Multiple consoles: A basis for communication growth in large systems

by DENNIS W. ANDREWS and RONALD A. RADICE

International Business Machines Corporation
Kingston, New York

INTRODUCTION

The intent of this paper is to discuss the development of multiple consoles which on the surface appears to be a simple and straightforward concept. This concept, however, should not be considered an ending in itself; it should be viewed as a basis from which a communication network can grow. Justification for more than one console is supported to show that multiple-consoles are necessary components in most if not all large systems. Beyond the basic and obvious requirements of a multiple console environment lie many different philosophies concerning the utilization of consoles, some of which are explored in this paper.

The recently announced Multiple Console Support (MCS) Option of the IBM System/360 Operating System constitutes one significant approach. Within its own environment MCS leaves room for meaningful growth in the communications network.

THE MULTIPLE CONSOLE CONCEPT

As one looks back over the history of computer evolution the evidence of development in system features shows a steady growth. Increased diversification, throughput capabilities and physical size have all contributed to overloading the communication network. The discontinuous flow of system messages and the proportionate increase in the number of messages with the growth pattern in system features, made the task of the operator demanding to the point of physical and mental exhaustion.

An informal study revealed that three messages per minute based on an average message mix is the maximum number an operator can handle without delaying system throughput. Information only messages were the easiest for the operator to handle, because all he had to do was remember what was outputted to him regarding the system status. But as the number of decision and action type messages increased so increased the complexity of the operator's job. His role in the man-computer communication network became more vital as the system and the number of system messages continued to grow. It soon became obvious any breakdown in the response time of the operator would only degrade the power and throughput of the system.

What was a user to do? He wanted more system features and was aware that with this increased power the system would need a corresponding increase in operator-computer communications. As one user described it, "With priority scheduling and with multipartition or multiprocessor operations, the operator at the console is in the position of trying to drink from a firehose." Another user found that their IBM 1052 typewriter console output was approximately 4 million characters a month based on a 24-hour day, 7 days per week operation, or 100 characters per minute. And considering that the maximum rate of a 1052 is 14.8 characters per second and allowing 500 milliseconds for a carrier return this meant that the console was busy on an average of one-eighth of each minute. However, realizing that console output rates follow probability the situation arose too often where the console had all it could do to keep up with message output. At these times, the operator was kept abnormally busy trying to interpret what was going on. Then, if required operator communication requests are considered, such as displaying information about job activity on the IBM System/360 Models 65 and up, chances were that what the operator got back in reply was a history of what existed rather than what was presently happening in the system. The IBM 2250 Display Console helped somewhat in offsetting the time required to display these types of informational messages.

Another problem was that even if the operator, or operators with the larger systems, could keep up with
the flood of output messages they were kept active trying to satisfy peripheral requests made by the system. Add to this fact that with some users it is either necessary or desirable to have peripheral equipment located in other rooms or even other floors, the inconvenience of satisfying peripheral requests is compounded. Voice communication between operators was previously the only way of handling many of these distant peripheral demands made by the system. But voice communication proved faulty either through unintelligible operator enunciations or because of the lack of readable copy for the peripheral operator who might forget a request when his area might be overly busy. The result was needless I/O delay.

Another significant point was that all communication was dependent on one relatively inexpensive console unit. If for some reason this device should fail, all communication would stop and the system would eventually stop.

The solution to this overloaded operator and overloaded console problem was to have multiple operators at multiple consoles. Now, for example, all the operator stationed in the tape pool had to do was scan the console output near his station for messages related to the tape pool. Clearly this was a trivial solution, but it was nonetheless a solution. The inherent problem, of course, was that each operator received all system messages. We shall see later how a specific implementation of the multiple console concept can eliminate this problem of message duplication at all active consoles.

DESIRABLE ASPECTS OF A SOLUTION

What was needed to satisfy the growth in system power was an encompassing communications network that would be versatile enough to accommodate future system growth while affording immediate solutions to the present day problems inherent in man-computer interactions.

At this point then, let us enumerate what should be the desirable aspects of such a solution:

(1) Reliability—A system component as vital as the communication network should afford safeguards against failure whenever possible. Also, there should be a guarantee that every generated message gets delivered.

(2) Flexibility—The network should have the capability of molding to fit the immediate needs of the system, and allow room for future growth.

(3) Standardization—To guard against the heretofore haphazard growth in the communication area, formalized procedures should be developed to eliminate needless duplication of messages and guard against multiple formats of messages.

(4) Communication Power—Presently, a major deficit is in speed of communication. The solution should offer a definite increase in output capability. This, to have value, would have to be coupled with a routing capability which could deliver messages to their most appropriate point.

(5) Auditability—The network should be able to maintain a trail of past communications to aid in system analysis and evaluations.

(6) Security—A user desiring to keep his computer interactions private should be given that capability.

MULTIPLE CONSOLE SUPPORT

One answer to the multiple console concept was provided by IBM with the Multiple Console Support (MCS) Option, made available in Release 18 of the IBM System/360 Operating System. Previous attempts at multiple consoles within IBM were made through the ASP-HASP spooling systems, but the MCS Option was the first supported package that not only made multiple consoles a working idea but provided a communications network that could grow with the system. Let us now look at some of the highpoints of this MCS Option.

As its name implies, MCS enables the Operating System to be configured with multiple consoles, with each console performing one or more dedicated functions.

The primary means of routing messages to their appropriate point is via 'routing codes' assigned to each message. One or more routing codes may be specified to indicate to which functional area a message is to be sent. More than one routing code may be assigned to a message.

Descriptor codes provide multiple means of message presentation or message deletion from display type devices. These codes provide the individual console device support with the means of determining how a message is to be printed or displayed, and how a message may be deleted from a display device.

The user specifies the console configuration when building the system. He may dynamically alter the configuration during system operation, however. One console must be specified as the Master Console, where all commands are valid. All other consoles are specified as secondary consoles with each console having a command entering authority and assigned routing codes.

Secondary Consoles are additional consoles (local or remote) to which selected messages are routed.
console may handle more than one routing code, and the same routing code may be handled by many consoles. The user specifies which operator commands and routing codes will be authorized for each secondary console when the system is built.

Alternate consoles provide backup capability when the original console device is inoperative. An alternate console can be a secondary console or the Master Console. MCS requires that an alternate console be specified for the Master Console. If an alternate console is not assigned to a secondary console, the Master Console will be assigned as the alternate. This alternate console concept enhances the network reliability by ensuring that messages are not lost when one console goes down or offline. Initially, each console's alternate is assigned during system definition but can be dynamically changed during operation.

Six console types are presently supported by MCS:

1. IBM 1052 Printer Keyboard Model 7 with a 2150 Console.
2. IBM 1052 Printer Keyboard Model 7 with a 1052 Adapter.
3. Composite Reader/Printer or Reader/Punch combinations.
4. IBM 2250 on IBM System/360 Models 50, 65, 75, and 91 using MVT. (Display Screen)
5. IBM System/360 Model 85 Integrated Operator's Console (220C). (Display Screen)
6. IBM 2740 Communication Terminal Model I.

Console switching, the act of moving one console's capabilities to another console, can be done automatically, dynamically, and manually. Automatically switching to an alternate console occurs when the console is determined to be inoperative by the software. An operator command has been provided for dynamic console switching and console reconfiguration. The external interrupt key on the operator's panel provides manual switching to a new Master Console. The facility for the DISPLAY of the console configuration is provided through the Display Console operator command.

HARD COPY LOG, ROUTING CODES, AND TIME STAMPING: MCS provides the capability to have buffered or immediate hard copy. Specification of the hard copy device is provided at system definition and at system initialization. It eliminates the loss of information when graphic console operators delete messages and operator commands from their screen. Hard Copy collects and records all messages that have routing codes which intersect an assigned set. This set can be dynamically changed by the Master Console operator. Messages when sent to Hard Copy are prefixed by their routing codes and a time stamp from the system clock.

User Exit: An option is provided to enable the inclusion of a resident, user-written exit routine. This routine receives in a separate buffer a copy of each message before it is routed. Available to the routine are the following:

1. Message Text  
   (read only)
2. Routing Codes  
   (modify or suppress)
3. Descriptor Codes  
   (modify or suppress)

This allows a user to impose his own functional routing mechanism.

These individual facts about MCS together form a multiple console environment which accomplishes some of the aforementioned 'desirable aspects.'

(a) Reliability—The extensive alternate chaining and automatic console switching combined with a wide variety of device types insure much greater dependability. Now one or more of the consoles can fail without a noticeable delay in system functioning.
(b) Communication Power—More consoles immediately increases message output capability by a factor almost equal to the number of consoles. The new devices supported such as the IBM 2250 and the IBM System/360 Model 85 Integrated Operator's Console 220C also deliver increased speed in terms of time between message issuance and appearance on the console.
(c) Auditability—The Hard Copy Log concept is a direct answer to this problem and affords a flexible means of recording the system's daily performance.
(d) Flexibility—The changing internal and external system environment can be coped with through the new operator's commands. The console configuration can be easily changed by bringing up new consoles or removing others from operation. Operating consoles can have their functional areas and command authorities changed. Flexibility on a shop level is given by the User Exit routine which allows a user to tailor a routing algorithm more in keeping with his own specific job types.

GROWTH IN THIS NETWORK

MCS clearly follows the lines of the solution. Although not a panacea for all the mentioned points it
does afford a basis upon which the remaining points could be obtained.

MCS routes messages through the use of predetermined codes attached to messages and consoles defined to receive specific message codes. This allows for splitting messages between the consoles but does not allow for messages that might be considered priority. Thus, if a volume of messages occurs the operator will attack the output serially and he may lose time responding to a number of less important messages before he gets to the priority message. MCS does provide, however, a mechanism for flagging messages that are considered needed action on the part of the operator, but still a priority message may be tucked at the end of the queue. This problem of priority messages will grow as the system continues to grow and the number of output messages and operator responses increase. Although priority routing is a valid point it should be noticed that a multiple console environment that properly uses functional routing minimizes problems in this area.

The security aspect has not been addressed during the MCS discussion. Certain abilities for security handling and monitoring exist through the USER EXIT and HARD COPY mechanism, and a routing code is assigned for security messages. Beyond this there is no defined mechanism for a security console to monitor system functions that may affect the security user. Since the master console always has the authority to issue all commands and other consoles may have limited command issuing authorities, either may inadvertently affect the security user by cancelling his job, halting his job, or varying system conditions. The security user should be able to monitor all system functions that may affect the status of his jobs.

MCS is device independent within the range of devices/consoles it presently supports and in some cases devices not presently supported can be hooked into an MCS environment if the user modifies internal restrictions that disallow such devices. A mechanism should be provided to allow total device independence. Thus, a user should be able to allow for a desired number of consoles, and later, say at system initialization time, decide what devices he wants to use as consoles.

Message standardization perhaps does not fall within the range of implementing a multiple console concept. However, standardizing message formats and text would have great impact on an MCS environment since the USER EXIT could have more effective text analysis. This would indeed make available more means of routing to the individual user. For example, MCS puts the job name on each message. Hence the USER EXIT could feasibly route by job name by incorporating a standard naming technique within the given shop.

OTHER PHILOSOPHIES

The last two sections dealt with one specific case, namely MCS, which should be recognized as just one example. It is not meant to be representative since any application is a function more of its environment than any other single factor and consequently cannot be considered a general solution. It (MCS) does however point out that more can be derived from a multiple console environment than meets the eye. The extremely obvious and trivial concept of more than one console, can play a significant role in large system enhancement if implemented in a meaningful manner. The basis upon which messages are routed seems to be the major point of difference between philosophies. No one method can be labeled as the best since different environments lend themselves to different routing algorithms.

Some systems have more than one Direct Access, Tape, or Unit Record pool so that functional routing as afforded by MCS would result in needless messages at certain locations. A more suitable algorithm in this case might be to route by unit address. In this way, a console could request all information concerning certain devices. This method also makes implementation of security measures more feasible since any operations concerning a particular volume could be monitored by receiving all messages concerning the unit upon which the volume is mounted.

Another routing algorithm which arises from the user who wishes to monitor his own program is the routing by job class or partition. With this concept a console would receive all messages referring to a particular job class. By properly assigning classes then, one could go to the appropriate console to watch his own job being processed.

These algorithms are for the most part batch oriented. A time-sharing environment would place different demands on the communication network and proper implementation of multiple consoles no doubt implies different routing algorithms.

SUMMARY

Beneath the existing multi-console answer to computer communication network problems lies the more significant matter of meaningful implementation. Proper implementation cannot be stereotyped since it depends for the most part on the environment.