Network Working Group
Internet-Draft
Expires: December 22, 2003

A. McDonald
R. Hancock
Siemens/Roke Manor Research
H. Tschofenig
C. Kappler
Siemens AG
June 23, 2003

A Quality of Service NSLP for NSIS
<draft-mcdonald-nsis-qos-nslp-00.txt>

Status of this Memo

Copyright Notice

Abstract

This draft describes a protocol to be used for signaling QoS reservations in the Internet. It is compatible with the framework and requirements for such signaling protocols developed within NSIS; in conjunction with the NSIS Transport solution, it provides functionality comparable to RSVP: it is independent of the details of QoS specification, and adds support for a greater variety of reservation models, but is simplified by the elimination of support for multicast flows.

This draft includes a model of reservation operation and a description of the individual protocol mechanisms, and discusses interactions of the reservation protocol with other protocols and mechanisms. It also includes an outline functional specification and example message flows.

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 [1]. This QoS-NSLP is part of a larger suite of signaling protocols, whose structure is outlined in [2]; 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 [3], and uses soft-state peer-peer refresh messages as the primary state management mechanism. However, there is no backwards compatibility at the protocol level, although interworking would be possible in some circumstances. QoS-NSLP extends the set of reservation mechanisms to meet the requirements of [1], in particular support of sender or receiver initiated reservations, as well as a type of bidirectional reservation. Note that 'sender' and 'receiver' initiation refers to the direction of reservation messages relative to the data flow; the actual signaling entities can be anywhere along the data path, not just at the endpoints. On the other hand, there is no support for IP multicast.

QoS-NSLP does not mandate any specific 'QoS Model', i.e. a language for QoS objects or any architecture for provisioning it within a network or any particular node; this is similar to (but stronger than) the decoupling between RSVP and the IntServ architecture [4]. It should be able to carry QoS objects of various different types; the specification of Integrated Services for use with RSVP given in [5] could form the basis of one QoS model.

1.2 Model of Operation

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 [3]. The model is shown in Figure 1.

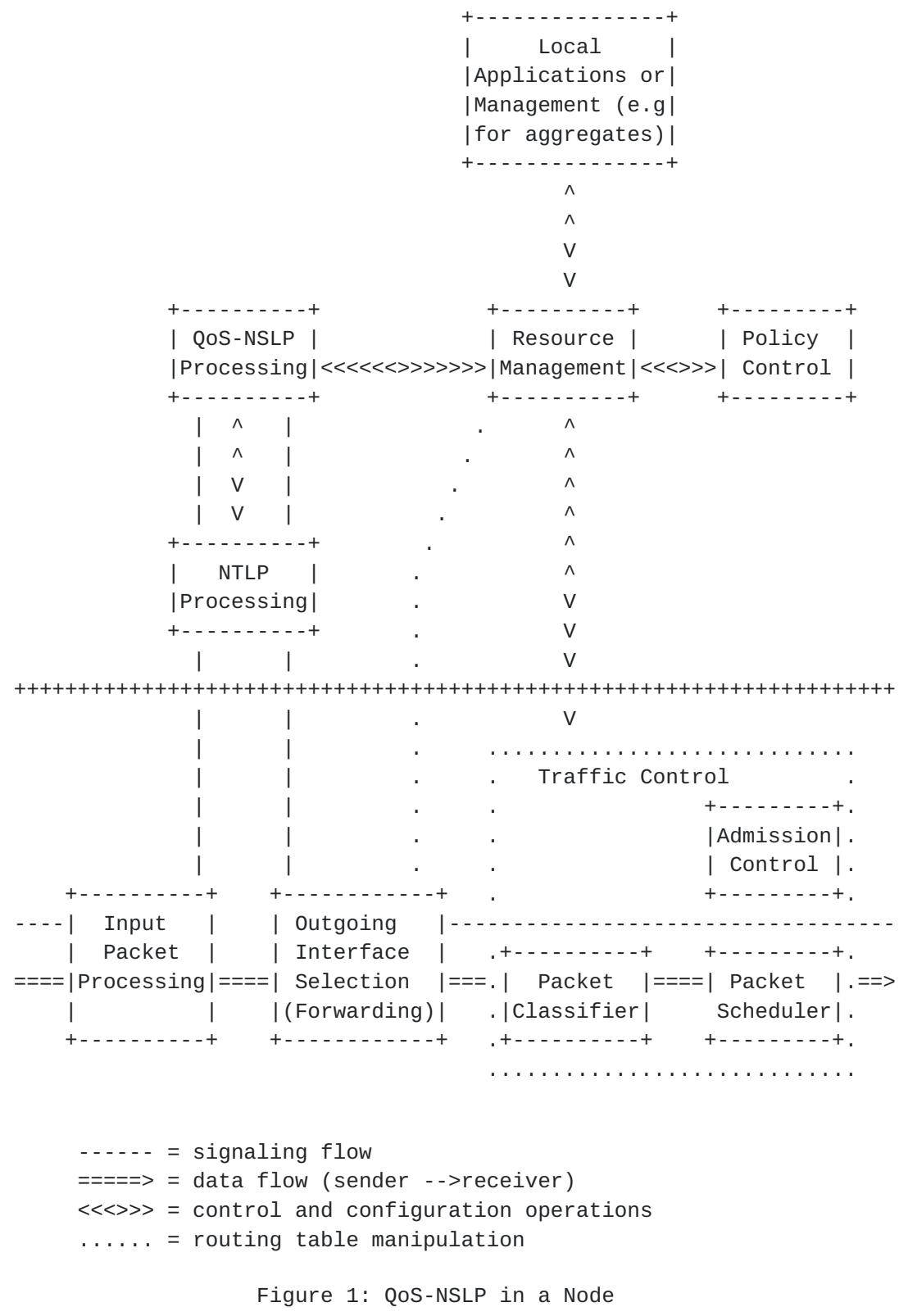


Figure 1: QoS-NSLP in a Node

The main features of the model are as follows:

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. a multimedia application), 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 [13] and [17]. 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; however, this is not visible to the protocol operation.

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 3.1.

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 the figure 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 [2] applies to this draft. In addition, the following terms are used:

- QNE - an NSIS Entity (NE) which supports the QoS-NSLP.
- QNI - a QoS NSLP node acting as an NSIS Initiator (NI), the first node in the sequence of QNEs that issues a reservation request.
- QNR - a QoS NSLP node acting as an NSIS Responder (NR), the last node in the sequence of QNEs that receives a reservation request.
- QNF - a QoS NSLP node acting as an NSIS Forwarder (NF).

In the document, where the phrase "message source" is used it generally refers to the adjacent QNE. Where it means one of the endpoints of the signalling session, this is highlighted by using the terms QNI or QNR.

2. Protocol Mechanisms

This section describes the conceptual building blocks of the QoS-NSLP, in order to explain the overall structure of the protocol. The message set of QoS-NSLP is deliberately designed to be simple, to ease future analysis of routing and mobility interactions and other extensions: there are just 5 messages, namely a RESERVE/RESPONSE pair (idempotent messages controlling all aspects of state management) and 3 stateless informational messages to handle queries and notifications. The default mode of operation delegates all message

routing responsibility to the NTLP.

2.1 State Management

The QoS-NSLP uses one message, the RESERVE message, for reservation state management at QNEs. The RESERVE message is idempotent, i.e. any given message has the same effect however many times it is repeated; whether a RESERVE is installing new state or refreshing, modifying or tearing down already established state is determined independently at each QNE, depending on the existence of related state at that QNE. The RESERVE messages being sent at different points along the path are conceptually independent so far as the protocol is concerned, each being sent by one QNE and received by its next peer.

The RESERVE message is used to support a similar function to the RSVP reserve, reservation refresh and teardown as all of these manipulate what is conceptually the same reservation state. Which of these three functions the message causes depends on the QoS objects and flow/session identifier it carries. For example, reservation state is created in QNEs by a RESERVE message with unknown flow/session ID and the topmost QoS object describing non-zero resources.

Reservation state is soft state and needs to be refreshed periodically. Refresh (and modify) is achieved by sending a RESERVE message with known flow/session identifier. Finally, a teardown is really a special case of a modify message, with the modification being that resources should be zero.

Any RESERVE may be replied to by a 'local' RESPONSE message (i.e. one sent directly back by the next QNE along the path); this happens when the reservation installed does not match the reservation requested (generally, when an error condition occurs), or if the QNE sending the RESERVE explicitly requests it. Any QNE can also create a RESERVE message tagged so it causes messages to be sent further along the path towards the QNR and RESPONSE messages sent all the way back.

2.2 Informational Messages

The QoS-NSLP uses three messages which do not affect the resource management state at all (are 'impotent') at QNEs. These are the QUERY, QUERY-REPLY and NOTIFY messages.

QUERY messages are used to support a similar function to the AdSpec Class in RSVP. A QUERY can be used to determine what kind of QoS services are available along the path, as well as any service-specific attributes and the amount of QoS resources available.

The QUERY message payloads can be extended or modified as the message is propagated; when it reaches the end of the path it is changed into a QUERY-REPLY message which carries the same data but in the reverse direction. QUERY messages can also be used purely between adjacent peers.

NOTIFY messages are not sent in direct response to other signaling messages. They do not directly modify network state, or require a signaling message in response. They can be sent asynchronously by any node in the path and are commonly used to indicate that a change has occurred in the network. Examples include: change of a reservation caused by a network event (e.g. where rerouting means that a reservation can no longer be supported), or when a reservation is no longer being honoured due to pre-emption.

QUERY, QUERY-REPLY and NOTIFY messages are forwarded along the path (they may go upstream or downstream, provided that the NTLP has the necessary path state).

2.3 Initiation and Termination

2.3.1 Initiating QoS Signaling

The simplest case is clearly where the sender initiates the reservation. It can do this simply by sending its first RESERVE message. The sender then becomes also the QoS-NSLP Initiator (QNI) for the reservation.

A node further along the data path may also carry out this role (for example, because the sender does not not support the QoS NSLP itself, or because QoS signaling is already done by the sender using some other mechanism.) This node must know the attributes of the flow, including the flow-id, and the QoS properties required. How this is done is not considered part of this protocol specification; options include the node taking part in the application's own signaling, or translating another QoS signaling protocol being used between itself and the sender. So far as the rest of the reservation processing is concerned at other nodes, this case is indistinguishable from the first, except that the QNI address (if visible in any messages) does not match the source address in the flow-id.

A receiver initiated QoS NSLP signaling session is essentially identical, except that it requires the QNI to be able to construct the flow identifier (including knowing the encapsulation or other marking used for data packets) and that NTLP reverse-path state should exist. Both of these may require coordination with the sender, which has to be done with end-to-end signaling outside the scope of the QoS-NSLP. If end-to-end reverse path state is created in the

NTLP, this should be signaled to the QoS-NSLP at the QNR to allow it to begin the signaling application.

2.3.2 Finding the Terminating Node

For a sender initiated case where the receiver supports the QoS NSLP, there is no difficulty in finding the terminating node. The receiver knows that it is the endpoint for the data flow, and so also knows to terminate the QoS signaling, and becomes the QNR for the session.

Similarly, a node on the path may decide to act as the QNR and terminate the signaling. This is needed in particular where the QoS NSLP signaling reaches a QNE after which there are no more nodes supporting the QoS NSLP before the receiver. This situation cannot be detected by the NSLP on its own, and requires support from the NTLP. This is easily detected and reported by the local NTLP if there are no further NSIS-aware nodes at all; if there are further NSIS-aware nodes but which don't support the QoS-NSLP, this condition must be forwarded back to the last QNE as an error at the NTLP level.

In the example shown in Figure 2, NQ is the 'last' node supporting the QoS-NSLP. N1 can simply forward the RESERVE message (treating it as an NTLP message with an 'unknown' payload). At N2 the next peer discovery fails, and it generates an error message which is passed in the reverse direction.

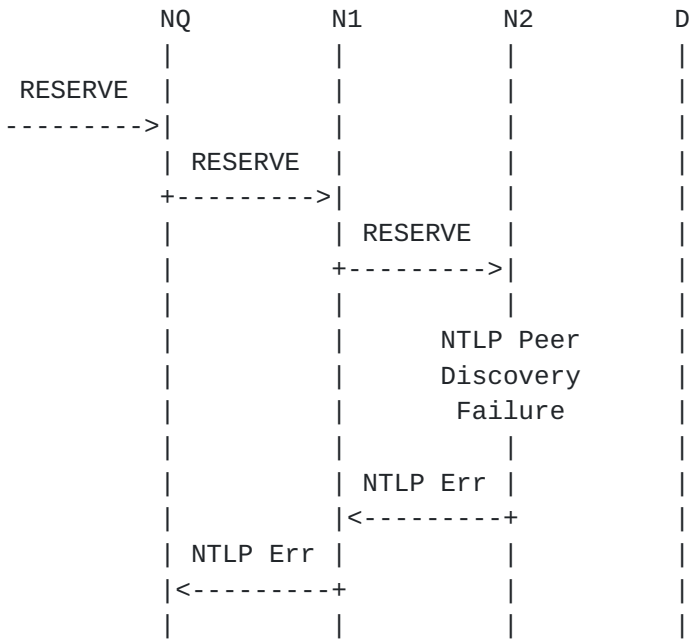


Figure 2: Terminating a Sender Initiated Session

In the corresponding receiver initiated scenario, the signaling will

reach a node with no 'previous hop'. If this is the sender then the signaling can be terminated immediately, the sender becoming the QNR. In other cases the node has to determine whether it is indeed the last node on the path, or simply lacking the relevant NTLP path state.

This situation may occur, for example, at the time of the initial reservation when reverse path state is available at the QNI (data flow receiver) because of a constrained topology (e.g. only one ingress router), but not further upstream. Therefore, when the receiver initiates a reservation, it has a valid route to the upstream peer, but this peer has no valid state to forward the message further. If the upstream peer wishes to propagate the signaling further, it has to generate a notification towards the QNI requesting an end to end (application layer) exchange which can trigger building the reverse path state. The QNI would also have to tag RESERVE messages to indicate that this installation of reverse path forwarding state has already been attempted (or is not possible), in which case the reservation could in general be propagated except in constrained network topologies.

2.4 State Source and Generation Identification

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 [3]; 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).

Reservation messages contain a sequence number allocated by the peer QNE which is the source of the message. Sequence numbers are allocated in ascending order and unique within the context of a given SII. Sequence numbers are used for a number of functions, including identification of the newest RESERVE message, simplifying the identification of RESERVE messages as being refreshes, and tying a RESPONSE back to a RESERVE message. Note that the sequence number may be different at different locations along the path; this is because

intermediate nodes may generate reservation updates autonomously while the reservation is in place.

2.5 QoS-NSLP Message Routing

2.5.1 Default Operation

The default mode of operation for the QoS-NSLP is that signaling messages are transported along the path currently taken by a data flow by the NTLP. The necessary information is held in the flow-id, which is used as flow-routing information by the NTLP. All that the QoS-NSLP needs to do is provide the flow-id and a tag to indicate whether the message should be sent upstream (towards the flow sender) or downstream (towards the flow receiver), and the NTLP will deliver it to the next QoS-NSLP-aware node or return an error condition (such as "no next node" or "no routing state available to reach next node"). Whether another message is sent beyond the next node is controlled by the QoS-NSLP at that node, and depends on the message in question.

2.5.2 Explicit Routing

There are circumstances where it is necessary to address state in an explicitly identified NSIS node, which isn't necessarily on the path associated with a particular (active or inactive) flow. Examples are tearing down state on the 'old' path after a route change, and sending a message which was only meaningful in the context of a previous message (such as a refresh which doesn't include the full reservation data).

The first case could be handled by depending on timeout of soft-state; this would lead to temporary waste of resource in those areas. The second could be handled by returning an error in the QoS-NSLP, since the message receiver can determine that the message sender is 'unknown' by checking the SII. An alternative approach, an extension to support explicit routing for such messages, is described in Appendix A.

2.5.3 Skip-Stop Routing

There may be cases where it is desirable to route a message directly to a QoS-NSLP peer (e.g. to send a confirmation directly to the initiator), rather than sending it via using NTLP. The motivation for doing this is to reduce the burden on the NTLP (especially NTLP-only intermediate nodes), both in message processing and the necessity to maintain reverse-path routing state for the associated flow.

The basic technique for doing this is that a subset of QNEs on the

path directly store some QNE IP address within the QoS-NSLP. These can send directly to their peer. However, this has to be done in a way which does not invalidate the services provided by the NTLP (error handling, security, congestion control, routing reactions and so on). There are several possible solutions to this problem; one is outlined in Appendix B.

2.6 Resource Description Objects

Each of the QoS-NSLP message types can carry QoS descriptions in Resource Description Objects (RDOs). The definitions of the possible RDO contents for each message together form a QoS Model. Multiple QoS Models can be defined for use with the QoS-NSLP; the general requirements on a QoS Model are discussed in Section 3.1, but which one is used does not affect the general operation of the QoS-NSLP itself.

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 kind of nesting of QoS Models by 'stacking' more than one RDO within a message. The mechanism for the RESERVE message is as follows; similar rules can be defined for the other QoS-NSLP message types. The QNI places an end-to-end RDO the RESERVE message. However, each QNF may add a further so-called local RDO in the RESERVE message that it forwards. The RDO on top of the stack is the one currently valid, and the others need not be parsed. This procedure allows mapping the QoS described in the end-to-end RDO onto local QoS paradigms. For example, a bandwidth and application type in the end-to-end RDO may be mapped onto a discrete set of available bandwidths and a particular traffic class in the local RDO. If a QNE does not understand the topmost (local) RDO, it generates an error RESPONSE which is sent backwards along the data path. The error message must be terminated and recovered by the last QNE which added it.

In general, local RDOs must contain scoping information such as "valid only in a particular domain A". In this case, all edge nodes of domain A must be QNEs and be configured to pop the topmost RDO on egress; they can then add a new one valid only on the inter-domain link, or fall back to the next lower one which may or may not be the end-to-end RDO. This is conceptually similar to the Aggregation Region concept of [9], where one option is that Deaggregators (which correspond to the last QNE in scope) are pre-configured to act in these roles. In general, it is up to Network Management to make QNFs knowledgeable about what scope they are in and what QoS Models they should use.

3. External Interactions

3.1 QoS Models

A QoS Model gathers together the set of ways of describing packet flow behavior (as RDOs), and describes how these RDOs can be used in particular QoS-NSLP transactions. The scope of a QoS model encompasses the tspec/rspec/adspec concepts of RSVP [3]; however the way they describe QoS can be different, and the QoS-NSLP specification leaves the details of how the RDOs in each message should be processed to the QoS Model description. For example, basic RSVP/IntServ [5] functionality would include the fact that RESERVE messages should contain a 'rspec' and QUERY messages an 'rspec' and optionally an 'adspec', and would also describe the parametrisation of those contents.

More generalised resource reservation signaling patterns are expected to be encoded in the QoS Model description, without changing the basic operation of the QoS-NSLP itself. For example, it may indicate that a partial reservation, successful only at some nodes, would be acceptable. In order to reduce the number of reservation message exchanges, the RDO in a RESERVE might also include a resource range, containing an upper and a lower bound. QNEs would attempt to reserve the highest amount of resources below the maximum and update the amount accordingly; the QoS Model would also define the possible contents of the RESPONSE if the maximum could not be reserved.

Another possibility is of advanced reservations, including a 2-stage reserve/commit mechanism. The intention here is that a QNI can request a firm guarantee that resources will be available at some future time, without actually needing the physical resources to be set aside at the time the request is made. Since this process does actually involve manipulating state in the other QNEs it can be described within a QoS Model as a particular type of reservation, where the initial RDO is processed by admission and policy control but not the packet scheduler. 'Activation' of the reservation takes place simply by sending another RESERVE with an updated RDO.

It is not clear whether there is a need for a standardised 'default' QoS Model that can be interpreted by all QNEs. This would naturally be used for the end-to-end RDOs which are handled by the QNI and QNR. Some applications may wish (and are able to) to detail a token bucket, peak rate etc, whereas others just provide the information "this is a delay-sensitive flow of such average bandwidth". The default QoS Model could provide fields for all these values, however not all of them need to be filled in by the application. It would seem that such a default model would be most useful at the network 'edge', and quite possibly inappropriate for core network or

inter-provider use. In addition, there would seem to be a need for some common elements to all QoS Models (such as the value 'No QoS' to be used when tearing down a reservation or reporting complete resource unavailability).

A local QoS Model in contrast is specific to the QoS mechanism used in a particular scoping region. For example, its RDOs may include full RSVP flowspec information. For some popular QoS mechanisms, an official standard may exist for the local QoS Model. However, private ('enterprise') QoS Models could also exist where all QNEs in a scoping region have to be configured specifically to understand it. It is also possible for a QNI to include definitions from a local QoS Model as well as a default one, for example in the case of a mobile node describing desired L2 properties on the air interface.

An IANA registry of well-known QoS Models would be required.

It is currently an open question whether policy related information, such as accounting and charging information or authorization tokens should be included as part of the QoS Model, or whether it should be defined separately. There is clearly coupling between the two types of information, since it is only possible to make a meaningful policy decision if the QoS description is understood; however, the way the policy information is given may itself be independent of that. This question depends on how AAA issues in general are to be handled in the QoS-NSLP [13] [17]

3.2 Implications for NTLP Functionality

The QoS-NSLP requires that a some particular features be available in the NTLP messages or be provided as triggers by the NTLP module (either locally or in a distributed fashion).

The features currently suggested in this document include:

- Last node detection: This trigger is provided by the NTLP to the QoS-NSLP when it determines that this 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. (See also Section 2.3.2).

- Rerouting 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.

- Reverse path state established: This trigger is provided at the QNI for receiver initiated reservations when the NTLP detects that a path-establishing message has been received from the flow

sender. This trigger is needed only when the path-establishing message is not itself a QoS-NSLP message (such as a QUERY). (See also Section 2.3.1.)

- Source Identification Information (SII): This information needs to be provided at the NTLP level to identify the QNE which a particular message came from. It needs to be stable across rerouting events which do not change QoS-NSLP adjacencies (but might, for example, change outgoing interfaces). (See also Section 2.4.)

- Explicit Routing: This facility might be needed to send messages explicitly to a known QNE, in which case the NTLP has to provide topologically correct routing information in a form which can be cached by the QoS-NSLP. (See also Appendix A.)

- Stateless Forwarding: This facility might be provided to enable the NTLP to operate in a mode where reverse path state is not kept in some region, analogous to the processing in [9]. Doing this robustly appears to be possible with a pair of additional flags at the NTLP level. (See the description in Appendix B.)

- Path length determination: In order to count the number of QoS-NSLP-aware/unaware hops, support from the NTLP is needed to provide the IP hop count between adjacent QNEs. This could be provided by the NTLP by default, if such a facility is felt to be generally useful.

Although these requirements are identified here in the context of a Quality of Service NSLP, some of them may also be applicable to other NSLP types. Also, it should be noted that the QoS-NSLP makes fairly cautious assumptions about the level of the transport service provided by the NTLP, for example regarding re-ordering protection across multiple hops or over route changes. A more sophisticated NTLP might allow us to remove QoS-NSLP functionality such as sequence numbering, or tagging state with source identification. However, it isn't clear how generic across other NSLPs such functionality can be made.

4. Outline Functional Specification

4.1 QoS NSLP Messages

The QoS-NSLP defines five message types:

- RESERVE

- RESPONSE

- QUERY
- QUERY-REPLY
- NOTIFY

The first two are idempotent (the resultant state in traffic control is the same however many times an identical message is repeated), and the remainder do not change traffic control state at all (although NOTIFY might indirectly trigger a state-changing action in the receiver).

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 destination, which is one of 'next hop', or 'previous hop'. On reception, the NTLP provides the same information to the QoS-NSLP along with the source identification. Possible additional features are described in Appendix A and Appendix B.

The rest of the message is opaque to the NTLP. QoS-NSLP messages have a common header providing protocol version, message type, and flags associated with protocol extensibility issues (rules about ignoring or rejecting unknown messages); these details are not described here.

4.1.1 RESERVE Message

RESERVE messages are idempotent. They manipulate reservation state in QNEs by creating, refreshing, modifying and deleting it. Each RESERVE message triggers three actions:

- Install, refresh, modify or delete reservation state
- Possibly send a RESPONSE
- Possibly create a modified message (e.g. adding or removing objects) and pass it to the NTLP

The RESERVE message format can be summarised as follows:

```
<Reserve Message> ::= <Common Header>
                      <RESERVATION_SEQUENCE_NUMBER>
                      <RDO_LIST>
                      <RESERVATION_LIFETIME>
                      [<RESPONSE_REQUESTED>]
                      [<POLICY_INFORMATION>]
```

4.1.1.1 Reserving resources

To reserve resources, the QNI sends a RESERVE message. A RESERVE message with unknown flow/session ID creates reservation state at each QNE, which is used by the local resource management function to grant and configure resources as described in Section 1.2. A RESERVE message carries at least one RDO describing the QoS desired and a lifetime for it. QNEs may add local RDOs, providing nesting of QoS information as in Section 2.6.

A RESERVE message carries a sequence number, which is increased for each new RESERVE message pertinent to the same flow/session ID. This allows for identifying the latest state that is being requested. Optionally, it may contain an object to control whether a local and/or end-to-end RESPONSE should be generated. A RESERVE message may additionally carry other local or global objects, such as accounting and charging information and so on.

4.1.1.2 Refreshing

In general, the NSIS protocol suite takes a soft state approach to managing reservation state in NEs. Note that although both NTLP and QoS-NSLP have soft state, it is managed independently to avoid interlayer coupling.

For NSLP, the state is created by the RESERVE message and must be periodically refreshed. Reservation state is deleted if no new RESERVE messages arrive before the expiration of the "reservation lifetime" interval specified as part of the reservation state. State can also be deleted by explicit teardown described in Section 4.1.1.3. At the expiration of a "refresh timeout" period, each QNE independently scans its state and sends a corresponding 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 in 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 that is advertised, and an assessment of the reliability of message delivery. The details of timer management and timer changes (slew handling and so on) should be similar to those of RSVP [3].

As well as a 'traditional' soft-refresh (simply repeating the original messages), a summary refresh can be sent if a RESPONSE has been received for the reservation. This is achieved by sending a

RESERVE message only containing a pointer to the corresponding reservation (flow/session ID and Sequence number of the RESPONSE), not the reservation information itself in order to speed up processing [7].

Reservations can be refreshed "as is". If reservation state needs to be modified, a RESERVE message containing explicit QoS information (new RDOs) is sent, with a strictly higher RESERVATION_SEQUENCE_NUMBER. If, for example, because of route changes, a QNE receives a refreshing RESERVE message, containing only a pointer which it does not understand (e.g. from an unknown source), it replies with an error RESPONSE message back to the originating (peer) QNE. This QNE replies with a full updated RESERVE message including corresponding RDOs. This way, failures can be repaired quickly locally. This is another advantage of peer-to-peer refreshing.

Regarding RFC 2961-style bundling of Refresh messages, there are two design options. Either a QNE may bundle refresh messages before handing them down to NTLP, or NTLP is solely responsible for bundling. The advantage of the former is there is only one NSLP header per bundle. On the other hand, NTLP is best placed to do bundling efficiently because it knows more about path properties (e.g. MTU, packetisation, latency) and whether messages should even follow the same path at the NTLP level. We therefore opt for the latter. NTLP may decide to bundle this bundle with refresh messages from other NSLPs and / or to synchronize and piggyback its own refreshes with QoS-NSLP refresh messages in order to save overhead. Details of this however are clearly out of scope of this document.

4.1.1.3 Teardown

A RESERVE message with 'zero' RDO removes reservation state of the corresponding flow/session ID immediately. Although because of soft state it is not necessary to explicitly tear down an old reservation, we recommend 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 an QNI or by a QNF 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.

4.1.2 RESPONSE Message

RESPONSE messages are any messages sent in reply to a RESERVE message. They are idempotent. Their semantics include error reports, simple acknowledgements, and so on. RESPONSE messages may be sent

with a scope of a single QoS-NSLP 'hop' or be sent further along the path towards a QNE which explicitly requested it. By default, they are sent within the NTLP and may require reverse-routing state to exist (in the case of a sender initiated reservation).

The RESPONSE message format can be summarized as:

```
<Response Message> ::= <Common Header>
                       <RESERVATION_SEQUENCE_NUMBER>
                       <CONFIRMATION_OR_ERROR_TYPE>
                       [<RDO_LIST>]
                       [<RESPONSE_REQUESTED>]
                       [<POLICY_INFORMATION>]
```

4.1.2.1 Error

An error RESPONSE message indicates a reservation has failed. It includes the sequence number of the failed RESERVE message. In addition, the SII of the QNE where the RESERVE failed is provided by the NTLP. It is interpreted first by the QNE which sent the RESERVE, which must either attempt corrective action, or tear the reservation down and propagate the error condition further backwards.

4.1.2.2 Confirmation

To request a confirmation for a reservation request, a QNE includes in the RESERVE message a confirmation-request object containing an identifier supplied by the QNE. If a confirmation-request has already been added by another QNE a second one need not be added, since this QNE will see the RESPONSE anyway.

The RESPONSE (whether an error or success indication) echoes back the confirmation-request object. If a RESPONSE contains a confirmation-request object not added by this QNE then it MUST forward the RESPONSE, until it reaches the QNE which provided the original request (matched by the identifier).

A confirmation must be issued when the reservation installed does not match the reservation requested, i.e. when - within a range possibly provided in the RESERVE - less resources have been reserved starting from a particular QNF_i towards the QNR. This allows the QNI to issue a new RESERVE in order to adapt resources up to QNF_i.

The confirm RESPONSE message may be sent by the QNR simply confirming the RESERVE was successful as requested, or it may be issued by a QNF to confirm it modified a Reservation (partial reservation or - within the bandwidth range - decreased reservation). The Confirm message

contains the sequence number of the RESERVE it is in response to.

4.1.3 QUERY and QUERY-REPLY Messages

QUERY messages do not change the NSLP state at any of the nodes that process them (though, like any NSIS message, they may cause NTLP path state to be created or modified). When the QUERY reaches the end of the path, the message is changed from a QUERY to a QUERY-REPLY and then sent back in the opposite direction.

One application of the QUERY message is similar to the use of an AdSpec object in a PATH message for RSVP. The QUERY message can carry parts specific to particular QoS Models, similar to the Guaranteed Service and Controlled-Load Service Fragments of the RSVP ADSPEC message [5]. These will be specified in documents defining particular QoS Models.

These may be used to determine what resources are present along the path (e.g. estimating the path bandwidth or minimum path latency). However, they do not guarantee the availability of resources for a subsequent RESERVE request.

The QUERY message format is as follows (the QUERY-REPLY is essentially identical):

```
<Query Message> ::= <Common Header>
                    <QUERY_IDENTIFIER>
                    <QOS_NSLP_COUNT>
                    [<qos model query list>]

<qos model query list> ::= <empty> |
                    <qos model query list> <QOS_MODEL_QUERY>
```

The QUERY_IDENTIFIER is an unstructured numerical identifier for the query, used for matching responses. The value of the identifier is otherwise not significant.

The QOS_NSLP_COUNT is initially zero, and is incremented by one at each QNE on the path. It counts the QoS-NSLP aware nodes along the path, and can be used (along with the total path length) to derive the number of non-QoS-NSLP hops, a generalisation of the way RSVP counts the number of non-IntServ hops. It can also accumulate the IP hop length of the path, if this information is provided by the NTLP.

A QOS_MODEL_QUERY object can be used to query information that is specific to the type of QoS Model being used.

4.1.4 NOTIFY Message

NOTIFY messages only provide information to an NE, they do not cause a change in state directly themselves.

They main difference between RESPONSE messages and NOTIFY messages is that RESPONSE messages are sent on receipt of a RESERVE message, whereas NOTIFY messages can be sent asynchronously, and in either direction relative to the RESERVE.

The message may contain information relating to particular QoS models. These can be used to provide more information when the notification is due to a change in the reservation.

The NOTIFY message format is as follows:

```
<Notify Message> ::= <Common Header> <NOTIFICATION_CODE>
                     [<qos model notification list>]

<qos model notification list> ::= <empty> |
                  <qos model notification list> <QOS_MODEL_NOTIFY>
```

The QOS_MODEL_NOTIFY contains any QoS Model specific information that needs to be carried as part of the QUERY, e.g. the reservation now being used after a reservation change.

4.2 Rerouting and Local Repair

The detection of rerouting can take place in multiple ways.

It can be done at the NTLP (including by the NTLP interacting with routing protocols or by path length monitoring and so on, as described in [18]). Rerouting detected by the NTLP may then be delivered as trigger information to the QoS-NSLP (at one or more locations along the signaling path).

Rerouting can also be detected at the QoS-NSLP itself, if a RESERVE arrives refreshing existing state but coming over a new interface; or, if a RESERVE claims to refresh state that does not exist at all. (Similar facts can be deduced from mis-delivered RESPONSE messages.) In either case, we assume for now that any necessary reverse-routing state already exists in the NTLP; actions to stimulate this state being set up will be considered in a later version of this document.

In either case, the QoS-NSLP needs to filter the event to avoid flapping a reservation in synchronization with flapping a route, and then carry out local repair actions to ensure that the reservation is set up on the new path and if possible torn down on the old. A

high-level outline of the necessary processing is as follows:

1. The QNE detecting the re-route issues a new RESERVE with a doubly-incremented sequence number, including a request to receive a confirmation from further down the path. The RESERVE should be given the same SII as previous reservations for the same flow, to avoid disrupting reservations in the case where the next QNE on the path is actually the same.

1a. At QNEs on the new part of the path, the RESERVE installs new reservation state, and is immediately propagated further to the QNR.

1b. At the QNE where the old and new paths merge, the QOS-NSLP should generate a RESPONSE which is returned to the QNE initiating the route change. The state already existing at that QNE is re-labelled with the SII of the new QNE which requested it.

At this stage, the reservation is essentially installed on the new path. Further reservation messaging might take place to adjust the QoS parameters along the path if the new path has very different characteristics from the old; this takes place (if at all) as a background activity.

2. If explicit routing (Appendix A) is supported, the QNE detecting the route change can now issue another new RESERVE with a null QoS request (i.e. a teardown) and lower sequence number than used in (1), and explicitly route it along the old path.

2a. A QNE not on the new path will tear down the reservation state and forward it further if it can.

2b. At the QNE where the old and new paths merge, the teardown will be ignored, either because it has a lower sequence number than the newly installed reservation, or because it is attempting to remove state installed under a different SII.

Note that, with the exception of the explicit routing, this method of re-routing support is purely an implementation issue at the QNE detecting the route change, it does not require any other rerouting-specific protocol features.

4.3 Mobility and Multihoming

There are several circumstances where it is desirable to associate together two reservations with different flow-ids (typically, different addresses) but which are conceptually for the same packet stream. One case is mobility with a change of address; a related example is of multihoming, where a node sets up reservations for its

flows on a new interface in preparation for handing them over. A third case is call waiting. In all cases, the wish is for resources to be shared (singly-booked) over the network region where the flows share a path. This is comparable to a restricted use of the Shared-Explicit filter style of [3], in a non-multicast context.

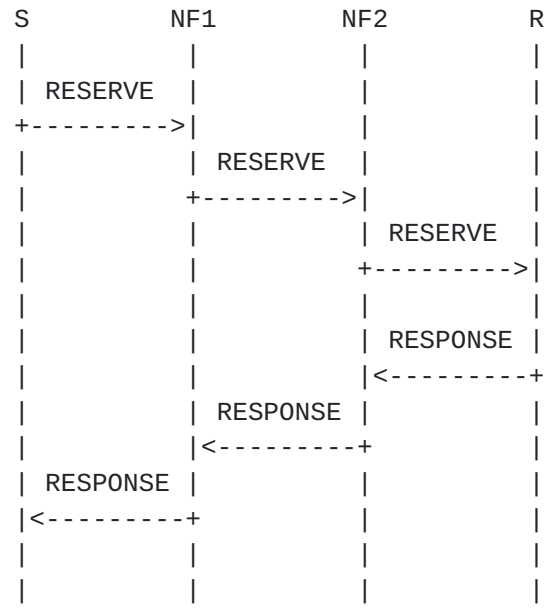
The NSIS protocol suite provides the session id for this purpose: resources are shared based on having a common session id, even though their flow ids are different. A later version of this draft will discuss the modifications to the protocol to support this functionality, mainly in terms of what identifiers are used to match state and messages. It should be noted that secure use of the session id is non-trivial; this problem is discussed in [16].

5. Example Message Flows

A number of message flows (at NSLP level) are shown here as examples of the QoS NSLP signaling process.

Section 5.1 below shows a sender initiated NSLP signaling flow; the RESPONSE messages have been generated from the QNR because of the RESPONSE REQUESTED object inserted by the QNI.

5.1 Basic Sender/Receiver Initiated Example



The receiver initiated case is essentially identical, with the difference that the leftmost node in Section 5.1 is now 'R' (the data receiver) and the rightmost node is now 'S' (the data sender). In order to perform the reservation, reverse path state needs to be installed. Some discussion of how this can be done is given in

Section 2.3.1.

5.2 Reservation Collision Example

This example shows an exchange, with a 'reservation collision' causing the new reservation to fail. A reservation collision occurs when the two endpoints of a data flow (or signaling proxies on the data path but not at the flow endpoints) fail to agree on who should make the reservation for a flow, leading to a QNE seeing RESERVE messages from both directions.

In this example (Figure 8), we assume that the QNE detecting the condition has adopted a 'sender wins' policy: the sender initiated reservation is accepted (or maintained), and the receiver initiated one is rejected (or torn down). It isn't clear whether this policy is reasonable; however, it does appear that some sort of default policy must be standardised. A slight increase in sophistication would be to include information in a RESERVE about whether it should be preferred over reservations in the opposite direction. Clearly, there are also interactions with AAA issues here.

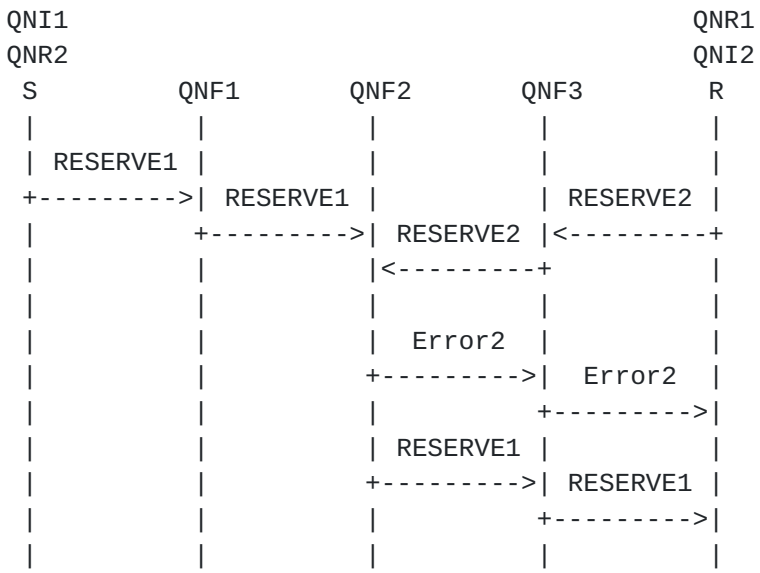


Figure 8: Reservation Collision

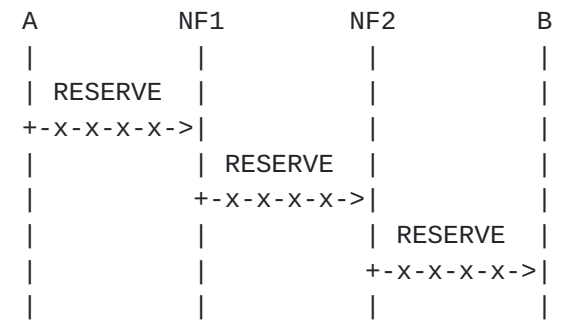
QNF2 detects the collision on receipt of RESERVE2. The reservation created by RESERVE2 is rejected with the appropriate error condition as RESERVE1 is propagated onwards towards the receiver.

Any RESPONSE message processing for RESERVE1 is performed in the normal way.

5.3 Bidirectional Reservation Example

A bidirectional reservation is actually a sender initiated reservation (for an outbound flow) and a receiver initiated reservation (for the corresponding inbound flow of a bi-directional flow) combined together and issued by a single QNI. It is implemented through the use of NTLP bundling, with the NSLP providing the two reservations together to the NTLP as an indication that they should if possible be delivered together.

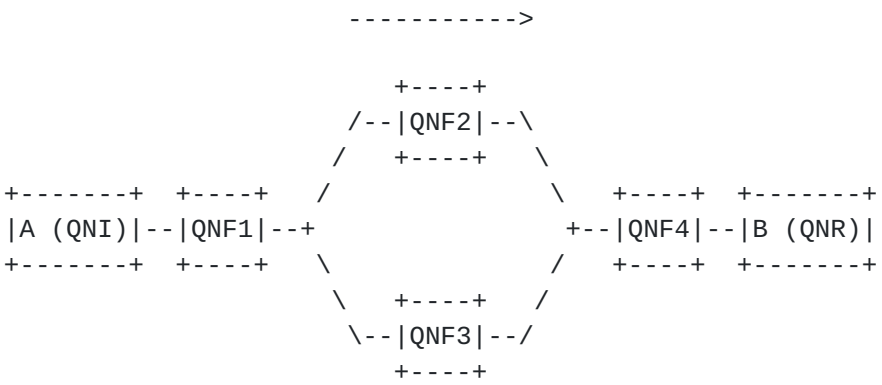
The diagram below shows a bidirectional reservation. RESPONSE messages can be provided in the normal manner.

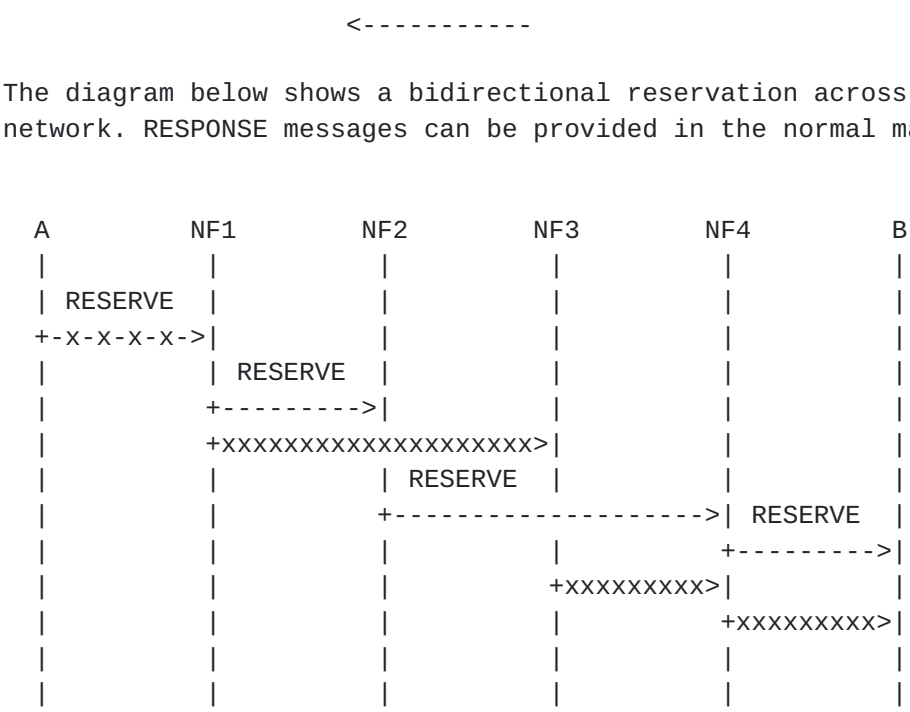


---> = Reservation for A->B direction
xxx> = Reservation for B->A direction
-x-> = Bundled reservation (A->B and B->A)

The NTLP path state for reverse path routing from A to B (for the B->A flow) must be set up before the reservation can be performed, otherwise the receiver initiated half of the reservation will fail.

If the routing is asymmetric then the reservation will be split into two where the paths diverge. The diagram below shows a network within an asymmetric route for a bidirectional flow between A and B.





---> = Reservation for A->B direction
xxx> = Reservation for B->A direction
-x-> = Bundled reservation (A->B and B->A)

At QNF1 the 'next hop' for the A to B flow is different to the 'previous hop' for the B to A flow, so the reservation bundle must be split. It then operates as two separate reservations - one sender initiated, the other receiver initiated.

Even if messages do not arrive bundled (as at QNF4 in the example), the QoS-NSLP is allowed to merge the state for the flow internally and use it to issue bundled refresh messages to neighboring QNEs.

5.4 Tunnels and Aggregation

QoS NSLP as defined above also allows dynamically aggregating reservations such that core network nodes are alleviated from keeping per-microflow state. Reservations for aggregate flows can be triggered by individual (per-microflow) reservations, or can be set up independently. The general advantages in terms of resource management, particularly in the context of DiffServ networks, are described in section 4.2.1 of [8].

Management must have configured QNEs, typically at the boundary of a domain, to act as aggregating and deaggregating QNEs. The configuration depends on the aggregation method being used. The two choices are:

- Tunnel-based, where traffic is encapsulated in an IP tunnel (using GRE, IP-in-IP tunnel, IPsec, and so on). The aggregating QNE initiates the tunnel and chooses the endpoint as one of the deaggregating QNEs at the domain edge.

- DiffServ-based, where normal routing is used within the domain, but the aggregating QNE marks the aggregated traffic with an appropriate DSCP.

5.4.1 Sender Initiated Tunnel Aggregation

Here we describe a sender-initiated example for aggregate reservation set-up. With receiver-initiated aggregate reservations other issues may arise which need to be investigated in future versions of this draft, if it is felt that receiver orientation is useful for reservations in this context.

Apart from the receiver/sender distinction, the method chosen here is conceptually similar to that of [6]. The tunnel is used as a single virtual link, in that the 'end-to-end' NSIS signaling for the data flows is tunneled between the same endpoints so as to be invisible to the routers between them, and a second signaling session is applied purely between the tunnel endpoints.

The aggregating QNE (which will be the QNI for the aggregate reservation) does the following:

- it tunnels 'forwards-path' NSIS messages referring to flows within the aggregate by adding an IP header addressing the deaggregating QNE. The aggregator also decapsulates 'reverse-path' NSIS messages tunneled from the deaggregator.

- whether or not these signaling messages are part of the aggregate reservation or use a distinct tunnel encapsulation is up to management; using a distinct encapsulation prevents the signaling and traffic having to share resources.

- it initiates a RESERVE towards the deaggregator describing resources to be reserved for the aggregate flow. The algorithm used to determine aggregate resources is a management and policy issue. They may e.g. exactly fit the resources needed currently, or - avoiding frequent reconfigurations - be based on an estimate of resources needed now and in the near future. Note that the aggregator will be able to see both directions of QoS-NSLP messages for all the flows within the aggregate, in particular RESERVE messages, and these can be used as the input to the calculation for the aggregate resource requirement. Therefore,

this technique is applicable regardless of whether the end-to-end signaling is sender or receiver initiated (or indeed a mixture of the two).

- depending on how aggregate flow and resources are described in the RESERVE, and depending on the local QoS mechanism, it tunnels data packets by appending an IP header fitting the aggregate flow ID and addressing the deaggregating QNE.

The deaggregating QNE (aka QNR for the aggregate reservation) does the following:

- it terminates the RESERVE for the aggregate and is the QNR for it.

- it receives and decapsulates the tunneled data packets.

- it receives and decapsulates tunneled QoS NSLP signaling packets and processes them just as any other signaling packet received in an ordinary fashion. If these are forwards path messages, the NTLP should be able to use them to install reverse routing state back up the virtual link (in exactly the same way it can install reverse routing state back up a real link), given that the other end of the link is also a QNE.

QNFs on the data path between aggregating and deaggregating QNEs do not know they are processing an aggregate reservation. Therefore they don't need any special information, nor do they perform special packet treatment. Indeed, it is clear from the above descriptions that aggregations can be nested by just re-applying the above steps.

5.4.2 Receiver Initiated DiffServ Aggregation

An alternative aggregation method is based on the DiffServ architecture rather than relying on the use of tunnels. It has some similarities with the description of RSVP aggregation in [9]; in particular, we assume a 'simple' routing infrastructure where a shortest path that includes two points (the aggregator and deaggregator) is also the shortest path between those two points themselves. We also assume that all DiffServ marked traffic within the region will be included in aggregate reservations.

The following description depends on the skip-stop routing extension described in Appendix B. The combination of the stateless mode of the NTLP and the storage of the ingress QNE identifier in the QoS-NSLP messages corresponds to the use of the special RSVP-E2E-IGNORE protocol number in [9]: state is (eventually) not stored in the interior of the network, and the egress QNE learns the address of the ingress QNE from the signaling messages. The method works as follows:

- End-to-end QoS-NSLP messages for the individual flows are sent using skip-stop routing. They must not be interpreted by the QoS-NSLP within the network; this could be done by giving them a different QoS Model from that used in the network interior.

- The egress QNE can determine which flows come from which ingress QNE, and also track the reservation requests for those flows. This allows it to build up a picture of what aggregate reservations are needed between it and each ingress QNE. The algorithm that assigns flows to aggregates (DSCPs) is the responsibility of network management.

- The egress QNE requests the ingress QNE for an aggregate to set up reverse path state in the network. The request can be sent as a special NOTIFY message sent outside the NTLP; the state can be set up with a QUERY sent from ingress to egress via the NTLP. (This is where the assumption that simple shortest path routing is being used.)

- Once the reverse path state is available, the egress QNE sets up a receiver initiated reservation for the DSCP along that path. Note that the classifier will be purely the DSCP, on the assumption that on any interface, all the traffic for any DSCP will be covered by some reservation (it doesn't matter which). This reservation can be maintained and modified by the egress QNE as it tracks the flow ingress point (and possibly DSCP) as derived from the end-to-end signaling.

5.5 Layered Reservations

The combination of end-to-end and local RDOs together with reservation aggregation as described in the last section can be used to perform layered reservations in the style described in [19] and [20]. Particularly, in [20], a framework (RMD) is proposed for resource management and reservation in DiffServ networks. The RMD proposes using two protocols, a Per Hop Reservation (PHR) protocol, and a Per Domain Reservation (PDR) protocol.

According to [20], "The PHR protocol is used within a DiffServ domain on a per-hop basis to augment the DiffServ Per Hop Behavior (PHB) with resource reservation. It is implemented in all nodes in a DiffServ domain. On the other hand, the PDR protocol manages the resource reservation per DiffServ domain, relying on the PHR resource reservation status in all nodes. The PDR is only implemented at the boundary of a domain (at the edge nodes)."

In [19], this framework is complemented by an end-to-end signaling

protocol, which transports per-flow QoS information to the edge nodes. This end-to-end protocol is invisible inside the DiffServ domain.

Tasks of PDR particularly are mapping of end-to-end signaled parameters on domain specific RMD parameters, specifically DSCPs. This information must be transmitted from the ingress to the egress QNF. Furthermore, [20] lists tasks such as admission control, resource reservation in edge nodes, congestion handling (refusing admission of new flows). However we believe that all of the latter are not protocol features but functionalities of the edge nodes (using information received via the protocols) with which we do not deal in this ID. They correspond to external interactions with the components shown in Figure 1.

There are currently two flavors of PHR. One flavor is reservation-based PHR. Here, the PHR protocol transports aggregate per-PHB resource requirements to each interior node. These nodes install corresponding reservation state. The other flavor is measurement-based PHB. Here, each interior node measures current load, and determines, based on these measurements, whether a new resource request arriving via PHB can be accommodated. The advantage of the latter is that interior nodes do not need to store reservation state. Furthermore, PHR issues congestion control notifications. As with PDR, we believe further PHR features such as per-interior node admission control etc. are functionalities of the interior nodes, independent of the protocol.

Following the discussion in Section 5.4 above, it is straightforward to implement the layered RMD signaling using QoS-NSLP. The edge nodes are (de)aggregating QNEs. They aggregate and tunnel the end-to-end (per-microflow) signaling. PDR signaling functionality is achieved by either stacking a local RDO onto end-to-end signaling messages, informing the deaggregating QNE about DSCP mapping, or by the aggregating QNE initiating an extra RESERVE towards the deaggregating QNE which is tunneled through the aggregate region. PHR signaling functionality is achieved by signaling for the aggregate initiated by the aggregating QNE. Reservation-based PHR signaling is equivalent to simply sending a RESERVE for each PHB, which installs reservation state at each QNF in the aggregation region. Measurement-based PHB always (also in [19] and [20]) depends on special configurations of the interior nodes - they are stateless and can measure their traffic load. Measurement-based PHB functionality can be realized by sending a QUERY (querying whether sufficient resources are available). For processing the QUERY, QNFs in this case do not consult their reservation-state database as they would normally, but perform traffic load measurements. However, from a protocol perspective, this is conformant message processing.

We do not discuss congestion handling in this version of the ID as it is still debated whether this functionality resides in NSLP or NTLP.

6. Open Issues

This section summarises some of the open issues that have arisen during the preparation of this Internet Draft. Needless to say, almost all of the proposals and assumptions made here can be questioned and alternatives proposed; we list here only the ones we have had most enjoyment and mental stimulation from discussing.

1. Do we need to have a standardised, well known, mandatory QoS Model?

2. Are the mechanisms for adapting to local QoS Models the most appropriate and useful? Do we need to include protocol support for discovering and agreeing these models?

3. Do we need explicit routing (Appendix A) or skip-stop routing (Appendix B), and should it be possible to extend the latter to multiple hierarchical levels?

4. How should message scoping be handled? Is it purely a matter of network management, or is some protocol support for it necessary or useful?

5. Is the messaging to set up reverse routing state (for the receiver initiated case) something that should be built into each application, or should there be out-of-NTLP QoS-NSLP messages to enable this?

6. Is 'sender wins' an appropriate default policy to handle the reservation collision problem?

7. Is it interesting for the QoS-NSLP to know the overall path length and how many nodes on it are QoS-NSLP aware?

8. How should the mobility/multihoming details look? In particular, how are session id and flow id matching rules modified?

9. Is it possible to send more than one RESPONSE for a RESERVE, e.g. to handle an error condition discovered after the original RESPONSE?

10. In particular, how should pre-emption within the network be signaled back towards the initiator of the reservation - by a modified RESPONSE or a NOTIFY?

11. Is there a notion of a QoS-NSLP proxy? Or are all QoS-NSLP nodes effectively proxies anyway?

12. Should we consider packet classifiers which are more (or less) granular than the flow id, and what effect does this have on the state matching rules (i.e. is the relevant matching really against packet classifier rather than flow id)? What should be done about overlapping packet classifiers in this case?

13. How are AAA issues really handled?

7. Security Considerations

To evaluate the security of the NSLP layer some assumptions regarding the security mechanism provided at the NTLP layer have to be taken into considerations.

To address the security threats described in Section 2.1, 2.3, 2.5, 2.6 of [15] it is assumed that an authentication and key exchange protocol is used to establish a security association between neighboring NTLP peers. Between neighboring administrative domains it is very likely that both peers are also NSLP nodes. By choosing an authentication and key exchange protocol which is resistant to denial of service attacks, man-in-the-middle attacks and provides strong authentication the described threats can be addressed. Details need to be analysed after choosing a specific authentication and key exchange protocol for the NTLP itself.

As a result the NTLP and therefore also NSLP messages can be authenticated, integrity, confidentiality and replay protected. Replay protection ensure that the threat described in Section 2.3 of [15] is prevented. Confidentiality protection prevents threats described in Section 2.2 of [15]. This is necessary when additional policy objects need to be exchanged or to protect the session identifier (or other payloads used for the same reason) as described in [16].

By fetching the authenticated identity used during the NTLP authentication it is possible to realize the two party authorization model described in Figure 1 of [17]. To realize either one of the two third party models shown in Figure 2 and Figure 3 of [17] additional security mechanisms are required at the NSLP to protect the authorization tokens and similar. This threat is also described in Section 2.4 of [15]. Non-repudiation seems to make sense only in the combination of authorization. The corresponding threat is described in Section 2.7 of [15].

Furthermore it might be necessary to protect NSLP message payloads in an end-to-middle, middle-to-middle or end-to-end fashion (an example for end-to-end protection is given in [13]). An association between non peer-to-peer protection and the above-described three party authorization is to stand to reason. A typical candidate for such a protection is CMS [12] or a modified Policy Object [10]. By modified we refer to enhanced functionality and possibly changed functionality.

Denial of service attacks (see Section 2.9 of [15]) are best prevented by separating the protocol functionality such as authentication and key exchange, signaling message delivery and discovery etc. This allows protocol functionalities from each protocol to be chosen in such a way that DoS attacks are prevented to the best possible extent. Some of these issues are therefore also applicable to the design of the NTLP - impacts can, however, also be seen at the NSLP. The authorization procedure itself is, to some extent, also vulnerable to DoS attacks since computing an authorization decision might require other entities (or even other networks) to be contacted. Specific authorization procedures need therefore be evaluated carefully against the vulnerability to introduce DoS attacks.

Security implications introduced by the session identifier are discussed in [16] but are applicable to this NSLP.

The security property of network topology hiding is controversial since to some extent it introduces security, however, on the other hand it makes debugging more difficult. QUERY messages, messages performing a record route, etc. are affected. It is left for future discussions whether this feature should be introduced. Based on previous work it seems that it is fairly simple to introduce network topology hiding (from a technical point of view).

The deployment threats addressed in Section 2.14 of [15] need to be addressed separately in the context of the [13] and [17] discussion.

Once certain mechanisms have been selected the issue of security parameter exchange and negotiation needs to be evaluated.

This section has to be re-evaluated once the NTLP design is finished and agreement exists on the security mechanisms.

8. Acknowledgements

This draft draws significant inspiration from RSVP (RFC2205).

The authors would like to express particular thanks to Henning

Schulzrinne.

This Internet Draft is built on previous discussions and inputs from Marcus Brunner, Jorge Cuellar, Jochen Eisl, Mehmet Ersue, Xiaoming Fu, Eleanor Hepworth, Holger Karl and Andreas Kassler.

References

[1] Brunner, M., "Requirements for Signaling Protocols", draft-ietf-nsis-req-08 (work in progress), June 2003.

[2] Hancock, R., "Next Steps in Signaling: Framework", draft-ietf-nsis-fw-02 (work in progress), March 2003.

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

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

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

[6] Terzis, A., Krawczyk, J., Wroclawski, J. and L. Zhang, "RSVP Operation Over IP Tunnels", RFC 2746, January 2000.

[7] Berger, L., Gan, D., Swallow, G., Pan, P., Tommasi, F. and S. Molendini, "RSVP Refresh Overhead Reduction Extensions", RFC 2961, April 2001.

[8] Bernet, Y., Ford, P., Yavatkar, R., Baker, F., Zhang, L., Speer, M., Braden, R., Davie, B., Wroclawski, J. and E. Felstaine, "A Framework for Integrated Services Operation over Diffserv Networks", RFC 2998, November 2000.

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

[10] Yadav, S., Yavatkar, R., Pabbati, R., Ford, P., Moore, T., Herzog, S. and R. Hess, "Identity Representation for RSVP", RFC 3182, October 2001.

[11] Awduche, D., Berger, L., Gan, D., Li, T., Srinivasan, V. and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels", RFC 3209, December 2001.

[12] Housley, R., "Cryptographic Message Syntax (CMS)", RFC 3369, August 2002.

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

[14] Tschofenig, H., "RSVP Security Properties", draft-ietf-nsis-rsvp-sec-properties-01 (work in progress), March 2003.

[15] Kroeselberg, D. and H. Tschofenig, "Security Threats for NSIS", draft-ietf-nsis-threats-01 (work in progress), January 2003.

[16] Tschofenig, H., Schulzrinne, H., Hancock, R., McDonald, A. and X. Fu, "Security Implications of the Session Identifier", draft-tschofenig-nsis-sid-00 (work in progress), June 2003.

[17] 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.

[18] Hancock, R., Hepworth, E. and A. McDonald, "Design Considerations for an NSIS Transport Layer Protocol", draft-mcdonald-nsis-ntlp-considerations-00 (work in progress), January 2003.

[19] Westberg, L., "A Proposal for RSVPv2-NSLP", draft-westberg-proposal-for-rsvpv2-nslp-00 (work in progress), April 2003.

[20] Westberg, L., "Resource Management in Diffserv (RMD) Framework", draft-westberg-rmd-framework-03 (work in progress), May 2003.

Authors' Addresses

Andrew McDonald
Siemens/Roke Manor Research
Old Salisbury Lane
Romsey, Hampshire SO51 0ZN
United Kingdom

EMail: andrew.mcdonald@roke.co.uk

Robert Hancock
Siemens/Roke Manor Research
Old Salisbury Lane
Romsey, Hampshire SO51 0ZN
United Kingdom

EMail: robert.hancock@roke.co.uk

Hannes Tschofenig
Siemens AG
Otto-Hahn-Ring 6
Munich 81739
Germany

EMail: hannes.tschofenig@siemens.com

Cornelia Kappler
Siemens AG
Siemensdamm 62
Berlin 13627
Germany

EMail: cornelia.kappler@siemens.com

Appendix A. Explicit Routing

There can be some cases where the QoS-NSLP interacts explicitly with the routing of messages by the NTLP. Examples are:

1. To tear down state in a node which is no longer on the path because of a mobility or rerouting event. Left to its own devices, the NTLP would send the teardown to the new node (which is presumably not what is wanted). The QoS-NSLP has to ensure that the message has to go to a specific physical node.

2. To send a message which depends on state previously established in the node, e.g. to send a message within a particular security association (see the 'next peer' problem of [14]), or to send a 'summary' refresh referring to an existing acknowledged reservation without including the full reservation data.

Both of these can be implemented by re-using the source identification information (SII) described in Section 2.4; the originator QoS-NSLP uses the SII provided in the messages in the reverse direction.

1. The tear message includes the SII; the NTLP is instructed to route the message based on the SII, rather than the flow identification. (This would include the NTLP unwinding topology hiding processing at intermediate nodes.) Once the message has left the originating node along the correct 'old' path, it can probably be routed normally by the NTLP.

2. The message is sent accompanied by the SII; the NTLP still routes the message on the flow identification, but as soon as it detects that the message would diverge from reaching the node given by the SII, it can generate a route change notification. (This might be done only at the receiving QoS-NSLP node.)

Using the SII in this way is somewhat analogous to the Explicit Routing Object (ERO) of [11]. It requires additional capabilities in the NTLP to make message routing SII-aware (as opposed to just transporting the SII as an identification tag). Note that some aspects of SII support (described in Section 4.2) may prevent it always being topologically correct for explicit reverse routing; in that case, it may be preferable to use an independent object for reverse routing - this can be seen as yet another case of the desirability of separation of routing and identifiers.

Appendix B. Skip-Stop Routing

This is an extension whereby 'interior' NSIS nodes are relieved of storing some per-flow state (e.g. reverse path routing state) and processing some message types, by allowing the routing of messages directly to a QoS-NSLP node (e.g. to send a confirmation directly to the initiator), rather than sending only using NTLP.

Difficulties of this are that the services provided by the NTLP (in terms of security, NAT traversal, and other message transport functions) are lost, because the NTLP only operates between adjacent on-path peers. Special consideration needs to be given to avoiding message loops. And the approach should be robust against the non-existence of a QoS-NSLP-aware downstream node - if no such node exists, lower layer errors (including the fact that no such node exists) cannot be forwarded upstream.

The following outlines one potential design supporting this functionality in a fairly robust way. It requires two additional NTLP-layer flags.

1. A edge node prepared to shield downstream nodes from processing/ state storage requirements inserts addressing information for itself in downstream messages sent via the NTLP, and additionally sets a 'stateless proposal' (SP) flag. The sending edge node must

be prepared to receive messages outside the NTLP (i.e. unsecured) at this address, and must also be able to route messages upstream itself (otherwise loops would form).

2. Intermediate nodes (including NSLP-unaware ones) can clear the SP flag if they wish (e.g. they are on an addressing or security boundary), but otherwise forward it. In the meantime, error messages can also be sent upstream.

3. A node wishing to act as the 'receiving edge' echoes the value of SP in the message it sends upstream, otherwise leaves it unset. If SP is set, the message may be sent directly to the sending edge node.

4. When the sending edge node receives messages with SP set, it sets a 'stateless requested' (SR) flag in downstream messages. Intermediate nodes can use this as a signal to flush per-flow routing state. Any reservation state is not deleted (typically, in circumstances where this technique is useful, only per-class reservation state is being stored anyway.) If the sending edge node stops receiving SP after some timeout, it must clear SR on the messages it sends.

5. An intermediate node that wishes to generate an upstream message - typically an error message - encapsulates this in a special payload and sends it downstream; it may also decide to clear SP. The receiving edge node can then send it back upstream.

In order not to violate assumptions about reliability and congestion management being managed by the NTLP, only a subset of QoS-NSLP messages can be sent 'out of band' in this way, namely RESPONSE messages (clocked by the rate of reservation messages) or vice versa, and notifications.

This functionality seems quite complex, but the state save seems non-trivial also. The consequent tradeoff should be carefully evaluated. Given that a significant amount of the complexity is caused by NTLP interactions (including the need to cope with error cases) it might be worth considering if this functionality should be built into the NTLP itself.

Full Copyright Statement

Acknowledgement

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