1. Introduction

1.1. Purpose

This white paper gives a high level description of the I-TDM protocol.

1.2. 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 backplanes 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.

1.3. What is I-TDM?

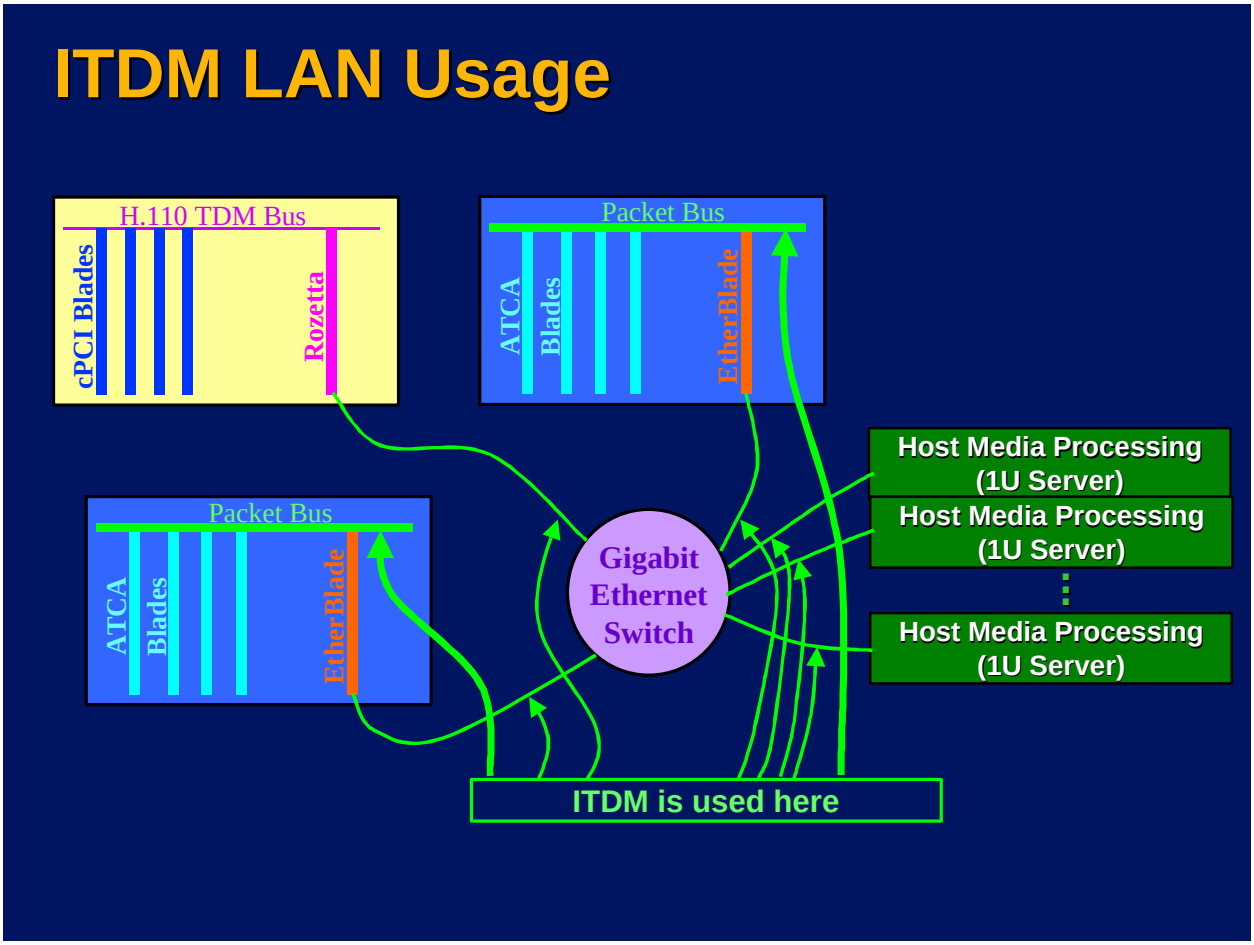
I-TDM stands for "Internal TDM" protocol. I-TDM is an aggregated voice over packet protocol that is optimized for Voice LANs and Packet Backplanes (i.e. for connecting telephony equipment within the same chassis, room or building). I-TDM does not aim to be an end user protocol. It typically exists only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

I-TDM Features:

- Low Latency (1 millisecond packets)
- Up to 148 TDM channels per packet
- Very low processing overhead (high density)
- 8-byte alignment (for 64-bit processors)
- Very easy to move TDM channels from one aggregate packet flow to another (grooming)

Contents

1.4. I-TDM LAN Usage

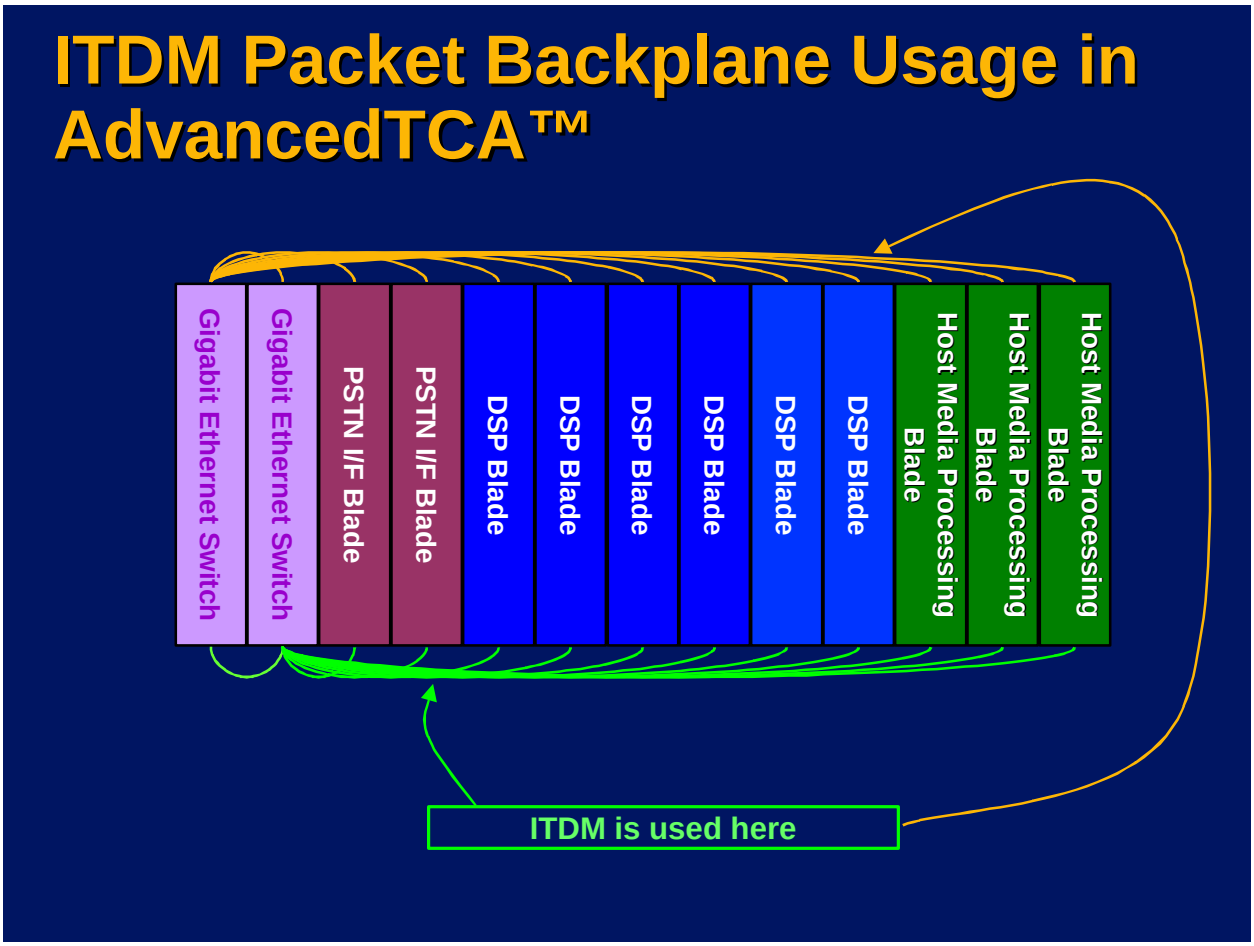


Media Processing such as Speech Recognition, Voice Conferencing, Voice Mail, etc. is being increasingly implemented using off-the-shelf Host Platforms (e.g. 1U PCs with no special add in cards). At low densities, RTP over IP works well for this. But when the density scales up to hundreds or thousands of voice channels per Host, RTP is no longer viable. The processing overhead of RTP is too high. Also, even for some lower density Host Media Processing applications, the latency of RTP 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 Computer 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.

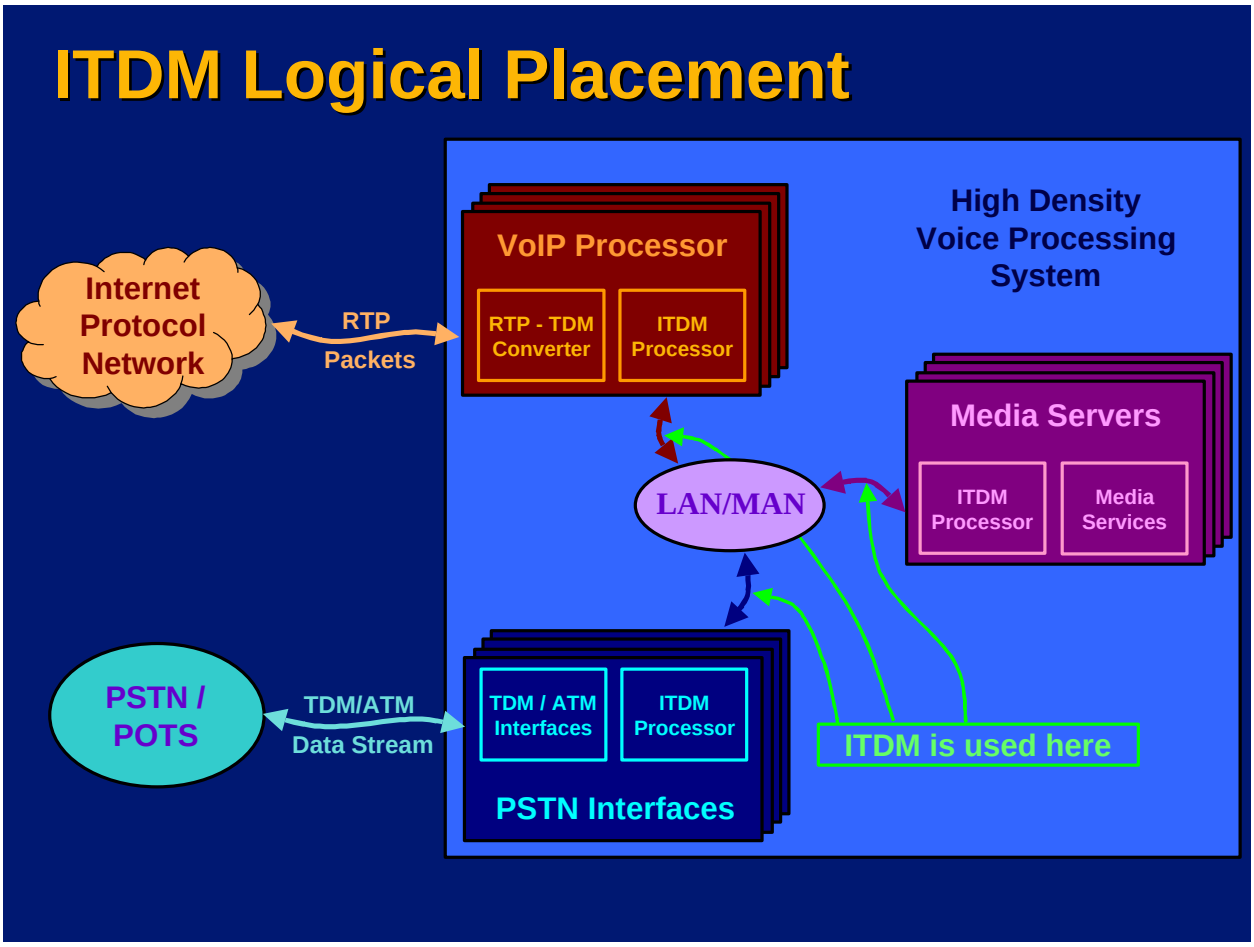
1.5. I-TDM Packet Backplane Usage

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.



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.



I-TDM does not aim to be an end user protocol. It typically exists only within the logical confines of the Voice Processing System. Note that various types of systems may be configured using the diagram above.

For example, a VoIP Gateway may be configured using just PSTN Interfaces and VoIP Processors (i.e. no Media Servers necessary).

In another example, a PBX or PSTN Switch may be configured using multiple PSTN interface modules connected via I-TDM (i.e. no Media Servers or VoIP Processors necessary).

In a third example, a VoIP Media Server may be configured using VoIP Processors and Media Servers (i.e. no PSTN interfaces necessary).

2. Driving Requirements for I-TDM

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

1. Support voice call switching within a Local Exchange Carrier environment

- Total latency must be less than around 10ms end to end (otherwise local echo becomes noticeable)
- This corresponds to 5ms in each direction
- Typical TDM to packet conversion buffers 1-2 packets
- Typical packet to TDM conversion buffers 2-3 packets
- Derived Requirement: Use 1ms packet rate

2. Support High Density

- Derived Requirement: Multiple TDM connections per packet (aggregated format)
- Derived Requirement: 8-byte alignment (for 64-bit processors)

3. Easy to control in High Call Rate applications

- Derived Requirement: Ability to easily move TDM connections between aggregate TDM packet flows (i.e. no end-to-end message required for this operation).

These requirements 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. Voice Switching within the Local Exchange Carrier

An example of this is when a customer uses debit-card to make a local call. 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.

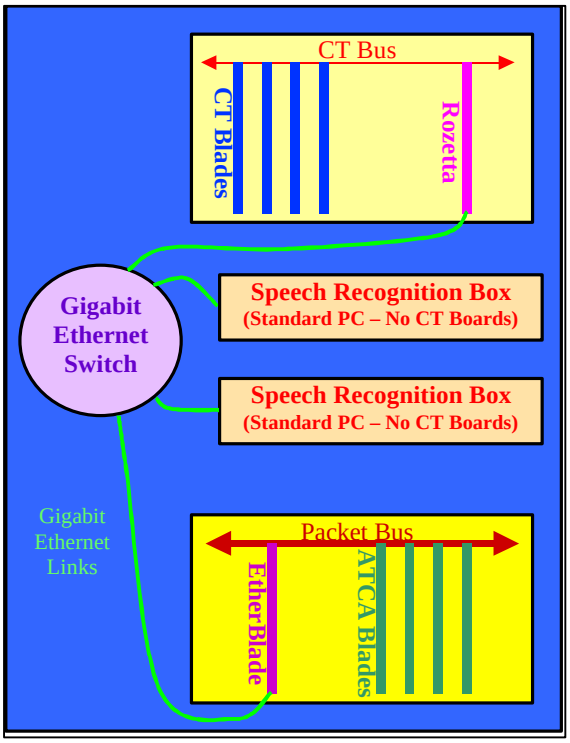
Another example of Voice Switching within the Local Exchange Carrier is when a voice over packet infrastructure is used to build the Local Exchange Carrier switch itself.

For these types of applications, the total end-to-end latency must be less than around 10ms (otherwise local echo becomes noticeable). End-to-end latency refers to the amount of time it takes to get from one phone to the other phone and back. This corresponds to 5ms in each direction. TDM to packet conversion implementations typically buffer 1-2 packets. Packet to TDM conversion implementations typically buffer 2-3 packets. This totals 3-5 packet buffers with no more than a 5ms latency in each direction.

Derived Requirement: Use 1ms packet rate

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.



In order to meet Barge-In requirements for Speech Recognition, the delivery protocol mechanism must have low latency. For this reason, an RTP 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 RTP) 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.

2.2. High Density

1. Aggregation. I-TDM allows many TDM channels in a single packet. This provides efficiency both in terms of Fabric bandwidth utilization and lower packets per second.

2. 8-byte alignment. Processors are being used increasingly in Voice Networks. Most modern processors have 64-bit data paths (e.g. Pentium*, Sparc*, PowerPC*, Network Processors, etc). These processors work much more efficiently when fields are aligned on 8-byte boundaries.

2.3. Grooming

I-TDM allows TDM timeslot connections to be moved between I-TDM aggregate 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. 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.

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 aggregation 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 aggregate voice 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.

2.4. I-TDM vs. Other Protocols

2.4.1. I-TDM vs. RTP

- RTP is designed for Voice over the Internet
- I-TDM is optimized for Voice over LAN
- Major Differences between I-TDM and RTP
    - I-TDM does not require the Internet Protocol (no IP Stack)
    - I-TDM allows up to 148 channels per packet (most RTP stacks only allow 1 channel per packet).
    - I-TDM is much lower latency (1ms vs. RTP typical minimum of 5 to 10ms)
    - I-TDM requires much less processing overhead

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

2.4.2. I-TDM vs. TDMoIP

- TDMoIP is 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)
    - I-TDM uses 64-bit (1ms) TDM channel transfer units. TDMoIP uses 8-bit (125us) TDM channel transfer units. The 8-bit transfer units make it difficult to endpoint large numbers of TDMoIP channels in software (e.g. Pentium*, Sparc*, PowerPC*, Network Processors, etc.) unless there is a TDMoIP hardware assist circuit.
    - I-TDM requires much less processing overhead for changing connections

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

2.4.3. I-TDM vs. VoMPLS, Multiplexed RTP, etc.

Recently, several other aggregated 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.

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 designed for WAN voice over packet applications where low latency and high call rates aren't required.

2.5. TDM Clocking issues

Most of the other protocols mentioned above (e.g. RTP, 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 stream 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.

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.

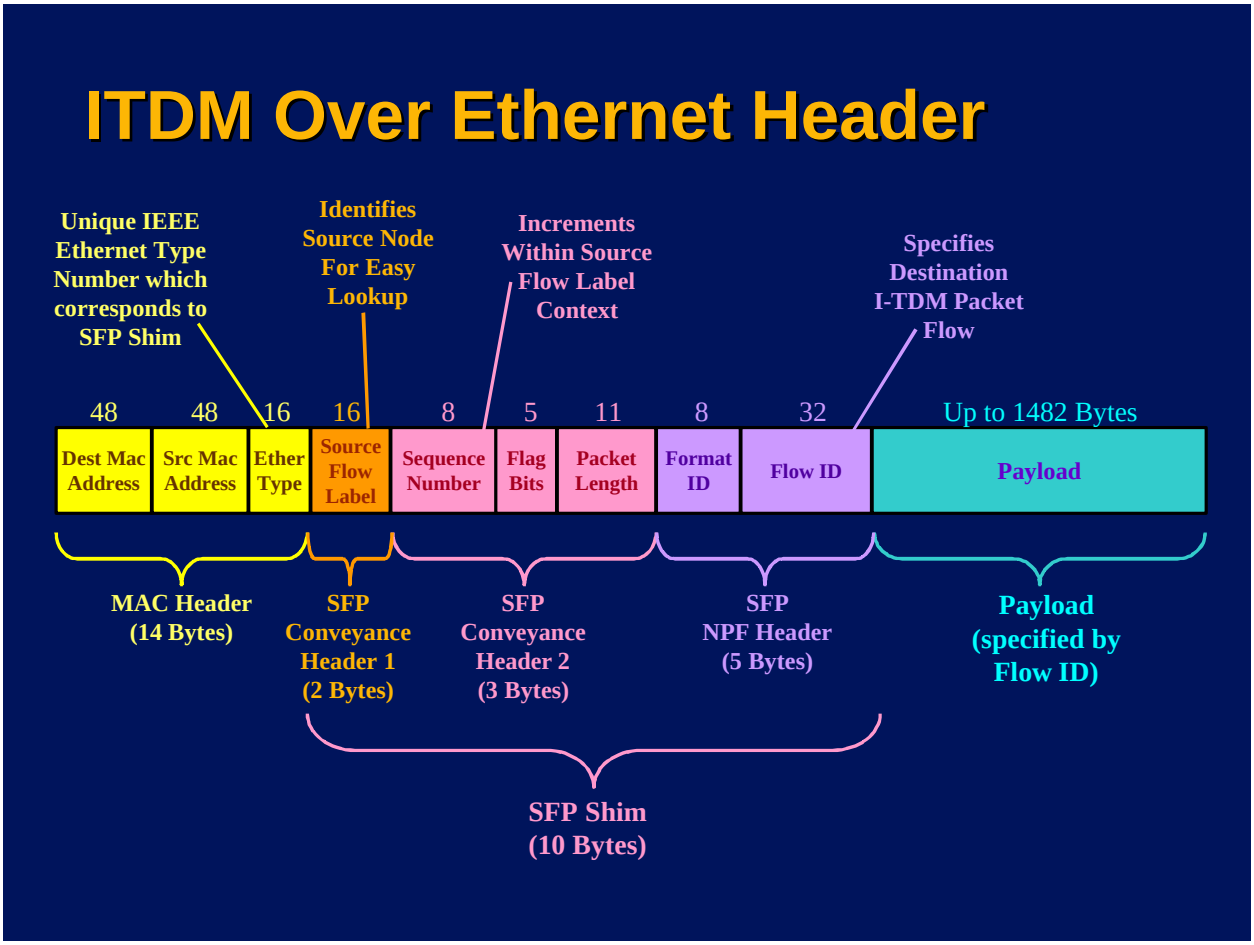
Between 2 chassis, T1/E1 wires are used. In a central office environment, these T1/E1 wires come from a central timing source (i.e. a "clock box"). For enterprise solutions, all you need is to lock the clock from one chassis to the next (e.g. fan-out or daisy-chain) using T1/E1 wires. The ATCA board 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 has added a sequence number. The combination of low latency packet arrival coupled with a sequence number to detect lost frames satisfies HMP timing requirements.

3. Protocol Overview

3.1. I-TDM Headers

I-TDM can easily be adapted to various packet formats (e.g. Ethernet, Infiniband, PCI Express*, MPLS, CSIX, etc). An example using Ethernet is given below.



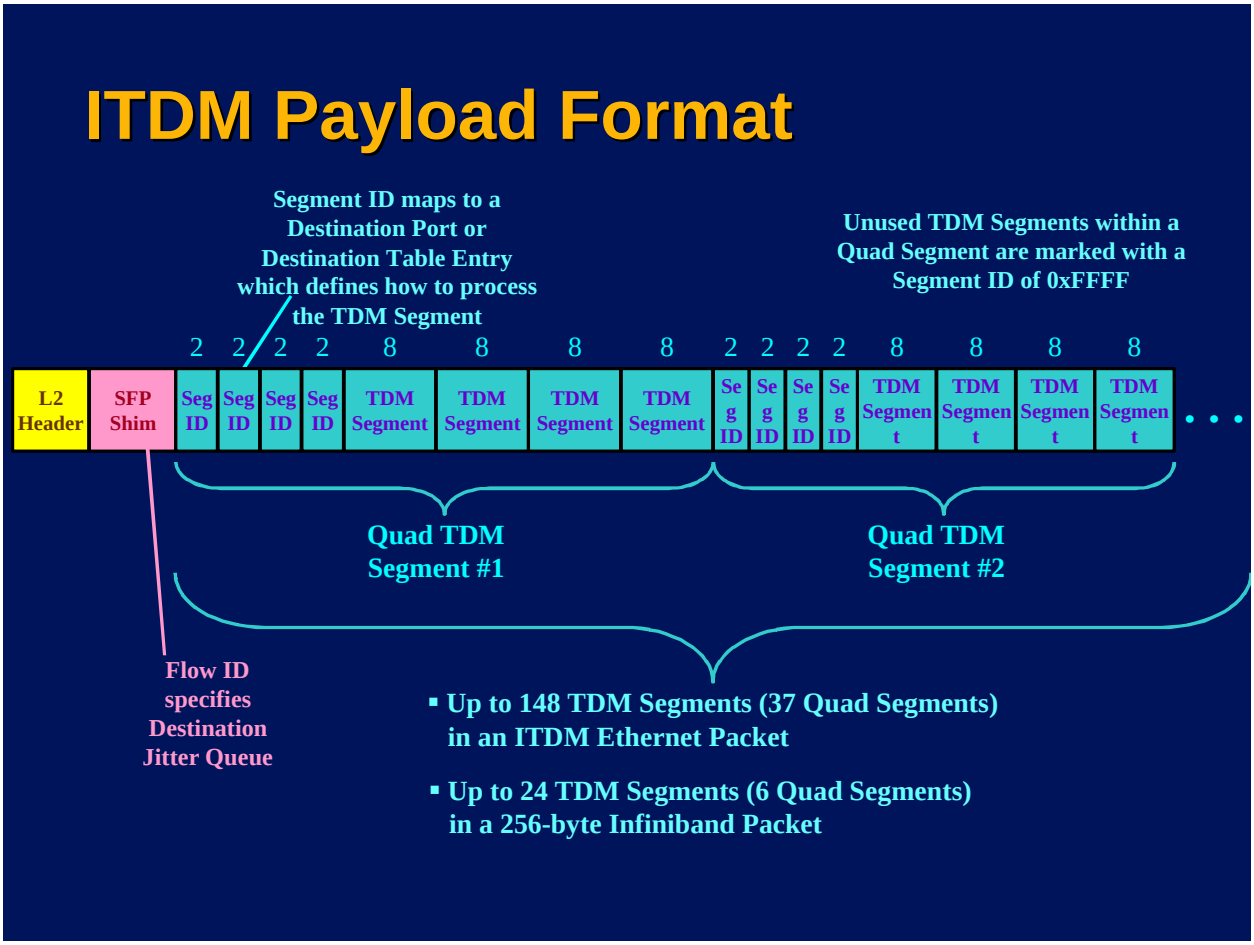
Note that I-TDM typically doesn't use Internet Protocol headers. All you really need is a Flow ID and Sequence Number.

One way to provide this is by using the System Fabric Plane (SFP) Shim. 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"). More information on the SFP shim is available in a separate white paper. Note that the Shim is sized so that the beginning of the payload starts on an 8-byte boundary. This increases performance when terminating I-TDM flows on processors with 64-bit data paths (e.g. Pentium*, Sparc*, PowerPC*, Network Processors, etc).

The Flow ID contains the destination packet flow that corresponds to a Jitter Queue at the Destination Node. The Flow ID also identifies the packet type (e.g. I-TDM Control or I-TDM Data).

The generation of the Sequence Number is mandatory. Checking the Sequence # is optional. For example, if a packet backplane contains a timing reference signal (e.g. 8K REF), then lost packets can be detected without ever reading the Sequence #. But if the destination node is a Host Media Processing platform, then the Sequence # would be the only way to detect lost packets.

3.2. I-TDM Payload Format



3.2.1. TDM Segment

This is 8 samples of PCM voice/data at an 8 KHz rate. Each sample is one byte. This represents 1 millisecond of TDM voice/data.

3.2.2. Segment ID

Identifies the destination TDM channel number of the corresponding TDM Segment. Note that this is not just identifying the TDM channel number within the packet, but rather identifies the TDM channel number within the destination endpoint node.

Note that, since the Segment ID completely identifies the TDM channel at the destination node, there is no need to translate this number at the destination. Additionally, if multiple packet flows are required to connect two endpoints, the TDM channels may be moved around within 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 may even switch cars in the middle of the ride. Similarly, if multiple packet flows exist between source and destination blades, a given Segment ID may 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 Segment 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 may 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 aggregation 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.

3.2.3. Quad TDM Segment

This is a structure that holds four TDM segments and associated Segment ID's while maintaining 8-byte alignment. It is the minimum unit of transfer. Specifically, it consists of 4 Segment ID's, followed by 4 TDM Segments.

Inactive TDM Segments are marked with the special Segment ID 0xFFFF.

3.3. I-TDM Control Protocol

One of the main differentiating features of I-TDM is that the aggregation of multiple TDM channels into a single packet is hidden from the higher layers. This scheme allows the lowest layers of control code to completely manage the I-TDM packet flows. The higher layers of control code only need to deal with channel connections. This, in turn, allows standard connection protocols (e.g. SIP) to be used for I-TDM.

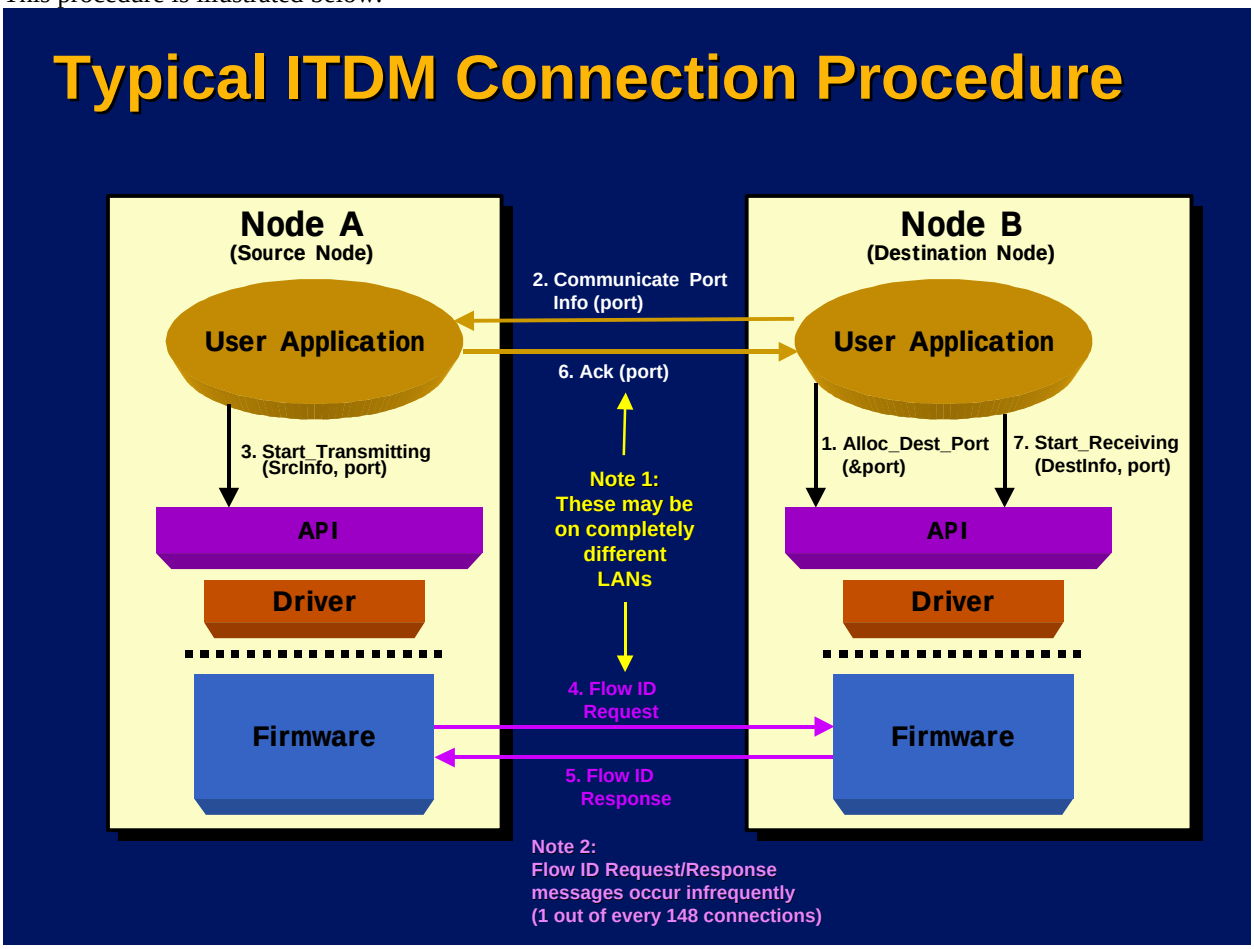
However, in order for the lowest layers of control code to manage the I-TDM packet flows, a few standard I-TDM control messages must be defined.

3.3.1. I-TDM Connection Procedure

The following list shows a typical I-TDM connection establishment sequence.

1. Allocate Destination Port (Note: this may be done once at initialization)
2. Communicate Destination Port Information to Source Node Application (e.g. using SIP session descriptor)
3. Start Transmitting on 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.



The Port Information includes the Destination Segment ID and LAN Address. This Port Information may be communicated between User Applications using a SIP session descriptor.

Of all the steps listed above, only steps 4 & 5 are part of the I-TDM control protocol. Everything else may 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 connections.

Also note that 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.

3.3.2. I-TDM Control Messages

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

- **Allocate Flow ID Request (AFI_REQ)**
    - The **AFI_REQ** is used by the I-TDM source to request a new Flow ID from the destination node.
- **Allocate Flow ID Response (AFI_RSP)**
    - The **AFI_RSP** is used by the I-TDM destination to return a new Flow ID to the I-TDM source in response to an AFI_REQ.
- **De-allocate Flow ID Request (DFI_REQ)**
    - The **DFI_REQ** is used by the I-TDM source to request the I-TDM destination to de-allocate the Flow ID.
- **De-allocate Flow ID Response (DFI_RSP)**
    - The **DFI_RSP** acknowledges the Flow ID de-allocation

3.4. I-TDM LAN Configurations

I-TDM requires low latency packet transmission (i.e. all I-TDM packet must arrive within 1ms).

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 has it's own priority (e.g. VLAN priority, Virtual Lane, etc).

4. Trade-Off Analysis – RTP vs. I-TDM

4.1. Latency

The delivery protocol mechanism must have low latency. For this reason, an off-the-shelf RTP stack for TDM transport would probably be configured for 5ms G.711 RTP packets. 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. By contrast; I-TDM only takes 3-5 ms for delivery latency.

4.2. 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 an RTP packet is not a big issue.

4.3. Packets Per Second

Consider a Host Media Processing Server that processes up to 100 ports. At this density:

Packet Type	Packet Time Unit	Packets Per Second Per Channel	Packets Required for 50 Channels	**Total Packets Per Second**
RTP	5ms	200	100	**20,000**
I-TDM	1ms	1000	1	**1000**

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.

With this in mind, an RTP/UDP/IP endpoint at 20,000 packets per second consumes at least 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. If the level of processing were the same for RTP and I-TDM packets, then I-TDM would require only 5% of the processor leaving around 95% of the platform available for Voice Processing. However, I-TDM actually requires less MIPs per packet, as shown below.

4.4. 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 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 thick UDP/IP Session layer. There is no RTP layer. Just read memory. This is similar to data streaming mechanisms typically used in DSP implementations.