PROJECT MAC

April 27, 1973

Computer Systems Research Division

Request for Comments No.24

Modification of the IMP DIM to ease system certification

by R. J. Feiertag

As part of the task of producing a certifiable Multics, the IMP DIM has been chosen as an initial case study. This document proposes changes to the structure of Multics that will allow the IMP DIM, as well as other DIMs to be recast into more easily certifiable form. The selection of the IMP DIM for study was made for several reasons:

1. One of the objectives of a certifiable Multics is to minimize the amount of code in the most protected region (ring 0). The IMP DIM is a large amount of code that does not need the access capabilities of ring 0 and is therefore a candidate for removal.
2. If the IMP DIM is removed from ring 0 the Network Control Program must also be removed from ring 0 and can be recast in a more easily certifiable form.
3. The IMP DIM is one of a set of device interface modules which reside in ring 0. The set of major problems which must be solved in order to remove the IMP DIM from ring 0 is a superset of the union of sets of major problems which must be solved in order to remove other DIMs from ring 0. Therefore, removing the IMP DIM from ring 0 will make removal of other DIMs and their recasting into a more certifiable form straightforward.
4. The strictures of writing programs in ring 0 has forced the functional modularity of Network associated programs to be compromised. Removal from ring 0 will permit proper modularization, thereby making certification easier.
5. The proposed solutions, given below, to problems of the IMP DIM have application to many areas other than DIMs and are generally desirable and needed facilities for Multics.

Difficulties

In order to motivate the proposals below, it is necessary to describe certain aspects of the IMP DIM. It is the function of the IMP DIM to control physical communication with the IMP. To do this it must maintain core resident buffers, generate DCW lists, execute connect instructions, and handle interrupts from the IMP. In addition the single physical IMP channel must be logically multiplexed into many logical channels called links. The IMP DIM must, therefore, maintain tables describing the links, including the association of a link with a particular process. Note that protection of the IMP DIM is necessary to insure the integrity of the objects held in common for all processes, i.e., the link tables, the buffer areas, and the IMP channel. This protection need not necessarily be at the most protected (ring 0) level. The 6180 hardware does require that the connect instruction be issued and that interrupts be handled in ring 0, however, the use of ring 0 primitives (with appropriate validation) can make these functions available to outer rings.

Why then, is the IMP DIM in ring 0? One reason is to avoid the validation of DCW lists. This validation is necessary, if the DCW lists are generated outside ring 0, in order to protect ring 0 information, but its cost is not very significant. Another reason is that interrupts must be handled in ring 0. However, as stated above, the ring 0 portion of the interrupt interceptor need be only a trivial procedure which simply transfers control to the handler in an outer ring.

Perhaps the most important reason why the IMP DIM is in ring 0 has to do with the real time response requirements of the ARPA Network. The most critical time requirement of the ARPA Network involves driving terminals connected to TIPs at full speed, since the TIP buffers only eight characters at a time it is necessary for Multics to send more data at least every 250 milliseconds to drive a 300 cps terminal at full speed (assuming a Network transmission time of 0), i.e., Multics must send the next 8 characters within 250 msec. of receiving the interrupt indicating that the last 8 characters were received. The only way to attain such real time response currently in Multics is to respond to the interrupt at interrupt time when in full control of the machine.

Solution

There is one straightforward implementation of the IMP DIM that provides the greatest promise for overcoming the above mentioned problems. This involves converting the IMP DIM interrupt handler to be a separate process. This process would normally be blocked waiting for an interrupt from the IMP channel and when one arrives the system interrupt handler would awaken the waiting process. As a separate process, the interrupt handler need no longer be in ring 0 and can have access rights independent of any other procedures, i.e., access to some IMP DIM data bases can be restricted to the IMP DIM process only and the IMP DIM can be restricted from accessing certain other system data bases. (Note that if all interrupt handlers are converted to processes and removed from ring 0, what happens at interrupt time in ring 0 can be greatly simplified.) Once removed from ring 0, the IMP DIM will require a ring 0 interface and validation will be necessary. As mentioned earlier, however, this should not be a significant expense.

The major problem remaining, then, is that of real time response. Multics was not designed to be used as a real time system, however, in order to be a useful system some facilities, e.g., I/O, must respond quickly. It can be very frustrating for a user to have to stop typing every eight characters or have a page of text printed out in spurts of eight characters. To attain real time response it is necessary that the IMP DIM process (or processes) be given high priority scheduling. If the process is given the highest priority then it can respond as quickly as the present interrupt handler does.

There are yet more subtle issues involving response and efficiency. Once the IMP DIM interrupt handler is made a separate process it is no longer subject to the current constraints of interrupt handlers in ring 0; this means it can take page faults, segment faults, and linkage faults. It is no longer necessary that the interrupt handler and all its data bases be core resident. Allowing paging has the desirable effect of more efficiently utilizing the critical resource, primary memory. However, it has the undesirable effect of slowing down the interrupt handler process and therefore reducing its real time response capabilities. It is difficult to predict exactly what effect paging will have upon the real time response of interrupt handlers. The effect will also change with time. For example, as the IMP becomes more heavily used, the effect of paging will probably diminish because its pages will effectively become core resident. If the typewriter DIM interrupt handler were allowed to page, it might very well remain core resident due to its high frequency of use.

As a defense against disaster, should paging cause a significant reduction in the response of interrupt handlers, it is necessary that we allow the handler processes to wire down pages if necessary. Clearly, this is a trade-off of the value of quick response versus the cost of primary memory pages.

To accomplish the solution described above, five major tasks need to undertaken:

1. The IMP DIM (and other DIMs should the scope of this design be enlarged) must be rewritten to execute outside ring 0 and with the interrupt handler portion running as a separate process;
2. The scheduler must be modified to have a more sophisticated scheduling strategy, i.e., be able to grant certain processes absolute high priority over all others;
3. A new form of interprocess communication is necessary to permit the types of communication necessary by these new handler processes;
4. Modification is required to the interrupt interceptor to wakeup the interrupt handler process instead of calling an interrupt handler program;
5. A means must be provided by which handler processes may wire down pages.

The Plan

The five tasks described above represent a significant change to the Multics supervisor. Because we do not fully understand the possible impact of these changes and because there may be unanticipated problems, it is more reasonable to implement these changes in small steps rather than a great leap forward. Proceeding in small steps allows one to measure the effects of isolated changes and to backtract if insurmountable problems arise. Fortunately, it is possible to achieve most of the least understood effects of the change, albeit in a none to clean manner, without extensive implementation changes. Most of the major work can be postponed to the latter stages of the plan when the exact requirements of the handler processes are better understood.

Stage 1 of the plan involves converting the IMP DIM interrupt handler (and other handlers, if desired) to a process in the simplest possible manner. The procedure will continue to run in ring 0. The iom_manager will be modified to wakeup the handler process rather than call the handler procedure. The handler procedure will go blocked rather than return to the iom_manager. A means of creating this special process will have to be produced and the traffic controller will be augmented to give high priority to this process. All associated procedures and data bases will remain core resident. A new means of interprocess communication is desirable, but not essential here as the present one can be patched up sufficiently to provide the necessary functions. These modifications are relatively simple to implement and yet will accomplish the important step of converting the handler to a process. At this point it will be possible to measure the cost that the extra process switching introduces to the new scheme and the effect of the delay due to scheduling and getting the process running.

Stage 2 consists of unwiring the handler procedures and data bases that need no longer be core resident. This will enable us to measure the effect of paging upon the real time response of the handler. If the real time response degrades significantly it is possible to rewire some of the pages although it is hoped that this is not the case.

The third stage is to move the IMP DIM and associated segments outside ring 0. The major work here involves creating a proper ring 0 interface for issueing the I/O requests. At this point we can measure the cost of this interface and the impact of taking segment faults as well as the burden of the increased working set necessitated by multi-ring operation.

Up to this point the IMP DIM has not changed in structure. The only changes have been to make the handler go blocked and to introduce a new ring 0 interface. Stage 4 involves rewriting the IMP DIM in order to remodularize it and take advantage of its new circumstances. At this point new forms of interprocess communication are probably mandatory. At the conclusion of stage 4 all the major tasks to be accomplished will have been completed.

At all stages during the plan it is expected that sufficient instrumentation be implemented to allow evaluation of that stage. It is hoped that none of these stages will significantly affect the performance of the system, but if this should not be the case it should at least be clear exactly what feature has caused the problem.