bcc	title An Efficient Multiplexing Algorithm	prefix/class-number.revision ATINFIC/W- 37

checked [illegible]	authors Paul Heckel [illegible]	approval date 12/16/69	revision date
checked [illegible]		classification Working Paper	
approved [illegible]		distribution Company Private	pages 8

ABSTRACT and CONTENTS

Consider the problem of multiplexing output data for several devices, each of which receive string bursts of moderate length. (average of 10 or more characters).

The Multiplexing algorithm uses subsidiary algorithms for scanning the line. Its value is that it will work for any scanning algorithm. The efficiency of the algorithm depends on the length of the string, but is about (100-300/n)% where n is the length of the average string burst. This gives an efficiency of 90% if the average string burst is 30 characters long, and 70% if the average string burst is 10 characters in length.

This algorithm is used in the Phase 2 Communicati ons System; but is considered to be of sufficient interest to be published separately.

Introduction

Teletype output and input have two basic dissimilarities. First there is more output because computers are more verbose than people. Second it is more regular because people are more spastic than computers. By more regular it is meant that once a device starts it will continue to deliver characters at regular predictable intervals. This is true in the case of the computer because it outputs bursts of strings which are buffered. At regular intervals characters are taken from the strings. When the character count drops below a certain amount, the process that outputs the characters is reawakened to provide more, which it will probably do before the buffered string is exhausted. A person inputting characters to a teletype, on the other hand, types in short irregular bursts. A teletype reading a paper tape would be regular, however.

The basic idea behind the algorithm is that there exists a (slowly changing) set of teletypes that are ON. If both the computer and the Remote Concentrator know what the set of "ON" teletypes is, then both computers can use the same algorithm for stepping to the next device, and the only information that has to be passed from the CHIO to the Remote Concentrator is information to tell the Remote

Concentrator when a device is turned ON or OFF. Since it requires 2 characters inserted in the string to turn a device ON and one to turn a device OFF, the length of the average character string must be large compared to 3.

In actual fact this is likely to be the case, especially when the line tends to be overloaded, as would be likely if several teletypes were typing listings, which contribute heavily to the load (and the average string length). In particular, a printer contributes a very heavy load to the communication line's bandwidth (one string of several thousand characters). It should be obvious that the reason that input to the system cannot use the algorithm is that most input strings are only one character long, as few typists can keep up with a teletype (the slowest device). But then again this is not of concern because the total volume of input data is much less than the volume of output data.

A Description of the Meta-Algorithm

Let there exist a table called the state table that describes the teletypes that are ON, their speed, and any other pertinent information. Let there also exist algorithms: GNL(Get Next Line), DOL (Delete Old Line), and INL (Insert New Line) which are defined on the state table. Only these algorithms may modify the table.

GNL when called should step to the next line to which a character should be sent. It has constraints imposed by the devices, and possibly other things; but not by the Meta Algorithm. In particular, it should not select an N character per second device any more often than once each 1/Nth of a second. If GNL is ill-behaved then only predictable things will happen. For example if GNL persists in skipping one teletype, then that output device will never have any output sent to it; but all of the other devices will have the expected output. In fact no matter how badly coded any of these algorithms are all devices will only receive characters intended for them, and only in the correct order.

INL is responsible for adding new lines to the set of ON teletypes. It does this by modifying the state table to indicate the new line added.

Similarly DOL when called should modify the table in such a way as to delete the current device.

In short INL activates devices, DOL deactivates devices, and GNL selects the next active device. These subroutines, and only these subroutines, may modify the state table. There is one helpful addition to this. Whenever a character is sent to the Remote Concentrator (or received by it, as we will shortly see) it may increment a counter in the state. This may be used as a timer by algorithm GNL.

To initialize this communication system we start with identical state tables and algorithms in both the CHIO and the Remote Concentrator. We then start the communications system by sending a special initialization character to the Remote Concentrator. After the initialization character has been sent the CHIO will then begin executing algorithm MAT (Met Algorithm for Transmit described in the appendix). When the Remote Concentrator receives the initialization character it begins to execute the algorithm MAR. (Meta Algorithm For Receive also described in the appendix).

In order to see what is happening consider an observer at the Remote Concentrator looking at its tables. He also looks at the tables in the CHIO while still at the Remote Concentrator, seeing them delayed by the same amount of time that it takes a message to go from the CHIO to the Remote Concentrator. The observer will note that both tables look the same. This is because MAT and MAR are synchronized, both calling the same sub-algorithms at the same time.

This is because both sets of algorithms are essentially identical. They select a device, then input to (or output from) it. The only time that this basic pattern is broken is if a control character is received (or sent). This will trigger a call of the appropriate subroutine. (DOL or INL). Since the only data that can affect the state of the tables (namely the two special control characters) are available to both ends of the communications line, both MAT and MAR can react identically to them.

Inserting and Deleting Lines

An outside subroutine, because it is not allowed to modify the state table, cannot directly turn on or off a device. The outside subroutine is allowed to set another bit, one that is not (logically) in the table. When MAT finds this bit set in the table it sends a control character (DL) to the Remote Concentrator indicating the current line turned off (DL) or a new line should be turned on (NL followed by the line number). MAT then calls DOL or INL in response to the control character(s) sent, rather than because it found a bit set. The bit's being set caused it to send the control characters. Since MAR will respond similarly to the control character it receives, both tables will be modified identically.

Conclusion

The algorithm described does not say anything about device speeds, or device selection. This is an important problem, one that will be discussed in another document. The algorithm is useful for applications that require multiplexing long strings of data.

The beauty of the algorithm is that it in no way depends upon the device selection mechanism.

APPENDIX

```
*MAT Meta-Algorithm for transmission

TLOOP:
   CALL GNL;                     *Sets up CLINE the current line
   IF DELBIT(CLINE) = ON DO;     *If delete bit is set for current
                                 *device
     TRANSMIT(DL);
     CLEARDB(CLINE);             *Clear Delete Bit
     CALL DOL;
     GOTO TLOOP;
   ENDIF;
   TRANSMIT(GET(CLINE));         *Character from current line;

LOOP2:
   IF AVNL() DO;                 *If available line to insert
     NLN ← NEWL();               *set NLN to the new line;
     CLEARIB(NLN);               *Clear insert bit
     TRANSMIT(NL);
     TRANSMIT(NLN);
     CALL INL(NLN);
     GOTO LOOP2;
   ENDIF;
   GOTO TLOOP;

*MAR Meta-Algorithm for reception

RLOOP:
   CALL GNL;                     *Sets up CLINE, the current line
   CCHAR ← RECEIVE();            *get the next input character
                                 *from CHIO
   IF CCHAR = DL DO;
     CALL DOL;
     GOTO RLOOP;
   ELSEIF CCHAR = NL DO;
     CALL INL(RECEIVE());        *Insert device specified by
                                 *next character
     GOTO RLOOP;
   ENDIF;
   PUT(CCHAR,CLINE);             *Put the character just received
                                 *in the current line
   GOTO RLOOP;
```