Network Working Group
INTERNET-DRAFT

D. Auerbach
D. Berg
K. Morneault
Cisco Systems

Expires in six months

25 Feburary 1999

SESSION MANAGER
<draft-ietf-sigtran-session-mgr-00.txt>

Status of This Memo

Abstract

This Internet Draft discusses a protocol layer, Session Manager, that provides network redundancy and redundant Media Gateway Controller configurations for signaling protocol applications. Session Manager provides this support transparently. Network and Media Gateway Controller redundancy are important for providing highly reliable commercial applications that provide transportation of signaling protocols over packet networks.

TABLE OF CONTENTS

1. Introduction

Session Manager provides support for network and Media Gateway Controller redundancy. Session Manager is intended to be used between a Media Gateway (MG) or Signaling Gateway (SG), which interfaces between the circuit world (PSTN) and the packet world (IP/ATM), and a Media Gateway Controller (MGC), which provides call processing. For the rest of this document, Media Gateways and Signaling Gateways will be more generically refered to as gateways.

1.0 Background

Before discussing the functions of Session Manager, it is important to define a session and the other terms associated with a session.

1.1 Terminology

Session - A session is defined by a local IP address and port and a remote IP address and port. It is a "physical" connection between a MGC and a gateway.

Session Group - One or more sessions make up a session group. A session group is used when network redundancy is required. Session Manager manages session group(s) for a signaling application.

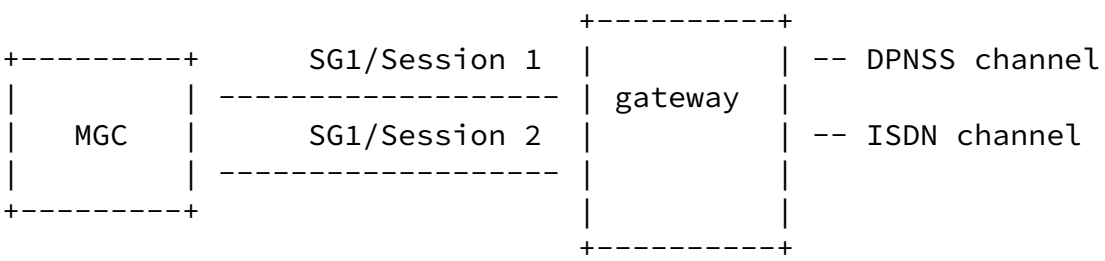
Session Set - A collection of session-group(s) each providing connectivity to a different MGC. A session set is used in a redundant MGC architecture.

Channel - A channel is a physical termination of a signaling line (T1/E1 timeslot, V.35, etc.). It is defined by the protocol family, protocol

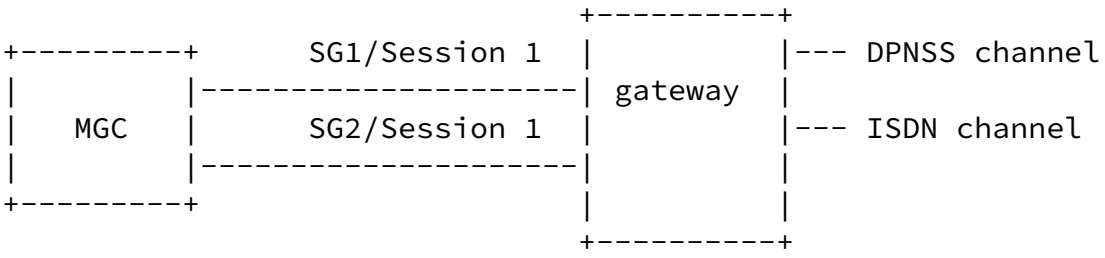
variant and layer terminated (i.e. Q.921). Each channel is associated with a path.

Path - A path consists of one or more channels. A path is defined by protocol family, variant and other relevant characteristics specific to the protocol supported by that path. A path will use a session, session group or a session set. Multiple paths between the same MGC and gateway can share the same sessions. For example, a SS7 path would be equivalent to a linkset and a SS7 channel would be equivalent to a link. An ISDN (FAS) path would be equivalent to a single ISDN D-channel.

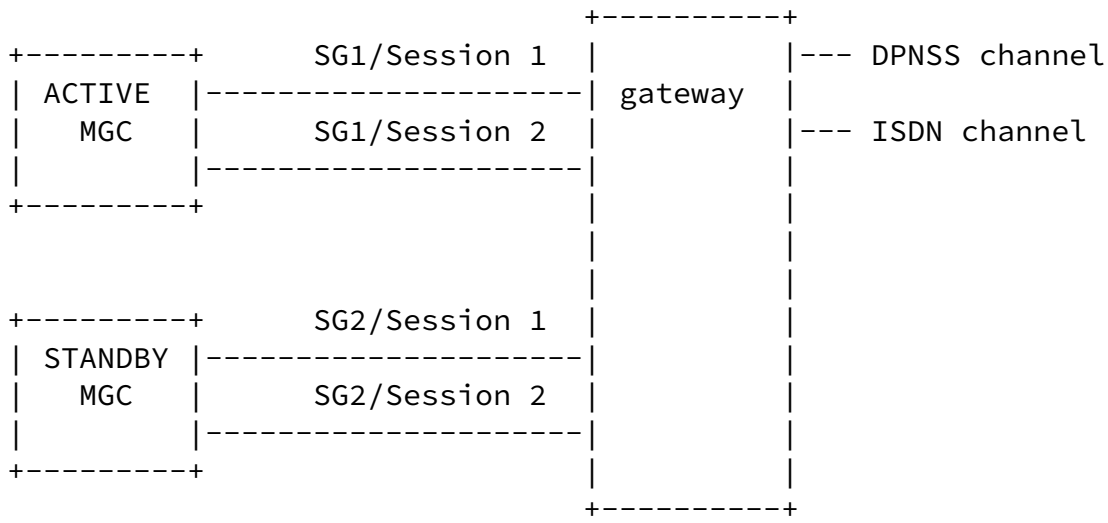
In the following diagram, there is an ISDN and DPNSS path. Both paths are associated with the same session group. The session group (SG1) consists of Session 1 and Session 2. For network redundancy, Session 1 would be on one IP network and Session 2 would be on a separate IP network.



The following diagram is the same except that the ISDN and DPNSS paths are carried over separate session group(s). Most likely, this configuration would be used when there was a desire to separate the DPNSS and ISDN signaling traffic. More sessions could be added to each session group for network redundancy.



The following diagram shows an example of the redundant MGC configuration. In this case, Session Manager provides the ability to manage ACTIVE/-STANDBY orientation on the client (gateway) side. In this configuration, Session Group 1 and Session Group 2 make up a Session Set.

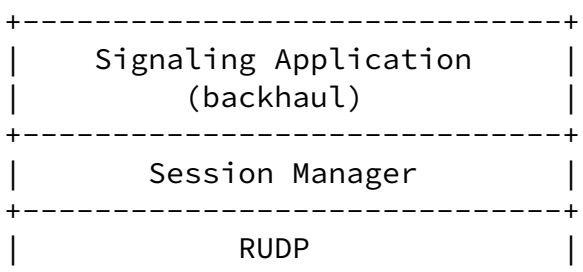


2.0 Problem Definition

There is a need for a lightweight protocol layer below the signaling protocol application to manage sessions used by signaling paths and channels. The use of a session group or session set is desirable to maintain multiple connections between a given gateway (client) and MGC (server). This increases the availability of the gateway. In addition, there is a need for managing redundant MGC hardware configurations transparently to signaling protocol application(s).

The intent is to have Session Manager provide these services to signaling application(s). As shown in the figure below, it sits below the application and above a reliable communication mechanism. Session Manager requires a reliable, connection-oriented transport mechanism

as its lower layer. In the figure below, Session Manager sits on top of Reliable UDP.



The signaling session management layer provides the following functions:

1. Manage sessions in a session group based on priority and availability. Only the client side needs to know priority.
2. Provide a mechanism to support session failure / switchover scenario.
3. Provide a mechanism to support redundant MGC configurations.
   a. notification of ACTIVE / STANDBY state of server (MGC) to client (gateway).
   b. Manage ACTIVE / STANDBY orientation on the client side
4. Allow the application to control the state of the session group, session set or sessions. If multiple applications are using the same session group or session set, it is up to the applications to coordinate this type of management function. In effect, this could provide an operator the ability to adminstratively cause a session switchover.
5. Provide a mechanism for querying the state of the session set, session group or sessions.
6. Provide a mechanism for autonomous notification of session set, session group or session change of service state.
7. Provide a mechanism for querying statistics of session set, session group or sessions.
8. If available, work with reliable transport to circumvent the duplication of packets on switchovers.
9. Monitor the frequency of how often the session is switched and recovered. If the frequency is larger than the threshold trigger sm_unstable_session, an alarm and appropriate operation is taken to report this faulty situation.
10. Avoid the ping-pong effects of failing over sessions in a short period of time.

3.0 Functionality

Session Manager provides two basic functions: the ability to manage sessions in a redundant network configuration and the ability to manage a session set in a redundant MGC configuration. Session Manager can provide this function for one or more applications.

3.1 Messages

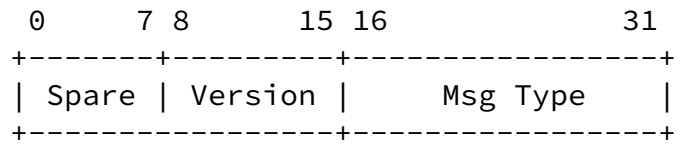
In order to fulfill its functions, the client and server implementations of Session Manager need to pass messages. The four primary messages are the Start, Stop, Active, and Standby message. The Start message is sent by the client to activate a session to make it available for transmission and receiption of PDUs. The Stop message is sent by the client to deactivate a session for transmission and receiption of PDUs. The Active and Standby messages are used by the server to indicate ACTIVE/STANDBY state.

3.2 Message Header

In order to support the above mentioned messages, a header is required. This header will contain a 16-bit message type, 8-bit version number and an 8-bit spare field. The valid messages types are:

* Start Message (0x0)
* Stop Message (0x1)
* Active Message (0x2) - used with redundant MGC configuration
* Standby Message (0x3) - used with redundant MGC configuration
* Q_HOLD Invoke Message (0x4) - used with redundant MGC configuration
* Q_HOLD Response Message (0x5) - used with redundant MGC configuration
* Q_RESUME Invoke Message (0x6) - used with redundant MGC configuration
* Q_RESUME Response Message (0x7) - used with redundant MGC configuration
* Q_RESET Invoke Message (0x8) - used with redundant MGC configuration
* Q_RESET Response Message (0x9) - used with redundant MGC configuration
* PDU (0x8000) - Non Session Manager message from application

Note that messages used by Session Manager (i.e. not passed to the application) are in the 0-0x7fff range.



The initial version number is 1.

3.3 Session States

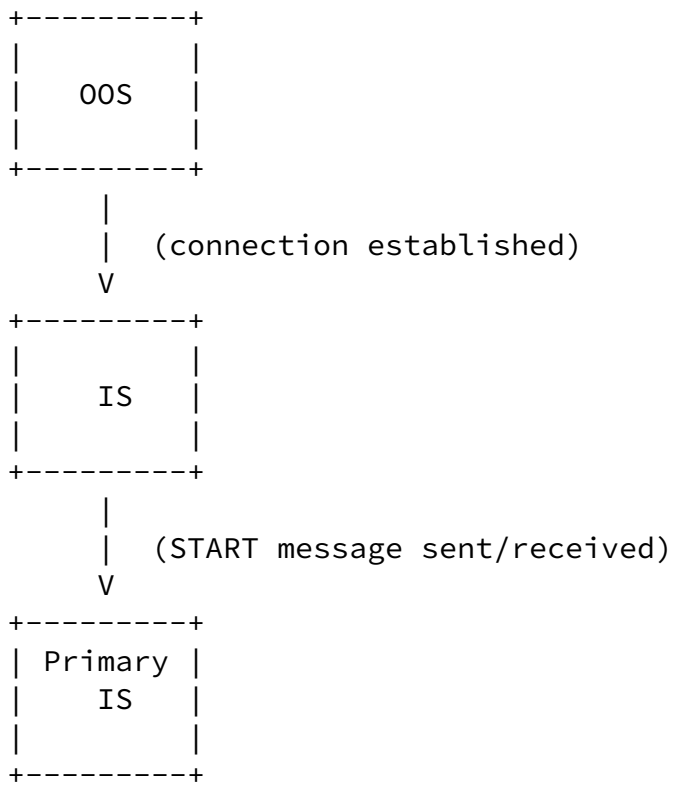
A session can be in one of three states:

1. Out-of-Service: session has been created, but is not connected at transport layer.
2. In-Service: session has been created and is connected at network

layer

3. Primary In-Service: session is In-Service and START message has been transferred from client to server

The figure below shows the state transitions.



3.4 Redundant Network

Session Manager should be used when redundant networks are required between a MGC (server) and a gateway (client). Each session is given a priority. The highest priority session will have a value of 1, lower priorities increase in value.

Session Manager will keep all sessions In-Service. If an In-Service session fails (goes Out-of-Service), Session Manager should periodically retry to bring that session back In-Serivce.

Session Manager only transmits and receives PDUs over the Primary In-Service session in order to avoid sequencing problems. Thus, only one session can be Primary In-Service at a given time. The client side implementation of Session Manager coordinates which In-Service session is primary. The algorithm the client uses to handle session

failovers is described in Section 3.4.2.

3.4.1 Server Side

The server side implementation of Session Manager in a redundant network configuration is fairly straightforward. A server can be in one of four states:

1. Idle: a session group has been created

2. Out-of-Service: a session or sessions have been added to the session group, no Primary In-Service session (there may be one or more In-Service sessions though).
3. In-Service: one of the sessions is in the Primary In-Service state Note: The server side must have received the START message from the client before it will transition to In-Service.
4. Switchover: the Primary In-Service session has failed, sm_switchover_timer set

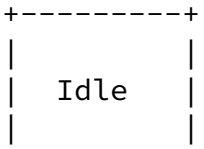
Initially, the server side is in the Idle state after the session group has been created. It transitions to the Out-of-Service state when a session(s) is added.

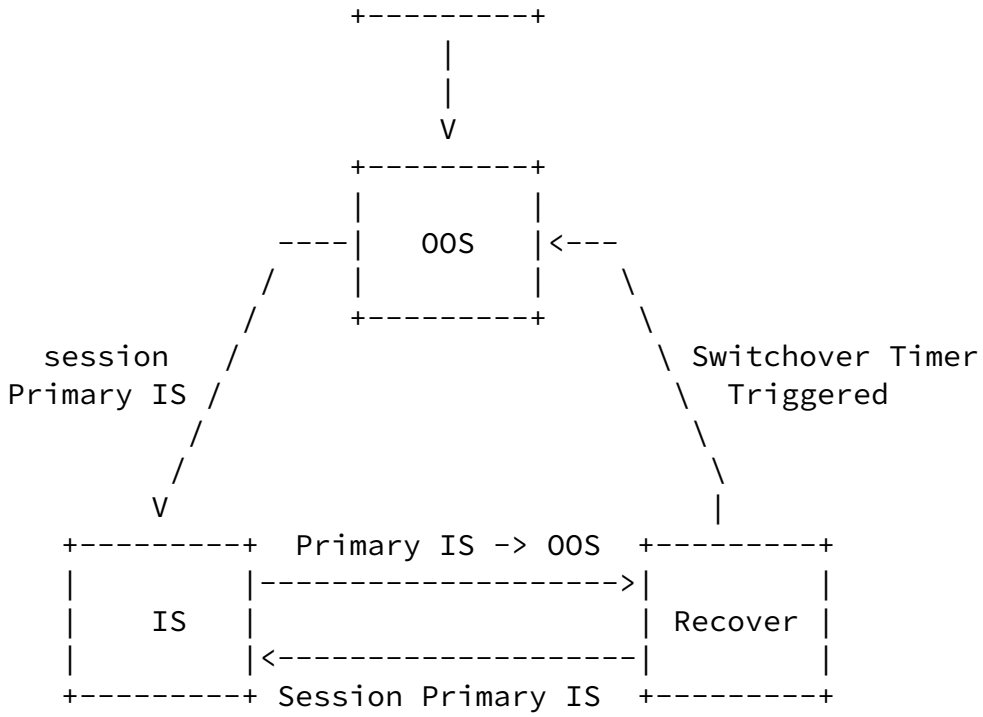
As sessions come In-Service, the server remains Out-of-Service until a session is selected to be primary. This occurs when the client side sends a START message over an In-Service session. At this point, the server side transitions to the In-Service state. It will remain In-Service unless the Primary In-Service session fails. In this case, it transitions to the Switchover state.

In the Switchover state, the server side sets the sm_switchover_timer and waits for the client to make another session primary. If the timer expires, the server side transitions to Out-of-Service.

Note that if the currently active session is a secondary session, the server may want to indicate a degradation in service (i.e. In-Service-Degraded).

The figure below shows an example of the state diagram for a server side implementation.





3.4.2 Client Side

The client side supports five states:

1. Idle: a session group has been created
2. Out-of-Service: a session or sessions have been added to the session group, no active session (meaning session is connected but Start message has not been sent).
3. In-Service: one of the highest priority sessions is Primary In-Service
4. Switchover: the active session has failed, try to bring next highest priority session In-Service.
5. IS-Degraded: one of the lower priority sessions is Primary In-Service

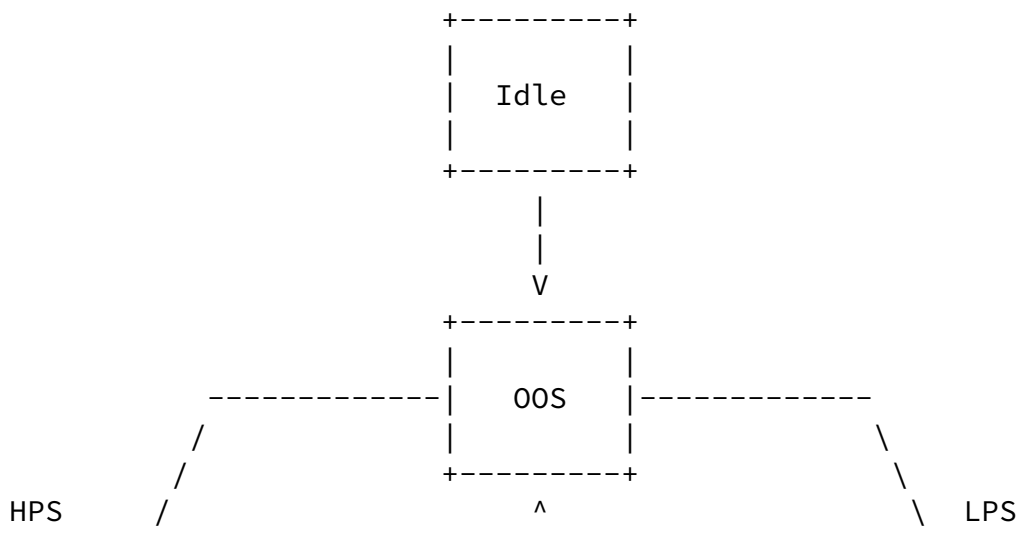
The client side begins in the Idle state when the session group has been created. It transitions to the Out-of-Service state as soon as a session is added. The client side periodically (sm_retry_timer) tries to bring all the sessions to the In-Service state. Once a session is In-Service, the client determines if it should be transitioned to Primary-In-Service based on priority. The session is made Primary-In-Service by passing a START message to the server side on that session. The client can remove the primary designation of a session by passing a STOP message to the server side on that session.

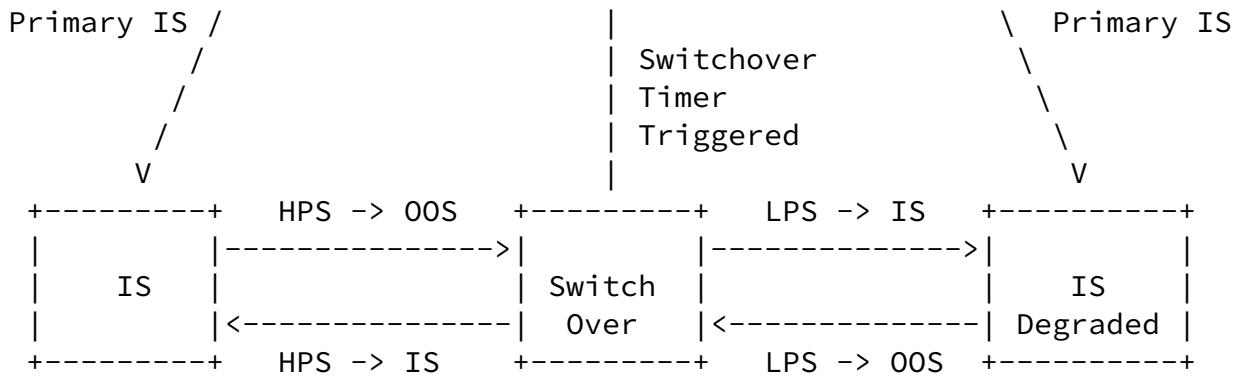
If one of the highest priority sessions is designated as primary, the client transitions to the In-Service. If a lower priority session is designated as primary, the client transitions to the In-Service-Degraded state.

If the primary session fails while In-Service, the client side transitions to the Switchover state. In this switchover state, the client does the following:

a. Find the highest priority session that is In-Service. If there are multiple high priority sessions, it picks the first.
b. If there are no sessions In-Service, it sets the sm_switchover_timer and it tries to bring all sessions In-Service in order of priority. If the switchover timer (sm_switchover_timer) expires, the client transitions to Out-of-Service.
c. Once the client has picked the session, it sends a START on that session to transition it to the Primary-In-Service state. The client then transitions its own state to In-Service or In-Service-Degraded based on whether the session is of the highest priority.

The figure below shows an example of the state diagram for a client side implementation.





* HPS = Highest Priority Session
  LPS = Lower Priority Session

3.4.2.1 Client Side with RUDP

When the client side Session Manager is using RUDP as its transport mechanism, it has the opportunity to initiate a session switchover to another session in a session group. In this configuration, the client does the following:

a. Receive the CONN_FAIL signal from RUDP which indicates the session has failed (disconnected)
b. Find the highest priority session that is In-Service. If there are multiple high priority sessions, it picks the first.
c. If there are no sessions In-Service, it sets the sm_switchover_timer and it tries to bring all sessions In-Service in order of priority. If the switchover timer (sm_switchover_timer) expires, the client transitions to Out-of-Service.
d. Once the client has picked the session, it sends a START on that session to transition it to the Primary-In-Service state. The client then transitions its own state to In-Service or In-Service-Degraded based on whether the session is of the highest priority.
e. If the AUTO_RESET signal has not yet been received from RUDP, Session Manager initiates a state transfer from the failed session to the Primary-In-Service session. RUDP is responsible

for ensuring the messages are retransmitted and avoiding the transmission/reception of duplicate messages.

3.5 Redundant MGC Configuration

In a redundant MGC configuration, a gateway is connected to an ACTIVE MGC and one or more STANDBY MGC(s). On the gateway side (client side), Session Manager is used to manage ACTIVE/STANDBY state for the signaling application. On the MGC side (server side), the signaling application informs Session Manager of its ACTIVE/STANDBY state.

The Primary In-Service state is separated into two states for this mode of operation:

1. Primary In-Service-ACTIVE: session is In-Service and START message has been transferred from client to server, server has transferred ACTIVE notification to client
2. Primary In-Service-STANDBY: session is In-Service and START message has been transferred from client to server, server has transferred STANDBY notification to client

3.5.1 Server Side

The server side implementation of this functionality is very straight-forward. The figure below shows the state diagram. The server side supports three states:

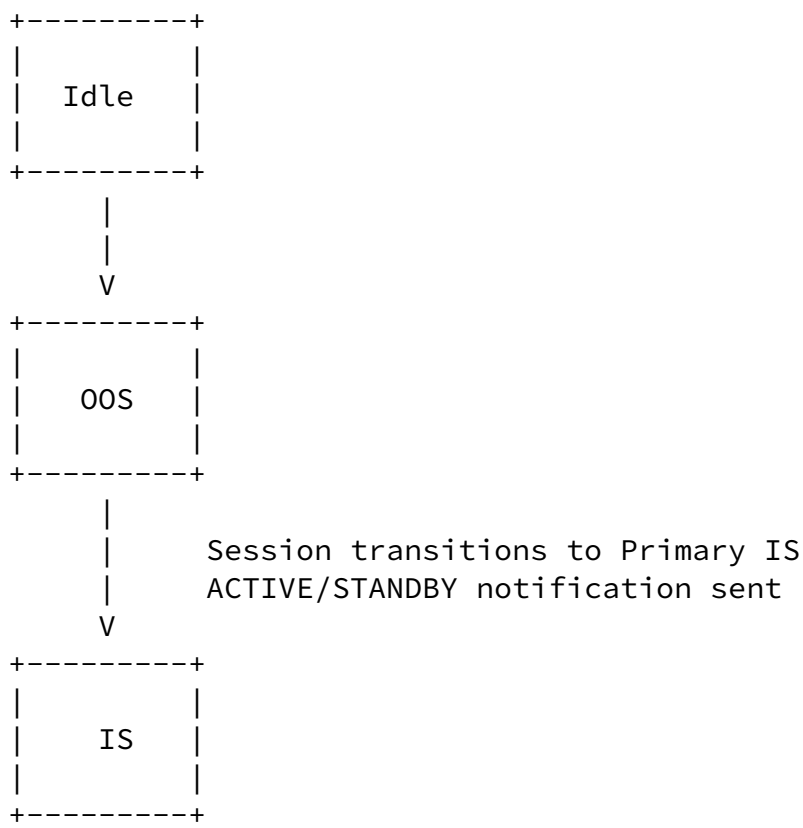
1. Idle: a session set has been created
2. Out-of-Service: session group(s) have been added to the session set, a session In-Service.
3. In-Service: a session is Primary In-Service and ACTIVE/STANDBY state has been sent

The server initializes to the Idle state when the session set is created. When the session set is created, the initial ACTIVE/STANDBY state is passed as an argument. After the session set has been created, the signaling application can change the ACTIVE/STANDBY state at any time.

The server transitions to the Out-of-Service state when a session is created. When a session transitions to the Primary In-Service state, it passes an ACTIVE or STANDBY message to the client. The server then transitions to the In-Service state.

Note: The server periodically sends its state (ACTIVE/STANDBY) based on sm_state_timer.

The figure below shows an example of the state diagram for a server side implementation.



3.5.2 Client Side

The client side implementation is a bit more complex since it needs to act on ACTIVE/STANDBY indications from each server. The figure below shows the state diagram. The client side supports five states:

1. SESS_IDLE State: a Session has been created.
2. SESS_OOS State: a session or sessions have been added to session group, no ACTIVE notification has been received from MGC.
3. SESS_ACTIVE_IS State: a ACTIVE notification has been received over one In-Service session, STANDBY notification has not been received on any available In-Service session(s).
4. SESS_STANDBY_IS State: a STANDBY notification is received, but there is no In-Service Active session available.
5. SESS_FULL_IS State: a session In-Service that has ACTIVE notification and at least one session In-Service that has STANDBY notification.

The client side only accepts PDU messages received from the ACTIVE MGC. The client can be configured to send to just the ACTIVE MGC or both the ACTIVE and STANDBY MGCs.

The client side session initializes to SESS_IDLE state when the

session is created. It transitions to the SESS_OOS state when a session is added, and then waits for an ACTIVE or STANDBY notification. If the notification is ACTIVE, it transitions to SESS_ACTIVE_IS state and informs all the applications using this Session Set that the Session Set is operational. If the notification is STANDBY, it transitions to SESS_STANDBY_IS state. The session that the ACTIVE

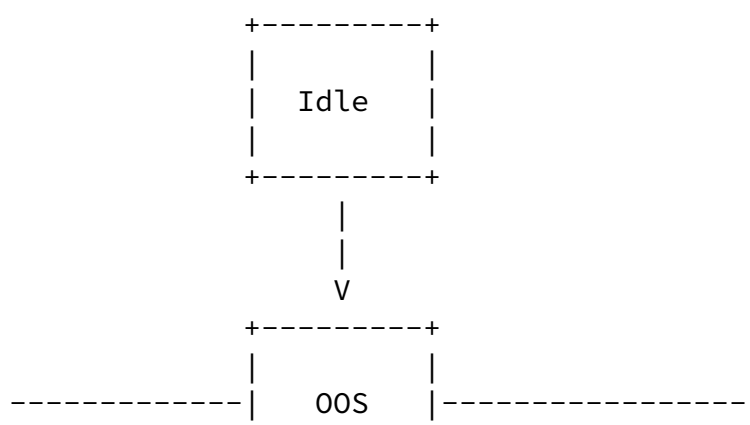
notification is received on will be called the Active session. The session that the STANDBY notifcation is received on will be called the Standby session.

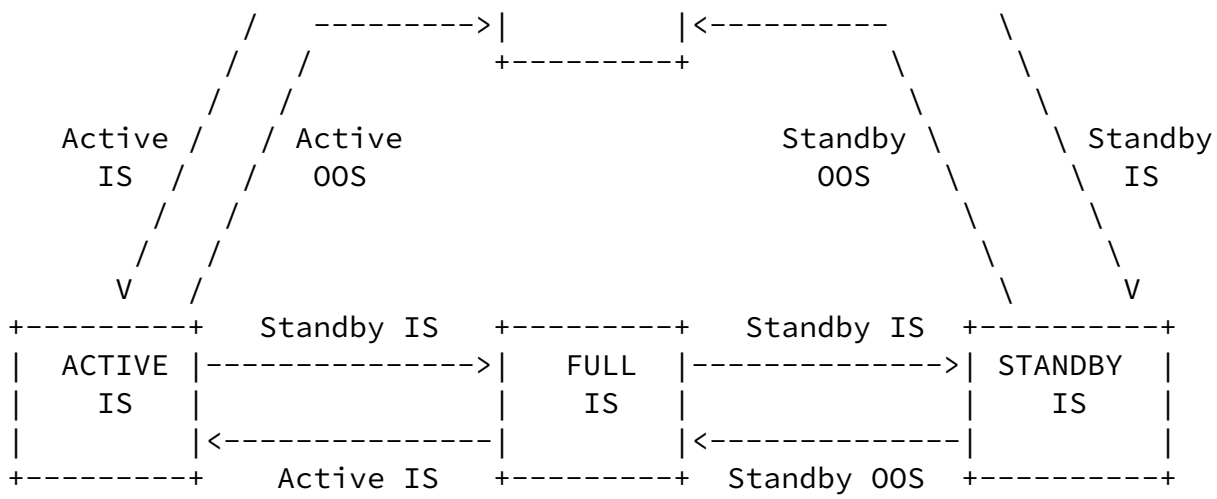
If in the SESS_ACTIVE_IS state, it transitions to the SESS_FULL_IS state if the STANDBY notification is received on any session other than the Active session. If on the other hand, it receives a STANDBY notification on the Active session, it transitions to the SESS_STANDBY_IS state.

If in the SESS_STANDBY_IS state, the client transitions to the SESS_FULL_IS state if the ACTIVE notification is received on a session other than the Standby session. If on the other hand, the client receives an ACTIVE notification on the Standby session, it transitions to the SESS_ACTIVE_IS state.

If in the SESS_FULL_IS state, the client transitions to the SESS_ACTIVE_IS state if an ACTIVE notification is received on the Standby session and no other Standby session(s) is/are available. Likewise, if in the SESS_FULL_IS state, the client transitions to the SESS_STANDBY_IS state if a STANDBY notification is received on the Active session.

The figure below shows an example of the state diagram for a client side implementation.





3.5.2.1 Session Queueing

In the redundant MGC configuration, there is a requirement to be able to queue messages on the client side while the MGC is in the process of failover/switchover. The ACTIVE MGC requests that messages be queued using the Q_HOLD Invoke message. The Q_HOLD Response message indicates to the ACTIVE MGC that messages are being queued. At this point, the MGC can freeze its state, transfer that state to the STANDBY MGC, then the MGCs will transition states with the STANDBY MGC becoming ACTIVE.

The Q_RESUME Invoke message is used to request that the client begin sending the queued messages to the ACTIVE MGC. The Q_RESET Invoke message is used to request that the client clear (delete) all queued messages.

In this SESS_ACTIVE_IS state, if Q_RESET Invoke notification is received the Q_HOLD condition of all the queues associated with the session are cleared and any queued messages are deleted. On the other hand, if Q_RESUME Invoke notification is received, the Q_HOLD condition of all the queues associated with the session are cleared and any queued messages are transferred. If Q_HOLD condition is not set, a Q_RESET Invoke or Q_RESUME Invoke notification are ignored. The same action is taken when Q_RESET Invoke or Q_RESUME Invoke notifications are received in any other state except SESS_IDLE and SESS_OOS states.

In this SESS_FULL_IS state, if the client receives a Q_HOLD Invoke notification it will not change state, but the status of all the queues associated with the session is set to Q_HOLD. The Q_HOLD Invoke notification is only acted on in the SESS_FULL_IS state. Once Q_HOLD

condition is set for queues, it remains until cleared by Q_RESUME Invoke or Q_RESET Invoke messages. No state transition clears the Q_HOLD condition.

3.6 Session Manager Timers

Session Manager uses the following timers:

* sm_retry_timer - used to retry connection of Out-of-Service sessions (default is 5 seconds)
* sm_switchover_timer - amount of time allowed for a switchover to another session before transitioning to Out-of-Service state (default is 3 seconds)
* sm_state_timer (redundant MGC configuration) - the server will send state (ACTIVE/STANDBY) periodically based on this timer (default is 60 seconds)

3.7 Session Manager Counters

* sm_unstable_session - alarm trigger - if 20 session recoveries in 60 minutes or less

3.8 Session Manager Statistics

Statistics should be maintained on a session, session group and session set basis. The application should be able to query and clear the statistics.

* number of bytes transmitted
* number of PDUs transmitted
* number of bytes received
* number of PDUs received
* number of session switchovers (kept at a session group level)
* number of MGC switchovers/failovers - redundant MGC configuration (kept at a session set level)

4.0 Author's Addresses

David Auerbach
Cisco Systems
13615 Dulles Technology Drive
Herndon, VA 20171

Tel: +1-703-484-3464
Email: dea@cisco.com

USA

Diane Berg
Cisco Systems
13615 Dulles Technology Drive
Herndon, VA 20171
USA

Tel: +1-703-484-3461
Email: dberg@cisco.com

Ken Morneault
Cisco Systems
13615 Dulles Technology Drive
Herndon, VA 20171
USA

Tel: +1-703-484-3323
Email: kmorneau@cisco.com