

Internet Engineering Task Force (IETF)  
Request for Comments: 5972  
Category: Informational  
ISSN: 2070-1721

T. Tsenov  
H. Tschofenig  
Nokia Siemens Network  
X. Fu, Ed.  
Univ. Goettingen  
C. Aoun  
Consultant  
E. Davies  
Folly Consulting  
October 2010

## General Internet Signaling Transport (GIST) State Machine

### Abstract

This document describes 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.

### Status of This Memo

This document is not an Internet Standards Track specification; it is published for informational purposes.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Not all documents approved by the IESG are a candidate for any level of Internet Standard; see Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at <http://www.rfc-editor.org/info/rfc5972>.

### Copyright Notice

Copyright (c) 2010 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must

include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

## Table of Contents

1. Introduction .....	3
2. Terminology .....	3
3. Notational Conventions Used in State Diagrams .....	3
4. State Machine Symbols .....	5
5. Common Rules .....	6
5.1. Common Procedures .....	7
5.2. Common Events .....	8
5.3. Common Variables .....	9
6. State Machines .....	11
6.1. Diagram Notations .....	12
6.2. State Machine for GIST Querying Node .....	12
6.3. State Machine for GIST Responding Node .....	16
7. Security Considerations .....	18
8. Acknowledgments .....	18
9. References .....	18
9.1. Normative References .....	18
9.2. Informative References .....	18
Appendix A. State Transition Tables .....	20
A.1. State Transition Tables for GIST Querying Node .....	20
A.2. State Transition Tables for GIST Responding Node .....	24

## 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].

RFC 4137 [3] reproduced the following text from Section 8.2.1 of IEEE 802-1X-2004 [4].

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 it 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 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. That is, 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 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 [1], such as Query, Response, and Confirm.

The state machine represents the handling of GIST messages that match a Message Routing State's Message Routing Information (MRI), NSIS Signaling Layer Protocol identifier (NSLPID), and session identifier (SID) and with no protocol errors. Separate parallel instances of the state machines should handle messages for different Message Routing States (MRSs).

The state machine represents the states and transitions of the upstream and downstream peers 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.

The state machines presented in this document 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 to as "lower-level pre-processing" step. Pre-processing provides to the appropriate MRS state machine only the messages that are matched against waiting Query/Response cookies, or the triplet (MRI, NSLPID, SID) of the established MRS. This is represented by "rx\_\*" events in the state machines.

Management of messaging associations (MAs) is considered in the document via procedures, events, and variables, which describe MA interaction with the MRS state machines. A state machine for MA management is not explicitly presented.

### 5.1. Common Procedures

**Tx\_Query:**  
Transmit of Query message.

**Tx\_Response:**  
Transmit of Response message.

**Tx\_Confirm:**  
Transmit of Confirm message.

**Tx\_Data:**  
Transmit of Data message.

**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 NSLP application.

**Tg\_NetworkNotification:**  
NSLP/GIST API message that informs NSLP application of change in MRS.

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

**Refresh MRS:**

Refreshes installed MRS.

**Queue NSLP info:**

Save NSLP messages in a queue until conditions for their sending are present, e.g., a required MA association is established.

**CheckPeerInfo:**

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

**Delete MA:**

Delete/disconnect used MA.

**Stop using shared MA:**

Stop using shared MA. If the shared MA is no longer being used by any other MRSs, it depends on the local policy whether it is deleted or kept.

**Tg\_Establish\_MA:**

Triggers establishment of a new MA.

**Start/Restart a timer variable (Section 5.3):**

Start/Restart of a certain timer.

**Install/Update/Delete UpstreamPeerInfo variable (Section 5.3):**

Management of upstream peer info in state machine of responding node.

**5.2. Common Events****Rx\_Query:**

Receive of Query message.

**Rx\_Response:**

Receive of Response message.

**Rx\_Confirm:**

Receive of Confirm message.

**Rx\_Data:**

Receive of Data message.



**Tg\_SendMsg:**

NSLP/GIST API message from NSLP application that requests transmission of a NSLP message.

**Tg\_SetStateLifetime(time\_period):**

NSLP/GIST API message providing info for the lifetime of a Routing State (RS), required by the application. "Time\_period = 0" represents the cancellation of established RSs/MAs, invoked by the NSLP application.

**Tg\_InvalidRoutingState:**

NSLP/GIST API notification from NSLP application for path change.

**Tg\_ERROR:**

General Error event / system level error.

**Tg\_MA\_Established:**

A new MA has been successfully established.

**Tg\_MA\_Error:**

Error event with used MA.

**Timeout a timer variable (Section 5.3):**

Timeout of a certain timer.

### 5.3. Common Variables

Variables listed in this section are defined as:

- Specific information carried in the received messages.
- Conditions that are results of processes not defined in the state machine model.

State machine logic is based on these general conditions and message parameters.

The type of mode and destination info is determined by NSLP application parameters and local GIST policy. Here it is represented by the common variables D-mode, C-mode, and MAinfo.

**C-mode:**

The message MUST be transmitted in C-mode. This is specified by "Message transfer attributes" set by NSLP application 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 C-mode (e.g., congestion control) [1].

**D-mode:**

The message **MUST** be transmitted in D-mode. This is specified by local policy rules. If the "Message transfer attributes" are not set by NSLP application to any of the following values, then:

"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 C-mode [1].

**MAinfo:**

GIST message parameters describing the required MA or proposed MA, e.g., "Stack-proposal" and "Stack-Configuration-Data" [1].

**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:**

Indicator that a Confirm message is required by the local policy rule for installation of a new MRS.

**NewPeer:**

Indicator that a Response message is received from a new responding peer.

**MAexist:**

Indicator that an existing MA will be reused in data transfer between peers.

**UpstreamPeerInfo:**

Upstream peer info that is saved in an established MRS.

**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.
- No\_MRS\_Installed:**  
Data sent by responding node via a Response message that indicates loss of Confirm message.

## 6. State Machines

The following section presents the state machine diagrams of GIST peers. RFC 5972 is published as a .txt file. A supplementary .pdf is being published as well.

In the .pdf document, the state machine diagrams are depicted in detail. All state machine information (triggering event, action taken, and variable status) is represented in the diagrams.

In the .txt document, state machine diagrams depict only transition numbers. Following each diagram is a list of state transition descriptions. Complete transition details (triggering event, action taken, and variable status) are given in state transition tables in Appendix A.

Please use the .pdf version whenever possible. It is the clearer representation of the state machine. In case of a difference between the two documents, please refer to the .pdf version.

### 6.1. Diagram Notations

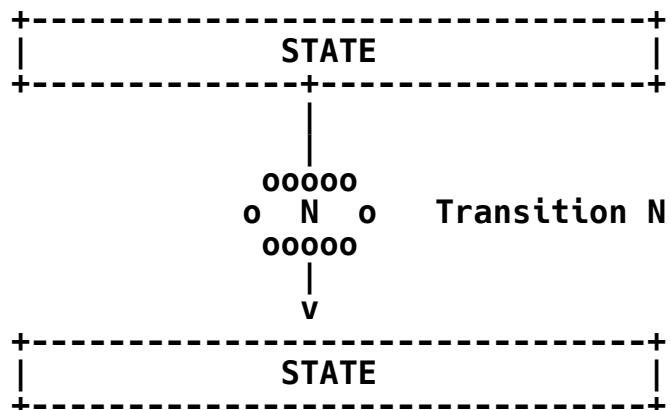


Figure 1: Diagram notations

### 6.2. State Machine for GIST Querying Node

The state machine diagram of the GIST querying node is below. Transition descriptions follow.

Please refer to Appendix A.1 for complete transition details (triggering event, action taken, and variable status).

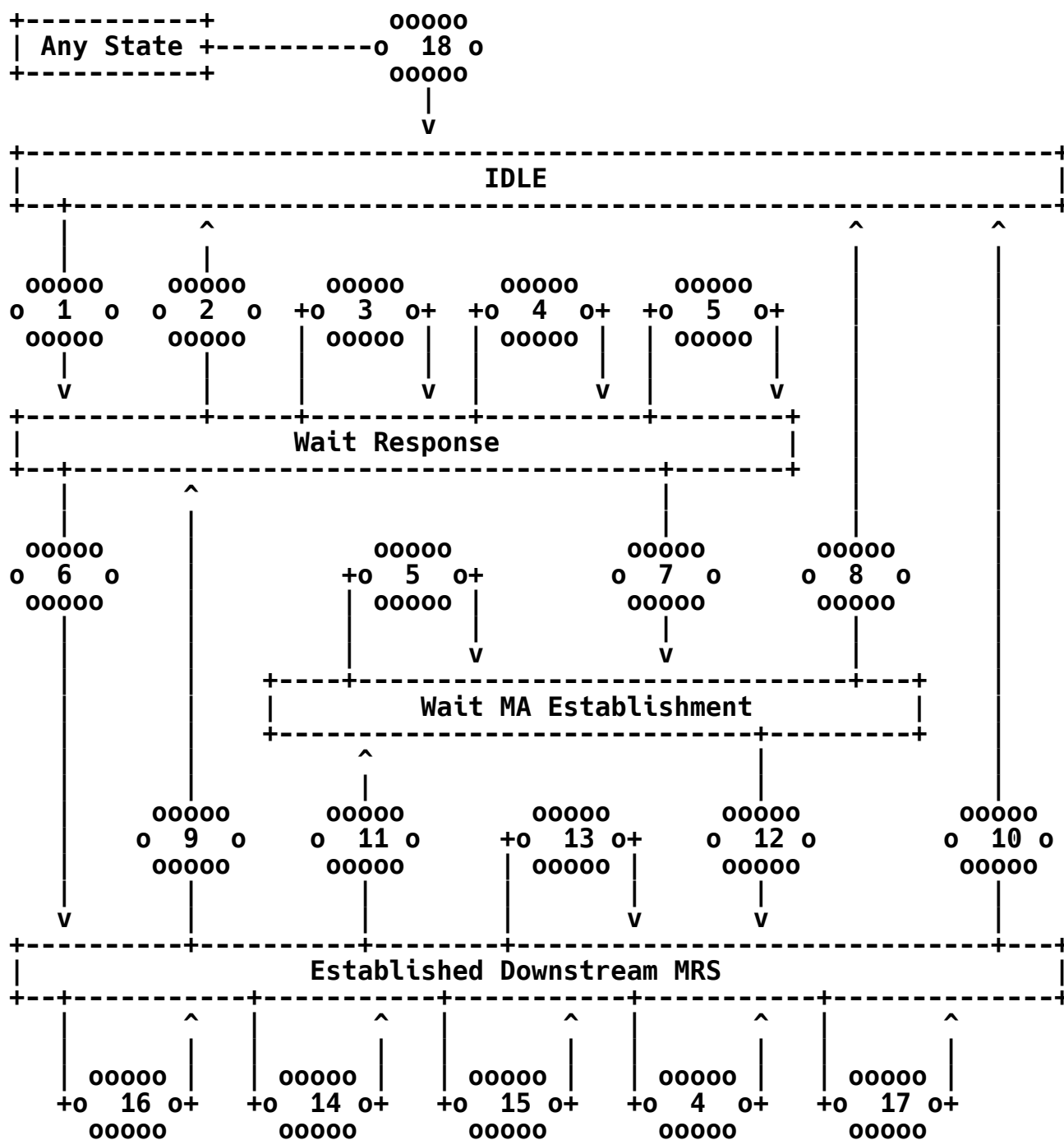


Figure 2: State Machine for GIST Querying Node

- 1\*\*) An initial request from the NSLP application is received, which triggers Query messages requesting either D-mode or C-mode. Depending on the node's local policy, the NSLP data might be piggybacked in the Query requesting D-mode. The Query may carry MAinfo if C-mode transport is needed.
- 2) T\_No\_Response timer expires, and the maximum number of retries has been reached. The NSLP application is notified of the GIST peer discovery failure.
- 3) T\_No\_Response timer expires. The Query is resent.
- 4) A Data message is received. It is checked to see whether its sender matches the installed downstream peer info in the MRS; if so, it is processed. In WaitResponse state, this event might happen in the process of an MA upgrade, when the downstream peer is still not aware of establishment of the new MA.
- 5) The NSLP application provides data for sending. NSLP data is queued because the downstream peer is not discovered or the required MA is still not established.
- 6) A Response message is received. If a D-mode connection is requested or the available MA can be reused for the requested C-mode, the MRS is established.
- 7\*) Response message is received. If a C-mode connection must be established, and there is no available MA to be reused, MA establishment is initiated and the system waits for it to be completed.
- 8) MA establishment fails. NSLP application is notified for unsuccessful message delivery.
- 9) The NSLP application provides data for sending, and the requested transport parameters require an upgrade of the established MRS from D-mode/C-mode to C-mode. Or, the NSLP application notifies the GIST instance of the path change. As a result, downstream GIST peer discovery is initiated.
- 10) The MRS lifetime expires or the NSLP application notifies that the MRS is no longer needed. The MRS is deleted. If not needed, the MA is deleted, too. The NSLP application is notified of the MRS change.
- 11\*) The path change is detected as a Response message from a new downstream GIST peer is received. A new MA must be established for the requested C-mode.

- 12\*) A new MA is established. The MRS is installed. The queued NSLP data is sent.
- 13) T\_Refresh\_QNode timer expires. The Query message is sent.
- 14) The NSLP application provides data for sending. It is sent via Data message towards the downstream GIST peer.
- 15) The Response message from the downstream GIST peer is received. The peer is not changed. The MRS is refreshed (T\_Refresh\_QNode timer is restarted).
- 16) The path change is detected as a Response message from a new downstream GIST peer is received. D-mode is requested, or the existing MA can be reused for the requested C-mode.
- 17) The responding peer indicates that it has not received a Confirm message and it has no established upstream MRS. The Confirm message is resent.
- 18) A general error or system-level error occurs. The MRS is deleted. If not needed, the MA is deleted, too. The NSLP application is notified of the MRS change.

Remarks:

- \*) Response and Confirm messages might be sent either in D-mode or C-mode, before or after MA establishment, depending on the node's local three-way handshake policy and the availability of the MAs to be reused. See [1] for details.
- \*\*) Depending on GIST local policy, NSLPdata might be sent as the payload of Query and Confirm messages (piggybacking).

### 6.3. State Machine for GIST Responding Node

The GIST responding node state machine diagram is below. Transition descriptions follow.

Please refer to Appendix A.2 for complete transition details (triggering event, action taken, and variable status).

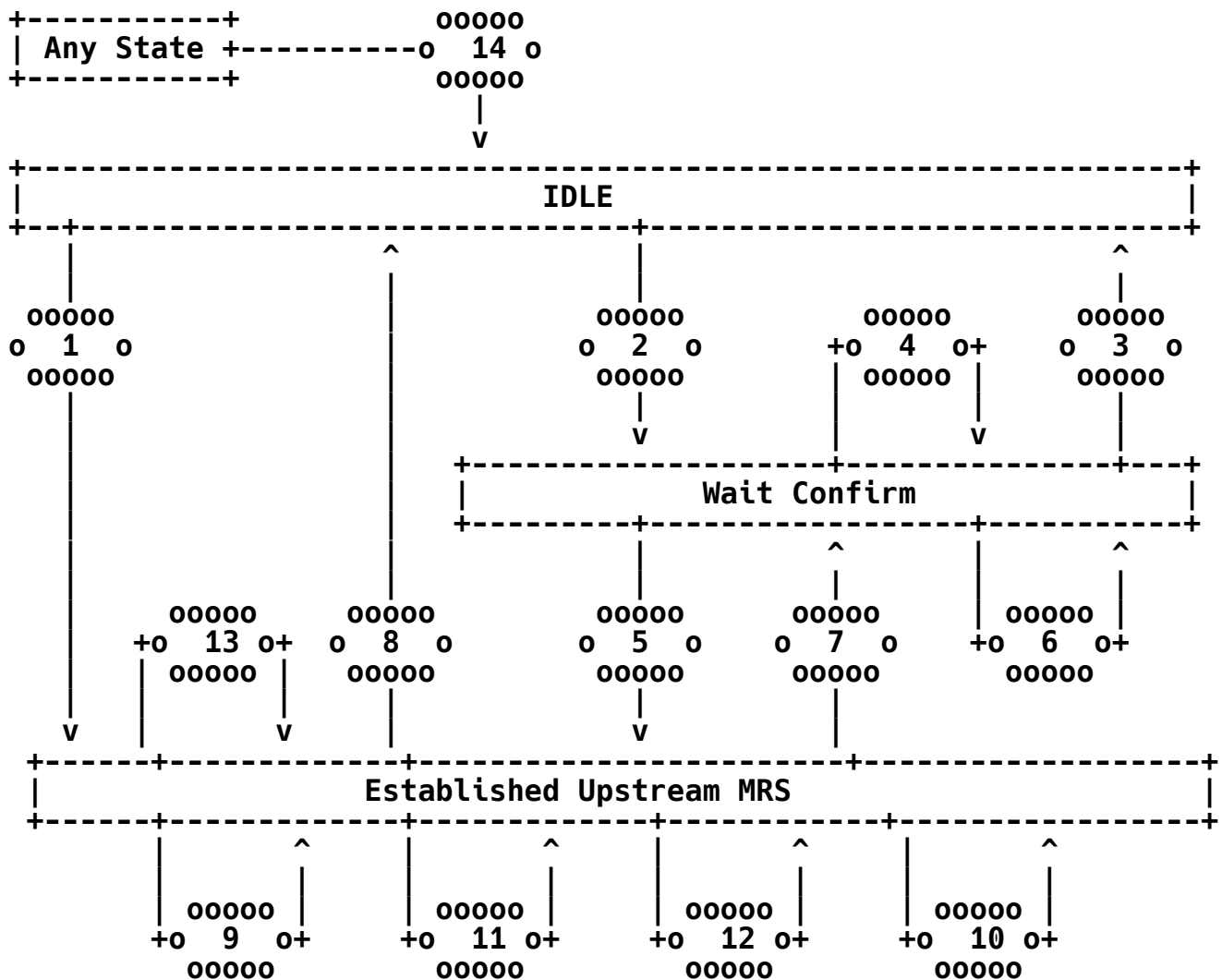


Figure 3: State Machine for GIST Responding Node

- 1) A Query message is received. The MRS is installed immediately because local policy permits it. The Query message might carry piggybacked NSLP data that will be provided to the NSLP application.



- 2) A Query message is received. Local policy requires an explicit Confirm message for MRS installation. The Query message might carry piggybacked NSLP data that will be provided to the NSLP application.
- 3) T\_No\_Confirm timer expires. Note that all cases of lost handshake GIST messages are handled only by the GIST querying node via resend of the Query message.
- 4) A Query message is received again. This means that the sent Response message has not been received by the upstream GIST peer. The Response message is resent.
- 5) A Confirm message is received that causes installation of the upstream MRS.
- 6) In case of a lost Confirm message, data messages might be received from the upstream GIST node (it is unaware of the lost Confirm message). A Response message indicating the loss of the Confirm is sent back to the upstream GIST node.
- 7) A Query message is received (from either an existing upstream GIST node or a new upstream GIST node) with a request to change the used GIST operation mode (from D-mode/C-mode to C-mode, if available; otherwise, it stays the same). Local policy requires an explicit Confirm message for MRS installation.
- 8) The MRS lifetime expires or the NSLP application notifies that the MRS is no longer needed. The MRS is deleted. If used and not needed, the MA is deleted, too. The NSLP application is notified of the MRS change.
- 9) The NSLP application provides data for sending. NSLP data is sent if the discovery process is successfully accomplished, or it is queued if a Confirm message is still expected to confirm establishment of an MA.
- 10) A Query message is received. If it is sent from a new upstream GIST node, then there is a path change. Local policy does not need an explicit Confirm message for MRS installation. The MRS data is updated.
- 11) A Query message is received with a request to change the used GIST operation mode (from D-mode/C-mode to C-mode, if available; otherwise, it stays the same). Local policy does not need an explicit Confirm message for MRS installation. The MRS data is updated.

- 12) A Data message is received. Data messages are accepted only if the complete MRS is installed, e.g., the upstream peer info is installed. If not, then a Confirm message is expected and the Data message is not accepted. A Response message indicating the loss of the Confirm is sent back to the upstream GIST node.
- 13) A Confirm message is received. It accomplishes assignment of an existing MA (or establishment of a new MA) needed for data transfer between peers. The information for the used MA is installed as the upstream peer info.
- 14) A general error or system-level error occurs. The MRS is deleted. If not needed, the MA is deleted, too. The NSLP application is notified of the MRS change.

## 7. Security Considerations

This document does not raise new security considerations. Security considerations are addressed in the GIST specification [1] and in [6].

## 8. Acknowledgments

The authors would like to thank Christian Dickmann who contributed to refining of the state machine.

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.

## 9. References

### 9.1. Normative References

- [1] Schulzrinne, H. and R. Hancock, "GIST: General Internet Signalling Transport", RFC 5971, October 2010.
- [2] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

### 9.2. Informative References

- [3] Vollbrecht, J., Eronen, P., Petroni, N., and Y. Ohba, "State Machines for Extensible Authentication Protocol (EAP) Peer and Authenticator", RFC 4137, 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., Ed., Ohba, Y., and R. Marin-Lopez, "State Machines for the Protocol for Carrying Authentication for Network Access (PANA)", RFC 5609, August 2009.
- [6] Tschofenig, H. and D. Kroeselberg, "Security Threats for Next Steps in Signaling (NSIS)", RFC 4081, June 2005.

## Appendix A. State Transition Tables

The state transition tables below represent the state diagrams in ASCII format. Please use the .pdf version whenever possible. It is the clearer representation of the state machine.

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 Transition Tables for GIST Querying Node

Please refer to the state machine diagram in Figure 2.

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

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

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

+Transition	Condition	Action	State
V			
2)	(timeout T_No_Response) &&(MaxRetry)	tg_MessageStatus	IDLE
3)	(timeout T_No_Response) &&(!MaxRetry)	Tx_Query restart T_No_Response	Wait Response
4)	rx_Data	IF(CheckPeerInfo) tg_RecvMsg to Appl.	Wait Response
5)	tg_SendMsg	Queue NSLP data	Wait Response
6)	rx_Response)   (rx_Response(MAinfo)&& (MAexist))	Install MRS IF (RespCookie) tx_Confirm(RespCookie) tx_Data(Queued NSLP data)	Established Downstream MRS
7) *	rx_Response(MAinfo)&& (!MAexist)	tg_Establish_MA (tx_Confirm)	Wait MA Establish.
18)	Tg_ERROR	(Delete MRS) IF (MA is used) ((Delete MA)   (Stop using shared MA)) Tg_NetworkNotification	IDLE

-----  
 State: Established Downstream MRS  
 -----

+Transition	Condition	Action	State
V-----	-----	-----	-----
4)	rx_Data	IF(CheckPeerInfo) tg_RecvMsg to Appl.	Established Downstream MRS
9)	((tg_SendMsg)&&(C-mode) &&(!MAexist))   (tg_MA_error)   (tg_InvalidRoutingState)	tx_Query Queue NSLP data	Wait Response
10)	(timeout T_Inactive_ QNode)   (tg_SetStateLifetime(0))	Delete MRS IF (MA is used) (Delete MA)   (Stop using shared MA) Tg_NetworkNotification	IDLE
11) *	(rx_Response(MAinfo)&& (NewPeer)&&(!MA_exist))	((Delete MA)   (Stop using shared MA)) tg_Establish_MA (tx_Confirm)	Wait MA Establish.
13)	timeout T_Refresh_QNode	tx_Query	Established Downstream MRS
14)	tg_SendMsg	tx_Data restart T_Inactive_QNode	Established Downstream MRS
15)	(rx_Response)&& (!NewPeer)	Refresh MRS restart T_Inactive_QNode	Established Downstream MRS
16)	(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) tx_Confirm(RespCookie)	Established Downstream MRS

17)	rx_Response(No MRS Installed)	tx_Confirm(RespCookie) tx_Data(Queued NSLP data)	Established Downstream MRS
18)	Tg_ERROR	(Delete MRS) IF (MA is used) ((Delete MA)   (Stop using shared MA)) Tg_NetworkNotification	IDLE

---

-----  
 State: Wait MA Establishment  
 -----

+Transition			
	Condition	Action	State
V	-----		
5)	tg_SendMsg	Queue NSLP data	Wait MA Establish.
8)	tg_MA_error	Delete MRS tg_MessageStatus	IDLE
12) *	tg_MA_Established	Install MRS (tx_Confirm) tx_Data(Queued NSLP data)	Established Downstream MRS
18)	Tg_ERROR	Delete MRS IF (MA is used) ((Delete MA)   (Stop using shared MA)) Tg_NetworkNotification	IDLE

---

## A.2. State Transition Tables for GIST Responding Node

Please refer to the state machine diagram in Figure 3.

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

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

-----  
 State: WAIT CONFIRM  
 -----

+Transition	Condition	Action	State
3)	timeout T_No_Confirm		IDLE
4)	rx_Query&& (ConfirmRequired)	tx_Response start T_No_Confirm IF(NSLPdata) tg_RecvMsg(NSLPdata) to Appl.	Wait Confirm
5)	rx_Confirm	Install Upstream MRS	Established Upstream MRS
6)	rx_Data	tx_Response(No_MRS installed)	Wait Confirm



14)	(Tg_ERROR)   (Tg_MA_Error)	(Delete MRS) IF (MA is used) ((Delete MA)   (Stop using shared MA)) Tg_NetworkNotification	IDLE
-----			

-----  
State: Established Upstream MRS  
-----

+Transition	Condition	Action	State
7)	(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
8)	(timeout T_Expire_RNode)    (tg_SetStateLifetime(0))	Delete MRS tg_NetworkNotification IF(MA is used) (Delete MA)   (Stop using shared MA)	IDLE
9)	tg_SendMsg	IF(!UpstreamPeerInfo) Queue NSLP data ELSE tx_Data	Established Upstream MRS
10)	rx_Query	IF (NewPeer) Update UpstreamPeerInfo tx_Response restart T_Expire_RNode	Established Upstream MRS
11)	rx_Query(MAinfo)&& (!ConfirmRequired)	Delete UpstreamPeerInfo restart T_Expire_RNode tx_Response(MAinfo)	Established Upstream MRS

12)	rx_Data	IF(UpstreamPeerInfo) (tg_RecvMsg to Appl.) &&(restart_T_Expire_ RNode) ELSE tx_Error(No MRS_ installed)	Established Upstream MRS
13)	rx_Confirm	Install UpstreamPeerInfo tx_Data(queued_NSLP_data)	Established Upstream MRS
14)	(Tg_ERROR)   (Tg_MA_Error)	(Delete MRS) IF (MA is used) ((Delete MA)   (Stop using shared MA)) Tg_NetworkNotification	IDLE
-----+-----+-----+-----			

**Authors' Addresses**

Tseno Tsenov  
Sofia, Bulgaria

EMail: [tseno.tsenov@mytum.de](mailto:tseno.tsenov@mytum.de)

Hannes Tschofenig  
Nokia Siemens Networks  
Linnoitustie 6  
Espoo 02600  
Finland

EMail: [Hannes.Tschofenig@nsn.com](mailto:Hannes.Tschofenig@nsn.com)

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

EMail: [fu@cs.uni-goettingen.de](mailto:fu@cs.uni-goettingen.de)

Cedric Aoun  
Consultant  
Paris, France

EMail: [cedaoun@yahoo.fr](mailto:cedaoun@yahoo.fr)

Elwyn B. Davies  
Folly Consulting  
Soham, Cambs, UK

Phone: +44 7889 488 335  
E-Mail: [elwynd@dial.pipex.com](mailto:elwynd@dial.pipex.com)