NSIS
Internet-Draft
Intended status: Informational
Expires: January 27, 2010

T. Tsenov
H. Tschofenig
Nokia Siemens Networks
X. Fu, Ed.
Univ. Goettingen
C. Aoun
E. Davies
Folly Consulting

July 27, 2009

GIST State Machine
draft-ietf-nsis-ntlp-statemachine-08.txt

Status of this Memo

Copyright Notice

Abstract

This document describes the state machines for the General Internet Signaling Transport (GIST). The states of GIST nodes for a given flow and their transitions are presented in order to illustrate how GIST

may be implemented.

Table of Contents

1. Introduction

The state machines described in this document are illustrative of how the GIST protocol defined in [1] may be implemented for the GIST nodes in different locations of a flow path. Where there are differences - [1] is authoritative. The state machines are informative only. Implementations may achieve the same results using different methods.

There are two types of possible entities for GIST signaling:

- GIST querying node - GIST node that initiates the discovery of the next peer;

- GIST responding node - GIST node that is the discovered next peer;

We describe a set of state machines for these entities to illustrate how GIST may be implemented.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [2].

3. Notational conventions used in state diagrams

The following text is reused from [3] and the state diagrams are based on the conventions specified in [4], Section 8.2.1. Additional state machine details are taken from [5].

The complete text is reproduced here:

State diagrams are used to represent the operation of the protocol by a number of cooperating state machines each comprising a group of connected, mutually exclusive states. Only one state of each machine can be active at any given time.

All permissible transitions between states are represented by arrows, the arrowhead denoting the direction of the possible transition. Labels attached to arrows denote the condition(s) that must be met in order for the transition to take place. All conditions are expressions that evaluate to TRUE or FALSE; if a condition evaluates to TRUE, then the condition is met. The label UCT denotes an unconditional transition (i.e., UCT always evaluates to TRUE). A transition that is global in nature (i.e., a transition that occurs from any of the possible states if the condition attached to the arrow is met) is denoted by an open arrow; i.e., no specific state is

identified as the origin of the transition. When the condition associated with a global transition is met, it supersedes all other exit conditions including UCT. The special global condition BEGIN supersedes all other global conditions, and once asserted remains asserted until all state blocks have executed to the point that variable assignments and other consequences of their execution remain unchanged.

On entry to a state, the procedures defined for the state (if any) are executed exactly once, in the order that they appear on the page. Each action is deemed to be atomic; i.e., execution of a procedure completes before the next sequential procedure starts to execute. No procedures execute outside of a state block. The procedures in only one state block execute at a time, even if the conditions for execution of state blocks in different state machines are satisfied, and all procedures in an executing state block complete execution before the transition to and execution of any other state block occurs, i.e., the execution of any state block appears to be atomic with respect to the execution of any other state block and the transition condition to that state from the previous state is TRUE when execution commences. The order of execution of state blocks in different state machines is undefined except as constrained by their transition conditions. A variable that is set to a particular value in a state block retains this value until a subsequent state block executes a procedure that modifies the value.

On completion of all of the procedures within a state, all exit conditions for the state (including all conditions associated with global transitions) are evaluated continuously until one of the conditions is met. The label ELSE denotes a transition that occurs if none of the other conditions for transitions from the state are met (i.e., ELSE evaluates to TRUE if all other possible exit conditions from the state evaluate to FALSE). Where two or more exit conditions with the same level of precedence become TRUE simultaneously, the choice as to which exit condition causes the state transition to take place is arbitrary.

In addition to the above notation, there are a couple of clarifications specific to this document. First, all boolean variables are initialized to FALSE before the state machine execution begins. Second, the following notational shorthand is specific to this document:

<variable> = <expression1> | <expression2> | ...

Execution of a statement of this form will result in <variable> having a value of exactly one of the expressions. The logic for which of those expressions gets executed is outside of the state

machine and could be environmental, configurable, or based on another state machine such as that of the method.

4. State Machine Symbols

( )
: Used to force the precedence of operators in Boolean expressions and to delimit the argument(s) of actions within state boxes.

;
: Used as a terminating delimiter for actions within state boxes. Where a state box contains multiple actions, the order of execution follows the normal English language conventions for reading text.

=
: Assignment action. The value of the expression to the right of the operator is assigned to the variable to the left of the operator. Where this operator is used to define multiple assignments, e.g., a = b = X the action causes the value of the expression following the right-most assignment operator to be assigned to all of the variables that appear to the left of the right-most assignment operator.

!
: Logical NOT operator.

&&
: Logical AND operator.

||
: Logical OR operator.

if...then...
: Conditional action. If the Boolean expression following the if evaluates to TRUE, then the action following the then is executed.

{ statement 1, ... statement N }
: Compound statement. Braces are used to group statements that are executed together as if they were a single statement.

!=
: Inequality. Evaluates to TRUE if the expression to the left of the operator is not equal in value to the expression to the right.

==
: Equality. Evaluates to TRUE if the expression to the left of the operator is equal in value to the expression to the right.

>
Greater than. Evaluates to TRUE if the value of the expression to the left of the operator is greater than the value of the expression to the right.

<=
Less than or equal to. Evaluates to TRUE if the value of the expression to the left of the operator is either less than or equal to the value of the expression to the right.

++
Increment the preceding integer operator by 1.

+
Arithmetic addition operator.

&
Bitwise AND operator.

5. Common Rules

Throughout the document we use terms defined in the [1], such as Query, Response, Confirm.

State machine represents handling of GIST messages that match a Message Routing State's MRI, NSLPID and SID and with no protocol errors. Separate parallel instances of the state machines should handle messages for different Message Routing States.

The state machine states represent the upstream/downstream peers states of the Message Routing State.

For simplification not all objects included in a message are shown. Only those that are significant for the case are shown. State machines do not present handling of messages that are not significant for management of the states.

Presented in this document state machines do not cover all functions of a GIST node. Functionality of message forwarding, transmission of NSLP data without MRS establishment and providing of the received messages to the appropriate MRS, we refer as "Lower level pre-processing" step. Pre-processing provides to the appropriate MRS FSM only the messages which are matched against waiting Query/Response cookies, or established MRS MRI+NSLPID+SID primary key. This is presented by "rx_*" events in the state machines.

Management of a MA is considered in the document (e.g., tg_Establish_MA, tg_MA_established events), but its FSM is not

explicitly presented.

5.1 Common Procedures

Tg_SendMsg:
NSLP/GIST API message that request transmission of a NSLP message.

Tg_SetStateLifetime(time_period):
NSLP/GIST API message providing info for the Lifetime of an RS, required by the application. "Time_period = 0" represents the cancellation of established RSs/MAs (invoked by NSLP application).

Tg_MessageStatus:
NSLP/GIST API message informing NSLP application of unsuccessful delivery of a message

Tg_RecvMsg:
NSLP/GIST API message that provides received message to the NSLP

Tg_NetworkNotification:
NSLP/GIST API message that informs NSLP for change in MRS

Tx_Query:
Transmit of Query message

Tx_Response:
Transmit of Response message

Tx_Confirm:
Transmit of Confirm message

Rx_Query:
Receive of Query message

Rx_Response:
Receive of Response message

Rx_Confirm:
Receive of Confirm message

Tx_Error:
Transmit of Error message

Rx_Error:
Receive of Error message

Queue NSLP info:
Save NLSP messages in a queue until a required MA association is

established

Tx_Data:
Transmit of Data message

Rx_Data:
Receive of Data message

T_Inactive_QNode:
Message Routing State lifetime timer in Querying Node

T_Expired_RNode:
Message Routing State lifetime timer in Responding Node

T_Refresh_QNode:
Message Routing State refresh timer in Querying Node

T_No_Response:
Timer for the waiting period for Response message in Querying Node

T_No_Confirm:
Timer for the waiting period for Confirm message in Responding Node

Install downstream/upstream MRS:
Install new Message Routing State and save the corespoding peer state info (IP address and UDP port or pointer to the used MA) for the current Message Routing State or update the coresponding peer state info.

DELETE MRS:
Delete installed downstream/upstream peer's info for the current Message Routing State and delete the Message Routing State if required.

Established MA:
A Message Association (MA) is established between the current node and its upstream peer. The initiator for the establishment is the upstream peer.

Re-use existing MA:
An existing MA between the current node and its peer is re-used.

DELETE MA:
Delete/disconnect used MA.

Stop using shared MA:
Stop using shared MA. If the shared MA is no more used by any

other MRSs, it depends on the local policy whether it is deleted or kept.

REFRESH MRS:
Refreshes installed MRS.

Tg_MA_Error:
Error event with used MA.

Tg_InvalidRoutingState:
Notification from NSLP application for path change

Tg_Establish_MA:
Trigers establishment of MA.

Tg_MA_Established:
MA has been successfully established.

Tg_ERROR:
General Error event / system level error.

No_MRS_Installed:
Error response, send by the Responding node indicating lost Confirm message.

5.2 Common Variables

It is assumed that the type of mode and destination info (which need to be taken from the application parameters and local GIST policy)is provided. This is represented by the common variables Dmode, Cmode, MAinfo, MApresent and Refresh.

Cmode:
The message MUST be transmitted in Cmode. This is specified by "Message transfer attributes" set to any of the following values:

"Reliability" is set to TRUE.

"Security" is set to values that request secure handling of a message.

"Local processing" is set to values that require services offered by Cmode (e.g., congestion control). [1]

Dmode:
The message MUST be transmitted in Dmode. This is specified by local policy rules and in case that the "Message transfer attributes" are not set to any of the following values:

"Reliability" is set to TRUE.

"Security" is set to values that request special security handling of a message.

"Local processing" is set to values that require services offered by Cmode [1]

MAinfo:
GIST message parameters describing the required MA or proposed MA e.g. "Stack-proposal" and "Stack-Configuration-Data".

NSLPdata:
NSLP application data.

RespCookie:
Responder Cookie that is being sent by the Responding node with the Response message in case that its local policy requires a confirmation from the querying node.

ConfirmRequired:
Confirm message is required by the local policy rule for installation of the new MRS.

NewPeer:
Response message is received from new responding peer.

MAexist:
Existing MA will be reused.

CheckPeerInfo:
The sender of the received data message is matched against the installed peer info in the MRS.

UpstreamPeerInstalled:
Upstream peer info is installed in the MRS.

6. State machines

The following section presents the state machine diagrams of GIST peers.

6.1 Diagram notations

(see the .pdf version for missing diagram or refer to Appendix A if reading the .txt version)

Figure 1: Diagram notations

6.2 State machine for GIST querying node

The following is a diagram of the GIST querying node state machine. Also included is clarification of notation.

(see the .pdf version for missing diagram or refer to Appendix A.1 if reading the .txt version)

Figure 1: GIST Querying Node State Machine

*) Response and Comfirm messages might be send either in Dmode or Cmode, before or after MA establishment depending on node s local 3-way handshake policy and the availability of MAs to be reused. See draft for details.

**) Depending on the local policy NSLPdata might be send as payload of Query and Confirm messages. (piggybacking)

1) Initial request from NSLP is received, which triggers Query messages requesting either D_mode or C_mode. Depending on node s local policy NSLP data might be piggybacked in the Query requesting D_mode. Query may carry Mainfo if C_mode transport is needed.
2) Response message is received. If C_mode connection must be established and there is no available MA to be reused, MA establishment is initiated and waited to be completed.
3) Response message is received. If D_mode connection is requested or available MA can be reused for requested C_mode, the MRS is established.
4) No_Response timer expires. Query is resent.
5) No_Response timer expires and maximum number of retries has been reached. NSLP application is notified for the GIST peer discovery failure.
6) NSLP data is queued, because downstream peer is not discovered or required MA is still not established.
7) Data message is received. It is checked if its sender matches the installed downstream peer info in the MRS and then processed. In WaitResponse state, this event might happen in the process of MA upgrade, when the downstream peer is still not aware of establishment of the new MA.
8) Provided NSLP data is sent via Data message towards downstream GIST peer.
9) Refresh_QNode timer expires. Query message is sent.
10) Response message from the downstream GIST peer is received. The peer is not changed. MRS is refreshed (Refresh_QNode timer is restarted).
11) Path change detected. Response message from a new downstream GIST peer is received. D_mode is requested or existing MA can be reused

for requested C_mode.
12) Path change detected. Response message from a new downstream GIST peer is received. A new MA must be established for requested C_mode.
13) Requested by NSLP application transport parameters requires upgrade of established MRS from D_mode/C_mode to C_mode. NSLP application notifies GIST for path change. Downstream GIST peer discovery is initiated.
14) Sent Confirm message has not been received by downstream GIST peer. Confirm message is resent.
15) MRS lifetime expires. Notification by NSLP application that MRS is no longer needed.
16) MA is established.
17) MA establishment failure.

6.3 State machine for GIST responding node

The following is a diagram of the GIST responding node state machine. Also included is clarification of notation.

(see the .pdf version for missing diagram or refer to Appendix A.2 if reading the .txt version)

Figure 3: GIST Responding Node State Machine

1) A Query message is received. Explicit Confirm message is required for MRS installation, based on the local policy. Query message might carry piggybacked NSLP data which is provided to the NSLP application.
2) A Query message is received. MRS is installed immediately, based on the local policy. Query message might carry piggybacked NSLP data which is provided to the NSLP application.
3) Confirm message is received which causes installation of the complete MRS or just installation of the used MA as a upstream peer info.
4) Sent Response message has not been received by upstream GIST peer. Response message is resent.
5) In case of lost Confirm message, data messages might be received from the upstream GIST node (it is unaware of the lost Confirm message). Response indicating the loss of the Confirm is sent back to the upstream GIST node.
6) No_Confirm timer expires. Note that all cases of lost handshake GIST messages are handled only by GIST querying node via resend of Query message.
7) NSLP data is sent if discovery process is successfully accomplished or is queued if Confirm message is still expected to confirm establishment of MA.
8) Data messages are accepted only if complete MRS is installed,

e.g., there is installed upstream peer info. If not, then Confirm message is expected and data message won t be accepted. Response indicating the loss of the Confirm is sent back to the upstream GIST node.

9) Change of the upstream GIST node (e.g., path change). Local policy does not need explicit Confirm message for MRS installation. MRS data is updated.

10) Change of the upstream GIST node or request for change of the used connection mode (from D_mode/C_mode to better C_mode). Local policy requires explicit Confirm message for MRS installation.

11) Request for change of the used connection mode (from D_mode/C_mode to better C_mode). Local policy does not need explicit Confirm message for MRS installation. MRS data is updated.

12) MRS lifetime expires. Notification by NSLP application that MRS is no longer needed.

7. Security Considerations

This document does not raise new security considerations. Any security concerns with GIST are likely reflected in security related NSIS work already (such as [1] or [6]).

8. IANA Considerations

This document has no actions for IANA.

9. Contributors

Christian Dickmann contributed to refining of the state machine since 01 version.

10. Acknowledgments

The authors would like to thank Robert Hancock, Ingo Juchem, Andreas Westermaier, Alexander Zrim, Julien Abeille Youssef Abidi and Bernd Schloer for their insightful comments.

11. References

11.1. Normative References

[1] Schulzrinne, H., "GIST: General Internet Signaling Transport", draft-ietf-nsis-ntlp-20 (work in progress), June 2009.

[2] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

11.2. Informative References

[3] Vollbrecht, J., Eronen, P., Petroni, N., and Y. Ohba, "State Machines for Extensible Authentication Protocol (EAP) Peer and Authenticator", RFC4137, August 2005.

[4] Institute of Electrical and Electronics Engineers, "Standard for Local and Metropolitan Area Networks: Port-Based
Network Access Control", IEEE 802-1X-2004, December 2004.

[5] Fajardo, V., Ohba, Y. and R. Lopez, "State Machines for Protocol for Carrying Authentication for Network Access (PANA)",
draft-ietf-pana-statemachine-12 (work in progress), April 2009.

[6] Tschofenig, H. and D. Kroeselberg, "Security Threats for NSIS", RFC 4081, June 2005.

Appendix A. ASCII versions of state diagrams

This appendix contains the state diagrams in ASCII format. Please use the PDF version whenever possible: it is much easier to understand.

For each state there is a separate table that lists in each row:
- an event that triggers a transition,
- actions taken as a result of the incoming event,
- and the new state at which the transitions ends.

A.1. State machine for GIST querying node (Figure 2)

```
-----------
State: IDLE
-----------

Condition                 Action                      State      Note
------------------------+--------------------------+-----------+---
tg_SendMsg               |tx_Query                  |Wait       |1)
                         |start T_No_Response       |Response   |**
                         |Queue NSLP data           |           |
                         |                          |           |
Tg_ERROR                 |Delete MRS                |IDLE       |
                         |IF (MA is used)           |           |
                         |  ((Delete MA)||          |           |
                         |  (Stop using shared MA))|           |
                         |Tg_NetworkNotification    |           |
                         |                          |           |
------------------------+--------------------------+-----------+---



-----------
State: WaitResponse
-----------

Condition                 Action                      State      Note
------------------------+--------------------------+-----------+---
rx_Response(MAinfo)&&    |tg_Establish_MA           |Wait MA    |*
(!MAexist)               |(tx_Confirm)              |Establish. |2)
                         |                          |           |
                         |                          |           |
rx_Response)||           |Install MRS               |Established|3)
(rx_Response(MAinfo)&&   |IF (RespCookie)           |Downstream |
(MAexist))               |   tx_Confirm(RespCookie)|MRS        |
                         |tx_Data(Queued NSLP data)|           |
                         |                          |           |
```

(timeout T_No_Response) &&(!MaxRetry)	Tx_Query restart T_No_Response	Wait Response	4)
(timeout T_No_Response) &&(MaxRetry)	tg_MessageStatus	IDLE	5)
tg_SendMsg	Queue NSLP data	Wait Response	6)
rx_Data	IF(CheckPeerInfo) tg_RecvMsg to Appl.	Wait Response	7)
Tg_ERROR	(Delete MRS) IF (MA is used) ((Delete MA)|| (Stop using shared MA)) Tg_NetworkNotification	IDLE	

State: Established Downstream MRS

Condition	Action	State	Note
tg_SendMsg	tx_Data restart T_Inactive_QNode	Established Downstream MRS	8)
timeout T_Refresh_QNode	tx_Query	Established Downstream MRS	9)
(rx_Response)&& (!NewPeer)	Refresh MRS restart T_Inactive_QNode	Established Downstream MRS	10)

(rx_Response)|| (rx_Response(Mainfo)&& (MAexist)))&&(NewPeer)	IF (MA is used)
	(Delete MA)||
	(Stop using shared MA)
	Install MRS
	restart T_Inactive_QNode
	IF (RespCookie)
		Established
		Downstream
		MRS
			11)

	tx_Confirm(RespCookie)		
(rx_Response(MAinfo)&& (NewPeer)&&(!MA_exist))	((Delete MA)|| (Stop using shared MA)) tg_Establish_MA (tx_Confirm)	Wait MA Establish.	12) *
((tg_SendMsg)&&(Cmode)&& (!MAexist))|| (tg_MA_error)|| (tg_InvalidRoutingState)	tx_Query Queue NSLP data	Wait Response	13)
rx_Response(No_MRS_ installed)	tx_Confirm(RespCookie) tx_Data(Queued NSLP data)	Established Downstream MRS	14)
(timeout T_Inactive_ QNode)|| (tg_SetStateLifetime(0))	Delete MRS IF (MA is used) (Delete MA)|| (Stop using shared MA) Tg_NetworkNotification	IDLE	15)
rx_Data	IF(CheckPeerInfo) tg_RecvMsg to Appl.	Established Downstream MRS	7)
Tg_ERROR	(Delete MRS) IF (MA is used) ((Delete MA)|| (Stop using shared MA)) Tg_NetworkNotification	IDLE	

State: Wait MA Establishment

Condition	Action	State	Note
tg_MA_Established	Install MRS (tx_Confirm) tx_Data(Queued NSLP data)	Established Downstream MRS	16) *
tg_MA_error	Delete MRS tg_MessageStatus	IDLE	17)

	|	|	|
tg_SendMsg	|Queue NSLP data	|Wait MA	|6)
	|	|Establish.	|
	|	|	|
Tg_ERROR	|Delete MRS	|IDLE	|
	|IF (MA is used)	|	|
	| ((Delete MA)||	|	|
	| (Stop using shared MA))|	|	|
	|Tg_NetworkNotification	|	|
	|	|	|

Figure 4

A.2. State Machine for GIST responding node (Figure 3)

State: IDLE

Condition	Action	State	Note
rx_Query&&	|tx_Response	|Wait	|1)
(ConfirmRequired)	|start T_No_Confirm	|Confirm	|
	|IF(NSLPdata)	|	|
	| tg_RecvMsg(NSLPdata)|	|	|
	| to Appl.|	|	|
	|	|	|
rx_Query&&	|tx_Response	|Established	|2)
(!ConfirmRequired)	|Install MRS	|Upstream	|
	|IF(NSLPdata)	|MRS	|
	| tg_RecvMsg(NSLPdata)|	|	|
	| to Appl.|	|	|
	|	|	|

State: WAIT CONFIRM

Condition	Action	State	Note
rx_Confirm	|Install Upstream MRS	|Established	|3)
	|	|Upstream	|
	|	|MRS	|
	|	|	|

rx_Query&& (ConfirmRequired)	tx_Response start T_No_Confirm IF(NSLPdata) tg_RecvMsg(NSLPdata) to Appl.	Wait Confirm	4)
rx_Data	tx_Error(No_MRS_ installed)	Wait Confirm	5)
timeout T_No_Confirm		IDLE	6)

State: Established Upstream MRS

Condition	Action	State	Note
tg_SendMsg	IF(!UpstreamPeerInfo) Queue NSLP data ELSE tx_Data	Established Upstream MRS	7)
rx_Data	IF(UpstreamPeerInfo) (tg_RecvMsg to Appl.) &&(restart_T_Expire_ RNode) ELSE tx_Error(No_MRS_ installed)	Established Upstream MRS	8)
rx_Query	IF (NewPeer) Update UpstreamPeerInfo tx_Response restart T_Expire_RNode	Established Upstream MRS	9)
(rx_Query)&& (ConfirmRequired)	Delete MRS tx_Response start T_No_Confirm IF(MA is used) (Delete MA)|| (Stop using shared MA) IF(NSLPdata) tg_RecvMsg(NSLPdata) to Appl.	Wait Confirm	

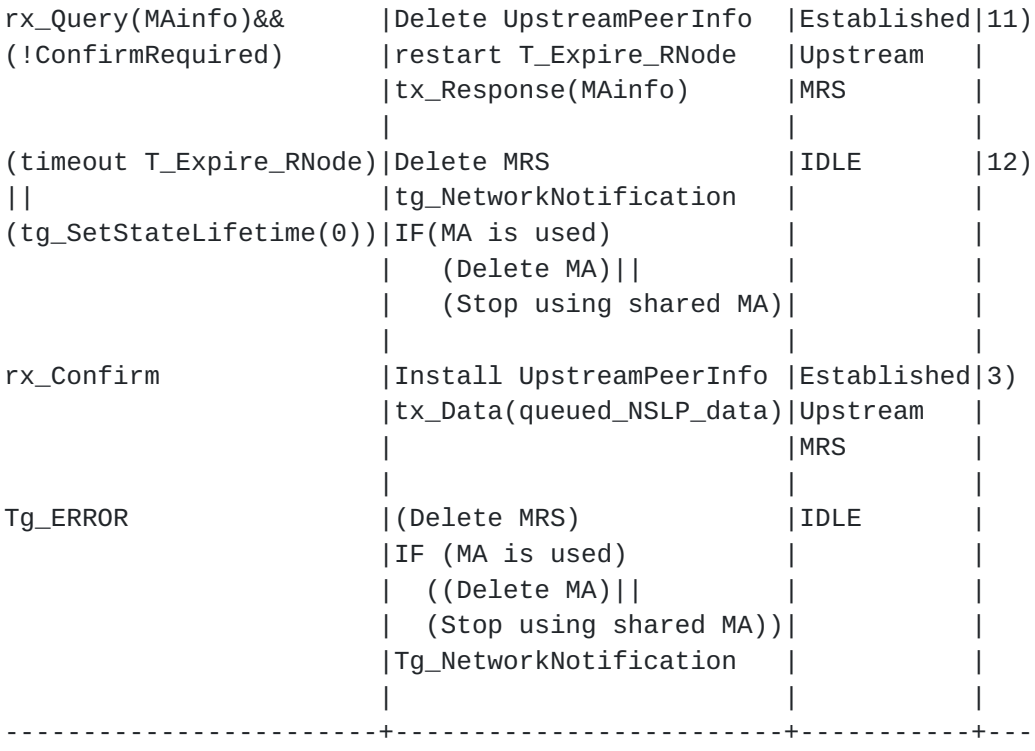
rx_Query(MAinfo)&&
(!ConfirmRequired)
Delete UpstreamPeerInfo
restart T_Expire_RNode
tx_Response(MAinfo)
Established
Upstream
MRS
11)
(timeout T_Expire_RNode)
||
(tg_SetStateLifetime(0))
Delete MRS
tg_NetworkNotification
IF(MA is used)
(Delete MA)||
(Stop using shared MA)
IDLE
12)
rx_Confirm
Install UpstreamPeerInfo
tx_Data(queued_NSLP_data)
Established
Upstream
MRS
3)
Tg_ERROR
(Delete MRS)
IF (MA is used)
((Delete MA)||
(Stop using shared MA))
Tg_NetworkNotification
IDLE

Figure 5

Authors' Addresses

Tseno Tsenov
Sofia, Bulgaria

Email: tseno.tsenov@mytum.de

Hannes Tschofenig
Nokia Siemens Networks
Linnoitustie 6
Espoo 02600
Finland

Email: Hannes.Tschofenig@nsn.com

Xiaoming Fu (Editor)
University of Goettingen
Computer Networks Group
Goldschmidtstr. 7
Goettingen 37077
Germany

Email: fu@cs.uni-goettingen.de

Cedric Aoun
Paris, France

Email: cedric@caoun.net

Elwyn B. Davies
Folly Consulting
Soham, Cambs, UK

Phone: +44 7889 488 335
Email: elwynd@dial.pipex.com