NSIS
Internet-Draft
Expires: September, 2006

X. Fu
B. Schloer
Univ. Goettingen
H. Tschofenig
Siemens
T. Tsenov
March, 2006

QoS NSLP State Machine
draft-fu-nsis-qos-nslp-statemachine-03.txt

Status of this Memo

Copyright Notice

Abstract

This document describes the state machines for the NSIS Signaling Layer Protocol for Quality-of-Service signaling (QoS NSLP). A set of state machines for QoS NSLP entities at different locations of a flow path are presented in order to illustrate how QoS NSLP may be implemented.

Table of Contents

1. Introduction

This document describes the state machines for QoS NSLP [1], trying to show how QoS NSLP can be implemented to support its deployment. The state machines described in this document are illustrative of how the QoS NSLP protocol defined in [1] may be implemented for the QNI QoS NSLP node, QNE QoS NSLP nodes, and QNR QoS NSLP node in the flow path. Where there are differences [1] are authoritative. The state machines are informative only. Implementations may achieve the same results using different methods.

According to [1], there are several possibilities for QoS NSLP signaling, at least including the following: - end-to-end signaling vs. scoped signaling - sender-initiated signaling vs. receiver-initiated signaling (which need to be incorporated into use scenarios when describing state machine. Note they are represented by way of certain objects/flags in Reserve and Query messages.)

The messages used in the QoS NSLP protocol can be summarized as follows:

Requesting message	Responding message
RESERVE	None or RESERVE or RESPONSE
QUERY	RESERVE or RESPONSE
RESPONSE	NONE
NOTIFY	NONE

We describe a set of state machines for different roles of entities running QoS NSLP to illustrate how QoS NSLP 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 flow sender, flow receiver, QUERY, RESERVE or RESPONSE.

5.1 Common Procedures

tx_RESERVE(<object>):
Transmit RESERVE message with <object>

tx_RESPONSE():
Transmit RESPONSE message

tx_QUERY(<object>):
Transmit QUERY message with <object>

tx_QUERY(w/o<object>):

```
      Transmit QUERY message without <object>

   tx_NOTIFY():
      Transmit NOTIFY message

   Install QoS state():
      Install the local QoS state.

   Refresh QoS state():
      Refresh the local QoS state.

   Delete QoS state():
      Delete the local QoS state.

   Send info to Application():
      Report information to the application.

   rx_RESPONSE():
      Receive RESPONSE message

   rx_QUERY():
      Receive QUERY message

   rx_RESERVE():
      Receive RESERVE message

   rx_NOTIFY():
      Receive NOTIFY message

   TIMEOUT_StateLifetime:
      State lifetime timer expiration

   TIMEOUT_Refresh:
      Refresh interval timer expiration

   TIMEOUT_Response:
      Wait-Response interval timer expiration

   tg_QUERY:
      External trigger to send a QUERY message (typically triggered by
      the application).
```

tg_RESERVE:
   External trigger to send a RESERVE message.

tg_TEARDOWN:
   External trigger to clear previously established QoS state
   (typically triggered by the application). It is translated to a
   tx_RESERVE(Ton) message.

RMF:
   Performs Resource Management Function and returns the following
   values{AVAIL, NO_AVAIL}.

SetRII:
   Sets the RII object of the messages e.g. the node requests
   explicit response to the message being sent. Returns values
   {0,1}.

CheckRII:
   Checks the RII object of received RESPONSE message if it is
   requested by current node or other upstream node. Returns values
   {LOCAL, NO_LOCAL}.

ProcessQUERY:
   Processes a Query message and provides the requested info

5.2 Common Variables

RII:
   Request Identification Information (RII) object. Logical variable
   representing if the RII is set or not. Takes values {0,1}.

SCOPING:
   Scoping flag of common message header. Takes values
   {"Next_hop","Whole_path"}.

RSN:
   Reservation Sequence Number object. Takes values:
   - recRSN - RSN object of the received message
   - currRSN - Current stored RSN value for installed QoS state.
   (Assumed to be the one for the direction where the message comes
   from e.g.Upstream/Downstream)

ACK:
   Acknowledgement flag of common message header. Takes values
   {"On","Off"}.

ReducedRefresh:
Keeps information if Reduced refresh method may be used for refreshing a installed QoS state. Takes value {"On","Off"}.

E_SPEC:
Error_Spec object. Takes values:
- 0x02? - Success values
- 0x04? - Transient Failure values

QSPEC:
QoS specification object.

FlowID:
Flow ID kept by the installed QoS state.

Replace:
Replace flag of common message header. Takes values {"On","Off"}.

SII:
Source Identification Information entry. Takes values:
- CurrSII - SII entry stored for current installed QoS state. (Assumed to be the one for the direction where the message comes from e.g.Upstream/Downstream)
- newSII - SII of the received message is different from the SII stored for the current installed QoS state.

5.3 Constants

5.4 Assumptions

- For simplification not all included objects in a message are showed. Only those that are significant for the case are showed. State machines do not present handling of messages that are not significant for management of the states such as certain NOTIFY and QUERY messages.
- State machines represent handling of messages of the same Session ID and with no protocol errors. Separate parallel instances of the state machines should handle messages for different Session IDs.
- Default message handling should be defined for messages with different Session IDs that have impact on current session state and error messages. This is not included in the current version.
- ACK flag in the common header is set "On" by default.

- Direction of receiving and sending messages is not specified. We assume it is implicit from the context.

6. State machines

The following section presents the state machine diagrams of QoS NSLP 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 QNI QoS NSLP node

The following are diagrams of the QNI QoS NSLP node state machine.

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

Figure 2: QNI node: "IDLE" state

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

Figure 3: QNI node: "WAITRESP1", "WAITRESP2" and "QoS state installed" state

6.3 State machine for QNE QoS NSLP node

The following are diagrams of the QNE QoS NSLP node state machine.

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

Figure 4: QNE node: "IDLE" state

Notes:
1) Successful reservation without Response request (1a) and with Scoping (1b).
2) Processing of Query msg for Receiver initiated reservation
3) Unsuccessful reservation with/without request for response from

previous node in the path.
5) Processing of Query msg triggered by the application layer.
7) Processing of Query msg received from an upstream node.

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

Figure 5: QNE node: "QoS state installed" state

Notes:
4) Unsuccessful reservation. RII requested at the local node.
NOTIFY(RSN) is sent further to the upstream nodes.

6) QoS State refresh procedures
8) We assume that handling of QoS state lifetime expiration event is based on the local policy of the node. NOTIFY/Reserve(Ton) messages might be sent to other peers.
9) Update QoS state and Re-route functionality

(see the .pdf version for missing diagram or

refer to Appendix A.2 if reading the .txt version)

Figure 6: QNE node: "QoS state installed & WaitRESP1" and "WaitRESP2" states

6.4 State machine for QNR QoS NSLP node

The following are diagrams of the QNR QoS NSLP node state machine.

(see the .pdf version for missing diagram or
refer to Appendix A.3 if reading the .txt version)

Figure 7: QNR node

Notes:
1) Initiation of Receiver-side reservation
2) Successful Reservation with& without response request from the QNI side
3) Unsuccessful Reservation with & without response request from the QNI side.
5) We assume that handling of QoS state lifetime expiration event is based on the local policy of the node. NOTIFY/Reserve(Ton) messages might be sent to other peers.
6) Successful Reservation update with& without response request from the QNI side.

7. Security Considerations

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

For the time being, the state machines described in this document do not consider the security aspect of QoS NSLP protocol itself. A future versions of this document will add security relevant states and state transitions.

8. Open Issues

This document tries to describe possible states and transitions for QoS NSLP according to its current specification [1], Section 5. We found some issues during the development of the state machines.

1. For receiver-initiated reservation, it is unclear who triggers a teardown.
2. Bi-directional reservation is difficult to support as the state machine becomes quite complex (note at one particular point in time the protocol state engine can be only in one state).
3. How to signal unsuccessful reservation for Receiver initiated reservation (No RII included; a resulting Response(RSN) cannot be forwarded further than the next peer). We use NOTIFY message.
4. If QoS state lifetime expires in QNI, should RESERVE(Ton) be sent downstream the path?
5. The case of unsuccessful reservation at a QNE node and no RII specified by upstream nodes. According to the spec RESPONSE(RSN) should not be forwarded further than the next peer. Currently we use NOTIFY(RSN) that is sent further to the upstream nodes.
6. We assume that handling of QoS state lifetime expiration event is based on the local policy of the node. NOTIFY/Reserve(Ton) messages might be sent to other peers.
7. The draft states that RESERVE message MUST be sent only towards the QNR. This is not the case when re-routing procedure is done and RESERVE(Ton) message should be sent from merging QNE node for deleting the old branch. We believe this is towards the QNI.

8. Re-routing functionality described in this document is not complete and need further consideration.

9. Change History

9.1 Changes in Version -01

1. Notation of the nodes changed to QNI, QNE and QNR.

2. Description of soft state refresh functionality.
3. Support of ACK flag in the common header.
4. Include of QoS NSLP objects, flags from the common header and entries stored with the installed QoS state in a node: ACK, Replace, RSN, Error_SPEC, QSPEQ, FlowID, SII.
5. Initial description of Re-routing functionality.
6. For support of all listed changes, some notations are changed.

9.2 Changes in Version -02

1. Switch to .pdf format of the draft and include graphic diagrams.
2. Update notation from "Summary refresh" to "Reduced refresh"
3. Description of QoS reservation update/upgrade

10. Acknowledgments

The authors would like to thank Sven Van den Bosch for his feedback.

11. References

11.1. Normative References

[1] Manner, J., Karagiannis, G., McDonald, A. and S. Van den Bosch "NSLP for Quality-of-Service signaling", Internet draft, draft-ietf-nsis-qos-nslp-07, July 2005.

[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", draft-ietf-eap-statemachine-06 (work in progress), December 2004.

[4] Institute of Electrical and Electronics Engineers, "DRAFT Standard for Local and Metropolitan Area Networks: Port-Based Network Access Control (Revision)", IEEE 802-1X-REV/D11, July 2004.

[5] Ohba, Y., "State Machines for Protocol for Carrying

Authentication for Network Access (PANA)",
draft-ohba-pana-statemachine-01 (work in progress),
February 2005.

[6] Tschofenig, H. and D. Kroeselberg, "Security Threats for NSIS", draft-ietf-nsis-threats-06 (work in progress), October 2004.

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.

The notation is as follows: 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 QNI QoS NSLP node (Figures 2,3)

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

Condition                 Action                      State      Note
------------------------+--------------------------+-----------+---
(tg_RESERVE) &&          |Send info to Application |IDLE       |
```

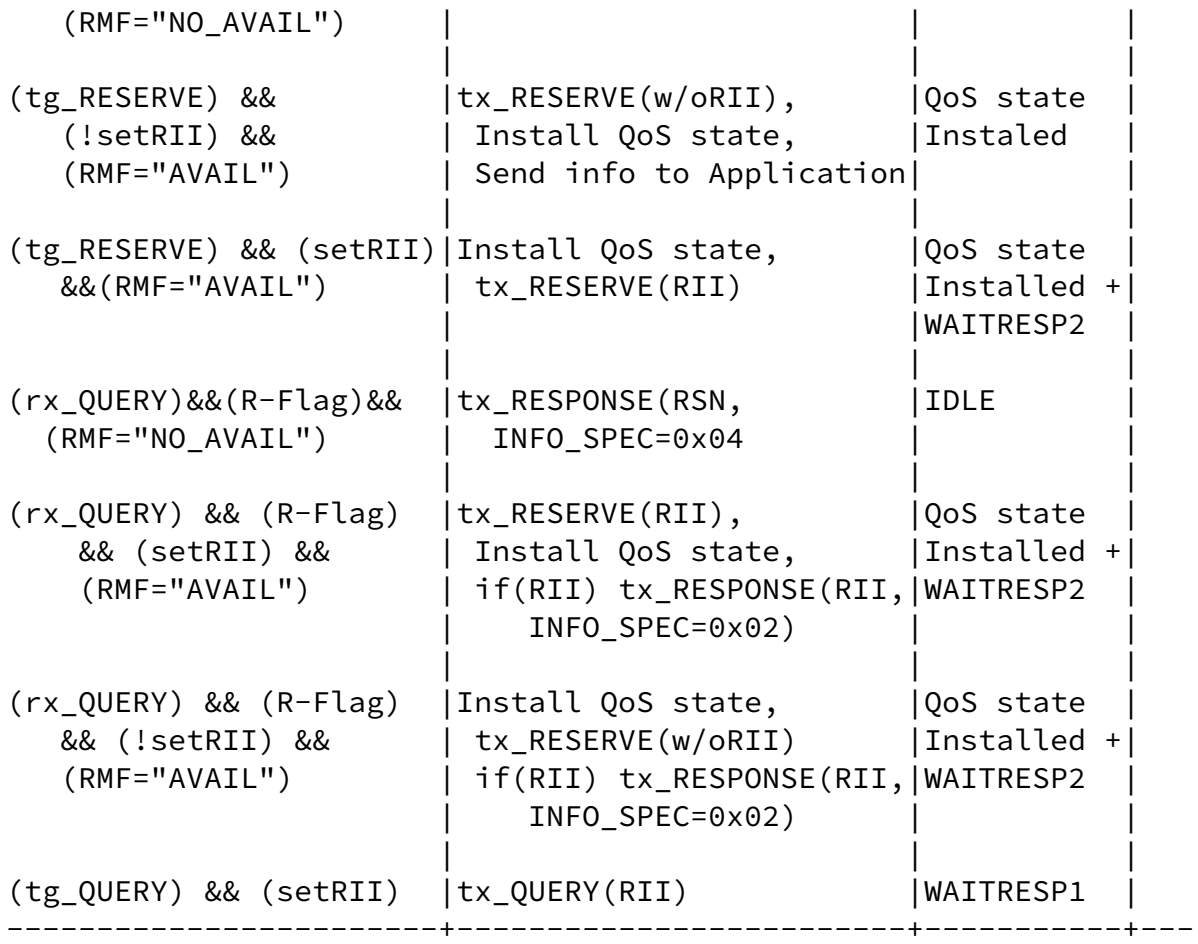

Figure 8

State: WAITRESP1

Condition	Action	State
(TIMEOUT_WaitResp) && (!MaxRetry)	tx_QUERY(RII)	WAITRESP1
(TIMEOUT_WaitResp) && (MaxRetry)	Send info to Application	IDLE

rx_RESPONSE	Send info to Application	IDLE
rx_QUERY(RII)	ProcessQUERY tx_RESPONSE(RII)	WAITRESP1

State: QoS state installed + WAITRESP2

Condition	Action	State
(TIMEOUT_WaitResp) && (!MaxRetry)	tx_RESERVE(RII)	QoS state installed + WAITRESP2
(TIMEOUT_WaitResp) && (MaxRetry)	Delete QoS state Send info to Application	IDLE
rx_RESPONSE(RII, INFO_SPEC="0x04?")	Delete QoS state Send info to Application	IDLE
rx_RESPONSE(RII, INFO_SPEC="0x02?")	Send info to Application iReducedRefresh="On"	QoS state installed
rx_QUERY(RII)	ProcessQUERY tx_RESPONSE(RII)	QoS state installed + WAITRESP2

State: QoS state installed

Condition	Action	State	Note
rx_QUERY(RII)	ProcessQUERY tx_RESPONSE(RII)	QoS state installed	

TIMEOUT_Refresh	If (ReducedRefresh="On")
	(Tx_RESERVE(RSN)) &&
	(ReducedRefresh="Off")
	Else
	Tx_RESERVE(RSN,QSPEC);
		QoS state installed	

rx_RESPONSE(RSN, INFO_SPEC="0x02?")	ReducedRefresh="On"	QoS state installed	
TIMEOUT_StateLifetime	Delete QoS state Send info to Application	IDLE	
tg_TEARDOWN	Delete QoS state, tx_RESERVE(Ton)	IDLE	
rx_NOTIFY(RSN, INFO_SPEC="0x04?")	Delete QoS state Send info to Application	IDLE	
(tg_RESERVE) && (!setRII) && (RMF="AVAIL")	tx_RESERVE(w/oRII), Update QoS state, Send info to Application	QoS state Instaled	
(tg_RESERVE) && (setRII) &&(RMF="AVAIL")	Update QoS state, tx_RESERVE(RII)	QoS state Installed + WAITRESP2	

Figure 9

A.2. State machine for QNE QoS NSLP node (Figures 4,5,6)

State: IDLE

Condition	Action	State	Note
(rx_QUERY) && (!RII)	tx_QUERY(w/oRII)	IDLE	2)
(rx_QUERY(RII, SCOPING="Next_hop")	ProcessQUERY, Tx_RESPONSE(RII)	IDLE	7)
(rx_QUERY) && (RII)	tx_QUERY(w/RII)	IDLE	7)
(rx_RESERVE(RII)) && (RMF="NO_AVAIL")	Tx_RESPONSE(RII, INFO_SPEC="0x04?")	IDLE	3)
(rx_RESERVE) && (!RII)&& (RMF="NO_AVAIL")	Tx_RESPONSE(RSN, INFO_SPEC="0x04?")	IDLE	3)
(rx_RESPONSE(RII)) && (CheckRII="Not_LOCAL")	Tx_RESPONSE(RII)	IDLE	

(rx_RESERVE)&& !(setRII) && (RMF="AVAIL")	Install QoS state,
	If(ACK="On")
	Tx_RESPONSE(RSN,
	INFO_SPEC="0x02?")
	If(RII) Tx_RESPONSE(RII)
	Else Tx_RESPONSE(w/oRII)
		QoS State Installed	1a)

(rx_RESERVE(SCOPING= "Next_hop")) && (RMF="AVAIL")	Install QoS state, If(RII) Tx_RESPONSE(RII, INFO_SPEC="0x02?") Else Tx_RESPONSE(RSN, INFO_SPEC="0x02?")	QoS State Installed	1b)
(rx_RESERVE) && (setRII) && (RMF="AVAIL")	Install QoS state, Tx_RESiERVE(RII), If(ACK="On") Tx_RESPONSE(RSN, INFO_SPEC="0x02?")	QoS State Installed + WAITRESP1	4)
(tg_QUERY) && (setRII)	tx_QUERY(RII)	WAITRESP2	5)

Figure 10

State: QoS State Installed

Condition	Action	State	Note
rx_RESERVE(Ton)	tx_RESERVE(Ton), Delete QoS state	IDLE	
rx_RESERVE	Refresh QoS state If(ACK="On") Tx_RESPONSE(RSN, INFO_SPEC="0x02?") tx_reserve()	QoS State Installed	6)
rx_RESPONSE(RSN, INFO_SPEC="0x02?")	ReducedRefresh="On"	QoS State Installed	6)

TIMEOUT_Refresh	If (ReducedRefresh="On")
	(Tx_RESERVE(RSN))
	&&(ReducedRefresh="Off")
	Else
	Tx_RESERVE(RSN,QSPEC)
		QoS State Installed	6)

(rx_RESPONSE(RII, INFO_SPEC="0x02?") &&(ChechRII="NOT_LOCAL")	ReducedRefresh="On" rx_RESPONSE(RII, INFO_SPEC="0x02?")	QoS State Installed	
(TIMEOUT_StateLifetime)	Delete QoS state	IDLE	8)
(rx_RESPONSE(RII, INFO_SPEC="0x04?") &&(ChechRII="NOT_LOCAL")	Delete QoS state tx_RESPONSE(RII, INFO_SPEC="0x04?")	QoS State	
rx_RESPONSE(RSN, INFO_SPEC="0x04?")	Delete QoS state rx_RESPONSE(RSN, INFO_SPEC="0x04?")	IDLE Installed	
rx_NOTIFY(RSN, INFO_SPEC="0x04?")	Delete QoS state rx_RESPONSE(RSN, INFO_SPEC="0x04?")	IDLE	
(Rx_RESERVE)&&(currSII) &&(Replace="On")	Update QoS state If (RII)	QoS State Installed	9)

&&(RMF="AVAIL")	Tx_RESERVE(RII,QSPEC)		
&&((recRSN>=currRSN)	else		
||(newFlowID))	Tx_RESERVE(RSN,QSPEC);		
	If (ACK="On")&&(!RII)		
	tx_RESPONSE(RSN,		
	INFO_SPEC="0x02?")		

(Rx_RESERVE)&&(newSII)	Update QoS state	QoS State	9)
&&(RMF="AVAIL")	If (RII)	Installed	
&&((recRSN>=currRSN)	Tx_RESERVE(RII,QSPEC)		
||(newFlowID))	else		
	Tx_RESERVE(RSN,QSPEC);		
	If (ACK="On")&&(!RII)		
	tx_RESPONSE(RSN,		
	INFO_SPEC="0x02?")		
	If (Replace="On")		
	tx_Reserve(Ton)		
	to currSII		
(rx_RESERVE)&& !(setRII)	Update QoS state,	QoS State	
&& (RMF="AVAIL")	If(ACK="On")	Installed	
	Tx_RESPONSE(RSN,		
	INFO_SPEC="0x02?")		
	If(RII) Tx_RESPONSE(RII)		
	Else Tx_RESPONSE(w/oRII)		
(rx_RESERVE(SCOPING=	Update QoS state,	QoS State	
"Next_hop")) &&	If(RII) Tx_RESPONSE(RII,	Installed	
(RMF="AVAIL")	INFO_SPEC="0x02?")		
	Else Tx_RESPONSE(RSN,		
	INFO_SPEC="0x02?")		
(rx_RESERVE) && (setRII)	Update QoS state,	QoS State	
&& (RMF="AVAIL")	Tx_RESPONSE(RII),	Installed +	
	If(ACK="On")	WAITRESP1	
	Tx_RESPONSE(RSN,		
	INFO_SPEC="0x02?")		
(rx_QUERY) && (!RII)	tx_QUERY(w/oRII)	QoS State	
		Installed	

```
                          |                              |            |
(rx_QUERY(RII,            |ProcessQUERY,                 |QoS State   |
    SCOPING="Next_hop")   |Tx_RESPONSE(RII)              |Installed   |
                          |                              |            |
(rx_QUERY) && (RII)       |tx_QUERY(w/RII)               |QoS State   |
                          |                              |Installed   |
--------------------------+------------------------------+------------+---
```

Figure 11


```
----------------
State: QoS State Installed + WAITRESP1
----------------
```

```
Condition                  Action                       State       Note
-------------------------+----------------------------+-----------+---
(TIMEOUT_WaitResp) &&    |tx_RESERVE(RII)             |WAITRESP1  |
   (!MaxRetry)           |                            |           |
                         |                            |           |
(TIMEOUT_WaitResp) &&    |Delete QoS State,           |           |
   (MaxRetry) &&         |tx_RESPONSE(RSN,            |IDLE       |
                         |  INFO_SPEC="0x04?")        |           |
                         |Send info to Application    |           |
                         |                            |           |
(rx_RESPONSE(RII,        |Delete QoS State,           |IDLE       |
      E_SPEC="0x04?"))   |tx_RESPONSE(RSN,            |           |
 &&(CheckRII="LOCAL")    |  INFO_SPEC="0x04?")        |           |
                         |Send info to Application    |           |
                         |                            |           |
(rx_RESPONSE(RII,        |Send info to Application    |QoS State  |
      E_SPEC="0x02?"))   |SummaryRefresh="On"         |Installed  |
 &&(CheckRII="LOCAL")    |                            |           |
                         |                            |           |
(rx_RESPONSE(RII)) &&    |Tx_RESPONSE(RII)            |QoS State  |
```

```
  (CheckRII="Not_LOCAL")|                          |Installed +|
                        |                          |WAITRESP1  |
                        |                          |           |
(rx_QUERY) && (!RII)    |tx_QUERY(w/oRII)          |QoS State  |
                        |                          |Installed +|
                        |                          |WAITRESP1  |
                        |                          |           |
(rx_QUERY(RII,          |ProcessQUERY,             |QoS State  |
    SCOPING="Next_hop") |Tx_RESPONSE(RII)          |Installed +|
                        |                          |WAITRESP1  |
                        |                          |           |
(rx_QUERY) && (RII)     |tx_QUERY(RII)             |QoS State  |
                        |                          |Installed +|
                        |                          |WAITRESP1  |
------------------------+--------------------------+-----------+---
```



```
----------------
State: WAITRESP2
----------------

Condition                Action                     State      Note
-------------------------+--------------------------+-----------+---
(TIMEOUT_WaitResp) &&    |tx_QUERY(RII)             |           |
   (!MaxRetry)           |                          |           |
                         |                          |           |
(TIMEOUT_WaitResp) &&    |Send info to Application  |IDLE       |
   (MaxRetry)            |                          |           |
                         |                          |           |
(rx_RESPONSE) &&         |Send info to Application  |IDLE       |
   (CheckRII="LOCAL")    |                          |           |
                         |                          |           |
(rx_RESPONSE(RII)) &&    |Tx_RESPONSE(RII)          |WAITRESP2  |
- (CheckRII="Not_LOCAL")|                          |           |
                         |                          |           |
(rx_QUERY) && (!RII)     |tx_QUERY(w/oRII)          |WAITRESP2  |
```

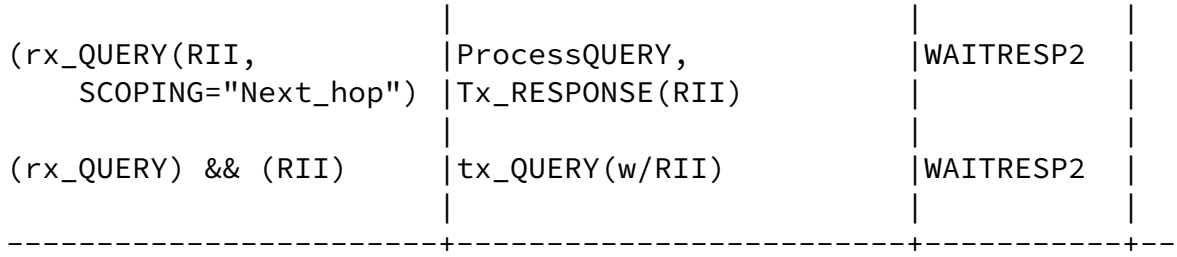

Figure 12

A.3. State machine for QNR QoS NSLP node (Figure 7)

State: IDLE

Condition	Action	State	Note
rx_QUERY(RII)	tx_RESPONSE(RII)	IDLE	
(rx_RESERVE)&&(!RII) && (RMF="NO_A")	Tx_RESPONSE(RSN, INFO_SPEC="0x04?")	IDLE	
(rx_RESERVE(RII)) && (RMF="NO_A")	Tx_RESPONSE(RII, INFO_SPEC="0x04?")	IDLE	
(tg_QUERY) && (R-Flag) && (RMF="AVAIL")	tx_QUERY(R-Flag)	WAITRESV	1)
(rx_RESERVE(RII)) && (RMF="AVAIL")	Install QoS state Tx_RESPONSE(RII, INFO_SPEC="0x02?")	QoS state installed	2)
(rx_RESERVE)&&(!RII)	Install QoS state	QoS state	2)

```
 && (RMF="AVAIL")       |Tx_RESPONSE(RSN,          |installed  |
                           INFO_SPEC="0x02?")     |           |
------------------------+------------------------+-----------+---
```

```
---------------
State: WAITRESV
---------------

Condition                Action                    State      Note
------------------------+------------------------+-----------+---
TIMEOUT_WaitResp        |Tx_QUERY(R-Flag)         |WAITRESV   |
                        |                         |           |
(TIMEOUT_WaitResp)      |Send info to Appl.       |IDLE       |
 && (MaxRetry)          |                         |           |
                        |                         |           |
rx_RESPONSE(RSN,        |Send info to Appl.       |IDLE       |
 INFO_SPEC="0x04?")     |                         |           |
                        |                         |           |
(rx_RESERVE)&&(!RII)    |Install QoS state        |QoS state  |
                        |Tx_RESPONSE(w/oRII)      |installed  |
                        |                         |           |
(rx_RESERVE)&&(RII))    |Install QoS State        |QoS state  |
                        |Tx_RESPONSE(RSN,         |installed  |
                        |   INFO_SPEC="0x02?")    |           |
                        |                         |           |
rx_QUERY(RII)           |tx_RESPONSE(RII)         |WAITRESV   |
------------------------+------------------------+-----------+---
```



```
------------------
State: QoS state installed
------------------

Condition                Action                    State      Note
```

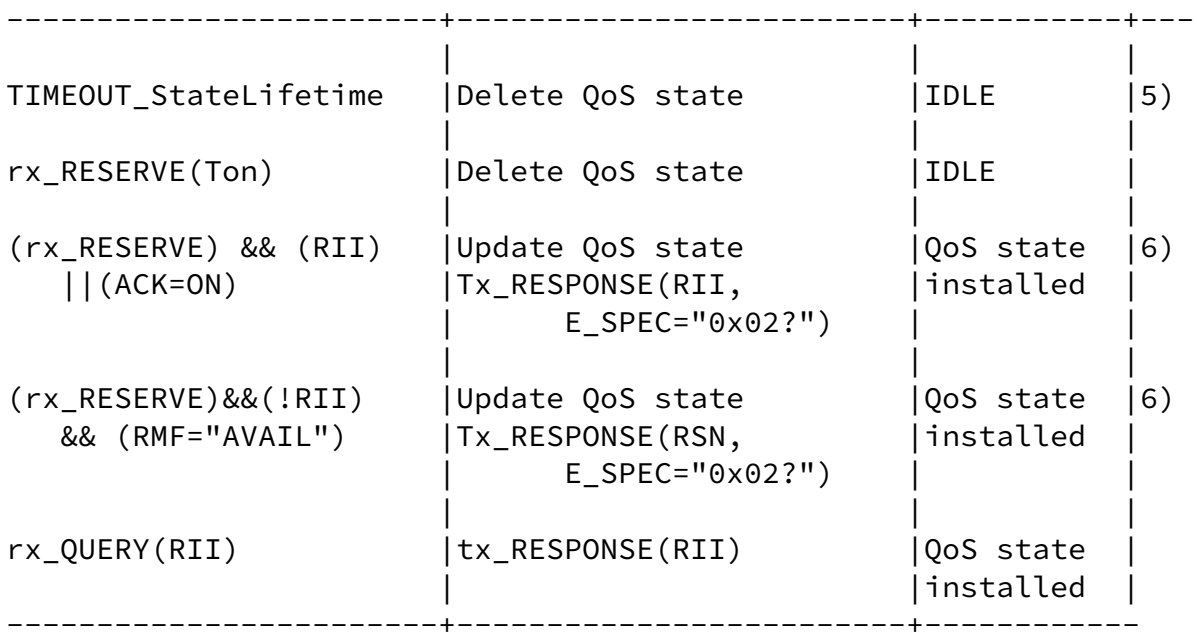
TIMEOUT_StateLifetime
Delete QoS state
IDLE
5)
rx_RESERVE(Ton)
Delete QoS state
IDLE
(rx_RESERVE) && (RII)
||(ACK=ON)
Update QoS state
Tx_RESPONSE(RII,
E_SPEC="0x02?")
QoS state
installed
6)
(rx_RESERVE)&&(!RII)
&& (RMF="AVAIL")
Update QoS state
Tx_RESPONSE(RSN,
E_SPEC="0x02?")
QoS state
installed
6)
rx_QUERY(RII)
tx_RESPONSE(RII)
QoS state
installed

Figure 13

Authors' Addresses

Xiaoming Fu
University of Goettingen
Telematics Group
Lotzestr. 16-18
Goettingen 37083
Germany

Email: fu@cs.uni-goettingen.de

Bernd Schloer
University of Goettingen
Telematics Group
Lotzestr. 16-18
Goettingen 37083
Germany

Email: bschloer@cs.uni-goettingen.de

Hannes Tschofenig
Siemens
Otto-Hahn-Ring 6
Munich, Bayern 81739
Germany

Email: Hannes.Tschofenig@siemens.com

Tseno Tsenov

Email: tseno.tsenov@mytum.de

Intellectual Property Statement

Disclaimer of Validity

Copyright Statement

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.