Network Working Group   J. Davidson
Request for Comments: 563   University of Hawaii
NIC: 18775   28 August 1973
References: RFC 357, RFC 560

Comments on the RCTE TELNET Option

RFC 560 describes a Remote Controlled Transmission and Echoing TELNET option. Its authors provide a framework wherein a serving host may control two aspects of TELNET communication over the (simplex) user-to-server path.

Commands are introduced which govern

1. when (and which) characters shall be echoed by the user, and
2. when (and which) characters shall be transmitted by the user.

Motivation for the option was based on two considerations:

1. the latency between striking and printing of a character which is to be echoed by a remote server is disconcerting to the human typist, and
2. character-at-a-time transmission introduces processing inefficiencies (for IMPS, for servers, for users) and decreases effective channel thruputs over the net.

The author feels that the RCTE description is in error (or at least unclear [1]) in its treatment of when characters are to be transmitted. However, discussion of the subject in the RCTE specification is incomplete, so it is difficult to point to a statement which is "wrong." Rather, the present objections are based on inferences drawn from the sample TENEX interaction

Perhaps there is some misunderstanding of the original issues to which RCTE now addresses itself.

Original Motivation for Remote Controlled Echoing (RCE)

RFC 357 (An Echoing Strategy for Satellite Links) introduced a need for RCE for users who are separated from a service host by a satellite link. The motivation was to lessen human frustration and confusion; no consideration was given to resulting processing inefficiencies or channel thruputs.

(In the remainder of this RFC, we consider character transmission apart from echoing considerations.)

It was recognized that the human’s best interests could be served if user-to-server transmission were performed on a character-by-character basis, (the implicit assumption being that this insured the most rapid server response possible). This scheme allowed for the classic overlap of (network) I/O and computation, and was thus efficient as far as the (human) user was concerned.

Concessions were made in the transmission strategy when it was accepted that the serving process could not in fact do any significant processing until a completed command was available. Ideally then, users should be able to buffer characters until they have a completed command and then fire off the entire command in a single "packet," with the resultant savings in channel usage and a greater per-packet data efficiency. The characters which delimited commands were called wakeup characters, in 357, for their effect on the serving process. RCTE calls them transmission characters for the effect they have at the User TELNET.

The key here is that it is quite possible for a human, separated by a satellite link from his remote host, to type several completed commands - and to therefore initiate several packet transmissions- all the while awaiting the server’s response to his first command. Again we see the overlap of I/O and computation, and again we achieve maximum efficiency from the human’s viewpoint.

The problem, however, is that wakeup (transmission) character sets change. And there will always be a finite amount of time [the one-way transmission time] during which the set definitions will differ between server and user. This says that during such times the user will be sending off packets which do not contain completed commands, (or contain more than a single completed command), or he will be buffering characters beyond the end of a completed command. (A fourth alternative is that he may actually still be doing the right thing by chance). Buffering beyond the end of a command is the only case which lessens processing efficiency for the human, however.

Dissatisfaction With RCTE

Here is the author’s complaint: RCTE [at least the sample interaction which allowed transmission (by default) only at break characters] would have the TELNET user wait until he knows exactly the wakeup (transmission) character set being used by the server ! Ideal channel utilization might be achieved, since no "unnecessary" packets are sent (and, strangely, no extra characters are allowed in the current packet) but the overlap of I/O and computation has been eliminated, and the human has an extra round-trip time added to the server’s processing time. This is wrong.

An Alternative Implementation

Unless a round-trip time penalty is to be paid by the human at every break interaction, the user TELNET must transmit characters based on the transmission character set in effect at the moment the characters are typed. And unless the step-by-step interaction developed in the RCTE TENEX example was not a true representation of the relative temporal occurances of events, RCTE did not do this.

The sample TENEX interaction showed the user typing

```
(T:) LOGIN ARPA <cr>
```

while the break set included <space> and <cr>. The only transmission characters in effect were the break characters - by default. The RCTE example showed that the LOGIN <space> phrase was, properly, a completed command; it was transmitted. But while the alternative transmission strategy of the current RFC would "recognize" the ARPA <cr> phrase as a second completed command, and thus initiate a second transmission, RCTE withholds judgment until the server respecifies the transmission classes. Response for the user suffers.

One might also ask what transmission strategy was to be undertaken when two users were, say, linked thru a TENEX. Transmission should obviously be at every character. RCTE would send the first single character packet and then wait to be sure that a single character did in fact delimit the next command also. It would wait a long time it would seem, since no break interaction would occur until the end of the line (<cr>). The user would be echoing like a champ, but no characters would be transmitted for the linked party’s inspection.

If we adopt the convention that transmission decisions should be based on the transmission set [and by default, the break set] in effect at the time the character is typed, then the sample interaction might in fact look like this:

```
P:  TENEX  1.31.18,   TENEX EXEC  1.50.2  <cr> <lf>@

T:  LOGIN <space>
P:  LOGIN <space> } >>>>>> NOTE: Typing and printing occurs simul-
U:  LOGIN <space>                taneously up to the <space> at
                                 which point the human "types-ahead."
T:              ARPA <cr>

U:  ARPA <cr>             <<key: the user transmits a second packet.
```

```
   S:  <space> <IAC> <SB> <RCTE> <0>

   P:  <space> AR

   S: <cr> <lf> (PASSWORD): <IAC> <SB> <RCTE> <7>

              [the server sends while text is printing]

   P:            PA <cr> <lf> (PASSWORD):

   T: WASHINGTON <space>

   U:  WASHINGTON <space>

   T:                   100

   S:  <space> <IAC> <SB> <RCTE> <3>

   P:  <space> 100

   T:                            0          [Again printing is
                                             simultaneous to typing]

   P:                 0

   T:                             <cr>

   P:                  <cr>

   U:  1000 <cr>

   S:  <cr> <lf> JOB ...
```

The interaction will not necessarily be the same each time. It depends on the typing speed of the user and response time of the server. For this example, both channel utilization and performance for the human are perfect, since the transmission set [even though it was only the default break set] did not change.

Unsolicited Output

The question of unsolicited output arise again. The treatment in 560 was simplified over that of 357 only because of the RCTE transmission strategy. No output could possibly be returning for a command which hasn’t been sent yet (!), so the message must be "SYSTEM GOING DOWN."

RFC 357 outlines when unsolicited output can be recognized and when it should be printed, in line with the alternate transmission scheme proposed. The requirement that such system alerts be terminated by RCTE commands is of course the proper way to handle such interrupts; this clarification of the unsatisfactory solution in 357 is appreciated.

TIP Buffering

RCTE as defined cannot allow a user to transmit when his buffer is full, else he might send a break character. [presumably the buffer fills because we are waiting for break (transmission) redefinition]. The response to the command delimited by the break character could return before the characters, of the command were "echoed." RCTE would thus demand that it be printed first, and the listing would be out of order.

The alternative transmission strategy eliminates this problem since transmission of a full buffer is no worse than guessing incorrectly that the last character in the buffer is a transmission character.

A further suggestion

All server-to-user echoing could be eliminated if control bytes were sent to indicate which break sets should be echoed and which shouldn’t.

Endnotes

[1] for example: statement 2E2F does not properly distinguish between the "occurrence" of a break character and the "occurrence" of a Transmission character. The present RFC shows that they are fundamentally different.