IETF Next Steps in Signaling
Internet-Draft
Intended status: Informational
Expires: August 13, 2010

C. Shen
H. Schulzrinne
Columbia U.
S. Lee
J. Bang
Samsung AIT

February 9, 2010

NSIS Operation Over IP Tunnels
draft-ietf-nsis-tunnel-08.txt

Abstract

NSIS QoS signaling enables applications to perform QoS reservation along a data flow path. When the data flow path contains IP tunnel segments, NSIS QoS signaling has no effect within those tunnel segments and the resulting QoS-untended tunnel segments could become the weakest QoS link which may invalidate the QoS efforts in the rest of the end-to-end path. The problem with NSIS signaling within the tunnel is caused by the tunnel encapsulation which masks packets' original IP header fields. Those original IP header fields are needed to intercept NSIS signaling messages and classify QoS data packets. This document defines a solution to this problem by mapping end-to-end QoS session requests to corresponding QoS sessions in the tunnel, thus extending the end-to-end QoS signaling into the IP tunnel segments.

This Internet-Draft will expire on August 13, 2010.

Copyright Notice

Table of Contents

1. Introduction

IP tunneling is a technique that allows a packet to be encapsulated and carried as payload within an IP packet. The resulting encapsulated packet is called an IP tunnel packet, and the packet being tunneled is called the original packet. In typical scenarios, IP tunneling is used to exert explicit forwarding path control (e.g., in Mobile IP [RFC3220]), facilitate the secure IP delivery architecture (e.g., in IPSEC [RFC2401]), and help packet routing in IP networks of different characteristics (e.g., between IPv6 and IPv4 networks [RFC4213]).

This document considers the situation when the packet being tunneled contains a Next Step In Signaling (NSIS) [RFC4080] message. NSIS is an IP network layer signaling architecture consisting of a Generic Internet Signaling Transport (GIST) [I-D.ietf-nsis-ntlp] sub-layer for signaling transport, and an NSIS Signaling Layer Protocol (NSLP) sub-layer customizable for different applications. We focus on the Quality of Service (QoS) NSLP [I-D.ietf-nsis-qos-nslp] which provides functionalities that extend those of the earlier RSVP [RFC2205] signaling. In this document the term "NSIS" and "NSIS QoS" are used interchangeably.

Without additional efforts, NSIS signaling does not work within IP tunneling segments of a signaling path. The reason is that tunnel encapsulation masks the original packet including its header and payload. However, information from the original packet is required both for NSIS peer node discovery and for QoS data flow packet classification. Without access to information from the original packet, an IP tunnel acts as an NSIS-unaware virtual link in the end-to-end NSIS signaling path.

This document defines a mechanism to extend end-to-end NSIS signaling for QoS reservation into IP tunnels. The NSIS-aware IP tunnel end-points that support this mechanism are called NSIS-tunnel-aware end-points. There are two main operation modes. On one hand, if the tunnel already has pre-configured QoS sessions, the NSIS-tunnel-aware end-points map end-to-end QoS signaling requests directly to existing tunnel sessions as long as there are enough tunnel session resources; on the other hand, if no pre-configured tunnel QoS sessions are available, the NSIS-tunnel-aware end-points dynamically initiate and maintain tunnel QoS sessions that are then associated with the corresponding end-to-end QoS sessions. Note that whether the tunnel pre-configures QoS sessions or not, and which pre-configured tunnel QoS sessions a particular end-to-end QoS signaling request should be mapped to are policy issues out of scope of this document.

The rest of this document is organized as follows. Section 2 defines

terminology. Section 3 presents the problem statement including common IP tunneling protocols and existing behavior of NSIS QoS signaling operating over IP tunnels. Section 4 introduces the design requirements and overall approach of our mechanism. More details about how NSIS QoS signaling operates with tunnels that use pre-configured QoS and dynamic QoS signaling are provided in Section 5 and Section 6. Section 7 describes a method to automatically discover whether a tunnel end-point node supports the NSIS-tunnel interoperation mechanism defined in this document. Section 8 discusses IANA considerations and Section 9 considers security.

2. Terminology

This document uses terminology defined in [RFC2473], [I-D.ietf-nsis-ntlp], and [I-D.ietf-nsis-qos-nslp]. In addition, the following terms are used:

Tunnel IP Header: The IP header prepended to the original packet during encapsulation. It specifies the tunnel end-points as source and destination.

Tunnel Specific Header: The header fields inserted by the encapsulation mechanism after the tunnel IP header and before the original packet. These headers may or may not exist depending on the specific tunnel mechanism used.

Tunnel Intermediate Node (Tmid): A node which resides in the middle of the forwarding path between the tunnel entry-point node and the tunnel exit-point node.

IP Tunnel: A tunnel configured as a virtual link between two IP nodes, on which the encapsulating protocol is IP.

Flow Identifier (Flow ID): The set of header fields which is used to identify a [Data] flow. For example, it may include flow sender and receiver addresses, protocol and port numbers.

End-to-end [QoS] Signaling: The signaling process that manipulates the QoS control information in the end-to-end path from the flow sender to the flow receiver. When the end-to-end flow path contains tunnel segments, this document uses end-to-end [QoS] signaling to refer specially to the [QoS] signaling outside the tunnel segments.

Tunnel [QoS] Signaling: The signaling process that manipulates the QoS control information in the path inside a tunnel, between the tunnel entry-point and the tunnel exit-point nodes.

[Adjacent] NSIS Peer: The next node along the signaling path, in the upstream or downstream direction, with which a NSIS node explicitly interacts.

NSIS-aware Node: A node that supports NSIS signaling.

NSIS-aware Tunnel End-point Node: A tunnel end-point node which is also an NSIS node.

NSIS-tunnel-aware [Tunnel] End-point Node: An NSIS-aware Tunnel End-point node which also supports the mechanism for NSIS operating over IP tunnels defined in this document.

3. Problem Statement

3.1. IP Tunneling Protocols

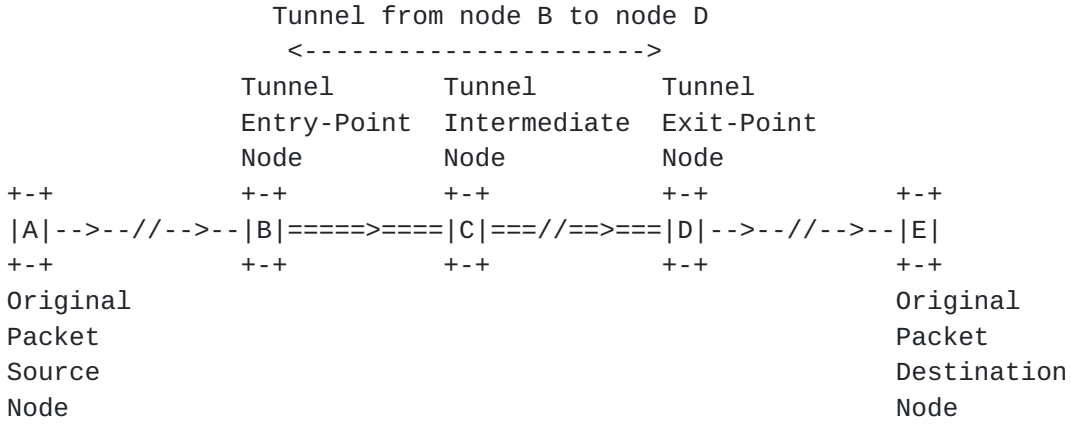


Figure 1: IP Tunnel

The following definition of IP tunneling is derived from [RFC2473] and adapted for both IPv4 and IPv6.

IP tunneling is a technique for establishing a "virtual link" between two IP nodes for transmitting data packets as payloads of IP packets (see Figure 1). From the point of view of the two nodes, this "virtual link", called an IP tunnel, appears as a point-to-point link on which IP acts like a link-layer protocol. The two IP nodes play specific roles. One node encapsulates original packets received from other nodes or from itself and forwards the resulting tunnel packets through the tunnel. The other node decapsulates the received tunnel packets and forwards the resulting original packets towards their

destinations, possibly itself. The encapsulating node is called the tunnel entry-point node (Tentry), and it is the source of the tunnel packets. The decapsulating node is called the tunnel exit-point node (Texit), and it is the destination of the tunnel packets.

An IP tunnel is a unidirectional mechanism - tunnel packet flow takes place in one direction between the IP tunnel entry-point and exit-point nodes (see Figure 1). Bi-directional tunneling is achieved by combining two unidirectional mechanisms, that is, configuring two tunnels, each in opposite direction to the other - the entry-point node of one tunnel is the exit-point node of the other tunnel.

Figure 2 illustrates the original packet and the resulting tunnel packet. In a tunnel packet, the original packet is encapsulated within the tunnel header. The tunnel header contains two components, the tunnel IP header and other tunnel specific headers. The tunnel IP header specifies tunnel entry-point node as IP source address and tunnel exit-point node as IP destination address, thus causing the tunnel packet to be routed inside the tunnel. The tunnel specific headers in between the tunnel IP header and the original packet in a tunnel packet are optional, depending on the tunneling protocol in use.

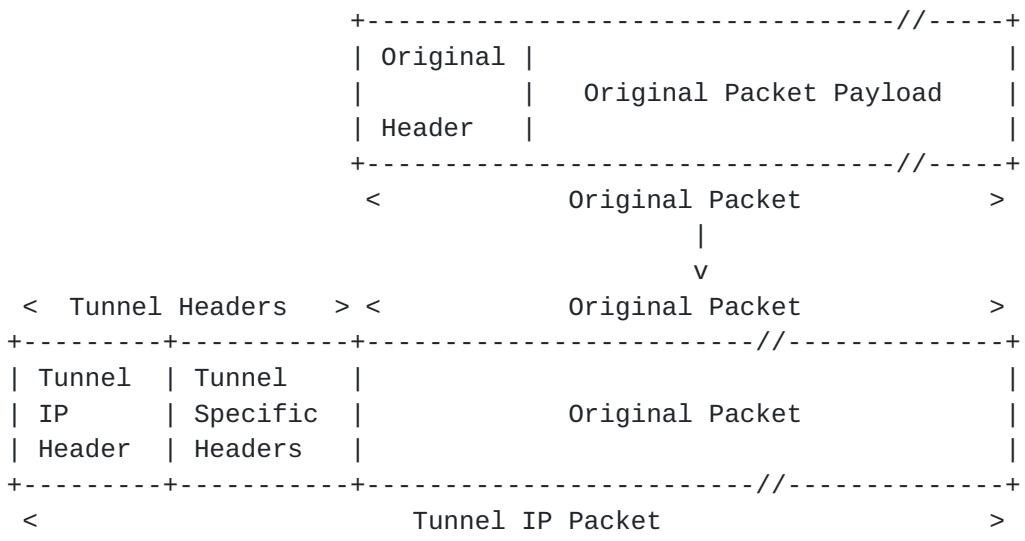


Figure 2: IP Tunnel Encapsulation

Commonly used IP tunneling protocols include Generic Routing Encapsulation (GRE) [RFC1701][RFC2784], Generic Routing Encapsulation over IPv4 Networks (GREIPv4) [RFC1702] and IP Encapsulation within IP (IPv4INIPv4) [RFC1853][RFC2003], Minimal Encapsulation within IP

(MINENC) [RFC2004], IPv6 over IPv4 Tunneling (IPv6INIPv4) [RFC4213], Generic Packet Tunneling in IPv6 Specification (IPv6GEN) [RFC2473] and IPSEC tunneling mode (IPSEC) [RFC4301][RFC4303]. Among these tunneling protocols, the tunnel headers in IPv4INIPv4, IPv6INIPv4 and IPv6GEN contain only a tunnel IP header, and no tunnel specific headers. All the other tunneling protocols have a tunnel header consisting of both a tunnel IP header and a tunnel specific header. The tunnel specific header is the GRE header for GRE and GREIPv4, the minimum encapsulation header for MINENC and the Encapsulation Security Payload (ESP) header for IPSEC tunneling mode. As will be discussed in Section 4.3, some of the tunnel specific headers may be used to identify a flow in the tunnel and facilitate NSIS operating over IP tunnels.

3.2. NSIS QoS Signaling in the Presence of IP Tunnels

Typically, applications use NSIS QoS signaling to reserve resources for a flow along the flow path. NSIS QoS signaling can be initiated by either the flow sender or flow receiver. Figure 3 shows an example scenario with five NSIS nodes, including flow sender node A, flow receiver node E, and intermediate NSIS nodes B, C and D. Nodes which are not NSIS QoS capable are not shown.

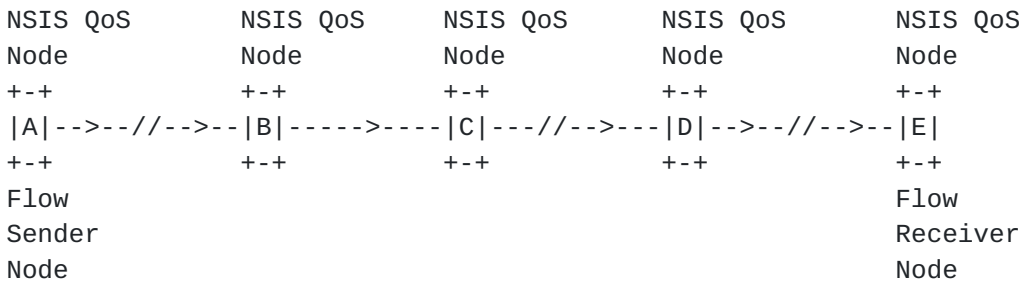


Figure 3: Example Scenario of NSIS QoS Signaling

Figure 4 illustrates a sender-initiated signaling sequence in the scenario of Figure 3. Sender node A sends a RESERVE message towards receiver node E. The RESERVE message gets forwarded by intermediate NSIS Nodes B, C, and D and finally reaches receiver node E. Receiver node E then sends back a RESPONSE message confirming the QoS reservation, again through the previous intermediate NSIS nodes in the data flow path.

There are two important aspects in the above signaling process that are worth mentioning. First, the flow sender does not initially know exactly which intermediate nodes are NSIS-aware and should be involved in the signaling process for a flow from node A to node E. Discovery of those nodes, namely node B, C and D is accomplished by a separate NSIS peer discovery process (not shown above, see

[I-D.ietf-nsis-ntlp]). The NSIS peer discovery messages contain special IP header and payload format or include a Router Alert Option (RAO) [RFC2113] [RFC2711]. The special formats of NSIS discovery messages allow node B, C and D to intercept them and subsequently insert themselves into the signaling path for the flow in question. After formation of the signaling path, all signaling messages corresponding to this flow will be passed to these nodes for processing. Other nodes which are not NSIS-aware simply forward all signaling messages like any other IP packets without additional handling.

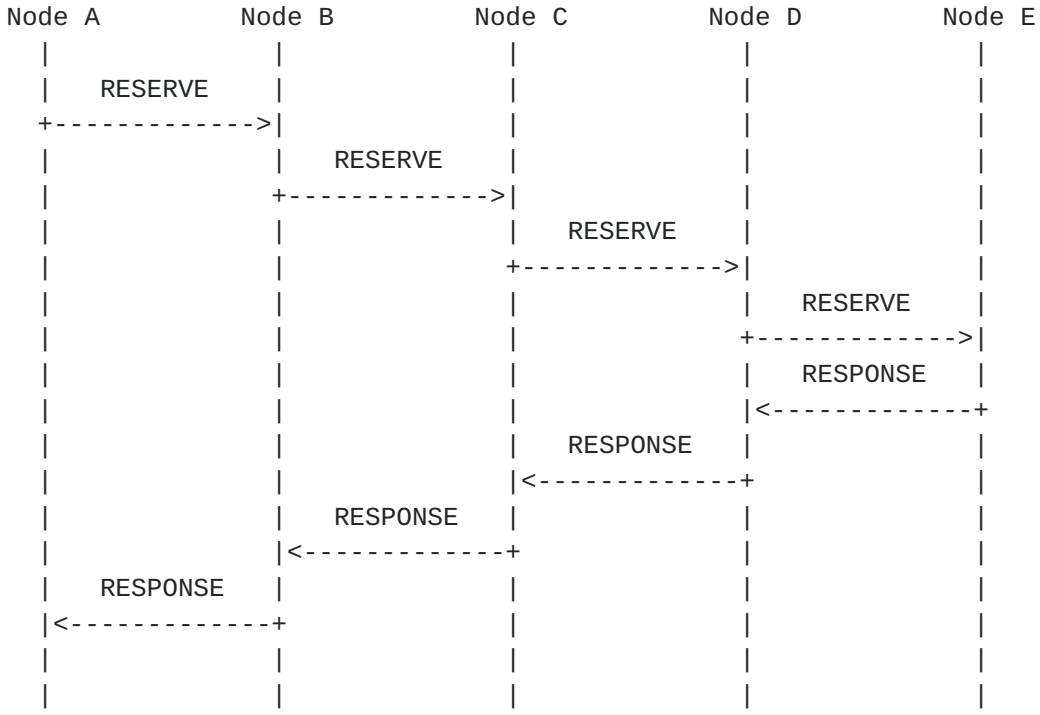


Figure 4: Sender-initiated NSIS QoS Signaling

Second, the goal of QoS signaling is to install control information to give QoS treatment for the flow being signaled. Basic QoS control information includes the data Flow ID for packet classification and the type of QoS treatment those packets are entitled to. The Flow ID contains a set of header fields such as flow sender and receiver addresses, protocol and port numbers.

Now consider Figure 5 where nodes B, C and D are end-points and intermediate nodes of an IP tunnel. During the signaling path discovery process, node B can still intercept and process NSIS peer discovery messages if it recognizes them before performing tunnel encapsulation; node D can identify NSIS peer discovery messages after performing tunnel decapsulation. A tunnel intermediate node such as node C, however, only sees the tunnel header of the packets and will

not be able to identify the original NSIS peer discovery message or insert itself in the flow signaling path. Furthermore, the Flow ID of the original flow is based on IP header fields of the original packet. Those fields are also hidden in the payload of the tunnel packet. So there is no way node C can classify packets belonging to that flow in the tunnel. In summary, the problem is that tunnel intermediate nodes are unable to intercept original NSIS signaling messages and unable to classify original data flow packets as a result of tunnel encapsulation. An IP tunnel segment appears just like a QoS-unaware virtual link. Since the best QoS of an end-to-end path is judged based on its weakest segment, leaving the tunnel path "untended" risks voiding other efforts to provide QoS in the rest of the path.

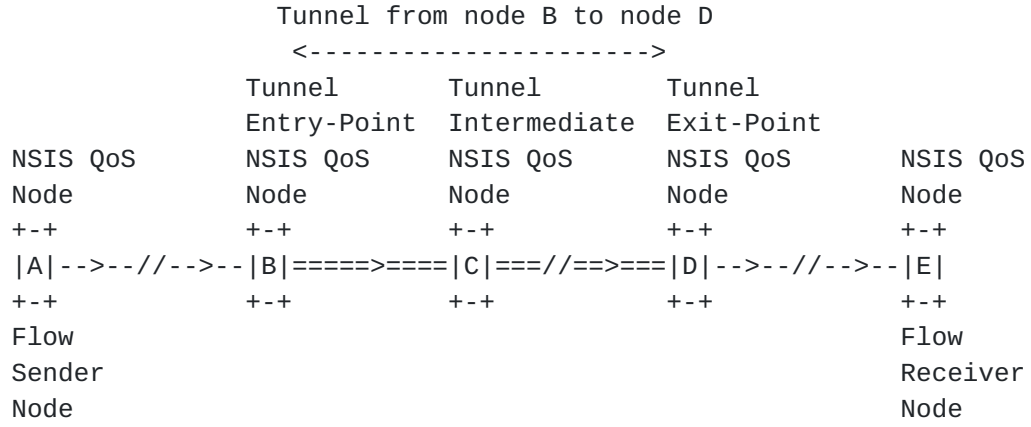


Figure 5: Example Scenario of NSIS QoS Signaling with IP Tunnel

4. Design Overview

4.1. Design Requirements

We identify the following design requirements for NSIS operating over IP tunnels.

- The mechanism should work with all common IP tunneling protocols listed in Section 3.1.
- Some IP tunnels maintain pre-configured QoS sessions inside the tunnel. The mechanism should work for IP tunnels both with and without pre-configured tunnel QoS sessions.
- The mechanism should minimize the required upgrade to existing infrastructure in order to facilitate its deployment. Specifically, we limit the necessary upgrade to NSIS-aware tunnel end-points. Only tunnel end-points need to support the mechanism defined in this document. Such tunnel end-points are called NSIS-

tunnel-aware end-points. All other nodes, both inside and outside the tunnel should be transparent to this mechanism.

- The mechanism should facilitate its incremental deployment by providing a method for one NSIS-tunnel-aware end-point to discover whether the other end-point is also NSIS-tunnel-aware.
- The mechanism should learn from design experience of previous work on RSVP over IP tunnels (RSVP-TUNNEL) [RFC2746], while also addressing the following major differences of NSIS from RSVP. First, NSIS is designed as a generic framework to accommodate various signaling application needs, and therefore is split into a signaling transport layer and a signaling application layer; RSVP does not have a layer split and is designed only for QoS signaling. Second, NSIS QoS NSLP allows both sender-initiated and receiver-initiated reservations; RSVP only supports receiver-initiated reservations. Third, NSIS deals only with unicast; RSVP also supports multicast. Fourth, NSIS integrates a new Session ID feature which is different from the session identification concept in RSVP.

4.2. Overall Design Approach

The overall design of this NSIS signaling and IP tunnel interworking mechanism draws similar concepts from RSVP-TUNNEL [RFC2746], but is tailored and extended for NSIS operation.

Since a flow is considered unidirectional, to accommodate flows in both directions of a tunnel, we require both tunnel entry-point and tunnel exit-point to be NSIS-tunnel-aware. If an NSIS-tunnel-aware end-point needs to know whether the other tunnel end-point is also NSIS-tunnel-aware, it may use the NSIS-tunnel capability discovery mechanism defined in Section 7.

Tunnel end-points need to always intercept NSIS peer discovery messages and insert themselves into the NSIS signaling path so they can receive all NSIS signaling messages and coordinate their interaction with tunnel QoS.

To facilitate the QoS handling in the tunnel, the end-to-end QoS session will be mapped to a tunnel QoS session, either pre-configured or dynamically created. An important property of a tunnel QoS session is its tunnel Flow ID which identifies the end-to-end data flow within the tunnel. In both tunnels with and without pre-configured QoS sessions, the tunnel Flow ID is assigned based on information available in the tunnel header, therefore solving the problem for tunnel-intermediate nodes to classify flow packets as discussed in Section 3.2. An example tunnel Flow ID contains the tunnel entry-point and exit-point IP addresses and a tunnel inserted UDP port number. We discuss more details about recommended choices

of tunnel Flow ID for different IP tunneling protocols in Section 4.3.

For tunnels that maintain pre-configured QoS sessions, upon receiving a request to reserve resources for an end-to-end session, the tunnel end-point maps the end-to-end QoS session to an existing tunnel session. To simplify the design, the mapping decision is always made by the tunnel entry-point regardless of whether the end-to-end session uses sender-initiated or receiver-initiated NSIS signaling mode. The details about which end-to-end session can be mapped to which pre-configured tunnel session depend on policy mechanisms outside the scope of this document.

For tunnels that do not maintain pre-configured QoS sessions, the NSIS-tunnel-aware end-points dynamically create and manage a corresponding tunnel QoS session for the end-to-end session. Since the initiation mode of both QoS sessions can be sender-initiated or receiver-initiated, to simplify the design, we require that the initiation mode of the tunnel QoS session follow that of the end-to-end QoS session. In other words, the end-to-end QoS session and its corresponding tunnel QoS session are either both sender-initiated or both receiver-initiated. To keep the handling mechanism consistent with the case for tunnels with pre-configured QoS sessions, the tunnel entry-point always initiates the mapping between the tunnel session and the end-to-end session.

As the mapping initiator, the tunnel entry-point records the association between the end-to-end session and its corresponding tunnel session, both in tunnels with and without pre-configured QoS sessions. This association serves two purposes, one at the signaling plane and the other at the data plane. At the signaling plane, the association enables the tunnel entry-point to coordinate necessary interaction, such as QoS adjustment in sender-initiated reservations, between the end-to-end and the tunnel QoS sessions. At the data plane, the association allows the tunnel entry-point to correctly encapsulate data flow packets according to the chosen tunnel Flow ID. Since the tunnel Flow ID uses header fields that are visible inside the tunnel, the tunnel intermediate nodes can classify the data flow packets and apply appropriate QoS treatment.

In addition to the tunnel entry-point recording the association between the end-to-end session and its corresponding tunnel session, the tunnel exit-point also needs to maintain the same association for similar reasons. At the signaling plane, this association at the tunnel exit-point enables the interaction of the end-to-end and the tunnel QoS session such as QoS adjustment in receiver-initiated reservations. At the data plane, this association tells the tunnel exit-point that the relevant data flow packets need to be

decapsulated according to the corresponding tunnel Flow ID.

The tunnel exit-point learns about the mapping between the tunnel and the end-to-end QoS sessions, including the tunnel Flow ID and the tunnel session's Session ID that corresponds to the end-to-end session, through the follow methods. In tunnels with pre-configured QoS sessions, the mapping information between the corresponding tunnel and end-to-end QoS sessions may be pre-configured as well. In tunnels without pre-configured QoS sessions, the tunnel exit-point knows the tunnel Flow ID through the NSIS signaling process that creates the tunnel QoS sessions inside the tunnel. Meanwhile, the tunnel exit-point maps the Session IDs of the tunnel QoS session and the end-to-end session through the QoS NSLP BOUND_SESSION_ID object [I-D.ietf-nsis-qos-nslp]. Specifically, when used for NSIS signaling over IP tunnels, the BOUND_SESSION_ID object carries the Session ID of the tunnel session and a Binding Code of value 0x01 indicating tunnel handling. The tunnel entry-point includes this tunnel binding object in appropriate end-to-end signaling messages. Upon receiving this binding object, the tunnel exit-point records the association between the tunnel QoS session and the corresponding end-to-end QoS session.

One problem for NSIS operating over IP tunnels which dynamically create QoS sessions is that it involves two signaling sequences. The outcome of the tunnel signaling session directly affects the outcome of the end-to-end signaling session. Since the two signaling sessions overlap in time, there are circumstances when a tunnel end-point has to decide whether it should proceed with the end-to-end signaling session while it is still waiting for results of the tunnel session. Sequential mode and parallel mode are two basic options for this problem. In sequential mode, end-to-end signaling pauses when it is waiting for results of tunnel signaling, and resumes upon receipt of the tunnel signaling outcome. In parallel mode, end-to-end signaling continues outside the tunnel while tunnel signaling is still in process and its outcome is unknown. The parallel mode may lead to reduced signaling delays if the QoS resources in the tunnel path are sufficient compared to the rest of the end-to-end path. If the QoS resources in the tunnel path are more constraint than the rest of the end-to-end path, however, the parallel mode may lead to wasted end-to-end signaling or necessitates re-negotiation after the tunnel signaling outcome becomes available. In those cases, the signaling flow of the parallel mode also tends to be more complicated. This document adopts a sequential mode approach. In addition, the actual signaling process uses the QoS NSLP message binding mechanism [I-D.ietf-nsis-qos-nslp] to convey the dependency relationship between corresponding messages of the tunnel session and the end-to-end session.

4.3. Tunnel Flow ID for Different IP Tunneling Protocols

A tunnel Flow ID identifies the end-to-end flow for packet classification within the tunnel. The tunnel Flow ID is based on a set of tunnel header fields. Different tunnel Flow ID can be chosen for different tunneling mechanisms in order to minimize the classification overhead. This document specifies the following Flow ID formats for the respective tunneling protocols.

- For IPv6 tunneling protocols (IPv6GEN), the tunnel Flow ID consists of the tunnel entry-point IPv6 address and the tunnel exit-point IPv6 address plus a unique IPv6 flow label [RFC3697].
- For IPSEC tunnel mode (IPSEC), the tunnel Flow ID contains the tunnel entry-point IP address and the tunnel exit-point IP address plus the Security Parameter Index (SPI).
- For all other tunneling protocols (GRE, GREIPv4, IPv4INIPv4, MINENC, IPv6INIPv4), the tunnel entry-point inserts an additional UDP header between the tunnel header and the original packet. The Flow ID consists of the tunnel entry-point and tunnel exit-point IP addresses and the source port number in the additional UDP header. In these cases, it is especially important that the tunnel exit-point also understands the additional UDP encapsulation, and therefore can correctly decapsulate both the normal tunnel header and the additional UDP header. In other words, both tunnel end-points need to be NSIS-tunnel-aware.

The above recommendations about choosing tunnel Flow ID apply to dynamically created QoS tunnel sessions. For pre-configured QoS tunnel sessions, the corresponding Flow ID is determined by the configuration mechanism itself. For example, if the tunnel QoS is DiffServ based, the DiffServ Code Point (DSCP) field value may be used to identify the corresponding tunnel session.

5. NSIS Operation over Tunnels with Pre-configured QoS Sessions

When tunnel QoS is managed by pre-configured QoS sessions, both the tunnel entry-point and tunnel exit-point also need to be configured with the Flow ID of the tunnel QoS session. This is to enable the tunnel end-points to correctly perform matching encapsulating and decapsulating operations. The procedures of NSIS operating over tunnels with pre-configured QoS sessions are slightly different depending on whether the end-to-end NSIS signaling is sender-initiated or receiver-initiated. But in either case, it is the tunnel entry-point that first creates the mapping between a tunnel session and an end-to-end session.

5.1. Sender-initiated Reservation

Figure 6 illustrates the signaling sequence when end-to-end signaling outside the tunnel is sender-initiated. Upon receiving a RESERVE message from the sender, Tentry checks tunnel QoS configuration, determines whether and how this end-to-end session can be mapped to a pre-configured tunnel session. The mapping criteria are part of the pre-configuration and outside the scope of this document. Tentry then tunnels the RESERVE message to Texit. Texit forwards the RESERVE message to the receiver. The receiver replies with a RESPONSE message which arrives at Texit, Tentry and finally the sender. If the RESPONSE message that Tentry receives confirms that the overall signaling is successful, Tentry starts to encapsulate all incoming packets of the data flow using the tunnel Flow ID corresponding to the mapped tunnel session. Texit knows how to decapsulate the tunnel packets because it recognizes the mapped tunnel Flow ID based on information supplied during tunnel session pre-configuration.

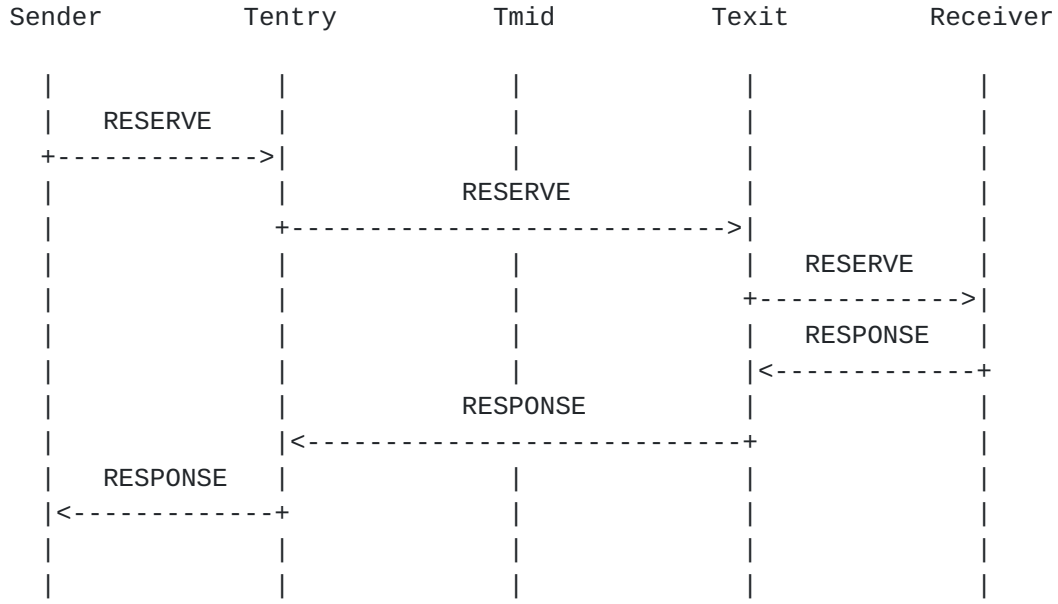


Figure 6: Sender-initiated End-to-end Session with Pre-configured Tunnel QoS Sessions

5.2. Receiver-initiated Reservation

Figure 7 shows the signaling sequence when end-to-end signaling outside the tunnel is receiver-initiated. Upon receiving the first end-to-end Query message, Tentry examines the tunnel QoS configuration, updates and tunnels the Query message to Texit. Texit decapsulates the QUERY message, processes it and forwards it toward the receiver. Later, the receiver sends back a RESERVE message passing through Texit and arriving at Tentry. Tentry decides on whether and how the QoS request for this end-to-end session can be

mapped to a pre-configured tunnel session based on an algorithm outside the scope of this document. Then Tentry tunnels the RESERVE message to Texit which forwards it to the receiver. The signaling continues until a RESPONSE message arrives at Tentry, Texit and finally the receiver. If the RESPONSE message that Tentry receives confirms that the overall signaling is successful, Tentry starts to encapsulate all incoming packets of the data flow using the tunnel Flow ID corresponding to the mapped tunnel session. Similarly, Texit knows how to decapsulate the tunnel packets because it recognizes the mapped tunnel Flow ID based on information supplied during tunnel session pre-configuration.

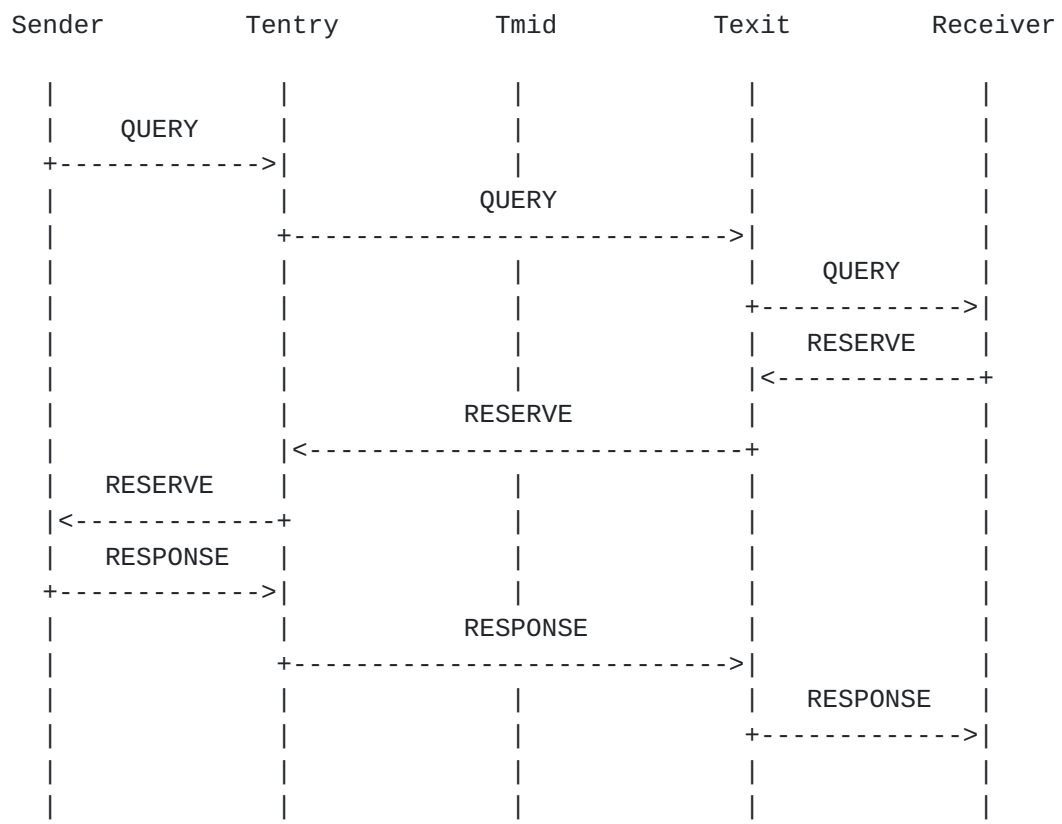


Figure 7: Receiver-initiated End-to-end Session with Pre-configured Tunnel QoS Sessions

Since tunnel QoS signaling is not involved in pre-configured QoS tunnels, Figure 6 and Figure 7 look as if the tunnel is a single virtual link. The signaling path simply skips all tunnel intermediate nodes. However, both Tentry and Texit need to deploy NSIS-tunnel related functionalities described above, including acting on the end-to-end NSIS signaling messages based on tunnel QoS status, mapping end-to-end and tunnel QoS sessions, and correctly encapsulating and decapsulating tunnel packets according to the tunnel protocol and the configured tunnel Flow ID.

6. NSIS Operation over Tunnels with Dynamically Created QoS Sessions

When there are no pre-configured tunnel QoS sessions, a tunnel can apply the same NSIS QoS signaling mechanism used for the end-to-end path to manage the QoS inside the tunnel. The tunnel NSIS signaling involves only those NSIS nodes in the tunnel forwarding path. The Flow IDs for the tunnel signaling are based on tunnel header fields. NSIS peer discovery messages inside the tunnel distinguish themselves using the tunnel header fields, which solves the problem for tunnel intermediate NSIS nodes to intercept signaling messages.

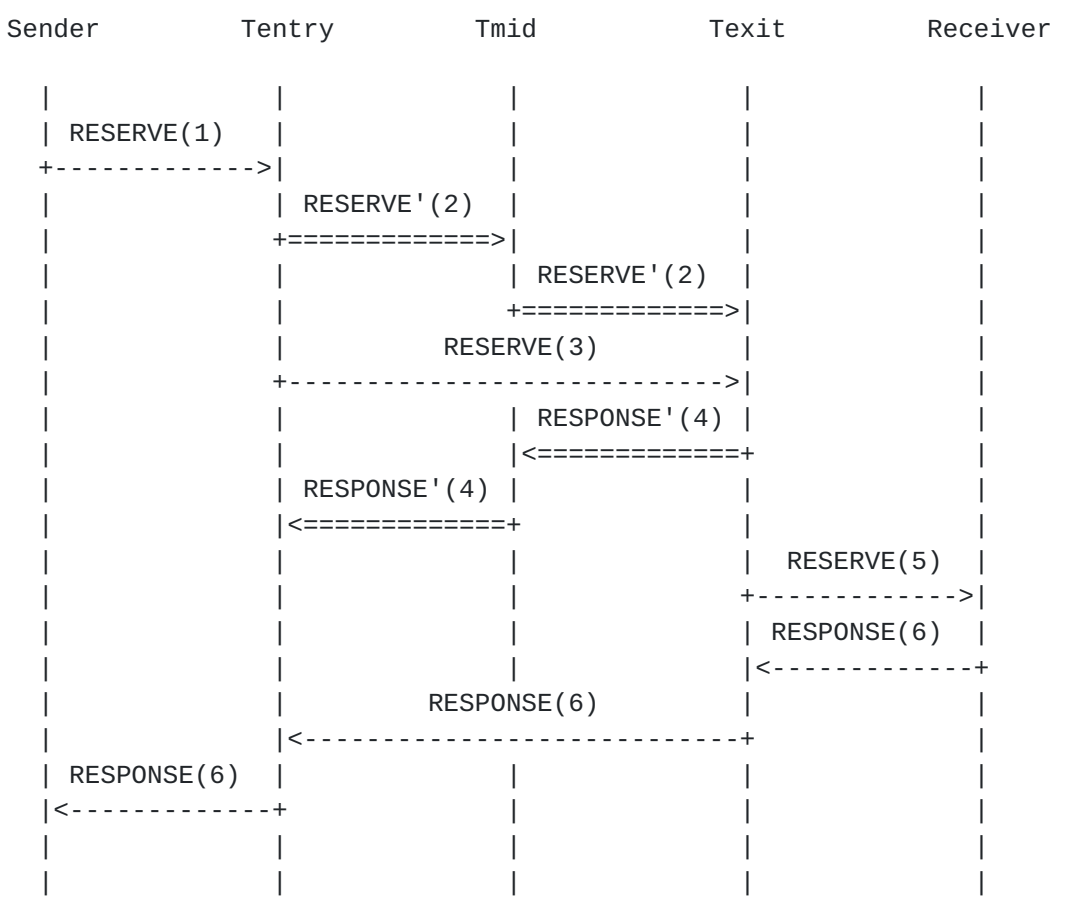
When tunnel end-points dynamically create tunnel QoS sessions, the initiation mode of the tunnel session always follows the initiation mode of the end-to-end session. Specifically, when the end-to-end session is sender-initiated, the tunnel session should also be sender-initiated; when the end-to-end session is receiver-initiated, the tunnel session should also be receiver-initiated.

The tunnel entry-point conveys the corresponding tunnel Flow ID associated with an end-to-end session to the tunnel exit-point during the tunnel signaling process. The tunnel entry-point also informs the binding between the corresponding tunnel session and the end-to-end session to the exit-point through the BOUND_SESSION_ID QoS NSLP message object. The reservation message dependencies between the tunnel session and end-to-end session is resolved using the MSG_ID and BOUND_MSG_ID objects of the QoS NSLP message binding mechanism.

6.1. Sender-initiated Reservation

Figure 8 shows the typical messaging sequence of how NSIS operates over IP tunnels when both end-to-end session and tunnel session are sender-initiated. Tunnel signaling messages are distinguished from end-to-end messages by a prime symbol after the message name. The sender first sends an end-to-end RESERVE message (1) which arrives at Tentry. Tentry chooses the tunnel Flow ID, creates the tunnel session and associates the end-to-end session with the tunnel session. Tentry then sends a tunnel RESERVE' message (2) matching the request of the end-to-end session towards Texit to reserve tunnel resources. This RESERVE' message (2) includes a MSG_ID object which contains a randomly generated 128-bit MSG_ID. Meanwhile, Tentry inserts a BOUND_MSG_ID object containing the same MSG_ID as well as a BOUND_SESSION_ID object containing the Session ID of the tunnel session into the original RESERVE message, and sends this RESERVE message (3) towards Texit using normal tunnel encapsulation. The Message_Binding_Type flag of both the MSG_ID and BOUND_MSG_ID objects in the RESERVE' and RESERVE messages (2, 3) is SET, indicating a bidirectional binding. The tunnel RESERVE' message (2) is processed hop-by-hop inside the tunnel for the flow identified by the chosen

tunnel Flow ID, while the end-to-end RESERVE message (3) passes through the tunnel intermediate nodes (Tmid) just like other tunneled packets. These two messages could arrive at Texit in different orders, and the reaction of Texit in these different situations should combine the tunnel QoS message processing rules with the QoS NSLP processing principles for message binding [I-D.ietf-nsis-qos-nslp], as illustrated below.



(1,5): RESERVE w/o BOUND_MSG_ID and BOUND_SESSION_ID
(2): RESERVE' w/ MSG_ID
(3): RESERVE w/ BOUND_MSG_ID and BOUND_SESSION_ID

Figure 8: Sender-initiated Reservation for Both End-to-end and Tunnel Signaling

The first possibility is shown in the example messaging flow of Figure 8, where the tunnel RESERVE' message (2), aka the triggering message in QoS NSLP message binding terms, arrives first. Since the message binding is bidirectional, Texit records the MSG_ID of the RESERVE' message (2), enques it and starts a MsgIDWait timer waiting

for the end-to-end RESERVE message (3), aka the bound signaling message in QoS NSLP message binding terms. The timer value is set to the default retransmission timeout period QOSNSLP_REQUEST_RETRY. When the end-to-end RESERVE message (3) arrives, Texit notices that there is an existing stored MSG_ID which matches the MSG_ID in the BOUND_MSG_ID object of the incoming RESERVE message (3). Therefore the message binding condition has been satisfied. Texit resumes processing of the tunnel RESERVE' message (2), creates the reservation state for the tunnel session, and sends a tunnel RESPONSE' message (4) to Tentry. At the same time, Texit checks the BOUND_SESSION_ID object of the end-to-end RESERVE message (3) and records the binding of the corresponding tunnel session with the end-to-end session. Texit also updates the end-to-end RESERVE message based on the result of the tunnel session reservation, removes its tunnel BOUND_SESSION_ID and BOUND_MSG_ID object and forwards the end-to-end RESERVE message (5) along the path towards the receiver. When the receiver receives the end-to-end RESERVE message (5), it sends an end-to-end RESPONSE message (6) back to the sender.

The second possibility is that the end-to-end RESERVE message arrives before the tunnel RESERVE' message at Texit. In that case, Texit notices a BOUND_SESSION_ID object and a BOUND_MSG_ID object in the end-to-end RESERVE message, but realizes that the tunnel session does not exist yet. So Texit enques the RESERVE message and starts a MsgIDWait timer. The timer value is set to the default retransmission timeout period QOSNSLP_REQUEST_RETRY. When the corresponding tunnel RESERVE' message arrives with a MSG_ID matching that of the outstanding BOUND_MSG_ID object, the message binding condition is satisfied. Texit sends a tunnel RESPONSE' message back to Tentry and updates the end-to-end RESERVE message by incorporating the result of the tunnel session reservation, as well as removing the tunnel BOUND_SESSION_ID and BOUND_MSG_ID objects. Texit then forwards the end-to-end RESERVE message along the path towards the receiver. When the receiver receives the end-to-end RESERVE message, it sends an end-to-end RESPONSE message back to the sender.

Yet another possibility is that the tunnel RESERVE' message arrives at Texit first but the end-to-end RESERVE message never arrives. In that case, the MsgIDWait timer for the queued tunnel RESERVE' message will expire. Texit should send a tunnel RESPONSE' message back to Tentry indicating a reservation error has occurred, and discard the tunnel RESERVE' message. The last possibility is that the end-to-end RESERVE message arrives at Texit first but the tunnel RESERVE' message never arrives. And in that case, the MsgIDWait timer for the queued end-to-end RESERVE message will expire. Texit should treat this situation as a local reservation failure, and according to [I-D.ietf-nsis-qos-nslp], Texit as a stateful QoS NSLP should generate an end-to-end RESPONSE message indicating the RESERVE error

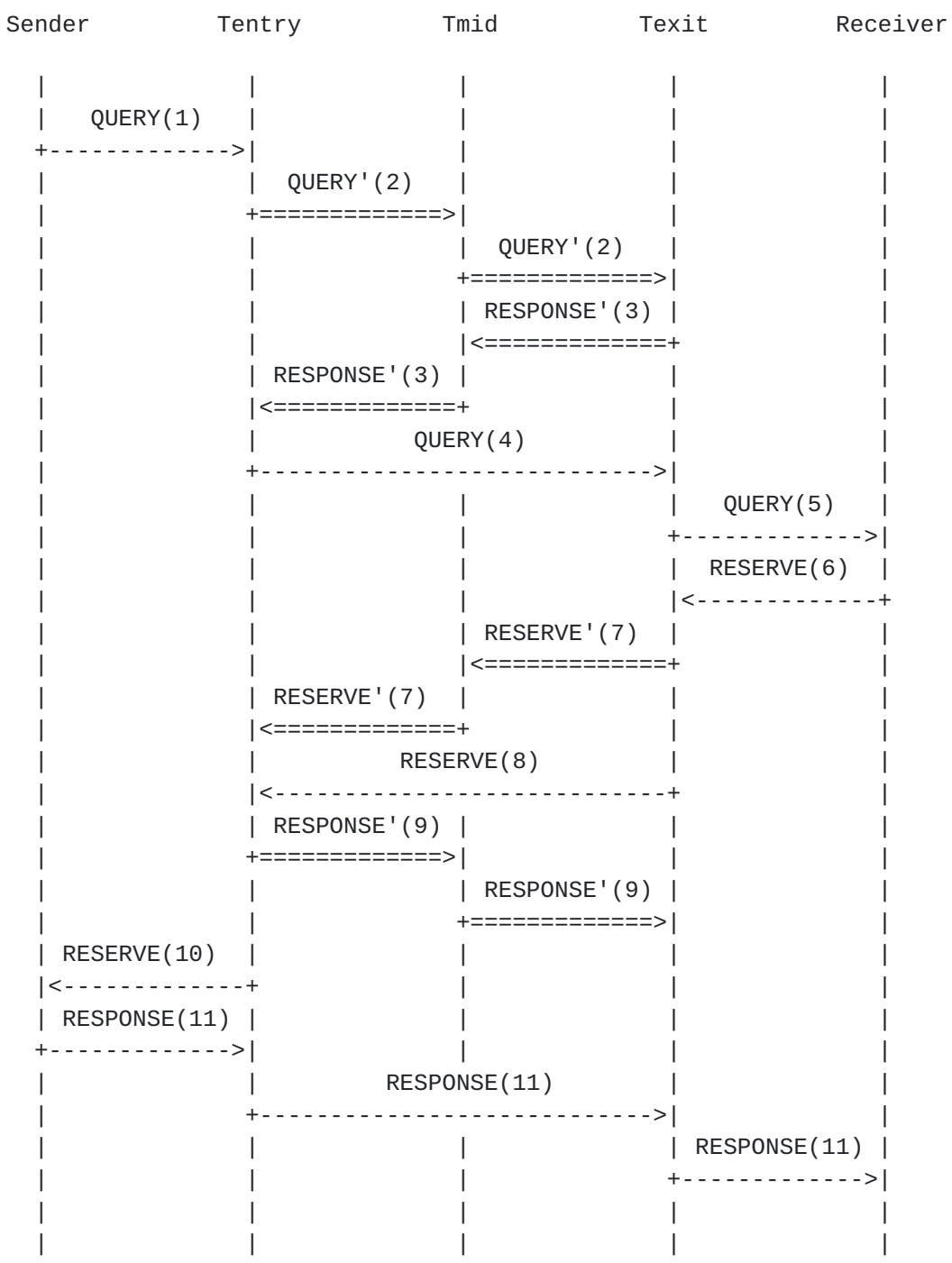
to the sender.

Once the end-to-end and the tunnel QoS session have both been successfully created and associated, the tunnel end-points Tentry and Texit coordinate the signaling between the two sessions and make sure that adjustment or teardown of either session may trigger similar actions for the other session as necessary, by invoking appropriate signaling messages.

6.2. Receiver-initiated Reservation

Figure 9 shows the typical messaging sequence of how NSIS signaling operates over IP tunnels when both end-to-end and tunnel sessions are receiver-initiated. Upon receiving an end-to-end QUERY message (1) from the sender, Tentry chooses the tunnel Flow ID and sends a tunnel QUERY' message (2) matching the request of the end-to-end session towards Texit. This tunnel QUERY' message (2) is meant to discover QoS characteristics of the tunnel path, rather than initiating an actual reservation. Therefore, it includes a Request Identification Information (RII) object but does not set the RESERVE-INIT flag. The tunnel QUERY' message (2) is processed hop-by-hop inside the tunnel for the flow identified by the tunnel Flow ID. When Texit receives this tunnel QUERY' message (2), it replies with a corresponding tunnel RESPONSE' message (3) containing the tunnel path characteristics. After receiving the tunnel RESPONSE' message (3), Tentry creates the tunnel session, generates an outgoing end-to-end QUERY message (4) considering the tunnel path characteristics, appends a tunnel BOUND_SESSION_ID object containing the tunnel Session ID, and sends it toward Texit using normal tunnel encapsulation. The end-to-end QUERY message (4) passes along tunnel intermediate nodes like other tunneled packets. Upon receiving this end-to-end QUERY message (4), Texit notices the tunnel session binding and creates the tunnel session state, removes the tunnel BOUND_SESSION_ID object and forwards the end-to-end QUERY message (5) further along the path.

The end-to-end QUERY message (5) arrives at the receiver and triggers a RESERVE message (6). When Texit receives the RESERVE message (6), it notices that the session is bound to a receiver-initiated tunnel session. Therefore, Texit triggers a RESERVE' message (7) toward Tentry for the tunnel session reservation. This tunnel RESERVE' message (7) includes a randomly generated 128-bit MSG_ID. Meanwhile, Texit inserts a BOUND_MSG_ID object containing the same MSG_ID and a BOUND_SESSION_ID object containing the tunnel Session ID into the end-to-end RESERVE message (8), and sends it towards Tentry using normal tunnel encapsulation. The Message_Binding_Type flag of the MSG_ID and BOUND_MSG_ID objects in the RESERVE' and RESERVE messages (7,8) is SET, indicating a bidirectional binding.



(1,5): QUERY w/ RESERVE-INIT (2): QUERY' w/ RII
(4): QUERY w/ RESERVE-INIT and BOUND_SESSION_ID
(6,10): RESERVE w/o BOUND_SESSION_ID (7): RESERVE' w/ MSG_ID
(8): RESERVE w/ BOUND_MSG_ID and BOUND_SESSION_ID

Figure 9: Receiver-initiated Reservation for Both End-to-end and Tunnel Signaling

At Tentry, the tunnel RESERVE' message (7) and the end-to-end RESERVE message (8) could arrive in different orders. In a typical case shown in Figure 9, the tunnel RESERVE' message (7) arrives first. Tentry records the MSG_ID of the tunnel RESERVE' message (7) and starts a MsgIDWait timer. When the end-to-end RESERVE message (8) with the BOUND_MSG_ID object containing the same MSG_ID arrives, the message binding condition is satisfied. Tentry resumes processing of the tunnel RESERVE' message (7), creates the reservation state for the tunnel session, and sends a tunnel RESPONSE' message (9) to Texit. At the same time, Tentry creates the outgoing end-to-end RESERVE message (10) by incorporating results of the tunnel session reservation and removing the BOUND_SESSION_ID and BOUND_MSG_ID objects, and forwards it along the path towards the sender. When the sender receives the end-to-end RESERVE message (10), it sends an end-to-end RESPONSE message (11) back to the receiver.

If the end-to-end RESERVE message arrives before the tunnel RESERVE' message at Tentry, or either of the two messages fails to arrive at Tentry, the processing rules at Tentry is similar to those of Texit in the same situation discussed in Section 6.1.

Once the end-to-end and the tunnel QoS session have both been successfully created and associated, the tunnel end-points Tentry and Texit coordinate the signaling between the two sessions and make sure that adjustment or teardown of either session can trigger similar actions for the other session as necessary, by invoking appropriate signaling messages.

7. NSIS-Tunnel Signaling Capability Discovery

When operating over a tunnel, NSIS-tunnel-aware end-points may need to perform special encapsulation and decapsulation such as inserting and removal of an extra UDP header, depending on the tunnel Flow ID format. Therefore, before the NSIS-tunnel-aware end-point decides to initiate such encapsulation, it needs to know whether the other entry-point is also NSIS-tunnel-aware and thus capable of performing matching decapsulation. This section defines a mechanism to enable this capability discovery for tunnels using dynamically created tunnel QoS sessions. For tunnels with pre-configured QoS sessions, an end-point can learn the NSIS-tunnel capability information of the other end-point during the pre-configuration process.

7.1. NODE_CAPABILITY Object Format

A GIST NODE_CAPABILITY object is defined to discover the NSIS-tunnel handling capability of a tunnel end-point. The format of the NODE_CAPABILITY object follows the general object definition in GIST.

The object contains a fixed header specifying the object type and object length, followed by the object value.

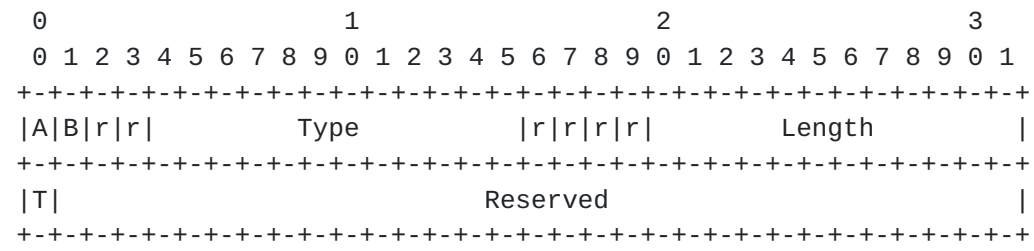


Figure 10: NODE_CAPABILITY Object Format

Type: NODE_CAPABILITY

Length: 1, measured in units of 32-bit word

Value: Value contains a single 'T' bit, indicating the node supports the NSIS-tunnel handling mechanisms defined in this document. The reserved bits in the value field can be used to signal other node characteristics in the future.

The bits marked 'A' and 'B' define the desired behavior for objects whose Type field is not recognized. If a node does not recognize the NODE_CAPABILITY object, the desired behavior is "Ignore". That is, the object must be deleted and the rest of the message processed as usual. This can be satisfied by setting 'AB' to '01' according to Appendix A.2 of the GIST specification [I-D.ietf-nsis-ntlp].

7.2. Using NODE_CAPABILITY Object

The NODE_CAPABILITY object is included in a QUERY or RESERVE message by a tunnel end-point that needs to learn about the other end-point's NSIS tunnel handling capability. If the receiving tunnel end-point is indeed NSIS-tunnel-aware, it recognizes this object and knows that the sending end-point is NSIS-tunnel-aware. The receiving tunnel end-point places the same object in a RESPONSE message to inform the sending end-point that it is also NSIS-tunnel-aware. Example procedures of how to use the NODE_CAPABILITY object over tunnels that dynamically creates QoS sessions are further detailed below.

First, assume that both end-to-end and tunnel session are sender-initiated (Section 6.1) and an NSIS-tunnel-aware Tentry wants to discover the NSIS-tunnel capability of Texit before starting the tunnel signaling. Tentry includes a Request Identification Information (RII) object (see [I-D.ietf-nsis-qos-nslp]) and a

NODE_CAPABILITY object with T bit set in the first end-to-end RESERVE message sent to Texit. When Texit receives this RESERVE message, if it is also NSIS-tunnel-aware, it learns that Tentry is NSIS-tunnel-aware and includes the same object with T bit set in the following end-to-end RESPONSE message sent back to Tentry. Otherwise, Texit ignores the NODE_CAPABILITY object. When Tentry receives the RESPONSE message, it knows whether Texit is NSIS-tunnel-aware by checking the existence of the NODE_CAPABILITY object and its T bit. If both tunnel endpoints are NSIS-tunnel-aware, the rest of the procedures follows those defined in Section 6.1.

Second, assume that both end-to-end and tunnel sessions are receiver-initiated (Section 6.2) and the NSIS-tunnel-aware Tentry wants to discover the NSIS-tunnel capability of Texit before creating a tunnel session. Tentry includes an RII object and a NODE_CAPABILITY object with T bit set in the first end-to-end QUERY message sent towards Texit. If Texit is NSIS-tunnel-aware, it learns from the NODE_CAPABILITY object that Tentry is also NSIS-tunnel-aware. In the later end-to-end RESPONSE message to this QUERY message, Texit includes a NODE_CAPABILITY object with T bit set to notify Tentry of its NSIS-tunnel capability. If both tunnel end-points are NSIS-tunnel-aware, the rest of the procedures follows those in Section 6.2.

8. IANA Considerations

This document defines a new object type called NODE_CAPABILITY for GIST. Its OType value needs to be assigned by IANA. The object format and the setting of the extensibility bits are defined in Section 7.

9. Security Considerations

This draft does not raise new security threats. Security considerations for NSIS NTLP and QoS NSLP are discussed in [I-D.ietf-nsis-ntlp] and [I-D.ietf-nsis-qos-nslp], respectively. General threats for NSIS can be found in [RFC4081].

10. Acknowledgements

The authors would like to thank Roland Bless, Hannes Tschofenig, Georgios Karagiannis and other members of the NSIS working group for comments to this work.

11. References

11.1. Normative References

[I-D.ietf-nsis-ntlp]
Schulzrinne, H. and M. Stiemerling, "GIST: General Internet Signalling Transport", draft-ietf-nsis-ntlp-20 (work in progress), June 2009.

[I-D.ietf-nsis-qos-nslp]
Manner, J., Karagiannis, G., and A. McDonald, "NSLP for Quality-of-Service Signaling", draft-ietf-nsis-qos-nslp-18 (work in progress), January 2010.

11.2. Informative References

[I-D.ietf-nsis-applicability-mobility-signaling]
Sanda, T., Fu, X., Jeong, S., Manner, J., and H. Tschofenig, "Applicability Statement of NSIS Protocols in Mobile Environments", draft-ietf-nsis-applicability-mobility-signaling-14 (work in progress), January 2010.

[I-D.ietf-nsis-nslp-natfw]
Stiemerling, M., Tschofenig, H., Aoun, C., and E. Davies, "NAT/Firewall NSIS Signaling Layer Protocol (NSLP)", draft-ietf-nsis-nslp-natfw-23 (work in progress), February 2010.

[I-D.ietf-nsis-qspec]
Bader, A., Kappler, C., and D. Oran, "QoS NSLP QSPEC Template", draft-ietf-nsis-qspec-24 (work in progress), January 2010.

[RFC1701] Hanks, S., Li, T., Farinacci, D., and P. Traina, "Generic Routing Encapsulation (GRE)", RFC 1701, October 1994.

[RFC1702] Hanks, S., Li, T., Farinacci, D., and P. Traina, "Generic Routing Encapsulation over IPv4 networks", RFC 1702, October 1994.

[RFC1853] Simpson, W., "IP in IP Tunneling", RFC 1853, October 1995.

[RFC2003] Perkins, C., "IP Encapsulation within IP", RFC 2003, October 1996.

[RFC2004] Perkins, C., "Minimal Encapsulation within IP", RFC 2004, October 1996.

[RFC2113] Katz, D., "IP Router Alert Option", RFC 2113, February 1997.

[RFC2205] Braden, B., Zhang, L., Berson, S., Herzog, S., and S. Jamin, "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification", RFC 2205, September 1997.

[RFC2207] Berger, L. and T. O'Malley, "RSVP Extensions for IPSEC Data Flows", RFC 2207, September 1997.

[RFC2401] Kent, S. and R. Atkinson, "Security Architecture for the Internet Protocol", RFC 2401, November 1998.

[RFC2473] Conta, A. and S. Deering, "Generic Packet Tunneling in IPv6 Specification", RFC 2473, December 1998.

[RFC2711] Partridge, C. and A. Jackson, "IPv6 Router Alert Option", RFC 2711, October 1999.

[RFC2746] Terzis, A., Krawczyk, J., Wroclawski, J., and L. Zhang, "RSVP Operation Over IP Tunnels", RFC 2746, January 2000.

[RFC2784] Farinacci, D., Li, T., Hanks, S., Meyer, D., and P. Traina, "Generic Routing Encapsulation (GRE)", RFC 2784, March 2000.

[RFC3220] Perkins, C., "IP Mobility Support for IPv4", RFC 3220, January 2002.

[RFC3697] Rajahalme, J., Conta, A., Carpenter, B., and S. Deering, "IPv6 Flow Label Specification", RFC 3697, March 2004.

[RFC4080] Hancock, R., Karagiannis, G., Loughney, J., and S. Van den Bosch, "Next Steps in Signaling (NSIS): Framework", RFC 4080, June 2005.

[RFC4081] Tschofenig, H. and D. Kroeselberg, "Security Threats for Next Steps in Signaling (NSIS)", RFC 4081, June 2005.

[RFC4213] Nordmark, E. and R. Gilligan, "Basic Transition Mechanisms for IPv6 Hosts and Routers", RFC 4213, October 2005.

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, December 2005.

[RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, December 2005.

Authors' Addresses

Charles Shen
Columbia University
Department of Computer Science
1214 Amsterdam Avenue, MC 0401
New York, NY 10027
USA

Phone: +1 212 854 3109
Email: charles@cs.columbia.edu

Henning Schulzrinne
Columbia University
Department of Computer Science
1214 Amsterdam Avenue, MC 0401
New York, NY 10027
USA

Phone: +1 212 939 7004
Email: hgs@cs.columbia.edu

Sung-Hyuck Lee
SAMSUNG Advanced Institute of Technology
San 14-1, Nongseo-ri, Giheung-eup
Yongin-si, Gyeonggi-do 449-712
KOREA

Phone: +82 31 280 9552
Email: starsu.lee@samsung.com

Jong Ho Bang
SAMSUNG Advanced Institute of Technology
San 14-1, Nongseo-ri, Giheung-eup
Yongin-si, Gyeonggi-do 449-712
KOREA

Phone: +82 31 280 9585
Email: jh0278.bang@samsung.com