Internet-Draft February, 1995
ST Working Group
M. Rajagopal and Sharon Sergeant
File: draft-ietf-ST2-state-01.ps Expires: July, 1996

Internet Stream Protocol Version 2 (ST2)
Protocol State Machines - Version ST2+

Status of this Memo

This document is an Internet-Draft. Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its Areas and Working Groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months. Internet-Drafts may be updated, replaced, or obsoleted by other documents at any time. It is not appropriate to use Internet-Drafts as reference material or cite them other than as “work in progress”.

To learn the current status of any Internet-Draft, please check the “lid-abstracts.txt” listing contained in the Internet-Drafts Shadow directories on ds.internic.net (US East Coast), nic.nordu.net (Europe), ftp.isi.edu (US West Coast), or munnari.oz.au (Pacific Rim).

Abstract:

This memo contains a description of state machines for the revised specification of the Internet STream Protocol Version 2 (ST2+) described in RFC 1819. The state machines in this document are descriptions of the ST2+ protocol states and message sequence specifications for normal behavior. Exception processsing issues are defined and discussed for protocol compliance and implementation options.

Editor's Note:

This memo is available both in ASCII format (file: draft-ietf-ST-state-01.txt) and in PostScript (file: draft-ietf-ST-state-01.ps). The PostScript version contains the essential state diagrams and is absolutely required.

TABLE OF CONTENTS

SECTION 1 Introduction 4

SECTION 2 ST Agent Architecture 7

ST Protocol Characteristics 7
The Stream FSM Model 8
ST Agent Roles in an Internetwork 10
The ST Agent Model 10
Origin, Next Hop, Previous Hop and Target Finite State Machines 13
Stream Finite State Machines 14
Externally Communicating FSMs 14
Internally Communicating FSMs 14
Queues between External Communicating FSMs 14
Queues Inside an Agent 15

SECTION 3 Stream Finite State Machines 17

Assumptions 18
State Machine Model Conventions 18
Naming Conventions and Notations 18
Transmissions and Receptions 19
Predicates 19
Normal Behavior versus Exception Processing 20
Context not Represented in Stream FSMs 20
Special Message Types 21
Classes of Response Types 21
Stream State Machines. 22
Origin State Machine (OSM) 22
Next Hop State Machine (NHSM) 26
Previous Hop State Machine (PHSM) 31
The Target State Machine (TSM) 34

SECTION 4 ST Agent FSMs 37

Agent Database Context 37

ST Dispatcher role for incoming Packet-switching, ACKnowledgement and PDU validation 38

ST Dispatcher functions for outgoing Packet switching, timer and retry settings 41

Retry FSM- RFSM for datalink reliability of PDU transmissions 42

Agent , Neighbor and Stream Supervision 46

The MonitorFSM (MFSM) for Agent and Stream Supervision 46

The Nieghbor Detection Failure FSM for Neighbor Management 48

Service Model Interactions 50

SECTION 5 Exception Processing 51

Additional Exception Processing 51

ST Dispatcher detected inconsistencies Reason Codes: 51

Monitor FSM issues with neighbor failure and stream recovery Reason Codes: 52

Retry and Timeout Failures Reason Codes: 52

Routing issues Reason Codes: 52

LRM issue Reason Codes: 53

SECTION 6 APPENDIX 54

Glossary 54

ST Control Message Flow 56

Message Type 56

Response 56

Possible causes for message 57

Internetwork Complexities 60

SECTION 1

Introduction

This section gives a brief overview of the ST protocol terms and the protocol FiniteState Machine (FSM) issues addressed in this document. It is assumed that the reader is familiar with the ST2+ Protocol Specification document listed in [1]. Unless otherwise stated, ST in this document refers to the enhanced ST protocol (ST2+).

ST+ is a connection-oriented internetworking protocol that operates at the same layer as connectionless IP. An ST *stream* is defined as a connection established between an *Origin* sending data to one or more *Targets*. An ST *Agent* is a network node that participates in resource reservation negotiations during stream setup along the path between the Origin and Targets. The resource reservation request is based on a *Flow Specification* sent by the Origin. The FlowSpec provides the basis for the negotiated *Quality of Service* (QOS).

This QOS is only established, monitored and maintained by nodes with ST Agent capabilities. Each hop in the ST stream routing tree is an ST Agent . ST Agents that are one hop away from a given node are called *Previous-Hops* in the *upstream* direction and *Next-Hops* in the *downstream* direction. ST Agent Previous-Hop and Next-Hop Agents are called ST *neighbors*.

Data transfer in the ST stream is simplex in the downstream direction. As such, a single Origin sending data to many Targets is similar to a media broadcast model. However, each ST Agent may simultaneously need to perform Origin, Previous-Hop, Next-Hop and Target functions for a number of different streams. These streams may be part of a conference (as in the telephone model) or *Group* of related streams, such that resource reservation and routing issues may be interrelated. The streams may also be unrelated to each other, but ranked by *Precedence* within an internetwork in the event that limited or changing resources need to be reallocated. Origin applications may request an automatic *Recovery* option in the event of network failure or a *Change* to the QOS after the original setup. Target applications may send a request to a stream ST Agent to allow that Target to *Join* the stream, with or without *Notify*ing the Origin.

Thus, an ST Agent may be required to support a complex web of intersecting streams with competing QOS requirements and changing resource allocations or members. The *ST Service Model* supports the ST protocol and ST QOS features for routing, resource management and packet-switching. This Protocol State Machine document addresses the ST protocol in any ST Agent, regardless of the implementation specifics of the ST Service Model.

Stream Control Message Protocol (SCMP) messages form a request-response protocol where the particulars of the Flow Specification, as well as other *Protocol Data Unit* (PDU) parameters, are interpreted by the chosen QOS algorithms for routing, *Local Resource Management* (LRM) and packet-switching. The ST2+ Specification explicitly defines all required and allowable, functions and sequences of SCMP message operations. The SCMP message types are: *ACCEPT,*

ACK, CHANGE, CONNECT, DISCONNECT, ERROR, HELLO, JOIN, JOIN-REJECT, NOTIFY, REFUSE, STATUS, STATUS-RESPONSE.

An ST Agent will direct incoming SCMP messages to the appropriate FSMs for each stream. An *Origin State Machine* OSM) is associated with every Origin ST application. A *Next-Hop State Machine* (NHSM) is associated with every downstream ST neighbor. A *Previous-Hop State Machine* (PHSM) is associated with every upstream ST neighbor. A *Target State Machine* (TSM) is associated with every Target ST application.

The OSM, NHSM, PHSM and TSM have the same four fundemental states: *IDLE, ESTABLISHED, ADD* and *CHANGE*. The basic transition from IDLE to ESTABLISHED is through a CONNECT request with an ACCEPT response. Additional CONNECT and JOIN requests may result in an ADD stream state, while a CHANGE request would result in a CHANGE stream state. DISCONNECT and REFUSE messages may remove one or more Targets while the stream is in any state.

A *Retry* FSM is used to monitor the datalink reliability between ST Agents.Each SCMP request-response sequence is defined with next hop ACKnowledgement , ERROR, timeout and retry conditions. Such exception processing is designed to resolve incomplete functions during times of network or ST Agent failure.

A *Monitor* FSM is used to manage ST neighbor and stream Recovery issues for all streams managed by the ST Agent. Each ST Agent maintains state information describing the streams flowing through it, and can actively gather and distribute such information.

If, for example, an Intermediate ST Agent fails, the neighboring Agents can recognize this via HELLO messages that are periodically exchanged between the ST Agents that share streams. STATUS packets can be used to ask other ST Agents about a particular stream. These agents then send back a STATUS-RESPONSE message. NOTIFY messages serve to inform ST Agents of additional mangement information.

ST Reason Codes are used to inform other ST Agents of the source and type of problem such that the correct response sequences will be followed. These Reason Codes are inserted in the appropriate SCMP PDUs and available to the ST Agent management functions. Thus an ST Agent not only manages the normal request- response protocol between the Origin and Targets of each stream, but also is actively involved in the detection and distribution of error and QOS implications.

The ST Agent architecture and FSM models contained in this document have been chosen to illustrate a method for an ST2+ protocol implementation. There are many alternative techniques. Every effort has been made to note the relevant tradeoffs between protocol requirements and implementation choices.The atomic components of this model may be rearranged to accomodate platform and implementation issues. Every effort has been made to ensure that the described state tables accurately reflect state transitions of the ST2+ version of SCMP, and that the described state diagrams accurately reflect the state transition tables. If there are discrepancies, the tables take precedence over the diagrams and the protocol specification takes precedence over the tables.

Section 2 : ST Agent Architecture describes the organization of the ST Agent model. Section 3: Stream Finite State Machines describes the OSM, NHSM, PHSM and TSM Section 4: Agent Finite State Machines describes the Monitor and Retry FSMs. Section 5: Exception Processing Issues details the Reason Codes by category.

SECTION 2

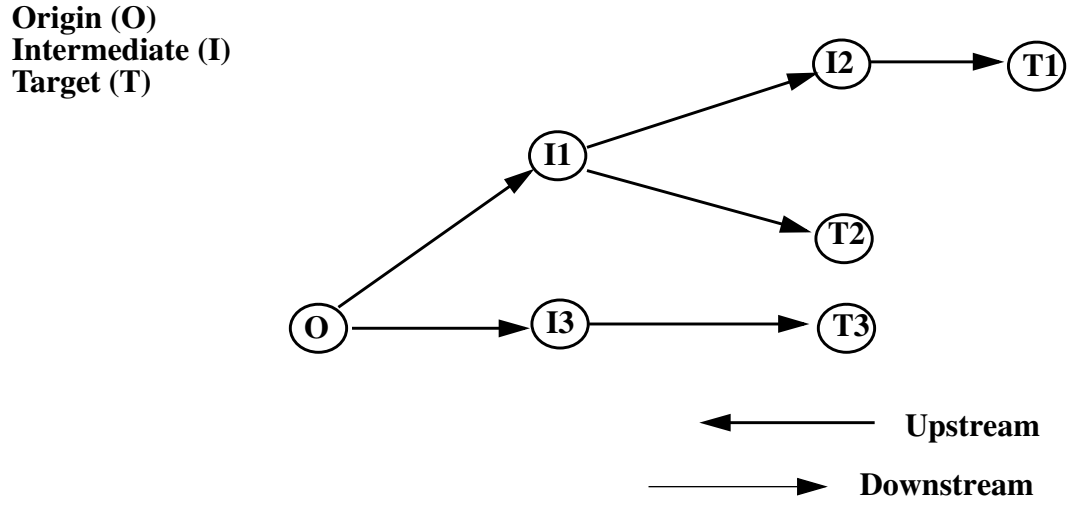
ST Agent Architecture

This section describes the ST Agent Finite State Machines (FSMs). The architectural descriptions are necessarily at a high level and are meant to serve as a guide to the protocol implementer. The state machine models are expected to provide the implementer with useful information such as valid message sequences. The ST2+ Specification provides the fully documented message detail.

2.1 ST Protocol Characteristics

The ST Agent FSM architecture is organized in a hierarchy of ST2 protocol characteristics. The characteristics are modeled as Agent roles (i.e., Origin, Intermediate, Target, Previous Hop and Next Hop), as well as protocol functions (e.g., individual SCMP Request/Response patterns and reliability at both the PDU and Agent level).

Figure 1. ST Agent Roles



Each ST Agent has an ST Dispatcher to filter incoming PDUs, intercept semantic errors and direct valid PDUs to the appropriate queues and FSMs. FIFO queues and a Message Separator/ Combinator in each ST Agent provide additional intra-Agent FSM communications.

Table 1 below shows the Request/Response patterns of each SCMP message name by the Agent type generating the message. The message type may be either a Request, a Response or a local control for the Agent and PDU reliability functions. The message direction can be downstream

(towards a Target) or upstream (towards the Origin). The Monitor FSM and the Retry FSM manage network, Agent and link reliability status with the local control SCMP messages..

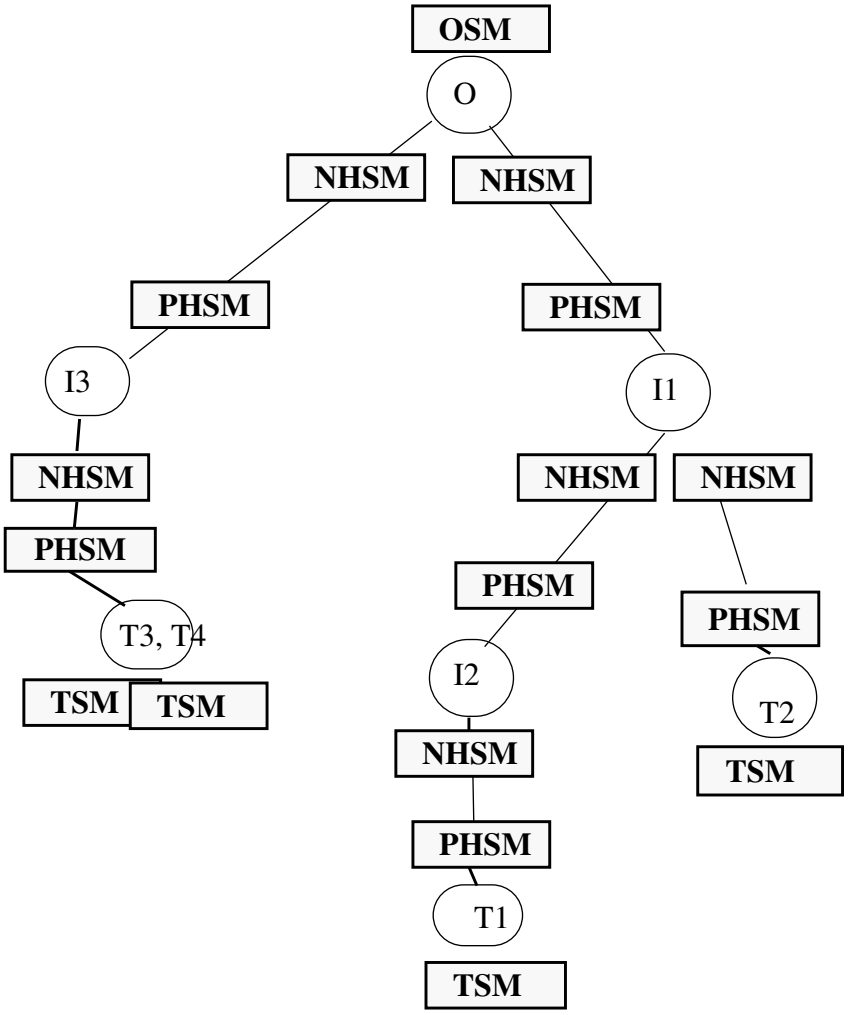
Table 1: Request/Response Patterns

Message Name	Generation by Agent Type	Message Type	Direction
ACCEPT	**Target**	**Response**	**Upstream**
ACK	All	local control	Both
CHANGE	**Origin**	**Request**	**Downstream**
CONNECT	**Origin, Intermediate (Join)**	**Request**	**Downstream**
DISCONNECT	**Origin, Intermediate**	**Request**	**Downstream**
ERROR	All	local control	Both
HELLO	All	local control	Both
JOIN-REJECT	**Origin, Intermediate**	**Response**	**Downstream**
JOIN	**Target**	**Request**	**Forwarded**
NOTIFY	All	Response local control	Both
REFUSE	**Intermediate, Target**	**Response, Request**	**Upstream**
STATUS	All	local control	Both
STATUS-RESP	All	local control	Both

2.2 The Stream FSM Model

Figure 2 below shows the relationship between the ST Agents and FSMs in each stream.

Figure 2. Stream FSM Model

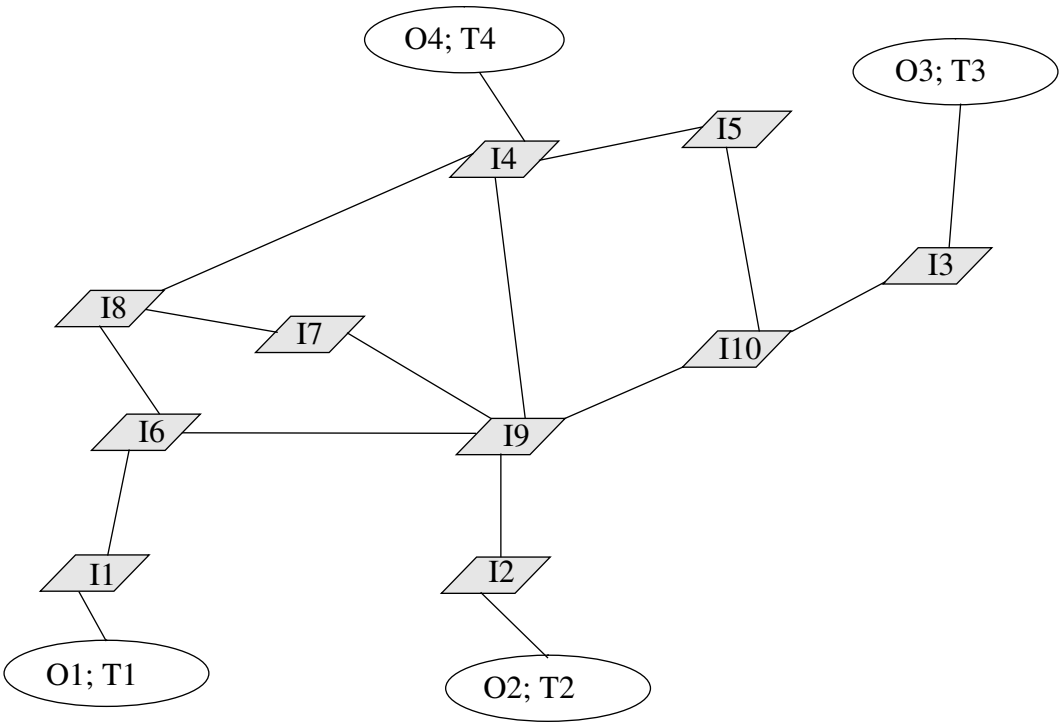


The Origin State Machine (OSM) provides communications between an Origin application and one or more Next Hop State Machines (NHSM).

An Intermediate Agent's Previous Hop State Machine (PHSM) has communications with an Origin Agent's Next Hop State Machine (NHSM) through their common network link and the respective Agent's ST Dispatchers. In the same fashion, an Intermediate Agent's Next Hop State Machine (NHSM) has communications with Intermediate and/or Target Agent's Previous Hop State Machines(PHSM).

The Target State Machine (TSM) provides communications between a Target application and a Previous Hop State Machine (PHSM) in the Target Agent.

Figure 3. Internetwork Diagram of ST Agent Roles



2.3 ST Agent Roles in an Internetwork

The internetwork diagram of ST Agents (Figure 3) indicates the Origin (O), Intermediate (I) and Target (T) roles of each ST Agent in a conference with 4 Origins. The Intermediate Agents I1, I2, I3 and I4 are each neighbors of an ST Agent acting as both an Origin and a Target for the other Origins. Each Origin is sending data in one outgoing stream to three Targets, and simultaneously recieving data on 3 incoming streams from the other Origins.

All ST Agents in this illustration have multiple ST neighbors, streams and interfaces to manage. Each ST Agent may be required to manage multiple FSMs for any one stream, as well as all competitive, intersecting streams in the internetwork topology.

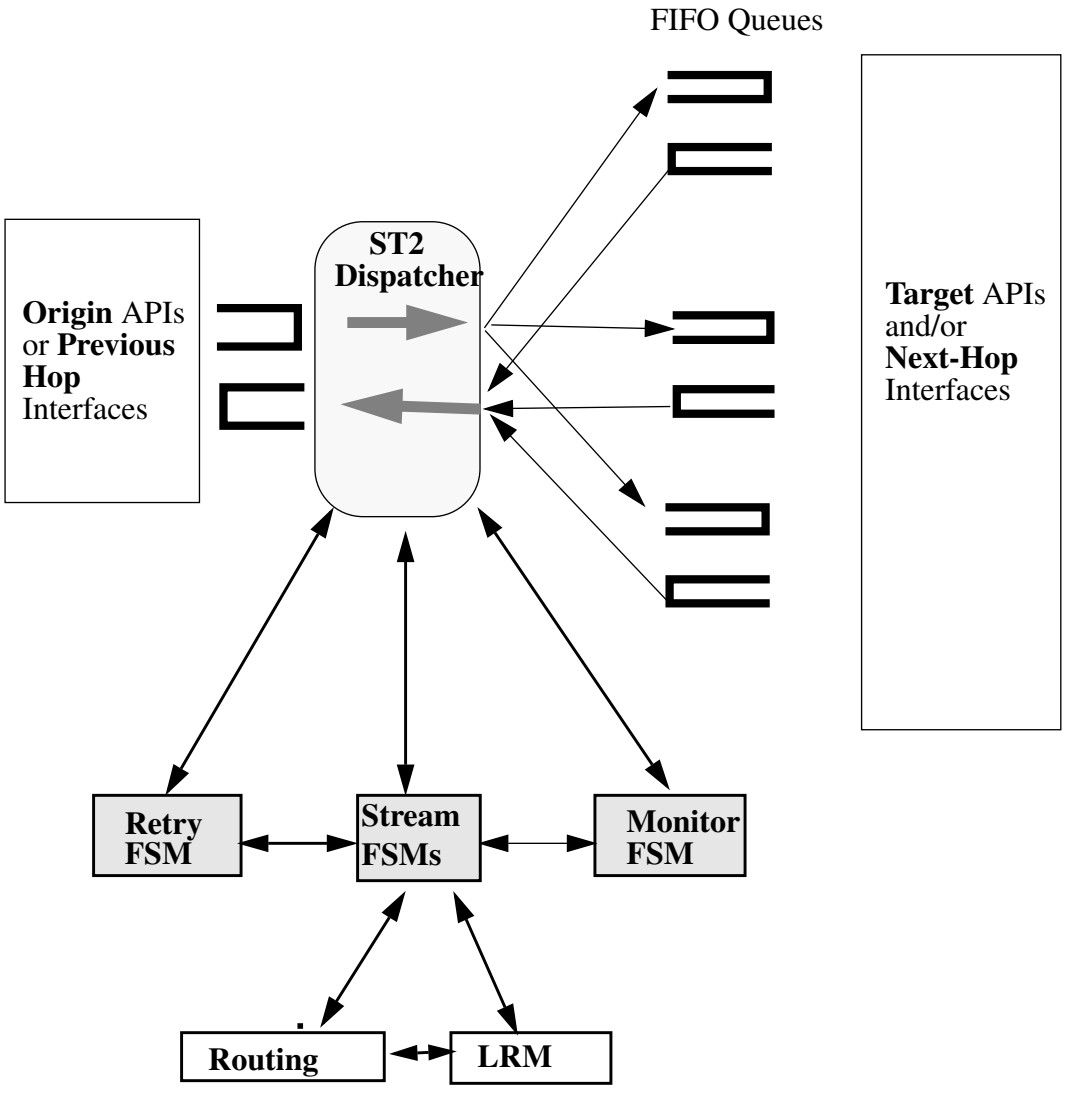
ST neighbor communications and SCMP exception processing can be used to create a first line of defense for the ST stream FSMs. Normal operations for stream FSMs may be protected and simplified. Network errors and conflicting message sequences can be filtered out of the fundemental stream state transitions.

2.4 The ST Agent Model

In Figure 4 , an ST Agent is depicted with an ST Dispatcher sending and receiving ST PDUs from interface queues (and PDU surrogates from application interfaces), representing a high order ST

message management scheme. This dispatcher unpacks or forwards incoming PDUs, and creates and forwards outgoing PDUs.

Figure 4. ST Agent Model



The forwarding of data or the forwarding of certain command sequences that are not following a negotiatied QOS path (i.e., JOIN and/or JOIN flooding messages) requires a packet-forward-

ing mechanism, separate from the stream operations that unpack, interpret or create PDUs.The efficient packet switching of ST PDUs through Intermediate hops is the main reason for this filtering priority.

A first level of filtering is designed to determine whether the incoming PDU (or PDU surrogate) is data or one of the JOIN sequences where the destination is, in fact, another ST Agent or an application. Such PDUs are then forwarded to the destination., whether a resident Target or for replication to multiple Next-hop Targets.

The ST PDU validation and delivery functions manage information about the the messaging success or failure, i.e. the Retry, timeout, ERROR , or ACK status of messages. This information concerns datalink, Agent and network reliability. Stream state transitions may occur as a result.

Many Retry and Monitor FSM transitions may occur while any particular stream state exists. The Monitor FSM interprets and sends messages with information relevant to multiple streams , i.e., HELLO, STATUS, STATUS-RESPONSE , NOTIFY.

Second-level filtering occurs when an ST Agent validates incoming SCMP PDUs and sends the required ACKnowledge (or ERROR PDU, if there are semantic errors) to the originator of the incoming PDU. Conversely, incoming ACKnowledgement and ERROR PDUs trigger the Retry FSM, where the timeout and retry values are updated or a signal is generated to the appropriate stream FSM for specialized exception processing.

All SCMP PDUs, except ACK, ERROR, HELLO, STATUS and STATUS-RESPONSE, require an ACK from the next hop Agent upon receipt. An Agent or datalink failure may be detected by either the Retry or the Monitor FSM. A signal is then sent to the appropriate stream FSMs. The Monitor FSM then manages a reCONNECT sequence for all streams that have specified this Recovery option.

Once an ST Dispatcher has validated and filtered the PDUs, the stream SCMP messages are separately queued into Requests (CONNECT, CHANGE, JOIN) and Responses (ACCEPT, DISCONNECT, JOIN-REJECT, REFUSE). Requests must wait for the completion of any preceding Requests for the same stream, while Responses must be handled immediately without regard to Request state transitions or queues.

The Request and Response queues are directed to the Origin (OSM), Next Hop (NHSM), Previous Hop (PHSM) and Target (TSM) state machines. These state machines are referred to as the stream state machines, rather than the Agent state machines.

The stream state machines are designed to focus on normal (typical) behavior rather than all pathological cases. Error control and recovery in the architecture (i.e., ST Dispatcher filters, Monitor and Retry FSMs) provide a firewall against many problems in the stream FSMs.

However, when the architecture of a particular Agent platform has ST intra-Agent communications that are actually between multiple processors, the Next-hop and Previous-hop FSM communications may require the concept of multiple ST Agents within what might otherwise appear

to be one ST Agent. Thus the rationale for the filtering and queueing of the SCMP messages , not just the particular method of illustration, is very important to the ST Agent Model.

The convention of discussing issues in terms of individual stream state transistions will be used throughout Section 3 (Stream FSMs) as a way of simplifying the discussion. Section 4 (ST Agent FSMs) and Section 5 (Exception Processing) will provide more detail about the architecture, filtering and queues used with the stream FSMs, such that network failures can be managed with competing and intersecting streams.

2.5 Origin, Next Hop, Previous Hop and Target Finite State Machines

Communicating Finite State Machine (CFSM) models have been extensively used in the trade to formally describe protocol behavior [2]. Many variations of the basic CFSM model exist and our model is also a variation of the basic model. Our model uses the basic CFSM model with FIFO queues combined with predicates. The model describes the ST protocol behavior and consists of ST SCMP messages along with a number of predicates. These predicates are not part of the formal ST Protocol specifications but are useful mechanisms that simplify the state machine specifications

Origin, Intermediate and Target ST Agents in Figure 2 are all modeled separately. Because a stream diverges in a tree-like graph, every Intermediate ST Agent has to communicate with one upstream ST Agent and one or more downstream Agents. An Intermediate Agent will therefore have exactly one PHSM and one or more NHSMs for each stream. Note that, it is possible to have more than one NHSM per physical interface, when that interface has more than one Agent on the associated communications link.

The state machine model architecture at an Origin is similar to the state machine architecture at an Intermediate (Fig. 2). The Origin may have one or more NHSMs. There is no PHSM in this case. However, in the place of the PHSM there is an Origin State Machine (OSM) which interfaces with the application. An OSM is a special case of the PHSM.

The Target is modeled with one PHSM (Fig. 2). There are no NHSMs in this example. However, in the place of a NHSM there are one or more Target State Machine (TSM) that interface with the application. The TSM is a special case of the NHSM.

Because the role of each ST Agent (Origin, Intermediate, or Target) is different, the finite state machine models are not identical. However, the model for communication between FSMs inside or outside an Agent is uniform.

Consider a stream topology shown in Figure 2. The figure shows an ST Origin (O) connected to 2 Intermediate Agents (I1 and I3). I1 is also connected to I2 and a target T2. I2 is connected to Target T1 and I3 is connected to Targets T3 and T4.

The Origin is modeled with one OSM and 2 NHSMs (one per next hop). Each Target is modeled with one PHSM and one or more TSMs. I1 and I3 are both modeled with one PHSM and 2 NHSMs; I2 is modeled with one PHSM and 2 NHSMs.

2.6 Stream Finite State Machines

2.6.1 Externally Communicating FSMs

Communication between two ST agents is External Communication and always happens between a NHSM and a PHSM pair (see Figure 2). Note that, in the case of Origin and Target Agent as direct neighbors, it is possible for a Target to be directly connected to an Origin . It is also possible that one Target Agent is an Intermediate Agent for another Target. in which case an Agent will have a PHSM communicating with a TSM and one or more NHSMs.

2.6.2 Internally Communicating FSMs

Communicating entities inside an ST Agent is different for each Agent type, i.e., Origin, Intermediate or Target. However, all FSMs inside an Agent communicate via a Message Separator/ Combiner box (MS/C). The function of the MS/C box is described later in this section.

Internal Communication within the Origin occurs:

- between the OSM and an Upper Layer module and
- between the OSM and one or more NHSMs via a MS/C box (Note that the NHSMs themselves do not communicate with each other)

Internal Communication within a Target occurs:

- between one or more TSMs and an Upper Layer module and
- between the TSM and a PHSM via a MS/C box

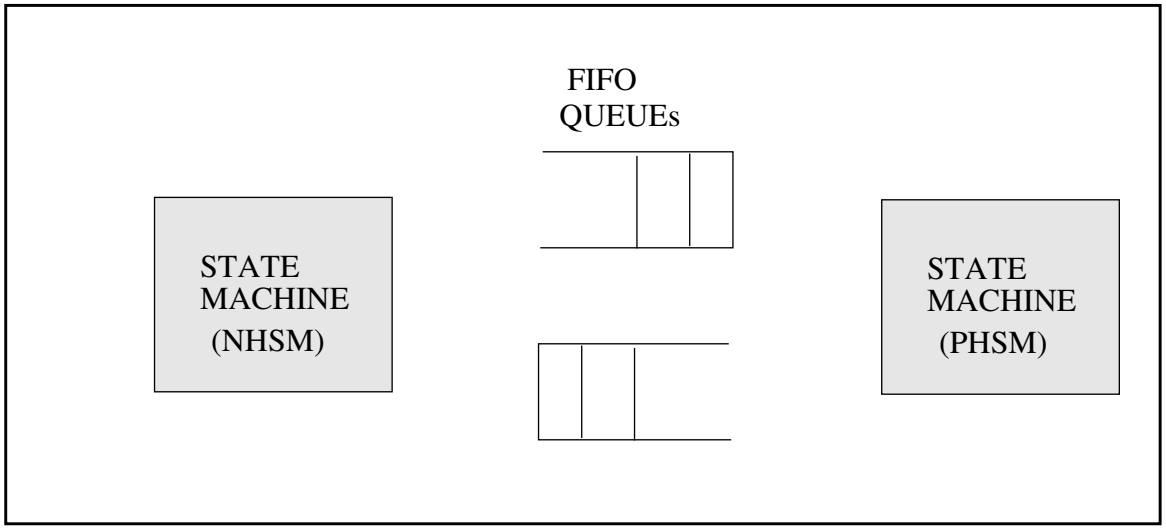
Internal Communication within an Intermediate Agent occurs:

- between a PHSM and one or more NHSMs via a MS/C box (Note that the NHSMs themselves do not communicate with each other)

2.7 Queues between External Communicating FSMs

For the purposes of modelling, assume that messages are filtered and queued in FIFO queues for the case of external Communicating FSM pairs, i.e. between any two ST Agents. However, as indicated in the previous discussion and diagrams of the ST dispatcher and filtering hierarchy, it is somewhat more complex in reality. The concept shown below in Figure 5 allows the discussion of the inter-Agent and intra-Agent state machines to focus on the stream FSM issues without regard to message and neighbor management issues.

Figure 5. Implicit Queues between an External Communicating FSM pair

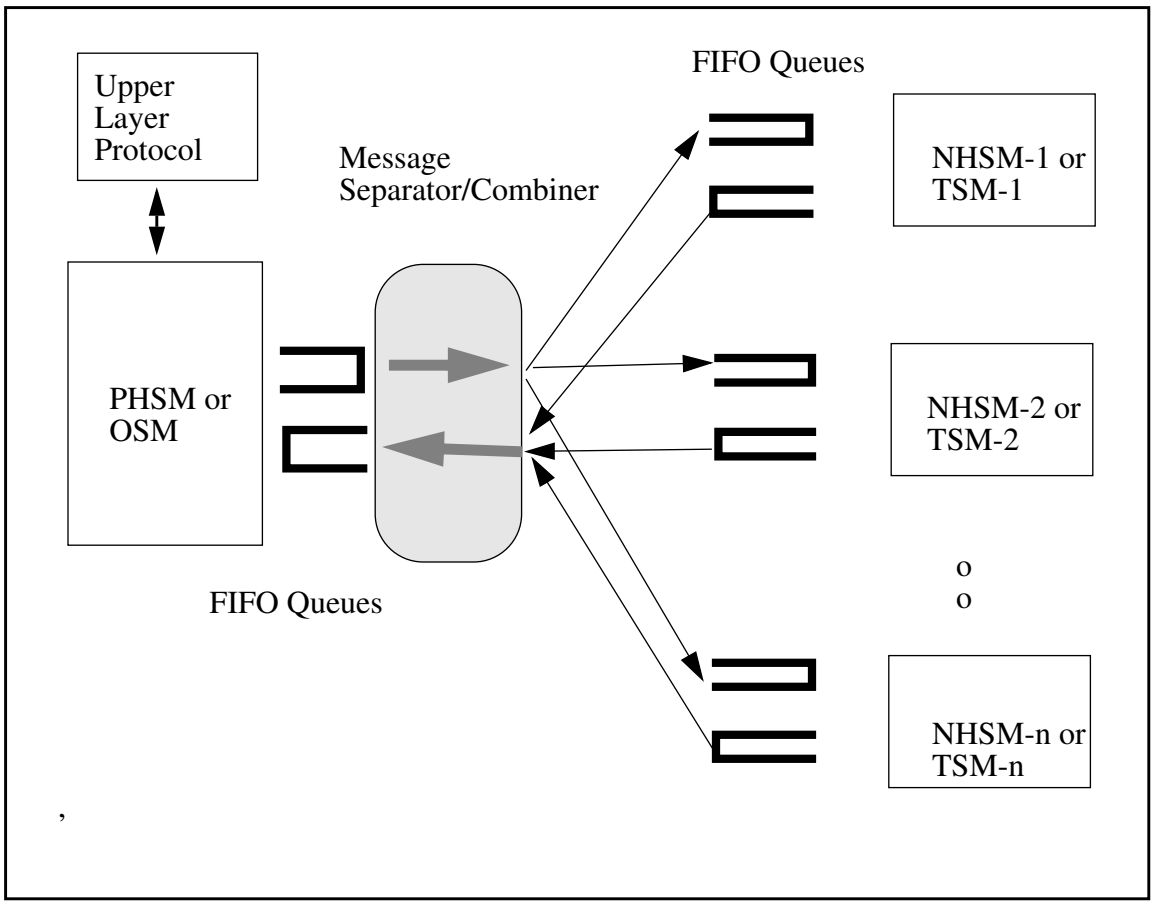


2.8 Queues Inside an Agent

Each Agent is modeled with at least 2 state machines for each stream. These state machines also need to communicate just like the external communicating FSM pairs described above. The queue model in this case also requires filtering mechanisms. This model requires a message Separating and Combining function shown as Message Separator/Combiner (MS/C) box in Fig. 6.

Figure 6 pictorially describes the multi-stage FIFO queue model for an Agent. Implicit FIFO queues are assumed between the PHSM and the MS/C, and also between the MS/C and one or more NHSMs. Use of such FIFO queues eliminates the need for a separate synchronizing state machine that would normally be required to synchronize the flows .

Figure 6. Queues between Internal Communicating FSMs inside an Agent



The function of this Message Separator/Combiner box is many:

- Performing a multicasting function by replicating an OSM or PHSM message and sending them to different NHSMs or TSMs
- Combining messages coming from different TSMs or NHSMs and sending them to the appropriate OSM or PHSM

Designing the Agent to contain separate upstream and downstream state machines (PHSM and NHSMs respectively) with FIFO queues as shown in Fig.6, offers several benefits:

- It simplifies the Agent design considerably by separating the neighbor upstream and downstream communications
- Use of FIFO queues simplifies the Agent management since no other synchronization mechanisms need to be used to streamline messages flowing through the Agent.

SECTION 3

Stream Finite State Machines

Each ST Agent must maintain state for each stream supported by that Agent. There are many ways to represent the state that must be maintained by Agents. This section presents the OSM, NHSM, PHSM and TSM as a reference set of state machines.

Implementations may support machines based on this section or may even support a completely different set of state machines. These stream FSMs represent normal operations for the stream request-response scenarios without regard to the functions performed by the Retry and Monitor FSMs. The model assumes that a data engine separate from the control engine exists.

This section represents stream state through four state machines. The defined machines are:

- The Origin State Machine, or OSM. It represents the state of a stream at the Origin Agent.
- The Next-Hop State Machine, or NHSM. It represents the state of the stream for Targets reached via a particular next-hop.
- The Previous-Hop State Machine, or PHSM. It represents the state of a stream at an Intermediate Agent or a Target Agent. The OSM is essentially a special case of the PHSM, where the delivery of SCMP to the Origin is via an API.
- The Target State Machine, or TSM. It represents the state of a stream for a particular target application at the Target Agent. This state machine is essentially a special case of the NHSM, where there is only a ever a single Target per TSM and delivery of SCMP to the Target is via an API.

A number of NHSMs related to the same stream, could conceivably all be running in parallel -one for each next hop. In some cases, where there is a network-layer multipoint link (e.g., ethernet), it is even possible to have more than one NHSM associated with the same physical interface.

A Message Separator/Combiner (MS/C) box separates all downstream messages modifying the Targetlist and placing them in the respective NHSM FIFO queues. The MS/C box also functions as a combiner of messages flowing up stream. In this role it multiplexes all local messages and places them in the PHSM FIFO queues. Note that the MS/C relies on separate routing and LRM functions to determine the appropriate separation since route and resource computation is not part of ST protocol. Full-duplex FIFO queues are assumed between the MS/C box and PHSM, and also between the MS/C box and the NHSMs.

The multi-machine Agent model breaks the complexity that results with only one large model with the aid of the FIFO queue buffers and a MS/C box. The FIFO queues eliminate the need for a separate synchronizing state machine while reducing the complexity. The MS/C reduces the explicit next-hop identification modelling that would otherwise be required..

The Intermediate Agent PHSM always communicates with a NHSM on the upstream side and the NHSM always communicates with a PHSM on the downstream side.

3.1 Assumptions

Some basic assumptions were made as part of the development of the enclosed state machines. These included:

- All state machines exist as part of an ST Agent and that the Agent will instantiate state machines as needed to represent state on a per stream basis.

- The ST Agent implements logic that unpacks incoming SCMP packets, validates the contents, updates the Agent databases and routes the message signal to the appropriate stream and it's associated state machine.

- Detection and handling of messages that are broken, duplicates, or not valid for a particular stream state does not affect stream state and is not represented in the state machines. The mechanisms to prevent such misleading signals to individual state machines are described in the Architecture, Agent FSM and Exception Processing Sections.

- All reliable delivery of intra- and inter-Agent SCMP messages is handled by the ST Agent independent of the described state machines except in the case where stream state is dependent on the outcome of the message delivery.

- All communication within the same Agent should follow the same Request Response paradigm as inter-Agent messages in order to be as reliable as SCMP communications. This assumes that all API communications and intra-agent communications recreate the reliability available with the ACK, timeout and retry paradigms. It is an implementation specific choice.

- The described state tables accurately reflect state transitions of the ST2+ version of SCMP. The described state diagrams accurately reflect the state transition tables for all states, input trigger events and state transitions. Output events are not shown in the diagrams, but are detailed in the tables. If there are discrepancies, the tables take precedence over the diagrams and the protocol specification takes precedence over the tables.

- API notations for the Origin and Target ST applications are shown to illustrate the OSM and TSM interactions. The actual defintion of an ST application API is outside the scope of this document.

3.2 State Machine Model Conventions

3.2.1 Naming Conventions and Notations

All state names are in bold and start out with a capital letter followed by the lower case letters. All message names are in capitals usually prefixed with a + or - sign .All messages with special response conditions have suffixes indicating the condition, i.e. **_last, _all, _change**. Predicates are in bold and lower case string.

Tables show states, events, output, and transitions.

Diagrams show states, events and transitions. Initial states are indicated by an asterisk “*”.

Messages that trigger events are proceeded by a plus sign “+”.

Outputs are proceeded by a minus sign “-”.

Transitions are represented by arrows in the diagrams and by “>>” in the tables.

3.2.2 Transmissions and Receptions

In all the state machine models, the standard convention of prefixing message transition labels, with a + or - symbol, is used to explicitly indicate a transmission and reception respectively. The prefixes are not part of the message syntax. In addition, the tables will show both transmitted and received messages, but the diagrams show only recieived messages. This simplifies the diagrams, but the tables must be referenced for the message outputs.

3.2.3 Predicates

State transitions are sometimes dictated by conditions outside the scope of the protocol specification. Predicates are mechanisms that allow such transitions to occur. For example, terminating a protocol session (a result of many conditions) should allow the Agent to transition to either the initial state or some idle state. This decision is of course Application-initiated but a means should nevertheless exist to allow transitioning to the correct state. In the ST protocol there is no message which accomplishes this.

Predicates allow a state machine to express conditions and control not explicitly possible with the protocol messages. Generally speaking, they add clarity to the state diagram while reducing the complexity in terms of states. The addition of control predicates allows user defined change of states. Predicates are meant to give hints to the protocol implementer and are not part of the ST protocol. A Glossary in the Appendix can be used to check the explicit meaning of each message or predicate

API predicates are used to illustrate the OSM and TSM interactions with ST applications. A predicate with an **api_** prefix shows an API message coming into the FSM. A predicate with an **_api** suffix shows a message being sent to the API from an FSM.

For example, an **api_open** and an **api_close** predicate are defined for the OSM as a means to transfer control to and from the **Init** state. The Origin application may open or maintain a stream in the **Establd** state without any Targets being active or in the TargetList.

NHSM, PHSM and TSM state machines have corresponding **nh_open, ph_open** and **tsm_open** predicate definitions to allow the Agent to bring the state machine into the **Establd** state when the Agent is ready to process the initial CONNECT. Unlike the OSM, these state machines return to the Init state when all Targets have been deleted, so no predicate is required to close the NHSM, PHSM or TSM.

Some triggers and events are combinations of implicit and explicit message conditions. This is particularly true for the **RetryTimeout** mechanisms, as well as the requirement that responses from all Targets in a Request be complete before the Request state can complete. See Section 3.3 below.

No attempt has been made to illustrate the API interactions with Routing and LRM functions. The results of these interactions affect both how the TargetList is partitioned and what Reason Code has been included in a DISCONNECT or REFUSE to indicate the source of a Request failure. ST Agent management of such failures is discussed in Section 4 ST Agent State Machines and Secton 5 Exception Processing.

3.3 Normal Behavior versus Exception Processing

The stream FSMs describe the protocol under normal conditions. In general, the architecture is designed to protect these stream FSMs from error conditions handled in the Monitor and Retry FSMs. The SCMP messages STATUS, STATUS-RESPONSE, NOTIFY and ERROR, as well as detailed error handling will be discussed in both Section 4 and Section 5. Otherwise, if a core message transition is not specified from a state, it implicitly means that this message is not allowed from that state.

3.3.1 Context not Represented in Stream FSMs

The OSM, NHSM, PHSM and TSM diagrams and tables in this section cannot represent state as the complete context of the stream. There are context issues that are handled by the ST Agent Dispatcher, Retry and Monitor FSMs and ST Agent database implementation. These stream FSMs define the atomic elements of stream setup, maintenance and teardown.

The G-bit (all Targets), the S-bit (stream Recovery), the I-bit and E-bit (CHANGE stream teardown risk) involve combinations of FSM and stream database interactions. The implementor must consider the best way to manage these conditions with the other elements of the ST Service Model.

Stream Recovery by the Monitor FSM is modeled such that the reconnection heuristics are outside of the basic CONNECT functionality in the stream FSMs. The Monitor FSM initiates stream teardown, and then initiates reCONNECT sequences.The individual stream FSMs are not directly concerned with the Recovery option.

MTU size limitations may cause multiple SCMP PDUs for the same transaction or an SCMP propagation failure. This type of problem is managed by the Dispatcher and Retry FSM filtering.

Another issue not specifically addressed in this section is the partitioning and management of the TargetList according to the NHSM and ST Agent neighbor associated with each Target or set of Targets.

3.3.2 Special Message Types

In addition to the described predicates for API transactions and state transitions, there are signals from the Retry FSM for ACK failures, a signal for the timer expiration for the End-to-End Response to a Request and special conditions for a REFUSE Response to a CHANGE.

The Retry FSM issues a **RetryTimeout** signal when no ACK has been received for a Request after the configured number of retries have been attempted. This signal is an implicit REFUSE to appropriate PHSM or NHSM.

In the following FSM explanations, you will note that a **RetryTimeout** is an indicated signal only to the NHSM and the PHSM. The NHSM and PHSM provide the inter-Agent communications for the stream FSMs. A **RetryTimeout** is generated by the Retry FSM and forwarded to the appropriate PHSM or NHSM, and that FSM then generates the appropriate DISCONNECT and REFUSE messages as intra-Agent communications. For example, an OSM would receive a REFUSE with Reason Code 41 RetransTimeout as the result of an NHSM receiving a **RetryTimeout.**

Once an Origin (or Agent acting as an Origin) receives an ACK to a Request in the Retry FSM, the End-to-End Response timer is set for the maximum time to wait for Responses to this Request. If this End-to-End timer expires before a Response has been recieved, the **E2ETimeout** becomes an implicit **REFUSE f**or all Targets that have not yet Responded. The Retry FSM communicates this failure to OSM (or PHSM, in the case of an Agent acting as Origin) as an **E2ETimeout**. The OSM issues the appropriate messages to the API and NHSM.

If a CHANGE request is made with the I-bit set, the LRM may risk losing the existing resources to allocate the requested resources. If the I-bit is not set, application does not want to risk losing the current resources for the sake of a CHANGE. Thus when a REFUSE to a CHANGE is recieved, and the E-bit is zero, it means the REFUSE will result in stream teardown. This is the normal result of a REFUSE. However, if the the E-bit is set, it is a **REFUSE_CHANGE**, indicating only that the CHANGE could not be completed, but the that the stream still has the original QOS resources.

3.3.3 Classes of Response Types

The ST2+ protocol requires that all Responses be received from all Targets in a TargetList before the Request state transition may be completed and any other Request may be processed. The protocol, however, allows immediate processing of all DISCONNECT and REFUSE messages whether or not they are not initiated by the current REQUEST. These requirements result in the need to differentiate three classes of Responses.

The first class is a Response that does not have any signifigance for state change, where such Responses are not specifically either the last one required to complete the TargetList of the current Request, nor a deletion of the last of all Targets associated with that stream's FSM. Completion of the Responses for the current TargetList is the second class. Removal of the last of all Targets for that stream's FSM is the third class.

All Responses in the second and third class are defined by predicates that identify the message type with a suffix for either **last** (class 2) or **all** (class 3). All API references are illustrative and are not intended to fully define the application interface.

Class 1 Responses:

api_accept, api_disconnect, api_refuse, ACCEPT, DISCONNECT, REFUSE, REFUSE_CHANGE and RetryTimeout indicate that an individual TargetList member has signaled a Response. The **api_disconnect, api_refuse, DISCONNECT and REFUSE** messages may also be a Request to delete a Target(whether or not it is in the current TargetList of an Add or Change state transaction). Individual and/or Global Target deletion may occur at any time, but any Global (G-bit set) Target Response or deletion Request falls into one of the second two classes.

Class 2 Responses:

api_accept_last, api_disconnect_last, api_refuse_last, ACCEPT_LAST, DISCONNECT_LAST, REFUSE_LAST, REFUSE_CHANGE_LAST, RetryTimeout_last, E2E_Timeout_last are only relevant to the current stream Request and refer to the completion of the Request state by occurring as the Response that incidently completes the TargetList .

Class 3 responses:

api_disconnect_all, api_refuse_all, DISCONNECT_ALL and REFUSE_ALL refer to the Requests or Responses that remove the last active Target from that FSM for that stream.

These classes delineate the asynchronous Request/Response activity that may occur. Network conditions may result in interruptions of any stream FSM operation.

The OSM, NHSM, PHSM and TSM diagrams and tables in this section define stream state as it relates to atomic setup and teardown functions. Every attempt has been made to delineate the atomic SCMP request-response specifications such that implementors may reorganize the Agent architecture to address implementation-specific issues.

3.4 Stream State Machines.

3.4.1 Origin State Machine (OSM)

The Origin State Machine (OSM) communicates with one or more NHSMs. The OSM also talks to the Upper Layer module via primitives. This OSM to Upper Layer Interface is outside the scope of this document, but examples of API predicates are illustrated in the diagrams and tables. All ST Dispatcher and MS/C Box diagrams have indicated that API messages could be included. The actual mechanism used for API communications should be decided by implementation factors.

The OSM consists of a small number of states: **Init, Establd, Add and Change.**

Init: The initial state is called **Init**. An **api_open** predicate moves the control to the **Establd** state. An **api_close** is required to return the stream to the **Init** state.

Establd: The **Establd** state is the stable state from which all **api_connect**, JOIN and **api_change** requests may cause a transition to the **Add** or **Change** states. All Requests that occur while a stream is in either an **Add** or **Change** state will be queued up until the stream returns to the **Establd** state. Data transfer may occur to established Targets. The removal of Targets from previous operations or current operations may occur in the **Establd, Add or Change** states with an **api_disconnect** or a REFUSE.

It is possible for an Application at the Origin to add new Targets to an existing stream any time after the stream has been established. A JOIN message received by an OSM indicates that the Origin Agent happens to be the first Agent for that stream in the path between the JOIN originator and the Origin.

JOIN messages from potential Targets require the authorization process to determine if the JOIN will be allowed. The OSM then issues either a JOIN-REJECT message or a CONNECT message. If this validation is complete and the stream JOIN option allows authorization to be completed,the ST Agent at the Origin transitions to the **Add** state and then issues a CONNECT message that contains the SID, the FlowSpec, and the TargetList specifying the new Target, waiting an ACCEPT or REFUSE response.

If this is not the case, a JOIN-REJECT message is sent to the Target with the appropriate ReasonCode (e.g., JoinAuthFailure, DuplicateTarget or RouteLoop). Issuing a JOIN-REJECT brings the OSM back to the **Establd** state.

Add: Once in the **Establd** state the API may issue an **api_connect**. A transition to **Add** will create a CONNECT message that is placed in the FIFO queue between the OSM and the MS/C box. The CONNECT message contains the SID, an updated FlowSpec, and a TargetList. The MS/C box will then make a copy of the CONNECT message, partition the Targetlist parameter and place it the NHSMs queues.The spliting (or separating) information is derived from the implementation's routing and LRM functions.

Once in the **Add** state the OSM waits to get ACCEPT or REFUSE responses. The stream will not transition back to the **Establd** state until all Targets have responded. A REFUSE may be generated by the local Routing or LRM functions, NHSM or Monitor FSMs as well as any Agent in the path between the Origin and the Target. Normal operations in the OSM treat all types of REFUSE responses to a CONNECT in the same manner. The Monitor FSM will manage the Recovery reCONNECT analysis and may also be expanded to include other ST Service Model functions.

The OSM will record the status of each response from each Target. As each ACCEPT is received, the OSM updates its database and records the status of each Target and the resources that were successfully allocated along the path to it, as specified in the FlowSpec contained in the ACCEPT message. The Application may then use the information to either adopt or terminate the portion of the stream to each Target. When either an ACCEPT or REFUSE from all Targets has been received at the Origin, the stream state returns to **Establd** and any additional queued up requests may then be processed**.**

Figure 7. **Origin State Machine (OSM)**

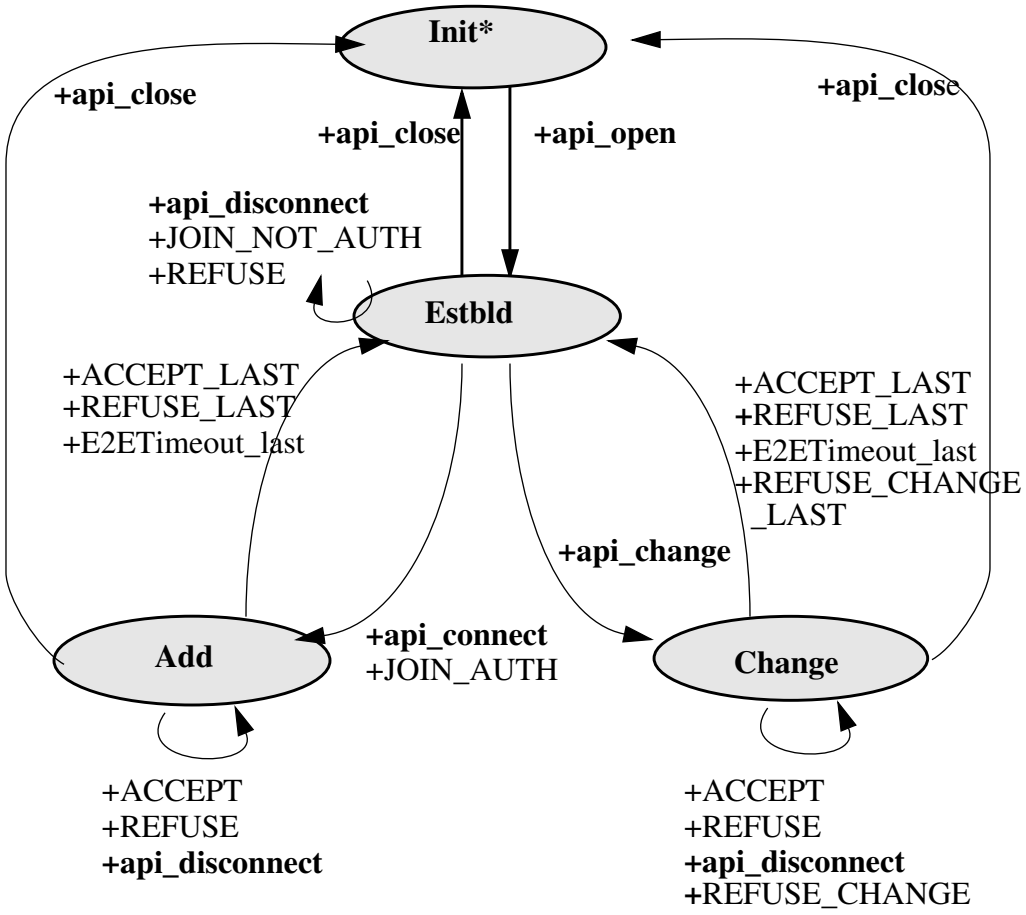


Table 2: OSM

OSM	Init	Estbld	Add	Change
+ACCEPT	-	-	>> Self -accept_api	>> Self -accept_api
+ACCEPT_LAST	-	-	>> Estbld -accept_api	>> Estbld -accept_api
+JOIN_AUTHORIZED	-	>> Add -CONNECT	>> Self -queue	>> Self -queue
+JOIN_NOT_AUTHORIZED	-	>> Self -JOIN_REJ	>> Self -queue	>> Self -queue

Table 2: OSM

OSM	Init	Estbld	Add	Change
+REFUSE	-	>> Self -refuse_api	>> Self -refuse_api	>> Self -refuse_api
+REFUSE_CHANGE	-	-	-	>> Self -refuse_ change_api
+REFUSE_LAST	-	-	>> Estbld -refuse_api	>> Estbld -refuse_api
+REFUSE_CHANGE_LAST	-	-	-	>> Establd -refuse_ change_api
+api_open	>> Estbld	-	-	-
+api_change	-	>> Change -CHANGE	>> Self -queue	>> Self -queue
+api_connect	-	>> Add -CONNECT	>> Self -queue	>> Self -queue
+api_disconnect	-	>> Self -DISCON	>> Self -DISCON	>> Self -DISCON
+api_close	-	>> Init -DISCON	>> Init -DISCON	>> Init -DISCON
+E2ETimeout_last	-	-	>> Estbld -DISCON -refuse_api	>> Estbld -refuse_ change_api

Once an ACCEPT is received by the OSM, the path to the Target is considered to be established and the ST Agent is allowed to forward the data along this path. When a REFUSE reaches the OSM, the OSM notifies the Application that the Target is no longer part of the stream. If there are no remaining Targets, the Application may wish to terminate the stream or keep the stream active to allow stream joining.

To ensure that all Targets receive the data with the desired quality of service, an Application should send the data only after the whole stream has been established. Depending on the local API, an Application may not be prevented from sending data before the completion of all stream Targets.

For each new Target in the TargetList, processing is much the same as for the original CONNECT. The CONNECT is acknowledged, propagated, and network resources are reserved. However, it may be possible to route to the new Targets using previously allocated paths or an existing multicast group. In that case, additional resources do not need to be reserved but more next-hops might have to be added to an existing multicast group. These issues are managed by the implementation of

the ST Service Model and stream state transitions remain the same. Intermediate or Target ST Agents that are not already nodes in the stream behave as in the case of stream setup.

The OSM may issue a DISCONNECT when an **api_disconnect** is received. This message may be processed in any state. The OSM then records this fact and appropriately updates its database.

A REFUSE message may arrive at the OSM asynchronously at any time.This message is sent as a result of an Intermediate Agent failure or a Target leaving a stream.

Change:The Application at the Origin may wish to change the FlowSpec of an established stream. To do so, it informs the ST Agent at the Origin of the new FlowSpec and of the list of Targets associated with the change with an **api_change**. The Origin then issues one CHANGE message with the new FlowSpec per next-hop and sends it to the relevant next-hop Agents. The control flow to the **Change** state is very similar to the control to the **Add** state from the **Establd** state. Depending on the CHANGE options selected and the resources avalailable in each of the stream paths, the CHANGE may result in either a simple refusal of any change or the disconnect of the entire stream. A REFUSE response to a CHANGE request with the E-bit set to zero means that the stream has been torn down.for that Target. A REFUSE_CHANGE is a REFUSE with the E-bit set to 1 indicating that the CHANGE has been refused but the prior stream resources are unchanged

3.4.2 Next Hop State Machine (NHSM)

The NHSM is pictorially shown in Figure 8. This model is common to the Origin as well as an Intermediate Agent .The NHSM consists of the same fundamental states as the OSM: **Init, Establd, Add and Change.**

Init: The state machine for each next hop enters its **Init** state at Agent start-up time. An asterisk indicates that this is the initial state. A **nexthop_open** predicate moves control to the **Establd** state when the next hop associated with an NHSM is required by Targets in a stream.

Establd: Once in the **Establd** state a number of things can happen. Targets may be added by the Origin or Targets may request to join the stream. However, the processing of a JOIN request is always handled by either an OSM or a PHSM. Within each ST Agent, the ST Dispatcher examines incoming JOIN requests and determines whether the stream referenced is a stream that that Agent supports. If not, the JOIN is forwarded on towards the Origin. Once a JOIN request reaches an Agent that can process the JOIN, the ST Dispatcher ACKs the JOIN and queues it up to the resident OSM or PHSM. The NHSM only sees the resultant CONNECT when stream authorization has completed successfully and the OSM or PHSM has issued a CONNECT through the MS/C Box.

As previously described in the OSM, an ST Agent can handle only one stream **Add** or **Change** at a time. If such a stream operation is already underway, further requests are queued and handled when the previous operation has been completed. Either a DISCONNECT or REFUSE for all Targets transfers control from the **Establd** state to the **Init** state.

Add: A CONNECT that has been propagated from the NHSM **Add** state to the next hop Agent PHSM and will require a Response in the form an ACK . If an ACK is not received, the timeout and retry mechanisms of the Retry FSM will invoke a RetryTimeout signal. Every PDU has a

unique reference number, so that all ACKs may be matched to the appropriate Request or Response.

The CONNECT message contains the SID, an updated FlowSpec, and a TargetList. In general, the FlowSpec and TargetList depend on both the next-hop and the intervening network. Each TargetList is a subset of the original TargetList, identifying the targets that are to be reached through the next-hop to which the CONNECT message is being sent. If the TargetList causes a PDU that is larger than the MTU size, CONNECT message to be generated, the CONNECT message is partitioned.

The ACK, if it is received, does not need to be reported to the NHSM. However, if the ACK is not received and the retries are exhausted, a RetryTimeout signal will be reported to the NHSM and interpreted as a REFUSE. The NHSM will record all Target Responses until the last Target in the TargetList has sent an ACCEPT or REFUSE (or an implicit REFUSE due to Retry exhaustion).An Origin DISCONNECT may terminate this process when the End-to-End Response timer is exceeded. A DISCONNECT or REFUSE signal may be due to the failure of a next hop or previous hop.

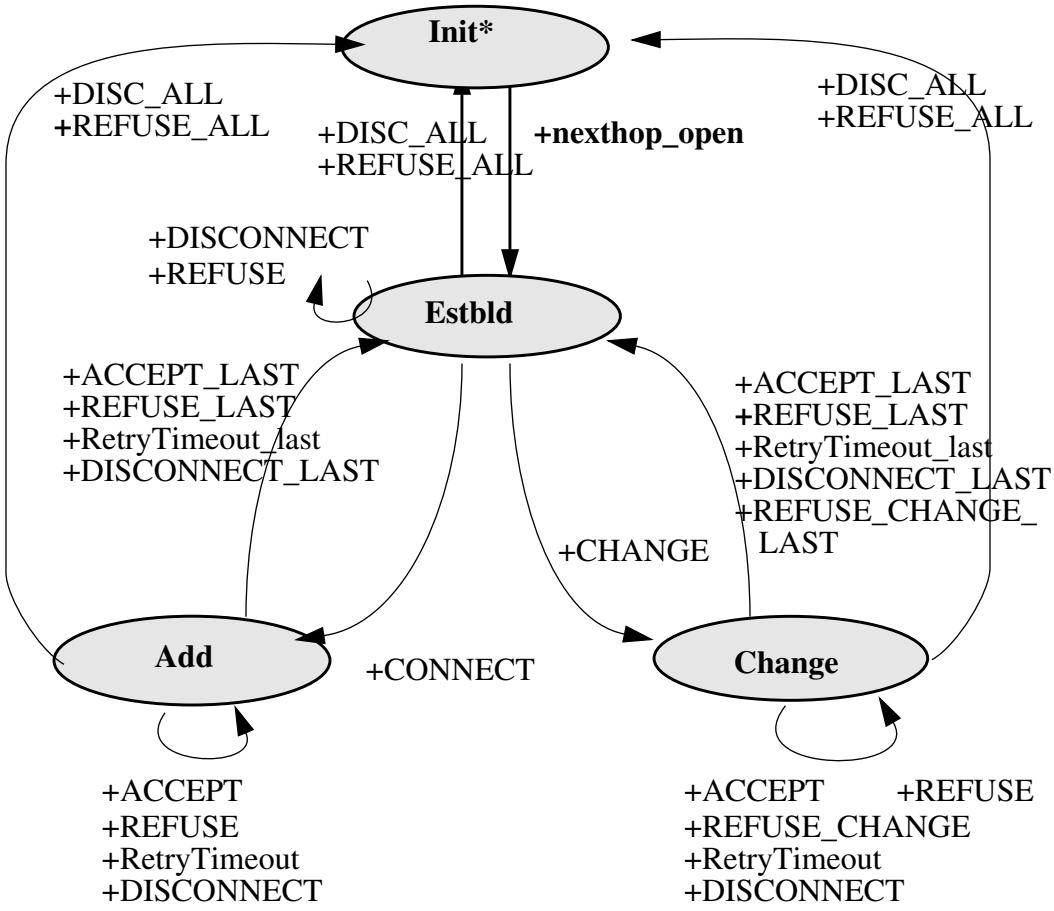
If an Application at a Target does not wish to participate in the stream, it sends a REFUSE message back to the Origin with a ReasonCode (ApplDisconnect). When an NHSM receives a REFUSE message with ReasonCode (ApplDisconnect), the acknowledgement has already been sent by the ST Dispatcher as an ACK to the next-hop. The Agent considers which resources are to be released, deletes the Target entry from the internal database, and propagates the REFUSE message back to the OSM or PHSM.

If, after deleting the specified Target, the next-hop has no remaining Targets, then those resources associated with that next-hop agent may be released. Note that network resources may not actually be released if network multicasting is being used since they may still be required for traffic to other next-hops in the multicast group.

Change: The Application at the Origin may wish to change the FlowSpec of an established stream. To do so, it informs the OSM of the new FlowSpec and of the list of Targets relative to the change. The OSM then issues one CHANGE message with the new FlowSpec per next-hop and sends it with the correct Targetlist. The MS/C box then places copies (as required) of this in the NHSM queues.This takes the control to the **Change** state from the **EstabId** state. CHANGE messages are structured and processed similar to CONNECT messages.

A next-hop agent that is an Intermediate Agent that receives a CHANGE message similarly determines if it can implement the new FlowSpec along the path to each of its next-hop agents, and if so, it propagates the CHANGE messages along the established paths. If this process succeeds, the CHANGE messages will eventually reach the Targets, which will each respond with an ACCEPT (or REFUSE) message that is propagated back to the OSM.

Figure 8. Next Hop State Machine (NHSM)



At this point the Application decides whether all replies have been received. If the change to the FlowSpec is in a direction that makes fewer demands of the involved networks, then the change has a high probability of success along the path of the established stream. Each ST agent receiving the CHANGE message makes the necessary request changes to the network resource allocations, and if successful, propagates the CHANGE message along the established paths. If the change cannot be made, but the E-bit indicates that stream should be torn down, then the ST Agent must recover using DISCONNECT and REFUSE messages as in the case of a network failure. Note that a failure to change the resources requested for specific Targets should not cause other targets in the stream to be deleted. A REFUSE response to a CHANGE request with the E-bit set to zero means that the stream has been torn down.for that Target. A REFUSE_CHANGE is a REFUSE with the E-bit set to 1 and the stream is unchanged

NHSM	Init	Estbld	Add	Change
+nexthop_open	>>Estbld	-	-	-

Table 3: NHSM

NHSM	Init	Estbld	Add	Change
+ACCEPT	-	-	>> Self -ACCEPT	>> Self -ACCEPT
+ACCEPT_LAST	-	-	>> Estbld -ACCEPT	>> Estbld -ACCEPT
+CHANGE	-	>> Change -CHANGE	>> Self -queue	>> Self -queue
+CONNECT	-	>> Add -CONNECT	>> Self -queue	>> Self -queue
+DISCONNECT	-	>> Self -DISCON	>> Self -DISCON	>> Self -DISCON
+DISCONNECT_LAST	-	-	>>Establd -DISCON	>>Establd -DISCON
+DISCONNECT_ALL	-	>>Init -DISCON	>>Init -DISCON	>>Init -DISCON
+REFUSE	-	>> Self -REFUSE	>> Self -REFUSE	>> Self -REFUSE -DISCON
+REFUSE_CHANGE				>> Self -REFUSE_ CHANGE
+REFUSE_LAST	-	-	>> Estbld -REFUSE	>> Estbld -REFUSE -DISCON
+REFUSE_CHANGE_ LAST				>> Estbld -REFUSE_ CHANGE
+REFUSE_ALL	-	>>Init -REFUSE	>>Init -REFUSE	>>Init -REFUSE -DISCON
+RetryTimeout	-	-	>> Self -REFUSE	>> Self -REFUSE -DISCON
+RetryTimeout_Last	-	-	>> Estbld -REFUSE	>> Estbld -REFUSE -DISCON

Table 3: NHSM

The Application at the Origin may specify a set of Targets that are to be removed from the stream with an appropriate ReasonCode (ApplDisconnect). The Targets are partitioned into multiple DISCONNECT messages based on the next-hop route towards the individual Targets. If the TargetList is too long to fit into one DISCONNECT message, it is partitioned.

If, after deleting the specified Targets, any next-hop has no remaining Targets, then those resources associated with that next-hop agent may be released. Note that the network resources may not actually be released if network multicasting is being used since they may still be required for traffic to other next-hops in the multicast group.

When the DISCONNECT reaches a Target, the Target Agent sends an ACK to the upstream NHSM and notifies the Application (at target) that it is no longer part of the stream and for which reason. The ST Agent at the Target deletes the stream from its database after performing any necessary management and accounting functions. Note that the stream is not deleted if the ST Target Agent is also an Intermediate Agent for the stream and there are remaining downstream Targets.

Data Forwarding: Once the Application or OSM determines that the stream is established Data may be transferred to the targets. An Application is not guaranteed that the data reaches its destinations: ST is unreliable and it does not make any attempt to recover from packet loss, e.g. due to the underlying network. In case the data reaches its destination, it does it accordingly to the negotiated quality of service. An ST Agent forwards the data only along already established paths to Targets.

Since a path is considered to be established when the ST next-hop agent on the path sends an ACCEPT message, it implies that the target and all other intermediate ST Agents on the path to the Target are ready to handle the incoming data packets. In no case will an ST Agent forward data to a next-hop Agent that has not explicitly accepted the stream.

At the end of the connection setup phase, the Origin, each Target, and each Intermediate ST Agent has a database entry that allows it to forward the data packets from the Origin to the Targets and to recover from failures of the Intermediate Agents or networks. The database should be optimized to make the packet forwarding task most efficient. The time critical operation is an Intermediate Agent receiving a packet from the previous-hop Agent and forwarding it to the next- hop Agents. The database entry must also contain the FlowSpec, utilization information, the address of the Origin and previous-hop, and the addresses of the Targets and next-hops, so it can perform enforcement and recover from failures. An ST Agent receives data packets encapsulated by an ST header. A data packet received by an ST Agent contains the SID. This SID was selected at the Origin so that it is globally unique and thus can be used as an index into the database, to obtain quickly the necessary replication and forwarding information.

The forwarding information will be network and implementation specific, but must identify the next-hop Agents. It is suggested that the cached information for a next-hop Agent include the local network address of the next- hop. If the data packet must be forwarded to multiple next- hops across a single network that supports multicast, the database may specify the next-hops by a (local network) multicast address. If the network does not support multicast, or the next-hops are on different networks, multiple copies of the data packet must be sent.

No data fragmentation is supported during the data transfer phase. The Application is expected to segment its PDUs according to the minimum MTU over all paths in the stream. The Application receives information on the MTUs relative to the paths to the Targets as part of the FlowSpec contained in the ACCEPT message. The minimum MTU over all paths has to be calculated from the MTUs relative to the single paths. If the Application at the Origin sends a too large data packet, the ST Agent at the Origin generates an error and it does not forward the data.

3.4.3 Previous Hop State Machine (PHSM)

The Previous Hop State Machine Model is common to a Target or Intermediate Agent. A PHSM communicates with an upstream NHSM and downstream with one or more NHSMs and/or a TSM via a MS/C box. When a CONNECT message is received, the Intermediate ST Agent invokes the routing function, reserves resources via the Local Resource Manager, and then propagates the CONNECT messages to its next-hops. For the most part the Intermediate Agent behaves like a relay. In the cases when the Intermediate Agent is not able to successfully send out a CONNECT message to a downstream PHSM, a REFUSE message from the PHSM is sent to the upstream NHSM..

The PHSM consists of a small number of states: **Init, Establd, Add and Change.**

Init: The ST Agent initially takes control from the **Init** state to the **Establd** state via the **phsm_open** predicate. A DISCONNECT or REFUSE of all Targets in a stream will take the stream from the **Establd** to a terminating state which is also the **Init** state.

Establd:Once in the **Establd** state, Targets may be added or changed by the Origin or Targets may request to join the stream. The processing of a JOIN request is always handled by either an OSM or a PHSM. Within each ST Agent, the ST Dispatcher examines incoming JOIN requests and determines whether the stream referenced is a stream that that Agent supports. If not, the JOIN is forwarded on towards the Origin. Once a JOIN request reaches an Agent that can process the JOIN, the ST Dispatcher ACKs the JOIN and queues it up to the resident OSM or PHSM. When stream authorization has completed successfully, the PHSM issues a CONNECT through the MS/C Box to either a NHSM or a TSM.

As previously described in the OSM, an ST Agent can handle only one stream Add or Change at a time. If such a stream operation is already underway, further requests are queued and handled when the previous operation has been completed. Either a DISCONNECT or REFUSE for all Targets transfers control from the **Establd** state to the **Init** state.

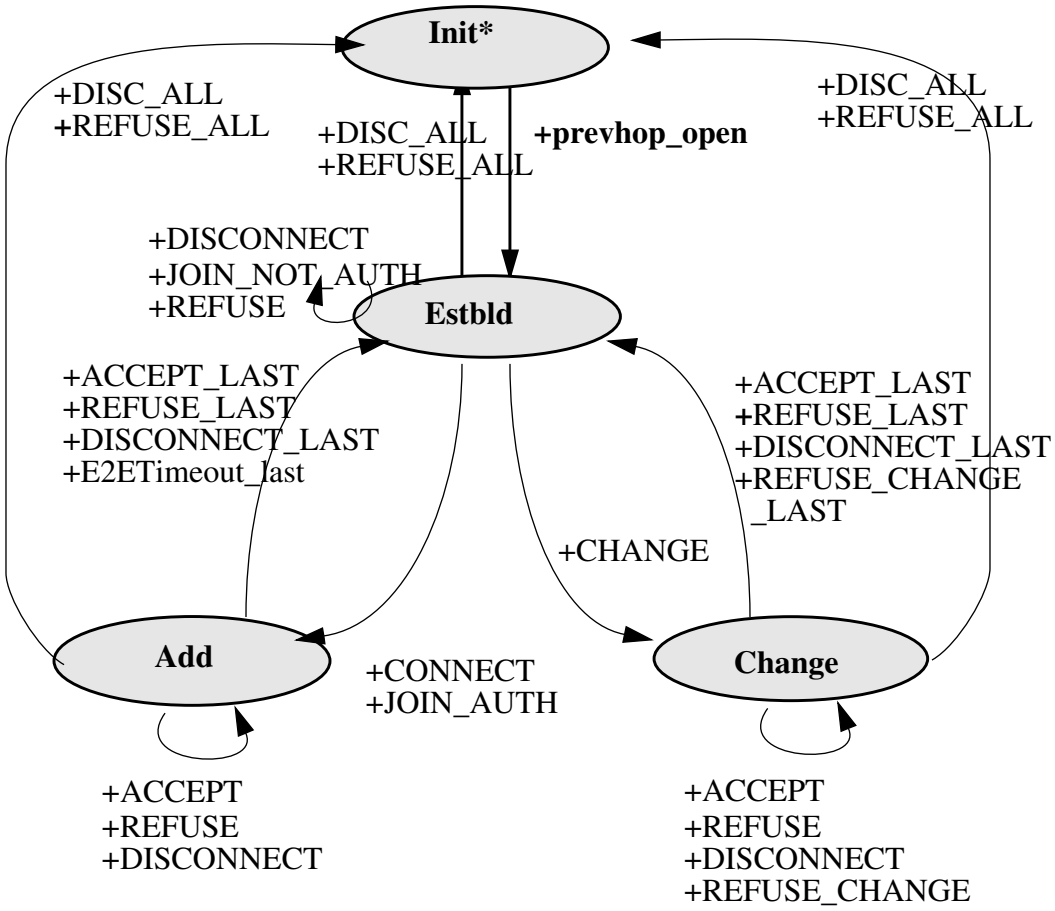
Add: Once in the **Establd** state the previous hop may relay a CONNECT message. A transition to **Add** will create a CONNECT message that is placed in the FIFO queue between the PHSM and the MS/C box. The CONNECT message contains the SID, an updated FlowSpec, and a TargetList. The MS/C box will then make a copy of the CONNECT message, partition the Targetlist parameter and place it the NHSM and/or TSM queues.The spliting (or separating) information is derived from the implementation's routing and LRM functions.

Once in the **Add** state the OSM waits to get ACCEPT or REFUSE responses. The stream will not transition back to the **Establd** state until all Targets have responded. The expiration of the retry

timer and count (if the next hop is not ACKing the request) or the expiration of the end-to-end timer will be interpreted as an implicit refuse.

Change:The Application at the Origin may wish to change the FlowSpec of an established stream. To do so, it informs the ST Agent at the Origin of the new FlowSpec and of the list of Targets relative to the change and this message will be propagated to through the NHSMs to the PHSMs and TSMs. The control flow to the **Change** state is very similar to the previous FSM discussions.

Figure 9. Previous Hop State Machine (PHSM)



PHSM	Init	Estbld	Add	Change
+ prevhop_open	>> Estbld	-	-	-

Table 4: PHSM

PHSM	Init	Estbld	Add	Change
+ACCEPT	-	-	>> Self -ACCEPT	>> Self -ACCEPT
+ACCEPT_LAST	-	-	>> Estbld -ACCEPT	>> Estbld -ACCEPT
+CHANGE	-	>> Change -CHANGE	>> Self -queue	>> Self -queue
+CONNECT	-	>> Add -CONNECT	>> Self -queue	>> Self -queue
+JOIN_AUTHORIZED	-	>> Add -CONNECT	>> Self -queue	>> Self -queue
+JOIN_NOT_AUTHORIZED	-	>> Self -JOIN-REJ	>> Self -queue	>> Self -queue
+DISCONNECT	-	>> Self -DISCON	>> Self -DISCON	>> Self -DISCON
+DISCONNECT_LAST	-	-	>>Establd -DISCON	>>Establd -DISCON
+DISCONNECT_ALL	-	>>Init -DISCON	>>Init -DISCON	>>Init -DISCON
+REFUSE	-	>> Self -REFUSE	>> Self -REFUSE	>> Self -REFUSE -DISCON
+REFUSE_CHANGE	-	-	-	>> Self -REFUSE_ CHANGE
+REFUSE_LAST	-	-	>> Estbld -REFUSE	>> Estbld -REFUSE -DISCON
+REFUSE_CHANGE_LAST	-	-	-	>> Estbld -REFUSE_ CHANGE
+REFUSE_ALL	-	>>Init -REFUSE	>>Init -REFUSE	>>Init -REFUSE
+E2ETimeout_last	-	-	>> Estbld -DISCON -NOTIFYi	>> Estbld -REFUSE_ CHANGE

Table 4: PHSM

3.5 The Target State Machine (TSM)

The Target State Machine (TSM) is a high level state machine which communicates with a PHSM, or OSM if residing the same Agent as the Origin. The TSM also talks to the Upper Layer module via primitives. The TSM consists of a small number of states: **Init, Establd, Add and Change.**

Init: The ST Agent initially takes control from the **Init** state to the **Establd** state via the **tsm_open** predicate. A Target Application may request to join an existing stream. It has to collect information on the stream including the stream ID (SID) and the IP address of the stream's Origin. This can be done out-of- band, e.g. via regular IP. The information is then passed to the local ST Agent together with the FlowSpec. The Application directs the TSM to generate a JOIN message containing the Application's request to join the stream and sends it to the PHSM which in turn sends it upstream toward the stream Origin.

An ST Agent receiving a JOIN message for which that Agent has a matching stream , responds with an ACK. The ACK message must identify the JOIN message to which it corresponds by including the Reference number indicated by the Reference field of the Join message. If the ST Agent is not traversed by the stream that has to be joined, it propagates the JOIN message toward the stream's Origin. Eventually, an ST Agent traversed by the stream or the stream's Origin itself is reached. In any case, the TSM will eventually receive a JOIN-REJECT or CONNECT response. This is shown as transitions to the **Establd** state and the **Add** state respectively.

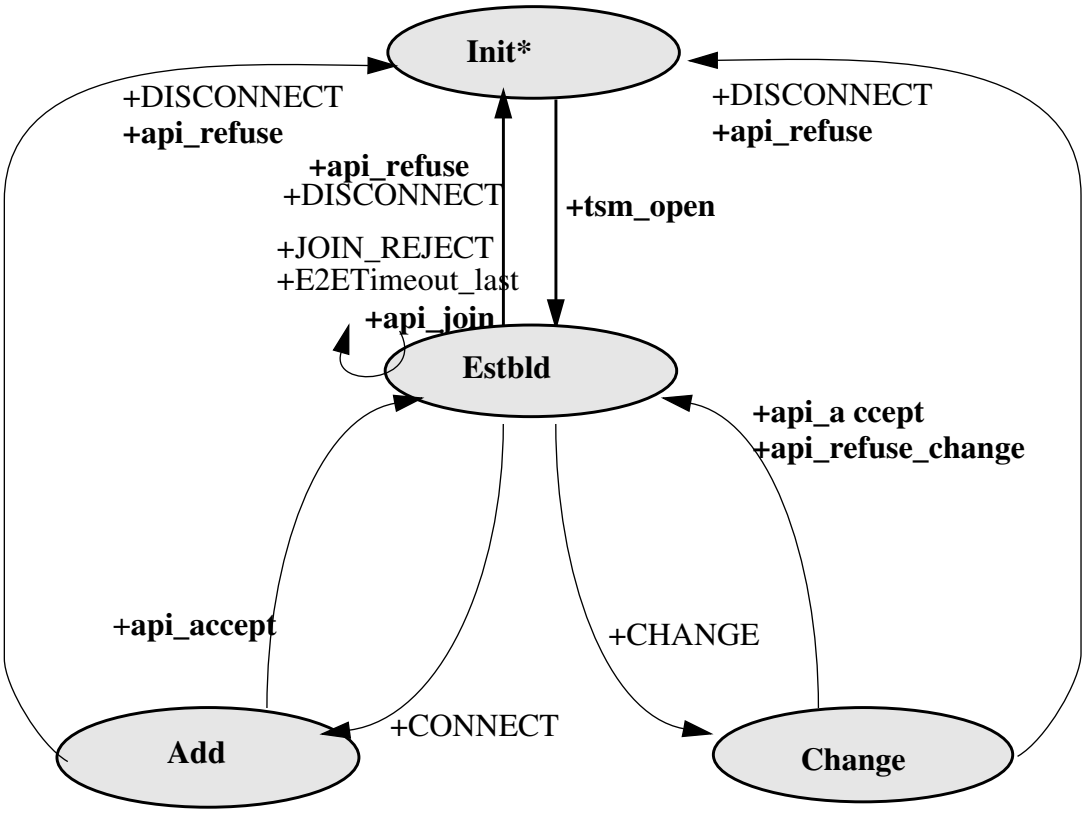
Add: The TSM may receive a CONNECT message any time . The ST Agent reserves local resources and inquires from the specified Application process whether or not it is willing to accept the connection. In particular, the Application must be presented with parameters from the CONNECT, such as the SID, FlowSpec, Options, and Group, to be used as a basis for its decision. The Application is identified by a combination of the NextPcol field and the SAP field included in the correspondent (usually single remaining) target of the TargetList. The contents of the SAP field may specify the port or other local identifier for use by the protocol layer above the host ST layer. Subsequently received data packets will carry the SID, that can be mapped into this information and be used for their delivery.

The TSM responds with an ACCEPT or REFUSE - a result of the Upper Layer module decision.

Change: The TSM may receive a CHANGE message any time it is in a **Establd** state. This happens always after a CONNECT. The TSM again responds with an ACCEPT or REFUSE after informing the Upper Layer Protocol.

The TSM may at any time want to terminate its membership in the stream. This is handled by the TSM sending out a REFUSE message. On the other hand it is possible for an Origin or IntermediateAgent to disconnect the Target from the stream. This is accomplished by the Agent or Origin sending a DISCONNECT message.

Figure 10. Target State Machine (TSM)



TSM	Init	Estbld	Add	Change
+tsm_open	>>Estbld	-	-	-
+api_accept	-	-	>> Estbld -ACCEPT	>> Estbld -ACCEPT
+CHANGE	-	>> Change -change_api	>>self -queue	>>self -queue
+CONNECT	-	>> Add -connect_api	-	-

Table 5: TSM

TSM	Init	Estbld	Add	Change
+DISCONNECT	-	>> Init -disconnect_api	>> Init -discon_api	>> Init -discon_api
+JOIN_REJECT	-	>> Init -join_reject_api	-	-
+api_join	-	>> Self -JOIN	-	-
+api_refuse	-	>> Init -REFUSE	>> Init -REFUSE	>> Init -REFUSE
+api_refuse_change	-	-	-	>> Estbld - REFUSE_ CHANGE
+E2ETimeout_last	-	>>Init -join_reject_api	-	-

Table 5: TSM

SECTION 4

ST Agent FSMs

This section describes the Retry FSM and the Monitor FSM for the datalink and ST Agent neighbor reliability functions. The OSM, NHSM, PHSM and TSM have been shown to model the stream specific Request-Response pattern. The Retry FSM models the datalink reliability provided by the ACK mechanisms with the associated timer and retry count. The Monitor FSM models the ST Agent reliabilty provided by the neighbor HELLO mechanisms, the STATUS and STATUS_RESPONSE messages, as well as the stream Recovery timer and retry count.

The ST Agent FSMs are the unifying aspect of the total FSM model architecture.These models are dependent on how the SCMP messages traverse the ST Agents, and impact Agent databases and FSMs.

4.1 Agent Database Context

ST Agent stream database entries are intitated by the first CONNECT for that Stream Id. The information initially correlated to each StreamId entry includes:

ST Neighbor Previous Hop and Next Hops

FlowSpec, Group, MulticastAddress, Origin, TargetList, ACK and Response timers

Stream Options for NoRecovery(S-bit) and Join Authorization Level (J-bit, N-bit)

Routing results for each Target's Next Hop

LRM results for each Next Hop's resource allocation

Each Agent database is modified when the CONNECT Responses indicate some variation specified by a downstream Agent or Target Response. Subsequent Requests can also modify the database and include additional CONNECT, JOIN and CHANGE requests. Origin, Network or Agent Recovery and LRM initiated stream teardown can occur in the form of explicit DISCONNECT, REFUSE and Recovery initiated CONNECT messages or implicit conditions detected through the HELLO, STATUS, STATUS-RESPONSE and NOTIFY messages.

The database context is then augmented with the history of Reason Codes and prior stream characteristics. Transient state characteristics can also include the G-bit (Global stream TargetList), I-bit (CHANGE risking teardown of old resources)and E-bit (CHANGE REFUSE without teardown), R-bit (Restarted Agent) or ACK and Response retry values.

While all control messages may have an indirect effect on stream state and databases , only ACCEPT, CHANGE, CONNECT, DISCONNECT, JOIN-REJECT, JOIN and REFUSE directly affect each Agent's defintion of each stream. ACK, ERROR, HELLO, NOTIFY, STATUS and

STATUS-RESPONSE are the control messages that are primarily used to maintain ST Agent databases for datalink, neighbor and network management functions.

As the ST PDUs traverse the network, each Agent presumably has a platform specific interface-to-packet-switching function that must intercept the ST packets for ST functions. The ST Dispatcher represents ST PDU validation, filtering and packetswitching. The ST Dispatcher in this model is organized as the Agent packet-switcher, rather than as a per-interface or per-next-hop packet-switcher. This function may be reorganized as a distributed function if the Agent platform architecture requires such distribution.

4.2 ST Dispatcher role for incoming Packet-switching, ACKnowledgement and PDU validation

An ST Dispatcher can validate an ST PDU for ST header and PDU syntax and semantic validity, and then rapidly switch Data packets , .i.e to a local Target application SAP or to the appropriate next hop interface for remote Targets.

When the PDU syntax are in error, an ERROR PDU with the corresponding Reason Code and the offending PDU contents are returned to the SenderIpAddress (instead of an ACK for those SCMP messages that require an ACK). The incoming PDU in ERROR is then discarded and does not directly impact any FSM state. The ERROR response is designed for The following Reason Codes detail the inconsistencies that could be reported in an ERROR Response:

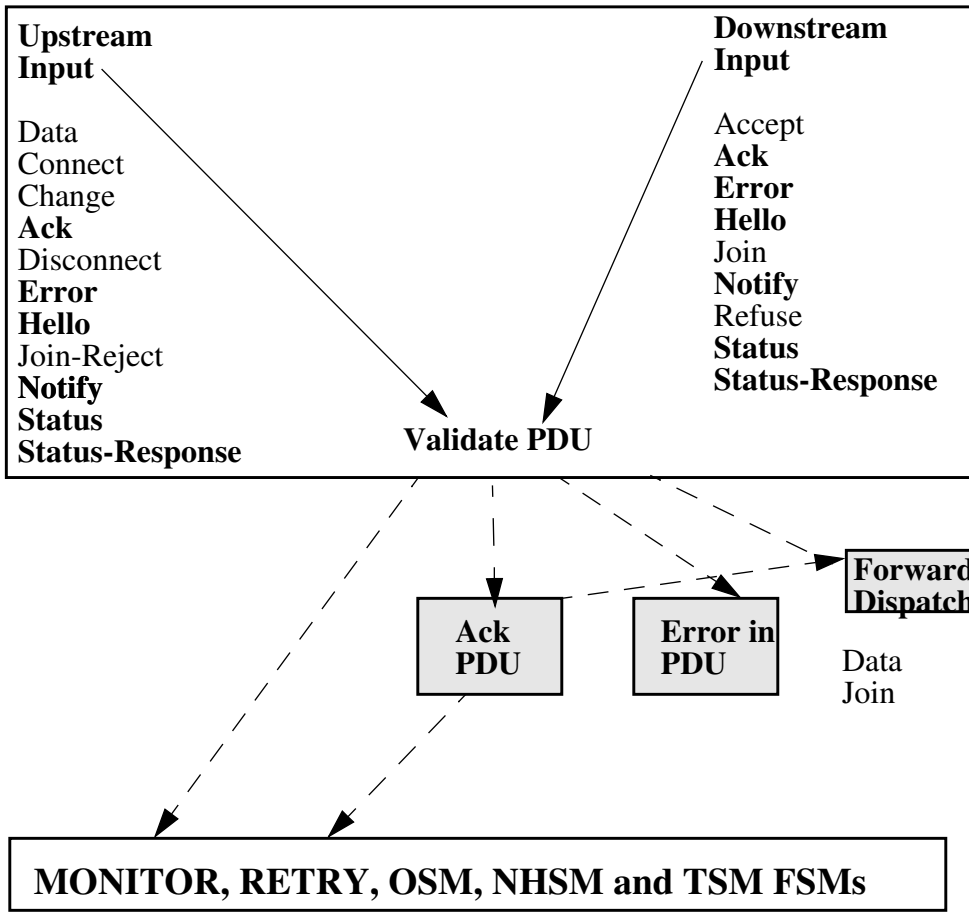
	2	ErrorUnknown	An error not contained in this list has been detected.
	8	AuthentFailed	The authentication function failed.
	13	CksumBadCtl	Control PDU has a bad message checksum.
	14	CksumBadST	PDU has a bad ST Header checksum.
	23	InvalidSender	Control PDU has an invalid SenderIPAddress field.
	24	InvalidTotByt	Control PDU has an invalid TotalBytes field.
*	26	LnkRefUnknown	Control PDU contains an unknown LnkReference.
	31	OpCodeUnknown	Control PDU has an invalid OpCode field.
	32	PCodeUnknown	Control PDU has a parameter with an invalid PCode.
	33	ParmValueBad	Control PDU contains an invalid parameter value.
	35	ProtocolUnknown	Control PDU contains an unknown next-higher layer protocol identifier.
	37	RefUnknown	Control PDU contains an unknown Reference.
	45	SAPUnknown	Control PDU contains an unknown next-higher layer SAP (port).

*	46	SIDUnknown	Control PDU contains an unknown SID.
	48	STVer3Bad	A received PDU is not ST Version 3.
	54	TruncatedCtl	Control PDU is shorter than expected.
	55	TruncatedPDU	A received ST PDU is shorter than the ST Header indicates.

In some cases, RFC1819 specifically requires that an error in a PDU result in an ACK and then a response with the error code. An example of this is when a CONNECT or CHANGE request with an unknown SID results in an ACK followed by a REFUSE with Reason Code 46. In any event, the ST Dispatcher function is to direct only valid PDUs to the inidividual FSM logic.

Figure 11. ST Dispatcher InputInput

r



The next level of PDU analysis involves Agent and stream consistency. The PDU is examined for content consistency with both Agent and stream database information.The following detected inconsistencies may result:

3	AccessDenied	Access denied.
4	AckUnexpected	An unexpected ACK was received.
15	DuplicateIgn	Control PDU is a duplicate and is being acknowledged.
16	DuplicateTarget	Control PDU contains a duplicate target, or an attempt to add an existing target.
49	StreamExists	A stream with the given SID already exists.
51	TargetExists	A CONNECT was received that specified an existing target.
52	TargetUnknown	A target is not a member of the specified stream.
53	TargetMissing	A target parameter was expected and is not included, or is empty.

Most SCMP PDUs (except ACK, ERROR, HELLO, STATUS, STATUS-RESPONSE,) will trigger an ACK to the ST neighbor that sent the PDU.

CONNECT, CHANGE and JOIN Requests will be directed to the appropriate stream PHSM.

Incoming Responses are first correlated with any corresponding Request Reference so that the appropriate next hop or Response timer may be terminated. Then ACCEPT and REFUSE messages are queued up to the appropriate stream NHSM, while DISCONNECT and JOIN-REJECT messages are queued up to the appropriate stream PHSM.

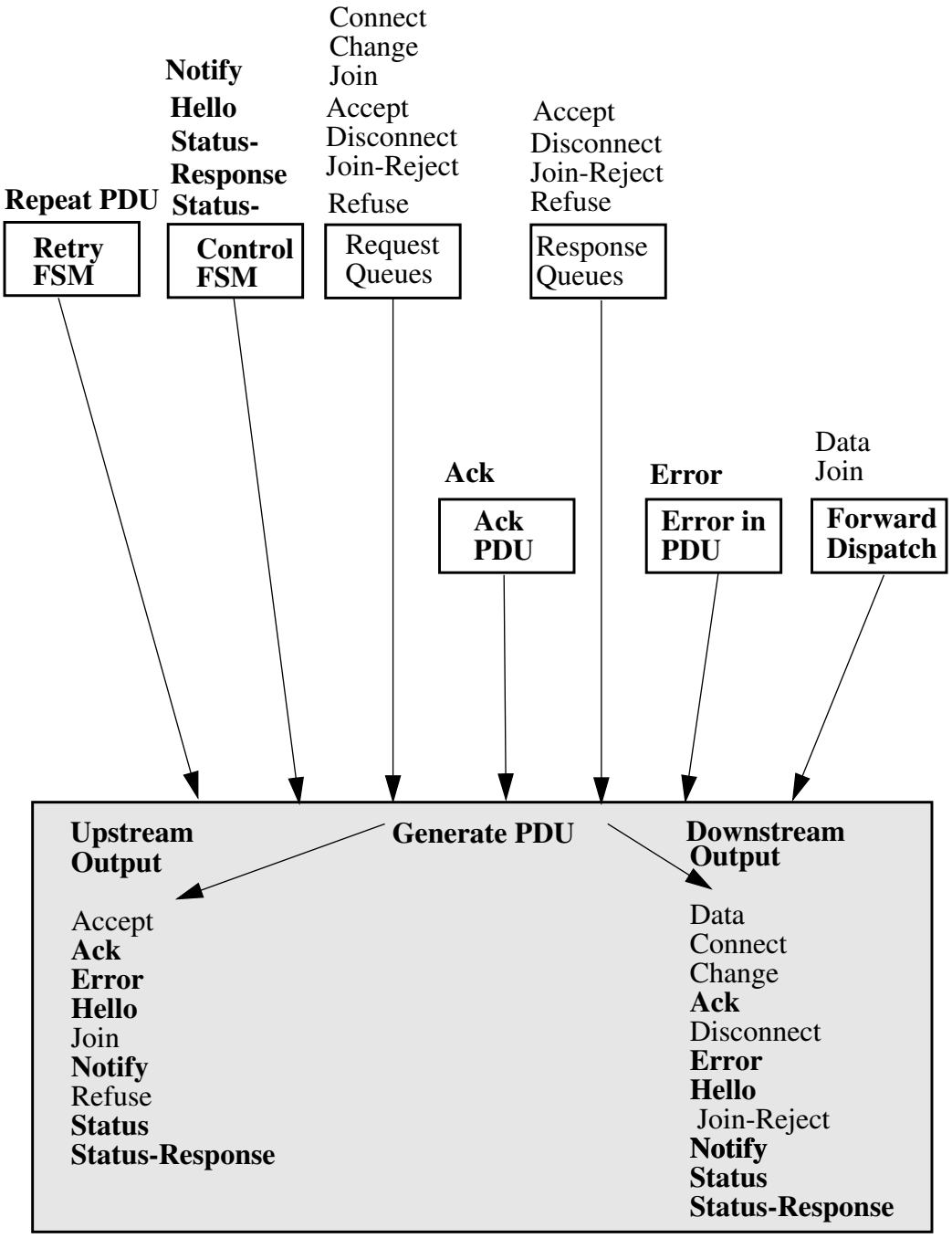
ACK and ERROR messages are correlated with a PDU Reference, terminating the appropriate timers, and then queued up to the stream Retry FSM.

HELLO, STATUS and STATUS-RESPONSE messages are correlated with a PDU Reference so that the appropriate timers may be terminated and then queued up to the Monitor FSM.

4.3 ST Dispatcher functions for outgoing Packet switching, timer and retry settings

Figure 12.

ST Dispatcher Output



The ST Dispatcher also has the role of packaging and forwarding outgoing PDUs to the apprropriate interfaces. The outgoing PDU must be given it's own PDU Reference number and any correlated PDU Referencer number, as well as the semantics and context of the PDU database entries. This Agent architecture model assumes that the Agent and stream databases are the intra-Agent repository of all activities, such that the ST Dispatcher can efficiently create and distribute the PDUs. However, it is entirely possible that the accumulated contents of a PDU has exceeded an outgoing MTU restriction and the PDU would be trunkated with the following Reason Codes:

6	UserDataSize	UserData parameter too large to permit a message to fit into a network's MTU.
36	RecordRouteSize	RecordRoute parameter is too long to permit message to fit a network's MTU.

4.4 Retry FSM- RFSM for datalink reliability of PDU transmissions

The following table provides a quick reference for ST Recovery and Retry implications across ST Agent FSMs. Each SCMP message type that requires an ACK has configured values for the ACK timer and Retry count. Each Request that requires an End-to-End Response has a configured value for the End-to-End Response timer. End-to-End Response timers are set by the Retry FSM when an ACK signals that the Request has successfully gone out to the network. The Recovery Option, STATUS and HELLO messages are managed by the Monitor FSM and follow a different paradigm.

.

Table 6: Table of local control and End-to-End retry parameterss

SCMP message type	Nbr ACK timer	Nbr Retry count	End to End explicit Responses	End to End Response timer	ST Agent FSM responsible
ACCEPT	ToAccept	NAccept			Retry
ACK					
CHANGE	ToChange	NChange	ACCEPT/ REFUSE	ToChange-Resp	Retry
CONNECT	ToConnect	NConnect	ACCEPT/ REFUSE	ToConnect-Resp	Retry
DISCON-NECT	ToDiscon-nect	Ndisconnect			Retry
ERROR					

Table 6: Table of local control and End-to-End retry parameterss

SCMP message type	Nbr ACK timer	Nbr Retry count	End to End explicit Responses	End to End Response timer	ST Agent FSM responsible
JOIN	ToJoin	NJoin	CONNECT/ JOIN-REFUSE	ToJoinResp	Retry
JOIN-REJECT	ToJoinReject	NJoinReject			Retry
NOTIFY	ToNotify	NNotify			Retry
REFUSE	ToRefuse	NRefuse			Retry
Recovery-CONNECT	ToConnect	NConnect	ACCEPT/ REFUSE	ToRetry-Route	Monitor-NRetry-Route
STATUS	ToStatus Resp	NStatus	STATUS-RESPONSE		Monitor-
HELLO	implicit not explicit ACK - DefaultRecoveryTimeout	implicit not explicit retry - Helloloss-Factor			Monitor-Hellotimer-Holddown

The Retry FSM conditions are explicit when an ST Agent neighbor ACK terminates the neighbor ACK timer and retry count for that transaction. Any End-to-End Response will terminate the End-to-End Response timer. Implicit conditions occur when any of the timer or the retry count values have been exhausted. The general paradigm is that an implicit REFUSE is generated for unsatisfied downstream Requests and an implicit DISCONNECT is generated for unsatisfied upstream Requests. The secondary consequence of a timeout is that explicit REFUSE and DISCONNECT messages may also be issued.

Each table entry has its own variation of this basic paradigm. In addition, the ST specification indicates many secondary and tertiary implications for SCMP message failures. As a particular example, once any ST Agent has completed a downstream Request-Response scenario, an upstream propagation problem may or may not cause the stream to be torn down. The I-bit (risk teardown)in CHANGE processing and the S-bit (Recovery) are examples of causes for the secondary and tertiary implications.

Figure 13. **Retry State Machine (RFSM)**

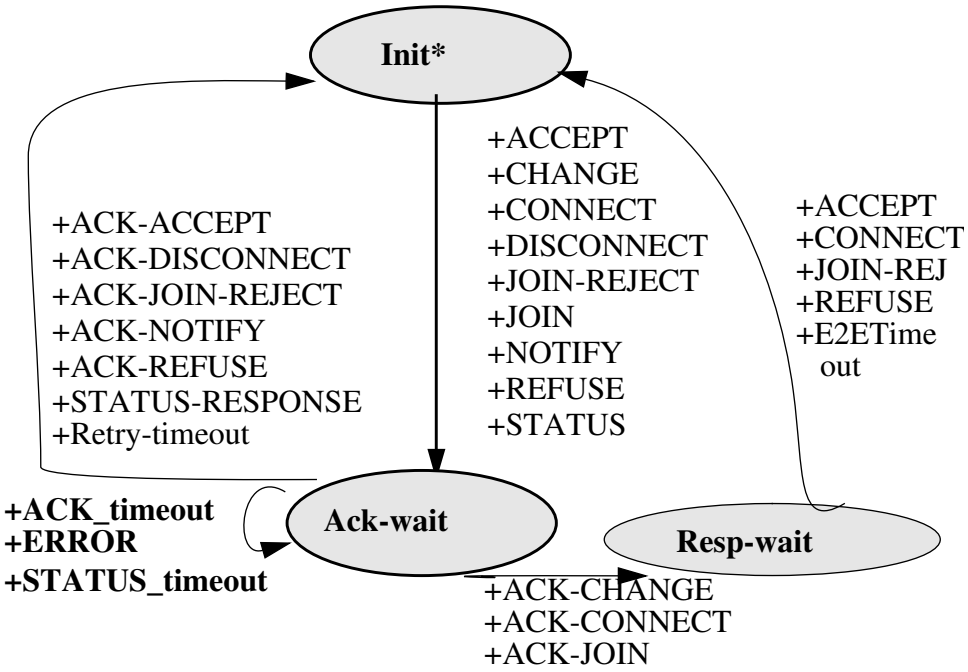


Figure 14.

The Retry FSM has three states - **Init, Ack-wait** and **Resp-wait**. The general paradigm for the Retry FSM is to move from the **Init** state to the **Ack-wait** state whenever a PDU requiring an ACK is sent. The STATUS message does not require an ACK, but the required STATUS-RESPONSE performs the same function as an ACK.

The Retry FSM waits for the resultant ACKS, Responses and/or timeouts. PDUs requiring ACKS cycle through resends for the appropriate NAccept, NChange, NConnect, NDisconnect, NJoin, NJoinReject, NNotify and NRefuse configured counts.

Since all ACKs are correlated by PDU Reference numbers, packets maybe correlated to the outstanding Retry FSM by the same mechanism. Either an ACK or a RetryTimeout that is correlated to an ACCEPT, DISCONNECT, JOIN-REJECT, NOTIFY or REFUSE results in the Retry FSM transitionsto Init. Such PDUs have no End-to-End response requirements and generally have no secondary error processing when it can be assumed that the neighbor Agent and/or link layer reliability is gone. An ACCEPT is an exception. The failure of an ACCEPT is an implicit REFUSE upstream and DISCONNECT downstream.since this ACCEPT was an End-to-End Response that has now failed to completely traverse the stream Agents.

An ACK on a CHANGE, CONNECT or JOIN causes the respective ToChange,ToConnect or ToJoin End-to-End timers to be set, and a state transition to **Resp-wait**.

A legitimate Response, or an E2ETimeout on CHANGE, CONNECT or JOIN causes the transition to the **Init** state with the signal to be replicated to the appropriate stream FSM.

Retry FSM	Init	Ack-wait	Resp-wait
+Ack_timeout	-	>>Self -resend PDU	-
+ERROR	-	>>Self -resend PDU	-
+ACK-ACCEPT	-	>>Init	-
+ACK-CHANGE	-	>>Resp-wait	-
+ACK-CONNECT	-	>>Resp-wait	-
+ACK-DISCONNECT	-	>>Init	-
+ACK-JOIN	-	>>Resp-wait	-
+ACK-JOIN-REJECT	-	>>Init	-
+ACK-NOTIFY	-	>>Init	-
+ACK-REFUSE	-	>>Init	-
+RetryTimeout-ACCEPT	-	>>Init -DISCONNECT -REFUSE	-
+RetryTimeout-CHANGE	-	>>Init -RetryTimeout	-
+RetryTimeout-CONNECT	-	>>Init -RetryTimeout	-
+RetryTimeout-DISCONNECT	-	>>Init	-
+RetryTimeout-JOIN	-	>>Init -RetryTimeout	-
+RetryTimeout-JOIN-REJECT	-	>>Init	-
+RetryTimeout-NOTIFY	-	>>Init	-
+RetryTimeout-REFUSE	-	>>Init	-
+ACCEPT	-	-	>> Init -ACCEPT
+REFUSE	-	-	>>Init -REFUSE

Retry FSM	Init	Ack-wait	Resp-wait
+CONNECT	-	-	>> Init -CONNECT
+JOIN-REJECT	-	-	>> Init -JOIN-REJECT
+STATUS-RESPONSE	-	>> Init -STATUS-RESPONSE	-

4.5 Agent , Neighbor and Stream Supervision

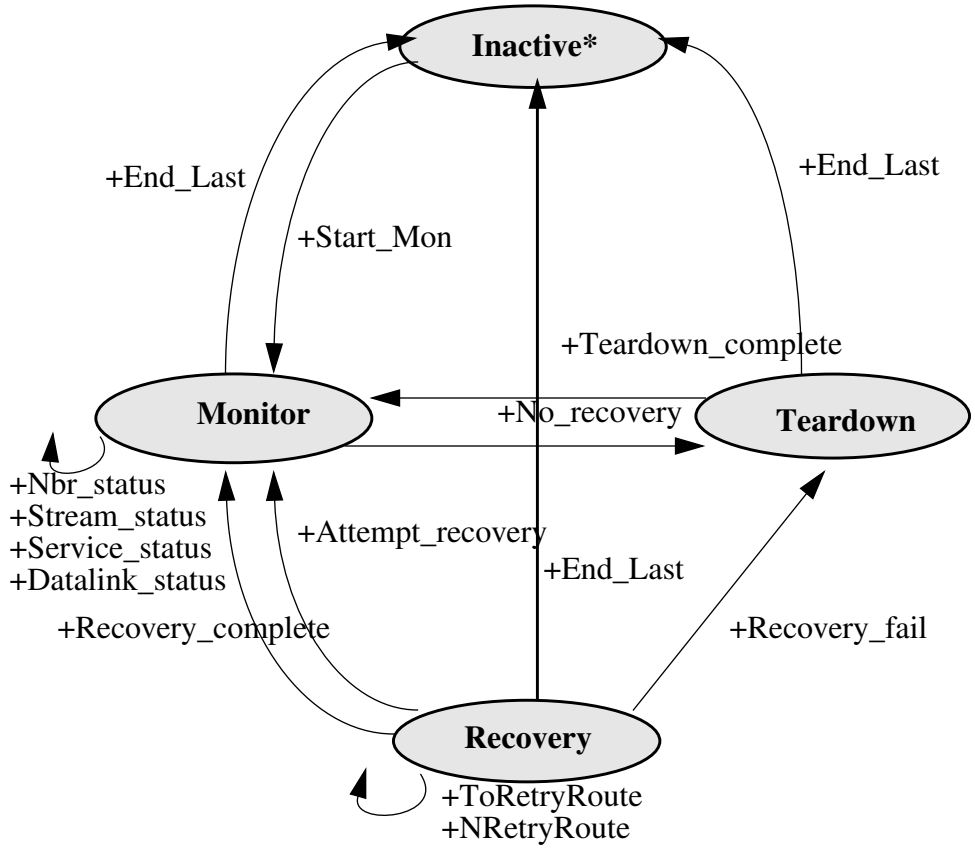
4.5.1 The MonitorFSM (MFSM) for Agent and Stream Supervision

Each ST Agent must monitor its own status, network conditions, neighbor Agent status and supervise the Recovery of streams whenever required and possible during a network failure.This MFSM is intended to be a general approach to these issues, rather than a fully specified FSM since the particular network, platform and implementation architecture will determine detail FSM considerations.

What this MFSM model does suggest is that the MFSM provides a superstructure for the management of the Neighbor Detection Failure FSM(NDFSM), as well as any Agent NOTIFY, STATUS or STATUS-RESPONSE implications. The Service Model management (including application issues, routing and LRM or other Agent implementation specific issues), as well as datalink statistics analysis (e.g., broken or dropped PDUs or accumulated routing errors) may also be incorporated into this FSM.

At the very least, stream Recovery requires careful analysis of the possible recursions in Agent, neighbor failure detection, routing and LRM conditions. The ST2+ specification defines parameters for a configured number of times that Recovery should be attempted (NRetryRoute), the configured time to wait for each Response (ToRetryRoute) and variations in the exception processing.

Figure 15. .MFSM



During the course of a stream setup, the CONNECT contains a Recovery Timeout, as specified by the Origin. The resultant ACCEPTs contain the Agent's "supportable" Recovery Timeout such that the stream Recovery Timeout becomes the smallest Recovery Timeout for all Targets. The HELLO timer must be smaller than the smallest Recovery Timeout for all streams between these Agents, but an Agent may have various HELLO timers between different Agents, such that the management function of such timers should fall into the MFSM also. A Round Trip Time (RTT) estimation function is available with STATUS and STATUS-RESPONSE messages to aid in this area.

The MFSM relies on the Nieghbor Detection Failure FSM (NDFSM) as the primary notification vehicle for stream and neighbor management. During the initial stream setup of any stream NHSM and PHSM, the MFSM is signalled to begin monitoring of the FSM neighbor Agents involved in the stream. The sending of HELLOs is begun once an ACCEPT is forwarded upstream. The receiving of HELLOs is acceptable as soon as an ACCEPT is received. HELLOs are terminated once an ACK is sent or received for the DISCONNECT or REFUSE associated with the last of all streams and Targets for that neighbor.This requires signalling and coordination with the ST Dispatcher, Retry FSM and database context, especially when the Restarted bit is active for either the local Agent of a neighbor.

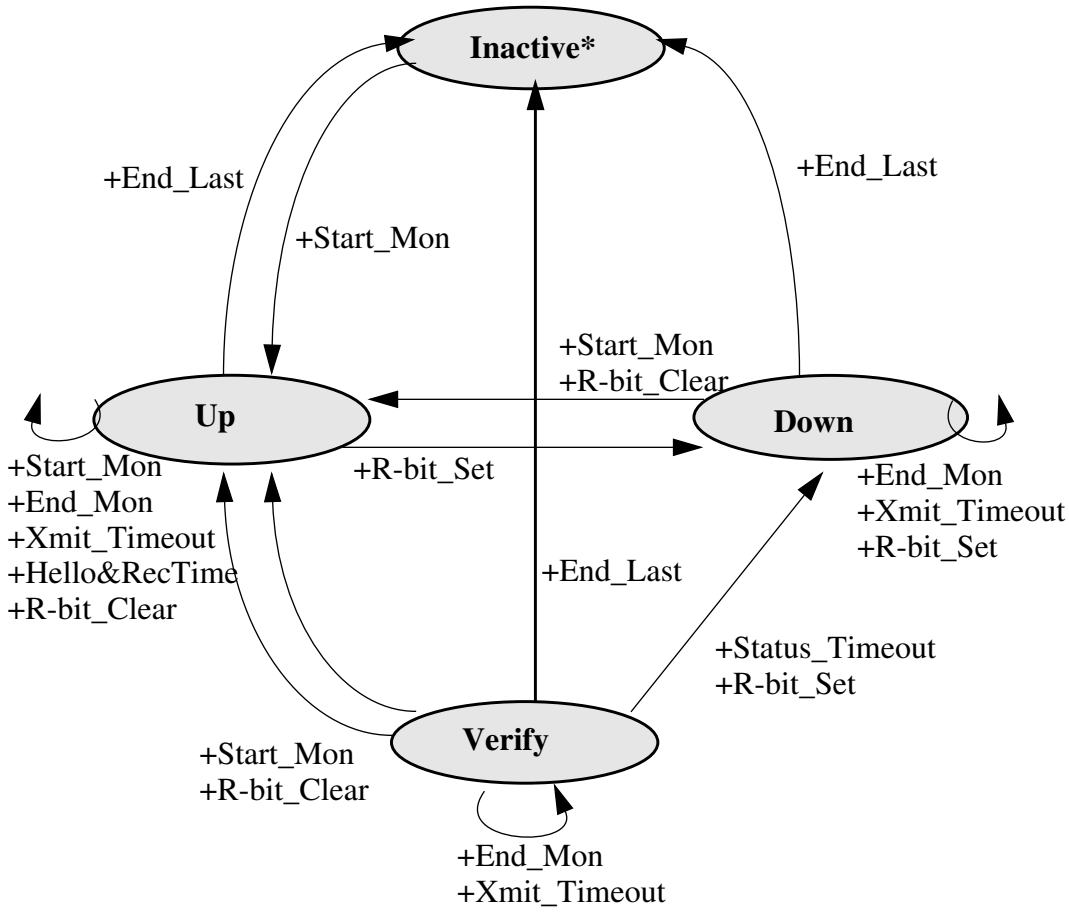
Agent network "inspection and repair" functions might also exist in the MFSM to extend the mechanisms of the NDFSM before attempting Recovery and/or stream teardown.

Group management for Bandwidth-sharing, Fate-sharing, Path-sharing and Subnet resource- sharing can be intiated by any ST Agent and it may be adviseable to incorporate optimization algorithms in the MFSM to interact with Routing and LRM functions, thus allowing the MFSM to monitor and gauge the impact on the stream Recovery analysis.

4.5.2 The Nieghbor Detection Failure FSM for Neighbor Management

This FSM has a more atomic focus in that ST neighbor HELLOs are maintained and monitored only while there are one or more shared streams active. When the neighbor HELLOs and subsequent STATUS inquiry fails or the neighbor R-bit has been set, the neighbor is considered down and the streams involved in that neighbor relationship must be examined for Recovery conditions.

Figure 16. NFDSM

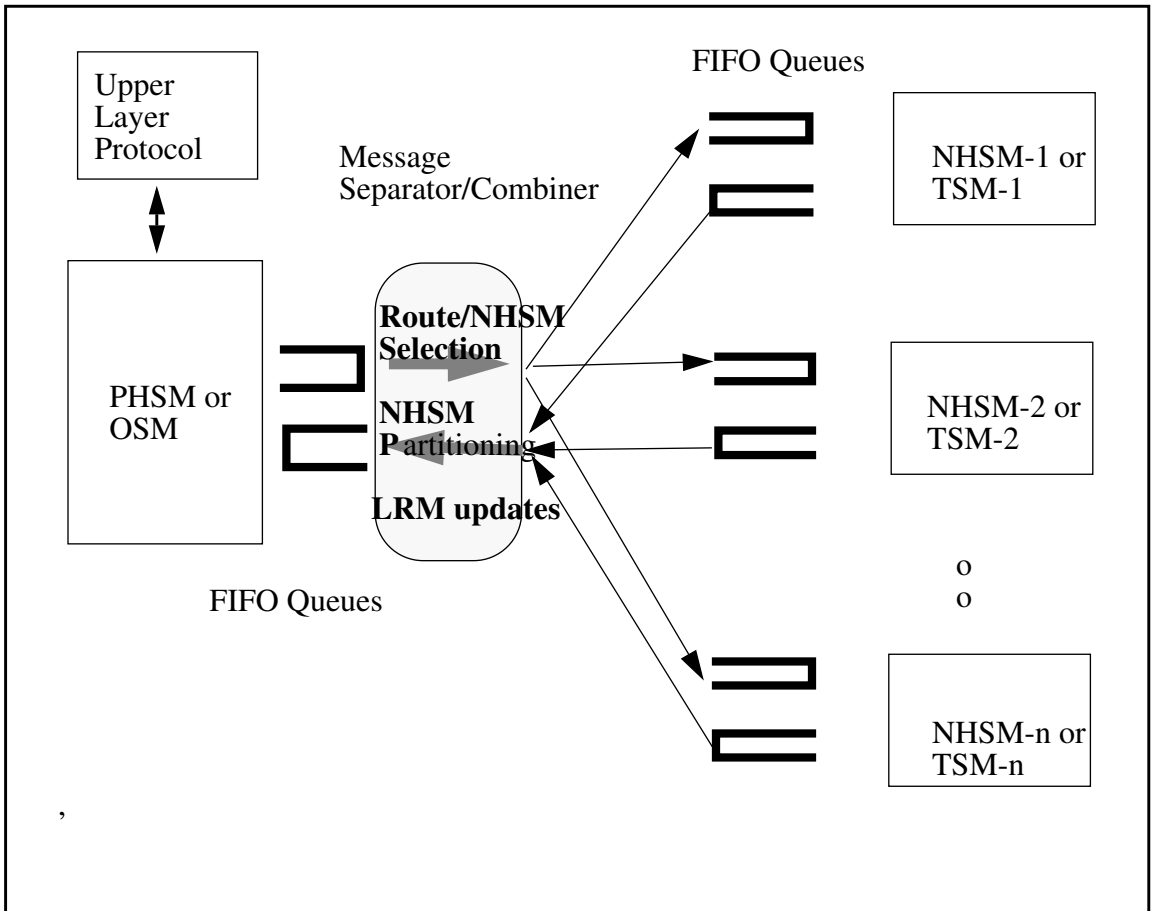


NFDSM	Inactive	Up	Verify	Down
+Start Monitoring	>> Up	>> Self	>> Up	>> Up
+End Monitoring	-	>>Self	>> Self	>> Self
+End Monitoring Last	-	>>Inactive	>>Inactive	>>Inactive
+Xmit Timeout	-	>> Self -Hello	>> Self -Hello	>> Self -Hello
+Hello & Recovery Timeout	-	>> Self	-	-
+NO Hello & Recovery Timeout	-	>> Verify -Status (SID)	-	-
+Status Timeout	-	-	>> Down -N_Down	-
+R-bit Set	-	>> Down -N_Down	>> Down -N_Down	>> Self
+R-bit Clear	-	>> Self	>> Up	>> Up

Table 7: NFDSM

4.5.3 Service Model Interactions

Figure 17. MS/C Box Communications inside an Agent



The optimization of route and LRM functions can affect the selection from multiple path routes to a Target on initial CONNECTs, as well as CHANGE and Recovery procedures. This document's model follows a sequential process of integrating the route and LRM services with the MS/C Box. for the atomic stream FSMs.

Additional algorithms may be used in the MFSM, such that algorithms for Options and Group factors may be optimized in relation to the stream Recovery decisions.

SECTION 5

Exception Processing

Various types of exception processing conditions have been referenced in the preceding sections. Not all have been spelled out in detail. The general paradigms fall into several categories and all of this document's models are based on a suggested approach. The secondary and tertiary conditions of some apects of exception processsing are especially subject to implementation preferences.

The first topic of discussion might be the category SCMP datalink reliability as generally characterized in the Retry FSM. This document favors maintaining a coordinating Retry FSM versus incorporating the Retry states in each of the OSM, NHSM, PHSM and TSM, which is naturally an alternative.

ERROR message generation for PDU semantics problems is discussed in Section 5 as an ST Dispatcher function. A special case occurs in PDU construction when the MTU size is exceeded, i.e.:

6	UserDataSize	UserData parameter too large to permit a message to fit into a network's MTU.
36	RecordRouteSize	RecordRoute parameter is too long to permit message to fit a network's MTU.

However, all of the analysis and potential REFUSE message or signal generation, still seems best suited to the ST Dispatcher.

5.1 Additional Exception Processing

5.1.1 ST Dispatcher detected inconsistencies Reason Codes:

The following errors can also be detected by the ST Dispatcher with the careful analysis of all Agent and stream database values:

3	AccessDenied	Access denied.
4	AckUnexpected	An unexpected ACK was received.
15	DuplicateIgn	Control PDU is a duplicate and is being acknowledged.
16	DuplicateTarget	Control PDU contains a duplicate target, or an attempt to add an existing target.
49	StreamExists	A stream with the given SID already exists.
51	TargetExists	A CONNECT was received that specified an existing target.

52 TargetUnknown A target is not a member of the specified stream.

53 TargetMissing A target parameter was expected and is not included, or is empty.

This means that the atomic FSMs do not have to incorporate this logic and this approach simplifies the atomic FSM paradigms.

5.1.2 MonitorFSM issues with neighbor failure and stream recovery Reason Codes:

The details of these specific instances can also be intertwined with Retry, Routing and LRM failures.

12 CantRecover Unable to recover failed stream.

22 IntfcFailure A network interface failure has been detected.

27 NetworkFailure A network failure has been detected.

39 RestartLocal The local ST agent has recently restarted.

40 RestartRemote The remote ST agent has recently restarted.

47 STAgentFailure An ST agent failure has been detected.

5.1.3 Retry and Timeout Failures Reason Codes:

38 ResponseTimeout Control message has been acknowledged but n**ot** answered by an appropriate control message.

41 RetransTimeout An acknowledgment has not been received after several retransmissions.

5.1.4 Routing issues Reason Codes:

Routing issues initiate special exception processing requirements. Some of these have been addressed in the ST2+ specification, but each implementation should consider the network and platform architecture, also.

9 BadMcastAddress IP Multicast address is unacceptable in CONNECT

28 NoRouteToAgent Cannot find a route to an ST agent.

29 NoRouteToHost Cannot find a route to a host.

30 NoRouteToNet Cannot find a route to a network.

34 PathConvergence Two branches of the stream join during the CONNECT setup.

42 RouteBack Route to next-hop through same interface as previous-hop and is not previous-hop.

43 RouteInconsist A routing inconsistency has been detected.

44 RouteLoop A routing loop has been detected.

5.1.5 LRM issue Reason Codes:

Optimization of routing and LRM issues can also initiate special exception processing requirements. Some of these have been addressed in the ST2+ specification, but each implementation should also consider the network and platform architecture.

10	CantGetResrc	Unable to acquire (additional) resources.
11	CantRelResrc	Unable to release excess resources.
17	FlowSpecMismatch	FlowSpec in request does not match existing FlowSpec.
18	FlowSpecError	An error occurred while processing the FlowSpec.
19	FlowVerUnknown	Control PDU has a FlowSpec Version Number that is not supported.
20	GroupUnknown	Control PDU contains an unknown Group Name.
21	InconsistGroup	An inconsistency has been detected with the streams forming a group.
50	StreamPreempted	The stream has been preempted by one with a higher precedence.

SECTION 6

APPENDIX

6.1 Glossary

All stream FSMs have the following 4 states in common:

Init: The stream has no active Targets.

Establd: The stream is established and may or may not have Target members.

Add: The stream is currently adding Targets as the result of a Connect or Join initiated Connect.

Change: The stream is currently attempting to Change according to a new FlowSpec.

A list of predicates, API interactions and combination conditions include the following:

api_close - the Origin API explicitly terminates a stream, since a stream with no Targets at the Origin may remain Established

api_open - the Origin API explicitly establishes a stream to initiate all database setup functions whether or not any Targets are initially specified.

api_connect- the Origin API adds Targets.

api_change - the Origin API initiates a CHANGE to the FlowSpec.

api_disconnect - the Origin API initiates a DISCONNECT to Targets.

accept_api - the OSM propagates an ACCEPT received from either a TSM or a NHSM to the Origin API.

notify_api - the OSM propagates a NOTIFY received from either a TSM or a NHSM to the Origin API.

refuse_api - the OSM propagates a REFUSE received from either a TSM or a NHSM to the Origin API.

nexthop_open - the first time each unique NHSM is invoked for each unique stream in an Agent, the Agent explicitly establishes a NHSM database and Establd state.

prevhop_open - the first time the PHSM is invoked for each unique stream in an Agent, the Agent explicitly establishes a PHSM database and Establd state.

api_join - the Target API initiates a JOIN request**.**

api_refuse - the Target API initiates a REFUSE to the TSM.

api_refuse_change - the Target API initiates a REFUSE of a CHANGE request to the TSM.

connect_api - the TSM propagates a CONNECT to the Target API.

change_api - the TSM propagates a CHANGE to the Target API.

join_reject_api - the TSM propagates a JOIN_REJECT to the Target API.

disconnect_api - the TSM propagates a DISCONNECT to the Target API.

JOIN_AUTH - a PHSM or OSM JOIN is authorized.

JOIN_NOT_AUTH - a PHSM or OSM JOIN is not authorized.

RetryTimeout - an FSM recieves an implicit REFUSE response to a CONNECT or CHANGE request to one Target in the TargetList by exceeding the ACK retry and timeout values (i.e., ToChange/NChange, ToConnect/NConnect timers and retry counts) for that particular transaction.

The Add and Change states cannot transition back to the Establd state until all Targets have given implicit or explicit responses.

ACCEPT_LAST - the last Target in the TargetList for a CONNECT or CHANGE has responded with an ACCEPT.

DISC_LAST - the last Target in the TargetList for a CONNECT or CHANGE has responded with a DISCONNECT.

REFUSE_LAST - the last Target in the TargetList for a CONNECT or CHANGE has responded with a REFUSE.

RetryTimeout_last - the last Target in the TargetList for a CONNECT or CHANGE has responded with an implicit REFUSE by exceeding the ACK retry and timeout values (i.e., ToChange/NChange, ToConnect/NConnect timers and retry counts).

E2E_Timeout_last - the last Target in the TargetList for a CONNECT or CHANGE has responded with an implicit REFUSE by exceeding the End-to-End timeout value (i.e., ToChange-Resp, ToConnectResp timers).

Except in the case of the OSM (which must be explicitly closed by the Origin API, the Establd, Add and Change states transition back to the Init state when all Targets in the unique FSM TargetList have given implicit or explicit stream teardown instructions.

DISC_ALL -a DISCONNECT has been received for the last Target in the entire TargetList for a stream FSM (as opposed to the TargetList for a particular CONNECT or CHANGE request).

REFUSE_ALL - a REFUSE has been received for the last Target in the entire TargetList for a stream FSM (as opposed to the TargetList for a particular CONNECT or CHANGE request

6.2 ST Control Message Flow

Control Message Types

ST control messages are generally of the Request -Response type. Table 1 summarizes these control messages alphabetically. The table has three major columns.

- Message type
- Response
- Possible causes for message

6.2.1 Message Type

Under the Message Type each control message is categorized either as a:

— Request message

— Response message

It is possible for a message to be more than one type depending on the usage, although this is not apparent from this table.

6.2.2 Response

The Response to each control message is given in the next major column under Response. Note that the Response to a message can be interpreted to mean either:

1. a Response to another control message
2. a Response to indicate the condition of receipt of the message, driven primarily by the error control function

The second interpretation of Response includes positive acknowledgments and negative acknowledgments (error response). Thus, this major column has the following categories:

— Error Response

— Mandatory Response

— Other response following mandatory response.

An X or an entry in the table indicates classification of a message under a particular category shown under each major column.

6.2.3 Possible causes for message

Finally, a control message might have been sent in response to another control message. This is shown in the last column. Note that it is possible that independently a number of control messages may be the cause for this control message in question Note that an entry does not necessarily mean that is the only cause. A blank entry in this column for instance means that the message was not invoked by another message.

For example, an ACCEPT message is a Response Type message to either a CONNECT or a CHANGE message. It will be acknowledged (Mandatory response) with an ACK. It may be responded with an ERROR in case of error conditions. The state diagrams illustrate this sequencing more completely. It may be noted that the sequencing of messages gives the protocol semantics.

.

Table 8: Message Types: Requests, Responses and Others

	Type		Response			Possible causes for message:
Message	Req.	Resp.	Error Resp.	Man-datory Resp.	Other response following Mandatory Resp.	
ACCEPT		X	ERROR	ACK		1. CONNECT 2. CHANGE
ACK		X	ERROR			1. ACCEPT 2. CHANGE 3. CONNECT 4. DISCONNECT 5. JOIN 6.JOIN-REJECT 7. NOTIFY 8. REFUSE
CHANGE	X		ERROR	ACK	ACCEPT REFUSE	1.Origin Change stream
CONNECT	X	X	ERROR	ACK	ACCEPT REFUSE	1.Target JOIN 2. Origin Connect TargetList

Table 8: Message Types: Requests, Responses and Others

	Type		Response			
Message	Req.	Resp.	Error Resp.	Man-datory Resp.	Other response following Mandatory Resp.	Possible causes for message:
DISCONNECT	X		ERROR	ACK		1.Origin Disconnect TargetList 2. Intermediate Agent detects upstream failure or is acting as an Origin
ERROR		X				Errored mesgs.: 1. ACCEPT 2. ACK 3. CHANGE 4. CONNECT 5. DISCONNECT 6. HELLO 7. JOIN 8. JOIN-REJECT 9. NOTIFY 10.REFUSE 11.STATUS 12. STATUS-RESPONSE
HELLO	X		ERROR			1.Periodic Message
JOIN-REJECT		X	ERROR	ACK		1. JOIN
JOIN	X		ERROR	ACK	CONNECT JOIN-REJECT	Target joining stream
NOTIFY		X	ERROR	ACK		1. Information message 2. Notification upstream to JOIN for some authorization levels

Table 8: Message Types: Requests, Responses and Others

	Type		Response			Possible causes for message:
Message	Req.	Resp.	Error Resp.	Man-datory Resp.	Other response following Mandatory Resp.	
REFUSE	X	X	ERROR	ACK		1.Target leaving stream 2. CONNECT 3. CHANGE 4.Intermediate Agent detects downstream stream failure
STATUS	X		ERROR	STATUS-RESPONSE		
STATUS-RESPONSE		X	ERROR			1.STATUS

6.3 Internetwork Complexities

The following internetwork diagram of ST Agents indicates the Origin (O), Intermediate (I) and Target (T) roles of each ST Agent in relation to two ST conferences. Conference 1 (C1) has 4 participants, all of which are sending and receiving data from each other. Intermediate Agents 1, 2, 3 and 4 each have an ST application with an Origin sending data to all other members, and simultaneously receiving data as a Target for each of the other members.

Figure 18.

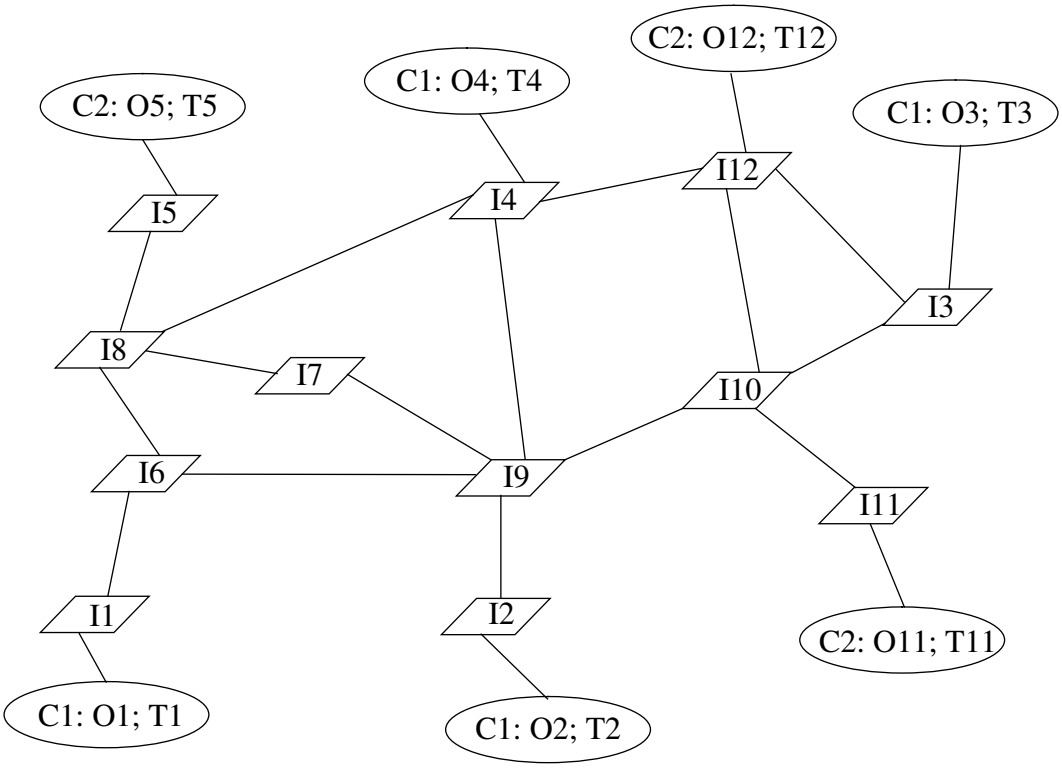
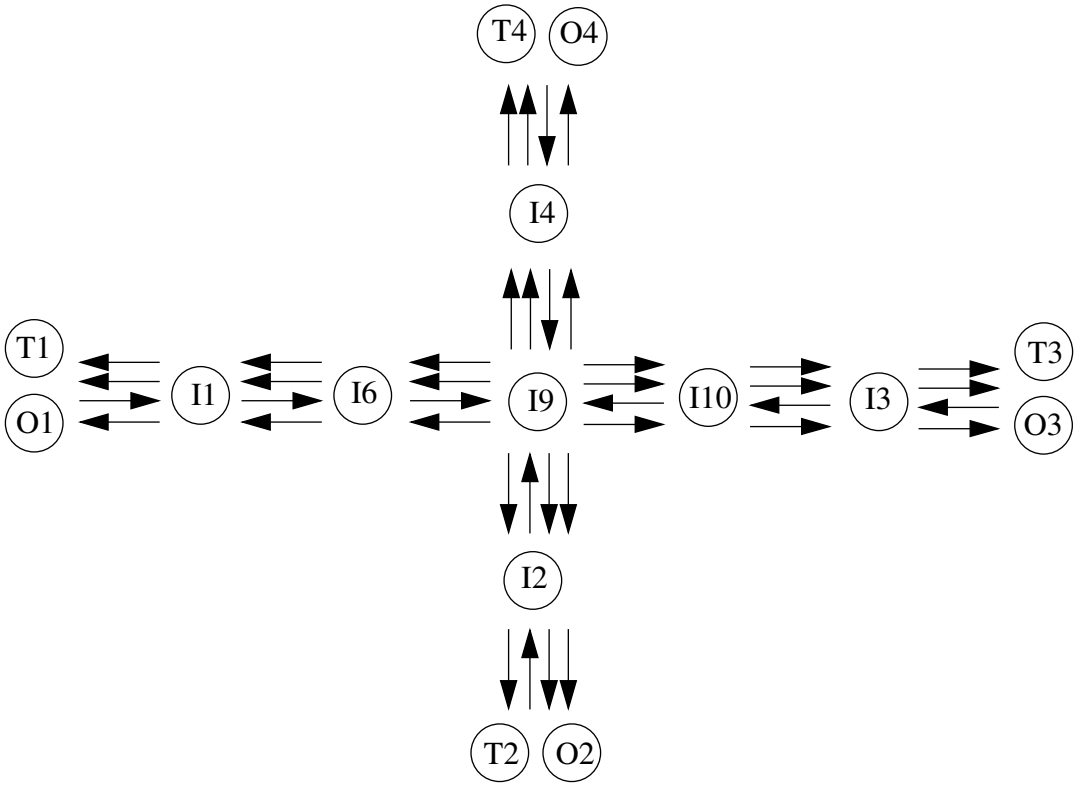


Figure 19.

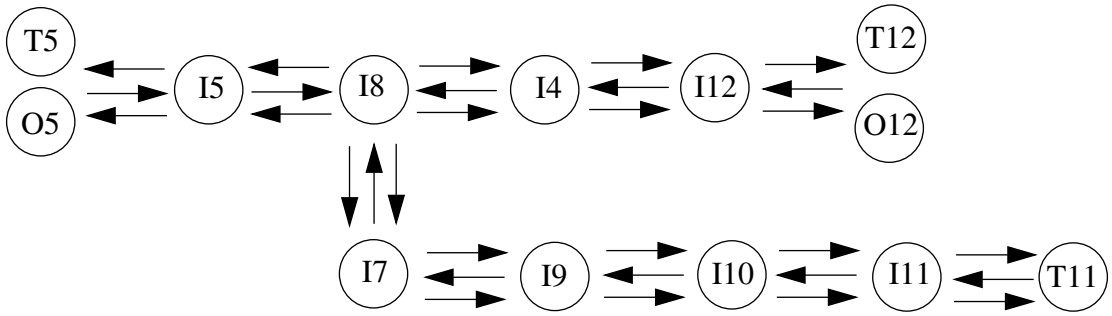


Each ST Agent participating in C1, as illustrated, has all four streams to manage, representing a fully meshed stream conference with Targets and Origins communicating along the same paths.

There are a number of other possible routes for each stream in C1. The above paths through Agents I6, I9 and I10 were chosen to illustrate a simple routing scheme for such a conference. Agents I8 and I12 could have just as easily been involved, if there were no other routing metrics to consider other than number of hops. However, it is also possible that the resources available at any Agent or interface may not actually be equal , such that Agents I7 and I12 became involved in branches of some of the streams. In such alternative routing and resource circumstances, some of the Intermediate Agents might only maintain one stream in the conference. However, in this illustration, Agent I9 happens to have an ST neighbor for every stream and the need to manage multiple targets for each stream.

Conference 2 (C2) has 3 participants and is also a fully meshed set of streams for each member of the conference. All ST Agents in the C2 illustration also have multiple ST neighbors, streams and interfaces to manage.

Figure 20. ,



In addition, since both conferences are being conducted simultaneously, several Agents are managing streams from both conferences, which may be Grouped for Resource or Fatesharing characteristics. The dynamics of such internetwork topology and resource issues can become complex stream management issues.

Internet-Draft **January, 1995**

ST Working Group M. Rajagopal and Sharon Sergeant

File: draft-ietf-ST-state-01.ps Expires:July, 1996

ACKNOWLEDGEMENTS and AUTHORS:

Many individuals have contributed to the work described in this memo. We thank the participants in the ST Working Group for their input, review, and constructive comments.

We would also like to thank Luca Delgrossi and Louis Berger for allowing us to adopt the text from their [1] document.

We would like to acknowledge inputs from Mark Pullen and his graduate students, Tim O'Malley, Eric Crowley, Muneyoshi Suzuki and many others.

Murali Rajagopal EMail: murali@fbcs.com, Phone: 714-764-2952

Sharon Sergeant EMail:sergeant@xylogics.com, Phone: 617-893-6142

LIST OF REFERENCES:

[1] L. Delgrossi and L. Berger: Internet STream Protocol Version 2 (ST) - Protocol Specification-Version ST2+, RFC 1819 , August 1995.

[2]D. Brand and P. Zafiropulo: On Communicating Finite-State Machines, J.ACM, 30, No.2, April 1983