AdvancedTCA ®

PICMG SFP.1

Revision 1.0

(Internal TDM) Short Form Specification

This short form specification is derived from the PICMG SFP.1 Revision 1.0 Internal TDM specification as approved on March 24, 2005 by the PICMG® Executive Membership. For guidelines on the design of the SFP.1 compliant boards and systems, refer to the full specification–do not use this short form for any design decisions.

SFP.1 – Internal TDM

1. INTRODUCTION

1.1. Purpose

This Short Form version of the SFP.1 specification is intended to provide an overview of the I-TDM protocol. Note that most of the I-TDM detailed low-level requirements have been removed in this Short Form version. Do not attempt to design to this Short Form specification.

1.1.1. Why I-TDM?

New modular architectures and standards (like PICMG 3.x) define communication systems that do not include a TDM backplane or TDM interconnect technology for LAN attached communication modules. TDM traffic has not been eliminated, just the legacy H.1x0 bus in next generation modular systems. Packet back-planes and LANs have replaced legacy physical interconnects, but there still needs to be a standard way of transporting TDM traffic from one module to another.

1.1.2. What is I-TDM?

I-TDM stands for "Internal TDM" protocol. I-TDM is a multiplexed voice over packet protocol that is optimized for Voice LANs and Packet Backplanes (i.e. for connecting telephony equipment within the same chassis, room or building). I-TDM does not aim to be an end user protocol. It typically exists only inside the logical confines of a Voice Processing or Voice Switching System, hence the name "Internal TDM". Note that a Voice Processing or Switching System may physically consist of multiple LAN attached boxes that span a room, building or campus.

1.1.3. I-TDM Features

- Low Latency (1 millisecond or 125 microsecond packet rate)
- Up to 512 TDM channels per I-TDM packet
- Low processing overhead (high density)
- 64-bit alignment (for modern processors)
- Very easy to move TDM channels from one position to another within a packet or between packet flows (grooming)
- Up to 64K channels per destination node for 1ms mode, and 16M channels per destination node in 125us mode.
- Optional transport of Channel Associated Signaling (CAS) bits
- Optional transport of Multi-timeslot connections
- Optional support for other packet rates

1.1.4. I-TDM LAN Usage

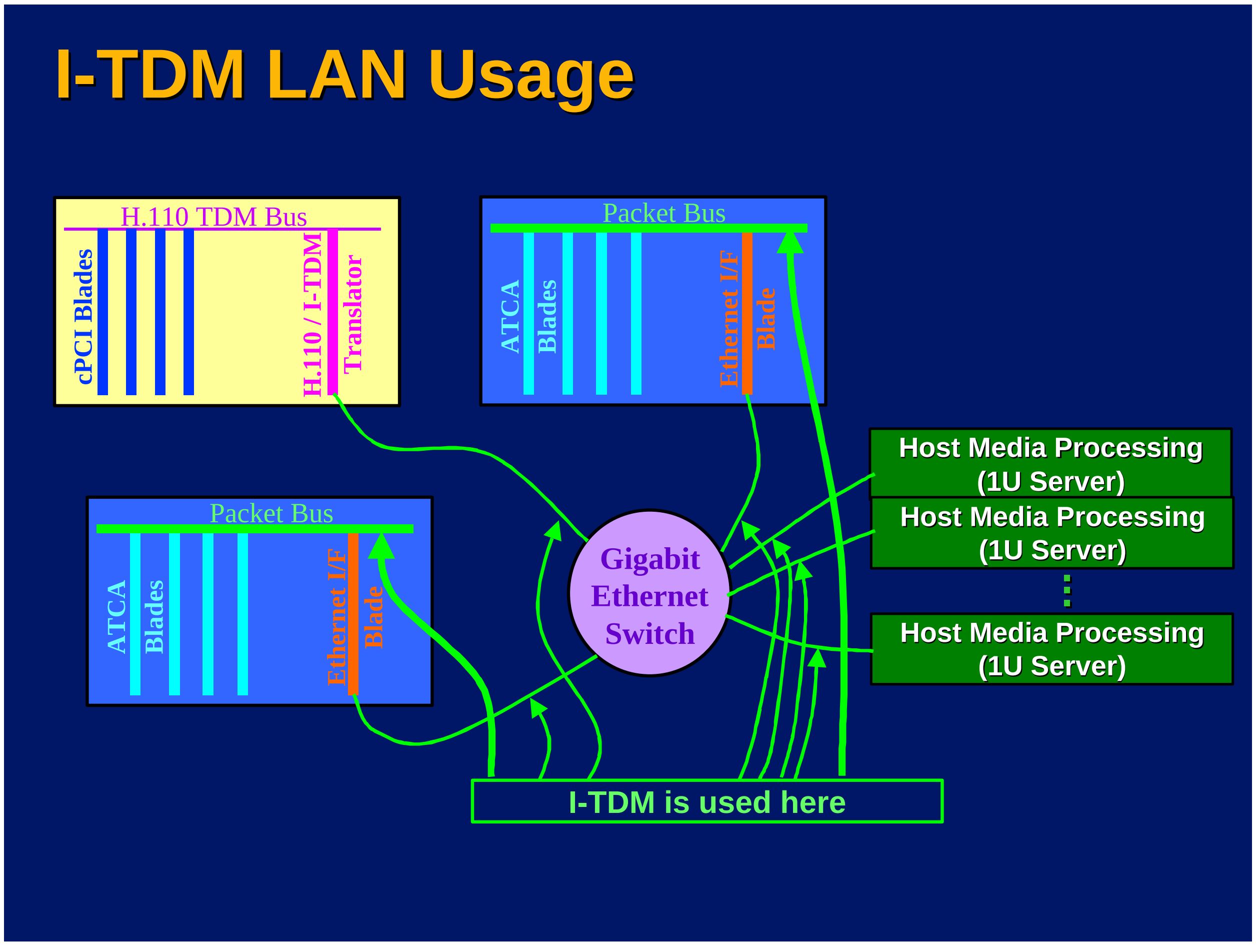


Figure 1: I-TDM LAN Usage

Media Processing such as Speech Recognition, Voice Conferencing, Voice Mail, etc. is being increasingly implemented using off-the-shelf Host Platforms (e.g. 1U PCs with no special add in cards). At low densities, VoIP works well for this. But when the density scales up to hundreds or thousands of voice channels per Host, VoIP is no longer viable. The processing overhead of VoIP is too high. Also, even for some lower density Host Media Processing applications, the latency of VoIP becomes an issue. So I-TDM is the right kind of protocol to connect high-density Host Media Processing platforms into a larger system.

Also, as Voice and Data networks converge, TDM busses (e.g. H.100 or H.110) will migrate to packet bus implementations. However, this migration will not occur all at once. Existing Telephony equipment will have to inter-work at some level with the newer packet bus equipment for some time. I-TDM is perfectly positioned to enable this migration.

1.1.5. I-TDM Packet Backplane Usage

I-TDM is also a perfect protocol for carrying many TDM voice/data calls over the packet bus itself. In fact, this is what really drove the creation of the I-TDM format to begin with.

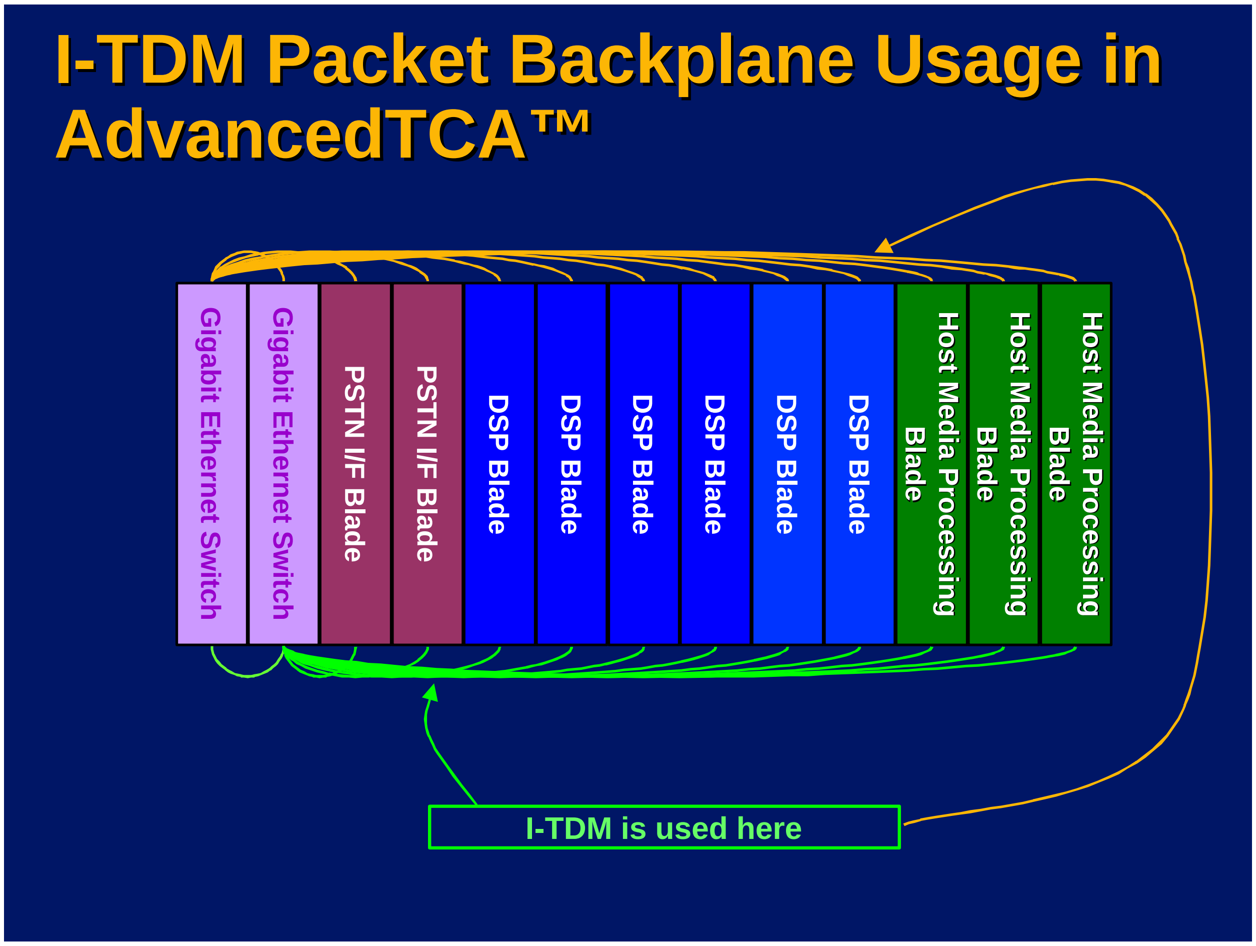


Figure 2: I-TDM Packet Backplane Usage in Advanced TCA

Standardizing I-TDM in the aTCA backplane allows the creation of an ecosystem of boards that can be used together to implement a variety of applications, including:

- Media Gateways
- Media Servers
- PBXs
- PSTN Switches
- Various combinations of the above

1.1.6. I-TDM Logical Placement

Regardless of the physical configuration of the system (e.g. Packet Backplane, LAN, MAN, etc), the logical positioning of the I-TDM protocol is the same.

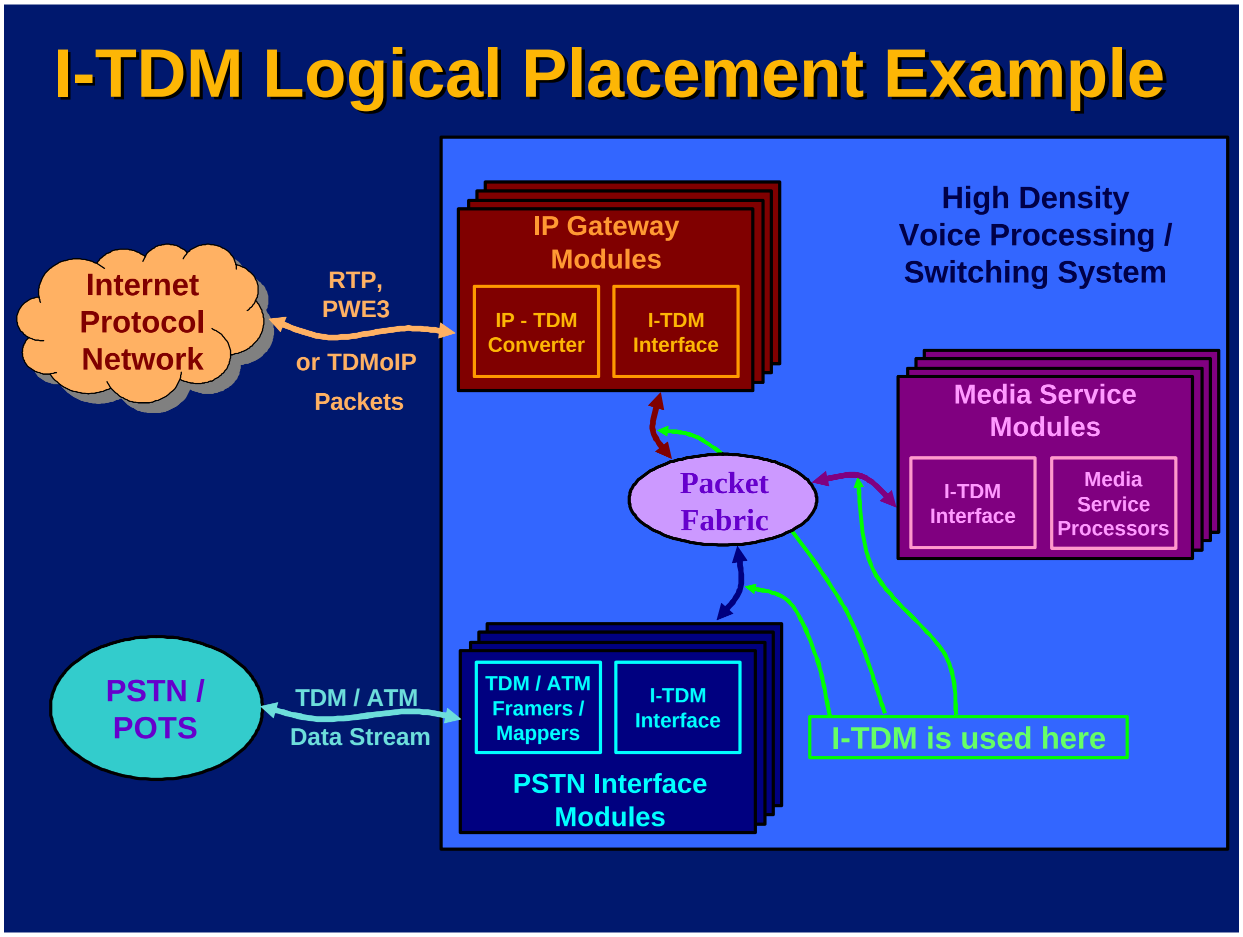


Figure 3: I-TDM Logical Placement Example

I-TDM does not aim to be an end user protocol. It typically exists only within the logical confines of the Voice Processing System, hence the name “Internal TDM” protocol.

Note that various types of systems may be configured using the diagram above.

For example, a VoIP Gateway may be configured using just PSTN Interface Modules and IP Gateway Modules (i.e. no Media Service Modules).

In another example, an IP Media Server may be configured using IP Gateway Modules and Media Service Modules (i.e. no PSTN Interface Modules necessary).

In a third example, a PBX or PSTN Switch may be configured using multiple PSTN Interface Modules connected via I-TDM (i.e. no Media Service Modules or IP Gateway Modules necessary).

In a fourth example, all 3 types of modules above maybe combined to form a Switching Gateway Server (for lack of a better name).

1.2. Scope

1.2.1. Clocking

Although I-TDM applications usually require that endpoints be tied to a common clock, the I-TDM protocol doesn't carry any clocking information. The timing information may be carried:

- by separate clock wires (e.g. 8KHz back plane wires, T1/E1 wires between chassis, etc.), or
- by endpoints synchronizing on the long term packet arrival rate of the I-TDM packets themselves, or
- by some other method.

The choice of methods depends on the jitter/latency requirements of the specific implementation, and is outside the scope of this specification. See Section 2.7 for details.

1.2.2. Line Status

I-TDM does not carry any line status information (e.g. Red Alarm, Yellow Alarm, Loop Status, etc.). This information can be communicated directly to the User Application through the Vendor Specific API.

1.2.3. Dropped Packets

I-TDM does not specify what action the destination node will take when the fabric drops a packet (e.g. insert silence, repeat last samples, etc.). Actions taken under these circumstances are implementation specific.

2. RATIONALE FOR I-TDM

This section overviews the driving application level requirements that led to the creation of the I-TDM protocol. Note that Section 2 is for informational purposes only and does not contain any items for compliance testing.

There are a number of existing TDM over packet protocols (e.g. VoIP, TDMoIP, PWE3, VoMPLS, Multiplexed RTP, etc.). However, none of these existing protocols meet all the requirements of I-TDM. This is because the existing protocols generally have an entirely different application in mind.

For example, VoIP generally aims to make the Internet look like a switching long distance carrier. Multiplexing is not required since VoIP calls typically don't share endpoints. Low latency is generally not an issue since most long distance gateways employ echo cancellers. If both endpoints are IP Phones (e.g. IP PBX), then echo will never occur, so low latency isn't required for these types of applications as well.

The new IETF protocol "PWE3" (Pseudo Wire Emulation Edge to Edge) aims to make the Internet look like a wide area leased line provider (hence the name "Wire Emulation"). TDMoIP has similar applications. Switching calls is not required since leased lines are statically configured. Also, the clock synchronization and jitter characteristics of these applications tend to demand hardware based endpoints, so requirements having to do with Processor based endpoints aren't as relevant for PWE3 or TDMoIP.

By contrast, I-TDM aims to make a Private LAN or Packet Backplane look like a switching TDM bus. Low Latency, High Call Rates, Multiplexing and Processor Based Endpoints are all strong requirements for I-TDM.

The list below summarizes the driving application level requirements that led to the creation of the I-TDM protocol.

1. **Low Latency**
    - For DS0 Switching applications, the total one-way latency must be less than 1.6 ms in each direction
    - For Media Server and Media Gateway applications, the total one-way latency should be around 3-5 ms in each direction
    - Typical TDM to Packet conversion requires 1-2 packet buffers
    - Typical Packet to TDM conversion requires at least 2-3 packet buffers
    - Typical packet delivery times on a properly engineered Fabric should be less than 300 us.
    - Typical PSTN Interface devices (e.g. Framers) buffer 125-250 us.
    - Implications:
        - Use 1ms packet rate for Media Service and Media Gateway applications
        - Use 125 us packet rate for DS0 Switching applications

2. Support High Density

- Implication: Multiple TDM connections per packet (multiplexed format)
- Implication: For processor endpoints, all data must be 64-bit aligned.

3. Easy to control in High Call Rate applications

- Implication: Ability to easily move TDM channels from one position to another within a packet or between packet flows (i.e. no end-to-end control message required for this operation).

These rationales are further detailed in the sections below.

2.1. Latency

Long Distance applications like Voice Over IP typically employ echo cancellers to deal with delay. While this approach works well for VoIP, many other applications can't tolerate high transmission delays. Some examples are given in the sections below.

2.1.1. DS0 Switching

An example of this is when a customer uses debit-card to make a local call. This is typically called a "hairpin connection". Since the call is routed back to the Local Exchange Carrier switch, there are no echo cancellers in the path. Echo cancellers are typically only used by long distance carriers.

Another example of DS0 switching is when the voice over packet infrastructure is used to build the Local Exchange Carrier switch itself.

For these types of applications, the total one way end-to-end latency must be less than 1.6 ms. One-way end-to-end latency refers to the amount of time it takes to get from the input port of the system to the output port of the system.

TDM to Packet conversion implementations typically buffer 1-2 packets. Packet to TDM conversion implementations typically buffer at least 2-3 packets. <u>This totals a 3 to 5 packet buffer delays.</u>

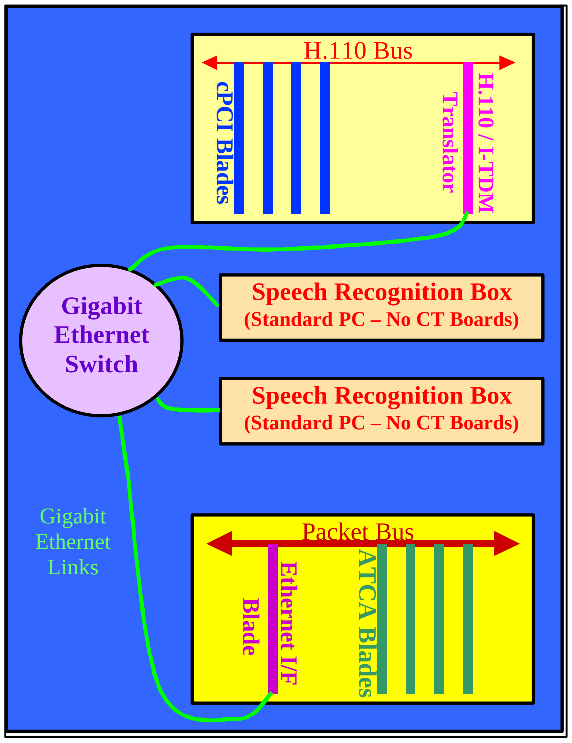
In addition, the packet delivery times on a properly engineered Fabric may reach 300 us. There are also some delays typically associated with the PSTN interfaces (e.g. Rx Slip buffers in the framer). With all this in mind, there is little choice than to send packets at the sample rate for these types of applications.

<u>Implication: For DS0 Switching applications, use 125 us packet rate</u>

2.1.2. Speech Recognition Barge-In

Speech recognition applications typically require a Barge-In time of less than 100ms. The term "Barge-In" refers to the amount of time it takes the system to stop playing a prompt after the user starts to speak. One key differentiating feature of Speech Recognition algorithms is this Barge-In time. If a significant amount of time is used up in the transport of TDM from the PSTN interface box to the Speech Processing Server, then the total Speech Recognition Barge-In time is increased, making the Speech Recognition solution less competitive.

Figure 4: Speech Recognition Example



In order to meet Barge-In requirements for Speech Recognition, the delivery protocol mechanism must have low latency. For this reason, a VoIP version TDM transport would use 5ms G.711. Note that the 5ms is not the delivery latency. To calculate latency, the packet unit (e.g. 5ms) must be multiplied by a factor of 3-5 to account for minimal buffering and jitter queues. This produces latency in the range of 15-25 ms. Barge-In should occur within 100 ms of the time the user begins his utterance. Higher delivery latencies (e.g. 10ms VoIP) make it harder for Speech Algorithms and control code to meet this 100 ms requirement.

By contrast, I-TDM only takes 3-5 ms for delivery latency. This represents only 3-5% of the total Barge-In time.

Note that some other Media Services (e.g. conferencing) also have latency issues.

<u>Implication: For Media Service and Gateway applications, use 1ms packet rate.</u>

2.2. High Density

<u>Multiplexing.</u> I-TDM allows many TDM channels in a single packet. This provides efficiency in Fabric bandwidth utilization, but more importantly this dramatically lowers packets per second which decreases the processing and/or hardware requirements at the endpoints.

<u>64-bit alignment for Processor endpoints</u>. Processors are being used increasingly in Voice Servers. Most modern processors have 64-bit data paths (e.g. Pentium®, Sparc®, PowerPC®, Network Processors, etc). These processors work much more efficiently when the data is aligned on 64-bit boundaries.

2.3. Destination Based Channel IDs

The basic architecture of I-TDM implies that each Channel ID within an I-TDM payload is directly related to some resource at the destination node (e.g. a local TDM port timeslot number). Since the destination node allocates the I-TDM Channel IDs, there is generally a 1:1 correspondence between in the incoming I-TDM Channel IDs and the local endpoints of the destination node, so that no table lookup is required to route the TDM channels at the destination.

2.4. Destination Based Packet Flow IDs

The basic architecture of I-TDM implies that the Packet Flow ID associated with an I-TDM payload is directly related to some resource at the destination node (e.g. a jitter buffer). Since the destination node allocates the I-TDM Packet Flow IDs, the Flow-ID coming into that destination node will be unique regardless of which source node sent it. So there is no need to look at a source node identifier or do a hash lookup to completely identify an I-TDM packet flow.

2.5. Grooming

I-TDM allows TDM timeslot connections to be moved between I-TDM multiplexed packet flows with no changes in the control layer.

Consider the following example. Let's say you have 400 TDM timeslot connections from Node A to Node B. In the 1ms mode, this would require 3 I-TDM packets to be sent every 1ms. A little while later, 300 of these connections are re-routed from Node A to Node C. The remaining 100 channel connections from Node A to Node B require only one packet every 1ms. But these remaining 100 connections are still scattered among 3 partially filled packets. The TDM connections must be moved to form 1 efficient packet. This is typically called grooming.

The diagram below shows a simplified example of this type of grooming.

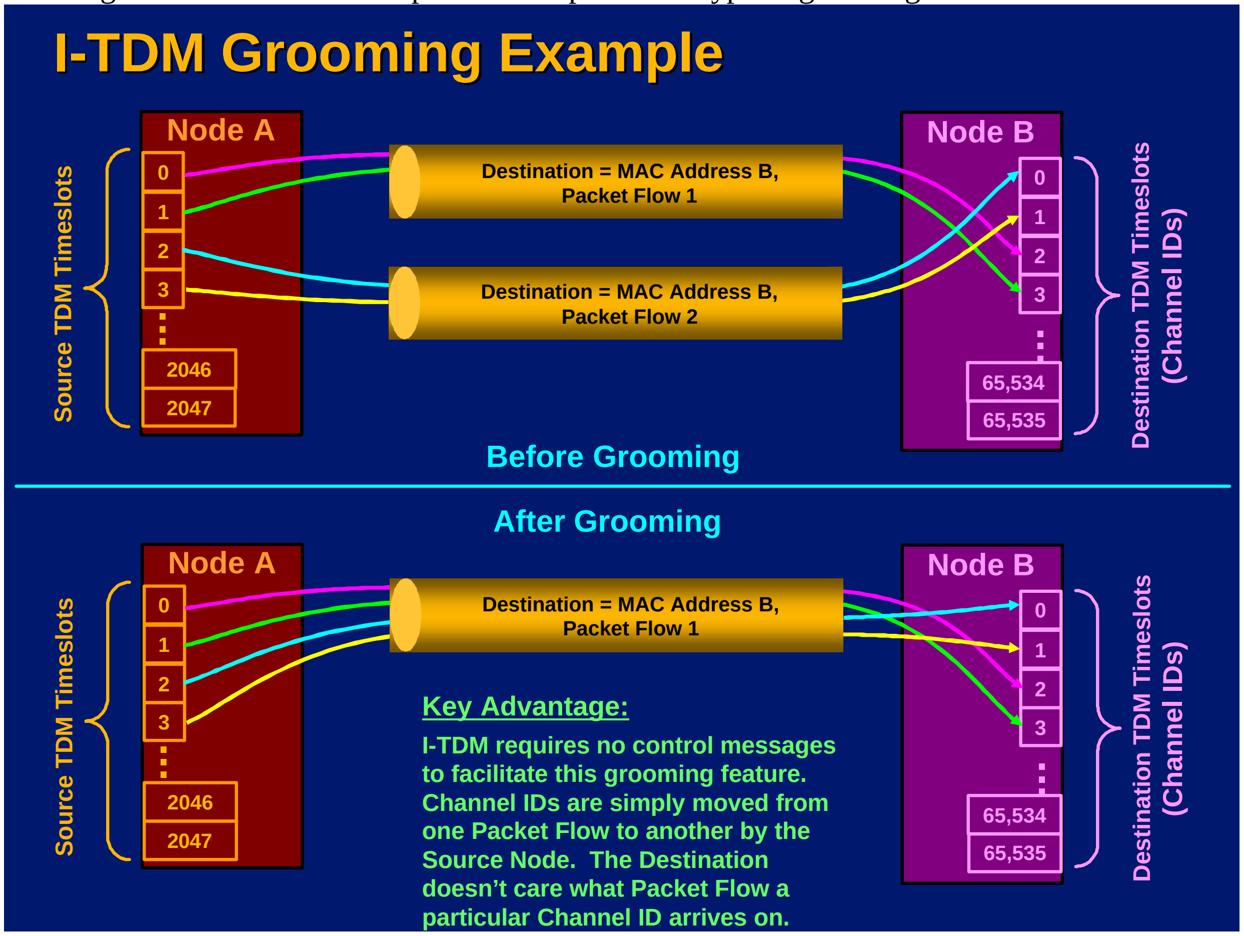

I-TDM Grooming Example
Node A
Source TDM Timeslots
0
1
2
3
2046
2047
Destination = MAC Address B, Packet Flow 1
Destination = MAC Address B, Packet Flow 2
Node B
0
1
2
3
65,534
65,535
Destination TDM Timeslots (Channel IDs)
Before Grooming
After Grooming
Node A
Source TDM Timeslots
Destination = MAC Address B, Packet Flow 1
Node B
Destination TDM Timeslots (Channel IDs)
Key Advantage:
I-TDM requires no control messages to facilitate this grooming feature. Channel IDs are simply moved from one Packet Flow to another by the Source Node. The Destination doesn't care what Packet Flow a particular Channel ID arrives on.

Figure 5: I-TDM Grooming Example

I-TDM allows this grooming to occur with no changes in the control layer (i.e. no end-to-end message required). In this way, the multiplexing of multiple TDM channels into a single packet is hidden from the higher layers. This, in turn, allows standard connection protocols (e.g. SIP) to be used for I-TDM.

The ability to easily move voice connections between multiplexed TDM packet flows is particularly important when the TDM connections change frequently. A system that requires a separate control message to groom each TDM connection will present a huge MIPs load on the

Control Processor. Configuring a Control Processor to keep up with this MIPs load will increase the hardware cost, space & power. Since I-TDM bypasses most of this message processing, the Control Processor required for I-TDM is relatively cheaper, smaller and burns fewer watts.

2.6. I-TDM vs. Other Protocols

2.6.1. I-TDM vs. VoIP (RFC1890)

- VoIP is designed for Voice over the Internet
- I-TDM is optimized for Voice over LAN or Backplane
- Major Differences between I-TDM and VoIP
    - I-TDM does not require the Internet Protocol (no IP Stack required)
    - Up to 512 channels per I-TDM packet
        - Most VoIP stacks only allow 1 channel per packet
    - I-TDM is much lower latency
        - I-TDM is 125us or 1ms vs. VoIP typical minimum of 5 or 10ms
    - I-TDM requires much less processing overhead

Bottom Line: VoIP is not an optimal protocol for connecting large numbers of TDM voice channels between equipment residing within the same room or building.

2.6.2. I-TDM vs. TDMoIP, PWE3, and other similar protocols

- TDMoIP and PWE3 are designed for emulating TDM Leased Lines over the Internet (i.e. static point to point connections)
- I-TDM is optimized for Voice processing systems (i.e. high call rate applications where connections change constantly)
- Major Differences between I-TDM and TDMoIP
    - I-TDM does not require the Internet Protocol (no IP Stack required)
    - I-TDM supports 64-bit (1ms) TDM channel transfer units. TDMoIP and PWE3 use only 8-bit (125us) TDM channel transfer units. The 8-bit transfer units make it difficult to endpoint large numbers of TDMoIP or PWE3 channels in software (e.g. Pentium*, Sparc*, PowerPC*, Network Processors, etc.) unless there is a TDMoIP / PWE3 hardware assist circuit.
    - I-TDM requires much less processing overhead for changing connections

Bottom Line: TDMoIP and PWE3 are not optimal protocols for high call rate applications where connections change constantly. TDMoIP and PWE3 are also not optimal protocols when the channel connection endpoint is a processor (unless there is specific TDMoIP / PWE3 hardware tied to the processor).

2.6.3. I-TDM vs. VoMPLS, Multiplexed RTP, and other similar protocols

Recently, several other multiplexed Voice over Packet formats have become available. These include:

- Voice over MPLS (VoMPLS)
- draft-ietf-avt-aggregation-00 (An RTP Payload Format for User Multiplexing)

While these formats address the optimizations of placing many voice calls in a single packet, there are still 2 key aspects that they do not address:

1) All of these types of formats have an 8-bit channel identifier. Given that most systems scale beyond 256 voice channels, the 8-bit channel identifier means that you also need a packet flow label to completely identify the channel. This effectively prohibits channels from moving between packet flows. The net result is that, with these types of protocols, the upper level application must manage which channels are allocated into which packets. This scenario makes it very difficult to control individual channel connections in high call rate applications. By contrast, I-TDM uses 16 or 24-bit channel identifiers, which allows for 64K to 16M channels per I-TDM endpoint (depending on the I-TDM mode used).
2) All of these types of formats have relatively high latency (i.e. 5-10 ms minimum).

Bottom Line: VoMPLS, Multiplexed RTP, and other similar protocols were generally designed for WAN voice over packet applications where low latency and high call rates aren't required.

2.7. TDM Clocking issues

Most of the other protocols mentioned above (e.g. VoIP, PWE3, TDMoIP, VoMPLS, etc) have mechanisms to translate the TDM timing (clock) information from the source to the destination. This is necessary for WAN Voice over Packet applications.

For I-TDM target applications, embedding clocking information into the packet flow is typically not required. This is because I-TDM equipment is typically all physically close to each other (i.e. in the same chassis, room or building). This allows the use of separate wires dedicated for clocking.

Note that, since I-TDM deals primarily with DS0 switching and end-pointing applications, the assumption is that I-TDM endpoints will typically be tied to a common clock.

For example, the ATCA chassis has a redundant pair of 8KHz clock reference wires running over the backplane. All ATCA boards have access to these two 8KHz clock signals.

Between 2 chassis, T1/E1 wires are used. In a central office environment, these T1/E1 wires come from a central timing source (i.e. a “clock box”). For enterprise solutions, all you need is to lock the clock from one chassis to the next (e.g. fan-out or daisy-chain) using T1/E1 wires. The ATCA board that receives the T1/E1 timing information then drives the 8KHz clock reference wires to all other boards in the system.

The only significant timing issue with I-TDM arises on a Host Media Processing platform. These modules typically do not use 8KHz clock reference wires or T1/E1 timing ports. However, the timing requirements for Host Media Processing (HMP) platform applications are relatively loose since there are no physical TDM interfaces to deal with. With this in mind, the I-TDM format includes a timestamp within the I-TDM headers (e.g. at the SFP layer). The combination of low latency packet arrival coupled with a timestamp to detect lost frames allows the HMP platform to lock the local process timing to the external master clock well enough to satisfy HMP latency/jitter requirements.

2.8. I-TDM LAN/Backplane Configuration Issues

I-TDM requires low latency packet transmission. When I-TDM is used in on Ethernet LAN, this is probably most easily configured using separate LAN (e.g. separate Gigabit Switch) just for I-TDM traffic. VLAN priorities could also be used, but this may be more expensive to support for LAN configurations.

When I-TDM is used in a packet backplane, data and control packets may coexist with I-TDM on the same fabric as long as these packets are managed properly (i.e. priority for I-TDM packets guarantees low latency delivery).
Bottom Line: I-TDM is not meant for Corporate LANs or the Internet; the latency of these networks is too high. For I-TDM over LAN, a private LAN just for I-TDM traffic is usually best. The recent price trend for Ethernet Switches makes this approach economical. A managed LAN will also work well, but requires more configuration. The managed approach is generally more suited to a packet backplane where data, voice and control packets each have their own priority (e.g. separate VLAN priority, Virtual Lane, etc.) for each traffic class.

3. I-TDM PROTOCOL

This section defines the I-TDM protocol.

This Short Form version of the SFP.1 specification is intended to provide an overview of the I-TDM protocol. Note that most of the I-TDM detailed low-level requirements have been removed in this Short Form version. Do not attempt to design to this Short Form specification.

3.1. *I-TDM Headers*

I-TDM can be adapted to various packet formats (e.g. Ethernet, Infiniband, AS, etc).

3.1.1. I-TDM over Ethernet

An example using Ethernet with VLAN is given below.

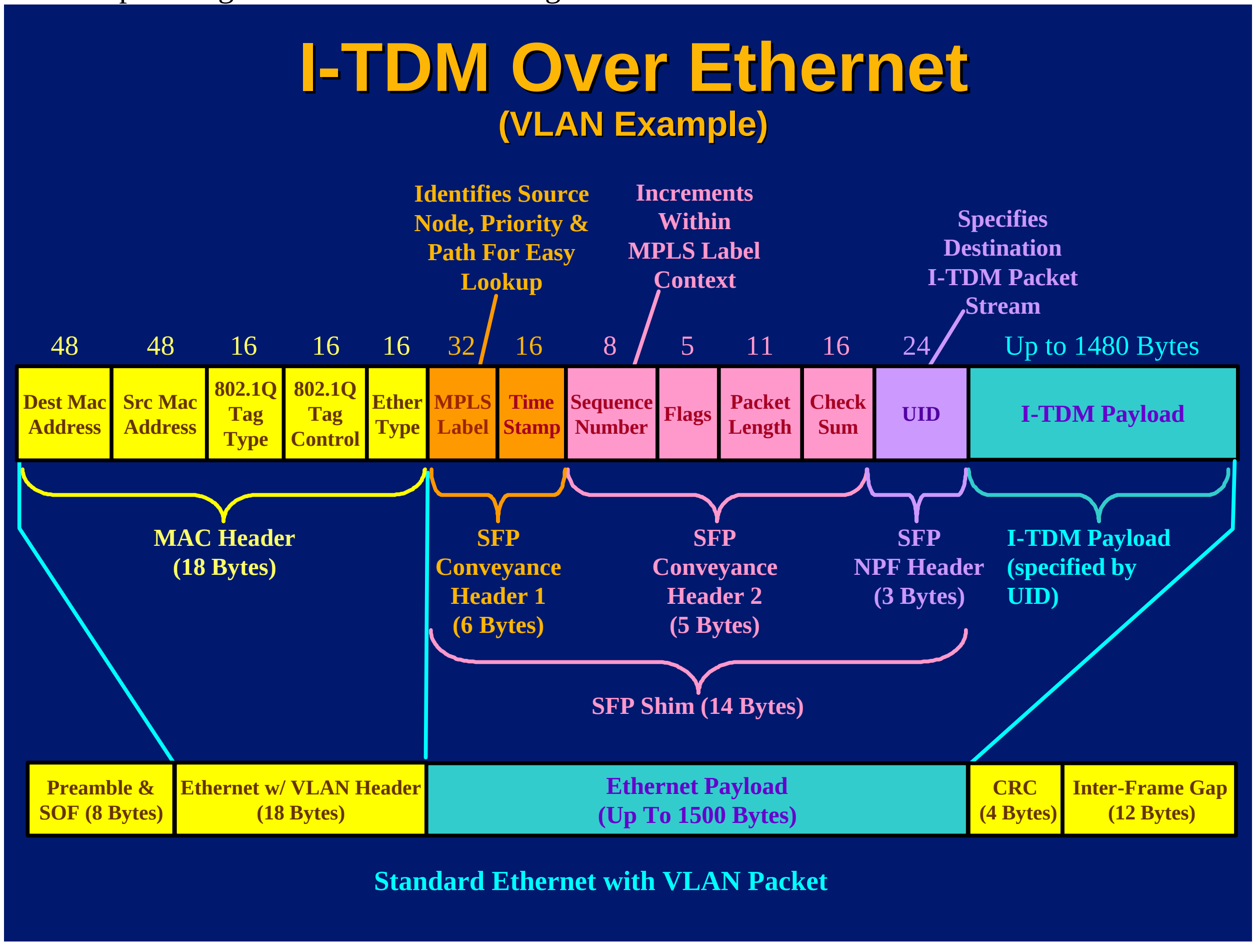

I-TDM Over Ethernet
(VLAN Example)
Identifies Source Node, Priority & Path For Easy Lookup
Increments Within MPLS Label Context
Specifies Destination I-TDM Packet Stream
48
48
16
16
16
32
16
8
5
11
16
24
Up to 1480 Bytes
Dest Mac Address
Src Mac Address
802.1Q Tag Type
802.1Q Tag Control
Ether Type
MPLS Label
Time Stamp
Sequence Number
Flags
Packet Length
Check Sum
UID
I-TDM Payload
MAC Header (18 Bytes)
SFP Conveyance Header 1 (6 Bytes)
SFP Conveyance Header 2 (5 Bytes)
SFP NPF Header (3 Bytes)
I-TDM Payload (specified by UID)
SFP Shim (14 Bytes)
Preamble & SOF (8 Bytes)
Ethernet w/ VLAN Header (18 Bytes)
Ethernet Payload (Up To 1500 Bytes)
CRC (4 Bytes)
Inter-Frame Gap (12 Bytes)
Standard Ethernet with VLAN Packet

Figure 6: I-TDM over Ethernet (VLAN Example)

Note that I-TDM doesn't require Internet Protocol headers. All that is required is a Flow ID and a timestamp for detecting dropped packets.

For Ethernet fabrics, the I-TDM Headers use the SFP.0 System Fabric Plane Format.

In other words, the terms "I-TDM Headers" and "SFP Shim" are used interchangeably for Ethernet based fabrics. The SFP Shim allows the encapsulation of many different protocols (e.g.

IP, AAL-2, I-TDM, Wireless Protocols, etc) on a common fabric. The use of a Shim is a common practice in Router architectures. Logically, the Shim sits between Layer 2 and Layer 3 of an OSI stack (hence the name "Shim").

Further details on the SFP shim are specified in the *"SFP.0 System Fabric Plane Format"* specification.

Note that the SFP Shim is sized so that the beginning of the payload starts on a 64-bit boundary. This increases performance when terminating I-TDM flows on processors with 64-bit data paths (e.g. Pentium®, Sparc®, PowerPC®, Network Processors, etc).

For Ethernet Fabrics, the I-TDM Flow ID is carried within the SFP.0 UID field.

The I-TDM Flow ID generally corresponds to a Jitter Queue at the Destination Node. The I-TDM Flow ID also identifies the basic packet type (e.g. I-TDM Control or I-TDM Data) as well as the Payload Format and the Packet Rate.

3.2. I-TDM Payload Formats

I-TDM supports 2 basic modes of TDM data transport:

- 1ms Mode
- 125us Mode

The 1ms Mode is required for all I-TDM implementations.

The 125us Mode is required for those implementations that support DS0 Switching functions (e.g. hairpin connections, PBXs, PSTN Switches, etc.).

For example, in the diagram below, all 3 types of modules need to support the 1ms mode, while only the PSTN Interface Modules requires support for the 125us mode. In other words, support for the 125us mode on the IP Gateway and Media Service Modules is optional.

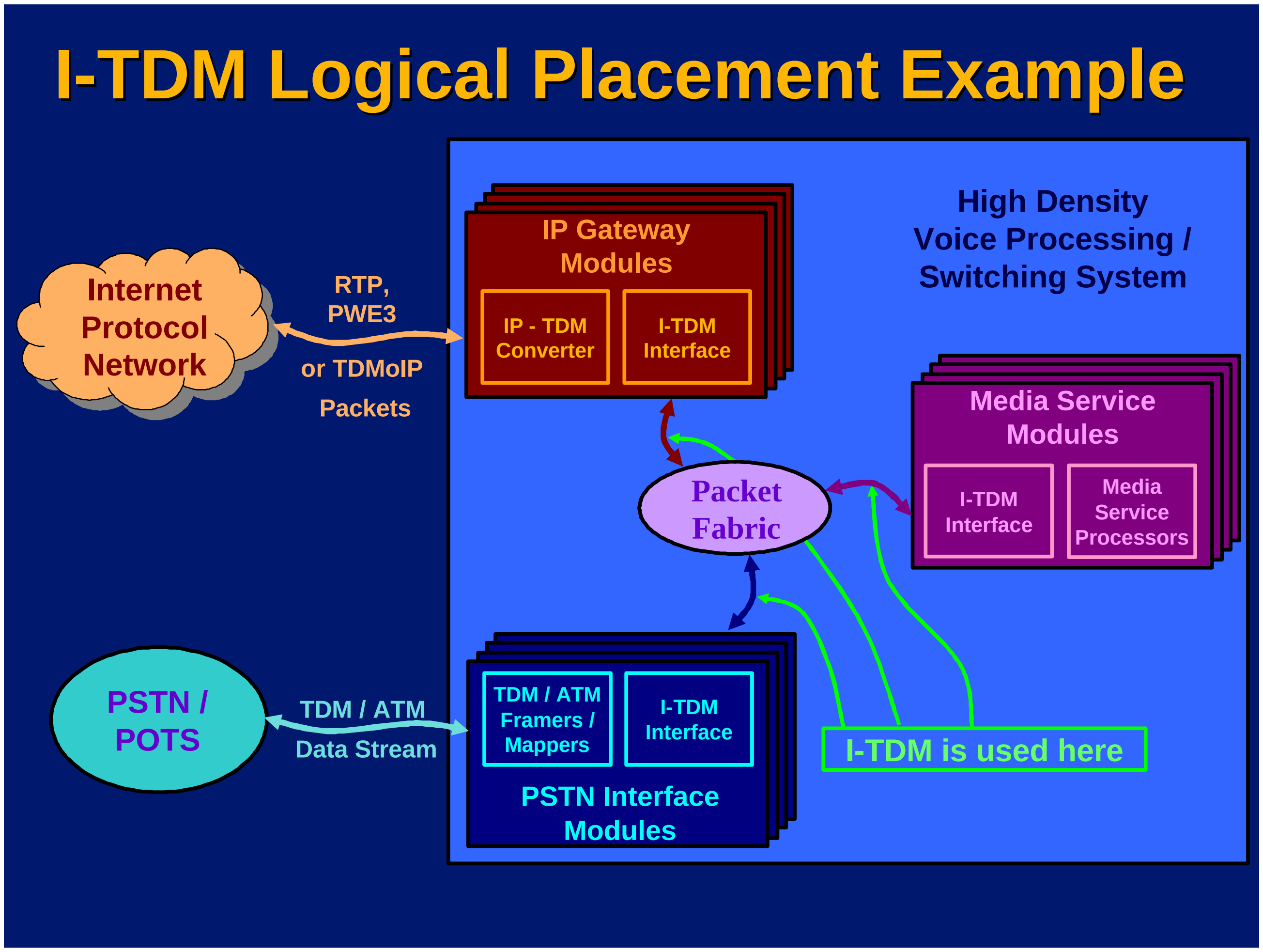


Figure 7: I-TDM Logical Placement Example

Note that, for implementations that provide DS0 Switching functions, support for the 1ms mode need not scale to the capacity of the endpoint node. In other words, the numbers of channels of 1ms and 125us mode is vendor specific and need not be equal. For example, one vendors implementation might support 32K channels of 125us mode and only 2K channels of 1ms mode.

The only requirement for the number of channels is this: an I-TDM endpoint will support a minimum of 1 (one) 1ms mode I-TDM channel.

It is anticipated that individual vendors will differentiate their products with the number of channels of each mode supported.

3.2.1. Implementation Implications

Although there are no explicit requirements for hardware or software based implementations, the near term implication is that the I-TDM 125us mode will typically be implemented in hardware based endpoints. The I-TDM 1ms mode can be implemented in hardware and/or software based endpoints.

3.2.2. Grooming

Within the context of this document, the term “grooming” corresponds to the ability to move TDM connections around, either within a packet or between packet flows.

All I-TDM modes allow the location of a channel to be moved within a packet. It is possible to groom Channel IDs across flows in either 125us or 1ms mode. However, due to the command timing and jitter buffer issues associated with the 125us mode, there would probably be lost samples when channels are groomed across flows. In 1ms mode, there typically would be no lost samples when channels are groomed across flows.

3.3. I-TDM 1ms Mode Payload Format

The 1ms mode uses a self identifying Channel ID scheme as shown in the diagram below. In other words, the Channel IDs for all TDM channels are present in every packet.

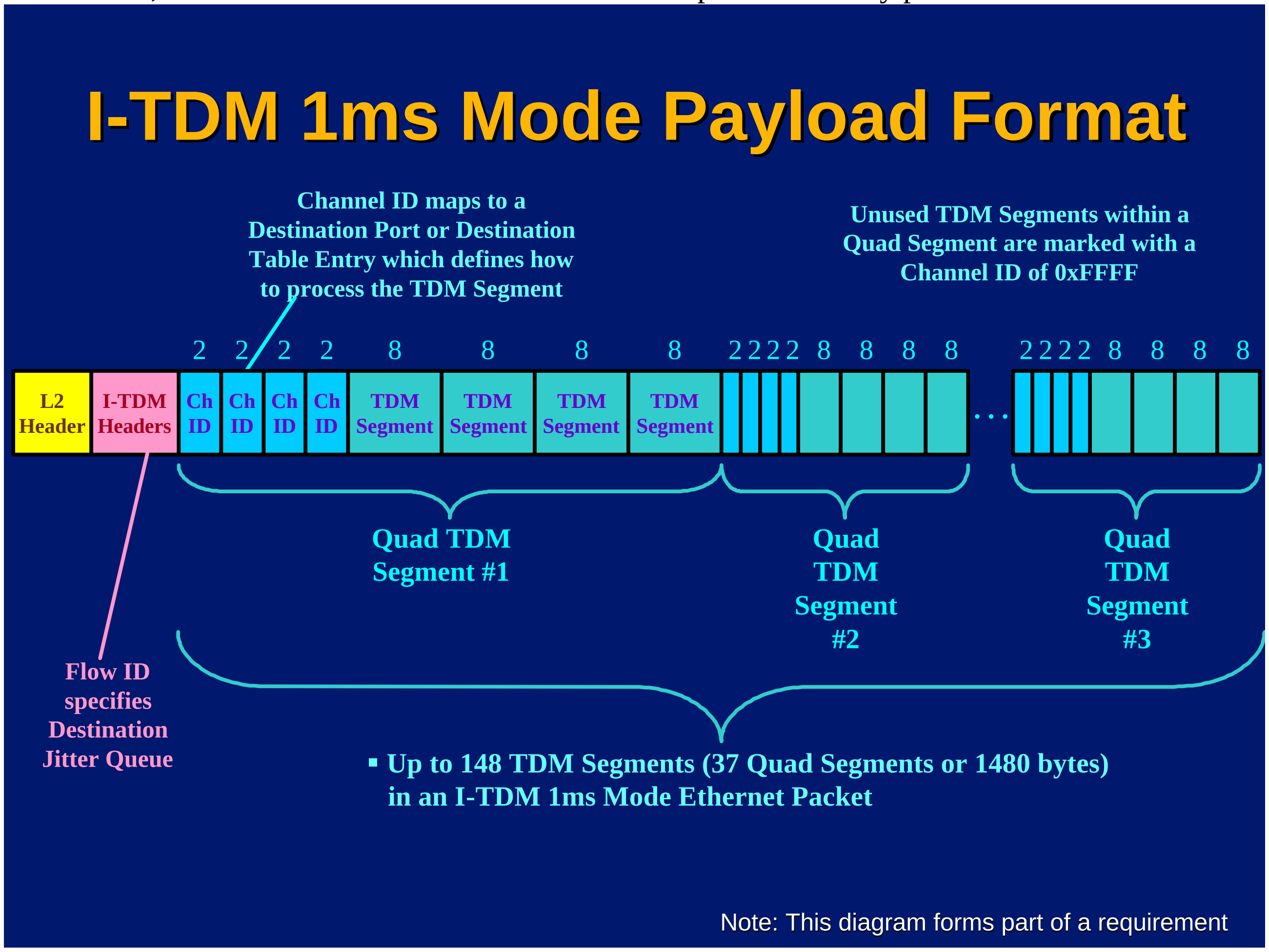

I-TDM 1ms Mode Payload Format
Channel ID maps to a Destination Port or Destination Table Entry which defines how to process the TDM Segment
Unused TDM Segments within a Quad Segment are marked with a Channel ID of 0xFFFF
2 2 2 2 8 8 8 8 2 2 2 2 8 8 8 8 2 2 2 2 8 8 8 8
L2 Header
I-TDM Headers
Ch ID
Ch ID
Ch ID
Ch ID
TDM Segment
TDM Segment
TDM Segment
TDM Segment
. . .
Quad TDM Segment #1
Quad TDM Segment #2
Quad TDM Segment #3
Flow ID specifies Destination Jitter Queue
▪ Up to 148 TDM Segments (37 Quad Segments or 1480 bytes) in an I-TDM 1ms Mode Ethernet Packet
Note: This diagram forms part of a requirement

Figure 8: I-TDM 1ms Mode Payload Format

Refer to the full specification of SFP.1 for details on the Packet Rate, TDM Segment, and Quad TDM Segment requirements.

3.3.1. Channel ID

The Channel ID field completely identifies the TDM channel number of the corresponding TDM Segment at the destination node.

Since the Channel ID generally corresponds to some resource at the destination node (e.g. destination port #), the destination node will generally restrict the number space of the I-TDM Channel IDs bound toward it. For example, if the destination node supports 2048 I-TDM channels, the Channel IDs toward that destination node would typically be numbered 0-2047.

In another example, let's say a given destination node contains 32 DSPs, each of which supports 100 channels (3200 channels total). In this case, the I-TDM Channel IDs bound to this destination would probably have the following mapping:

Channel ID 0-99	Valid
Channel ID 100-127	Invalid
Channel ID 128 - 227	Valid
:	:
Channel ID 3968 - 4067	Valid
Channel ID 4068 - 65534	Invalid
Channel ID 65,535 (0xFFFF)	Inactive channel

Table 1: Channel ID assignment example

The intent of this is to minimize any translations between the I-TDM Channel ID and the physical channel resource at the destination node.

The Destination node is responsible for allocating the Channel ID number space. Note that this destination Channel ID allocation could be implemented in a variety of ways, such as:

- A Vendor Specific API is used to dynamically query a destination node for an available valid Channel ID.
- A Vendor simply documents the Channel ID number space restrictions and the User Application implements these restrictions.
- etc.

So the Channel ID is not just identifying the TDM channel number within the packet, but rather identifies the TDM port number within the destination endpoint node.

Since the Channel ID completely identifies the TDM channel at the destination node, there is no need to translate this number at the destination. Additionally, if multiple I-TDM packet flows are required between two endpoints, the TDM channels can be moved around between these packets at the source node with no affect on the TDM channel connection.

The best analogy for this is a passenger train. When you buy a ticket for a particular destination, nobody cares which boxcar of the train you get on. In fact, if the train allows, you can even switch cars in the middle of the ride. Similarly, if multiple I-TDM packet flows exist between source and destination blades, a given Channel ID might start off a connection over one packet flow and switch to another packet flow during the TDM channel connection. The destination just looks at the Channel ID to determine what the channel number is. The destination node doesn't care which packet flow it arrives on. In this way, a source can add or remove packet flows depending on the number of active channel connections to a given destination.

This scheme allows the lowest layers of control code (e.g. Drivers and/or Firmware) to completely manage the I-TDM packet flows. The higher layers of control code only need to deal with channel connections. In this way, the multiplexing of multiple TDM channels into a single packet is hidden from the higher layers. This, in turn, allows standard connection protocols (e.g. SIP) to be used for I-TDM.

3.4. I-TDM 125us Mode Payload Format

The self identifying Channel ID scheme used in the I-TDM 1ms mode is not practical for the 125us packet rate. In other words, if the Channel IDs for all TDM channels were present in every packet, this would incur too much bandwidth overhead on the Fabric for 125us mode packets.

So instead, the I-TDM 125us Mode uses a dynamic positional Channel ID scheme as shown in the diagram below. In other words, TDM channels are identified by their position within the packet, but channels can move this position dynamically without the use of any separate control messages.

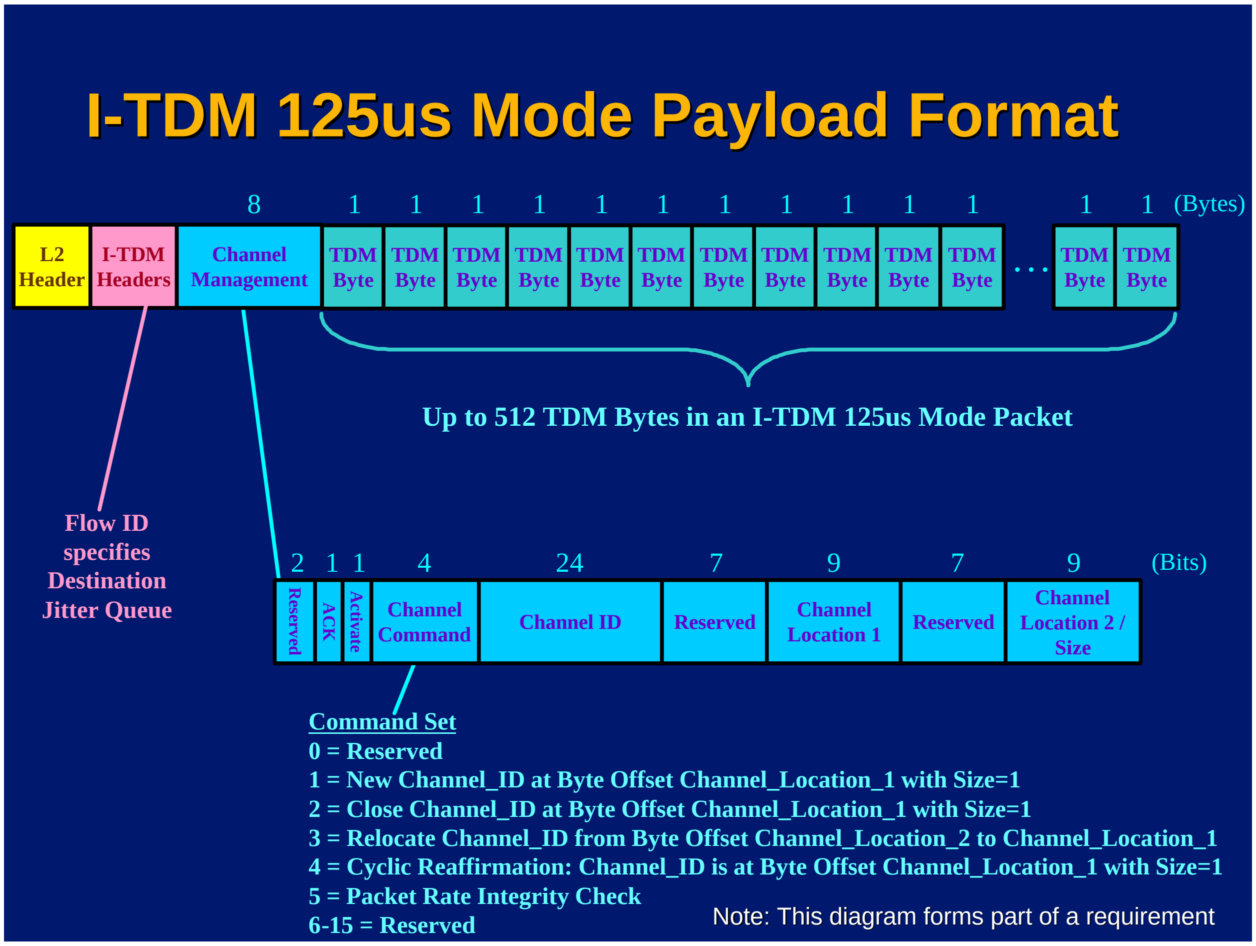

I-TDM 125us Mode Payload Format
8
1
(Bytes)
L2 Header
I-TDM Headers
Channel Management
TDM Byte
. . .
Up to 512 TDM Bytes in an I-TDM 125us Mode Packet
Flow ID specifies Destination Jitter Queue
2 1 1 4 24 7 9 7 9 (Bits)
Reserved
ACK
Activate
Channel Command
Channel ID
Reserved
Channel Location 1
Reserved
Channel Location 2 / Size
Command Set
0 = Reserved
1 = New Channel_ID at Byte Offset Channel_Location_1 with Size=1
2 = Close Channel_ID at Byte Offset Channel_Location_1 with Size=1
3 = Relocate Channel_ID from Byte Offset Channel_Location_2 to Channel_Location_1
4 = Cyclic Reaffirmation: Channel_ID is at Byte Offset Channel_Location_1 with Size=1
5 = Packet Rate Integrity Check
6-15 = Reserved
Note: This diagram forms part of a requirement

Figure 9: I-TDM 125us Mode Payload Format

Refer to the full specification of SFP.1 for the I-TDM 125us Mode requirements.

3.4.1. Channel Management Example

The dynamic positional Channel ID scheme supported by the Channel Management Structure works as follows. Let’s start with no TDM channels (i.e. there are no bytes in the packet after the Channel Management Structure). The first “New Channel ID” command announces the addition of a new TDM channel at location 0 (i.e. the byte just after the Channel Management Structure). The second “New Channel ID” command announces the addition of a new TDM channel at location 1, and so on until there are, for example, 100 TDM samples in the packet.

Note that, while the Channel Locations of the TDM samples are numbered sequentially from 0-99, the Channel IDs associated with these locations are somewhat random. For example, location #0 could be associated with Channel ID #663, while location #1 is associated with Channel ID #423. The Channel ID corresponds to the active channel number at the destination node where the TDM sample will be connected (e.g. the destination port #). The location (or position within the packet) that carries this Channel ID is somewhat arbitrary and, as we will see, can move around.

Now let's say that Channel ID #428 at location #43 is no longer needed. The "Close Channel ID" command is used to remove the channel. But now there is a hole at location #43 (i.e. there is no Channel ID associated with location #43. At this point, one of three things can happen:

1) The hole could persist (e.g. location #43 can continue to have no Channel ID)
2) The hole might be filled with a new connection (e.g. another "New Channel ID" command can be issued at location #43)
3) The Packet could shrink to fill the Hole. This is done using the "Relocate Channel ID" command. The Channel ID associated with the last byte offset in the previous packet is moved to fill the hole, and the size of the packet is decremented by 1.

In this way, Channel IDs change their position dynamically.

3.4.2. Channel ID

As with the 1ms I-TDM mode, the Channel ID completely identifies the TDM channel at the destination node. Since the Channel ID generally corresponds to some resource at the destination node (e.g. destination port #), the destination node will generally restrict the number space of the I-TDM Channel IDs bound toward it. The Destination node is responsible for allocating the Channel ID number space. Refer to the "Channel ID" section within the 1ms Mode for further explanations and examples of Channel ID number space restrictions.

3.4.3. ACK and Activate Bits

These bits are used to implement a low-level reliability protocol for managing the dynamic I-TDM channel locations.

The intent of this low-level reliability protocol is to allow a hardware-based implementation of the reliability mechanism. This scenario alleviates the local processor from sending and receiving messages to move I-TDM connections within or between packet flows, which in turn greatly relieves the MIPs load on the local processor.

The reliability protocol scheme relies on the following I-TDM 125us mode characteristics:

- The Command Structure is in the same packet as the I-TDM data.
- The packet rate is relatively high (i.e. 125us for the mode being defined here).
- The end-to-end latency is required to be low (see latency section below).
- The destination node can acknowledge commands quickly (e.g. in hardware).
- Only one command per outgoing Flow ID is in process at a time.
- The available command bandwidth greatly exceeds the requirements for call rates.

Note that the reliable protocol involves a pairing of I-TDM Flow IDs (e.g. one Flow ID from Node A to Node B, and another associated Flow ID from Node B back to Node A). The endpoint nodes are responsible for maintaining information about this incoming/outgoing I-TDM Flow ID pairing. Refer to the I-TDM Control Protocol sections below for further details on establishing I-TDM Flow ID pairs.

Refer to the full specification of SFP.1 for details on the Command Scheduling, ACK Processing, and Destination Behavior requirements associated with the ACK and Activate bits.

3.4.4. Command Bandwidth

A typical TDM connection would go through 3 phases:

1. New Channel ID command (location is typically at end of packet)
2. … Duration of call …
3. Close Channel ID or Relocate Channel ID command

The Relocate Channel ID command can be used to close the inactive channel as well as shorten the packet. The Close Channel ID command is used when the channel to be closed is in the last location of the packet.

The packet rate is 125us (i.e. 8000 packets per second). The maximum number of TDM time-slots per packet is 512. This allows 15-16 commands per second for each channel. A typical channel connection requires at worst case 16 packet intervals (8 command slots for New Channel ID plus 8 command slots for Close Channel ID or Relocate Channel ID commands).

This means that the available command bandwidth would support channel connections that last as short as about 1 second. Note that typical average call rates are much higher than this. For example, the lowest known call rate specifications require a 6 second minimum average call duration, and this is an aggressive specification (i.e. rarely if ever reached in real life).

So the bottom line is that the available command bandwidth far exceeds any and all requirements.

As another data point, a full (i.e. 512 channel) I-TDM 125us mode packet may be populated in 512ms or less (i.e. using New Channel ID commands worst case once every 8 packet intervals).

Note that the limit of 512 channels per packet was chosen to accommodate a balance between bandwidth efficiency, command bandwidth per channel, and look-up table size issues.

For Ethernet based fabrics, since the max size packet is 2-3 times this 512 channel limit, SFP.0 multiplexing can be used to combine multiple 125us mode I-TDM packets between the same endpoints into a single SFP frame.

3.5. Optional I-TDM Payload Formats

The I-TDM protocol also specifies 3 optional data payload formats/modes:

- Implicit Multi-Timeslot Mode
- Explicit Multi-Timeslot Mode
- CAS Signaling Transport
- Optional Negotiable Packet Rates for 125us Mode and Explicit Multi-Timeslot Mode.

Although support for each of these modes is optional, a claim of compliance for a given optional mode must meet all of the requirements for that optional mode. In other words, a vendor can't claim compliance to the optional mode unless all of the requirements for that optional mode are met.

3.6. TDM Connections that Span Multiple Timeslots

Some applications require support for TDM connections consisting of multiple DS0s. I-TDM specifies 2 optional modes for carrying these multi-timeslot connections.

- Implicit Multi-Timeslot Mode
- Explicit Multi-Timeslot Mode

3.6.1. Implicit Multi-Timeslot Mode

The Implicit Multi-Timeslot Mode supports a mixture of different TDM connection sizes in the same packet. For example, a TDM video stream might require 6 DS0s to obtain a 384Kbit/sec TDM data rate. These multi-DS0 type connections have been given various names (bundles, hyper-channels, etc.). The Implicit Multi-Timeslot Mode allows these multi-timeslot connections to be transported in the same packet as more traditional 1-byte DS0 timeslots.

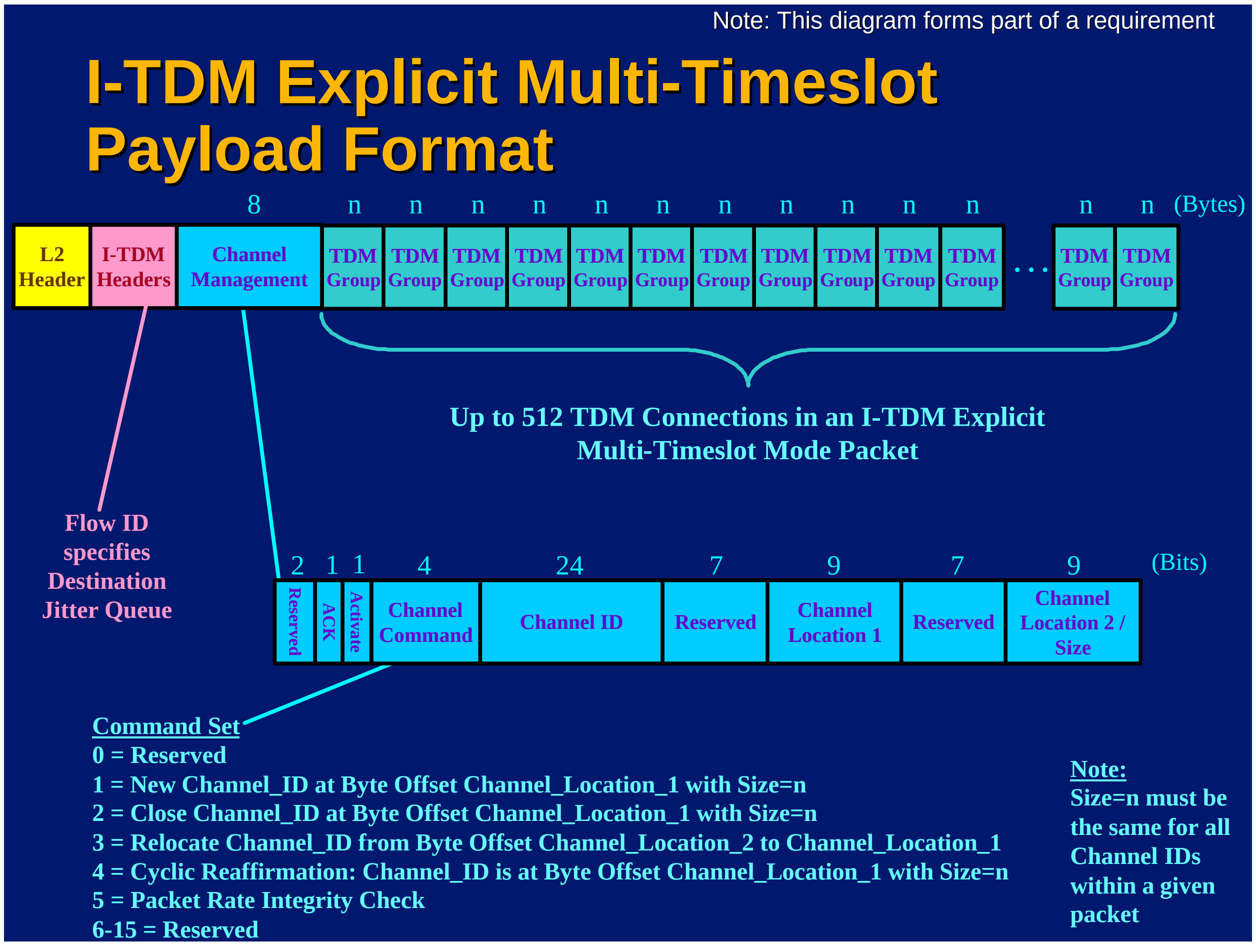
Note that there are no specific additional I-TDM payload formats, commands or modes to transport this mixture of data in the Implicit Multi-Timeslot Mode.

Rather, the I-TDM Implicit Multi-Timeslot Mode uses multiple Channel IDs (one Channel ID per DS0 timeslot), and the management of these multiple Channel IDs as a single connection will occur at the endpoints (i.e. above the basic I-TDM transport layer).

Refer to the full specification of SFP.1 for details on the Implicit Multi-Timeslot Mode.

3.6.2. Explicit Multi-Timeslot Mode

The Explicit Multi-Timeslot Mode supports a more efficient means of grooming large numbers of Multi-Timeslot connections of the same size. The restriction is that each I-TDM packet is limited to one size connection. This limitation greatly simplifies the movement of these types of connections.


Note: This diagram forms part of a requirement
I-TDM Explicit Multi-Timeslot Payload Format
8
n
(Bytes)
L2 Header
I-TDM Headers
Channel Management
TDM Group
. . .
Up to 512 TDM Connections in an I-TDM Explicit Multi-Timeslot Mode Packet
Flow ID specifies Destination Jitter Queue
2 1 1 4 24 7 9 7 9
(Bits)
Reserved
ACK
Activate
Channel Command
Channel ID
Reserved
Channel Location 1
Reserved
Channel Location 2 / Size

Command Set
0 = Reserved
1 = New Channel_ID at Byte Offset Channel_Location_1 with Size=n
2 = Close Channel_ID at Byte Offset Channel_Location_1 with Size=n
3 = Relocate Channel_ID from Byte Offset Channel_Location_2 to Channel_Location_1
4 = Cyclic Reaffirmation: Channel_ID is at Byte Offset Channel_Location_1 with Size=n
5 = Packet Rate Integrity Check
6-15 = Reserved

Note:
Size=n must be the same for all Channel IDs within a given packet

Figure 10: I-TDM Explicit Multi-Timeslot Payload Format

Refer to the full specification of SFP.1 for details on the Explicit Multi-Timeslot Mode.

Note that the packet rate of the Explicit Multi-Timeslot Mode can vary to accommodate multiple 125us samples per TDM group. For example the optional Explicit Multi-Timeslot Mode can be used to accommodate 250us or 500us packet transmission intervals. Refer to the "Optional Packet Rates" section below for details.

3.7. Channel Associated Signaling (CAS) Transport

I-TDM optionally supports the transport of the Channel Associated Signaling (CAS) bits (e.g. T1 Robbed bits or E1 Timeslot-16 bits).

The repetition rate for CAS signaling bits is given below:

- For T1, 4 CAS bits per channel are transferred every 3 ms.
- For E1, 4 CAS bits per channel are transferred every 2 ms.

3.7.1. CAS Overview

Its worth some time to study the application scenarios associated with CAS bit transport. There 3 basic ways of establishing a TDM connection via the Public Switched Telephone Network (PSTN):

1. Pulse Dialing
2. Tone Dialing
3. Packet-Based Dialing (e.g. ISDN, SS7)

CAS bits are only used in Pulse Dialing and Tone Dialing.

For Tone Dialing, the CAS bits are periodically monitored to see if a port is “off-hook” (i.e. to determine which ports are active). This type of monitoring might require some amount of processing to de-bounce any spurious CAS bits on the wire.

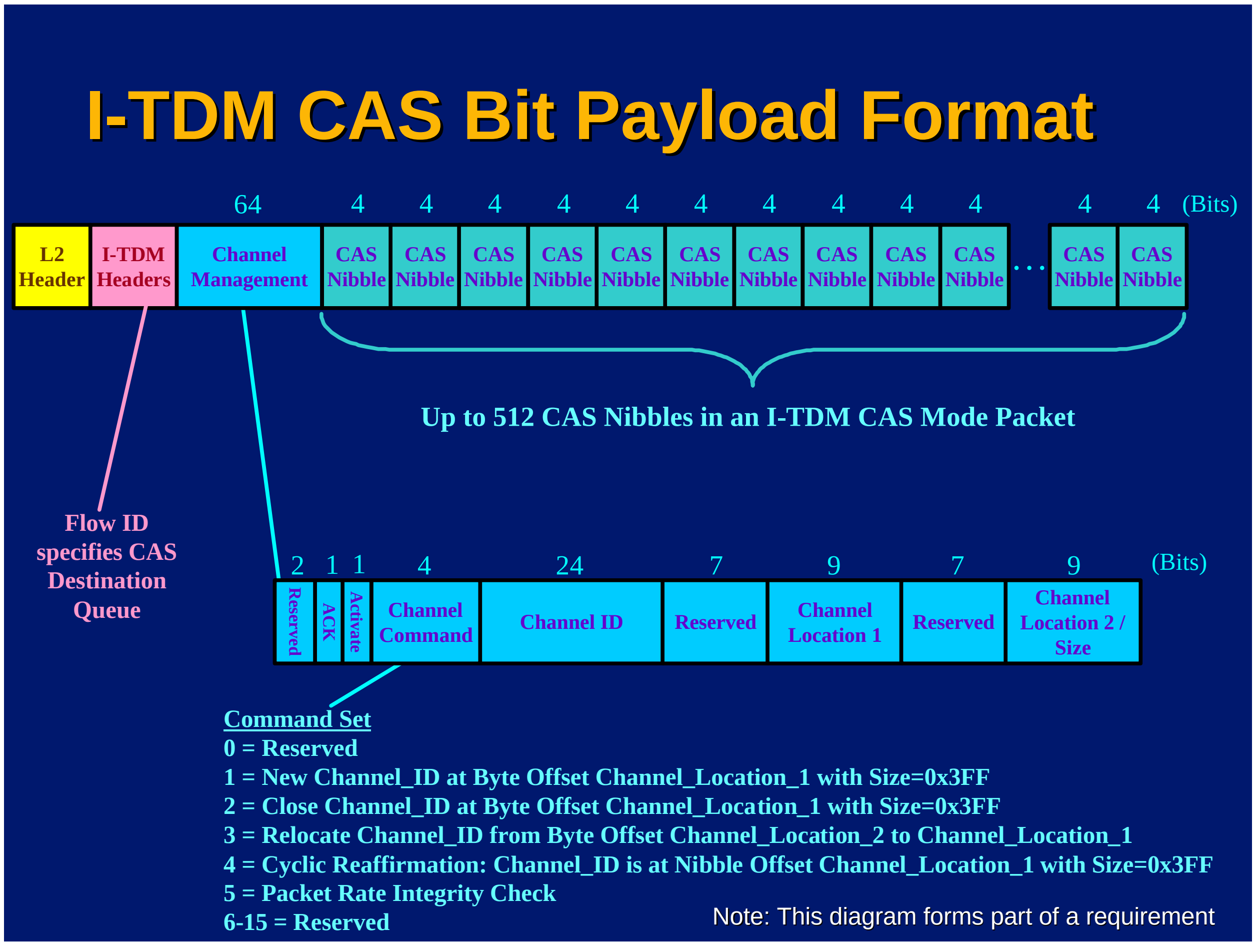
In Pulse Dialing, dial digits are encoded by pulsing the “hook” (i.e. the switch that moves when you hang up) at a nominal rate of 10 pulses per second (10 PPS). However, since mechanical friction rotary dial phones are still in use, the 10 PPS rate could vary. In addition, the duty cycle of the pulses can vary. There might also be noise on the line that causes spurious CAS bit values. All of this tends to call for a fair amount of processing to decode the Dial Pulse digits.

With this in mind, it is possible that a given system implementation would centralize this Dial Pulse processing. In other words, instead of implementing the Pulse Dialing processing function on every PSTN interface module, the system could implement a centralized Pulse Dialing processor module. This requires an I-TDM transport format to carry these CAS bits. Note that the CAS bit connections associated with this application scenario tend to be rather static (e.g. CAS bits for all channels connect to the centralized Pulse Dialing processor module). So for this scenario, the I-TDM CAS transport format doesn’t need to move channels around frequently.

Another scenario would be a very high density Voice Over Packet (VoP) or Media Gateway application where CAS dial pulse termination and relay could be distributed through system VoP processing resources to help with scalability (i.e. remove the central CAS processing resource, which can become a bottleneck as densities increase). In order to maintain a non blocking architecture and simplify control plane elements this requires that CAS channels are terminated on the same resource as the associated bearer VoP channel (i.e. the I-TDM data connection associated with the same channel). This would require an I-TDM CAS transport format that allows CAS channels to move around dynamically.

3.7.2. I-TDM CAS Bit Transport Format

The optional I-TDM CAS Transport Mode uses a dynamic positional Channel ID scheme, as shown in the diagram below. This is similar to the I-TDM 125us mode.


I-TDM CAS Bit Payload Format
64
4
(Bits)
L2 Header
I-TDM Headers
Channel Management
CAS Nibble
Up to 512 CAS Nibbles in an I-TDM CAS Mode Packet
Flow ID specifies CAS Destination Queue
2 1 1 4 24 7 9 7 9 (Bits)
Reserved
ACK
Activate
Channel Command
Channel ID
Reserved
Channel Location 1
Reserved
Channel Location 2 / Size

Command Set
0 = Reserved
1 = New Channel_ID at Byte Offset Channel_Location_1 with Size=0x3FF
2 = Close Channel_ID at Byte Offset Channel_Location_1 with Size=0x3FF
3 = Relocate Channel_ID from Byte Offset Channel_Location_2 to Channel_Location_1
4 = Cyclic Reaffirmation: Channel_ID is at Nibble Offset Channel_Location_1 with Size=0x3FF
5 = Packet Rate Integrity Check
6-15 = Reserved

Note: This diagram forms part of a requirement

Figure 11: I-TDM CAS Bit Payload Format

Refer to the full specification of SFP.1 for details on the Explicit Multi-Timeslot Mode.

Note that the I-TDM CAS Transport Mode is optional.

Also note that SFP.0 multiplexing can be used to combine a CAS format payload with another I-TDM format (e.g. 125us mode) payload in the same Ethernet packet.

3.8. Optional Packet Rates

I-TDM optionally supports packet rates other than 125us for the following two I-TDM data transport modes:

- I-TDM 125us Mode
- I-TDM Explicit Multi-Timeslot Mode

Specific optional packet rates can be negotiated between participating endpoints by using the Packet Rate field in the I-TDM control messages. Refer to the full specification of SFP.1 for details on the Optional Packet Rates.

3.9. Endian

The I-TDM protocol uses the normal Endian for network protocols (i.e Big Endian).

3.10. Multicast

The I-TDM protocol does not support broadcasting or multicasting I-TDM data packets. Adding support for I-TDM multicast would significantly complicate the I-TDM control protocol (see section below).

Note that the concept of multicast for I-TDM is particularly complicated since I-TDM is a multiplexed format.

However, many user applications require transport of a single DS0 timeslot to multiple I-TDM destinations. In addition, with the ecosystem that I-TDM envisions, it would be rare to know exactly what system level applications could be associated with a particular I-TDM based module.

Therefore, in order not to constrain user applications of a given vendor's implementation, this specification requires that all I-TDM implementations be capable of replicating the TDM time-slot data at the source (i.e. above the I-TDM protocol layer). In other words, all I-TDM implementations must be capable of sending data from a single DS0 TDM channel to multiple I-TDM fabric destinations over multiple I-TDM packet flows.

Note that support for true multicast could be added in some future version of this I-TDM specification.

3.11. Fabric Latency / Jitter Buffer Issues

The maximum latency for I-TDM 125us mode data traffic over the fabric is 300 microseconds.

The maximum latency for I-TDM 1ms mode data traffic over the fabric is 900 microseconds.

The combination of the User Application, Source Node, Destination Node and Fabric Switching Elements will enforce these fabric latency requirements.

Note that if the system implements the following:

- Properly assigned bandwidth allocation for I-TDM data (e.g. typically no more than 50% of the bandwidth of any fabric link)
- Highest fabric priority (e.g. VLAN priority) for I-TDM data, lower priorities for I-TDM control and other data packets
- Properly defined endpoint behavior (e.g. properly shaped so no large bursts of TDM traffic occur)

then the required fabric latencies specified above are achievable with most of today's fabric switching devices.

3.12. I-TDM Software Model

Since the main goal of I-TDM is to enable an ecosystem of aTCA boards and/or LAN attached modules, much of the software will be vendor specific. In other words, I-TDM aims to specify the minimum amount of software necessary to enable the ecosystem.

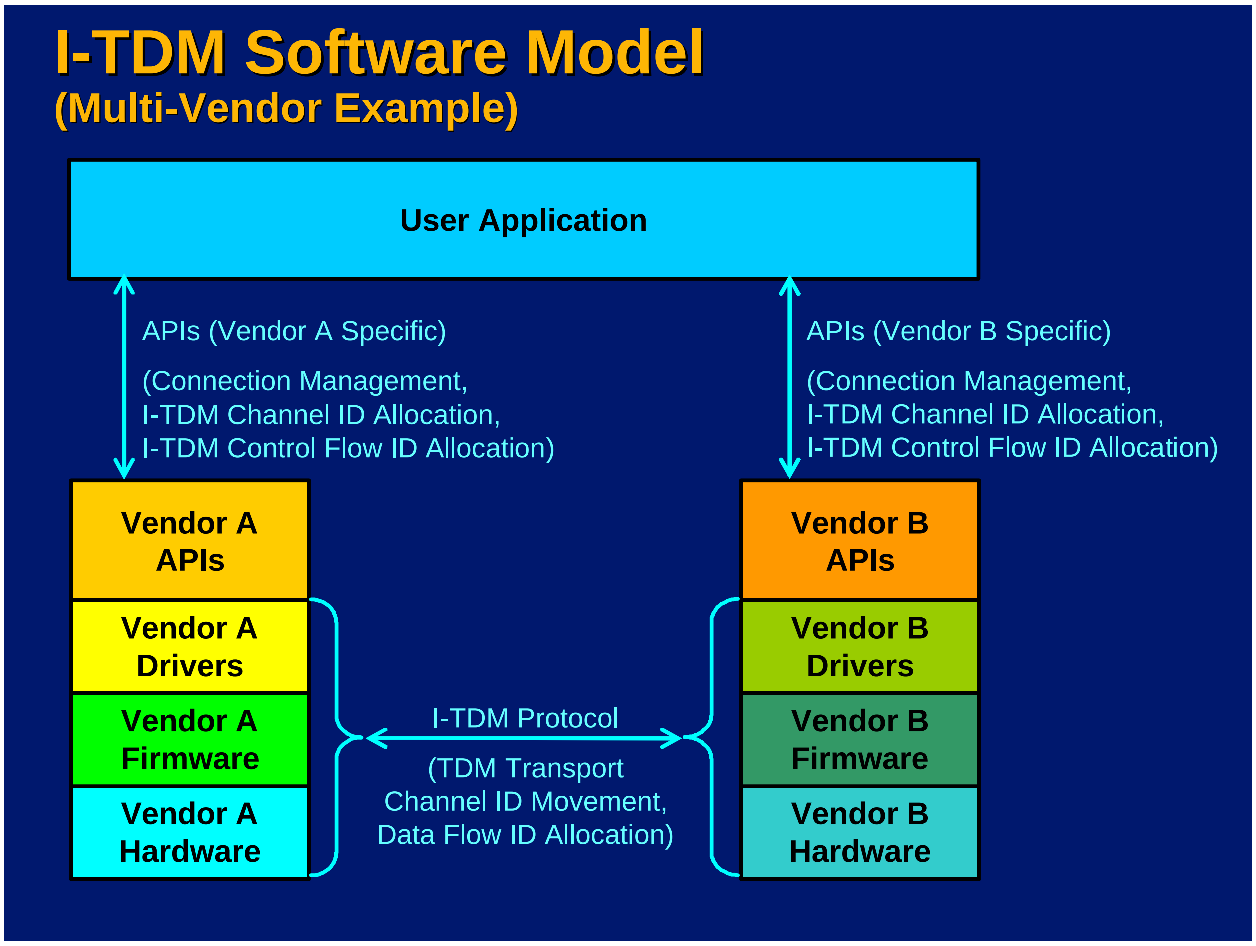


Figure 12: I-TDM Software Model

However, one of the main differentiating features of I-TDM is that the multiplexing of multiple TDM channels into a single packet is hidden from the higher software layers. This scheme allows the lowest layers of control code to completely manage the I-TDM packet flows. This, in turn, enables the User Application to manage individual TDM channel connections using standard call control stacks (e.g. SIP, MGCP, Megaco, etc.).

In order for the lowest layers of control code to manage the I-TDM packet flows, a few standard I-TDM control messages need to be defined. These are detailed in the "I-TDM Control Messages" section below.

3.12.1. I-TDM Connection Procedure

The following list shows an example I-TDM connection establishment sequence.

1. Allocate Destination Port
   - Port includes the Channel ID and Destination LAN Address (e.g. Ethernet MAC address).
   - Destination Port can be communicated using a SIP Session Descriptor Package (SIP SDP).
   - All Ports can be allocated once at initialization and then managed by the User Application.
2. Communicate Destination Port to Source Node Application (e.g. using SIP SDP)
3. Issue request to Start Transmitting at the Source Node
4. Send Flow ID Request to Destination Node (if necessary)
5. Flow ID Response reply back to Source Node (if necessary)
6. Send an ACK message back to the Destination Node Application
7. Start processing the connection at the destination node

This procedure is illustrated below.

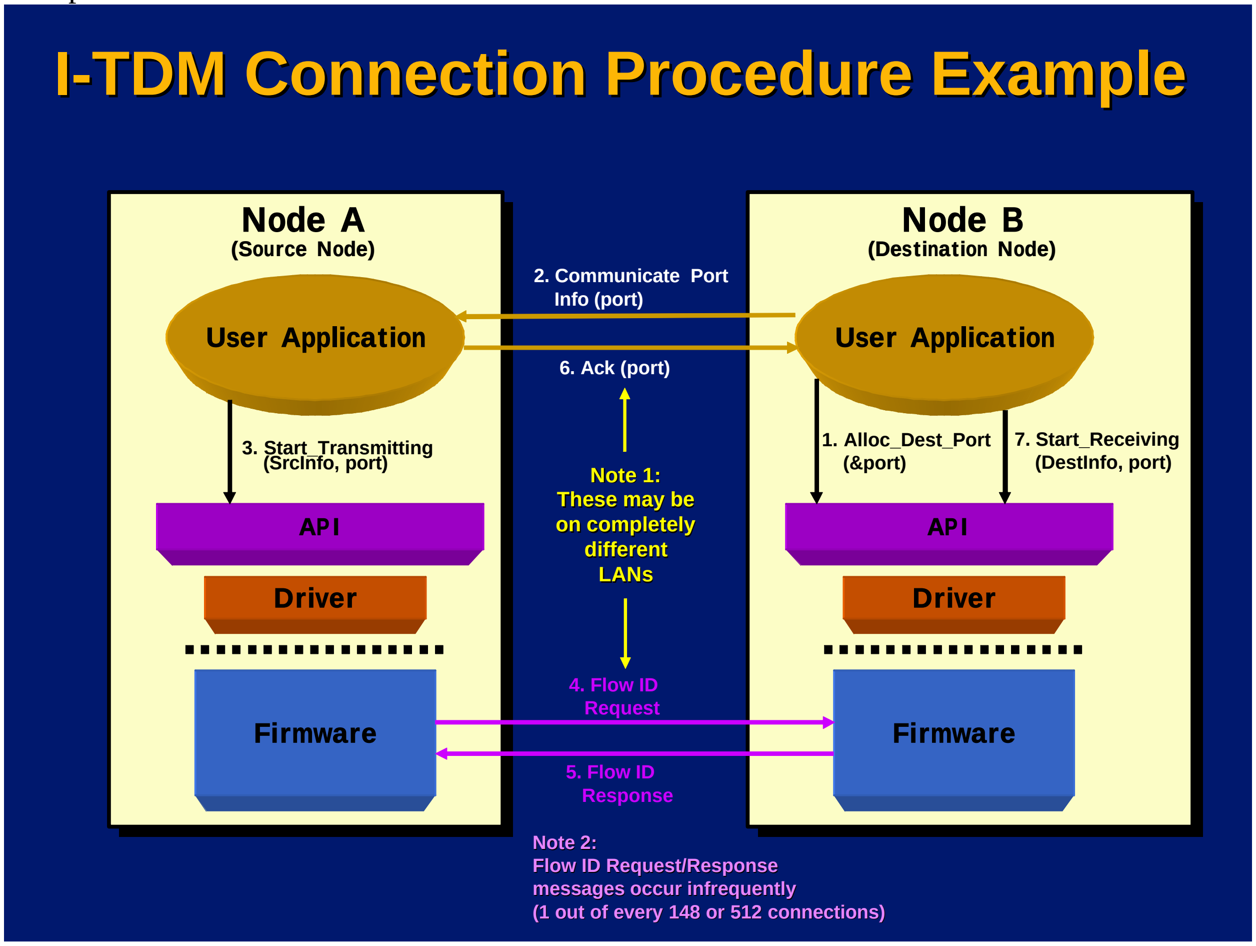

I-TDM Connection Procedure Example
Node A
(Source Node)
User Application
3. Start_Transmitting (SrcInfo, port)
API
Driver
Firmware
Node B
(Destination Node)
User Application
1. Alloc_Dest_Port (&port)
7. Start_Receiving (DestInfo, port)
API
Driver
Firmware
2. Communicate Port Info (port)
6. Ack (port)
Note 1: These may be on completely different LANs
4. Flow ID Request
5. Flow ID Response
Note 2: Flow ID Request/Response messages occur infrequently (1 out of every 148 or 512 connections)

Figure 13: I-TDM Connection Procedure Example

Of all the steps listed above, only steps 4 & 5 are part of the I-TDM control protocol. Everything else can vary from one implementation to another.

In addition, steps 4 & 5 are only invoked when there is no more room left in the current combination of packet flows to allow the new TDM connection. So this type of message interaction typically occurs only once in every 148 or 512 connections (1ms or 125us modes respectively).

3.12.2. Destination Based Packet Flow IDs

The Flow ID is unique within the destination node. For example, let's say there are 4 nodes, A, B, C & D. Nodes A, B & C are all driving TDM connections to Node D. That is, Node D is the destination and Nodes A, B & C are the sources. In this scenario, when Node D receives Flow ID #5, it doesn't look to see where it came from. There will be only one Flow ID #5 coming in to Node D. So the Flow ID can be used directly by the destination node to identify the packet flow.

This enables the destination node to use the I-TDM Flow ID to directly identify a local resource (e.g. an I-TDM jitter queue).

In addition, the destination node may dedicate certain Flow ID numbers to specific types of flows, and this may be done differently for each implementation, as shown in the examples below.

Example Implementation #1		
Flow ID #s	# of Flows	Type of Flow
0	1	I-TDM Control
1 - 127	127	I-TDM 1ms Mode

Example Implementation #2		
Flow ID #s	# of Flows	Type of Flow
0	1	I-TDM Control
1 - 15	15	I-TDM Explicit Multi-Timeslot Mode
16-127	112	Not Used
128 - 255	128	I-TDM 125us Mode
256 - 383	128	I-TDM 1ms Mode

Table 2: Destination Flow ID Number Space Allocation Examples

Note that these are only two of many possible implementations. The point is that there are no restrictions on the way an implementation assigns the destination based Flow ID #s.

For these reasons, the Destination node is responsible for allocating the I-TDM Flow ID number space. The I-TDM Control protocol facilitates this destination based Flow ID allocation, as specified in the sections below.

3.12.3. Half Duplex Channel Connections

All I-TDM channel connections are managed as half-duplex. In other words, if a full duplex connection is required by the application, I-TDM will require 2 half-duplex connections to implement this. This is because all I-TDM Channel IDs are destination based (see Channel ID sub-sections within payload format sections above for details).

3.12.4. Paired Packet Flow IDs

Although I-TDM channel connections are managed as half duplex, the I-TDM Flow IDs are managed as full duplex pairs. This is because the reliability mechanisms associated with many of the I-TDM data modes specified above require an acknowledge command to be returned back on a corresponding paired I-TDM data path.

Note that the I-TDM 1ms mode doesn’t require a corresponding paired I-TDM data path. However, since the vast majority of I-TDM 1ms mode connections are full duplex, and since there are many benefits for having the same I-TDM control protocol work for all I-TDM modes, the I-TDM 1ms mode uses paired Flow IDs as well.

Since the I-TDM packet Flow ID generally corresponds to a resource at the destination node (e.g. a jitter queue), then the Flow IDs for the associated forward and reverse paths will usually be different (e.g. Flow ID from Node A to Node B is probably different than Flow ID from Node B back to Node A). For this reason, the I-TDM endpoint node maintains a table that pairs each incoming Flow ID # with its associated outgoing Flow ID #.

Note that, since channel connections are managed as half duplex, the number of I-TDM channel connections in each direction of a paired set of Flow IDs can be very different. In fact, there will be some cases where an I-TDM Flow ID is established and never used for any channel connection (i.e. if all channel connections are unidirectional).

3.12.5. I-TDM Control Messages

The following 4 messages form the basis of the I-TDM Control Protocol.

- **Allocate Flow ID Request (AFI_REQ)**
- **Allocate Flow ID Response (AFI_RSP)**
- **De-allocate Flow ID Request (DFI_REQ)**
- **De-allocate Flow ID Response (DFI_RSP)**

Allocate Flow ID Request (AFI_REQ)

The **AFI_REQ** message shown in the diagram below is used by the I-TDM source node to request a new Flow ID to be allocated by the destination node. The AFI_REQ also serves as a request to the destination to allocate a paired Flow ID in the reverse direction.

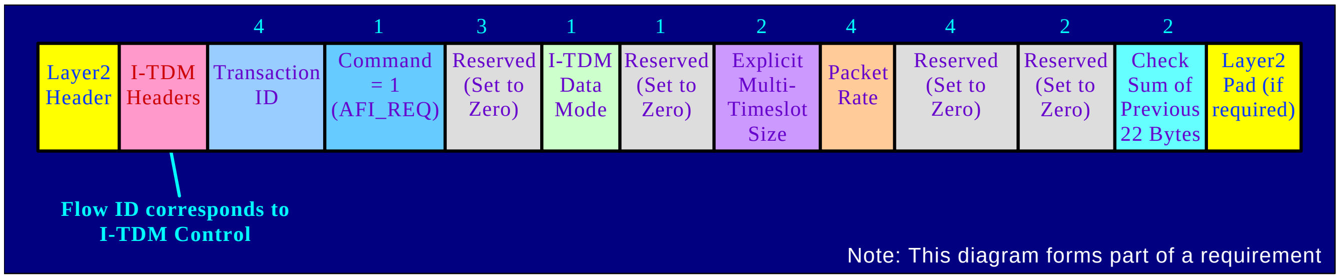


Figure 14: AFI_REQ

Allocate Flow ID Response (AFI_RSP)

The **AFI_RSP** message shown in the diagram below is used by the I-TDM destination node to return a new Flow ID to the I-TDM source node in response to an AFI_REQ, as well as communicate the Transaction ID of the associated paired flow in the reverse direction.

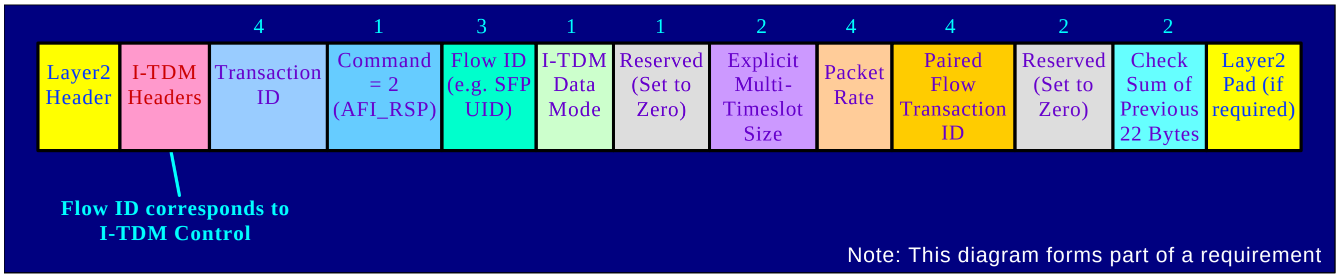


Figure 15: AFI_RSP

De-allocate Flow ID Request (DFI_REQ)

The **DFI_REQ** message shown in the diagram below is used by the I-TDM source node to request the I-TDM destination node to de-allocate the given Flow ID. The DFI_REQ also serves as a request to the destination to de-allocate the paired Flow ID in the reverse direction.

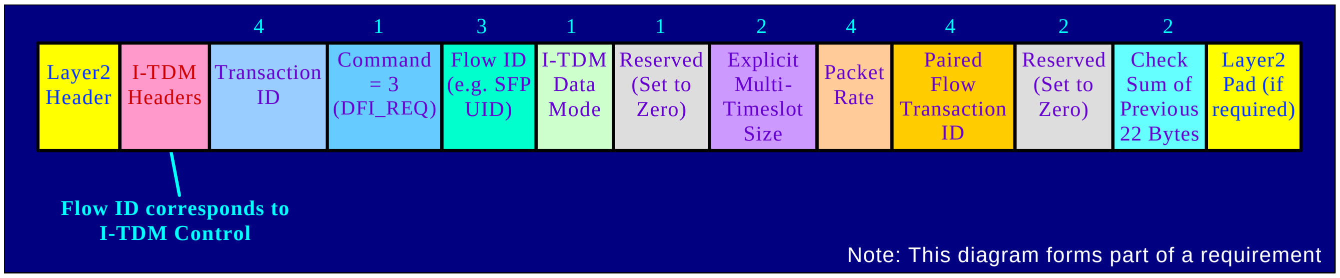


Figure 16: DFI_REQ

De-allocate Flow ID Response (DFI_RSP)

The **DFI_RSP** message shown in the diagram below is used by the I-TDM destination node to acknowledge the Flow ID de-allocation in response to a DFI_REQ back to the source node.

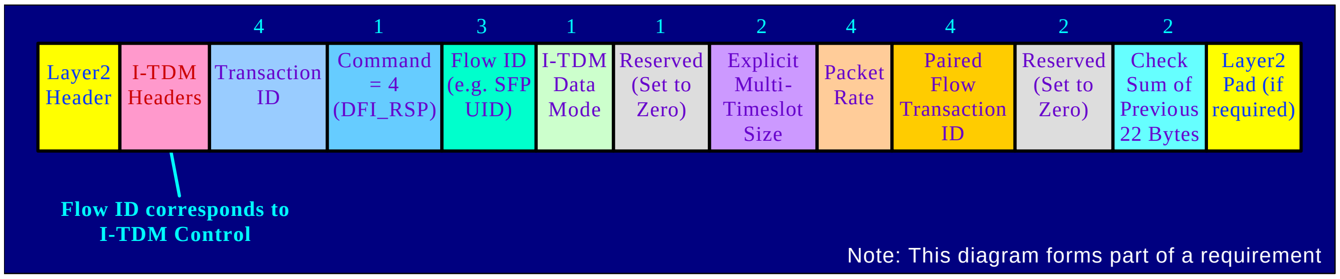


Figure 17: DFI_RSP

Refer to the full specification of SFP.1 for details on the I-TDM Control Messages.

4. APPENDIX

This section adds additional explanatory information.

4.1. IEEE 754 Single Precision Floating Point Format

I-TDM packet repetition rate fields are given in IEEE 754 single precision floating point format. The intent of using this format is to allow future versions of the I-TDM protocol to define specific new packet rates in a pre-defined and standard way.

The currently defined packet rates are:

I-TDM Data Mode	Packets Per Second	IEEE 754 Floating Point Value
I-TDM 1ms Data Mode	1000	0x447A0000
I-TDM 125us Data Mode	8000	0x45FA0000
I-TDM Explicit Multi-timeslot Mode	8000	0x45FA0000
I-TDM T1 CAS Mode	333.333333	0x43A6AAAB
I-TDM E1 CAS Mode	500	0x43FA0000

Table 3: Packet Rate Values

Note that there is no implied requirement to support a wide range of packet rates either now or in the future. Rather, it is envisioned that a typical implementation of a future I-TDM protocol version would have only a limited number of specific packet rates, and that these rates would be hard coded as a specific set of constants. This means there is no need for software to process in IEEE 754 floating point numbers, either now or in the future (i.e. floating point processor not required).

Since it is impossible to know at this time what the future set of specific packet rates will be, it was decided to use a flexible standard way to define this set of constant values, hence the choice of the IEEE 754 single precision floating point format.

To ease the generation of this set of constant values, a number of sites on the Internet offer calculators that convert decimal numbers to IEEE 754 single precision floating point format, just type **"IEEE 754 floating point conversion"** in your favorite search engine.

4.2. TRADE-OFF ANALYSIS – VOIP vs. I-TDM

4.2.1. Latency

The delivery protocol mechanism needs low latency. For this reason, an off-the-shelf VoIP stack for TDM transport would probably be configured for 5ms G.711 VoIP packets. Note that the 5ms is not the delivery latency. To calculate latency, the packet unit (e.g. 5ms) is multiplied by a factor of 3-5 to account for minimal buffering and jitter queues. This produces latency in the range of 15-25 ms. By contrast; I-TDM only takes 3-5 ms for delivery latency.

4.2.2. Packet Header Size

At first glance, it would appear that the sheer number of header bytes in a 5ms RTP/UDP/IP packet is an issue. However, at the densities we are considering (2000 full duplex TDM connections per node), Gigabit Ethernet should not have a problem with this. Additionally, the processing MIPs associated with adding and removing bytes to or from a packet is minimal. The real MIPs issues are associated with the implied processing of the fields within these headers (i.e. the IP and RTP Stacks). The size of the headers in a VoIP packet is not a big issue.

4.2.3. Packets Per Second

Consider a Host Media Processing Server that processes up to 50 ports. At this density:

Packet Type	Packet Time Unit	Packets Per Second Per Channel	Packets Required for 50 Channels	**Total Packets Per Second**
VoIP	5ms	200	50	**10,000**
I-TDM	1ms	1000	1	**1000**

Table 4: VoIP vs. I-TDM Packets per Second Example

As a data point, Microsoft once advertised the NT RRAS Server software at 40,000 Packets per Second on a 500MHz processor. Note that this is Routing software that does not process the TCP/UDP checksums or session layers. Since all VoIP packets are end-pointed, it's probably safe to assume that handling VoIP packets will require twice the MIPs of NT RRAS. With this in mind, an RTP/UDP/IP endpoint at 10,000 packets per second consumes around 50% of a 500MHz Pentium processor, leaving only 50% of the platform available for Voice Processing.

As the processor speed is increased, the number of voice ports desired tends to increase linearly, so the problem tends to scale. So, for example, if a 5 GHz processor were used for 500 TDM channels, the RTP/UDP/IP processing alone would probably consume around 50% of a 5 GHz processor.

If the level of processing were the same for VoIP and I-TDM packets, then I-TDM would require only 5% of the processor (i.e. I-TDM is 10-30 times lower packet rate). This would leave around 95% of the processor available for Voice Processing. However, I-TDM actually requires less MIPs per packet, as shown below.

4.2.4. Processing per Packet

The Option of using I-TDM on a Host Media Processing Server Box would require the use of a separate I-TDM over Ethernet (I-TDMoE) Driver (e.g. Windows NDIS or Linux Network Driver). The new Network Driver would sit along-side the other Network drivers (e.g. IP, IPX, PPP, etc). The Operating System determines the specific Network Driver used for each packet using the EtherType in the MAC header. Each Network Driver registers which EtherTypes it handles. This is a standard O.S. feature for developing Network Drivers. Since we have a unique EtherType for SFP/I-TDM, this approach fits very well.

Since I-TDM has no IP, UDP, or RTP headers, the processing of the packets in the I-TDMoE Network Driver is greatly simplified. The main Network Driver function required is to distribute the various TDM Segments to their proper Circular Queues. Since all fields are 64-bit aligned, the movement of data to the circular queues is greatly accelerated. The circular Queues may then be Direct Mapped in memory to the User Application. So the User API for voice data is minimal (i.e. just read a Queue in memory). There is no UDP/IP Session layer. There is no RTP layer. Just read memory. This is similar to data streaming mechanisms typically used in DSP implementations.