BELL TELEPHONE LABORATORIES
INCORPORATED

SUBJECT: Software Tools for Monitoring and Tracing in Multics - Case 39199-14

DATE: February 15, 1966

FROM: J. F. Gimpel

MEMORANDUM FOR FILE

PREFACE

With respect to monitoring the performance of the multics system, the following three tenets seem to be widely held.

(1) Performance monitoring is not only good but essential.

(2) Designing performance monitoring capabilities into multics should parallel the development of the system itself so that the necessary hooks can be embedded before the system is "cast in concrete".

(3) It is impossible to say (and dangerous to fix) in advance the specific monitoring which will be required.

These three articles of belief suggested a flexible approach to performance monitoring; in particular, initial work aimed at the development of generally usable tools rather than specific plans to thereby sidestep the seeming paradox posed by (2) and (3).

The methods developed seem also applicable to debugging and tracing both system and processes. Thus it may be possible for these other system functions to share a common mechanism even though the interface with the user may differ widely.

A variety of people have made contributions to this work in the form of suggestions, information and encouragement. Dave Farber of BTL was responsible for the initial inspirational impetus. Others include Jerry Saltzer, Don Widrig and D. Wagner of MIT and Chester Jones and Vic Vyssotsky of BTL.

I. INTRODUCTION

In contradistinction to much of the supervisory software in the multics system, certain modules concern themselves not with servicing the user directly but with obtaining information about the operating system. The information is intended for consumption by management, system programmers and staff personnel generally charged with the responsibility of building and maintaining the facility.

Experience with project MAC has shown, and it can readily be imagined, that monitoring the performance of a time-shared system is more important (and more demanding) than monitoring the performance of a system where physical input, and physical output are centralized, serial and easily managed.

Performance monitoring activities in Multics will probably serve at least the following purposes

(1) System debugging

(2) Compiling statistics to improve software strategies (e.g., scheduling algorithm, paging algorithm).

(3) Real-time statistics gathering to affect parameterized algorithms.

(4) On-line data gathering for users (provided safeguards exist).

(5) Off-line and on-line data gathering to affect operations (i.e., computer center) policy.

(6) Long range statistics gathering to aid in the design of new machines and new software.

(7) Data gathering for purposes unforseen at system design time.

Not only is it true that these purposes do not furnish specific data to be gathered, but they also imply that one specific form of data collection will not really suffice.

Consider the following example. Suppose every once in a while the system strangely and unaccountably does down. Suppose by standard monitoring techniques it can be determined that the breakdown is accompanied by a wild uncontrolled growth of the concealed stack which rapidly fills all of core. The cancerous growth of the concealed stack may be a symptom of or may be closely associated with the cause of the system undoing and in general only further monitoring can really tell. To determine the precipitating cause may require a great deal of extra monitoring which under normal use would imply an unbearable amount of overhead. The happiest circumstance would be if the growth of the concealed stack could itself trigger this excess monitoring. The excess monitoring could consist of copying key data bases which could be "played back" at a programmer's leisure after the system's collapse, or, could consist of methods to interrupt and arrest the growth of the concealed stack.

The monitoring and tracing package described in this memo, is being designed to realize just this sort of interactive monitoring.

II. THE MONITORING AND TRACING PACKAGE

The Monitoring and Tracing package (MT-package) will permit at certain user-designated events certain user-specified activities. An event, for example, can be the call of procedure P in process A (or in any class of processes) or the occurrance of one of a class of interrupts or a particular moment of time. The invoked activity could be the execution of a procedure, the restarting of a process, or some subset of fixed data gathering activities. In all cases the activities permitted will be dependent upon the events requested and the user who is requesting the action. Thus a user may request that procedure Beta be called every time procedure Alpha is called in his own process for debugging purposes, but unless he has the necessary authority he may not initiate this action for all processes. On the other hand system monitoring and debugging will generally require control across all processes.

Section III deals with a tentative list of events; Section IV outlines the activities; Section V deals with the organizational structure of the package.

III. EVENTS

The power and flexibility of the MT-package will be largely dependent on the number and kinds of events which the user can designate. The following list may be regarded as tentative.

1. calls and returns to linked procedures
2. process faults
3. process and system interrupts
4. a particular moment of time
5. the opening and closing of files
6. intra-segment references

Each of the above classes of events can be further modified. For example, the call-return event can (and should) be modified by specifying a class of procedures and a class of processes.

In the simple case of timing a procedure both the call to and return from a procedure must initiate activity. Moreover, let a fault occur within the procedure being monitored. Then another routine is called into play, works for some time, and relinquishes control, so that the original procedure resumes its activity at the point of departure. In this case, both the fault and the concomitant return must be monitored. These kind of events in which monitoring activity is to be provoked on two occasions rather than one are called bi-polar. Those kinds of events in which activity is provoked just once are called unipolar. For example a particular moment of time represents a unipolar event.

IV. ACTIVITIES

Activities prescribed by the user can be of three types.

(1) a subset of fixed data gathering.
(2) the call of a procedure
(3) the restart of a process

Though, in a sense, (2) subsumes (1) it will in general require less authority to specify (1) than to specify (2) especially for system-wide events (i.e., all processes). On the other hand, for process debugging, a user may specify any arbitrary procedure provided the events which trigger this activity occur during his own process.

Type (3) activity requires a call to entry restart in the process exchange; in general this activity will require little, if any, privilege. The scheduled process itself may have a self-favorable schedule to guarantee a rapid process switching if this is needed for certain system debugging and/or monitoring. But this is left up to the process which is being restarted and is of no direct concern to the MT-package.

V. GENERAL ORGANIZATION

The MT-package is divided functionally into three areas, viz. control, activity distribution, and collaboration. Generally, the control section forms an interface with the user, activity distribution concerns itself with activities, and collaboration, with events.

CONTROL

The control section of the MT-package (Fig. 1) consists of two modules working on two data bases.* The two data bases are called the plan of SYSTEM monitoring (pSm) and the plan of PROCESS monitoring (pPm). As their names imply, the pSm describes the monitoring of the system as a whole, while the pPm describes, and in fact controls, the monitoring which occurs within a single process. Hence there is one pSm whereas there is a pPm for each process.

When a user issues a request to monitor certain events, a call is made to the MT-manager. The MT-manager checks the request and either grants or denies permission to carry out the activity.

---

*a half arrow pointing from a module to a data base indicates writing; the reverse direction indicates reading. A full arrow as usual is used to indicate a call-return.

If permission is granted, the information which effectively mirrors the request is placed in the pSm data base. The MT-manager sets a flag called the System Monitoring flag (SM-flag) to indicate that some monitoring activity is to take place. This flag is not turned off as long as any monitoring activity is being requested.

Although the pSm is immediately updated, the pPm is only periodically updated, viz. at "process-resume time".* At such times a suitable system module checks the SM-flag and, finding it ON, calls the MT-planner which prepares the pPm data base. This preparation may be thought of as recompiling a system procedure known as the MT-interceptor which is described later.

The pPm is actually a macro data-base consisting of three sub-bases. The auxiliary linkage section (ALS) for the MT-interceptor, the auxiliary definitions section (ADS) associated with the ALS and the collaborator action specification (CAS). The MT-planner also sets flags to indicate that collaboration is required.

ACTIVITY DISTRIBUTION

The MT-interceptor serves to receive control at designated events and to pass on this control to routines which will carry out the specified activity (see Fig. 2). Its behavior is a function of the information located in the ALS and ADS. Upon receiving control the MT-interceptor does the following:

(1) For type-1 activity it calls the pm-data gatherer.
(2) For type-2 activity it calls any prescribed procedures (P1 through Pn in Fig. 2).
(3) For type-3 activity it calls restart (in the Traffic Controller) to reschedule any prescribed processes.
(4) In the case of bipolar activity it calls the target procedure. Upon return it repeats (1), (2) and (3).
(5) It returns.

Each time an event occurs for which MT-activity was specified (left-hand side of Fig. 2) control is passed to the auxiliary linkage section (ALS). There, two instructions are executed; the first sets the base pair bp to the current location and the

---

*This is perhaps not a well-defined notion at the present time. What is meant here is the time after unlinking occurs and the process is resumed with its linkage section full of fault tag 2 modifiers. This will occur frequently enough for our purposes because each time a procedure is recompiled the process must unlink.

second transfers control to the normal entry point in the linkage section of the MT-interceptor. Each distinct event (e.g., call of procedure alpha) causes control to pass to a distinct location in the ALS. The MT-interceptor, which is pure, determines its activity by operating on fixed offsets from the entrypoint in the ALS. In particular links to type-2 activity procedures are located there and the definitions section for these links are located in the auxiliary definitions section. A link to the target procedure is also located in the ALS.

THE COLLABORATORS

Located throughout the system, there must exist a variety of mechanisms whereby, at appropriate events, control can be made to pass to the MT-interceptor. These mechanisms will be called collaborators, and in general, there will be a collaborator for each event-type. A collaborator will in general operate upon information in the CAS. For some events, e.g., a moment of time, the mechanism exists outside the MT-package and is essentially provided free. On the other hand, for an event such as the call-return, the mechanism does not already exist and will have to be written and embedded within the system.

Note that a collaborator, itself, need not gain control on every event associated with it. For example, in the case of a call-return event, we can replace the normal linker with a collaborator called the MT-linker. Suppose, then, that all calls to a procedure T are to be monitored. If procedure A attempts to call T a first-time fault will give control to the MT-linker. The MT-linker checks to see that this procedure is to be monitored. If so, it stuffs the link to T with a pointer to an entrypoint in the ALS. The pointer to T is placed in the ALS. Thereafter interception occurs automatically without the necessity of invoking the collaborator.

HO-3345-JFG-SC J. F. GIMPEL

Att.

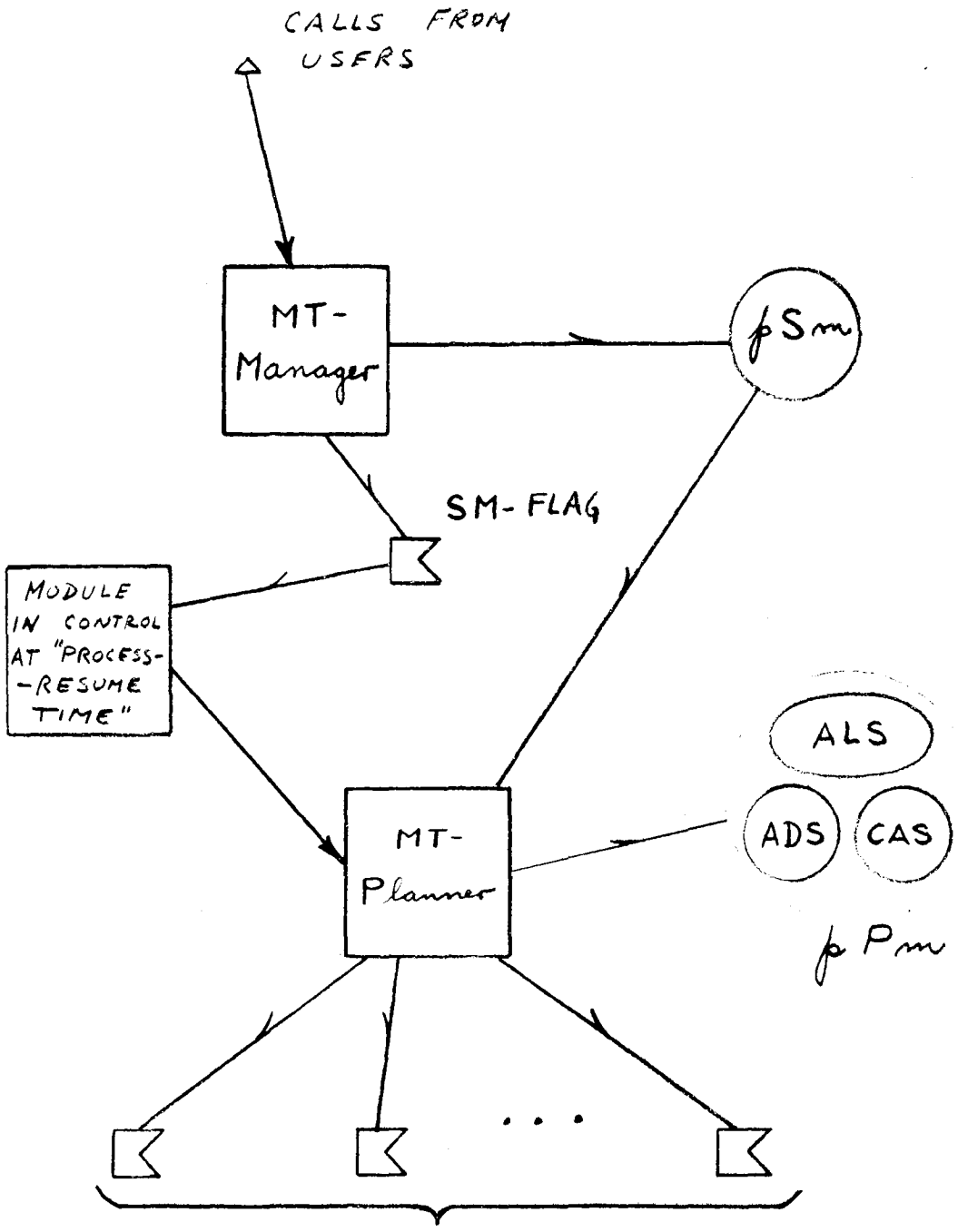

FIGURE 1

THE CONTROL SECTION OF THE MT-PACKAGE

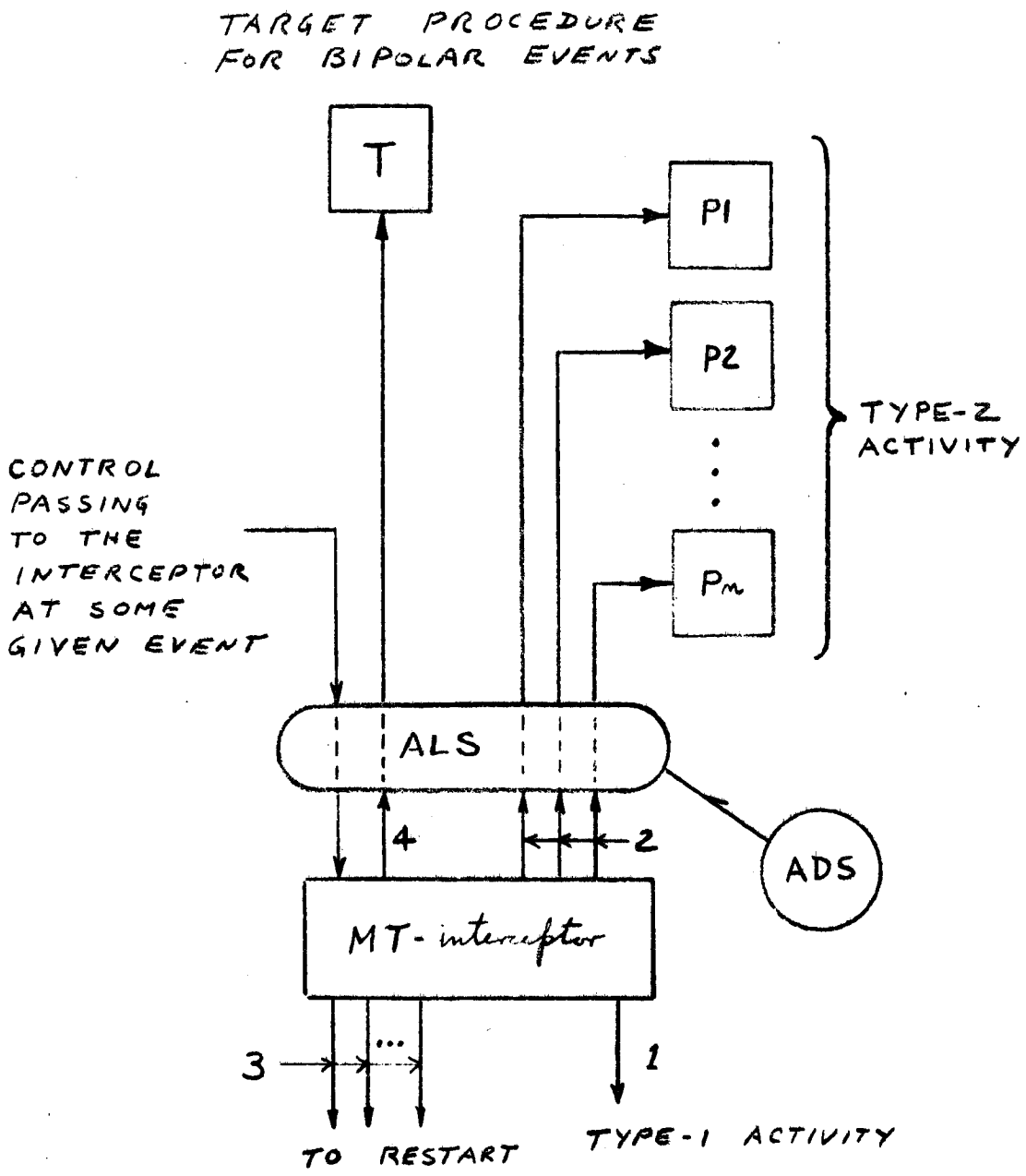

FIGURE 2

ACTIVITY DISTRIBUTION