



# Technology Backgrounder

## TDMoIP Environment

### Introduction to TDMoIP

TDMoIP is a pseudowire technology for emulating TDM circuits over IP networks. TDMoIP transport capabilities include E1, fractional E1, T1 and fractional T1 data streams, with and without channel-associated signaling. Voice can also be transported.

The TDMoIP technology enables transmitting the continuous data stream generated by TDM equipment as a stream of discrete packets, having a structure suitable for transmission over packet-switched networks. The data stream consists of individual timeslots retrieved from the E1 or T1 frame structure; each independent set of timeslots is referred to as a **bundle**. The TDMoIP packets are encapsulated in UDP packets and transported using IP (this is referred to as UDP over IP, or UDP/IP).

In many applications, the TDMoIP technology must transfer not only the data stream, but also the original timing (clock rate).

### User Datagram Protocol

The User Datagram Protocol (UDP) is a transport protocol (that is, a Layer 4 protocol in the OSI model) that provides an unreliable connectionless delivery service. UDP uses IP to transport messages among host machines.

One of the main characteristics of UDP is support for multiple destinations within a given host machine, which means that it enables the multiplexing of traffic from multiple sources, for example, different software processes running on the host machine, over the same IP link. The term used to refer to a UDP source (or destination) is **port**.

UDP messages are called **user datagrams**. A datagram consists of two parts:

- **User datagram header.** The UDP datagram header is divided into four 16-bit fields that specify the port from which them was sent, the port to which the message is destined, the message length, and a UDP checksum.

Figure 1 shows the structure of a UDP datagram header.

0	4	8	12	16	20	24	28	31
Source UDP Port (2 Bytes)				Destination UDP Port (2 Bytes)				
Datagram Length (2 Bytes)				UDP Checksum (2 Bytes)				

Figure 1. Structure of UDP Datagram Header

- The source and destination fields carry the 16-bit UDP protocol port numbers used to demultiplex datagrams.  
In UDP, the source port is optional: when used, it specifies the port to which replies should be sent; if not used, it should be zero.
- The length field carries the number of bytes in the complete UDP datagram (UDP header and user data).
- The UDP checksum is optional; 0 in the checksum field means that the checksum has not been computed and the field must be ignored at the receive side.
- **User datagram data field.** The datagram data field can include up to 65527 bytes.

### Structure of TDMoIP Packets

Figure 2 shows the structure of a basic TDMoIP packet. The packet includes the standard IP packet header, followed by the UDP datagram carrying the payload, which is encapsulated in the IP packet. However, the UDP datagram header is slightly modified relative to the standard header:

- The UDP source port field is divided into two sections:
  - Three bits that identify the TDMoIP protocol version
  - The source port number field, comprising 13 bits, is used to identify the TDMoIP bundle number. The bundle number serves as a pseudowire label. The available range of pseudowire labels is 1 to 8063; 0 is not allowed, the range of ports 8064 to 8190 is reserved, and 8191 is used for OAM control messages.
- The destination UDP port number is 085E.
- The checksum is 0 (not calculated).

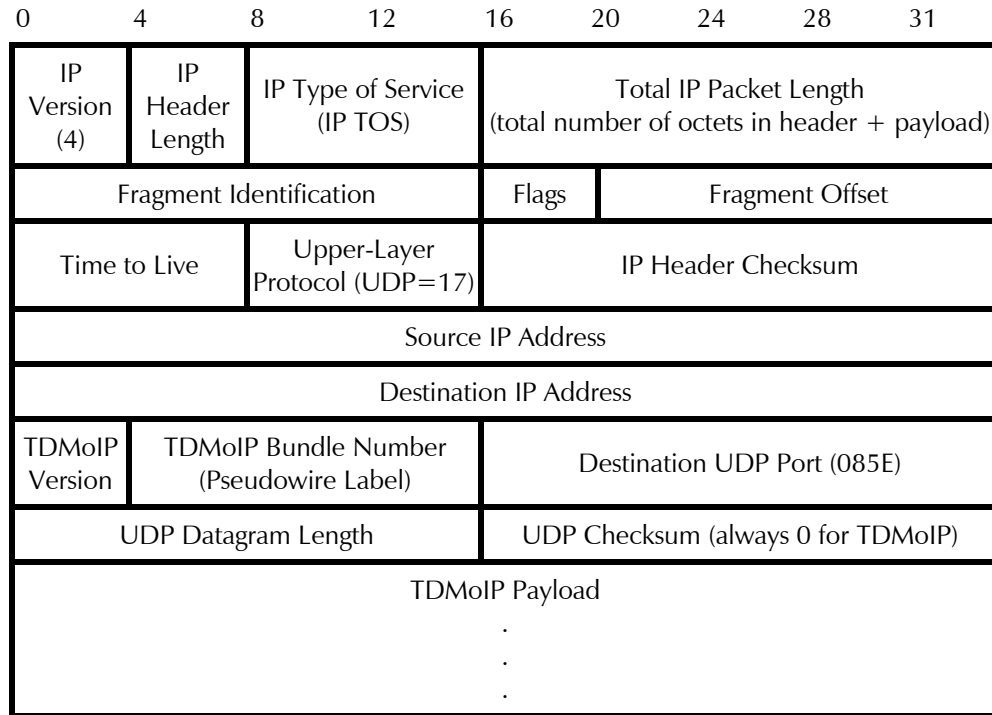


Figure 2. Structure of Basic TDMoIP Packets

The TDMoIP payload section of the packet starts with a control word having the structure shown in [Figure 3](#).

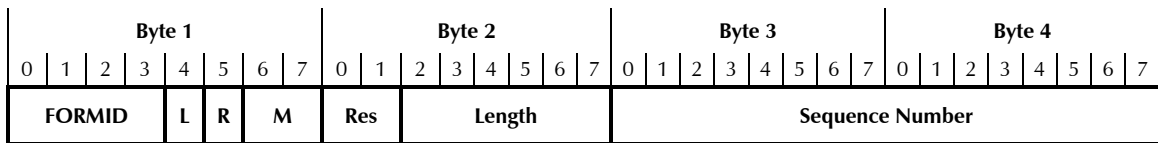


Figure 3. TDMoIP Control Word Structure

**FORMID** Format identifier, comprises 4 bits, used to specify the payload format:

- 0000 — unspecified payload format
- 1100 — AAL1 unstructured payload
- 1101 — AAL1 structured payload
- 1110 — AAL1 structured with CAS payload
- 1001 — AAL2 payload
- 1111 — HDLC payload

**L** Local TDM failure indicator: can be set to 1 to indicate physical layer loss of signal at the sending gateway, that should trigger AIS generation towards the destination.

<b>R</b>	Remote PSN failure indicator: can be set to 1 to indicate that the sending gateway does not receive packets from the PSN, for example, because of a failure of the reverse direction of the bidirectional connection, as a result of congestion or other network-related fault. May be used to generate RDI.
<b>M</b>	Defect modifier indicator (optional).
<b>RES</b>	Reserved.
<b>Length</b>	6-bit field used to indicate the length of the TDMoIP packet, in case padding is needed to meet the minimum transmission unit requirements of the PSN. Must always be used when the total packet length is less than 64 bytes.
<b>Sequence Number</b>	The TDMoIP sequence number (16 bits) provides the common pseudowire sequencing function, and enables detection of lost and misordered packets.

## TDMoIP Packetizing Process

This section describes the process used to build TDMoIP packets.

A simplified diagram of this process, which identifies the main steps of the process, is shown in [Figure 4](#). The packetizing process comprises the following main steps:

1. Splitting the continuous TDM data stream into discrete slices of appropriate size. The slice size is always an integer number of bytes. For example, in [Figure 4](#), the number of TDM bytes per slice, **K**, is 2.
2. Adding the overhead necessary to transmit each slice over the packet network and enable reaching the desired destination. Basically, this process includes the following steps:
  1. Inserting the TDM bytes into the payload field of a UDP packet, and adding the overhead data needed to build a UDP packet.
  2. Inserting the UDP packet into the payload field of an IP packet, and adding the overhead data needed to build an IP packet for transmission to the desired IP destination.
  3. Inserting the IP packet into the payload field of an Ethernet frame, and adding the MAC overhead needed to build an Ethernet frame for transmission to the destination MAC address (the MAC address needed to reach the desired IP destination is determined using the ARP protocol). For example, in [Figure 4](#) the resultant overhead comprises a total of 54 bytes.

**Note** *The actual overhead depends on several factors, one of them being the encoding method used to transmit CAS information. [Figure 4](#) also ignores the minimum interpacket gap, which further increases the overhead.*

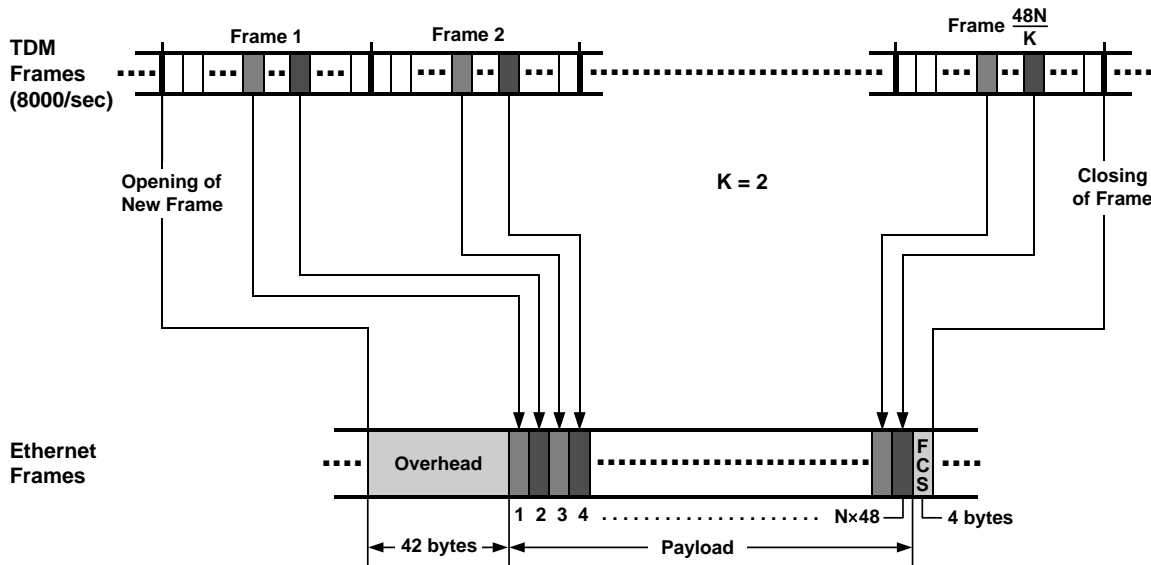


Figure 4. Building an Ethernet Frame with TDMoIP Payload

The receiving end performs the steps necessary to reconstruct the original TDM data stream from the received frames, in the “best” possible way, where “best” means meeting a set of criteria that describe a compromise among several conflicting requirements. A simplified diagram of this process, which identifies the main steps of the process, is shown in

Figure 5.

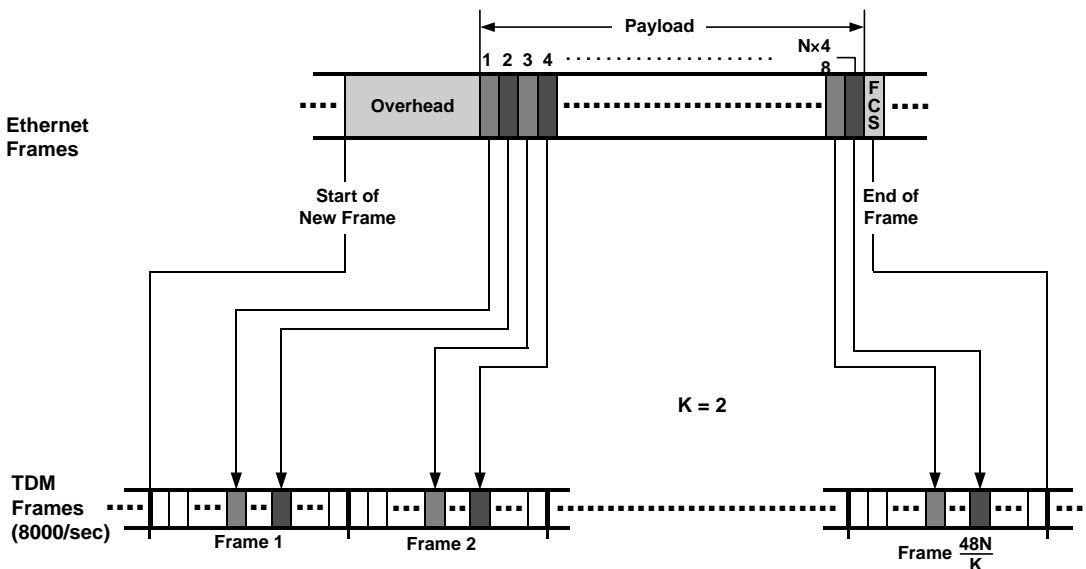


Figure 5. Retrieving the Payload from an Ethernet Frame with TDMoIP Payload

The number of TDM bytes inserted in each Ethernet frame sent to the network, which is actually the size of the UDP packet payload field, affects two important performance parameters:

- Bandwidth utilization: because of the relatively short TDMoIP payload, the bandwidth utilization efficiency depends on the overhead that must be transmitted to the LAN in order to support the transmission of a certain amount of payload.
- Packetizing delay and the associated delay variance. Bandwidth utilization efficiency increases when using a large payload size per frame. However, there are additional aspects that must be considered when selecting the size of the TDMoIP payload:
  - **Filling time:** the filling time, which is the time needed to load the payload into an Ethernet frame, increases in direct proportion to the number of bytes in the TDMoIP payload. This is particularly significant for bundles with few timeslots; for example, a voice channel could be carried by a single-timeslot bundle. Considering the nominal filling rate (approximately one byte every 0.125 msec), the time needed to fill a single-timeslot bundle is as follows:
    - At 48 TDM bytes per frame: 5.5 msec with CAS support and 5.9 msec without CAS support
    - At 384 TDM bytes per frame: 44 msec with CAS support and 47 msec without CAS support.

Therefore, before considering any other delays encountered along the end-to-end transmission path, the round-trip (or echo) delay for the voice channel example presented above is 92 msec at 384 TDM bytes per frame (including the additional intrinsic delay of module – see below).

Such long delays may also cause time-out in certain data transmission protocols.

- **Intrinsic jitter:** the transmission of packets to the network is performed at nominally equal intervals (usually, the interval is 1 msec). This means that every 1 msec the TDMoIP packet processor sends to the network (through the appropriate Ethernet interface) all the frames ready for transmission. As a result, the actual payload transmission intervals vary in an apparently random way whose peak value depends on the bundle size, an effect called **delay variance** (or jitter).

For example, a bundle with 6 timeslots will fill a 48-byte payload field of an Ethernet frame every 1 msec. If the sending instants are not perfectly synchronized with the filling instants, the sending time will sometimes occur just in time and sometimes will be delayed by 1 msec relative to the ideal, creating a peak delay variance of 1 msec at the transmitting side.

The intrinsic jitter in other cases is lower, therefore the delay variance generated by the Gmux-2000 TDMoIP modules will not exceed 2 msec.

## Using Jitter Buffers

The method used to mitigate the effects of delay variation is the use of a jitter buffer.

## Background

Ideally, since frames are transmitted at regular intervals, they should reach the destination after some fixed delay. If the transmission delay through the network were indeed constant, the frames would be received at regular intervals and in their original transmission order.

In practice, the transmission delay varies because of several factors:

- Intrinsic jitter at the transmit side, described above
- Variations in the transmission time through the network, caused by the frame handling method: frames pass through many switches and routers, and in each of them the frame (or the packet encapsulated in the frame) is first stored in a queue with frames or packets from other sources, and is then forwarded to the next link when its time arrives.
- Intrinsic jitter at the receive side, due to the variation in the time needed to extract the payload from the received packets.

## Jitter Buffer Functions

Any network designed for reliable data transmission must have a negligibly low rate of data loss. Therefore, it is reasonable to assume that essentially all the transmitted frames reach their destination. Under these circumstances, the rate at which frames are received from the network is equal to the rate at which frames are transmitted by their source (provided that the measurement is made over a sufficiently long time).

As a result, it is possible to compensate for transmission delay variations by using a large enough temporary storage. This storage, called **jitter buffer**, serves as a first-in, first-out buffer.

The buffer has two clock signals:

- Write clock, used to load packets into the buffer. Since each packet is loaded immediately after being successfully received from the network, packets are written into the buffer at irregular intervals.
- Read clock, used to transfer packets to the packet processor at a fixed rate.

The jitter buffer operates as follows:

- At the beginning of a session, the buffer is loaded with a conditioning pattern until it is half full. No bits are read from the buffer at this time. Therefore, a delay is introduced in the data path.
- After the buffer reaches the half-full mark, the read-out process is started. The data bits are read out at an essentially constant rate. To prevent the buffer from either overflowing or becoming empty (underflow), the read-out rate must be equal to the average rate at which frames are received from the network. Therefore, the buffer occupancy remains near the half-full mark.

The buffer stores the frames in accordance with their arrival order.

## Selecting an Optimal Jitter Buffer Size

For reliable operation, the jitter buffer must be large enough to ensure that it is not emptied when the transmission delay increases temporarily (an effect called **underflow**, or **underrun**), nor fills up to the point that it can no longer accept new frames when the transmission delay decreases temporarily (an effect called **overflow**).

The minimum size of a jitter buffer depends on the intrinsic jitter: usually, the minimum value is 3 msec. The maximum size is 300 msec.

The theoretically correct value for the size of the jitter buffer of any given bundle is slightly more than the maximum variation in the transmission delay through the network, as observed on the particular link between the bundle source and the destination. For practical reasons, it is sufficient to select a value that is not exceeded for any desired percentage of time: for example, a value of 99.93% means that the jitter buffer will overflow or underflow for an accumulated total of only one minute per day.

Jitter buffers are located at both ends of a link, therefore the delay added by the buffers is twice the selected value. The resultant increase in the round-trip delay of a connection may cause problems ranging from inconvenience because of long echo delays on audio circuits (similar to those encountered on satellite links) to time-out of data transmission protocols.

Therefore, the size of each jitter buffer must be minimized, to reduce the round-trip delay of each connection in as far as possible, while still maintaining the link availability at a level consistent with the application requirements.

## Adaptive Timing

Because of the transmission characteristics of packet switching networks, which use statistical multiplexing, the average rate must be measured over a sufficiently long interval. The optimal measurement interval is equal to the difference between the maximum and minimum transmission delays expected in the network.

As explained above, the buffer is used to store packets for an interval equal to the maximum expected delay variation. Therefore, this buffer can be used by the adaptive timing mechanism, to recover a clock having a frequency equal to the average transmit rate.

The method used to recover the payload clock of a bundle is based on monitoring the fill level of the jitter buffer: the clock recovery mechanism monitors the buffer fill level, and generates a read-out clock signal with adjustable frequency. The frequency of this clock signal is adjusted so as to read frames out of the buffer at a rate that keeps the jitter buffer as near as possible to the half-full mark. This condition can be maintained only when the rate at which frames are loaded into the buffer is equal to the rate at which frames are removed.

Assuming that the IP network does not lose data, the average rate at which payload arrives will be equal to the rate at which payload is transmitted by the

source. Therefore, the adaptive clock recovery mechanism actually recovers the original payload transmit clock.

This mechanism described above also generates a clock signal having the frequency necessary to read-out frames at the rate that keeps the jitter buffer as near as possible to the half-full mark.

The bundle used as the basis for recovering the adaptive clock can be selected by the user.