To: Distribution

From: Jerry Stern

Date: 06/29/78

Subject: Support of Multiplexed Communications Channels

Introduction

This MTB describes a proposed enhancement to the Multics Communications System (MCS), called demultiplexing or deconcentration, intended to provide a new form of support for multiplexed communications channels. The key objective of demultiplexing is to make each subchannel of a multiplexed channel appear to be autonomous. This permits different subchannels to be independently controlled by different processes without any explicit cooperation. Such a feature is desired primarily in order to support cluster terminals (e.g. Honeywell VIP 7700 and IBM 3270) as login devices. However, this feature can also be used to support any other types of multiplexed devices including networks.

Design Alternatives

The principal design decision concerning demultiplexing is choosing the level at which it is performed. Several alternatives were considered.

Starting at the lowest level, it is possible to perform demultiplexing within the FNP. This scheme has the advantage that it relieves the central system of the burden of demultiplexing. Unfortunately, however, memory space within the FNP is already tight. Unless one were willing to dedicate an FNP to handling a particular kind of multiplexed device, there would be insufficient space for new demultiplexing programs and their data bases. Requiring an additional FNP to run a multiplexed device seems unreasonable. Also, the inferior programming environment of the FNP would make implementation and debugging difficult.

Ring 0 is the next level at which demultiplexing could be implemented. At this level it is still possible to preserve the current user interface while making multiplexed subchannels appear equivalent to non-multiplexed channels.

Demultiplexing can also be performed outside ring 0, i.e., in ring 1 or in the user ring. This possibility is attractive for two reasons. First, it would spare ring 0 the additional complexity. Second, it would be somewhat easier for sites to provide their own support for multiplexed devices not standardly supported. Unfortunately, however, outer ring demultiplexing would require that each multiplexed channel have a single owner process. This process would coordinate I/O operations for the subchannel processes. The added cost of involving a second process in every I/O operation would be excessive.

Of the several alternatives, only ring 0 demultiplexing has no major drawbacks. Therefore, ring 0 is the chosen level.

Basic Design

The basic design for ring 0 demultiplexing centers around a new data base called the Logical Channel Table (LCT) and a class of new programs called multiplexer modules.

The LCT contains one entry for every channel and subchannel managed by MCS. An LCT entry is identified by a device index (devx). LCT entries correspond to different levels of multiplexing. Starting at the bottom level, there is one LCT entry for each FNP. At the next level, there is one LCT entry for each physical channel on each FNP. If one of these physical channels is multiplexed, then there is one LCT entry for each subchannel. A subchannel itself can be multiplexed, in which case there is yet another level of subchannels.

Each LCT entry contains a major channel devx field. A major channel is the multiplexed channel from which a subchannel is derived. Thus, all LCT entries are implicitly arranged into tree structures that reflect the multiplexing hierarchy. The root node of a tree is always an FNP. The leaf nodes of a tree are always non-multiplexed channels. Intermediate nodes correspond to intermediate levels of multiplexing.

A separate multiplexer module is required for each type of multiplexed device. Each such module must provide a standard set of interfaces that will be invoked through an iox_-like call switch. Some of these interfaces are invoked in response to user calls while others are invoked in response to interrupts. The call-side component of a multiplexer module is called the "multiplexer" and the interrupt-side component is called the "interrupt handler".

The relationship between multiplexer modules and logical channels can be described as follows:

Let A be a multiplexer type.
Let M be a multiplexer module for type A.

Let C be a multiplexed channel of type A.
Let S be a subchannel of C.

then we say:

M is the multiplexer for S.
M is the interrupt handler for C.

The multiplexer and interrupt handler for each channel is specified in the associated LCT entry.

Taken together, the LCT and the multiplexer modules yield a systematic approach to handling arbitrary levels of multiplexing. Each call-side operation propagates downward through one or more levels of multiplexing until reaching the FNP multiplexer. Each interrupt-side operation propagates upward through one or more levels of demultiplexing until reaching a non-multiplexed logical channel.

CDT and CMF Changes

The LCT will be constructed from information contained in the Channel Definition Table (CDT). Therefore, the CDT and its source segment, the Channel Master File (CMF), must be expanded to describe multiplexed channels.

At present, the CDT contains two types of entries: FNP entries and channel entries. A third type of entry, called a multiplexer entry, will be added to define multiplexed channels. Channel entries will be used to define the non-multiplexed subchannels of a multiplexed channel. Thus, a one-to-one correspondence will exist between CDT entries and LCT entries.

Multiplexer entries will resemble channel entries to some extent. For example, a baud rate and a line type can be specified for a multiplexer entry. A terminal type, however, cannot be specified. Instead, a multiplexer type must be specified. An optional argument string may also be specified to guide multiplexer initialization.

The current scheme for naming channels must be discarded since it represents an encoding of FNP channel numbers. Instead, multi-component names will be used where each component corresponds to a level of multiplexing. The first component of a channel name (or multiplexer name) will always be the FNP identification tag. The second component will represent the FNP channel number. For example, the name "a.h107" indicates channel h107 on FNP a. (Here, h107 means hsla 1, subchannel 7). Channel names will have more than two components if multiplexing is involved. For example, suppose that a.h107 is a multiplexer. Then, a.h107.13 is subchannel 13 on that multiplexer.

The implications of introducing new channel names are discussed later. Note that network channel names, both for telnet and ftp, are not affected by this new naming scheme.

Initialization

Initialization of ring 0 demultiplexing is essentially a matter of constructing the LCT and causing all multiplexer modules to initialize themselves. This, of course, is an integral part of the overall initialization of MCS.

MCS initialization is managed by the Initializer process at answering service startup time. Every logical channel (i.e., every CDT entry) must be initialized. The procedure begins by initializing an FNP. Next, all channels on the FNP are initialized. If one of these is multiplexed, then its subchannels are initialized, and so on until the entire multiplexer hierarchy is finished.

Whenever a multiplexer channel is initialized (including FNPs), an initialization routine of the associated multiplexer module is run. This routine receives a list of multiplexer subchannels extracted from the CDT. It assigns an LCT entry to each subchannel and assigns itself to be the multiplexer for each subchannel. Also, it assigns itself to be the interrupt handler for the major channel. Multiplexer data bases are allocated and initialized at this time.

Reinitialization

When a mutiplexer module detects that a multiplexed channel has "crashed" due to physical disconnection or other causes, it must reinitialize the channel. This is accomplished in the following way. First the multiplexer signals each subchannel that the major channel has crashed. Any subchannel which is itself multiplexed must reiterate this procedure. Each subchannel then signals the Initializer that it has hung up and is not to be reinitialized. When all subchannels are finished, control returns to the major channel multiplexer, which then proceeds to clean up after the major channel. All subchannel LCT entries are discarded and all major channel data bases are deallocated. Finally, a signal is sent to the Initializer indicating that the major channel has crashed and should be reinitialized. The Initializer then begins the previously described initialization procedure starting with the crashed channel.

Channel Names

The introduction of a new genre of channel names is an

incompatible user interface change. Therefore, existing interfaces must continue to accept old-style channel names as well as the new type. This situation is complicated by the fact that old-style names have a maximum length of six characters. New-style names for all channels that exist today will not exceed this length. However, new-style names for multiplexed subchannels can be longer.

The following approach is suggested:

1. tty_$tty_attach, hcs_$tty_attach, and hcs_$tty_index will accept both old and new-style names.

2. user_info_$terminal_data will return the new-style channel name. However, if the caller provides insufficient space, an error code will be returned.

3. [user device_channel] will return the new-style name.

4. Dial messages sent by the Initializer will contain a devx instead of the channel name. convert_dial_message_ will convert this devx to a new-style name. However, if the caller provides insufficient space, an error code will be returned.

Programs that call user_info_$terminal_data or convert_dial_message_ may require a minor change. These changes can be made at any time, even before new channel names are introduced. Such programs, if not changed, will not encounter problems until used in conjunction with multiplexed devices.

Implementation Plans

Whenever a new type of multiplexed device is to be attached to Multics, a new multiplexer module will have to be added to the supervisor and, in general, a new or modified set of control tables to run the device will have to be added to the FNP. For Multics release 7.0, we plan to implement the general mechanism described in this document, and a set of control tables and a multiplexer module for one type of device, namely the VIP 7700. Other multiplexer types, such as the IBM 3270, will be added in future releases.