NETWORKING ANSI COLOR GRAPHICS

by Brian A. Lantz/KO4KS
TPALAN Tampa Local Area Network

BACKGROUND

A typical network supports two kinds of users; regular users and super-users. The regular users provide the NEED for the network. Without those who wish to USE the network, the network is useless! It doesn't matter if it is ROSE, NETROM, TCP/IP, or SplatRouter-2000; if there are no users, there is no need.

The second class of users, super-users, try to MEET the needs of the regular users. There are usually BBSs, but can also be information providers, Internet mail routers, non-radio wormholes to distant places, and any other resources that might be needed or desired.

While both classes of users should be given equal status pertaining to the use of network resources, the regular users, obviously, should be given special consideration. Their needs and desires should be constantly evaluated by the local administrators of the network, as well as by the writers of packet software.

The users in the early days of packet, the users had low expectations, since this was all a new experience. The users of 1993 have developed a desire to do more. They want new capabilities. They want to be able to do things more like landline networks. This brings us to ANSI color graphics.

This paper will briefly describe the needs, the shortcomings, the facts, and the solutions to making our networks support ANSI color graphics.

THE NEEDS

ANSI color graphics might seem to some to be only unnecessary fluff. In fact, they are correct! But so is almost all non-emergency, Amateur communications. In Amateur radio, perceived needs determine what is necessary.

The landline BBS world has for many years greeted its users with ANSI color text and graphics. While this doesn't change the content, it does make it ***seem*** friendlier and nicer. This environment has led to many public domain programs for creating ANSI color portraits, allowing the user to express his/her self with crude but expressive tools.

Once this budding artist has a creation that he has developed a deal of pride over, he now wants to SHARE the portrait with his peers. Acceptance in the ANSI art galleries is very important to those who have labored so long and hard.

So, the artist gleefully posts his creation in a message on the packet BBS system and sends messages to all his/her friends to tell them to check it out. What follows has broken lesser men, caused nations to crumble, and cancelled many good TV series.

The friends all send back messages saying that SOMETHING HAPPENED to the creation and it was not received as it must have been sent!

THE SHORTCOMINGS - BBS CHARACTER HANDLING

The current BBS software in use have totally different features and functionality, but they keep the message handling portions compatible. They might have OTHER WAYS to do these features in addition to the standard (i.e. SP, SB), but the interface for sending mail IS consistent.

Most of the BBSs have one or more control characters that can be used to abort the message, disconnect, etc. These character fall in the range of characters with values from 0 to 31. These characters (with the exception of the carriage return and line feed) are not usually used in ANSI color files. Because of this, it does not SEEM like there is a problem.

The problem starts, though, with ANSI graphic characters, which fall into the range of values 128 through 255 (characters with the most significant bit set). Some BBS software cannot handle characters in this range. They either bit-bucket these characters, choke on them, or strip off the eighth bit.

While most BBSs don't have a problem with MOST of these characters, a few have a problem with at least one of them, the character value 255 ($FF). The reason for this is-that character oriented libraries usually return a value of -1 to indicate an error or the end of file. If you are: using, for example, the C programming language, and define the variable that you are filling with the next character to be a SIGNED CHARACTER (the default) rather than an UNSIGNED CHARACTER, that -1 looks the same as a 255 that has been sign-extended.

THE SHORTCOMINGS - TNC CHARACTER HANDLING

TNC firmware contributes to the problem, in most cases. Each manufacturer has commands that allow the 8th bit to be stripped from data coming in. These need to be properly set by each user.

In addition, most TNC firmware pre-processes the data that gets placed into its Mini-BBS. This pre-processing CAN corrupt the data.

THE FACTS - ABOUT ANSI CONTROL SEQUENCES

The biggest problems come from improper perception and inefficient packaging of the ANSI sequences within a packet.

While this is not the place for a complete dissertation of the details of how ANSI control sequences are handled, a little explanation is necessary for those who have not dealt with them before.

An ANSI control sequence starts with the ESCAPE character (27 decimal, $1B hexadecimal). This is followed by the left bracket ('[') character. All ANSI sequences begin with these two characters.

The length of an ANSI sequence is variable. After the beginning sequence, you may have zero or more numbers (in ASCII) separated by semi-colons (';'). Each number can be from 0 to 255. These numbers are the parameters for the ANSI command.

An ANSI sequence is concluded with a single alpha character (A-Z or a-z). This command code determines the function that this ANSI sequence is instructing to occur. A good source of reference for most of the normal ANSI sequences is the MS-DOS version *5.0 User's* Guide *and Reference,* pages 593-600 (ANSI.SYS).

```
$1B          [   1        40   H
```

EXAMPLE 1 - Sample ANSI control sequence

In the example, the "ESC [" starts the sequence, it has two parameters (1 and 40), and the ANSI command code is "H". The "H" command sequence is a cursor positioning sequence. This particular example moves the cursor to line 1, column 40 of the display. Note the semi-colon between the two numbers.

Since all of these characters can be generated from a keyboard, the ANSI standard helps keep from generating FALSE sequences by placing an inter-character timer into the ANSI parser. Any pause between the beginning ESCAPE character and the terminating command code causes the parser to treat the interrupted sequence as an ordinary sequence of characters.

This gets complicated by packet radio, where the first portion of the sequence could be at the end of one packet, and the end of the sequence being found in the next packet. They could be separated by several seconds, far long enough for the ANSI parser to determine that it is not to be treated specially.

While there are some existing problems with the handling of 8-bit characters, most of the ANSI woes of the users are because of this broken sequencing.

THE SOLUTIONS

First, BBS and TNC authors, get into your code and find out what it is REALLY doing with characters that have the 8th bit set. Re-think your defaults in regards to these. The majority of the users are NOT using dumb terminals that need the 8th bit stripped. 8th bit transparency should be the default. Always treat your data buffers as UNSIGNED CHARACTERS. Try to make as few assumptions and limitations on the data as you can.

Second, TNC Mini-BBSs should store 8th bit characters correctly. By the time of this conference, PacComm should already be shipping release 3.2 of our Amateur firmware, which supports color file messages in the PMS.

Third, BBS authors should consider adding the few lines of code needed to properly packetize ANSI sequences. In TNOS all I had to do was check the queued data size when I came across an ANSI sequence. If the size was above 200 characters of data, I would flush the packet data BEFORE starting the ANSI sequence. The result is that the ANSI sequence begins a new packet and is never broken in the midst. That one simple change means the difference between a perfectly displayed creation, or a horrible fake.

CONCLUSION

While many of us are ready to start tackling digital voice across a network, some have much simpler needs. And while it may seem minor, this is the first step to such bigger hurdles. We can't begin to pass other more complex data forms over the network, if we can't work together on this one.

The Protocol Wars are (hopefully) over, and one things that I hope that we've all learned from them is that different people approach similar needs in different ways. The Vulcan edict IDIC stands for "Infinite Diversity in Infinite Combinations". Our differences make us (collectively) better, when we can work together and learn from each other.

Brian A. Lantz
Manager of Software Development, PacComm Packet Radio Systems
President, Tampa Local Area Network
Packet: KO4KS@KO4KS.#TPA.FL.USA.NA
Internet: brianlantz@delphi.com