Evaluation of a DTN Convergence Layer for the AX.25 Network Protocol

John Ronan
Telecommunications Software & Systems Group
Waterford Institute of Technology
Waterford
Ireland
jronan@tssg.org

Kristian Walsh
Telecommunications Software & Systems Group
Waterford Institute of Technology
Waterford
Ireland
kwalsh@tssg.org

Darren Long
Bury St Edmunds
England
darren.long@mac.com

ABSTRACT

The AX.25 Link Access Protocol for Amateur Packet Radio is a data link layer protocol derived from the ITU-T X.25 data link protocol with modifications for use by amateur radio operators. One of the authors has produced a prototype AX.25 connected mode DTN Convergence Layer (CL) based on the existing DTN2 reference implementation. Initial testing of this implementation was undertaken on Linux in order to compare the performance of the implementation with the performance of native AX.25 links. Initial results appear to indicate that the current prototype can be up to 25 percent more efficient than using the Linux TCP/IP over AX.25 implementation in certain circumstances. The experimental results also reveal situations where obvious improvements can still be made to the implementation.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: Data communications

General Terms

Performance, Experimentation

Keywords

Delay Tolerant Network, DTN, AX.25

1. INTRODUCTION

Amateur Radio emergency communications networks have used the AX.25 protocol [3] for many years. At its peak utilisation, during the late 1980s and early 1990s, the worldwide AX.25 Bulletin Board System (BBS) network moved email and other messages over terrestrial and satellite (AO-51, GO-32 and others) links across long distances, on a store-and-forward network.

Since the growth of the Internet, most of this infrastructure is no longer in place, with much of the previously AX.25 links replaced by fixed-line TCP/IP [15] [14] links between systems. Other AX.25 networks have been replaced over the last number of years by much cheaper IEEE 802.11 point-to-point links.

Recently, however, there has been a resurgence of interest in the AX.25 protocol due to the deployment of the Winlink 2000 [20] system, which commonly uses AX.25 as the "last mile" between an internet-connected gateway and a radio network. However, in the event of a failure anywhere in the chain connecting the radio equipment to the core Winlink network, the client has to be manually reconfigured to use an alternate gateway (or another mechanism) to connect to the Winlink core servers. This need for manual configuration precludes the use of Winlink in *ad hoc* radio networks.

In recent years, DARPA and the Internet Research Task Force have developed an architecture [5] and protocol [17] for Delay or Disruption Tolerant Networking (DTN) . DTN uses in-network or node-level storage to provide an overlay network on top of a number of heterogeneous network infrastructures (see figure 1). Node-level storage allows application messages—called *bundles* in the DTN architecture—to be stored at DTN gateways for arbitrary lengths of time while waiting for a onward path to become available. This differs from IP's "store-and-forward" model where the IP packets must either be forwarded immediately or dropped.

Because DTN operates as an overlay network over transport layers, it allows for different transport protocols to be used at different points along the path. This allows the selection of transport parameters suitable for the local environment. In the context of the DTN overlay network, the various transport layers used between DTN routers are termed *Convergence Layers*. This use of Convergence Layers provides a framework for interconnecting heteroge-

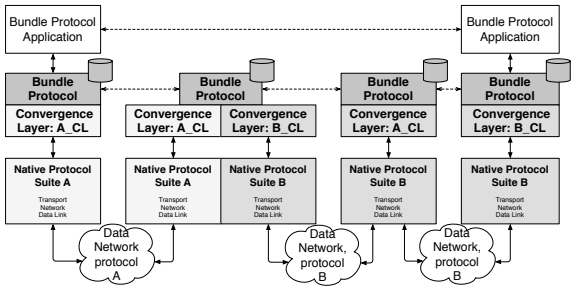

Figure 1: DTN network stacks in a heterogenous network. The DTN Bundle Protocol accesses the underlying transport networks using Convergence Layers which map existing protocol suites to a common set of functions.

MobiOpp '10, February 22-23, 2010, Pisa, Italy.

neous network segments.

In this work, we explore the suitability of the existing AX.25 network infrastructure for use as part of a more general *ad hoc* emergency communications network. A DTN Convergence Layer for the AX.25 protocol was developed to allow AX.25 nodes to partake in DTN transfers. Tools [9] already exist to use a DTN network for the transport of email that present a standard SMTP [13] and POP [4] interface to a user's existing email application.

However, before considering AX.25 networks for use as links in a DTN network, it was necessary to evaluate the performance characteristics of AX.25 itself when used as the transport for DTN traffic. To this end, we examined the throughput of DTN transfers over point-to-point radio links using the AX.25 transmission protocol.

AX.25 links are normally low bandwidth (1200 bits per second is still a common configuration), and so it was essential to determine the overhead incurred by the use of DTN, and how a DTN solution compared to existing transfer mechanisms. For this reason, tests were also performed using not only the DTN protocol, but also native AX.25 and TCP/IP-over-AX.25 in order to be able to perform a meaningful comparison.

2. AMATEUR RADIO COMMUNICATIONS NETWORKS

2.1 Packet BBS Systems

It is worth noting that in 1985, Karn et al [11] describe what work will be required in AX.25 networks to implement distributed protocols to allow for automatic exchange of connectivity information across heterogeneous *ad hoc* connections that have a wide range of performance levels. The following passage is particularly relevant:

> "In addition, given the nature of the amateur service, some links might have some rather unusual characteristics such as asymmetries in station RF performance and part-time availability due to changing propagation conditions or satellite visibility. At the same time, the amount of storage needed to store routing tables in each node must be minimized in order to keep costs down"

In the heyday of the amateur packet network, this was done by building an extensive AX.25 radio network, including satellite gateways on every continent, all using protocols exclusive to Amateur Radio. Long term store-and-forward systems formed an important part of this network's infrastructure, especially for satellite gateways. Today this network still exists, albeit as islands of AX.25 traffic interconnected via TCP/IP tunnels.

2.2 Winlink 2000

One existing approach to integrate the Amateur Radio data network into the wider Internet is the "Winlink 2000" global radio email system [20]. Winlink 2000 is designed to provide access to email to and from radio networks. To do so, the system comprises five mirrored Common Message Servers (CMS) distributed around the globe, whose purpose is to exchange information between the Winlink 2000 network and the broader internet. Each of these CMS nodes serves a number of local Radio Message Servers (RMS), which provide the access points for end-users. Despite support for some additional services (e.g., position broadcast and weather reports), Winlink 2000 is primarily an email service. It provides a mechanism for exchange of SMTP messages between the internet and the AX.25 data network, but this is performed at the application level of the network stack; it is not possible to exchange traffic with arbitrary ports on an internet host.

Winlink 2000 is a statically-configured network: although every RMS is functionally identical, and interchangeable in use, clients must be configured by their operator to speak to a specific one. While disruption tolerance is achieved in the core network by mirroring the five CMS hosts, the "last mile" still relies on a circuit-switched transport model which requires operator intervention in the event of disruption.

3. DTN CONVERGENCE LAYER IMPLEMENTATION

The DTN2 reference implementation [8] is provided as a flexible software framework for experimentation, extension and real-world deployment of Delay Tolerant Networking (DTN) systems. We have taken this framework and used it to produce a Convergence Layer (CL) for the AX.25 networking protocol.

The AX.25 Connected Mode Convergence Layer (AX.25CM-CL) is a convergence layer implementation for AX.25 sockets on the Linux platform.

The AX.25CM-CL transports the DTN "bundles" described by RFC-5050 [17] directly over an AX.25 connection that operates solely as a Layer 2 protocol. In this respect, the AX.25 Convergence Layer (CL) is similar to the existing Bluetooth CL.

Currently, the only major difference between the AX.25CM-CL and the TCP-CL [7]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, as well as providing additional means to detect protocol errors introduced by the implementation.

On occasion, malformed AX.25 packets were sent into the Convergence Layer, which was placed into an invalid state as a result. At its most benign, this invalid state caused the convergence layer link to be dropped (and subsequently re-established, if viable); in more extreme cases, the entire protocol daemon (dtnd) would crash. After adding the CRC32 checksum to each transmitted segment, the AX.25CM-CL now detects malformed messages and terminates the connection, assuming that the connection will be re-established and transfer will be resumed, if connectivity permits.

3.1 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.

4. 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 [3], a data link layer protocol derived from the ITU-T X.25 data link protocol [10] with modifications for use by Amateur Radio operators.

4.1 Experimental settings

Table 1 lists the significant parameters of the radio link used in these experiments.

Table 1: Characteristics of experimental transfer

parameter	value
link speed, bit/s	1200
T_{slot}, s	0.020
$T_{txdelay}$, s	0.150
T_{tail}, s	0.020
p	0.250
data length, bytes	7182

4.2 Model Transfer times

AX.25 is most commonly deployed on half-duplex radio links, with link access managed using a p-persist CSMA algorithm [18] [12]. 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 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:

$$T_{frame} = \frac{(bytes \times 8.004) + 160}{(bitrate)}$$

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

The value 8.004 is used to take into account the additional bits stuffed into payload data octets with the value 01111110. Assuming a uniformly distributed payload byte values, such extra bits will occur on average once in every 256 octets.

Each acknowledgement is a single transmission of a frame with zero payload bits. Allowing for transmission setup and release times, T_{ack}, the average time required to send an acknowledgement is:

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

The number of acknowledgements sent depends on $maxframe$, the acknowledgement window size for the link. Also, 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 2(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:

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

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

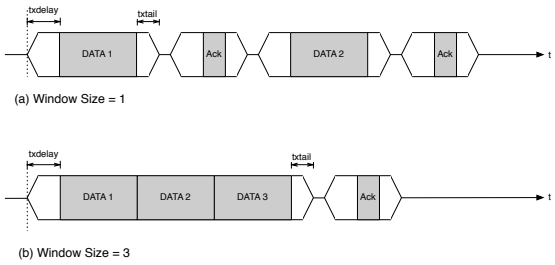

Figure 2: Effect of increased window sizes on link efficiency.

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 2 shows how increasing the window size can reduce the amount of time required to send data.

Using the experimental parameters in Table 1, we calculate 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:

$$T_{frame} = \frac{(255 \times 8.004) + 160}{1200} \qquad = 1.933971s$$

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

$$\begin{aligned} T_{ack} &= T_{txdelay} + \frac{160}{bitrate} + T_{txtail} + p \times T_{slot} \\ &= 0.150 + \frac{160}{1200} + 0.25 \times 0.020 \\ &= 0.308s \end{aligned}$$

Using these values, and the formula for $T_{message}$, previously, the values in Table 2 were obtained.

Table 2: Theoretical minimum transfer times, raw AX.25 transfer

Window size	timings from model (seconds)
1	67.2
2	60.4
3	58.0
4	57.1
5	56.1
6	55.6
7	55.6

* 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 represent peak performance of the link, and 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 RS-232 [16] serial interfaces. As these model figures do not take account of these additional overheads or the time required to process higher-level protocol commands, none

of the experimental results were expected to reach this level of performance.

5. EXPERIMENTAL NETWORK

5.1 BBS and KISS mode

Traditionally, AX.25 Terminal Node Controller (TNC) devices operate interactively. A human user connects, and issues commands to a remote TNC. The remote TNC presents itself as a simple Bulletin Board System, and is thus said to be operating in "BBS Mode". The TNC firmware manages the AX.25 protocol stack, and runs a small set of applications, including a simple message/email server.

Some AX.25 Terminal Node Controller (TNC) devices can also be operated in "KISS" mode, an acronym of "Keep It Simple, Stupid" [6]. This mode allows host software to bypass the higher-level AX.25 implementation built in to the TNC equipment, and use the TNC simply as a means of sending and receiving Layer 2 frames. Using KISS mode requires that all higher functions of the AX.25 protocol be implemented on the host system, but this also allows the host to have greater control over how the AX.25 implementation operates.

The performance of the TNC equipment was recorded in both BBS and KISS modes.

5.2 Network

Figure 3 shows the experimental network used to measure the system performance.

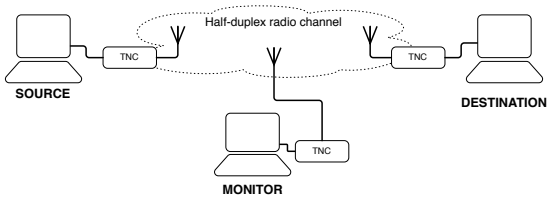

Figure 3: Experimental setup used to measure AX.25 performance. Source and Destination devices were connected on a single RF data channel (i.e., half-duplex)

As the AX.25 TNC and radio transmitters used for Source and Destination nodes were not identical, transfers were performed in both directions in an effort to minimise the effects of using different equipment.

Equipment used for the source node was a Yaesu FT-847 transciever, with a SCS-PTCIIex radio modem. For the destination node a Yaesu FT-1500M transciever and Kantronics KPC3+ radio modem were used. The monitor used a Kenwood TH-D7 with integrated radio modem. The antennas were in close proximity (less than 10 metres), thus power levels were kept low at 5 watts or less where possible. All nodes were static throughout all testing.

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.

To obtain a valid set of readings, five 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[1] of ASCII-encoded text data. For BBS mode tests, this data file was pre-loaded into the source TNC's built in Bulletin Board System (BBS) server.

For testing of the linux AX.25 implementation, we used a combination of *ax25d* (part of the Linux AX25 [1] subsystem) to respond to the AX.25 connection request, and *uronode* [19] to deal with the connection itself. The *axmail* [2] program was used to access a local SMTP mailbox to which the test file had been pre-loaded.

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 according to Table 3 on the Linux host before each transfer was begun. 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 3: TCP/IP test settings

Window Size	MTU (Bytes)	TCP Window (Bytes)
1	255	255
2	255	510
3	255	765
4	255	1020
5	255	1275
6	255	1530
7	255	1785

For the DTN test, The *dtncp* utility was used to send the test file.

Obviously the application-layer protocols used by the *ftp*, *axmail* and *dtncp* tools all 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.

6. RESULTS

Tables 4 and 5 list the results obtained for transfers between the two TNC devices. The recorded values are within $\pm 2\%$ (on the worst case) of the mean with a confidence interval of 99%.

7. DISCUSSION

7.1 AX.25 Acknowledgement timeouts

The "Both KISS" entries in Tables 4 and 5 contain some anomalous timings, marked with an asterisk. Following investigation of the logs, it was discovered that these are due to the message transfer ending with a window containing only one or two frames. In these cases, the receiver does not send an acknowledgement immediately, but instead waits to see if any more frames arrive that would fill the window. As no such frames follow, the receiver times out, and acknowledges what it has received.

[1]This was the size of a ASCII file containing Two-Line Element (TLE) sets for various amateur radio satellites. These TLEs describe the orbit of an earth satellite. The file happened to be on the source machine, and seemed a reasonable starting point

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

Table 4: PTCIIex reading from KPC3+
average transfer times in seconds for window sizes from 1 to 7

Win. size	BBS	KPC3+ KISS	Both KISS	TCP/ IP	DTN	*model*
1	119.2	75.0	75.6	146.0	84.6	67.2
2	87.2	65.0	67.0	104	75.8	60.4
3	86.2	60.6	61.6	99.8	73.8	58.0
4	86.2	59.6	* 63.0	98.4	† 72.0	57.1
5	86.2	57.6	58.4	97.2	† —	56.1
6	86.2	57.0	* 60.2	97.6	† —	55.6
7	86.2	57.2	* 61.0	95.8	† —	55.6

* receiver timeout on last frame group ; †window sizes above 3 are not honoured, see section 7.3.

Table 5: KPC3+ reading from PTCIIex
average transfer times in seconds for window sizes from 1 to 7

Win. size	BBS	PTCIIex KISS	Both KISS	TCP/ IP	DTN	*model*
1	76.6	76.2	76.2	173.2	85.8	67.2
2	65.0	65.0	67.0	104.2	75.6	60.4
3	61.4	62.0	62.0	98.8	74.8	58.0
4	59.6	59.8	* 63.2	97.2	† 71.25	57.1
5	58.2	58.6	59.6	96	† —	56.1
6	57.8	58.0	* 60.8	95.8	† —	55.6
7	57.6	57.6	* 61.6	95.2	† —	55.6

* receiver timeout on last frame group ; †window sizes above 3 are not honoured, see section 7.3.

Table 6: Comparison
transfer times all in seconds

TCP/IP	DTN	best AX.25	*model*
146.0	84.6	76.6	67.2
104	75.8	65	60.4
99.8	73.8	61.4	58.0
98.4	†72.0	59.6	57.1
97.2	†-	58.2	56.1
97.6	†-	57.8	55.6
95.8	†-	57.6	55.6

†window sizes of 4 or more are not honoured, see section 7.3.

When both TNCs were operating in "BBS mode", this final delay was not observed. The timings for the "BBS mode" operation were used as the "Best AX.25" measurements in Table 6 and Figure 5.

7.2 Firmware bottleneck in KPC3+

The "BBS" column in Table 4 shows for window sizes above 3 frames, the TNC's own firmware was the bottleneck to data transfer. Using KISS mode on this device shows a marked improvement. After testing with difference versions of this TNC (including newer firmware), we concluded that the firmware implementation of the "BBS" has this limitation "hard-coded".

7.3 Problems with large window sizes in DTN

Tables 4 and 5 are missing readings for window sizes greater than 4 (AX.25 $maxframe$ parameter). This is because of a bug which was discovered during testing. 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. As it was not possible to obtain five consecutive measurements, the result for $maxframe$ of 4 in Tables 4 and 5 is actually an average of four readings.

Once manifested, this behaviour persisted across all subsequent runs of the DTN test, which would suggest that the problem caused the internal state of the Linux kernel AX.25 implementation itself to become corrupted.

7.4 Over-Acknowledgement

Currently, the AX25CM-CL produces a flurry of (TCP-CL Protocol) segment ACK messages (one for each segment/frame) and sends these as distinct frames. (Figure 4) A more conservative approach to ACK generation would be prudent, by aggregating more than one ACK into a frame and/or adopting a selective ACK mechanism. The first approach (aggregating multiple ACKs into a single frame) should in theory already happen, but does not in practice – further investigation is required.

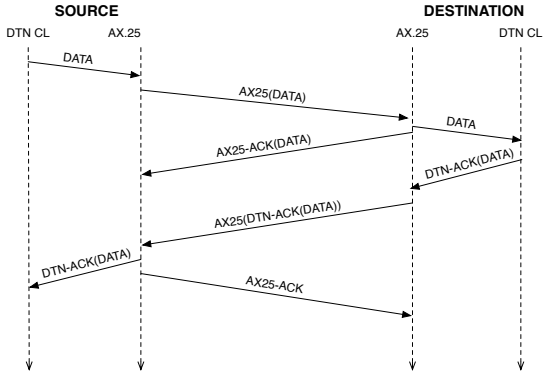

Figure 4: Multiple acknowledgements produced by the DTN CL

Deferring the first AX.25 acknowledgement in such a manner that it is sent in the frame containing the DTN (or TCP) acknowledgement message is one strategy which could be used to combat this problem.

7.5 Overhead of DTN

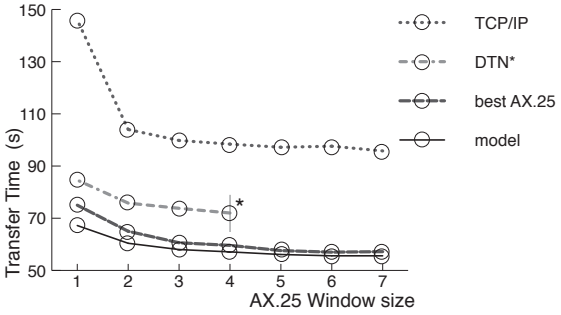

Figure 5: Comparison of transfer times for AX.25, AX.25 DTN CL and TCP/AX.25 with reference to theoretical model. * Reliable DTN results were not available for window sizes of 4 or more, see section 7.3

Overall, the performance of the DTN layer was acceptable, and disproved the authors' "gut instinct" which suggested a much greater overhead from using DTN bundling. While a full set of measurements could not be obtained, the CL implementation shows considerable promise, and comfortably out-performs TCP/IP.

8. CONCLUSIONS

The prototype DTN CL performs well, considering its early stage of development[3]. However, bugs remain in the implementation that could be addressed to further improve performance. Finding the cause of the failure to honor larger window sizes would be a priority task. An other factor that adversely affects performance is the duplication of DTN and AX.25 acknowledgement messages. This resulted in considerable bandwidth being consumed by "acknowledgement-of-acknowledgement" packets. A more efficient mechanism for stimulating reactive fragmentation should be adopted in the AX.25 Connected Mode Convergence Layer. Once these issues are resolved, we could move forward by evaluating the performance in more challenging RF environments and different file sizes.

The analysis of the performance TCP/IP over AX.25 provides a simple illustration of why so few operators use this combination in practice: TCP's assumption of a full-duplex transfer link results in traffic patterns which markedly impair the performance of the half-duplex radio link.

As expected, the highest-performance transfer was found to be native AX.25 point-to-point traffic. However, certain single message transfers in KISS-mode using a larger window size were subject to a delay at end of transfer, as the receiving process would wait for a time-out before passing its received data up the stack.

The measurements taken provide an illustration that, even without considering robustness in the face of disruption, the ubiquitous TCP/IP protocol may not *always* be the best choice.

9. ACKNOWLEDGMENTS

This work was partly funded by the European Commission via the 7th Framework Programme Integrated Project EFIPSANS (grant no. 215549).

[3]We are in the process of attempting to merge our modifications back into the reference implementation. See http://www.dtnrg.org/wiki/AX25ConnectedModeConvergenceLayer for more information.

10. REFERENCES

[1] LinuxAX25, http://www.linux-ax25.org/wiki/linuxax25. Accessed on 2009-01-02.

[2] axMail, ftp://ftp.uroweb.net/pub/ax25/. Accessed on 2009-01-02.

[3] 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. Accessed on 2009-01-02.

[4] M. Butler, J. Postel, D. Chase, J. Goldberger, and J. Reynolds. Post Office Protocol: Version 2. RFC 937 (Historic), Feb. 1985.

[5] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, and H. Weiss. Delay-Tolerant Networking Architecture. RFC 4838 (Informational), Apr. 2007.

[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, 1987. ARRL.

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

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

[9] T. Hyyryläinen, T. Kärkkäinen, C. Luo, V. Jaspertas, J. Karvo, and J. Ott. Opportunistic email distribution and access in challenged heterogeneous environments. In *CHANTS '07: Proceedings of the second ACM workshop on Challenged networks*, pages 97–100, New York, NY, USA, 2007. ACM.

[10] 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.

[11] P. Karn, H. Price, and R. Diersing. Packet radio in the amateur service. *Selected Areas in Communications, IEEE Journal on*, 3(3):431–439, May 1985.

[12] L. Kleinrock and F. Tobagi. Packet switching in radio channels: Part i–carrier sense multiple-access modes and their throughput-delay characteristics. *Communications, IEEE Transactions on [legacy, pre - 1988]*, 23(12):1400–1416, 1975.

[13] J. Klensin. Simple Mail Transfer Protocol. RFC 5321 (Draft Standard), Oct. 2008.

[14] J. Postel. Internet Protocol. RFC 791 (Standard), Sept. 1981. Updated by RFC 1349.

[15] J. Postel. Transmission Control Protocol. RFC 793 (Standard), Sept. 1981. Updated by RFCs 1122, 3168.

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

[17] K. Scott and S. Burleigh. Bundle Protocol Specification. RFC 5050 (Experimental), Nov. 2007.

[18] F. A. Tobagi. *Random access techniques for data transmission over packet switched radio networks.* PhD thesis, 1974.

[19] Uronode, ftp://ftp.uroweb.net/pub/ax25/. Accessed on 2009-01-02.

[20] Winlink 2000 global radio email system, http://www.winlink.org. Accessed on 2009-09-22.