IETF Next Steps in Signaling
Internet-Draft
Intended status: Informational
Expires: June 6, 2010

C. Shen
H. Schulzrinne
Columbia U.
S. Lee
J. Bang
Samsung AIT
December 3, 2009

NSIS Operation Over IP Tunnels
draft-ietf-nsis-tunnel-07.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 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 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 June 6, 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 NSIS signaling for end-to-end QoS reservation into IP tunnels. The NSIS-aware IP tunnel end-points that support this mechanism is 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 matched to is a policy issue 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 NSIS operation over IP tunnels mechanism defined in this document.

3. Problem Statement

3.1. IP Tunneling Protocols

The following definition of IP tunnel 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.

```
                  Tunnel from node B to node D
                   <---------------------->
                Tunnel       Tunnel        Tunnel
                Entry-Point  Intermediate  Exit-Point
                Node         Node          Node
 +-+            +-+          +-+           +-+            +-+
 |A|-->--//-->--|B|=====>====|C|===//==>===|D|-->--//-->--|E|
 +-+            +-+          +-+           +-+            +-+
 Original                                                 Original
 Packet                                                   Packet
 Source                                                   Destination
```

```
    Node                                                        Node
```

Figure 1: IP Tunnel

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.

```
                             +-----------------------------------//-----+
                             | Original |                               |
                             |          |   Original Packet Payload     |
                             | Header   |                               |
                             +-----------------------------------//-----+
                              <           Original Packet            >
                                                 |
                                                 v
         <Tunnel Headers>     <       Original Packet                >

        +---------+ - - - - - +-----------------------//-------------+
        | Tunnel  | Tunnel    |                                      |
        | IP      | Specific  |       Original Packet                |
        | Header  | Headers   |                                      |
        +---------+ - - - - - +-----------------------//-------------+
         <                      Tunnel IP Packet                     >
```

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.

```
NSIS QoS       NSIS QoS     NSIS QoS      NSIS QoS       NSIS QoS
Node           Node         Node          Node           Node
+-+            +-+          +-+           +-+            +-+
|A|-->--//-->--|B|----->----|C|---//-->---|D|-->--//-->--|E|
+-+            +-+          +-+           +-+            +-+
Flow                                                     Flow
Sender                                                   Receiver
Node                                                     Node
```

Figure 3: Example Scenario of Sender-initiated NSIS QoS Signaling

```
Node A     Node B     Node C     Node D     Node E
|          |          |          |          |
| RESERVE  |          |          |          |
+--------->|          |          |          |
|          | RESERVE  |          |          |
|          +--------->|          |          |
|          |          | RESERVE  |          |
|          |          +--------->|          |
|          |          |          | RESERVE  |
|          |          |          +--------->|
|          |          |          | RESPONSE |
|          |          |          |<---------+
```

```
       |          |          | RESPONSE |          |
       |          |          |<---------+          |
       |          | RESPONSE |          |          |
       |          |<---------+          |          |
       | RESPONSE |          |          |          |
       |<---------+          |          |          |
       |          |          |          |          |
       |          |          |          |          |
```

Figure 4: Example Scenario of Sender-initiated NSIS QoS Reservation

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.

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 packets classification and 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 might still be able to intercept and process NSIS peer discovery messages if it recognizes them before performing tunnel encapsulation; node D might be able to 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.

```
                   Tunnel from node B to node D
                    <----------------------->
                 Tunnel       Tunnel        Tunnel
                 Entry-Point  Intermediate  Exit-Point
  NSIS QoS       NSIS QoS     NSIS QoS      NSIS QoS       NSIS QoS
  Node           Node         Node          Node           Node
  +-+            +-+          +-+           +-+            +-+
  |A|-->--//-->--|B|=====>====|C|===//==>===|D|-->--//-->--|E|
  +-+            +-+          +-+           +-+            +-+
  Flow                                                     Flow
  Sender                                                   Receiver
  Node                                                     Node
```

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.

- o The mechanism should work with all common IP tunneling protocols listed in Section 3.1.
- o 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.
- o 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 needs 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.

- o 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.
- o 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 uni-directional, 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 uses 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 within tunnel QoS.

For tunnels that maintain pre-configured QoS sessions, upon receiving a request to reserve resources for an end-to-end session, an NSIS-tunnel-aware entry-point will try to map 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 initiate and manage a corresponding tunnel QoS session for each end-to-end session. 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.

An important characteristics 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 tunnel-intermediate node flow packet classification problem 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.

Ultimately QoS handling needs to be enforced in the data plane. To achieve that, NSIS-tunnel-aware entry-point nodes not only encapsulate data flow packets according to the specific tunnel protocol, but also insert any necessary header fields according to the chosen tunnel flow ID. All those header fields are visible to tunnel intermediate nodes. Tunnel intermediate nodes then classify those data flow packets and apply appropriate QoS treatment. At the tunnel exit-point, the data flow packet is decapsulated accordingly and forwarded as usual.

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.

- o For IPv6 tunneling protocols (IPv6GEN), the tunnel flow ID consists of the tunnel entry-point IPv6 address and the tunnel exit-point IPv6 addresses plus a unique IPv6 flow label [RFC3697].
- o For IPSEC tunnel mode (IPSEC), the tunnel flow ID contains tunnel the entry-point IP address and the tunnel exit-point IP address plus the Security Parameter Index (SPI).
- o 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 this case, 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. This requires both tunnel end-points 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

```
 Sender    Tentry      Tmid      Texit     Receiver

   |          |          |          |          |
   | RESERVE  |          |          |          |
   +--------->|          |          |          |
   |          |       RESERVE       |          |
   |          +-------------------->|          |
   |          |          |          | RESERVE  |
   |          |          |          +--------->|
   |          |          |          | RESPONSE |
   |          |          |          |<---------+
   |          |       RESPONSE      |          |
   |          |<--------------------+          |
   | RESPONSE |          |          |          |
   |<---------+          |          |          |
   |          |          |          |          |
   |          |          |          |          |
```

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

Tunnel QoS Sessions

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 indicates success or failure of the mapping between the end-to-end session and a tunnel session in the RESERVE message being tunneled to Texit. The signaling proceeds as usual until a RESPONSE message arrives at Texit, Tentry and finally the sender. If the RESPONSE message that Tentry received 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.

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 the Query message accordingly, and tunnel 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 reaches 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, again based on an algorithm outside the scope of this document, and then indicates the outcome in the outgoing RESERVE message. The signaling continues until a RESPONSE message arrives at Tentry, Texit and finally the receiver. If the RESPONSE message that Tentry received 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.

```
     Sender    Tentry      Tmid      Texit     Receiver

       |          |          |          |          |
       |  QUERY   |          |          |          |
       +--------->|          |          |          |
       |          |        QUERY        |          |
       |          +-------------------->|          |
```

```
          |          |          |          |  QUERY   |
          |          |          |          +--------->|
          |          |          |          | RESERVE  |
          |          |          |          |<---------+
          |          |       RESERVE       |          |
          |          |<--------------------+          |
          |  RESERVE |          |          |          |
          |<---------+          |          |          |
          | RESPONSE |          |          |          |
          +--------->|          |          |          |
          |          |       RESPONSE      |          |
          |          +-------------------->|          |
          |          |          |          | RESPONSE |
          |          |          |          +--------->|
          |          |          |          |          |
          |          |          |          |          |
```

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 also distinguish themselves using the tunnel header fields, which solves the problem for tunnel intermediate NSIS nodes to intercept signaling messages as described in Section 3.2.

When tunnel end-points dynamically create tunnel QoS sessions, they need to determine whether the tunnel NSIS signaling is sender-initiated or receiver-initiated. In order to reduce complexity, we decide that the end-to-end session and the tunnel session should share the same signaling initiation mode. Since the end-to-end

session is the originator that causes the establishment of the tunnel session, 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.

6.1. Conveying Tunnel Flow ID Between Tunnel End-points

Depending on what type of tunnel flow ID in Section 4.3 is used, dynamically created tunnel QoS sessions may involve packet encapsulation other than standard tunneling mechanisms. For example, when a particular tunnel session inserts an additional UDP header for its flow ID, that additional UDP header added by the NSIS-tunnel-aware entry-point is not part of the standard tunnel encapsulation process. In these cases, the NSIS-tunnel-aware exit-point needs to be notified about the special encapsulation so it can perform correct decapsulation by removing both the standard tunnel header and the additional UDP header. The NSIS-tunnel-aware exit-point can learn about the tunnel flow ID through the NSIS signaling process inside the tunnel that creates the tunnel QoS sessions. However, it needs to further understand that the specific tunnel QoS session is associated with an end-to-end session and therefore needs to be decapsulated based on the corresponding tunnel flow ID. This is achieved by using one of the objects in QoS NSLP messages called BOUND_SESSION_ID [I-D.ietf-nsis-qos-nslp]. When used for NSIS signaling over tunnels, the BOUND_SESSION_ID object carries the session ID of the corresponding tunnel session and a Binding Code of value 0x01 indicating tunnel handling. The NSIS-tunnel-aware entry-point includes this tunnel binding object in appropriate end-to-end signaling messages. The NSIS-tunnel-aware exit-point that receives this tunnel session binding object then records the association between the tunnel QoS session and the end-to-end session. With this association, the NSIS-tunnel-aware exit-point can perform correct decapsulation for the data packets belonging to the end-to-end session.

6.2. Sender-initiated Reservation

```
  Sender    Tentry      Tmid      Texit     Receiver

    |          |          |          |          |
    | RESERVE  |          |          |          |
    +--------->|          |          |          |
    |          | RESERVE' |          |          |
    |          +=========>|          |          |
```

```
       |          |          | RESERVE' |          |
       |          |          +=========>|          |
       |          |       RESERVE       |          |
       |          | w/ BOUND_SESSION_ID |          |
       |          +-------------------->|          |
       |          |          | RESPONSE'|          |
       |          |          |<=========+          |
       |          |          |          | RESERVE  |
       |          |          |          +--------->|
       |          | RESPONSE'|          |          |
       |          |<=========+          |          |
       |          |          |          | RESPONSE |
       |          |          |          |<---------+
       |          |       RESPONSE      |          |
       |          |<--------------------+          |
       | RESPONSE |          |          |          |
       |<---------+          |          |          |
       |          |          |          |          |
       |          |          |          |          |
```

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

Figure 8 shows the 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 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 matching the request of the end-to-end session towards Texit to reserve tunnel resources. Tentry also appends a tunnel BOUND_SESSION_ID object to the original RESERVE message containing the session ID of the tunnel session and sends it towards Texit using normal tunnel encapsulation.

The tunnel RESERVE' message is processed hop-by-hop inside the tunnel for the flow identified by the chosen tunnel flow ID. When Texit receives the tunnel RESERVE' message, a reservation state for the tunnel session is created. Texit also sends a tunnel RESPONSE' message to Tentry. On the other hand, the end-to-end RESERVE message passes through the tunnel intermediate nodes (Tmid) just like other tunneled packets. When Texit receives the end-to-end RESERVE message, it notices the binding of a tunnel session and updates the end-to-end RESERVE message based on the result of the tunnel session

reservation. Then Texit removes the tunnel BOUND_SESSION_ID object and forwards the end-to-end RESERVE message further 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.

6.3. Receiver-initiated Reservation

Figure 9 shows the messaging sequence of how NSIS signaling operates over IP tunnels when both end-to-end and tunnel sessions are receiver-initiated. Tentry first receives an end-to-end QUERY message from the sender, it chooses the tunnel flow ID, creates the tunnel session and sends a tunnel QUERY' message matching the request of the end-to-end session towards Texit. Tentry also appends a tunnel BOUND_SESSION_ID object containing the session ID of the tunnel session to the original QUERY message and sends it toward Texit using normal tunnel encapsulation.

The tunnel QUERY' message is processed hop-by-hop inside the tunnel for the flow identified by the chosen tunnel flow ID. When Texit receives the tunnel QUERY' message, it creates a reservation state for the tunnel session without sending a tunnel RESERVE' message immediately.

The end-to-end QUERY message passes along tunnel intermediate nodes like other tunneled packets. When Texit receives the end-to-end QUERY message, it notices the binding of a tunnel session and checks the state information for the tunnel session. When the tunnel session state is available, Texit updates the end-to-end QUERY message with information from tunnel session state (e.g., QoS availability in the tunnel), removes the tunnel BOUND_SESSION_ID object and forwards the end-to-end QUERY message further along the path.

```
  Sender    Tentry      Tmid      Texit     Receiver

    |          |          |          |          |
    |  QUERY   |          |          |          |
    +--------->|          |          |          |
    |          |  QUERY'  |          |          |
    |          +=========>|          |          |
    |          |          |  QUERY'  |          |
    |          |          +=========>|          |
    |          |        QUERY        |          |
    |          | w/ BOUND_SESSION_ID |          |
    |          +-------------------->|          |
    |          |          |          |  QUERY   |
```

```
       |          |          |          +--------->|
       |          |          |          | RESERVE  |
       |          |          |          |<---------+
       |          |          | RESERVE' |          |
       |          |          |<=========+          |
       |          | RESERVE' |          |          |
       |          |<=========+          |          |
       |          |       RESERVE       |          |
       |          |<--------------------+          |
       |  RESERVE |          |          |          |
       |<---------+          |          |          |
       |          | RESPONSE'|          |          |
       |          +=========>|          |          |
       |          |          | RESPONSE'|          |
       |          |          +=========>|          |
       | RESPONSE |          |          |          |
       +--------->|          |          |          |
       |          |       RESPONSE      |          |
       |          +-------------------->|          |
       |          |          |          | RESPONSE |
       |          |          |          +--------->|
       |          |          |          |          |
       |          |          |          |          |
```

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

When Texit receives the first end-to-end RESERVE message issued by the receiver, it finds the reservation state of the tunnel session and triggers a tunnel RESERVE' message for that session. Meanwhile Texit forwards the end-to-end RESERVE message towards Tentry. When Tentry receives the tunnel RESERVE' message, it creates the reservation state for the tunnel session and sends a tunnel RESPONSE' message back to Texit. When Tentry receives the end-to-end RESERVE message, it updates the end-to-end RESERVE message with the result of the corresponding tunnel session reservation. Then Tentry forwards the end-to-end RESERVE message upstream toward the sender. When the sender receives the end-to-end RESERVE message, it sends an end-to-end RESPONSE message back to the receiver.

6.4. Timing Issues of End-to-end and Tunnel Signaling

NSIS operation over tunnels that dynamically create QoS sessions involves two correlated but separate signaling sessions, the end-to-end session and the corresponding tunnel session. 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. Our design in Section 6 adopts a hybrid mode in order to strike a balance between speed and efficiency. The rule is that end-to-end signaling should wait for tunnel signaling if it is expecting information that is required to initiate the end-to-end reservation; but end-to-end signaling does not need to wait for tunnel signaling if the end-to-end QoS reservation is already in progress.

An example of end-to-end signaling having to wait for tunnel signaling outcome is the end-to-end QUERY message from Texit to the receiver in Figure 9. That Query message needs to wait for the QUERY' message about the tunnel QoS information and then be forwarded toward the receiver. The tunnel QoS information learned from the QUERY' message supplies important information needed to initiate the end-to-end reservation. This timing rule ensures that the first end-to-end RESERVE message originated from the receiver will have the correct view of the whole path, both inside and outside the tunnel.

An example of end-to-end signaling not having to wait for tunnel signaling outcome is a RESERVE message which is about to leave the tunnel, such as the RESERVE message from Texit to the receiver in Figure 8 and the RESERVE message from Tentry to sender in Figure 9. Those RESERVE messages can be forwarded immediately even if the tunnel QoS reservation outcome is still unknown. However, the tunnel end-point should try to learn about results of the corresponding tunnel session reservation either through proactive polling after a specific amount of time, or before a refresh message is scheduled to be sent. Once the tunnel session reservation information is available, the tunnel end-point should immediately trigger an end-to-end RESERVE message subject to the results of the tunnel reservation. That is, if the tunnel reservation is successful, the message would be a normal RESERVE refresh; otherwise, the RESERVE message should indicate that some error has occurred in the reservation path.

7. NSIS-Tunnel Signaling Capability Discovery

As discussed in Section 6.1, 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.

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_CHAR Object Format

A GIST NODE_CHAR object is defined to discover the NSIS-tunnel handling capability of a tunnel end-point. The format of the NODE_CHAR 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.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|A|B|r|r|         Type          |r|r|r|r|        Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|T|                           Reserved                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 10: NODE_CHAR Object Format

Type: NODE_CHAR

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_CHAR 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_CHAR Object

The NODE_CHAR 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_CHAR 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 6.2) 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_CHAR 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_CHAR object. When Tentry receives the RESPONSE message, it knows whether Texit is NSIS-tunnel-aware by checking the existence of the NODE_CHAR object and its T bit. If both tunnel endpoints are NSIS-tunnel-aware, the rest of the procedures follows those defined in Section 6.2.

Second, assume that both end-to-end and tunnel sessions are receiver-initiated (Section 6.3) 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_CHAR 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_CHAR object that Tentry is also NSIS-tunnel-aware. In the later end-to-end RESPONSE message to this QUERY message, Texit includes a NODE_CHAR 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 6.3.

8. IANA Considerations

This document defines a new object type called NODE_CHAR 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 Tannes Tschofenig 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-17 (work in progress), October 2009.

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-13 (work in progress), July 2009.

[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-20 (work in progress), November 2008.

[I-D.ietf-nsis-qspec]
Bader, A., Ash, G., Kappler, C., and D. Oran, "QoS NSLP QSPEC Template", draft-ietf-nsis-qspec-22 (work in progress), November 2009.

[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