Chapter 26

Ethernet frames, Wireshark® and FAT32

TEC/NOT/051

This paper explains how a network recorder operates. The following topics are discussed:

- "26.1 Overview" on page 1
- "26.2 Encapsulating data in Ethernet frames" on page 1
- "26.3 Recording Ethernet frames as Wireshark PCAP files" on page 4
- "26.4 Storing PCAP files in a FAT32 file system" on page 6
- "26.5 Appendix" on page 7

26.1 Overview

An Ethernet network recorder can capture data stored in Ethernet frames and write them to a file using the Wireshark Packet CAPture (PCAP) file format. This paper outlines the process of packing data samples in Ethernet frames for transmission across an Ethernet networked data acquisition system.

The PCAP file format is a flexible and efficient means of recording networked data which can then be processed and viewed on a range of open and closed applications. During a recording session, multiple PCAP files may be created. These PCAP files are then stored in a FAT32 file system.

26.2 Encapsulating data in Ethernet frames

Packet encapsulation is the process where sampled data is packetized to be transmitted. The TCP/IP model has four abstraction layers and is an accepted alternative to the seven-layer OSI (Open System Interconnect) reference model. Each abstract layer attaches a descriptive header—this header contains specific information required to correctly route the data and unpack the data when it reaches its intended destination.

In summary, packet encapsulation creates a packet by prepending a header to the data and passing the packet onto the next abstract layer. The following figure illustrates the packetization process and displays several Ethernet frames consecutively transmitted on the wire.

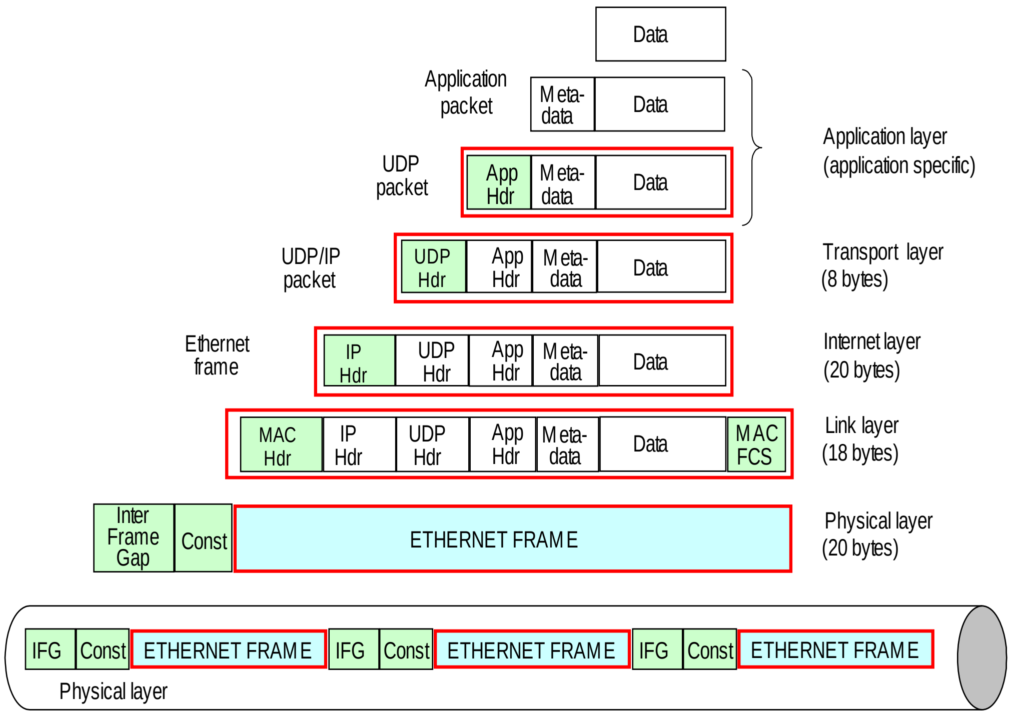


Figure 26-1: Packet encapsulation

The Application layer gathers data to be transmitted, and prepends the associated metadata in the application header. Typically, the application header contains information such as sequence numbers and time stamps.

NOTE: This paper assumes a generic Application layer similar to IENA iNET-X or RTP as compared in “26.5.2 Existing Application layer protocols” on page 9. For more information on the Application layer used in the SSR-500 recorder, see *TEC/NOT/067 - IENA and iNET-X packet payload formats*.

The application packet is then passed to the Transport layer where the User Datagram Protocol (UDP) header is added. The UDP header contains source and destination port numbering.

To correctly route the data to its destination, the Internet layer prepends the Internet Protocol (IP) header, which provides the necessary logical source IP address and destination IP address.

The Link layer creates the Ethernet frame by prepending the MAC (Medium Access Control) header, which contains the source and destination hardware address, and appends the MAC Frame Check Sequence (FCS).

NOTE: Ethernet requires networked devices to have both a logical system-wide unique IP address and a unique hardware MAC address. Both the logical and hardware addresses are required to route data to its destination.

Finally, the Ethernet frame is transmitted over the Physical layer. The Physical layer (see the previous figure) imposes an Inter-Frame Gap (IFG), that is, the duration of 96-bit times of the physical link. For example, the duration of 96-bit times on a 100BaseT (100 Mbps twisted pair link) is 960 ns. Following the IFG, there is a constant preamble and Start Of Frame (SOF) delimiter, signaled before the Ethernet frame is transmitted over the link. On high-speed links, the IFG and SOF are not easily observed using an oscilloscope.

26.2.1 Packetization efficiency

As described in “26.2 Encapsulating data in Ethernet frames” on page 1, each layer adds additional header overhead to the data to be transmitted over the network. Since bandwidth is a finite resource, it is important to maximize the packetization efficiency of the data to be transmitted.

Assuming a generic Application layer adds a 16-byte overhead, the total overhead to transmit each packet is 82 bytes, as summarized in the following table.

Table 26-1: Example: packet encapsulation overhead (82 bytes)

Abstract encapsulation layer	Layer overhead (bytes)
Application layer	16
Transport layer – UDP	8
Internet layer – IPv4	20
Link layer – Ethernet MAC	18
Physical layer	20 12 bytes of an Inter-Frame-Gap 96-bit times (12 bytes), and lasts 960 ns on 100 Mbps link 7 bytes of preamble, for example 10101010… 1-byte Start Of Frame (SOF) delimiter

If every 2-byte sample is to be transmitted in a single packet, the total bandwidth required to transmit a sample is 84 bytes. The packing efficiency of carrying 2 bytes of data can be calculated as follows:

Packing efficiency = data transmitted / (overhead + data transmitted)

$2.4\% \approx 2$ bytes / (82 bytes + 2 bytes)

This is clearly an inefficient use of bandwidth. However, if 512 samples of the same parameter are transmitted in a single Ethernet frame, the total bandwidth required to transmit these samples is 1106 bytes.

Using the same formula, packing more samples into a single Ethernet frame greatly improves packing efficiency. This is described in the following example:

93% ≈ (512 × 2 bytes) / (82 bytes + (512 × 2 bytes))

The following figure illustrates how packing efficiency increases with the number of 2-byte samples carried within the packet. By packing more samples into a single Ethernet frame, there is greater packing efficiency and therefore better bandwidth management. However, there is an upper limit to the number of samples that can be contained in a single packet; this is described in “26.2.2 Ethernet frame sizes and fragmentation” on page 3.

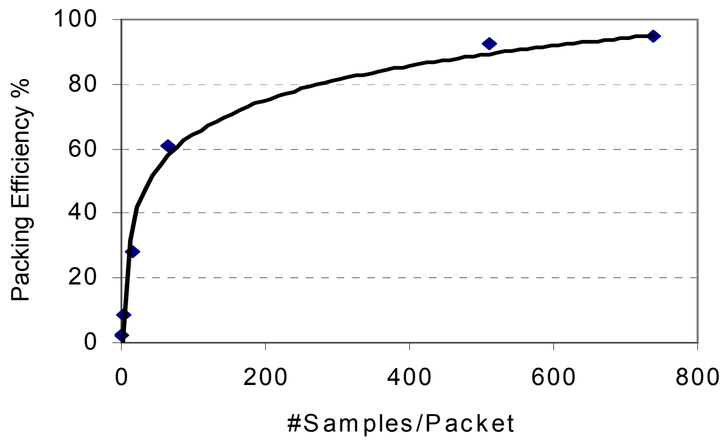


Figure 26-2: Packing efficiency with number of samples per packet

26.2.2 Ethernet frame sizes and fragmentation

Ethernet frames are subject to certain rules imposed by the IEEE 802.3 Ethernet standard. Valid Ethernet frames must be sized between 64 bytes and 1518 bytes. Ethernet frames that are outside this range may be either discarded or fragmented by the switch, router, or network interface card. (Frames that are less than 64 bytes are known as runts while frames greater than 1518 bytes are known as giants.)

NOTE: Some routers and switches allow Ethernet frames of 1,522 bytes (this includes the 4-byte VLAN 802.1q tagged Ethernet frames).

IP fragmentation occurs when the size of the IP packet exceeds the Maximum Transmission Unit (MTU) on the network link. The MTU is the maximum sized packet that is allowed to be transferred across the network link. IP packets transmitted over Ethernet have an MTU of 1,500 bytes.

Fragmentation should be avoided for the following reasons:

- Fragments may arrive out of order and complicate reassembly of the data.
- Fragments may be lost, rendering all other fragments for the frame useless.
- The process of fragmenting and reassembly of the data takes time.

It is generally accepted that, excluding the Link layer overhead, the IP packet may not exceed 1,500 bytes (inclusive of the IP header overhead). The maximum number of samples (S) that can be packed into a single unfragmented packet can be calculated as follows:

S × SampleSize = 1,500 bytes - (IP + UDP + generic application overhead)

Assuming a 16-byte generic application header overhead is:

S × 2 bytes = 1,500 - (20 byte IP + 8 byte UDP + 16 byte application header)

then,

S = 728 samples

26.2.3 Packetization delay

As discussed in “26.2.2 Ethernet frame sizes and fragmentation” on page 3, it is clear that the more samples there are contained in a single packet, the greater the efficiency of the bandwidth. However, there is a trade-off in terms of packetization delay (see the following figure).

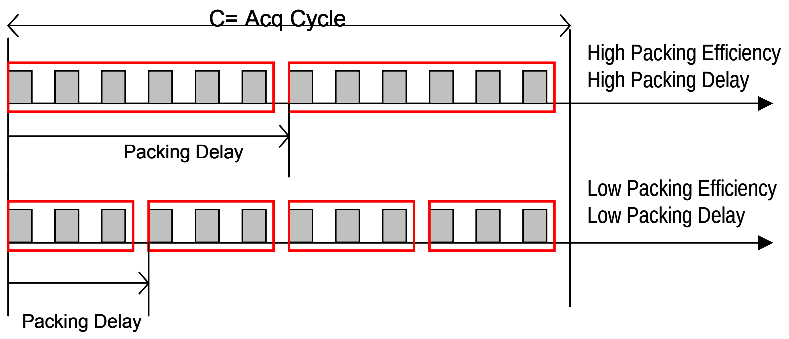


Figure 26-3: Trade-off of packing efficiency against packing delay

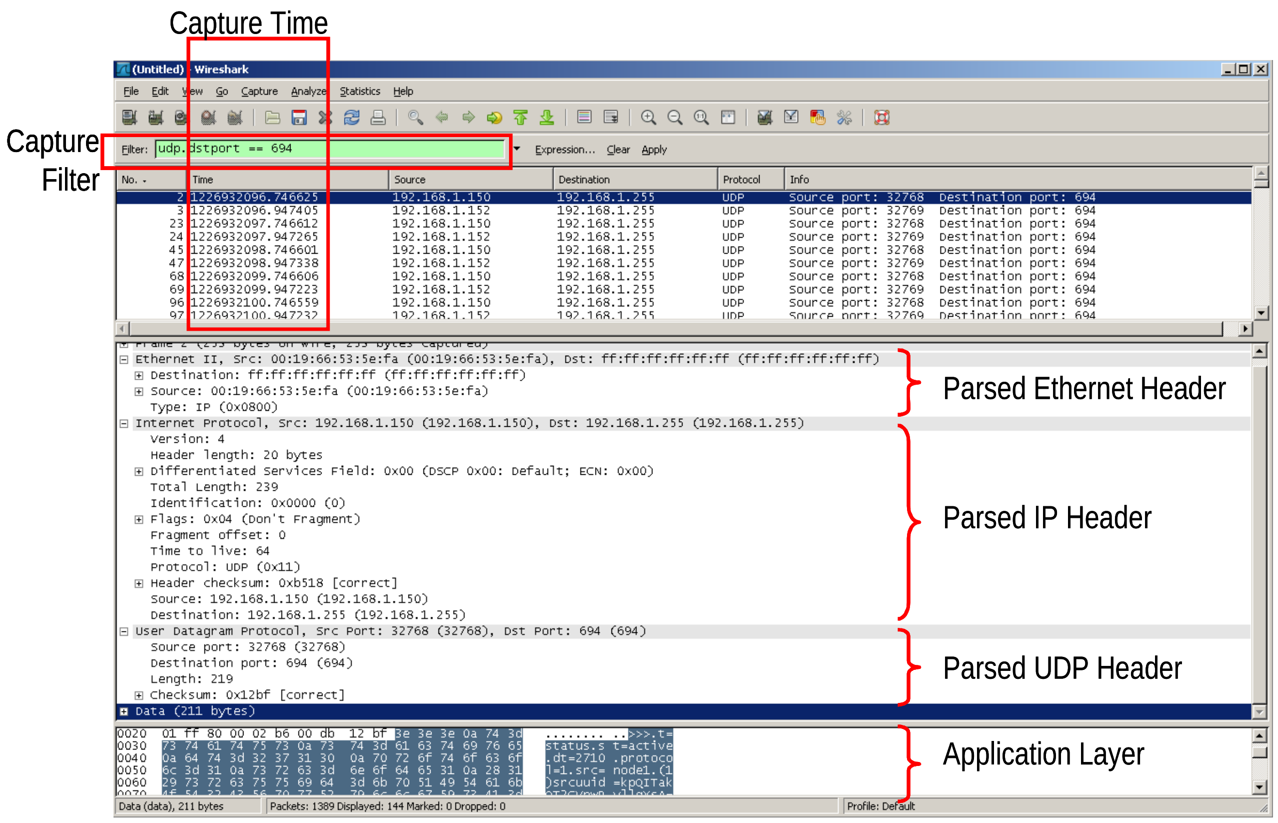
For example, in order to pack 512 two-byte samples of a single parameter sampled at 1,024sps (samples per second) into one packet (to achieve 93% bandwidth packing efficiency), the first sample in the packet would be 500 ms old before the packet is transmitted. A delay of 500 ms may not be a problem if the data is being stored for analysis, however if the data is being viewed in real-time, a delay of even 250 ms can be significant.

26.3 Recording Ethernet frames as Wireshark PCAP files

Known previously as Ethereal, Wireshark is a widely used network protocol analyzer. It is an open source software project, and is released under the GNU General Public License (GPL).

Wireshark non-intrusively reads and captures live packet data on physical networking technologies including Ethernet, IEEE 802.11, PPP/HDLC, ATM, Bluetooth, USB, Token Ring, Frame Relay and Fibre Distributed Data Interface (FDDI).

WinPCAP, the Wireshark packet-capturing engine, is the industry-standard tool for link layer network access and enables the capture of Ethernet frames. Filters can be used to capture only packets that are of interest, for example, packets with a given destination IP address.



Filter: udp.dstport == 694

```
No. -  Time                Source          Destination     Protocol  Info
     2 1226932096.746625   192.168.1.150   192.168.1.255   UDP       Source port: 32768  Destination port: 694
     3 1226932096.947405   192.168.1.152   192.168.1.255   UDP       Source port: 32769  Destination port: 694
    23 1226932097.746612   192.168.1.150   192.168.1.255   UDP       Source port: 32768  Destination port: 694
    24 1226932097.947265   192.168.1.152   192.168.1.255   UDP       Source port: 32769  Destination port: 694
    45 1226932098.746601   192.168.1.150   192.168.1.255   UDP       Source port: 32768  Destination port: 694
    47 1226932098.947338   192.168.1.152   192.168.1.255   UDP       Source port: 32769  Destination port: 694
    68 1226932099.746606   192.168.1.150   192.168.1.255   UDP       Source port: 32768  Destination port: 694
    69 1226932099.947223   192.168.1.152   192.168.1.255   UDP       Source port: 32769  Destination port: 694
    96 1226932100.746559   192.168.1.150   192.168.1.255   UDP       Source port: 32768  Destination port: 694
    97 1226932100.947232   192.168.1.152   192.168.1.255   UDP       Source port: 32769  Destination port: 694
```

```
⊟ Ethernet II, Src: 00:19:66:53:5e:fa (00:19:66:53:5e:fa), Dst: ff:ff:ff:ff:ff:ff (ff:ff:ff:ff:ff:ff)
  ⊞ Destination: ff:ff:ff:ff:ff:ff (ff:ff:ff:ff:ff:ff)
  ⊞ Source: 00:19:66:53:5e:fa (00:19:66:53:5e:fa)
    Type: IP (0x0800)
⊟ Internet Protocol, Src: 192.168.1.150 (192.168.1.150), Dst: 192.168.1.255 (192.168.1.255)
    Version: 4
    Header length: 20 bytes
  ⊞ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
    Total Length: 239
    Identification: 0x0000 (0)
  ⊞ Flags: 0x04 (Don't Fragment)
    Fragment offset: 0
    Time to live: 64
    Protocol: UDP (0x11)
  ⊞ Header checksum: 0xb518 [correct]
    Source: 192.168.1.150 (192.168.1.150)
    Destination: 192.168.1.255 (192.168.1.255)
⊟ User Datagram Protocol, Src Port: 32768 (32768), Dst Port: 694 (694)
    Source port: 32768 (32768)
    Destination port: 694 (694)
    Length: 219
  ⊞ Checksum: 0x12bf [correct]
⊞ Data (211 bytes)
```

```
0020  01 ff 80 00 02 b6 00 db  12 bf 3e 3e 3e 0a 74 3d   ........ ..>>>.t=
0030  73 74 61 74 75 73 0a 73  74 3d 61 63 74 69 76 65   status.s t=active
0040  0a 64 74 3d 32 37 31 30  0a 70 72 6f 74 6f 63 6f   .dt=2710 .protoco
0050  6c 3d 31 0a 73 72 63 3d  6e 6f 64 65 31 0a 28 31   l=1.src= node1.(1
0060  29 73 72 63 75 75 69 64  3d 6b 70 51 49 54 61 6b   )srcuuid =kpQITak
```

Data (data), 211 bytes | Packets: 1389 Displayed: 144 Marked: 0 Dropped: 0 | Profile: Default

Figure 26-4: Wireshark packet capturing and filtering

The previous figure illustrates how a filter has been applied to display only UDP packets whose destination port number is 694. When a packet meeting this criterion is received, its capture is time stamped. Wireshark dissects the Link, Internet, and Transport encapsulation layers. The physical layer IFG, preamble, and SOF delimiter need not be recorded since these are constant for all Ethernet frames that are captured. Moreover, the network interface stack removes the 4-byte Ethernet FCS for valid error-free Ethernet frames.

Wireshark offers a number of options for recording to file by generating a new capture file every N minutes, N megabytes or N packets. In Record mode, Wireshark writes the captured packets using the PCAP file format. PCAP files are designed to support recording on a number of different networking technologies including Ethernet, FDDI and Token Ring. The PCAP file format is stable (last revision 1998) and is not expected to change in the future. It is supported and used on a range of open and closed network recording and analysis tools.

The following figure illustrates the format of PCAP files. Each PCAP file has a 24-byte Global header containing global information describing the recording session, followed by zero or more records for each captured packet. Each captured packet is prepended with a 16-byte packet header.

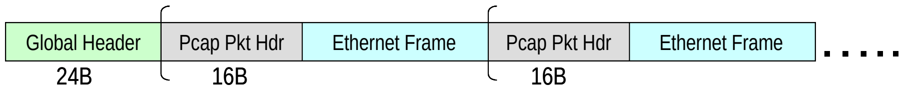


Figure 26-5: PCAP file format

The Global header contains basic metadata on the packet capture recording session (see the following table) while the PCAP packet header records the length and capture time of the packet (see Table 26-3 on page 5).

Table 26-2: Global header

Field name	Length (bytes)	Description
Magic number	4	Used to detect the file format and the byte ordering. The writing application writes 0xa1b2c3d4 (using its native byte ordering format) to this field. The reading application reads either 0xa1b2c3d4 (identical) or 0xd4c3b2a1 (swapped). If the reading application reads the swapped 0xd4c3b2a1 value, it indicates that all the following fields must also be swapped.
Version major:minor	2:2	The version number of this file format (current version is 2.4).
This zone	4	The correction time, in seconds, between UTC and the local time zone of the following packet header time stamps. For example, if time stamps are in UTC, this zone is simply 0; if time stamps are in Central European time, such as Amsterdam or Berlin (which is UTC + 1:00), this zone must be -3600.
Sig flags	4	The accuracy of the capture timestamp. Not supported in most PCAP-compatible tools.
Snap length	4	To capture the entire Ethernet frame, this is typically set to 65535.
Network	4	Data link layer type, for example 1 for Ethernet. This can be various types such as Token Ring or FDDI.

Table 26-3: PCAP packet header

Field name	Length (bytes)	Description
Ts Sec	4	The date and time when this packet was captured. This value relates to seconds since January 1, 1970 00:00:00 UTC.
Ts USec	4	The time, in microseconds, when this packet was captured, as an offset to ts_sec.
Incl Len	4	The number of bytes of packet data actually captured and saved in the file.
Orig Len	4	The length of the packet, as it appeared on the network when it was captured.

26.3.1 PCAP time stamp

The granularity of a PCAP time stamp is microseconds. This has sufficient accuracy to record packet capture times. The inter-arrival time between consecutively captured packets is at least 6720 ns, calculated as the sum of a minimum sized Ethernet frame of 64 bytes or 512 bits (including the Link, Internet, and Transport layers), fixed IFG of 96-bit times (or 960 ns on a 100 Mbps link), preamble and SOF of 64 bits.

26.3.2 PCAP storage efficiency

As seen in “26.3 Recording Ethernet frames as Wireshark PCAP files” on page 4, when storing packets in the Wireshark PCAP file format an additional 16-byte overhead is imposed to record each packet; the total overhead to store a single packet is 74 bytes. This includes the PCAP packet (16 bytes), generic application header (16 bytes), UDP transport header (8 bytes), IP internet header (20 bytes), and 14 bytes MAC link layer header (excluding the MAC FCS). The PCAP file format does not record the physical layer constants such as the preamble, SOF, or the IFG.

If each 2-byte sample is stored in a single packet, the total memory required to record a single sample is 76 bytes. The efficiency of storing 2 bytes of data can be calculated as follows:

Storage efficiency = (data stored) / (storage overhead + data stored)

2.6% ≈ 2 bytes / (74 bytes + 2 bytes)

This is clearly an inefficient use of memory. However, if 512 samples of the same parameter were to be recorded in a single Ethernet frame, the total memory required is 1,098 bytes. By packing more samples into one frame, storage efficiency is vastly improved.

Storage efficiency = (data stored) / (storage overhead + data stored)

93% ≈ (512 × 2 bytes) / (74 bytes + (512 × 2 bytes))

Despite the overhead imposed to support the PCAP file format, the storage efficiency is comparable to the packing efficiency (see the following figure).

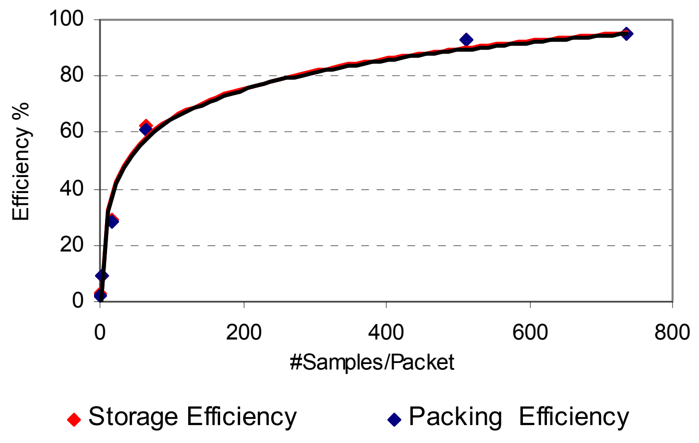


Figure 26-6: Comparison of storage efficiency and packing efficiency

26.4 Storing PCAP files in a FAT32 file system

The File Allocation Table (FAT) system is particularly suited to solid-state memory cards, and is a convenient way of sharing data between operating systems. The maximum file size supported on FAT32 systems is just under 4 GB with a maximum partition size of 2 TB.

The first sector of the FAT system (see the following figure) is the Reserved Region. This sector contains information about the file system and includes the following:

- Bytes per sector (512 - 4,096)
- Number of sectors per cluster
- Number of root directories
- Number of sectors per FAT

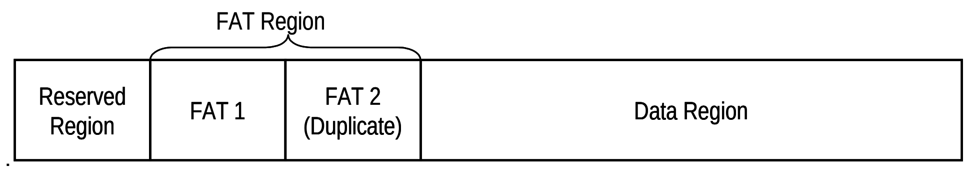


Figure 26-7: FAT32 partition layout

After the Reserved Region is the FAT Region, which records the files or directories contained in the Data Region. The FAT region typically contains two copies of the FAT. A duplicate copy (see FAT2 in the previous figure) is for redundancy checking and error recovery in case FAT1 becomes corrupt. The duplicate copy is rarely used.

The FAT records which files or directories occupy which parts of the Data Region. The Data Region is divided into clusters. The FAT contains one entry for each cluster.

Each FAT entry, which represents part of a file, contains the number of the next FAT entry in the file; the end of the file is marked by a FAT entry containing a special END value (see the following figure). In this way, the FAT defines individually linked lists of clusters in the Data Region used by files.

Directories are stored in the same way as files. Unused clusters and bad (physically damaged) clusters are recorded as special FAT entry values.

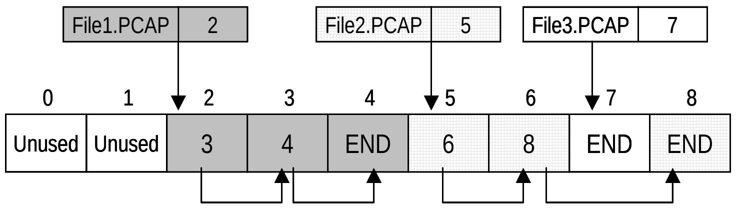


Figure 26-8: Linked list FAT32 overview

Each entry in a FAT32 system is 32 bits long (although only 28 bits are actually used). The upper four bits are usually zero but are reserved and should be left untouched.

The FAT contains one entry for each cluster. For historical reasons, clusters are numbered from 2; the first two table entries of the FAT are unused. Files are stored as whole clusters, so the space occupied in the volume is a multiple of the cluster size. This means that, if the clusters are 32 KB in size, up to 32 KB can be wasted at the end of every file. Data should be stored in large files: a 1-KB file takes up 32 KB of disk space, with 3,100% wastage; whereas a 31,969 KB file takes 32,000 KB which is 0.1% wasted space.

26.5 Appendix

26.5.1 Encapsulation header overhead

Table 26-4: Transport layer - UDP (8 bytes)

Field Name	Length (bytes)
Source port	2
Destination port	2
Length	2
Checksum	2

Table 26-5: Internet layer - IPv4 (20 bytes)

Field Name	Length (bytes)
Version	4 bits
Header length	4 bits
Type Of Service (TOS)	1
Total length	2
ID fragment	2
IP flags	3 bits
Fragment offset	13 bits
Time To Live (TTL)	1
Protocol	1
Header checksum	2
Source IP address	4
Destination IP address	4
Options: The IP header may additionally carry header extension options, as indicated by the header length.	

Table 26-6: Link layer - basic Ethernet MAC frame (18 bytes)

Field Name	Length (bytes)
Destination MAC address	6
Source MAC address	6
Type/length	2
Trailer Frame Check Sequence (FCS)	4

Table 26-7: Link layer - Ethernet MAC Frame with 802.1Q tags (22 bytes)

Field Name	Length (bytes)
Destination MAC address	6
Source MAC address	6
Tag	4
Type/length	2
Trailer Frame Check Sequence (FCS)	4

Table 26-8: Physical layer - 100Mbps Fast Ethernet (20 bytes)

Field Name	Length (bytes)	Description
Inter-Frame Gap (IFG)	12	There is a mandatory idle gap between successive Ethernet frames. This is typically 96 bits (12 bytes), which amounts to an interval of 960 ns on a 100 Mbps link.
Preamble	7	
SOF	1	

26.5.2 Existing Application layer protocols

A number of Application layer protocols have been developed for the specific requirements of telemetry data. These include: IENA (AirBus), Chapter 10 (IRIG 106), TmNS Data Message Protocol (iNET-X), and the iNET-X packet structure. In addition the Real-time Transport Protocol (RTP) that is commonly used for the transmission of real-time data is briefly described.

26.5.2.1 IENA - Airbus

The IENA Application layer protocol partitions logical groupings of data into packet streams. An IENA Key field in the Application header uniquely identifies each packetized group of data. The IENA Key is mapped to a metadata description of the payload contents to facilitate decoding of the received data. IENA defines structures and placement rules that indicate how data can be placed in the packet. These structures may be Positional, Standard or Message type. A packet may contain multiple data structures, however all structures must be of the same type.

26.5.2.2 Chapter 10 – IRIG-106

Chapter 10 is part of the IRIG-106 Telemetry standards. The focus of Chapter 10 is on-board and ground-based recording for which a number of packetized data formats have been defined (for example general and data source specific data formats including, Analog, Discrete, ARINC-429, Video, and Ethernet). The Chapter 10 standard is revised every two years with a release due in 2009. Chapter 10 files consist of a Setup Record followed by data stored in structures known as data specific packets interleaved with time packets. Data packets have three elements: a header; body; and optional trailer. The data packet header has fixed header fields including a Channel ID (analogous to the IENA key) that is used to associate packets from the same source. Following the header, the data is prepended with a Data header.

26.5.2.3 iNET-X and TmNS

The TmNS protocol from iNET-X is in the standards definition phase. It is an ongoing process and subject to future protocol revisions. As of January 2009, only the Application layer header has been fully defined. The approach taken was to identify the elements in the Application layer header that encompass the merits of each of the existing protocols, IENA and Chapter 10. The Application payload consists of Packages that are partitioned subsets of data. It is expected that Packages will have a fixed Package header with the possibility of source specific Package header extensions.

iNET-X packets use the standard iNET-X TmNS Application layer packet structure and are compatible with iNET-X packets. However, iNET-X packets have an additional 4-byte extension field, called the iNET-X Payload Information Field, appended directly following the standard iNET-X header. The iNET-X Payload Information Field contains Curtiss-Wright-specific metadata to facilitate decoding and decommutation of the iNET-X payload.

NOTE: For more information on the iNET-X packet structure and the packetization rules, see *TEC/NOT/067 - IENA and iNET-X packet payload formats*.

26.5.2.4 Real-time Transport Protocol (RTP) - IETF

The RTP protocol is an open-standard protocol that is published by the Internet Engineering Task Force (IETF) defined in the RFC 3550 (Standard 64) document. RTP was originally defined for multimedia applications with real-time transport requirements. Profiles define the rules and packetization structures for different data types in the RTP payload. To date, there is no profile defined that specifically targets telemetry applications.

26.5.2.5 Application layer protocol comparison

These protocols are compared in the following table. There is a notable difference in the terminology used between the different protocol implementations. To unify these terms, the descriptive field (far left column) relates a generic term to a specific protocol term.

Table 26-9: Comparison of existing application layer protocols

Descriptive Field		**IENA** **Airbus**	**Chapter 10** **IRIG 106**	**iNET-X** **iNET-X with Curtiss-Wright header extension**	**RTP** IETF RFC 3550 Standard 64	**Common**
Packet Header	Version	X	√	√	√	NO
	Timestamp	√ - 48bits	√ - 48bits	√ - 64bits	√ - 32bits	YES
	Sequence Number	√	√	√	√	YES
	Stream Identifier	√ - IENA Key	√ - Channel ID	√ - Stream Identifier	√ - Synchronization Source Identifier	YES
	Status/Flags	√ - Key status, Application status	√ - Packet Flags	√	√ - Padding, Marker, Extension	YES
	Length/Size	√	√ - Packet Length/Data length	√	X	YES
	Data Block Type	X – inferred from Key	√ - Data Type	√ - Payload Type	√ - Payload Type Identifier	NO
	Optional Header Extension	X	√ - Secondary header (Time, Reserved, Header Checksum)	√	√	NO
	Other Fields	None	None	None	Optional multiplexing facilities	N/A
Payload Data (Blocks)	Payload Types	√ - General	√ - Data type specific	√ - General and Data Type specific	√ - General and Payload specific	YES
	Packet Body Format	√ - Positional or Standard/Message	√ - Channel specific data + Intra-packet	√ - Blocks	√ - General and Payload specific	YES
	Multiple Data Blocks	√	√	√	TBD for Telemetry	YES
	Block Header	√ - Standard (Param ID); Message (Param ID Length) For Standard (D) and Message (Q) type Param ID followed by Time	√ - Intra-packet time, header. Data	√ - Dependent on Payload Type - Bit-aligned, Video, Placed, Parser-aligned, Error/Event	TBD for Telemetry	NO
	Fixed Length Blocks	X – Message types not fixed length	√ - Unclear, assumed to be fixed from Data Length in Packet Header	X	TBD for Telemetry	NO
	Block Padding	X	Optional - Filler bits in Intra-packet	√	TBD for Telemetry	NO

Table 26-9: Comparison of existing application layer protocols (continued)

Descriptive Field		IENA Airbus	Chapter 10 IRIG 106	iNET-X iNET-X with Curtiss-Wright header extension	RTP IETF RFC 3550 Standard 64	Common
Packet Trailer	Padding	Optional	Optional - Filler	X	Optional	NO
	Trailer	√ - End Field 0xDEAD	√ - Header Checksum	X	X	NO
Packing Rules	Payload Filling Method	√ - Fill to max size or max delay	X – Not specified	√ - Fill to max size or max delay	TBD for Telemetry	NO

This page is intentionally blank