The KISS TNC: A simple Host-to-TNC communications protocol

Mike Chepponis, K3MC
Phil Karn, KA9Q

ABSTRACT

The KISS TNC provides direct computer to TNC communications using a simple protocol described here. Many TNCs now implement it, including the TAPR TNC-1 and TNC-2 (and their clones), the venerable VADCG TNC, the AEA PK-232/PK-87 and all TNCs in the Kantronics line. KISS has quickly become the protocol of choice for TCP/IP operation and multi-connect BBS software.

1. Introduction

Standard TNC software was written with human users in mind; unfortunately, commands and responses well suited for human use and ill-adapted for host computer use, and vice versa. This is especially true for multi-user servers such as bulletin boards which must multiplex data from several network connections across a single host/TNC link. In addition, experimentation with new link level protocols is greatly hampered because there may very well be no way at all to generate or receive frames in the desired format without reprogramming the TNC.

The KISS TNC solves these problems by eliminating as much as possible from the TNC software, giving the attached host complete control over and access to the contents of the HDLC frames transmitted and received over the air. This is central to the KISS philosophy: the host software should have control over all TNC functions at the lowest possible level.

The AX.25 protocol is removed entirely from the TNC, as are all command interpreters and the like. The TNC simply converts between synchronous HDLC spoken on the full- or half-duplex radio channel, and a special asynchronous, full-duplex frame format spoken on the host/TNC link. Every frame received on the HDLC link is passed intact to the host once it has been translated to the asynchronous format; likewise, asynchronous frames from the host are transmitted on the radio channel once they have been converted to HDLC format.

Of course, this means that the bulk of AX.25 (or another protocol) must now be implemented on the host system. This is acceptable, however, considering the greatly increased flexibility and reduced overall complexity that comes from allowing the protocol to reside on the same machine with the applications to which it is closely coupled.

It should be stressed that the KISS TNC is intended only as a stopgap. Ideally, host computers would have HDLC interfaces of their own, making separate TNCs unnecessary. [15] Unfortunately, HDLC interfaces are rare, although they are starting to appear for the IBM PC. The KISS TNC therefore becomes the “next best thing” to a real HDLC interface, since the host computer only needs an ordinary asynchronous interface.

“Keep It Simple, Stupid”

2. Asynchronous Frame Format

The “asynchronous packet protocol” spoken between the host and TNC is very simple, since its only function is to delimit frames. Each frame is both preceded and followed by a special FEND (Frame End) character, analogous to an HDLC flag. No CRC or checksum is provided. In addition, no RS-232C handshaking signals are employed.

The special characters are:

Abbreviation	Description	Hex value
FEND	Frame End	C0
FESC	Frame Escape	DB
TFEND	Transposed Frame End	DC
TFESC	Transposed Frame Escape	DD

The reason for both preceding and ending frames with FENDs is to improve performance when there is noise on the asynch line. The FEND at the beginning of a frames serves to “flush out” any accumulated garbage into a separate frame (which will be discarded by the upper layer protocol) instead of sticking it on the front of an otherwise good frame. As with back-to-back flags in HDLC, two FEND characters in a row should not be interpreted as delimiting an empty frame.

3. Transparency

Frames are sent in 8-bit binary; the asynchronous link is set to 8 data bits, 1 stop bit, and no parity. If a FEND ever appears in the data, it is translated into the two byte sequence FESC TFEND (Frame Escape, Transposed Frame End). Likewise, if the FESC character ever appears in the user data, it is replaced with the two character sequence FESC TFESC (Frame Escape, Transposed Frame Escape).

As characters arrive at the receiver, they are appended to a buffer containing the current frame. Receiving a FEND marks the end of the current frame. Receipt of a FESC puts the receiver into “escaped mode”, causing the receiver to translate a following TFESC or TFEND back to FESC or FEND, respectively, before adding it to the receive buffer and leaving escaped mode. Receipt of any character other that TFESC or TFEND while in escaped mode is an error; no action is taken and frame assembly continues. A TFEND or TFESC received while not in escaped mode is treated as an ordinary data character.

This procedure may seem somewhat complicated, but it is easy to implement and recovers quickly from errors. In particular, the FEND character is never sent over the channel except as an actual end-of-frame indication. This ensures that any intact frame (properly delimited by FEND characters) will always be received properly regardless of the starting state of the receiver or corruption of the preceding frame.

This asynchronous framing protocol is identical to “SLIP” (Serial Line IP), a popular method for sending ARPA IP datagrams across asynchronous lines. It could also form the basis of an asynchronous amateur packet radio link protocol that avoids the complexity of HDLC on slow speed channels.

4. Control of the KISS TNC

Each asynchronous data frame sent to the TNC is converted back into “pure” form and queued for transmission as a separate HDLC frame. Although removing the human interface and the AX.25 protocol from the TNC makes most existing TNC commands unnecessary (i.e., they become host functions), the TNC is still responsible for keying the transmitter’s PTT line and deferring to other activity on the radio channel. It is therefore necessary to allow the host to control a few TNC parameters, namely the transmitter keyup delay, the transmitter persistence variables and any special hardware that a particular TNC may have.

To distinguish between command and data frames on the host/TNC link, the first byte of each asynchronous frame between host and TNC is a “type” indicator. This type indicator byte is broken into two 4-bit nibbles so that the low-order nibble indicates the command number (given in the table below) and the high-order nibble indicates the port number for that particular command. In systems with only one HDLC port, it is by definition Port 0. In multi-port TNCs, the upper 4-bits of the type indicator byte can specify one of up to sixteen ports. The following commands are defined in frames to the TNC (the “Command” field is in hexadecimal):

Command	Function	Comments
0	Data Frame	The rest of the frame is data to be sent on the HDLC channel.
1	TXDELAY	The next byte is the transmitter keyup delay in 10 ms units. The default start-up value is 50 (i.e., 500 ms).
2	P	The next byte is the persistence parameter, p, scaled to the range 0 - 255 with the following formula: $P = p * 256 - 1$ The default value is P = 63 (i.e., p = 0.25).
3	SlotTime	The next byte is the slot interval in 10 ms units. The default is 10 (i.e., 100 ms).
4	TXtail	The next byte is the time to hold up the TX after the FCS has been sent, in 10 ms units. This command is obsolete, and is included here for compatibility with some existing implementations.
5	FullDuplex	The next byte is 0 for half duplex, nonzero for full duplex, The default is 0 (i.e., half duplex).
6	SetHardware	Specific for each TNC. In the TNC-1, this command sets the modem speed. Other implementations may use this command for other hardware-specific functions.
FF	Return	Exit KISS and return control to a higher-level program. This is useful only when KISS is incorporated into the TNC along with other applications.

The following types are defined in frames to the host:

Type	Function	Comments
0	Data Frame	Rest of frame is data from the HDLC channel.

No other types are defined; in particular, there is no provision for acknowledging data or command frames sent to the TNC. KISS implementations must ignore any unsupported command types. All KISS implementations must implement commands 0,1,2,3 and 5; the others are optional.

5. Buffer and Packet Size Limits

One of the things that makes the KISS TNC simple is the deliberate lack of TNC/host flow control. The host computers run a higher level protocol (typically TCP, but AX.25 in the connected mode also qualifies) that handles flow control on an end-to-end basis. Ideally, the TNC would always have more buffer memory than the sum of all the flow control windows of all of the logical connections using it at that moment. This would allow for the worst case (i.e., all users sending simultaneously). In practise, however many (if not most) user connections are idle for long periods of time, so buffer memory may be safely "overbooked". When the occasional "bump" occurs, the TNC must drop the packet gracefully, i.e., ignore it without crashing or losing packets already queued. The higher level protocol is expected to recover by "backing off" and retransmitting the packet at a later time, just as it does whenever a packet is lost in the network for any other reason. As long as this occurs infrequently, the performance degradation is slight; therefore the TNC should provide as much packet buffering as possible, limited only by available RAM.

Individual packets at least 1024 bytes long should be allowed. As with buffer queues, it is recommended that no artificial limits be placed on packet size. For example, the K3MC code running on a TNC-2 with 32K of RAM can send and receive 30K byte packets, although this is admittedly rather extreme. Large packets reduce protocol overhead on good channels. They are essential for good performance when operating on high speed modems such as the new WA4DSY 56 kbps design.

6. Persistence

The P and SlotTime parameters are used to implement true p-persistent CSMA. This works as follows:

Whenever the host queues data for transmission, the TNC begins monitoring the carrier detect signal from the modem. It waits indefinitely for this signal to go inactive. When the channel clears, the TNC generates a random number between 0 and 1. If the number is less than or equal to the parameter *p*, the TNC keys the transmitter, waits .01 * TXDELAY seconds, and transmits all queued frames. The TNC then unkeys the transmitter and goes back to the idle state. If the random number is greater than *p*, the TNC delays .01 * SlotTime seconds and repeats the procedure beginning with the sampling of the carrier detect signal. (If the carrier detect signal has gone active in the meantime, the TNC again waits for it to clear before continuing). Note that $p = 1$ means "transmit as soon as the channel clears"; in this case the p-persistence algorithm degenerates into the 1-persistent CSMA generally used by conventional AX.25 TNCs.

p-persistence causes the TNC to wait for an exponentially-distributed random interval after sensing that the channel has gone clear before attempting to transmit. With proper tuning of the parameters *p* and SlotTime, several stations with traffic to send are much less likely to collide with each other when they all see the channel go clear. One transmits first and the others see it in time

---

To conform to the literature, here *p* takes on values between 0 to 1. However fractions are difficult to use in a fixed point microprocessor so the KISS TNC actually works with *P* values that are rescaled to the range 0 to 255. To avoid confusion, we will use lower-case *p* to mean the former (0-1) and upper-case *P* whenever we mean the latter (0-255).

to prevent a collision, and the channel remains stable under heavy load. See references [1] through [13] for details.

We believe that optimum *p* and SlotTime values could be computed automatically. This could be done by noting the channel occupancy and the length of the frames on the channel. We are proceeding with a simulation of the p-persistence algorithm described here that we hope will allow us to construct an automatic algorithm for *p* and SlotTime selection.

We added p-persistence to the KISS TNC because it was a convenient opportunity to do so. However, it is not inherently associated with KISS nor with new protocols such as TCP/IP. Rather, persistence is a *channel access* protocol that can yield dramatic performance improvements regardless of the higher protocol in use; we urge it be added to *every* TNC, whether or not it supports KISS.

7. Implementation History

The original idea for a simplified host/TNC protocol is due to Brian Lloyd, WB6RQN. Phil Karn, KA9Q, organized the specification and submitted an initial version on 6 August 1986. As of this writing, the following KISS TNC implementations exist:

TNC type	Author	Comments
TAPR TNC-1 & clones	Marc Kaufman, WB6ECE	Both download and dedicated ROM versions.
TAPR TNC-2 & clones	Mike Chepponis, K3MC	First implementation, most widely used. Exists in both download and dedicated ROM versions.
VADCG TNC & Ashby TNC	Mike Bruski, AJ9X	Dedicated ROM.
AEA PK-232 & PK-87	Steve Stuart, N6IA	Integrated into standard AEA firmware as of 21 January 1987. The special commands “KISS ON” and “KISS OFF” (!) control entry into KISS mode.
Kantronics	Mike Huslig	Integrated into standard Kantronics firmware as of July 1987.

The AEA and Kantronics implementations are noteworthy in that the KISS functions were written by those vendors and integrated into their standard TNC firmware. Their TNCs can operate in either KISS or regular AX.25 mode without ROM changes. The TNC-1 and TNC-2 KISS versions were written by different authors that the original AX.25 firmware. Because of the specialized development environment used for the TNC-1 code, and because original source code for the TNC-2 was not made available, the KISS authors wrote their code independently of the standard AX.25 firmware. Therefore these TNCs require the installation of nonstandard ROMs. Two ROMs are available for the TNC-2. One contains “dedicated” KISS TNC code; the TNC operates only in KISS mode. The “download” version contains standard N2WX firmware with bootstrap loader overlay. When the TNC is turned on or reset, it executes the loader. The loader will accept a memory image in Intel Hex format, or it can be told to execute the standard N2WX firmware through the “H” command. The download version is handy for occasional KISS operation, while

---

For “Howie” of course.

the dedicated version is much more convenient for full-time or demo KISS operation.

The code for the TNC-1 is also available in both download and dedicated versions. However, at present the download ROM contains only a bootstrap; the original ROMs must be put back in to run the original TNC software.

8. Credits

The combined "Howie + downloader" ROM for the TNC-2 was contributed by WA7MXZ. This document was expertly typeset by Bob Hoffman, N3CVL.

9. Bibliography

1. Tanenbaum, Andrew S.: "Computer Networks" pp. 282-292. Prentice-Hall 1981.

2. Tobagi, F. A.: "Random Access Techniques for Data Transmission over Packet Switched Radio Networks," Ph.D. thesis, Computer Science Department, UCLA, 1974.

3. Kleinrock, L., and Tobagi, F.: "Random Access Techniques for Data Transmission over Packet-Switched Radio Channels," *Proc. NCC*, pp. 187-201, 1975.

4. Tobagi, F. A., Gerla, M., Peebles, R. W., and Manning, E. G.: "Modelling and Measurement Techniques in Packet Communications Networks," *Proc. IEEE*, vol. 66, pp. 1423-1447, Nov. 1978.

5. Lam, S. S.: "Packet Switching in a Multiaccess Broadcast Channel", Ph.D. thesis, Computer Science Department, UCLA, 1974.

6. Lam, S. S., and Kleinrock, L.: "Packet Switching in a Multiaccess Broadcast Channel: Dynamic Control Procedures," *IEEE Trans. Commun.*, vol COM-23, pp. 891-904, Sept. 1975.

7. Lam, S. S.: "A Carrier Sense Multiple Access Protocol for Local Networks," *Comput. Networks,* vol 4, pp. 21-32, Feb. 1980.

8. Tobagi, F. A.: "Multiaccess Protocols in Packet Communications Systems," *IEEE Trans. Commun.*, vol COM-28, pp. 468-488, April 1980.

9. Bertsekas, D., and Gallager, R.: "Data Networks", pp. 274-282 Prentice-Hall 1987.

10. Kahn, R. E., Gronemeyer, S. A., Burchfiel, J., and Kungelman, R. C.: "Advances in Packet Radio Technology," *Proc. IEEE.* pp. 1468-1496. 1978.

11. Takagi, H.: "Analysis of Polling Systems," Cambridge, MA MIT Press 1986.

12. Tobagi, F. A., and Kleinrock, L.: "Packet Switching in Radio Channels: Part II - The Hidden Terminal Problem in CSMA and Busy-Tone Solution," *IEEE Trans. Commun.* COM-23 pp. 1417-1433. 1975.

13. Rivest, R. L.: “Network Control by Bayessian Broadcast,” Report MIT/LCS/TM-285. Cambridge, MA. MIT, Laboratory for Computer Science. 1985.

14. Karn, P. and Lloyd, B.: “Link Level Protocols Revisited,” *ARRL Amateur Radio Fifth Computer Networking Conference*, pp. 5.25-5.37, Orlando, 9 March 1986.

15. Karn, P.: “Why Do We Even Need TNCs Anyway,” *Gateway*, vol. 3 no. 2, September 5, 1986.