IETF Next Steps in Signaling
Internet-Draft
Intended status: Informational
Expires: May 7, 2009

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

November 3, 2008

NSIS Operation Over IP Tunnels
draft-ietf-nsis-tunnel-05.txt

Status of this Memo

Abstract

This draft presents an NSIS operation over IP tunnel scheme using QoS NSLP as the NSIS signaling application. Both sender-initiated and receiver-initiated NSIS signaling modes are discussed. The scheme creates individual or aggregate tunnel sessions for end-to-end sessions traversing the tunnel. Packets belonging to qualified end-to-end sessions are mapped to corresponding tunnel sessions and assigned special flow IDs to be distinguished from the rest of the tunnel traffic. Tunnel endpoints keep the association of the end-to-end and tunnel session mapping, so that adjustment in one session can

be reflected in the other.

Table of Contents

1. Introduction

When IP tunnel mechanism is used to transfer signaling messages, e.g., NSIS messages, the signaling messages usually become hidden inside the tunnel and are not known to the tunnel intermediate nodes. In other words, the IP tunnel behaves as a logical link that does not support signaling in the end-to-end path. If true end-to-end signaling support is desired, there needs to be a scheme to enable signaling at the tunnel segment of the end-to-end signaling path. This draft describes such a scheme for NSIS operation over IP tunnels. We assume QoS NSLP as the NSIS signaling application.

1.1. IP Tunneling Mechanisms and Tunnel Signaling Capability

There are a number of common IP tunneling mechanisms, such as Generic Routing Encapsulation (GRE) [4][15], Generic Routing Encapsulation over IPv4 Networks (GREIP4) [5] , IP Encapsulation within IP (IP4INIP4) [7], Minimal Encapsulation within IP (MINENC) [8], Generic Packet Tunneling in IPv6 Specification (IP6GEN) [11], IPv6 over IPv4 tunneling (IP6INIP4) [9], IPSEC tunneling mode [19][10]. These mechanisms can be differentiated according to the format of the tunnel encapsulation header. IP4INIP4, IP6INIP4 and IP6GENIP4 can be seen as normal IP in IP tunnel encapsulation because their tunnel encapsulation headers are in the form of a standard IP header. All GRE-related IP tunneling (GRE,GREIP4), MINENC and IPSEC tunneling mode can be seen as modified IP in IP tunnel encapsulation because the tunnel encapsulation header contains additional information fields besides a standard IP header. The additional information fields are the GRE header for GRE and GREIP4, the minimum encapsulation header for MINENC and the Encapsulation Security Payload (ESP) header for IPSEC tunneling mode.

By default any end-to-end signaling messages arriving at the tunnel endpoint will be encapsulated the same way as data packets. Tunnel intermediate nodes do not identify them as signaling messages. A signaling-aware IP tunnel can participate in a signaling network in various ways. Prior work on RSVP operation over IP tunnels (RSVP-TUNNEL) [16] identifies two types of QoS-aware tunnels: a tunnel that can promise some overall level of resources but cannot allocate resources specifically to individual data flows, or a tunnel that can make reservations for individual end-to-end data flows. This classification leads to two types of tunnel signaling sessions: individual tunnel signaling sessions that are created and torn down dynamically as end-to-end session come and go, and aggregate tunnel sessions that can either be fixed, or dynamically adjusted as the actually used session resources increase or decrease. Aggregate tunnel sessions are usually pre-configured but can also be dynamically created. A tunnel may contain only individual tunnel

sessions or aggregate tunnel sessions or both.

1.2. NSIS Tunnel Operation Overview

This NSIS operation over IP tunnel scheme is designed to work with most, if not all, existing IP in IP tunneling mechanisms. The scheme requires the tunnel endpoints to support specific tunnel related functionalities. Such tunnel endpoints are called NSIS-tunnel capable endpoints. Tunnel intermediate nodes do not need to have special knowledge about this scheme. When tunnel endpoints are NSIS-tunnel capable, this scheme enables the proper signaling initiation and adjustment inside the tunnel to match the requests of the corresponding end-to-end sessions. In cases where tunnel session signaling status is uncertain or not successful, the end-to-end session will be notified about the existence of possible NSIS-unaware links in the end-to-end path.

The overall design of this NSIS operation over IP tunnel scheme is conceptually similar to RSVP-TUNNEL [16]. However, the details of the scheme address all the important differences of NSIS from RSVP. For example,

- NSIS is based on a two-layer architecture, namely a signaling transport layer and a signaling application layer. It is designed as a generic framework to accommodate various signaling application needs. The basic RSVP protocol does not have a layer split and is only for QoS signaling.
- NSIS QoS NSLP allows both sender-initiated and receiver-initiated reservations; RSVP only supports receiver-initiated reservations.
- NSIS deals only with unicast; RSVP also supports multicast.
- NSIS integrates new features, such as the Session ID, to facilitate operation in specific environments (e.g. mobility and multi-homing).

From a high level point of view, there are two main issues in a signaling operation over IP tunnel scheme. First, how packet classification is performed inside the tunnel. Second, how signaling is carried out inside the tunnel.

Packets belonging to qualified data flows need to be recognized by tunnel intermediate nodes to receive special treatment. Packet classification is traditionally based on flow ID. After a typical IP-in-IP tunnel encapsulation, packets from different flows appear as having the same flow ID which usually consists of the Tunnel Entry (Tentry) address and Tunnel Exit (Texit) address. Therefore, the flow ID for a signaled flow needs to contain further demultiplexing information to make it distinguishable from non-signaled flows and from other signaled flows.

The special flow ID for signaled flows inside the tunnel needs to be carried in tunnel signaling messages, along with tunnel adjusted QoS parameters, to set up or modify the state information in tunnel intermediate nodes. This process creates separate tunnel signaling sessions between the tunnel endpoints. In most cases, it is necessary to maintain the state association between an end-to-end session and its corresponding tunnel session so that any change to one session may be reflected in the other.

2. Protocol Design Decisions

2.1. Flow Packet Classification over the Tunnel

A flow can be an individual flow, or an aggregate flow consisting of multiple individual flows.

For individual flows that need to be distinguished from each other inside the tunnel, by default an additional UDP header is inserted during the tunnel IP encapsulation. The resulting UDP encapsulated flow will then use the Tentry IP address, Texit IP address along with the source port number in the additional UDP header as flow ID inside the tunnel. To ensure this mechanism work, the Tentry doing UDP encapsulation needs to know the Texit has the corresponding UDP decapsulation capability. Tentry knows the capability of the Text either by pre-configuration or through tunnel signaling capability discovery defined in Section 5.

Not all individual flows must use the UDP encapsulation to form the tunnel flow ID. In particular, for an IPv6 flow with unique flow label [6], the tunnel signaling can use the Tentry and Texit IP addresses plus the IPv6 flow label as the flow ID; for an IPSEC flow with Security Parameter Index (SPI), the tunnel signaling can use the Tentry and Texit IP addresses plus the SPI as the tunnel flow ID.

For aggregate flows, the tunnel signaling can still use UDP encapsulation for flow ID; when the DiffServ Code Point (DSCP) field is in use, the aggregate tunnel flow ID can also be Tentry and Texit IP addresses plus the DSCP value; when additional interfaces are available, the tunnel signaling may also use the IP address of an additional interface at Tentry plus the IP address of the Texit as the aggregate flow ID.

The choice of the tunnel flow ID format is made by the tunnel signaling initiator and then conveyed to the other end of the tunnel as part of Message Routing Information (MRI, see [2]) in regular NSIS signaling messages.

2.2. Tunnel Signaling and its Association with End-to-end Signaling

Tunnel signaling messages contain tunnel specific parameters such as tunnel MRI and tunnel adjusted QoS parameters. But in general, the formats of tunnel signaling messages are the same as end-to-end signaling messages. Tunnel signaling is carried out according to the same signaling rules as for end-to-end signaling. The main challenge is, therefore, the interaction between tunnel signaling and end-to-end signaling. The interaction is achieved by special functionalities supported in the NSIS-tunnel aware tunnel endpoints. These special functionalities include assigning tunnel flow IDs, creating tunnel session association, notifying the other endpoint about tunnel association, adjusting one session based on change of the other session, encapsulating (decapsulating) packets according to the chosen tunnel flow ID at Tentry (Texit), and etc. In most cases, we expect to have bi-directional tunnels, where both tunnel endpoints are NSIS-tunnel aware.

When both Tentry and Texit are NSIS-tunnel aware, the endpoint that creates the tunnel session notifies the other endpoint of the association between the end-to-end and tunnel session using the QoS NSLP BOUND_SESSION_ID object with a Binding Code indicating tunnel handling as the reason for binding. In the rest of this document, we refer to a BOUND_SESSION_ID object with its tunnel Binding Code set as a tunnel BOUND_SESSION_ID object or a tunnel binding object. This tunnel binding object is carried in the end-to-end signaling messages and contains the session ID of the corresponding tunnel session. NSIS-tunnel aware endpoints that receive this tunnel BOUND_SESSION_ID object should perform tunnel related procedures and then remove it from any end-to-end signaling messages sent out of the tunnel.

3. Protocol Operation with Dynamically Created Tunnel Sessions

The tunnel session corresponding to the end-to-end session can be dynamically created or pre-configured, the former case is much more complicated. It is a policy decision over which method should be used. We discuss the dynamically created tunnel session case in this section and then the pre-configured tunnel session case in the next.

3.1. Operation Scenarios

When tunnel sessions are dynamically created for end-to-end sessions, there could be four scenarios based on the sender-initiated and receiver-initiated reservation modes of NSIS QoS NSLP:

- End-to-end session is sender-initiated; tunnel session is sender-initiated.

- o End-to-end session is receiver-initiated; tunnel session is receiver-initiated.
- o End-to-end session is sender-initiated; tunnel session is receiver-initiated.
- o End-to-end session is receiver-initiated; tunnel session is sender-initiated.

Whether sender-initiated or receiver-initiated reservation should be used is determined by the signaling initiator. When both the end-to-end session and the tunnel session are concerned, this decision will need to be made twice. In order to reduce complexity, we decide that both the end-to-end session and the tunnel session should use the same initiation mode. Since the end-to-end session is the originator that causes the establishment of the tunnel session, we use the decision made by the end-to-end session as a reference. Specifically, when the end-to-end session is sender-initiated, then the tunnel session should be sender-initiated too. If the end-to-end session is receiver-initiated, then the tunnel session should be receiver-initiated too.

In the following we describe the typical NSIS end-to-end and tunnel signaling interaction process during the tunnel setup phase in each of the two recommended scenarios. The end-to-end QoS flow is assumed to be one that qualifies an individual dynamic tunnel session.

3.1.1. Sender-initiated Reservation for both End-to-end and Tunnel Signaling

This scenario assumes both end-to-end and tunnel sessions are sender-initiated. Figure 1 shows the messaging flow of NSIS operation over IP tunnels in this case. Tunnel signaling messages are distinguished from end-to-end messages by a prime symbol after the message name. Tnode denotes an intermediate tunnel node that participates in tunnel signaling. The sender first sends an end-to-end RESERVE message which arrives at Tentry. If Tentry supports tunnel signaling and determines that an individual tunnel session needs to be established for the end-to-end session, it chooses the tunnel flow ID, creates the tunnel session and associates the end-to-end session with the tunnel session. It then sends a tunnel RESERVE' message matching the requests of the end-to-end session towards the Texit to reserve tunnel resources. Tentry also appends to the original RESERVE message a tunnel BOUND_SESSION_ID object 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 will be created. Texit may also send a tunnel RESPONSE' message to Tentry. On the other hand, the end-to-end RESERVE message passes through the tunnel intermediate nodes just like any 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 end-to-end reservation finishes, the receiver may send an end-to-end RESPONSE back to the sender.

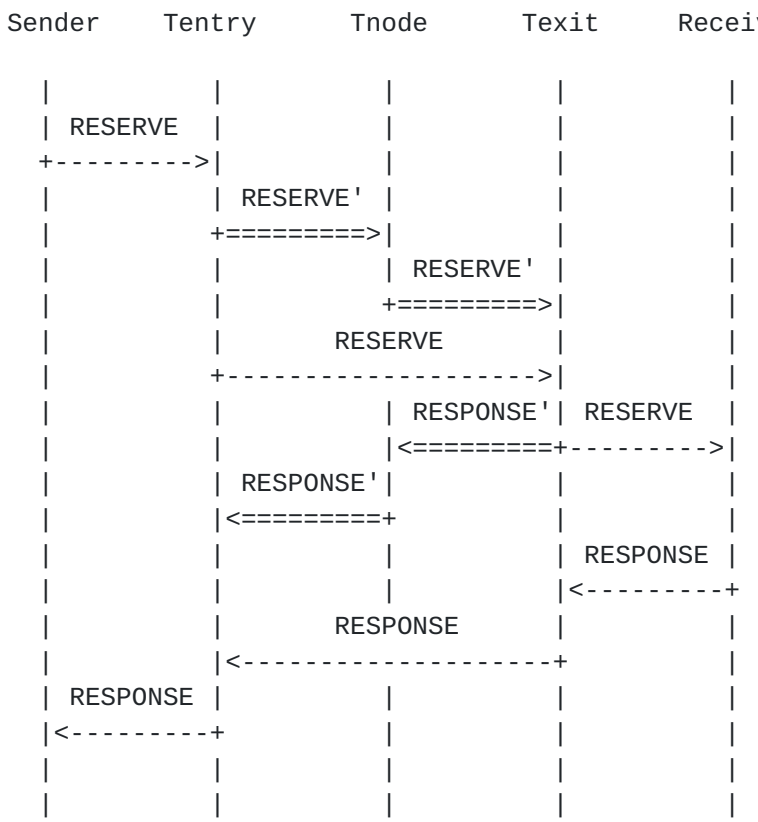

Figure 1: Sender-initiated Reservation for both End-to-end and Tunnel Signaling

3.1.2. Receiver-initiated Reservation for both End-to-end and Tunnel Signaling

This scenario assumes both end-to-end and tunnel sessions are

receiver-initiated. Figure 2 shows the messaging flow of NSIS operation over IP tunnels in this case. When Tentry receives the first 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 toward the Texit. Tentry also appends to the original QUERY message with a tunnel BOUND_SESSION_ID object containing the session ID of the tunnel session and sends it toward the Texit using normal tunnel encapsulation.

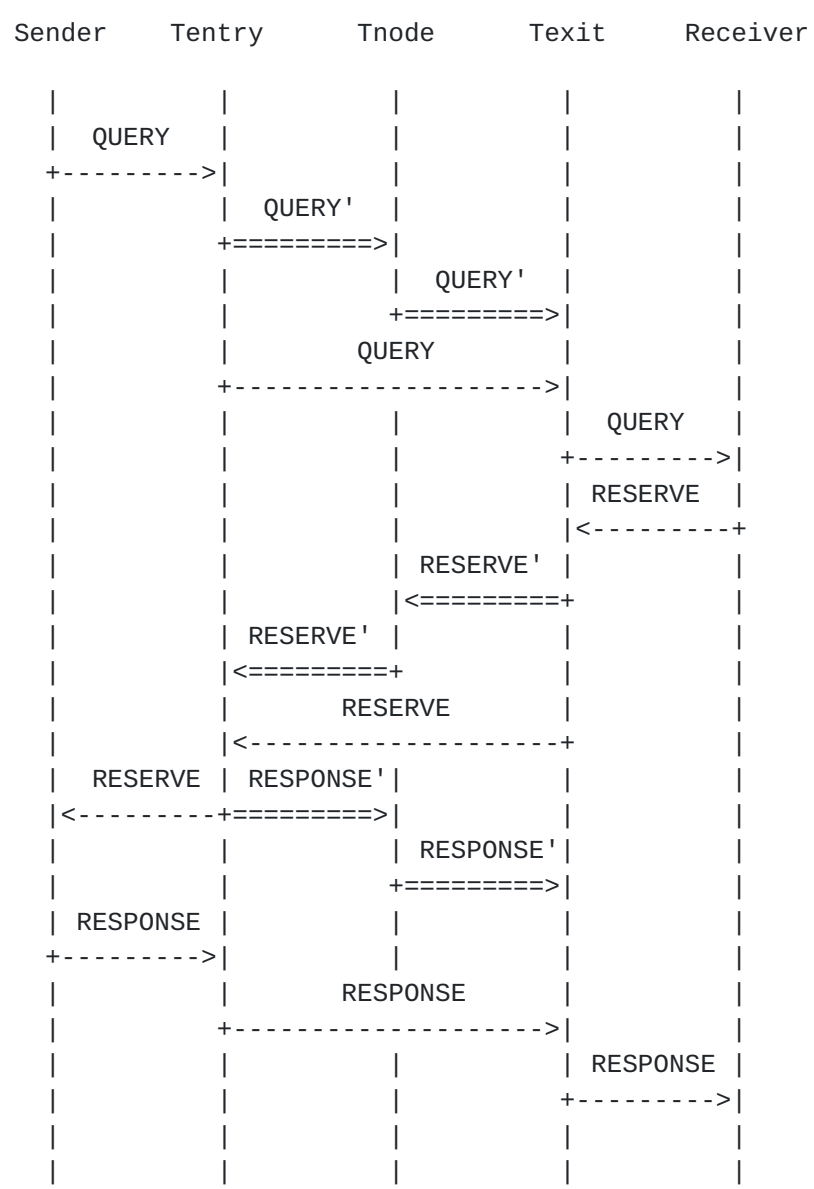

Figure 2: Receiver-initiated Reservation for both End-to-end and Tunnel Signaling

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 out a tunnel RESERVE' message immediately.

The end-to-end QUERY message passes along tunnel intermediate nodes just like any other tunneled packets. When Texit receives the end-to-end QUERY message, it notices the binding of a tunnel session and checks the state for the tunnel session. When the tunnel session state is available, Texit updates the end-to-end QUERY message using the tunnel session state, removes the tunnel BOUND_SESSION_ID object and forwards the end-to-end QUERY message further along the path.

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 the end-to-end RESERVE message will be appended with a tunnel BOUND_SESSION_ID object and forwarded towards Tentry. When Tentry receives the tunnel RESERVE' message, it creates the reservation state for the tunnel session and may send 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 it removes the BOUND_SESSION_ID object and forwards the end-to-end RESERVE message upstream toward the sender. When the end-to-end signaling finishes, the sender may send a RESPONSE message to the receiver.

3.2. Implementation Specific Issues

3.2.1. End-to-end and Tunnel Signaling Interaction

There could be many ways through which the end-to-end signaling and tunnel signaling may interact with each other. In general, different interaction approaches can be grouped into sequential mode and parallel mode. 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 decision in this document is a hybrid model. The rule is that the end-to-end signaling waits for tunnel signaling only if pre-conditions is needed to initiate the end-to-end session reservation. The example of this is the end-to-end QUERY message in Section 3.1.2, which needs to wait for the QUERY' message about the tunnel information and then be forwarded further. This ensures that the first end-to-end RESERVE message originated from the receiver will have the correct view of the whole path. After that, as long as the amount of resources the

end-to-end session requests for has been decided, the end-to-end reservation and tunnel reservation go parallel to speed up the whole process, as illustrated in both Section 3.1.1 and Section 3.1.2.

When the RESERVE messages of the end-to-end session and the tunnel session are propagated in parallel, if by the time an end-to-end RESERVE message carrying tunnel binding object arrives at the exit endpoint of the tunnel (which could be the Texit in Section 3.1.1 or the Tentry in Section 3.1.2), the results of the corresponding tunnel session reservation is already available, then the tunnel exit endpoint can remove the tunnel BOUND_SESSION_ID object, update the end-to-end RESERVE message accordingly and send it out of the tunnel immediately. However, it is also possible that the tunnel session state is not yet available when the end-to-end RESERVE message with a tunnel binding object is received. In the latter case, the exit tunnel endpoint should still remove the tunnel BOUND_SESSION_ID object, but sets the NON-QoSM Hop field [12] to indicate the possible existence of non-QoS link and then forward the message out immediately. The exit tunnel endpoint should then try to learn the results of the corresponding tunnel session reservation. This could be done by proactive polling after a specific amount of time, or when a refresh message is scheduled to send. In any case, once the state of the tunnel session is available, the exit tunnel endpoint should immediately trigger an end-to-end RESERVE message subject to the results of the tunnel reservation. If the tunnel reservation is successfully confirmed, the message would be a normal RESERVE refresh but with the NON-QoSM Hop field reset. Otherwise, the QSPEC in the RESERVE message should indicate error happened in the reservation.

3.2.2. Aggregate vs. Individual Tunnel Session Setup

The operation outlined in Section 3.1 applies to a flow that qualifies an individual dynamic tunnel session. For a tunnel that may contain multiple end-to-end sessions, it is more efficient to keep aggregate tunnel sessions rather than individual tunnel sessions whenever possible. This will avoid the new tunnel session setup overhead. Therefore, when the tunnel endpoint creates a reservation for a tunnel session based on the individual end-to-end session, it is up to local policy whether it wants to actually create an aggregate session by requesting more resources than the current end-to-end session requires. If it does, other end-to-end sessions arrived later may make use of this aggregate tunnel session. The tunnel endpoint will also need to determine how long to keep the tunnel session if no active end-to-end session is currently mapped to the aggregate tunnel session. The decision may be based on knowledge of likelihood of traffic in the future. It should be noted that once these kinds of on-demand aggregate tunnel sessions are set up, they are treated the same as pre-configured tunnel sessions to future end-

to-end sessions. Therefore, the adjustment of such aggregate sessions should follow Section 4.

Note that the session ID of an aggregate tunnel session should be different from that of the end-to-end session because they usually have separate lifetime. If the tunnel endpoint is certain that the tunnel session is for an individual end-to-end session alone, it may in some cases want to reuse the same session ID for both sessions. This will require additional manipulation of the NSLP state at the tunnel endpoints, since the NSLP state is usually keyed based on the session ID.

4. Protocol Operation with Pre-configured Tunnel Sessions

This section discusses NSIS operation over tunnels that are pre-configured through management interface with one or more tunnel sessions. A pre-configured tunnel session may be mapped to one session as an individual tunnel session but are usually mapped to multiple end-to-end sessions as an aggregate tunnel session.

4.1. Tunnel with Exactly One Pre-configured Aggregate Session

If only one aggregate session is configured in the tunnel and all traffic will receive the reserved tunnel resources, all packets just need to be IP-in-IP encapsulated as usual. If there is only one aggregate session configured in the tunnel but only some traffic should receive the reserved tunnel resources through the aggregate tunnel session, then the aggregate tunnel session should be assigned an appropriate flow ID. Qualified packets need to be encapsulated with this special flow ID. The rest of the traffic will be IP-in-IP encapsulated as usual.

4.2. Tunnel with Multiple Pre-configured Aggregate Sessions

If there are multiple pre-configured aggregate sessions over a tunnel setup, these sessions must be distinguished by their different aggregate tunnel flow IDs. In this case it is necessary to explicitly bind the end-to-end sessions with specific tunnel sessions. This binding is conveyed between tunnel endpoints by the tunnel BOUND_SESSION_ID object. Once the binding has been established, Tentry should encapsulate qualified data packets according to the associated aggregate tunnel flow ID. Intermediate nodes in the tunnel will then be able to filter these packets to receive reserved tunnel resources.

4.3. Adjustment of Pre-configured Tunnel Sessions

Adjustment of pre-configured tunnel sessions upon the change of its mapped end-to-end sessions is up to local policy mechanisms. RSVP-TUNNEL [16] described multiple choices to accomplish this. First, the tunnel reservation is never adjusted, which makes the tunnel a rough equivalent of a fixed-capacity hardware link ("hard pipe"). Second, the tunnel reservation is adjusted whenever a new end-to-end reservation arrives or an old one is torn down ("soft pipe"). Doing this will require the Texit to keep track of the resources allocated to the tunnel and the resources actually in use by end-to-end reservations separately. The third approach adopts some hysteresis in the adjustment of the tunnel reservation parameters. The tunnel reservation is adjusted upwards or downwards occasionally, whenever the end-to-end reservation level has changed enough to warrant the adjustment. This trades off extra resource usage in the tunnel for reduced control traffic and overhead.

5. NSIS-Tunnel Signaling Capability Discovery

There are several reasons why NSIS-tunnel signaling capability discovery is needed. First, when the Tentry decides to use UDP encapsulation to distinguish the tunnel flows, it needs to make sure the Texit is capable of doing UDP decapsulation when the flows leave the tunnel. Second, full operations of the NSIS tunnel mechanism, such as association of the end-to-end and tunnel session and adjustment of one session based on the state change of the other session, require the involvement of both Tentry and Texit. Therefore, one tunnel endpoint wants to know whether the other endpoint is also NSIS-tunnel signaling capable in deciding whether or not to initiate the related operations.

Manual configuration is one possible solution to the NSIS-tunnel signaling capability discovery problem. This section defines a NODE_CHAR object for GIST to automate the NSIS-tunnel capability discovery process.

The format of the NODE_CHAR object follows the general object definition in GIST [2]. It contains a fixed header giving the object type and object length, followed by the object value as shown in Figure 3 and Figure 4.

The Value field contains a single 'T' bit, indicating the NSIS-tunnel scheme defined in this document. It is also possible to use multiple bits to define NSIS-tunnel capability in finer granularity. We have adopted the simplest approach by using only one bit. The remaining reserved bits 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 GIST specification .

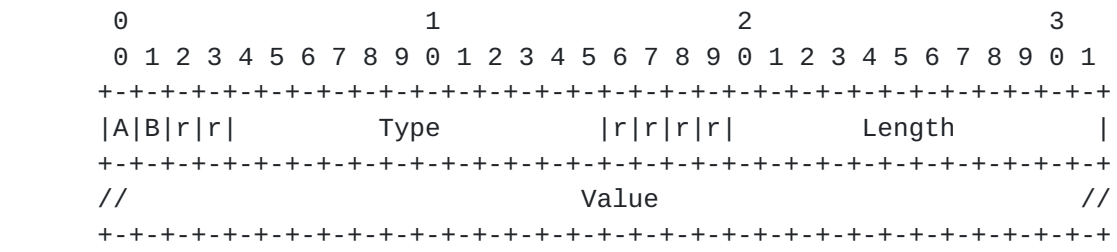

Figure 3: NODE_CHAR Object Format

Type: NODE_CHAR

Length: Fixed (1 32-bit word)

Value:

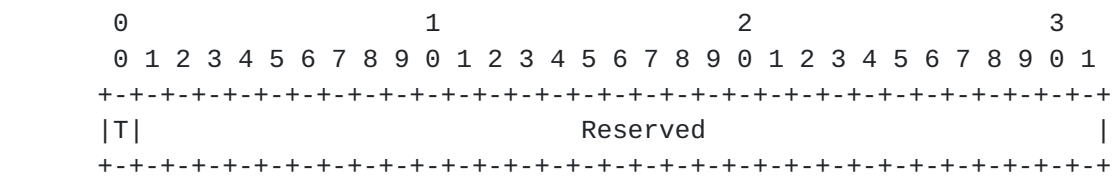

Figure 4: NODE_CHAR Object Value

This NODE_CHAR object is included in a QUERY or RESERVE message by a tunnel endpoint who wishes to learn about the other endpoint's tunnel handling capability. The other endpoint that receives this object will know that the sending endpoint is NSIS-tunnel capable, and place the same object in a RESPONSE message to inform the sending endpoint of its own tunnel handling capability. The procedures for using NODE_CHAR object in the two dynamically created tunnel session scenarios are further detailed below.

When both the end-to-end and the tunnel session are sender-initiated (Section 3.1.1) and Tentry is NSIS-tunnel capable, the Tentry includes a Request Identification Information (RII) object (see [3]) (if it is not present already) 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 supports NSIS tunneling, it learns that Tentry is NSIS-tunnel capable 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 learns whether Texit is NSIS-tunnel capable by examining the existence of the NODE_CHAR object and its T-bit. If both tunnel endpoints are NSIS-tunnel capable, the rest of the procedures will follow those defined in Section 3.1.1. Alternatively, Tentry may send out tunnel RESERVE message before the RESPONSE message confirming the NSIS-tunnel capability of Texit is received. If later it deduces that the Texit is not NSIS-tunnel capable, it should send out teardown messages to cancel the tunnel session reservation that has already been made. This way the signaling process is faster when Texit is NSIS-tunnel capable, but it can lead to temporary waste of tunnel resources if Texit is not NSIS-tunnel capable.

If both end-to-end and tunnel sessions are receiver-initiated (Section 3.1.2) and Tentry is NSIS-tunnel capable, the Tentry includes an RII object (if it is not present already) and a NODE_CHAR object with T bit set in the first end-to-end QUERY message sent toward Texit. An NSIS-tunnel capable Texit learns from the NODE_CHAR object whether Tentry is NSIS-tunnel capable. In the later end-to-end RESPONSE message to this QUERY message, the NSIS-tunnel capable Texit includes a NODE_CHAR object with T bit set to notify Tentry of its own tunnel capability. If both tunnel endpoints are NSIS-tunnel capable, the rest of the procedures will follow those defined in Section 3.1.2. Otherwise, Texit will not initiate tunnel session reservations.

6. 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 5.

7. Security Considerations

This draft does not draw new security threats. Security considerations for NSIS NTLP and QoS NSLP are discussed in [2] and [3], respectively. General threats for NSIS can be found in [18].

8. Appendix

8.1. NSIS-tunnel Operation for other Types of NSLP

This document discusses tunnel operation using QoS NSLP. It will be desirable to have the scheme work with other NSLPs as well. Since NSIS-tunnel operation involves specific NSLP itself and different NSLPs have different message exchange semantics, the NSIS-tunnel specification would not be the same for all NSLPs. However the basic aspects behind NSIS-tunnel operation could indeed be similar for different types of NSLPs. For example, in the case of NATFW NSLP [13], one of the most important signaling operations is CREATE. Assuming Tentry is a NATFW NSLP, the tunnel handling for the CREATE operation is expected to be very similar to the sender-initiated QoS reservation case. There are also a number of reverse directional operations in NATFW NSLP, such as RESERVE_EXTERNAL_ADDRESS and UCREATE. Detailed discussion of their operations inside the tunnel will be the scope of a separate document.

8.2. NSIS-tunnel Operation and Mobility

NSIS-tunnel operation needs to interact with IP mobility in an efficient way. In places where pre-configured tunnel sessions are available, the process is relatively straightforward. For dynamic individual signaling tunnel sessions, one way to improve NSIS mobility efficiency in the tunnel is to reuse the session ID of the tunnel session when tunnel flow ID changes during mobility. This works as follows. With a mobile IP tunnel, one tunnel endpoint is the Home Agent (HA), and the other endpoint is the Mobile Node (MN) if collocated Care-of-Address (CoA) is used, or the Foreign Agent (FA) if FA CoA is used. When MN is a receiver, Tentry is the HA and Texit is the MN or FA. In a mobility event, handoff tunnel signaling messages will start from HA, which may use the same session ID for the new tunnel session. When MN is a sender and collocated CoA is used, Tentry is the MN and Texit is the HA. Handoff tunnel signaling is started at the MN. It may also use the session ID of the previous tunnel session for the new tunnel session. When MN is a sender and FA CoA is used, the situation is complicated because Tentry has changed from the old FA to the new FA. In this case the new FA does not have the session ID of the previous tunnel session.

When mobile IP is operating on a bi-directional tunneling mode, NSIS-tunnel operation with mobility may be further improved by localizing the handoff tunnel signaling process by bypassing the path between HA and CN.

General aspects of NSIS interaction with mobility are discussed in [14].

8.3. Various Design Alternatives

8.3.1. End-to-end and Tunnel Signaling Integration Model

The contents of the original end-to-end singling messages are not directly examined by tunnel intermediate nodes. To carry out tunnel signaling we choose to maintain a separate tunnel session for the end-to-end session by generating tunnel specific signaling messages. An alternative approach is to stack tunnel specific objects on top of the original end-to-end messages and make these messages visible to tunnel intermediate nodes. Thus, these new messages serve both the end-to-end session and tunnel session. The latter approach turns out to be difficult because the actual tunnel signaling messages differ from the end-to-end signaling message both in GIST layer and NSLP layer information, such as MRI, PACKET CLASSIFIER and QSPEC. Although QSPEC can be stacked in an NSLP message, there doesn't seem to be a handy way to stack MRI and the PACKET CLASSIFIER in the NSLP layer. In addition, the stacking method only applies to individual signaling tunnels. The separate end-to-end and tunnel session signaling model adopted in this document handles both individual and aggregate signaling tunnels in a consistent way.

8.3.2. Packet Classification over the Tunnel

Packet classification over the tunnel may be done in either of the two ways: first, retaining the end-to-end packet classification rules; second, using tunnel specific classification rules. In the first approach, tunnel packet classification is not tied with the tunnel MRI. This is a useful property especially in handling tunnel mobility. Mobility causes changes in the tunnel MRI. If at the same time the packet classification rule does not change, the common path after a handoff does not need to be updated about the packet classification, which results in a better handoff performance. The main problem with this approach is that most existing routers do not support inspection of inner IP headers in an IP tunnel, where the tunnel independent packet classification fields usually reside. Therefore this document adopts the second approach which does not pose special classification requirements on intermediate tunnel nodes.

8.3.3. Tunnel Binding Methods

In this document, the end-to-end session and its mapping tunnel session use different session IDs and they are associated with each other using the BOUND_SESSION_ID object. This choice is obvious for aggregate tunnel sessions because in those cases the original end-to-end session and the corresponding aggregate tunnel session require independent control.

Sessions in individual signaling tunnels are created and deleted along with the related end-to-end session. So association between the end-to-end session and the corresponding individual tunnel

session has another choice: the two sessions may share the same session ID. Instead of sending a BOUND_SESSION_ID object, it may be possible to define a BOUND_FLOW_ID object, to bind the flow ID of the end-to-end session to the flow ID of the tunnel session at the tunnel endpoints. However, since flow ID is usually derived from MRI, if a NAT is present in the tunnel, this BOUND_FLOW_ID object will have to be modified in the middle, which makes the process fairly complicated. Furthermore, it is not desirable to have different session association mechanisms for aggregate signaling tunnels and individual signaling tunnels. Therefore, we decide to use the same tunnel BOUND_SESSION_ID mechanism for both individual and aggregation tunnel sessions. Note that in this case the mobility handling inside the tunnel can still be optimized in certain situations as discussed in Section 8.2.

In this document we used the existing BOUND_SESSION_ID object with a tunnel Binding Code to indicate the reason of binding. Two other options were considered.

1. Define a designated "tunnel object" to be included when tunnel binding needs to be conveyed.
2. Define a "tunnel bit" in corresponding NSLP message headers.

These options are not chosen because they either require the creation of an entirely new object, or change of basic message headers. They are also not generic solutions that can cover other binding causes.

There are basically three ways to carry the binding object between Tentry and Texit, using (a) end-to-end signaling messages, (b) tunnel signaling messages, (c) both end-to-end and tunnel signaling messages. In option (a) only tunnel endpoints see the tunnel binding information. In option (b) every tunnel intermediate node sees the binding information. Since there will be no state for the end-to-end session in tunnel intermediate nodes, they will all generate a message containing an "INFO_SPEC" object indicating no bound session found according to [3], which is not desirable. Option (c) suffers the same problem as in (b). Therefore the choice in this document for carrying the tunnel binding object is option (a).

9. Acknowledgements

10. References

10.1. Normative References

[1] Bradner, S., "Key words for use in RFCs to Indicate Requirement

Levels", BCP 14, RFC 2119, March 1997.

[2] Schulzrinne, H. and R. Hancock, "GIST: General Internet Signalling Transport", draft-ietf-nsis-ntlp-17 (work in progress), October 2008.

[3] Manner, J., Karagiannis, G., and A. McDonald, "NSLP for Quality-of-Service Signaling", draft-ietf-nsis-qos-nslp-16 (work in progress), February 2008.

10.2. Informative References

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

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

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

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

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

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

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

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

[12] Ash, G., Bader, A., Kappler, C., and D. Oran, "QoS NSLP QSPEC Template", draft-ietf-nsis-qspec-20 (work in progress), April 2008.

[13] 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.

[14] 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-10 (work in progress), July 2008.

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

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

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

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

[19] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, 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: schulzrinne@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

Full Copyright Statement

Intellectual Property