Network Working Group
Request for Comments: 596
NIC: 15372

E. Taft
PARC-MAXC

8 December 1973

Second Thoughts on Telnet Go-Ahead

INTRODUCTION

In this RFC we present objections to the requirement that hosts implement the Telnet Go-Ahead (GA) command, as specified in the Telnet Protocol Specification (NIC #15372). The thrust of these objections is in three major directions:

1. The GA mechanism is esthetically unappealing, both to myself and to many other people I have talked to. I shall attempt to describe why this is so.

2. As specified in the Protocol, GA will not, in general, work; i.e. it will not serve its intended purpose unless hosts make various unwarranted assumptions about how other hosts operate.

3. GA is impossible for most hosts to implement correctly in all cases. This is certainly true of the PDP-10 operating systems with which I am familiar (10/50 and Tenex).

The purpose of this RFC is to advocate either complete removal of the GA mechanism or relegating it to the status of a negotiated option whose default state is that it be suppressed.

TERMINOLOGY

"Half-duplex" is a two-way communication discipline in which transmission takes place in only one direction at a time and the receiving party is constrained not to transmit until the transmitting party has explicitly given up control of the communication path ("turned the line around").

This definition is distinct from a common (but incorrect) use of the terms "half-duplex" and "full-duplex" to designate local and remote character echoing.

"Reverse break" is a means by which a computer connected to a terminal by a half-duplex path may regain control of the path for further typeout after previously having relinquished it.

This is the complement of the "break" or "attention" mechanism, implemented by all half-duplex terminals, by means of which the user may gain control of the line while it is in use by the computer.

ESTHETIC OBJECTIONS TO GA

One assumption that permeates the Telnet Protocol specification (and is explicitly stated on Page 7) is that the "normal" mode of communication between computers and terminals is half-duplex, line-at-a-time. While historically this is partially true, it is also clear, both within the ARPA Network community and elsewhere, that the trend is toward highly interactive man-machine communication systems which are difficult to implement under half-duplex communication disciplines.

The GA mechanism is an attempt to solve a specific problem, that of switching control between computer and user in a subset of those hosts utilizing IBM 2741 or equivalent terminals. I say "a subset" because in fact the problem arises only in the case of TIPs from 2741s (with reverse break); from what experience I have had, I think the TIP does a very good job of turning the line around at the right moments. (I am told this is also the case at Multics).

Given the trend toward more interactive communication, and given the fact that terminals on the Network requiring a Go-Ahead mechanism are a distinct minority of all terminals, I think we should be reluctant to burden our protocols with kludges that are so clearly a concession to obsolete design.

> I have little doubt that before long somebody (if not IBM) will produce a full-duplex 2741-like terminal (indeed, perhaps it has already been done). There is an obvious need for a terminal with Selectric quality keyboard and hard-copy better suited to interactive applications (i.e. full-duplex).

As a more practical consideration, it makes little sense to have the default state of the GA option be the one that benefits the least number of hosts and terminals.

> There is no question that most parties to Telnet communication will immediately negotiate to suppress GA. To do otherwise will double the amount of network traffic generated by character-at-a-time typein, and will increase it by a non-negligible amount even for a line-at-a-time typein.
>
> It strikes me as worthwhile to minimize the number of such "necessary" option negotiations, especially in view of the large number of TIPs and mini-hosts on the Network. Many such hosts

must, due to resource constraints, implement only a limited subset of the available options. It follows, then, that the default state of all options should be the one most hosts will be willing to use.

WHY GA WON'T WORK

We now show that a server process's being "blocked on input" (as specified in the Protocol) is not itself a sufficient condition for sending out GA.

This is due to the fact that the user Telnet has no control over the packaging of a "line" of information sent to the server; rather, this is a function of the NCP, which must observe constraints such as allocation and buffering. Consider the following example:

A user types a line of text, which is buffered by his host's user Telnet until he signals end-of-line. His keyboard then becomes locked (this being the behavior of half-duplex terminals while the computer has control of the line), and stays locked in anticipation of the server's eventual response and subsequent GA command.

The user Telnet transmits this text line over the connection; however, due to insufficient allocation or other conditions, the text actually gets packaged up and sent as two or more separate messages, which arrive at the server host in the correct order but separated by some amount of time.

The server Telnet passes the contents of the first message to the appropriate process, which reads the partial text line and immediately blocks for further input. At this moment (assuming the second message hasn't arrived yet), the server telnet, in accordance with the Protocol, sends back a GA command.

The rest of the text then arrives in response, the server process may generate a large volume of output. Meanwhile, however, the GA command has caused the user's keyboard to become unlocked and computer output thereby blocked. Hence we have a deadlock, which will be resolved only when the user recognizes what has happened and (manually) gives control back to the computer.

Of course, this particular problem is avoided if the Telnet protocol is modified to specify that the server Telnet will transmit GA only if the server process is blocked for input AND the most recent character passed to that process was end-of-line.

I claim that this solution is bad in principle because it assumes too much knowledge on the part of the serving host as to what constitutes "end-of-line" in the using host.

Furthermore, the Protocol explicitly (and quite rightly) specifies that the user Telnet should provide some means by which a user may signal that all buffered text should be transmitted immediately, without its being terminated by end-of-line.

One must conclude, then, that in general the server Telnet has no precise way of knowing when it should send GA commands.

IMPLEMENTATION PROBLEMS

The foregoing analysis illustrates the problems that arise with the GA mechanism in communication between servers and users whose normal mode of operation is half-duplex, line-at-a-time. When we turn to hosts that provide full-duplex service, such as the PDP-10s and many other hosts on the Network, the problems are much more severe.

This is particularly true of operating system such as Tenex that exercise such tight control over terminal behavior that they prefer to operate in server echoing, character-at-a-time mode. This will probably become less necessary as protocols such as Remote Controlled transmission and Echoing Option come into general use, enabling servers to regulate echoing and break character classes in user Telnets.

Even in hosts such as 10/50 systems that provide reasonable service to line-at-a-time users for most subsystems (e.g. excluding DDT and TECO), GA is impossible to implement correctly. This is true for several reasons.

First, there are a number of subsystems that never block for terminal input but rather poll for it or accept it on an interrupt basis. In the absence of typein, such processes go on to do other tasks, possibly generating terminal output.

Processes of this sort come immediately to mind. The user telnet, FTP, and RJE programs are implemented in this fashion by almost all hosts. 10/50 has a subsystem called OPSER, used to control multiple independent subjobs from a single terminal.

Since these programs never block for input, GA commands will never be sent by the server Telnet in such cases even though the processes are prepared to accept terminal input at any time.

Second, there is not necessarily a one-to-one relationship between processes and terminals, as seems to be assumed by the Telnet Protocol specification.

For example, in Tenex one process may be blocked for terminal input while another process is generating output to the same terminal. (Such processes are typically parallel forks of the same job).

Third, there is the possibility of inter-terminal links, such as are provided in many systems.

By this I do not mean special Telnet connections established between a pair of NVTs for the express purpose of terminal-to-terminal communication, as is suggested on page 9 of the Protocol specification. Rather, I am referring to facilities such as the Tenex LINK facility, in which any number and any mixture of local and Network terminals and processes may have their input and output streams linked together in arbitrarily complex ways. Clearly the GA mechanism will fall flat on its face in this case.

Also, the notion that one user of an inter-terminal link should have to "manually signal that it is time for a GA to be sent over the Telnet connection" in order to unblock another user’s keyboard offends me to no end.

Finally, most systems provide means by which system personnel and processes may broadcast important messages to all terminals (e.g. SEND ALL in 10/50, NOTIFY in Tenex). Clearly such asynchronous messages will be blocked by a half-duplex terminal that has been irrevocably placed in the typein state by a previous GA.

This strikes me as such an obvious problem that I am forced to wonder how half-duplex hosts handle it even for their local terminals.