CCITT X.224 TRANSPORT LAYER PROTOCOL BASIC DESCRIPTION

Terry Fox, WB4JFI
President, AMRAD
1819 Anderson Rd.
Falls Church, VA 22043

Overview

In order to assure absolute data integrity through the amateur network, some form of transport layer protocol should be employed between the entry and exit points of the network. In datagram service, this transport comes in two basic forms, the Transmission Control Procedure (the TCP in TCP/IP) and the User Datagram Protocol, or UDP. The UDP is a very small transport protocol, and as such does not provide absolute data integrity under all conditions. The TCP is a much more robust protocol, and as such is capable of assuring absolute data integrity through the network, with a much higher overhead.

The Virtual Circuit amateur network concept using the CCITT standards (X.25/X.75) has generally relied on the use of the delivery confirmation, or D-bit procedures to maintain data integrity. Not only is this potentially a violation of the IS0 seven-layer model, but is also inadequate. If a virtual circuit connection is lost due to an intermediate packet switch malfunction, the D-bit procedures alone may not provide an accurate indication of what data was lost due to the malfunction. In addition, use of the D-bit provides no mechanism of detecting errors within the packet switches (such as memory errors) that might corrupt otherwise good data flowing through the amateur network.

This paper proposes the use of another CCITT X series protocol to correct for these potential deficiencies. This is a new recommendation, called X.224, and it describes a multi-class Transport Layer protocol that can be used on top of the X.25/X.75 Level 3 protocols. I will not give a detailed protocol specification here, but rather describe the different classes of the protocol and some of how they function.

Transport Layer Responsibilities

The basic function of the Transport Layer is to make sure that data traveling from the source end-point of a network to the destination end-point of the network does so in the proper order and without data corruption (if necessary). The Transport Layer relies on the Network Layer for providing a method of getting the data through the network from the source end-point to the destination end-point. Once the Transport Layer entity is sure that the data leaving it matches the data entering its peer Transport Layer entity, it will pass the data up to the next layer for further processing if required. The Transport Layer should have at its disposal some method of detecting errors, informing its Transport peer of these errors if necessary, and possibly correcting some forms of errors commonly encountered.

Errors Encountered By The Transport Layer

Before the X.224 Transport Layer recommendation is discussed, it might be helpful to describe some of the errors and situations a Transport Layer might have to deal with. Among these situations are:

A. Data Loss.
B. Data Corruption.
c. Data Duplication.
D. Data Misordering.
E. Data Misdelivery.
F. Network flow-controlled.
G. Upper layer flow-controlled.
H. Network Concatenation and Separation.
I. Segmenting and Reassembly.
J. Splitting and Recombining.
K. Multiplexing and Demultiplexing.

Data Loss

Loss of data can occur for a number of reasons in a network. If an intermediate packet switch goes down while holding data it acknowledged receiving but before sending to the next switch is one example. Total network failure is example of absolute data loss. These type of failures should be detected and corrected. This may mean a request for the missing data should be sent, or possibly tearing down the malfunctioning connection through the network and informing the user of the network failure.

Data Corruption

Data corruption is when data passes through the network from the source end-point to the destination end-point, but something happens to it along the way to create errors in it. While most of the time data passed through the amateur network will be passed from point-to-point using a reliable Link Layer protocol (such as AX.25), if an intermediate packet switch has a memory malfunction, data passed through that switch will be corrupted. Since the network corrupted the data, there should be some method of detecting and correcting this situation, if necessary.

Data Duplication

It is possible to have the same data delivered to the Transport Layer more than once. This happens most frequently when retransmissions due to loss of acknowledgements occur. In datagram networks can happen a lot when some type of flooding mechanism is used to correct network deficiencies. There should be some sort of detection of duplicated data, which results in the duplications being thrown out.

Data Misordering

With some network desi ns, it is possible for a group of data send be4ore another group to arrive at the destination AFTER the second send group. This is common with datagram networks employing some sort of dynamic network routing scheme. Not only should this be detected, but some method of data re-ordering must be used to regain the original data order.

Data Misdelivery

Whenever the Transport Layer receives data from the Network Layer that it does not understand what to do with for reasons other than specified, it should have some method for acting upon the error. This can be as simple as ignoring the bad data, or as drastic as tearing down the Transport connection and notifying the user that the network has failed. This is kind of a catch-all for the "I'm so confused" feeling.

Network Flow-Controlled

While not an error, if not properly handled network flow control could cause the Transport Layer to become congested when the upper layer(s) keeps sending data that the Transport Layer cannot send over the network. Some method of flow-controlling the upper layer(s) should be employed to prevent data loss due to this potential situation.

Upper Layer Flow Controlled

This is also not an error, but rather a situation that could cause errors. If the upper layer(s) cannot accept more data from the network-through the Transport Layer, that upper layer(s) may try to stop further data from reaching it from the Transport Layer. If the Transport Layer is not capable of reacting properly, data may become lost between the layers.

Network Concatenation and Separation

Some networks allow the Network Layer protocol to concatenate more than one Transport Layer data unit into a single Network Layer data unit at the source end-point, and then separate the Transport Layer data units at the destination end-point. This may cause some problems at the Network/Transport Layer interface, which must be handled by the Transport Layer.

Segmentation and Reassembly

If the Transport Layer receives data from the upper layer(s) in a group or size that it cannot handle in a single Transport Layer data unit, it should break the data into multiple smaller Transport data units for sending through the network. The destination Transport Layer end-point should be capable of reassembling these samller data units into their original form before passing the data to the upper layer protocol(s). Some method of indicating this segmenting must be employed, along with some numbering scheme to make sure the data is reassembled properly at the destination end-point.

Splitting and Recombining

Occasionally Transport Layer protocols may use more than one network connection to support the same Transport Layer connection. If this happens, some of the above mentioned data errors may occur, especially data misordering. Special procedures may be required to support this function.

Multiplexing and Demultiplexing

This is when a single Network Layer connection is shared multiple Transport Layer connections. This will happen frequently, since it will result in reduced Network Layer overhead. Some method of assuring proper demultiplexing should be employed, otherwise the wrong user may get someone elses data.

There are more potential errors that can creep into the network system. The above highlights(?) some of the more common problem areas.

CCITT X.224 Recommendation

The CCITT has written Recommendation X.224 to serve as a Transport Layer Specification for Open Systems Interconnection networks. It is designed to be flexible enough to be used with a variety of Network Layers operating underneath it, while requiring a minimum amount of overhead.

Like most Transport Layer protocols, X.224 is designed to establish a "virtual connection" between two Transport Layer peers. In order not to ruffle the feathers of the datagramies, I will use the term "logical connection" in place of "virtual connection" throughout the rest of this paper.

One of the peers in this logical connection is called the source end-point, and the other is called the destination end-point. This is not meant to imply that data flows only in one direction, but rather that when an upper layer protocol sends a Transport Layer some data, as far as that data is concerned, it is at the source end-point of the Transport Layer connection. The source Transport Layer end-point then adds any overhead to the data that it may require for proper Transport Layer operation, and sends the resulting data to the Network Layer below it for transferring the data across the network to the destination Transport end-point.

The destination Transport Layer end-point then receives the data from its Network Layer, strips off and acts upon the overhead added by the source Transport end-point? and if everything is fine, passes the resulting data to its upper layer(s) for further processing.

As long as it is required to transfer data between network users, and no major errors occur, the logical connection between the two Transport Layer peers is maintained. Some errors can be recovered from using X.224 procedures, while others require tearing down, and re-establishing the logical connection, if it is still wanted after the error.

In order to provide one recommendation that is robust enough to handle different qualities of service provided by different networks, X.224 uses five different classes of protocols. This has the advantage of not requiring unnecessary overhead when using network protocols that provide a high degree of reliability, while allowing the overhead to grow to support network protocols that less quality of service.

CCITT X.224 Classes

The five classes of X.224 are:

Class 0. Simple Class.
Class 1. Basic Error Recovery Class.
Class 2. Multiplexing Class.
Class 3. Error Recovery and Multiplexing.
Class 4. Error Detection and Recovery Class.

The class of Transport Layer protocol used is inversely proportional to the quality of service provided by the Network Layer. If the network and the Network Layer protocol provides a high quality of service, then a simple Transport Layer protocol can be used. When the network service quality is poorer., a more sophisticated Transport Layer protocol might be required. There are three classifications of network quality defined in X.224:

Type A. Network operation with acceptable residual error rate (errors not signalled by disconnect or reset) and an acceptable rate of signalled errors.

Type B. Network operation with acceptable residual error rate, but unacceptable rate of signalled errors.

Type C. Network operation with unacceptable residual error rate.

Class 0.

Class 0 provides the simplest type of Transport Layer connection. It is designed to be used with type A network connections. It provides the functions necessary for logical connection establishment, data transfer with segmenting, and error reporting. Flow control relies on network provided flow control, and disconnection based on network service disconnection.

Class 1.

Class 1 provides a basic transport connection with minimal overhead, and is usually used with type B networks. The main purpose of class 1 is to recover from a network disconnect or reset. This is the basic class recommended for use in the amateur network over AX.25/AX.75 Network Layer protocols when absolute data integrity is not required.

Class 1 provides transport connections with flow control based on network provided flow control, error recovery, expedited data transfer, disconnection, along with the ability to provide consecutive transport connections on a network connection. In addition to providing Class 0 functions, Class 1 also provides the ability to recover from Network failures without affecting the network user. This is the main advantage of Class 1, and corrects for a potential problem in AX.25/AX.75 network designs.

Class 2.

Class 2 is designed to be used with type A networks. It provides a way to multiplex several transport connections onto a single network connection. Class 2 also allows the use of transport flow controls to help avoid congestion at Transport Layer connections end-points. No error detection or recovery is provided by class 2.

If the network connection resets or disconnects, the transport connection is terminated, and the user is informed.

Class 3.

Class 3 provides the characteristics of class 2 with explicit flow control, plus the ability to recover from network disconnects or resets from class 1. This is usually used with type B networks.

Class 4.

Class 4 is the most sophisticated Transport Layer protocol, and is designed for use with type C networks. In addition to providing Class 3 services, it also detects and recovers from errors which occur as a result of a low grade of service from the network. The kinds of errors detected include:

A. Data Loss.
B. Out-of-Sequence Data Delivery.
C. Data Duplication.
D. Data corruption.

Detection and recovery from errors is enhanced by the extended use of data sequence numbering, timeout procedures, and a simple checksum system.

The class of protocol used can be negotiated at connection request time. If the preferred class is not available, an alternate may be selected, if appropriate.

X.224 Headers

Figure 1 shows the basic structure for the Transport Layer headers used in X.224. These headers are an integral number of octets long.

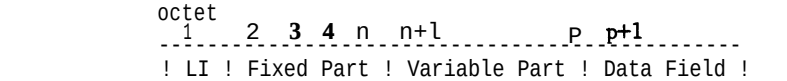

Figure 1. X.224 Transport Layer Header

LI is the Length Indicator field. It indicates how long the Transport header is in binary (less the LI itself). The maximum value is 254 octets (1111 1110).

The fixed part contains frequently occuring parameters, including the coding of the Transport Protocol Data Unit (TPDU) type. The length of the fixed part is determined by the type of data unit, protocol class, and format in use (normal or extended). The coding of the TPDU types are shown in Figure 2.

TPDU Name	0	1	2	3	4	coding
CR connect request	x	x	x	x	x	1110xxxx
CC connect confirmation	x	x	x	x	x	1101xxxx
DR disconnect request	x	x	x	x	x	10000000
DC disconnect confirm.		x	x	x	x	11000000
DT data	x	x	x	x	x	11110000
ED expedited data		x	F	x	x	00010000
AK data acknowledgement		C	F	x	x	0110zzzz
EA expedited data ack.		x	F	x	x	00100000
RJ reject		x		x		0101zzzz
ER TPDU error	x	x	x	x	x	01110000
PI Transport Protocol ID						00000001
						00000000
already in use by other protocols						00110000
not allowed in CCITT X.224						1001xxxx
						1010xxxx

Figure 2. TPDU Coding and Class Usage

Where:

xxxx indicates initial credit allocation in classes 2, 3, and 4. 0000 in classes 0, and 1.

zzzz indicates initial credit allocation in classes 2, 3, and 4. 1111 in class 1.

F Not available when non explicit flow control option is selected.

C Not available when receipt confirmation option is selected.

The variable part is used to indicated less frequently used parameters, if any are needed. The first octet of each parameter holds the parameter code, the second octet contains the parameter value length indicator, and the rest holds the parameter value itself. An example of use of the variable part is the use of checksums for the class 4 protocol.

The data field contains transparent user data, if any. Size restrictions depend on each TPDU type.

TPDU Header Structures

Figure 3 gives an outline of the structures of the headers used. It is not meant to be a complete description, just a brief indication of TPDU header size.

As mentioned above, the LI field is used to indicate how long (less the LI field itself) the Transport Layer header is.

The second byte contains the TPDU type, and optionally a credit field.

The DST-REF field is used to contain the address of the destination Transport end-point, and the SRC-REF holds the address of the source Transport end-point.

Connection requests and Connection confirmation headers contain a single-octet field that hold the preferred class the requesting Transport device wants to use, along with two option bits (use/non-use of explicit flow control and normal/extended formats in classes 2, 3, or 4.

Additional parameters may be requested in the variable part field, and are selected from among the following:

A. Transport Data unit size.
B. Version Number of Transport protocol.
C. Security parameters.
D. Checksum operation (class 4).
E. Alternative protocol class(es).
F. Acknowledge time adjustment.
G. Throughput adjustment.
H. Residual error rate.
I. Priority of data.
J. Transit delay.
K. Reassignment/resynchronization time.
L. Expidated data, additional options.

If a disconnect TPDU is sent, it should include a reason field which indicates why the disconnect was requested.

Data TPDUs contain a field that contains a bit (called EOT) which indicates when a TPDU is the end of a sequence of TPDUs. This field also contains the TPDU-NR, the TPDU send-sequence number. The TPDU-NR is set to zero in class 0, and may be any value in class 2 without explicit flow control.

Acknowledgement TPDUs contain a field called YR-TU-NR. This field contains the sequence number of the next expected data TPDU. It is used to indicate the last correctly received data TPDU to the source Transport end-point.

The last type of field used is the reject cause field in error TPDUs. As implied, it informs the source Transport end-point that the destination Transport end-point has rejected a TPDU, and why.

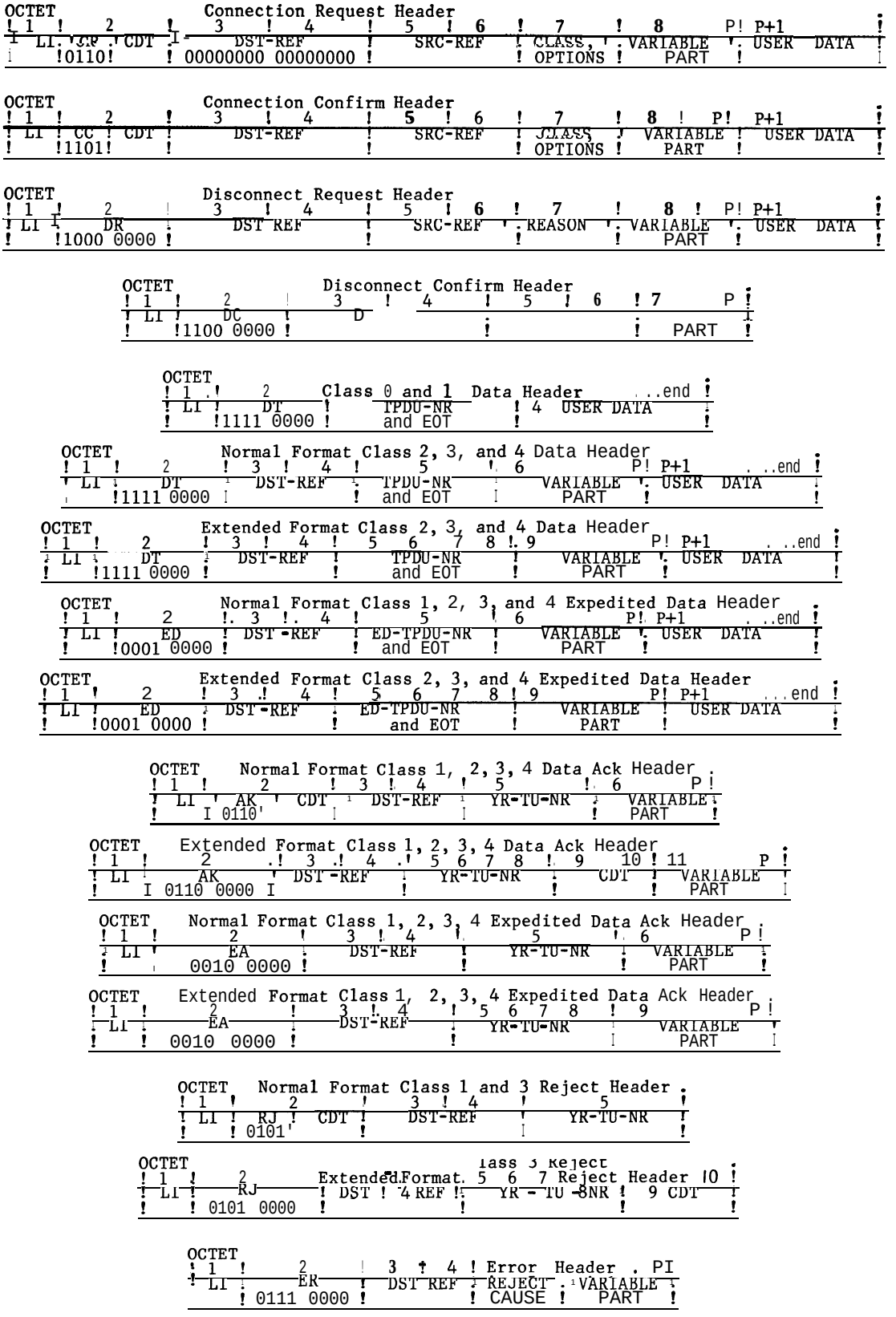

Figure X.224 Transport Protocol Data Unit (TPDU) Header Formats

Procedures Available By Class

Figure 4 shows the various procedures of X.224, and in which classes they are available.

Procedure	Variant	Class 0	1	2	3	4
Assignment to network cnct		x	x	x	x	x
TPDU data transfer		x	x	x	x	x
Segmenting, reassembly		x	x	x	x	x
Concatenation, separation			x	x	x	x
Connect. establishment		x	x	x	x	x
Connection refusal		x	x	x	x	x
Normal Release	implicit	x				
	explicit		x	x	x	x
Error release		x	x	x		
Assoc. of TPDU with TCs		x	x	x	x	x
Data TPDU numbering	normal		x	m	m	m
	extended			o	o	o
Expedited data transfer	net normal		m	x	x	x
	net explic.		o			
Reassignment after fail.			x		x	x
Retention until acknowledgement of TPDUs	Conf. rcpt.		o			
	AK		m		x	x
Resynchronization			x		x	x
Multiplexing, de-muxing				x	x	x
Explicit Flow Control	with			m	x	x
Checksum	without	x	x	o		
	use of					x
	non-use of	x	x	x	x	o
Frozen references			x		x	x
Retransmission on timeout						x
Resequencing						x
Inactivity Control						x
Treatment protocol errors		x	x	x	x	x
Splitting and Recombining						x

X.224 Procedures

It is beyond the scope of this paper to describe the full operation procedures of the various X.224 classes. The above information is presented to show that there is an alternative Transport protocol to TCP that would work very effectively on top of an X.251X.75 type Network Layer protocol. I will continue to process the X.224 document into a form that will be presentable to amateurs, along with making sure that it is 100% amateur compatible.

The basic operational procedures of X.224 is designed to allow a logical connection to be established between a source Transport end-point and a destination Transport end-point using the connection request and connect confirm TPDUs, which may be passed along as part of the data field of an X.25/X.75 Ntework Layer fast-select connect request.

Once a connection has been established, data may flow in both directions, with optional flow controls, checksums, and sequence numbers. Optionally, expedited data can also be sent. Bad data can be rejected if necessary, and protocol errors can be detected and signalled.

When the connection is no longer needed, it may be terminated either explicitly, or by inference when the network connection servicing the Transport Layer is torn down.

Conclusion

I believe that X.224 is a viable Transport Layer protocol to use over an X.25/X.75 network. X.224 provides the small extra amount of protection over the X.25/X.75 network layer to insure absolute data integrity when necessary at a very low amount of overhead. Since X.224 is similar to AX.25 level 2 operation and X.25/X.75 network operation, an extensive software development campaign is not necessary to implement this protocol. Level 2 and level 3 protocol machines could be modified to provide much of the basic core of the X.224 protocol.

Interested users are encouraged to write the author for the latest in X.224 development, along with AX.25/AX.75 Network Layer development. It is also recommended to join AMRAD, as the AMRAD Newsletter contains fairly up-to-date information on packet radio development.