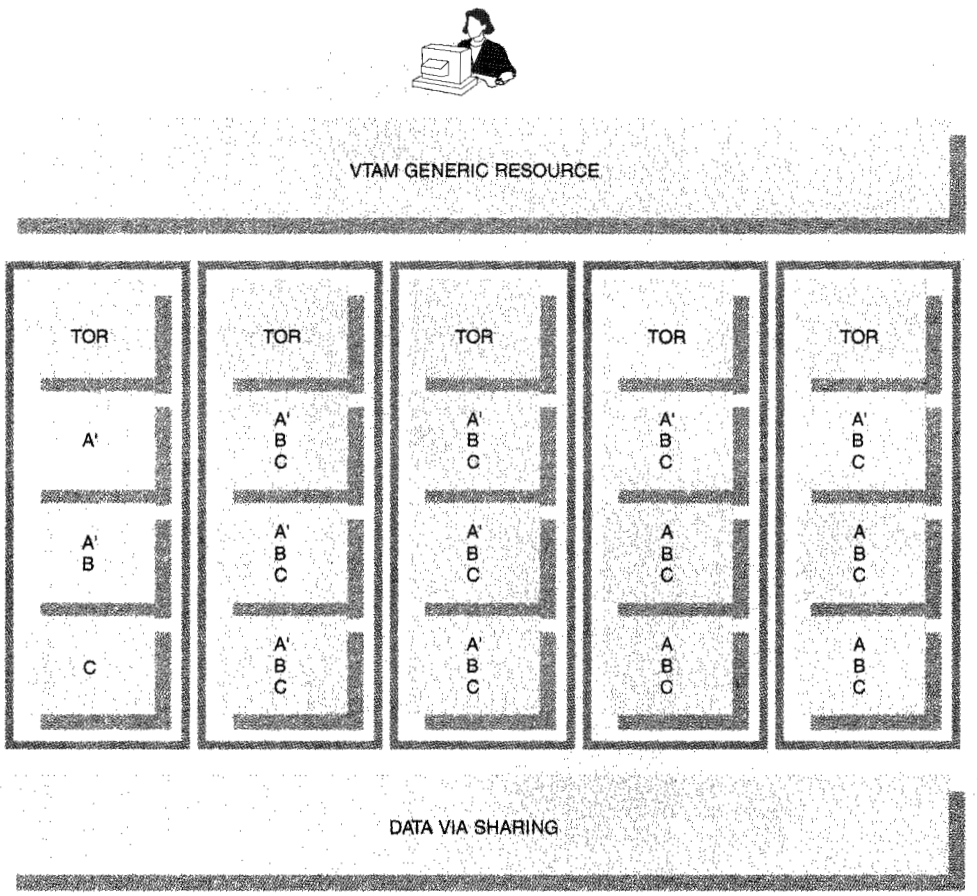


plications as Figure 2; however, a new version of A (A′) has been introduced in some regions. Separation of certain applications may also be desirable, for example, some applications may be capable of bringing down a CICS region due to poor application design, or different versions of the same application may be in use by different sets of users.

Of course, many changes to the CICS product have been made in order to execute within, and maximize the potential of, the Parallel Sysplex.[2] In addition to the subsystem enhancements, there are migration issues that must be considered by the systems and application programmers when moving to a Parallel Sysplex environment. Whether simple or complex, the move needs to be planned. Some of these issues are discussed in this paper.

Systems management tools. Although many of the issues affecting CICSplexes have been resolved, the increase in the number of CICS regions has brought a new set of problems. Each region must be managed and transactions must be dynamically balanced within the CICSplex. Automating the running of the CICS regions, tracking the various components of a given transaction instance throughout the CICSplex, and the dynamic nature of the number and physical location of resources all contribute to increased com-

Figure 3 Clustering of applications



plexity for the systems programmer, operator, and help desk personnel. Of course, these same problems exist in traditional CICSplexes. The problem is compounded by the increase in the number of CICS regions being managed and the dynamic nature of the configuration.

The key problem is that we now have many address spaces to manage and the physical location of these CICS resources may change. However, the only tools to manage them are the single-address-space tools provided by CICS.[3] Similarly, third-party vendor applications act on a single address space, or at best provide a single point of control but not a single-system image.

In order to effect a change to the running CICSplex we need to know where the resources reside (that is, we need a topology map). This information is typically contained in an operations workbook that is subject to change when new applications are moved, changed, or added. Once we know the locations, we must then sign on to each affected CICS region and issue the appropriate command. Should the location or number of the resources change, the operations workbook must be updated. If the configuration is

as in Figure 2 the location of resources is simpler; they exist in all clones. However, there is still the problem of multiple copies of the applications. In the more typical clustering environment of Figure 3, full knowledge of the resource topology is required in order to navigate to the appropriate regions.

The problems of multiplicity and dynamically changing locations cannot be overemphasized. A simple change in application code, followed by loading the new code into CICS storage, is straightforward if there is only one region. It is quite another thing if there are 30, or even hundreds, of regions. Prior to the invention of CICSPlex SM, it could take close to an hour to achieve this apparently simple task. Using CICSPlex SM, it can be achieved with a single command in seconds.

It is even more difficult to track a transaction thread through a CICSplex. Each thread, although executing the same transactions, may be distributed differently each time it is run due to dynamic routing. To track it we could write down the relevant information, capture screen images, or have multiple windows open at the same time displaying fragments of information. With CICSPlex SM this correlation of information is simply achieved.

What is required is a tool that provides access to the data without regard to the physical location or the number of instances of the resources, and is sensitive to the dynamic nature of the CICSplex. User interactions with the tool are then independent of those properties. The tool must provide a single-system image for systems management. CICSPlex SM for MVS/ESA meets these requirements.

CICSPlex System Manager for MVS/ESA V1R2

CICSPlex SM provides enterprise-wide single-system-image systems management for CICS/ESA* V3.3 and above, CICS/MVS* V2.1.2, CICS/VSE* V2.3, and CICS OS/2* V2 and V3. It is available as either a stand-alone product or as a component of the CICS Transaction Server for OS/390. In this paper we concentrate on the capabilities it provides for the Parallel Sysplex environment.[4]

In Figure 4 we see a typical Parallel Sysplex implementation similar to Figure 2, but with the CICSPlex SM address spaces added.

The principal components of CICSPlex SM are:

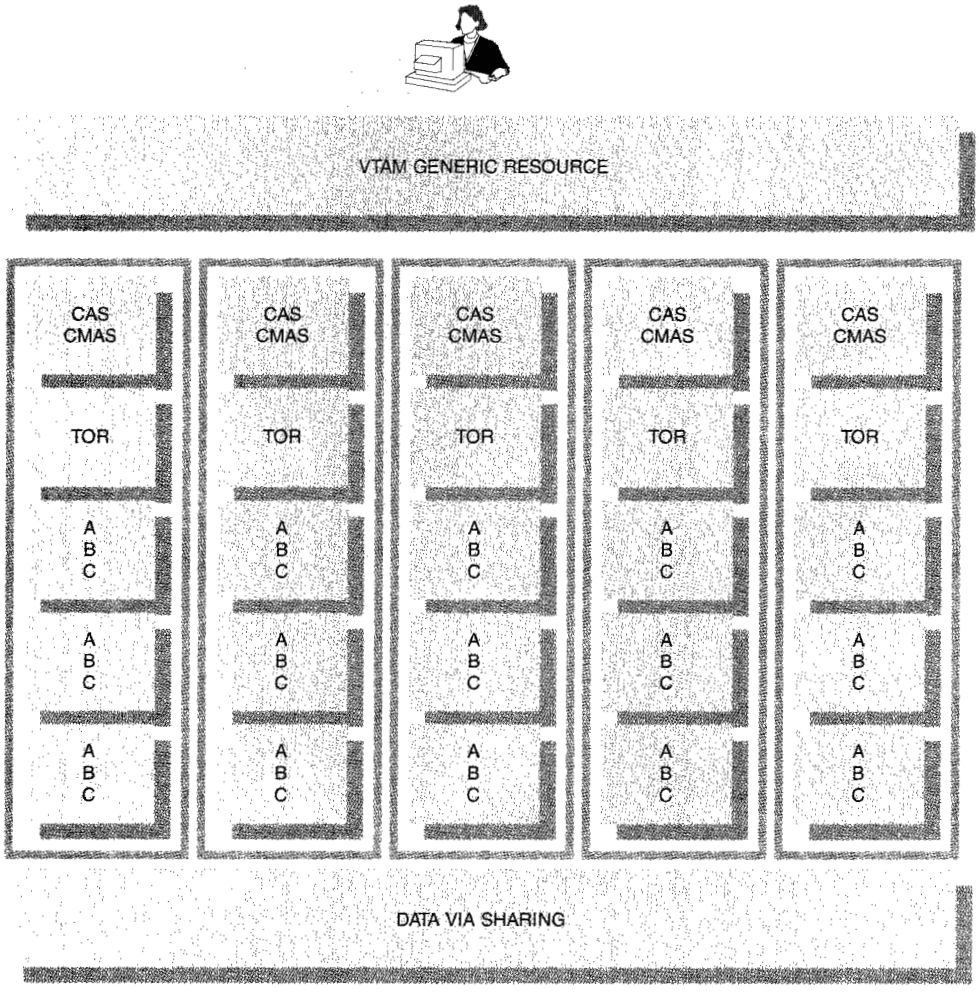
- The coordinating address space (CAS), which provides the connection point for the TSO (Time Sharing Option) end-user interface (EUI)
- The CICS managing address space (CMAS), which provides most of the CICSPlex SM function, including the single-system image
- CICSPlex SM agent code, which resides in user CICS systems that are to be managed by CICSPlex SM. These CICS systems are referred to as managed application systems (MASs).

Coordinating address space component. This component provides a single point of control for the CICSplex. A TSO session on the MVS image local to the CAS connects to it and obtains access to all CICSplexes that this CAS, or another CAS directly connected to this one, knows about. Communication from TSO to local CAS is cross-memory communication;[5] CAS-to-CAS communication is via LU 6.2.[6] For complete management from any CAS, all links between all CASs (i.e., $n[n-1]$ links) must be defined. A CAS provides a single point of control but *not* a single-system image, which is provided by the services within the CMAS.[7]

CICS managing address space component. Internally CICSPlex SM is similar to CICS, i.e., its functions are provided via domains with associated actions.[8] The CMAS component provides basic infrastructure services such as message, trace, and program call. Locking services and event notification are also provided to enable event-driven management of the CICSplex. This greatly reduces the hardware cycles executed by the management code, thereby maximizing the amount available for user application code.

Data space acquisition, extension, and management services are provided for various data types. These data types range from simple storage blocks to more complex data structures such as queues and lists. The management services include addition, deletion, merge, sort, and search facilities. Data spaces provide the ability to efficiently contain and communicate the large amounts of data that could be involved when a query is executed. (Consider the amount of data that a query of active transactions could generate, if issued for 300 CICS systems.) Data spaces also help minimize the execution time overhead within the managed application systems.

Figure 4 CICSplex with CAS and CMAS address spaces added



Data spaces are also used to shield user applications from management code failure. Consider, for example, dynamic transaction routing. It is essential that failure of the management code should not seriously affect the dynamic routing of application code. For this reason, the data required to perform workload management are kept in a data space that is accessible by the routing code in the event of CMAS failure. (The data spaces are actually owned by a limited-function address space started by the CMAS during initialization).

Requests are distributed by a communications component, built upon CICS MRO/ISC (multiregion operation/intersystems communication) or cross-memory communication, as appropriate. It provides logical connection services without regard to the physical location of the originator or target system(s).

Routing of requests is based on an active CMAS-to-CMAS topology map, and routing will occur through the set of CMASs that have the least number of "hops" between source and target. There is therefore no need to connect every CMAS to every other CMAS. The communications component also identifies to the CAS the CICSplexes that this CMAS is able to manage. A request can be routed from the CAS to the nearest CMAS that manages this CICSplex and its communications component can then distribute the request across the CMAS-to-CMAS network.

A real-time distributed topology component tracks the location of every CICS system and its resources within the CICSplex. Communications and topology allow other components to direct and distribute requests to both local and remote systems without regard to their physical location or the number of instances to be managed. The components that provide operations, monitoring, real-time analysis, dynamic workload management, and all the other functions of CICSPlex SM are built on these basic functions.

CICSPlex SM agent code. The CICSPlex SM CMAS code directs requests, extracts data, and processes information. Eventually the request for information or action will be passed via the communications component to the MAS agent code for execution. This code is responsible for accepting CICSPlex SM requests, transforming them into various EXEC CICS requests, and managing the responses from their execution. It is also responsible for detecting topology changes in the system and for interfacing to the dynamic routing code provided by CICSPlex SM.

A typical request flow. Consider a request from an end user on TSO for information about where some set of programs are currently defined within the CICSplex. The user may do this simply to identify where this program is currently defined, or in preparation for loading an updated copy of the program into CICS storage for execution. Suppose the program exists on all AORs, as shown in Figure 4.

The request goes from TSO to the local CMAS via cross-memory services. The local CMAS uses its topology component to find all CMASs that have CICS systems containing instances of the program. A single request is then made to the communications component, passing the set of remote CMASs (via CICS MRO/ISC or MRO/XCF [cross-MVS communications] in this case). At each target CMAS, through its topology component, the CICS systems managed by this CMAS are found and the target MASs identified. The request is then forwarded to each target MAS agent (via cross-memory communication), which issues an EXEC CICS INQUIRE PROGRAM request and obtains the response data. Each target MAS agent puts the response data into a data space queue and passes it back to the CMAS code. When responses have been received from all local MAS agents, the resultant queues are appended together, transformed, and transmitted back to the originating CMAS via the communications component. The originating CMAS appends together the result queues from the various CMASs, then passes them back to the TSO EUI code for presentation to the user.

Single-system image. Single-system image is supported across all the systems management functions provided by CICSPlex SM. In fact, CICSPlex SM provides the ability to define a CICSplex in terms of its constituent CICS systems. This is the default single-system image. One can, however, define subsets of CICS regions within the CICSplex to reduce the scope of the single-system image. These subsets can be overlapping in content. Subsets could include all the TORs in the sysplex, all the AORs in the sysplex, all the CICS regions on a particular MVS image, or whatever logical groups are required.

By restricting the scope of the actions at the interface, the end user can choose to affect any set of CICS regions within the CICSplex at the logical level. Should the location of the CICS systems change, (e.g., as a result of MVS automatic restart manager [ARM]), or the number of resources change (e.g., by adding further cloned regions) *no change in the interaction at the end-user interface is required.*

Single point of control. CICSPlex SM provides a single point of control to manage the CICSplex via a TSO end-user interface. It can be accessed on any MVS image that has an associated CAS available for connection. Information solicited from all the CICS systems in the enterprise can therefore be presented and acted upon from a single screen with a single command.

Single points of control can, of course, be used to support multiple concurrent users in multiple MVS address spaces. In this way, centralized or decentralized systems management can be provided according to the specific needs of the enterprise. Access to resources is controlled via RACF* (Resource Access Control Facility) or any security-access-facility-(SAF-) compliant security product at the appropriate levels of granularity.

Figure 5 Single-system image and single point of control

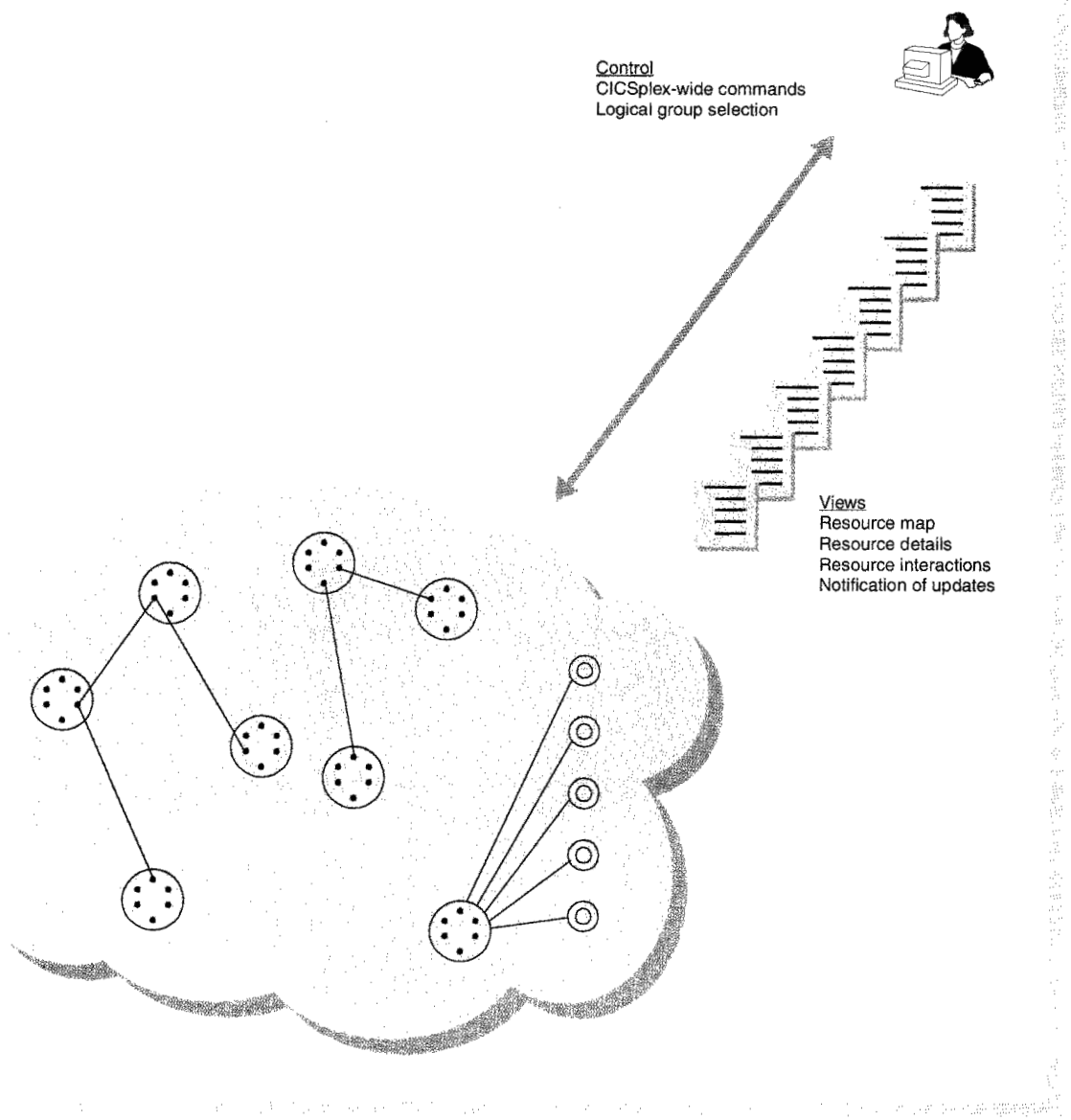
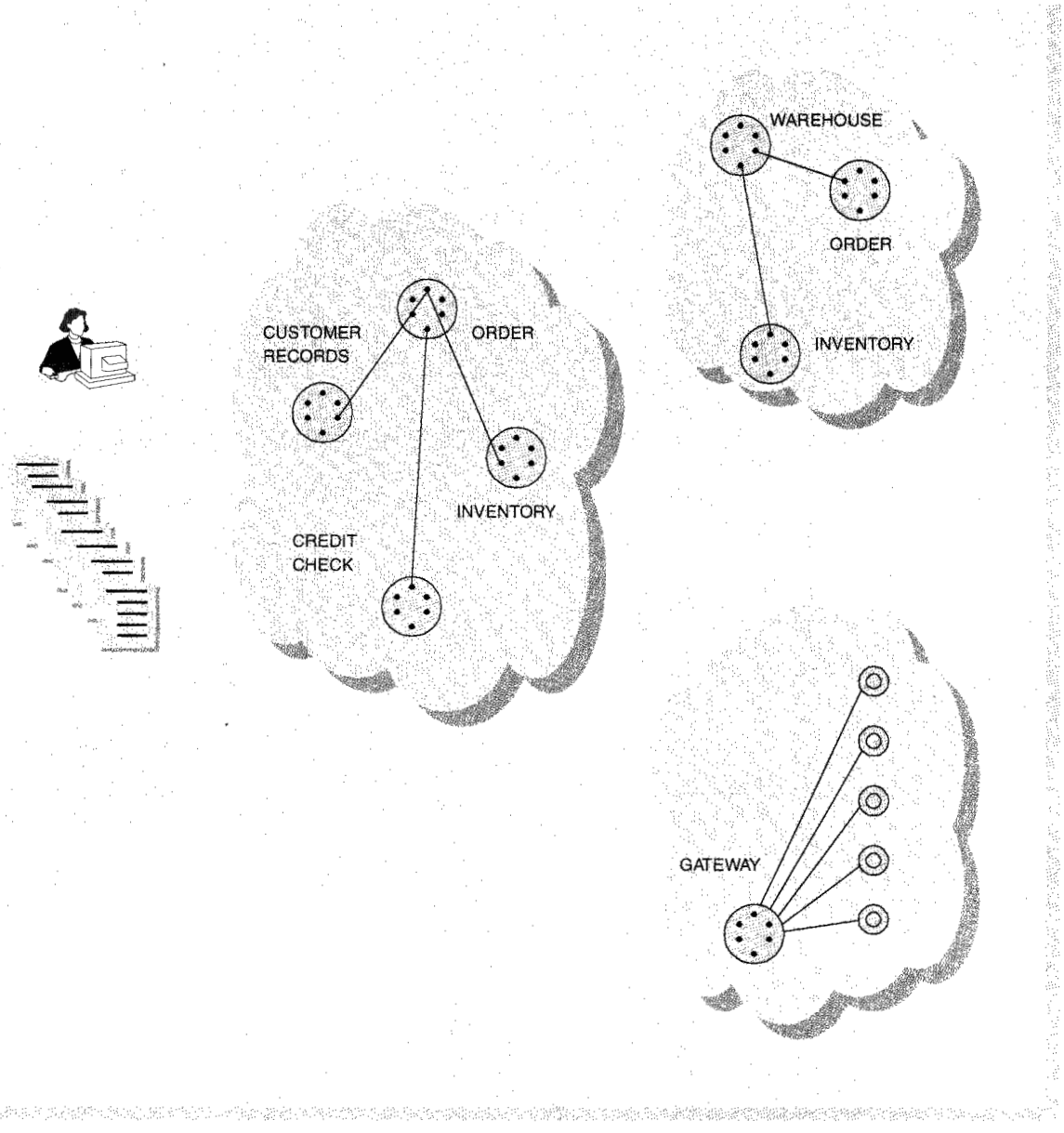


Figure 5 shows a CICSPlex SM user with a single point of control and a single-system image. The "cloud" represents a CICSplex, and the circles within it represent CICS regions. Contained within the regions are CICS resources. In Figure 6 the user has multiple, overlapping views of logical groups of systems. Here each cloud represents a single-system image, but with a smaller scope than the entire CICSplex.

Navigation between views of data. CICSPlex SM provides three levels of views for a given resource type: tabular, detail, and summary. Tabular views provide key data for each instance of a resource within the current scope of the request command. Detail views provide details of all attributes of a given instance of a resource. Summary views are generated from tabular views by performing spreadsheet-like oper-

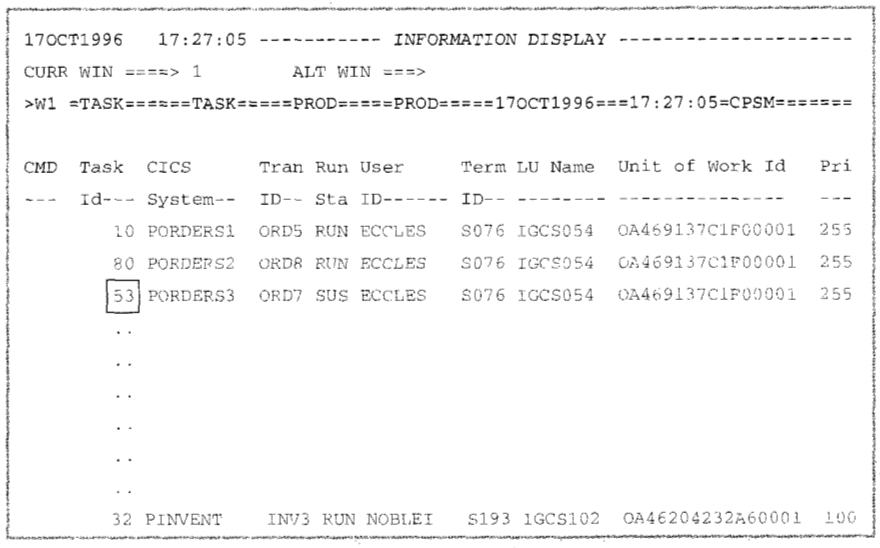
Figure 6 Logical groups of systems



ations on the rows of data. Summarization rules for individual attributes can be specified and summaries on specific columns can be generated.

Navigation between views occurs via hyperlink fields. Navigation can be within a resource type, as from tabular to detail, or across resource types, for example from transaction, to program, to program library. This might be used if a transaction successfully executes in some regions, but not in others. Identifying the target program libraries may uncover different libraries being used for loading the same program. Multiple views of several object types can also be displayed simultaneously on a single screen.

Figure 7 Operations TASK tabular view



```
17OCT1996   17:27:05 ----------- INFORMATION DISPLAY ---------------------
CURR WIN ====> 1        ALT WIN ===>
>W1 =TASK======TASK=====PROD=====PROD=====17OCT1996===17:27:05=CPSM=======

CMD  Task  CICS      Tran Run User     Term LU Name  Unit of Work Id   Pri
---  Id--- System--  ID-- Sta ID------ ID-- --------  --------------    ---
       10 PORDERS1  ORD5 RUN ECCLES   S076 IGCS054  OA469137C1F00001  255
       80 PORDERS2  ORD8 RUN ECCLES   S076 IGCS054  OA469137C1F00001  255
       53 PORDERS3  ORD7 SUS ECCLES   S076 IGCS054  OA469137C1F00001  255
        ..
        ..
        ..
        ..
        ..
        ..
       32 PINVENT    INV3 RUN NOBLEI   S193 IGCS102  OA46204232A60001  100
```

Figures 7 and 8 illustrate a typical help desk scenario. An end user calls the help desk because his or her application is "hung up." All that the user provides is the user identification. A request for all tasks associated with this user results in the display in Figure 7. It is apparent that part of the application is suspended in PORDERS3. The hyperlink from the task identification leads to the detail view for that specific region (Figure 8), where the cause is discovered. (SOS means "short on storage.")

This illustrates some of the power that CICSPlex SM provides. This time the application ran in PORDERS3. Next time, due to dynamic work balancing, it could run in any of the other regions allocated to the ORDERS application. A static map, such as an operations book, is no longer appropriate in this rapidly changing environment.

Operations. CICSPlex SM provides a single-system image; users can inquire and take actions on CICS resources throughout the CICSplex. All CICS resources and their attributes are supported by CICSPlex SM. Disabling a transaction within the entire CICSplex can be performed by a single command. Tracking down elements of a given transaction thread in the CICSplex is a simple matter with CICSPlex SM, as shown in Figure 7.

Monitoring. The monitoring component of CICSPlex SM provides equivalent views of all monitoring and statistics data from the CICS systems in real time. Information such as response time for transactions and usage for the dynamic storage area (DSA) across the CICSplex can be displayed.

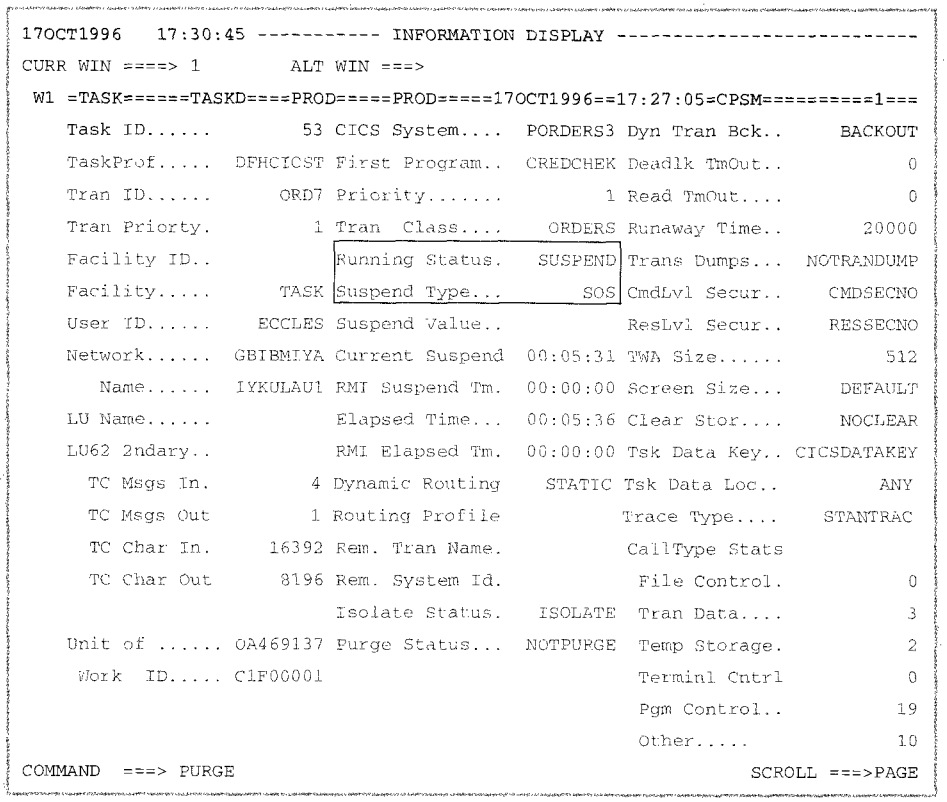
Real time analysis. CICSPlex SM is not limited to manual interaction, it also supports automation of the operation of the CICSplex. This is provided via the real-time analysis (RTA) component of CICSPlex SM.

RTA consists of the following components:

- System availability monitoring (SAM)
- MAS resource monitoring (MRM)
- Analysis point monitoring (APM)

All of these functions can cause manual notification via the CICSPlex SM end-user interface, a message to the MVS console, or an SNA (Systems Network Architecture) generic alert for detection (and resolution) via an automation product. As we shall see in

Figure 8 Operations TASK detail view



a following section, automation within CICSPlex SM itself is possible without resorting to external automation products.

System availability monitoring. SAM provides basic system "health" data for the CICSplex. By simply identifying to CICSPlex SM the standard times that the CICS systems are in operation, it monitors

- System unavailable
- Short on storage
- Maximum tasks reached
- Transaction or system dumping
- Potential stall situation within the region

for all CICS systems in the CICSplex. Rapid notification of any situations that might affect the availability of the CICSplex is therefore possible, with very little definitional effort.

MAS resource monitoring. MRM provides the ability to specify criteria based on any of the objects and their attributes within the CICSplex. These criteria can be combined to identify conditions within the CICSplex that might require action. This provides completely generalized state management of the CICSplex.

Analysis point monitoring. APM provides similar facilities to (and uses the same definitions as) MRM. The difference is in the way that events are created. MRM provides one event per instance, and APM produces one event per set of CICS systems.

Status probes. The final component of RTA supports user-written status probes that can contribute to the evaluation of the state of the running CICS systems. Status probes are programs that CICSPlex SM invokes at a user-specified interval. When invoked, a status probe returns the value "true" or "false" to RTA (via a CICS COMMAREA control block). If the value is true, an associated severity is also returned. This component allows RTA function to be extended to any information the user can access. Monitoring transaction "scratch-pad" data, database manager status, or distributed CICS systems on other platforms is therefore possible.

Automation within CICSPlex SM. The CICSPlex SM user can specify an action to be taken upon detecting a condition within the CICSplex. An application of this facility is testing the status of all the connections in the CICSplex. CICSPlex SM can be instructed to attempt to reacquire any connection on the CICSplex that goes out of service. This can be defined once and remain unchanged as new connections are added. Prior to CICSPlex SM, this type of checking was performed by problem-specific user-written software, rather than general-purpose software.

Another use of this facility might be to cause MVS ARM (automatic restart manager) to restart the CICS region according to ARM policy currently in effect. A region can be cancelled by RTA if the CICS system is currently registered to ARM. In this way, a diverse set of problems within CICS regions can be quickly detected and the affected regions shut down and scheduled for restart without delay.

Automation via NetView. For situations where the problem cannot be resolved internally by CICSPlex SM, NetView* or another automation product can be notified, via either SNA generic alert or console message. For message-based automation, the message is identified in the NetView message automation table and the standard interfaces can be used to resolve the event (MVS "modify" commands, etc.). This would require knowledge of the location of the resource. Automation can also be done using the CICSPlex SM API.

CICSPlex SM API. CICSPlex SM provides an API (application program interface) for programs written in C, assembler, PL/I, COBOL, and REXX (Restructured Extended Executor). The API is available in CICS, MVS batch processing, TSO, and NetView. In addition, a language binding is provided, by EXEC CPSM (CICSPlex System Manager), which is similar to the EXEC CICS language processed by the CICS translator. The API provides access to all the data available to an end user and also allows access to the events that occur within the CICSPlex SM product (e.g., events could be resource changes within the managed CICS regions, or generated by RTA).

The API programs can access data anywhere throughout the CICSplex and can be run in any MVS address space that has access to the CMAS, without change. One possible use would be to close application files prior to running in batch work. Invocation of these programs can be controlled via scheduling products such as IBM's OPC*/A (Operations Planning and Control/Advanced). These programs can also be used to detect and log changes to the CICSplex, start and stop systems when conditions change, vary active RTA policy, and any other task that the user wants to automate within the CICSplex.

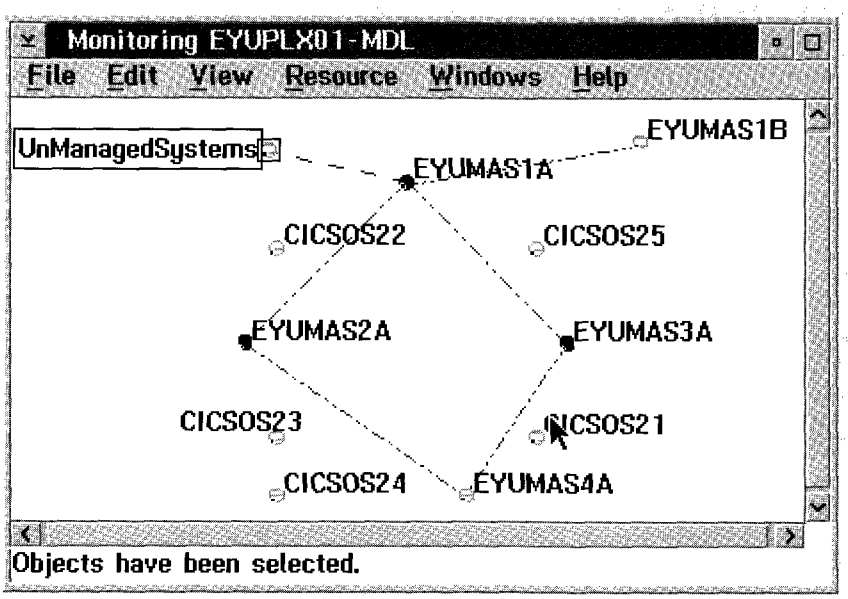
CICSPlex SM and NetView RODM. CICSPlex SM supports the population of CICS objects and their key status indicators (such as enabled or disabled) within the resource object data model (RODM) component of NetView. RODM is the strategic integration point in SystemView* and can be used to correlate CICS resources with resources from other subsystems within the sysplex. The user can identify to CICSPlex SM which objects are required. CICSPlex SM, via an agent within NetView, will dynamically update RODM as resources are installed or discarded within the CICSplex, or their status changes. CICSPlex SM uses the NetView MultiSystem Manager (MSM) to represent instances of the CICS objects.

These data can be viewed via the NetView graphical monitoring facility (NGMF) on an OS/2* (Operating System/2*) workstation. A sample view (Figure 9) displays the CICS systems being monitored and the status of the connections between those systems.

The default view provided by CICSPlex SM is a containment hierarchy. Within the top level aggregate is the CMAS and managed CICSplexes. Each CICSplex contains CICS systems. Each CICS system contains the types of resources managed, and within those the individual resource instances (Figures 10 and 11).

Once RODM has been populated with the resources, the user can build aggregates from them using the FLCBLDV tool provided with MSM. This allows the

Figure 9 NGMF connectivity map of CICSplex members showing the regions and status of the links between them



user to represent key business applications as aggregates on the NGMF screen (Figure 12).

NGMF provides facilities to set thresholds on the states of the monitored objects. When the threshold is reached, NGMF changes the color of the displayed object, thus providing color cues to the state of the underlying objects contained within the aggregate. Users can specify a single screen with a single icon that represents the state of all their key business applications. Should the icon turn red, then some problem has occurred and the cause must be determined.

The object causing the change could be many levels down the containment hierarchy. To aid quick navigation, NGMF provides a "fast path to failing resource" that goes directly to the object that caused the state change. By selecting that object, then choosing "show parents," the containment hierarchy can be displayed. This provides additional information as to the location of the resource that caused the failure.

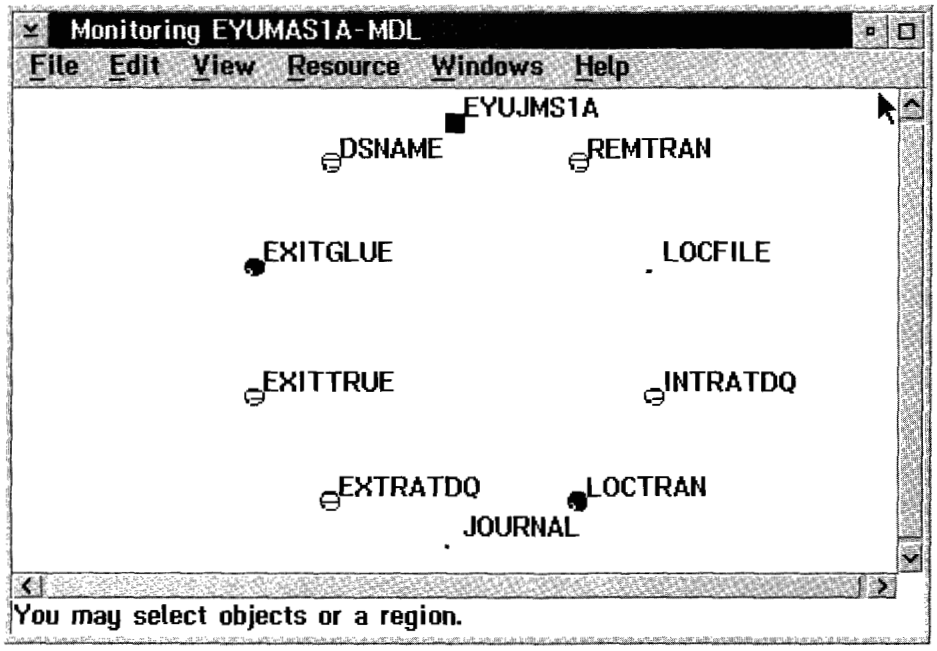
NGMF also provides a facility to execute commands in NetView. With this facility enabled, CICSPlex SM API REXX applications in NetView could be invoked, from NGMF, to resolve problems in the underlying CICSplex.

The utilization of graphical displays allows the operations personnel to rapidly detect, track down, and resolve problems, sometimes even before the end user is aware that a problem exists.

CICS dynamic workload management. CICSPlex SM provides dynamic transaction workload management for CICS. This function is central to the deployment of CICS on Parallel Sysplex technology.

The CICS transaction model that has evolved over time is the "pseudoconversational" model. It is somewhat different than the model used in other application environments. Conventional "conversational" programs remain in storage from initial invocation until termination, holding resources for much longer than necessary. When resources are held during the

Figure 10 Resource types being monitored for a CICS region



time that data entry and operator decisions are being made, other tasks may be blocked. The pseudoconversational model was developed to give the illusion of conversational interaction.

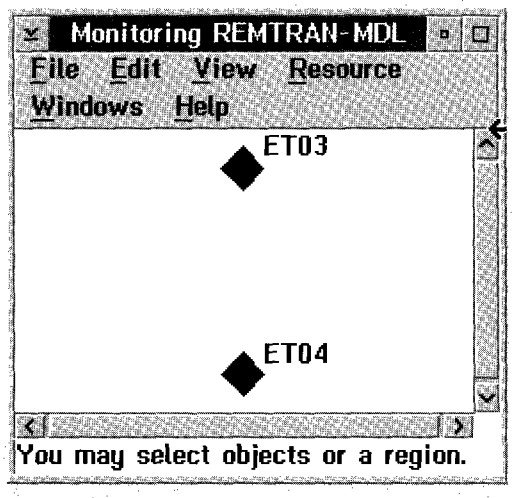
In this model, between interactions with the end user, the state data are kept separate from the program. Thus the program can be removed from main storage and its resources freed until the next interaction is initiated. The program is then brought back into storage, passed its state data, and executed. A single execution of a transaction could therefore result in multiple executions of the program, each program execution being a segment of the pseudoconversation. When the program finishes processing, the pseudoconversation is terminated.

CICS provided this function via EXEC CICS RETURN NEXTTRANID (tranid) COMMAREA (state_data) to execute the next segment, and EXEC CICS RETURN to end the pseudoconversation. Other methods of passing data between transaction segments evolved over time through EXEC CICS API extensions and other facilities.

As CICS evolved into CICSplex, this pseudoconversational technique allowed CICS transactions initiated in the TOR (where the state information was maintained) to execute application code in the AOR, with each segment of the transaction thread potentially executing in a different AOR. Several of the earlier programming techniques employed inhibited this potential, or at least reduced it, by saving state data in the AOR. This led to *affinities*,[9] as outlined below. Consequently, dynamic transaction routing has to tolerate any intertransaction and transaction-to-system affinities that may exist within the workload (unless the application provider removes such affinities).

As well as affinities, workload management must assess the health of the target AORs and their links,

Figure 11 Transaction instances being monitored for this CICS system



the probability of the transaction abending within a given region, workload separation criteria, and workload balancing.

Transaction affinities. Transaction affinities come about through various application programming techniques that were employed by developers before dynamic routing was available. These techniques cause data to exist in an AOR for longer than the task that created the data. An example would be using CICS main temporary storage to contain data, rather than passing the data via a COMMAREA control block for subsequent use by the transaction. Another example would be leaving data in the CICS work area (CWA) control block (Figure 13). Since the data remain in the AOR, disaster can occur if the next step in the application is routed to another AOR. The affinity can be associated with various attributes of the environment, for example a user identifier or a terminal. These affinity types and their lifetimes are reflected in Table 1.

Affinity lifetimes may be:

- Pseudoconversational, where the data exist only for the length of the CICS pseudoconversation (normally a short period of time)
- Sign-on (between sign-on and sign-off of a user to the CICS TOR)
- Log-on (from VTAM log on to log off)
- System (while the CICS AOR is active)
- Permanent, where once created, the data exist for all time (thankfully few exist)

Various programming techniques that utilized pseudoconversational behavior before CICS provided the ability gave rise to:

- Delimited affinities, where a given transaction "delimits" the affinity
- Specific identification of transactions that start and end an affinity

Any dynamic workload management solution must handle affinities and route to CICS AORs appropriately.

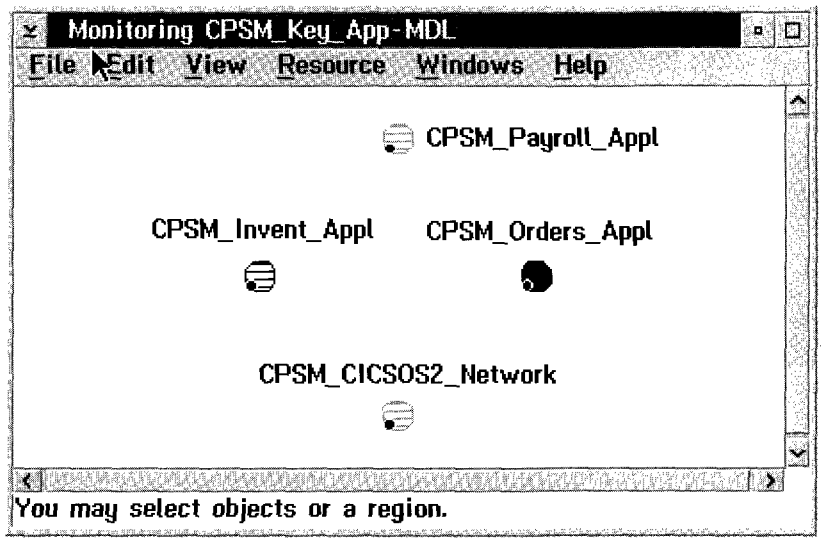
IBM provides the CICS affinities utilities to help in detecting affinities.[10] This has both on-line and off-line components. Another component of this utility is the affinity builder that creates batch definitions in order to simplify input into CICSPlex SM.

The affinities described so far are statically defined, i.e., only a relationship between the transaction names and their affinity is defined. CICSPlex SM also supports the ability to dynamically create and destroy affinities based on information accessible to the dynamic routing program. This requires customization of the CICSPlex SM routing program to utilize the *create* and *destroy* CICSPlex SM verbs. The common area might contain sufficient data for the routing program to deduce whether an affinity needed to be cre-

Table 1 Affinities and their corresponding valid lifetimes

Affinities	Lifetimes					
	Pseudoconversational	Sign-on	Log-on	System	Permanent	Delimited
User identifier	X	X		X	X	X
Logical unit name	X		X	X	X	X
Global				X	X	

Figure 12 Example of a user-built view that represents key applications and resources



ated or destroyed. The various techniques that could be employed are beyond the scope of this paper.

Health data and RTA. When routing transactions to CICS AORs, it obviously makes sense to send the work to the regions most capable of executing it. In the section on real-time analysis, we saw various conditions identified through systems availability monitoring that could cause a CICS region to be less eligible to process work. These conditions relate to the "health" of a region. The user can also specify an RTA event that can affect routing into an AOR. The full power of RTA can therefore be employed in the routing decision.

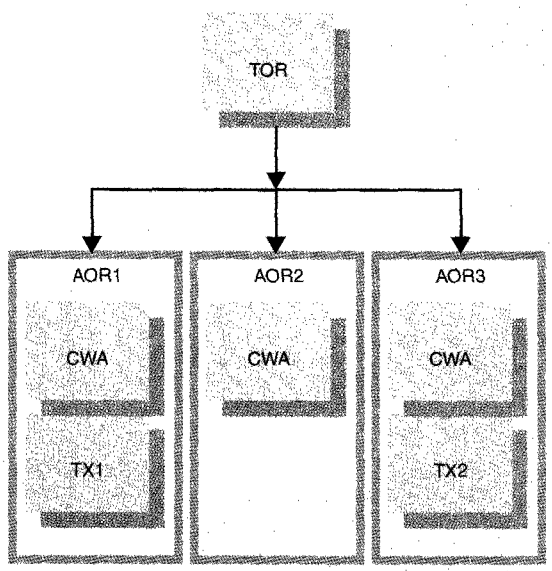
Link data. The types of links between the TOR and AOR are considered when choosing a target AOR in order to determine the relative cost of shipping a request. Weights are applied as follows, in increasing order: same region (AOR is TOR), cross region (AOR via MRO), cross MVS (AOR via MRO/XCF), and cross system (AOR via ISC).

Abend avoidance. An optional feature provided by CICSPlex SM dynamic workload management is to avoid abnormal termination (abend). We can route a transaction away from a candidate AOR if the transaction has abended within the AOR in the recent past. After a given time, CICSPlex SM will reactivate the AOR and send it a "sacrificial lamb" transaction to test whether the problem still exists. If no abend occurs, then the AOR is brought back "into the fold" and it will be reactivated for transaction routing.

This is achieved as follows. If a transaction abends within an AOR its probability for abending is set. From the abend probability an abend factor is calculated, based on the abend health and abend load values provided via CICSPlex SM workload management administration. Successful completions are then simulated for this transaction, which bring down the abend probability and associated abend factor. Ultimately the weight reduces to such a point that the AOR is chosen to route a real instance of the transaction, the sacrificial lamb.[11]

Workload separation. As we have seen, the user may want to partition the incoming work to various subsets of the potential AORs. This could be due to different versions of an application, separation based

Figure 13 Example of affinities



1. Transaction 1 (TX1) and Transaction 2 (TX2) pass data by using the CWA.
2. They are dynamically routed to different AORs.
3. The application fails!

on geographical location, or departmental activities. CICSPlex SM allows the user to partition workloads to different sets of AORs based on user identification, logical unit name, and groups of transactions. This is illustrated by the triple (userid, luname, trangrp) elements shown in Figure 14. The corresponding target scope, across which balancing is to occur for that triplet (AORSET) is denoted graphically by the ellipse surrounding the target AORs.

The transactions to be partitioned are contained within the transaction group (trangrp) also illustrated in Figure 14. Finally, Figure 14 illustrates that this routing knowledge can be limited to a given set of TORs within the CICSplex.

Workload balancing. All three criteria, health, link, and abend, are taken into account when identifying potential target AORs, as summarized in Figure 14. However, even though we have identified the potential targets, we have still to choose one for this transaction instance. CICSPlex SM provides two balancing algorithms, queue and goal. We next describe the implementation of these algorithms.

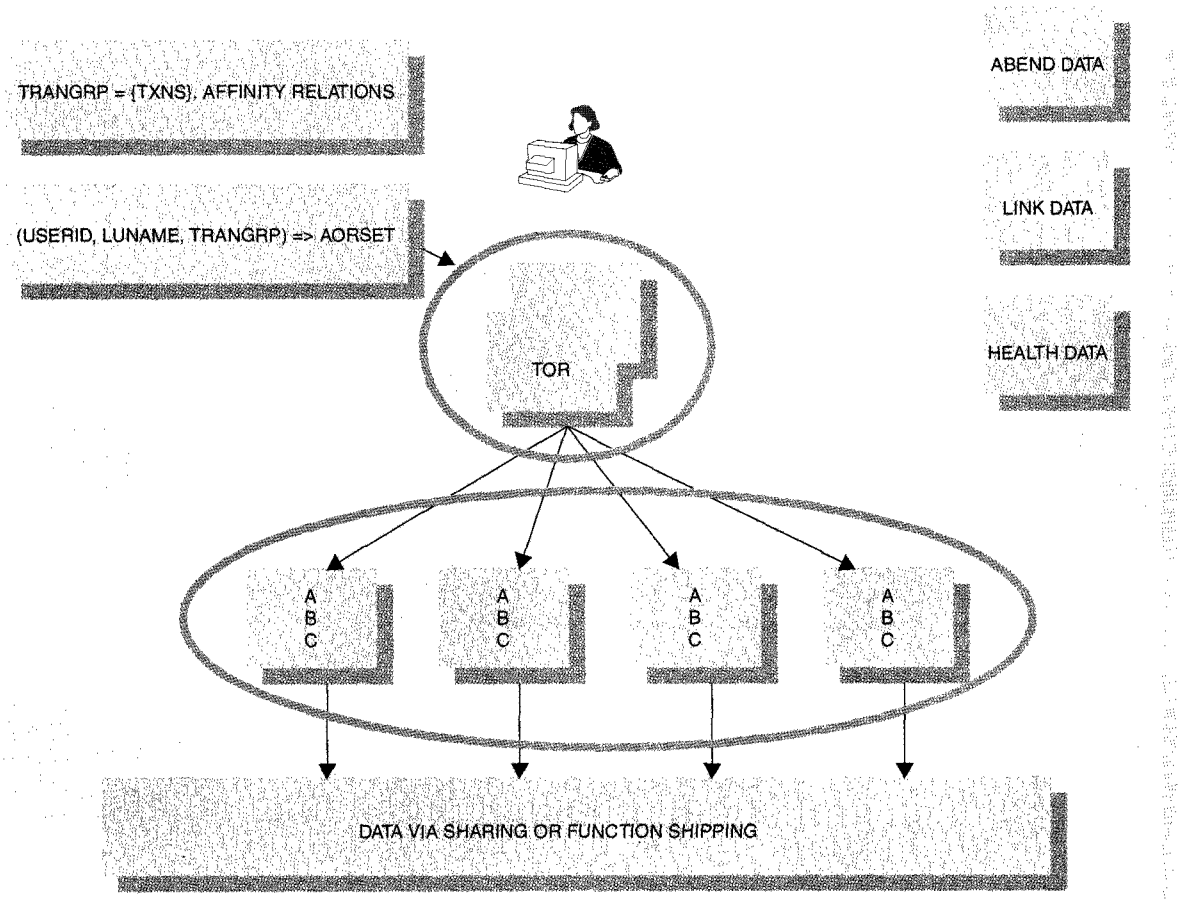
The queue algorithm works by calculating the ratio of the current task count to maximum task count. This corresponds to "load" in Figure 15. The health, link, and abend factors described earlier are also taken into consideration and a weight is calculated for each target AOR. The AOR with the lowest weight is chosen to be the target (assuming no affinity exists). Figure 15 shows an example of routing using this algorithm. AOR3 is rejected because of its "stall" state; AORs 1 and 2 because of nonzero abend probabilities. The values shown are for illustrative purposes only. The actual values used for the link factor are defined by the customer, and the values for the abend factor are calculated from abend probability, abend load, and abend health.

MVS workload management (WLM) in V5 provides the ability to schedule work within the sysplex based on goals, rather than system resources manager (SRM) policy. When running workload management in goal mode, CICSPlex SM provides dynamic transaction routing in collaboration with MVS WLM. For this to occur the MVS images must be in goal mode and the CICS TORs must be CICS 4.1. CICS performs the functions necessary to identify work to MVS WLM via performance blocks (PBs) as described by Banks.[1] It is the responsibility of CICSPlex SM to route transactions to AORs in such a way that MVS WLM can alter the dispatch priority of, and storage allocation to, the AORs in order to achieve the response time objectives specified.

Response time criteria for CICS transactions (and other address space types) are specified via MVS WLM policy. The brief description that follows is in terms of the components related to the CICSPlex SM goal algorithm.

For MVS WLM, a *service definition* per sysplex is created, within which service policies and classification rules are defined. Only one service policy can be active at one time; it defines the workload definitions and service class definitions that control the allocation of resources within MVS and the associated report classes and goals. Goals can be of various types: velocity, discretionary, or response time. Response time goals can be given either by average or by per-

Figure 14 CICSPlex SM dynamic routing model



centile. CICSPlex SM supports only average response time goals.

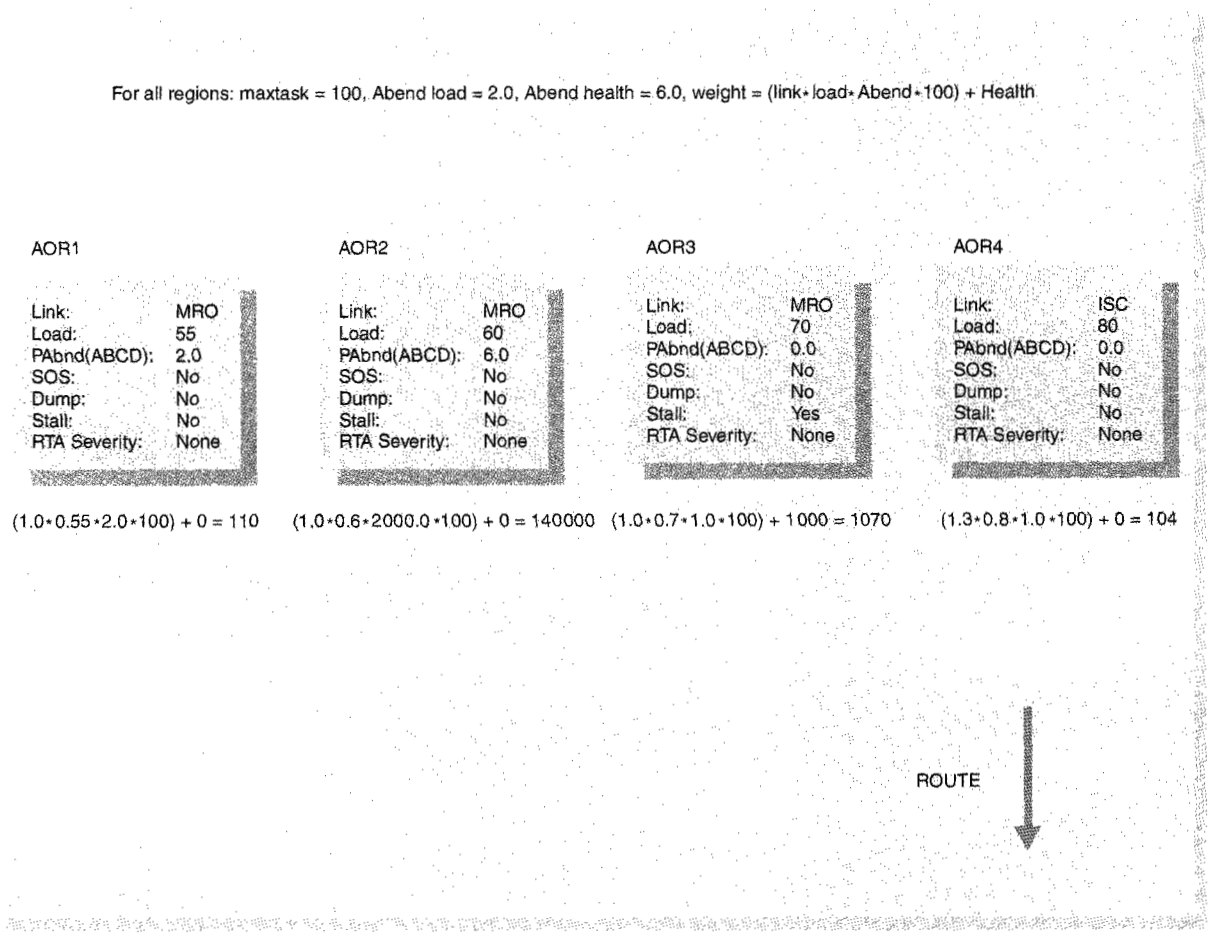
As a transaction enters the TOR, CICS classifies it and passes a service class token to the dynamic routing exit. This allows CICSPlex SM to obtain the response time goal and to calculate a performance index (PI) for the transaction. The PI is defined as:

$$\text{PI} = \text{average successful response time} / \text{average response time goal}$$

Transactions with a PI of less than 1 are therefore achieving their goal, and those with a PI greater than 1 are missing their goal. CICSPlex SM calculates this PI for all active service classes in the workload. This calculation is performed periodically, and when no new arrivals of a given service class occur, that entry is discarded (that is, we maintain only active service classes).

When CICSPlex SM allocates the service classes to the target set of AORs, it takes into account the relative frequencies of transactions within the active service classes. A service class may have more AORs allocated if the incidence of transactions in its service class greatly outweighs those in the other service classes. By ordering the service classes by PI and scaling the number of AORs to service classes by inci-

Figure 15 Routing using the queue algorithm



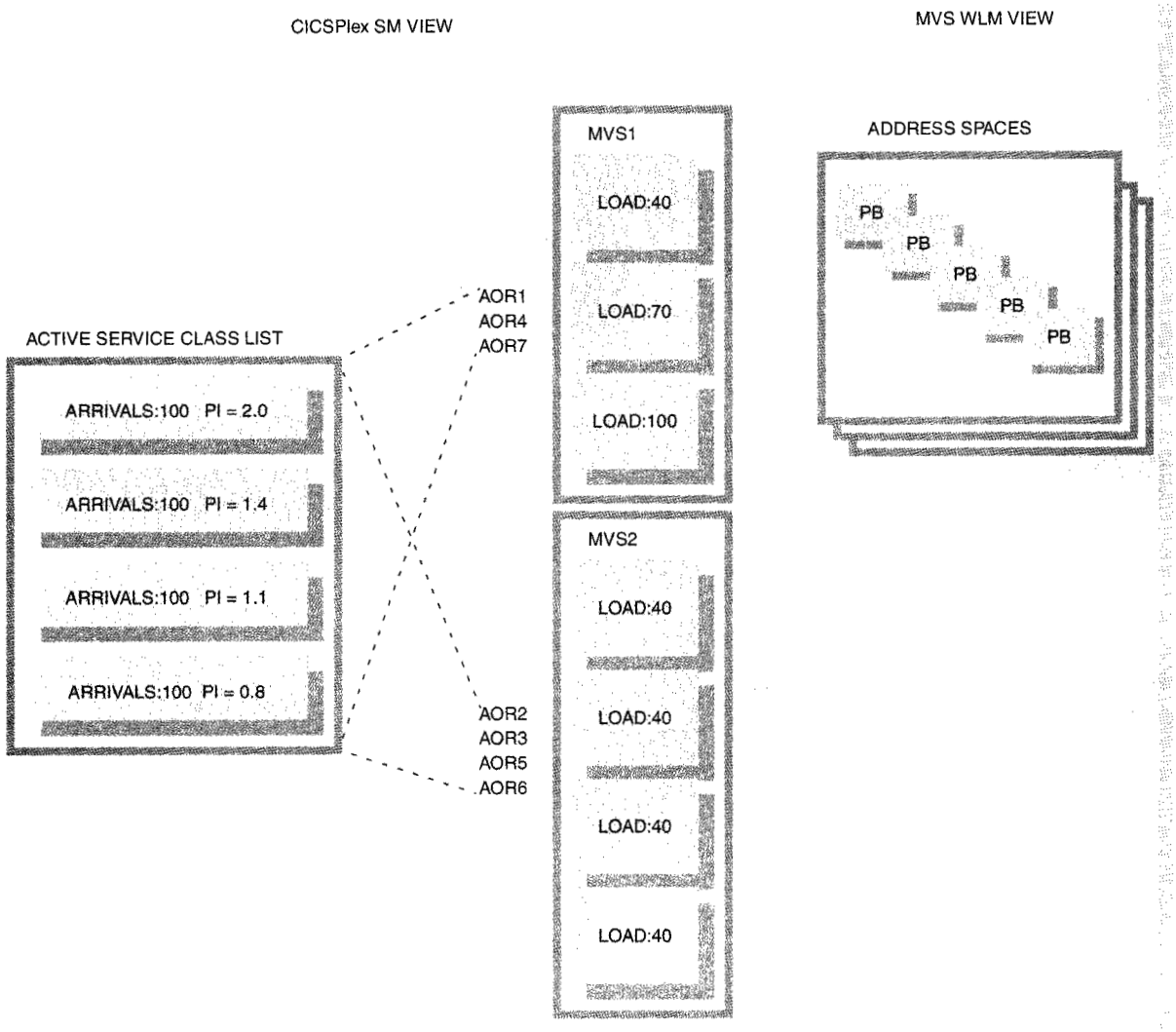
dence rate, we effectively partition the range of PIs. Each AOR thus receives only a portion of the spectrum of PIs within the MVS image. This allows MVS WLM to successfully allocate dispatch priority and storage in order to attain the response time objectives set for the transactions.

Figure 16 is a simplified view of the responsibilities of CICSPlex SM and MVS WLM when routing is done using average response time goals. It is the responsibility of CICSPlex SM to assign transactions to the AORs. MVS WLM sees transactions as performance blocks within address spaces.

MVS WLM determines the dispatch priority and the storage to allocate for each address space (AOR) based on the active performance blocks. Each performance block has a state: initial (the transaction is starting), running (the transaction is executing and suspended, i.e., waiting for input or output), notify in the AOR (the transaction is involved in a pseudoconversation), or report in the TOR (the transaction has finished). MVS WLM samples the performance blocks for delay information. During the report state MVS WLM can obtain information needed to calculate its own PI for the address space.

Other uses of CICSPlex SM dynamic workload management. CICSPlex SM 1.2 introduced the ability to invoke the dynamic routing function from outside the CICS dynamic transaction routing exit routine. The function may therefore be used to obtain target information for distributing work within the

Figure 16 Routing using the goal algorithm

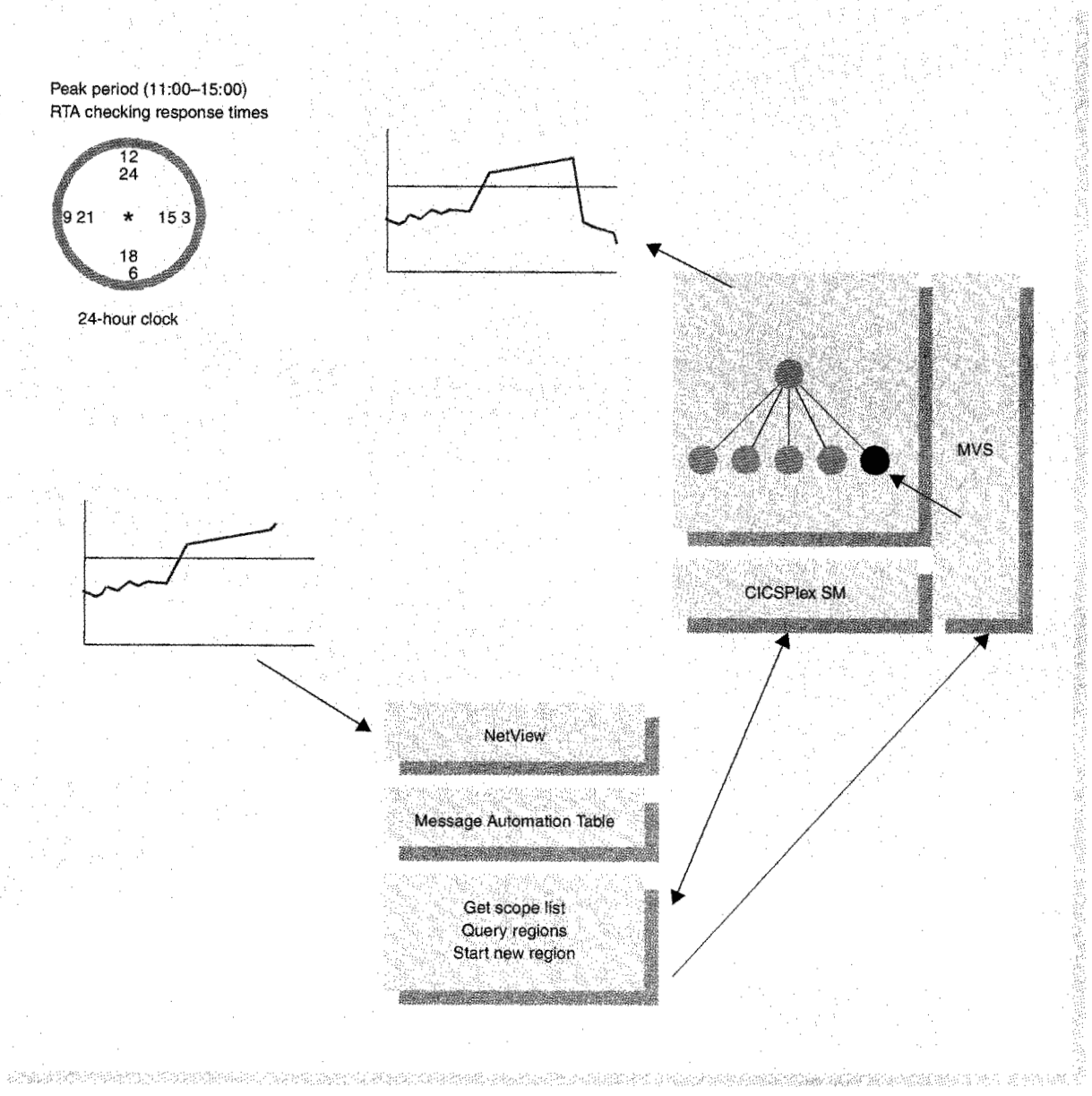


CICSplex for a variety of reasons, for example, dynamic program link requests.

Putting it all together. We have seen how the various components of CICSPlex SM can be utilized to provide systems management with a single-system image for CICS within the CICSplex. We now look at how we can integrate these functions to enhance the management of the CICSplex in a Parallel Sysplex.

First let us consider the migration to a Parallel Sysplex from an existing bipolar machine running a CICS workload. The customer might install CICSPlex SM to gain

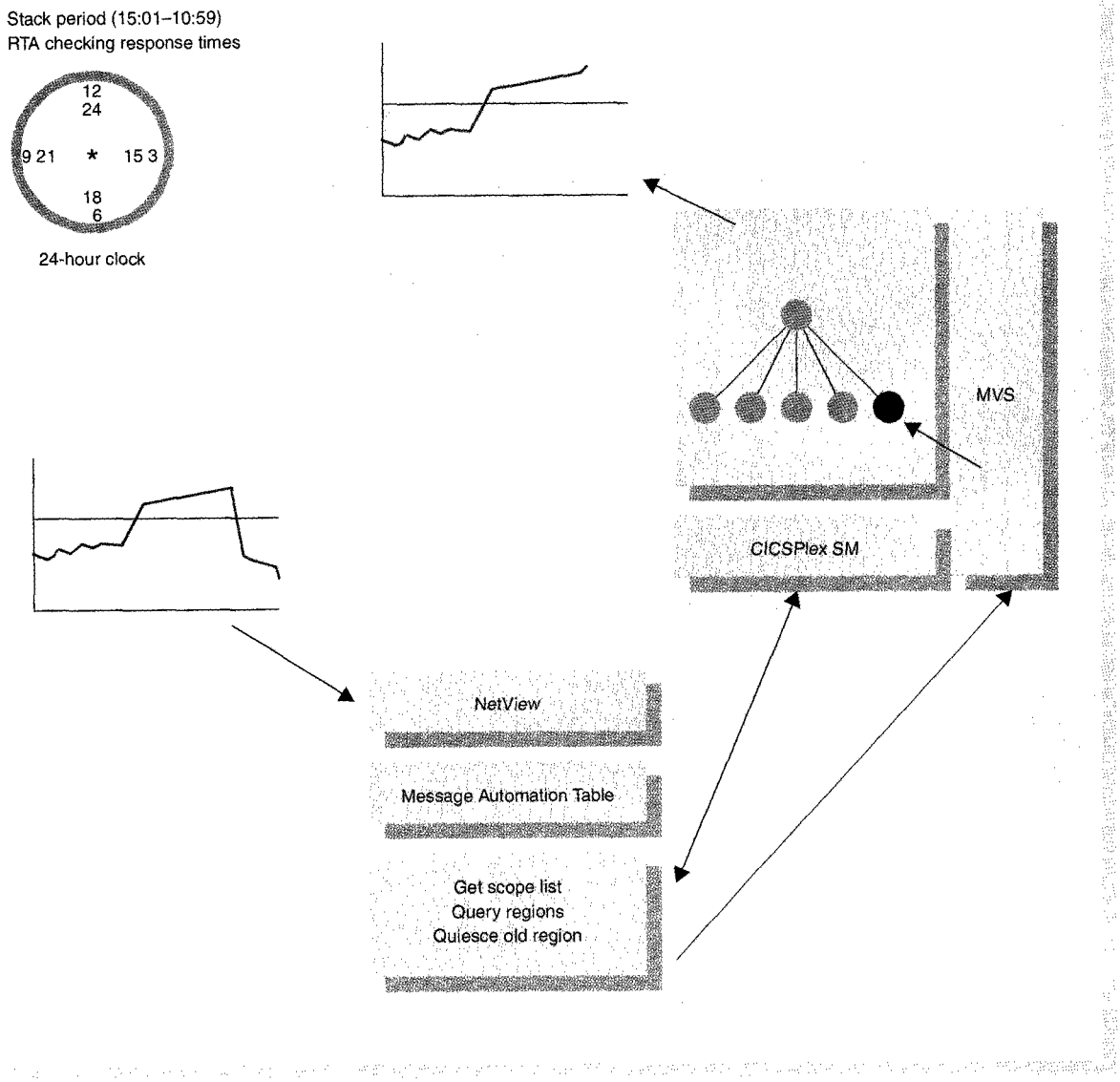
Figure 17 Dynamic addition of a new CICS region



experience in order to simplify migration to CMOS. Since CICSPlex SM provides a single-system image, the migration of CICS regions to CMOS does not affect users of CICSPlex SM. When CICS TORs and AORs are cloned it is a simple matter to add these CICS regions to the CICSplex definition and to the system groups already defined. The large increase in the number of CICS regions is therefore easily controlled.

Consider error recovery after migrating from CICS to CICSPlex SM. If the facilities of MVS ARM are used to restart failed CICS systems, again there will be no change in the interactions at the end user interface. Even if some regions fail or are moved to another processor, the interactions will remain the same. Indeed, it may be as a result of RTA processing that the region is shut down for restart by MVS ARM.

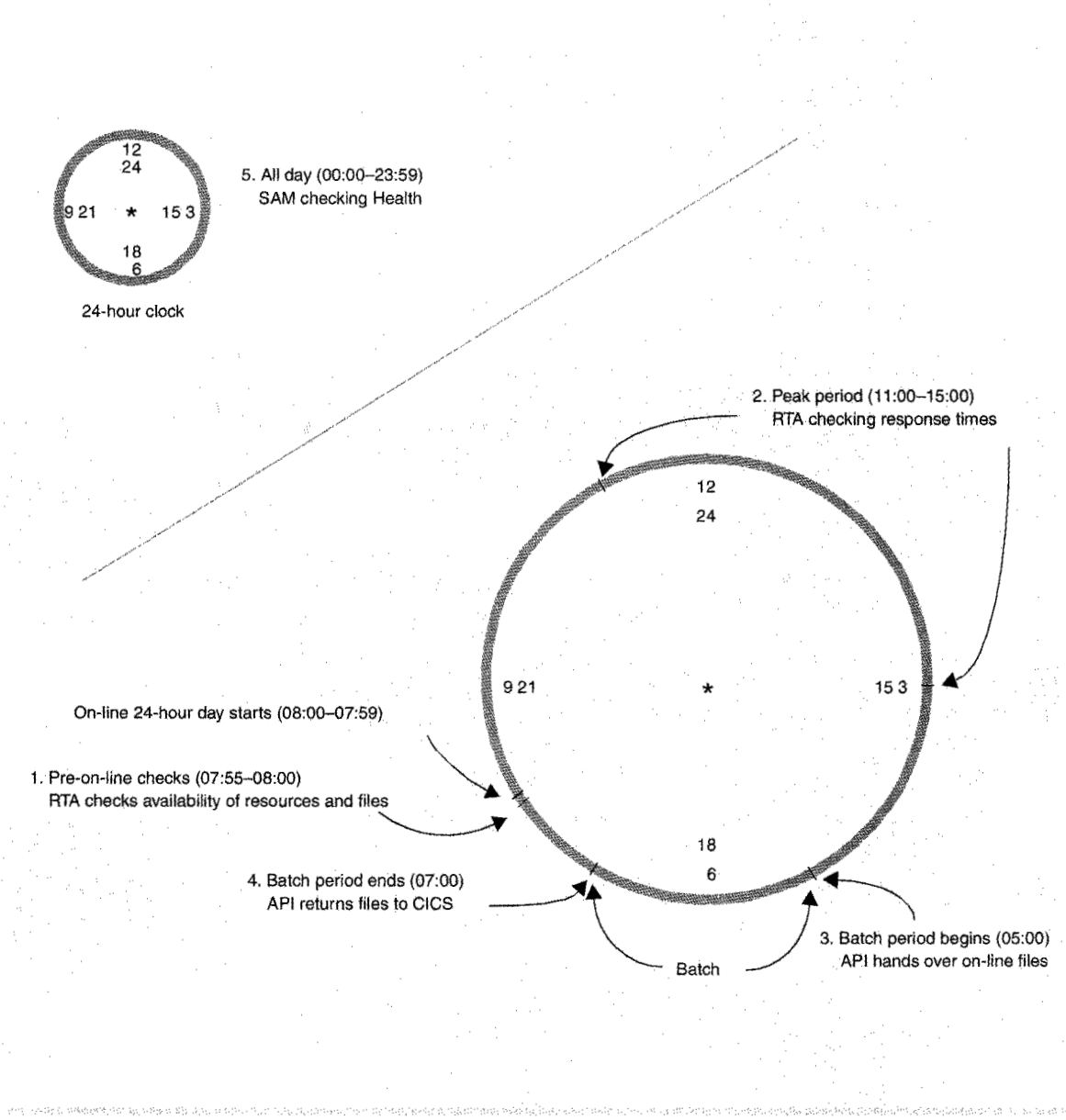
Figure 18 Dynamic removal of a CICS region



Now consider the day-to-day operations. Balancing of the workload, workload separation, etc., can be performed by CICSPlex SM in either queue or goal mode. With goal mode, transactions are managed to the goals specified for them through MVS WLM. CICS address space dispatching priorities and storage will be adjusted by MVS WLM in concert with other address spaces within the sysplex to give optimum performance with respect to the defined goals. RTA can be used to check response times and to start or stop CICS AORs by using the CICSPlex SM API with the collaboration of automation products such as NetView.

As we see in Figure 17, during the peak period between 11:00 and 15:00, RTA finds that the transaction response time has gone above a threshold. Net-

Figure 19 Structured RTA and API batch via OPC/A



View is monitoring this condition, and requests that another MVS region be started for the CICSplex.

In Figure 18, we see that during the slack period between 15:01 and 10:59, RTA detects that transaction response time has moved back below the threshold, and NetView requests that the CICS region started earlier be quiesced.

RTA can be used to monitor the overall health of the CICS systems in the CICSplex. Interaction with batch processing can be controlled through products such as OPC/A, and any other calendar-driven event can be controlled via programs utilizing the CICSPlex SM API. Figure 19 gives an example of scheduling CICS files for backup in batch processing before returning them to CICS, prechecking file availability and

response times during peak load. Again, all of this will continue to execute unchanged if resources are moved within the sysplex due to an outage or for maintenance, because of the single-system image of the CICSPlex SM product.

Tracking a transaction throughout the CICSplex has already been shown to be simple (Figure 7). Graphical representation of the status of the CICSplex can be achieved via RODM and NGMF (Figures 9–12).

Client server systems management with CICSPlex SM. Of course the Parallel Sysplex does not stand alone. Today CICS OS/2 client server applications are developed that access the CICS mainframe server. Since CICSPlex SM also provides a single-system image for management of CICS on CICS/VSE (Virtual Storage Extended) and CICS OS/2, it allows the correlation of data from the CICS OS/2 workstation to the CICS mainframe server. RTA threshold monitoring is also available against CICS OS/2 objects and attributes. Status probes can be used to monitor CICS/400*, CICS for AIX* (Advanced Interactive Executive), and any of the other CICS platforms, and even the databases in IMS* (Information Management System) and DB2* (DATABASE 2*).

The future. CICSPlex SM today provides a TSO end-user interface. A statement of direction exists for a fully customizable graphical user interface. CICS Transaction Server for OS/390 Version 1.1 also contains the ability to dynamically install CICS resource definitions. Until recently that facility has only been available via CICS administration transaction. A natural extension to the function of CICSPlex SM would be to provide the ability to define, share, and install CICS resource definitions across the CICSplex. This would open up possibilities of raising the interaction level from that of the base CICS resources to that of business applications, and for increased integration of the various tasks supported via CICSPlex SM. Integration across the various other disciplines and subsystems within MVS and the other distributed platforms must also be addressed. Only when this challenge has been met will we truly have systems management with a single-system image.

Summary

As this paper has described, CICSPlex SM provides advances in many areas of systems management. There is a significant reduction in the complexity of the end-user interface through the single-system image, which makes resource locations transparent and provides data correlation. Because CICSPlex SM is independent of CICS platforms and releases, it allows management of distributed CICS client/server applications.

The dynamic transaction routing provided by CICSPlex SM improves the availability of CICS resources over previous solutions. Through the elimination of various performance measurement tasks it also reduces cost. In addition, it increases reliability by providing dynamic resource topology maps, eliminating the dependence on outdated workbooks.

Problems are detected early, allowing them to be resolved either manually or automatically within the product. External automation products can be used as well, and the CICSPlex SM API allows the integration of arbitrary user data (via status probes) into the automation solution. Problem data can be correlated with data from other subsystems using NetView and RODM.

Perhaps most important, CICSPlex SM can be used to define a high-level view of the resources of the business enterprise, and the viewer can shift his or her focus away from the details to see the business applications and their integration with other supported tasks.

Cited references and notes

1. "Function shipping" is the ability to route a request to another CICS region for access to a resource (e.g., reading a file in a remote CICS region).
2. T. Banks, K. E. Davies, and C. Moxey, "The Evolution of CICS/ESA in the Sysplex Environment," *IBM Systems Journal* **36**, No. 2, 352–360 (1996, this issue).
3. An example of such a tool is the CICS master terminal transaction (CEMT).
4. *CICSPlex SM for MVS/ESA Concepts and Planning*, GC33-0786-01, IBM Corporation (1995); available through IBM branch offices. Also see P. Johnson, "CICSPlex SM Technical Overview," *Proceedings, CICS Technical Conference*, San Francisco, CA (May 1996).
5. MVS provides address spaces within which programs execute and data spaces within which data can be placed; both reside in virtual storage. "Cross-memory" communication refers to the ability for programs executing in different address spaces to communicate with each other via MVS system services rather than via a communications product such as VTAM. Cross-memory communication can cope with large volumes of data with high performance.
6. LU (logical unit) 6.2 is an IBM protocol for peer-to-peer communications.
7. In contrast, the CICSPlex SM API (application program interface), the batched repository update (batchrep) facility, and the graphical user interface (alluded to in a statement

of direction) connect directly to the CMAS without using CAS services, thereby using the dynamic communication capabilities of the CMAS network.
8. The CICS product was restructured in V3.1 into a domain-based architecture. Data encapsulation, architected interfaces, etc., were provided in an object-based way. A domain is similar to an object.
9. P. Johnson, "Dynamic Routing in a CICSPlex," *Proceedings, CICS Technical Conference*, New Orleans, LA (May 1995); see also P. Johnson, "CICSPlex SM Trends and Directions," *Proceedings, CICS Technical Conference*, San Francisco, CA (May 1996).
10. *IBM CICS Affinities Utility MVS/ESA User's Guide*, SC33-1159-00, IBM Corporation (1994); available through IBM branch offices.
11. R. A. Cieslak, D. F. Ferguson, and J. Sairamesh, *A Methodology for Routing Transactions in the Presence of Failing Servers*, U.S. Patent 5,475,813, IBM (December 1995); see also:

    A. Ephremides, P. Varaiya, and J. Walrand, "A Simple Dynamic Routing Problem," *IEEE Transactions on Automatic Control* **25**, No. 4, 690–693 (August 1980).

    D. Ferguson, C. Nikolaou, L. Goergiadis, and K. Davies, *Satisfying Response Time Goals in Transaction Processing Systems*, available as IBM Research Report RC18139 from IBM Thomas J. Watson Research Center (July 1992).

    P. Johnson, "Dynamic Workload Management," *Proceedings, CICS Technical Conference*, New Orleans, LA (May 1995).

    L. Kleinrock, *Queuing Systems, Volume 1*, Wiley & Sons, NY (1974).

    R. Weber, "On the Optimal Assignment of Customers to Parallel Servers," *Journal of Applied Probability* **15**, No. 2, 406–413 (June 1978).

    W. Winston, "Optimality of the Shortest Line Discipline," *Journal of Applied Probability* **14**, No. 1, 181–189 (1977).

Accepted for publication January 27, 1997.

Paul Johnson *IBM UK Laboratories Ltd., Hursley Park, Winchester, Hampshire S021 2JN, United Kingdom (electronic mail: paul_johnson@vnet.ibm.com).* Dr. Johnson was awarded a B.Sc. with honors and a Ph.D. in high energy physics from Liverpool University. After several years in postdoctoral research he joined IBM in Hursley to work in CICS/ESA development. Since joining IBM he has worked in RDO (resource definition on line), CEMT, and national language support and has been involved in the application and development of formal programming techniques. In 1990 he joined the CICS systems management group, which was formed to create the CICSPlex System Manager for MVS/ESA. He has been involved with this product from its initial requirements and external specification phases through development and release of versions 1.0, 1.1, and 1.2. He is currently architect and lead developer of the CICSPlex SM product and involved in future release development. To visit the CICSPlex SM Web site, see *http//www.hursley.ibm.com/*.

Reprint Order No. G321-5645.