MCTP 2.0 Overview WIP v0.6

PMCI August 16th 2022

Disclaimer

- The information in this presentation represents a snapshot of work in progress within the DMTF.
- This information is subject to change without notice. The standard specifications remain the normative reference for all information.
- For additional information, see the DMTF website.
- This information is a summary of the information that will appear in the specifications. See the specifications for further details.

Introduction

- MCTP specs were first released in 2009. Since then, the specifications have become widely adopted. Enabling MCTP use on current and future platforms requires addressing some key developments:
  - The number of managed devices in a single platform has increased dramatically
  - Message sizes have increased significantly in some use cases
  - The number of message types that are being transferred over MCTP has grown
  - Security & encryption are now a requirement in most environments
  - The transports for MCTP have increased beyond the original small packet interfaces
- In order to keep MCTP as the management transport protocol of choice in the coming decade, the industry has reached the point where these issues should be addressed. And yet, this needs to be done carefully to leverage the success of MCTP 1.0, which will remain in some environments for years to come.

Overview

- MCTP 2.0 is intended to address some of the deficiencies and limitations of MCTP 1.0
- MCTP 2.0 may be introduced in incremental steps. This document defines the content for each of the new capabilities which are planned to be included in MCTP 2.0

General Requirement

- MCTP 2.0 allows us to start from scratch. However, there are a lot of good things in existing MCTP 1.0 base. The request is to leverage as much as possible the functionality that exists in MCTP 1.0.
  - This whole slide set assumes that a lot of MCTP 1.0-base feature and function carries forward into 2.0 in some shape or form (e.g. discovery, bridging, commands, MTU discovery, etc). It does not call out every feature and function in 1.0

MAIN FEATURES LIST

Reliable and Non-reliable MCTP transport modes definition

Reliable mode

- Transfer mode which assures reliable delivery of application data over MCTP transport using segmentation, re-assembly, acknowledgement, errors-detection and re-transmission

Non-reliable mode

- Transfer mode which allows delivery of application data over MCTP transport using segmentation, re-assembly without guaranteeing application data transfer reliability

Planned Features list

Feature #	Feature name	Reliable mode	Non-Reliable mode
1	Extending EID range	Mandatory	Mandatory
2	Enabling coexistence of MCTP 1.0/2.0	Mandatory	Mandatory
4	Hot-Insertion/removal support	Optional	Optional
5	Backward compliance with existing HW	Mandatory	Mandatory
6a	Increasing concurrent messages count	Mandatory	Mandatory
6b	Increase the number of outstanding packets	Mandatory	Mandatory
9	Forward looking header format	Mandatory	Mandatory
10	Medium and protocol agnostic header	Mandatory	Mandatory
11	Negotiate transmission unit size	Optional	Optional
13	Support for Secured Messages	Optional	Optional
14	Provide support for reliable & efficient large transfers	Mandatory	Not-required
16a	Keep Rate-based Flow control support	Mandatory	Mandatory
17	Reliably detect message corruption	Mandatory	Optional
18	Pause/Resume flow control	Optional	Optional
19	Error/State Reporting	Mandatory	Mandatory

Feature #1 - Extending EID range

- Problem:
  - Scalability beyond 255 EIDs
  - Large MCTP network can easily consume 255 endpoint IDs.

- Requirement:
  - Add support for up to 64K EIDs
  - Increase the endpoint IDs namespace or whatever the equivalent of endpoint ID is for MCTP 2.0
- Comment
  - Current MCTP binding specifications may have to be adapted to accommodate EIDs beyond 255

Feature #2 - Enabling coexistence of MCTP 1.0/2.0

- MCTP 1.0 and MCTP 2.0 shall be allowed to be used on the same physical bus
- MCTP 1.0 and MCTP 2.0 shall be allowed to be used on the same MCTP network
- MCTP 2.0 compliant bridge shall be able to route both MCTP 1.0 and MCTP 2.0 packets without being required to perform translations

Feature #4 - Hot-Insertion/removal support

- Add new method for propagating the hot-add/remove info up in the hierarchy of bus owners up to the topmost bus owner.
  - Information about endpoints added/removed on a local bus is not visible to other devices, except the local bus owner assigning the EID.
  - Having current information about MCTP endpoints is important for devices that manage the MCTP network, such as a BMC.
    The request is to define such a capability.
- Add new method for endpoints to subscribe for async notifications for Hot-Insertion/removal events
  - Need to address security hazards and considerations with such subscription

Feature #5 - Backward compliance with existing HW

- Any change for MCTP packet header must
  - Add method for discovery of MCTP version support by bus owner and endpoint
  - Allow any endpoint to publish its MCTP version support
  - Allow co-existence of MCTP 1.X and MCTP 2.X messages on the same physical bus
    - Mandatory (shall)
      - 2.X Bus owners and management controllers shall be able to send and receive messages using both formats
      - 2.X Bridges shall support both MCTP 2.X and MCTP 1.X
    - Recommended (should)
      - Require MCTP 2.X capable devices to be able to send and receive messages using both formats
      - Designate EIDs such that there will be no conflict in EIDs at system-level
        - Example: assume all MSBs of EIDs of MCTP 1.X to be all 0's
  - Allow MCTP 2.X capable devices to communicate with MCTP 1.X only devices

Feature #6a - Increasing concurrent messages count

- Allow increased usage of MCTP via:
  - Increase the number of tags

Feature #6b - Increase the number of outstanding packets

- Allow increased usage of MCTP via:
  - Increase the number of outstanding packets - more bits for sequence Number

Feature #9 Forward looking header format

- Problem:
  - It is hard to accommodate changes to MCTP header for future request and issues that we have to do a new 2.0 base.
    - Example, we have to do a 2.0 just to increase the endpoint namespace size.
- Requirement:
  - Allow for future changes to MCTP base or header without requiring a 3.0 spec change.
    - Example, decouple the “Hdr version” from the physical binding/requirements

Feature #10 Medium and protocol agnostic header

- Problem:
  - It is hard to transfer an MCTP message over various mediums especially when they are bridged.
    - This is because of the “Hdr version” having language that binds it to the physical transport
  - Also, it is not clear in a mixed environment how various bridges and endpoints of different MCTP protocol can work together.
- Requirement:
  - Make it clear and easy that an MCTP message and its header can go across various mediums and endpoints with various MCTP versions without changing the MCTP message and MCTP packet header contents.
    - Example: the MCTP header should not have any bindings to physical layer.

Feature #11 Negotiate transmission unit size

- Problem:
  - The minimum packet size is too restrictive for present day use. Thus, lots of overhead at the packet level (i.e. many packets need to be sent).
- Requirement:
  - Enable negotiating the transmission unit size (while retaining the default BTU)
  - Negotiating the Transmission Unit Size (shall be >= BTU) and be medium specific across all message types applied for both directions
  - Negotiation of transmission unit size should **only** be part of MCTP base
  - Support negotiating MCTP packet size between two MCTP Endpoints
- Comments
  - How to address bridged connections?
    - Can we make the negotiation End2End?
    - Should it include the bus-owner in the negotiation?
    - Should we support message-type specific MSMTU negotiation?

Feature #13 Support for secured messages

- Provide secured message capability integrated directly into MCTP 2.0 without requirement for separate message type
  - ie MCTP Type 6 no longer would be required
    - MCTP Type-6 is mutex with MCTP native secured messages
- Comment
  - Definition of Message Integrity within MCTP base
  - Support of native secured messages shall be optional
  - MCTP Bridges shall be secured message agnostic
    - MCTP bridging and routing function shall be secured messages agnostic
  - Negotiation for support for secured messages shall be end-to-end

Feature #14 Provide support for reliable & efficient multi-packet message transfers

- Problem
  - If a packet is dropped or damaged in multi-packet MCTP message, MCTP relies on the higher level to retry. This means the entire message needs to be retried. This can reduce reliability in the transport and cause additional overhead that consumes bandwidth.
  - Error criteria shall be within the list of causes to drop a packet
- Requirement:
  - Support reliable message delivery between two MCTP endpoints
    - Allowing a destination receiver to request a retransmission of MCTP packet(s) or segment(s)
    - Allowing a source transmitter to support of retransmission of MCTP packet(s) or segment(s)
    - Handling an error shall be part of the same message
  - Enable negotiating the max message size that can use re-transmission/re-assembly across all message types between two MCTP endpoints

Feature #16a Keep Rate-based Flow control support

- A Negotiated sender-based rate-limiting feature:
  - Rate-based flow control
- Transport-Level only capability
- Decided that flow-control is
  - End-to-end feature

Feature #17 Reliably detect message corruption

- Problem:
  - Need a common method for MCTP to reliably detect message corruption
- Requirement:
  - Define a mechanism in MCTPv2 Base (optional but recommended) to use integrity check for fragmented messages - simple CRC or SPDM could be the options
- Motivation:
  - Packet loss detection – improve incorrect message reassembly detection
    - MCTPv1 does not define any mechanism (other than the seqNo/TAG field, which is very weak) to detect message payload corruption due to packet loss
    - Higher-layer protocols **assume** that MCTP layer delivers uncorrupted messages, for example, the CERTIFICATE message in SPDM spec, which will likely be fragmented into many fragments, does not have any integrity check
  - Efficiency – integrity (signature of the payload) is a common need nowadays
    - side-effect of such a signature check is corruption detection so the proposal is to reuse it for efficiency reasons

Feature #18 Pause/Resume Flow Control

- Define a mechanism to allow a Management Controller to pause/resume traffic transmitted from a Management Endpoint
- Message Type agnostic
- Pause is at the packet level
- Three potential methods to pause:
  - Control Message
  - Bit in header of first packet to auto-pause
  - Physical transport layer specific method when the mux is switched or the device is transmitting
    - SMBus NACK-based pause
- Replay semantics can be used to resume
  - Management Controller specifies the packet to resume from
- More Processing Required
  - Device auto pauses; sends back time estimate; replay updates time estimate
- An optional discoverable feature for Management Endpoints per physical layer.
- All of these are at the MCTP Base Spec scope except SMBus NACK-based which is at the SMBus/I2C Binding Spec scope. Control Messages only need to be enabled for SMBus/I2C initially, but architecture should support expansion to additional physical binding in the future.
- Support for Pause/Resume flow control is not required for MCTP bridges initially.

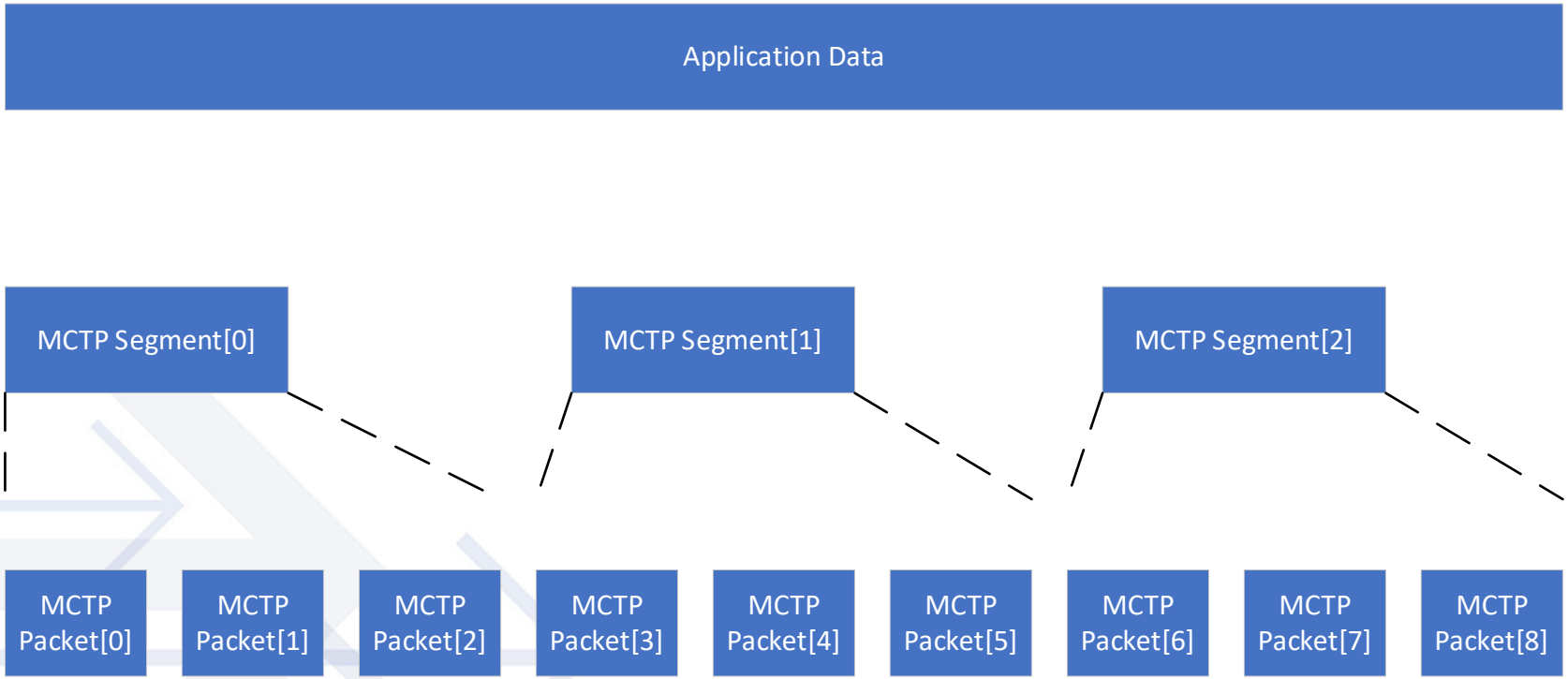
Feature #19 Error/State Reporting

- Define a mechanism to allow a Management Controller to request MCTP error/state information from a Management Endpoint (Mandatory)
- Status bits for dropped packets/messages (Mandatory)
  - Readable and clearable by Management Controller
- State info such as paused state (Mandatory)
- Counters such as Total Received Packets, Total Transmitted Packets, Total Dropped Packets, Total Dropped Packets Due to Message Integrity Check Failure, etc. (Optional)
- Scope: MCTP Base Spec

MCTP 2.0 technical discussion

DETAILED DEFINITIONS

Simplified Segmentation illustration



Application data

Data that is specific to the application and whose definition and format is outside the scope of this specification. Application data usually exist at the application layer, which is, in general, the layer above MCTP. Examples of data that could be application data include messages carried as DMTF MCTP payloads such as PLDM, NC-SI, NVMe-MI;

MCTP Segment

An MCTP segment is a portion of application data transferred over MCTP using one or more consecutive MCTP packets

- *An MCTP segment represents the maximal re-transmittable portion of Application Data at the MCTP layer*
- *Within an application data transfer, all segments shall be equally sized except for the last segment which may be shorter*

MCTP Packet

The unit of data transfer used for MCTP communication on a given physical medium (same as DSP0236 1.x)

- MCTP packets within the MCTP segments shall carry equally sized payload data except for the last packet of each segment which may be shorter

MCTP packet payload

Refers to the portion of the MCTP segment that is carried in a single MCTP packet

- The MCTP payload is sized to be the transmission unit for all packets except for the last packet within a segment which may be shorter

ACK packet

- A control packet sent by an endpoint to acknowledge the reception of a complete segment with no errors.

ACK packet intent

- ACK packets are only used on reliable MCTP transfers (after negotiation)
- Acknowledge reception of a complete segment with no errors
- Once a segment is acknowledged it can be discarded from the sender buffer
- Acknowledgement is done at a segment granularity
- An ACK packet shall be sent within a TBD1 delay following the successful reception of the complete segment
- After transmitting an complete segment, if an ACK is not received by the sender within TBD2 time, it is an implicit NACK
- Transmission of a next segment within a given application data shall only start after receiving an ACK for a previously transmitted segment

NACK packet definition

- A control packet sent by an endpoint that rejects a partial or complete segment due to an error or other condition

NACK packet intent

- NACK shall be sent by a receiver on detection of an error within a received segment
- An error is any one of: missing-packet(s), payload integrity, unexpected payload or transport error and other error as defined in DSP0236 1.x
- A segment size exceeding the negotiated segment size shall be dropped and marked as a bad segment
- Internal processing error on the receiver side (which requires a re-transmission to recover)
- ACK/NACK Receive timeout by the segment transmitter is considered an implied NACK
- If retransmission count has not exceeded, a sender shall attempt retransmission within TBD3 of receiving a NACK packet or implied NACK timeout

ABORT packet definition (details are WIP)

- A control packet sent by an endpoint indicating termination of application data transfer

MCTP 2.0 future discussion

OPEN ITEMS

Future work items

- Reliable transfer negotiation
  - How to negotiate which message types are transferred as reliable and which message-types are transferred as non-reliable
- Pause & Resume flow control
  - How to negotiate support for flow control
  - When and how to send flow control
- Bus owner functions
  - How and if Bus owner shall be tracking added/removed endpoints
- Bridges and other non-endpoint devices
  - How and if these shall be detecting bus disconnections
- Resolving segment size
- MCTP 1.x and 2.0 co-existence
  - How to map/interpret EIDs
- Detailed MCTP packet format and Segment format
- Abort packet details

MCTP 2.0 Header fields (medium agnostic per Req #10)

Field name	Mandatory (Y/N)	Size	Comments
MCTP Header Version	Yes	4b	Same location as 1.x
Reserved	Yes	4b	
SEID	Yes	16b	== 1.x
DEID	Yes	16b	== 1.x
SOM	Yes	1b	
EOM	Yes	1b	
Sequence #	Yes	?	
TO	Yes	1b	*Document* as *Origin/Reply*
MSG TAG	Yes	?	Response shall use the request' #
IC			Re-discuss per use-case & Security
(MSG Type)	Yes	?	
SOR?	TBD		Start of Replay?

Header Fields description

- SOM – Start Of Message. Set to 1 only on the 1st packet of a **(non replayed)** message. When SOM==1 Sequence # shall be set to 0.
- EOM – End Of Message. Set to 1 on the last or only packet of a message. Therefore, on a single-packet message EOM & SOM are both set.
- TO – Set to 1 on every packet by the message originator and set to 0 on every packet by the responder
- Sequence number – An N bit counter which increments on every packet of a message. Shall always be set to 0 on 1st packet of a **(non replayed)** message.
- Message Tag – An identifier of a message. Message originator sets this value with the TO bit set to 1. Message responder shall use in the response the same Message Tag value that was received from the message originator and shall set TO bit to 0.

MCTP Segment Fields

Within a Segment

1. Application data type (formerly message type)
2. Application data ID (needed if multiple streams co-exists)
3. Segment Sequence #
4. Start/End of Application Data
5. Integrity check code flag (formerly IC bit)
6. Integrity check code (when enabled)
7. Encryption (on/off) flag
8. Request/Response flag
9. Encryption header/trailer including authentication tag (when applicable)

MCTP Packet fields

Within a Packet

1. Tag-Owner (or Segment owner?) or should it be replaced with Request/Response flag?
2. Segment Sequence #
3. Tag?
4. Start/End of Segment
5. Packet-type? (differentiating “normal/data” from “ack/control” etc.)
6. Packet Sequence #
7. Re-transmitted flag (and retransmission trial #)
8. Source/destination EID