NATO Standardization Agreement:
Profile for Maritime High Frequency (HF) Radio Data Communications
STANAG 5066
Version 1.0.2~~1~~

2501 CD The Hague, The Netherlands
Radio Branch
Communications Systems Division

Changes from Version 1.0 to Version 1.0.2 (this should be deleted from the final version):

- Main body, page 5 - added List of Annexes
- In List of Annexes and in each Annex, indicated whether Annex is mandatory or information only

Annex A:

- Corrections to Table A-1
- S_BIND_ACCEPTED: specified that maximum MTU of 4096 for broadcast-only nodes
- S_UNBIND_INDICATION: added reasons
- S_EXPEDITED_UNIDATA_REQUEST: added guidance on use
- S_HARD_LINK_REJECTED: added reasons
- S_HARD_LINK_TERMINATED: added reasons
- Section A.2.1: Added version and size fields to all S_primitives (Figure A-1(a))
- Figure A-1(f): changed wording of TTL to match (new) resolution of TTD
- Figure A-1(g): changed _INDICATION primitives to allow indication of missing as well as blocks with errors
- Figure A-1(i): changed text following to indicate that "deliver in order" applies only to ARQ service.

Section A.3.1

- DATA S_PDU: changed text to match figure here and to match Figure A-4(a)
- DATA DELIVERY FAIL S_PDU: Added reasons
- HARD LINK ESTABLISHMENT REJECTED S_PDU: Added reasons
- HARD LINK TERMINATE S_PDU: Added reasons
- Text following Figure A-4(b): changed definition of TTD
- Section A.3.2.2.1: changed wording and rules

Annex B

- Figure B-3(a): modified logic to allow multiple connections
- Figure B-3(b): expanded

Annex C

- Section C.1: editorial changes. Also that in the "non-reliable" mode, delivery of complete C_PDUs is not guaranteed.

Section C.3

- Previously advertised EOT **shall** not be violated
- Each D_PDU description placed in its own subsection (C.3.2 - C.3.11)
- Section C.3.1: change "first 3 bytes" $\rightarrow$ "first 4 bytes"
- Section C.3.1.2, Figure C-2: added figure C-2(d), EOW Message Type 3 (version number): **The Type 3 EOW message shall be placed in all non-ARQ D_PDUs**
- Section C.3.1.3: EOT clarifications; all zeros reserved for full duplex or broadcast operation
- Section C.3.1.4: clarified address encoding
- Section C.3.1.5: added example CRC calculation
- New section C.3.3: deleted sentence "If the bit representing … it will always be zero)." Changed Figure C-6 to match.
- New section C.3.3: Changed Table C-4 and text so that repetitions are not mandatory.
- New section C.3.5: added reference to C.4.4 for detailed FULL RESET procedure; changed Table C-5 and text so that repetitions are not mandatory.
- New section C.3.8: changed table C-6 to reserve message types 5 and 6 for ALM. Changed Figure C-12(a) to apply to Message types 0 and 4.
- New section C.3.9: Added reference to new type 4 EOW message and specified that it shall be sent when non-ARQ D_PDUs are transmitted.

- New section C.3.11: added statement that a WARNING D_PDU shall not be sent in response to another WARNING D_PDU; Table C-12 under review
- Section C.4: added statement that state machine implementation is not mandatory.
- Section C.4.1.1: made Tables C-13, C-14, C-15 and C-16 consistent
- Section C.4.2: revised link layer flow control rules.
- New section C.4.4: Mandatory procedures for FULL RESET using type 6 D_PDU.

Annex D: now mandatory

- interface shall be synchronous

Annex I (new): added procedures and MANAGEMENT messages to support ALM (frequency change using for example an ALE system)

STANAG 5066

NAVY

NATO STANDARDIZATION AGREEMENT (STANAG)

PROFILE FOR MARITIME HF RADIO DATA COMMUNICATIONS

Annexes:

RELATED DOCUMENTS:

STANAG 4203	TECHNICAL STANDARDS FOR SINGLE CHANNEL HF RADIO EQUIPMENT
STANAG 4285	CHARACTERISTICS OF 1200/2400/3600 BITS PER SECOND SINGLE TONE MODULATORS/ DEMODULATORS FOR HF RADIO LINKS
STANAG 4529	CHARACTERISTICS OF SINGLE TONE MODULATORS/ DEMODULATORS FOR HF RADIO LINKS WITH 1240 HZ BANDWIDTH
MIL-STD-188-110A	INTEROPERABILITY AND PERFORMANCE STANDARDS FOR DATA MODEMS
CCITT V.41	CODE INDEPENDENT ERROR CONTROL SYSTEM
CCITT V.42	ERROR-CORRECTING PROCEDURES FOR DCES USING ASYNCHRONOUS-TO-SYNCHRONOUS CONVERSION

AIM

The aim of this agreement is to define the functions and interfaces required for networked, error-free communication over maritime HF radio channels.

AGREEMENT

The participating nations agree to implement the profile defined in this STANAG (including mandatory Annexes) to provide communications over maritime ship-to-shore HF radio circuits.

IMPLEMENTATION OF THE AGREEMENT

This STANAG is implemented by a nation when data communication on maritime ship-to-shore HF radio circuits complies with the characteristics detailed in this agreement.

DEFINITIONS

node	An implementation of the profile described in the main body of and mandatory annexes to this STANAG. The node is generally assumed to include the HF (modem and radio) and cryptographic equipment required for communications.
profile	A document describing a set of functions (some or all of which may be defined in separate documents or standards), segregated logically into layers, together with the interfaces, data formats, and procedures required for interoperability.
subnetwork	A collection of nodes. As a whole, a subnetwork provides a reliable networked transport service for external users or clients.

1 INTRODUCTION

This document describes a profile for data communication over HF radio. The technical characteristics which are required to ensure interoperability and reliable system operation are described in the main body of and mandatory annexes to the document. Information-only annexes provide information on possible implementation of interfaces and subnetwork clients, and implementation advice based on extensive experience during the development of the protocols.

This document is organised so that the main body gives an overview of the structure of the profile and the capabilities that should be realised when it is implemented. The details of the interfaces, data formats, and procedures are described in a number of mandatory annexes.

The HF profile provides interoperability at the two major interfaces: first, the “common air interface”, describing how information is exchanged between nodes by radio; and second, the non-HF interfaces which allow external users or clients to interact with the subnetwork and with each other over the subnetwork. While physical interfaces are left up to the system implementer (i.e., Ethernet, FDDI, internal bus), data formats (primitives) and procedures are defined so that client applications can make use of the subnet.

1.1 Common Air Interface: Reliable Data Communications over HF Radio

Reliable data communications over HF radio is provided by using modern, equalized single-tone HF data modems[1] in combination with an ARQ data link protocol.

The data transfer sublayer defined in the profile supports automatically changing the user data rate (that is, code rate) of the HF modem in response to changing channel conditions (adaptive data rate). This capability requires remote control of the HF modem. The profile is defined so that nodes in which remote control of the modem, and hence adaptive data rate, is not available will interoperate with nodes which do have the capability. The profile includes adaptive data rate with STANAG 4285 and STANAG 4529 waveforms.

The HF radio associated with a node is assumed to be an HF SSB radio with specifications appropriate to the modems mentioned above. The profile defined here does not assume that any remote control of the radio (specifically, frequency) is available.

[1]Although this document could be used with other HF data modems, it has been developed for use with the MIL-STD-188-110A single-tone, STANAG 4285, and STANAG 4529 waveforms.

1.2 Interoperability Into the Subnetwork

The profile also defines the interface between a node and external users of the subnetwork. While physical interfaces are left up to the system implementer (ie, Ethernet, FDDI, internal bus), data formats (primitives) and procedures are defined so that external applications can make use of the subnet There are two reasons for including these definitions as a mandatory part of the STANAG: first, with a standard interface definition, any vendor can develop an application which makes use of the subnetwork; and second, without such a definition, interoperability is only guaranteed between nodes implemented by the same vendor. Annex F to this STANAG defines the data formats and procedures which will allow interoperation for a limited subset of applications. Other vendors may define applications which make use of other procedures; while they should be able to make use of the subnet to communicate with another application of the same type, there is no guarantee of interoperability between vendors. The application itself is not defined in this document.

For the purposes of clearer discussion, this document divides the functions of the HF profile into a number of sublayers as shown in figure 1. These sublayers contain the functions which, in OSI terms, would be found in the Physical and Link Layers (with a few network layer functions). Figure 1 contains, for completeness, a number of sublayers (shaded) which are not addressed in this document.

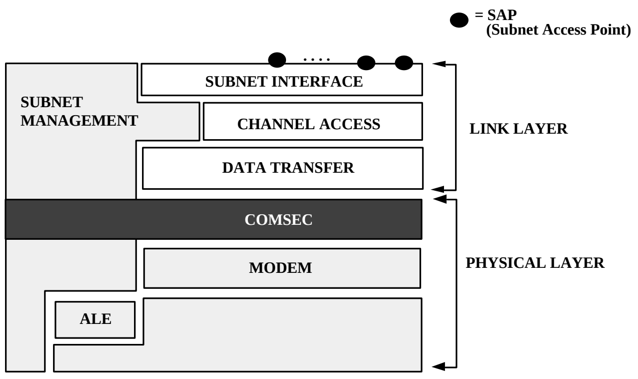


Figure 1. Sublayers within the Profile for Maritime HF Data Communication

1.3 Communications between adjacent sublayers and peer sublayers

Communications between adjacent sublayers within a node is done with “primitives”. Primitives at certain sublayer interfaces must be defined to achieve interoperability; the main example is primitives entering the system at the subnet interface. Other primitives which are not required for interoperability are not defined here, but are discussed in detail in [1].

Communications between a sublayer and the corresponding sublayer in a different node is done with Protocol Data Units (PDUs). For interoperability, PDUs at all sublayers must be defined, together with the protocols for their use. These definitions are given in the annexes to this document.

A brief description of the functions associated with each sublayer follows:

The **Subnetwork Interface Sublayer** provides a common, standard interface to all users. This is the interface between the subnet and the “rest of the world”. Annex A contains a detailed definition of primitives which are exchanged between this sublayer and external users (clients), and PDUs which are exchanged (over HF radio) between peer subnetwork interface sublayers.

Annex F contains a definition of how the primitives can be used to support specific applications. This is provided so that a subset of applications will be interoperable across vendors.

The **Channel Access Sublayer** provides additional functionality as needed to allow different forms of channel access. For the purposes of this document, this sublayer supports communication over a “dedicated” HF radio channel, under the assumption that the processes required to place the two ends of the link on the proper channel are handled by procedures which are external to this system (mechanisms for handling unintentional interference are provided). The PDUs exchanged (over HF radio) between peer sublayers are defined in Annex B of this document. The primitives exchanged between this sublayer and adjacent sublayers within a node are not defined.

The **Data Transfer Sublayer** contains the data transfer protocol. This protocol provides a reliable (ARQ) data link service. The PDUs exchanged (over HF radio) between peer sublayers are defined in Annex C of this document. The primitives exchanged between this sublayer and adjacent sublayers within a node are not defined. Annex C also contains the protocol for adaptive control of the HF modem data rate when using the STANAG 4285 or 4529 waveforms. The interface between this sublayer and the sublayers below (either communications security sublayer or modem sublayer) is defined in Annex D to this document.

The **Communications Security Sublayer** provides communications security using hardware crypto equipment. A number of NATO approved cryptos, including BID-950 and KG-84C, have been shown to be suitable to provide this function; detailed information may be found in [2].

The **Modem Sublayer** provides a means for transmitting digital data over an analogue channel. This STANAG has been developed specifically for use with HF modems defined in STANAG 4285 and STANAG 4529. The interface between this sublayer and radio equipment is not covered in this document for two reasons: first, it is outside the scope of this document, and second, it is covered in other STANAGs. Current trends in system security indicate that encryption will, in the future, be implemented at or near the application layer. If this change occurs while this STANAG is still in service, the interface between the Data Transfer and modem sublayers shall be as defined in Annex D. Annex E contains a recommended outline of a modem remote control interface. This is provided to allow migration toward a more flexible system architecture, in which systems would not be specific to a single vendor’s HF modem.

The **Automatic Link Establishment Sublayer** automates the process of establishing a radio path (link) with one or more remote nodes. This sublayer is not addressed in this document. The system as it is defined is fully compatible with the use of ALE, so if in the future ALE is added, no changes to the other sublayers will be required and only minor changes to the implementation will be required (assuming that implementors follow a layered approach in implementing the profile).

The **Radio Equipment Sublayer** comprises the equipment required to establish a radio link between two or more nodes, i.e. transmitters, receivers, transceivers, antennas, etc. This sublayer is not defined in this document. A NATO STANAG (STANAG 4203) exists which specifies minimum standards for transmitters and receivers.

The **Subnet Management Sublayer** is shown in Figure 1 as a vertical column with interfaces to each sublayer. The main subnet management function, in the context of this STANAG, is automatic link maintenance (ALM) in the form of adaptive control of the HF modem. The management sublayer messages and associated procedures which are required for ALM are defined in Annex C of this document, in the context of the MANAGEMENT D_PDU. The other functions of the Subnet Management Sublayer, which may be critically important to a successful implementation, need not be standardized for interoperability and are not addressed further in this document.

~~Additional~~List of Annexes

Annex A: Subnetwork Interface Sublayer (mandatory)
Annex B: Channel Access Sublayer (mandatory)
Annex C: Data Transfer Sublayer (mandatory)
Annex D: Interface between Data Transfer Sublayer and Communications Equipment (mandatory)
Annex E: HF Modem Remote Control Interface (information only)
Annex F: Subnetwork Client Definitions (information only)
Annex G: Waveforms for Data Rates Above 2400 BPS (information only)
Annex H: Implementation Guide and Notes (information only)
Annex I: Messages and Procedures for Frequency Change (information only)

~~Annex G provides a definition of waveforms which will provide data rates above 2400 bps. The mandatory annexes are written to support data rates up to 9600 bps, however no standards currently exist for HF modems at these data rates. Annex H provides implementation guidance and notes.~~

References

1. Clark, D., and N. Karavassillis, “Open Systems for Radio Communications: A Subnet Architecture for Data Transmission over HF Radio”, TM-9~~40~~37 (in preparation)

2. Miller, T., and P. Reynolds, “Experience with Approved Cryptographic Equipment in HF ARQ Systems”, NC3A TN 638, NATO CONFIDENTIAL, November 1996

Annex A: Subnetwork Interface Sublayer (mandatory)

This annex defines the interface between the users of the HF subnetwork and the node through which the user accesses the subnetwork.

A.1 Subnetwork Service Definition

The interaction between the HF subnetwork and the users of the subnetwork are governed by a Client-Server relation. The users (clients) request the services provided by the HF subnetwork (server). The service provided by the server is application independent and common to all clients irrespective of the task they may perform.

Clients are attached to the Subnetwork Interface Sublayer at Subnetwork Access Points (SAPs). There can be multiple clients simultaneously attached to the Subnetwork Interface Sublayer. Each SAP is identified by its SAP Identifier (SAP ID).[2] The SAP ID is a number in the range 0-15; hence there can be a maximum of 16 clients attached to the Subnetwork Interface Sublayer of a single node.

Annex F contains a recommended definition of the various subnetwork clients. For the purposes of this STANAG, the subnetwork client definitions in Annex F are not mandatory. Data submitted by the clients to the Subnetwork Interface Sublayer must be in the form of primitives with the format as described in this document. Clients are responsible for segmenting larger messages into U_PDUs. A U_PDU format which supports this segmentation is defined in Annex F.

The Subnetwork Interface Sublayer treats all clients connected to it in the same manner irrespective of the application performed by these clients. The only distinguishing factor between clients is their **Rank** which is a measure of their importance. See Annex H.5 for further information on the rank of clients. Certain service requests made by higher ranked clients may take precedence over requests made by lower ranked clients.

A.1.1 Initiating Data Exchange Sessions

The Subnetwork Interface Sublayer is responsible for initiating the establishment and termination of Sessions with its peers at remote nodes. There are four types of sessions:

1. Soft Link Data Exchange Session

2. Hard Link Data Exchange Session

3. Broadcast Data Exchange Session

4. Reserved

All sessions apart form the broadcast data exchange session require the making of a point-to-point physical link with a specified remote node.

A.1.1.1 Soft Link Data Exchange Session

[2] SAPs are equivalent to the "ports" of the TCP protocol.

The establishment of a Soft Link Data Exchange Session is initiated unilaterally by the Subnetwork Interface Sublayer which has queued ARQ U_PDUs and from which a client has not requested a Hard Link Data Exchange Soft Link Data Exchange Sessions shall be initiated as needed, following the procedure described in Section A.3.2.1. When the data for this node has been transmitted, the session is terminated immediately or after a suitable time-out period (the time out period would allow for the possibility of new U_PDUs which can be serviced by the existing soft link session). The procedure for terminating a Soft Link Data Exchange Session is described in Section A.3.2.1.2. In order to provide “balanced” servicing of the queued U_PDUs, a Soft Link Data Exchange Session should not be maintained for a period which exceeds a specified maximum time if U_PDUs of appropriate priorities are queued for different node(s). The specific values of the parameters governing the establishment and termination of Soft Link Data Exchange Sessions (e.g. time-out periods etc.) must be chosen in the context of a particular configuration (i.e. size of network, etc).

A.1.1.2 Hard Link Data Exchange Session

The second type of data exchange session is the Hard Link Data Exchange Session. This session is initiated at the explicit request of a client. A client can request a hard link in order to ensure that a physical link to a specified node is maintained (irrespective of the destinations of other queued U_PDUs) and optionally to partially or fully reserve the capacity of this link. The procedures for establishing and terminating hard link sessions are described in Sections A.3.2.2.1 and A.3.2.2.2.

A.1.1.3 Broadcast Data Exchange Session

- The third type of data exchange session is the Broadcast Data Exchange Session. During this session only non-ARQ U_PDUs can be serviced. A Broadcast Data Exchange Session can be initiated and terminated by a management process, e.g.. a local or network administrator management client. A node configured to be a broadcast-only node is put in a “permanent” Broadcast Data Exchange Session during which no hard link requests or ARQ Data U_PDUs can be serviced. Alternatively the Subnetwork Interface Sublayer can unilaterally initiate and terminate Broadcast Data Exchange Sessions. The rules for initiating and terminating broadcast data exchange sessions are in Annex C.

A.2 Primitives Exchanged with Clients

Table A-1 lists the primitives exchanged between clients and the Subnetwork Interface Sublayer. The names of these primitives are prefixed with an “S_” to indicate that they are exchanged across the interface between the subnetwork interface sublayer and the subnetwork clients. This table is intended to provide a general guide and overview to the primitives. For detailed specification of the primitives, refer to Figure A-1.

Table A-1. Primitives Exchanged with Clients

CLIENT -> SUBNETWORK INTERFACE	SUBNETWORK INTERFACE -> CLIENT
S_BIND_REQUEST (Service Type, Rank, SAP ID)	S_BIND_ACCEPTED (SAP ID, MTU)
	S_BIND_REJECTED (Reason)
S_UNBIND_REQUEST ( )	S_UNBIND_INDICATION (Reason)
S_HARD_LINK_ESTABLISH (Link Priority, Link Type, Remote Node Address, Remote SAP ID)	S_HARD_LINK_ESTABLISHED (Remote Node Status, Link Priority, Link Type, Remote Node Address, Remote SAP ID)
	S_HARD_LINK_REJECTED (Reason, Link Priority, Link Type, Remote Node Address, Remote SAP ID)
S_HARD_LINK_ACCEPT (Link Priority, Link Type, Remote Node Address, Remote SAP ID)	S_HARD_LINK_INDICATION (Remote Node Status, Link Priority, Link Type, Remote Node Address, Remote SAP ID)
S_HARD_LINK_REJECT (Reason, Link Priority, Link Type, Remote Node Address, Remote SAP ID)	
S_HARD_LINK_TERMINATE (Remote Node Address)	S_HARD_LINK_TERMINATED (Reason, Link Priority, Link Type, Remote Node Address, Remote SAP ID)
	S_SUBNET_AVAILABILITY (Subnet Status, Reason)
S_UNIDATA_REQUEST (Destination Node Address, Destination SAP ID, Priority, TimeToLive, Delivery Mode, U_PDU)	S_UNIDATA_REQUEST_CONFIRM (Destination Node Address, Destination SAP ID, U_PDU)
	S_UNIDATA_REQUEST_REJECTED (Reason, Destination Node Address, Destination SAP ID, U_PDU)
	S_UNIDATA_INDICATION (Source Node Address, Source SAP ID, Destination Node Address, Destination SAP ID, Priority, U_PDU)
S_EXPEDITED_UNIDATA_REQUEST (Destination Node Address, Destination SAP ID, ~~Priority,~~ TimeToLive, Delivery Mode, U_PDU)	S_EXPEDITED_UNIDATA_REQUEST_CONFIRM (Destination Node Address, Destination SAP ID, U_PDU)
	S_ EXPEDITED_UNIDATA_REQUEST_REJECTED (Reason, Destination Node Address, Destination SAP ID, U_PDU)
	S_ EXPEDITED_UNIDATA_INDICATION (Source Node Address, Source SAP ID,

	Destination Node Address, Destination SAP ID, ~~Priority,~~ U_PDU)
	S_DATA_FLOW_ON( )
	S_DATA_FLOW_OFF ( )
S_MANAGEMNT _MSG_REQUEST (MSG)	S_MANAGEMENT_MSG_INDICATION (MSG)
S_KEEP_ALIVE ( )	S_KEEP_ALIVE ( )

The description of the primitives follows:

Name :
S_BIND_REQUEST ( )
Arguments :
1. Service Type,
2. RANK,
3. SAP ID

Direction :
Client -> Subnetwork Interface
Description :
This primitive must be issued by a new client when it first connects to the subnetwork. Unless this primitive is issued the client can not be serviced. With this primitive the client uniquely identifies itself and declares that it is “on-line” and ready to be serviced by the subnetwork. The first argument of this primitive is “*Service Type*” and identifies the default type of service requested by the client. This service type will apply to all data units submitted by the client unless explicitly overridden (see S_UNIDATA_REQUEST primitive). The “*Service Type*” argument is a complex argument and has a number of attributes. The second argument of this primitive is “Rank”. This is a measure of the importance of a client; itis used by the subnetwork to allocate resources. A detailed description of the Service Type argument is given at the end of this section. A description of the use of the Rank argument may be found in Annex H and [1]. The range of possible rank arguments is from 0 to 15. Clients which are not authorised to make changes to a node or subnetwork configuration shall not bind with rank of 15. The third argument of this primitive is the “*SAP ID”* which the client wishes to be assigned. The SAP ID must be node level unique, i.e. not assigned to another connected client.

Name :
S_UNBIND_REQUEST ( )
Arguments :
NONE
Direction :
client -> Subnetwork Interface ( )
Description :
This primitive is issued by a client in order to declare itself as “off-line”. Going off-line results in the release of the SAP ID of this client which can be allocated to another client that may request it. A client which went off-line by issuing this primitive can come on-line again by issuing a new S_BIND_REQUEST. A client can also go off-line by physically disconnecting itself (e.g. powering down the computer which runs the client program) or disconnecting the physical cable (RS232, Ethernet,

etc.) which may connect the client to the node. The Subnetwork Interface Sublayer must be able to sense whether a client is physically disconnected in order to unilaterally declare this client as off-line; this capability is provided by the S_KEEP_ALIVE and S_KEEP_ALIVE_RESPONSE primitives.
[The omission of SAP ID as an argument in this and other primitives implies a requirement on the stack supporting this connection to associate a SAP ID with a lower level connection (ie, socket) and maintain this association.]

Name :
S_BIND_ACCEPTED ( )
Arguments :
1. SAP ID
2. Maximum Transmission Unit (MTU)

Direction :
Subnetwork Interface -> Client
Description :
This primitive is issued by the Subnetwork Interface Sublayer as a positive response to a client's S_BIND_REQUEST. If the arguments of S_BIND_REQUEST are valid (e.g. the SAP ID requested by the client is not already allocated and conforms to the rules of valid SAP IDs), and certain conditions are met (e.g. maximum number of connected clients is not exceeded, sufficient Subnet Access Sublayer resources are available), then the client is accepted as on-line by the HF Subnetwork. The *SAP ID* argument is the SAP ID assigned to the client and it is a copy of the *SAP ID* argument of the S_BIND_REQUEST. The MTU argument allows the subnetwork interface to inform the client of the maximum size U_PDU (in bytes or octets) which will be accepted as an argument of the S_UNIDATA_REQUEST primitive. S_UNIDATA_REQUEST primitives containing U_PDUs larger than the MTU will be rejected by the subnetwork interface. Note that this restriction applies only to U_PDUs received through the subnetwork interface. U_PDUs which are received from the lower HF sublayers (i.e., received by radio) shall be delivered to clients regardless of size.

For general purpose nodes, the MTU shall be 2048 bytes. For broadcast-only nodes, the MTU may be set by the implementation up to a maximum of 4096 bytes.

Name :
S_BIND_REJECTED ( )
Arguments :
1. Reason

Direction :
Subnetwork Interface -> Client
Description :
This primitive is issued by the Subnetwork Interface Sublayer as a negative response to a client's S_BIND_REQUEST. If certain conditions are not met then the Subnetwork Interface Sublayer rejects the client's request. The *Reason* argument explains why the client's request was rejected. Possible *Reason* values are given in the table. The binary representation of the value in the table shall be mapped into the Reason field of the primitive by placing the LSB of the value into the LSB of the field.

Reason	Value
Not Enough Resources	1
Invalid SAP ID	2
SAP ID already allocated	3

Name :
S_UNBIND_INDICATION ( )
Arguments :
1. Reason

Direction :
Subnetwork Interface->Client
Description :
This primitive is issued by the Subnetwork Interface Sublayer to unilaterally declare a client as off-line. If the client wants to come on-line again, it must issue a new a S_BIND_REQUEST primitive. The S_UNBIND_INDICATION primitive provides a means for the Subnetwork Interface Sublayer to manage the clients connected to it. As an example, if a new “High Ranked” client requests to come on-line but not enough resources are available, the Subnetwork Interface Sublayer may decide to unilaterally declare a “Lower Ranked” client off-line in order to release resources for the Higher-Ranked client. The *Reason* argument explains why the client was declared off-line. The binary representation of the value in the table shall be mapped into the Reason field of the primitive by placing the LSB of the value into the LSB of the field.

Reason	Value
Connection pre-empted by higher ranked client	1
Inactivity (failure to respond to “Keep alive”)	2
Too many invalid primitives	3
Too many expedited data request primitives	4

Name :
S_UNIDATA_REQUEST
Arguments :
1. Destination Node Address
2. Destination SAP ID
3. Priority
4. TimeToLive (TTL)
5. Delivery Mode
6. U_PDU (User Protocol Data Unit)

Direction :
Client->Subnet Interface
Description :
This primitive is used by connected clients to submit a U_PDU to the HF subnetwork for delivery to a receiving client. The argument *Destination Node Address* specifies the address of the physical HF node to which the receiving client is bound. The argument *Destination SAP ID* specifies the SAP ID of the receiving client and

distinguishes it from the other clients bound to the destination node. The argument *Priority* represents the priority of the U_PDU. The U_PDU priority can take a value in the range 0-15. The node shall make a “best effort” to give precedence to high priority U_PDUs over lower priority U_PDUs which are queued in the system. The argument *TimeToLive (TTL)* specifies the maximum amount of time the submitted U_PDU is allowed to stay in the HF Subnetwork before it is delivered to its final destination. If the TTL is exceeded the U_PDU shall be discarded. A TTL value of 0 is defined to mean infinite TTL**,** i.e. the subnetwork should try *forever* to deliver the U_PDU. The subnetwork should have a default maximum TTL which will normally be a very large value. As soon as the Subnetwork Interface Sublayer accepts a S_UNIDATA_REQUEST primitive, it immediately calculates its *TimeToDie (TTD)* by adding the specified TTL (or the default maximum value if the specified TTL is equal to 0) to the current Time of Day, e.g. GMT. The TTD attribute of a U_PDU accompanies it during its trip within the subnetwork. It should be noted that the TTD is an absolute time while the TTL is a time interval relative to the instant of the U_PDU submission. The argument *Delivery Mode* is a complex argument and has a number of attributes. This argument can be given the value of “Default” which means that the delivery mode associated with the U_PDU will be the delivery mode specified by the client during “binding” (*Service Type* argument of S_BIND_REQUEST). The other values of the *Delivery Mode* can be used to override the default delivery mode for this U_PDU. A detailed discussion of this argument is provided at the end of this section. The final argument, *U_PDU,* is the actual Data Unit submitted by the client to the HF Subnetwork.

Name :
S_UNIDATA_REQUEST_CONFIRM
Arguments :
1. Destination Node Address
2. Destination SAP ID
3. U_PDU (User Protocol Data Unit or part of it)

Direction :
Subnetwork Interface->Client
Description :
This primitive is issued by the Subnetwork Interface Sublayer to acknowledge the successful delivery of a S_UNIDATA_REQUEST submitted by the client. This primitive is issued only if the client has requested Data Delivery Confirmation (either during binding or for this particular data unit). The three arguments of this primitive are copies of the corresponding arguments of the S_UNIDATA_REQUEST which caused the S_UNIDATA_REQUEST_CONFIRM. Using these arguments, the client should be able to uniquely identify the U_PDU which is being acknowledged. Depending on the implementation, the last argument, *U_PDU,* may not be a complete copy of the original U_PDU but only a partial copy, e.g. only the first x bytes are copied. If a partial U_PDU is returned, the first x bytes shall be returned so that the client will have the U_PDU segment information.

Name :
S_UNIDATA_REQUEST_REJECTED
Arguments :
1. Reason
2. Destination Node Address
3. Destination SAP ID
4. U_PDU (User Protocol Data Unit or part of it)

Direction :

Subnetwork Interface->client

Description :

This primitive is issued by the Subnetwork Interface Sublayer to inform a client that a S_UNIDATA_REQUEST was not delivered successfully. This primitive is issued if the client has requested Data Delivery Confirmation (either during Binding or for this particular U_PDU), or if a U_PDU larger than the MTU is submitted. The argument *Reason* explains why the delivery failed; valid reasons are given in the table below. The rest of the arguments are copied from the corresponding arguments of the S_UNIDATA_REQUEST which caused the S_UNIDATA_REQUEST_REJECTED. As explained in the S_UNIDATA_REQUEST_CONFIRM primitive, the *U_PDU* may only be a partial copy of the original U_PDU. The binary representation of the value in the table below shall be mapped into the Reason field of the primitive by placing the LSB of the value into the LSB of the field.

Reason	Value
TTL Expired	1
Destination SAP ID not bound	2
Destination node not responding	3
U_PDU larger than MTU	4
Tx Mode not specified	5

Name :

S_UNIDATA_INDICATION

Arguments :

1. Source Node Address
2. Source SAP ID
3. Destination Node Address
4. Destination SAP ID
5. Priority
6. Transmission Mode
7. U_PDU

Direction :

Subnetwork Interface->client

Description :

This primitive is used by the Subnetwork Interface Sublayer to deliver a received U_PDU to the client. The first two arguments, *Source Node Address* and *Source SAP ID* represent the node sddress and the SAP ID of the client which sent the U_PDU. The *Destination Node Address* argument is normally the address of the local (receiving) node. It can however be a “group” address to which the local node has subscribed (Group Adresses and their subscribers are defined during configuration). The *Destination SAP ID* argument is normally the SAP ID of the client to which this primtive is delivered. The *Transmisson Mode* argument represents the mode at which the U_PDU was transmitted by the remote node and received by the local node; ie, ARQ, Non-ARQ (Broadcast) transmission, etc. The *Priority* argument represents the priority of the PDU, while the *U_PDU* argument represents the actual received user data.

Name :

S_EXPEDITED_UNIDATA_REQUEST

Arguments :

1. Destination Node Address
2. Destination SAP ID
3. TimeToLive (TTL)
4. Delivery Mode
5. U_PDU (User Protocol Data Unit)

Direction :

Client->Subnet Interface

Description :

This primitive is used to submit a U_PDU to the HF Subnetwork for Expedited Delivery to a receiving client. The five arguments of this primitive have the same meaning as the corresponding arguments of the S_UNIDATA_REQUEST primitive. There is no *Priority* argument in the S_EXPEDITED_UNIDATA_REQUEST primitive. Although seemingly equivalent, there is a important difference between a S_UNIDATA_REQUEST primitive of the highest priority and a S_EXPEDITED_UNIDATA_REQUEST primitive. S_UNIDATA_REQUEST primitives of all priority levels are processed according to a set of rules that apply to Normal Data. U_PDUs submitted using S_EXPEDITED_UNIDATA_REQUEST primitives are treated differently, e.g. expedited U_PDUs are queued separately from normal U_PDUs. When an expedited U_PDU is received, the transmission of normal data is halted and the expedited data is transmitted. When the expedited data has been sent, the transmission of normal data is resumed again.

The 5066 node management should track the submission of S_EXPEDITED_UNIDATA_REQUEST primitives by various clients. If a client submits too many S_EXPEDITED_UNIDATA_REQUEST primitives, the node management should unilaterally disconnect the client (using a S_UNBIND_INDICATION with reason 4). What is “too many” remains an implementation decision.

Name :

S_EXPEDITED_UNIDATA_REQUEST_CONFIRM

Arguments :

1. Destination Node Address
2. Destination SAP ID
3. U_PDU (User Protocol Data Unit or part of it)

Direction :

Subnetwork Interface->Client

Description :

This primitive is issued by the Subnetwork Interface Sublayer to acknowledge the successful delivery of a S_EXPEDITED_UNIDATA_REQUEST primitive. This primitive is issued only if the client has requested Data Delivery Confirmation (either during Binding or for this particular U_PDU).The three arguments of this primitive are copies of the corresponding arguments of the S_EXPEDITED_UNIDATA_REQUEST primitive which caused the S_EXPEDITED_UNIDATA_REQUEST_CONFIRM. Using these arguments, the client should be able to identify the U_PDU which is being acknowledged. Depending on the implementation, the last argument *U_PDU* may not be a complete copy of the original U_PDU but only a partial copy.

Name :

S_EXPEDITED_UNIDATA_REQUEST_REJECTED

Arguments :

1. Reason
2. Destination Node Address
3. Destination SAP ID
4. U_PDU (User Protocol Data Unit or part of it)

Direction :

Subnetwork Interface->Client

Description :

This primitive is issued by the Subnetwork Interface Sublayer to inform a client that a S_EXPEDITED_UNIDATA_REQUEST was not delivered successfully. This primitive is issued if the client has requested Data Delivery Confirmation (either during Binding or for this particular U_PDU), or if a U_PDU larger than the MTU is submitted. The argument *Reason* explains why the delivery failed; values defined for this field are given in the table below. The remaining arguments are copied from the corresponding arguments of the S_EXPEDITED_UNIDATA_REQUEST which caused the S_EXPEDITED_UNIDATA_REQUEST_REJECTED.

Reason	Value
TTL Expired	1
Destination SAP ID not bound	2
Destination node not responding	3
U_PDU larger than MTU	4

Name :

S_EXPEDITED_UNIDATA_INDICATION

Arguments :

1. Source Node Address
2. Source SAP ID
3. Destination Node Address
4. Destination SAP ID
5. Transmission Mode
6. U_PDU

Direction :

Subnetwork Interface->Client

Description :

This primitive is used by the Subnetwork Interface Sublayer to deliver a received Expedited U_PDU to a client. The format of this primitive is similar to that of the S_UNIDATA_INDICATION, with the only difference that there is no *Priority* argument.

Name :

S_DATA_FLOW_ON
S_DATA_FLOW_OFF

Arguments :

NONE

Direction :

Subnetwork Interface-> Client

Description :

These two primitives are issued by the Subnetwork Interface Sublayer to flow control the transfer of U_PDUs submitted by a client (Depending on the implementation, the physical connection between the client(s) and the Subnetwork Interface Sublayer may provide an implicit flow control mechanism. For example, if the connection is implemented as TCP/IP Berkeley Sockets, the implicit flow control mechanism of the TCP protocol may be utilised in which case these two primitives are redundant). The Subnetwork Interface Sublayer can use these two primitives (or other mechanisms) to control the flow of data from locally attached clients. A client shall not control the flow of data *from* the subnetwork by any mechanism, explicit or implicit. All clients shall be ready to accept at all times data received by the HF Node to which it is connected; clients not following this rule may be disconnected by the node.

Name :
S_MANAGEMENT_MSG_REQUEST
Arguments :
1. MSG

Direction :
Client-> Subnet Interface
Description :
This primitive is issued by a client to submit a "Management" message to the Subnetwork. Depending on the value of the complex argument *MSG*, this primitive can take the form of a Command (e.g. Go-To-EMCON, Go-Off-Air, etc.) or of a Request (e.g. Request-For-Subnetwork-Statistics, Request-For-Connected-client-Information, etc.). The subnetwork shall accept the command form of this primitive only from clients which have bound with a rank of 15. The contents of this message are not yet defined.

Note that this primitive is not intended to allow for the transmission of management coordination messages over the air. This is an interaction between peer subnet management clients and as such shall be accomplished using the UNIDATA primitives defined elsewhere in this annex.

Name :
S_MANAGEMENT_MSG_INDICATION
Arguments :
1. MSG

Direction :
Subnetwork Interface-> Client
Description :
This primitive is issued by the Subnetwork to send a "Management" message to a client. The complex argument *MSG* can take several values such as : Subnetwork-Statistics, Connected-client-Information, etc. This primitive can be issued either in response to a S_MANAGEMENT_MSG_REQUEST or asynchronously by the Subnetwork. The contents of this message are not yet defined.

Name :
S_KEEP_ALIVE
Arguments :
NONE

Direction :
Client-> Subnetwork Interface
Subnetwork Interface-> Client

Description :
This primitive can be issued as required (e.g. during periods of inactivity) by the clients and/or the Subnetwork Interface to sense whether the physical connection between the client and the Subnetwork is alive or broken. This primitive may be redundant if the implementation of the physical connection provides an implicit mechanism for sensing the status of the connection. When this primitive is received, the client or node shall respond with the same primitive within 10 seconds. If a reply is not sent within 10 seconds, no reply shall be sent. A client or node shall not send this primitive more frequently than once every 120 seconds to the same destination.

Name :
S_HARD_LINK_ESTABLISH

Arguments :
1. Link Priority
2. Link Type
3. Remote Node Address
4. Remote SAP ID

Direction :
Client-> Subnetwork Interface

Description :
This primitive is used by a client to request the establishment of a Hard Link between the local Node to which it is connected and a specified remote Node. A Hard Link can only be terminated by the client who requested it, or by another client on either end of the Link by requesting a Hard Link of a higher priority than the existing one. Physical Links between Nodes are normally made and broken unilaterally by the HF subnetwork according to the destinations of the queued U_PDUs. Such links are classified as Soft Links. The S_HARD_LINK_ESTABLISH primitive, allows a client to override these procedures and request a Physical Link to be made to a specific Node and be maintained until the client decides to break it. The argument *Link Priority* defines the priority of the Link. It can take a value in the range 0-3. Hard Link requests made by higher Ranked clients shall take precedence over requests of lower Ranked clients regardless of the value of the *Link Priority* argument. The *Link Type* argument can be used by the requesting client to fully or partially reserve the bandwidth of the Link. It can take a value in the range 0-2. The *Link Type* argument is further discussed at the end of this section. The *Remote Node Address* argument specifies the physical HF Node Address to which a connection must be established and maintained. The *Remote SAP ID* argument identifies the single client connected to the remote Node, to and from which traffic is allowed. This argument is only valid if the *Link Type* argument has a value of 2 (see discussion at the end of this section for clarification).

Name :
S_HARD_LINK_TERMINATE

Arguments :
1. Remote Node Address

Direction :
Client-> Subnetwork Interface

Description :

This primitive is issued by a client to terminate an existing Hard Link. Only the client which requested the establishment of the Hard Link can terminate it using this primitive. The single argument, *Remote Node Address* specifies the Address of the Node at the remote end of the Link. This argument is redundant in that only one Hard Link can exist at any time. It may however be used for checking the validity of the primitive. Upon receiving this primitive, the node will take all necessary steps to terminate the Hard Link[3]. This primitive is always accepted, so there is no need for the Subnetwork Interface Sublayer to issue a primitive confirming the successful termination of the Link.

Name :

S_HARD_LINK_ESTABLISHED

Arguments :

1. Remote Node Status
2. Link Priority
3. Link Type
4. Remote Node Address
5. Remote SAP ID

Direction :

Subnetwork Interface-> Client

Description:

This primitive is issued by the Subnetwork Interface Sublayer as a positive response to a client's S_HARD_LINK_ESTABLISH primitive. This primitive is issued only after all the negotiations between the appropriate peer sublayers of the two nodes have been completed and the remote node has accepted the establishment of the Hard Link.

The first argument, *Remote Node Status*, informs the requesting client of any special status of the remote node, e.g. Remote Node in EMCON, etc. Valid arguments for *Remote Node Status* are given in the table below. The remaining arguments are copies of the corresponding arguments of the S_HARD_LINK_ESTABLISH primitive. Until additional values for Remote Node Status have been defined, successful establishment of a Hard Link shall imply a status of "OK" for the remote node; this shall be indicated by the value 1 in this field.

Name :

S_HARD_LINK_REJECTED

Arguments :

1. reserved for Reason
2. Link Priority
3. Link Type
4. Remote Node Address
5. Remote SAP ID

Direction :

Subnetwork Interface-> Client

Description:

This primitive is issued by the Subnetwork Interface Sublayer as a negative response to a client's S_HARD_LINK_ESTABLISH primitive. The *Reason* argument explains

[3] The Link can be terminated immediately or in a "graceful" manner according to the requirements of a specific application. A graceful termination might, for example, allow completion of the current transmission interval before the link is broken and/or allow transmission of queued high priority U-PDUs from other clients to the same destination to be transmitted before the link is terminated.

why the Hard Link Request was rejected; values defined for this field are given in the table below. ~~Possible values of the *Reason* argument are: Remote-Node-Busy~~, ~~Higher-Priority-Link-Existing~~, ~~Remote-Node-Not-Responding, etc. The values for the Reason argument have not yet been defined. Until they are defined, the client shall ignore this field.~~ The remaining arguments are copies of the corresponding arguments of the S_HARD_LINK_ESTABLISH primitive.

Reason	Value
Remote-Node-Busy	1
Higher-Priority-Link-Existing	2
Remote-Node-Not-Responding	3
Destination SAP ID not bound	4

Name :

S_HARD_LINK_TERMINATED

Arguments :

1. Reason
2. Link Priority
3. Link Type
4. Remote Node Address
5. Remote SAP ID

Direction :

Subnetwork Interface-> Client

Description:

This primitive is issued by the Subnetwork Interface Sublayer to inform a client which has been granted a Hard Link that the Link has been terminated unilaterally by the Subnetwork. ~~O~~For hard link types 0 and 1, only the client which originally requested the Hard Link will receive this primitive. For type 2 hard links, both called and calling clients will receive this primitive. The *Reason* argument explains why the Hard Link was terminated; values defined for this field are given in the table below. ~~Possible values of the *Reason* argument are: Remote-Node-Not-Responding-Time-Out, Higher-Priority-Hard-Link-Requested, Remote-Node-Requested-Disconnection, etc. The values for the Reason argument have not yet been defined. Until they are defined, the client shall ignore this field.~~ The remaining arguments are copies of the corresponding arguments of the S_HARD_LINK_ESTABLISH primitive.

Reason	Value
Link terminated by remote node	1
Higher priority link requested	2
Remote node not responding (time out)	3
Destination SAP ID unbound	4

Name :

S_HARD_LINK_INDICATION

Arguments :

1. Remote Node Status
2. Link Priority
3. Link Type
4. Remote Node Address
5. Remote SAP ID

Direction :
Subnetwork Interface-> Client
Description:
This primitive is used only for link types 2 and 3. With this primitive the Subnetwork Interface Sublayer signals to one of his local clients, that a client at a remote node requested a hard link of type 2 or 3 to be established between them. The five arguments of this primitive are similar to those for the S_HARD_LINK_ESTABLISHED primitive, with the difference that the remote node and remote SAP ID refer to the calling node and SAP ID.

Name :
S_HARD_LINK_ACCEPT
Arguments :
1. Link Priority
2. Link Type
3. Remote Node Address
4. Remote SAP ID

Direction :
Client-> Subnetwork Interface
Description:
This primitive is issued by a client as a positive response to a S_HARD_LINK_INDICATION primitive. With this primitive the client tells the Subnetwork Interface Sublayer that it accepts the hard link of type 2 or 3 requested by a client at a remote node. The arguments of this primitive are simply copies of the corresponding arguments of the S_HARD_LINK_INDICATION primitive.

Name :
S_HARD_LINK_REJECT
Arguments :
1. Reason
2. Link Priority
3. Link Type
4. Remote Node Address
5. Remote SAP ID

Direction :
Client-> Subnetwork Interface
Description:
This primitive is issued by a client as a negative response to a S_HARD_LINK_INDICATION primitive. with this primitive the client tells the Subnetwork Interface Sublayer that it rejects the hard link of type 2 or 3 requested by a client at a remote node. The *Reason* argument explains why the hard link is rejected. Possible values of this argument are: Mode-Not-Supported (for Link Type 3), I-Have-Higher-Priority-Data, etc. The rest of the arguments of this primitive are simply copies of the corresponding arguments of the S_HARD_LINK_INDICATION primitive.

Name :
S_SUBNET_AVAILABILITY
Arguments :
1. Node Status
2. Reason

Direction :
Subnetwork Interface-> Client

Description:

This primitive can be sent asynchronously to all or selected clients connected to the Subnetwork Interface Sublayer to inform them of changes in the status of the node to which they are attached. For example, this primitive can be used to inform clients that all available resources (i.e. bandwidth) have been temporarily reserved by a high ranked client or that the node has entered an EMCON state and as a result they should only expect to receive Data but will not be allowed to transmit data. The *Node Status* argument specifies the new Status of the node. Possible values of this argument are: ON, OFF, Receive-Only, Transmit-Only-to-Specific-Destination-Node/SAP, etc. If the Subnetwork Status is other than ON, the *Reason* argument explains why. Possible values of this argument are: Local-Node-In- EMCON, Hard-Link-Requested-By High-Priority-client, etc. The contents of this primitive are not yet defined.

A.2.1 Encoding of Primitives

Figure A-1 (a-h) below shows the encoding of the S_Primitives. . The bits shall be mapped into fields in accordance with CCITT V.42, 8.1.2.3, which states that:

- when a field is contained within a single octet, the lowest bit number of the field represents the lowest-order value
- when a field spans more than one octet, the order of bit values within each octet progressively decreases as the octet number increases. The lowest bit number associated with the field represents the lowest-order value.

The 4 byte address field in the S_primitives shall carry the 3.5 byte address defined in C.3.1.4. The lowest order bit of the address shall be placed in the lowest order bit position of the field (generally bit 0 of the highest byte number of the field), consistent with the mapping in D_PDUs.

As shown in Figure A-1(a), ~~A~~all primitives shall begin with the 16-bit Maury-Styles sequence shown below, with the least significant bit (LSB) transmitted first:

(MSB) 1 1 1 0 1 0 1 1 1 0 0 1 0 0 0 0 (LSB)

Following the Maury-Styles sequence, the next 8 bits shall carry the 5066 version number. For this version of 5066, the version number shall be all zeros.

The next 16 bits shall carry the size in bytes of the S_primitive to follow, exclusive of the Maury-Styles sequence, version field, and this size field. The LSB of the of the size value shall be mapped into the low order bit of the field.

Unless specified otherwise, the order of bit transmission shall be as described in CCITT V.42 paragraph 8.1.2.2, which specifies that the least significant bit (LSB, bit 0 in the figures below) of byte 0 shall be transmitted first. The remaining bytes shall be transmitted sequentially, also beginning with the LSB of each byte.

2 BYTE HEADER	1 BYTE VERSION	2 BYTE SIZE OF PRIMITIVE	S_PRIMITIVE

Figure A-1 (a). Encoding of “S_” Primitives

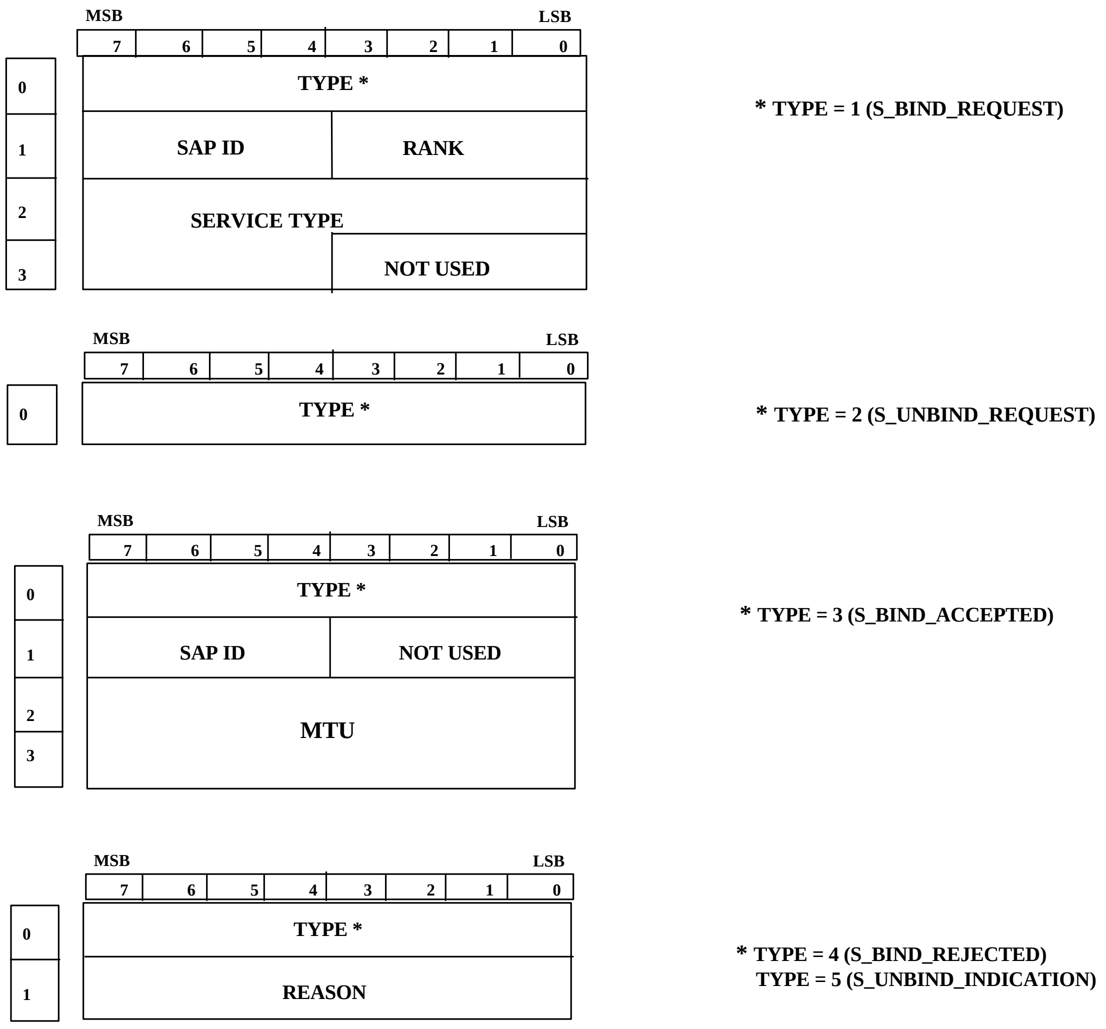
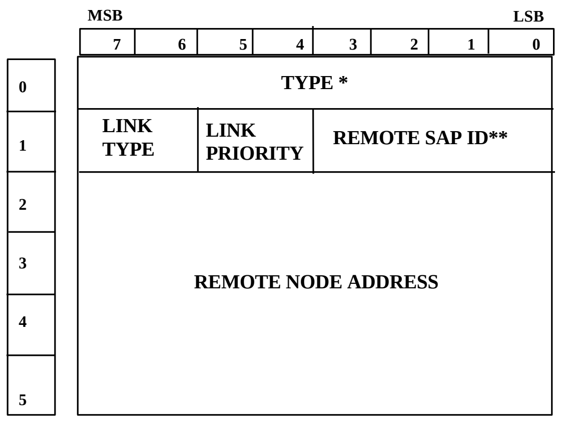
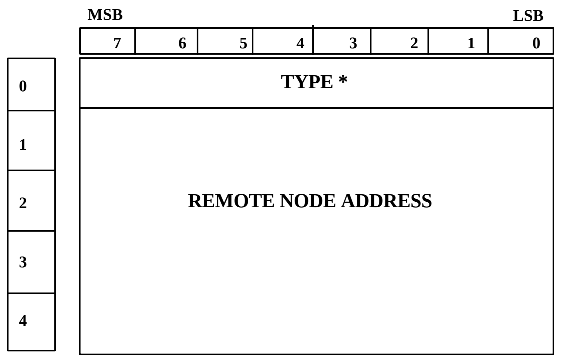

MSB
LSB
7 6 5 4 3 2 1 0
0 1 2 3
TYPE *
SAP ID
RANK
SERVICE TYPE
NOT USED
* TYPE = 1 (S_BIND_REQUEST)
0
TYPE *
* TYPE = 2 (S_UNBIND_REQUEST)
0 1 2 3
TYPE *
SAP ID
NOT USED
MTU
* TYPE = 3 (S_BIND_ACCEPTED)
0 1
TYPE *
REASON
* TYPE = 4 (S_BIND_REJECTED)
TYPE = 5 (S_UNBIND_INDICATION)

Figure A-1 (b): Encoding of “S_” Primitives (Cont.)



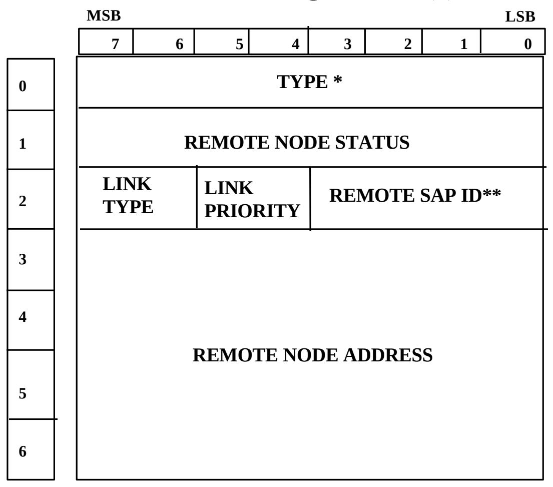
* TYPE = 6 (S_HARD_LINK_ESTABLISH)
TYPE = 12 (S_HARD_LINK_ACCEPT)

**ONLY VALID IF LINK TYPE = 2 or 3



* TYPE = 7 (S_HARD_LINK_TERMINATE)

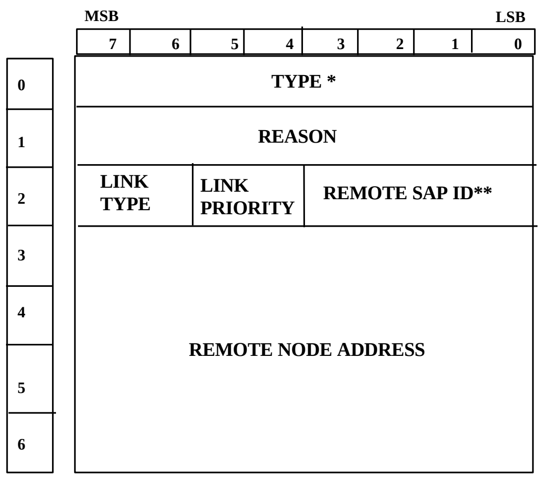
Figure A-1 (c): Encoding of "S_" Primitives



* TYPE = 8 (S_HARD_LINK_ESTABLISHED)
TYPE = 11 (S_HARD_LINK_INDICATION)

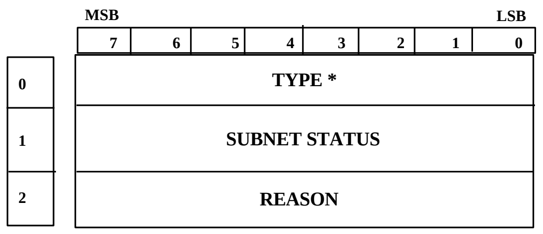
**ONLY VALID IF LINK TYPE = 2 or 3

Figure A-1 (d): Encoding of "S_" Primitives (Cont.)

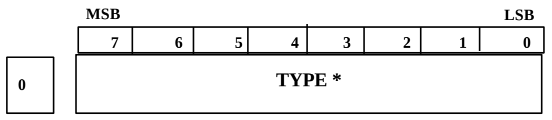


* TYPE = 9 (S_HARD_LINK_REJECTED)
TYPE = 10 (S_HARD_LINK_TERMINATED)
TYPE = 13 (S_HARD_LINK_REJECT)

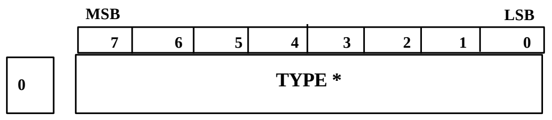
**ONLY VALID IF LINK TYPE = 2 or 3



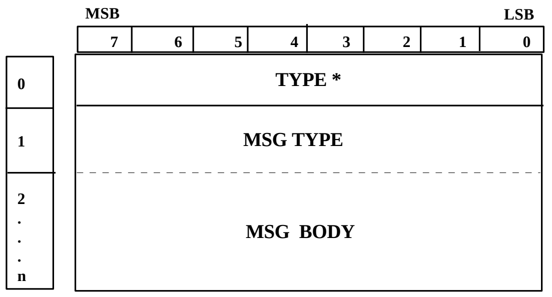
* TYPE = 14 (S_SUBNET_AVAILABILITY)



* TYPE = 15 (S_DATA_FLOW_ON)
TYPE = 16 (S_DATA_FLOW_OFF)

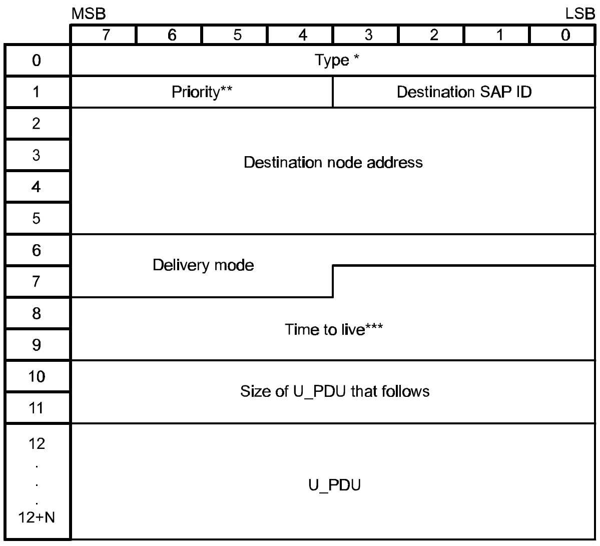


* TYPE = 17 (S_KEEP_ALIVE)



* TYPE = 18 (S_MANAGEMENT_MESSAGE_REQUEST)
TYPE = 19 (S_MANAGEMENT_MESSAGE_INDICATION)

Figure A-1 (e): Encoding of "S_" Primitives (Cont.)



* Type = 20 (S_UNIDATA_REQUEST)
Type = 24 (S_EXPEDITED_UNIDATA_REQUEST)

** Priority field is not used in the S_EXPEDITED_UNIDATA_REQUEST

*** TTL is defined in 2-second units with a maximum value of 648000 (15 days). The LSB of the TTL field is the LSB of byte 9.

Figure A-1 (f): Encoding of "S_" Primitives (Cont.)

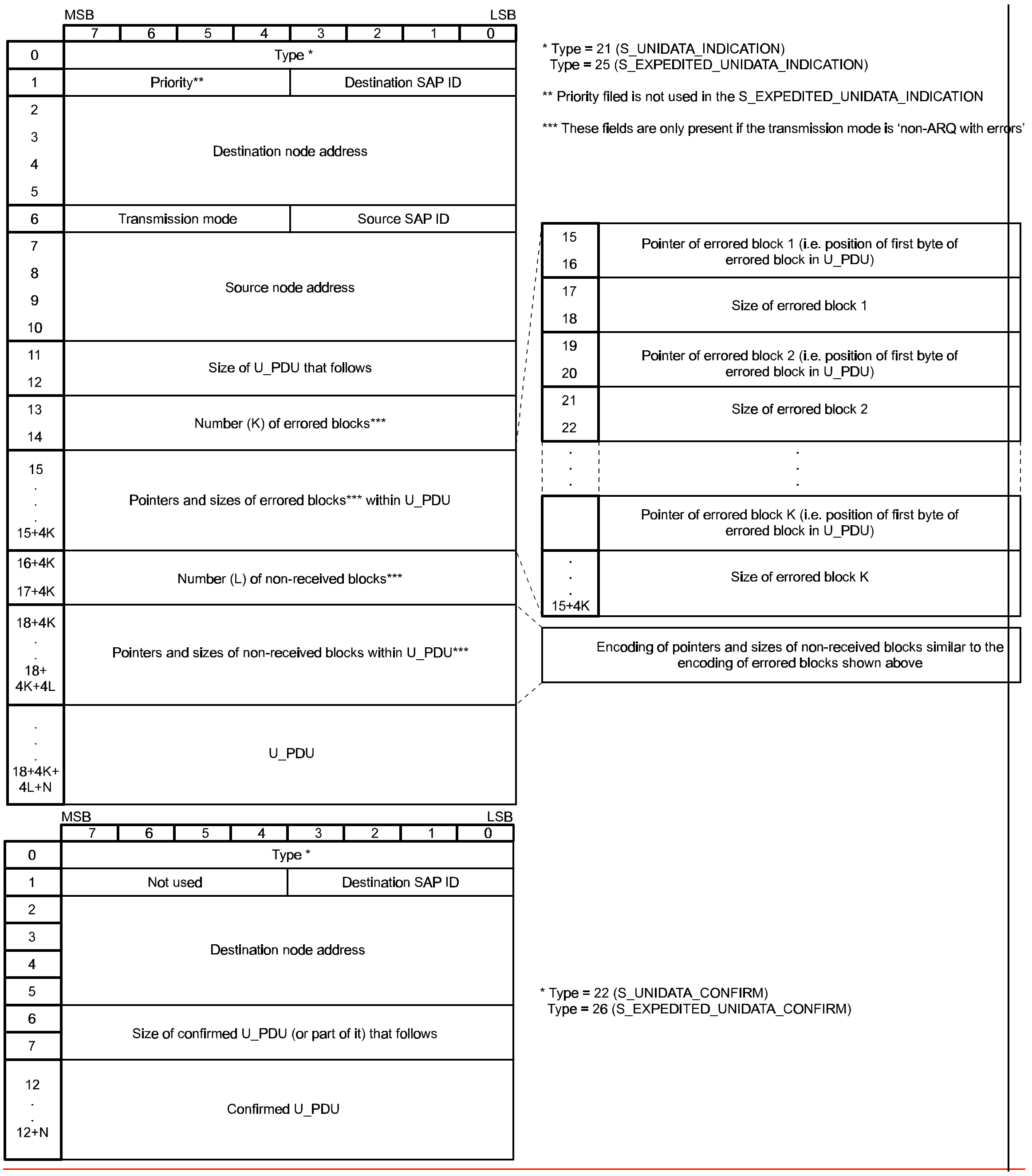
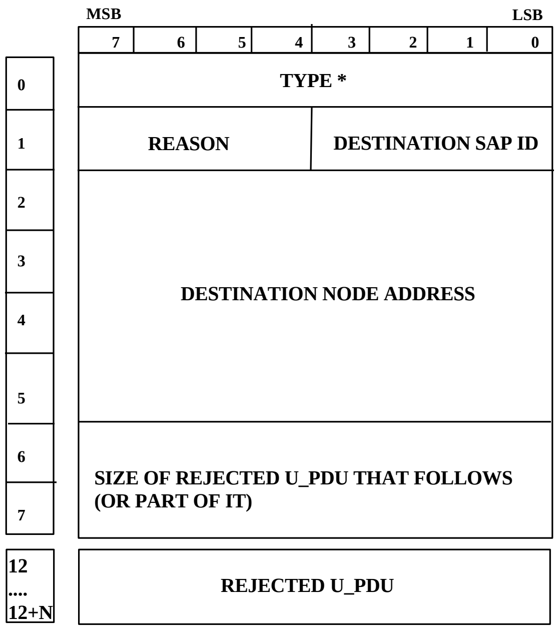

MSB
LSB
7 6 5 4 3 2 1 0
0 Type *
1 Priority** Destination SAP ID
2 3 4 5 Destination node address
6 Transmission mode Source SAP ID
7 8 9 10 Source node address
11 12 Size of U_PDU that follows
13 14 Number (K) of errored blocks***
15 . . . 15+4K Pointers and sizes of errored blocks*** within U_PDU
16+4K 17+4K Number (L) of non-received blocks***
18+4K . . 18+ 4K+4L Pointers and sizes of non-received blocks within U_PDU***
. . . 18+4K+ 4L+N U_PDU
* Type = 21 (S_UNIDATA_INDICATION)
Type = 25 (S_EXPEDITED_UNIDATA_INDICATION)
** Priority filed is not used in the S_EXPEDITED_UNIDATA_INDICATION
*** These fields are only present if the transmission mode is 'non-ARQ with errors'
15 16 Pointer of errored block 1 (i.e. position of first byte of errored block in U_PDU)
17 18 Size of errored block 1
19 20 Pointer of errored block 2 (i.e. position of first byte of errored block in U_PDU)
21 22 Size of errored block 2
Pointer of errored block K (i.e. position of first byte of errored block in U_PDU)
15+4K Size of errored block K
Encoding of pointers and sizes of non-received blocks similar to the encoding of errored blocks shown above
MSB
LSB
7 6 5 4 3 2 1 0
0 Type *
1 Not used Destination SAP ID
2 3 4 5 Destination node address
6 7 Size of confirmed U_PDU (or part of it) that follows
12 . . 12+N Confirmed U_PDU
* Type = 22 (S_UNIDATA_CONFIRM)
Type = 26 (S_EXPEDITED_UNIDATA_CONFIRM)

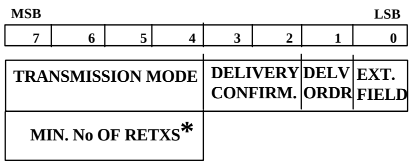
Figure A-1 (g): Encoding of "S_" Primitives (Cont.)



* TYPE = 23 (S_UNIDATA_REJECTED)
TYPE = 27 (S_EXPEDITED_UNIDATA_REJECTED)

Figure A-1 (h): Encoding of "S_" Primitives (Cont.)

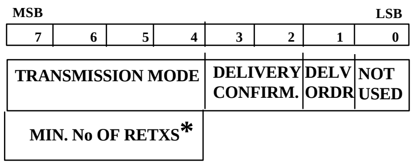
“SERVICE TYPE” FIELD OF S_BIND_REQUEST



*Only used if TRANSMISSION MODE is a Non-ARQ subtype. Otherwise it is “don’t care”

TRANSMISSION MODE	DELIVERY CONFIRMATION	DELIVERY ORDER	EXTENDED FIELD
0000 = Ignore Service type field. 0001 = ARQ 0010 = Non-ARQ (Broadcast) 0011-.... = other Non-ARQ types etc.	00 = No Confirmation 01 = Node Delivery Conf. 10 = Client Delievery Conf. 11 = Not defined	0 = In-order Delivery 1 = As they arrive	0 = No extended field 1 = Extended field follows

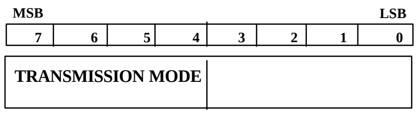
“DELIVERY MODE” FIELD OF S_UNIDATA_REQUEST AND S_EXPEDITED_UNIDATA_REQUEST



*Only used if TRANSMISSION MODE is a Non-ARQ subtype. Otherwise it is “don’t care”

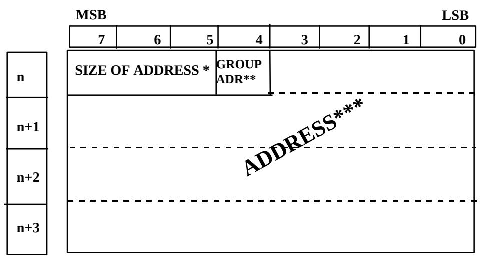
TRANSMISSION MODE	DELIVERY CONFIRM.	DELIVERY ORDER	EXTENDED FIELD
0000 = Ignore this field. Use default mode as specified during *S_BIND_REQUEST* time 0001 = ARQ 0010 = Non-ARQ (Broadcast) 0011-.... = other Non-ARQ types, etc.	00 = No Confirmation 01 = Node Delivery Conf. 10 = Client Delievery Conf. 11 = Not defined	0 = In-order Delivery 1 = As they arrive	0 = No extended field 1 = Extended field follows

“TRANSMISSION MODE” FIELD OF S_UNIDATA_INDICATION AND S_EXPEDITED_UNIDATA_INDICATION



TRANSMISSION MODE
0000 = Not Used 0001 = ARQ 0010 = Non-ARQ (Broadcast) 0011 = Non-ARQ WITH ERRORS 0100 ....= to be defined

FORMAT OF ADDRESS FIELDS USED IN THE “S_” PRIMITIVES”



* SPECIFIES THE ACTUAL ADDRESS SIZE IN “1/2 “ BYTES (Max 3 1/2 bytes)

** 0 = INDIVIDUAL ADDRESS
1 = GROUP ADRESS
Note:: A Group address can only be specified as a *Destination address of non-ARQ PDUs.*

*** Only the first 1/2 bytes specified by the “SIZE OF ADDRESS” field are valid. The rest are “don’t care”

Figure A-1 (i): Encoding of “S_” Primitives (Cont.)

Clarifications on selected arguments of Primitives

In order to clarify some of the procedures and tasks executed by the sublayers, additional details concerning some of the arguments of the Primitives described in previous sections are provided below.

Arguments :	SAP ID, NODE ADDRESS.
Primitives :	ALL "UNIDATA" primitives and "S_HARD_LINK" primitives.

Group node addresses can only be specified as the destination node address of Non-ARQ PDUs.
Destination SAP IDs and destination node addresses of ARQ PDUs and source SAP IDs and source node addresses of all PDUs *must be* individual SAP IDs and individual node addresses respectively.
Remote node addresses and remote SAP IDs of all "S_HARD_LINK" primitives *must be* individual SAP IDs and individual node addresses respectively.

Argument : SERVICE TYPE
Primitive : S_BIND_REQUEST

This argument specifies the default type of service requested by the client. This type of service will apply until the client unbinds itself from the node. Only by unbinding and binding again can a client change the default service type. This is a complex primitive consisting of a number of attributes. Although the exact number of attributes and their encoding is left for future definition, the following attributes are mandatory:

1. *Order of delivery of submitted U_PDUs*. A client may request that its U_PDUs are delivered to the destination client "in-order" (as they are submitted) or in the order they are received by the destination node.

2. *Transmission Mode.* This argument can take the values: *ARQ Transmission of U_PDUs, Non-ARQ (Broadcast) transmission of U_PDUs.*, Non-ARQ transmission can have submodes such as: *Error-Free-Only* delivery to destination client, delivery to destination client even with *some* errors.

3. *Data Delivery Confirmation*. The client can request explicit confirmation from the Subnetwork as to whether its U_PDUs have been properly delivered to their destination. Delivery confirmation can be requested only in combination with ARQ delivery. There are three types of data delivery confirmation:

    - None
    - Node-to-Node Delivery Confirmation
    - Client-to-Client Delivery Confirmation

The Node-to-Node Delivery Confirmation does not require any explicit peer-to-peer communication between the Subnetwork Interface Sublayers and hence it does not introduce extra overhead. It simply uses the ACK (ARQ) confirmation provided by the Data Transfer Sublayer. Client-to-Client Delivery Confirmation requires explicit peer-to-peer communication between the Sublayers and therefore introduces overhead. It should be used only when it is absolutely critical for the client to know whether or not its data was delivered to the destination client (which may, for instance, be disconnected).

4. *Delivery Order*. In-order delivery is supported only for ARQ modes. This bit shall be ignored if a non-ARQ transmission mode is specified.

The above attributes apply to all U_PDUs submitted by the client. However, the value of some of these attributes can be overridden by the client for individual U_PDUs (see *Delivery Mode* argument of S_UNIDATA_REQUEST primitive below).

Argument : DELIVERY MODE
Primitive : S_UNIDATA_REQUEST , S_EXPEDITED_UNIDATA_REQUEST

This is a complex argument consisting of a number of attributes. The value of this argument can be "DEFAULT" which means that the delivery mode for this U_PDU is the one specified in the *Service Type* argument of the S_BIND_REQUEST. A non-DEFAULT value is used to override the default settings of the Service Type for this U_ PDU. The attributes of this argument are similar to those described in *the Service Type* argument:

1. *Order of delivery of this PDU to the receiving client.*

2. *Transmission Mode of this U_PDU.* (ARQ or Non-ARQ Transmission can be requested).

3. *Data Delivery Confirmation for this PDU* (node-to-node, client-to-client or none).

Argument : LINK TYPE
Primitive : S_HARD_LINK_ESTABLISH

The Link Type argument is used by a client to partially or fully reserve the capacity of the Hard Link. This argument can have three values:

- A value of 0 indicates that the physical link to the specified node address must be maintained, but all clients connected to the two nodes can make use of the link capacity according to normal procedures, i.e. there is no bandwidth reservation.

- A value of 1 indicates that the physical link to the specified node address must be maintained and traffic is only allowed between the requesting client and any of the clients on the remote Node, i.e. partial bandwidth reservation.

- A value of 2 indicates that the physical link to the specified node address must be maintained and traffic is only allowed between the requesting Client and the specific Client on the remote node specified by the remote SAP ID argument, i.e. full bandwidth reservation.

Figure A-2 illustrates the allowed communication paths for the three types of Hard Links.

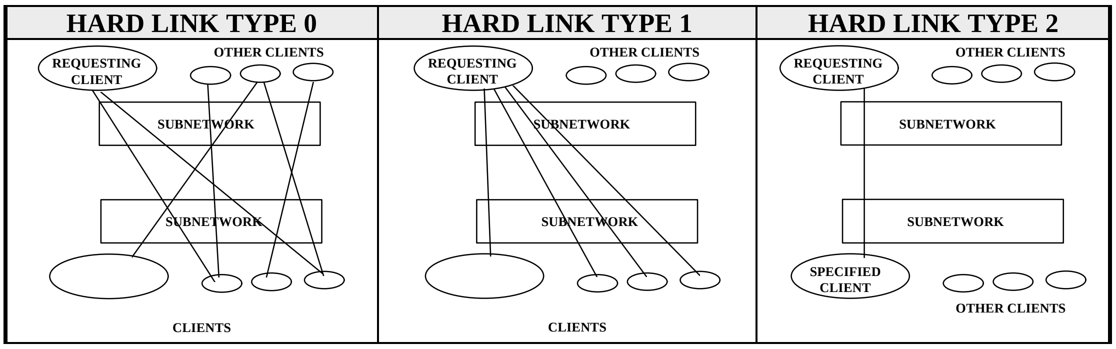


Figure A-2: Hard Link Types

It is emphasised that the multiple communication paths that are shown in the above Figure are multiplexed over the same physical link between the nodes.

A.3 Peer-to-Peer Communication Protocols and S_PDUs

Peer Subnetwork Interface Sublayers (generally in different nodes) communicate with each other by exchanging Subnetwork Interface Sublayer Protocol Data Units (S_PDUs). For the Subnetwork configurations currently defined, Peer-to-Peer Communication is required for:

1. Establishment and Termination of Hard Link Data Exchange Sessions

2. Exchange of Client Data

Explicit Peer-to-Peer communication is not required for the establishment or termination of Soft Link or Broadcast Data Exchange sessions. The Peer-to-Peer communication required for the exchange of Client Data is similar for all Data exchange sessions. The encoding of the S_PDUs and the protocol governing the Peer-to-Peer Communication are described in the following sections.

A.3.1 Subnetwork Interface Sublayer Protocol Data Units (S_PDUS)

There are currently seven types of S_PDUs. Additional S_PDU types may be defined in the future. The generic encoding of the seven S_PDU types showing the fields and subfileds of the S_PDUs is shown in Figure A-3.

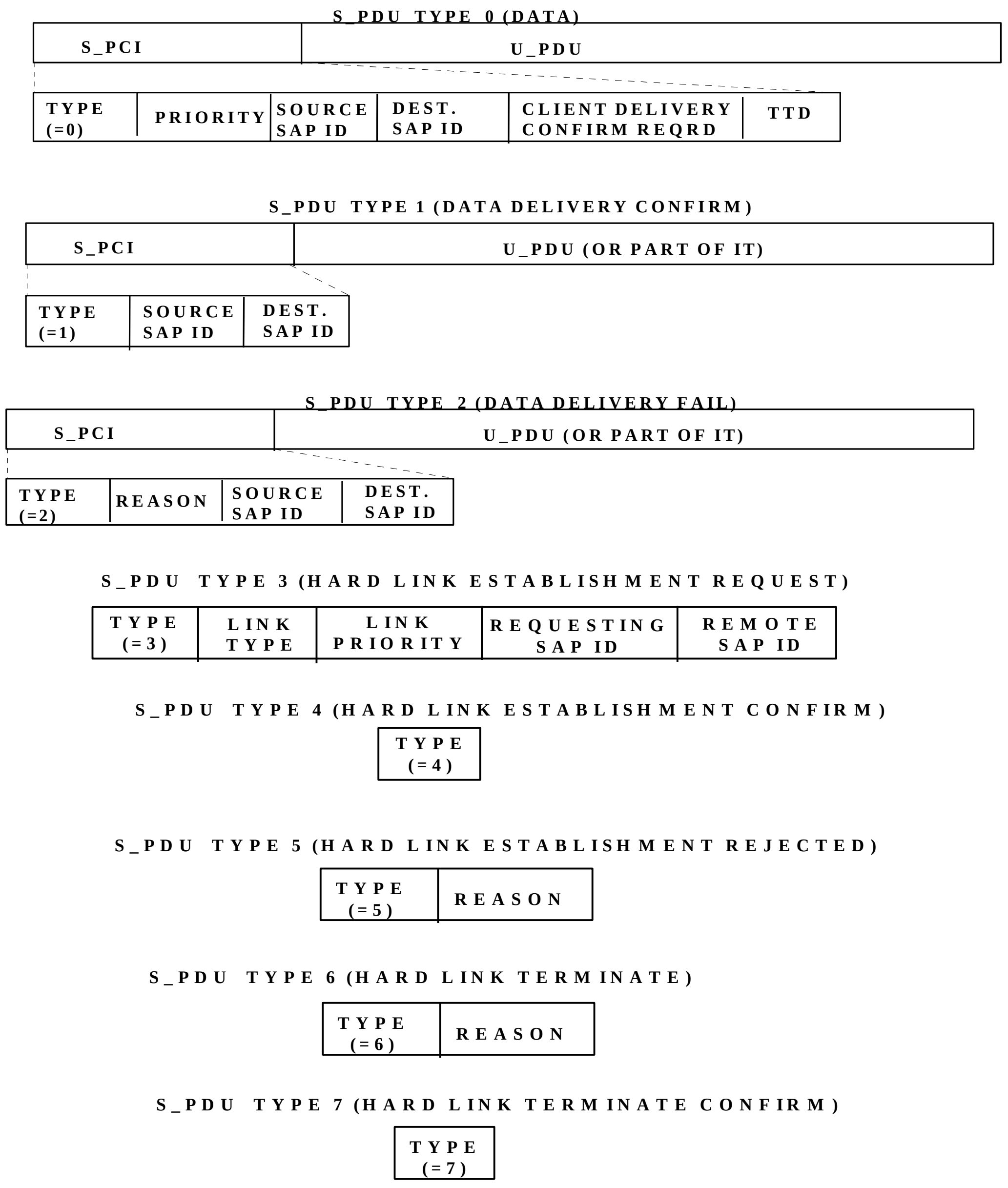

S_PDU TYPE 0 (DATA)
S_PCI
U_PDU
TYPE (=0)
PRIORITY
SOURCE SAP ID
DEST. SAP ID
CLIENT DELIVERY CONFIRM REQRD
TTD
S_PDU TYPE 1 (DATA DELIVERY CONFIRM)
S_PCI
U_PDU (OR PART OF IT)
TYPE (=1)
SOURCE SAP ID
DEST. SAP ID
S_PDU TYPE 2 (DATA DELIVERY FAIL)
S_PCI
U_PDU (OR PART OF IT)
TYPE (=2)
REASON
SOURCE SAP ID
DEST. SAP ID
S_PDU TYPE 3 (HARD LINK ESTABLISHMENT REQUEST)
TYPE (=3)
LINK TYPE
LINK PRIORITY
REQUESTING SAP ID
REMOTE SAP ID
S_PDU TYPE 4 (HARD LINK ESTABLISHMENT CONFIRM)
TYPE (=4)
S_PDU TYPE 5 (HARD LINK ESTABLISHMENT REJECTED)
TYPE (=5)
REASON
S_PDU TYPE 6 (HARD LINK TERMINATE)
TYPE (=6)
REASON
S_PDU TYPE 7 (HARD LINK TERMINATE CONFIRM)
TYPE (=7)

Figure A-3: Generic Encoding of S_PDUs

The first field is common to all S_PDUs. It is called "TYPE" and identifies the type of the S_PDU.
The other fields are described below for each S_PDU type and defined in more detail in Figure A-4:

Type :

"0" = DATA S_PDU

Encoding :

S_PCI	U_PDU

TYPE (=0)	PRIORITY	SOURCE SAP ID	DEST. SAP ID	CLIENT DELIVERY CONFIRM REQRD	TTD

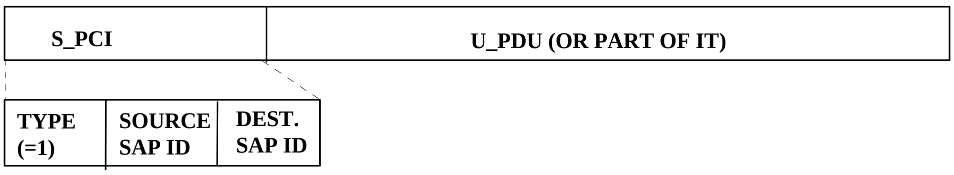
Description :

This S_PDU is transmitted by the Subnetwork Interface Sublayer in order to send client data to a remote peer sublayer. This S_PDU consists of two parts. The first part is the S_PCI (Subnetwork Interface Sublayer Protocol Control Information) and represents the overhead added by the sublayer. The second part is the actual client data (U_PDU). The first field of the S_PCI is "TYPE". It is equal to 0 and identifies the S_PDU as being of type DATA. The second field is "PRIORITY" and represents the priority of the client's U_PDU. The third ~~second~~ field of the S_PCI is the "SOURCE SAP ID" and identifies the client of the transmitting peer which sent the data. The fourth ~~third~~ field is the "DESTINATION SAP ID" and identifies the client of the receiving peer which must take delivery of the data. There is no need to encode the source and destination node addresses in the S_PDU as this information is relayed between the peers by the underlying sublayers. ~~The fourth field is "PRIORITY" and represents the priority of the client's U_PDU. The fifth field is "TTD" and represents the TimeToDie for this U_PDU. The "DESTINATION SAP ID", "PRIORITY" and "TTD" fields are copied directly from the corresponding arguments of the S_UNIDATA_REQUEST primitive submitted by the client.~~
The fifth ~~final~~ field of the S_PCI is shown in the figure above as "CLIENT DELIVERY CONFIRM REQUIRED". The field is made up of 4 bits, one for delivery confirmation, one indicating the presence of a valid TTD, and two reserved; see figure A-4(a) for a more detailed description. ~~This field~~ The "CLIENT DELIVERY CONFIRM REQUIRED" bit can take the values "YES" (=1) or "NO" (=0). The value of this bit ~~field~~ will be set according to the *Service Type* requested by the sending client during binding (see S_BIND_REQUEST primitive) or according to the *Delivery Mode* requested explicitly for this U_PDU (see S_UN~~U~~IDATA_REQUEST Primitive). If this field has the value of "YES", the receiving peer should confirm to the transmitting peer the successful delivery of the U_PDU to the destination client. Upon receiving a DATA S_PDU, the receiving peer must attempt to deliver the U_PDU to the destination client by issuing a S_UNIDATA_INDICATION, or a S_EXPEDITED_UNIDATA_INDICATION. The last field of the S_PCI is "TTD" and represents the TimeToDie for this U_PDU. The "DESTINATION SAP ID", "PRIORITY" and "TTD" fields are copied directly from the corresponding arguments of the S_UNIDATA_REQUEST primitive submitted by the client.

Type :

"1" = DATA DELIVERY CONFIRM

Encoding :



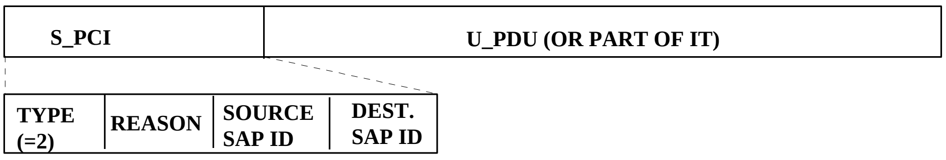
Description :

This S_PDU is transmitted in response to a successful delivery to a Client of a U_PDU which was received in a DATA type S_PDU in which the "CLIENT DELIVERY CONFIRM REQUIRED" field was set to "YES". The first part of this S_PDU is the S_PCI, while the second part is a full or partial copy of the U_PDU that was received and delivered to the destination Client. The first field of the S_PCI is "TYPE". It is equal to 1 and identifies the S_PDU as being of type DATA DELIVERY CONFIRM. The remaining fields of the S_PCI are direct copies of the corresponding fields of the DATA S_PDU which caused the transmission of the DATA DELIVERY CONFIRM. The peer sublayer which receives the DATA DELIVERY CONFIRM, must inform the Client which originated the U_PDU that its Data has been successfully delivered to its Destination (by issuing a S_UNIDATA_REQUEST_CONFIRM or a S_EXPEDITED_UNIDATA_REQUEST_CONFIRM).

Type :

"2" = DATA DELIVERY FAIL

Encoding :



Description :

This S_PDU is transmitted in response to a failed delivery to a Client of a U_PDU which was received in a DATA type S_PDU with the "CLIENT DELIVERY CONFIRM REQUIRED" field set to "YES". The first part of this S_PDU is the S_PCI. The second part is a full or partial copy of the U_PDU that was received but not delivered to the destination Client. The first field of the S_PCI is "TYPE". It is equal to 2 and identifies the S_PDU as being of type DATA DELIVERY FAIL. The second field is "REASON" and explains why the U_PDU failed to be delivered. It can take a value in the range 0-15; valid reasons are defined in the table below.The rest of the fields of the S_PCI are direct copies of the corresponding fields of the DATA S_PDU which caused the transmission of the DATA DELIVERY FAIL. The Peer sublayer which receives the DATA DELIVERY FAIL S_PDU, must inform the Client which originated the U_PDU that its Data was not delivered to the Destination (by issuing a S_UNIDATA_REQUEST_REJECTED or a S_EXPEDITED_UNIDATA_REQUEST_REJECTED).

Reason	Value
Destination SAP ID not bound	1

Type :
“3” = HARD LINK ESTABLISHMENT REQUEST

Encoding :

TYPE (=3)	LINK TYPE	LINK PRIORITY	REQUESTING SAP ID	REMOTE SAP ID

Description :
This S_PDU is transmitted by a Peer in response to a Client’s request for a Hard Link. Since the establishment of a Hard Link overrides the normal procedures of making Links based on the destinations of the queued U_PDUs (Soft Links), it is important that both Peers use a handshake procedure in order to confirm the successful Hard Link establishment The first field of the S_PDU is “TYPE”. It is equal to 3 and identifies the S_PDU as being of type HARD LINK ESTABLISHMENT REQUEST. The “LINK TYPE” and “LINK PRIORITY” fields are direct copies of the corresponding arguments of the S_HARD_LINK_ESTABLISH Primitive submitted by the Client. The “REQUESTING SAP ID” field is the SAP ID of the Client which requested the Hard Link Establishment. The “REMOTE SAP ID” field identifies the single Client connected to the remote Node, to and from which traffic is allowed. This argument is only valid if the “LINK TYPE” field has a value of 2 (see discussion at the end of this section for clarification). ~~For other values of “LINK TYPE” the “REMOTE SAP ID” is “don’t care”.~~

Type :
“4” = HARD LINK ESTABLISHMENT CONFIRM

Encoding :

TYPE (=4)

Description :
This S_PDU is transmitted as a positive response to the reception of a HARD LINK ESTABLISHMENT REQUEST S_PDU. Its only field is “TYPE”, which is equal to 4 and identifies the S_PDU as being of type HARD LINK ESTABLISHMENT CONFIRM. The peer which receives this S_PDU should inform its appropriate Client accordingly with a S_HARD_LINK_ESTABLISHED Primitive.

Type :
“5” = HARD LINK ESTABLISHMENT REJECTED

Encoding :

TYPE (=5)	REASON

Description :
This S_PDU is transmitted as a negative response to the reception of a HARD LINK ESTABLISHMENT REQUEST S_PDU. The first field is “TYPE” and contains a value of 5 to identify the S_PDU as being of type HARD LINK ESTABLISHMENT REJECTED. The second field is “REASON” and explains why the Hard Link

Request was rejected. It can take a value in the range 0-15. Reasons which have been defined are given in the table. ~~Possible interpretations of the value of this field can be : Higher-Priority-Link-In-Progress, Destination-SAP_ID-Not-Bound, etc.~~ The Peer which receives this S_PDU, should inform its appropriate Client accordingly with a S_HARD_LINK_REJECTED Primitive.

Reason	Value
Remote-Node-Busy	1
Higher-Priority-Link-Existing	2
Remote-Node-Not-Responding	3
Destination SAP ID not bound	4

Type :

“6” = HARD LINK TERMINATE

Encoding :

TYPE (=6)	REASON

Description :

This S_PDU can be transmitted by either of the two Peer sublayers involved in a Hard Link session to request termination of the Hard Link. Under normal circumstances a Hard Link is terminated at the request of the Client which originated it or as a result of a request by another Client to establish a higher priority Hard Link. The first field is “TYPE” and is set to 6 to identify the S_PDU as being of type HARD LINK TERMINATE. The second field is “REASON” and explains why the Hard Link is being terminated. Values which have been defined are shown in the table below. ~~It can take a value in the range 0-15. Possible interpretations of the value of this field can be : Client-Request, Higher-Priority-Link-Requested, etc.~~ In order to ensure a graceful termination of the Hard Link, the Peer which sent the HARD LINK TERMINATE must await (within a specified time-out-period) the confirmation of its Peer before it declares the Link as terminated.

Reason	Value
Client request	1
Higher priority link requested	2
reserved	3
Destination SAP ID unbound	4

Type :

“7” = HARD LINK TERMINATE CONFIRM

Encoding :

TYPE (=7)

Description :

This S_PDU is transmitted in response to the reception of a HARD LINK TERMINATE S_PDU.

The only field of this S_PDU is “TYPE”. A value of 7 identifies that the S_PDU is of type HARD LINK TERMINATE CONFIRM.

Bit-Maps of the S_PDUs types are shown in Figs. A-4 (a) and (b).

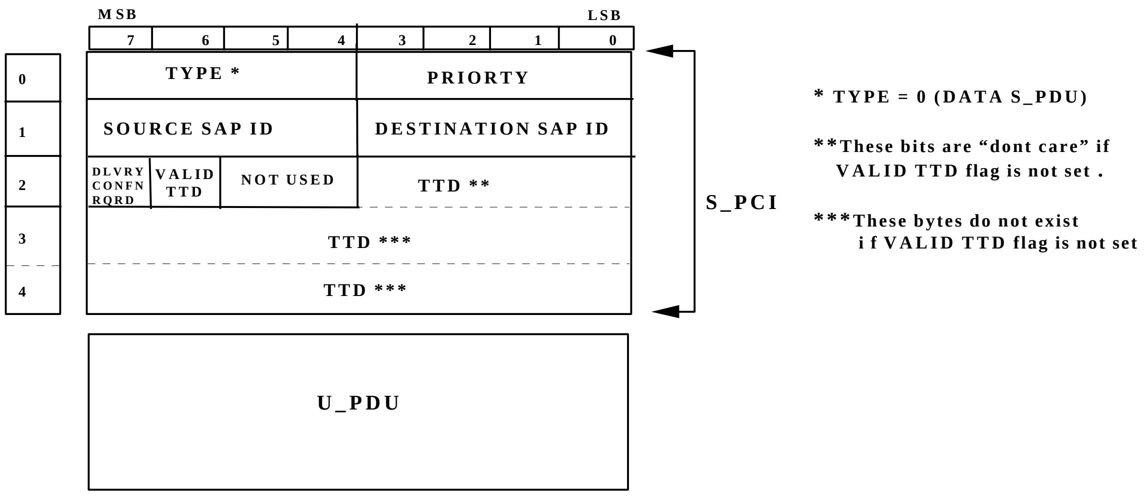
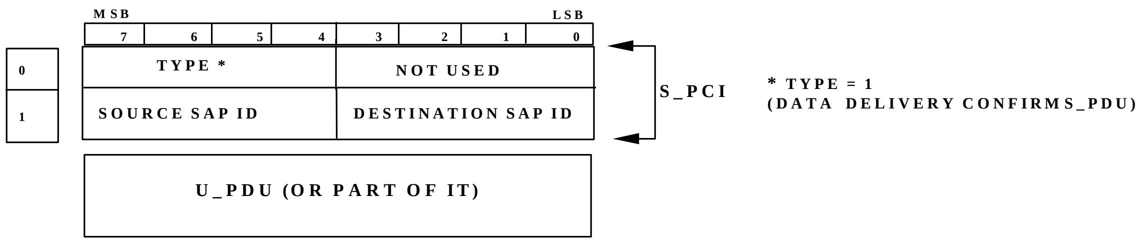
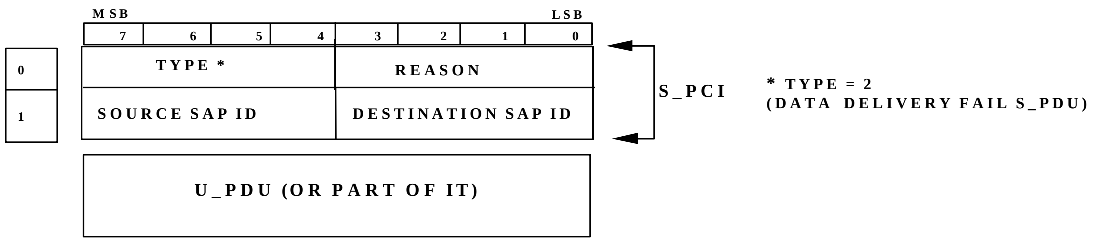


Figure A-4 (a): Bit Maps of S_PDUs





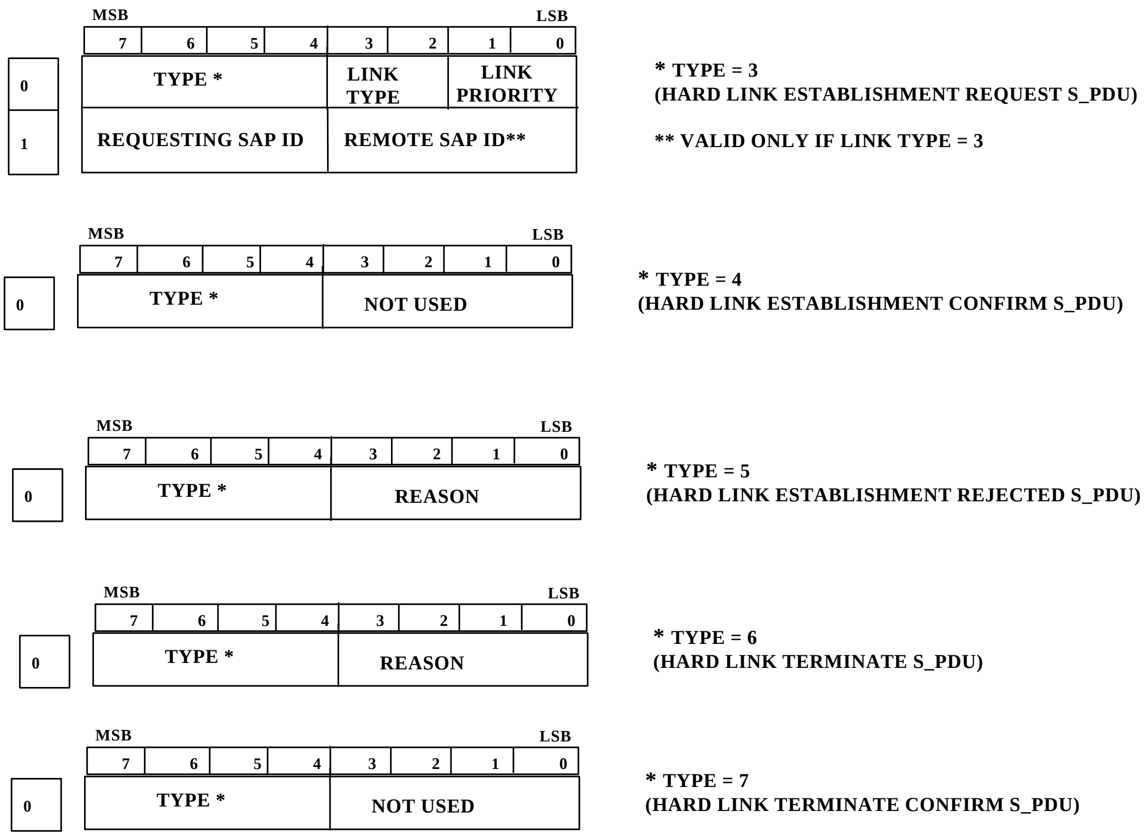

MSB
LSB
7 6 5 4 3 2 1 0
0
1
TYPE *
LINK TYPE
LINK PRIORITY
REQUESTING SAP ID
REMOTE SAP ID**
* TYPE = 3
(HARD LINK ESTABLISHMENT REQUEST S_PDU)
** VALID ONLY IF LINK TYPE = 3
TYPE *
NOT USED
* TYPE = 4
(HARD LINK ESTABLISHMENT CONFIRM S_PDU)
TYPE *
REASON
* TYPE = 5
(HARD LINK ESTABLISHMENT REJECTED S_PDU)
TYPE *
REASON
* TYPE = 6
(HARD LINK TERMINATE S_PDU)
TYPE *
NOT USED
* TYPE = 7
(HARD LINK TERMINATE CONFIRM S_PDU)

Figure A-4 (b): Bit Maps of S_PDUs

~~The TTD field shall be filled with the GMT in 1/8 seconds after which time the S_PDU must be discarded by the receiving peer if it has not yet been delivered to the client.~~ The TTD field shall be filled with the Julian date[4] modulo 16, and the GMT in seconds after which time the S_PDU must be discarded if it has not yet been delivered to the client. The Julian date mod 16 part of the TTD shall be mapped into bits 0-3 of byte 2 of the S_PDU. The 16 high bits of the GMT part of the TTD shall be mapped into the 2 remaining bytes of the TTD field; the LSB of the GMT shall be discarded. If the "VALID TTD" flag bit of a DATA S_PDU is set (=1) then the complete TTD 20-bit field is present and its value must be used. If this flag bit is not set (=0), the last two bytes of the TTD field are not present (to conserve overhead) and the TTD must not be used. The "VALID TTD" flag bit allows the transmitting peer to specify whether the S_PDU should be discarded by the receiving peer based on TTD or it should be delivered to the client without consideration of the TTD.

The bit-maps of the remaining fields of the S_PDUs do not require additional explanation as there is a one-to-one association with the corresponding fields shown in the generic encoding of Figure A-3.

A.3.2 Peer-to-Peer Communication Protocol

[4] The simple Julian date system, which numbers the days of the year consecutively starting with 001 on 1 January and ending with 365 on 31 December (or 364 on leap years).

The Protocols governing the Peer-to-Peer communication for Establishing and Terminating Soft Link Data Exchange Sessions, Establishing and Terminating Hard Link Data Exchange Sessions, Establishing and Terminating Broadcast Data Exchange Sessions and Exchanging Client Data are described below.

A.3.2.1 Soft Link Data Exchange Session

A.3.2.1.1 Protocol for Establishing a Soft Link Data Exchange Session

In the absence of a hard link request by a client, the Subnetwork Interface Sublayer initiates Soft Link Data Exchange Sessions with remote peers, based on the destinations of queued client U_PDUs. In contrast with the establishment of a Hard Link Session, explicit peer-to-peer handshaking is not required for the establishment of Soft Link Data Exchange Sessions. The caller peer *implicitly establishes* a Soft Link Session by requesting the making of a physical link to the required remote node. The procedure for making physical links is explained in Annex B. Both peer Subnetwork Interface Sublayers (Caller and Called) are informed about the successful making of a physical link between their nodes by their respective Channel Access Sublayers. After the physical link is made, both peer Subnetwork Interface Sublayers assume that a Soft Link Data Exchange Session has started, unless one of them (usually the caller) explicitly sends a “HARD LINK ESTABLISHMENT REQUEST” S_PDU.

A.3.2.1.2 Protocol for Terminating a Soft Link Data Exchange Session

A Soft Link Data Exchange Session can be terminated by either of the two peers involved by simply breaking the Physical Link. No peer-to-peer communication is needed. The procedure for breaking Physical Links is explained in Annex B. Both peers will be informed about the breaking of the Physical link by their respective Channel Access Sublayers. After the Physical Link has been broken, both peer Subnetwork Interface Sublayers declare the Soft Link Exchange Session as terminated.

The caller peer normally initiates the termination of the session (by breaking the physical link) based on the destinations of its queued U_PDUs, and on any ongoing communication with the distant node. The inter-layer signaling required to coordinate this would normally be carried out via the subnetwork management sublayer. The called peer can also terminate the Session if it has high priority data destined for a different Node. However, called peers should wait a configurable minimum time before unilaterally terminating sessions, otherwise an unstable condition may arise, i.e. all nodes in the Network have data to transmit and Called Peers immediately close Sessions in order to establish other Sessions as Callers. If such a situation arises, the efficiency of the Subnetwork will deteriorate as a result of Nodes continuously establishing and terminating Sessions without actually transmitting data. The minimum amount of time that a Called Peer should wait before it attempts to terminate a Soft Link Session must be carefully chosen and will depend on a number of factors such as the Subntework size and configuration.

A.3.2.2 Hard Link Data Exchange Session

A.3.2.2.1 Protocol for Establishing a Hard Link Data Exchange Session

Upon receiving a S_HARD_LINK_ESTABLISH Primitive from a client, the Subnetwork Interface Sublayer first checks if can accept the request from the client, i.e. there is not an existing higher priority hard link, etc. If it can accept the request it will first terminate the current hard link if one exists. The peer-to-peer communication protocol for terminating an existing hard link is described in Section A.3.2.2.2. The Subnetwork Interface Sublayer then initiates the making of a physical link to the node specified by the client. The procedure for making the physical link is explained in Annex B. After the physical link has been made, the Subnetwork Interface Sublayer sends a “HARD LINK ESTABLISHMENT REQUEST” (type 3) S_PDU to its peer sublayer at the remote node. In order to ensure that the S_PDU will overtake all routine DATA S_PDUs which may be queued and in various stages of processing by the lower sublayers, the “HARD LINK ESTABLISHMENT REQUEST” S_PDU should ~~be submitted to the Channel Access Sublayer as a C_EXPEDITED_UNIDATA_REQUEST Primitive~~ use the expedited data service.

When the called Subnetwork Access Sublayer receives the “HARD LINK ESTABLISHMENT REQUEST” S_PDU ~~(which is normally delivered to it by the Channel Access Sublayer as a C_EXPEDITED_UNIDATA_INDICATION Primitive)~~ it checks if it can accept the request, e.g. it will verify that one of its own clients has not requested a higher priority hard link, and verify that the a client is bound to the called SAP~~etc~~. The node shall not accept the request if no client is bound to the called SAP ID. If the request is accepted, the called peer must return, within a specified maximum time-out period, a “HARD LINK ESTABLISHMENT CONFIRM” (type 4) S_PDU to its calling peer. If the request cannot be accepted by the called peer, a “HARD LINK ESTABLISHMENT REJECTED” (type 5) S_PDU must be sent to the calling peer. Both these S_PDUs are normally sent using the expedited data service~~submitted to the Channel Access Sublayer as C_EXPEDITED_UNIDATA_REQUEST Primitives~~. When the calling peer receives the response from the called peer, it declares the hard link establishment as successful (or unsuccessful) and informs its clients accordingly.

The procedures for establishing a hard link for both the caller and called peers are shown in Figure A-5.

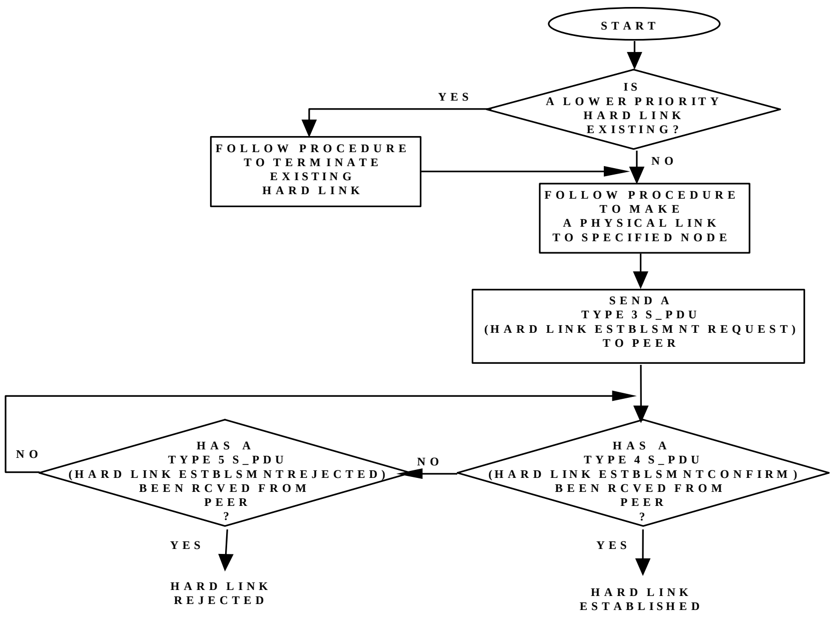

START
IS A LOWER PRIORITY HARD LINK EXISTING?
YES
NO
FOLLOW PROCEDURE TO TERMINATE EXISTING HARD LINK
FOLLOW PROCEDURE TO MAKE A PHYSICAL LINK TO SPECIFIED NODE
SEND A TYPE 3 S_PDU (HARD LINK ESTBLSMNT REQUEST) TO PEER
HAS A TYPE 5 S_PDU (HARD LINK ESTBLSMNTREJECTED) BEEN RCVED FROM PEER ?
NO
YES
HARD LINK REJECTED
HAS A TYPE 4 S_PDU (HARD LINK ESTBLSMNTCONFIRM) BEEN RCVED FROM PEER ?
NO
YES
HARD LINK ESTABLISHED

Figure A-5 (a): Procedures for Establishing a Hard Link: CALLER PEER

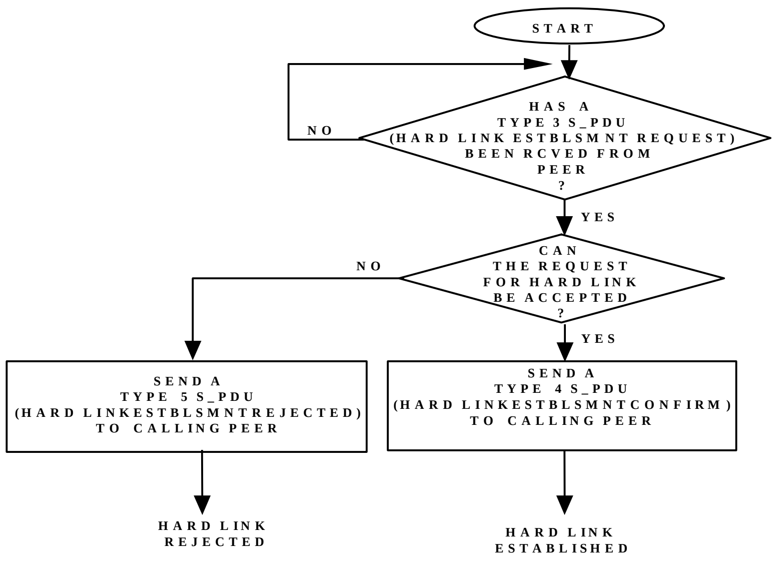

START
HAS A TYPE 3 S_PDU (HARD LINK ESTBLSMNT REQUEST) BEEN RCVED FROM PEER ?
NO
YES
CAN THE REQUEST FOR HARD LINK BE ACCEPTED ?
NO
YES
SEND A TYPE 5 S_PDU (HARD LINKESTBLSMNTREJECTED) TO CALLING PEER
SEND A TYPE 4 S_PDU (HARD LINKESTBLSMNTCONFIRM) TO CALLING PEER
HARD LINK REJECTED
HARD LINK ESTABLISHED

Figure A-5 (b): Procedures for Establishing a Hard Link: CALLED PEER

A.3.2.2.2 Protocol for Terminating a Hard Link Data Exchange Session

The termination of the Hard Link can be initiated by either of the two peer sublayers. Normally the Hard Link will be terminated by the Calling Peer at the request of the Client who initiated it, or by either of the peers if they receive a higher Priority Hard Link Request from one of their other Clients. The peer which wishes to terminate the Hard Link, sends a "HARD LINK TERMINATE" (type 6) S_PDU. The receiving Peer immediately declares the Hard Link as terminated and responds with a "HARD LINK TERMINATE CONFIRM" (type 7) S_PDU. The initiating peer upon receiving this response (or after a specified time-out) also declares the Hard Link as terminated. Both peers inform their clients according to the rules associated with the S_HARD_LINK_TERMINATED primitive~~ly~~. These S_PDUs should be submitted to the Channel Access Sublayer as C_EXPEDITED_UNIDATA_REQUEST Primitives
.

After termination of the Hard Link with a subnetwork client, the Physical Link between the nodes may need to be broken. Normally the breaking of the Physical Link is left to the peer which requested the termination of the Hard Link session. The reason for this is that this peer may want to start another session using the existing Physical Link in which case breaking and making procedures may be avoided. The procedures for breaking a Physical Link are explained in Annex B.

The procedures for Terminating a Hard Link by both the Requesting and Responding Peers are shown in Figure A-6.

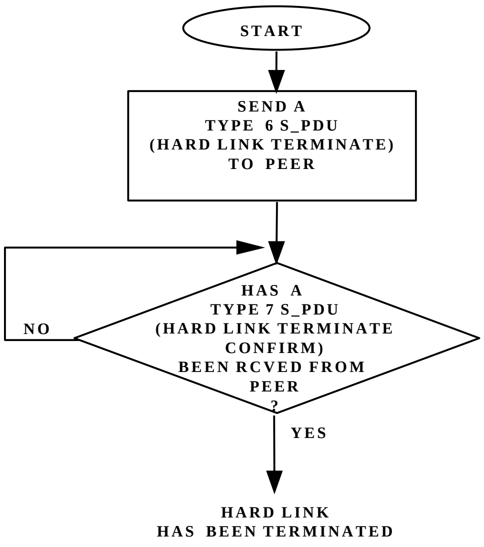

START
SEND A TYPE 6 S_PDU (HARD LINK TERMINATE) TO PEER
HAS A TYPE 7 S_PDU (HARD LINK TERMINATE CONFIRM) BEEN RCVED FROM PEER ?
NO
YES
HARD LINK HAS BEEN TERMINATED

Figure A-6 (a): Procedures for Terminating a Hard Link: REQUESTING PEER

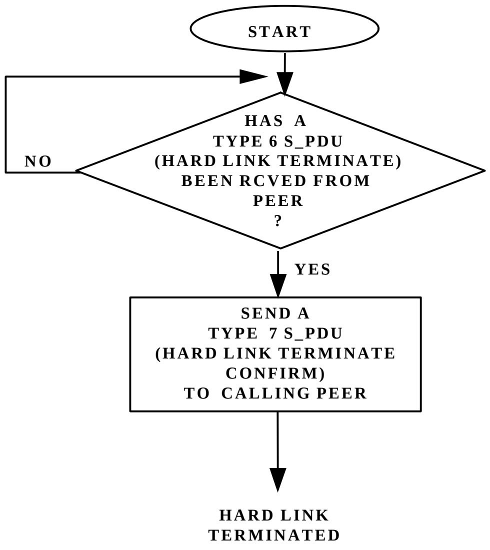

START
HAS A TYPE 6 S_PDU (HARD LINK TERMINATE) BEEN RCVED FROM PEER ?
NO
YES
SEND A TYPE 7 S_PDU (HARD LINK TERMINATE CONFIRM) TO CALLING PEER
HARD LINK TERMINATED

Figure A-6 (b): Procedures for Terminating a Hard Link: RESPONDING PEER

Apart from the procedures above, a peer will unilaterally declare a Hard Link as terminated, if at any time it is informed by the Channel Access Sublayer that the Physical Link has been permanently broken.

A.3.2.3 Protocol for Establishing and Terminating a Broadcast Data Exchange Session

No explicit peer-to-peer communication is required in order to establish and terminate a Broadcast Data Exchange Session. A Broadcast Data Exchange Session is established and terminated either by a management process or unilaterally by the Subnetwork Interface Sublayer based on a number of criteria as explained in section A.1.1.3.

A.3.2.4 Protocol for Exchanging Client Data

After a Data Exchange Session has been established DATA S_PDUs (type 0) can be exchanged.

The sending peer retrieves Client U_PDUs from its Queues (according to Priority and/or other criteria), encodes them into appropriate DATA S_PDUs and sends them to its peer at the remote end of the Link. A S_PDUs is submitted to the Channel Access Sublayer as a C_UNIDATA_REQUEST Primitive of the appropriate priority or as a C_EXPEDITED_UNIDATA_REQUEST Primitive, depending upon whether the encoded U_PDU was submitted as a S_UNIDATA_REQUEST or as a S_EXPEDITED_UNIDATA_REQUEST Primitive by the Client.

The Receiving Peer decodes the S_PDU and extracts from it the U_PDU, the Destination SAP_ID and the other encoded attributes described in Figure A-4. The U_PDU is then delivered to the Destination Client as a S_UNIDATA_INDICATION Primitive or as a S_EXPEDITED_UNIDATA_INDICATION Primitive, depending upon whether the received S_PDU was submitted by the Channel Access Sublayer as C_UNIDATA_INDICATION Primitive or C_EXPEDITED_UNIDATA_INDICATION Primitive. If the received S_PDU has the “CLIENT DELIVERY CONFIRM REQUIRED” field set equal to “YES”, then the Receiving Peer must respond with either a “DATA DELIVERY CONFIRM” S_PDU (type 1) or a “DATA DELIVERY FAIL” S_PDU (type 2) depending upon whether the U_PDU was successfully delivered to the destination Client or not. Normally these response S_PDUs should have the same Priority (or be of “Expedited” format) as the DATA S_PDU that caused them. However, as the Priority of these Response S_PDUs does not affect interoperability, it is left as an implementation issue.

It should be noted that, as shown in Figure A-3, there is no explicit indication that the S_PDU is a “normal” or an “expedited” one. The reason for this is that the underlying sublayers are expected to treat Expedited S_PDUs differently and implicitly pass the information to the receiving peer by (for example) delivering Expedited S_PDUs as C_EXPEDITED_UNIDATA_INDICATION Primitives rather than normal C_UNIDATA_INDICATION Primitives.

The procedures for exchanging DATA S_PDUs for both the Sending and Receiving Peers are shown in Figure A-7.

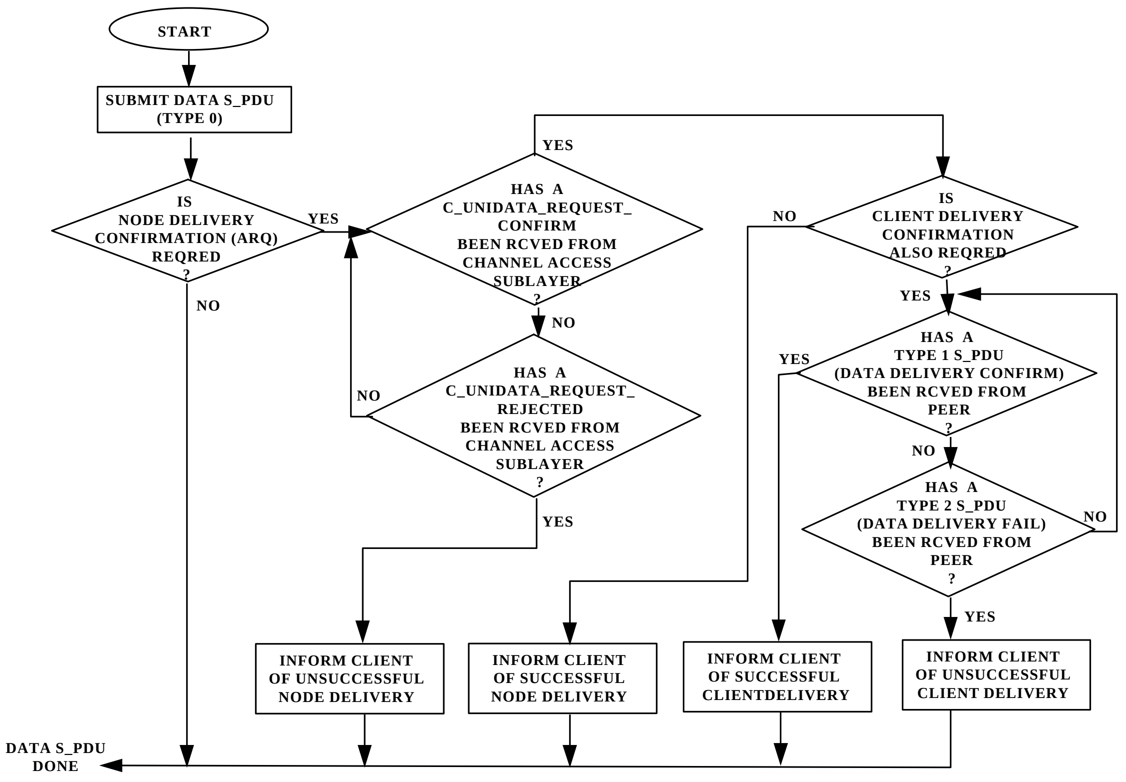

START
SUBMIT DATA S_PDU (TYPE 0)
IS NODE DELIVERY CONFIRMATION (ARQ) REQRED ?
YES
NO
HAS A C_UNIDATA_REQUEST_ CONFIRM BEEN RCVED FROM CHANNEL ACCESS SUBLAYER ?
YES
NO
HAS A C_UNIDATA_REQUEST_ REJECTED BEEN RCVED FROM CHANNEL ACCESS SUBLAYER ?
NO
YES
IS CLIENT DELIVERY CONFIRMATION ALSO REQRED ?
NO
YES
HAS A TYPE 1 S_PDU (DATA DELIVERY CONFIRM) BEEN RCVED FROM PEER ?
YES
NO
HAS A TYPE 2 S_PDU (DATA DELIVERY FAIL) BEEN RCVED FROM PEER ?
NO
YES
INFORM CLIENT OF UNSUCCESSFUL NODE DELIVERY
INFORM CLIENT OF SUCCESSFUL NODE DELIVERY
INFORM CLIENT OF SUCCESSFUL CLIENTDELIVERY
INFORM CLIENT OF UNSUCCESSFUL CLIENT DELIVERY
DATA S_PDU DONE

Figure A-7 (a): Data Exchange Procedures: SENDING PEER

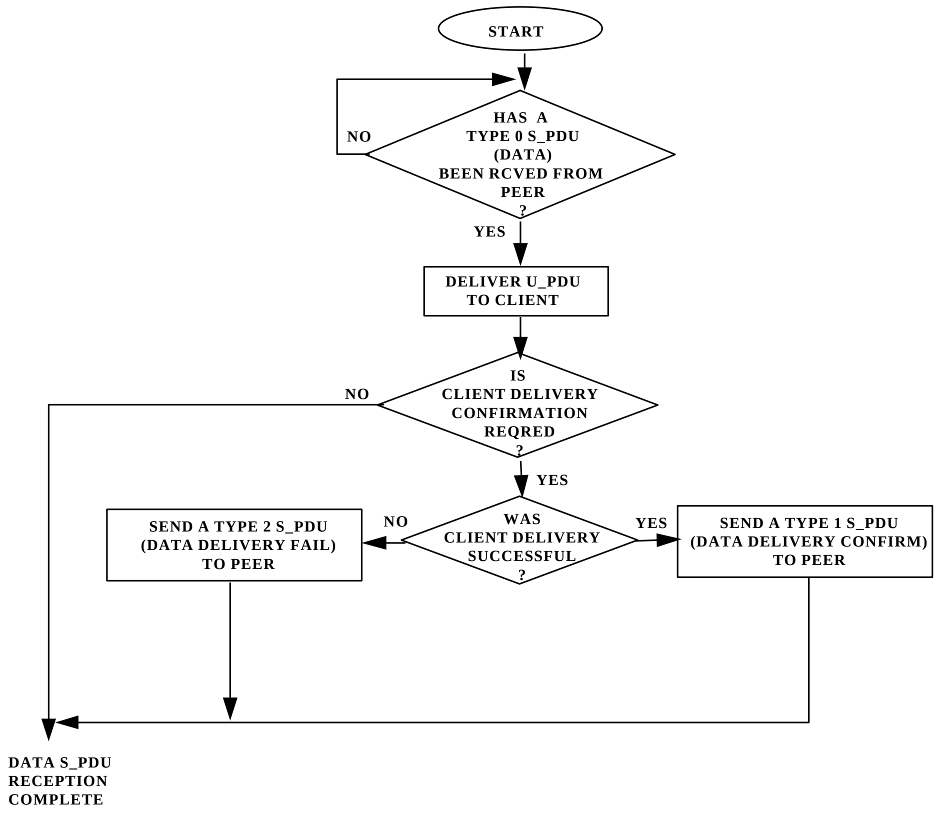

START
HAS A TYPE 0 S_PDU (DATA) BEEN RCVED FROM PEER ?
NO
YES
DELIVER U_PDU TO CLIENT
IS CLIENT DELIVERY CONFIRMATION REQRED ?
NO
YES
WAS CLIENT DELIVERY SUCCESSFUL ?
NO
YES
SEND A TYPE 2 S_PDU (DATA DELIVERY FAIL) TO PEER
SEND A TYPE 1 S_PDU (DATA DELIVERY CONFIRM) TO PEER
DATA S_PDU RECEPTION COMPLETE

Figure A-7 (b): Data Exchange Procedures: RECEIVING PEER

Annex B: Channel Access Sublayer (mandatory)

The functions required of the channel access sublayer are quite limited in the ship-shore application.

B.1 Channel Access Sublayer Service Definition

The Channel Access Sublayer provides services to the Subnetwork Interface Sublayer. These services are:

1. Execute Physical Link "*Make and Break*" requests.

2. Notify changes in the state of the Physical Link.

3. Accept S_PDUs (in the appropriate primitive) for transmission.

4. Deliver received S_PDUs (in the appropriate primitive).

In order to provide these services, the Channel Access Sublayer implements a protocol which specifies the tasks that must be executed and the rules that must be obeyed by the sublayer. While a number of different channel access protocols are possible, the one that is suitable for this document is referred to as the type 1 protocol.

B.2 Channel Access Protocol Type 1 and C_PDUs

The Type 1 Channel Access Protocol supports the following subnetwork configuration:

1. Pairs of Nodes are linked "point-to-point" on a "common" HF frequency channel or on dedicated frequency channels selected from a pool of assigned frequencies by an external process[5] which is not under the control of any of the sublayers (an ALE sublayer is not present or not used).
2. The co-ordination of the making and breaking of Physical Links between two nodes (after a common frequency has already been selected by an external process) is performed solely by the Channel Access Sublayer.
3. The Nodes may or may not be within ground-wave distances.

B.2.1 Type 1 Channel Access Sublayer Peer-to-Peer Communication

The type 1 Channel Access Sublayer communicates with peer sublayers in other nodes in order to:

1. Make and break physical links

2. Exchange S_PDUs submitted by the Subnetwork Interface Sublayer.

[5] An appropriate frequency may be simply selected manually by an operator or by an external (semi)automated process e.g. "available" frequencies are negotiated between a node which needs to establish a link and a master station which "manages" the assigned frequency pool. Protocol type 1 provides no mechanism for automatic interaction between the external process and the Channel Access Sublayer. The Channel Access Sublayer "assumes" that a common frequency channel has already been found.

B.2.1.1 Type 1 Channel Access Sublayer Data Protocol Units (C_PDUs)

- Six C_PDU types are defined for the type1 Channel Access Protocol. The generic encoding of these C_PDUs is shown in Figure B-1.

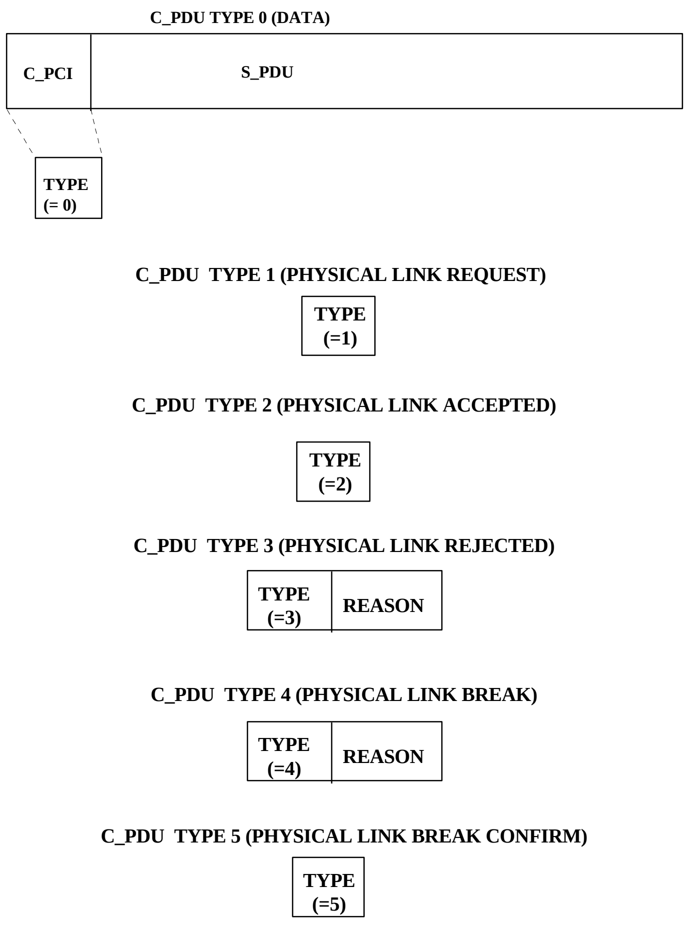

C_PDU TYPE 0 (DATA)
C_PCI
S_PDU
TYPE (= 0)
C_PDU TYPE 1 (PHYSICAL LINK REQUEST)
TYPE (=1)
C_PDU TYPE 2 (PHYSICAL LINK ACCEPTED)
TYPE (=2)
C_PDU TYPE 3 (PHYSICAL LINK REJECTED)
TYPE (=3)
REASON
C_PDU TYPE 4 (PHYSICAL LINK BREAK)
TYPE (=4)
REASON
C_PDU TYPE 5 (PHYSICAL LINK BREAK CONFIRM)
TYPE (=5)

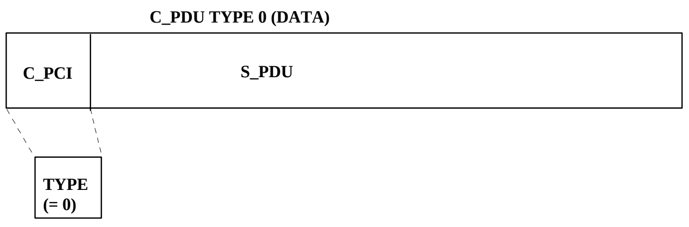
Figure B-1. Generic Encoding of C_PDUs used in the Type 1 Channel Access Protocol

The description of the C_PDUs follows:

Type :

“0” = DATA C_PDU

Encoding :



Description :

The first field of the C_PDU identifies its type while the second part of the C_PDU contains the S_PDU.

Type :

“1” = PHYSICAL LINK REQUEST

Encoding :

TYPE (=1)

Description :

This C_PDU is transmitted by the peer sublayer which requests the making of the Physical Link. This C_PDU consists only of one field: *Type*. When this C_PDU is transmitted the local Node is not linked to another Node and therefore the ARQ transmission mode is not supported by the Data Transfer Sublayer. This C_PDU shall be sent by the Data Transfer Sublayer using type 8 D_PDUs. The peer sublayer which receives this C_PDU must respond with either a type 2 or type 3 C_PDU described next.

Type :

“2” = PHYSICAL LINK ACCEPTED

Encoding :

TYPE (=2)

Description :

This C_PDU is transmitted by a peer sublayer as a positive response to the reception of a type 1 C_PDU (PHYSICAL LINK REQUEST). This C_PDU consists only of one field, i.e. *Type*. When this C_PDU is transmitted the local Node is not linked to another Node and therefore the ARQ transmission mode is not supported by the Data Transfer Sublayer. This C_PDU shall be sent by the Data Transfer Sublayer using type 8 D_PDUs.

Type :

“3” = PHYSICAL LINK REJECTED

Encoding :

TYPE (=3)	**RESERVED**

Description :

This C_PDU is transmitted by a peer sublayer as a negative response to the reception of a type 1 C_PDU (PHYSICAL LINK REQUEST). This C_PDU consists of two fields: *Type and Reason.* The reserved field will be used in the future to identify the reason for which the Physical Link Request is being rejected. . Until the reasons and codes are defined, the reserved bits shall be set to 0. Possible values include: Broadcast-Only-Node, Higher-Priority-Link-Request-Pending, etc.[6] When this C_PDU is transmitted the local Node is not linked to another Node and therefore the

[6] “Link busy” is not included in the list of possible reasons for rejecting node’s request to establish a link since this function is provided by the WARNING frame type of the Data Transfer Sublayer. See Section 3.3.6.

ARQ transmission mode is not supported by the Data Transfer Sublayer. This C_PDU shall be sent by the Data Transfer Sublayer using type 8 D_PDUs.

Type :

“4” = PHYSICAL LINK BREAK

Encoding :

TYPE (=4)	**RESERVED**

Description :

This C_PDU is transmitted by the peer sublayers in order to request the breaking of the Link. This C_PDU consists of two fields: *Type and Reason.* The *reserved* field will be used in the future to identify the reason for which the Physical Link Request is being broken. Possible values can be : Higher-Layer-Request, Going-Into-Broadcast-Only, etc. Until the reasons and codes are defined, the reserved bits shall be set to 0. The peer sublayer which receives this C_PDU must immediately declare the Physical Link as broken and respond with a type 5 C_PDU described next. This C_PDU shall be sent by the Data Transfer Sublayer using type 8 D_PDUs. The reason is that since the receiving peer will immediately declare the Link as broken, its local Data Transfer Sublayer will not have time to send ARQ ACKs for this C_PDU.

Type :

“5” = PHYSICAL LINK BREAK CONFIRM

Encoding :

TYPE (=5)

Description :

This C_PDU consists of only one field :*Type*. This C_PDU must be transmitted by the peer sublayer as a response to a type 4 “PHYSICAL LINK BREAK” C_PDU. This C_PDU shall be sent by the Data Transfer Sublayer using type 8 D_PDUs. Upon receiving a type 5 C_PDU, the peer which initiated the breaking of the Link can declare the Link as broken.

Bit-Maps of the C_PDUs used in the Type 1 Channel Access Protocol are shown in Figure B-2. The bits shall be mapped into fields in accordance with CCITT V.42, 8.1.2.3, which states that:

- when a field is contained within a single octet, the lowest bit number of the field represents the lowest-order value
- when a field spans more than one octet, the order of bit values within each octet progressively decreases as the octet number increases. The lowest bit number associated with the field represents the lowest-order value.

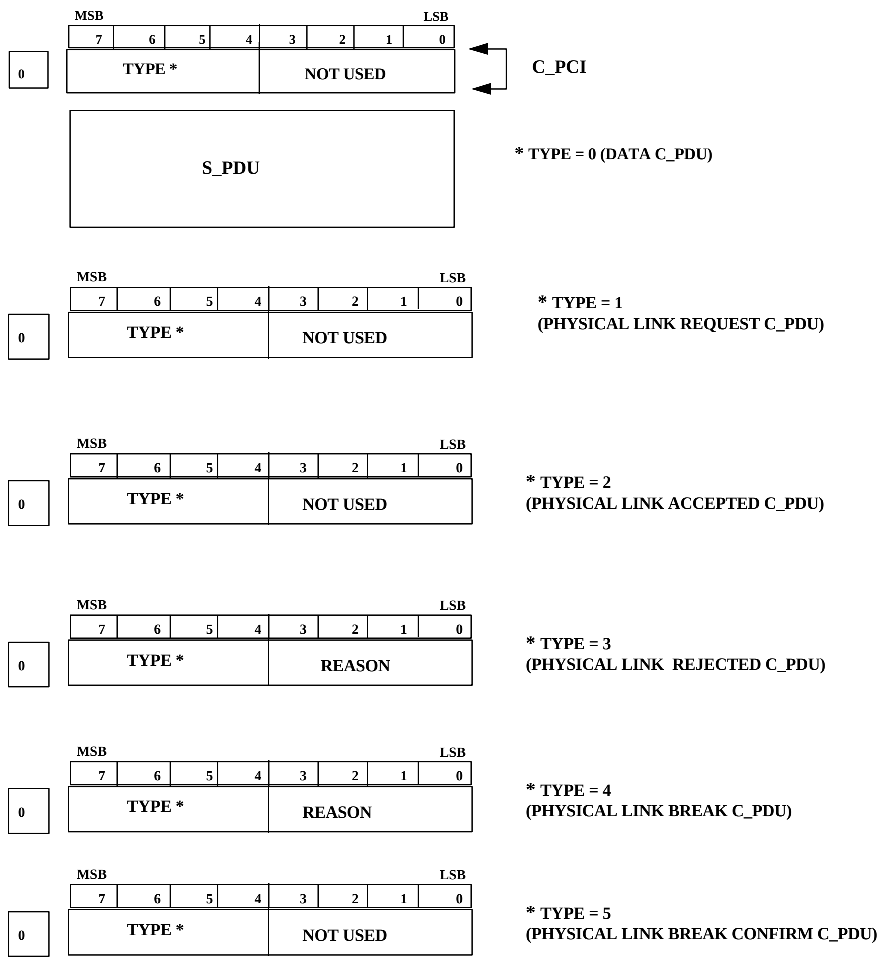

MSB
LSB
7 6 5 4 3 2 1 0
0
TYPE *
NOT USED
C_PCI
S_PDU
* TYPE = 0 (DATA C_PDU)
* TYPE = 1
(PHYSICAL LINK REQUEST C_PDU)
* TYPE = 2
(PHYSICAL LINK ACCEPTED C_PDU)
REASON
* TYPE = 3
(PHYSICAL LINK REJECTED C_PDU)
* TYPE = 4
(PHYSICAL LINK BREAK C_PDU)
* TYPE = 5
(PHYSICAL LINK BREAK CONFIRM C_PDU)

Figure B-2. Bit Maps of C_PDUs used in Type 1 Channel Access Protocol

B.2.1.2 Type 1 Channel Access Sublayer Peer-to-Peer Communication Protocol

All Channel Access Sublayer peer-to-peer communications, that is, sending type 1, 2, 3, 4, and 5 C_PDUs shall be done via type 8 (expedited non-ARQ) D_PDUs.

B.2.1.2.1 Protocol for Making a Physical Link

The node which requests the physical link is referred to as the **Caller**, while the node which receives the request will be referred to as the **Called** node. The procedure for making the physical link consists of the following steps:

Step 1-Caller.

The Channel Access Sublayer sends a Type 1 C_PDU (PHYSICAL LINK REQUEST). Upon sending this C_PDU the Channel Access Sublayer should start a timer which is set to a value greater than or equal to the maximum time required by the Called Node to send its response (this time depends on the modem parameters, number of re-transmissions, etc.).

Step 2-Called.

On receiving the type 1 C_PDU, the called node responds with either a type 2 or type 3 C_PDU. After the type 2 C_PDU is sent, the called channel access sublayer shall declare physical link made (transition to DATA state). If further type 1 C_PDUs are received from the same address after the link is made, the CAS shall again reply with they type 2 C_PDU.

Step 3. Caller

There are two possible outcomes to the link process: success or failure. Upon receiving a Type 2 C_PDU (PHYSICAL LINK ACCEPTED), the calling Channel Access Sublayer declares the Physical Link as *successfully Made*.

Upon receiving a Type 3 C_PDU (PHYSICAL LINK REJECTED), the Channel Access Sublayer declares the Physical Link as *Failed*.

Upon expiration of its timer without any response having been received from the remote node, the Channel Access Sublayer repeats step 2 and awaits again a response from the remote node. After having repeated step 2 a maximum number of times (left as a configuration parameter) without any response having been received from the remote node, the Channel Access Sublayer declares the Physical Link as *Failed*.

Figure B-3 (a) and (b) show the procedures followed by the Caller and Called Channel Access Sublayers for Making a Physical Link.

Figure B-3 (a). Type 1 Channel Access Sublayer protocol for Making a Physical Link (Caller Peer)

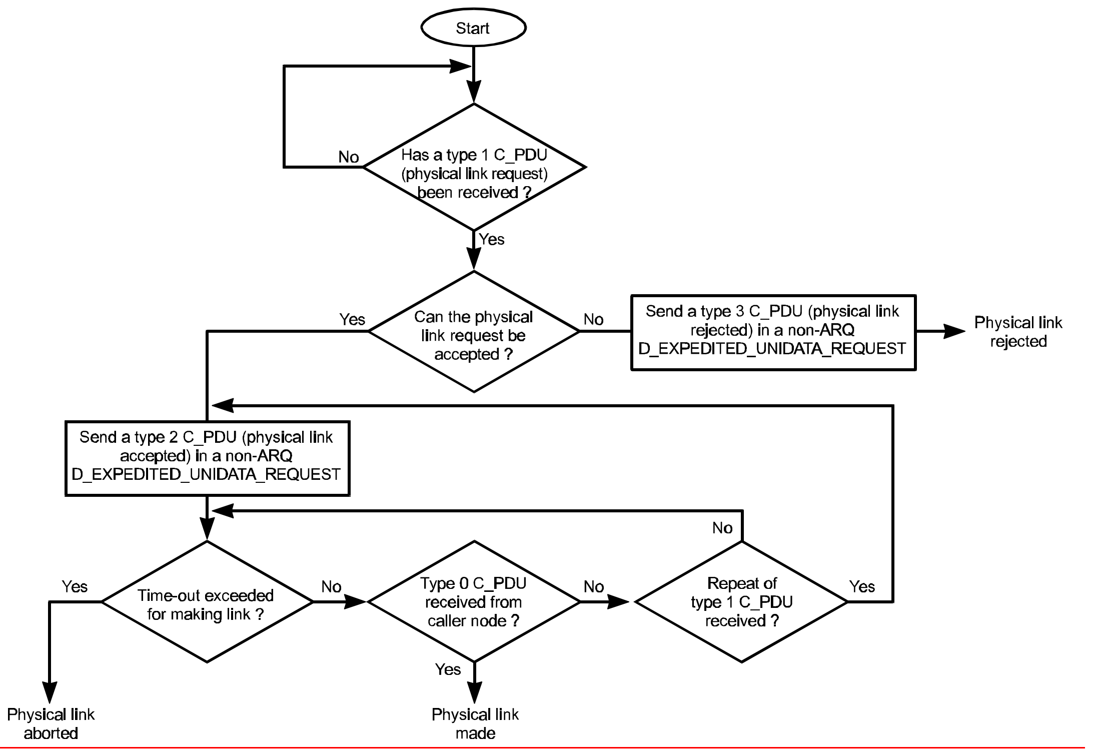

Start
No
Has a type 1 C_PDU (physical link request) been received ?
Yes
Yes
Can the physical link request be accepted ?
No
Send a type 3 C_PDU (physical link rejected) in a non-ARQ D_EXPEDITED_UNIDATA_REQUEST
Physical link rejected
Send a type 2 C_PDU (physical link accepted) in a non-ARQ D_EXPEDITED_UNIDATA_REQUEST
No
Yes
Time-out exceeded for making link ?
No
Type 0 C_PDU received from caller node ?
No
Repeat of type 1 C_PDU received ?
Yes
Yes
Physical link aborted
Physical link made

Figure B-3 (b). Type 1 Channel Access Sublayer Protocol for Making a Physical Link

(Called Peer)

After having declared the Physical Link as successfully made, the peer Channel Access Sublayers do not carry out any further handshake in order to confirm the successful operation of the Link.

B.2.1.2.2 Protocol for Breaking a Physical Link

The node which requests the breaking of the Physical Link will be referred to as the **Initiator**, while the node which receives the request will be referred to as the **Responder**. The procedure for breaking the Physical Link consists of the following four steps:

Step 1: Initiator

The Channel Access Sublayer sends a type 4 C_PDU (PHYSICAL LINK BREAK). Upon sending this C_PDU the Channel Access Sublayer should start a timer which is set to a value greater than or equal to the maximum time required by the Called Node to send its response (this time depends on the modem parameters, number of re-transmissions, etc.).

Step 2: Responder

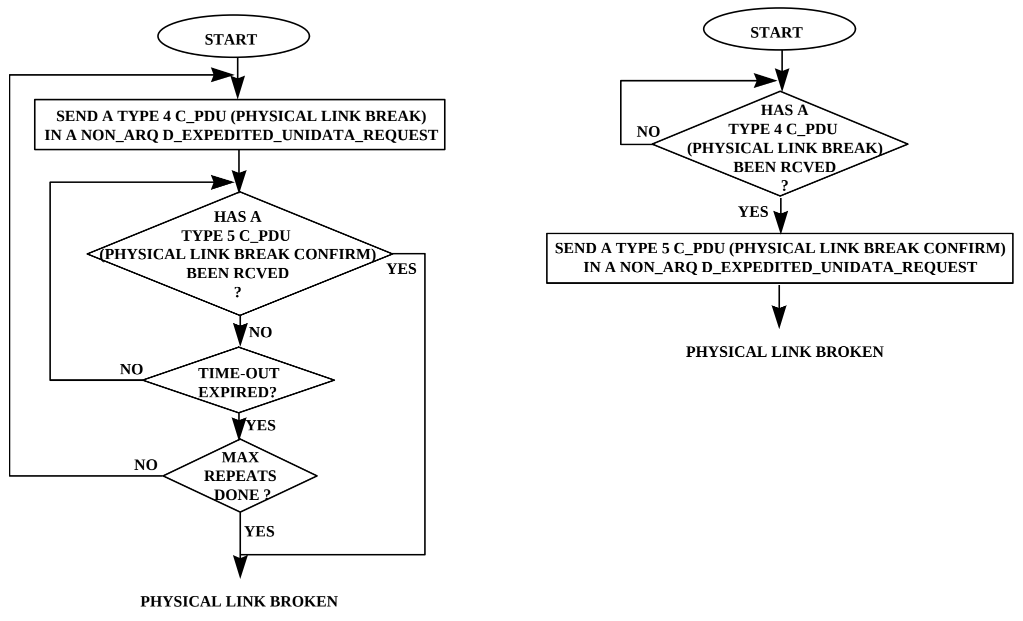
Upon receiving the type 4 C_PDU the Channel Access Sublayer declares the Physical link as *broken* and responds by sending a type 5 C_PDU (PHYSICAL LINK BREAK CONFIRM).

Step 3: Initiator

Upon receiving a Type 5 C_PDU (PHYSICAL LINK BREAK CONFIRM), the Channel Access Sublayer declares the Physical Link as *broken*.

Upon expiration of its timer without any response having been received from the remote node, the Channel Access Sublayer repeats step 1 and waits again for a response from the remote node. After having repeated step 1 a maximum number of times (left as a configuration parameter) without any response having been received from the remote node, the Channel Access Sublayer declares the Physical Link as *broken*.

Figures C-4 (a) and (b) show the procedures followed by the Initiator and Responding Channel Access Sublayers for Breaking a Physical Link.


START
SEND A TYPE 4 C_PDU (PHYSICAL LINK BREAK) IN A NON_ARQ D_EXPEDITED_UNIDATA_REQUEST
HAS A TYPE 5 C_PDU (PHYSICAL LINK BREAK CONFIRM) BEEN RCVED ?
YES
NO
NO
TIME-OUT EXPIRED?
YES
NO
MAX REPEATS DONE ?
YES
PHYSICAL LINK BROKEN
START
NO
HAS A TYPE 4 C_PDU (PHYSICAL LINK BREAK) BEEN RCVED ?
YES
SEND A TYPE 5 C_PDU (PHYSICAL LINK BREAK CONFIRM) IN A NON_ARQ D_EXPEDITED_UNIDATA_REQUEST
PHYSICAL LINK BROKEN

(a) (b)

Figure B-4. Type 1 Channel Access Sublayer protocol for Breaking a Physical Link (a) Initiator Peer. (b) Responding Peer.

B.3 Protocol for Exchanging Data C_PDUs.

The sending peer accepts S_PDUs from the Subnetwork Interface Sublayer, envelopes them in a "DATA" C_PDU (by adding the C_PCI) and sends them to its receiving peer. At the receive side, the receiving peer receives DATA C_PDUs from the Data Transfer Sublayer, checks them for validity, strips off the C_PCI, and delivers the enveloped S_PDU to the Subnetwork Interface Sublayer.

Annex C: Data Transfer Sublayer
(mandatory)

The Data Transfer Sublayer is responsible for the efficient transfer of C_PDUs across the HF link. Usually, but not always, this means the error free delivery of C_PDUs to the Channel Access Sublayer. Efficient transmission of D_PDUs (which contain the C_PDUs) over the HF channel is achieved principally by two mechanisms: segmentation of C_PDUs into smaller D_PDUs and by the selective repetition by the sending node of D_PDUs which were received in error.

C.1 Sublayer service definition

The Data Transfer Sublayer provides a data transfer service with different delivery modes (ARQ~~reliable~~ and non-ARQ ~~reliable~~) and “sub-modes” which influence the characteristics of the service.

In the ARQ~~reliable~~ mode, the following submodes may be specified:

- node and client level delivery confirmation
- regular data service
- expedited data service[7]
- deliver C_PDUs to the peer Channel Access Sublayer in the order submitted or in the order they are received

By definition, in the reliable mode only complete error free C_PDUs are ultimately delivered to the clients attached to the subnet.

In the non-ARQ ~~reliable~~ mode, which is commonly known as a broadcast mode, the following submodes may be specified:

- regular data service
- expedited data service

Note that, in the non reliable mode:

1. Data is delivered to/from and processed within the Data Transfer Sublayer as standard PDUs.

2. “In order” delivery of C_PDUs is not guaranteed.
3. Delivery of complete C_PDUs is not guaranteed.

Finally, it is noted that it is possible to mix frames corresponding to the ARQ and non ARQ modes of operation in the same transmission interval.

C.2 Communications with lower sublayers

Communications with adjacent sublayers is normally handled with primitives. Typically, a primitive will contain the PDU from the layer above plus some header information. In the case of the data transfer sublayer, the next lower layer is the HF modem (either directly or via

[7] Note that despite its use of an ARQ protocol, this submode is not reliable. Under certain conditions, data queued for expedited service will be discarded.

the link encryption device). In this case, the primitive is simply one or more D_PDUs. The hardware handshaking can be considered a form of header for the D_PDUs.

C.3 Structure of sublayer protocol data units (D_PDUs)

In order to provide the data transfer services described earlier, the Data Transfer Sublayer exchanges D-PDUs with its peer. The ten D_PDU types which are currently defined to support the ARQ and non ARQ data transfer service are shown in the table below. The same D_PDU types support half duplex and a full duplex operating modes.

Table C-1. D_PDUs

D_PDU Type	Function	D_PDU type
DATA-ONLY	One-way data transfer	0
ACK-ONLY	Acknowledgement of one-way data transfer	1
DATA-ACK	Two-way data transfer	2
RESET/WIN-RESYNC	Reset/resynchronize peer protocol entities (ie, ARQ window sync)	3
EXPEDITED-DATA-ONLY	One way expedited data transfer	4
EXPEDITED-ACK-ONLY	Acknowledgement of one way expedited data transfer	5
MANANGEMENT	Subnet management peer-to-peer message transfer	6
NON-ARQ DATA	Non-ARQ data	7
EXPEDITED NON-ARQ DATA	Expedited non-ARQ data	8
WARNING	Sent in response to unexpected or unrecognized D_PDU type	15

Different D_PDU types may be combined in a transmission, subject to limitations imposed by the state which the data link sublayer of a node is in. These states and the corresponding rules are given in section C.4. When receiving, a node must continually search for D_PDU headers [using sync and CRC], examine each received D_PDU header and act accordingly. When D_PDU types are combined in a transmission, the previously advertised EOT shall ~~must~~ not be violated.

The data link protocol for transfer of D_PDU types 0, 2, and 4 uses a selective repeat ARQ. The data link protocol for transfer of type 6 D_PDUs is a stop and wait ARQ. Type 7, 8, and 15 D_PDUs are not acknowledged.

C.3.1 General D_PDU structure

The structure of the various D_PDUs shall be as shown in Figure C-1.

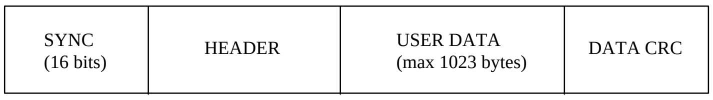


Figure C-1 (a). Format for DATA-ONLY, DATA-ACK, EXPEDITED DATA-ONLY, NON-ARQ DATA, and EXPEDITED NON-ARQ DATA D_PDUs

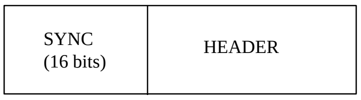


Figure C-1 (b). Format for all other D_PDU types (1, 3, 5, 6 and 15)

All D_PDUs, regardless of type, begin with the same 16 bit synchronisation sequence. The 16 bit sequence shall be the 16-bit Maury-Styles sequence shown below, with the least significant bit (LSB) transmitted first:

(MSB) 1 1 1 0 1 0 1 1 1 0 0 1 0 0 0 0 (LSB)

~~Unless specified otherwise, the order of bit transmission shall be as described in CCITT V.42 paragraph 8.1.2.2, which specifies that the least significant bit (LSB, bit 0 in the figures below) of byte 0 of all D_PDUs shall be transmitted first. The remaining bytes shall be transmitted sequentially, also beginning with the LSB of each byte.~~

The first ~~3~~4 bytes of all headers contain the same fields: a 4 bit type field, which identifies the type of D_PDU; an engineering order wire (EOW) field or MANAGEMENT message field (12 bits); an end of transmission (EOT) field (8 bits), and one byte which indicates both the size of the address field (3 bits) and the size of the header (5 bits). The last two bytes of every header are a CRC (these bytes are not included in the size of header value).

The next 0 to 7 bytes of every header contain address information.

Bits shall be mapped into fields in accordance with CCITT V.42, 8.1.2.3, which states that:

- when a field is contained within a single octet, the lowest bit number of the field represents the lowest-order value
- when a field spans more than one octet, the order of bit values within each octet progressively decreases as the octet number increases. The lowest bit number associated with the field represents the lowest-order value.

Unless specified otherwise, the order of bit transmission shall be as described in CCITT V.42 paragraph 8.1.2.2, which specifies that the least significant bit (LSB, bit 0 in the figures below) of byte 0 of all D_PDUs shall be transmitted first. The remaining bytes shall be transmitted sequentially, also beginning with the LSB of each byte.

C.3.1.1 D_PDU type

The D_PDU types are as defined in the table above and the figures below. The ~~hex~~ value of the D_PDU type number shall be used to indicate the D_PDU type. The four bits available allow for 16 types of D_PDUs, ten of which are defined in this STANAG. The MSB of the D_PDU type shall be placed in the MSB of the type field.

C.3.1.2 Engineering orderwire (EOW)

The 12 bit EOW field carries EOW messages (except for the MANAGEMENT D_PDU, where this field carries the first 12 bits of the MANAGEMENT message). EOW messages are advisory and are not explicitly acknowledged (while the D_PDUs of which they are a part may be). The EOW messages are used in this STANAG for adaptive control of the modem data rate. If a node wishes to communicate to another node the optimum modem data rate for transmissions directed to it, the EOW field shall be used to do so, as defined below. The following EOW messages are defined (their use will be detailed later):

Table C-2. EOW Message Types

EOW Message Type	Function	Contents
0	No recommendation (ignore message contents)	reset to 0
1	Support adaptive control of distant (peer) modem operating parameters	Recommended modem speed and interleaving setting for distant (peer) node
2	node capability	capability bitmap
3	version number	version number
4 - 8	reserved	

The type 1 EOW message allows the node producing the EOW message to specify if it must transmit and receive data at the same rate (due, for example, to modem limitations).
Type 1 EOW messages shall take precedence over all other types of EOW message.
The format of the type 0, 1, 2, and 3 EOW messages is shown in Figure C-2.

MSB LSB

11	10	9	8	7	6	5	4	3	2	1	0
0	**0**	**0**	**0**	**reset to 0**							
TYPE											

Figure C-2 (a). EOW Message Type 0 Format

MSB LSB

11	10	9	8	7	6	5	4	3	2	1	0
0	**0**	**0**	**1**	**Data Rate**				**Interleaving**		**Other parameters**	
TYPE											

Figure C-2 (b). EOW Message Type 1 Format

MSB LSB

11	10	9	8	7	6	5	4	3	2	1	0
0	**0**	**1**	**0**	**capability bit map**							
TYPE											

Figure C-2 (c). EOW Message Type 2 Format

MSB LSB

11	10	9	8	7	6	5	4	3	2	1	0
0	**0**	**1**	**1**	**version number**							
TYPE											

Figure C-2 (d). EOW Message Type 3 Format

The TYPE field of the message shall be filled with the hexadecimal value of the appropriate EOW message type (units only), with the LSB of the TYPE value placed in the LSB of the TYPE field. The version number shall be all zeros for this version. The other fields are filled as shown in table C-3.

Table C-3 (a). Values for Type 1 EOW Message

Data Rate Parameter
Message Type 1

MSB - LSB	Interpretation
0 0 0 0	75 bps
0 0 0 1	150 bps
0 0 1 0	300 bps
0 0 1 1	600 bps
0 1 0 0	1200 bps
0 1 0 1	2400 bps
0 1 1 0	3200 bps
0 1 1 1	3600 bps
1 0 0 0	4800 bps
1 0 0 1	6400 bps
1 0 1 0	7200 bps
1 0 1 1	9600 bps
others	reserved

Interleaver Parameter
Message Type 1

MSB - LSB	Interpretation
0 0	no interleaving
0 1	short interleaving
1 0	long interleaving
1 1	reserved

Table C-3 (b). Values for Type 1 EOW Message (Other Parameters)

MSB - LSB	Interpretation
0 0	node sending EOW has independent data rate (change applies only to transmit data rate at remote node)
0 1	Tx and Rx data rate at node sending EOW must be equal (change will apply to both transmitand receivedata rates)

Table C-3 (c). Capability Bit Map

bit	meaning
7 (MSB)	Adaptive modem parameters (DRC) capable [note 1] (0 = no, 1 = yes)
6	STANAG 4529 available [note 2] (0 = no, 1 = yes)
5	MIL-STD-188-110A available [note 2] (0 = no, 1 = yes)
4	extended data rate capable [note 3] (0 = no, 1 = yes)
3	full duplex supported [note 4] (0 = no, 1 = yes)
2	split frequency supported [note 4] (0 = no, 1 = yes)
1	non-ARCS ALE capable
0 (LSB)	ARCS capable (0 = no, 1 = yes)

Notes
1. If a node is DRC capable, it must implement at minimum 75 bps through 2400 bps (1200 bps for STANAG 4529) with short and long interleaving in accordance with the relevant document.
2. All nodes shall have, at minimum, the STANAG 4285 waveform available.
3. Annex G describes waveforms for data rates above 2400 bps.
4. A full duplex node must have split frequency operation available.

The type 1 EOW message is used to advise the other node of a recommended or requested transmission data rate. The reception of a type 1 EOW message will, in general, initiate a data rate change procedure which will be defined later. If a node has a type 1 EOW message to send, the message should be placed in as many D_PDUs as possible to insure reception. The type 1 EOW message shall be placed in at least 5 D_PDUs in one transmission, unless the node has less than 5 D_PDUs to send, in which case it shall be placed in all D_PDUs.

Use and definition of EOW Type 2 message

This message is provided to ~~give the capability to~~ allow a node to advise a distant node of its capabilities. This message is intended primarily to supplement the DRC procedures described later in this section. It also gives information about other capabilities such as full duplex operation. The capabilities bitmap shall be as shown in Table C-3 (c).

Use and definition of EOW Type 3 Message

This message is provided to allow a node to advise a distant node of the version number of STANAG 5066 which it implements. The version number specified for the version field of the S_primitives (Annex A) shall be mapped into the version number field of the Type 3 EOW message. The Type 3 EOW message shall be placed in all non-ARQ (Type 7 and Type 8) D_PDUs.

C.3.1.3 End of Transmission (EOT)

The 8 bit EOT field provides an estimate of the time remaining in the current transmission interval. This information is provided for the timing of exchanges between peer nodes to minimize collisions and the link turnaround time (time between the end of a transmission by one node and the start of a transmission by another node). The number in this field shall be a binary number expressing of the number of half seconds remaining in the current transmission from the beginning of the current D_PDU, including sync bytes. ~~The bits shall be mapped into the field in accordance with CCITT V.42, 8.1.2.3, which states that:~~

- ~~when a field is contained within a single octet, the lowest bit number of the firld represents the lowest-order value~~
- ~~when a field spans more than one octet, the order of bit values within each octet progressively decreases as the octet number increases. The lowest bit number associated with the field represents the lowest-order value.~~

When operating in half duplex mode, ~~In all states,~~ a node shall not make a transmission during a transmission by another node (ie, before the EOT expires). Once an EOT is sent, the EOT in each subsequent D_PDU in that transmission shall contain a consistent (that is, monotonically decreasing) EOT and the system shall stop transmitting and shift to receive mode at the advertised time.

When a node is configured for and operating in full duplex or broadcast only mode, the EOT field shall be filled with all zeros.

C.3.1.4 Address

Each D_PDU transmitted by a node shall contain the address of the source node (which shall be identical for all D_PDUs transmitted by that node) and the address of the destination. Half of the bits are assigned to the source and the other half to the destination. The first half (in order of transmission) shall be the destination address and the second half shall be the source address. Addresses shall be in the form of a binary number; with 7 bytes available, the largest possible address field is 3.5 bytes, or 28 bits, for each of the user and the destination. This allows more than 268 million addresses, if the maximum field size is used. Since the header must be made up of an integer number of bytes, addresses are available in 4 bit increments of size: 4 bits, 1 byte, 1.5, 2, 2.5, 3, and 3.5 bytes. Each byte or fractional byte of an address shall be represented by a decimal number, and the binary equivalent shall be mapped into the corresponding byte. Any fractional bytes shall be mapped into the first (leftmost) non-zero number of the address. For example, if 3.5 bytes are used, the address would be expressed as w.x.y.z, where w can be any value from 0 to 15, and x, y and z can be any value from 0 to 255. The value w will represent the most significant bits, and z the least significant bits of the address. The address bits will be mapped into the address field by placing the MSB of the address into the MSB of the first byte of the address field, and the LSB into the LSB of the last byte of the address field (consistent with CCITT V.42, 8.1.2.3).

Leading address bytes which are zero may be dropped from the address. Trailing address bytes which are zero shall be sent.

C.3.1.5 Cyclic Redundancy Check (CRC)

The last two bytes of every D_PDU header are a 16 bit CRC. The CRC shall be calculated using the polynomial $x^{16} + x^{12} + x^5 + 1$, in the manner described in CCITT V.41 part 2 ("Encoding and Checking Process"). When calculating the CRC, the shift registers shall be initially set to all zeros. The CRC shall be calculated over all bits in the header, excluding the sync bytes. A node shall process the information contained in a header with a valid CRC, regardless of the result of the CRC over any data that may be a part of the D_PDU.

The CRC bits shall be mapped into the CRC bytes of the header by placing the MSB of the CRC into the LSB of the first byte of the CRC field, and the LSB of the CRC into the MSB of the last byte of the CRC field (as specified in CCITT V.42, 8.1.2.3, Figure 7: 16 bit FCS). This will result in the MSB of the most significant byte of the CRC being sent first, followed by the remaining bits in descending order (which is consistent with the order of transmission for CRC bits specified in CCITT V.41 part 2).

The following C code can be used to calculate the CRC value according to the CCITT polynomial.

```
/* CRC16 According to the CCITT standard */
/* Polynomial x(16) + x(12) + x(5) + 1 */
unsigned short CRC16(unsigned char DATA, unsigned short CRC)
{
unsigned char I, Bit;

for (i=0x01; i; i<<=1)
{
Bit=(((CRC & 0x0001) ? 1:0)^((Data&i) ? 1:10));
CRC>>=1;
if(Bit) CRC^=0x8408;                    /*  CRC-CCITT  polynomial */
}
return(CRC);
}
```

The result of this code for the D_PDU described below is 0xCE5D. Note that this code calculates the CRC bytes in the proper order for transmission as defined above; no reversal of bytes is required.

D_PDU used in above CRC calculation:

D_PDU Type:	Warning
EOW:	Type 0
EOT:	all zeros
Address size:	2
Destination address:	0.0.0.5 (leading zeros not encoded)
Source address:	0.0.0.100 (leading zeros not encoded)
Received D_PDU type:	0
Reason warning sent:	2
CRC:	0xCE5D

The full D_PDU is then (including sync bytes):
0x90, 0xEB, 0xF0, 0x00, 0x00, 0x47, 0x05, 0x64, 0x02, 0x5D, 0xCE

~~The structure of the header for each type of D_PDU, and the components specific to each D_PDU type, will be described below.~~

C.3.2 DATA_ONLY D_PDU

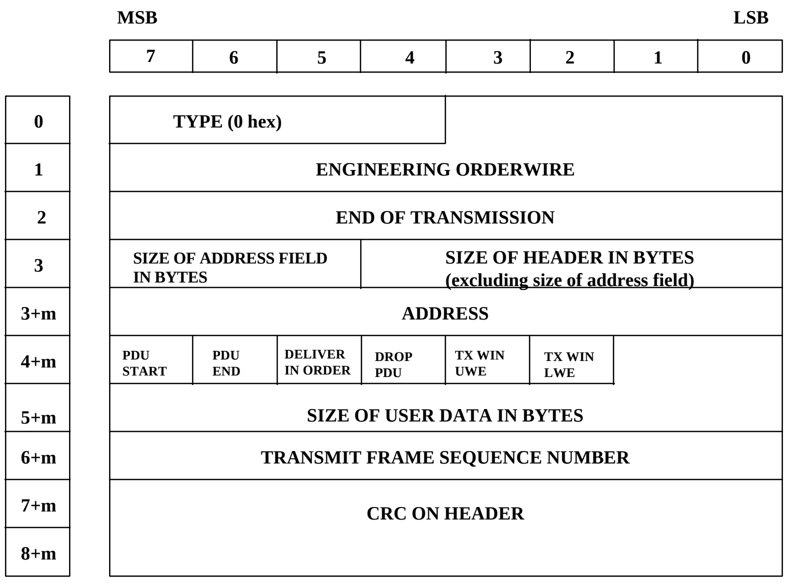


Figure C-3. Header format for DATA-ONLY D_PDU

The unique bytes of the DATA-ONLY D_PDU begin with byte 4+m, following the ADDRESS field. The first four bits of byte 4+m indicate the handling of PDUs. PDU START and PDU END are set to indicate the start and end of a PDU; when a D_PDU is received with the PDU END bit set, the link protocol should assemble and deliver the PDU, if it is complete. When the DELIVER IN ORDER bit is not set on the D_PDUs composing a C_PDU, the C_PDU shall be delivered to the upper layer as soon as it is complete. When the DELIVER IN ORDER bit is set on the D_PDUs composing a C_PDU, the C_PDU shall be delivered to the upper layer when both the following conditions are met: 1) the C_PDU is complete, and 2) all previous C_PDUs which also had the DELIVER IN ORDER bit set have been delivered. ~~set, the link protocol must deliver received PDUs in the order in which they are present in the flow control window (if this bit is not set for consecutive PDUs, it will be ignored).~~ If the DROP PDU bit is set, the protocol will discard the contents of the data field of the current D_PDU and all other previously recieved portions of the C_PDU of which the current D_PDU is a part.

The next two bits are used by the node to verify that its receive flow control window pointers (which govern the acknowledgement of DATA-ONLY and DATA-ACK D_PDUs) are in synchronization with the corresponding transmit flow control window pointers of the sending node. The TX UWE flag is set when the current D_PDU is at the TX UWE; and similarly, the TX LWE flag is set when the TRANSMIT FRAME SEQUENCE NUMBER is equal to the LWE of the transmit flow control window.

The 10 bit field SIZE OF USER DATA indicates the size of the information field in bytes. This figure does not include the data CRC. A DATA-ONLY D_PDU with zero data bytes may be sent, for example, when the DROP PDU flag is set as discussed above.

The TRANSMIT FRAME SEQUENCE NUMBER field contains the sequence number of the current D_PDU.

The format of the data portion of the D_PDU is described below.

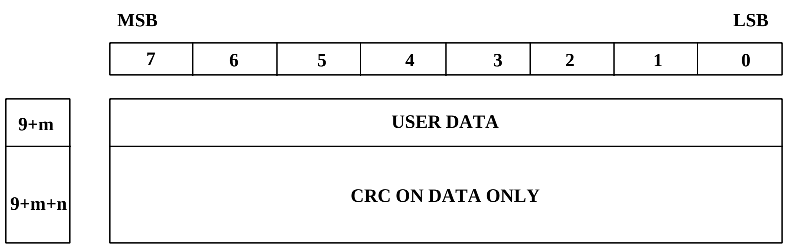


Figure C-4. Format of DATA portion of D_PDU (DATA-ONLY, DATA-ACK, NON-ARQ DATA)

D_PDUs which carry data, including DATA-ONLY, DATA-ACK, and NON-ARQ DATA D_PDUs, shall transmit the user data immediately following the corresponding D_PDU header. The last bit of the D_PDU header CRC shall be followed by the first bit of the user data.

The octets of the DATA and CRC fields are transmitted in ascending numerical order; inside an octet, bit 0 is the first bit to be transmitted. The CRC shall be computed as defined in V.41, para 2 over the data field only. The CRC bits shall be mapped into the CRC bytes by placing the MSB of the CRC into the LSB of the first byte of the CRC field, and the LSB of the CRC into the MSB of the last byte of the CRC field (as specified in CCITT V.42, 8.1.2.3, Figure 7: 16 bit FCS). This will result in the MSB of the most significant byte of the CRC being sent first, followed by the remaining bits in descending order (which is consistent with the order of transmission for CRC bits specified in CCITT V.41 part 2).

C.3.3 ACK_ONLY D_PDU

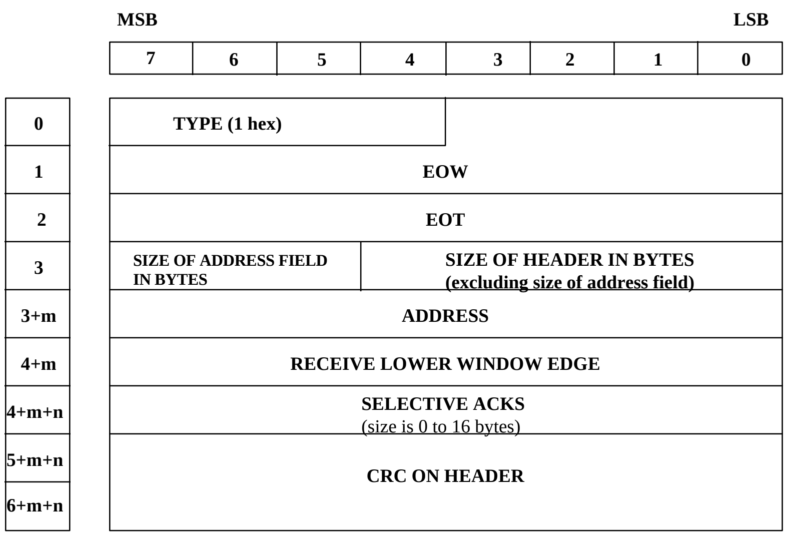

MSB
LSB
7
6
5
4
3
2
1
0
0
1
2
3
3+m
4+m
4+m+n
5+m+n
6+m+n
TYPE (1 hex)
EOW
EOT
SIZE OF ADDRESS FIELD IN BYTES
SIZE OF HEADER IN BYTES (excluding size of address field)
ADDRESS
RECEIVE LOWER WINDOW EDGE
SELECTIVE ACKS (size is 0 to 16 bytes)
CRC ON HEADER

Figure C-5. Format for ACK-ONLY D_PDU

The ACK-ONLY D_PDU is used to acknowledge received D_PDUs when the receiving node has no PDUs to send. The unique fields in the ACK-ONLY D_PDU header are the RECEIVE LWE NUMBER and the SELECTIVE ACKS fields. The RECEIVE LWE field contains the D_PDU sequence number of the “lower window edge (LWE)” pointer associated with the node’s receive ARQ flow control window. If the value in the RECEIVE LWE field is greater than the current transmit LWE, the transmitting system shall update its LWE to the value in the RECEIVE LWE field.

The SELECTIVE ACKS field can be from 0 to 16 bytes in length and contains a bit mapped representation of the ARQ status of all received D_PDUs with sequence numbers from the LWE to the UWE pointers (inclusive) of the receive flow control window (excluding the first sequence of correctly received D_PDUs). A set bit shall indicate a positive acknowledgement (D_PDU received OK) and a reset bit (0) shall indicate a negative acknowlegement.

The LSB of the first byte of the SELECTIVE ACKS field corresponds to the D_PDU whose sequence number is equal to 1 + the value in the RECEIVE LWE field of that D_PDU [the bit corresponding to RX LWE is always zero, by definition, and is therefore not sent]. Each subsequent bit in the SELECTIVE ACKs field shall represent a subsequent D_PDU in the receive flow control window. The mapping of D_PDUs to bits is shown in Figure C-6. If the bit representing the receive UWE is not the MSB of a byte, the remaining bits in the byte (until and including the MSB) shall be set to 0. ~~If the bit representing the receive UWE is the LSB of a byte, it is not necessary to send the bit (because that bit represents the next new D_PDU to be received, it will always be zero).~~ If the number in the RECEIVE LWE field is greater than the transmit LWE, all D_PDUs between the transmit LWE and the receive LWE value are acknowledged. Bits of the most significant byte of the SELECTIVE ACKS header field which correspond to D_PDU sequence numbers equal to or greater than a node’s receive buffer UWE shall be 0 (reset). No further SELECTIVE ACK bytes should be transmitted after such bits are required.

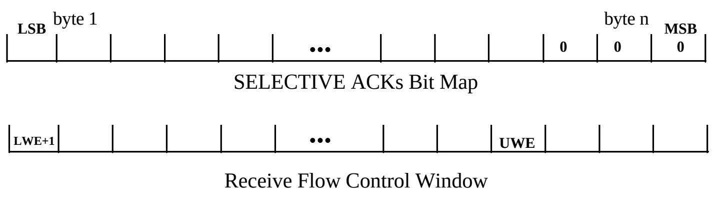


Figure C-6. Constructing the SELECTIVE ACKs Field

In order to increase reliability, while not adding to the overhead of the transmission, if a node has only a single or small number of ACK-ONLY D_PDUs to transmit, the ACK-ONLY D_PDU should ~~shall~~ be repeated. ~~as indicated in the table:~~ Table C-4 gives a minimum suggested number of times that a message should be transmitted, based on minimizing the use of stuff bits in the interleaver. Other considerations could make a larger number of repetitions desireable.

Table C-4. Minimum ~~N~~number of ACK-ONLY messages to be transmitted at various data rates when using STANAG 4285 waveform

Data rate	~~repetitions~~ ~~(~~short interleave~~)~~	~~repetitions~~ ~~(~~long interleave~~)~~
75	1	9
150	1	18
300	1	37
600	3	75
1200	7	150
2400	15	300

Note to Table C-4: when using waveforms with different interleave depth and data rates, the number of repetitions should be adjusted so as to minimise the use of “stuff bits” in the modem to fill the interleaver.

C.3.4 DATA-ACK D_PDU

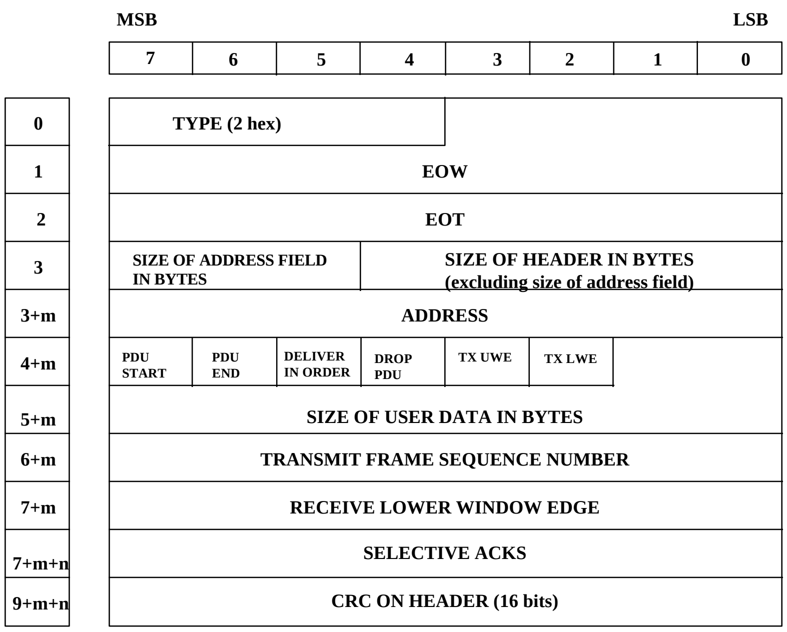

MSB
LSB
7
6
5
4
3
2
1
0
0
1
2
3
3+m
4+m
5+m
6+m
7+m
7+m+n
9+m+n
TYPE (2 hex)
EOW
EOT
SIZE OF ADDRESS FIELD IN BYTES
SIZE OF HEADER IN BYTES (excluding size of address field)
ADDRESS
PDU START
PDU END
DELIVER IN ORDER
DROP PDU
TX UWE
TX LWE
SIZE OF USER DATA IN BYTES
TRANSMIT FRAME SEQUENCE NUMBER
RECEIVE LOWER WINDOW EDGE
SELECTIVE ACKS
CRC ON HEADER (16 bits)

Figure C-7. Header format for DATA-ACK D_PDU

The DATA-ACK D_PDU is a combination of the DATA-ONLY and ACK-ONLY D_PDU types. All of the field descriptions from the previous D_PDUs apply to this one.

C.3.5 RESET/WIN_RESYNC D_PDU

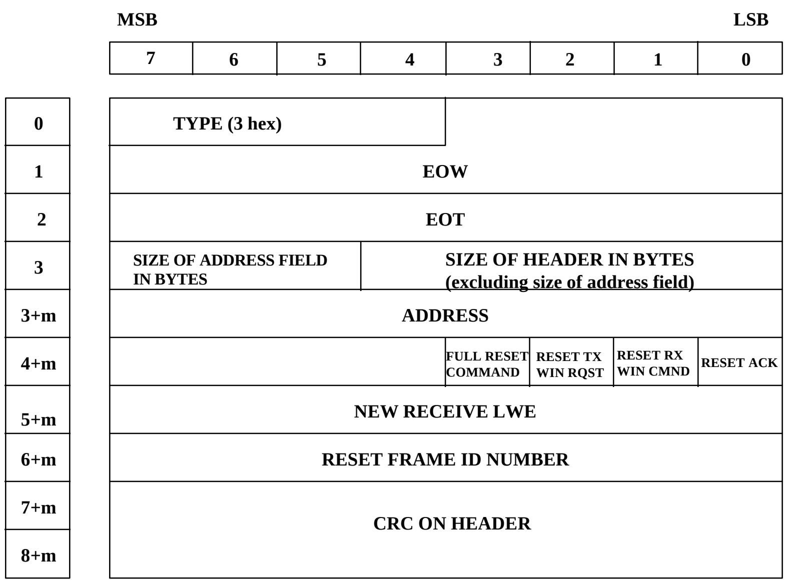


Figure C-8. Header for RESET/WIN RESYNC D_PDU

This D_PDU type is used to control the synchronisation of the transmit and receive window edges. Two major types of action can be taken using this D_PDU: first, reset, and second, re-synchronisation. This D_PDU shall be sent only when the node is in the Data state. Detailed procedures for the use of this D_PDU are described in C.4.4, and a discussion of ~~More detailed information on~~ the use of this D_PDU is given in Annex H.

Reset

A reset is initiated by sending a RESET/WIN-RESYNC D_PDU with the FULL RESET CMND flag set (value = 1). The initiating node assigns a **RESET FRAME ID #** which is derived from a sequence of D_PDU ID numbers (modulo 255) that is reserved (independently by each node) for reset/resynchronisation operations.[8] A new RESET FRAME ID NUMBER is assigned for each **reset event**. A reset event is defined as one or more D_PDUs, which may span more than one transmission interval, sent by a node to accomplish a given reset operation, e.g. initiate and receive an acknowledgement for a full reset. Thus, for example, if a node transmits multiple (i.e. redundant) RESET/WIN-RESYNC D_PDUs in the same or over several transmission intervals, each RESET/WIN-RESYNC D_PDU is assigned the same RESET FRAME ID NUMBER The node that received the reset request replies with a RESET/WIN-RESYNC D_PDU with the RESET ACK flag set (value = 1) and the other three reset/resynchronisation flags reset (value = 0). The RESET FRAME ID NUMBER field is not given a value. The effect of the FULL RESET CMND is to reset (zeroise) the Tx *and* Rx window pointers and buffers in the *currently active* peer DATA state protocol machines, i.e. at both ends of the link. Because a reset effects the transmit and receive buffers at both

[8] The term ID number is used to avoid confusion with the concept of a frame sequence number. RESET/WIN-RESYNC frames do not carry a frame sequence number.

ends of a link, it should be used only when necessary or when there will be no loss of data, e.g. before beginning data transmission over a newly established link.

In order to increase reliability, while not adding to the overhead of the transmission, if a node has only a single or small number of RESET/WIN RESYNC D_PDUs to transmit, the RESET/WIN RESYNC D_PDU should be repeated. Table C-5 gives a minimum suggested number of times that a message should be transmitted, based on minimising the use of stuff bits in the interleaver. Other considerations could make a larger number of repetitions desireable.

Table C-5. Number of RESET/WIN RESYNC messages to be transmitted at various data rates when using STANAG 4285

Data rate	repetitions (short interleave)	repetitions (long interleave)
75	1	9
150	1	18
300	1	37
600	3	75
1200	7	150
2400	15	300

Note to Table C-5: when using waveforms with different interleave depth and data rates, the number of repetitions should be adjusted so as to minimise the use of “stuff bits” in the modem to fill the interleaver. ~~The number of repetitions is selected to nearly fill the interleave buffer.~~

C.3.6 EXPEDITED DATA ONLY D_PDU

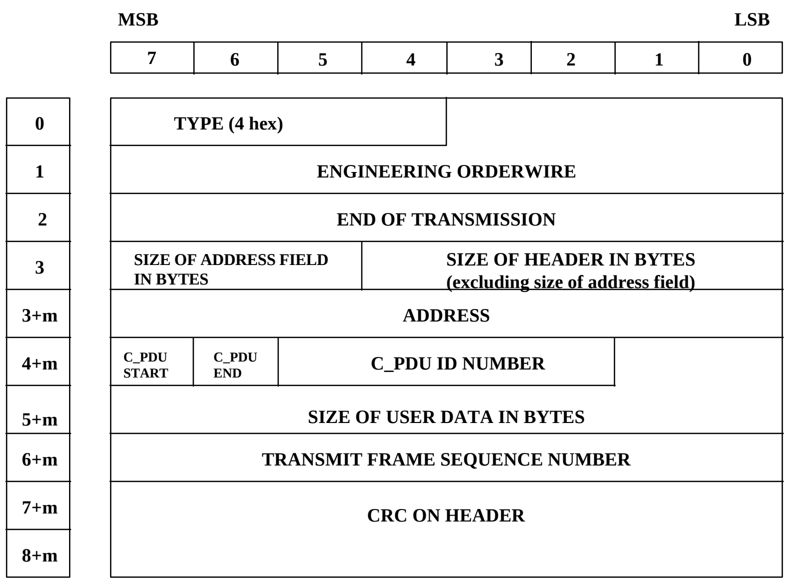

MSB
LSB
7
6
5
4
3
2
1
0
0
1
2
3
3+m
4+m
5+m
6+m
7+m
8+m
TYPE (4 hex)
ENGINEERING ORDERWIRE
END OF TRANSMISSION
SIZE OF ADDRESS FIELD IN BYTES
SIZE OF HEADER IN BYTES (excluding size of address field)
ADDRESS
C_PDU START
C_PDU END
C_PDU ID NUMBER
SIZE OF USER DATA IN BYTES
TRANSMIT FRAME SEQUENCE NUMBER
CRC ON HEADER

Figure C-9. Header for EXPEDITED DATA ONLY D_PDU

With the exception of byte 4+m, this D_PDU is identical in structure to the DATA ONLY D_PDU. The definitions given there apply here as well.

The processing of D_PDUs in the EXPEDITED DATA state differs from the processing of D_PDUs in the DATA state in the following ways:

- Expedited data is transferred using EXPEDITED DATA-ONLY and EXPEDITED ACK-ONLY D_PDUs. If required, EXPEDITED DATA-ONLY and EXPEDITED ACK-ONLY D_PDUs can be placed together in a transmission interval.
- EXPEDITED DATA-ONLY D_PDUs are assigned frame numbers from a frame sequence number queue (0, 1 ...255) that is reserved exclusively for the transmission of EXPEDITED DATA-ONLY and EXPEDITED ACK-ONLY D_PDUs. The FRAME SEQUENCE NUMBER is used, in this D_PDU, to sequence the D_PDUs which make up a C_PDU. When the C_PDU ID NUMBER is incremented, the FRAME SEQUENCE NUMBER shall be reset to zero.
- DPUs containing expedited data are not queued behind PDUs containing non-expedited data.
- Upon entering the EXPEDITED DATA state , the transmit and receive ARQ window pointers and the PDU ID # counter are automatically zeroized. This counter is distinct from the counter with the same name for D_PDU types 7 and 8.
- Starting another ARQ machine (i.e. establishing a link with a new node or re-establishing a link with a previously connected node) resets the EXPEDITED DATA state ARQ machine.

- The processing of D_PDUs containing expedited data proceeds according to a *C_PDU level* stop and wait protocol. Each time a D_EXPEDITED_UNIDATA_REQUEST Primitive is accepted, the expedited data D_PDU sequence counter is reset and the C_PDU ID counter is incremented.

- Upon exiting the EXPEDITED DATA state to another *data transfer state*, all unsent C_PDUs (and portions of C_PDUs) are discarded. Similarly at the receiving node, a transition from the EXPEDITED DATA state to another data transfer state results in the deletion of a partially assembled C_PDU. The decision to delete partially processed C_PDUs when a transition is made to another data transfer state reflects the primary usage of the expedited data transfer service, i.e. the transfer of short, high priority, upper layer peer-to-peer M_PDUs that require immediate acknowledgement. D_PDUs are not discarded and the expedited data ARQ machine is not reset following a transition to/from the PAUSE state.

- There is an implied limit on the size of the C_PDU which can be transmitted using type 4 D_PDUs. The limit is 256 times the size of the data portion of the D_PDU. This limit will not be important for systems which observe the MTU of 2048 bytes.

C.3.7 EXPEDITED ACK ONLY D_PDU

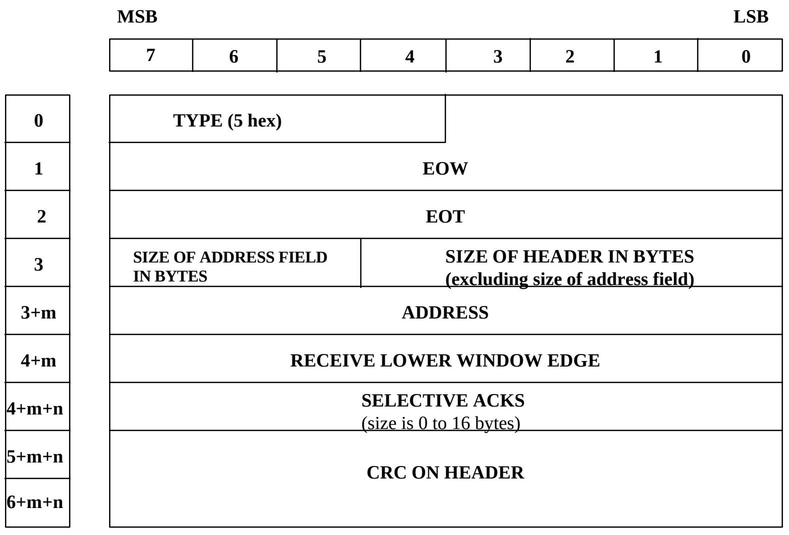


Figure C-10. Header for EXPEDITED ACK ONLY D_PDU

The EXPEDITED ACK-ONLY D_PDU type carries acknowledgements in a one-way transfer of expedited data. This D_PDU type has the same format as the ACK ONLY D_PDU.

This D_PDU should be repeated as described in Table C-4 when one or a small number of type 5 D_PDUs are to be transmitted.

C.3.8 MANAGEMENT D_PDU

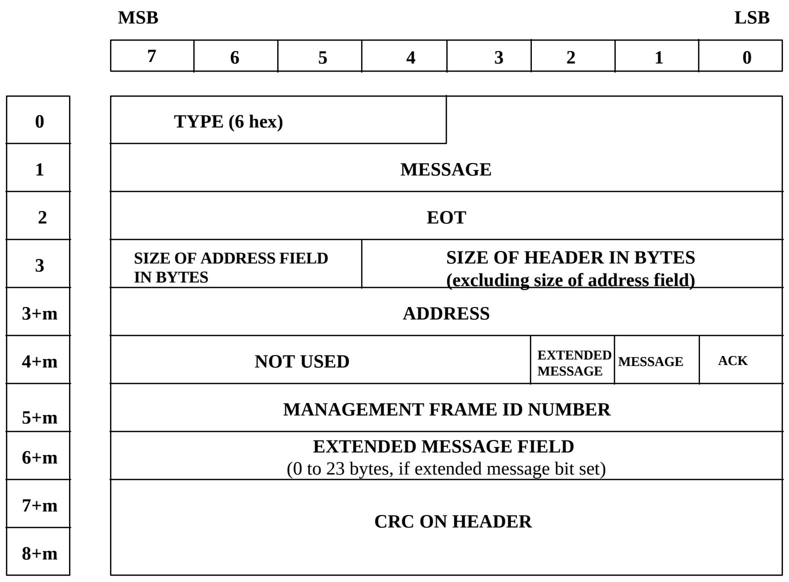

MSB
LSB
7
6
5
4
3
2
1
0
0
1
2
3
3+m
4+m
5+m
6+m
7+m
8+m
TYPE (6 hex)
MESSAGE
EOT
SIZE OF ADDRESS FIELD IN BYTES
SIZE OF HEADER IN BYTES (excluding size of address field)
ADDRESS
NOT USED
EXTENDED MESSAGE
MESSAGE
ACK
MANAGEMENT FRAME ID NUMBER
EXTENDED MESSAGE FIELD
(0 to 23 bytes, if extended message bit set)
CRC ON HEADER

Figure C-11. Header format for MANAGEMENT D_PDU

The MANAGEMENT D_PDU provides a reliable, quick response way to coordinate data link management activity between two nodes (for example, to coordinate data rate changes when using the STANAG 4285 or 4529 modems). These D_PDUs are only transmitted while the node is in the Management state and can bypass all other pending D_PDU types. The exchange of MANAGEMENT frames is regulated by a stop-and-wait protocol, that is there can be only one unacknowledged type 6 D_PDU at any one time. A node shall periodically retransmit type 6 D_PDUs until they are acknowledged or the link fails.

When a new (as indicated by the MANAGEMENT FRAME ID NUMBER) type 6 D_PDU is received with the MESSAGE bit set, the receiving node shall acknowledge the D_PDU by sending a type 6 D_PDU with the ACK bit set. The MESSAGE bit shall not be set unless the new D_PDU contains a management message beyond the acknowledgement.

On initialisation (such as a new connection), a node shall set its current TX MANAGEMENT FRAME ID NUMBER to zero and shall set its current RX MANAGEMENT FRAME ID NUMBER to an out of range value (greater than 255). The current TX MANAGEMENT FRAME ID NUMBER shall be placed in the appropriate field of each unique type 6 D_PDU transmitted, and the current value shall be incremented by one. Repeated D_PDUs (i.e., according to Table C-4) shall have the same MANAGEMENT FRAME ID NUMBER. The MANAGEMENT FRAME ID NUMBER of received type 6 D_PDUs shall be compared to the current RX MANAGEMENT FRAME ID NUMBER. If the MANAGEMENT FRAME ID NUMBER in the received frame differs from the current RX MANAGEMENT FRAME ID value, the D_PDU shall be treated as a new D_PDU, and shall update the current RX MANAGEMENT FRAME ID value to the received MANAGEMENT FRAME ID NUMBER. If the value in the received D_PDU is equal to the current RX MANAGEMENT FRAME ID value, the node shall assume that the frame is a repetition of a type 6 D_PDU that has already been received.

There is a one-to-one correspondence between MANAGEMENT messages and MANGEMENT D_PDUs; that is, each message is placed into a separate D_PDU (which may be repeated a number of times as shown in Table C-4). The 12-bit MSG section of the D_PDU shall carry the standard (non-extended) MANAGEMENT message. The types of MANAGEMENT messages are defined in the table below.

Table C-6. MANAGEMENT Message Types

Message Type	Function	Contents
0	RESERVED	all bits reset to 0
1	Data Rate Change Request (DRC_Req)	New HF modem transmit data rate and interleaving setting for DRC master
2	Data Rate Change Response (DRC_Resp)	Positive or negative response (including reason if negative)
3	Unrecognized user defined message type	message type field which is not recognized
4	capabilities	bit map describing capabilities of node
5	Frequency Change/ALM Request	
6	Frequency Change/ALM Response	
~~5~~7-8	reserved	

The format of the MANAGEMENT message types are shown in Figure C-12.

MSB											LSB
11	10	9	8	7	6	5	4	3	2	1	0
TYPE				contents							

Figure C-12 (a). ~~Message Type 0~~ Format of Type 0 and Type 4 Messages

MSB											LSB
11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	Data Rate				Interleaving		Other parameters	
TYPE											

Figure C-12 (b). Message Type 1 Format

MSB											LSB
11	10	9	8	7	6	5	4	3	2	1	0
TYPE				Response			Reason				

Figure C-12 (c). Format of Type 2 and 3 Message

The TYPE field of the message shall be filled with the hexadecimal value of the appropriate message type (units only), with the LSB of the TYPE value placed in the LSB of the TYPE field. The other fields are filled according to the following tables.

Table C-7. Contents for Type 1 Message (Data Rate and Interleaving)

Data Rate Parameter
Message Type 1

MSB - LSB	Interpretation
0 0 0 0	75 bps
0 0 0 1	150 bps
0 0 1 0	300 bps
0 0 1 1	600 bps
0 1 0 0	1200 bps
0 1 0 1	2400 bps
0 1 1 0	3200 bps
0 1 1 1	3600 bps
1 0 0 0	4800 bps
1 0 0 1	6400 bps
1 0 1 0	7200 bps
1 0 1 1	9600 bps
others	reserved

Interleaver Parameter
Message Type 1

MSB - LSB	Interpretation
0 0	no interleaving
0 1	short interleaving
1 0	long interleaving
1 1	reserved

Table C-8. Contents for Type 1 Message (Other Parameters)

MSB - LSB	Interpretation
0 0	DRC Request: master has independent data rate (change applies to Tx data rate only)
0 1	DRC Request: Tx and Rx data rate at master must be equal (change will apply to both Tx and Rx data rates)

Table C-9. Contents for Type 2 Message (Response)

MSB - LSB	Interpretation
0 0 0	accept
0 0 1	refuse
0 1 0	cancel
0 1 1	confirm

Table C-10. Contents for Type 2 Message (Reason)

MSB - LSB	Interpretation
0 0 0 0 0	no reason (used to indicate unconditional acceptance of DRC_Request)
0 0 0 0 1	Tx and Rx parameters must be the same (conditionally accept)
0 0 0 1 0	Not possible to change modem data rate
0 0 0 1 1	Not possible to change modem interleaving
0 0 1 0 0	Not possible to change modem data rate or interleaving
0 0 1 0 1	Not consistent with local conditions

The parameters for the Type 1 message shall be

1. the new data rate at which the node producing the message (DRC master; see section on Data Rate Change Procedures) will transmit data following the DRC; and
2. the capabilities of the modem in use by the node producing the message (for that link, if multiple links/modems are in use) with respect to transmit and receive data rates.

The Type 3 message has the type of the unrecognised user defined message type mapped into the four most significant bits of the "reason" field. The MSB of the type of the unrecognized message type shall be mapped into the MSB of the "reason" field. The unused LSB of the "reason" field, and all bits in the "response" field, shall be reset to 0.

Message type 4 allows a node to inform another node of its capabilities. For this message type, the contents of the message field are defined in the following table.

Table C-11. Contents of Message field for MANAGEMENT Message Type 4

bit	meaning
7 (MSB)	Adaptive modem parameters (DRC) capable note 1 (0 = no, 1 = yes)
6	STANAG 4529 available note 2 (0 = no, 1 = yes)
5	MIL-STD-188-110A available note 2 (0 = no, 1 = yes)
4	extended data rate capable note 3 (0 = no, 1 = yes)
3	full duplex supported note 4 (0 = no, 1 = yes)
2	split frequency supported note 4 (0 = no, 1 = yes)
1	non-ARCS ALE capable
0 (LSB)	ARCS capable (0 = no, 1 = yes)

Notes

1. If a node is DRC capable, it must implement at minimum 75 bps through 2400 bps (1200 bps for STANAG 4529) with short and long interleaving in accordance with the relevant document.
2. All nodes shall have, at minimum, the STANAG 4285 waveform available.
3. Annex G describes waveforms for data rates above 2400 bps.
4. A full duplex node must have split frequency operation available.

The EXTENDED MESSAGE field may also be used to transmit other vendor-specific messages. When the EXTENDED MESSAGE field is in use, the EXTENDED MESSAGE bit shall be set.

The type 6 D_PDU should ~~shall~~ be repeated as shown in Table C-4, for the reasons explained there.

C.3.9 NON-ARQ DATA D_PDU

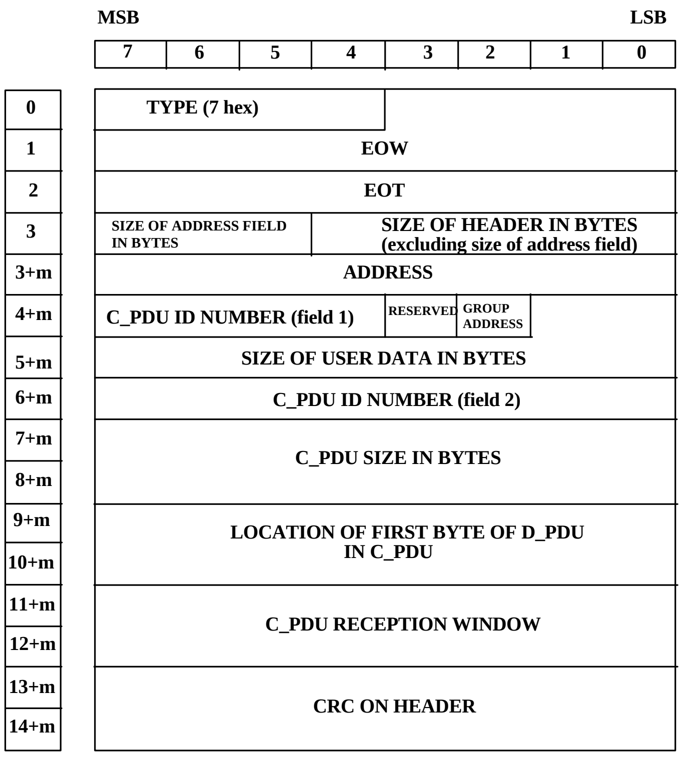

MSB
LSB
7
6
5
4
3
2
1
0
0
1
2
3
3+m
4+m
5+m
6+m
7+m
8+m
9+m
10+m
11+m
12+m
13+m
14+m
TYPE (7 hex)
EOW
EOT
SIZE OF ADDRESS FIELD IN BYTES
SIZE OF HEADER IN BYTES (excluding size of address field)
ADDRESS
C_PDU ID NUMBER (field 1)
RESERVED
GROUP ADDRESS
SIZE OF USER DATA IN BYTES
C_PDU ID NUMBER (field 2)
C_PDU SIZE IN BYTES
LOCATION OF FIRST BYTE OF D_PDU IN C_PDU
C_PDU RECEPTION WINDOW
CRC ON HEADER

Figure C-13. Header format for NON-ARQ DATA D_PDU

The first ~~3~~4 + m bytes of the header, and the CRC field, of the NON-ARQ DATA D_PDU are similar to that of the other DATA D_PDUs. The type 3 EOW message shall be included in all non-ARQ D_PDUs. The unique parts of the non-ARQ D_PDUs are described below.

The C_PDU ID NUMBER is a 12 bit integer which identifies the C_PDU to which the D_PDU belongs. These numbers can be reused as soon as the C_PDU RECEPTION WINDOW expires but should be used in sequence. In the header for this D_PDU, the C_PDU ID NUMBER field has been placed in non-adjacent bytes to provide maximum commonality between different D_PDU headers. The C_PDU ID NUMBER shall be mapped into the field such that the MSB of the C_PDU ID is in bit 7 of byte 4+m (C_PDU ID NUMBER field 1), and the LSB of the C_PDU ID is in bit 0 of byte 6+m.

The SIZE OF USER DATA field is as described for other DATA D_PDU types.

The C_PDU SIZE IN BYTES is a 16 bit integer which indicates the size of the C_PDU of which the D_PDU is a part.

The LOCATION OF FIRST BYTE OF D_PDU IN C_PDU is a 16 bit integer which indicates the location of the first byte of the D_PDU with respect to the start of the C_PDU. For the purposes of this field, the bytes of the C_PDU shall be numbered consecutively starting with 0.

The C_PDU RECEPTION WINDOW is a 16 bit integer which indicates the maximum remaining time in units of 0.5 seconds relative to the start of the D_PDU during which portions of the associated C_PDU may be received. As in the case of the EOT field, the C_PDU RECEPTION WINDOW is updated just prior to transmitting each D_PDU. The receiving node can use this information to determine when to release a partially received C_PDU.

The GROUP ADDRESS bit indicates that the destination address should be interpreted as a group address rather than an individual address. The use of a bit to designate a "group address" allows the same number (268 million!) and structure of group addresses to be designated (rather than requiring some portion of the total address space for group addresses). Group addresses are intended to allow PDUs to be addressed to specific groups when using NON-ARQ DATA D_PDUs. If a specific PDU is intended for only one node, the normal address may also be used with the NON-ARQ DATA D_PDU.

The D_PDU sequence number for this type of D_PDU is assigned from a unique sequence, separate from those used for other D_PDU types.

~~The header format for EXPEDITED NON-ARQ DATA D_PDUs is identical to the NON_ARQ DATA D_PDU except that the TYPE field has a value of 8 hex.~~

The C_PDU ID NUMBER space for the type 7 and type 8 D_PDUs should be the same.

C.3.10 EXPEDITED NON-ARQ DATA D_PDU

The header format for EXPEDITED NON-ARQ DATA D_PDUs is identical to the NON_ARQ DATA D_PDU except that the TYPE field has a value of 8.

C.3.11 WARNING D-PDU

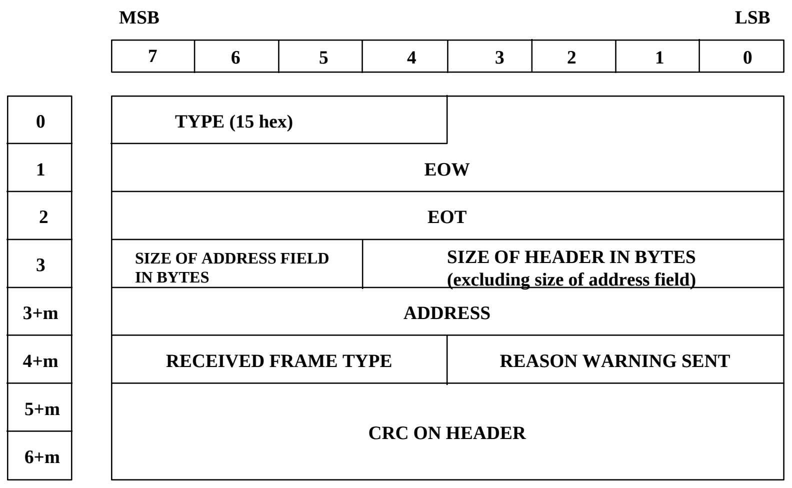


Figure C-14. Header format for WARNING D_PDU

The transmission of WARNING type D_PDUs is initiated independently by the Data Transfer Sublayer in response to certain D_PDUs. WARNING type D_PDUs are not assigned a TX FRAME SEQUENCE NUMBER and are not acknowledged. WARNING type D_PDUs may be inserted into an ongoing transmission interval provided that the transmission interval period is not exceeded.

A WARNING D_PDU shall be sent in the following conditions:

1. A node receives a D_PDU header addressed to itself with a valid CRC and an unrecognized D_PDU type (value 0000)
2. A node is not in the IDLE/BROADCAST state and it receives a D_PDU header addressed to itself, from a node with which it is not currently connected. (value 0001)
3. A node is in IDLE/BROADCAST state and it receives a D_PDU header addressed to itself which is other than type 7 or type 8 D_PDU (value 0010)
4. A node receives any D_PDU which is recognized but is not of the allowed type for the state which the receiving node is in (value 0011; this is the general case of the preceding)

A WARNING D_PDU shall not be sent in response to a WARNING D_PDU.

The reason codes are defined in Table C-12.

Table C-12. Reason Codes for Warning D_PDUs

Binary Value (MSB LSB)	Reason
0000	Undefined D_PDU type
0001	Busy (node is engaged in an ARQ exchange with another node)
0010	Connection assumed when none exists
0011	Improper D_PDU for current state

A node which receives a warning D_PDU must stop the action which caused the warning. A node which receives a ~~type 1 or type 2~~ WARNING D_PDU with binary reason code 0001 shall transition the related connection to the IDLE/BROADCAST state.

C.4 Peer-to-peer communication protocols

This section discusses the interactions between nodes in terms of states and state diagrams. This STANAG does not mandate a state machine implementation. The requirement for interoperability is that the system behave consistently with the rules presented in this document.

C.4.1 Data Transfer Sublayer States and Transitions

The expected and allowed interactions of one node with another depend on the states of the node's various sublayers. This section applies specifically to the states of the Data Transfer sublayer. Receiving certain PDUs will cause a node, depending on its state, to transmit certain PDUs and/or change to another state. The allowed states are as shown in the figure below. The tables following give, for each state, the actions and transitions which are required on reception of various PDUs, and the PDU types which may be transmitted.

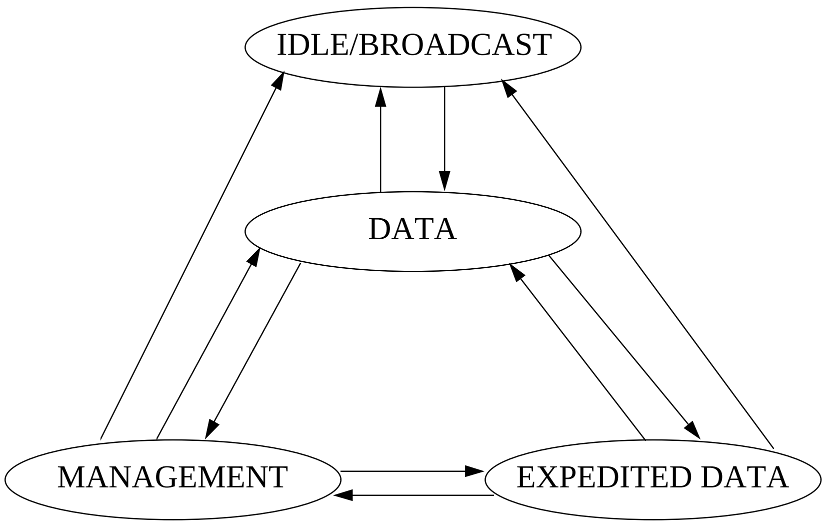


Figure C-15. State and Transition Diagram (Data Transfer Sublayer)

C.~~1~~4.~~4~~1.1 State Transition Rules

In all of the tables below, "PDU received" refers to a PDU received which is addressed to the node in question. State transitions shall not occur based on PDUs addressed to other nodes.

Table C-13. Rules for IDLE/BROADCAST State

PDU received	Transition to	action/notes
type 0-6 D_PDU	no transition	transmit type 15(2) D_PDU (note 1)
type 7 or 8 D_PDU	no transition (note 2)	process and deliver data (C_PDU)
type 1 C_PDU	DATA	transmit C_PDU type 2 or 3 (in ~~type 7 or~~ 8 D_DPU)
type 15 D_PDU	no transition	stop the action indicated

Note 1. The number in brackets for the type 15(2) D_PDU indicates the decimal translation of the binary value in the reason field of the type 15 D_PDU (Table C-12). In this case, 15(2) translates to “connection assumed when none exists”.
Note 2. No transition is made based on the D_PDU. Transition to the DATA state is made based on receipt of type 1 C_PDU. Table C-13 applies to the process described in Figure B-3(b) (Called Peer). The rules for “calling” (Figure B-3(a)) are slightly different.

Table C-14. Rules for EXPEDITED DATA State

PDU received	Transition to	action/notes
type 0-2 D_PDU	DATA	process and respond
type 3 D_PDU	no transition	process and respond
type 4,5 D_PDU	no transition	process and respond
type 6 D_PDU	MANAGEMENT	process and respond
type 7,8 D_PDU	no transition (note 1)	process and deliver
type 4 C_PDU	IDLE/BROADCAST	send type 5 C_PDU
type 5 C_PDU	IDLE/BROADCAST	none (note 2)
type 15 D_PDU	~~no transition~~see note 3	stop the action indicated

Note 1. No transition is made based on the D_PDU. Transition to the IDLE/BROADCAST state is made based on receipt of type 4 C_PDU (see Annex B). ~~This table applies to the process described in Figure B-4(b) (Responding Peer). The rules for initiating a physical link break (Figure B-4(a)) are slightly different.~~
Note 2. All states other than IDLE/BROADCAST are associated with a specific node address. If a node is not in IDLE/BROADCAST state and any D_PDU other than type 7 or 8 is received from a node other than the node to which it is currently connected, the node shall send a Type 15(1) D_PDU.
Note 3. A node which receives a WARNING D_PDU with binary reason code 0001 shall transition the related connection to the IDLE/BROADCAST state.

Table C-15. Rules for DATA State

PDU received	Transition to	action/notes
type 0-3 D_PDU	no transition	process and respond
type 4 D_PDU	EXPEDITED DATA	process and respond
type 5 D_PDU	no transition	transmit type 15(3) D_PDU
type 6 D_PDU	MANAGEMENT	process and respond (acknowledge)
type 7 or 8 D_PDU	no transition (note 1)	process and deliver
type 1 C_PDU	see note 6	
type 4 C_PDU	IDLE/BROADCAST	send type 5 C_PDU
type 5 C_PDU	IDLE/BROADCAST	none (note 2)
type 15 D_PDU	see note 5~~no transition~~	stop the action indicated

Note 1. No transition is made based on the D_PDU. Transition to the IDLE/BROADCAST state is made based on receipt of type 5 C_PDU.
Note 2. Regardless of whether the node receiving the type 5 C_PDU has initiated a disconnect, when a type 5 C_PDU is received, the assumption shall be that the distant node is in the IDLE/BROADCAST state.
Note 3. All states other than IDLE/BROADCAST are associated with a specific node address. If a node is not in IDLE/BROADCAST state and any D_PDU other than type 7 or 8 is received from a node other than the node to which it is currently connected, the node shall send a Type 15(1) D_PDU.
Note 4. The pointers, queues, and other data structures associated with the DATA state should be retained on transition to EXPEDITED DATA or MANAGEMENT states.
Note 5. A node which receives a WARNING D_PDU with binary reason code 0001 shall transition the related connection to the IDLE/BROADCAST state.
Note 6. If the type 1 C_PDU corresponds to an existing link, the rule is no transition/send type 2 C_PDU (see Annex B).

Table C-16. Rules for MANAGEMENT State

PDU received	Transition to	action/notes
type 0-3 D_PDU	DATA (note 1)	process and respond
type 4 D_PDU	EXPEDITED DATA (note 2)	~~discard~~process and respond
type 5 D_PDU	no transition	transmit type 15(3) D_PDU
type 6 D_PDU	no transition	process and respond (acknowledge)
type 7 or 8 D_PDU	no transition (note 3)	process and deliver
type 4 C_PDU	IDLE/BROADCAST	send type 5 C_PDU
type 5 C_PDU	IDLE/BROADCAST	none
type 15 D_PDU	note 4~~no transition~~	stop the action indicated

Note 1. If (the node believes that) the current management process is not complete, it may remain in the management state and continue the management process, delaying response to type 0-3 D-PDUs; however, the contents shall be processed and acted upon appropriately when the current and any pending management processes are completed.
Note 2. If (the node believes that) the current management process is not complete, it may remain in the management state and continue the management process.
Note 3. No transition is made based on the D_PDU. Transition to the IDLE/BROADCAST state is made based on receipt of type 5 C_PDU.
Note 4. A node which receives a WARNING D_PDU with binary reason code 0001 shall transition the related connection to the IDLE/BROADCAST state.

C.4.1.2 State Rules for Sending D_PDUs

A node shall transmit only D_PDUs which are allowed for its current state :

- IDLE/BROADCAST: type 7, 8, or 15 D_PDUs
- DATA: type 0, 1, 2, 3, 7, 8, or 15 D_PDUs
- EXPEDITED DATA: type 4, 5, 8, or 15 D_PDUs
- MANAGEMENT: type 6, 8, or 15 D_PDUs

C.4.2 D_PDU numbering and flow control for DATA STATE

Primitives containing PDUs from upper layers arriving at the Data Transfer sublayer are segmented into D_PDUs for HF transmission (these will form the data portion of DATA-ONLY and DATA-ACK D_PDUs). Because the size of the data portion of the D_PDU is indicated in the D_PDU itself, it is not necessary to define the size of the D_PDU. A C_PDU will thus be divided into one or more D_PDUs for HF transmission. D_PDUs making up a C_PDU shall be assigned D_PDU numbers in sequential order. D_PDU numbers shall be assigned in the range from 0 to 255.

Because frame numbers are assigned using modulo arithmetic, it is convenient to represent the ARQ sliding window as a segment of a circular buffer. Each node shall maintain a transmit and a receive flow control window buffer for each link supported. The following conventions will be followed in this STANAG when discussing link layer flow control.

- The frame numbers shall be assigned sequentially[9], modulo 256.
- The window size at any moment is variable (“current window size”) and shall not exceed 127 (“maximum window size”).
- The transmit lower window edge (TX LWE) shall indicate the “oldest” unacknowledged D_PDU (lowest D_PDU number, allowing for the modulo 256 operation).
- The transmit upper window edge (TX UWE) shall be the number of the last new D_PDU that was transmitted. When the distance (in a modulo 256 sense) between the TX UWE and TX LWE is equal to 127, no additional new D_PDUs shall be transmitted until the TX UWE is advanced and the distance is ~~less than~~ 127 or less. The number of unacknowledged frames (if no frames between the window edges have been acknowledged) is equal to UWE - LWE + 1.
- The receive lower window edge (RX LWE) shall indicate the smallest (mod 256) or oldest D_PDU number that has not been received.
- The receive upper window edge (RX UWE) shall be the number of the last new D_PDU that was received. More explicitly, it is the TX FSN of the received D_PDU which is the greatest distance (mod 256) from the RX LWE. The RX UWE increases monotonically as D_PDUs with higher (mod 256) TX FSN are received; it does not move back when retransmitted D_PDUs are received.
- The initial condition of the window edge pointers (e.g., on the initialization of a new link) shall be
    - TX LWE: 0
    - TX UWE: 255
    - RX LWE: 0
    - RX UWE: 255

[9]Different sequences of numbers are used for different types of frames. Frame types 0 and 2 share a common sequence of frame numbers. EXPEDITED DATA (type 4) and MANAGEMENT (type 6) D_PDUs shall each use their own, distinct frame number sequence. The two non-ARQ D_PDU types shall share a common sequence of C_PDU ID numbers.

~~If at any time t~~The separation between the node's TX LWE and TX UWE ~~is equal to or greater than~~ shall not exceed 127.~~,~~ ~~t~~If the separation is equal to 127, the node shall not transmit any more DATA-ONLY or DATA-ACK D_PDUs until outstanding D_PDUs are positively acknowledged and the TX LWE can be advanced. Note that, although the flow control limitations would allow 128 D_PDUs to be transmitted in one transmission, other protocol parameters will also affect this in an indirect way; the structure of the EOT parameter allows a maximum transmission interval of about 2 minutes. If a D_PDU size of 200 bytes is used at 75 bps, only 5 D_PDUs can be transmitted. However, even if no acks are received, it would be possible to transmit many more D_PDUs before transmission would halt due to flow control restrictions.

The LWE and UWE are defined slightly differently for the receive flow control buffer. In the receive buffer, the LWE indicates the smallest D_PDU number that has not been recieved (therefore all D_PDUs prior to the LWE have been received). The receive buffer UWE assumes the value of the highest received D_PDU number~~plus one (that is, the next expected D_PDU number)~~.

C.4.3 **Data Rate Control**

Initial (Default) Data Rate

All connections shall be initiated at 300 bps, using short interleaving. The waveform shall be selected by the user.[10]

Adaptive Data Rate Change Procedure

The receiving node has available to it complete information on the signal it is receiving and is therefore best able to determine the optimum data rate at which data should be sent to it. The EOW type 1 message is provided as a way for the receiving node to advise the sending node of the optimum data rate. The action and the final decision remain with the sending node. Mechanisms other than the type 1 EOW message may be used to arrive at a decision to change data rate; however, these other mechanisms are not defined here and will not be interoperable between vendors. At a minimum, systems shall implement and support data rate changes in accordance with the procedures defined here.

On receiving an EOW type 1 message, a node shall comply with the parameters specified in the message unless some specific reason prevents it doing so (such as lack of remote modem control, or local conditions do not support a change if modem tx and rx data rate must be the same). Generally this means that the node will initiate a data rate change (DRC) procedure.

Following a decision to change data rate, a node shall use type 6 D_PDUs (MANAGEMENT D_PDUs) containing type 1 and type 2 MANAGEMENT messages to implement and coordinate the change. The change procedure will be executed in Management mode. If the waveform used supports automatic recognition of coding and interleaving parameters[11], the data rate change procedures defined here are not necessary. Following receipt of an EOW advisory message, a system using such a waveform can simply change the transmit data rate. See Annex H for more details on this subject.

[10] If the waveform used supports automatic recognition of coding and interleaving parameters, defined initial settings are not necessary.
[11] Examples of such waveforms include the waveforms defined in Annex G to this document, and the MIL-STD-188-110A single tone waveform.

Data rate changes are effective only for a single connection. If a node has a number of connections active with different nodes, the data rate change decisions and procedures shall be executed independently for each connection.

The node initiating the data rate change is referred to as the data rate change master (DRC master) for this DRC procedure. Figure C-16 shows an example of a successful DRC procedure. The numbers in brackets in the figure are included for reference from the text.

The message numbered [1] in the figure indicates an EOW advisory message. The DRC master shall send a type 1 MANAGEMENT D_PDU [2] with the parameters equal to the intended new transmit data rate for the DRC master. The "other parameter" field of the type 1 MANAGEMENT message shall be set to indicate the data rate capabilities of the modem in use at the DRC master, for this link. In the figure, the modem at the DRC master has independent transmit and receive data rate.

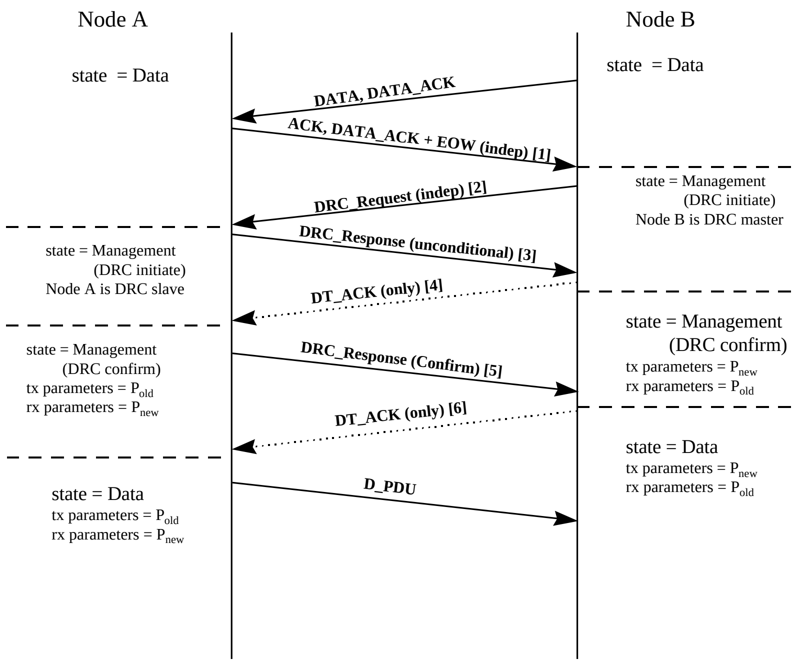

Node A
Node B
state = Data
state = Data
DATA, DATA_ACK
ACK, DATA_ACK + EOW (indep) [1]
state = Management (DRC initiate) Node B is DRC master
DRC_Request (indep) [2]
DRC_Response (unconditional) [3]
state = Management (DRC initiate) Node A is DRC slave
DT_ACK (only) [4]
state = Management (DRC confirm) tx parameters = P_{new} rx parameters = P_{old}
state = Management (DRC confirm) tx parameters = P_{old} rx parameters = P_{new}
DRC_Response (Confirm) [5]
DT_ACK (only) [6]
state = Data tx parameters = P_{new} rx parameters = P_{old}
state = Data tx parameters = P_{old} rx parameters = P_{new}
D_PDU

Figure C-16. Data Rate Change Procedure (Example 1)

The only state transition allowed due to timeout during this procedure is to the IDLE/BROADCAST state. The management message stop-and-wait protocol, with the repetitions shown, is extremely robust. If it is not possible to complete the procedure in the management state, it is unlikely that returning to another connected state will be productive. Timeouts should be set to allow a reasonable number of retransmissions, perhaps 3, before failure is declared.

If this procedure was initiated in response to an EOW Type 1 message, the modem data rate and interleaving parameters shall be identical to the parameters in the EOW message, unless they specify a speed for which the DRC master is not equipped. The DRC Request D_PDU shall be repeated depending on the data rate at which it is transmitted (see table C-17), with the same FRAME ID NUMBER in each of the copies. Valid EOT information shall be supplied in each copy. The same EOT shall not be transmitted in each copy (unless necessary due to EOT resolution, i.e., D_PDU duration is less than half the EOT resolution).

When a node recognises a MANAGEMENT D_PDU addressed to it, containing a DRC_Request message, the node (referred to as the DRC slave) shall transition to the management state. The DRC slave shall respond to the DRC_Request D_PDU with a DRC_Response message. The DRC Response message shall indicate either “accept” or “refuse”, in accordance with the tables describing the MANAGEMENT messages. If the DRC slave accepts the DRC_Request, the “reason” field shall indicate either “unconditional acceptance” or “Tx and Rx parameters must be the same”. If the DRC slave refuses the request, the reason field shall indicate the reason for the refusal. Only the four reasons defined in the table are valid reasons for refusing a DRC_Request. The reasons which apply to the modem capabilities imply that all attempts to change that modem parameter should stop (for the duration of the current link with the node that gives this reason for refusing a DRC). The “Not consistent with local conditions” parameter shall only be used to refuse a DRC_Request which indicates a less robust mode (ie, higher data rate or shorter interleaver). This response should only be used in the event that conditions have changed at the DRC slave between the time that the last EOW advisory message was sent and the DRC_Request was received. Subsequent DRC procedures may be initiated after additional information on channel conditions is obtained and exchanged.

This response is shown as DRC_Response [3] in the figure. The figure shows an example in which the modem at the DRC slave also has independent transmit and receive data rate. Multiple copies of this D_PDU should~~shall~~ be transmitted depending on the modem parameters~~data rate~~. ~~, as shown in~~ Table C-17 gives a recommended minimum number of times to transmit the message; more may be advisable under certain circumstances.

Table C-17. Minimum ~~N~~number of DRC messages to be transmitted at various data rates using STANAG 4285 modem

Data rate	~~repetitions~~ ~~(~~short interleave~~)~~	~~repetitions~~ ~~(~~long interleave~~)~~
75	1	9
150	1	18
300	1	37
600	3	75
1200	7	150
2400	15	300

The number of ~~re~~transmissions is selected to (nearly) fill the interleave buffer. For waveforms and interleaver settings not shown, the number of transmissions should ~~repetitions shall~~ be adjusted ~~selected~~ as required to minimise the use of “stuff bits” to fill the modem interleave buffer.

After receiving the DRC_Response message the DRC master shall review its contents and determine the appropriate response [4]. The various DRC_Response messages, and the allowed responses from the DRC master, are shown in table C-18.

Table C-18. Possible DRC_Responses and Allowed DRC Master Actions

DRC_Response	DRC_Response reason	allowed from DRC master

accept	unconditional	DT_ACK only
accept	transmit and receive parameters must be the same	DT_ACK only, or DRC_Response (cancel, or DRC Request)[note 1]
refuse	not possible to change modem data rate	DRC_Response (cancel)[note 2] or DRC_Request [note 3] (with DT_ACK)
refuse	not possible to change modem interleave	DRC_Response (cancel)[note 2] or DRC_Request [note 4](with DT_ACK)
refuse	not possible to change modem data rate or interleave	DRC_Response (cancel)[note 2](with DT_ACK)
refuse	not consistent with local conditions (see note 5)	DRC_Response (cancel)[note 2] or DRC_Request [note 6](with DT_ACK)

Notes to Table:

1. If the procedure is initiated in response to EOW type 1 message, the DRC master should already know that the DRC slave's transmit and receive parameters must be the same. Therefore, the DRC master should generally reply with a DT_ACK, accepting that the new parameters will apply to both transmit and receive.
2. DRC Slave shall acknowledge the cancel message with DT_ACK only; then the DRC procedure is discontinued.
3. DRC_Request may be sent by master to request a different interleave setting at the same data rate.
4. DRC_Request may be sent by master to request a different data rate setting at the same interleave.
5. This reply shall only be sent in response to a request for a less robust set of parameters, i.e., higher data rate and/or shorter interleave than currently in use.
6. If the nodes make use of the EOW type 1 message to initiate DRC procedure, the master shall send the DRC_Response (cancel), and await an updated EOW recommendation before initiating another DRC procedure. If EOW type 1 messages are not used, DRC_Request may be sent by master to request different modem parameters which may be consistent with the local conditions.

In the table above, the DT_ACK refers to a data transfer sublayer acknowledgement of the preceding MANAGEMENT message (shown at [4] in Figure C-16). The DT_ACK reply indicates that the node has nothing further to communicate. If the DT_ACK (with no further management message) is sent in reply to a DRC_Response "accept" (as shown in Figure C-16), the nodes change their respective modem parameters and proceed to the "confirmation" phase. The DRC slave shall NOT change its modem parameters until it has received the DT_ACK (with no further management message) from the DRC master. If the DT_ACK (with no further management message) is sent by the DRC slave in reply to a DRC_Response "cancel", both nodes abandon the procedure and return to the prior state without changing modem parameters. If node A (formerly the DRC slave) has no queued data or acknowledgements to send to node B, it shall send a data D_PDU, expedited data D_PDU, or non-ARQ D_PDU, with zero data attached.

In the figure, the slave's DRC_Response with an "accept/unconditional" message generates the allowed DT_ACK from the DRC master.

After sending the DT_ACK [4], the master changes its modem parameters and waits to receive a DRC Confirm message (type 2 MANAGEMENT message with response set to "confirm" and reason set to "none") from node A ("confirmation phase"). After receiving the DT_ACK [4], the slave changes its modem parameters and transmits a DRC Confirm message [5] to the master. On receiving the DRC Confirm message, the master shall respond with a DT_ACK and then return to the previous state. After sending the DRC Confirm message [5] to the master and receiving the DT_ACK from the master, the slave shall return to the previous state and send any queued D_PDUs to node B. If node A (formerly the DRC slave) has no queued data to send to node B, it shall send a data D_PDU or expedited data D_PDU with zero data attached.

Additional DRC Examples

The following figures provide additional examples of the DRC procedure in different conditions.

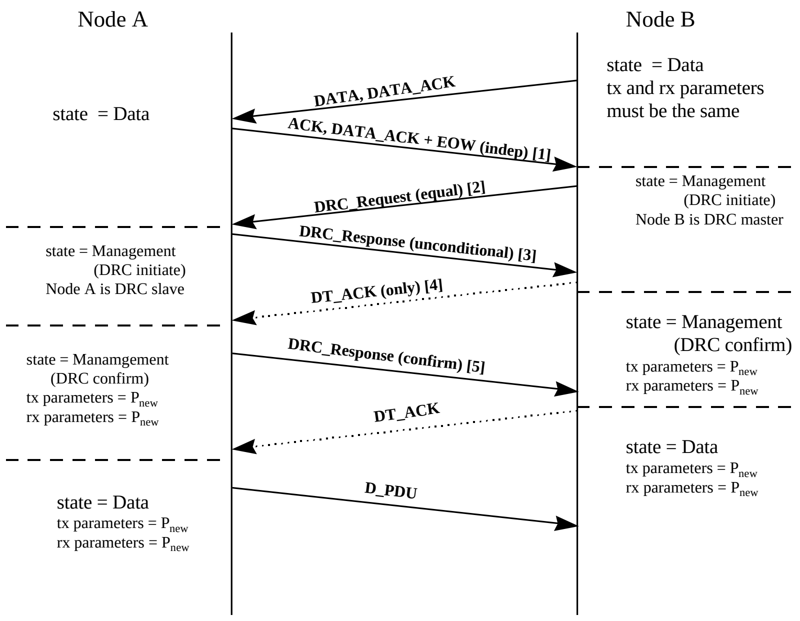


Figure C-17. Data Rate Change Procedure (Example 2)

Figure C-17 gives an example in which node B, due to HF modem limitations, must operate with the same HF modem transmit and receive parameters. The following numbered paragraphs correspond to the numbers in [brackets] in Figure C-17.

1. In this transmission, the advisory message sent by A carries the recommended tx speed for B (You should tx at x bps) and also tells B about the capabilities of A’s modem (that A’s modem can operate at independent tx and rx data rates.
2. The DRC_Request message from B carries the requested new modem parameters for B: “I (B) will transmit at x bps with y interleaving”; these parameters shall be as recommended by A. The message also tells A about the capabilities of B’s modem, in this case, if I change my tx parameters (and your rx parameters), I will have to change my rx parameters (and your tx parameters) as well.)
3. DRC_Response message accepts the requested change.
4. The DT_ACK confirms that B received the prior message and triggers B to change its parameters.
5. This DRC_Response (confirm) is a confirmation that the link is up at the new parameters.

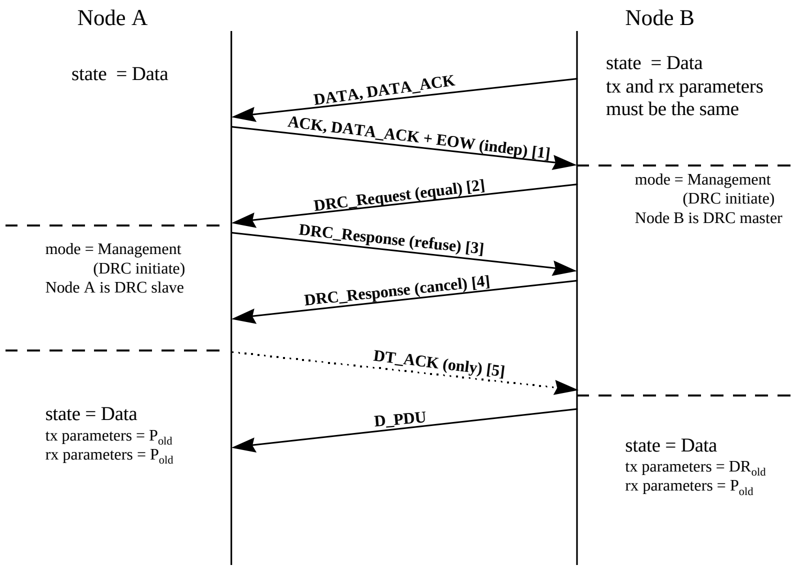


Figure C-18. Data Rate Change Procedure (Example 3)

Figure C-18 shows an example in which the procedure is refused by the slave, and hence cancelled by the DRC master. In this example, node B has a modem which must have the same transmit and receive parameters. The requested new parameters would then apply to the transmit and receive parameters at both nodes. The following numbered paragraphs correspond to the numbers in [brackets] in Figure C-18.

1. In this transmission, the advisory message sent by A carries the recommended tx speed for B (You should tx at x bps) and also tells B about the capabilities of A’s modem (that A’s modem can operate at independent tx and rx data rates.
2. The DRC_Request message from B carries the requested new modem parameters for B: “I (B) will transmit at x bps with y interleaving”; these parameters shall be as recommended by A. The message also tells A about the capabilities of B’s modem, in this case, if I change my tx parameters (and your rx parameters), I will have to change my rx parameters (and your tx parameters) as well. The net result is that both nodes have the same rx and tx parameters.
3. DRC_Response message refuses the requested change; a likely reason would be that the local conditions at node A do not support use of the proposed parameters.
4. The DRC master responds by canceling the procedure.
5. The DT_ACK confirms that B received the prior message and terminates the procedure.

C.4.4 Procedure for use of Type 3 RESET/WIN RESYNC D_PDU

The Type 3 RESET/WIN RESYNC D_PDU, described in C.3.5, supports a number of different resynchronization functions. A procedure for the FULL RESET function is described in this section.

A FULL RESET procedure is initiated by a node sending a type 3 D_PDU as follows:

- FULL RESET CMD flag set to 1
- RESET FRAME ID NUMBER as selected from RESET FRAME ID NUMBER sequence
- NEW RECEIVE LWE field reset to zero
- RESET TX WIN RQST and RESET RX WIN CMD flags reset to zero

The D_PDU described above will be referred to as a FULL RESET CMD D_PDU. A node receiving a FULL RESET CMD D_PDU shall

- set the transmit and receive window pointers to initial values (as defined in C.4.2)
- discard from its transmit queue any partially completed C_PDUs
- flush its receive buffers
- respond by sending a type 3 D_PDU as follows:

    - RESET ACK flag set to 1
    - NEW RECEIVE LWE and RESET FRAME ID NUMBER fields reset to zero
    - RESET TX WIN RQST, FULL RESET CMD and RESET RX WIN CMD flags reset to zero

The D_PDU described above will be referred to as a FULL RESET ACK D_PDU. The FULL RESET ACK D_PDU shall be sent only in response to the FULL RESET CMD D_PDU. On receiving the FULL RESET ACK D_PDU, the node initiating the FULL RESET shall:

- set the transmit and receive window pointers to initial values (as defined in C.4.2)
- discard from its transmit queue any partially completed C_PDUs
- flush its receive buffers

This concludes the FULL RESET procedure.

~~Information Only Annexes~~

Annex D: Interface between Data Transfer Sublayer and Communications Equipment (mandatory)

This annex defines the interface between the Data Transfer Sublayer and the communications equipment. This is currently cryptographic equipment but in time, the crypto is expected to move to the application layer. When this change occurs, this definition shall apply to the interface between the Data Transfer Sublayer and the modem.

This interface shall be configurable, with EIA-232D/423 for unbalanced connections and EIA-422 for balanced connections. A synchronous interface is mandatory. Clock source shall be either configurable or DCE (crypto or modem) for DTE data out. Clock source shall be DCE for DTE data in. Full hardware handshaking, in accordance with any standard recommendations, is mandatory. Compatability with MIL-STD-188C polarity, levels, and slew rates will be specified if required for interoperability with existing (cryptographic) equipment.

Annex E: HF Modem remote control interface (information only)

This Annex defines a common remote control interface with a minimum set of modem commands required for operation of the system. This Annex has not been implemented or tested.

The electrical interface for remote control of the modem shall be configurable, with EIA-232D/423 for unbalanced connections and EIA-422 for balanced connections. The remote control interface shall include transmit data and receive data.

Remote control commands shall consist of ASCII alpha-numeric strings with 8 data bits, even parity, and one stop bit, at 9600 bps.

At minimum, the following commands shall be supported . In all cases, the <address> field shall be replaced by a decimal number between 0 and 255 which shall indicate the address of a specific modem. The modem shall be configurable to select this address and shall respond only to properly addressed commands.

<address> INITIALIZE This command shall instruct the modem to load a predefined set of operating parameters. The mechanism for defining this set is not standardized.

<address> MODEM RATE <x> (replace x with 75, 150, 300, 600, 1200, 2400, etc; ie, <address> MODEM RATE <1200>); selects the coding applied to the user data before transmission.

<address> MODEM INTERLEAVE <x> (replace x with ZERO, LONG, SHORT, ie, <address> MODEM INTERLEAVE <SHORT>); selects the interleaving used in the modem

<address> MODEM SNR? Returns the signal to noise ratio of the received signal, in the following format:
<address> MODEM SNR=<value> where value is replaced with the SNR in dB.

<address> MODEM IDENTIFY?   Returns the manufacturer, model, and software/firmware revision information for a modem, in the following format: <address> MFR <x> ; MODEL <y> SW/FW <software/firmware revision information> where the quantities in angle brackets are replaced by the appropriate ASCII strings. The angle bracket characters shall not appear in any of these strings.

<address> MODEM WAVEFORM <x> Replace x with 4285 (for 4285 with annex E coding), 4529, 110A (MIL-STD-188-110A single tone), or HIGH (for high data rate waveforms described in annex G); ie, MODEM WAVEFORM <4529> to select the STANAG 4529 waveform.

Annex F. Subnetwork Client Definitions (information only)

This Annex defines interactions with the subnetwork for a set of clients to which the HF subnet defined in the mandatory part of this STANAG provides data transport service (subnet clients). These clients are defined in order to provide an initial set of interoperable applications which make use of the HF subnet. Note that one subnetwork client may act as a server to another subnetwork client. Each section of the Annex describes the actions of two different subnetwork clients which carry on a client-server relationship between themselves over the subnetwork. This annex is a minimum requirement; compliance with this annex does not prohibit additional interactions which are not defined. Table F-1 defines standard SAP IDs for the client applications defined in this annex.

Table F-1: SAP ID Assignments

Client Application (see Annex F)	SAP ID
Subnet management client (Annex F.8)	0
Acknowledged message (Annex F.1)	1
Un-acknowledged message (Annex F.2)	2
HMTP (Annex F.3)	3
HF-POP3 (Annex F.4)	4
Z-modem file transfer (Annex F.5)	5
Operator orderwire (Annex F.6)	6
FAI client (Annex F.7)	7
reserved for future assignment	8-13
available for undefined use	14-15

All subnetwork clients shall segment the U_PDUs submitted in S_UNIDATA_REQUEST primitives to comply with the MTU advertised by the subnetwork interface sublayer in the S_BIND_ACCEPTED primitive. The format of all U_PDUs shall be as shown in Figure F-2. Order of transmission of bits and bytes, and mapping of bits into fields, shall be as defined in the mandatory parts of this document.

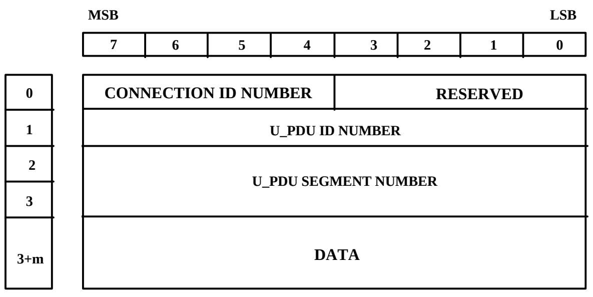


Figure F-2. Format for U_PDUs

Notes for figure F-2:

1. The connection ID number shall be a value from 0-15. Assignment and co-ordination of connection ID numbers is not specified here. Connection ID number 0 shall be reserved for non-multiplexed connections.
2. The reserved bits shall be set to 0.
3. The U_PDU ID numbers shall be assigned consecutively to U_PDUs.
4. The U_PDU segment number shall be assigned consecutively to segments within a single U_PDU. The first segment transmitted shall be assigned segment number 0. If a U_PDU is not segmented, the single segment that is transmitted shall be assigned number 0.

If the subnetwork interface sublayer receives a S_UNIDATA_REQUEST primitive with a U_PDU larger than the maximum PDU size, the S_UNIDATA_REQUEST shall be rejected.

Note that U_PDUs which are received from lower layers are not subject to this limitation.

The subnet clients which are defined in this annex are gateways which translate protocols designed for wide bandwidth links with little or no turnaround time, to protocols which are more suitable for HF radio links. While the advantages and disadvantages of a gateway approach can be debated, experience has demonstrated that the advantages outweigh the disadvantages, especially over HF radio links. If there is a need for an "IP router" subnet client, there is no reason that one cannot be developed. A full duplex system supported by a fast ALE/ARCS and high data rate waveforms (see Annex G) might give a useful IP routing capability.

F.1 Acknowledged message application

This annex defines a protocol which supports acknowledged message transfer over a radio link using the HF subnet described in the mandatory part of this STANAG. This would be suitable, for example, to replace the current TARE-based ship-shore systems.

F.1.1 SENDING

Sending shall be used to support transfer of messages between client and server. The primary difference between sending and mailing is that the server for “sending” does not necessarily support mail forwarding. Sending supports transfer of messages from a local user at one node to a local user at another node

The following command are defined to support the sending options. These are used in the mail transaction instead of the MAIL command and inform the receiver-SMTP of the special semantics of this transaction:

SEND <SP> FROM:<reverse-path> <CRLF>

The SEND command requires that the message data be delivered to the user's local mailbox, if it exists. If the user does not have a local mailbox with the SEND server, and the server does not support forwarding, a 450 reply may returned to a RCPT command. The transaction is successful if the message is delivered to the local mailbox.

The same reply codes that are used for the MAIL commands are used for these commands.

SEND MULTIPLE

This command is used to advise the server that a sequence of more than one SEND transaction will follow. The default, if this command is not sent, is a single SEND transaction.

SEND (SEND)

This command is used to initiate a transaction in which the message data is delivered to one or more local mailboxes, or forwarded to a mail server. The argument field contains a reverse-path. This command is successful if the message is delivered to the specified local mailboxes, or forwarded to a mail server.

The reverse-path consists of the sender's mailbox.

This command clears the reverse-path buffer, the forward-path buffer, and the mail data buffer; and inserts the reverse-path information from this command into the reverse-path buffer.

RESET (RSET)

This command specifies that the current mail transaction is to be aborted. Any stored sender, recipients, and mail data must be discarded, and all buffers and state tables cleared. The receiver must send an OK reply.

QUIT (QUIT)

This command specifies that the receiver must send an OK reply, and then close the transmission channel.

The receiver should not close the transmission channel until it receives and replies to a QUIT command (even if there was an error). The sender should not close the transmission channel until it send a QUIT command and receives the reply (even if there was an error response to a previous command). If the connection is closed prematurely the receiver should act as if a RSET command had been received (canceling any pending transaction, but not undoing any previously completed transaction), the sender should act as if the command or transaction in progress had received a temporary error (4xx).

Example of the SEND procedure

```
C: SEND MULTIPLE
C: SEND FROM:<user1>
C: RCPT TO:<user2>
C: RCPT TO:<user3>
C: RCPT TO:<user4>
C: DATA
C: Blah blah blah...
C: ...etc. etc. etc.
C: <CRLF>.<CRLF>
C: SEND FROM: <user1>
C: …
C: <CRLF>.<CRLF><CRLF>.<CRLF>


S: 250 Source <user1> OK
```

```
S: 250 Destination <user2.> OK
S: 550 Destination <user3> not known
S: 250 Destination <user4> OK
S: 250 DATA OK
S: 250 Source <user1> OK
S: etc.
```

The message has now been accepted for user2 and user4.

F.2 Un-acknowledged message application

This annex defines the use of primitives to support unacknowledged transfer of messages over a radio link using the HF subnet described in the mandatory part of this STANAG. This would be suitable, for example, to replace the current TARE-based shore-ship broadcast systems. Because this type of service is inherently unreliable, mail services are not supported. It is assumed that any acknowlegement (positive or negative) is the responsibility of the application level and communication for it is via some other channel.

In this case, there is no client-server relationship between subnetwork clients. The sending subnetwork client simply sends messages which the receiving subnetwork client or clients will receive. Messages for broadcast, or non-ARQ, transmission are submitted using the normal S_UNIDATA_REQUEST primitives with the service type set to non-ARQ. The address in the primitive will be either an individual address or a group address. More discussion of broadcast applications may be found in Annex H.8.

F.3 SMTP FOR USE OVER STANAG 5066 TRANSPORT (HMTP)

This annex defines a set of primitives which support acknowledged transfer of SMTP email messages over a radio link using the HF subnet described in the mandatory part of this STANAG. This would be suitable, for example, to provide the HF communications service for an application-level gateway of email to HF radio.

Note: source routing will not be supported. Routing will be the responsibility of the HMTP server, which enjoys better access to shore directory services.

F.3.1. Introduction

The objective of HF Mail Transfer Protocol (HMTP) is to transfer mail reliably and efficiently over an HF radio link using the system described in the mandatory part of this STANAG.

F.3.2. THE SMTP MODEL

The HMTP is an adaptation of the SMTP, therefore the SMTP model is presented here. The SMTP design is based on the following model of communication: as the result of a user mail request, the SMTP client establishes a two-way transmission channel to an SMTP server. The SMTP server may be either the ultimate destination or an intermediate. SMTP commands are generated by the SMTP client and sent to the SMTP server. SMTP replies are sent from the SMTP server to the SMTP client in response to the commands.

In order to increase efficiency and reduce response time over low data rate channels, the HMTP combines certain steps which are separate in the SMTP. Once the transmission channel is established, the SMTP client sends a MAIL command indicating the sender of the mail. If the SMTP-server can accept mail it responds with an OK reply. The SMTP client then sends a RCPT command identifying a recipient of the mail. If the SMTP-server can accept mail for that recipient it responds with an OK reply; if not, it responds with a reply rejecting that recipient (but not the whole mail transaction). The SMTP client and SMTP-server may negotiate several recipients. When the recipients have been negotiated the SMTP client sends the mail data, terminating with a special sequence. If the SMTP-server successfully processes the mail data it responds with an OK reply. The SMTP dialog is purposely lock-step, one-at-a-time.

HMTP combines the MAIL, RCPT, and the mail data for multiple mail messages into a single transmission.

The HMTP server shall provide a relay capability for the client. The argument to the MAIL command is in the form “user@hostname”, which specifies who the mail is from. The argument to the RCPT command is in the same form and specifies the ultimate destination of the mail. The HMTP server shall forward mail. A destination will only be rejected if it cannot be understood by the server. Source routing shall not be used; the argument for the RCPT command shall be the ultimate destination of the mail.

Example of the HMTP Procedure

This example shows mail sent by Smith at host Alpha, to Jones, Green, and Brown at different domains. Each individual message is terminated with a <CRLF>.<CRLF>, and the sequence with <CRLF>.<CRLF><CRLF>.<CRLF>.

```
C: MAIL MULTIPLE
C: MAIL FROM:<Smith@Alpha >
C: RCPT TO:<Jones@Beta>
C: RCPT TO:<Green@gamma>
C: RCPT TO:<Brown@delta>
C: DATA
C: Blah blah blah...
C: ...etc. etc. etc.
C: <CRLF>.<CRLF>
C: MAIL FROM: etc
C: …
C: <CRLF>.<CRLF><CRLF>.<CRLF>


S: 250 Source <Smith@Alpha> OK
S: 250 Destination <Jones@Beta.> OK
S: 550 Destination <Green@gamma> not known
S: 250 Destination <Brown@delta> OK
S: 250 DATA OK
S: etc
```

The mail has now been accepted for Jones and Brown.

F.3.3 THE HMTP SPECIFICATIONS

F.3.3.1 HMTP COMMANDS

F.3.3.1.1 COMMAND SEMANTICS

The HMTP commands define the mail transfer or the mail system function requested by the user. HMTP commands are character strings terminated by <CRLF>. The command codes themselves are alphabetic characters terminated by <SP> if parameters follow and <CRLF> otherwise. The syntax of mailboxes must conform to server site conventions. The HMTP commands are discussed below. The HMTP replies are discussed in the Section 4.2.

A mail transaction involves several data objects which are communicated as arguments to different commands. The reverse-path is the argument of the MAIL command, the forward-path is the argument of the RCPT command, and the mail data is the argument of theDATA command. These arguments or data objects must be transmitted and held pending the confirmation communicated by the end of mail data indication which finalizes the transaction. The model for this is that distinct buffers are provided to hold the types of data objects, that is, there is a reverse-path buffer, a forward-path buffer, and a mail data buffer. Specific commands cause information to be appended to a specific buffer, or cause one or more buffers to be cleared.

HELLO (HELO)

This command is used to identify the HMTP client to the HMTP server. The argument field contains the host name of the HMTP client.

The HMTP server identifies itself to the HMTP client in the connection greeting reply, and in the response to this command.

This command and an OK reply to it confirm that both the HMTP client and the HMTP server are in the initial state, that is, there is no transaction in progress and all state tables and buffers are cleared.

MAIL MULTIPLE

This command is used to advise the server that a sequence of more than one MAIL transaction will follow. The default, if this command is not sent, is a single MAIL transaction.

MAIL (MAIL)

This command is used to initiate a mail transaction in which the mail data is delivered to one or more mailboxes. The argument field contains a reverse-path.

The reverse-path consists of an optional list of hosts and the sender mailbox. When the list of hosts is present is a "reverse" source route and indicates that the mail was relayed through each host on the list (the first host in the list was the most recent relay). This list is used as a source route to return non-delivery notices to the sender. As each relay host adds itself to the beginning of the list, it must use its name as known in the IPCE to which it is relaying the mail rather than the IPCE from which the mail came (if they are different). In some types of error reporting messages (for example, undeliverable mail notifications) the reverse-path may be null (see Example 7).

This command clears the reverse-path buffer, the forward-path buffer, and the mail data buffer; and inserts the reverse-path information from this command into the reverse-path buffer.

RECIPIENT (RCPT)

This command is used to identify an individual recipient of the mail data; multiple recipients are specified by multiple use of this command.

The forward-path consists of a required destination mailbox. Source routing shall not be used.

When mail is relayed, the relay host must put itself at the beginning of the reverse-path. When mail reaches its ultimate destination, the HMTP server inserts it into the destination mailbox in accordance with its host mail conventions.

DATA (DATA)

The server treats the lines following the command as mail data from the sender. This command causes the mail data from this command to be appended to the mail data buffer. The mail data may contain any of the 128 ASCII character codes.

The mail data is terminated by a line containing only a period, that is the character sequence "<CRLF>.<CRLF>" (see Section 4.5.2 on Transparency). This is the end of mail data indication.

The end of mail data indication requires that the server must now process the stored mail transaction information. This processing consumes the information in the reverse-path buffer, the forward-path buffer, and the mail data buffer, and on the completion of this command these buffers are cleared. If the processing is successful the server must send an OK reply. If the processing fails completely the server must send a failure reply.

When the HMTP server accepts a message either for relaying or for final delivery it inserts at the beginning of the mail data a time stamp line. The time stamp line indicates the identity of the host that sent the message, and the identity of the host that received the message (and is inserting this time stamp), and the date and time the message was received. Relayed messages will have multiple time stamp lines.

When the HMTP server makes the "final delivery" of a message it inserts at the beginning of the mail data a return path line. The return path line preserves the information in the <reverse-path> from the MAIL command. Here, final delivery means the message leaves the HMTP world. Normally, this would mean it has been delivered to the destination user, but in some cases it may be further processed and transmitted by another mail system.

It is possible for the mailbox in the return path be different from the actual sender's mailbox, for example, if error responses are to be delivered a special error handling mailbox rather than the message senders.

The preceding two paragraphs imply that the final mail data will begin with a return path line, followed by one or more time stamp lines. These lines will be followed by the mail data header and body [2]. See Example 8.

Special mention is needed of the response and further action required when the processing following the end of mail data indication is partially successful. This could arise if after accepting several recipients and the mail data, the HMTP server finds that the mail data can be successfully delivered to some of the recipients, but it cannot be to others (for example, due to mailbox space allocation problems). In such a situation, the response to the DATA command must be an OK reply. But, the HMTP server must compose and send an "undeliverable mail" notification message to the originator of the message. Either a single notification which lists all of the recipients that failed to get the message, or separate notification messages must be sent for each failed recipient (see Example 7). All undeliverable mail notification messages are sent using the MAIL command (even if they result from processing a SEND, SOML, or SAML command).

RESET (RSET)

This command specifies that the current mail transaction is to be aborted. Any stored sender, recipients, and mail data must be discarded, and all buffers and state tables cleared. The server must send an OK reply.

QUIT (QUIT)

This command specifies that the server must send an OK reply, and then close the transmission channel.

The server should not close the transmission channel until it receives and replies to a QUIT command (even if there was an error). The client should not close the transmission channel until it send a QUIT command and receives the reply (even if there was an error response to a previous command). If the connection is closed prematurely the server should act as if a RSET command had been received (canceling any pending transaction, but not undoing any previously completed transaction), the client should act as if the command or transaction in progress had received a temporary error (4xx).

F.3.3.1.2 COMMAND SYNTAX

The commands consist of a command code followed by an argument field. Command codes are four alphabetic characters. Upper and lower case alphabetic characters are to be treated identically. Thus, any of the following may represent the mail command:

MAIL   Mail   mail   MaIl   mAIl

This also applies to any symbols representing parameter values, such as "TO" or "to" for the forward-path. Command codes and the argument fields are separated by one or more spaces. However, within the reverse-path and destination arguments case is important. In particular, in some hosts the user "smith" is different from the user "Smith".

The argument field consists of a variable length character string ending with the character sequence <CRLF>. The receiver is to take no action until this sequence is received.

Square brackets denote an optional argument field. If the option is not taken, the appropriate default is implied.

The following are the HMTP commands:

HELO HMTP <SP> <domain> <CRLF>

MAIL MULTIPLE

MAIL <SP> FROM:<reverse-path> <CRLF>

RCPT <SP> TO:<destination> <CRLF>

DATA <CRLF>

RSET <CRLF>

QUIT <CRLF>

F.3.3.1.3 SMTP REPLIES

Replies to HMTP commands are devised to ensure the synchronization of requests and actions in the process of mail transfer, and to guarantee that the HMTP client always knows the state of the HMTP server. Every command must generate exactly one reply.

An HMTP reply consists of a three digit number (transmitted as three alphanumeric characters) followed by some text. The number is intended for use by automata to determine what state to enter next; the text is meant for the human user. It is intended that the three digits contain enough encoded information that the HMTP client need not examine the text and may either discard it or pass it on to the user, as appropriate. In particular, the text may be server-dependent and context dependent, so there are likely to be varying texts for each reply code. A discussion of the theory of reply codes can be found in Appendix E to RFC 822. Formally, a reply is defined to be the sequence: a three-digit code, <SP>, one line of text, and <CRLF>. In general, replies in HMTP will be multiline replies.

F.3.3.1.3.1. REPLY CODES BY FUNCTION GROUPS

500 Syntax error, command unrecognized [This may include errors such as command line too long]
501 Syntax error in parameters or arguments
502 Command not implemented
503 Bad sequence of commands
504 Command parameter not implemented

220 <domain> HMTP Service ready
221 <domain> Service closing transmission channel
421 <domain> Service not available, closing transmission channel [This may be a reply to any command if the service knows it must shut down]

250 Requested mail action okay, completed
450 Requested mail action not taken: mailbox unavailable [E.g., mailbox busy]
550 Requested action not taken: mailbox unavailable [E.g., mailbox not found, no access]
451 Requested action aborted: error in processing
452 Requested action not taken: insufficient system storage

552 Requested mail action aborted: exceeded storage allocation
553 Requested action not taken: mailbox name not allowed [E.g., mailbox syntax incorrect]
354 Start mail input; end with <CRLF>.<CRLF>
554 Transaction failed

F.3.4 TRANSPARENCY

Without some provision for data transparency the character sequence "<CRLF>.<CRLF>" ends the mail text and cannot be sent by the user. In general, users are not aware of such "forbidden" sequences. To allow all user composed text to be transmitted transparently the following procedures are used.

1. Before sending a line of mail text the sender-HMTP checks the first character of the line. If it is a period, one additional period is inserted at the beginning of the line.

2. When a line of mail text is received by the receiver-HMTP it checks the line. If the line is composed of a single period it is the end of mail. If the first character is a period and there are other characters on the line, the first character is deleted.

The mail data may contain any of the 128 ASCII characters. All characters are to be delivered to the recipient's mailbox including format effectors and other control characters. If the transmission channel provides an 8-bit byte (octets) data stream, the 7-bit ASCII codes are transmitted right justified in the octets with the high order bits cleared to zero.

In some systems it may be necessary to transform the data as it is received and stored. This may be necessary for hosts that use a different character set than ASCII as their local character set, or that store data in records rather than strings. If such transforms are necessary, they must be reversible -- especially if such transforms are applied to mail being relayed.

F.4 POP3 FOR USE OVER STANAG 5066 TRANSPORT (HF-POP3)

This section of this Annex presents an adaptation of the POP3 protocol, as described in RFC 1939, for use over the transport system described in the main body of and mandatory annexes to this STANAG. Large parts of the RFC are reproduced here, however there are differences so read carefully.

F.4.1 Introduction

It will in general be impractical to maintain a message transport system (MTS) over HF radio to allow immediate delivery of messages to a node when that node relies on HF radio for connectivity. It is still necessary to be able to manage the transfer of mail to these smaller nodes. To solve this problem, a node which can support an MTS entity offers a maildrop service to these less endowed nodes. The Post Office Protocol - Version 3 (POP3), as modified in this Annex, is intended to permit a workstation to dynamically access a maildrop on a server host over HF radio. Usually, this means that the POP3 protocol is used to allow a node to retrieve mail that the server is holding for it. This annex presents a revision of the POP3 protocol which is intended to provide efficient operation over HF radio links. This revised protocol will be referred to as “HF-POP” in this annex.

For the remainder of this annex, the term "client host" refers to a host making use of the HF-POP service, while the term "server host" refers to a host which offers the HF-POP service.

This annex does not specify how a client host enters mail into the transport system, although a method consistent with the philosophy of this annex is presented here:

When the user agent on a client host wishes to enter a message into the transport system, it establishes an HMTP connection to its relay host and sends all mail to it.

F.4.2 Basic Operation

Initially, the server host starts the HF-POP3 service by listening for incoming connections over the HF subnet. When a client host wishes to make use of the service, it establishes a connection with the server host. When the connection is established, the HF-POP3 server sends a greeting. The client and POP3 server then exchange commands and responses (respectively) until the connection is closed or aborted.

Commands in the HF-POP consist of a case-insensitive keyword, possibly followed by one or more arguments. All commands are terminated by a CRLF pair. Keywords and arguments consist of printable ASCII characters. Keywords and arguments are each separated by a single SPACE character. Keywords are three or four characters long. Each argument may be up to 40 characters long.

Responses in the HF-POP consist of a status indicator and a keyword possibly followed by additional information. All responses are terminated by a CRLF pair. Responses may be up to 512 characters long, including the terminating CRLF. There are two status indicators: positive ("+OK") and negative ("-ERR"). Servers shall send the "+OK" and "-ERR" in upper case.

Responses to certain commands are multi-line. In these cases, which are clearly indicated below, after sending the first line of the response and a CRLF, any additional lines are sent, each terminated by a CRLF pair. When all lines of the response have been sent, a final line is sent, consisting of a termination octet (decimal code 046, ".") and a CRLF pair. If any line of the multi-line response begins with the termination octet, the line is "byte-stuffed" by pre-pending the termination octet to that line of the response. Hence a multi-line response is terminated with the five octets "CRLF.CRLF". When examining a multi-line response, the client checks to see if the line begins with the termination octet. If so and if octets other than CRLF follow, the first octet of the line (the termination octet) is stripped away. If so and if CRLF immediately follows the termination character, then the response from the POP server is ended and the line containing ".CRLF" is not considered part of the multi-line response.

A HF-POP3 session progresses through a number of states during its lifetime. Once the SnAP connection has been opened and the HF-POP3 server has sent the greeting, the session enters the AUTHORIZATION state. In this state, the client must identify itself to the HF-POP3 server. Once the client has successfully done this, the server acquires resources associated with the client's maildrop, and the session enters the TRANSACTION state. In this state, the client requests actions on the part of the HF-POP3 server. When the client has issued the QUIT command, the session enters the UPDATE state. In this state, the HF-POP3 server releases any resources acquired during the TRANSACTION state and says goodbye. The connection is then closed.

A server MUST respond to an unrecognized, unimplemented, or syntactically invalid command by responding with a negative status indicator. A server MUST respond to a command issued when the session is in an incorrect state by responding with a negative status indicator. There is no general method for a client to distinguish between a server which does not implement an optional command and a server which is unwilling or unable to process the command.

A HF-POP3 server MAY have an inactivity autologout timer. Such a timer MUST be of at least 10 minutes' duration. The receipt of any command from the client during that interval should suffice to reset the autologout timer. When the timer expires, the session does NOT enter the UPDATE state--the server should close the connection without removing any messages or sending any response to the client.

F.4.3 The AUTHORIZATION State

Once the connection has been opened by a HF-POP3 client, the HF-POP3 server shall issue a one line greeting as follows:

S: +OK HF-POP3 (STANAG 5066) server ready <process-ID.clock@hostname>

(Responses which do not conform to this format shall be interpreted by the HF_POP3 client as negative responses.) The HF-POP3 session is now in the AUTHORIZATION state. The client must now identify and authenticate itself to the HF-POP3 server. The APOP command shall be used for this purpose.

APOP name digest

Arguments:
a string identifying a mailbox and a MD5 digest string (both required)

Restrictions:
may only be given in the AUTHORIZATION state after the POP3 greeting or after an unsuccessful USER or PASS command

Discussion:

In the context of this standard, the APOP command provides both origin authentication and replay protection. An HF-POP3 servers shall include a timestamp in its banner greeting. The syntax of the timestamp corresponds to the `msg-id' in [RFC822], and MUST be different each time the POP3 server issues a banner greeting. For example, on a UNIX implementation in which a separate UNIX process is used for each instance of a POP3 server, the syntax of the timestamp might be:

<process-ID.clock@hostname>

where `process-ID' is the decimal value of the process's PID, clock is the decimal value of the system clock, and hostname is the fully-qualified domain-name corresponding to the host where the POP3 server is running.

The POP3 client makes note of this timestamp, and then issues the APOP command. The `name' parameter has identical semantics to the `name' parameter of the USER command. The `digest' parameter is calculated by applying the MD5 algorithm [RFC1321] to a string consisting of the timestamp (including angle-brackets) followed by a shared secret. This shared secret is a string known only to the POP3 client and server. Great care should be taken to prevent unauthorized disclosure of the secret, as knowledge of the secret will allow any entity to successfully masquerade as the named user. The `digest' parameter itself is a 16-octet value which is sent in hexadecimal format, using lower-case ASCII characters.

When the POP3 server receives the APOP command, it verifies the digest provided. If the digest is correct, the POP3 session enters the TRANSACTION state and the POP3 server issues a positive response including the LIST response for the maildrop, as shown below. Otherwise, a negative response is issued and the POP3 session remains in the AUTHORIZATION state.

Note that as the length of the shared secret increases, so does the difficulty of deriving it. As such, shared secrets should be long strings (considerably longer than the 8-character example shown below).

Possible Responses:

```
S: +OK maildrop locked and ready; maildrop has 1 message (320 octets)
S: 1 120
S: 2 200
S: .

S: -ERR permission denied
```

Examples:

```
S: +OK POP3 server ready <1896.697170952@dbc.mtview.ca.us>
C: APOP mrose c4c9334bac560ecc979e58001b3e22fb
S: +OK maildrop has 1 message (369 octets)
S: 1 369
```

In this example, the shared secret is the string `tanstaaf'. Hence, the MD5 algorithm is applied to the string

<1896.697170952@dbc.mtview.ca.us>tanstaaf

which produces a digest value of

c4c9334bac560ecc979e58001b3e22fb

Once the POP3 server has determined through the use of the APOP command that the client should be given access to the appropriate maildrop, the HF-POP3 server then acquires an exclusive-access lock on the maildrop, as necessary to prevent messages from being modified or removed before the session enters the UPDATE state. If the lock is successfully acquired, the POP3 server responds with a positive status indicator plus the response to the LIST command with no arguments (described in the following section on the TRANSACTION state). The HF-POP3 session now enters the TRANSACTION state, with no messages marked as deleted. If the maildrop cannot be opened for some reason (for example, a lock can not be acquired, the client is denied access to the appropriate maildrop, or the maildrop cannot be parsed), the HF-POP3 server responds with a negative status indicator. (If a lock was acquired but the HF-POP3 server intends to respond with a negative status indicator, the HF-POP3 server must release the lock prior to rejecting the command.) After returning a negative status indicator, the server may close the connection. If the server does not close the connection, the client may either issue a new authentication command and start again, or the client may issue the QUIT command.

After the HF-POP3 server has opened the maildrop, it assigns a message- number to each message, and notes the size of each message in octets. The first message in the maildrop is assigned a message-number of "1", the second is assigned "2", and so on, so that the nth message in a maildrop is assigned a message-number of "n". In HF-POP3 commands and responses, all message-numbers and message sizes are expressed in base-10 (i.e., decimal).

Here is the summary for the QUIT command when used in the AUTHORIZATION state:

QUIT

Arguments: none

Restrictions: none

Possible Responses:
+OK

Examples:
C: QUIT
S: +OK dewey POP3 server signing off

F.4.4 The TRANSACTION State

Once the client has successfully identified itself to the POP3 server and the POP3 server has locked and opened the appropriate maildrop, the POP3 session is now in the TRANSACTION state. Note that this transition occurs after the APOP line from the client is processed and before the server generates the UIDL response. The client may now issue any of the following POP3 commands repeatedly. After each command, the POP3 server issues a response. Eventually, the client issues the QUIT command and the POP3 session enters the UPDATE state.

Here are the POP3 commands valid in the TRANSACTION state:

LIST [msg]

Arguments:
a message-number (optional), which, if present, may NOT refer to a message marked as deleted

Restrictions:
may only be given in the TRANSACTION state

Discussion:
If an argument was given and the POP3 server issues a positive response with a line containing information for that message. This line is called a "scan listing" for that message.

If no argument was given and the POP3 server issues a positive response, then the response given is multi-line. After the initial +OK, for each message in the maildrop, the POP3 server responds with a line containing information for that message. This line is also called a "scan listing" for that message. If there are no messages in the maildrop, then the POP3 server responds with no scan listings--it issues a positive response followed by a line containing a termination octet and a CRLF pair.

In order to simplify parsing, all POP3 servers are required to use a certain format for scan listings. A scan listing consists of the message-number of the message, followed by a single space and the exact size of the message in octets, followed by a CRLF pair. Methods for calculating the exact size of the message are described in the "Message Format" section of RFC 1939.

Note that messages marked as deleted are not listed.

Possible Responses:
+OK scan listing follows
-ERR no such message

Examples:
C: LIST
S: +OK 2 messages (320 octets)
S: 1 120
S: 2 200
S: .
...
C: LIST 2
S: +OK 2 200
...
C: LIST 3
S: -ERR no such message, only 2 messages in maildrop

RETR nn

Arguments:
a message-number (optional) which may NOT refer to a message marked as deleted

Restrictions:
may only be given in the TRANSACTION state

Discussion:
If the POP3 server issues a positive response, then the response given is multi-line. If an argument was given, after the initial +OK, the POP3 server sends the message corresponding to the given message-number, being careful to byte-stuff the termination character (as with all multi-line responses). If no argument was given, the server sends all messages in the drop which are not marked as deleted. Use UIDL to identify boundary between messages? If so, need to include the UIDL response as part of the response to APOP.

Possible Responses:
+OK message nn follows
+OK m messages follow
-ERR no such message

Examples:
C: RETR 1

S: +OK 120 octets
S: <the POP3 server sends the entire message here>
S: .

DELE msg

Arguments:
a message-number (required) which may NOT refer to a message marked as deleted

Restrictions:
may only be given in the TRANSACTION state

Discussion:
The POP3 server marks the message as deleted. Any future reference to the message-number associated with the message in a POP3 command generates an error. The POP3 server does not actually delete the message until the POP3 session enters the UPDATE state.

Possible Responses:
+OK message deleted
-ERR no such message

Examples:
C: DELE 1
S: +OK message 1 deleted
...
C: DELE 2
S: -ERR message 2 already deleted

RSET

Arguments: none

Restrictions:
may only be given in the TRANSACTION state

Discussion:
If any messages have been marked as deleted by the POP3 server, they are unmarked. The POP3 server then replies with a positive response.

Possible Responses:
+OK

Examples:
C: RSET
S: +OK maildrop has 2 messages (320 octets)

STAT

Arguments: none

Restrictions:
may only be given in the TRANSACTION state

Discussion:
The POP3 server issues a positive response with a line containing information for the maildrop. This line is called a "drop listing" for that maildrop.

In order to simplify parsing, all POP3 servers are required to use a certain format for drop listings. The positive response consists of "+OK" followed by a single space, the number of messages in the maildrop, a single space, the size of the maildrop in octets, and the line shall be terminated with a CRLF pair.

Note that messages marked as deleted are not counted in either total.

Possible Responses:
+OK nn mm

Examples:
C: STAT
S: +OK 2 320

UIDL [msg] (OPTIONAL)

Arguments:
a message-number (optional), which, if present, may NOT refer to a message marked as deleted

Restrictions:
may only be given in the TRANSACTION state.

Discussion:
If an argument was given and the POP3 server issues a positive response with a line containing information for that message. This line is called a "unique-id listing" for that message.

If no argument was given, then the response given is multi-line. After the initial +OK, for each message in the maildrop, the HF-POP3 server responds with a line containing information for that message. This line is called a "unique-id listing" for that message.

In order to simplify parsing, all POP3 servers are required to use a certain format for unique-id listings. A unique-id listing consists of the message-number of the message, followed by a single space and the unique-id of the message. No information follows the unique-id in the unique-id listing.

The unique-id of a message is an arbitrary server-determined string, consisting of one to 70 characters in the range 0x21 to 0x7E, which uniquely identifies a message within a maildrop and which persists across sessions. This persistence is required even if a session ends without entering the UPDATE state. The server should never reuse an unique-id in a given maildrop, for as long as the entity using the unique-id exists.

Note that messages marked as deleted are not listed.

While it is generally preferable for server implementations to store arbitrarily assigned unique-ids in the maildrop, this specification is intended to permit unique-ids to be calculated as a hash of the message. Clients should be able to handle a situation where two identical copies of a message in a maildrop have the same unique-id.

Possible Responses:
+OK unique-id listing follows
-ERR no such message

Examples:

```
C: UIDL
S: +OK
S: 1 whqtswO00WBw418f9t5JxYwZ
S: 2 QhdPYR:00WBw1Ph7x7
S: .
...
C: UIDL 2
S: +OK 2 QhdPYR:00WBw1Ph7x7
...
C: UIDL 3
S: -ERR no such message, only 2 messages in maildrop
```

F.4.5 The UPDATE State

When the client issues the QUIT command from the TRANSACTION state, the POP3 session enters the UPDATE state. (Note that if the client issues the QUIT command from the AUTHORIZATION state, the POP3 session terminates but does NOT enter the UPDATE state.)

If a session terminates for some reason other than a client-issued QUIT command, the POP3 session does NOT enter the UPDATE state and MUST not remove any messages from the maildrop.

QUIT

Arguments: none

Restrictions: none

Discussion:
The POP3 server removes all messages marked as deleted from the maildrop and replies as to the status of this operation. If there is an error, such as a resource shortage, encountered while removing messages, the maildrop may result in having some or none of the messages marked as deleted be removed. In no case may the server remove any messages not marked as deleted.

Whether the removal was successful or not, the server then releases any exclusive-access lock on the maildrop and closes the connection.

Possible Responses:
+OK
-ERR some deleted messages not removed

Examples:
C: QUIT
S: +OK dewey POP3 server signing off (maildrop empty)
...
C: QUIT
S: +OK dewey POP3 server signing off (2 messages left)
...

TOP msg n

Arguments:
a message-number (required) which may NOT refer to to a message marked as deleted, and a non-negative number of lines (required)

Restrictions:
may only be given in the TRANSACTION state

Discussion:
If the POP3 server issues a positive response, then the response given is multi-line. After the initial +OK, the POP3 server sends the headers of the message, the blank line separating the headers from the body, and then the number of lines of the indicated message's body, being careful to byte-stuff the termination character (as with all multi-line responses).

Note that if the number of lines requested by the POP3 client is greater than than the number of lines in the body, then the POP3 server sends the entire message.

Possible Responses:
+OK top of message follows
-ERR no such message

Examples:
C: TOP 1 10
S: +OK
S: <the POP3 server sends the headers of the message, a blank line, and the first 10 lines of the body of the message>
S: .
...
C: TOP 100 3
S: -ERR no such message

8. Scaling and Operational Considerations

The HF-POP3 protocol implements a special case of a site policy in order to prevent users from accumulating large mail queues on the server system. These large queues can result in a full disk and system crashes; this problem is frequently observed in POP3 servers. Messages may only be downloaded once from the server, and shall be deleted after this has been accomplished. This could be implemented in HF-POP3 server software by the following mechanism: "following a POP3 login by a client which was ended by a QUIT, delete all messages downloaded during the session with the RETR command". It is important not to delete messages in the event of abnormal connection termination (ie, if no QUIT was received from the client) because the client may not have successfully received or stored the messages. Servers implementing a download-and-delete policy may also wish to disable or limit the optional TOP command, since it could be used as an alternate mechanism to download entire messages.

F.5 Z-modem (modified) file transfer

This application allows the transfer of files using a modified Z-modem protocol over the HF subnet. This allows file transfer without using the HMTP service. It has the added benefit that an interrupted file transfer will resume at the point of interruption rather than starting over again.

Details to be added.

F.6 Operator orderwire application

This annex specifies the use of existing primitives for transfer of operator orderwire messages over a radio link using the HF subnet described in the mandatory part of this STANAG. These messages would generally be displayed on the HF node host system, and the subnet client would likewise reside on the same host as the node. This is an implementation decision and is described only as an example.

For acknowledged orderwire, the procedures described in section F.1 shall be used. For non-acknowledged orderwire, the procedures described in section F.2 shall be used. The orderwire clients shall attach to SAP ID 6.

F.7 Subnet Management Client

This annex will define messages for control of the local node using the S_MANAGEMENT_MSG_REQUEST, S_MANAGEMENT_MSG_INDICATION, and S_SUBNET_AVAILABILITY primitives, and for coordination with distant subnet management clients using the S_UNIDATA primitives. Subnetwork management clients shall attach to SAP ID 0. The rank of the client must be 15 if the client is entitled to submit commands which will change the configuration of the node or subnetwork.

F.8 Frequency Availability Broadcast

Overview

Frequency availability information will be compiled by the shore site and made available by non-ARQ (broadcast) service to units which want to use the ship-shore facilities. This section describes the interaction of the associated clients with the subnetwork.

FAB clients

When the frequency availability information (FAI) is distributed by non-ARQ (broadcast) service, it will be referred to as a frequency availability broadcast, or FAB. There will be two types of clients: a FAB generator or compiler, and a FAB receiver.

The “FAB generator” client (at the shore site) shall accept information from an external source and shall then format this information as described below and pass it to one or more

HF nodes using the primitives specified below and defined in the mandatory annexes of this document (STANAG 5066). The U_PDU format described elsewhere in this Annex shall be followed.

The “FAB receiver” client at the ship shall receive FAI updates from the HF node via SAP ID 7. This information may be displayed to an operator by the FAB receive client, or passed to an external destination.

The “FAB generator” client shall attach to SAP ID 7 of the local HF node. The “FAB generator” client shall submit FIA updates to the HF node(s) using S_UNIDATA_REQUEST primitives. Within this primitive,

- The destination node address shall be set to the appropriate broadcast address; this address shall be selectable.
- The destination SAP ID shall be 7.
- TTL shall not exceed the interval between FAI updates.
- The Delivery mode (also the bind request primitive) shall be set so that the FAI update is sent using non-ARQ (broadcast) D_PDUs.

The format of frequency availability information is not defined in this document.

Recommendations and Implementation Notes

For a FAB, the updates should be generated and transmitted on a periodic schedule and in addition any time there is a change in the broadcast status. The updates should not be generated so often that queues of FAB information build up. These queues will delay the transmission of current FAI while old FAI is transmitted. Mathematically, this rule is

Update rate x (characters in FAI x bits/character)+240 bits 5066 overhead < broadcast data rate

If we have a FAI of 200 Baudot characters and a 300 bps broadcast, this becomes

update rate x 1640 bits/update < 300 bps

update rate < 300 bps/1640 bits per update = 0.183 updates/second

So, the interval between FAI updates should be at least 5.47 seconds for this example. If the system will use a common shore answering frequency for FAB plus ARQ responses, this number is a very loose lower bound. A more reasonable figure would be when the status information changes , and in the absence of changes, an update period on the order of once every 30 or 60 seconds.

The FAB should be addressed to the broadest possible (global) group address.

When using the STANAG 4285 and 4529 waveforms, the FAB must be sent at a known data rate. While it may be intermingled with ARQ traffic on a general shore answering frequency, the transmit rate on that frequency must be maintained at the fixed, known data rate. Note that, depending on the implementation of the shore node, it may be possible to vary the data rate of the connection oriented (ARQ) transmissions while leaving the connectionless, broadcast data at a fixed known data rate.

Shore answering frequencies should never be unavailable (unless they are assigned to an MRL circuit) - it is always possible to use the same shore answering frequency to transmit CARB/CAB/FAB and ARQ responses for a number of different ship-shore circuits.

It would be possible to allow a manufacturer to implement the FAB receive client and integrate it with their equipment in a closed fashion, ie, a proprietary interface. This would limit to some extent the ability to replace the equipment in the future because the FAB receive client would also have to be replaced.

Annex G: Waveforms for Data Rates Above 2400 bps (information only)

G.1.0 Introduction

This Annex presents a modem waveform and coding specification for data rates of 3200, 4800, 6400 and 9600 bps. This Annex can be viewed as analogous to STANAG 4285 with Annex E coding. As much as possible, commonality with STANAG 4285 has been maintained. The serial-tone waveforms specified in this Annex use modulation techniques of greater complexity and data blocks larger than those found in other standards in order to achieve the efficiencies necessary to obtain the required data rates. A single coding option, a constraint length 7, rate 1/2 convolutional code, punctured to rate 3/4, is used for all data rates. A convolutional interleaver, very similar in design to that in Annex E of STANAG 4285, is used to obtain interleaving delays of 0.72 s and 10.8 s. In contrast to STANAG 4285, an option for bypassing the interleaver is also included in the specification.

The waveforms specified in this Annex can be used in the same manner as the waveforms specified in STANAG 4285, with code rates, interleaver settings and data rates specified in advance. Alternatively, however, provision has been made to allow the receiver to determine the waveform and interleaving settings from the received signal. Interleaver settings are explicitly transmitted as a part of the waveform; data rates can be deduced by using modulation recognition to ascertain the modulation used, and hence the data rate. This ‘‘autobaud’’ feature is critical in developing an efficient ARQ protocol for HF channels. The capability for the receive modem to deduce the data rate and interleaving setting should be considered to be an optional, though highly desirable, feature of the waveforms specified in this Annex.

G.2.0 Modulation

The symbol rate for all symbols will be 2400 baud with a minimum symbol clock accuracy of 10^{-5}. Phase-shift-keying (PSK) and quadrature amplitude modulation (QAM) modulation techniques are used. The sub-carrier (or pair of quadrature sub-carriers in the case of QAM) is centered at 1800 Hz. The phase of the Quadrature sub-carrier relative to the In-phase carrier will be -90 degreees. The correct relationship can be achieved by making the In-phase sub-carrier cos(1800 Hz) and the Quadrature sub-carrier -sin(1800 Hz). The frequency of the 1800 Hz sub-carriers will be accurate to a minimum of 10^{-5}.

The power spectral density of the modulator output signal should be constrained to be at least 20 dB down outside of the band from 300 Hz to 3300 Hz. Appropriate filtering will result in a ripple of no more than plus or minus 2 dB in the range from 800 Hz to 2800 Hz.

G.2.1 Known Symbols

For all known symbols, the modulation used will be PSK, with the symbol mapping shown in Table G-1 and Figure G-1. No scrambling will be applied to the known symbols.

Table G-1. 8PSK Symbol Mapping.

Symbol Number	Phase	In-Phase	Quadrature
0	0	1.000000	0.000000
1	π/4	0.707107	0.707107
2	π/2	0.000000	1.000000
3	3π/4	–0.707107	0.707107
4	π	–1.000000	0.000000
5	5π/4	–0.707107	–0.707107
6	3π/2	0.0000000	–1.000000
7	7π/4	0.707107	–0.707107

The complex number exp[jnπ/4] is linked with the symbol number n.

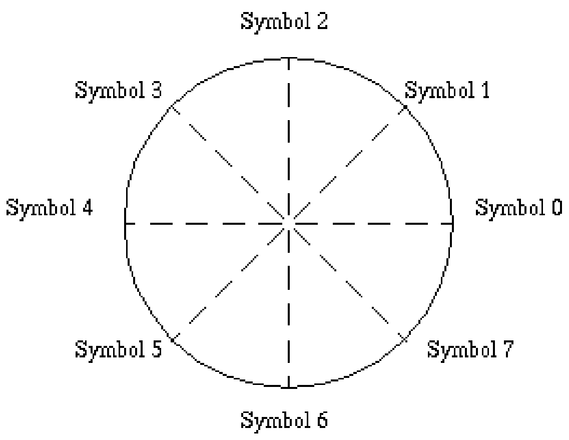


Figure G-1: 8 PSK signal constellation and symbol numbering.

G.2.2 Data Symbols

For data symbols, the modulation used will depend upon the data rate. Table G-2 specifies the modulation to be used with each data rate.

Table G-2. Modulation used to obtain each data rate.

Data Rate (bps)	Modulation
3200	QPSK
4800	8PSK
6400	16QAM
9600	64QAM

The 3200 bps QPSK constellation is scrambled to appear, on-air, as an 8PSK constellation. Both the 16QAM constellation and 64QAM constellations are variations of the standard square QAM constellations which have been modified to improve their peak-to-average ratios.

For the PSK constellations, a distinction is made between the data bits and the symbol number for the purposes of scrambling the QPSK modulation to appear as 8PSK, on-air. Scrambling is applied as a modulo 8 addition of a scrambling sequence to the 8PSK symbol number. This follows the convention used in STANAG 4285.

Transcoding is an operation which links a symbol to be transmitted to a group of data bits. For the 3200 bps user data rate, transcoding is achieved by linking one of the symbols specified in Table G-1 to a set of two consecutive data bits (dibit) as shown in the table below. In this table, the leftmost bit of the dibit is the oldest bit and the rightmost bit is the most recent bit.

Dibit	Symbol
00	0
01	2
11	4
10	6

For the 4800 bps user data rate, transcoding is achieved by linking one symbol to a set of three consecutive data bits (tribit) as shown in the table below. In this table, the leftmost bit of the tribit is the oldest bit and the rightmost bit is the most recent bit.

Tribit	Symbol
000	1
001	0
010	2
011	3
100	6
101	7
110	5
111	4

For the QAM constellations, no distinction is made between the number formed directly from the data bits and the symbol number. Each set of 4 bits (16QAM) or 6 bits (64QAM) is mapped directly to a QAM symbol. For example, the four bit grouping 0111 would map to symbol 7 in the 16QAM constellation while the 6 bit grouping 100011 would map to symbol 35 in the 64QAM constellation. Again, in each case the leftmost bit is the oldest bit and the rightmost bit is the most recent bit.

The mapping of bits to symbols for the QAM constellations has been selected to minimize the number of bit errors incurred when errors involve adjacent signalling points in the constellation. The 16QAM constellation points are shown in Figure G-2 below and specified in terms of their In-phase and Quadrature components in Table G-3 below.

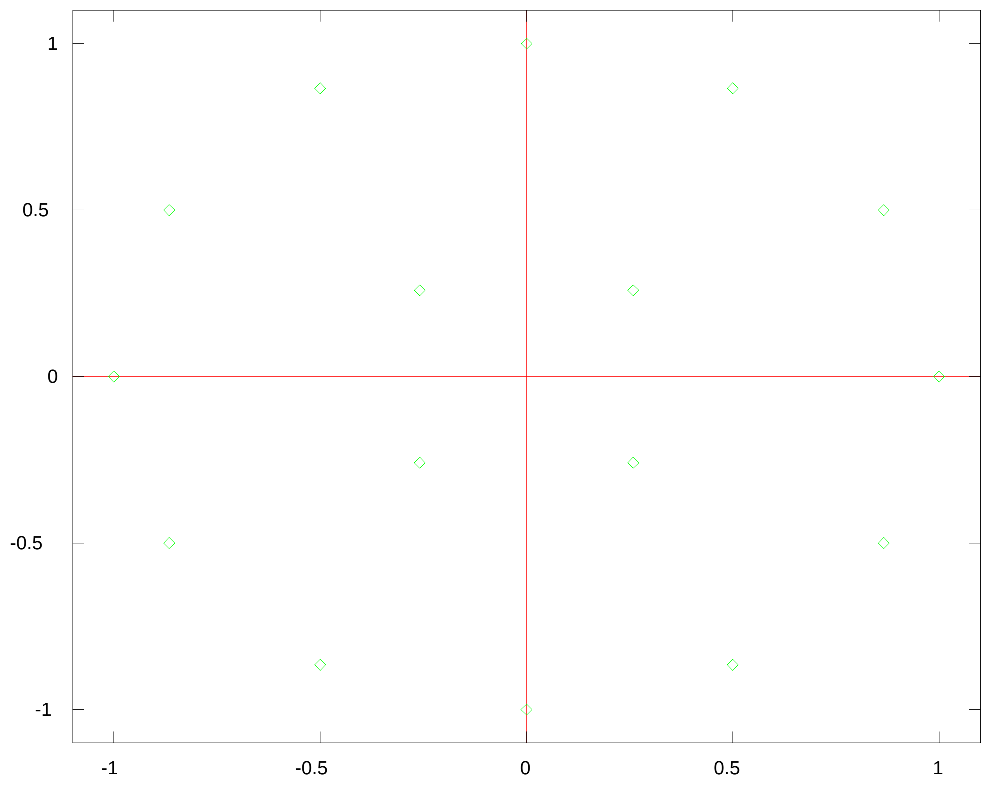


Figure G-2: 16QAM Signalling Constellation.

Table G-3. In-phase and Quadrature components of each 16QAM symbol.

Symbol Number	In-Phase	Quadrature
0	0.866025	0.500000
1	0.500000	0.866025
2	1.000000	0.000000
3	0.258819;	0.258819
4	-0.500000	0.866025
5	0.000000	1.000000
6	-0.866025	0.500000
7	-0.258819	0.258819
8	0.500000	-0.866025
9	0.000000	-1.000000
10	0.866025	-0.500000
11	0.258819	-0.258819
12	-0.866025	-0.500000
13	-0.500000	-0.866025
14	-1.000000	0.000000
15	-0.258819	-0.258819

The constellation points used by the 64QAM modulation are shown in Figure G-3 below. This constellation is a variation on the standard 8 x 8 square constellation which achieves better peak-to-average without sacrificing the very good pseudo-Gray code properties of the square constellation.

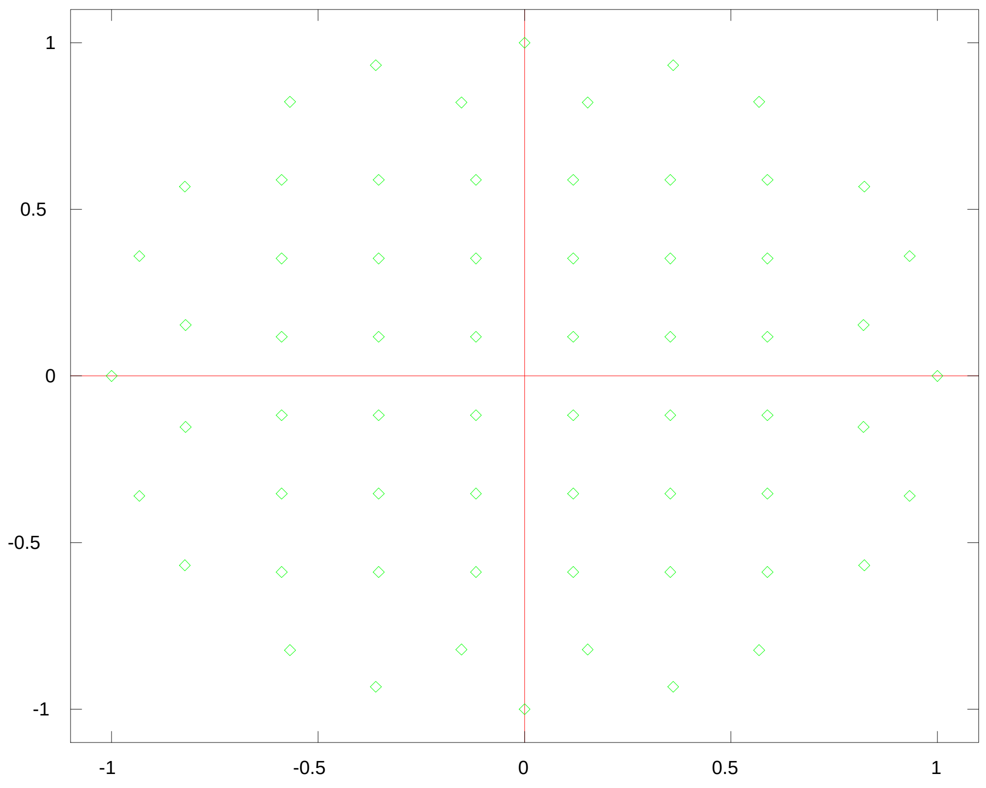


Figure G-3: 64QAM signalling constellation.

The In-phase and Quadrature components of each of the 64QAM symbols are specified in table G-4.

Table G-4. In-Phase and Quadrature Components of each 64QAM symbol.

Symbol Number	In-Phase	Quadrature	Symbol Number	In-Phase	Quadrature
0	1.000000	0.000000	32	0.000000	1.000000
1	0.822878	0.568218	33	-0.822878	0.568218
2	0.821137	0.152996	34	-0.821137	0.152996
3	0.932897	0.360142	35	-0.932897	0.360142
4	0.000000	-1.000000	36	-1.000000	0.000000
5	0.822878	-0.568218	37	-0.822878	-0.568218
6	0.821137	-0.152996	38	-0.821137	-0.152996
7	0.932897	-0.360142	39	-0.932897	-0.360142
8	0.568218	0.822878	40	-0.568218	0.822878
9	0.588429	0.588429	41	-0.588429	0.588429
10	0.588429	0.117686	42	-0.588429	0.117686
11	0.588429	0.353057	43	-0.588429	0.353057
12	0.568218	-0.822878	44	-0.568218	-0.822878
13	0.588429	-0.588429	45	-0.588429	-0.588429
14	0.588429	-0.117686	46	-0.588429	-0.117686
15	0.588429	-0.353057	47	-0.588429	-0.353057
16	0.152996	0.821137	48	-0.152996	0.821137
17	0.117686	0.588429	49	-0.117686	0.588429
18	0.117686	0.117686	50	-0.117686	0.117686
19	0.117686	0.353057	51	-0.117686	0.353057
20	0.152996	-0.821137	52	-0.152996	-0.821137
21	0.117686	-0.588429	53	-0.117686	-0.588429
22	0.117686	-0.117686	54	-0.117686	-0.117686
23	0.117686	-0.353057	55	-0.117686	-0.353057
24	0.360142	0.932897	56	-0.360142	0.932897
25	0.353057	0.588429	57	-0.353057	0.588429
26	0.353057	0.117686	58	-0.353057	0.117686
27	0.353057	0.353057	59	-0.353057	0.353057
28	0.360142	-0.932897	60	-0.360142	-0.932897
29	0.353057	-0.588429	61	-0.353057	-0.588429
30	0.353057	-0.117686	62	-0.353057	-0.117686
31	0.353057	-0.353057	63	-0.353057	-0.353057

G.2.3 Data Scrambling

Data symbols are scrambled by modulo 8 addition with a scrambling sequence for the 8PSK symbol constellation (3200 bps, 4800 bps). The data symbols for the 16QAM and 64QAM constellations are scrambled by using an exclusive or (XOR) operation. Sequentially, the data bits forming each symbol (4 for 16QAM and 6 for 64QAM) are XOR'd with an equal number of bits from the scrambling sequence.

In all cases, the scrambling sequence generator polynomial is x^9+x^4+1 and the generator is initialized to 1 at the start of each data block. A block diagram of the scrambling sequence generator is shown in Figure G-4.

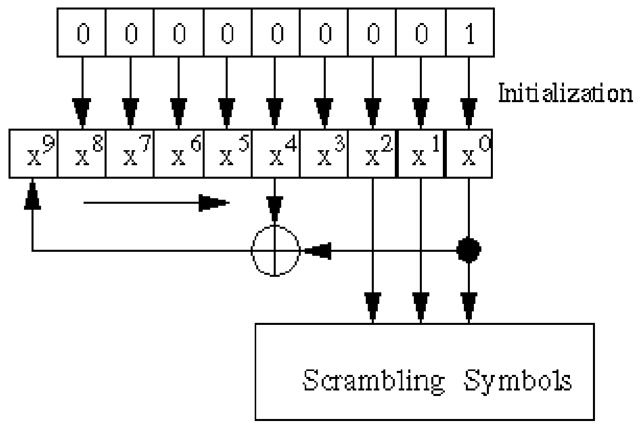


Figure G-4: Scrambling sequence generator. Symbol generation for 8PSK symbols shown, with 3 bits generated per symbol. 16QAM symbols require 4 bits from the shift register per symbol while 64 QAM symbols require 6 bits per symbol.

For 8PSK symbols, the scrambling is carried out by adding the triplet consisting of the last three bits in the shift register modulo 8 to the symbol number. For example, if the last three bits in the scrambling sequence shift register were 010 and the symbol number before scrambling was 6 (i.e. 110), symbol 0 (000) would be transmitted.

For 16QAM symbols, scrambling is carried out by XORing the 4 bit number consisting of the last four bits in the shift register with the symbol number. For example, if the last 4 bits in the scrambling sequence shift register were 0101 and the 16QAM symbol number before scrambling was 3 (i.e. 0011), symbol 6 (0110) would be transmitted.

For 64QAM symbols, scrambling is carried out by XORing the 6 bit number formed by the last six bits in the shift register with the symbol number.

After each data symbol is scrambled, the generator is iterated the required number of times to produce all new bits for use in scrambling the next symbol (i.e. 3 iterations for 8PSK, 4 iterations for 16QAM and 6 iterations for 64QAM).

The length of the scrambling sequence is 511 bits. For a 256 symbol frame with 6 bits per symbol, this means that the scrambling sequence will be repeated just slightly more than 3 times, although in terms of symbols, there will be no repetition.

G.3.0 Frame Structure

The frame structure used by the waveforms specified in this Annex is shown in Figure G-5. An initial 200 symbol preamble is followed by 72 blocks of alternating data and known symbols. Each data block, consisting of 256 data symbols, is followed by a mini-probe consisting of 31 symbols of known data. After 72 blocks, a 72 symbol subset of the initial preamble is reinserted to facillitate late acquisition, Doppler shift removal and sync adjustment. It should be noted that the total length of this segment of known data is actually 103 symbols - the 72 reinserted symbols plus the 31 symbol mini-probe segment following the last 256 symbol data block.

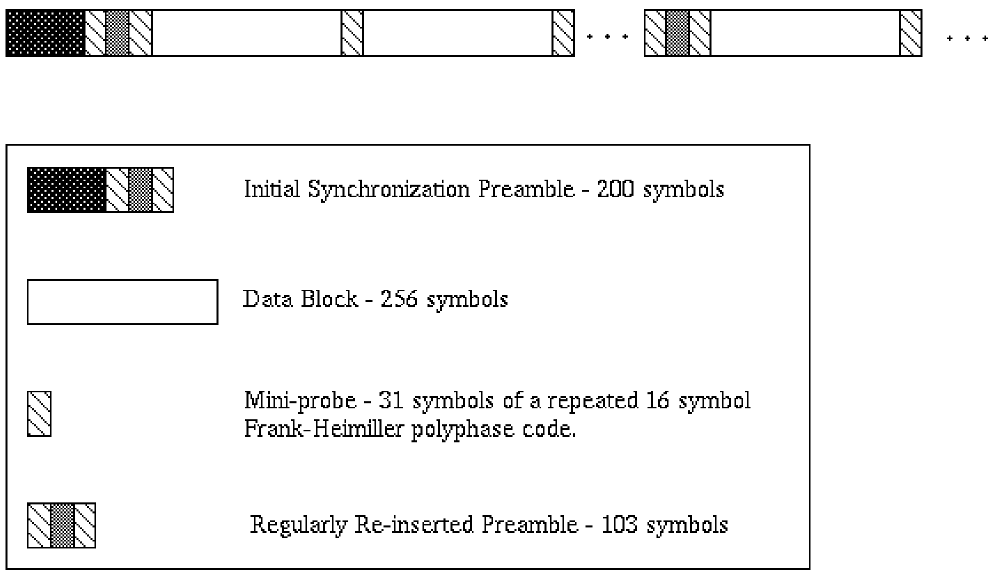

. . .
. . .
Initial Synchronization Preamble - 200 symbols
Data Block - 256 symbols
Mini-probe - 31 symbols of a repeated 16 symbol Frank-Heimiller polyphase code.
Regularly Re-inserted Preamble - 103 symbols

Figure G-5: Frame structure for all waveforms.

G.4.0 Synchronization and Reinserted Preamble

G.4.1 Synchronization Preamble

The synchronization preamble consists of 200 symbols. The first 97 symbols are intended exclusively for synchronization and Doppler offset removal purposes while the final 103 symbols, which are common with the reinserted preamble, also carry information regarding the interleaver setting.

Expressed as a sequence of 8PSK symbols, using the symbol numbers given in Table G-1, the synchronization preamble is:

1, 3, 3, 7, 0, 3, 6, 1, 2, 0, 5, 0, 7, 0, 0, 3, 3, 6, 7, 3, 1, 2, 0, 3, 2, 3, 3, 1, 1, 0, 4, 7, 5, 4, 6, 5, 4, 0, 6, 2, 1, 1, 6, 1, 3, 7, 2, 3, 3, 1, 6, 0, 2, 0, 6, 4, 7, 4, 0, 5, 7, 6, 6, 1, 1, 4, 6, 5, 5, 4, 1, 0, 7, 2, 3, 7, 2, 3, 6, 3, 1, 2, 3, 1, 0, 1, 0, 4, 1, 3, 1, 6, 6, 5, 6, 1, 4,

0, 0, 0, 0, 0, 2, 4, 6, 0, 4, 0, 4, 0, 6, 4, 2, 0, 0, 0, 0, 0, 2, 4, 6, 0, 4, 0, 4, 0, 6, 4,

D_0,D_1,D_2,6, 4, 5, 6, 1, 2, 5, 6, 4, 7, 3, 1, 0, 6, 1, 7, D_0,D_1,D_2,
2, 6, 6, 1, 2, 0, 2, 7, 4, 4, 2, 6, 4, 3, 1, 0,D_0,D_1,D_2,

4, 4, 4, 4, 4, 6, 0, 2, 4, 0, 4, 0, 4, 2, 0, 6, 4, 4, 4, 4, 4, 6, 0, 2, 4, 0, 4, 0, 4, 2, 0;

where the data symbols D_0, D_1, and D_2 take one of 3 sets of values chosen from Table G-5 to indicate the interleaver setting.

Table G-5.

Interleaver Setting	**D_0, D_1, D_2 (8 PSK symbols)**
No (0 s)	4,0,0
Short (0.72 s)	2,4,2
Long (10.8 s)	6,2,4

G.4.2 Reinserted Preamble

The reinserted preamble takes the same form as the final 72 symbols of the synchronization preamble. In fact, the final 103 symbols are common between the synchronization preamble and the contiguous block consisting of the reinserted preamble and the mini-probe which immediately precedes it. The 103 symbols of known data are thus:

0, 0, 0, 0, 0, 2, 4, 6, 0, 4, 0, 4, 0, 6, 4, 2, 0, 0, 0, 0, 0, 2, 4, 6, 0, 4, 0, 4, 0, 6, 4,

D_0,D_1,D_2,6, 4, 5, 6, 1, 2, 5, 6, 4, 7, 3, 1, 0, 6, 1, 7, D_0,D_1,D_2,
2, 6, 6, 1, 2, 0, 2, 7, 4, 4, 2, 6, 4, 3, 1, 0,D_0,D_1,D_2,

4, 4, 4, 4, 4, 6, 0, 2, 4, 0, 4, 0, 4, 2, 0, 6, 4, 4, 4, 4, 4, 6, 0, 2, 4, 0, 4, 0, 4, 2, 0;

where the data symbols D_0, D_1, and D_2 again take one of 3 sets of values chosen from Table G-5 to indicate the interleaver setting.

G.4.3 Mini-Probes

Mini-probes 31 symbols in length will be inserted following every 256 symbol data block and at the end of each preamble (where they are considered to be part of the preamble). For convenience, each mini-probe will be sequentially numbered, with mini-probe number 1 following the first data block after a preamble.

Using the 8PSK symbol mapping, each mini-probe shall be based on the repeated Frank-Heimiller sequence. One such sequence, specified in terms of the 8PSK symbol numbers, is given by

0, 0, 0, 0, 0, 2, 4, 6, 0, 4, 0, 4, 0, 6, 4, 2, 0, 0, 0, 0, 0, 2, 4, 6, 0, 4, 0, 4, 0, 6, 4.

This mini-probe will be designated '+'.

The phase inverted version of this is:

4, 4, 4, 4, 4, 6, 0, 2, 4, 0, 4, 0, 4, 2, 0, 6, 4, 4, 4, 4, 4, 6, 0, 2, 4, 0, 4, 0, 4, 2, 0

and mini-probes using this sequence will be designated '-', as the phase of each symbol has been rotated 180 degrees from the '+'.

If the no (or 0 s.) interleaver setting is selected, all mini-probes will employ the '+' sequence above.

Pictorally, this is: - | + + + + + + + + + ... + + |

where vertical bars have been used to indicate preamble boundaries. The initial '-' indicates the negative phase Frank sequence which forms the last 31 symbols of all preambles.

If short interleaving (0.72 s.) is selected, the phase inverted sequence will be sent for 3 consecutive mini-probe segments out of every group of 6 mini-probe segments beginning with the first 3 mini-probes. Thus mini-probes 1,2, and 3 are phase inverted, 4,5 and 6 are reference non-phase inverted sequences, 7,8 and 9 are phase inverted and so on.

Pictorally: - | - - - + + + - - - + + + - - - ... - - - + + + |

If long interleaving is selected, the phase inverted sequence will be sent for every mini-probe whose sequence number is not evenly divisible by 3. For example, mini-probes 1, 2, 4 and 5 would be phase inverted in this way, while mini-probes 3 and 6 would employ the reference non-phase inverted phase sequence.

Pictorally: - | - - + - - + - - + - - + ... - - + |

G.5.0 Coding and Interleaving

G.5.1 Rate 1/2 Convolutional Codec

The constraint length 7, rate 1/2 convolutional codec employed is the same as is used in Annex E of STANAG 4285. Figure G-6 is a pictoral representation of the encoder. The two generator polynomials used are

$$T_1 = x^6 + x^4 + x^3 + x + 1$$

$$T_2 = x^6 + x^5 + x^4 + x^3 + 1$$

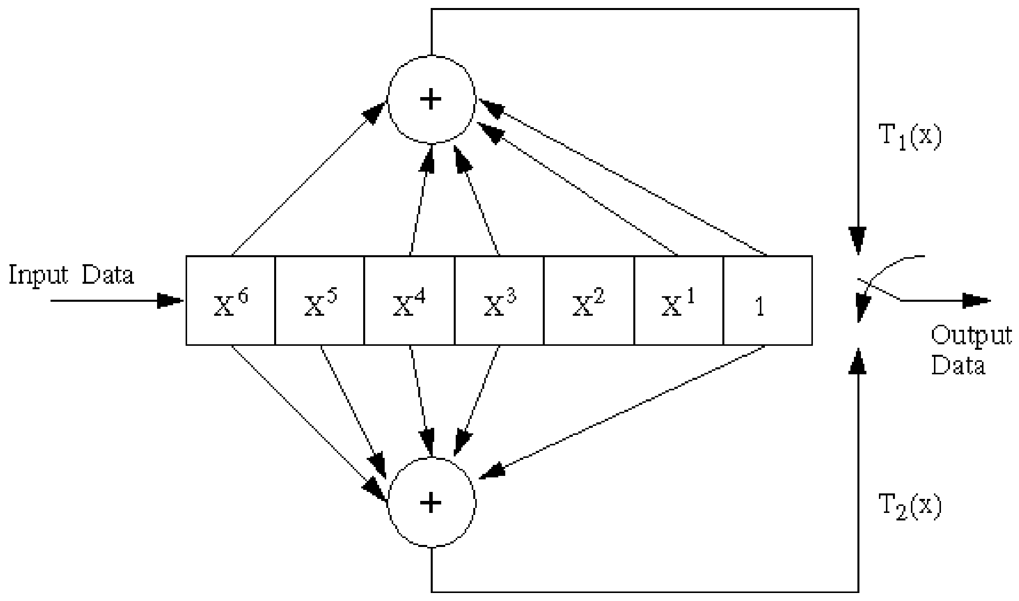


Figure G-6: Constraint length 7, rate 1/2 convolutional encoder.

The two summing nodes in the figure represent modulo 2 addition. For each bit input to the encoder, two bits are taken from the encoder, with the upper output bit, $T_1(x)$, taken first. Coded bit streams of 19200 bps, 12800 bps, 9600 bps and 6400 bps are generated for the input data rates of 9600 bps, 6400 bps, 4800 bps and 3200 bps respectively.

Puncturing of the rate 1/2 code to the required rate 3/4 is done at the output of the interleaver, or in place of the interleaver when the no interleaver option is selected.

G.5.2 Puncturing to Rate 3/4 with No Interleaver

In order to obtain a rate 3/4 code from the rate 1/2 code used, the output of the encoder must be punctured by not transmitting 1 bit out of every 3. A good puncturing is obtained by using a puncturing mask of 1 1 1 0 0 1, applied to the bits output from the encoder. In this notation a 1 indicates that the bit is retained and a 0 indicates that the bit is not transmitted. For a generated sequence of

$T_1(k), T_2(k), T_1(k+1), T_2(k+1), T_1(k+2), T_2(k+2) \ldots$

the transmitted sequence would be

$T_1(k), T_2(k), T_1(k+1), T_2(k+2) \ldots$

The data stream is punctured in this manner when the no-interleaver setting is selected; otherwise the puncturing takes place at the output of the interleaver.

G.5.3 Convolutional Interleaver Structure

The convolutional interleaver used is a modification of that presented in Annex E of STANAG 4285. A conceptual representation of a convolutional interleaver and deinterleaver is shown in Figure G-7. In order to obtain the desired puncturing to rate 3/4, a nominal 48 row interleaver is employed.

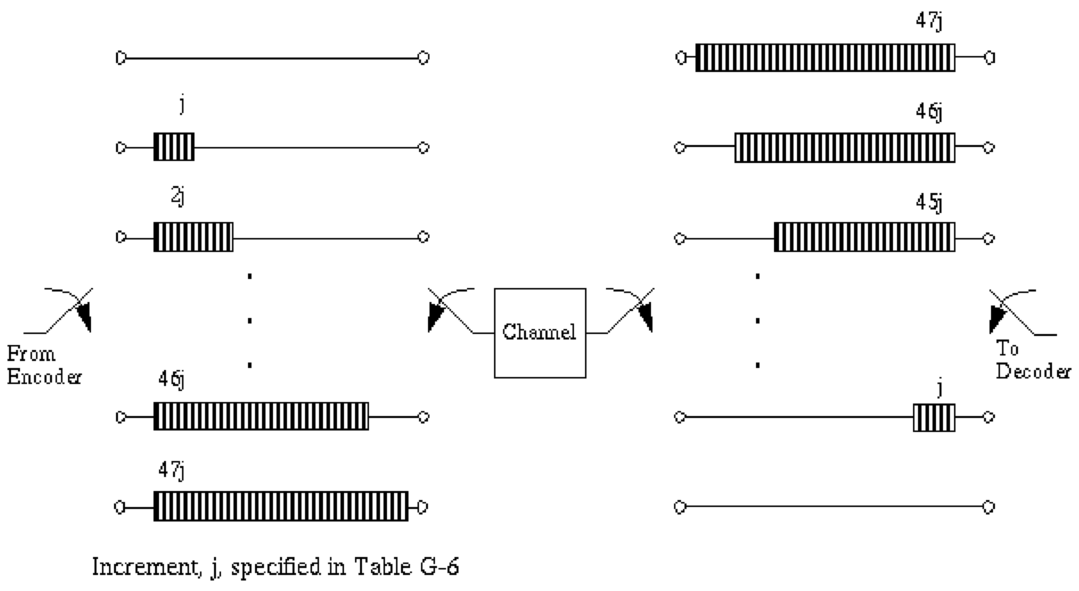


Figure G-7: Convolutional interleaver structure.

For a normal implementation, coded bits are shifted into interleaver shift-registers (rows) on the left side of the diagram. On each new bit, the commutator switches into the next lower row (shift-register). Each shift-register has j more bits of storage than the preceding one above it. Bits are extracted through the output commutator in a similar fashion and transmitted over the channel. In the receiver, deinterleaving is performed by a similar operation, however, each shift-register in the deinterleaver has j fewer bits of storage than the preceding one above it. This normal interleaving technique is modified by making the commutators at the input of the interleaver and the output of the deinterleaver cycle through all of the rows in a nonsequential pattern as specified below. To achieve the rate
3/4 code rate required using a good puncturing, 4 bits are taken out of the interleaver for every 6 bits entered using a puncturing mask of 1 1 1 0 0 1.

The proper timing relationship between input and output bits with these modifications of the row-sequencing is maintained if all rows of the shift-registers are shifted simultaneously after each complete cycle of the commutators. The specific parameters of the convolutional interleaver used are as indicated below.

The number of rows is nominally 48 for all data rates, although with the puncturing to rate 3/4 used, only 32 of these need be maintained.

Table G-6. Delay increment ‘j’ for each successive row.

Data Rate	Short (0.72 s)	Long (10.8 s)
3200 bps	2	30
4800 bps	3	45
6400 bps	4	60
9600 bps	6	90

If the rate 1/2 code was used without puncturing, the commutator row sequence for outputting bits from the interleaver would simply involve taking 1 bit from each row in numerical sequence, returning to row zero after a bit from row 47 was output. With the selected puncturing to rate 3/4, the commutator row sequence for outputting bits from the interleaver is:

0, 1, 2, 5, 6, 7, 8, 11, 12, 13, 14, 17, 18, 19, 20, 23, 24, 25, 26, 29, 30, 31, 32, 35, 36, 37, 38, 41, 42, 43, 44, 47

The sequence above is repeated every 32 output bits.

3 cycles through the interleaver produces

48 QPSK symbols or
32 8PSK symbols or
24 16QAM symbols or
16 64QAM symbols.

The commutator row sequence for inputting bits to the interleaver is:

0, 19, 38, 9, 28, 47, 18, 37, 8, 27, 46, 17, 36, 7, 26, 45, 16, 35, 6, 25, 44, 15, 34, 5, 24, 43, 14, 33, 4, 23, 42, 13, 32, 3, 22, 41, 12, 31, 2, 21, 40, 11, 30, 1, 20, 39, 10, 29

where the sequence has been generated by using the modulo 48 result of the multiplication of each number in the traditional (increment by 1) sequence by 19.

The interleaver and deinterleaver are synchronized when the two center commutators shown in Figure G-7 are synchronized, i.e. a bit taken from the ith row of the interleaver is sent to the ith row of the deinterleaver. Synchronization shall be maintained by making the top (0th) row always be the starting position for the 1st bit sent or received in the first data block following a synchronization or reinserted preamble and, consequently, every data block thereafter for 8PSK and 64QAM modulations and every 3rd data block thereafter for QPSK and 16QAM modulations. Interleaver synchronization can thus be established at the beginning of every third data block for all modulations. In a sky-wave channel, it is likely that interleaver synchronization will only be possible by acquiring one of the preambles, either the initial synchronization preamble, or, if this is missed, one of the reinserted preambles which occur every 8.64 s. In a surface wave channel, it should be possible to use the mini-probes for signal acquisition and interleaver identification and synchronization.

G.5.4 Initialization and Message Protocols

The encoder shift-register and interleaver shift-registers should be set to all zeroes before the start of any message.

A unique 32 bit start-of-message (SOM) pattern is inserted into the bit stream sent to the encoder before the first bit of a message. The SOM, expressed as in hexadecimal notation is 03873C3C, where the left most bit is sent first.

A unique 32 bit end-of-message (EOM) pattern is appended after the last bit of the message. The EOM, expressed in hexadecimal notation is 4B65A5B2, where the left most bit is sent first. When the interleaver is used, a string of flush zeroes, equal in length to the interleaver delay plus 102 is appended to the EOM in the bit stream. These zeroes are used to flush the interleaver, coder and allow for the maximum practical traceback delay in the receiver’s decoder. Note that the encoder and interleaver will be filled with all zeroes, ready for the next message.

When no interleaver is employed, flush zeroes are used to guarantee that the data block containing the EOM and the first 102 flush zeroes will be completely filled. Again, the flush zeroes are appended to the EOM in the bit stream.

Only after the last of the flush zeroes is input to the encoder is the transmission terminated.

<SOM > < Message Bits > < EOM > < Flush Zeroes >

The number of flush zeroes for each of the data rates and interleaver settings are specified in the table below.

Table G-7. Flush zeroes required for specified rates and interleaver settings.

Data Rate	**Interleaver Delay**		
	None (0 s)	**Short (0.72 s)**	**Long (10.8 s)**
3200	784	2406	34662
4800	1125	3558	51942
6400	1467	4710	69222
9600	2149	7014	103782

G.6.0 Associated Communications Equipment

Transmitters and receivers satisfying the specifications set out in Appendix 1 of Annex A to STANAG 4285 will perform satisfactorily when used with modems implementing the waveforms in this Annex.

Annex H. Implementation Guide and Notes (information only)

This Annex contains rules and guidelines for adaptive speed control and other implementation issues based on our experience with earlier systems. The best and most complete source of information on implementation topics is NC3A TM-937 "Open Systems for Radio Communications: A Subnet Architecture for Data Transmission over HF Radio".

H.1 Flow Control

Flow control imposed by a client on the Subnetwork could cause the receiving queues of the HF Node to be filled, which in turn could cause older queued data for other clients to be discarded or result in a temporary pause in accepting and acknowledging error free PDUs. This situation is not acceptable since a client, of even a low Rank, could, in principle, cause a deterioration of the service provided to the other clients connected to the Node.

H.2 Reasons for Data Transfer D_PDUs (D_PDUs) with Different Rules

Annex C to this document defines a number of different D_PDUs which may be used to transfer data: normal data transfer (D_PDU types 0-3), expedited data (D_PDU types 4 and 5), management message (D_PDU type 6), and non-ARQ (types 7, 8, and 15). This section reviews the use of the different D_PDUs and the reasons for having them.

H.2.1 Normal Data Transfer

The normal data transfer D_PDUs are intended to be used in most cases for the transfer of U_PDUs. It is difficult to efficiently handle high priority data in the context of this type of D_PDU alone. When a PDU from a higher layer reaches the data transfer sublayer, it is segmented into a number of D_PDUs which are assigned sequence numbers. While it may be possible to "unqueue" or cancel parts of higher level PDUs, this will either result in the loss of data (cancellation) or delays in transmitting the high priority traffic. Thus, the additional types of D_PDUs have been introduced to efficiently accommodate high-priority traffic.

H.2.2 Expedited Data

Expedited data D_PDUs are intended for use to support peer-to-peer communications (primarily making and breaking physical links), and exceptionally, to provide a path for U_PDUs of the highest priority which bypasses all existing queues. An example of the first would be if it is desired to immediately break a link which has long queues of traffic pending. If this "break" C_PDU was handled as a normal data PDU, it might (depending on the system implementation) have fairly long delays (minutes) before it could be transmitted. With the expedited data mechanism, the C_PDU bypasses all the normal data queues and is transmitted at the beginning of the next transmission interval.

Expedited data bypasses all existing normal data queues (which are maintained during the handling of the expedited data). A number of expedited data D_PDUs (corresponding to, and not more than, a single expedited data C_PDU) may be transmitted in a single transmission. A stop and wait protocol is used for the acknowledgement of (the group of) expedited data D_PDUs. This service is intended for occasional use to transmit small amounts of data. Frequent use, or use with large U_PDUs, will degrade system performance.

H.2.3 Management

A third type of service is provided exclusively for system management functions, for example, to co-ordinate the adaptive change of the HF modem data rate. This function, and others like it, requires a service with the smallest possible delay and with maximum robustness. This is provided by the management data D_PDU. Management D_PDUs bypass all pending data D_PDUs (when the system enters the management mode, all data queues are put on hold).

Transmission of this D_PDU type follows a D_PDU-by-D_PDU stop-and-wait protocol, with D_PDUs repeated as necessary to fill the HF modem interleave buffers and provide maximum robustness and efficiency. Only a single type 7 D_PDU D_PDU may be outstanding (unacknowledged) at any moment; the D_PDU is repeated until acknowledged (or the link fails due to a time out).

H.2.4 Non-ARQ

Non-ARQ, or unacknowledged, D_PDU types are provided in order to allow the transfer of data which is not acknowledged. This is useful for “broadcast” modes of operation, in which nodes are not allowed to transmit due to EMCON restrictions, and to support functions which occur when the node is not in an ARQ state, such as establishing a connection.

H.3 Other topics

EOT definition gives a maximum transmission interval of 127.5 seconds, or just over two minutes. EOT calculation must be rounded up to avoid collisions.

Because the D_PDU header is generally shorter than the data, errors are more likely in the data part of a D_PDU than the header. Protecting the header with its own CRC allows the possibility to detect and use uncorrupted header information even if the data part of a D_PDU contains errors.

If a D_PDU is to be sent with the DROP PDU bit set, it is inefficient to send the data portion of the D_PDU. However, all D_PDUs which make up the PDU to be dropped must be sent to maintain window synchronisation. A D_PDU that is received with the DROP PDU flag set must still be acknowledged.

The 10 bit field SIZE OF USER DATA indicates the size of the information field in bytes. This figure does not include the data CRC. 2^{10}=1023 bytes=8184 bits=109 seconds at 75 bps

To allow the Channel Access Sublayer to effectively monitor the channel activity, the Data Transfer Sublayer should deliver all identifiable *D_PDUs* (i.e D_PDUs for which the header portion of the D_PDU was received without errors) which are *not addressed* to the receiving node to the Channel Access Sublayer. The Channel Access Sublayer filters the received C_PDUs according to address and type and, as appropriate, locally processes the C_PDU or passes the C_PDU to the Subnet Interface Sublayer.

In the non-reliable mode of operation of the Data Transfer Sublayer, all identifiable *C_PDUs* (i.e. C_PDUs in which the header portion of one or more associated D_PDU’s) was received without error are delivered to the Channel Access Sublayer. This procedure was adopted so that: i) the Channel Access Sublayer could effectively monitor the channel activity, and ii) a client could specify that only error free or, alternatively, that all identified S_PDUs be delivered. This latter option may be useful if the client handles printable text and/or implements additional error control functionality.

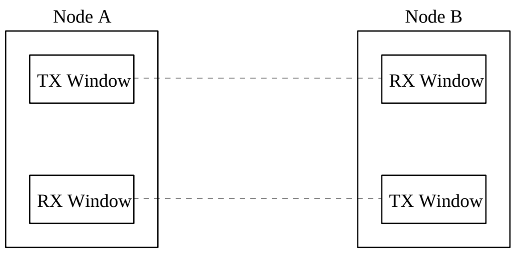
H.4 Synchronisation of the ARQ Machine

H.4.1 Initial Synchronisation

Establishment of a “connection” requires the initial synchronisation of the peer protocol ARQ machines in the Data Transfer Sublayer. This process can be viewed as automatic in the sense that the peer ARQ machines associated with a new connection (i.e. one for which an ARQ machine must be established) will be automatically reset and the ARQ machines associated with a revived data state connection (i.e. one for which an existing ARQ machine is re-activated) is assumed to be still valid. Because this assumption may, for various reasons, *not* be valid, a synchronisation verification procedure should be executed whenever a link is made.[12]

H.4.2 Verification and Maintenance of Synchronisation

In addition to a synchronisation verification procedure which may be executed following (re)establishment of a link, a mechanism is available to verify on an *ongoing basis* if the peer ARQ processes are in synchronisation and, if required, to effect a reset or re-synchronisation of the peer ARQ window pointers.[13]



This figure illustrates the fact that, at each end of the node, there is one transmit and one receive window. The dotted lines show which pairs of windows need to remain in sync.

Synchronisation Tests Performed at the Destination Node

Synchronisation tests performed at the destination node make use of the TX lower window edge (LWE) and TX upper window edge (UWE) flags in conjunction with the TX FRAME SEQ # contained in the header of DATA-ONLY and DATA-ACK frames. The appropriate flag is raised (value = 1) when the TX FRAME SEQ # corresponds to the originating node’s transmit ARQ LWE or UWE pointers. A *loss of synchronisation* is detected under one or more of the following conditions (mod 256):

[12] It has been our experience that synchronisation between peer ARQ machines is frequently lost due to factors not related to propagation, e.g. interruption of primary power.

[13] Verification of synchronisation on an ongoing basis and, if required, re-synchronisation is the responsibility of the Data Transfer Sublayer. However, under some circumstances a reset or re-synchronisation may be initiated by the Management Sublayer, e.g. following the (re)establishment of a link and as part of some link maintenance procedures.

TX UWE POINTER < RX UWE POINTER (1a)

or

(TX UWE - RX LWE) > 127 (1b)

Equation (1a) is TRUE when a frame is acknowledged as correctly received before it is transmitted; equation (1b) is TRUE when the apparent size (i.e. as seen from the perspective of the destination node) of the originating node's transmit ARQ window exceeds 127. These relationships are graphically illustrated in Figure H-1.

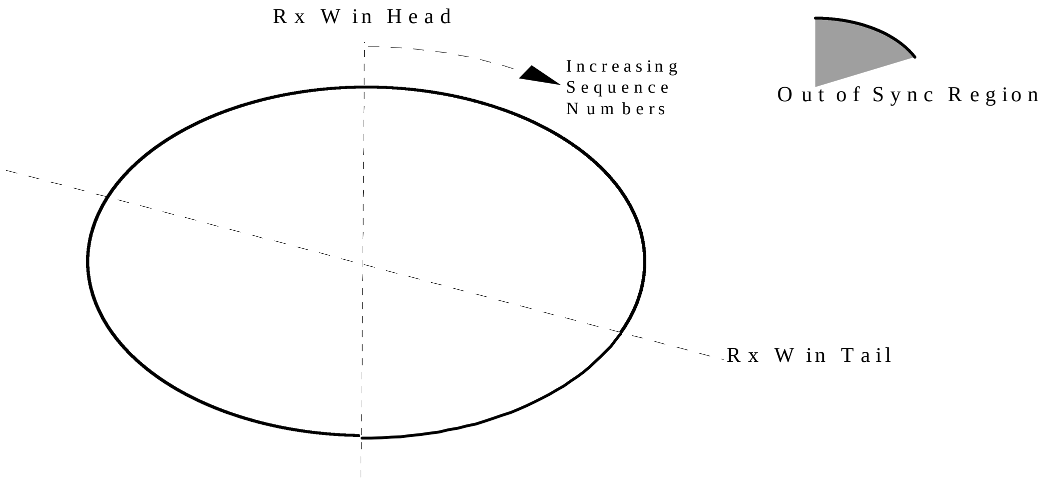


Figure H-1. Showing IN-SYNC and OUT-OF-SYNC Regions of the FSN Circle (Equation 1)

A second set of tests that may be performed at a destination node are given below (mod 256):

TX LWE POINTER > RX LWE POINTER (2a)

or

(RX UWE - TX LWE) > 127 (2b)

These relationships indicate that the peer ARQ machines are not properly synchronised, when (2a) the originating node has advanced its transmit LWE pointer when the destination node has not yet acknowledged those frames and (2b) when the apparent size (i.e. as seen from the perspective of the destination node) of the originating Node's transmit ARQ window exceeds 127. These relationships are graphically illustrate in Figure H-2.

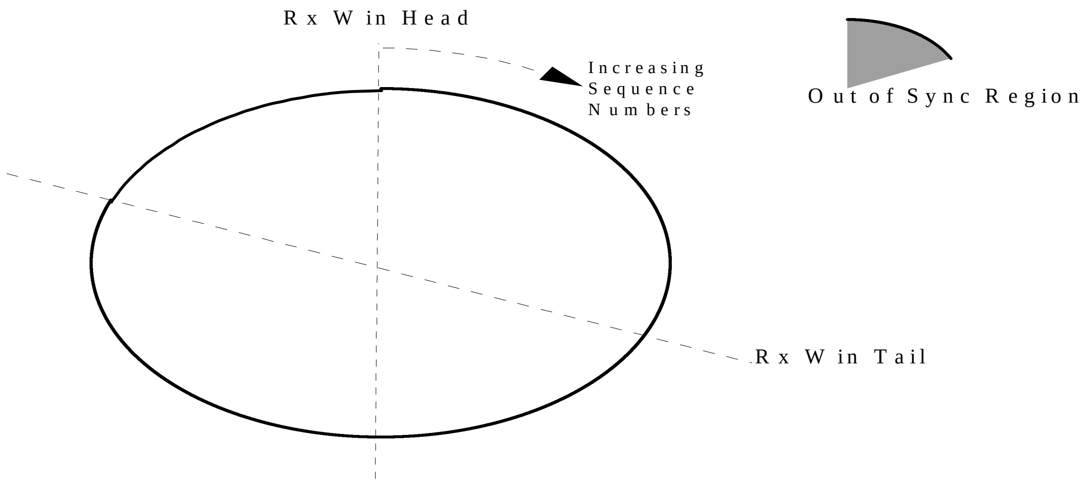


Figure H-2. Showing IN-SYNC and OUT-OF-SYNC Regions of the FSN Circle (Equation 2)

A third set of tests that may be performed at a receiving node are given below (mod 256):

$$(\text{TX FSN} - \text{RX LWE}) > 127 \quad (3a)$$

and

$$(\text{RX UWE} - \text{TX FSN}) > 127 \quad (3b)$$

Equation (3a) is a generalisation of equation (1b) while equation (3b) is a generalisation of equation (2b). These relationships are graphically illustrated in Figure H-3.

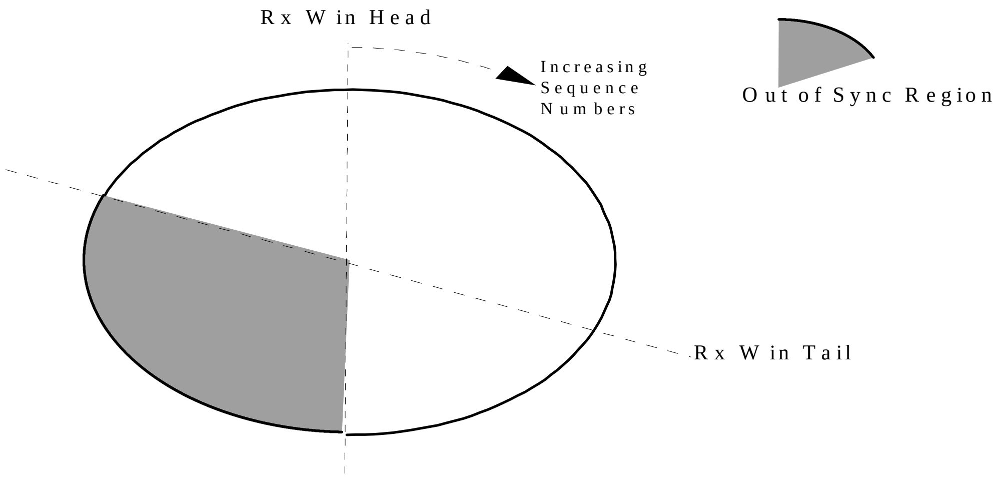


Figure H-3. Showing IN-SYNC and OUT-OF-SYNC Regions of the FSN Circle (Equation 3)

Synchronisation Tests Performed at the Originating Node

A single set of tests can performed at the originating node to detect an out-of-synchronisation condition. These tests can be considered "loop around" tests in that the originating Node compares the FSNs contained in received ACK or ACK-ONLY frames with its transmit ARQ pointers. These tests apply to any received FSN, i.e. whether a group ACK (RX WIN LWE pointer) or derived from the selective ACK header field. A loss of synchronisation is detected under one or more of the following conditions (mod 256):

$$\text{Acknowledged FSN} < \text{TX LWE} \qquad (4a)$$

or

$$\text{Acknowledged FSN} > \text{TX UWE} \qquad (4b)$$

Test (4a) indicates a loss of synchronisation if a frame has been recorded as having been acknowledged at the transmitting Node which the receiving Node indicates has *not* been positively acknowledged. Test (4b) indicates a loss of synchronisation when the receiving Node indicates an acknowledgement of a frame that the receiving Node has not yet transmitted. These relationships are graphically illustrate in Figure H-4.

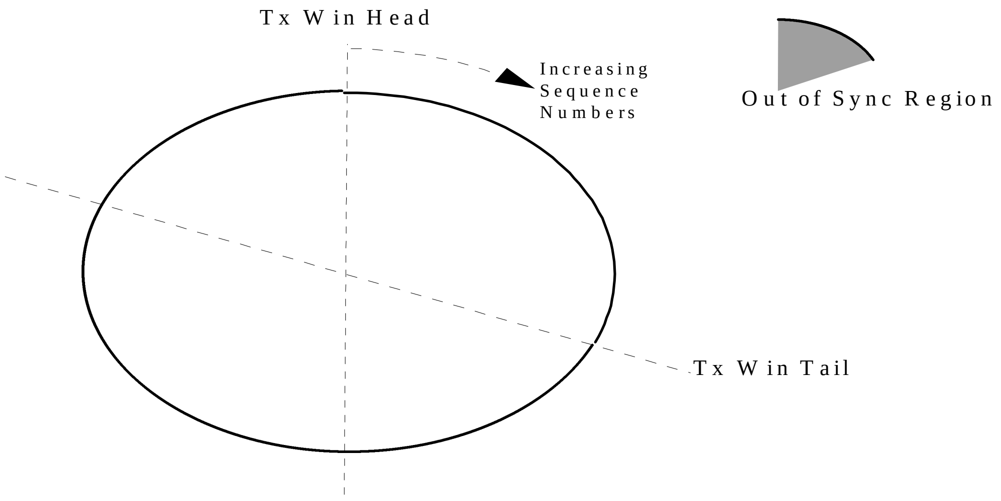


Figure H-4. Showing IN-SYNC and OUT-OF-SYNC Regions of the FSN Circle (Equation 4)

Re-synchronisation Procedures Initiated at the Destination Node

If a loss of synchronisation between the peer ARQ processes is detected, several possibilities exist for re-synchronisation. The destination node can independently re-synchronise its receive LWE and UWE pointers to the indicated TX FRAME SEQ # *UWE* pointer of the originating node. This is equivalent to signalling a group ACK to all frames transmitted by the originating node. Alternatively, the destination node can initiate a re-synchronisation or reset procedure with the peer node. All options result in some loss of data although, in general, a negotiated re-synchronisation results in the loss of a smaller number of frames. The re-synchronisation options available using a handshaking procedure are described below.

The simplest but most drastic option is a reset of the ARQ window pointers. This is initiated by sending a RESET/WIN-RESYNC frame with the FULL RESET CMND flag set (value = 1). The requesting node assigns a **RESET FRAME ID #** which is derived from a sequence of frame ID numbers (modulo 255) that is reserved (independently by each node) for reset/re-synchronisation operations.[14] A new RESET FRAME ID # is assigned for each **reset event**. A reset event is defined as one or more frames, which may span more than one transmission interval, sent by a node to accomplish a given reset operation, e.g. initiate and receive an acknowledgement for a full reset. Thus, for example, if a node transmits multiple (i.e. redundant) RESET/WIN-RESYNC frames in the same or over several transmission intervals, each RESET/WIN-RESYNC frame is assigned the same RESET FRAME ID # and is considered part of the same reset event. The node that received the reset request replies with a RESET/WIN-RESYNC frame with the RESET ACK flag set (value = 1) and the other three reset/resynchronisation flags reset (value = 0). The RESET FRAME ID # field is not given a value (i.e. don't care condition as indicated in Fig. 21). The effect of the FULL RESET CMND is to reset (zeroise) the Tx *and* Rx window pointers and buffers in the *currently active* peer DATA state protocol machines, i.e. at both ends of the link. Because a reset effects the transmit and receive buffers at both ends of a link, it should be used only when necessary or when there will be no loss of data, e.g. before beginning data transmission over a newly established link.

A node can also request that its peer re-adjust its transmit ARQ buffer pointers by sending a RESET/WIN-RESYNC frame with the RESET TX WIN RQST flag set (value = 1). The requesting node supplies a RESET FRAME ID # as previously described from its queue of ID numbers reserved for the transmission of RESET/WIN-RESYNC frames. The peer node responds with a RESET/WIN-RESYNC frame with the RESET ACK and RESET RX WIN CMND flags set (value = 1). The node also assigns a value to the NEW RX WINDOW LWE # and the RESET FRAME ID #. The NEW RX WINDOW LWE # is used by the node that *requested* the re-synchronisation event to re-sync its receive ARQ LWE pointer to the same value as the peer node's newly adjusted transmit ARQ LWE pointer.[15] The requesting node should also set its UWE pointer to the newly adjusted receive ARQ LWE pointer. Finally, the requesting node must respond with a RESET/WIN-RESYNC frame with the ACK flag set (value = 1) and all other flags cleared. The requesting node can initiate the resynchronisation procedure without first waiting for DATA-ONLY and DATA-ACK frames which have already been assigned sequence numbers to be transmitted and acknowledged, i.e. the RESET/WIN-RESYNC frames can be added to an ongoing transmission interval. The only restriction is that, for half duplex operation, the transmission interval should not exceed the maximum value that appears in the frame header of any frames that have already been sent.

Re-synchronisation Procedures Initiated at the Originating Node

[14] The term ID number is used to avoid confusion with the concept of a frame sequence number. RESET/WIN-RESYNC frames do not carry a frame sequence number.

[15] The selection of the new transmit ARQ window LWE pointer location by the node which *received* the resynchronisation *request* is not an interoperability issue and can be done in several ways. For example, the pointer could be advanced (in the direction of the node's transmit ARQ window UWE pointer) until there are no positively ACK'd frames with sequence numbers greater than the new position of the LWE pointer. This procedure is suitable for an implementation in which frames which have been acknowledged as being correctly received are immediately deleted from the transmit queue. Alternatively, if frames with sequence numbers between the transmit LWE and UWE pointers are retained, the node receiving the reset request can set the NEW RX WINDOW LWE # to the current value of the transmit LWE pointer. This is the most "efficient" option in that the least number of frames are lost. Finally, the responding node can simply set its transmit LWE pointer (and the NEW RX WINDOW LWE # field) to the current value of its transmit UWE pointer. This is the least "efficient" option.

The originating Node can request that its peer re-adjust its receive ARQ buffer pointers by sending a RESET/WIN-RESYNC frame with a value assigned to the RESET ID #, the RESET RX WIN CMD flag set (value = 1) and the NEW RX WINDOW LWE # equal to the desired value of the originating Node's transmit LWE pointer. This causes the peer Node to set its receive LWE *and* UWE pointers to the value of the originating node's transmit LWE pointer. The originating Node, after having received an acknowledgement (i.e. a REST/WIN-RESYNC frame with the RESET ACK flag set and the other three reset/resynchronisation flags cleared) can proceed with retransmission of user data.

Summary

Tests (1) - (3) are considered the primary node synchronisation tests in that they i) apply to all DATA state frame types (Tests (1) and (2) provide tighter bounds than Test (3) but can only be applied when the LWE and/or UWE flag is set in the frame header) ii) can be applied independently by each node and iii) do not depend on the state of the reverse channel in the case of ARQ data. In addition, by performing synchronisation tests at the receiving node it is likely that the loss of synchronisation will be detected and corrected in a shorter time period than if synchronisation tests are applied only at the originating Node.

The reset/resynchronisation procedures discussed above apply only to the DATA state of the ARQ protocol machine (the EXPEDITED DATA and MAN DATA states utilise a different set of procedures that ensure synchronisation of the peer ARQ machines). These tests and the associated resynchronisation procedures can be applied by the Data Transfer Sublayer on an "as required basis" without the intervention of the Management Sublayer. Finally, note that the transfer of data is automatically resumed by the Data Transfer Protocol following completion of the Reset/Resynchronisation exchange.

H.5 Use of Rank and Priority Arguments (Subnet Interface Sublayer)

The rank of subnet clients is used to determine the allocation of subnet resources. As an example, if a new client attempts to come on-line (bind to a node) but not enough resources are available, the Subnetwork Interface Sublayer is permitted to unilaterally declare a client with lower rank off-line in order to release resources for the higher-ranked client.

Rank is also used to determine management privileges. The node shall not accept command-type S_MANAGEMENT_MSG_REQUEST primitives from a client with rank less than 15. The node shall accept request-type S_MANAGEMENT_MSG_REQUEST primitives (which cannot change the configuration of the node or subnetwork) from any client.

Priority can take a value in the range 0-15. The node "does its best" to service high priority U_PDUs before lower priority U_PDUs which are queued in the system. This means that the node is not required to *guarantee* that the higher priority U_PDUs will overtake all queued lower priority U_PDUs (depending on the implementation of the node, it may not be possible for a higher priority U_PDU to overtake a queued lower priority U_PDU which has entered an advanced stage of processing).

Client rank and the priority of data submitted by that client are not necessarily dependent.

H.6 **Implementation notes for DRC Procedure**

Although many 4285 modems are implemented so that the transmit and receive data rates must be the same at any instant, this is not an absolute limitation. Some modems may be implemented so that they can use different transmit and receive data rates. The system, including the modem, may be implemented in such a way as to circumvent the problem if it exists, ie, fast remote control of the modem (in a half-duplex system), or multiple modems (in a half or full duplex system). However, the DRC procedures defined in Annex C will support the case in which the system is limited by this constraint.

The notes below refers to Figure C-16 in the DRC procedure section. The numbers here refer to the numbers in square brackets in the figure.

1. In this transmission, the advisory message sent by A carries recommended transmit speed for B (You should transmit at x bps) and also tells B about the capabilities of A's modem (that A's modem can operate at independent transmit and receive data rates. (The alternative message says that: If you change your transmit data rate (and my receive data rate), I will have to change my transmit data rate (and your receive rate) as well.)
2. The DRC_Request message from B carries the requested new modem parameters for B: "I (B) will transmit at x bps with y interleaving"; these parameters shall be as recommended by A. The message also tells A about the capabilities of B's modem.
3. DRC_Response message accepts the requested change.

If a node uses D_PDU error statistics to make data rate adaptation decisions, the decision to change data rate should be made only based on a transmission containing DATA-ONLY or DATA_ACK D_PDUs; or after a transmission made up of only ACK_ONLY D_PDUs is received with errors. These conditions are imposed because it is difficult to make reliable DRC decisions based on analysis of short signals. While these conditions are not required for interoperability, they are required for reliable system operation. Experience with algorithms based on counting D_PDUs with errors shows that short transmissions are usually either received completely, with no errors, or not received at all (this is caused by the steep BER vs E_b/N_o curves for the STANAG 4285 and MIL-110A modems). This can cause the data rate on an acknowledgement link, carrying only short transmissions, to increase steadily until, after one increase too many, the acknowledgements are no longer received. This is also the reason for transmitting ACK-ONLY D_PDUs multiple times when they are the only D_PDU type transmitted. This rule will generally lead to the ACK-ONLY D_PDUs being sent at a lower data rate, when data is flowing in (mainly) one direction on a link. In order to avoid the situation where the data transmissions are at 75 bps and the ACKs are at 2400 bps, the following rule should be followed: one end of the link shall transmit at not less than 1/8 the data rate of the other end.

If a node is only receiving acknowledgements, the transmit data rate for acks should be less than the transmit data rate for data but not less than 1/8 of the transmit data rate for data. The result of this rule will be that adaptive changes to data rate on one direction of the link will in some cases "pull" the data rate on the other direction of the link. This recommendation does not apply to a circuit which includes a shore answering frequency which must operate at a fixed data rate.

DRC Procedures With Different Waveforms

It is possible to have adaptive data rate from 75 bps through 9600 bps, using a combination of waveforms, for example STANAG 4285 for data rates of 2400 bps and below, and the Annex G waveform for the higher data rates. In this case, the DRC procedures would be required for 4285 but not for the Annex G waveform. In such a situation, the following rules should be observed:

- when the new and old data rates fall within the same waveform which does **not** support automatic recognition of the coding and interleave parameters, the Annex C DRC procedures shall be used. (Example: STANAG 4285, 600 bps, short interleave to 1200 bps, short interleave.)

- when the new and old data rates fall within the same waveform which **does** support automatic recognition of the coding and interleave parameters, the Annex C DRC procedures shall be used. (Example: STANAG 5066 Annex G, 4800 bps, short interleave to 6400 bps, short interleave.)
- when the new and old data rates **do not** fall within the same waveform, the Annex C DRC procedures shall be used. (Example: STANAG 5066 Annex G, 3200 bps, short interleave to STANAG 4285, 2400 bps, short interleave.) This is necessary in order to allow the modem to change waveforms, from 5066 Annex G to 4285.

H.7 Optimum D_PDU Size

A number of studies have examined the impact of the size of D_PDUs (or HF frames) on the throughput of the ARQ protocol. Perhaps the most relevant of these studies to performance in the context of STANAG 5066 has been a study which combined on-the-air trials with OPNET modelling, done by DRA Portsdown. Other studies have been done by STC and by Rohde and Schwartz.

The first work in this area was done by STC in 1992 and documented in "Laboratory and Field Tests of the High Frequency OSI Data Link Protocol", STC TN-506, August 1993. Some of the major findings in this report are[16]

1. the optimum frame size varies with the data rate and with channel conditions (fading as well as SNR)
2. throughput is not strongly sensitive to frame size
3. adaptive control of data rate with a compromise frame size of 200 bytes gives throughput very nearly identical to that realised with the "optimum" frame size for each data rate.

These results formed the basis of STC's, and later NC3A's, work with adaptive data rate HF ARQ protocols from 1992 until the present.

More recently, the DRA trials focused on the effect of varying error rate at 1200 bps. The DRA study has confirmed that throughput at a given BER is not strongly dependent on the frame size. At the higher BERs, there is a throughput maximum for frame sizes between 100 and 200 bytes. As the BER decreases, the frame size for maximum throughput also increases but very slowly, and the curve becomes even flatter.

The DRA throughput data are also valid for non-ARQ use of STANAG 5066 D_PDUs. The DRA data show the error free data bytes received, rather than the overall throughput; the small amount of overhead due to the short ARQ transmissions on a half-duplex link is not included. Thus, the 200 byte frame size is also a "good compromise" for use with non-ARQ 5066 transmission.

The DRA trials have confirmed (at least for 1200 bps)

1. the conclusion that the throughput is not strongly sensitive to frame size
2. the selection of 200 bytes as a good compromise frame size.

If there is a desire to vary frame size in some way, STANAG 5066 supports the use of variable frame sizes. While the data available to date suggest that the benefit may be

[16] One result from this report which is not relevant to the subject at hand but is more relevant to adaptive data rate control, is the very small improvement in throughput gained from switching from 1200 to 600 bps (figures 24 and 25).

marginal, it would be possible, for example, to associate a certain frame size with each data rate.

The variable frame size scheme mentioned in the preceding paragraph brings with it a number of implementation issues (as does any variable frame size). One of these issues is the fact that, when a data rate change is made, there will generally be some number of D_PDUs queued for transmission. It will, in general, be advisable to transmit these (at least up to the end of the next C_PDU) without changing the frame size and start using a new frame size when the next D_PDUs are created. This avoids some difficult synchronisation issues, and the data available suggest that the performance penalty will be negligible.

H.8 Application Note: Use of STANAG 5066 To Support Multiple Broadcasts on a Single HF Channel (frequency)

This application note describes how STANAG 5066 can provide a multichannel broadcast, and reviews some of the 5066 implementation issues to achieve a good capability. Some of the differences between this approach and the traditional, physically multiplexed (either FDMA or TDMA) multichannel approaches are described.

It is assumed in the discussion below that an HF node which is handling multiple broadcast clients will be dedicated to this task. It would seem in general inappropriate (although perhaps not inconceivable) to have this node abandon multiple broadcasts in order to do some other task.

Multi-user Subnetwork Client

In order to support multiple simultaneous clients attached to SAP ID 1 (broadcast message), a multi-user subnetwork client is required. This client will sit between a number of clients and the subnetwork interface and perform a number of functions, including multiplexing and segmentation. The arrangement is shown in figure H-x.

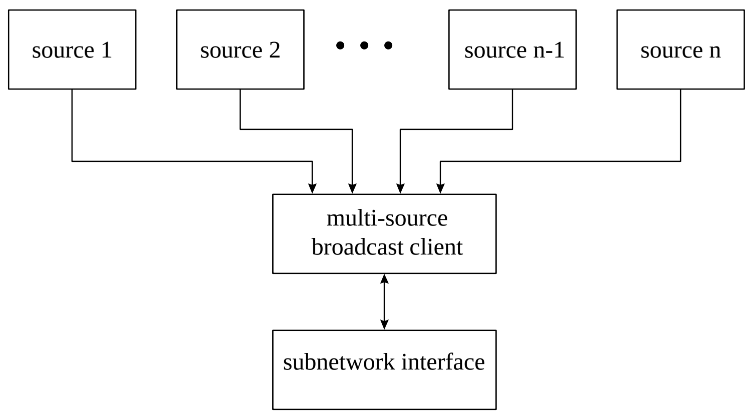


Figure H-x. Multi-source Broadcast (MSB) Client

Data will be accepted from the sources according to the interface spec of the source; the MSB client then forms a buffer between the standard interface to the 5066 subnetwork and the sources, which may be existing equipment.

Segmentation by the MSB client (or any other client) allows for more responsive and adaptive behaviour by the subnetwork. The client must segment large U_PDUs into 2 kb segments and submit these segments in the appropriate primitive to the subnet interface sublayer. The reasoning behind the requirement for the client to segment large U_PDUs will be explained later.

Insuring that each broadcast source gets appropriate access to the channel would seem to be a task appropriate to the channel access sublayer. However, since the MSB client is already responsible for queuing the data from the various sources, it knows the load that each source offers the subnetwork. The MSB client also knows the rank of the sources and the priority of the data. It therefore has all of the information required to allocate capacity to the various sources. Therefore, the MSB client should insure that U_PDUs or segments from multiple broadcast sources are passed down to the lower layers as is appropriate to

- the current (recent average) bandwidth of the source
- the rank of source
- the priority of the data from the source

Subnetwork Interface Sublayer

At the receiving node, if a node is in more than one broadcast group, U_PDUs may (will!) arrive for various broadcast groups (perhaps composed by different broadcast compilers) and be delivered to SAP ID 0. The client will be able to tell the arriving U_PDUs apart by the information in the S_UNIDATA_INDICATION primitives (sent to the client) which have the addressing information in them (S_PDUs do not). Exchanging this information between the Subnet Access and Data Transfer sublayers is an implementation issue (ie, the definition of the internal primitives) which is not appropriate for a STANAG. Detailed guidance is available in NC3A TM-937.

The subnetwork interface sublayer **does not segment** U_PDUs because reassembly is not supported in the S_PDU format.

Channel Access Sublayer

The channel access sublayer is simply a pipe for type 0 (data) S_PDUs, adding a few bits to convert them into type 0 (data) C_PDUs.

Data Transfer Sublayer

The Data Transfer Sublayer will segment the arriving C_PDUs into D_PDUs and queue them for transmission. Note that, as described in Annex H.5, the optimum size of 200 bytes for HF transmission also applies to broadcast. So a C_PDU of 2 kb will be segmented into 10 D_PDUs, and queued for transmission (at 300 bps, these D_PDUs will take about 1 minute to transmit). The next C_PDU to arrive will be likewise segmented and queued.

At this point the reason for limiting the size of U_PDUs, and thereby also S_PDUs, is clear. If a 100 kb S_PDU (inside a C_PDU) were to arrive at the DT sublayer for ARQ transmission, then the DT sublayer would segment it into some 500 D_PDUs and queue them for transmission. If the transmit speed is 300 bps, it will be about 45 minutes before any other client has access to the channel. Depending on the implementation of the DT sublayer, this situation may or may not also arise if a large PDU arrives at the DT sublayer for non-ARQ service.

Comparison to FDMA and TDMA multiplexing

The addressing structure adopted for non-ARQ (type 7) D_PDU allows some 268 million different group addresses. (This large address space will in all probability be reduced by an hierarchical addressing scheme.) There remains the possibility to have a unit (ship) belong to a number of broadcast groups depending on its operational role. This large address space, **combined with the ability to belong to more than one broadcast group**, offers the potential for significant efficiencies in broadcasts, because a close match between a group and the intended audience of a message is more likely. When a node can belong to only one group, one is forced to trade off between very large groups, with members receiving large amounts of unwanted messages; or smaller groups, with messages being retransmitted to a number of different groups.

This addressing scheme also allows a broadcast compiler to send a message to a single destination node, by simply using that node's individual address rather than a group address.

This scheme would require cryptographic equipment settings which differ from those for the current broadcast. Another possible approach would be to replace the broadcast crypto equipment with TADPOLE equipment. The result of changing to TADPOLE equipment for the broadcast would be that the **identical (receive) equipment string would be used to work either broadcast or ship-shore**, which is an advantage.

This approach is adaptive. If one of the broadcast sources is idle, or disconnects from the subnet, the capacity becomes available for other users (whereas in a TDMA or FDMA scheme it is more difficult or impossible to reallocate capacity). Because the allocation of capacity is done in software, it is flexible - it can take account of client rank, data priority, offered load, etc.

Use in BRASS

The BRASS broadcast compiler can certainly act as a source of one of the broadcast sources mentioned above. Some software will be required to interface the BRASS output to a 5066 broadcast client (this software could also convert Baudot to ASCII, or not as desired).

To take full advantage of the new capability, the special-purpose BRASS broadcast/5066 broadcast interface software would have to "peek" into the message (which is a layering violation, but actually the violation is having the RI and destinations in the message) at the routing indicators and destinations, and assign a group or individual address in some intelligent way.

Annex I: Messages and Procedures for Frequency Change (information only)

This annex defines PDUs and procedures to allow the use of automatic link establishment (ALE) systems to select a new operating frequency. It could also be used to support manual frequency change; however the required coordination procedures are not defined here.

For some systems it may be desirable to adapt the data rate along with the frequency change. Three cases may be considered:

1. Leave data rate at current settings (assumed in Option 1 above)
2. Return to default data rate
3. Set to new values

Initially, a simple procedure which implemented case 1 or 2 was envisioned. However, it turned out that the “simple procedure” was so similar to the DRC procedure defined in Annex C that it seemed overall a good idea to extend the DRC PDUs and procedures to the ALM procedure. This would cover all three cases above and is defined as option 1 below; the “simple procedure” is defined in option 2 below. Only one will be present in the final version of the STANAG.

I.1 Option 1: Combined Frequency and Data Rate Change (Automatic Link Maintenance - ALM)

This section defines PDUs and procedures for automatic link maintenance which combines data rate and operating frequency changes. This is an extension to the DRC PDUs and procedures defined in Annex C. Table I-1 defines the additional MANGEMENT message types which are used to implement this function.

Table I-1. ALM MANAGEMENT Messages

Message Type	Function	Contents
5	Automatic Link Maintenance Request (ALM_Req)	New HF modem transmit data rate and interleaving setting for ALM master
6	Automatic Link Maintenance Response (ALM_Resp)	Positive or negative response (including reason if negative)

The format and contents of the type 5 message shall be as defined for the type 1 message in C.3.8. The format and contents of the type 6 message will be as defined for the type 2 message in C.3.8, with a new reason as shown in table I-2.

Since the procedures defined in this Annex are extensions of the Annex C DRC procedures, they allow either

1. the ALM master tx parameters may be changed and the rx parameters left unchanged
2. the tx and rx parameters can be changed to identical new values

As the procedures stand now, there is no capability to specify during ALM new tx and rx parameters which are asymmetric. Another change which should be considered is to add a data rate parameter (extension of table C-7) which explicitly indicates that the parameters on the new frequency will be determined externally (ie, policy, or an ALE system).

Table I-2. Contents for Type 6 Message (Reason)

MSB - LSB	Interpretation
0 0 0 0 0	no reason (used to indicate unconditional acceptance of ALM_Request)
0 0 0 0 1	Tx and Rx parameters must be the same (conditionally accept)
0 0 0 1 0	Not possible to change modem data rate
0 0 0 1 1	Not possible to change modem interleaving
0 0 1 0 0	Not possible to change modem data rate or interleaving
0 0 1 0 1	Not consistent with local conditions
0 0 1 1 0	Not possible to change frequency

Procedures

Following a decision to change frequency, a node shall use type 6 D_PDUs containing type 5 and type 6 MANAGEMENT messages to coordinate the change. The node initiating the frequency change is referred to as the ALM master for this ALM procedure. The data rate and interleaving fields shall carry the data rate and interleaving for the ALM Confirm phase of the procedure (see Figure I-1). The data rate and interleave parameters to be used on the new frequency may be selected based on some external information (ie, a sounding or ALE system) or by policy (ie, existing or default parameters shall be used on the new frequency).

An advisory EOW message has not been defined which requests a frequency change, because the need to change frequencies will generally be caused by link conditions which have deteriorated below some threshold, and active measures are more appropriate than advisory measures in that situation.

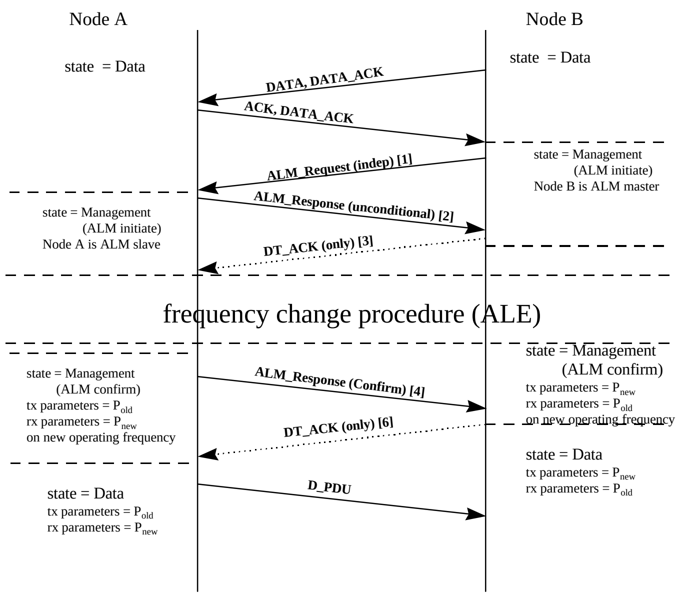

Node A
Node B
state = Data
state = Data
DATA, DATA_ACK
ACK, DATA_ACK
state = Management (ALM initiate) Node B is ALM master
ALM_Request (indep) [1]
state = Management (ALM initiate) Node A is ALM slave
ALM_Response (unconditional) [2]
DT_ACK (only) [3]
frequency change procedure (ALE)
state = Management (ALM confirm) tx parameters = P_old rx parameters = P_new on new operating frequency
ALM_Response (Confirm) [4]
state = Management (ALM confirm) tx parameters = P_new rx parameters = P_old on new operating frequency
DT_ACK (only) [6]
state = Data tx parameters = P_new rx parameters = P_old
D_PDU
state = Data tx parameters = P_old rx parameters = P_new

Figure I-1: Automatic Link Maintenance Procedure

(Data Rate and Frequency Change)

The node initiating the ALM procedure by sending a ALM Request (type 5) management message (shown at [1] in figure I-1) will be referred to as an "ALM master" (node B in figure I-1). When a node recognises a MANAGEMENT D_PDU addressed to it, containing a ALM_Request message, the node (referred to as the ALM slave, node A in Figure I-1) shall transition to the management state. The ALM slave shall respond to the ALM_Request D_PDU with a ALM_Response (type 6) message (shown at [2] in Figure I-1). The ALM Response message shall indicate either "accept" or "refuse", in accordance with Table I-2. If the ALM slave accepts the ALM_Request, the "reason" field shall indicate either "unconditional acceptance" or "Tx and Rx parameters must be the same". If the ALM slave refuses the request, the reason field shall indicate the reason for the refusal. Only the five reasons defined in the table are valid reasons for refusing a ALM_Request.

The figure shows an example in which the modem at the ALM slave also has independent transmit and receive data rate.

In order to increase reliability, the ALM message should be repeated. Table I-3 gives a minimum suggested number of times that a message should be transmitted, based on minimising the use of stuff bits in the interleaver. Other considerations could make a larger number of repetitions desirable.

Table I-3. Suggested minimum number of ALM messages to be transmitted at various data rates using STANAG 4285 modem

Data rate	repetitions (short interleave)	repetitions (long interleave)
75	1	9
150	1	18
300	1	37
600	3	75
1200	7	150
2400	15	300

The number of retransmissions is selected to (nearly) fill the modem interleave buffer. For waveforms and interleaver settings not shown, the number of repetitions should be selected as required to minimise the use of "stuff bits" to fill the modem interleave buffer.

After receiving the ALM_Response message the ALM master shall review its contents and determine the appropriate response [4]. The various ALM_Response messages, and the allowed responses from the ALM master, are shown in table I-4.

Table I-4. Possible ALM_Responses and Allowed ALM Master Actions

ALM_Response	ALM_Response reason	allowed from ALM master
accept	unconditional	DT_ACK only
accept	transmit and receive parameters must be the same	DT_ACK only, or ALM_Response (cancel, or ALM Request)$^{\text{note 1}}$
refuse	not possible to change modem data rate	ALM_Response (cancel)$^{\text{note 2}}$ or ALM_Request $^{\text{note 3}}$ (with DT_ACK)
refuse	not possible to change modem interleave	ALM_Response (cancel)$^{\text{note 2}}$ or ALM_Request $^{\text{note 4}}$(with DT_ACK)
refuse	not possible to change modem data rate or interleave	ALM_Response (cancel)$^{\text{note 2}}$(with DT_ACK)
refuse	not consistent with local conditions (see note 5)	ALM_Response (cancel)$^{\text{note 2}}$ or ALM_Request $^{\text{note 6}}$(with DT_ACK)

Notes to Table:

1. If EOW messages have been sent before the ALM procedure is initiated, the ALM master should already know that the ALM slave's transmit and receive parameters must be the same. Therefore, the ALM master should generally reply with a DT_ACK, accepting that the new parameters will apply to both transmit and receive.
2. ALM Slave shall acknowledge the cancel message with DT_ACK only; then the ALM procedure is discontinued. Note that this situation may frequently lead to failure of the link.
3. ALM_Request may be sent by master to request a different interleave setting at the same data rate.
4. ALM_Request may be sent by master to request a different data rate setting at the same interleave.
5. This reply shall only be sent in response to a request for a less robust set of parameters, i.e., higher data rate and/or shorter interleave than currently in use. It is expected that this will
6. ALM_Request may be sent by master to request different modem parameters which may be consistent with the local conditions.

In the table above, the DT_ACK refers to a data transfer sublayer acknowledgement of the preceding MANAGEMENT message (shown at [3] in Figure I-1). The DT_ACK reply indicates that the node has nothing further to communicate. If the DT_ACK (with no further management message) is sent in reply to a ALM_Response "accept" (as shown in Figure I-1), the nodes initiate the frequency change procedure, which may be controlled by an ALE system. Following the completion of the frequency change procedure, the nodes proceed to the "confirmation" phase. The ALM slave shall NOT initiate the frequency change procedure until it has received the DT_ACK (with no further management message) from the ALM master. If the DT_ACK (with no further management message) is sent by the ALM slave in reply to a ALM_Response "cancel", both nodes abandon the procedure and return to the prior state. If node A (formerly the ALM slave) has no queued data or acknowledgements to send to node B, it shall send a data D_PDU, expedited data D_PDU, or non-ARQ D_PDU, with zero data attached.

In the figure, the slave's ALM_Response with an "accept/unconditional" message generates the allowed DT_ACK from the ALM master.

After sending the DT_ACK [3], the ALM master initiates the frequency change procedure. On completion of the frequency change procedure, the ALM master sets its modem parameters and waits to receive a ALM Confirm message (type 6 MANAGEMENT message with response set to “confirm” and reason set to “none”) from node A (“confirmation phase”).

After receiving the DT_ACK [3], the ALM slave changes its modem parameters and transmits a ALM Confirm message [4] to the master. On receiving the ALM Confirm message, the master shall respond with a DT_ACK and then return to the previous state. After sending the ALM Confirm message [4] to the master and receiving the DT_ACK from the master, the slave shall return to the previous state and send any queued D_PDUs to node B. If node A (formerly the ALM slave) has no queued data to send to node B, it shall send a data D_PDU or expedited data D_PDU with zero data attached.

I.2 Option 2: Frequency Change without Data Rate Change

This Annex defines a data transfer sublayer procedure which will support a pause in data transfer for purposes of link management involving a change of frequency. The procedures for selecting the new frequency are not defined here; this is included for the benefit of systems which make use of some form of ALE for that purpose.

The procedures defined here make use of additional management messages (type 6 D_PDUs).

Table I-1. MANAGEMENT Message Types

Message Type	**Function**	**Contents**
5	Frequency Change Request (FC_Req)	
6	Frequency Change Response (FC_Resp)	Positive or negative response (including reason if negative)

MSB											LSB
11	10	9	8	7	6	5	4	3	2	1	0
0	**0**	**0**	**0**	**reset to 0**							
TYPE											

Figure I-1 (a). Message Type 5 Format

MSB											LSB
11	10	9	8	7	6	5	4	3	2	1	0
TYPE				**Response**			**Reason**				

Figure I-1 (b). Message Type 6 Format

Table I-2. Contents for Type 6 Message (Response)

MSB - LSB	Interpretation
0 0 0	accept
0 0 1	refuse
0 1 0	cancel
0 1 1	confirm

Procedures

Following a decision to change frequency, a node shall use type 6 D_PDUs containing type 5 and type 6 MANAGEMENT messages to coordinate the change. The node initiating the frequency change is referred to as the frequency change master (FC master) for this FC procedure.

An advisory EOW message has not been defined to which requests a frequency change, because the need to change frequencies will generally be caused by link conditions which have deteriorated below some threshold, and active measures are more appropriate than advisory measures in that situation.

Table I-2. Possible FC_Responses and Allowed FC Master Actions

FC_Response	allowed from FC master
accept	DT_ACK only
refuse	ALM_Response (cancel)note1

Notes to Table:

1. FC Slave shall acknowledge the cancel message with DT_ACK only; then the FC procedure is discontinued (as shown in Figure I-3).

In the table above, the DT_ACK refers to a data transfer sublayer acknowledgement of the preceding MANAGEMENT message (shown at [3] in Figure I-2). The DT_ACK reply indicates that the node has nothing further to communicate. If the DT_ACK (with no further management message) is sent in reply to a FC_Response "accept" (as shown in Figure I-2), the nodes switch to ALE mode in order to select a new frequency for the connection. When ALE is completed, the nodes enter the "confirmation" phase to complete the procedure.

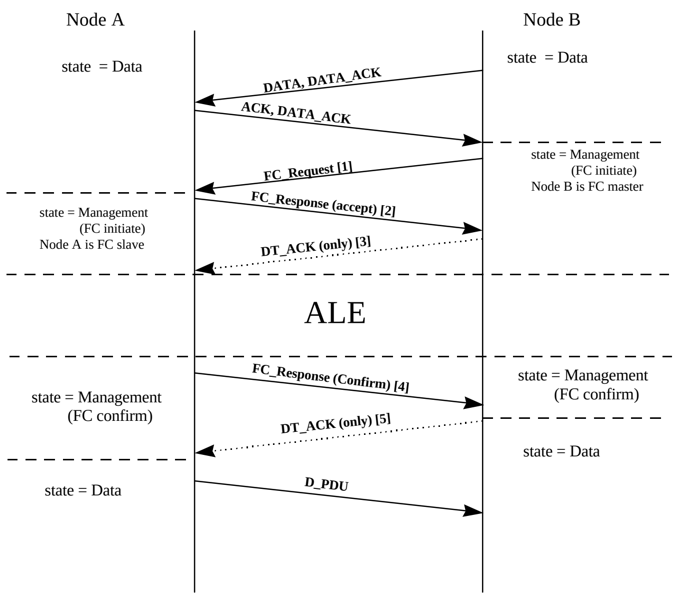

Node A
Node B
state = Data
state = Data
DATA, DATA_ACK
ACK, DATA_ACK
state = Management (FC initiate) Node B is FC master
FC_Request [1]
state = Management (FC initiate) Node A is FC slave
FC_Response (accept) [2]
DT_ACK (only) [3]
ALE
FC_Response (Confirm) [4]
state = Management (FC confirm)
state = Management (FC confirm)
DT_ACK (only) [5]
state = Data
state = Data
D_PDU

Figure I-2. Frequency Change Procedure (Example 1)

Figure I-2 presents a "normal", successful frequency change procedure. In this diagram, node B determines that a frequency change is needed. Node B then enters the management state and sends an FC_Request message to node A (shown at [1] in Figure I-2). On receiving this message, node A enters the management state and sends an FC_Response(accept) message (shown at [2]). Node B responds with a DT_ACK (shown at [3]) and enters the

ALE state as the active party. Node A enters the ALE state as the passive party, ie, expecting to receive from node B. On completion of the ALE procedure, node A sends a FC_Response(confirm) message (shown at [4]). Node B responds with a DT_ACK (shown at [5]), unless node B wishes to initiate a ALM procedure.

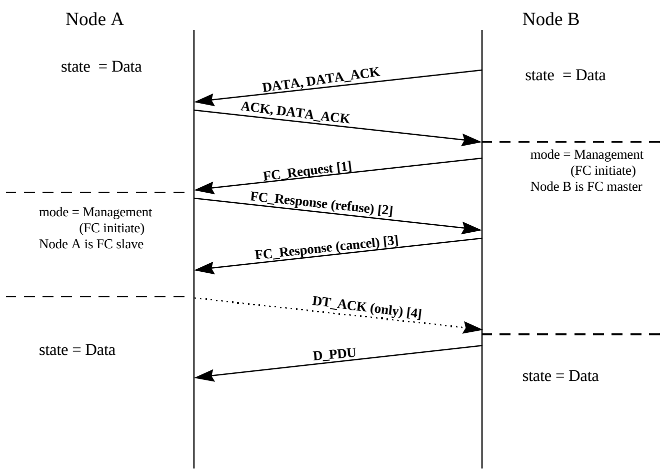


Figure I-3. Frequency Change Procedure (Example 2)