NSIS Working Group
Internet Draft

Sven Van den Bosch (Editor)
Alcatel
Georgios Karagiannis
Univ. of Twente/Ericsson
Andrew McDonald
Siemens/Roke Manor Research

Document: draft-ietf-nsis-qos-nslp-01.txt
Expires: April 2004 October 2003

NSLP for Quality-of-Service signaling

Status of this Memo

Abstract

This draft describes an NSIS Signaling Layer Protocol (NSLP) for signaling QoS reservations in the Internet. It is in accordance with the framework and requirements developed in NSIS.

Together with the NTLP, it provides functionality similar to RSVP and extends it. The QoS-NSLP is independent of the underlying QoS specification or architecture and provides support for different reservation models. It is simplified by the elimination of support for multicast flows.

This version of the draft focuses on the basic protocol structure. It identifies the different message types and describes the basic operation of the protocol to create, refresh, modify and teardown a reservation or to obtain information on the characteristics of the associated data path.

Conventions used in this document

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 RFC-2119 [1].

Table of Contents

1. Introduction

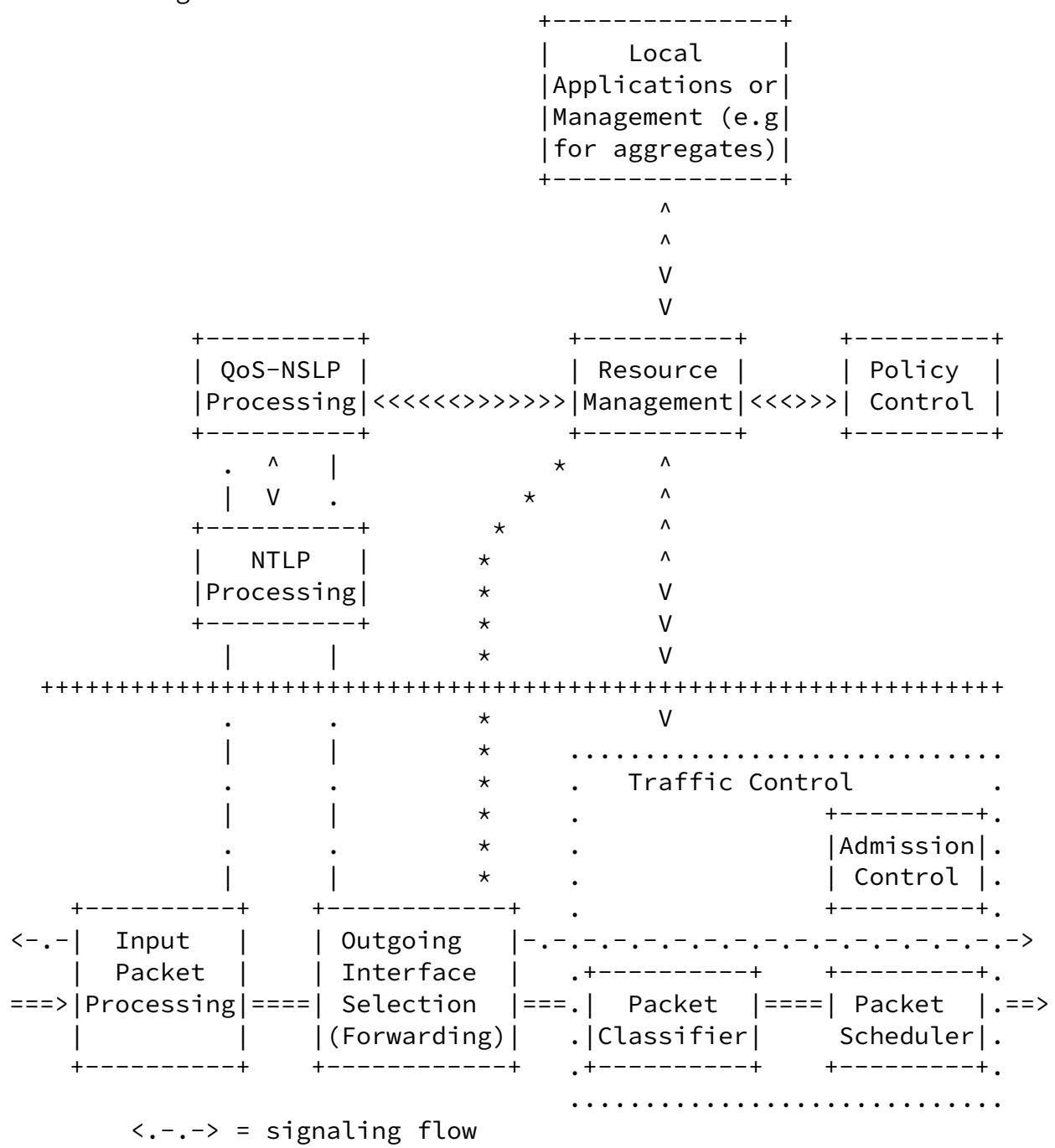
1.1 Scope and background

This document defines a Quality of Service (QoS) NSIS Signaling Layer Protocol (NSLP), henceforth referred to as the "QoS-NSLP". This protocol establishes and maintains state at nodes along the path of a data flow for the purpose of providing some forwarding resources for that flow. It is intended to satisfy the QoS-related requirements of [2]. This QoS-NSLP is part of a larger suite of signaling protocols, whose structure is outlined in [3]; this defines a common NSIS Transport Layer Protocol (NTLP) which QoS-NSLP uses to carry out many aspects of signaling message delivery.

The design of QoS-NSLP is conceptually similar to RSVP [4], and uses soft-state peer-to-peer refresh messages as the primary state management mechanism. Although there is no backwards compatibility at the level of protocol messages, interworking with RSVP at a signaling application gateway would be possible in some circumstances. QoS-NSLP extends the set of reservation mechanisms to meet the requirements of [2], in particular support of sender or receiver-initiated reservations, as well as a type of bi-directional reservation and support of reservations between arbitrary nodes, e.g. edge-to-edge, end-to-access, etc. On the other hand, there is no support for IP multicast.

QoS-NSLP does not mandate any specific 'QoS Model', i.e. a particular QoS provisioning method or QoS architecture; this is similar to (but stronger than) the decoupling between RSVP and the IntServ architecture [5]. It should be able to carry information for various QoS models; the specification of Integrated Services for use with RSVP given in [6] could form the basis of one QoS model.

This section presents a logical model for the operation of the QoS-NSLP and associated provisioning mechanisms within a single node. It is adapted from the discussion in section 1 of [4]. The model is shown in Figure 1.



```
=====> = data flow (sender -->receiver)
<<<>>> = control and configuration operations
****** = routing table manipulation
```

Figure 1: QoS-NSLP in a Node

This diagram shows an example implementation scenario where QoS conditioning is performed on the output interface. However, this does

not limit the possible implementations. For example, in some cases traffic conditioning may be performed on the incoming interface, or it may be split over the input and output interfaces.

From the perspective of a single node, the request for QoS may result from a local application request, or from processing an incoming QoS-NSLP message.

- The 'local application case' includes not only user applications (e.g. multimedia applications) but also network management (e.g. initiating a tunnel to handle an aggregate, or interworking with some other reservation protocol - such as RSVP). In this sense, the model does not distinguish between hosts and routers.

- The 'incoming message' case requires NSIS messages to be captured during input packet processing and handled by the NTLP. Only messages related to QoS are passed to the QoS-NSLP. The NTLP may also generate triggers to the QoS-NSLP (e.g. indications that a route change has occurred).

The QoS request is handled by a local 'resource management' function, which coordinates the activities required to grant and configure the resource.

- The grant processing involves two local decision modules, 'policy control' and 'admission control'. Policy control determines whether the user has administrative permission to make the reservation. Admission control determines whether the node has sufficient available resources to supply the requested QoS.

- If both checks succeed, parameters are set in the packet classifier and in the link layer interface (e.g., in the packet scheduler) to obtain the desired QoS. Error

notifications are passed back to the request originator. The resource management function may also manipulate the forwarding tables at this stage, to select (or at least pin) a route; this must be done before interface-dependent actions are carried out (including forwarding outgoing messages over any new route), and is in any case invisible to the operation of the protocol.

Policy control is expected to make use of a AAA service external to the node itself. Some discussion can be found in [7] and [8]. More generally, the processing of policy and resource management functions may be outsourced to an external node leaving only 'stubs' co-located with the NSLP; this is not visible to the protocol operation,

although it may have some influence on the detailed design of protocol messages to allow the stub to be minimally complex.

The group of user plane functions, which implement QoS for a flow (admission control, packet classification, and scheduling) is sometimes known as 'traffic control'.

Admission control, packet scheduling, and any part of policy control beyond simple authentication have to be implemented using specific definitions for types and levels of QoS; we refer to this as a QoS model. Our assumption is that the QoS-NSLP is independent of the QoS model, that is, QoS parameters (e.g. IntServ service elements) are interpreted only by the resource management and associated functions, and are opaque to the QoS-NSLP itself. QoS Models are discussed further in section 2.2.

The final stage of processing for a resource request is to indicate to the QoS-NSLP protocol processing that the required resources have been configured. The QoS-NSLP may generate an acknowledgement message in one direction, and may propagate the resource request forwards in the other. Message routing is (by default) carried out by the NTLP module. Note that while Figure 1 shows a unidirectional data flow, the signaling messages can pass in both directions through the node, depending on the particular message and orientation of the reservation.

1.3 Terminology

The terminology defined in [3] applies to this draft. In addition, the following terms are used:

- edge QNE - a QNE that is located at the boundary of an administrative domain, e.g., Diffserv.

- egress QNE - an edge QNE that handles the traffic as it leaves the domain.

- ingress QNE - an edge QNE that handles the traffic as it enters the domain.

- interior QNE - a QNE that is part of an administrative domain, e.g., Diffserv, and is not an edge QNE.

- QNE - an NSIS Entity (NE), which supports the QoS-NSLP.

- QNI - the first node in the sequence of QNEs that issues a reservation request.

- QNR - the last node in the sequence of QNEs that receives a reservation request.

- QoS-NSLP stateful operation - mode of operation where per-flow reservation states are created, maintained and used.

- QoS-NSLP reduced-state operation - mode of operation where reservation states with a coarser granularity (e.g. per-class) are created, maintained and used.

- QoS-NSLP stateless operation - mode of operation where reservation state is not needed and not created.

- reduced state QNE - a QNE that supports the QoS NSLP reduced state operation.

- Source or message source - QoS NSLP messages are sent peer-to-peer. This means that a QNE considers its adjacent stateful upstream or downstream peer to be the source of a QoS NSLP message. I.e. the source of a QoS NSLP message is not necessarily the QNI.

- Stateful QNE - a QNE that supports the QoS NSLP stateful operation.

- Stateless QNE - a QNE that supports the QoS NSLP stateless operation.

2. Protocol design principles

2.1 Basic protocol structure

Messages are normally passed from the NSLP to the NTLP via an API, which also specifies the signaling application (as QoS-NSLP), the flow/session identifier, and an indication of the intended direction - towards data sender or receiver. On reception, the NTLP provides the same information to the QoS-NSLP.

The protocol specifies four message types (RESERVE, QUERY, RESPONSE and NOTIFY) and two objects (QSpec and Policy object).

- Each protocol message includes header information that identifies the message type and determines which objects it may carry.

- A protocol message also contains Control Information (CI); this is a group of objects that qualify and/or restrict the action that a QNE may perform on the QoS message. Control Information is always associated to a QSpec, i.e. CI and QSpec always come in pairs. This is important when we have locally valid objects e.g. for reduced state operation, because both a CI and a QSpec will be added.

- QSpec object: The QoS parameters defined by the QoS model are carried in a QSpec object. It describes the QoS that is being reserved.

- Policy object: The Policy object authenticates and authorizes the requester of the QoS reservation.

This memo intends to define message types and control information for the QoS-NSLP (generic to all QoS models). It also introduces the QoS model (section 2.2) and Policy object (section 2.3). However, a detailed description of these objects will need to be provided in separate documents.

2.2 QoS model

A QoS model is a defined mechanism for achieving QoS as a whole. The specification of a QoS model includes a description of its QoS parameter information, as well as how that information should be treated or interpreted in the network. In that sense, the QoS model goes beyond the QoS-NSLP protocol level in that it could also describe underlying assumptions, conditions and/or specific provisioning mechanisms appropriate for it.

A QoS model provides a specific set of parameters to be carried in the QSpec, or restricts the values these parameters can take. Integrated Services [5], Differentiated Services[9] and RMD [11] are all examples of QoS models. There is no restriction on the number of QoS models. QoS models may be local, meaning that they are private to one network, implementation or vendor specific or global, meaning that they must be implementable by different networks and vendors.

The QSpec contains a set of parameters and values describing the requested resources. It is opaque to the QoS-NSLP and similar in purpose to the TSpec, RSpec and AdSpec specified in [4,6]. At each QNE, its content is interpreted by the resource management function for the purposes of policy control and traffic control (including admission control and configuration of the packet classifier and scheduler).

Different QoS models may share a common set of QoS parameters. Standardizing a QSpec template would allow for a consistent specification of common QoS parameters in different QoS models. It would simplify the introduction of new QoS models as well as greatly increasing interoperability. Other examples of QoS description for which a template was standardized are e.g. the MEF Ethernet Service definition [10]. The QSpec template is envisaged to be useful for specifying a wide range of QoS descriptions. It would, for instance, need to support both qualitative and quantitative QoS specifications in order to provide for terminals that are not willing or capable to quantitatively describe the traffic behavior they require. A strawman QSpec template is described in Appendix A. In addition to standardizing a QSpec template, individual QoS models could be standardized, but we would expect them to have local significance only.

2.3 Authentication and authorization

Future versions of this document will contain a discussion on the

Policy object, based on [7,8].

2.4 Control Information

Any QoS-NSLP message may contain a set of objects for conveying information about how these messages should be handled. This set of objects is collectively known as Control Information (CI).

Control Information can have a local scope (Local Control Information or LCI) or global scope.

The ability of a QNE to explicitly request a RESPONSE to its messages (section 2.4.1) and the ability to limit the scope of a message are both examples of Control Information (section 2.4.2). Future versions of the draft may include more examples of Control Information objects.

2.4.1ResponseRequest

Some QNEs may require explicit responses to messages they send. A QUERY message (section 3.2), for instance, will always cause a RESPONSE message (section 3.3) to be sent. The QNE that generates a RESERVE message (section 3.1) may explicitly request that it triggers a RESPONSE.

If a RESPONSE message is required or desired, the QNE inserts a ResponseRequest object in the message. The exact format of this object will be determined in a future version of this specification.

In order to be able to match a response back to the corresponding request , the ResponseRequest object contains Response Identification Information (RII). The RII is inserted by the QNE that requests the response and is copied into the corresponding RESPONSE message. QNEs that receive a RESPONSE containing their own RII should not forward the message further.

The ResponseRequest object also contains a flag to indicate whether it is requesting a response from the next node (a 'local' response), or a response from the QNR. More general ways of specifying scoping information will be investigated in future versions of this document.

Note that if an intermediate QNE desires a RESPONSE as well, it should not change the RII or add another ResponseRequest since the

RESPONSE will pass through it anyway.

2.4.2Message and object scoping

QoS-NSLP supports locally defined QoS model specifications by means of local QSpecs and local Control Information. Messages containing such local information need to be scoped, i.e. it should be possible to restrict the forwarding of such a message to the domain in which it is applicable. It may also be needed to scope individual objects in the message.

One example of message scoping uses the RII to limit the forwarding of a RESPONSE to the QNE that requested it. Another example might be controlling the scope of a tearing RESERVE.

The two scopes of "single QoS-NSLP hop" and "whole path" appear to be useful concepts. In case no scoping information is specified, the default case of "whole path" must be assumed. It is still to be determined what the best way(s) of specifying more flexible scoping information, such as per-domain are.

2.5 Nested protocol operation

For various reasons an operator may want to use a different type of QoS specification (i.e. a different QoS model) within its network. This may be done, for example, in order for QNEs to be able to store less reservation state. In order to allow some local selection of which QoS Model to use without destroying all end-to-end aspects of

the signaling, QoS-NSLP allows a nesting of QoS Models by 'stacking' more than one pair of Control Information / QSpec object within a message.

The details of this operation will be covered in future versions of this draft. This nested operation would allow localized QSpec objects and control information to be used along parts of the path. It also has dependencies on the scoping facilities provided by the protocol.

2.6 Protocol extensibility

QoS-NSLP is designed in modular way making it possible to support different QoS models and other future extensions of the protocol. Extensions can be provided by specifying new QoS models and their usage or defining new QoS-NSLP objects (similar to the current QSpec

and Policy object) and their usage.

In QoS-NSLP the basic operation mechanism of the protocol is clearly separated from the control and traffic information being transported. This logical separation makes it possible to develop a variety of new QoS models within one protocol frame. Each of the QoS-NSLP message types can carry, for each supported QoS Model, QoS descriptions by using object templates. This memo discusses a template for the QoS specification (QSpec). A future version will also cover the Policy object.

A new QoS model is specified by defining the internal content of the template objects and by defining how QoS provisioning is done in different nodes. Another important aspect of defining a new QoS model is the specification of the associated CI used in these templates, which allow particular setting in different nodes.

A QoS model may have internal structure into different components, some of which may have application limited to particular nodes while being opaque to others.

2.7 Implementation flexibility for scalability

The QoS-NSLP protocol supports QoS architectures in which some QNEs may not be able or willing to store per-session state. A stateless operation is conceivable, in which QNEs interior to a domain store neither NSLP nor NTLP state. They rather e.g. just send and receive messages to provide information on currently available resources, and react upon overload detection. Also reduced-state operation is conceivable, in which QNEs do not store full per session (per-flow or per-aggregate) state in both NSLP and NTLP, but rather, e.g. one per-

class state in NSLP and no state in NTLP. Examples of how QoS can be achieved with stateless and reduced-state operation are described in RMD [11].

Stateless and reduced state operation is only applicable under certain circumstances. Stateless or reduced state QNEs are not able to perform message bundling, message fragmentation and reassembly (at the NSLP) or congestion control. They are not able to establish and maintain security associations with their neighbors, which means they can only be applied in a trusted environment. For these reasons, a typical application of stateless or reduced state QNEs is for signaling within a single domain where the edges of the domain are

stateful QNEs.

Stateless and reduced state QoS-NSLP operation makes the most sense when (some nodes of) the underlying NTLP is (are) able to operate in a stateless way as well. Such nodes should not worry about keeping reverse path state, message fragmentation and reassembly (at the NTLP), congestion control or security associations. A stateless or reduced state QNE will be able to inform the underlying NTLP of this situation. We rely on the NTLP design to allow for a mode of operation that can take advantage of this information (section 6.1).

3. Basic message types

The QoS-NSLP contains four message types: RESERVE, QUERY, RESPONSE and NOTIFY.

These messages have different conceptual properties; the fundamental properties of the message determine how it should be analyzed from the perspective of re-ordering, loss, end-to-end reliability and so on. It is important for applications to know whether NSLP messages can be repeated, discarded or merged and so on (e.g. for edge mobility support, rerouting, etc). Indeed, the ordering of messages that don't manipulate state at QNEs matters little, whereas the way that messages that manipulate state are interleaved matters very much. Therefore NSLP is designed such that the message type identifies whether a message is manipulating state (in which case it is idempotent) or not (it is impotent).

3.1 RESERVE

The RESERVE message is used to manipulate QoS reservation state in QNEs. A RESERVE message may create, refresh, modify or remove such state.

The RESERVE message opaquely transports a QSpec object, describing the desired service level and a Policy object, authorizing the requestor of the service. It also carries the lifetime of the reservation (most likely in the Control Information part). Each node may insert a local QSpec object and or local Control Information (LCI) provided it has a way of scoping this information (e.g. at the boundary of a domain).

In some cases, a QNE needs to be able to distinguish between newly

created, modified state or refreshed state based on the RESERVE message alone (not in combination with state information obtained from previous messages). Therefore, one or more additional flags that provide this differentiation may be needed. Future versions of this draft will describe how such flag(s) will be provided and used.

In order to clearly distinguish between a RESERVE message that sets the reserved resources to zero and a RESERVE message that tears down QoS-NSLP state completely, a tear bit should be foreseen. Note that the potential initiation of (reverse path) state removal at the NTLP is a separate issue. This will be signaled over the API between NTLP and QoS-NSLP.

RESERVE messages are sent peer-to-peer. This means that a QNE considers its adjacent upstream or downstream peer to be the source of the RESERVE message. Note that two nodes that are adjacent at the QoS-NSLP layer may in fact be separated by several NTLP hops. A QoS-NSLP node may want to be able to detect changes in its QoS-NSLP peers, or send a message to an explicitly identified node, e.g. for tearing down a reservation on an old path. This functionality can be provided by keeping track of a Source Identification Information (SII) object that is similar in nature to the RSVP_HOP object described in [4]. We assume such an SII (section 6.2) to be available as a service from the NTLP.

The RESERVE message is idempotent; the resultant effect is the same whether a message is received once or many times. In addition, the ordering of RESERVE messages matters - an old RESERVE message does not replace a newer one. Both of these features are required for protocol robustness - messages may be re-ordered on route (e.g. because of mobility, or at intermediate NTLP nodes) or spuriously retransmitted.

In order to tackle these issues, the RESERVE message contains a Reservation Sequence Number (RSN). An RSN is an incrementing sequence number that indicates the order in which state modifying actions are performed by a QNE. The RSN has local significance only, i.e. between QNEs. Attempting to make an identifier that was unique in the context of a session identifier but the same along the complete path would be

very hard. Since RESERVEs can be sent by any node on the path that maintains reservation state (e.g. for path repair) we would have the difficult task of attempting to keep the identifier synchronized along the whole path. The ordering only matters between a pair of

nodes maintaining reservation state, i.e. stateful QNEs. This means that we can instead make the Reservation Sequence Number unique just between a pair of neighboring stateful QNEs. Note that an alternative might be for the NTLP to guarantee in-order delivery between the NSLP peers.

A Flow Identifier groups together state items for a single flow. The RSN is one of these state items, and is used to identify reordering of messages and to allow the use of partial refresh messages. The state items for a number of flows can be linked together and identified as part of a single reservation using a Session Identifier. The identifiers play complementary roles in the management of QoS NSLP state.

The sender of a RESERVE message may want to receive some confirmation from a downstream node. For this purpose the RESERVE message may contain a ResponseRequest object (section 2.4.1).

3.2 QUERY

A QUERY message is used to request information about the data path without making a reservation. This functionality can be used to 'probe' the network for path characteristics or for support of certain QoS models. The information obtained from a QUERY may be used in the admission control process of a QNE (e.g. in case of measurement-based admission control). Note that a QUERY does not change existing reservation state, nor does it cause state to be installed in nodes other than the one that generated the QUERY.

A QUERY message contains a ResponseRequest object containing Response Identification Information (RII) that allows matching back RESPONSE to the QUERY request. It is transported unchanged along the data path and may be used to scope the RESPONSE to a QUERY message (section 3.3).

The QUERY message can gather information along the data path in a number of objects. Some of these objects may be part of the QoS model. Others may be generic to the QoS-NSLP protocol.

A QUERY message may be scoped. If scoping information is present, this means that the QUERY is not supposed to go down the entire path to the QNR but rather that is has a maximum number of QNE hops it can travel. Else, the default case (whole path) is assumed. It is

currently an open issue what the best way of message scoping is (section 7.1)

3.3 RESPONSE

The REPONSE message is used to provide information about the result of a previous QoS-NSLP message, e.g. confirmation, error or information resulting from a query. The RESPONSE message is impotent, it does not cause any state to be installed or modified.

A QNE may want to receive a RESPONSE message to inform it that the reservation has been successfully installed. The RESERVE message may contain a ResponseRequest object for this purpose. Such a ResponseRequest object can be used to request an explicit confirmation of the state manipulation signaled in the RESERVE message.

The forwarding of the RESPONSE message along the path does not necessarily imply the existence of NTLP reverse-path state at every node. For example, the NTLP may have a mechanism to pass a message directly from the egress to the ingress of a region of QNEs that do not store per-flow reverse-path state.

If a QNE or the QNR is unable to provide the requested information or if the response is negative, the RESPONSE message may be used to carry an error message.

A QUERY always causes a RESPONSE to be sent. Therefore, a QUERY message will always contain a ResponseRequest object. A RESERVE may cause a RESPONSE to be sent if this is explicitly requested or when an error occurs.

3.4 NOTIFY

NOTIFY messages are used to convey information to a QNE. NOTIFY messages are impotent (they do not cause a change in state directly).

NOTIFY messages differ from RESPONSEs in that they need not refer to any particular state or previously received message. They are sent asynchronously. The NOTIFY message itself does not trigger or mandate any action in the receiving QNE.

The information conveyed by a NOTIFY message is typically related to error conditions. One example would be notification to an upstream peer about state being torn down.

4. Basic outline of operation

4.1 Making a reservation

To make a new reservation, the QNI builds a RESERVE message containing a QSpec specifying the resources required and a Policy object containing user identification and authorization information. It initializes a Reservation Sequence Number (RSN) counter for the flow and provides the initial value in the RESERVE message. This message is passed to the NTLP, which delivers it to the next QNE. A ResponseRequest object may be introduced if a confirmation of successful reservation is desired.

At the next QNE the RESERVE message is examined. Policy control and admission control decisions are made. The node performs appropriate actions based on the content of any QSpec object in the message. The QoS NSLP generates a new RESERVE message based on the one it received. This is passed to the NTLP, which forwards it to the next QNE.

The same processing is performed at further QNEs on the path, up to the QNR.

At the QNR, having determined that it is the last QNE (section 6.3) on the path, the attempt to send a further RESERVE message is aborted. The NR may rely on information obtained from the NTLP to determine that it is the last node (section 6.3).

A QNE may want to store per-flow state for a number of reasons. It may be that per-flow reservations are required to provide better granularity of reserved resources. Some additional functions can also be provided by the NSLP by storing NSLP state.

If the QNE wishes to be able to detect changes in the neighboring QNE (i.e. that a future RESERVE message did not come from the same node as the one that sent this RESERVE), then it records the identity of that node (e.g. SII). The SII on the outgoing message is defined by the NTLP.

If the QNE also wants to detect re-ordered or duplicated RESERVE messages then it must also store the Reservation Sequence Number. When an NSLP node that maintains per-flow reservation state (and may generate refreshes) generates a RESERVE message to forward on to the next peer it must replace the Reservation Sequence Number in the message it received with one of its own. If the NSLP does not maintain any per-flow reservation state (and thus cannot generate new

RESERVE messages (refreshes, tears, etc) on its own) then it can use the RSN from the RESERVE message it received, rather than maintaining its own sequence numbers. By managing the sequence numbers in this manner, the source of the RESERVE does not need to determine how the next NSLP node will process the message.

Layering approaches (as outlined in section 2.5) can be used by an ingress QNE to translate end-to-end NSLP messages into a form which is particularly suited to the local network, where it is expected that interior nodes will benefit from this. For example, where only per-class state or no state is stored by nodes (as for stateless or reduced-state operation in section 2.7). At the egress of the region, this local QSpec can be removed.

4.2 Refreshing a reservation

Since the NSIS protocol suite normally takes a soft-state approach to managing reservation state in QNEs, the state created by a RESERVE message must be periodically refreshed. Reservation state is deleted if no new RESERVE messages arrive before the expiration of the reservation lifetime. The Reservation lifetime is indicated in the RESERVE message when the reservation is made. Maintaining the reservation beyond this lifetime can be done by sending a RESERVE message with the same QSpec and Policy objects as the original message that created the reservation and by indicating that it is refreshing existing state. Note that a refreshing RESERVE should not increment the Reservation Sequence Number.

At the expiration of a "refresh timeout" period, each QNE independently examines its state and sends a refreshing RESERVE message to the next QNE peer where it is absorbed. This peer-to-peer refreshing (as opposed to the QNI initiating a refresh which travels all the way to the QNR) allows QNEs to choose refresh intervals as appropriate for their environment. For example, it is conceivable that refreshing intervals in the backbone, where reservations are relatively stable, are much larger than in an access network. The "refresh timeout" is calculated within the QNE and is not part of the protocol; however, it must be chosen to be compatible with the reservation lifetime, and an assessment of the reliability of message delivery.

The details of timer management and timer changes (slew handling and so on) are left for future versions of this document.

If a RESPONSE has been received to acknowledge that reservation state has been installed, then an abbreviated form of refreshing RESERVE message ('summary refresh') can be sent which references the

reservation using the Reservation Sequence Number, rather than including the full reservation specification. Note that this functionality requires an explicit acknowledgment of state installation to ensure that the RSN reference will be understood. It is up to a QNE that receives a ResponseRequest to decide whether it wants to accept summary refreshes and provide this explicit acknowledgment. For example, reduced state QNEs that cannot support summary refreshes would not send this acknowledgement.

It is currently an open point which parts of the RESERVE message are reused by the summary refresh. A summary refresh containing only the RSN seems to be the minimal case of a broad spectrum of varying amounts of data that we send in an update. Future versions of this document will attempt to identify some objects as 'needing refresh'.

When a QNE receives a partial refresh message it compares the RSN against that of the currently installed reservation. If it matches then the state is refreshed. If the RSN in the message is less than that of the currently installed state (i.e. it refers to 'old' state) then the message is discarded (possibly the messages got reordered between the nodes). If the RSN in the message is greater than that of the currently installed state then an error MUST be signalled back. It means that the peer QNE believes that state is installed when it is not. If not informed it would continue to attempt to refresh the non-existent state. A partial refresh containing either an unknown session identifier or flow identifier MUST also be responded to with an error message (this is likely to occur following a rerouting event).

4.3 Sending a response

A QNE sending a RESERVE message may also request a RESPONSE message to be sent back to it. To do this it includes a ResponseRequest object with a RII object in it.

When a node processes a received RESERVE message it should examine it to see if it contains a local ResponseRequest object. If it does, then a RESPONSE message is generated, and the contents of this object are copied into it. The local ResponseRequest object is removed from the RESERVE message being sent out.

At the QNR, the processing of local ResponseRequest objects is carried out normally (this may happen before this QNE has determined that it is the QNR). If the RESERVE message received by the QNR contained a ResponseRequest object, then a RESPONSE message is created with the contents of the ResponseRequest object copied into it.

On receiving a RESPONSE message a QNE should check the contents of the ResponseRequest object echoed back to see if it contains an identifier supplied by it (the RII). If it does, then it should inspect the contents of the message and send it no further. This should always be the case for local ResponseRequest objects. If one of these is received with an incorrect identifier, then the RESPONSE must be discarded.

For other QNR responses, the QNE may inspect the message and then must pass it back to the NTLP to be sent on.

When a RESPONSE message reaches the edge of a stateless domain, the egress QNE will map/interchange the control information used by the "end to end" and "local" scope QoS model types. The generated LCI information will be encapsulated into the RESPONSE message. By using the stored SII of the ingress QNE the RESPONSE message will be sent to the particular ingress QNE node.

Stateless and reduced state interior QNEs are not using the RESPONSE message.

When the RESPONSE message is received by the ingress QNE node, the LCI information is used by the "local" scope QoS model. Afterwards, the LCI information is removed from the RESPONSE message, which is sent towards the QNI.

4.4 Performing a query

In order to perform a query along a path, a QNE constructs a QUERY message. The message must contain a ResponseRequest object in the same manner as described above to allow it to match any received RESPONSE messages back to the original QUERY. The QUERY message may contain general QoS NSLP query elements, as well as objects specific to a given QoS model. The message is then passed to the NTLP to be passed along the path.

A QNE (including the QNR) receiving a QUERY message should inspect it and manipulate the general query objects and QoS model specific query objects as required. It then passes it to the NTLP for further forwarding unless it knows it is the QNR.

At the QNR, a RESPONSE message is generated. Into this is copied the ResponseRequest object, and other general and QoS model specific information from the QUERY message. It is then passed to the NTLP to be forwarded peer-to-peer back along the path.

Each QNE receiving the RESPONSE message should inspect the ResponseRequest object to see if it contains an RII supplied by it. If it does not then it simply passes the message back to the NTLP unless it is a local RESPONSE. If it does, it uses the RII to match the RESPONSE back to the original QUERY, and performs any appropriate action based on the result of the query.

4.5 Tearing down a reservation

Although the use of soft state means that it is not necessary to explicitly tear down an old reservation, it is RECOMMENDED that QNIs send a teardown request as soon as a reservation is no longer needed. A teardown deletes reservation state and travels towards the QNR from its point of initiation. A teardown message may be initiated either by an application in a QNI or by a QNE along the route as the result of a state timeout or service preemption. Once initiated, a teardown message must be forwarded QNE peer - to - QNE peer without delay.

A QNE can remove a reservation by building a RESERVE message with the 'tear' indicator set to inform the next-hop QNE to remove the reservation state that it may hold. The tearing RESERVE is passed to the NTLP to be forwarded along the NEs up to the next-hop QNE. It is currently an open issue whether the tearing RESERVE will automatically tear down state in each NE. The next-hop QNE either tears down the corresponding reservation and passes on the tearing RESERVE, or, if it already has torn down the reservation, discards the message.

In addition, the QNE should construct a NOTIFY message and attempt to send it back to the QNE that requested the reservation originally. The NOTIFY message will inform the other QNE that a reservation has been removed. On receipt of such a NOTIFY message a QNE should

determine an appropriate course of action. This may include removing its own reservation state, and passing the NOTIFY message back further along the path.

The attempt to send a NOTIFY message may be abandoned if the NTLP is not able to route the message along the reverse-path (e.g. because it has not stored the necessary state). In case of a stateless or reduced state domain, the ingress QNE of the domain will be notified by the egress QNE, which has kept track of its SII. The ingress QNE can then send the NOTIFY message further upstream. It can also initiate a scoped tearing RESERVE message towards the egress QNE.

5. IANA section

This section provides guidance to the Internet Assigned Numbers Authority (IANA) regarding registration of values related to the QoS-NSLP, in accordance with BCP 26 [12].

Future versions of this draft will identify name spaces in QoS-NSLP that require registration and contain recommendations for registration policies.

6. Requirements for the NSIS Transport Layer Protocol (NTLP)

For the moment this section will merely describe what we assume and/or request to be available from the NTLP. This section will later be updated to describe the eventual interface when NTLP work gets finalized.

6.1 Support for stateless operation

Stateless or reduced state QoS-NSLP operation makes the most sense when (some nodes of) the underlying NTLP is (are) able to operate in a stateless way as well. Such nodes should not worry about keeping reverse state, message fragmentation and reassembly (at the NTLP), congestion control or security associations. A stateless or reduced state QNE will be able to inform the underlying NTLP of this situation. We rely on the NTLP design to allow for a mode of operation that can take advantage of this information.

6.2 Support for Source Identification Information (SII)

There are several circumstances where it is necessary for a QNE to

identify the adjacent QNE peer, which is the source of a signaling application message; for example, it may be to apply the policy that "state can only be modified by messages from the node that created it".

We rely on the NTLP to provide this functionality. By default, all outgoing QoS-NSLP messages are tagged like this at the NTLP layer, and this is propagated to the next QNE, where it can be used as an opaque identifier for the state associated with the message; we call this the Source Identification Information (SII). The SII is logically similar to the RSVP_HOP object of [4]; however, any IP (and possibly higher level) addressing information is not interpreted in the QoS-NSLP. Indeed, the intermediate NTLP nodes could enforce topology hiding by masking the content of the SII (provided this is done in a stable way).

Keeping track of the SII of a certain reservation also provides a means for the QoS-NSLP to detect route changes. When a QNE receives a RESERVE referring to existing state but with a different SII, it knows that its upstream peer has changed. It can then use the old SII to send initiate a teardown along the old section of the path. This functionality would require the NTLP to be able to route based on the SII. We would like this functionality to be available as a service from the NTLP.

6.3 Last node detection

There are situations in which a QNE needs to determine whether it is the last QNE on the data path (QNR), e.g. to construct and send a RESPONSE message.

A number of conditions may result in a QNE determining that it is the QNR:

- the QNE may be the flow destination
- the QNE have some other prior knowledge that it should act as the QNR
- the QNE may be the last NSIS-capable node on the path
- the QNE may be the last NSIS-capable node on the path supporting the QoS NSLP

Of these four conditions, the last two can only be detected by the NTLP. We rely on the NTLP to inform the QoS-NSLP about these cases by providing a trigger to the QoS-NSLP when it determines that it is the last NE on the path, which supports the QoS-NSLP. It requires the NTLP to have an error message indicating that no more NSLPs of a particular type are available on the path.

6.4 Re-routing detection

This trigger is provided when the NTLP detects that the route taken by a flow (which the QoS-NSLP has issued signaling messages for) has changed.

6.5 Performance requirements

The QoS-NSLP will generate messages with a range of performance requirements for the NTLP. These requirements may result from a

prioritization at the QoS-NSLP (section 7.3.4) or from the responsiveness expected by certain applications supported by the QoS-NSLP.

The NTLP design should be able to ensure that performance for one class of messages was not degraded by aggregation with other classes of messages. The different classes of performance requirements will be listed in future versions of this document.

7. Open issues

7.1 Refinements of this version

Some topics raised in this document would benefit from more detailed discussion. These include:
- description of the operation under re-routing conditions
- description of the modification operation
- description of the operation under error conditions
- detailed discussion of message timer
- detailed discussion of control information, more particularly message scoping
- detailed discussion on the impact of a tearing RESERVE on state teardown at the NTLP

7.2 Content for next (-02) version

In addition to refining the discussion of topics currently described in this document, we intend the next version of this document to discuss the following issues.

7.2.1 Sender/Receiver-initiated operation

A sender-initiated reservation is made when the QNI for that flow is located upstream of the QNR with respect to the data path of the flow that is being signaled for. A receiver-initiated reservation is then made when the QNI is located downstream of the QNR with respect to the data path of the flow that is being signaled for. Note however, that the actual QoS-NSLP entities that initiate these signaling procedures can be anywhere along the data path, not just at the endpoints.

There are signaling scenarios in which the receiver-initiated approach is more advantageous, while in others the sender-initiated

approach is required. QoS-NSLP stateless operation, for example, is only possible when the sender-initiated approach is applied.

The next version of this draft will describe the use of sender-initiated and receiver-initiated reservations in the protocol. Note that the solution of this issue will also impact the way bi-directional reservations are supported (see Section 7.3.3).

7.2.2 Message formats

This version of the draft describes the functionality provided by the QoS-NSLP messages. Encoding this functionality into a message format is left for the next version.

7.2.3Message flows

This version of the draft briefly describes basic protocol operation. Future versions of this draft will include message flow diagrams for informative purposes (potentially in an appendix).

7.3 Content for future (-0x) versions

7.3.1 Aggregation

Future versions of this document will describe the use of aggregation

to reduce the signaling overhead (message bundling) and the routing state (similar to [13]).

7.3.2 Tunnel management

Future versions of this document will describe the use of QoS-NSLP over tunnels and the associated tunnel management.

7.3.3 Bi-directional/proxy operation

A future version of this draft will discuss the use of bi-directional reservations, i.e. combining the reservations for a pair of coupled flows going in opposite directions. The main difficulty here is that the two flows, although between the same end points, may traverse different paths across the Internet.

Two proposals for tackling this are:

- generate both a sender initiated and a receiver-initiated reservation at the QNI (section 7.2.1), and allow them to be bundled together by the NTLP (the bundle can be split if the paths diverge).
- generate a sender-initiated reservation that includes a request for the QNR to generate a sender-initiated reservation for the flow in the other direction.

Both methods make some assumption about the flow routing. The first method requires that both flows pass through the same QNI. The second assumes that the QNR for one direction is on the path of the flow in the other direction.

A future version will also include discussion on the operation of QoS-NSLP with (path-coupled) proxies.

7.3.4 Priority and preemption

The QoS-NSLP will generate messages with different priority. Prioritization at the level of the QoS-NSLP includes reservation priority and message priority.

Reservation priority enables some reservations to be setup even if resources would normally not be available (e.g. for emergency services). This requires a priority indication in the message. Note that such an indication may be restricted to the QoS model scope, although every QoS model is expected to need this functionality.

Whether resources are freed by means of pre-emption (the act of tearing down an existing reservation to free resource for a new one) or whether high-priority resources should be pre-provisioned is an implementation issue for the Resource Management Function.

Message priority allows QoS-NSLP messages to be processed and forwarded in a different order than in which they arrived. This may be needed to ensure responsiveness to reservation requests or modifications (e.g. a reservation caused by a mobility event of an in-service session may need to be handled faster than a query or a new reservation request). This kind of prioritization should then be supported in the QoS-NSLP message set and may pose requirements on the NTLP (section 6.5).

Future versions of this document will describe the support and use of prioritization for the QoS-NSLP.

7.3.5 Network Address Translation (NAT)

Since the QoS-NSLP is used for requesting resources for data flows, the messages inevitably involve IP addresses and, possibly, port numbers. However, the addresses/ports of the data flow can be modified by Network Address Translators (NATs) along the path. It is hoped that some (or all) of this problem can be pushed down into the NTLP, so that only generic NSIS-awareness is required at the NAT, rather than requiring specific support for QoS-NSLP (as described in section 5.3 of [3]). Future versions of this document may contain additional discussion on this issue.

7.3.6 Security components

7.3.7 Mobility

A future version of this draft will provide details on mobility support in QoS-NSLP, following the requirements in [2], [14]. A number of issues are associated with mobility support, such as localizing the new reservation to the new segment of the path, removing reservation state along the old segment of the path, recognizing a RESERVE message for an already established reservation although the IP address of the mobile node changed, possible concealment of QoS-NSLP messages due to IP-in-IP tunneling utilized in mobile IP, etc. Some of these issues can be solved by employing a session ID that is independent of the IP address, in addition to the

flow ID (this has security implications, see [15]), and by supporting the SII and reservation sequence numbers. Some mobility support will be provided by NTLP, such as detection of route changes. More details are discussed in [16].

8. Security Considerations

The security of the NSLP is provided by a combination of security functions at the NTLP and NSLP. Subsequent versions of this draft will need to contain a specification of the NSLP security features, and how these interact with those at the NTLP.

Some consideration of the problems can be found in [17][15][8]

9. Change History

10. Appendix A: A strawman QSpec template

This appendix describes a strawman QSpec template. The QSpec template could include objects and fields for conveying the following information:

- QSpec ID: This would allow IANA reservation of well known QSpec parameter combinations
- Traffic Envelope and traffic conformance (similar to RSVP's TSpec)
- Excess treatment: what happens to non-conforming traffic of a certain reservation (may be dropped but could be remarked as well)
- Offered guarantees: this may be delay, jitter, loss or throughput, both qualitatively (low delay) and quantitatively (delay < 100ms, optionally even with quantiles
  Note that the specification may support ranges of acceptable values
- Service schedule: for when is the service requested: this would allow a RESERVE/COMMIT functionality
- Reliability: what percentage of the time do you expect to get the offered guarantee (this will allow e.g. a local resource

management function to map the traffic to an APS protected link)

- Monitoring requirements: what information do you require about the reservation. This is related to the AdSpec functionality and may contain parameters or sub object to be used in QUERY.

Note that Flow identification information is also needed but that this is not assumed to be part of the QSpec template but rather available over the API between NTLP and QoS-NSLP.

It is not the intention that all QoS models support all fields and sub objects defined here. The definition of a QoS model entails a selection of one or more of these fields and/or sub objects. For instance, one QoS model could only allow QSpec ID and DSCP to be specified.

References

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

2 M. Brunner, Ed., "Requirements for QoS signaling protocols." draft-ietf-nsis-req-09.txt, August 2003.

3 R. Hancock, Ed., "Next Steps in Signaling: Framework", draft-ietf-nsis-fw-03.txt (work in progress), June 2003.

4 Braden, B., Zhang, L., Berson, S., Herzog, S. and S. Jamin, "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification", RFC 2205, September 1997.

5 Braden, B., Clark, D. and S. Shenker, "Integrated Services in the Internet Architecture: an Overview", RFC 1633, June 1994.

6 Wroclawski, J., "The Use of RSVP with IETF Integrated Services", RFC 2210, September 1997.

7 Tschofenig, H., "NSIS Authentication, Authorization and Accounting Issues", draft-tschofenig-nsis-aaa-issues-01 (work in progress), March 2003.

8 Tschofenig, H., Buechli, M., Van den Bosch, S. and H. Schulzrinne, "QoS NSLP Authorization Issues", draft-tschofenig-nsis-qos-authz-issues-00 (work in progress), June 2003.

9 S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss, "An Architecture for Differentiated Services", RFC 2475, December 1998.

10 Metro Ethernet Forum, "Ethernet Services Model," letter ballot document, August 2003.

11 Westberg, L., "Resource Management in Diffserv (RMD) Framework", draft-westberg-rmd-framework-04 (work in progress), September 2003.

12 Alvestrand, H. and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 2434, October 1998.

13 Baker, F., Iturralde, C., Le Faucheur, F. and B. Davie, "Aggregation of RSVP for IPv4 and IPv6 Reservations", RFC 3175, September 2001.

14 H. Chaskar, "Requirements of a QoS solution for Mobile IP," RFC 3583, Sept. 2003.

15 H. Tschofenig, H. Schulzrinne, et al., "Security implications of the session identifier" Internet Draft, June 2003.

16 X. Fu, H. Schulzrinne, H. Tschofenig "Mobility Support in NSIS", Internet Draft, June 2003.

17 H. Tschofenig and D. Kroeselberg, "Security Threats for NSIS", draft-ietf-nsis-threats-02.txt (work in progress), June 2003.

Acknowledgments

This section will contain acknowledgments.

Contributors

This draft combines work from three individual drafts. The following authors from these drafts also contributed to this document: Robert Hancock (Siemens/Roke Manor Research), Hannes Tschofenig and Cornelia Kappler (Siemens AG), Lars Westberg and Attila Bader (Ericsson) and Maarten Buechli (Dante) and Eric Waegeman (Alcatel).

Contact information

Georgios Karagiannis
University of Twente
P.O. BOX 217
7500 AE Enschede
The Netherlands
email: karagian@cs.utwente.nl

Andrew McDonald
Roke Manor Research
Old Salisbury Lane
Romsey
Hampshire
SO51 0ZN
United Kingdom
email: andrew.mcdonald@roke.co.uk

Sven Van den Bosch
Alcatel
Francis Wellesplein 1
B-2018 Antwerpen
Belgium
email: sven.van_den_bosch@alcatel.be

Full Copyright Statement