NSIS

Internet-Draft
Expiration Date: December 2006

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

June 25, 2006

QoS NSLP State Machine
draft-fu-nsis-qos-nslp-statemachine-04.txt

Status of this Memo

Copyright Notice

Abstract

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

Table of Contents

1. Introduction

This document describes a state machine for QoS NSLP [1], trying to show how QoS NSLP can be implemented to support its deployment. The state machine described in this document is illustrative of how the QoS NSLP protocol defined in [1] may be implemented for QoS NSLP nodes in the flow path. Where there are differences [1] are authoritative. The state machine diagrams 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.

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
------------------------+--------------------------
```

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 defi ned in the [1], such as flow sender, flow receiver, QUERY, RESERVE or RESPONSE.

5.1 Common Procedures

tx_reserve():
Transmit RESERVE message

tx_response():
Transmit RESPONSE message

tx_query():
Transmit QUERY message

tx_notify():
Transmit NOTIFY message

install_qos_state():
Install the local QoS state.

delete_qos_state():
Delete the local QoS state.

send_info_to_app():
Report information to the application.

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

is_local(RII):
Checks the RII object of received RESPONSE message if it is requested by current node or other upstream node. Returns values {true, false}.

is_local(RSN):
Checks The RSN object of the received RESPONSE message if it is requested by current node. Returns values {true, false}.

process_query():
Processes a Query message and provides the requested info

5.2 Common Variables

RII:
Request Identifi cation Information (RII) object.

RSN:
Reservation Sequence Number (RSN) object.

INFO:
Info_Spec object. Takes values:
- 0x02 - Success values
- 0x04 - Transient Failure values

QSPEC:
QoS specifi cation object.

T-Flag:
Tear-Flag. Indicates to tear down reservation state.

A-Flag:
Acknowledgement-Flag of common message header. Takes values {true, false}.

R-Flag:
Reserve-Init. Indicates a Receiver Initiated Reservation request in a QUERY message.

S-Flag:
Scoping flag of common message header. Takes values {true="Next_hop", false="Whole_path"}.

setRII:
If set a RII object will be included into the message. Takes values {true, false}.

setACK:
If set a RSN object will be included into the message. Takes values {true, false}.

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

FlowID:
Flow ID kept by the installed QoS state.

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

nodepos:
Position of the QoS NSLP node. Takes values {"QNI", "QNE", "QNR"}.

TOGGLE:
Flag to indicate whether the direction of a new message has to be changed compared to the direction of a received one. Takes values {true, false}.

DIRECTION:
Direction, in which the message has to be sent. Takes values {DOWNSTREAM, UPSTREAM}.

SII:
Source Identifi cation 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 Events

EV_TIMEOUT_STATE_LIFETIME:
State lifetime timer expiration

EV_TIMEOUT_REFRESH:
Refresh interval timer expiration

EV_TIMEOUT_REFRESH:
Wait-Response interval timer expiration

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

EV_TG_RESERVE:
External trigger to send a RESERVE message.

EV_TG_TEARDOWN:
External trigger to clear previously established QoS state (typically triggered by the application). It is translated to a tx_RESERVE(T-Flag) message.

EV_RX_RESPONSE:
RESPONSE message received

EV_RX_QUERY:
QUERY message received

EV_RX_RESERVE:
RESERVE message received

EV_RX_NOTIFY:
NOTIFY message received

5.4 Assumptions

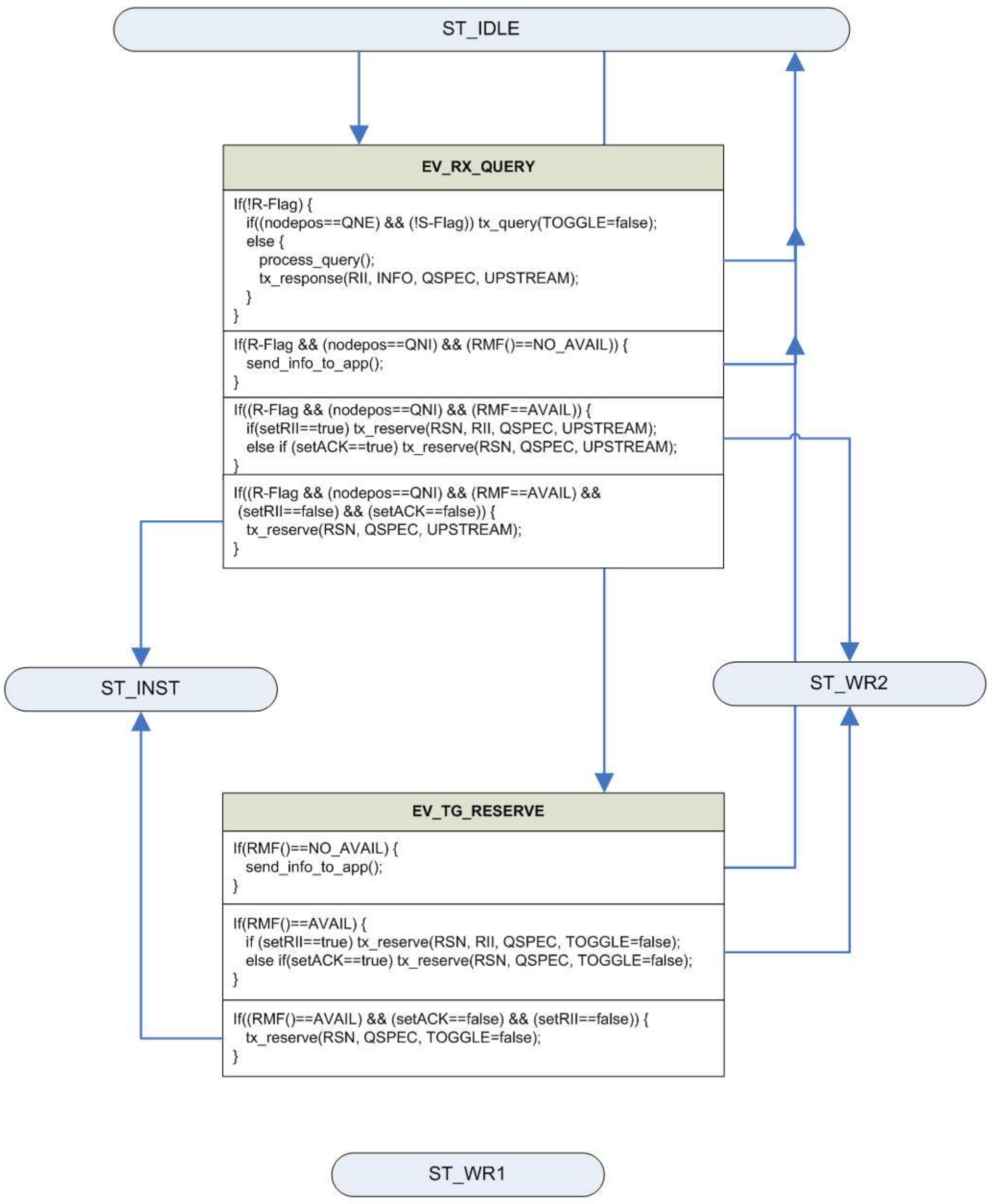
- For simplifi cation not all included objects in a message are shown. Only those that are signifi cant for the case are showed. State machines do not present handling of messages that are not signifi cant for management of the states such as certain NOTIFY and QUERY messages.
- State machines represent 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.

6. State machine

The following section presents the state machine diagrams of QoS NSLP

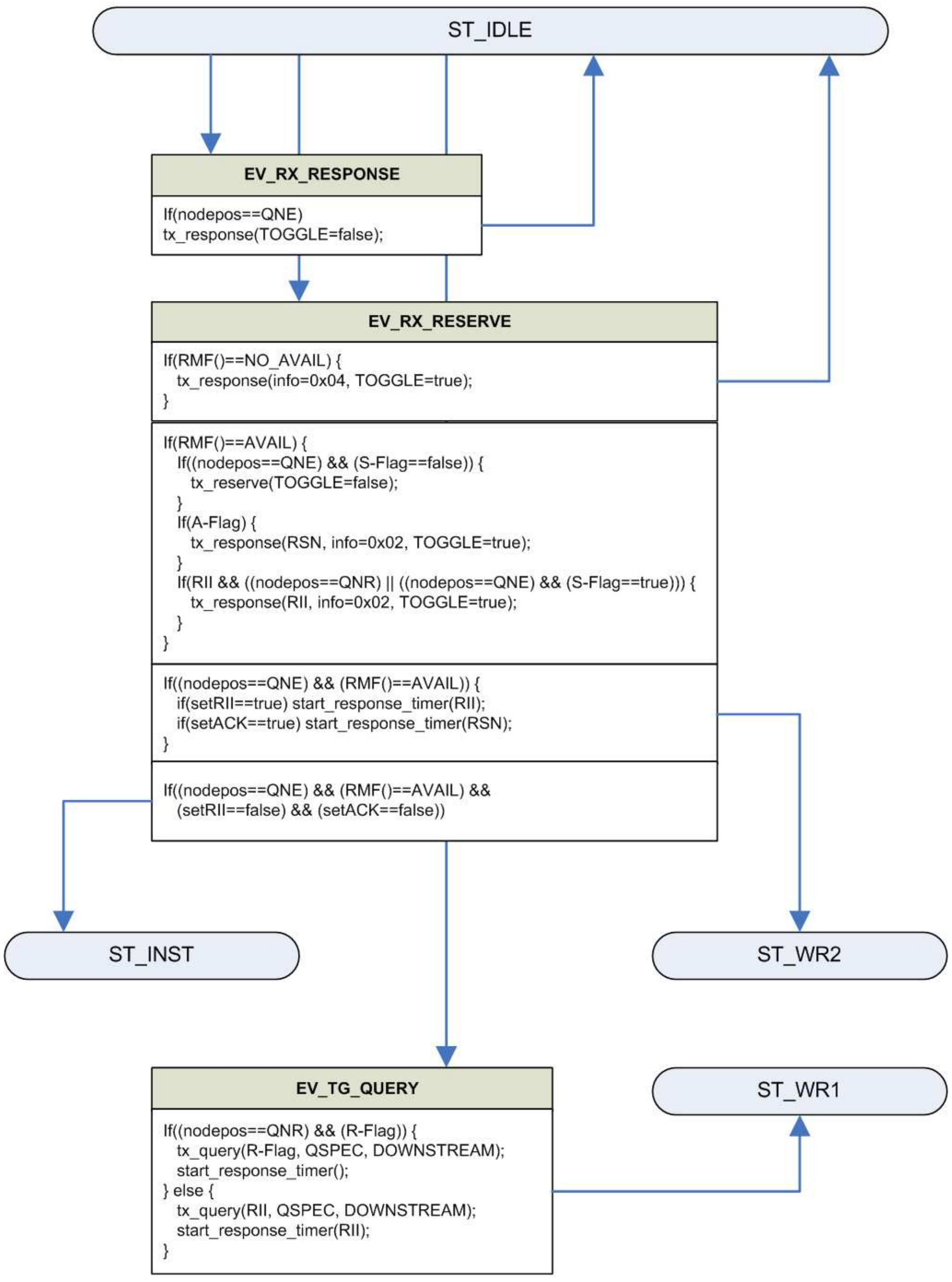
6.1 State ST_IDLE



```
If(!R-Flag) {
  if((nodepos==QNE) && (!S-Flag)) tx_query(TOGGLE=false);
  else {
    process_query();
    tx_response(RII, INFO, QSPEC, UPSTREAM);
  }
}
If(R-Flag && (nodepos==QNI) && (RMF()==NO_AVAIL)) {
  send_info_to_app();
}
If((R-Flag && (nodepos==QNI) && (RMF==AVAIL)) {
  if(setRII==true) tx_reserve(RSN, RII, QSPEC, UPSTREAM);
  else if (setACK==true) tx_reserve(RSN, QSPEC, UPSTREAM);
}
If((R-Flag && (nodepos==QNI) && (RMF==AVAIL) &&
 (setRII==false) && (setACK==false)) {
  tx_reserve(RSN, QSPEC, UPSTREAM);
}
```

```
If(RMF()==NO_AVAIL) {
  send_info_to_app();
}
If(RMF()==AVAIL) {
  if (setRII==true) tx_reserve(RSN, RII, QSPEC, TOGGLE=false);
  else if(setACK==true) tx_reserve(RSN, QSPEC, TOGGLE=false);
}
If((RMF()==AVAIL) && (setACK==false) && (setRII==false)) {
  tx_reserve(RSN, QSPEC, TOGGLE=false);
}
```

Figure 1: State ST_IDLE



EV_RX_RESPONSE

```
If(nodepos==QNE)
tx_response(TOGGLE=false);
```

EV_RX_RESERVE

```
If(RMF()==NO_AVAIL) {
  tx_response(info=0x04, TOGGLE=true);
}
```

```
If(RMF()==AVAIL) {
  If((nodepos==QNE) && (S-Flag==false)) {
    tx_reserve(TOGGLE=false);
  }
  If(A-Flag) {
    tx_response(RSN, info=0x02, TOGGLE=true);
  }
  If(RII && ((nodepos==QNR) || ((nodepos==QNE) && (S-Flag==true))) {
    tx_response(RII, info=0x02, TOGGLE=true);
  }
}
```

```
If((nodepos==QNE) && (RMF()==AVAIL)) {
  if(setRII==true) start_response_timer(RII);
  if(setACK==true) start_response_timer(RSN);
}
```

```
If((nodepos==QNE) && (RMF()==AVAIL) &&
  (setRII==false) && (setACK==false))
```

EV_TG_QUERY

```
If((nodepos==QNR) && (R-Flag)) {
  tx_query(R-Flag, QSPEC, DOWNSTREAM);
  start_response_timer();
} else {
  tx_query(RII, QSPEC, DOWNSTREAM);
  start_response_timer(RII);
}
```

Figure 2: State ST_IDLE

6.2 State ST_WR1

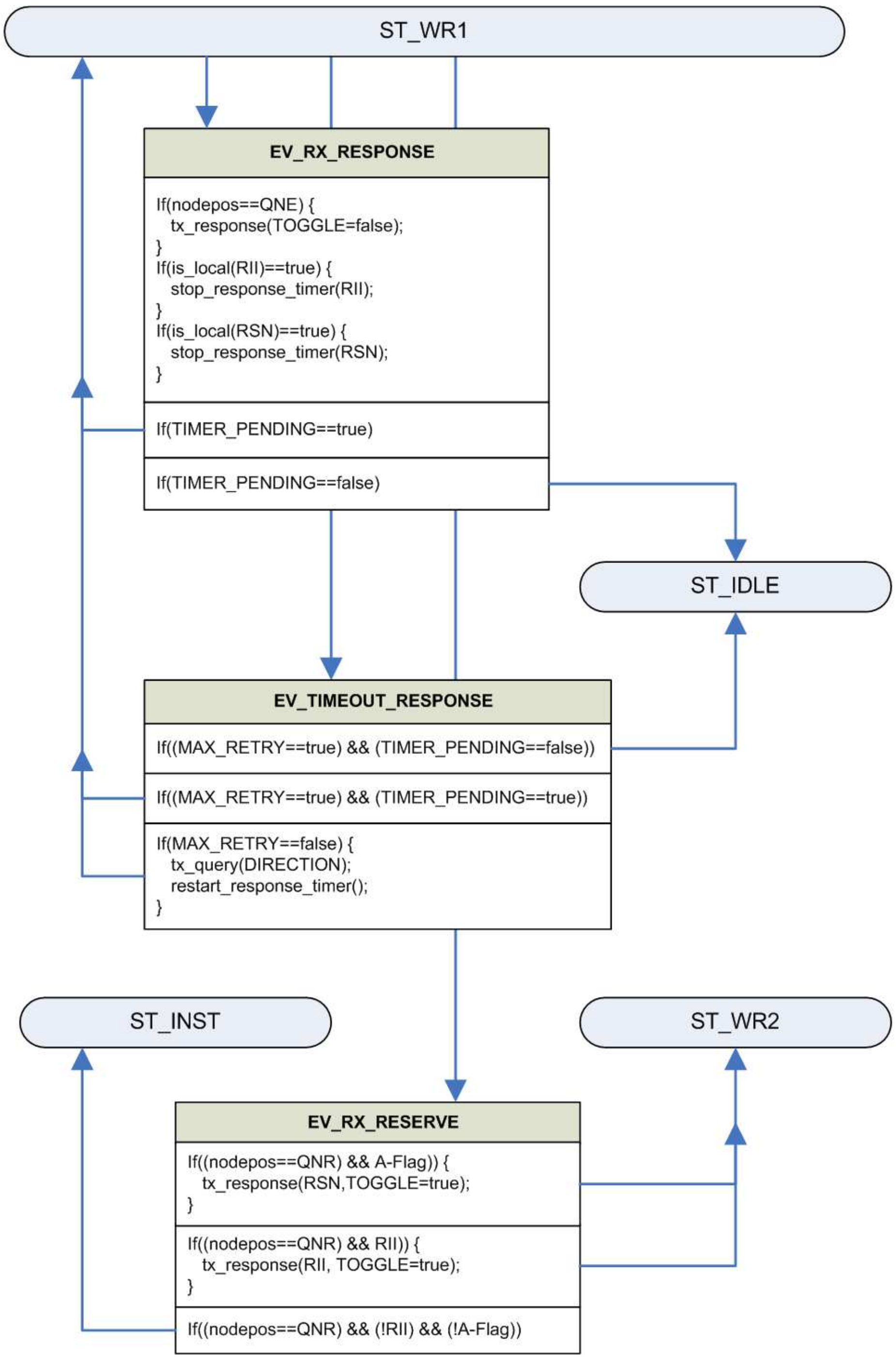
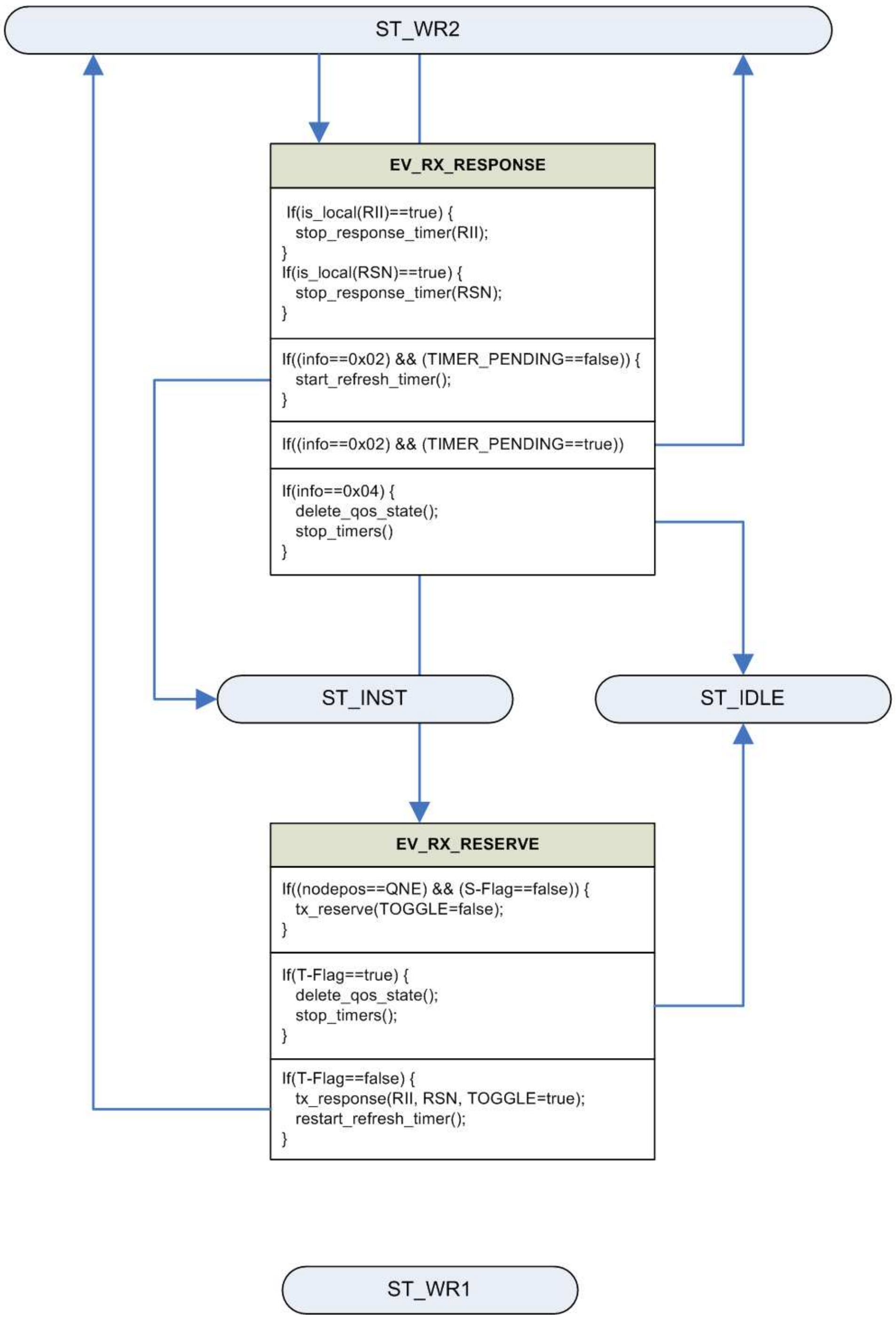

ST_WR1
EV_RX_RESPONSE
If(nodepos==QNE) {
tx_response(TOGGLE=false);
}
If(is_local(RII)==true) {
stop_response_timer(RII);
}
If(is_local(RSN)==true) {
stop_response_timer(RSN);
}
If(TIMER_PENDING==true)
If(TIMER_PENDING==false)
ST_IDLE
EV_TIMEOUT_RESPONSE
If((MAX_RETRY==true) && (TIMER_PENDING==false))
If((MAX_RETRY==true) && (TIMER_PENDING==true))
If(MAX_RETRY==false) {
tx_query(DIRECTION);
restart_response_timer();
}
ST_INST
ST_WR2
EV_RX_RESERVE
If((nodepos==QNR) && A-Flag)) {
tx_response(RSN,TOGGLE=true);
}
If((nodepos==QNR) && RII)) {
tx_response(RII, TOGGLE=true);
}
If((nodepos==QNR) && (!RII) && (!A-Flag))

Figure 3: State ST_WR1

6.3 State ST_WR2


ST_WR2
EV_RX_RESPONSE
If(is_local(RII)==true) {
stop_response_timer(RII);
}
If(is_local(RSN)==true) {
stop_response_timer(RSN);
}
If((info==0x02) && (TIMER_PENDING==false)) {
start_refresh_timer();
}
If((info==0x02) && (TIMER_PENDING==true))
If(info==0x04) {
delete_qos_state();
stop_timers()
}
ST_INST
ST_IDLE
EV_RX_RESERVE
If((nodepos==QNE) && (S-Flag==false)) {
tx_reserve(TOGGLE=false);
}
If(T-Flag==true) {
delete_qos_state();
stop_timers();
}
If(T-Flag==false) {
tx_response(RII, RSN, TOGGLE=true);
restart_refresh_timer();
}
ST_WR1
Figure 4: State ST_WR2

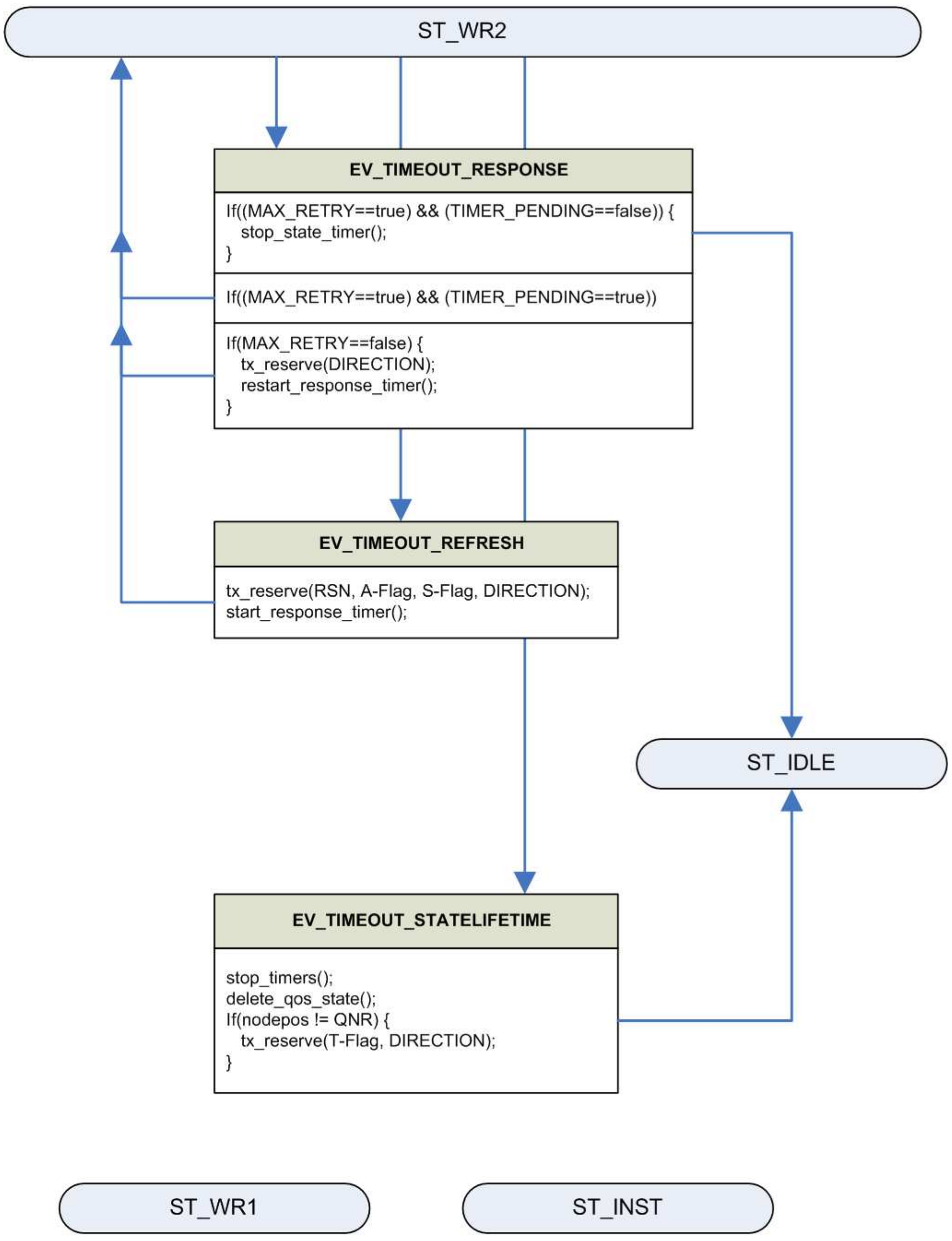


Figure 5: State ST_WR2

6.4 State ST_INST

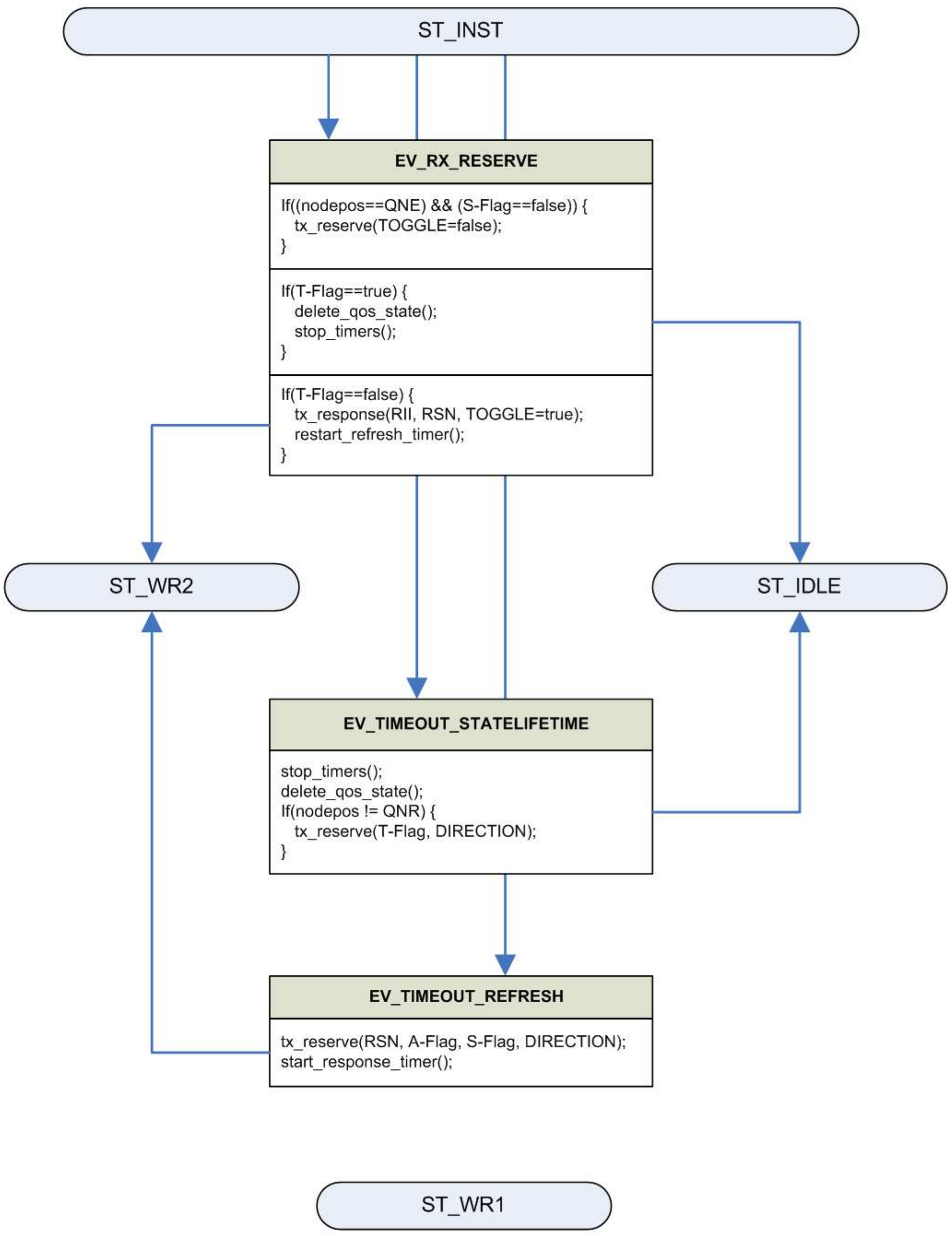


Figure 6: State ST_INST

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. 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).
2. 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.
3. 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.
4. 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.
5. 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.
6. 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

9.3 Changes in Version -03

1. Deep review of the state machine archtitecure

9.4 Changes in Version -04

1. Reduced the three state machines of QNI, QNE and QNR to one for all nodes.
2. Introduced new flags to have a finer control of the direction of the message to be sent.

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. and McDonald, A., "NSLP for Quality-of-Service Signaling", Internet draft, draft-ietf-nsis-qos-nslp-09, March 2006.

[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 ST_IDLE

```
Condition: EV_RX_QUERY
+-------------------------------------------------------+-----------+
| Action                                                | new State |
+-------------------------------------------------------+-----------+
|                                                       |           |
| If(!R-Flag) {                                         |  ST_IDLE  |
|    if((nodepos==QNE) && (!S-Flag)) {                  |           |
|       tx_query(TOGGLE=false);                         |           |
|    } else {                                           |           |
|       process_query();                                |           |
|       tx_response(RII, INFO, QSPEC, UPSTREAM);        |           |
|    }                                                  |           |
| }                                                     |           |
|                                                       |           |
+-------------------------------------------------------+-----------+
|                                                       |           |
| If(R-Flag && (nodepos==QNI) && (RMF()==NO_AVAIL)) {   |  ST_IDLE  |
|    send_info_to_app();                                |           |
| }                                                     |           |
|                                                       |           |
+-------------------------------------------------------+-----------+
|                                                       |           |
| If((R-Flag && (nodepos==QNI) && (RMF==AVAIL)) {       |  ST_WR2   |
|    if(setRII==true) {                                 |           |
|       tx_reserve(RSN, RII, QSPEC, UPSTREAM);          |           |
|    } else if (setACK==true) {                         |           |
|       tx_reserve(RSN, QSPEC, UPSTREAM);               |           |
|    }                                                  |           |
| }                                                     |           |
|                                                       |           |
+-------------------------------------------------------+-----------+
|                                                       |           |
| If((R-Flag && (nodepos==QNI) && (RMF==AVAIL) &&       |  ST_INST  |
|  (setRII==false) && (setACK==false)) {                 |           |
|    tx_reserve(RSN, QSPEC, UPSTREAM);                   |           |
| }                                                      |           |
|                                                        |           |
+--------------------------------------------------------+-----------+
```

```
Condition: EV_TG_RESERVE
+--------------------------------------------------------+-----------+
| Action                                                 | new State |
+--------------------------------------------------------+-----------+
|                                                        |           |
| If(RMF()==NO_AVAIL) {                                  |  ST_IDLE  |
|    send_info_to_app();                                 |           |
| }                                                      |           |
|                                                        |           |
+--------------------------------------------------------+-----------+
|                                                        |           |
| If(RMF()==AVAIL) {                                     |  ST_WR2   |
|    if (setRII==true) {                                 |           |
|       tx_reserve(RSN, RII, QSPEC, TOGGLE=false);       |           |
|    } else if(setACK==true) {                           |           |
|       tx_reserve(RSN, QSPEC, TOGGLE=false);            |           |
|    }                                                   |           |
| }                                                      |           |
|                                                        |           |
+--------------------------------------------------------+-----------+
|                                                        |           |
| If((RMF()==AVAIL) && (setACK==false) &&                |  ST_INST  |
|    (setRII==false)) {                                  |           |
|    tx_reserve(RSN, QSPEC, TOGGLE=false);               |           |
| }                                                      |           |
|                                                        |           |
+--------------------------------------------------------+-----------+
```

```
Condition: EV_RX_RESPONSE
+--------------------------------------------------------+-----------+
| Action                                                 | new State |
+--------------------------------------------------------+-----------+
|                                                        |           |
| If(nodepos==QNE) {                                     |  ST_IDLE  |
|    tx_response(TOGGLE=false);                          |           |
| }                                                      |           |
|                                                        |           |
+--------------------------------------------------------+-----------+
```

```
Condition: EV_RX_RESERVE
+--------------------------------------------------------+-----------+
| Action                                                | new State |
+-------------------------------------------------------+-----------+
|                                                       |           |
| If(RMF()==NO_AVAIL) {                                 |  ST_IDLE  |
|    tx_response(info=0x04, TOGGLE=true);               |           |
| }                                                     |           |
|                                                       |           |
+-------------------------------------------------------+-----------+
|                                                       |           |
| If(RMF()==AVAIL) {                                    |           |
|    If((nodepos==QNE) && (S-Flag==false)) {            |           |
|       tx_reserve(TOGGLE=false);                       |           |
|    }                                                  |           |
|    If(A-Flag) {                                       |           |
|       tx_response(RSN, info=0x02, TOGGLE=true);       |           |
|    }                                                  |           |
|    If(RII && ((nodepos==QNR) || ((nodepos==QNE) &&    |           |
|       (S-Flag==true))) {                              |           |
|       tx_response(RII, info=0x02, TOGGLE=true);       |           |
|    }                                                  |           |
| }                                                     |           |
|                                                       |           |
+-------------------------------------------------------+-----------+
|                                                       |           |
| If((nodepos==QNE) && (RMF()==AVAIL)) {                |  ST_WR2   |
|    if(setRII==true) start_response_timer(RII);        |           |
|    if(setACK==true) start_response_timer(RSN);        |           |
| }                                                     |           |
|                                                       |           |
+-------------------------------------------------------+-----------+
|                                                       |           |
| If((nodepos==QNE) && (RMF()==AVAIL) &&                |  ST_INST  |
|      (setRII==false) && (setACK==false))              |           |
|                                                       |           |
+-------------------------------------------------------+-----------+

Condition: EV_TG_QUERY
+-------------------------------------------------------+-----------+
| Action                                                | new State |
+-------------------------------------------------------+-----------+
|                                                       |           |
| If((nodepos==QNR) && (R-Flag)) {                      |   ST_WR1  |
|    tx_query(R-Flag, QSPEC, DOWNSTREAM);               |           |
|    start_response_timer();                            |           |
| } else {                                              |           |
|    tx_query(RII, QSPEC, DOWNSTREAM);                  |           |
|    start_response_timer(RII);                         |           |
| }                                                     |           |
|                                                        |           |
+--------------------------------------------------------+-----------+
```

Figure 7

A.2. State ST_WR1

```
Condition: EV_RX_RESPONSE
+--------------------------------------------------------+-----------+
| Action                                                 | new State |
+--------------------------------------------------------+-----------+
|                                                        |           |
| If(nodepos==QNE) {                                     |           |
|   tx_response(TOGGLE=false);                           |           |
| }                                                      |           |
| If(is_local(RII)==true) {                              |           |
|     stop_response_timer(RII);                          |           |
| }                                                      |           |
| If(is_local(RSN)==true) {                              |           |
|   stop_response_timer(RSN);                            |           |
| }                                                      |           |
|                                                        |           |
+--------------------------------------------------------+-----------+
|                                                        |           |
| If(TIMER_PENDING==true)                                |  ST_WR1   |
|                                                        |           |
+--------------------------------------------------------+-----------+
|                                                        |           |
| If(TIMER_PENDING==false)                               |  ST_IDLE  |
|                                                        |           |
+--------------------------------------------------------+-----------+

Condition: EV_TIMEOUT_RESPONSE
+--------------------------------------------------------+-----------+
| Action                                                 | new State |
+--------------------------------------------------------+-----------+
|                                                        |           |
| If((MAX_RETRY==true) && (TIMER_PENDING==false))        |  ST_IDLE  |
|                                                        |           |
+--------------------------------------------------------+-----------+
|                                                        |           |
| If((MAX_RETRY==true) && (TIMER_PENDING==true))         |  ST_WR1   |
|                                                        |           |
+--------------------------------------------------------+-----------+
|                                                        |           |
|  If(MAX_RETRY==false) {                                |  ST_WR1   |
|    tx_query(DIRECTION);                                |           |
|    restart_response_timer();                           |           |
| }                                                      |           |
|                                                        |           |
+--------------------------------------------------------+-----------+

Condition: EV_RX_RESERVE
+--------------------------------------------------------+-----------+
| Action                                                 | new State |
+--------------------------------------------------------+-----------+
|                                                        |           |
| if((nodepos==QNR) && (A-Flag)) {                       |  ST_WR2   |
|    tx_response(RSN, TOGGLE=true);                      |           |
| }                                                      |           |
|                                                        |           |
+--------------------------------------------------------+-----------+
|                                                        |           |
| If((nodepos==QNR) && RII)) {                           |  ST_WR2   |
|    tx_response(RII, TOGGLE=true);                      |           |
| }                                                      |           |
|                                                        |           |
+--------------------------------------------------------+-----------+
|                                                        |           |
| If((nodepos==QNR) && (!RII) && (!A-Flag))              |  ST_INST  |
|                                                        |           |
+--------------------------------------------------------+-----------+
```

Figure 8

A.3. State ST_WR2

```
Condition: EV_RX_RESPONSE
+--------------------------------------------------------+-----------+
| Action                                                 | new State |
+--------------------------------------------------------+-----------+
|                                                        |           |
| If(is_local(RII)==true) {                              |           |
|    stop_response_timer(RII);                           |           |
| }                                                      |           |
| If(is_local(RSN)==true) {                              |           |
|    stop_response_timer(RSN);                           |           |
| }                                                      |           |
|                                                        |           |
| If((info==0x02) && (TIMER_PENDING==false)) {           |  ST_INST  |
|    start_refresh_timer();                              |           |
| }                                                      |           |
|                                                        |           |
+--------------------------------------------------------+-----------+
|                                                        |           |
| If((info==0x02) && (TIMER_PENDING==true))              |  ST_WR2   |
|                                                        |           |
+--------------------------------------------------------+-----------+
|                                                        |           |
| If(info==0x04) {                                       |  ST_IDLE  |
|    delete_qos_state();                                 |           |
|    stop_timers()                                       |           |
| }                                                      |           |
|                                                        |           |
+--------------------------------------------------------+-----------+

Condition: EV_RX_RESERVE
+--------------------------------------------------------+-----------+
| Action                                                 | new State |
+--------------------------------------------------------+-----------+
|                                                        |           |
| If((nodepos==QNE) && (S-Flag==false)) {                |           |
|    tx_reserve(TOGGLE=false);                           |           |
|  }                                                     |           |
|                                                        |           |
| If(T-Flag==true) {                                     |  ST_IDLE  |
|    delete_qos_state();                                 |           |
|    stop_timers();                                      |           |
| }                                                      |           |
|                                                        |           |
+--------------------------------------------------------+-----------+
|                                                        |           |
| If(T-Flag==false) {                                    |  ST_WR2   |
|    tx_response(RII, RSN, TOGGLE=true);                 |           |
|    restart_refresh_timer();                            |           |
| }                                                      |           |
|                                                        |           |
+--------------------------------------------------------+-----------+

Condition: EV_TIMEOUT_RESPONSE
+--------------------------------------------------------+-----------+
| Action                                                 | new State |
+--------------------------------------------------------+-----------+
|                                                        |           |
| If((MAX_RETRY==true) && (TIMER_PENDING==false)) {      |  ST_IDLE  |
|    stop_state_timer();                                 |           |
| }                                                      |           |
|                                                        |           |
+--------------------------------------------------------+-----------+
|                                                        |           |
| If((MAX_RETRY==true) && (TIMER_PENDING==true))         |  ST_WR2   |
|                                                        |           |
+--------------------------------------------------------+-----------+
|                                                        |           |
| If(MAX_RETRY==false) {                                 |  ST_WR2   |
|    tx_reserve(DIRECTION);                              |           |
|    restart_response_timer();                           |           |
| }                                                      |           |
|                                                        |           |
+--------------------------------------------------------+-----------+

Condition: EV_TIMEOUT_REFRESH
+--------------------------------------------------------+-----------+
| Action                                                 | new State |
+--------------------------------------------------------+-----------+
|                                                        |           |
| tx_reserve(RSN, A-Flag, S-Flag, DIRECTION);            |  ST_WR2   |
| start_response_timer();                                |           |
|                                                        |           |
+--------------------------------------------------------+-----------+

Condition: EV_TIMEOUT_STATELIFETIME
+--------------------------------------------------------+-----------+
| Action                                                 | new State |
+--------------------------------------------------------+-----------+
|                                                        |           |
| stop_timers();                                         |  ST_IDLE  |
| delete_qos_state();                                    |           |
| If(nodepos != QNR) {                                   |           |
|    tx_reserve(T-Flag, DIRECTION);                      |           |
| }                                                      |           |
|                                                        |           |
+--------------------------------------------------------+-----------+
```

Figure 9

A.4. State ST_INST

```
Condition: EV_RX_RESERVE
+--------------------------------------------------------+-----------+
| Action                                                 | new State |
+--------------------------------------------------------+-----------+
|                                                        |           |
| If((nodepos==QNE) && (S-Flag==false)) {                |           |
|    tx_reserve(TOGGLE=false);                           |           |
| }                                                      |           |
|                                                        |           |
+--------------------------------------------------------+-----------+
|                                                        |           |
| If(T-Flag==true) {                                     |  ST_IDLE  |
|    delete_qos_state();                                 |           |
|    stop_timers();                                      |           |
| }                                                      |           |
|                                                        |           |
| If(T-Flag==false) {                                    |  ST_WR2   |
|    tx_response(RII, RSN, TOGGLE=true);                 |           |
|    restart_refresh_timer();                            |           |
| }                                                      |           |
|                                                        |           |
+--------------------------------------------------------+-----------+

Condition: EV_TIMEOUT_STATELIFETIME
+--------------------------------------------------------+-----------+
| Action                                                 | new State |
+--------------------------------------------------------+-----------+
|                                                        |           |
| stop_timers();                                         |  ST_IDLE  |
| delete_qos_state();                                    |           |
| If(nodepos != QNR) {                                   |           |
|    tx_reserve(T-Flag, DIRECTION);                      |           |
| }                                                      |           |
|                                                        |           |
+--------------------------------------------------------+-----------+

Condition: EV_TIMEOUT_REFRESH
+--------------------------------------------------------+-----------+
| Action                                                 | new State |
+--------------------------------------------------------+-----------+
|                                                        |           |
| tx_reserve(RSN, A-Flag, S-Flag, DIRECTION);            |  ST_WR2   |
| start_response_timer();                                |           |
|                                                        |           |
+--------------------------------------------------------+-----------+
```

Figure 10

Authors' Addresses

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

Email: fu@cs.uni-goettingen.de

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

Email: Hannes.Tschofenig@siemens.com

Tseno Tsenov
Siemens
Otto-Hahn-Ring 6
Munich, Bayern 81739
Germany

Email: tseno.tsenov@mytum.de

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

Email: bschloer@cs.uni-goettingen.de

Acknowledgement

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