Internet Engineering Task Force (IETF)
Request for Comments: 5979
Category: Experimental
ISSN: 2070-1721

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

March 2011

NSIS Operation over IP Tunnels

Abstract

NSIS Quality of Service (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. Therefore, the resulting tunnel segments could become the weakest QoS link and 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 that 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.

Status of This Memo

This document is not an Internet Standards Track specification; it is published for examination, experimental implementation, and evaluation.

This document defines an Experimental Protocol for the Internet community. This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Not all documents approved by the IESG are a candidate for any level of Internet Standard; see Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc5979.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

1. Introduction

IP tunneling [RFC1853] [RFC2003] 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 [RFC5944]), implement secure IP data delivery (e.g., in IPsec [RFC4301]), and help packet routing in IP networks of different characteristics (e.g., between IPv6 and IPv4 networks [RFC4213]). Section 3.1 summarizes a list of common IP tunneling protocols.

This document considers the situation when the packet being tunneled contains a Next Step In Signaling (NSIS) [RFC4080] packet. NSIS is an IP signaling architecture consisting of a Generic Internet Signaling Transport (GIST) [RFC5971] 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 [RFC5974] which provides functionalities that extend those of the earlier RSVP [RFC2205] signaling. In this document, the terms "NSIS" and "NSIS QoS" are used interchangeably.

Without additional efforts, NSIS signaling does not work within IP tunnel 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 endpoints that support this mechanism are called NSIS-tunnel-aware endpoints. There are two main operation modes. On one hand, if the tunnel already has preconfigured QoS sessions, the NSIS-tunnel-aware endpoints 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 preconfigured tunnel QoS sessions are available, the NSIS-tunnel-aware endpoints dynamically initiate and maintain tunnel QoS sessions that are then associated with the corresponding end-to-end QoS sessions. Note that whether or not the tunnel preconfigures QoS sessions, and which preconfigured tunnel QoS sessions a particular end-to-end QoS signaling request should be mapped to are policy issues that are 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 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 preconfigured QoS and dynamic QoS signaling are provided in Sections 5 and 6. Section 7 describes a method to automatically discover whether a tunnel endpoint 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], [RFC5971], and [RFC5974]. In addition, the following terms are used:

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

Tunnel IP Header: The IP header prepended to the original packet during encapsulation. It specifies the tunnel endpoints 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. An example of such header fields is the Encapsulation Security Payload (ESP) header for IPsec [RFC4301] tunneling mode.

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

Flow Identifier (Flow ID): The set of header fields that is used to identify a data flow. For example, it may include flow sender and receiver addresses, and 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 to the QoS signaling outside the tunnel segments. This document uses "end-to-end QoS signaling" and "end-to-end signaling" interchangeably.

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. This document uses "tunnel QoS signaling" and "tunnel signaling" interchangeably.

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

NSIS-Aware Tunnel Endpoint Node: A tunnel endpoint node that is also an NSIS node.

NSIS-Tunnel-Aware Endpoint Node: An NSIS-aware tunnel endpoint node that also supports the mechanism for NSIS operating over IP tunnels defined in this document.

3. Problem Statement

3.1. IP Tunneling Protocols

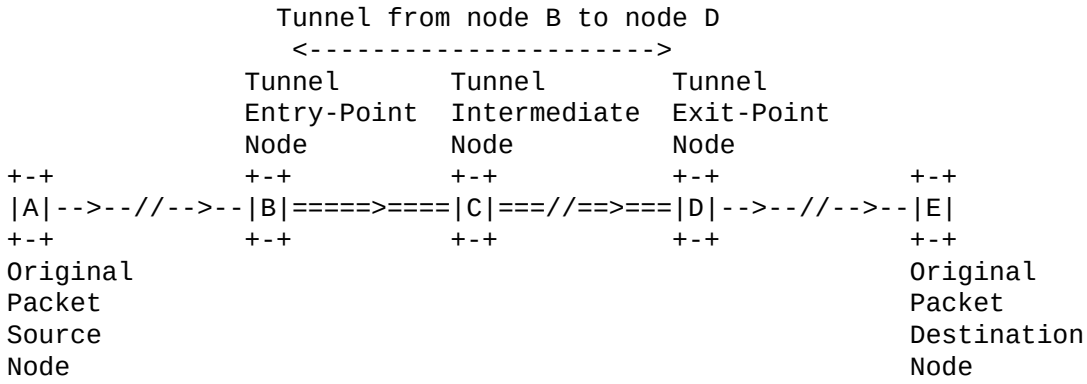

Figure 1: IP Tunnel

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

IP tunneling (Figure 1) is a technique for establishing a "virtual link" between two IP nodes for transmitting data packets as payloads of IP packets. 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 - the tunnel packet flow takes place in one direction between the IP tunnel entry-point and exit-point nodes. Bidirectional 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 the tunnel entry-point node as the IP source

address and the tunnel exit-point node as the IP destination address, causing the tunnel packet to be forwarded in the tunnel. The tunnel-specific header between the tunnel IP header and the original packet is optional, depending on the tunneling protocol in use.

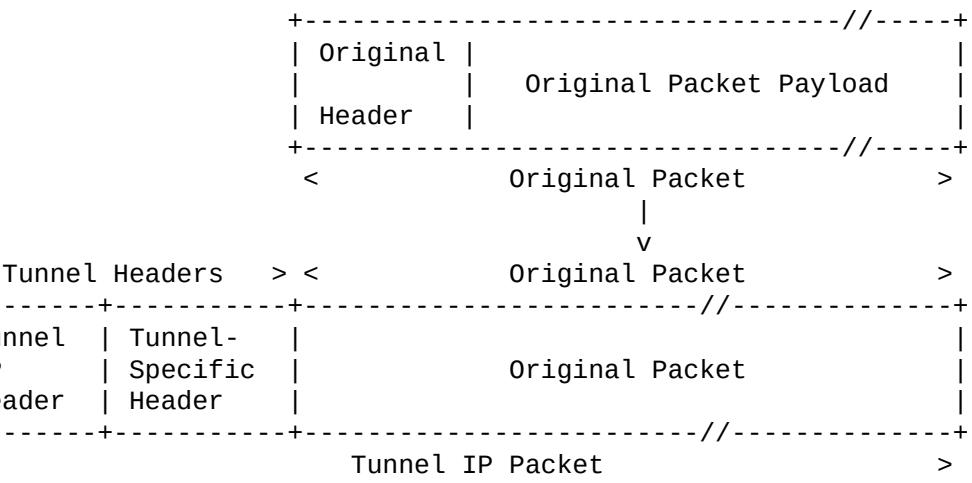

Figure 2: IP Tunnel Encapsulation

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 [RFC4301] [RFC4303]. Among these tunneling protocols, the tunnel headers in IPv4INIPv4, IPv6INIPv4, and IPv6GEN contain only a tunnel IP header, and no tunnel-specific header. 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 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 that are not NSIS QoS capable are not shown.

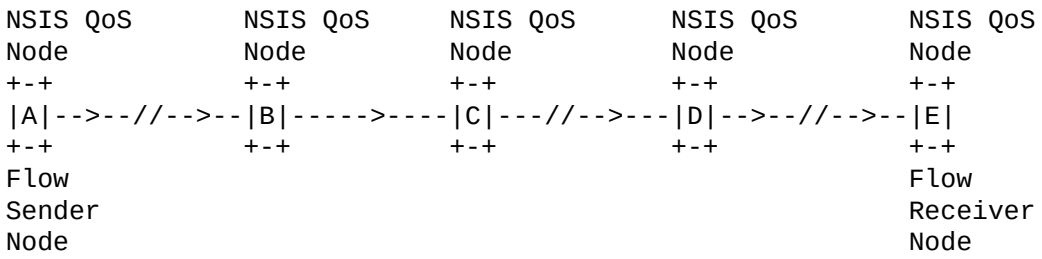

Figure 3: Example Scenario of NSIS QoS Signaling

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 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, nodes B, C, and D) is accomplished by a separate NSIS peer discovery process (not shown above; see [RFC5971]). The NSIS peer discovery messages contain special IP header and payload formats or include a Router Alert Option (RAO) [RFC2113] [RFC2711]. The special formats of NSIS discovery messages allow nodes B, C, and D to intercept the messages 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 that are not NSIS-aware simply forward all signaling messages, as they would any other IP packets that do not require additional handling.

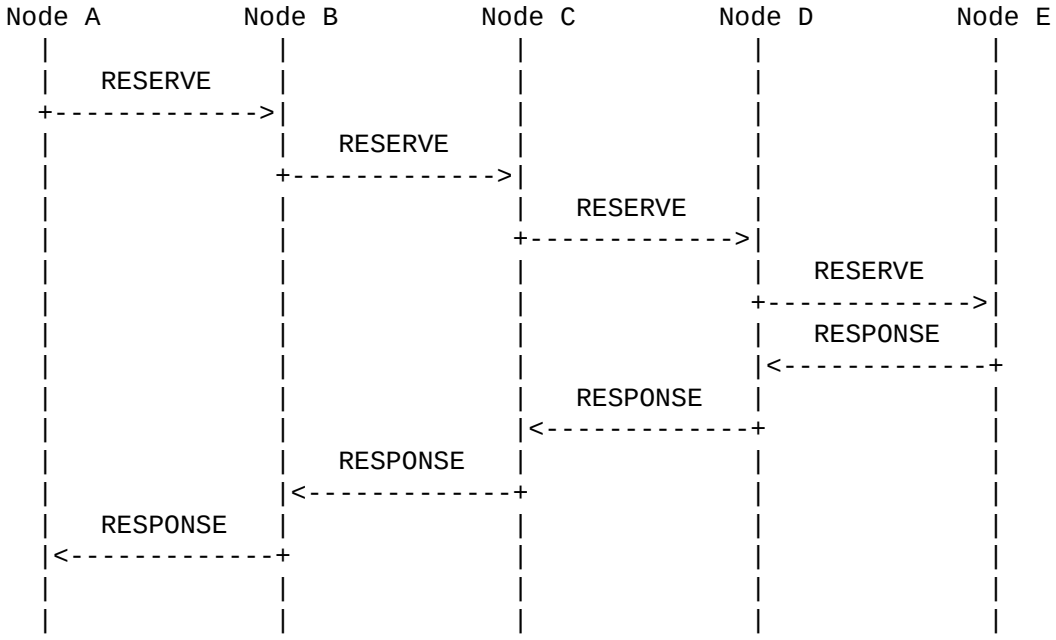

Figure 4: Sender-Initiated NSIS QoS Signaling

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, and protocol and port numbers.

Now consider Figure 5 where nodes B, C, and D are endpoints 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.

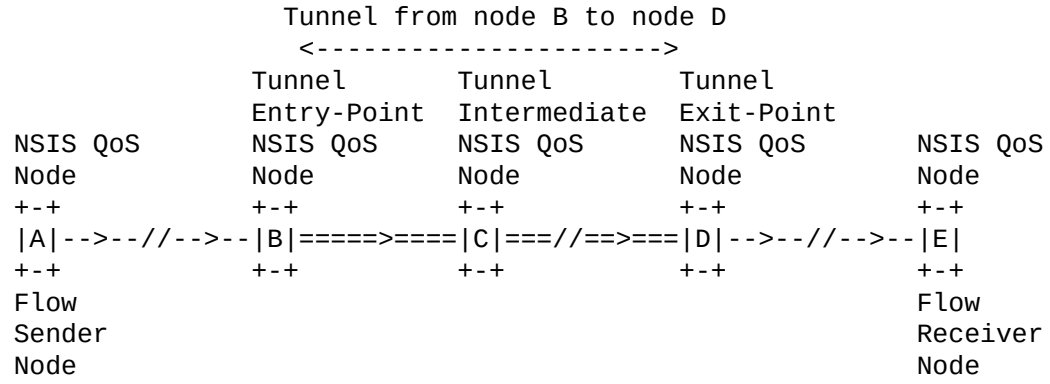

Figure 5: Example Scenario of NSIS QoS Signaling with IP Tunnel

In summary, an IP tunnel segment normally appears like a QoS-unaware virtual link. Since the best QoS of an end-to-end path is judged based on its weakest segment, we need a mechanism to extend NSIS into the IP tunnel segments, which should allow the tunnel intermediate nodes to intercept original NSIS signaling messages and classify original data flow packets in the presence of tunnel encapsulation.

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 preconfigured QoS sessions inside the tunnel. The mechanism should work for IP tunnels both with and without preconfigured tunnel QoS sessions.

- The mechanism should minimize the required upgrade to existing infrastructure in order to facilitate its deployment. Specifically, we should limit the necessary upgrade to the tunnel endpoints.

- The mechanism should provide a method for one NSIS-tunnel-aware endpoint to discover whether the other endpoint is also NSIS-tunnel-aware, when necessary.

- The mechanism should learn from the design experience of previous related 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 we only consider unidirectional flows, to accommodate flows in both directions of a tunnel, we require both tunnel entry-point and tunnel exit-point to be NSIS-tunnel-aware. An NSIS-tunnel-aware endpoint knows whether the other tunnel endpoint is NSIS-tunnel-aware either through preconfiguration or through an NSIS-tunnel capability discovery mechanism defined in Section 7.

Tunnel endpoints 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 QoS handling in the tunnel, an end-to-end QoS session is mapped to a tunnel QoS session, either preconfigured or dynamically created. The tunnel session uses a tunnel Flow ID based on information available in the tunnel headers, thus allowing tunnel intermediate nodes to classify flow packets correctly.

For tunnels that maintain preconfigured QoS sessions, upon receiving a request to reserve resources for an end-to-end session, the tunnel endpoint 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 preconfigured tunnel session depend on policy mechanisms outside the scope of this document.

For tunnels that do not maintain preconfigured QoS sessions, the NSIS-tunnel-aware endpoints 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 follows 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 preconfigured 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 preconfigured QoS sessions. This association serves two purposes, one for the signaling plane and the other for the data plane. For the signaling plane, the association enables the tunnel entry-point to coordinate necessary interactions between the end-to-end and the tunnel QoS sessions, such as QoS adjustment in sender-initiated reservations. For 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. For 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. For 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.

In tunnels with preconfigured QoS sessions, the tunnel exit-point may also learn about the mapping information between the corresponding tunnel and end-to-end QoS sessions through preconfiguration as well. In tunnels without preconfigured QoS sessions, the tunnel exit-point knows the mapping between the corresponding tunnel and end-to-end QoS sessions through the NSIS signaling process that creates the tunnel QoS sessions inside the tunnel, with the help of appropriate QoS NSLP session-binding and message-binding mechanisms.

One problem for NSIS operating over IP tunnels that 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

endpoint 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. This problem can be addressed in two ways, namely sequential mode and parallel mode. In sequential mode, end-to-end signaling pauses while 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 may necessitate renegotiation after the tunnel signaling outcome becomes available. In those cases, the signaling flow of the parallel mode also tends to be complicated. This document adopts a sequential mode approach for the two signaling sequences.

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 IDs 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. The source port number is dynamically chosen by the tunnel entry-point and conveyed to the tunnel exit-point. In these cases, it is especially important that the tunnel exit-point 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 endpoints need to be NSIS-tunnel-aware.

The above recommendations about choosing the tunnel Flow ID apply to dynamically created QoS tunnel sessions. For preconfigured 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 Preconfigured QoS Sessions

When tunnel QoS is managed by preconfigured QoS sessions, both the tunnel entry-point and tunnel exit-point need to be configured with information about the Flow ID of the tunnel QoS session. This allows the tunnel endpoints to correctly perform matching encapsulating and decapsulating operations. The procedures of NSIS operating over tunnels with preconfigured QoS sessions depend on whether the end-to-end NSIS signaling is sender-initiated or receiver-initiated. But in both cases, 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 the tunnel QoS configuration, determines whether and how this end-to-end session can be mapped to a preconfigured tunnel session. The mapping criteria are part of the preconfiguration 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 that 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 preconfiguration.

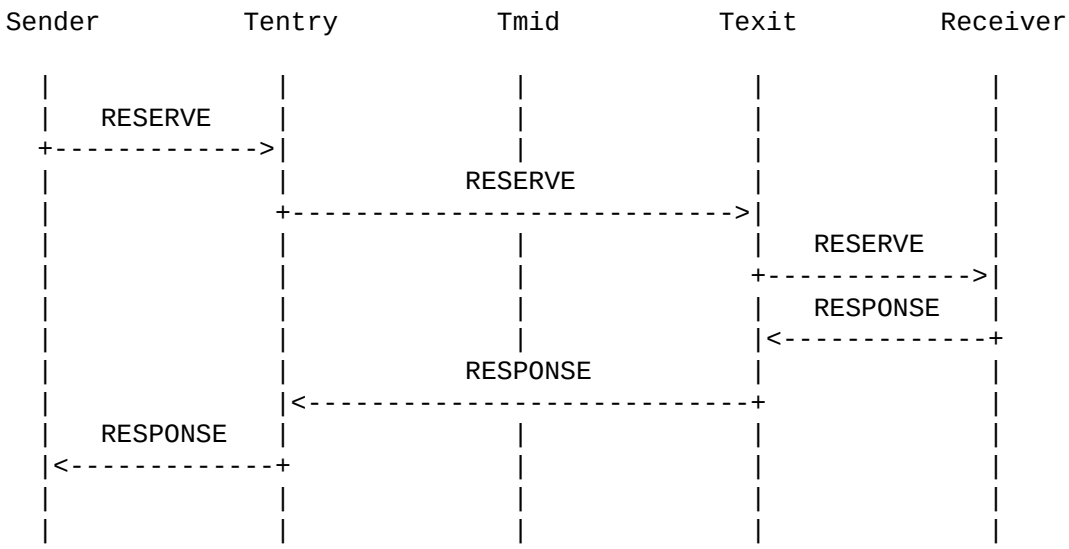

Figure 6: Sender-Initiated End-to-End Session with Preconfigured Tunnel QoS Sessions

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, then updates and tunnels the Query message to Texit. Texit decapsulates the QUERY message, processes it, and forwards it toward the receiver. 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 preconfigured tunnel session based on criteria outside the scope of this document. Then, Tentry forwards the RESERVE message towards the sender. 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 preconfiguration.

Since separate tunnel QoS signaling is not involved in preconfigured QoS tunnels, Figures 6 and 7 make the tunnel look like a single virtual link. The signaling path simply skips all tunnel intermediate nodes. However, both Tentry and Texit need to deploy the 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.

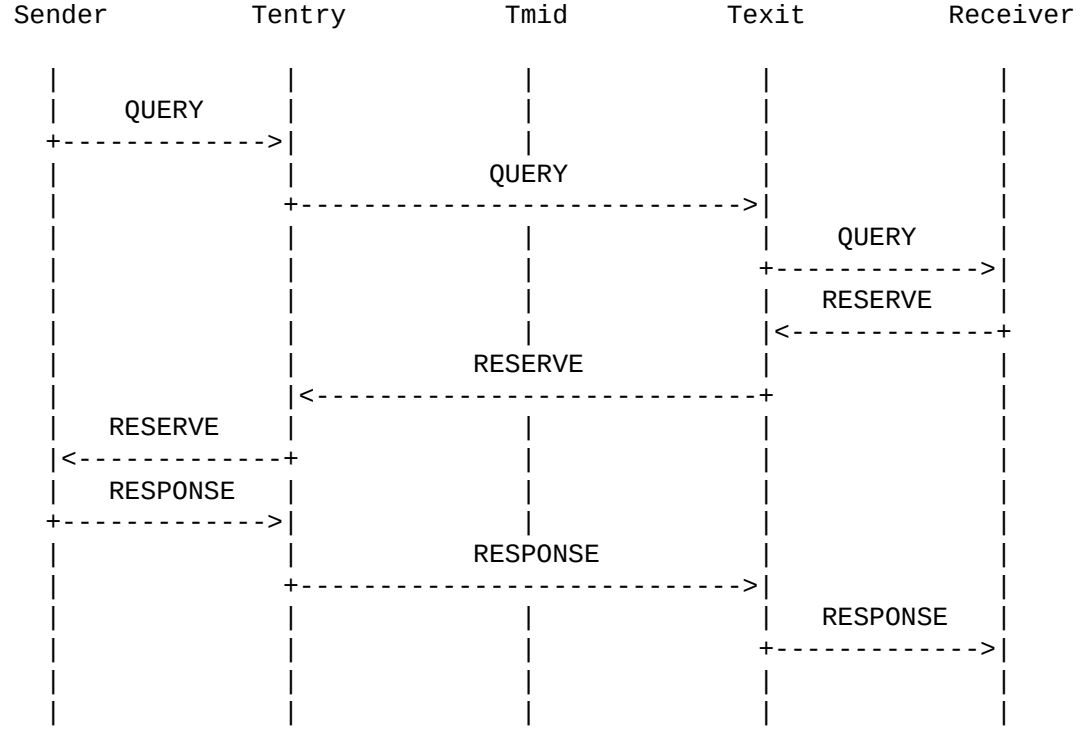

Figure 7: Receiver-Initiated End-to-End Session with Preconfigured Tunnel QoS Sessions

6. NSIS Operation over Tunnels with Dynamically Created QoS Sessions

When there are no preconfigured 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 endpoints 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 exit-point of the binding between the corresponding tunnel session and end-to-end session through the BOUND_SESSION_ID QoS NSLP message object. The reservation message dependencies between the tunnel session and end-to-end session are 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 the 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) that 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 that 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 flags of both the MSG-ID and BOUND-MSG-ID objects in the RESERVE' and RESERVE messages (2, 3) are 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 [RFC5974], as illustrated below.

The first possibility is shown in the example messaging flow of Figure 8, where the tunnel RESERVE' message (2), also known as 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), enqueues it and starts a MsgIDWait timer waiting for the end-to-end RESERVE message (3), also known as the bound signaling message in QoS NSLP message binding terms. The timer

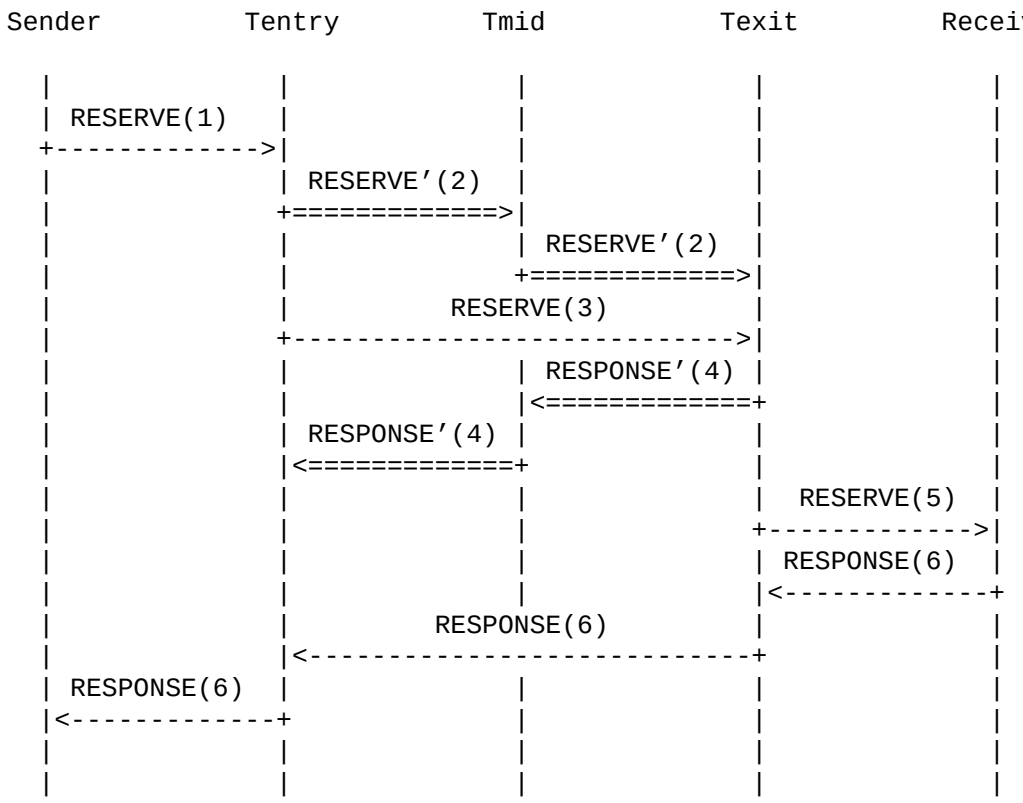

Figure 8: Sender-Initiated Reservation for Both End-to-End and Tunnel Signaling

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 enqueues 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 then 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. In that case, the MsgIDWait timer for the queued end-to-end RESERVE message will expire. Texit should then treat this situation as a local reservation failure, and according to [RFC5974], Texit as a stateful QoS NSLP should generate an end-to-end RESPONSE message indicating RESERVE error to the sender.

Once the end-to-end and the tunnel QoS session have both been successfully created and associated, the tunnel endpoints 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

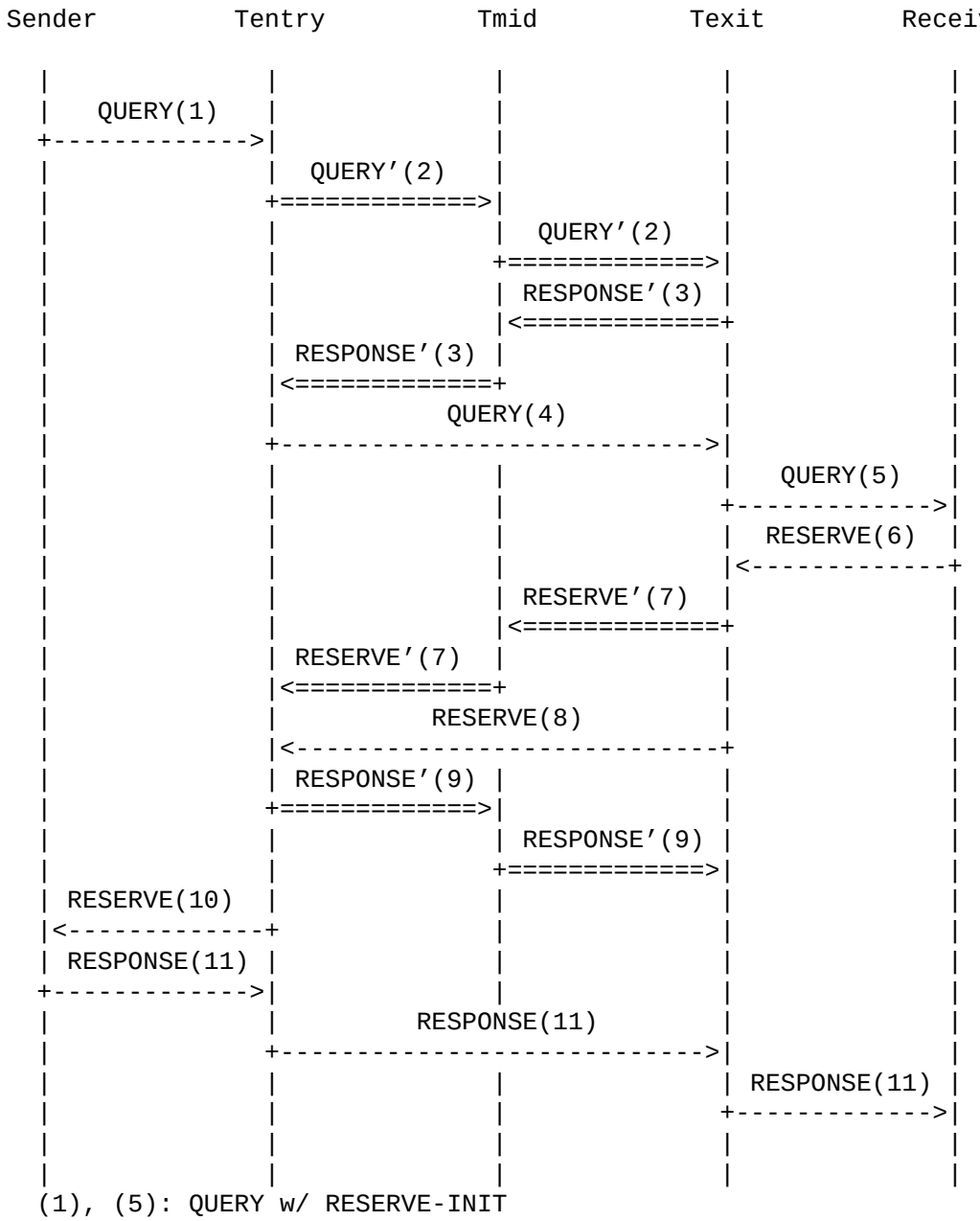

Figure 9: Receiver-Initiated Reservation for Both End-to-end and Tunnel Signaling

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 initiate 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, 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 flags of the MSG-ID and BOUND-MSG-ID objects in the RESERVE' and RESERVE messages (7,8) are SET, indicating a bidirectional binding.

At Tentry, the tunnel RESERVE' message (7) and the end-to-end RESERVE message (8) could arrive in either order. In a typical case shown in Figure 9, the tunnel RESERVE' message (7) arrives first. Tentry then 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 are similar to those of Texit in the 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 endpoints 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

The mechanism of NSIS operating over IP tunnels requires the coordination of both tunnel endpoints in tasks such as special encapsulation and decapsulation of data flow packets according to the chosen tunnel Flow ID, as well as the possible creation and adjustment of the end-to-end and tunnel QoS sessions. Therefore, one NSIS-tunnel-aware endpoint needs to know that the other tunnel endpoint is also NSIS-tunnel-aware before initiating this mechanism of NSIS operating over IP tunnels. In some cases, especially for IP tunnels with preconfigured QoS sessions, an NSIS-tunnel-aware endpoint can learn about whether the other tunnel endpoint is also NSIS-tunnel-aware through preconfiguration. In other cases where such preconfiguration is not available, the initiating NSIS-tunnel-aware endpoint may dynamically discover the other tunnel endpoint's capability through a QoS NSLP NODE_CAPABILITY_TUNNEL object defined in this section.

The NODE_CAPABILITY_TUNNEL object is a zero-length object with a standard NSLP object header as shown in Figure 10.

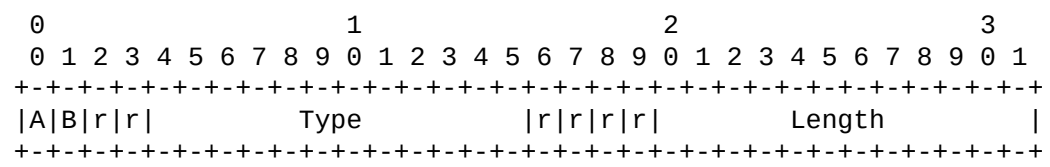

Figure 10: NODE_CAPABILITY_TUNNEL Object Format

Type: NODE_CAPABILITY_TUNNEL (0x015) from the shared NSLP object type space

Length: 0

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_TUNNEL object, the desired behavior is "Forward". That is, the object must be retained unchanged and forwarded as a result of message processing. This is satisfied by setting 'AB' to '10'.

The 'r' bit stands for 'reserved'.

The NODE_CAPABILITY_TUNNEL object is included in a tunnel QUERY' or RESERVE' message by a tunnel endpoint that needs to learn about the other endpoint's capability for NSIS tunnel handling. If the receiving tunnel endpoint is indeed NSIS-tunnel-aware, it recognizes this object and knows that the sending endpoint is NSIS-tunnel-aware. The receiving tunnel endpoint places the same object in a tunnel RESPONSE' message to inform the sending endpoint that it is also NSIS-tunnel-aware. The use of the NODE_CAPABILITY_TUNNEL object in the cases of sender-initiated reservation and receiver-initiated reservation are as follows.

First, assume that the end-to-end session is sender-initiated as in Figure 8, and the NSIS-tunnel-aware Tentry wants to discover the NSIS tunnel capability of Texit. After receiving the first end-to-end RESERVE message (1), Tentry inserts an RII object and a NODE_CAPABILITY_TUNNEL object into the tunnel RESERVE' message (2) and sends it to Texit. If Texit is NSIS-tunnel-aware, it learns from the NODE_CAPABILITY_TUNNEL object that Tentry is also NSIS-tunnel-aware and includes the same object into the tunnel RESPONSE' message (4) sent back to Tentry.

Second, assume that the end-to-end session is receiver-initiated as in Figure 9, and the NSIS-tunnel-aware Tentry wants to discover the NSIS tunnel capability of Texit. Upon receiving the first end-to-end QUERY message (1), Tentry inserts an RII object and a NODE_CAPABILITY_TUNNEL object in the tunnel QUERY' message (2) and sends it toward Texit. If Texit is NSIS-tunnel-aware, it learns from the NODE_CAPABILITY_TUNNEL object that Tentry is also NSIS-tunnel-aware and includes the same object tunnel RESPONSE' message (3) sent to Tentry.

8. IANA Considerations

This document defines a new object type called NODE_CAPABILITY_TUNNEL for QoS NSLP. Its Type value (0x015) has been assigned by IANA. The object format and the setting of the extensibility bits are defined in Section 7.

9. Security Considerations

This NSIS and IP tunnel interoperation mechanism has two IPsec-related security implications. First, NSIS messages may require per-hop processing within the IPsec tunnel, and that is potentially incompatible with IPsec. A similar problem exists for RSVP interacting with IPsec, when the Router Alert option is used (Appendix A.1 of RFC 4302 [RFC4302]). If this mechanism is indeed used for NSIS and IPsec tunnels, a so-called covert channel could exist where someone can create spurious NSIS signaling flows within the protected network in order to create signaling in the outside network, which then someone else is monitoring. For highly secure networks, this would be seen as a way to smuggle information out of the network, and therefore this channel will need to be rate-limited. A similar covert channel rate-limit problem exists for using Differentiated Services (DS) or Explicit Congestion Notification (ECN) fields with IPsec (Section 5.1.2 of RFC 4301 [RFC4301]).

Second, since the NSIS-tunnel-aware endpoint is responsible for adapting changes between the NSIS signaling both inside and outside the tunnel, there could be additional risks for an IPsec endpoint that is also an NSIS-tunnel-aware endpoint. For example, security vulnerability (e.g., buffer overflow) on the NSIS stack of that IPsec tunnel endpoint may be exposed to the unprotected outside network. Nevertheless, it should also be noted that if any node along the signaling path is compromised, the whole end-to-end QoS signaling could be affected, whether or not the end-to-end path includes an IPsec tunnel.

Several other documents discuss security issues for NSIS. General threats for NSIS can be found in [RFC4081]. Security considerations for NSIS NTLP and QoS NSLP are discussed in [RFC5971] and [RFC5974], respectively.

10. Acknowledgments

The authors would like to thank Roland Bless, Francis Dupont, Lars Eggert, Adrian Farrel, Russ Housley, Georgios Karagiannis, Jukka Manner, Martin Rohricht, Peter Saint-Andre, Martin Stiemerling, Hannes Tschofenig, and other members of the NSIS working group for comments. Thanks to Yaron Sheffer for pointing out the IPsec-related security considerations.

11. References

11.1. Normative References

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

[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.

[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.

[RFC5971] Schulzrinne, H. and R. Hancock, "GIST: General Internet Signalling Transport", RFC 5971, October 2010.

[RFC5974] Manner, J., Karagiannis, G., and A. McDonald, "NSIS Signaling Layer Protocol (NSLP) for Quality-of-Service Signaling", RFC 5974, October 2010.

11.2. Informative References

[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.

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

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

[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.

[RFC4302] Kent, S., "IP Authentication Header", RFC 4302, December 2005.

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

[RFC5944] Perkins, C., Ed., "IP Mobility Support for IPv4, Revised", RFC 5944, November 2010.

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
Convergence Technologies & Standardization Lab
Samsung Information System America, INC.
95 West Plumeria Drive
San Jose, CA 95134
USA

Phone: 1-408-544-5809
EMail: sung1.lee@samsung.com

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

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