IPv6 Maintenance Working Group   K. Lynn, Ed.
Internet-Draft   Consultant
Intended status: Standards Track   J. Martocci
Expires: September 13, 2012   Johnson Controls
                                                        C. Neilson
                                                     Delta Controls
                                                      S. Donaldson
                                                         Honeywell
                                                    March 12, 2012

                 Transmission of IPv6 over MS/TP Networks
                     draft-ietf-6man-6lobac-01

Abstract

   MS/TP (Master-Slave/Token-Passing) is a contention-free access method
   for the RS-485 physical layer that is used extensively in building
   automation networks. This document describes the frame format for
   transmission of IPv6 packets and the method of forming link-local and
   statelessly autoconfigured IPv6 addresses on MS/TP networks.

Status of this Memo

   This Internet-Draft is submitted in full conformance with the
   provisions of BCP 78 and BCP 79.

   Internet-Drafts are working documents of the Internet Engineering
   Task Force (IETF). Note that other groups may also distribute
   working documents as Internet-Drafts. The list of current Internet-
   Drafts is at http://datatracker.ietf.org/drafts/current/.

   Internet-Drafts are draft documents valid for a maximum of six months
   and may be updated, replaced, or obsoleted by other documents at any
   time. It is inappropriate to use Internet-Drafts as reference
   material or to cite them other than as "work in progress."

   This Internet-Draft will expire on September 13, 2012.

Table of Contents

1. Introduction

MS/TP (Master-Slave/Token-Passing) is a contention-free access method
for the RS-485 [TIA-485-A] physical layer that is used extensively in
building automation networks. This document describes the frame
format for transmission of IPv6 [RFC2460] packets and the method of
forming link-local and statelessly autoconfigured IPv6 addresses on
MS/TP networks. The general approach is to adapt elements of the
6LoWPAN [RFC4944] specification to constrained wired networks.

An MS/TP device is typically based on a low-cost microcontroller with
limited processing power and memory. Together with low data rates
and a small address space, these constraints are similar to those
faced in 6LoWPAN networks and suggest some elements of that solution
might be applied. MS/TP differs significantly from 6LoWPAN in at
least three respects: a) MS/TP devices typically have a continuous
source of power, b) all MS/TP devices on a segment can communicate
directly so there are no hidden node or mesh routing issues, and c)
proposed changes to MS/TP will support payloads of up to 1501 octets,
eliminating the need for link-layer fragmentation and reassembly.

The following sections provide a brief overview of MS/TP, then
describe how to form IPv6 addresses and encapsulate IPv6 packets in
MS/TP frames. This document also specifies a header compression
mechanism, based on [RFC6282], that is recommended in order to make
IPv6 practical on low speed MS/TP networks.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",
"SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this
document are to be interpreted as described in [RFC2119].

1.2. Abbreviations Used

ASHRAE: American Society of Heating, Refrigerating, and Air-
         Conditioning Engineers (http://www.ashrae.org)

BACnet: An ISO/ANSI/ASHRAE Standard Data Communication Protocol
         for Building Automation and Control Networks

CRC: Cyclic Redundancy Check

MAC: Medium Access Control

MSDU: MAC Service Data Unit (MAC client data)

UART: Universal Asynchronous Transmitter/Receiver

1.3. MS/TP Overview

   This section provides a brief overview of MS/TP, which is specified
   in Clause 9 of ANSI/ASHRAE 135-2010 [BACnet] and included herein by
   reference. [BACnet] also covers physical layer deployment options.

   MS/TP is designed to enable multidrop networks over shielded twisted
   pair wiring. It can support segments up to 1200 meters in length or
   data rates up to 115,200 baud (at this highest data rate the segment
   length is limited to 1000 meters). An MS/TP link requires only a
   UART, a 5ms resolution timer, and a [TIA-485-A] transceiver with a
   driver that can be disabled. These features combine to make MS/TP a
   cost-effective field bus for the most numerous and least expensive
   devices in a building automation network.

   The differential signaling used by [TIA-485-A] requires a contention-
   free MAC. MS/TP uses a token to control access to a multidrop bus.
   A master node may initiate the transmission of a data frame when it
   holds the token. After sending at most a configured maximum number
   of data frames, a master node passes the token to the next master
   node (as determined by node address). Slave nodes transmit only when
   polled and are not considered part of this specification.

   MS/TP frames have the following format*:

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |      0x55     |      0xFF     |  Frame Type* |      DA       |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |      SA       |    Length (MS octet first)   |   Header CRC  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   /                                                              /
   / Data and Extended Data CRC fields*                           /
   / encoded using Consistent Overhead Byte Stuffing [COBS]       /
   /                                                              /
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   | optional 0xFF |
   +-+-+-+-+-+-+-+-+
```

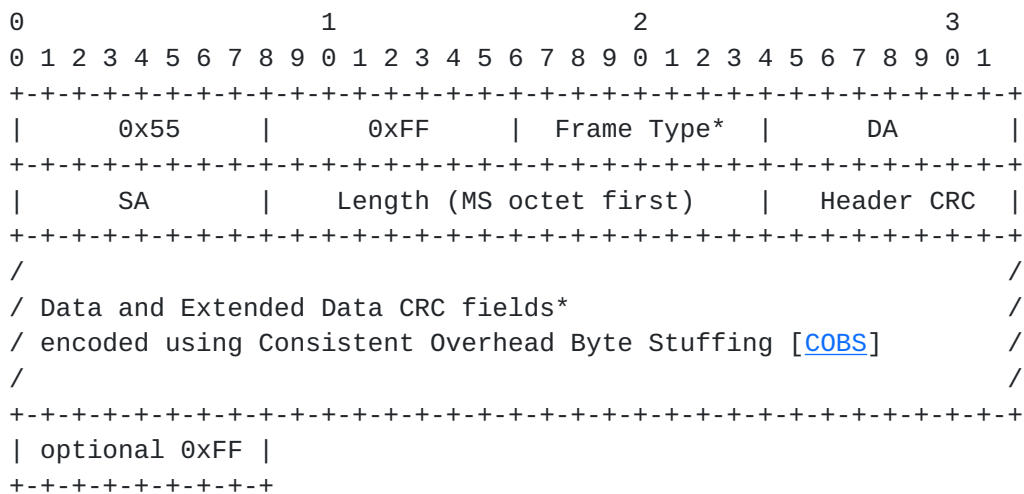   Figure 1: MS/TP Extended Frame Format

   *Note: A BACnet proposal [Addendum_an], now in public review, assigns
    a new Frame Type for IPv6 Encapsulation, extends the maximum length
    of the Data field to 1501 octets, and specifies a 32-bit Extended
    Data CRC [CRC32K] for these frames. The Data and Extended Data CRC
    fields are [COBS] encoded and present only if Length is non-zero.

The MS/TP frame fields have the following descriptions**:

```
Preamble              two octet preamble: 0x55, 0xFF
Frame Type            one octet
Destination Address   one octet address
Source Address        one octet address
Length                two octets, most significant octet first
Header CRC            one octet
Data                  0 - 1501 octets**
                      (present only if Length is non-zero)
Extended Data CRC     four octets**, least significant octet first
                      (present only if Length is non-zero)
(pad)                 (optional) at most one octet of trailer: 0xFF
```

The Frame Type is used to distinguish between different types of MAC
frames. Currently defined types (in decimal) are:

```
00  Token
01  Poll For Master
02  Reply To Poll For Master
    ...
10  IPv6 over MS/TP Encapsulation**
```

Frame Types 11 through 127 are reserved for assignment by ASHRAE.
All master nodes MUST understand Token, Poll For Master, and Reply to
Poll For Master frames. See Section 2 for additional details.

The Destination and Source Addresses are each one octet in length.
See Section 3 for additional details.

A non-zero Length field specifies the length of the [COBS] encoded
Data and Extended Data CRC fields in octets, minus two. (Note: This
trick is required for co-existence with legacy MS/TP devices.) See
Section 4 and Appendices for additional details.

The Header CRC field covers the Frame Type, Destination Address,
Source Address, and Length fields. The Header CRC generation and
check procedures are specified in [BACnet].

**The Data and Extended Data CRC fields are conditional on the Frame
Type and the Length and will always be present in frames specified by
this document. These fields are concatenated and then encoded before
transmission using Consistent Overhead Byte Stuffing [COBS] to remove
preamble sequences from the fields. The Extended Data CRC and COBS
encoding procedures are specified in the BACnet [Addendum_an] change
proposal and briefly summarized in Appendices A and B below.

1.4. Goals and Non-goals

   The primary goal of this specification is to enable IPv6 directly to
   wired end devices in building automation and control networks, while
   leveraging existing standards to the greatest extent possible. A
   secondary goal is to co-exist with legacy MS/TP implementations.
   Only the minimum changes necessary to support IPv6 over MS/TP are
   proposed in BACnet [Addendum_an] (see note in Section 1.3).

   Non-goals include making changes to the MS/TP frame header format,
   control frames, Master Node state machine, or addressing modes.
   Also, while the techniques described here may be applicable to other
   data links, no attempt is made to define a general design pattern.

2. MS/TP Mode for IPv6

   The BACnet [Addendum_an] change proposal allocates a new MS/TP Frame
   Type from the ASHRAE reserved range to indicate IPv6 Encapsulation.
   The new Frame Type for IPv6 Encapsulation is 10 (0x0A).

   All MS/TP master nodes (including those that support IPv6) must
   understand Token, Poll For Master, and Reply to Poll For Master
   control frames and support the Master Node state machine as specified
   in [BACnet]. MS/TP master nodes that support IPv6 must also support
   the Receive Frame state machine as specified in [BACnet] as extended
   by [Addendum_an].

3. Addressing Modes

   MS/TP node (link-layer) addresses are one octet in length. The
   method of assigning node addresses is outside the scope of this
   document. However, each MS/TP node on the link MUST have a unique
   address or a misconfiguration condition exists.

   [BACnet] specifies that addresses 0 through 127 are valid for master
   nodes. The method specified in Section 6 for creating the Interface
   Identifier (IID) ensures that an IID of all zeros can never result.

   A Destination Address of 255 (0xFF) denotes a link-level broadcast
   (all nodes). A Source Address of 255 MUST NOT be used. MS/TP does
   not support multicast, therefore all IPv6 multicast packets MUST be
   sent as link-level broadcasts and filtered at the IPv6 layer.

   This document assumes that each MS/TP link maps onto a unique IPv6
   subnet prefix. Hosts learn IPv6 prefixes via router advertisements
   according to [RFC4861].

4. Maximum Transmission Unit (MTU)

   The BACnet [Addendum_an] change proposal specifies that the MSDU be
   increased to 1501 octets and covered by a 32-bit CRC. This is
   sufficient to convey an MTU of at least 1280 octets as required by
   IPv6 without the need for link-layer fragmentation and reassembly.

   However, the relatively low data rates of MS/TP still make a
   compelling case for header compression. An adaptation layer to
   indicate compressed or uncompressed IPv6 headers is specified below
   in Section 5 and the compression scheme is specified in Section 10.

5. LoBAC Adaptation Layer

   The encapsulation formats defined in this section (subsequently
   referred to as the "LoBAC" encapsulation) comprise the payload (MSDU)
   of an MS/TP frame. The LoBAC payload (i.e., an IPv6 packet) follows
   an encapsulation header stack. LoBAC is a subset of the LoWPAN
   encapsulation defined in [RFC4944], therefore the use of "LOWPAN" in
   literals below is intentional. The primary differences between LoBAC
   and LoWPAN are: a) exclusion of the Fragmentation, Mesh, and
   Broadcast headers, and b) use of LOWPAN_IPHC [RFC6282] in place of
   LOWPAN_HC1 header compression (which is deprecated by [RFC6282]).

   All LoBAC encapsulated datagrams transmitted over MS/TP are prefixed
   by an encapsulation header stack. Each header in the stack consists
   of a header type followed by zero or more header fields. Whereas in
   an IPv6 header the stack would contain, in the following order,
   addressing, hop-by-hop options, routing, fragmentation, destination
   options, and finally payload [RFC2460]; in a LoBAC encapsulation the
   analogous sequence is (optional) header compression and payload. The
   header stacks that are valid in a LoBAC network are shown below.

       A LoBAC encapsulated IPv6 datagram:

         +---------------+-------------+---------+
         | IPv6 Dispatch | IPv6 Header | Payload |
         +---------------+-------------+---------+

       A LoBAC encapsulated LOWPAN_IPHC compressed IPv6 datagram:

         +---------------+-------------+---------+
         | IPHC Dispatch | IPHC Header | Payload |
         +---------------+-------------+---------+

   All protocol datagrams (i.e., IPv6 or compressed IPv6 headers) SHALL
   be preceded by one of the valid LoBAC encapsulation headers. This

permits uniform software treatment of datagrams without regard to
their mode of transmission.

The definition of LoBAC headers consists of the dispatch value, the
definition of the header fields that follow, and their ordering
constraints relative to all other headers. Although the header stack
structure provides a mechanism to address future demands on the LoBAC
(LoWPAN) adaptation layer, it is not intended to provided general
purpose extensibility. This format document specifies a small set of
header types using the header stack for clarity, compactness, and
orthogonality.

5.1. Dispatch Value and Header

The LoBAC Dispatch value begins with a "0" bit followed by a "1" bit.
The Dispatch value and header are shown here:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0|1| Dispatch  |  Type-specific header
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Dispatch   6-bit selector. Identifies the type of header
                      immediately following the Dispatch value.

Type-specific header   A header determined by the Dispatch value.

Figure 2: Dispatch Value and Header

The Dispatch value may be treated as an unstructured namespace. Only
a few symbols are required to represent current LoBAC functionality.
Although some additional savings could be achieved by encoding
additional functionality into the dispatch value, these measures
would tend to constrain the ability to address future alternatives.

```
   Pattern        Header Type
 +------------+-----------------------------------------------------+
 | 00  xxxxxx | NALP        - Not a LoWPAN (LoBAC) frame            |
 | 01  000000 | ESC         - Additional Dispatch octet follows     |
 | 01  000001 | IPv6        - Uncompressed IPv6 Addresses           |
 |   ...      | reserved    - Defined or reserved by [RFC4944]      |
 | 01  1xxxxx | LOWPAN_IPHC - LOWPAN_IPHC compressed IPv6 [RFC6282] |
 | 1x  xxxxxx | reserved    - Defined or reserved by [RFC4944]      |
 +------------+-----------------------------------------------------+
```

Figure 3: Dispatch Value Bit Patterns

NALP: Specifies that the following bits are not a part of the LoBAC
   encapsulation, and any LoBAC node that encounters a Dispatch
   value of 00xxxxxx shall discard the packet. Non-LoBAC protocols
   that wish to coexist with LoBAC nodes should include an octet
   matching this pattern immediately following the MS/TP header.

ESC: Specifies that the following header is a single 8-bit field for
   the Dispatch value. It allows support for Dispatch values larger
   than 127 (see [RFC6282] section 5).

IPv6: Specifies that the following header is an uncompressed IPv6
   header [RFC2460].

LOWPAN_IPHC: A value of 011xxxxx specifies a LOWPAN_IPHC compression
   header (see Section 10.)

Reserved: A LoBAC node that encounters a Dispatch value in the range
   01000010 through 01011111 or 1xxxxxxx SHALL discard the packet.

6. Stateless Address Autoconfiguration

This section defines how to obtain an IPv6 Interface Identifier. The
general procedure is described in Appendix A of [RFC4291], "Creating
Modified EUI-64 Format Interface Identifiers".

The Interface Identifier may be based on an [EUI-64] identifier
assigned to the device (but this is not typical for MS/TP). In this
case, the Interface Identifier is formed from the EUI-64 by inverting
the "u" (universal/local) bit according to [RFC4291]. This will
result in a globally unique Interface Identifier.

If the device does not have an EUI-64, then the Interface Identifier
MUST be formed by concatenating its 8-bit MS/TP node address to the
seven octets 0x00, 0x00, 0x00, 0xFF, 0xFE, 0x00, 0x00. For example,
an MS/TP node address of hexadecimal value 0x4F results in the
following Interface Identifier:

```
|0              1|1              3|3              4|4              6|
|0              5|6              1|2              7|8              3|
+----------------+----------------+----------------+----------------+
|0000000000000000|0000000011111111|1111111000000000|0000000001001111|
+----------------+----------------+----------------+----------------+
```

Note that this results in the universal/local bit set to "0" to
indicate local scope.

An IPv6 address prefix used for stateless autoconfiguration [RFC4862]

of an MS/TP interface MUST have a length of 64 bits.

7. IPv6 Link Local Address

   The IPv6 link-local address [RFC4291] for an MS/TP interface is
   formed by appending the Interface Identifier, as defined above, to
   the prefix FE80::/64.

```
    10 bits            54 bits                   64 bits
  +----------+-----------------------+----------------------------+
  |1111111010|       (zeros)         |     Interface Identifier   |
  +----------+-----------------------+----------------------------+
```


8. Unicast Address Mapping

   The address resolution procedure for mapping IPv6 non-multicast
   addresses into MS/TP link-layer addresses follows the general
   description in Section 7.2 of [RFC4861], unless otherwise specified.

   The Source/Target Link-layer Address option has the following form
   when the addresses are 8-bit MS/TP node (link-layer) addresses.

```
                    0                   1
                    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
                   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                   |     Type      |   Length=1    |
                   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                   |     0x00      | MS/TP Address |
                   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                   |                               |
                   +-         Padding            -+
                   |          (all zeros)          |
                   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```


   Option fields:

   Type:

      1: for Source Link-layer address.

      2: for Target Link-layer address.

   Length: This is the length of this option (including the type and
      length fields) in units of 8 octets. The value of this field is 1
      for 8-bit MS/TP node addresses.

MS/TP Address: The 8-bit address in canonical bit order [RFC2469].
   This is the unicast address the interface currently responds to.

9. Multicast Address Mapping

All IPv6 multicast packets MUST be sent to MS/TP Destination Address
255 (broadcast) and filtered at the IPv6 layer. When represented as
a 16-bit address in a compressed header (see Section 10), it MUST be
formed by padding on the left with a zero:

```
                0                   1
                0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
               |     0x00      |      0xFF      |
               +-+-+-+-+-+-+-+-+-+---------------+
```


10. Header Compression

LoBAC uses LOWPAN_IPHC IPv6 compression, which is specified in
[RFC6282] and included herein by reference. This section will simply
identify substitutions that should be made when interpreting the text
of [RFC6282].

In general the following substitutions should be made:

* Replace "6LoWPAN" with "MS/TP network"

* Replace "IEEE 802.15.4 address" with "MS/TP address"

When a 16-bit address is called for (i.e., an IEEE 802.15.4 "short
address") it MUST be formed by padding the MS/TP address to the left
with a zero:

```
                0                   1
                0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
               |     0x00      | MS/TP address |
               +-+-+-+-+-+-+-+-+-+---------------+
```


11. IANA Considerations

This document uses values previously reserved by [RFC4944] and
[RFC6282] and makes no further requests of IANA.

Note to RFC Editor: this section may be removed upon publication.

12. Security Considerations

   The method of deriving Interface Identifiers from MAC addresses is
   intended to preserve global uniqueness when possible. However, there
   is no protection from duplication through accident or forgery.

13. Acknowledgments

   We are grateful to the authors of [RFC4944] and members of the IETF
   6LoWPAN working group; this document borrows extensively from their
   work.

14. References

14.1. Normative References

   [BACnet]   American Society of Heating, Refrigerating, and Air-
               Conditioning Engineers, "BACnet, A Data Communication
               Protocol for Building Automation and Control Networks
               (ANSI Approved)", ANSI/ASHRAE 135-2010, April 2011.

   [RFC2119]   Bradner, S., "Key words for use in RFCs to Indicate
               Requirement Levels", BCP 14, RFC 2119, March 1997.

   [RFC2460]   Deering, S. and R. Hinden, "Internet Protocol, Version 6
               (IPv6) Specification", RFC 2460, December 1998.

   [RFC4291]   Hinden, R. and S. Deering, "IP Version 6 Addressing
               Architecture", RFC 4291, February 2006.

   [RFC4861]   Narten, T., Nordmark, E., Simpson, W., and H. Soliman,
               "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861,
               September 2007.

   [RFC4862]   Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless
               Address Autoconfiguration", RFC 4862, September 2007.

   [RFC4944]   Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler,
               "Transmission of IPv6 Packets over IEEE 802.15.4
               Networks", RFC 4944, September 2007.

   [RFC6282]   Hui, J. and P. Thubert, "Compression Format for IPv6
               Datagrams over IEEE 802.15.4-Based Networks", RFC 6282,
               September 2011.

14.2. Informative References

[Addendum_an]
          ASHRAE, "BSR/ASHRAE Addendum an to ANSI/ASHRAE Standard
          135-2010, BACnet - A Data Communication Protocol for
          Building Automation and Control Networks (Advisory Public
          Review Draft)", September 2011,
          <https://osr.ashrae.org/default.aspx>.

[COBS]   Cheshire, S. and M. Baker, "Consistent Overhead Byte
          Stuffing", IEEE/ACM TRANSACTIONS ON NETWORKING, VOL.7,
          NO.2 , April 1999,
          <http://www.stuartcheshire.org/papers/COBSforToN.pdf>.

[CRC32K] Koopman, P., "32-Bit Cyclic Redundancy Codes for Internet
          Applications", IEEE/IFIP International Conference on
          Dependable Systems and Networks (DSN 2002) , June 2002, <h
          ttp://www.ece.cmu.edu/~koopman/networks/dsn02/
          dsn02_koopman.pdf>.

[EUI-64] IEEE, "Guidelines for 64-bit Global Identifier (EUI-64)
          Registration Authority", March 1997, <http://
          standards.ieee.org/regauth/oui/tutorials/EUI64.html>.

[RFC2469] Narten, T. and C. Burton, "A Caution On The Canonical
          Ordering Of Link-Layer Addresses", RFC 2469,
          December 1998.

[TIA-485-A]
          Telecommunications Industry Association, "TIA-485-A,
          Electrical Characteristics of Generators and Receivers for
          Use in Balanced Digital Multipoint Systems (ANSI/TIA/
          EIA-485-A-98) (R2003)", March 2003.

Appendix A. **Extended Data CRC [CRC32K]**

   This Appendix is informative and not part of the standard.

   Extending the payload of MS/TP to 1501 octets requires upgrading the
   Data CRC from 16 bits to 32 bits. Koopman has published several
   papers on choosing the right CRC polynomial for the application. In
   [CRC32K], he surveyed the entire 32-bit polynomial space and
   identified some that exceed the 802.3 polynomial in performance.

   The BACnet MS/TP change proposal [Addendum an] specifies the CRC32K
   (Koopman) polynomial. An example C implementation is shown below.
   The specified use of the function is that 'crcValue' is initialized
   to all ones before the function is first called and, upon running the
   function over all octets in the payload, the ones complement of
   'crcValue' be sent in LSB-first order. Upon reception, the data
   field and modified 'crcValue' are passed again through the function.
   If these fields were properly received, the result of the function
   will be 'CRC32K_RESIDUE'.

```
#include <stdint.h>

/* See ASHRAE 135-2010 Addendum an, section G.3.2 */
#define CRC32K_INITIAL_VALUE (0xFFFFFFFF)
#define CRC32K_RESIDUE (0x0843323B)
/* CRC-32K polynomial, 1 + x**1 + ... + x**30 (+ x**32) */
#define CRC32K_POLY (0xEB31D82E)

/*
 * Accumulate 'dataValue' into the CRC in 'crcValue'.
 * Return updated CRC.
 *
 * Note: crcValue must be set to CRC32K_INITIAL_VALUE
 * before initial call.
 */
uint32_t
CalcExtendedDataCRC(uint8_t dataValue, uint32_t crcValue)
{
  uint8_t  data, b;
  uint32_t crc;

  data = dataValue;
  crc  = crcValue;

  for (b = 0; b < 8; b++) {
    if ((data & 1) ^ (crc & 1)) {
      crc >>= 1;
      crc ^= CRC32K_POLY;
    } else {
      crc >>= 1;
    }
    data >>= 1;
  }
  return crc;
}
```

Appendix B. Consistent Overhead Byte Stuffing [COBS]

   This Appendix is informative and not part of the standard.

   The BACnet change proposal [Addendum_an] corrects a long-standing
   issue with the MS/TP specification; namely that preamble sequences
   were not escaped whenever they appeared in the Data or Data CRC
   fields. In some cases, this could result in dropped frames due to
   mis-alignment. The solution is encode the Data and Extended Data CRC
   fields before transmission using Consistent Overhead Byte Stuffing
   [COBS] and decode these fields upon reception.

   COBS is a run-length encoding method that effectively removes '0x00'
   octets from its input. The worst-case overhead is bounded at approx.
   one octet in 254, or less than 0.5%, as described in [COBS]. An
   aribtrary octet value may be removed by XOR'ing the COBS output with
   the specified value. In the case of MS/TP, the '0x55' preamble octet
   is specified for removal.

   Encoding proceeds logically in three passes. First, the Extended
   Data CRC is calculated over the data, modified for transmission as
   described in Appendix A, and appended to the data. The combined
   fields are then passed through the COBS encoder. The resulting
   output is then XOR'd with the MS/TP preamble octet '0x55'. The
   length of the encoded fields, minus two octets for compatibility with
   existing MS/TP devices, is placed in the MS/TP header Length field
   before transmission.

   An example C implementation that combines these passes is shown
   below. The decode() function is the inverse of the encode()
   function.

```
#include <stdint.h>

#define MSTP_PREAMBLE_55 (0x55)

/*
 * Encodes 'length' octets of data at the location pointed to by 'input'
 * and writes the output to the location pointed to by 'output'.
 * Returns the number of octets written to 'output'.
 */
size_t
frame_encode (uint8_t *output, const uint8_t *input, size_t length)
{
  size_t code_index = 0;
  size_t read_index = 0;
  size_t write_index = 1;
  uint8_t code = 1;
  uint8_t data;
  int i;

  uint32_t crc32K = CRC32K_INITIAL_VALUE;

  while (read_index < length) {
    data = input[read_index++];
    crc32K = CalcExtendedDataCRC(data, crc32K);
    /*
     * In the common case, simply copy input to output and
     * increment the number of octets copied.
     */
    if (data != 0) {
      output[write_index++] = (data ^ MSTP_PREAMBLE_55);
      code++;
      if (code != 0xFF)
        continue;
    }
    /*
     * In the special case of encountering a zero in the input or
     * having copied the maximum number (254) of non-zero octets,
     * store the count and re-initialize encoder variables.
     */
    output[code_index] = (code ^ MSTP_PREAMBLE_55);
    code_index = write_index++;
    code = 1;
  }
  /*
   * Run the one's complement of the CRC value through the encoder,
   * LSB first.
   */
  crc32K = ~crc32K;
```

```
  for (i = 0; i < 4; i++) {
    data = ((uint8_t *) &crc32K)[i];

    if (data != 0) {
      output[write_index++] = (data ^ MSTP_PREAMBLE_55);
      code++;
      if (code != 0xFF)
        continue;
    }
    /*
     * In the special case of encountering a zero in the input or
     * having copied the maximum number (254) of non-zero octets,
     * store the count and re-initialize encoder variables.
     */
    output[code_index] = (code ^ MSTP_PREAMBLE_55);
    code_index = write_index++;
    last_code = code;
    code = 1;
  }
  /* Append a "phantom zero" to the output stream. */
  output[code_index] = (code ^ MSTP_PREAMBLE_55);
  /*
   * Return the combined value of the Data and Extended CRC fields.
   * Subtract two before use as the MS/TP frame Length field.
   */
  return write_index;
}
```

```
   /*
    * Takes COBS encoded Data and Extended Data CRC fields as 'input'.
    * The 'length' contains the actual length of these fields in
    * octets (that is, the header Length field plus two).
    * Decodes the Data and Extended Data CRC fields into 'output'.
    * Returns length of decoded Data in octets.
    * Note: Safe to call with 'output' <= 'input'.
    */
   size_t
   frame_decode (uint8_t *output, const uint8_t *input, size_t length)
   {
     uint16_t read_index  = 0;
     uint16_t write_index = 0;
     uint8_t  code, data;
     int i;

     crc32 = CRC32K_INITIAL_VALUE;
     while (read_index < length) {
       code = (input[read_index] ^ MSTP_PREAMBLE_55);
       /*
        * Sanity check the encoding to prevent the for() loop below
        * from overrunning the output buffer.
        */
       if ((read_index + code) > length)
         return 0;

       read_index++;

       for (i = 1; i < code; i++) {
         data = (input[read_index++] ^ MSTP_PREAMBLE_55);
         crc32 = CalcExtendedDataCRC(data, crc32);
         output[write_index++] = data;
       }
       if ((code < 0xFF) && (read_index < length)) {
         data = '\0';
         crc32 = CalcExtendedDataCRC(data, crc32);
         output[write_index++] = data;
       }
     }
     if (crc32K == CRC32K_RESIDUE)
       return write_index - sizeof(uint32_t);
     else
       return 0;
   }
```

Authors' Addresses

    Kerry Lynn (editor)
    Consultant

    Phone: +1 978 460 4253
    Email: kerlyn@ieee.org

    Jerry Martocci
    Johnson Controls, Inc.
    507 E. Michigan St
    Milwaukee, WI 53202
    USA

    Phone: +1 414 524 4010
    Email: jerald.p.martocci@jci.com

    Carl Neilson
    Delta Controls, Inc.
    17850 56th Ave
    Surrey, BC V3S 1C7
    Canada

    Phone: +1 604 575 5913
    Email: cneilson@deltacontrols.com

    Stuart Donaldson
    Honeywell Automation & Control Solutions
    6670 185th Ave NE
    Redmond, WA 98052
    USA

    Email: stuart.donaldson@honeywell.com