CHAPTER 28

RF Network Management

Table of Contents

List of Figures

List of Tables

Acronyms

AES	Advanced Encryption Standard
BSN	block sequence number
CCMP	Counter with Cipher Block Chaining Message Authentication Code mode Protocol
FCS	frame check sequence
FPSH	fragmentation/packing sub-header
GPS	Global Positioning System
IETF	Internet Engineering Task Force
IP	Internet Protocol
LLC	logical link control
MAC	media access control
MDL	Metadata Description Language
ms	millisecond
MSDU	MAC service data unit
OSI	Open Systems Interconnection
QoS	Quality of Service
RF	radio frequency
RFC	Request for Comment
SDU	service data unit
TCP	Transmission Control Protocol
TE	Traffic Engineering
TLV	Type Length Value
TmNS	Telemetry Network Standard
TSS	TmNS Source Selector
TxOp	transmission opportunity
UDP	User Datagram Protocol

This page intentionally left blank.

CHAPTER 28

RF Network Management

28.1   Introduction

This chapter defines the mechanisms and processes for managing radio frequency (RF) links with the RF network. The RF network implements an Open Systems Interconnection (OSI) model approach (Figure 28-1) to data transmission, where data moves through the OSI stack from the application layer to the physical layer, from physical layer to physical layer through some transmission medium, then back up the stack to another application on the receiving side. For the most part, the RF network operates just like any other Transmission Control Protocol (TCP)/Internet Protocol (IP) network, where a message is created using a standard data management protocol, such as Simple Network Management Protocol; is encapsulated at the transport layer to TCP or User Datagram Protocol (UDP); and then is further encapsulated into an IP packet that contains the logical addressing and path routing determination. Where the RF network differs from the standard OSI model is in the data link and physical layers, where media access controls (MACs) have been modified to support transmission over RF links.

OSI Model				
Layer		**Data Unit**	**Function**	**Examples**	
Host Layers	7. Application	Data	High Level APIs, including resource sharing, remote file access, directory services and virtual terminals	HTTP, FTP, SNMP, SSH, TELNET	
	6. Presentation		Translation of data between a networking service and an application, including character encoding, data compression and encryption/decryption	HTML, CSS, GIF	
	5. Session		Managing communications sessions, i.e. continuous exchange of information in the form of multiple back-and-forth transmissions between two nodes	RPC, PAP, SSL, SQL	
	4. Transport	Segments/Datagram	Managing communications sessions, i.e. continuous exchange of information in the form of multiple back-and-forth transmissions between two nodes	TCP, UDP, NETBEUI	
Media Layers	3. Network	Packet	Structuring and managing a multi-node network, including addressing, routing and traffic control	IPv4 IPv6, IPSec, Apple Talk, ICMP	
	2. Data Link	Frame	Reliable transmission of data frames between two nodes connected by a physical layer	PPP, IEE 802.2 L2TP, MAC, LLDP	**Covered by this chapter**
	1. Physical	Bit	Transmission and reception of raw bit streams over a physical medium	Ethernet physical layer, DSL USB, ISDN, DOCSIS	**IRIG 106 Chapter 27**

Figure 28-1. OSI Model as Related to the Telemetry Network Standard (TmNS) RF Network

The RF network manages communications at the data link and physical layers of the OSI model. This chapter focuses on the RF network with respect to managing the data link layer of the OSI model. The RF network is a multi-node network with a network layer control and data plane. The control plane for managing the RF link layer multiple access is described in this document. For information on the physical layer of the RF network, refer to Chapter 27. The RF network's control plane uses the existing ground network and adds RF connectivity to allow changes to RF transmission and capacity allocation in transceivers during missions. Radios need not establish direct bidirectional links with other transceivers in order to be part of the RF network. Rather, they are part of the network based on the principles and standards associated with normal IP routing protocols and need only support one or more paths to and from the overall RF network.

The RF network media access utilizes an epoch-structure transmission scheme. Management of the epoch schedule is coordinated through the interfaces defined in this document.

In order to support dynamic updates to the epoch structure, transceivers update their currently active policies based on RF network messages. These messages define when the transceiver has been given authority to transmit. This chapter covers the setting of transmission opportunities (TxOps) and the associated supporting RF network messages, which are defined in Chapter 24. The end product of the link layer control plane messaging is the coordinated definition of the start and stop times associated with transmit opportunities in the RF network. An overview of the RF network's link layer concepts is given in Section 28.2.

When coordinating the transmission of multiple transmitting entities, it is the responsibility of RF link management to sequence the sending of TxOp messages and check for feedback of assigned TxOps from the transmitting entities in order to avoid multiple concurrent uses of the channel by multiple transmitters as required by range policy. The TxOps are assigned with either a finite or an infinite timeout value. An example of an epoch structure of an RF network is given in Section 28.3.

A specific RF link manager implementation performing the RF link management function and the transceivers being managed are all RF network components and should provide the interfaces covered in Chapters 22 through 25.

The bit numbering, bit ordering, and byte ordering conventions used in this chapter are described in Chapter 21 Appendix 21-B.

28.2 RF Network Management Concepts and Definitions

28.2.1 Data Link Layer Framing

The RF network provides a standards-based IP network (Internet Engineering Task Force [IETF] Request for Comment [RFC] 791[1] and RFC 2474[2]). Layers supporting this IP layer are unique to the TmNS RF network. Figure 28-2 shows an overview of the protocol layers associated with sending an IP packet over the data link layer and RF physical interface. The IP packets are referred to as MAC service data units (MSDUs) and are comprised of complete IP packets containing user data. The MSDUs are placed into payload blocks with aggregation and fragmentation performed to meet the maximum transmission unit of the RF channel. Advanced Encryption Standard (AES)-Counter with Cipher Block Chaining Message Authentication Code mode Protocol (CCMP) encryption may be used to provide added security on the RF link by encrypting the payload blocks. The length-limited payload blocks are separated into RF MAC frames and link layer header information is added. Forward error correction is added to the RF MAC frames to create LDPC blocks suitable for transmission over the RF interface. Details of the lower levels of this protocol are covered in Chapter 27.

---

[1] Internet Engineering Task Force. "Internet Protocol." RFC 791. Updated by RFC 2474, RFC 6864, and RFC 1349. September 1981. Available at https://datatracker.ietf.org/doc/rfc791/.

[2] Internet Engineering Task Force. "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers." RFC 2474. Updated by RFC 3260 and RFC 3168. December 1998. Retrieved 18 April 2017. Available at https://datatracker.ietf.org/doc/rfc2474/.
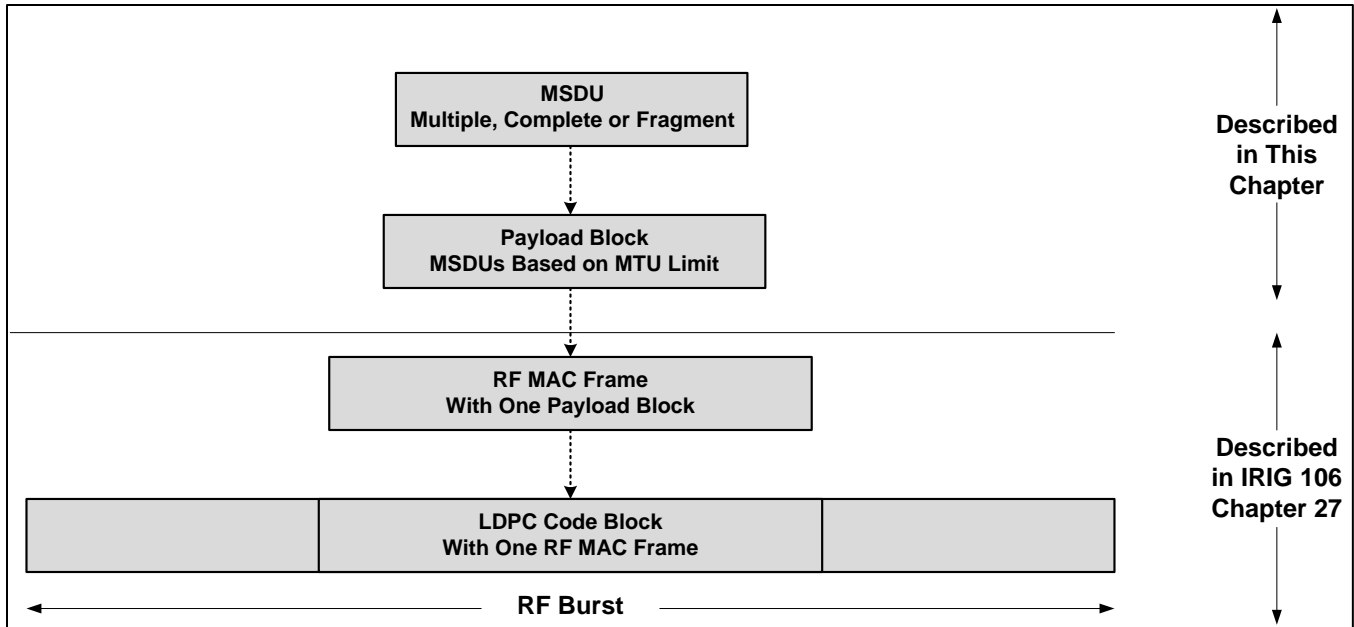
Figure 28-2.   Data Link Layer Framing Overview

28.2.2   RF Link Management

The RF link management is responsible for scheduling RF transmissions in TmNS radios. Time synchronization is critical to the orchestration of the scheduling. The RF link management also performs route selection for packets transiting from the wired networks to the RF networks.

There is at least one RF link management source associated with a ground station. The RF link management on the range may be controlling one or more transmitting entities on the ground. The RF link management also controls transmitting entities that are networked through the ground network, such as those contained on test articles.

The RF link management provides a set of control protocols for managing a transmitting entity's RF spectrum access. These protocols include:

- Transmission scheduling
- RF transmission capacity management
- Handoff management
- Power management - TBD
- Link and RF Traffic Loading Status

28.2.3   Epoch Structure

The RF network implements an epoch structure to provide an efficient utilization over a shared bandwidth. Link management messages support dynamic adjustment of the epoch structure being utilized by components comprising an RF network.

28.2.4   Transmission Opportunities

A TxOp is a window in time over which a transmitting entity may transmit over its associated RF interface. The TxOp contains a start time and stop time that is relative to a

repeating time boundary that is referred to as an epoch. The epoch is settable to a number of discrete times during initialization.

28.2.5   Guard Time

Guard time is the time utilized by an overall RF network epoch schedule to assure clock jitter and RF propagation delays do not create undesired RF collisions. There are no RF network messages associated that specify guard time. Rather, the TxOps that are allocated to components within an RF network should be set up in a fashion in order to support the desired guard time. Chapter 23 specifies an interface for communicating guard time to an RF link manager component. Default guard time is recommended to be 1 millisecond.

28.2.6   Traffic Engineering Queues

The RF network components that are intended to behave as IETF IP routers shall provide Quality of Service (QoS) handling of traffic that is delivered to the RF interface. The QoS interface is implemented in the form of Traffic Engineering (TE) queues. As such, the behavior concerning the ingress and subsequent egress (or in overload situations, drop) of messages shall comply with the details specified in Chapter 22.

28.2.7   Handoff

Handoff is a process by which the RF path is changed to use a different RF network interface. In this way, the route (path) of RF propagation that is experiencing undesired receive RF quality can be directed to a different path. This updated path (post-handoff) will be through a different RF network interface that may be in the current RF network or it may include switching to another RF network. This chapter does not provide a particular mechanism for performing handoff operations but rather a sequence of use of RF management interfaces that can accomplish a variety of handoff operations.

NOTE	It is expected that handoff will be performed when a test article transceiver begins to move out of range of a ground antenna, moves from one RF network to another, or moves from one range to another. This chapter does not specify a policy for when a handoff is to be performed. It can be automatically or manually directed.

28.2.8   Heartbeat

The Heartbeat mechanism defines a relative time to automatically cease transmissions and clear all TxOps from the transmission schedule for a transmitting entity. The initial heartbeat value shall be loaded from a Metadata Description Language (MDL) configuration file, and the value shall be updated upon reception of an RF network message containing a Heartbeat Type Length Value (TLV). Reception of an RF network message containing a Heartbeat TLV from RF link management that is directed to any RF network interface on the receiving transmitting entity will refresh the heartbeat counter for all RF network interfaces (e.g., links) on which the transmitting entity is processing TxOps. Links are defined in Subsection 28.2.10.

28.2.9 RF MAC Header Addressing

An RF MAC address source/destination pair that defines the endpoint of an RF transmission is called a "link". Links are individually managed, including scheduled, by the RF link management.

The RF MAC address shall be a 16-bit value subdivided into a Vendor ID field (most significant 4 bits) and an RF Interface field (least significant 12 bits), whereby the Vendor ID field uniquely identifies the manufacturer and the RF Interface field further uniquely identifies an RF component produced by that manufacturer. One of the Vendor ID fields is reserved for RF multicast group addresses. Table 28-1 displays RF MAC header vendor IDs.

Table 28-1. RF MAC Header Vendor IDs	
Vendor IDs	**Description**
4'b0000	Reserved
4'b0001– 4'b1101	Vendors
4'b1110	Experimental
4'b1111	Multicast

The assignment of a unique Vendor ID value to each manufacturer is outside the scope of this chapter. Manufacturers are responsible for ensuring the uniqueness of the RF MAC address space of the components they produce.

28.2.10 Timing

The RF link management and all entities under its control shall have their clocks synchronized. Methods of time synchronization are defined in Chapter 22. The format of the time in RF network messages is defined in Chapter 24.

28.2.11 Virtual TxOps

When a transmitting entity does not have any TxOps allocated for the current or any future epochs, a Virtual TxOp time slot shall be available for responding to link layer control plane TCP connection over the RF interface. The Virtual TxOp shall be a single burst in size and shall be located at the beginning of each Global Positioning System (GPS) second. This allows the RF link management to receive return messages from a transmitting entity in the exchange of TCP control messaging after which time a TxOp Assignment TLV can be sent to the transmitting entity in order to maintain long-term communications.

NOTE	The RF link management uses a TCP connection that requires a bidirectional handshake to occur before TxOp Assignment TLVs can be received. The use of the Virtual TxOp transmission allows standard TCP methods to establish the link layer control path without using unprotected messaging.

28.2.12 Independent Operation

An RF network shall be capable of operating independent of RF link management control. In this mode of operation, TxOp allocations are the result of configuring with an MDL file that contains the transmission schedule. A heartbeat value of infinite allows TxOp

assignments to remain in effect indefinitely, assuming the TxOp timeout value remains greater than zero, or until RF link management changes the value by sending a non-infinite heartbeat value in a Heartbeat TLV. The transmitting entity operating independent of RF link management as described shall allow RF link management control to take over independent operations. An independent transmitting entity allows externally received Heartbeat TLVs and TxOp Assignment TLVs to overwrite MDL-provided values.

28.3   RF Media Access Control Layer

The RF MAC layer is responsible for providing access to the physical media (i.e., the wireless RF network). On the transmission side, it is responsible for framing IP packets for physical transmission (adding in the layer 2 hardware addresses for the source/destination pair of the link). On the receive side, it is responsible for validating the checksum sent with each packet (known as the frame check sequence [FCS]) and de-framing the received packet.

28.3.1   Epoch

The RF channel will be supported by an epoch structure to separate transmission signals. It emulates full-duplex communication over a half-duplex link and is utilized by the TmNS RF network. The transmitting entity shall synchronize the epoch start time with a commonly referenced external time synchronization mechanism that is common to the RF network.

28.3.1.1   Epoch Structure

An epoch structure shall contain an integer multiple of TxOp assignments. The number of TxOps and their durations are assigned according to the need and policy that has been put in place by RF link management. Figure 28-3 depicts an example of a schedule with four transmitting entities in a network with TxOps to access the RF network.
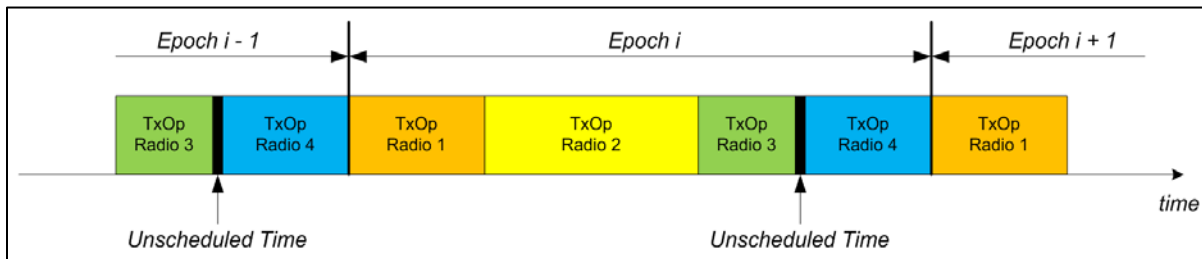

Figure 28-3. Typical Epoch Structure

Transmitting entities in the network are only aware of the start and stop times of a TxOp, and it is the responsibility of RF link management to provide unscheduled time in order to provide sufficient guard times when scheduling TxOps. For times other than the assigned TxOp periods in an epoch, a transceiver shall transition to receive mode, i.e., capable of receiving transmissions destined for it. When there are no packets ready to be transmitted when a scheduled TxOp period occurs, the transmitting entity shall not transmit. A detailed description of the TmNS RF burst sequence requirements for the transmitting entity can be found in Chapter 27.

28.3.1.2   Epoch Timing

The duration of the epoch frame period shall be constrained to the following allowable values: 1000 ms (maximum epoch size), 500 ms, 250 ms, 125 ms, 100 ms (default epoch size), 50 ms, 40 ms, 25 ms, 20 ms, 10 ms (minimum epoch size). Epoch size for a transmitting entity shall only be set during configuration through the MDL configuration file. It is not required that RF link management update the allocated capacity at the same rate as the epoch period used in the RF network. Epochs shall be aligned with time-synchronized seconds as defined in Chapter 22 corresponding with a transition between two adjacent epochs.

NOTE	While this chapter allows for different components within an RF network to be configured with different epochs, it is expected that all epochs will be the same.

28.3.1.3   Guard Times

Between TxOp allocations, a period of no transmissions referred to as a guard time may be provided by RF link management or independent MDL configuration to allocate time between transmissions. Guard times also can be used to account for RF propagation delays across the range. It is the responsibility of the RF link management to provide guard times between the TxOps that it allocates. Guard times are intentional gaps in the epoch structure in which no transmitting entity has a TxOp.

28.3.1.4   Transmission Opportunity

The use of TxOps provides a mechanism of provisioning coordinated channel capacity across multiple transmitting entities on an RF network based on the policies in place for the range. To access the RF media, one or more TxOps is assigned to a transmitting entity. A single TxOp is the authority of the transmitting entity to transmit to its RF interface between two time periods, referred to as start and stop times. The start and stop times of a TxOp are defined relative to a recurring epoch start time. The epoch start time of a transmitting entity should be synchronized with GPS seconds or external time synchronization as defined in Chapter 22.

Each TxOp has a defined destination RF MAC address (e.g., link) that is set to the RF MAC address of an individual transceiver or an RF multicast group address. Transceivers can be set to listen on one or more multicast addresses in addition to their RF MAC address. Multiple TxOps can be assigned to a transmitting entity that are for the same, different, or a combination of RF MAC addressed destinations.

If a transmitting entity does not have sufficient data to generate burst sequences that completely fill a TxOp, then it may cease transmission after all needed burst sequences have been transmitted, i.e., a transmitting entity is not required to pad to fill a TxOp.

If a transmitting entity has ceased transmission in a TxOp and more data becomes available to be transmitted while there is sufficient time remaining in the TxOp both to allow the minimum time between transmissions and to transmit one or more complete burst sequences, then the transmitting entity shall resume transmission, transmitting one or more contiguous burst sequences.

NOTE	The timeout value in the TxOp TLVs allows epochs of transmissions and not each transmission to be scheduled. This relaxes the requirement of RF link management processing and supporting network speed. Prior to the addition of the timeout functionality, the RF link management was required to generate the complete network complement of RF network messages containing TxOp TLVs and transfer them to the transmitting entities, meeting tight setup times, each epoch for proper RF data plane management.

28.3.2 Media Access Control Frame Structure

The MAC frame structure determines what RF transmissions are received. The RF MAC filters received traffic, accepting only those RF transmissions that the receiving entity is interested in receiving.

28.3.2.1 Header Format

Frame headers for RF MAC control frames contain the destination, source, and a length. The destination address is either an RF MAC address of the destination transceiver or an RF multicast address that specifies the RF multicast group that receiving entities can listen to. The source address is always the RF MAC address of the transmitting entity. A length is included that indicates the length of the MAC payload in the MAC frame. Additional information about the RF MAC frame format can be found in Chapter 27.

28.3.2.2 Unprotected Payload

All link layer control frames are contained in secure TCP payload streams with only the RF MAC layer header including the source and destination sent unprotected. The FCS is also sent unprotected to allow for physical layer verification of a correctly received frame.

28.3.2.3 Protected Payload

The link layer control frames are contained in TCP secure IP packets. These frames contain RF network messages to control transmissions on the RF network. If AES encryption is used, then the secure TCP packets will have an additional level of encryption.

28.3.3 RF MAC Payloads

The MSDUs are received from upper protocol layers by the RF link layer and are aggregated, fragmented, or directly placed into payload blocks. The payload blocks have a fragmentation/packing sub-header (FPSH) header added to preserve the original MSDU shape when reconstruction is performed at the link layer on the receiving end. The combined payload block and FPSH header are then encapsulated either encrypted or unencrypted into the RF MAC frame described in Chapter 27. This is shown in Figure 28-4.
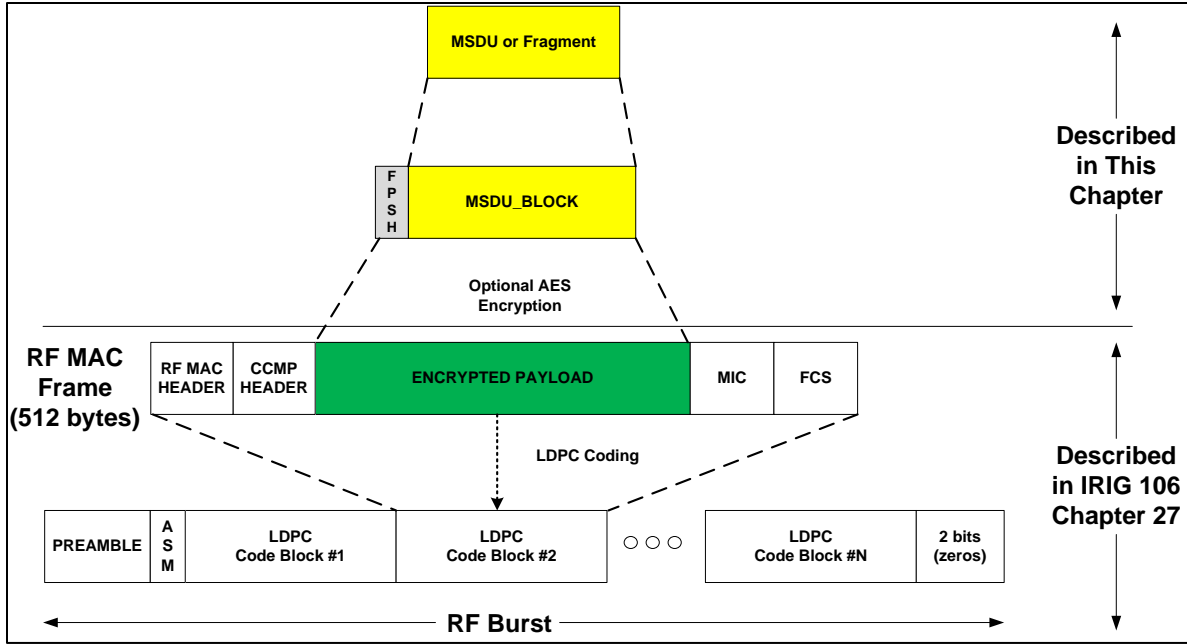
Figure 28-4.   MSDU Insertion into RF MAC Frame

28.3.3.1   RF MAC Service Data Units

The RF MAC service data units (MSDUs) are portions of messages to be transmitted across the RF link. The MSDUs are placed into blocks with aggregation and fragmentation performed.

28.3.3.2   RF MAC Frame Fragmentation

If an IP packet is too long to fit into an RF MAC frame, then it is fragmented, which is the process by which an RF MSDU is divided into one or more MSDU_Blocks. An MSDU_Block contains a full MSDU or a fragment of an MSDU. The fragmentation process is undertaken to allow efficient use of available payload in an RF MAC frame. Capabilities of fragmentation and reassembly are required. Fragments are tagged with their position in their parent SDU in accordance with the values defined for the Fragmentation Control field shown in Table 28-2.

Table 28-2. Definition of Fields in Fragmentation/Packing Sub-Header		
Field	Width (Bits)	Description
FC	2	Fragmentation Control Indicates the fragmentation state of the payload MSDU: 00 = no fragmentation 01 = last fragment 10 = first fragment 11 = continuing (middle) fragment
Reserved	3	Reserved
BSN	11	Block sequence number (BSN) for this MSDU_Block [0 … 2047] modulo 2048

Priority	3	Priority ranking for this MSDU_Block
Length	13	Length (in bytes) of this MSDU_Block, including this six-byte FPSH [7 … 500]
Protocol	16	Type of Protocol Use standard Ethernet values
TOTAL	48	

Multiple short RF MSDUs and/or fragments of RF MSDUs can be packed in the same RF MAC frame. Capabilities of packing and unpacking are required. When an RF MSDU is not fragmented, the Fragmentation Control field in the FPSH shall be set to 00 ("No Fragmentation").

An FPSH precedes each fragment or packed entity. Each fragment or packed message is itself an MSDU_Block. The FPSH structure is depicted in Figure 28-5.
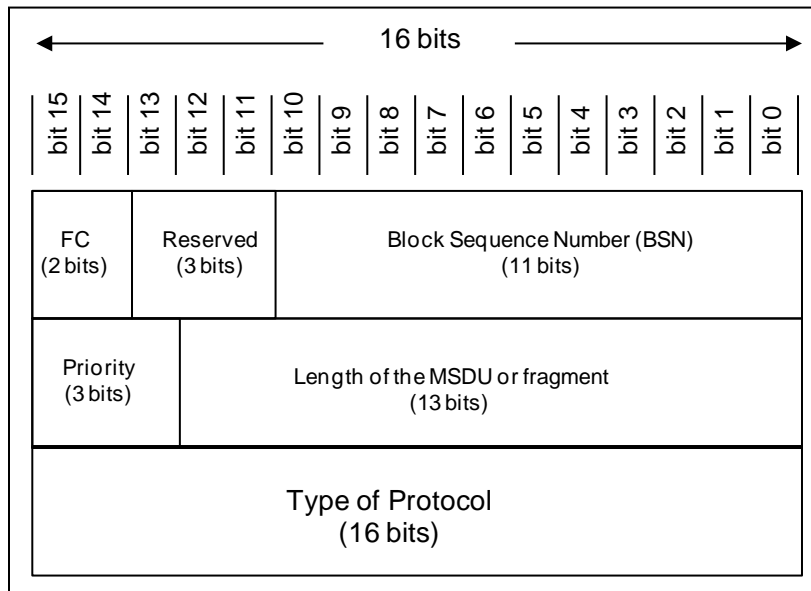
Figure 28-5. Fragmentation/Packing Sub-Header

The MSDU_Block size shall be variable with a maximum of 494 bytes in an unprotected RF MAC frame and 478 bytes in an RF MAC frame protected with AES-CCMP encryption.

The link layer control messages described in this document are sent within standard IP packets that are marked as IP type in the Protocol field.

If an RF MAC frame has a payload field, the payload shall comprise one or more MSDU_Blocks each with its associated FPSH. See the notional diagram in Figure 28-6.

Figure 28-6.   Notional Diagram of RF MAC Frame Containing Two
MSDU_Blocks

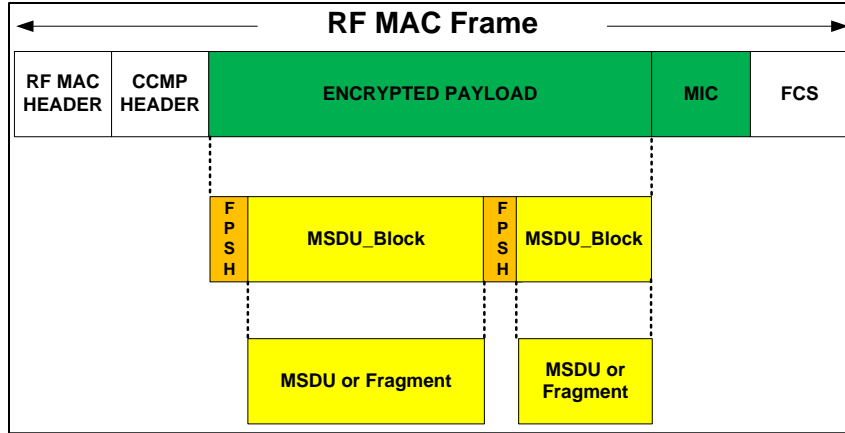Whenever possible, the payload of an RF MAC frame shall be completely filled. This may necessitate fragmenting the next MSDU that is to be transmitted.

If there are 6 bytes or fewer remaining in a payload, the remaining bytes shall be padded with zeroes. The length field in each FPSH tells the size of the following MSDU_Block.

If an MSDU is fragmented into multiple MSDU_Blocks, the multiple MSDU_Blocks shall be transmitted in the same order in which they occurred in the MSDU and shall have BSNs that are consecutive integers, modulo the BSN modulus.

If an MSDU is fragmented into multiple MSDU_Blocks and all the MSDU_Blocks are not transmitted in a single TxOp, then the remaining MSDU_Blocks should be the first new MSDU_Blocks sent at this priority level. When MSDU_Blocks are available for more than one priority level, MSDU_Blocks marked with the higher numeric priority values shall be chosen over lower numeric priority values.

28.3.4   Frame Check Sequence

The FCS contained at the end of an RF MAC frame shall serve as a link layer error-checking mechanism. The FCS generation and verification is covered in Chapter 27. Additionally, the higher-layer protocols (e.g., IP checksums) perform their own error checking.

28.4   Logical Link Control Layer

The logical link control (LLC) layer of the RF network provides media access control and transfer of data frames. The LLC layer provides the control mechanisms for dynamically managing transmission bandwidth.

Epoch-based RF link management is accomplished by sending RF network messages to the transmitting entities to modify the current transmission schedules.

NOTE	All RF link management messages are sent using a secure TCP connection with the transmitting entity. Details of secure connections are provided in Chapter 22 Subsection 22.4.3.

The RF network messages are sent with IP Precedence as defined in Chapter 22 Subsection 22.5.3.2.

An RF network message may contain multiple types of TLVs. When an RF network message contains multiple TLVs, each TLV shall be processed in the order in which they are packed.

28.4.1   TxOp Processing

The external loading of a schedule of TxOps from external MDL shall be equivalent to receiving a sequence of RF network messages with TxOp Assignment TLVs. An export of a transmitting entity's configuration in MDL shall include all the currently scheduled TxOps regardless of whether they originate from TxOp Assignment TLVs in RF network messages, an MDL configuration file, or a combination of both input sources. The transmitting entity shall allow TxOp Assignment TLVs received in RF network messages to modify existing scheduled TxOps, including those initially set during configuration with an MDL file.

A TxOp assignment for a transmitting entity may be different in each epoch, and multiple TxOp Assignment TLVs for use in a single epoch may be sent in the same RF network message from RF link management. Received TxOps shall take effect within two complete epoch start times after reception. For each allocated TxOp in a transmitting entity's schedule, the transmitting entity shall send a status RF network message containing TLVs with both information on all QoS transmit queue levels and receiver link quality to RF link management.

An RF network message containing one or more TxOp Assignment TLVs can be sent by RF link management and likewise be received at any time during the epoch by a transceiver. TxOp Assignment TLV adjustments are state-based, that is they adjust the epoch schedule based on top of all prior adjustments that were accomplished. A TxOp Assignment TLV can add, remove, or modify an existing TxOp window for transmissions. A modification of a currently active TxOp Assignment TLV occurs when a new TxOp Assignment TLV completely subsumes an existing TxOp. The start time of the new subsuming TxOp should be equal to or earlier than the existing TxOp and the stop time should be equal to or greater than the existing TxOp.

The TxOp Assignment TLV also contains a timeout value that specifies the lifetime of the TxOp as measured in epochs. The timeout value can be zero, which will result in the removal of one or more TxOps that are completely subsumed by the TxOp Assignment TLV received regardless of their former timeout values. If the timeout value is infinity, which is defined as the value 255, the TxOp defined by the received TLV is put in place with no epoch-based timeout, and it shall replace all TxOps that are subsumed by the start and stop times of the TxOp TLV.

The TxOp Assignment TLVs are associated with a particular RF link that is a source and destination pair of RF MAC addresses that identifies the source, e.g., the radio that is to use the particular TxOp, and the destination RF MAC address to which it is to transmit. Any transmitting entity that transmits to multiple destination groups shall have separate TxOp Assignment TLVs sent to it in order to differentiate which RF interface is being allocated for transmission. The associated link is contained in the RF network message header.

Each time the transmission schedule of a transmitting entity is updated, it shall report to RF link management with an RF network message containing a TxOp ID Acknowledgement Report TLV. The report shall be generated after the new schedule has been applied. The report

may be generated in the epoch prior to the first use of the associated TxOp if the relative start time and stop time have already occurred within the current epoch if the new TxOp will be first utilized during the next epoch. The TxOps with ID values of zero (16'h0000) shall not be acknowledged by transmission entities. The TxOp ID Acknowledgement Report TLVs are not specific to the particular source and destination RF MAC addresses contained in the message header of the RF network message containing the TLV. If the RF network message only contains the TxOp ID Acknowledgement Report TLV, then the RF network message header should use its own RF MAC address as the source address and a valid RF MAC address for the destination address, such as the address pair associated with the RF network messages that contain the TxOp Assignment TLV messages from RF link management.

A transmitting entity may only transmit over a particular RF interface at the frequency and for the epoch-based periodic time slot that has been specified in a valid TxOp Assignment TLV. Any time a TxOp is executed, the frequency of the transceiver is set according to the associated frequency value that was specified in the corresponding TxOp Assignment TLV, and transmission is allowed for the duration of the TxOp. If a TxOp Assignment TLV's start time is equal to its stop time (i.e., time duration is zero), it shall be considered a valid TxOp Assignment TLV.

NOTE	A zero-duration TxOp may be used to support handoff scenarios that involve a frequency change. That is, they provide a mechanism that supports commanding a transceiver to change its receiving frequency but does not give authority to transmit over the new frequency. Rather, a TxOp received on the new frequency from a different RF network management entity would then give authority to transmit.

Each TxOp is given a lifetime in terms of the number of epochs that it is valid for. The lifetime may be updated before it expires on its own if a new TxOp Assignment TLV arrives with a start and stop time that completely subsumes an existing scheduled TxOp.

Multiple TxOp Assignment TLVs may be packed into a single RF network message if the associated TxOps are destined for the same source over the same RF interface (link).

Because start and stop times are relative to the size of the epoch, a time window equal to the size of the epoch is used to define when the start time and stop time in a TxOp Assignment TLV are considered to be valid. The TxOp Assignment TLVs that define TxOps falling outside this window shall be discarded.

Because the start and stop times of TxOps are quantized values (limited to integers that represent microseconds within an epoch), the start time shall correspond to times greater than or equal to the time represented by the value of the start time field. The stop time shall correspond to the exact instance of the greatest value that is less than the stop time field plus 1. Thus, the valid transmission time associated with a TxOp shall be according to the following equation:

$$t_{start} \leq valid\ transmission\ range < t_{stop} + 1$$

If the start time of one TxOp is equal to the stop time + 1 of the previous TxOp, the TxOps can be called back-to-back TxOps. From a transmitter's perspective, back-to-back

TxOps allow continual transmission without a requirement to turn off the transmitter power prior to the conclusion of the first TxOp.

From a transmitting entity's perspective of the system, each time an epoch is processed, all scheduled TxOps that contain a non-zero timeout value will be executed. After execution, the timeout value associated with that TxOp will be decremented by one unless it is an infinite TxOp with a timeout value of 255. Once the timeout value of a TxOp reaches the value of zero, it is removed from any future processing. For infinite TxOps, the timeout does not decrement after execution.

28.4.1.1   TxOp Processing After Power Interruption

In the event of power interruption, the transceiver shall configure itself to receive using the relevant parameters of the last processed TxOp or the configuration data from the last loaded MDL configuration file, whichever occurred more recently. If no valid TxOps exist, it shall be assumed that no transmission slots are allocated to it for the next epoch. The transceiver shall continue to perform its receiver functionality.

NOTE	In the event of a power interruption, the transceiver should store its current TxOps and associated timeout values. After booting back up, the transceiver shall resume transmissions if the TxOps have not timed out and the transceiver's heartbeat timeout is not zero. This can be determined by calculating the number of epochs that have passed since the reboot event. The transceiver should still obtain time synchronization prior to executing any TxOps.

28.4.1.2   TxOp Processing When Heartbeat Times Out

If the timeout value for the last received heartbeat from RF link management expires before the receipt of the next heartbeat, the transceiver shall flush all scheduled TxOps with a remaining non-zero timeout value that was received.

Any new TxOp Assignment TLVs that are received whose current heartbeat timeout value is zero shall be discarded. A non-zero heartbeat timeout is required in order to process new TxOp Assignment TLVs.

See Subsection 28.4.4 for more information regarding Heartbeat TLVs.

28.4.2   Queue Management Processing

A MAC Queue Status Report TLV in an RF network message shall be generated to provide the state of traffic loading in the TE queue interface to the RF channel as defined in this chapter as well as Chapter 22, Chapter 23, and Chapter 27. The MAC Queue Status Report TLVs shall contain traffic loading for each DiffServ Code Point type contained in the transmitting entity. The traffic loading shall include all storage in traffic bearers contained within the transmitting entity. This RF network message is generated once per TxOp and occurs at the start of each burst. These chapters specify that there are eight TE queues, each of which corresponds to a specific IETF Precedence Class as summarized by the table contained in Chapter 22. Chapter 23 defines the TE queue to IETF class mapping and also includes details of the MDL grammar that provides a method to define user-specific per hop behaviors for a mission and how they apply to each of the TE queues (Precedence Classes). Transmitting entities shall comply with the QoS concepts defined in this chapter, Chapter 22, Chapter 23, and Chapter 27

by selecting MSDUs to send when a TxOp occurs based on the overall policy that has been specified through MDL. This selection process is therefore the overall configured policy for the mission and specifies the behavior choice process that is to be used across the TE queues. As long as time within the TxOp remains, further messages shall be sent based on the TE queue policies until the TxOp is over.

NOTE	It is expected that typical RF link management uses a combination of instantaneous historical queue status TLVs to determine potential adjustment to link layer TxOp allocations.

28.4.3   Link Metric Processing

The Link Metric TLV is used to inform RF link management of the quality of the received data signal in the transceiver. Once a TCP connection is established between RF link management and the transceiver, the transceiver shall send Link Metric TLV(s) at a minimum of once every epoch.

28.4.4   Heartbeat Processing

The initial heartbeat value is obtained through configuration via an MDL configuration file. An RF network message containing a Heartbeat TLV shall be used to overwrite the current value of its heartbeat value.

The Heartbeat TLV, which is described in more detail in Chapter 24, provides a numeric timeout value that represents the number of epochs in the future that the transmitting entity is authorized to execute TxOps. These TLVs are generated by the RF link management and sent to the transmitting entities. The timeout value of a newly received Heartbeat TLV shall replace the existing heartbeat timeout value. It is the responsibility of the RF link management to issue new Heartbeat TLVs before the heartbeat timeout expires. A transmitting entity whose heartbeat value has reached zero shall remove all active TxOps, and the transmitting entity shall not transmit further until it receives a non-zero heartbeat value, either from an RF network message with a Heartbeat TLV or through reconfiguration with an MDL configuration file.

While the heartbeat timeout value of a transmitting entity is zero, any newly received TxOp Assignment TLVs shall be discarded. A Heartbeat TLV may be sent in the same RF network message alongside TxOp Assignment TLVs. Because TLVs within an RF network message are processed in order, the Heartbeat TLV should be first before any TxOp Assignment TLV.

A heartbeat value is a global configuration parameter that affects all RF interfaces on the entity. Reception of an RF network message containing a heartbeat TLV from RF link management that is directed to any RF network interface on the receiving entity will refresh the heartbeat counter for all RF network interfaces (e.g., links).

Heartbeat values that are set to the value representing an infinite lifetime shall never expire; however, these values may be overwritten through RF network messaging as described above.

The current heartbeat value shall be provided in the MDL file produced by a transceiver during an MDL export operation.

NOTE	In the event of a power interruption, the transceiver should store its current heartbeat timeout value. The heartbeat value should be used after the transceiver boots back up, but only after the proper number of decrements to the timeout are made. The number of epochs missed due to the transceiver being rebooted or powered off should be used to determine the proper number of decrements.

28.5   Tunnel Management

Network tunnels provide a mechanism to ease the complexity of transporting TmNS-based data and command and control network packets across pre-existing networks. For example, routing, QoS, multicast delivery, and handoff can be supported by the tunnel, thereby not requiring customized range IT solutions. These tunnels can be created, removed, and adapted through management functions.

Tunnel management provides support for multiple receiving entities of a single transmission in a seamless manner (message-selection-from-many). Doing this allows the network to appear as a packet funnel; that is, it makes the multiple packet receptions appear as a single packet reception. Likewise, tunnel management supports the selection of a particular transmitting entity from a set of possible transmitting entities (transmitter-selection) such that it can appear as a single continuous transmission stream, even in dynamic switching scenarios. Receiving entity message-selection-from-many is a process of allowing multiple receivers to receive the same RF transmissions from a single source. The selection portion chooses the first received packet and forwards it on to its destination while all duplicate packets received are discarded. The transmitter-selection portion of tunnel management allows for the dynamic switching of transmitting entities.

NOTE	Message-selection-from-many and transmitter-selection capabilities of tunnel management can be utilized on a range to support handoff. The message-selection-from-many choice can support a kind of packet-based source selection for the air-to-ground transmissions. The transmitter-selection involves the selection of the transmitting entity for delivering packets to the target network.

Tunnel management shall be implemented by using virtual network interfaces, which appear just like any real network interface to the host operating system, but they can also be accessed by application programs. These virtual interfaces are commonly called "tuns" and are first-class interfaces in Linux-based operating systems. Implementations for Windows-based operating systems and macOS-based operating systems are also prevalent. Open-source examples of tun implementation are available on the Internet. The following links contain information regarding tuns for Linux, Windows, and macOS, respectively:

- https://kernel.org/doc/Documentation/networking/tuntap.txt (See Appendix 28-A)
- https://community.openvpn.net/openvpn/wiki/ManagingWindowsTAPDrivers
- http://tuntaposx.sourceforge.net (See Appendix 28-B)

The overall process of using the virtual interfaces and tunnel selection process is a functionality referred to as the TmNS Source Selector (TSS) capability. The TmNS-compliant transceivers shall provide tuns that can be connected through TCP tunnels by RF link management. Likewise, RF link management shall provide tuns to connect to TmNS-compliant

transceivers. These tuns are referred to as TSS interfaces. The TSS tunnels between TSS interfaces shall be implemented using Secure Sockets Layer over TCP. See Chapter 22, Subsection 22.4.3 for more information on SSL and Subsection 22.4.1 for more information on TCP.

28.5.1   Tunnel Connection

A TSS client initiates a TCP connection to the TSS listening port on a TSS server in order to establish the connection to be used as the tunnel. The default port for listening to incoming TSS connections shall be 55000.

28.5.2   TSS Interface Initialization

Each time a TCP connection is established between a TSS client and the target TSS server, the TSS initialization sequence shall occur. Each tunnel end point shall follow its own initialization sequence.

28.5.2.1   Initialization of TSS Server Interface

The TSS server shall create its virtual interface, apply interface properties to it, and then bring up the interface. Once the interface is up and active on the TSS server, it shall send its virtual interface properties through the tunnel to the connected TSS client. The interface properties shall be carried through TSS initialization messages, which are described in Chapter 24 (including the order in which they shall be sent). The TSS initialization messages shall be the first messages sent through the tunnel. After these six messages are sent, any post-initialization operations, such as routing table updates, may be made.

The TSS interface parameter values for TSS servers may be initialized during configuration through MDL. Default values shall be used if not explicitly described in MDL during component configuration. Default values for a TSS server interface are shown in Table 28-3.

Table 28-3. Default Interface Values for TSS Server Interfaces	
Value	**Description**
192.168.1.255	Broadcast address to associate with the TSS server interface
192.168.1.1	IP address to assign to this TSS server interface
55000	Port to listen on for incoming TCP connection
tap0	Name to give the TSS server interface
255.255.255.0	Netmask of the TSS server interface

28.5.2.2   Initialization of TSS Client Interface

The TSS client shall create a separate virtual interface for each TSS server that a TSS client establishes a TSS tunnel with. The virtual interface shall have its interface properties applied, and then the interface shall be brought up. Once it is up, it shall listen for the six TSS initialization messages being sent from the TSS server interface of the connected TSS server. Once the six messages are received, the TSS client interface has been initialized. If the interface is to be immediately available for routing, a post-initialization routing rule should be added to the TSS client's routing table.

28.5.3   Tunnel Operation

From the RF link management perspective, transmitter-selection IP routing shall be performed by writing to only one of the tunnels associated with the destination IP address. Message-selection-from-many shall use the Cyclic Redundancy Check field of the TSS data message in order to identify duplicated received packets. For message-selection-from-many, the first instance of a received IP packet shall be forwarded towards its destination. All subsequent duplicate packets shall be discarded.

With the exception of TSS initialization messages, all packets that traverse the tunnels are TSS data messages, and they shall conform to the structure described in Chapter 24.

28.5.4   Tunnel Selection

When there are multiple potential RF links (e.g., routes) to a single receiving entity, the RF link management shall route all IP traffic to the target network through the appropriate tunnel. The appropriate tunnel is defined as the tunnel whose endpoint is associated with the transmitting entity that is actively being scheduled with TxOps for the target network.

NOTE	During handoff scenarios, it is expected that RF link management will manage the tunnel selection in conjunction with the TxOp scheduling of the transmitting entities.

Appendix 28-A

Documentation of TunTap Device Driver for Linux

NOTE	This appendix was written and published by somebody not affiliated with the RCC; therefore, it is edited only for formatting and not for content.

Universal TUN/TAP device driver.
Copyright (C) 1999-2000 Maxim Krasnyansky <max_mk@yahoo.com>

Linux, Solaris drivers
Copyright (C) 1999-2000 Maxim Krasnyansky <max_mk@yahoo.com>

FreeBSD TAP driver
Copyright (c) 1999-2000 Maksim Yevmenkin <m_evmenkin@yahoo.com>

Revision of this document 2002 by Florian Thiel <florian.thiel@gmx.net>

1. Description
TUN/TAP provides packet reception and transmission for user space programs. It can be seen as a simple Point-to-Point or Ethernet device, which, instead of receiving packets from physical media, receives them from user space program and instead of sending packets via physical media writes them to the user space program.

In order to use the driver a program has to open /dev/net/tun and issue a corresponding ioctl() to register a network device with the kernel. A network device will appear as tunXX or tapXX, depending on the options chosen. When the program closes the file descriptor, the network device and all corresponding routes will disappear.

Depending on the type of device chosen the userspace program has to read/write IP packets (with tun) or ethernet frames (with tap). Which one is being used depends on the flags given with the ioctl().

The package from http://vtun.sourceforge.net/tun contains two simple examples for how to use tun and tap devices. Both programs work like a bridge between two network interfaces.
br_select.c - bridge based on select system call.
br_sigio.c - bridge based on async io and SIGIO signal.
However, the best example is VTun http://vtun.sourceforge.net :))

2. Configuration
Create device node:
    mkdir /dev/net (if it doesn't exist already)
    mknod /dev/net/tun c 10 200

Set permissions:
   e.g., chmod 0666 /dev/net/tun
There's no harm in allowing the device to be accessible by non-root users, since
CAP_NET_ADMIN is required for creating network devices or for connecting to network
devices that aren't owned by the user in question. If you want to create persistent devices and
give ownership of them to unprivileged users, then you need the /dev/net/tun device to be usable
by those users.

Driver module autoloading
Make sure that "Kernel module loader" - module auto-loading support is enabled in your kernel.
The kernel should load it on first access.

Manual loading
   insert the module by hand:
    modprobe tun

If you do it the latter way, you have to load the module every time you need it, if you do it the
other way it will be automatically loaded when /dev/net/tun is being opened.

3. Program interface
3.1 Network device allocation:

char *dev should be the name of the device with a format string (e.g., "tun%d"), but (as far as I
can see) this can be any valid network device name. Note that the character pointer becomes
overwritten with the real device name (e.g., "tun0")

```
#include <linux/if.h>
#include <linux/if_tun.h>

int tun_alloc(char *dev)
{
    struct ifreq ifr;
    int fd, err;

    if( (fd = open("/dev/net/tun", O_RDWR)) < 0 )
       return tun_alloc_old(dev);

    memset(&ifr, 0, sizeof(ifr));

    /* Flags: IFF_TUN   - TUN device (no Ethernet headers)
     *        IFF_TAP   - TAP device
     *
     *        IFF_NO_PI - Do not provide packet information
     */
    ifr.ifr_flags = IFF_TUN;
    if( *dev )
       strncpy(ifr.ifr_name, dev, IFNAMSIZ);
```

```
    if( (err = ioctl(fd, TUNSETIFF, (void *) &ifr)) < 0 ){
        close(fd);
        return err;
    }
    strcpy(dev, ifr.ifr_name);
    return fd;
}
```

3.2 Frame format:
If flag IFF_NO_PI is not set each frame format is:
    Flags [2 bytes]
    Proto [2 bytes]
    Raw protocol(IP, IPv6, etc) frame.

3.3 Multiqueue tuntap interface:
From version 3.8, Linux supports multiqueue tuntap, which can uses multiple file descriptors (queues) to parallelize packets sending or receiving. The device allocation is the same as before, and if user wants to create multiple queues, TUNSETIFF with the same device name must be called many times with IFF_MULTI_QUEUE flag.

char *dev should be the name of the device, queues is the number of queues to be created, fds is used to store and return the file descriptors (queues) created to the caller. Each file descriptor were served as the interface of a queue that could be accessed by userspace.

```
#include <linux/if.h>
#include <linux/if_tun.h>

int tun_alloc_mq(char *dev, int queues, int *fds)
{
    struct ifreq ifr;
    int fd, err, i;

    if (!dev)
        return -1;

    memset(&ifr, 0, sizeof(ifr));
    /* Flags: IFF_TUN   - TUN device (no Ethernet headers)
     *        IFF_TAP   - TAP device
     *
     *        IFF_NO_PI - Do not provide packet information
     *        IFF_MULTI_QUEUE - Create a queue of multiqueue device
     */
    ifr.ifr_flags = IFF_TAP | IFF_NO_PI | IFF_MULTI_QUEUE;
    strcpy(ifr.ifr_name, dev);

    for (i = 0; i < queues; i++) {
        if ((fd = open("/dev/net/tun", O_RDWR)) < 0)
            goto err;
        err = ioctl(fd, TUNSETIFF, (void *)&ifr);
```

```
        if (err) {
            close(fd);
            goto err;
        }
        fds[i] = fd;
    }

    return 0;
err:
    for (--i; i >= 0; i--)
        close(fds[i]);
    return err;
}
```

A new ioctl(TUNSETQUEUE) were introduced to enable or disable a queue. When calling it with IFF_DETACH_QUEUE flag, the queue were disabled. And when calling it with IFF_ATTACH_QUEUE flag, the queue were enabled. The queue were enabled by default after it was created through TUNSETIFF.

fd is the file descriptor (queue) that we want to enable or disable, when enable is true we enable it, otherwise we disable it

```
  #include <linux/if.h>
  #include <linux/if_tun.h>

  int tun_set_queue(int fd, int enable)
  {
      struct ifreq ifr;

      memset(&ifr, 0, sizeof(ifr));

      if (enable)
         ifr.ifr_flags = IFF_ATTACH_QUEUE;
      else
         ifr.ifr_flags = IFF_DETACH_QUEUE;

      return ioctl(fd, TUNSETQUEUE, (void *)&ifr);
  }
```

Universal TUN/TAP device driver Frequently Asked Question.

1. What platforms are supported by TUN/TAP driver ?
Currently driver has been written for 3 Unices:
  Linux kernels 2.2.x, 2.4.x
  FreeBSD 3.x, 4.x, 5.x
  Solaris 2.6, 7.0, 8.0

2. What is TUN/TAP driver used for?

As mentioned above, main purpose of TUN/TAP driver is tunneling. It is used by VTun (http://vtun.sourceforge.net).

Another interesting application using TUN/TAP is pipsecd (http://perso.enst.fr/~beyssac/pipsec/), a userspace IPSec implementation that can use complete kernel routing (unlike FreeS/WAN).

3. How does Virtual network device actually work ?
Virtual network device can be viewed as a simple Point-to-Point or Ethernet device, which instead of receiving packets from a physical media, receives them from user space program and instead of sending packets via physical media sends them to the user space program.

Let's say that you configured IPX on the tap0, then whenever the kernel sends an IPX packet to tap0, it is passed to the application (VTun for example). The application encrypts, compresses and sends it to the other side over TCP or UDP. The application on the other side decompresses and decrypts the data received and writes the packet to the TAP device, the kernel handles the packet like it came from real physical device.

4. What is the difference between TUN driver and TAP driver?
TUN works with IP frames. TAP works with Ethernet frames.

This means that you have to read/write IP packets when you are using tun and ethernet frames when using tap.

5. What is the difference between BPF and TUN/TAP driver?
BPF is an advanced packet filter. It can be attached to existing network interface. It does not provide a virtual network interface. A TUN/TAP driver does provide a virtual network interface and it is possible to attach BPF to this interface.

6. Does TAP driver support kernel Ethernet bridging?
Yes. Linux and FreeBSD drivers support Ethernet bridging.

This page intentionally left blank.

Appendix 28-B

Documentation of TunTap Project for macOS

NOTE	This appendix was written and published by somebody not affiliated with the RCC; therefore, it is edited only for formatting and not for content.

Overview
What is it?
The TunTap project provides kernel extensions for Mac OS X that allow to create virtual network interfaces. From the operating system kernel's point of view, these interfaces behave similar to physical network adapters such as an Ethernet network interface. However, the virtual interface does not send the packets into a wire, but makes them available to programs running in the system.

The software comes as a pair of kernel extensions that create virtual network interfaces on the IP and Ethernet level, respectively. These kind of network interfaces are commonly referred to as tun and tap devices on Unix-like platforms. This way of interfacing with the operating system's network stack is available on many platforms (cf. the TUN/TAP wikipedia article).

Who needs it?
By design, virtual network interfaces can be very flexibly used by any program that wants to receive packets from and inject them into the network stack. Generally, tun and tap devices are most commonly used in two distinct application scenarios: The first one is VPN software (such as OpenVPN). In this scenario, the kernel sends its network packets to the tun or tap devices. The VPN software will then encrypt and forward them to the other side of the VPN tunnel where they get decrypted and delivered to their destination. The second area in which tun and tap devices are popular are system virtualization/emulation packages. In this case, the virtualized operating system instance talks to a fake network device (commonly a virtual Ethernet adapter). The virtualization software then creates a tap device and interconnects the two such that the host system can talk to the guest and vice versa.

How does it work?
The TunTap package is comprised of a pair of kernel extensions, one providing tun and one providing tap interfaces. They create a set of character devices dev/tunX and /dev/tapX, respectively, where X is a number between zero and the maximum number of supported virtual interfaces. Once an application opens the character device, say <code>/dev/tap0</code>, a virtual network interface is created in the system, which will be named accordingly, i.e., tap0. The network interface can be assigned addresses just like any other network interfaces. After configuring the interface, packets that the kernel sends through this interface (as determined

by the routing table) can be read one packet at a time from the character device. Likewise, packets written to the character device will be injected into the kernel's network stack. For tun interfaces, the packets that are read and written are IP packets. For tap interfaces, the packet format is Ethernet frames.

Mailing list
There is a mailing list available through the Sourceforge project that is meant for general discussion about the TunTap software, asking questions, reporting bugs etc. It is called `tuntaposx-users`. If you are interested, you can register (https://lists.sourceforge.net/lists/listinfo/tuntaposx-users) or have a look at the archives (http://sourceforge.net/mailarchive/forum.php?forum_name=tuntaposx-users).

Donations
So far, I have spent my spare time to run this project. If you want to show your gratitude or support further maintenance or development of the software, you can donate either via the Sourceforge donation system or directly to me via PayPal, just click the appropriate button below. In either case, your money enables me to buy copies of upcoming Mac OS X releases as well as development machines. A huge thank you goes to the nice people at mozilla.com who have given me the Mac Mini I currently use as development machine.

Appendix 28-C

Citations

Internet Engineering Task Force. "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers." RFC 2474. Updated by RFC 3260 and RFC 3168. December 1998. Retrieved 18 April 2017. Available at https://datatracker.ietf.org/doc/rfc2474/.

———. "Internet Protocol." RFC 791. Updated by RFC 2474, RFC 6864, and RFC 1349. September 1981. Available at https://datatracker.ietf.org/doc/rfc791/.

* * * END OF CHAPTER 28 * * *

* * * END OF CHAPTER 28 * * *