More experiments with Delay and Disruption Tolerant Networking over AX.25 Networks

John Ronan, EI7IG
TSSG, Waterford I.T.
Cork Road, Waterford, Ireland
jronan@tssg.org

Darren Long, G0HHW
Bury St. Edmonds
England
darren.long@mac.com

Kristian Walsh
Earlwood Estate
The Lough
County Cork, Ireland
walshk@byteform.com

***Abstract*—Previously the authors gave an overview of a Delay Tolerant Network Convergence Layer implementation that operates over Connected Mode AX.25, and detailed the results of some performance tests using it to transfer data over Amateur Radio channels. These results were compared to both a native AX.25 and a TCP/IP-over-AX.25 implementation. The investigation of TCP/IP was undertaken because, while it is generally accepted that TCP is unsuited to wireless links, it has become the dominant protocol in real-world applications, with over 50% of internet traffic now accounted for by TCP over port 80 [1].**

As some issues were highlighted in experiments leading to the authors' prior publication, these have been worked on and have been largely resolved. It was also found that our model for an ideal AX.25 communications channel had deficiencies, so a correction is offered. Additionally, we also tested our Convergence Layer alongside a TCP/IP over AX.25 implementation on both a 1200 and 9600 baud point-to-point link and give comparative results between our Convergence Layer implementation and TCP/IP. Real-world behaviour of the data link still diverges from the model, but the authors provide some possible reasons for this.

***Keywords*-Disruption tolerant networking; Internetworking; Packet radio networks; TCPIP; Transport Protocols**

I. Introduction

Previously, the authors gave an overview of a Delay Tolerant Network (DTN) Convergence Layer (CL) implementation which was under development [2]. This was developed as a Convergence Layer in the DTNRG's DTN2 reference implementation on the Linux platform using its AX.25 stack.

During our testing we found some issues with our AX.25 Connected Mode Convergence Layer (AX.25CM-CL) implementation and we have worked to fix those and improve efficiency.

We proposed that the use of DTN over AX.25 as an alternative may allow for a more *ad hoc*, self-configuring network to be created [2].

Due to the low transmission speed of AX.25 links (typically 1200 bits-per-second) we compared our AX.25CM-CL and TCP/IP to an ideal AX.25 model. Since then we have improved the efficiency of the AX.25CM-CL, adjusted the TCP/IP parameters for better performance, and also corrected a mistake in our model of an "ideal" AX.25 link.

II. DTN Convergence Layer Implementation

The DTN2 reference implementation is provided as a flexible software framework for experimentation, extension and real-world deployment of Delay Tolerant Networking systems [3]. We have taken this framework and used it to produce a Convergence Layer for the AX.25 networking protocol.

The AX.25CM-CL is a convergence layer implementation for AX.25 sockets on the Linux platform which transports the DTN "bundles" described by RFC-5050 [4] directly over an AX.25 connection that operates solely as a Layer 2 protocol. In this respect, the AX.25CM-CL is similar to the existing Bluetooth CL.

Currently, the only major difference between the AX.25CM-CL and the TCP-CL [5] Protocol is that the AX.25 implementation is extended to include a 32-bit CRC appended to each TCP-CL Protocol segment. This is necessary in order to ensure that any corruption of AX.25 KISS [6] data frames

can be detected[1], as well as providing additional means to detect protocol errors introduced by the implementation.

A. Capabilities and Limitations

The AX.25CM-CL code has been in active development since January 2007, when it was first branched from the TCP-CL and Oasys support classes. Currently, the AX.25CM-CL allows point-to-point links between two peers, and also paths containing a single repeater operating at the AX.25 link layer.

At time of writing, no announcement or discovery mechanism had been implemented and therefore links have to be manually configured and initiated. However, it has been accepted into the reference implementation. It is envisaged that existing announcement and discovery mechanisms could be adapted to work within the framework of the AX.25 Protocol.

III. AX.25 Theoretical Performance

In assessing the performance of the DTN implementation, it is useful to consider the theoretical performance limits of the underlying data transport. In this case, that transport is the AX.25 Link Access Protocol for Amateur Packet Radio [7], a data link layer protocol derived from the ITU-T X.25 data link protocol [8] with modifications for use by Amateur Radio operators.

A. Experimental settings

Table I lists the significant parameters of the radio link used in these experiments as in our previous paper.

TABLE I
Characteristics of experimental transfer

parameter	value
link speed, bit/s	1200
T_{slot}, ms	20
$T_{txdelay}$, ms	150
T_{tail}, ms	20
p	0.250
data length, bytes	7182

[1]In theory corruption should not happen, but it does in practice.

While we have kept these the same as in our previous paper. It should be noted that using the Kenwood radios, with built in radio modems, it is possible to reduce the Transmit delay, $T_{txdelay}$ to 100ms and still maintain a reliable connection.

B. Model Transfer times

AX.25 is most commonly deployed on half-duplex radio links, with link access managed using a p-persist CSMA algorithm [9] [10]. Transfer times on such networks have a small probabilistic component, as a random delay is used for Media Access control. To minimise the effect of collisions on the experimental results, a point-to-point link was used on an unused UHF frequency, and the frequency was continually monitored for any other users during the running of all tests. The probabilistic factor, p, was set to 0.25, which entails an average delay of $0.25T_{slot}$ to each transmission.

T_{frame}, the transmission time, in seconds, for one data frame is obtained using the following formula:

$$T_{frame} = \frac{63}{62} \times \frac{(bits) + 160}{(bitrate)}$$

where 160 is the number of bits comprising the AX.25 preamble, header, check sequence, and end-of-frame marker.

As in HDLC, a zero is inserted after every five consecutive "ones" in order to make sure that there is no ambiguity about the location of delimiters which have the value 01111110. Such an extra bit will occur on average every 62 bits [11].

Each acknowledgement is a single transmission of a frame with no payload. Allowing for transmission setup and release times, T_{ack}, the average time required to send an acknowledgement is:

$$T_{ack} = T_{txdelay} + \frac{63}{62} \times \frac{160}{bitrate} + T_{txtail} + p \times T_{slot}$$

The number of acknowledgements sent depends on the acknowledgement window size for the link, $maxframe$: where a $maxframe$ of greater than 1 is chosen, the transmitter is allowed to send multiple data frames in one transmission, which eliminates all but one set of $T_{txdelay}$ and T_{txtail} delays in each group of $maxframes$ frames, as illustrated in Figure 1(b).

As each acknowledgement window of data packets is sent in a single transmission, and each such transmission will generate one acknowledgement, the following formula for transmission time of a message containing $frames$ number of data frames can be easily derived:

$$windows = Ceiling(\frac{frames}{windowsize})$$

$$\begin{aligned} T_{message} &= frames \times T_{frame} \\ &+ windows \times \\ &(T_{txdelay} + T_{txtail} + p \times T_{slot} + T_{ack}) \end{aligned}$$

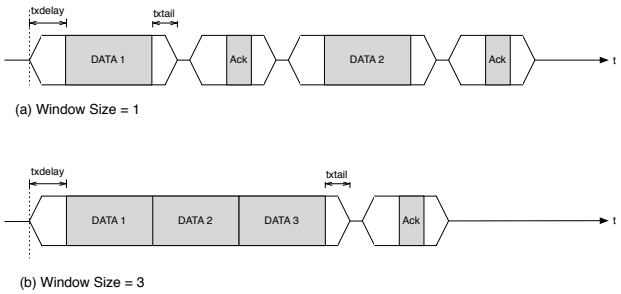

Fig. 1. Effect of increased window sizes on link efficiency.

For a window size of 1, each packet requires an acknowledgement, which will negatively affect the throughput of the link. The nature of the physical link used means that these acknowledgement frames incur a very high cost. On a radio data link, each transmission, be it a single frame, or a group of frames, must also include an initial period of time, $T_{txdelay}$ to allow the transmitter to stabilise before data can be sent. Figure 1 shows how increasing the window size can reduce the amount of time required to send data.

Using the experimental parameters in Table I, we calculate the frame transmission time for a transfer of n 255-byte frames as follows. First, the transfer time for a single frame, without link stabilisation or release delays, T_{frame}, is determined, as this is constant regardless of the size of the acknowledgment window:

$$T_{frame} = \frac{63}{62} \times \frac{(255 \times 8) + 160}{1200} \approx 1.8629s$$

T_{ack}, the time required to send an acknowledgement, is also constant for all window sizes:

$$\begin{aligned} T_{ack} &= T_{txdelay} + \frac{63}{62} \times \frac{160}{bitrate} + T_{txtail} + p \times T_{slot} \\ &= 0.150 + \frac{63}{62} \times \frac{160}{1200} + 0.25 \times 0.020 \\ &\approx 0.2904s \end{aligned}$$

Using these values, and the formula for $T_{message}$, previously, the values in Table II were obtained.

Changing the bit rate to 9600 bits-per-second, and using the formula for $T_{message}$, previously, the values in Table III were obtained.

TABLE II
THEORETICAL MINIMUM TRANSFER TIMES, RAW AX.25 TRANSFER, 1200 BPS

Window size	timings from model (seconds)
1	68.0
2	61.2
3	58.8
4	57.8
5	56.9
6	56.4
7	56.4

* values are same for window sizes of 6 or 7 as both settings generate only 5 acknowledgement frames for a 7182-byte transfer

TABLE III
THEORETICAL MINIMUM TRANSFER TIMES, RAW AX.25 TRANSFER, 9600 BPS

Window size	timings from model (seconds)
1	17.4
2	12.2
3	10.4
4	9.7
5	8.9
6	8.6
7	8.6

* values are same for window sizes of 6 or 7 as both settings generate only 5 acknowledgement frames for a 7182-byte transfer

It should be noted that these figures do not account for collisions, interference or the delay incurred by the transfer of data between the host system and the AX.25 radio modem over the RS-232 serial interfaces [12]. As these model figures

do not take account of these additional overheads or the time required to process higher-level protocol commands for transmission, none of the experimental results were expected to reach this level of performance; the figures here serve primarily to define the performance envelope of the experiment.

IV. Experimental Network

A. Network

Figure 2 shows the experimental network used to measure the system performance.

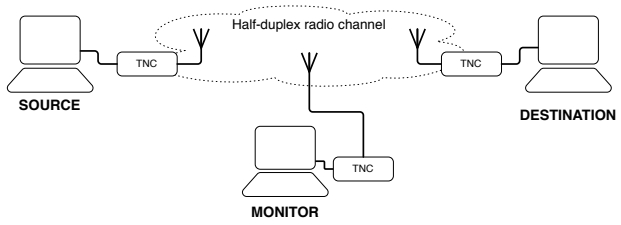

Fig. 2. Experimental setup used to measure AX.25 performance. Source and Destination devices were connected on a single RF data channel (i.e., half-duplex)

Equipment used for the source node was a Kenwood TM-D710E with an integrated radio modem. For the destination node, a Kenwood TH-D72E also with an integrated radio modem was used.

A third transceiver and TNC was used to monitor the radio channel to log all transmitted AX.25 frames and allow for the measuring of transfer times.

The monitor used a Kenwood TH-D7 with integrated radio modem. All antennas were in close proximity (less than 10 metres), thus power levels were kept low at 5 Watts or less where possible.

To obtain a valid set of readings, ten transfers of the candidate test file were completed for each setting of the AX.25 window parameter ($maxframe$). These readings were then combined using a simple average in order to give an indicative time for the given window setting.

The test file used contained 7182 bytes. When it came to testing using TCP/IP, both TNCs were first configured into KISS mode and then the Maximum Transmission Unit (MTU) and window sizes were set on both Linux hosts, according to Table IV, before each transfer commenced. This was to ensure coherence between the AX.25 and TCP/IP windowing. Transfer of the file data for TCP/IP tests was performed using the FTP protocol.

TABLE IV
TCP/IP test settings. MTU, MSS & TCP Window are all in Bytes

Window Size	MTU	MSS	Window
1	168	128	256
2	296	256	512
3	424	384	768
4	552	512	1024
5	680	640	1280
6	808	768	1536
7	936	896	1792

For the AX.25CM-CL test, The *dtncp* utility was used to send the test file.

Obviously the *ftp*, and *dtncp* applications add their own small amount of overhead to the file transfer (above that already added by AX.25). However, it was considered to be valid to include this in the final results, as the amount of additional data is quite small in relation to the file being transferred, and will be representative of "real world" usage.

That said, for the purposes of generating comparable data, great lengths were taken to make sure that there were no collisions at the MAC layer[2], thus removing one unknown. Consequently, we are confident that the figures obtained are a true and accurate reflection of the performance of the protocols tested in an ideal RF environment.

V. Discussion

Tables V and VII list the results obtained for transfers between the two TNC devices. As was previously mentioned, it was not expect that the actual transfer times would approach those of the model, but the gap here is quite large. One possible explanation is that collisions occurred during transmission, forcing a re-broadcast of certain packets. However, during the experiment, great care was taken to make sure that there were no collisions at the physical layer, so this cause can be eliminated.

The timing model does not account for buffering and host-to-TNC data transfers, and it is conceivable that these are responsible for the observed

[2]great care was taken to monitor the frequency for any interference during the tests

shortfall in performance. True to its name, the KISS protocol used here favours simple implementation over performance, and is controlled entirely by the host computer.

TABLE V
COMPARISON, 1200BPS

transfer times all in seconds

Window	Model	AX.25-CL	*TCP/IP*
1	68.0	110.36	169.18
2	61.2	82.82	114.91
3	58.8	73.64	96.45
4	57.8	70.45	90.36
5	56.9	65.64	88.18
6	56.4	63.73	86.09
7	56.4	64.55	86.82

The discrepancy between model and actual results was inversely proportional to the transmission size, and a simple division of the model-to-experiment error by the number of transmitted groups yielded a strong correlation between number of TX/RX swaps and the additional model delay which can be see in Table VI.

TABLE VI
MODEL VS AX.25CM-CL ERROR, 1200BPS

all times all in seconds

Window	TX/RX cycles	Difference	per group
1	29	42.4	1.46
2	15	21.6	1.44
3	10	14.8	1.48
4	8	12.6	1.58
5	6	8.8	1.46
6	5	7.4	1.47
7	5	8.2	1.63

The approximate 1.4 second additional "turnaround" time between sending one group of packets and the next may be due to multiple buffering in the chain from transmitter to receiver, slow computation of frame checksums by the devices, interrupt latency, or any number of other factors. Adapting the model to account for this constant delay would improve its accuracy for this experiment, but there is no guarantee that a different combination of TNC and transmission equipment would exhibit the same intrinsic delays.

TABLE VII
COMPARISON, 9600BPS

transfer times all in seconds

Window	Model	AX.25-CL	*TCP/IP*
1	17.4	56.82	71.50
2	12.2	33.82	42.91
3	10.4	25.00	34.10
4	9.7	24.27	30.50
5	8.9	18.64	27.00
6	8.6	16.91	25.27
7	8.6	16.80	26.09

A. Problems with large window sizes in DTN

Previously a bug was encountered a bug such that on the fifth consecutive run with $maxframe$ set to 4, the AX.25 implementation in the host computer appeared to enter an unstable condition: instead of obeying the chosen $maxframe$ setting, the source unit flooded the receiver with all the frames (over 20) of the data transfer, causing a breakdown in flow control for both source and destination [2].

On examination of the logs, it was determined that the AX.25 T1 timer (How long AX.25 will wait before retransmitting an unacknowledged frame) needed to be increased from its default value of 10 seconds for an AX.25 window of 4 or more. With an AX.25 window of 4, it took almost exactly 10 seconds to receive an acknowledgement. Once the T1 timer was increased sufficiently, the problem disappeared.

B. Resolving the issue of over-acknowledgement

Previously, the AX25CM-CL produced a flurry of (TCP-CL Protocol) segment ACK messages (acknowledgements - one for each segment/frame) and sends these as distinct frames (Figure 3). This was due to the our implementation sending a TCP-CL ACK for every TCP-CL segment.

The purpose of the TCP-CL Protocol ACK is to allow for reactive fragmentation of bundles in scenarios where an AX.25 connection is dropped. We chose the approach of allowing the specification of an acknowledgement window option `ack_window`, in units of TCP-CL Protocol segments, in the AX25CM-CL link configuration parameters. This allows the granularity over which re-

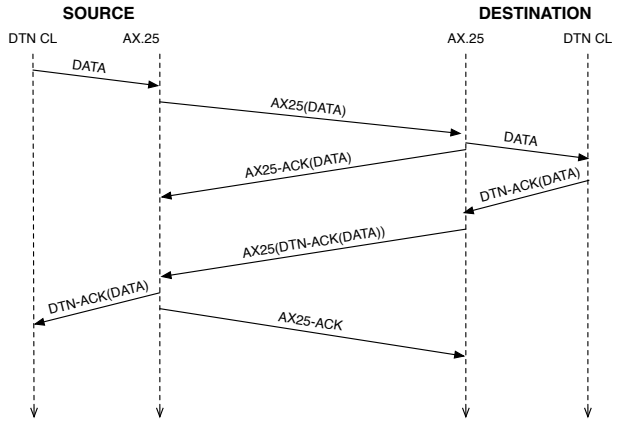

Fig. 3. Multiple acknowledgements produced by the DTN CL

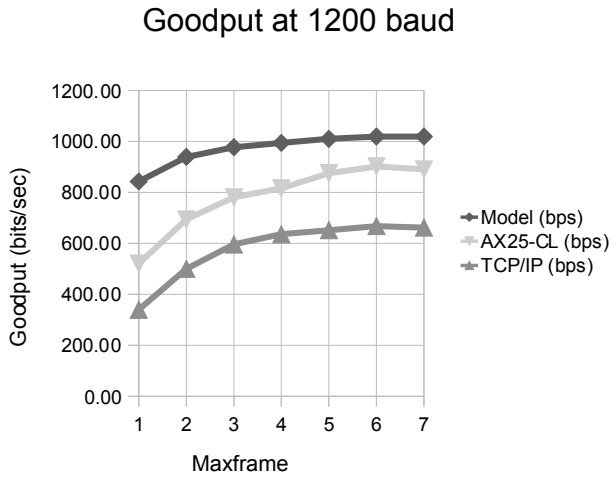

Fig. 4. Comparison of transfer times for AX.25CM-CL and TCP/AX.25 with reference to theoretical model at 1200bps.

active fragmentation is performed to be defined on a link by link basis. If links are stable on dedicated channels, then a higher value of `ack_window` can be specified (e.g. 32). If link opportunities are transient, then shorter `ack_window` values are recommended (e.g. 8, the default). All completely transferred bundles are acknowledged immediately, so as to avoid redundant retransmission.

C. Possible bug in the TH-D72 Firmware

This issue manifested itself as an occasional "beep" from the TH-D72, with the TH-D72 ceasing transmissions. There seemed to be two different bugs. In the first case, as the TH-D72 attempted to transmit, a "beep" was heard from the radio and the packet sent to the TH-D72 for transmission did not

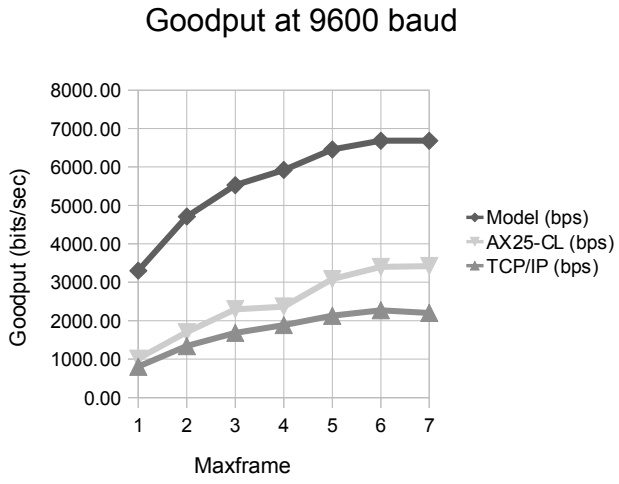

Fig. 5. Comparison of transfer times for AX.25CM-CL and TCP/AX.25 with reference to theoretical model at 9600bps

actually get transmitted. After receiving a packet over-the-air, the next transmission went through as normal. In the second case, the TH-D72 appears to stop receiving. This resolves itself on the next packet transmitted. During our test runs, any time this happened, we discarded the result and added a further test run at the end.

VI. Conclusion

The AX.25CM-CL now performs well, in both the throughput testing described here and in longer term usage testing for single-hop, point-to-point links, on dedicated radio channels. The previously reported over-acknowledgement within the Convergence Layer protocol can now be managed by defining the acknowledgement window appropriately based on the expected link characteristics. Static links with good link budget margin can be tailored for with larger ACK windows. Intermittent links can be optimised for shorter ACK windows to mitigate against unwarranted duplication within reactively fragmented bundles.

The measurements taken provide an illustration that, on low-bandwidth links, the overhead of TCP/IP can be significant, and careful thought should be given to whether its use is really required. In this case, with a window of 1, a 20 byte (minimum) overhead on a 255 byte frame is over 7% overhead. When comparing TCP/IP over AX.25 with the AX.25CM-CL for DTN2, we are

comparing a protocol stack that is understood to perform badly when carried on challenged links with one that is explicitly designed for such environments. The PILC working group reported on the use of TCP Performance Enhancing Proxies (PEP) [13] to mitigate against TCPs issues in this area. By choosing to deploy the Bundling Protocol over AX.25 instead of the TCP/IP stack we avoid the undesirable consequences of breaking the end-to-end principle with PEPs by using DTN as an overlay network. However, this still leaves us without any ability to guarantee end-to-end reliability. The Bundle Security Protocol would offer this, but to date the authors have not yet attempted to deploy this protocol in our testing environment.

In future work, we hope to continue development of an AX.25 discovery mechanism for DTN2 which will inter-operate with the APRS [14] network. Also, a new LTP [15], [16] over AX.25 Convergence Layer for DTN2, tailored for use with low earth orbit Amateur Radio satellites will be investigated.

REFERENCES

[1] C. Labovitz, S. Iekel-Johnson, D. McPherson, J. Oberheide, and F. Jahanian, "Internet inter-domain traffic," in *Proceedings of the ACM SIGCOMM 2010 conference on SIGCOMM*, ser. SIGCOMM '10. New York, NY, USA: ACM, 2010, pp. 75–86. [Online]. Available: http://doi.acm.org/10.1145/1851182.1851194

[2] J. Ronan, K. Walsh, and D. Long, "Evaluation of a DTN convergence layer for the AX.25 network protocol," in *Proceedings of the Second International Workshop on Mobile Opportunistic Networking*, ser. MobiOpp '10. New York, NY, USA: ACM, 2010, pp. 72–78. [Online]. Available: http://doi.acm.org/10.1145/1755743.1755757

[3] "Delay Tolerant Networking Research Group - Code," http://www.dtnrg.org/wiki/Code.

[4] K. Scott and S. Burleigh, "Bundle Protocol Specification," RFC 5050 (Experimental), Internet Engineering Task Force, Nov. 2007. [Online]. Available: http://www.ietf.org/rfc/rfc5050.txt

[5] M. Demmer and J. Ott, "Delay Tolerant Networking TCP Convergence Layer Protocol," November 2008, internet draft, draft-irtf-dtnrg-tcp-clayer-02.txt, work in progress.

[6] M. Chepponis and P. Karn, "The KISS TNC: A simple Host-to-TNC communications protocol," in *6th Computer Networking Conference*. 225 Main Street, Newington, CT 06111-1494, USA: ARRL, 1987.

[7] W. A. Beech, D. E. Dielsen, and J. Taylor, "AX.25 Link Access Protocol for Amateur Packet Radio, version 2.2 Revision July 1998," 1998.

[8] ITU-T, "X.25 : Interface between data terminal equipment (DTE) and data circuit-terminating equipment (DCE) for terminals operating in the packet mode and connected to public data networks by dedicated circuit," Oct. 1996. [Online]. Available: http://www.itu.int/rec/T-REC-X.25-199610-I/en

[9] L. Kleinrock and F. Tobagi, "Random access techniques for data transmission over packet-switched radio channels," in *Proceedings of the May 19-22, 1975, national computer conference and exposition*, ser. AFIPS '75. New York, NY, USA: ACM, 1975, pp. 187–201. [Online]. Available: http://doi.acm.org/10.1145/1499949.1499984

[10] ——, "Packet Switching in Radio Channels: Part I–Carrier Sense Multiple-Access Modes and Their Throughput-Delay Characteristics," *Communications, IEEE Transactions on [legacy, pre - 1988]*, vol. 23, no. 12, pp. 1400–1416, 1975. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1092768

[11] J. Ma, "On the Impact of HDLC Zero Insertion and Deletion on Link Utilization and Reliability," *Communications, IEEE Transactions on*, vol. 30, no. 2, pp. 375 – 381, Feb 1982.

[12] "EIA Standard RS-232-C Interface Between Data Terminal Equipment and Data Communication Equipment Employing Serial Data Interchange," August 1969.

[13] J. Border, M. Kojo, J. Griner, G. Montenegro, and Z. Shelby, "Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations," RFC 3135 (Informational), Internet Engineering Task Force, Jun. 2001. [Online]. Available: http://www.ietf.org/rfc/rfc3135.txt

[14] APRS Working Group, *APRS Protocol Reference*, I. Wade, Ed. Tucson Amateur Packet Radio Corporation, August 2000.

[15] S. Burleigh, M. Ramadas, and S. Farrell, "Licklider Transmission Protocol - Motivation," RFC 5325 (Informational), Internet Engineering Task Force, Sep. 2008. [Online]. Available: http://www.ietf.org/rfc/rfc5325.txt

[16] M. Ramadas, S. Burleigh, and S. Farrell, "Licklider Transmission Protocol - Specification," RFC 5326 (Experimental), Internet Engineering Task Force, Sep. 2008. [Online]. Available: http://www.ietf.org/rfc/rfc5326.txt