Header Sheet

This sheet is required for the configuration control of this document.

It is not a part of the delivered document.

Title: Transport Protocol Design {Title, Summary Block }

Doc. No: 5000 xxxx.00 {Subject/Rev, Summary Block }

Date: 21 June, 1997 {Create Date, auto}

File: TP.DOC {Filename, auto}

Directory: _____ (Server Directory, enter when filing)

Prepared by: Matthew Gream {Author, Summary Block }

_____ (signature)

Reviewed by: _____ (Reviewers Name, enter at start)

_____ (signature)

Approved by: _____ (Approvers Name, enter at start)

_____ (signature)

Document History: (Enter when filing)

Rev.	ECN	Date	Name	Page(s)	Description
01					
02					

ENGINEERING DESCRIPTION

Transport Protocol Design

June, 96

No. 5000 xxxx.00
A.C.N. 003 169 088

Table of Contents

1. Introduction

1.1. Scope

This document describes the design of the Transport Protocol Task. This task is responsible for providing unreliable and reliable data transport facilities for a software data path. It is used initially to convey signalling across the Frame Relay data path.

The following sections provide a brief description of the high level design, including the requirements, context, architecture and design. Following this, interface details are given, after which the detailed aspects of the design are presented in more detail.

1.2. Document Layout

This document is structured along the following guidelines.

- Section one describes the document layout and conventions.
- Section two provides a general description of the subject of the document. This includes a summary of the major functions, the user(s) and the constraints of the system. Any assumptions ore dependencies associated with the subject are also included.
- Section three describes external interfaces to this design.
- Section four gives the detailed algorithms, timing diagrams, and other notations for the subject of the document.
- Support information such as references and definitions are provided at the end of the document.

1.2.1. Conventions

Boxed paragraphs of Warning Text style highlight omissions and issues subject to change, debate or further study.

2. General Description

2.1. Requirements

The Transport Protocol is required to provide an Unreliable or Reliable data delivery service. Specifically, it must support the following requirements.

1. It can be configured by the Resource Manager.

    a. It must send a Sign On to the Resource Manager.

    b. It must accept a Connect Request from the Resource Manager.

    c. It must accept a Disconnect Request from the Resource Manager.

    d. It can send a Disconnect Indication to the Resource Manager.

2. It can act as an Unreliable Protocol for the Data Path.

    a. It must accept Upper Data and pass as Lower Data.

    b. It must accept Lower Data and pass as Upper Data.

    c. It must accept Connect and Disconnect requests.

3. It can act as a Reliable Protocol for the Data Path.

    a. It must accept Upper Data and pass as Lower Data.

    b. It must accept Lower Data and pass as Upper Data.

    c. It must accept Connect and Disconnect requests.

    d. It must ensure reliable and sequenced delivery for the Upper Data.

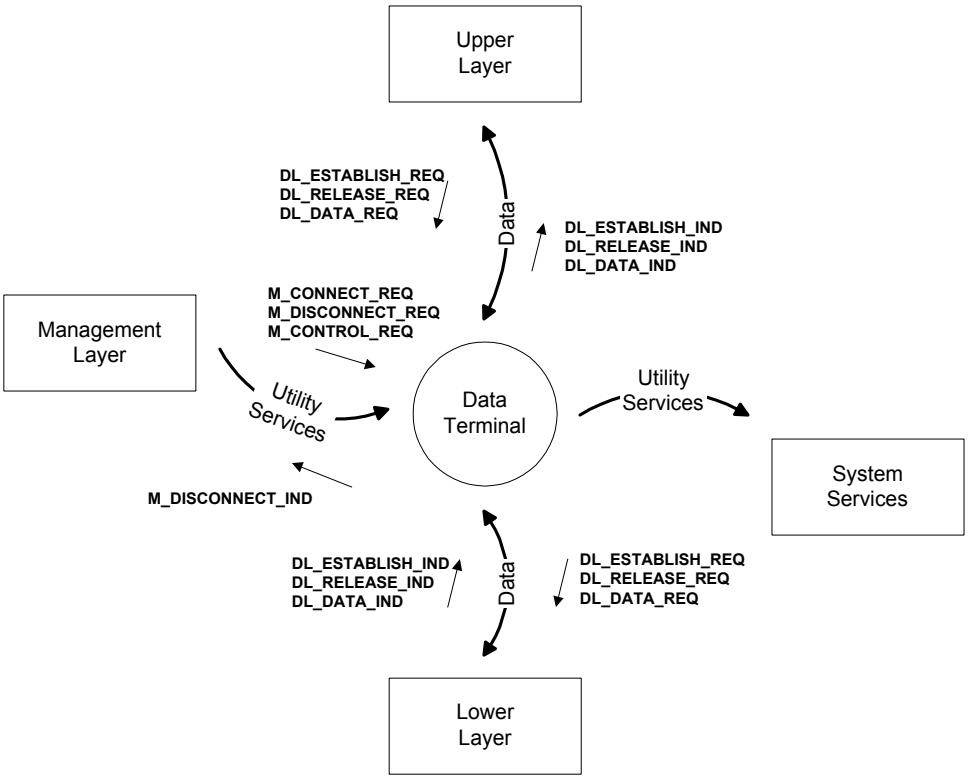
    e. It must provide a Failure request to the Upper Layer if it cannot provide these facilities.

4. It must interface to a Fast Packet Switch Lower Layer.

    a. It must accept and utilise fast packet addressing information.

2.2. Context

The Transport Protocol is a JEXEC Task. There are four primary external entities as illustrated in the following diagram.



1. **Management Layer** -- This is the entity that configures the Transport Protocol, and associates it with its Upper and Lower layers. It uses management messages to transfer configuration and association information.

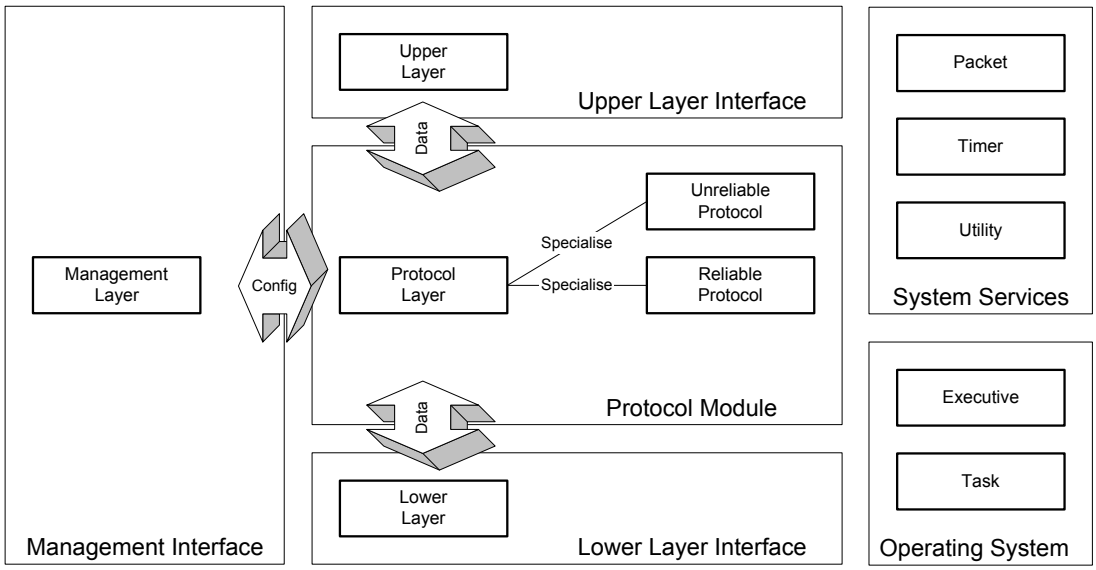
2. **Lower Layer** -- This is a source or sink of data on the data path, it is associated with this layer via the Management layer. It uses data link messages to transfer data.

3. **Upper Layer** -- This is a source or sink of data on the data path, it is associated with this layer via the Management layer. It uses data link messages to transfer data.

4. **System Services** -- These provide buffers, instrumentation, system time, operating system facilities and so on. These are always available for use through global function calls.

2.3. Architecture

The architecture of the Transport Protocol builds upon a representation of the context and provides a modular structure that is re-used in other items of software. There are a number of primary modules as illustrated in the following diagram.



1. **Management Layer Module** -- Provides the interface to the Management Layer, accepting and issuing messages, along with co-ordinating the establishment of the Upper, Lower and Protocol Layer Modules (i.e. binding them together and configuring their peers).

2. **Lower Layer Module** -- Provides the interface to the external Lower Layer, and to the internal Protocol Layer Module. It translates data for the Lower Layer upon reception from, or direction to, the Protocol Layer Module and is configured by the Management Layer Module.

3. **Upper Layer Module** -- Provides the interface to the external Upper Layer, and to the internal Protocol Layer Module. It translates data for the Upper Layer upon reception from, or direction to, the Protocol Layer Module and is configured by the Management Layer Module.

4. **Protocol Layer Module** -- Provides the implementation of the basic protocol, which interacts with the Lower Layer Module and Upper Layer Module and is configured by the Management Layer Module. There are a number of specialised Protocol Layer Modules.

5. **Executive Module** -- Provides the central point for co-ordinating the establishment of the modules, accepting external inputs (Messages) and directing them to the appropriate Module. This is an interface to the Operating System.

6. **Task Module** -- Provides the mechanisms to install, remove and execute the Executive Module as a JEXEC Task, by specifying operating system parameters. This is an interface to the Operating System.

7. **Timer Module** -- Provides timer services to the other Modules allowing them to activate and utilise timers. This is an interface to the Operating System.

8. **Packet Module** -- Provides packets for use by the other Modules. The packets are the containers used to manipulate data internally, between the layers and so on. This is an interface to the System Services.

9. **Utility Module** -- Provides instrumentation and related services. This is an interface to the System Services.

10. **Unreliable Protocol Layer Module** -- This is a specialised instance of the Protocol Layer Module that provides an unreliable transport service.

11. **Reliable Protocol Layer Module** -- This is a specialised instance of the Protocol Layer Module that provides a reliable transport service.

2.4. Design

2.4.1. System

The Transport Protocol is designed as a JEXEC task. It is assigned the Task Identifier *JEXEC_TASKID_TRANSPORT_PROTOCOL (0xEC)*. This is specified in *RMTASKID.H*.

The Transport Protocol is managed by the Resource Manager, for which it is assigned the Resource Identifier *RM_RESOURCE_ID_FP_TRANSPORT*. This is specified in *RM_RES_N.H*.

A total of 16 copies (instances) are made available for use by the system. Each of these may be individually configured, activated and deactivated. If required, more copies can be made available by changing the define TP_INSTANCE_MAX in *tp.h*.

2.4.2. Configuration

The Transport Protocol is configured through PIDs that are conveyed in an *M_CONNECT_REQ* message. This message specifies a particular copy (instance) of the Transport Protocol that is to be activated, and conveys configuration to configure the modules that are to be activated.

The PID *PID_TP_TYPE* is used to specify which of the two protocols (**Unreliable** or **Reliable**) are to be used for the copy (instance) that is being activated. Remaining PIDs then configure specific protocol parameters.

The following table illustrates all configuration PIDs and the modules that use them.

PID	MGMT	LOWR	UPPR	UNRL	RELI
PID_TP_TYPE	3				

Key:

MGMT:	Management Module
LOWR:	Lower Interface
UPPR:	Upper Interface
UNREL:	Unreliable Module
RELI:	Reliable Module

The following configuration PIDs are supported in order to select a particular protocol type.

PID_TP_TYPE	
Description	Specifies the protocol that is to be used for the instance of the task that is being activated.
Value	0xA0 : PID_TASK_BASE + 0x00
Length	1 octet
Content	0x00 : No Protocol, an invalid configuration 0x01 : Unreliable Protocol. 0x02 : Reliable Protocol.

2.4.3. Lower Layer Processing

The Lower Layer Module accepts Datalink Messages from the external Lower Layer and behaves according to the following table.

Message	Description
DL_ESTABLISH_IND	This indicates that the Lower Layer is established, no action is taken.
DL_RELEASE_IND	This indicates that the Lower Layer has released, no action is taken.

DL_ESTABLISH_CON	This may be provided in response to a DL_ESTABLISH_REQ, no action is taken.
DL_RELEASE_CON	This may be provided in response to a DL_RELEASE_REQ, no action is taken.
DL_DATA_IND	This conveys data from the Lower Layer; which is extracted from the message and delivered to the Protocol Layer.

The Lower Layer Module also accepts a number of events from the internal Modules. This includes Upper Layer (i.e. Protocol Layer Module) data input, Management Layer Module configure, connect and disconnect and Lower Layer message input. An exception can also be generated to be trapped by the Management Layer Module.

The Lower Layer acts in a different manner when it is communicating with the Packet Switch. If it is, then it ignores the copy provided in parameter [0], and uses the *VirtualCircuitID* encoded in the message. It locates the appropriate instance by matching its *address* with the *address* specified in the initial *M_CONNECT_REQ* message. It uses a Frame Type of *FR_FRAME_E2E_SIGNALLING.* When encoding or decoding the lower layer message, it uses the global *build_jtec_frame_relay_header* and *remove_jtec_frame_relay_header* functions.

2.4.4. Upper Layer Processing

The Upper Layer Module accepts Datalink Messages from the external Upper Layer and behaves according to the following table.

Message	**Description**
DL_ESTABLISH_REQ	This indicates that the Upper Layer requests activation of the data path.
DL_RELEASE_REQ	This indicates that the Upper Layer requests deactivation of the data path.
DL_ESTABLISH_RES	This may be provided as a response to a DL_ESTABLISH_IND, no action is taken.
DL_RELEASE_RES	This may be provided as a response to a DL_RELEASE_IND, no action is taken.
DL_DATA_REQ	This conveys data from the Upper Layer; which is extracted from the message and delivered to the Protocol Layer.

The Upper Layer Module also accepts a number of events from the internal Modules. This includes Lower Layer (i.e. Protocol Layer Module) data input, Management Layer Module configure, connect and disconnect and Upper Layer message input. An exception can also be generated to be trapped by the Management Layer Module.

2.4.5. Unreliable Protocol Processing

The Unreliable Protocol Layer Module accepts events from the internal Modules and provides a facility for the transparent unacknowledged delivery of data. It performs the following activities:

- **Upper Layer Data Input** -- Pass the data to the Lower Layer Output.
- **Lower Layer Data Input** -- Pass the data to the Upper Layer Output.
- **Connect** -- Indicate that the protocol is connected.
- **Disconnect** -- Indicate that the protocol is disconnected.
- **Timer Expiry** -- Nothing.

2.4.6. Reliable Protocol Processing

The Reliable Protocol Layer Module accepts events from the internal Modules and provides a facility for the reliable acknowledged delivery of data. It uses Reliable Data Protocol (RFC-0908) [RFC-0908] for this: Refer to the Detailed Design for more specific information. It performs the following activities:

- **Upper Layer Data Input** -- A SendRequest event is exercised on the RDP module; to pass the data into the RDP state machine.
- **Lower Layer Data Input** -- A SegmentArrival event is exercised on the RDP module; to pass the data into the RDP state machine, after which ReceiveRequest events are exercised to extract awaiting data. Any awaiting data is delivered to the upper layer.
- **Connect** -- A OpenRequest event is exercised on the RDP module, if this fails then an exception is raised.
- **Disconnect** -- A CloseRequest event is exercised on the RDP module.
- **Timer Expiry** -- A Timeout event is exercised on the RDP module.

2.4.7. Management Processing

The Management Module accepts message events and uses these to co-ordinate the operation of the Lower, Upper and Protocol Layers. It performs the following activities:

- **Layer Initialisation** -- A Connect message from the Resource Manager is used to configure a stack of Upper, Protocol and Lower layers; then to bind those layers together and connect them for operation.
- **Layer Termination** -- A Disconnect message from the Resource Manager is used to disconnect layers from operation.
- **Layer Control** -- A Control message from the Resource Manager is used to provide additional configuration and control information for a previously connected layer.
- **Layer Exception** -- An exception generated by a connected layer is used to send a Disconnect message to the Resource Manager, therefore indicating that the layer is not valid anymore.

3. External Interfaces

3.1. Lower Layer Interface

The Lower Layer uses messages for communication. The following messages are exchanged with the Lower Layer. These either indicate activation or deactivation of the data path, or provide information to convey on the data path.

Message	Description
DL_ESTABLISH_IND	Indicate that the Lower Layer is activated.
DL_RELEASE_IND	Indicate that the Lower Layer is deactivated.
DL_ESTABLISH_CON	Indicate that the Lower Layer is acknowledging a DL_ESTABLISH_REQ.
DL_RELEASE_CON	Indicate that the Lower Layer is acknowledging a DL_RELEASE_REQ.
DL_DATA_IND	Provision of data from the Lower Layer.

The Buffer Descriptors are encoded and decoded into *DL_DATA_IND* messages using the global buffer encoding and decoding mechanisms. This involves the use of *IsolateBufferDescriptorPID* and *AddBufferStructPID*.

The messages are addressed using the Resource Id and Resource Copy as configured by the Resource Manager, however when communicating with the Packet Switch, then VirtualCircuitId is used with DLCI addressing.

3.2. Upper Layer Interface

The Lower Layer uses messages for communication. The following messages are exchanged with the Lower Layer. These either indicate activation or deactivation of the data path, or provide information to convey on the data path.

Message	Description
DL_ESTABLISH_REQ	A request from the Upper Layer to indicate that it wants to be activated.

DL_RELEASE_REQ	A request from the Upper Layer to indicate that it wants to be deactivated.
DL_ESTABLISH_RES	An acknowledgement from the Upper Layer to a DL_ESTABLISH_IND.
DL_RELEASE_RES	An acknowledgement from the Upper Layer to a DL_RELEASE_IND.
DL_DATA_REQ	Provision of data from the Upper Layer.

The Buffer Descriptors are encoded and decoded into *DL_DATA_REQ* messages using the global buffer encoding and decoding mechanisms. This involves the use of *IsolateBufferDescriptorPID* and *AddBufferStructPID*.

The messages are addressed using the Resource Id and Resource Copy as configured by the Resource Manager.

3.3. Management Interface

The Management Layer uses messages for communication. The following messages are exchanged with the Management Layer.

Message	**Description**
M_STATUS_IND	Used to convey Sign On information to the Resource Manager about the availability of the Data Terminal.
M_CONNECT_REQ	Accepted from the Resource Manager to Connect and Configure an instance of the Data Terminal.
M_DISCONNECT_REQ	Accepted from the Resource Manager to Disconnect an instance of the Data Terminal.
M_DISCONNECT_IND	Used to inform the Resource Manager about a Disconnect of an instance of the Data Terminal.
M_CONTROL_REQ	Used to provide control information to the Data Terminal from the Resource Manager.

The Sign On message indicates that there are 128 copies available, and provides the Resource Identifier *(RM_RESOURCE_ID_SW_TERMINAL)*.

The Connect message specifies the Resource Copy to connect, and provides any other configuration information within the message itself.

The messages are addressed using parameter [0]. This is where the Resource Copy is placed.

4. Detailed Description

4.1. Reliable Data Protocol (RFC-0908)

The Reliable Data Protocol (RFC-0908) [RFC-0908] is an efficient and concise reliable delivery protocol. The specification should be consulted for more complete detail.

4.1.1. Features

The protocol supports the following features.

- **Initialisation Phase** -- the protocol has a TCP like initialisation phase where both sides perform synchronisation and acknowledgement steps in order to exchange initial sequence numbers. Having done this, they enter the data transfer phase.
- **Active and Passive Initialisation** -- the protocol can establish itself in an active or passive mode, where the active side of the protocol will initiate the synchronisation phase with the passive side. The protocol does support two active sides and a simultaneous synchronisation phase.
- **Synchronisation Options** -- the synchronisation segments contain options and parameters that are used to specify operational attributes for each side of the connection.
- **Termination Phase** -- the termination phase is not graceful; once shut down, all subsequent remote segments are discarded; and retransmissions will not be performed.
- **Segmented Delivery** -- data is transfered in segments, unlike other transport protocols that are stream oriented.
- **Window Based** -- data is transfered in segments using 16bit transmit and receive windows of size 16. This bounds the amount of data that can be contained within the network at any one point in time.
- **Cumulative Acknowledgements** -- acknowledgements can be provided in a cumulative fashion.
- **Selective Acknowledgements** -- acknowledgements can be provided in a selective fashion in response to out of order segments. This allows the transmitter to remove segments from its queue prior to retransmission.

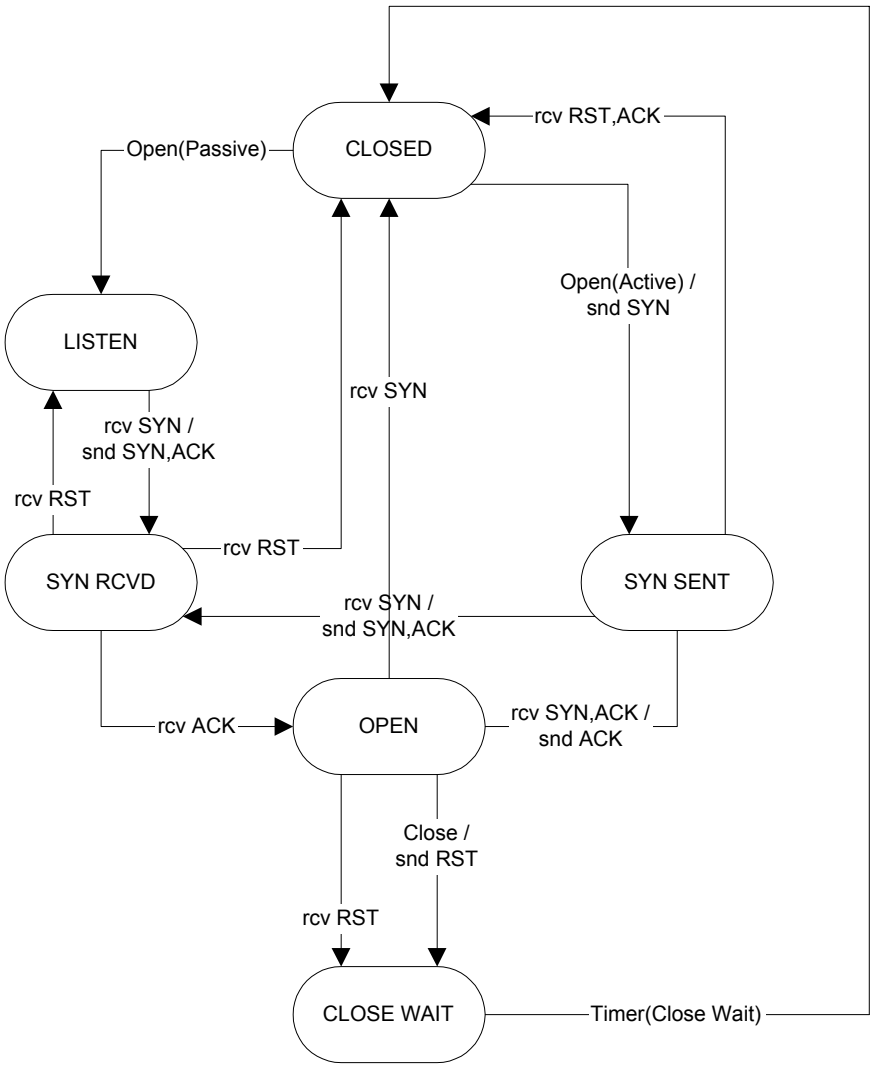
- **Retransmission Timers** -- the protocol maintains retransmission timers for each packet, and retransmits the packet when an acknowledgement for it has not been received within a set period of time.

- **Fast Recovery** -- the reception of 4 selective acknowledgements in a row is used to pre-empt the retransmission timer and force re-delivery of unacknowledged segments.

- **Checksums** -- checksums are supported, but not enabled in this implementation.

4.1.2. States

The following are the primary states in the protocol.



- **CLOSED** -- this is the initial state that occurs when the protocol is not yet available for operation. It also occurs after the protocol has shut down due to the reception of a RST Segment from the remote end, or a local Close Request. An

Open Request in passive mode causes transition to the LISTEN state, whereas an Open Request in active mode causes a SYN Segment to be issued and then a transition to the SYN SENT state.

- **LISTEN** -- this is the state that occurs after an Open Request in passive mode is issued. The protocol is awaiting a SYN Segment from the remote end, and when received it will make the transition to the SYN RCVD state.

- **SYN SENT** -- this is the state that occurs after an Open Request in active mode is issued. A SYN Segment has been sent and the protocol either awaits a SYN/ACK Segment in reply to move to the OPEN state, or a simultaneous SYN Segment in which it replies with a SYN/ACK Segment and moves to the SYN RCVD state.

- **SYN RCVD** -- this is the state that occurs after a SYN Segment has been received from the remote end and a SYN/ACK Segment issued in reply. The protocol is awaiting an ACK Segment in response before it will transition to the OPEN state.

- **OPEN** -- this is the primary state in which data transfer occurs. ACK Segments and EACK Segments are processed containing Data, and Send Requests are issued, and Receive Requests are honoured. The reception of an RST Segment causes a transition to the Close Wait state, as does a local Close Request -- which also prompts the delivery of an RST Segment.

- **CLOSE WAIT** -- when this state is entered, a Close Wait timer is activated. When it expires, the connection moves to the idle CLOSED state.

4.1.3. Events

The following are the primary events of the protocol. The table illustrates whether the events are valid or invalid in a particular state, and the transition states that are possible.

Event/State	**CLOSED**	**LISTEN**	**SYN-SENT**	**SYN-RCVD**	**OPEN**	**CLOSE-WAIT**
Open Request (Active)	**Valid SYN-SENT**	Invalid	Invalid	Invalid	Invalid	Invalid
Open Request (Passive)	**Valid LISTEN**	Invalid	Invalid	Invalid	Invalid	Invalid
Close Request	Invalid	**Valid CLOSED**	**Valid CLOSED**	**Valid CLOSED**	**Valid CLOSE-WAIT**	Invalid

Send Request	Invalid	**Valid**	**Valid**	**Valid**	**Valid**	Invalid
Status Request	**Valid**	**Valid**	**Valid**	**Valid**	**Valid**	**Valid**
Receive Request	Invalid	Invalid	Invalid	Invalid	**Valid**	Invalid
Segment Arrival	**Valid**	**Valid SYN-RCVD**	**Valid CLOSED OPEN SYN-RCVD**	**Valid LISTEN CLOSED OPEN**	**Valid CLOSE-WAIT CLOSED**	**Valid CLOSED**
Retransmit Timeout	Invalid	Invalid	Invalid	Invalid	**Valid**	Invalid
Close Wait Timeout	Invalid	Invalid	Invalid	Invalid	Invalid	**Valid**

- **Open Request** -- the connection record is initialised, and the connection enters either the SYN SENT or LISTEN states. In the case of the former, a SYN Aegment is generated and delivered.

- **Close Request** -- the connection is placed into the CLOSED or CLOSE WAIT states. In the case of the latter, the Close Wait timer is activated and a RST Aegment is delivered.

- **Send Request** -- the provided data is queued in the Tx Queue, and if in the OPEN state is also delivered in an ACK Aegment.

- **Status Request** -- status information about the connection record is shown.

- **Receive Request** -- the Rx Queue is polled to extract the next availale in order segment. It is possible that the segment is not available because it has not arrived yet, in which case NULL is returned.

- **Segment Arrival** -- a significant amount of activity occurs here. In the CLOSED state, the segments are discarded and RST Segments are delivered in reply. In the LISTEN state, we are awaiting the acceptance of a SYN or SYN/ACK Segment to move to the SYN RECV state. In the SYN SENT state, we are waiting for a SYN or SYN/ACK Segment to move to the SYN RECV or OPEN state respectively. In the SYN RECV state, we are waiting for an ACK Segment to move us to the OPEN state. In the OPEN state, we process ACK and EACK Segments to transfer data, and RST Segments to move to the CLOSE WAIT state. In the CLOSE WAIT state, we discard segments.

- **Retransmission Timer Expiry** -- when this timer expires, we examine any packets in the Tx Queue that have an expired timer field. If they do, then we retransmit the segment.

- **Close Wait Timer Expiry** -- when this timer expires, the connection moves from the CLOSE WAIT to the CLOSED state and the connection record is terminated.

5. Glossary

...

6. References

[RFC-0908] Velten D., Hinden R., Sax J., "Reliable Data Protocol". July 1994, Internet Engineering Task Force, Request For Comments 0908. http://globecom.net/(nobg)/ietf/rfc/rfc908.shtml