6Lo Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 5, 2015

K. Lynn, Ed.
Consultant
J. Martocci
Johnson Controls
C. Neilson
Delta Controls
S. Donaldson
Honeywell
July 4, 2014

Transmission of IPv6 over MS/TP Networks
draft-ietf-6lo-6lobac-00

Abstract

Master-Slave/Token-Passing (MS/TP) is a contention-free access method for the RS-485 physical layer, which is used extensively in building automation networks. This specification defines 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 January 5, 2015.

Copyright Notice

Table of Contents

1. Introduction

Master-Slave/Token-Passing (MS/TP) is a contention-free access method for the RS-485 [TIA-485-A] physical layer, which is used extensively in building automation networks. This specification defines 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 leveraged. 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) recent 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

MTU: Maximum Transmission Unit

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 ANSI/ASHRAE 135-2012 (BACnet) Clause 9 [Clause9] and included herein by reference. BACnet [Clause9] also covers physical layer deployment options.

MS/TP is designed to enable multidrop networks over shielded twisted pair wiring. It can support a data rate of 115,200 baud on segments up to 1000 meters in length, or segments up to 1200 meters in length at lower baud rates. An MS/TP link requires only a UART, an RS-485 [TIA-485-A] transceiver with a driver that can be disabled, and a 5ms resolution timer. These features 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 SHALL NOT be considered part of this specification.

MS/TP COBS-encoded* frames have the following format:

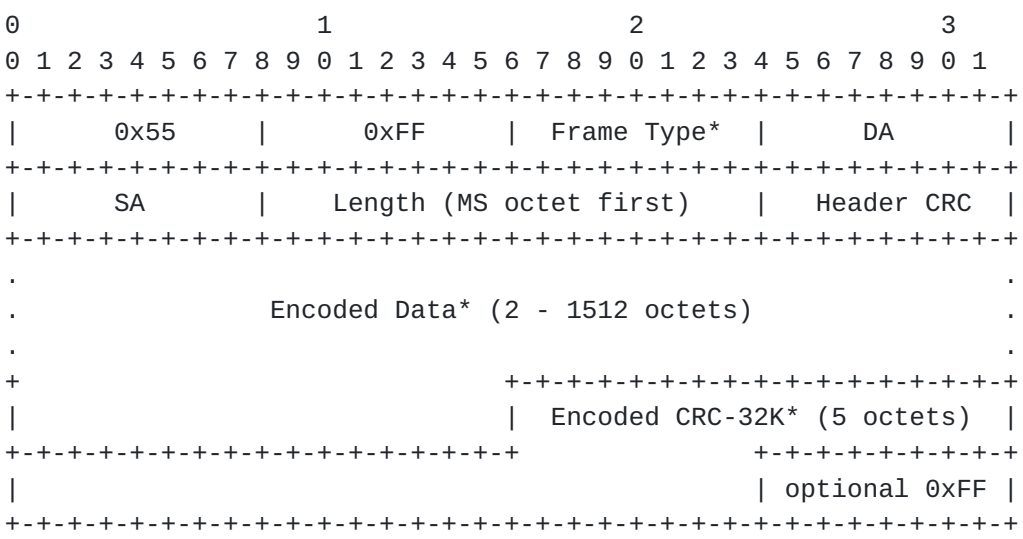


Figure 1: MS/TP COBS-Encoded Frame Format

*Note: BACnet Addendum 135-2012an [Addendum an] defines a range of Frame Type values to designate frames that contain data and data CRC fields encoded using Consistent Overhead Byte Stuffing [COBS] (see Appendix B). The purpose of COBS encoding is to eliminate preamble sequences from the Encoded Data and Encoded CRC-32K fields. The maximum length of an MSDU as defined by this specification is 1501 octets (before encoding). The Encoded Data is covered by a 32-bit CRC [CRC32K] (see Appendix C), which is then itself COBS encoded.

MS/TP COBS-encoded 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
Encoded Data	2 - 1512 octets (see Appendix B)
Encoded CRC-32K	five octets (see Appendix C)
(pad)	(optional) at most one octet of trailer: 0xFF

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

```
   0  Token
   1  Poll For Master
   2  Reply To Poll For Master
      ...
  34  IPv6 over MS/TP (LoBAC) Encapsulation
```

ASHRAE reserves undefined MS/TP Frame Type values 8 through 31 and 34 through 127, inclusive. Frame Types 32 through 127 designate COBS-encoded frames and MUST convey Encoded Data and Encoded CRC-32K fields. All master nodes MUST understand Token, Poll For Master, and Reply to Poll For Master control frames. See Section 2 for additional details.

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

For COBS-encoded frames, the Length field specifies the combined length of the [COBS] Encoded Data and Encoded CRC-32K fields in octets, minus two. (This adjustment is required for backward compatibility 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 [Clause9].

1.4. Goals and Non-goals

The primary goal of this specification is to enable IPv6 directly on wired end devices in building automation and control networks by leveraging existing standards to the greatest extent possible. A secondary goal is to co-exist with legacy MS/TP implementations. Only the minimal changes necessary to support IPv6 over MS/TP are specified 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

ASHRAE must assign a new MS/TP Frame Type to indicate IPv6 over MS/TP Encapsulation from the range reserved for designating COBS-encoded frames. The Frame Type requested for IPv6 over MS/TP Encapsulation is 34 (0x22).

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 [Clause9]. MS/TP master nodes that support IPv6 must also support the Receive Frame state machine as specified in [Clause9] and extended by BACnet [Addendum_an].

3. Addressing Modes

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

BACnet [Clause9] 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 specification assumes that a unique IPv6 subnet prefix is assigned to each MS/TP segment. Hosts learn IPv6 prefixes via router advertisements according to [RFC4861].

4. Maximum Transmission Unit (MTU)

BACnet [Addendum_an] supports MPDUs up to 2032 octets in length. This specification defines an MPDU length of at least 1281 octets and at most 1501 octets. This is sufficient to convey the minimum MTU required by IPv6 [RFC2460] without the need for link-layer fragmentation and reassembly.

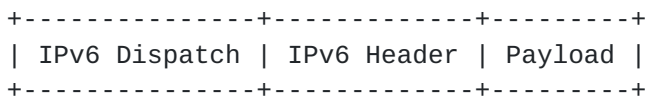
However, the relatively low data rates of MS/TP still make a compelling case for header compression. An adaptation layer to indicate compressed or uncompressed IPv6 headers is specified in Section 5 and the compression scheme is specified in Section 10.

5. LoBAC Adaptation Layer

The encapsulation formats defined in this section (subsequently referred to as the "LoBAC" encapsulation) comprise the MSDU (payload) 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) omission 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:



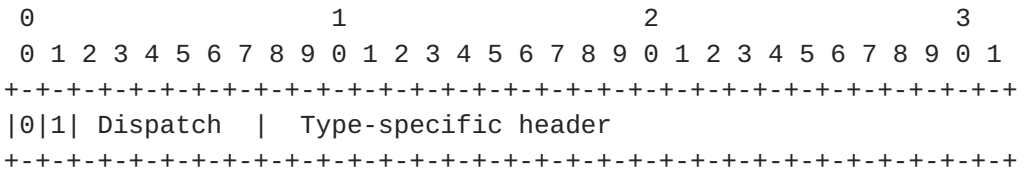
A LoBAC encapsulated LOWPAN_IPHC compressed IPv6 datagram:

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:



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", as updated by [RFC7136].

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 EUI-64 to IID transformation defined in the IPv6 addressing architecture [RFC4291] MUST be used. This will result in a globally unique Interface Identifier.

If the device does not have an EUI-64, then the Interface Identifier SHOULD 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:

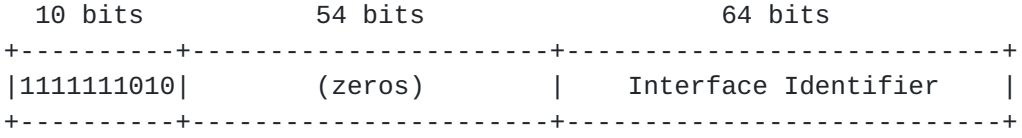
```
|0              1|1              3|3              4|4              6|
|0              5|6              1|2              7|8              3|
+----------------+----------------+----------------+----------------+
|0000000000000000|0000000011111111|1111111000000000|0000000001001111|
+----------------+----------------+----------------+----------------+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

This is the RECOMMENDED method of forming an IID, as it supports the most efficient header compression provided by the LOWPAN_IPHC [RFC6282] scheme specified in Section 10.

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

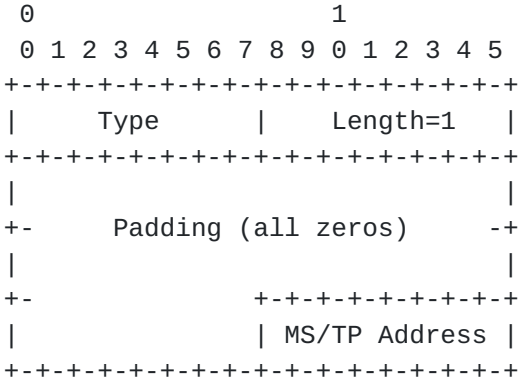
The IPv6 link-local address [RFC4291] for an MS/TP interface is formed by appending the Interface Identifier, as defined above, to the prefix FE80::/64.



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 link-layer (node) addresses.



Option fields:

Type:

1: for Source Link-layer address.

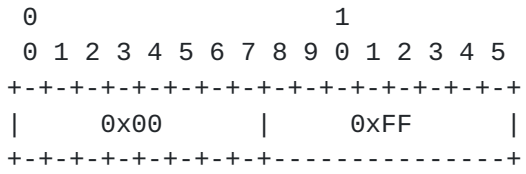
2: for Target Link-layer address.

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:



10. Header Compression

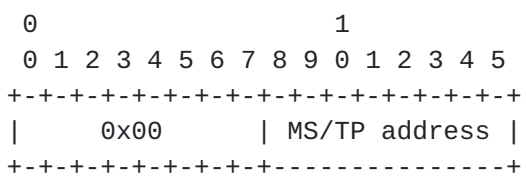
LoBAC uses LOWPAN_IPHC IPv6 compression, which is specified in [RFC6282] and included herein by reference. This section will simply identify substitutions that should be made when interpreting the text of [RFC6282].

In general the following substitutions should be made:

- Replace instances of "6LoWPAN" with "MS/TP network"

- Replace instances of "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:



If LOWPAN_IPHC compression [RFC6282] is used with context, the border router(s) directly attached to the MS/TP segment MUST disseminate the 6LoWPAN Context Option (6CO) as specified in [RFC6775].

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 liberally from their work.

14. References

14.1. Normative References

[Addendum_an] ASHRAE, "Proposed Addendum an to ANSI/ASHRAE Standard 135-2012, BACnet - A Data Communication Protocol for Building Automation and Control Networks (Second Public Review)", March 2014, <http://www.bacnet.org/Addenda/Add-135-2012an-PPR2-draft-rc4_chair_approved.pdf>.

[Clause9] American Society of Heating, Refrigerating, and Air-Conditioning Engineers, "BACnet - A Data Communication Protocol for Building Automation and Control Networks", ANSI/ASHRAE 135-2012 (Clause 9), March 2013.

[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.

[RFC6775] Shelby, Z., Chakrabarti, S., Nordmark, E., and C. Bormann, "Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)", RFC 6775, November 2012.

[RFC7136] Carpenter, B. and S. Jiang, "Significance of IPv6 Interface Identifiers", RFC 7136, February 2014.

14.2. Informative References

[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, <http://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>.

[IEEE.802.3] "Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements - Part 3: Carrier Sense Multiple Access with Collision Detection (CMSA/CD) Access Method and Physical Layer Specifications", IEEE Std 802.3-2008, December 2008, <http://standards.ieee.org/getieee802/802.3.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. Abstract MAC Interface

This Appendix is informative and not part of the standard.

BACnet [Clause9] defines support for MAC-layer clients through its SendFrame and ReceivedDataNoReply procedures. However, it does not define a protocol independent abstract interface for the data link. This is provided below as an aid to implementation.

A.1. MA-DATA.request

A.1.1. Function

This primitive defines the transfer of data from a MAC client entity to a single peer entity or multiple peer entities in the case of a broadcast address.

A.1.2. Semantics of the Service Primitive

The semantics of the primitive are as follows:

```
MA-DATA.request (
                 destination_address,
                 source_address,
                 data,
                 priority,
                 type
                )
```

The 'destination_address' parameter may specify either an individual or a broadcast MAC entity address. It must contain sufficient information to create the Destination Address field (see Section 10) that is prepended to the frame by the local MAC sublayer entity. The 'source_address' parameter, if present, must specify an individual MAC address. If the source_address parameter is omitted, the local MAC sublayer entity will insert a value associated with that entity.

The 'data' parameter specifies the MAC service data unit (MSDU) to be transferred by the MAC sublayer entity. There is sufficient information associated with the MSDU for the MAC sublayer entity to determine the length of the data unit.

The 'priority' parameter specifies the priority desired for the data unit transfer. The priority parameter is ignored by MS/TP.

The 'type' parameter specifies the value of the MS/TP Frame Type field that is prepended to the frame by the local MAC sublayer entity.

A.1.3. When Generated

This primitive is generated by the MAC client entity whenever data shall be transferred to a peer entity or entities. This can be in response to a request from higher protocol layers or from data generated internally to the MAC client, such as a Token frame.

A.1.4. Effect on Receipt

Receipt of this primitive will cause the MAC entity to insert all MAC specific fields, including Destination Address, Source Address, Frame Type, and any fields that are unique to the particular media access method, and pass the properly formed frame to the lower protocol layers for transfer to the peer MAC sublayer entity or entities.

A.2. MA-DATA.indication

A.2.1. Function

This primitive defines the transfer of data from the MAC sublayer entity to the MAC client entity or entities in the case of a broadcast address.

A.2.2. Semantics of the Service Primitive

The semantics of the primitive are as follows:

```
  MA-DATA.indication (
                      destination_address,
                      source_address,
                      data,
                      priority,
                      type
                     )
```

The 'destination_address' parameter may be either an individual or a broadcast address as specified by the Destination Address field of the incoming frame. The 'source_address' parameter is an individual address as specified by the Source Address field of the incoming frame.

The 'data' parameter specifies the MAC service data unit (MSDU) as received by the local MAC entity. There is sufficient information associated with the MSDU for the MAC sublayer client to determine the length of the data unit.

The 'priority' parameter specifies the priority desired for the data unit transfer. The priority parameter is ignored by MS/TP.

The 'type' parameter is the value of the MS/TP Frame Type field of the incoming frame.

A.2.3. When Generated

The MA_DATA.indication is passed from the MAC sublayer entity to the MAC client entity or entites to indicate the arrival of a frame to the local MAC sublayer entity that is destined for the MAC client. Such frames are reported only if they are validly formed, received without error, and their destination address designates the local MAC entity. Frames destined for the MAC Control sublayer are not passed to the MAC client.

A.2.4. Effect on Receipt

The effect of receipt of this primitive by the MAC client is unspecified.

Appendix B. Consistent Overhead Byte Stuffing [COBS]

This Appendix is informative and not part of the standard.

BACnet [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 rare cases, this resulted in dropped frames due to loss of frame synchronization. The solution is to encode the Data and 32-bit 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 nominally removes '0x00' octets from its input. Any selected octet value may be removed by XOR'ing that value with each octet of the COBS output. BACnet [Addendum_an] specifies the preamble octet '0x55' for removal.

The minimum overhead of COBS is one ectet per encoded field. The worst-case overhead is bounded to one octet in 254, or less than 0.5%, as described in [COBS].

Frame encoding proceeds logically in two passes. The Extended Data field is prepared by passing the MSDU through the COBS encoder and XOR'ing the preamble octet '0x055' with each octet of the output. The Extended Data CRC field is then prepared by calculating a CRC-32K over the Extended Data field and formatting it for transmission as described in Appendix C. The combined length of these fields, minus two octets for compatibility with existing MS/TP devices, is placed in the MS/TP header Length field before transmission.

Example COBS encoder and decoder functions are shown below for illustration. Complete examples of use and test vectors are provided in BACnet [Addendum_an].

```
#include <stddef.h>
#include <stdint.h>

#define CRC32K_INITIAL_VALUE (0xFFFFFFFF)
#define MSTP_PREAMBLE_X55 (0x55)

/*
 * Encodes 'length' octets of data located at 'from' and
 * writes one or more COBS code blocks at 'to', removing any
 * 'mask' octets that may present be in the encoded data.
 * Returns the length of the encoded data.
 */

size_t
cobs_encode (uint8_t *to, const uint8_t *from, size_t length,
             uint8_t mask)
{
  size_t code_index = 0;
  size_t read_index = 0;
  size_t write_index = 1;
  uint8_t code = 1;
  uint8_t data, last_code;

  while (read_index < length) {
    data = from[read_index++];
    /*
     * In the case of encountering a non-zero octet in the data,
     * simply copy input to output and increment the code octet.
     */
    if (data != 0) {
      to[write_index++] = data ^ mask;
      code++;
      if (code != 255)
        continue;
    }
    /*
     * In the case of encountering a zero in the data or having
     * copied the maximum number (254) of non-zero octets, store
     * the code octet and reset the encoder state variables.
     */
    last_code = code;
    to[code_index] = code ^ mask;
    code_index = write_index++;
    code = 1;
```

```
    }
    /*
     * If the last chunk contains exactly 254 non-zero octets, then
     * this exception is handled above (and returned length must be
     * adjusted). Otherwise, encode the last chunk normally, as if
     * a "phantom zero" is appended to the data.
     */
    if ((last_code == 255) && (code == 1))
      write_index--;
    else
      to[code_index] = code ^ mask;

    return write_index;
  }
```

```
#include <stddef.h>
#include <stdint.h>

#define CRC32K_INITIAL_VALUE (0xFFFFFFFF)
#define CRC32K_RESIDUE (0x0843323B)
#define MSTP_PREAMBLE_X55 (0x55)

/*
 * Decodes 'length' octets of data located at 'from' and
 * writes the original client data at 'to', restoring any
 * 'mask' octets that may present in the encoded data.
 * Returns the length of the encoded data or zero if error.
 */
size_t
cobs_decode (uint8_t *to, const uint8_t *from, size_t length,
             uint8_t mask)
{
  size_t read_index = 0;
  size_t write_index = 0;
  uint8_t code, last_code;

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

    read_index++;
    while (--code > 0)
      to[write_index++] = from[read_index++] ^ mask;
    /*
     * Restore the implicit zero at the end of each decoded block
     * except when it contains exactly 254 non-zero octets or the
     * end of data has been reached.
     */
    if ((last_code != 255) && (read_index < length))
      to[write_index++] = 0;
  }
  return write_index;
}
```

Appendix C. Encoded CRC-32K [CRC32K]

This Appendix is informative and not part of the standard.

Extending the payload of MS/TP to 1501 octets required upgrading the Data CRC from 16 bits to 32 bits. P.Koopman has authored several papers on evaluating CRC polynomials for network applications. In [CRC32K], he surveyed the entire 32-bit polynomial space and noted some that exceed the [IEEE.802.3] polynomial in performance. BACnet [Addendum_an] specifies the CRC-32K (Koopman) polynomial.

The specified use of the calc_crc32K() function is as follows. Before a frame is transmitted, 'crc_value' is initialized to all ones before the function is called. After passing all octets of the [COBS] Encoded Data through the function, the ones complement of the resulting 'crc_value' is arranged in LSB-first order and is itself [COBS] encoded.

Upon reception of a frame, 'crc_value' is initialized to all ones. The octets of the Encoded Data field are accumulated by the calc_crc32K() function before decoding. The Encoded CRC-32K field is then decoded and the resulting four octets are accumulated by the calc_crc32K() function. If the result is the expected residue value 'CRC32K_RESIDUE', then the frame was received correctly.

An example CRC-32K function in shown below for illustration. Complete examples of use and test vectors are provided in BACnet [Addendum_an].

```
#include <stdint.h>

/* See BACnet Addendum 135-2012an, 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 'data_value' into the CRC in 'crc_value'.
 * Return updated CRC.
 *
 * Note: crcValue must be set to CRC32K_INITIAL_VALUE
 * before initial call.
 */
uint32_t
calc_crc32K (uint8_t data_value, uint32_t crc_value)
{
  uint8_t  data, b;
  uint32_t crc;

  data = data_value;
  crc  = crc_value;

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

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