Internet Draft
draft-shore-ntlp-00.txt
May 2003
Expires November 2003

Melinda Shore
Cisco Systems

The NSIS Transport Layer Protocol (NTLP)

<draft-shore-ntlp-00.txt>

Abstract

The RSVP [RFC2205] model for communicating requests to network devices along a datapath has proven useful for a variety of applications beyond what the protocol designers envisioned, and while the architectural model generalizes well the protocol itself has a number of features that limit its applicability to applications other than IntServ [RFC1633]. The NSIS working group is developing a modernized version that, among other things, is based on a "two-layer" architecture that divides protocol function into transport and application. This document describes the transport protocol.

1. Terminology

Generator

NSLP

Path-coupled signaling

Receiver

Reservation

Sender

2. Introduction

RSVP is based on an "path-coupled" signaling model, in which signaling messages between two endpoints follow a path that is tied to the data path between the same endpoints, and in which the signaling messages are intercepted and interpreted by RSVP-capable routers along the path. While RSVP was originally designed to support QoS signaling for Integrated Services [RFC1633], this model has proven to generalize to other problems extremely well. Some of these problems include topology discovery, QoS signaling, communicating with firewalls and NATs, discovery of IPSec tunnel endpoints, test applications, and so on.

The IETF's NSIS working group has been chartered to develop an updated version of RSVP [Hancock, Brunner] -- one that is not tied directly to IntServ and in which the protocol machinery itself is sufficiently generalized to be able to support a variety of applications. What this means in practice is that there will be different NSIS applications, all of which share a base NSIS transport protocol. This is similar to the concepts used in secsh, where authentication and connection protocols run on top of a secsh transport protocol (see [Ylonen] for details).

[Hancock] describes the NSIS architectural framework. The protocol machinery is based heavily on RSVP [RFC2205] with refresh overhead reduction extensions [RFC2961].

NTLP differs from RSVP in several important ways, in addition to the mandatory support for RFC 2961. One of the most significant of these is that NTLP does not itself trigger reservations in NSIS nodes. The NSIS application will do that, and, indeed, some NSIS applications may not carry reservation requests at all (discovery protocols, for example). Because of this NSIS does not support reservation styles (those would be also be attributes of an application). Another significant difference is that that reservations may be installed by an NSLP in either a forward (from the sender

toward the receiver) or backward (from the receiver toward the sender) direction -- this is application-specific.

3. NTLP Messages

3.1. Message Processing Overview

Unlike RSVP, NTLP has only one fundamental message type. Directionality remains significant, but it is indicated through the use of a "direction" flag and directionally-linked differences in message processing may be inferred from the flag.

Even so, the basic operation of the protocol is similar to RSVP. A message is injected into a network by the sender towards a receiver with the receiver's address in the destination address in the IP header. NSIS entities along the path the message traverses will intercept it, store path state, act on (or not) the application payload data, and forward the message towards its destination. In NTLP, "path state" refers specifically to the unicast IP address of the previous hop node.

When the message arrives at the receiver (or its proxy), the receiver originates another NTLP message in response, with the directionality flag set to indicate that the message is being sent from the receiver to the sender. The primary difference in message handling is that this message is routed hop-by-hop between the NSIS nodes that handled the message in the forward direction. That is to say that the destination address in the IP header is the address of the next NSIS entity along the route in the reverse direction. Path state that was collected in the forward direction is used to accomplish this.

3.2. NTLP Message Format

NTLP messages consist of an NTLP header followed by optional TLV fields followed by an NSLP payload.

3.2.1. The NTLP Header

All NTLP messages (and by implication, all NSIS messages) start with an NTLP header. The header is formatted as follows:

```
       0             1              2             3
+-------------+-------------+-------------+-------------+
|   Version   |    Flags    |      Message Length       |
+-------------+-------------+-------------+-------------+
|     TTL     | (Reserved)  |         Checksum          |
+-------------+-------------+-------------+-------------+
|                         Flow ID                       |
+-------------+-------------+-------------+-------------+
```

where the fields are as follows:

Version:
8 bits. The protocol version number; in this case 0x01.

Flags:
8 bits. Flag bits include

0x01 directionality 0x02 teardown

Message Length:
16 bits. The total number of octets in the message, including the NTLP header and complete payload.

TTL: 8 bits. The IP TTL value with which the message was sent.

Checksum:
16 bits. The one's complement of the one's complement sum of the entire message. The checksum field is set to zero for the purpose of computing the checksum. This may optionally be set to all zeros. If a message is received in which this field is all zeros, no checksum was sent.

Flow ID:
32 bits. This is what is described as "Flow Identification" in [Hancock]. It is a value which, combined with the source IP address of the message, provides unique identification of a message, which may be used for later reference for actions such as quick teardowns, status queries, etc. The mechanism used for generating the value is implementation-specific.

3.2.2. NTLP TLVs

NTLP carries additional transport-layer information and requests as type-length-value fields, which are inserted after the header and before the NSLP payload. The TLV format is as follows:

```
       0             1              2             3
+-------------+-------------+-------------+-------------+
|           Length          |            Type           |
+-------------+-------------+-------------+-------------+
|                                                       |
//                        Value                        //
|                                                       |
+-------------+-------------+-------------+-------------+
```

where the fields are as follows:

Length:
16 bits. Total TLV length in octets. It must always be at least 4 and be a multiple of 4.

Type:
16 bits. The type of information or request. Defined below.

Value:
Variable length -- at least 4 octets and a multiple of 4 octets). The TLV semantic content.

3.2.2.1. IPv4_HOP

```
+-------------+-------------+-------------+-------------+
|                    IPv4 Hop Address                   |
+-------------+-------------+-------------+-------------+
|                Logical Interface Handle               |
+-------------+-------------+-------------+-------------+
```

The IPv4_HOP TLV carries the IPv4 address of the interface through which the last NSIS entity forwarded the message. The logical interface handle may be used to distinguish between multiple interfaces on the same entity, or it may be set to all 0s.

3.2.2.2. IPv6_HOP

```
+-------------+-------------+-------------+-------------+
|                                                       |
+                                                       +
|                                                       |
+             IPv6 Next/Previous Hop Address            +
|                                                       |
+                                                       +
|                                                       |
+-------------+-------------+-------------+-------------+
|                Logical Interface Handle               |
+-------------+-------------+-------------+-------------+
```

The IPv6_HOP TLV carries the IPv6 address of the interface through which the last NSIS entity forwarded the message. The logical interface handle may be used to distinguish between multiple interfaces on the same entity, or it may be set to all 0s.

3.2.2.3. IPv4_ERROR_CODE

```
+-------------+-------------+-------------+-------------+
|            IPv4 Error Node Address (4 octets)         |
+-------------+-------------+-------------+-------------+
|    Flags    |  Error Code |        Error Value        |
+-------------+-------------+-------------+-------------+
```

The IPv4_ERROR_CODE TLV carries the address of a node at which an NTLP error occurred, along with an error code and error value.

IPv4 Error Node Address:
    4 octets. The IPv4 address of the interface on the node that generated the error.

Flags:
    8 bits. None currently defined.

Error Code:

Error Value:

3.2.2.4. IPv6_ERROR_CODE

```
+------------+------------+------------+------------+
|                                                   |
+                                                   +
|                                                   |
+           IPv6 Error Node Address (16 octets)     +
|                                                   |
+                                                   +
|                                                   |
+------------+------------+------------+------------+
|   Flags    | Error Code |        Error Value      |
+------------+------------+------------+------------+
```

The IPv6_ERROR_CODE TLV carries the address of a node at which an NTLP error occurred, along with an error code and error value.

IPv6 Error Node Address:
16 octets. The IPv6 address of the interface on the node that generated the error.

Flags:
8 bits. None currently defined.

Error Code:

Error Value:

3.2.2.5. MESSAGE_ID

The MESSAGE_ID TLV is used to support reliable message delivery. This, as well as the MESSAGE_ID_ACK and MESSAGE_ID_NACK TLVs, were copied from RFC 2961.

The format is as follows:

```
+------------+------------+------------+------------+
|   Flags    |                 Epoch                |
+------------+------------+------------+------------+
|                  Message_Identifier               |
+------------+------------+------------+------------+
```

Flags: 8 bits
0x01 = ACK_Desired flag

Indicates that the sender requests the reciever to send an acknowledgment for the message.

Epoch: 24 bits

A value that indicates when the Message_Identifier sequence has reset. SHOULD be randomly generated each time a node reboots or the RSVP agent is restarted. The value SHOULD NOT be the same as was used when the node was last operational. This value MUST NOT be changed during normal operation.

Message_identifier: 32 bits

When combined with the message generator's IP address, the Message_Identifier field uniquely identifies a message. The values placed in this field change incrementally and only decrease when the Epoch changes or when the value wraps.

Note that this field is not redundant with the Flow ID in the NTLP header. The MESSAGE_ID TLV may be local between nodes, rather than end-to-end. The Flow ID is an end-to-end identifier.

3.2.2.6. MESSAGE_ID_ACK and MESSAGE_ID_NACK

These TLVs are also used to support reliable operations, and MESSAGE_ID_NACK is used to support summary refreshes.

The MESSAGE_ID_ACK format is as follows:

```
+-------------+-------------+-------------+-------------+
|    Flags    |                  Epoch                  |
+-------------+-------------+-------------+-------------+
|                   Message_Identifier                  |
+-------------+-------------+-------------+-------------+
```

Flags: 8 bits

No flags are currently defined. This field MUST be zero on transmission and ignored on receipt.

Epoch: 24 bits

The Epoch field copied from the message being acknowledged.

Message_Identifier: 32 bits

The Message_Identifier field copied from the message being acknowledged.

The format of the MESSAGE_ID_NACK TLV is identical to the format of the MESSAGE_ID_ACK TLV.

3.2.2.7. TIME_VALUES TLV

```
+-------------+-------------+-------------+-------------+
|                   Refresh Period R                    |
+-------------+-------------+-------------+-------------+
```

The TIME_VALUES TLV carries the refresh timeout period R, in milliseconds, used to generate the message in which this TLV appears. This TLV is used only in RFC 2205-style messaging and state maintenance.

3.2.2.8. MESSAGE_ID_LIST TLV

```
       0             1              2             3
+-------------+-------------+-------------+-------------+
|    Flags    |                 Epoch                   |
+-------------+-------------+-------------+-------------+
|                  Message_Identifier                   |
+-------------+-------------+-------------+-------------+
|                           :                           |
//                          :                          //
|                           :                           |
+-------------+-------------+-------------+-------------+
|                  Message_Identifier                   |
+-------------+-------------+-------------+-------------+
```

Flags: 8 bits
No flags are currently defined. This field must be zero on transmission and ignored on receipt.

Epoch: 24 bits
The Epoch field from the MESSAGE_ID TLV corresponding to the trigger message that advertised the state being refreshed.

Message_Identifier: 32 bits
The Message_Identifier field from the MESSAGE_ID TLV corresponding to the trigger message that advertised the state being refreshed. One or more Message_Identifiers may be

included.

4. Sending NTLP Messages

NTLP messages are sent as raw IP datagrams with protocol number <xx>, or may optionally be encapsulated in UDP packets.

When an endhost or its proxy wishes to initiate an NTLP session, it creates an NTLP message with the "directionality" bit in the NTLP header set to 0. The destination address in the IP header is the address that is expected to terminate the path along which signaling is expected to be sent. It may be a application peer host or terminal, or it may be a proxy.

NTLP messages should be sent with the router alert bit set in IPv4 headers or with the IPv6 router alert option [RFC2711].

In this version of the protocol, each NTLP message must fit in one IP datagram. [we need text discussing fragmentation here - ed]

5. Messaging and State Maintenance

NTLP supports both RFC 2205-style messaging and RFC 2961-style message for state installation and maintenance. RFC 2205 messaging is used by default. Features from RFC 2961 may be used individually with the indication of their use being feature-dependent.

5.1. RSVP v1 (RFC 2205)

RSVP takes a "soft state" approach to state maintenance in routers and hosts, as described in [RFC2205]. State is installed and kept fresh by periodically sending a full representation of installed state in idempotent Path and Resv messages, and is deleted if no matching Path/Resv refresh messages arrive before the timeout period expires. State may also be deleted through the use of an explicit teardown message.

5.1.1. Path Messages

When a host or its proxy wishes to initiate an NTLP session, it generates a Path message and periodically retransmits it. The Path message is addressed to a terminating host or its proxy. For example, if the NSLP is making a QoS reservation for a particular data flow, the Path message would be sent from the originator of a data flow and addressed to the host to whom the data will be sent. The source address in the IP header must be the address of the

interface from which the message was originated.

The Path message must begin with an NTLP header and include a HOP TLV. Other TLVs may be included, including for security purposes. Note that the presence of some TLVs, such as the MESSAGE_ID TLV, may indicate reliable message processing rather than RFC 2205-style messaging. The directionality flag in the NTLP header flags field must be set to 0x0.

The HOP TLV of each Path message contains the IP address of the interface through which the Path message was most recently sent (i.e. on the previous NSIS node). It may optionally carry a logical interface handle as well. Each NSIS-capable node along the path captures the Path message and processes it to create path state. Specifically, it stores the address and interface from the HOP TLV and overwrites it with its own address. Additionally, it will pass the NSLP payload to the appropriate NSLP module for processing [for efficiency purposes, should we have a flag that says not to do this in a particular direction? - ed]. Note that it does not modify the source or destination addresses in the IP header.

5.1.2. Resv Messages

Resv messages carry requests from the receiver towards the sender. Unlike Path messages, Resv messages are explicitly addressed to NSIS nodes along the data path using the addresses discovered and conveyed in the HOP fields in the precedent Path message. The source address in the IP header is the address of the interface on the NSIS node (i.e. not the "receiver") which sent the message. The directionality flat in the NTLP header flags field must be set to 0x1.

Unlike RFC 2205, instead of using a RESV_CONFIRM object we use the reliable messaging extensions from RFC 2961, described below.

5.1.3. Path Teardown Messages

Receipt of a teardown message indicates that matching path state must be deleted. A teardown message is one in which the teardown bit is set in the NTLP header. Note that this is independent of directionality, and the teardown message may be sent in either direction. The applications which have reservations that were installed by a message containing a matching Flow ID must be notified.

Unlike RFC 2205, if there is no matching path state the teardown message must be forwarded. There may be path state in support of an NSIS application that is not running on every node, and the teardown message must not be lost.

5.1.4. Timers

NTLP messages may contain a TIME_VALUES TLV describing the refresh period (R) between state refresh messages. This value is also used to derive the local state's lifetime value when the message is received and processed. The use of the TIME_VALUES TLV is discussed in section 3.7 of RFC 2205.

5.2. RFC 2961

In order to reduce the amount of network traffic associated with RSVP soft state maintenance, RFC 2961 specifies RSVP extensions for refresh reduction. It also addresses reliability and latency problems stemming from message loss (increasing the interval between refresh messages reduces the amount of state maintenance traffic on the wire but increases the latency in response to routing changes and increases the effects of dropped messages).

RFC 2961 reduces state maintenance traffic on the wire using several mechanisms. One is to allow "bundling" of requests. Another is the use of "summary" refresh messages, which obviates the need to send entire refresh messages.

NTLP supports RFC 2961 message processing, and because of that the "Refresh reduction capable" bit in the Flags field of the NTLP header is redundant and therefore unnecessary. Similarly, in NTLP all messages may carry multiple NSLP payloads and it is not necessary to separately identify a "bundled" message type. Unlike RFC 2961, bundled messages may be carried end-to-end and the router alert option/bit should be set. An NSIS node has the option of unbundling bundled messages into multiple messages (because of a smaller MTU on the outgoing link, for example, or to reduce transmission latency of high-priority requests) or bundling multiple NTLP messages into a single NTLP message.

5.3. MESSAGE_ID Usage

The MESSAGE_ID TLV may be included in any NTLP message other than an ACK. It MUST NOT be used in a message that includes more than one NSLP payload, which means that when an NSIS node wishes to receive an ACK for a particular message, and that message has more

than one NSLP payload, it must unbundle the NSLP payloads and transmit the ones requiring acknowledgment individually. [let's think about doing this in bundled messages - ed]

In messages containing a HOP TLV, the generator of the MESSAGE_ID TLV appears in the HOP. If there is no HOP, it is taken from the source address in the IP header.

The Epoch field contains a value selected by the generator. The value is used to indicate when the sender resets the values used in the Message_Identifier field. On startup, a node should randomly select a value to be used in the Epoch field. It should ensure that the selected value is not the same as was used when the node was last operational. The value must not be changed unless the node or agent is restarted.

The Message_Identifier field contains a generator-selected value. This value, when combined with the generator's IP address, identifies a particular NTLP message and the specific state information it represents. When a node is sending a refresh message with a MESSAGE_ID TLV, it should use the same Message_Identifier value that was used in the NTLP message that identified the NSLP state being refreshed. When a node is sending a trigger, the Message_Identifier value must have a value that is greater than any other Message_Identifier value previously used with the same Epoch field value (note that this provides a way to determine if messages are being delivered out-of-order). A value is considered to have been used when it has been sent in any message using the associated IP address with the same Epoch field value.

The ACK_Desired flag is set when the MESSAGE_ID generator wants an acknowledgement with a MESSAGE_ID_ACK TLV in response. Such information can be used to ensure reliable delivery of RSVP messages in the face of network loss. Nodes setting the ACK_Desierd flag should retransmite unacknowledged messages at a more rapid interval than the standard refresh period until the message is acknowledged or until the "rapid" retry limit is reached. Rapid retransmission rate must be based on the exponential back-off procedures defined below. The ACK_Desired flag will typically be set only in trigger messages. The ACK_Desired flag may be set in a refresh message.

Nodes processing incoming MESSAGE_ID TLVs should check to see if a newly-received message is out-of-order. Out-of-order messages should be ignored. To determine ordering, the received Epoch value must match the value previously received from the message generator. If the values differ the receiver must not treat the message

as out of order. When the Epoch values match and the Message_Identifier value is less than the largest value previously received from the sender, then the NSLP processor for the application payload should check for existing state for that payload. If none exists, the receiver must not treat the message as out of order. If local state does exist, the message should be treated as out of order.

Note that the Message_Identifier value may wrap. To cover the wrap case, the following expression may be used to test if a newly received Message_Identifier value is less than a previously received value:

```
if ((int) old_id - (int) new_id > 0)  {
    new value is less than old value;
}
```

MESSAGE_ID TLVs of messages that are not out of order should be used to aid in determining if the message represents new state or a state refresh. Note that this refers only to NTLP state -- NSLP application state is determined by the NSLP layer.

If the received Epoch value differs from the value previously received, the message is a trigger message and the receiver must fully process the message. If the message contains the same Message_Identifier value that was used in the most recently received message for the same session, then the receiver should treat the message as a state refresh. If the Message_Identifier value is greater than the most recently received value, then the receiver must fully process the message. Note these guidelines will apply to NSLP payloads in addition to NTLP state.

Nodes receiving a message that is not out-of-order containing a MESSAGE_ID TLV with the ACK_Desired flag set should respond with a MESSAGE_ID_ACK TLV. MESSAGE_ID TLVs received in messages containing errors (for example, not syntactically valid) must not be acknowledged.

5.4. MESSAGE_ID_ACK and MESSAGE_ID_NACK Usage

The MESSAGE_ID_ACK TLV is used to acknowledge receipt of messages containing MESSAGE_ID TLVs that were sent with the ACK_Desired flag set. A MESSAGE_ID_ACK object must not be generated in response to a received MESSAGE_ID TLV when the ACK_Desired flag is not set.

The MESSAGE_ID_NACK object is used as part of the summary refresh function. The generation and processing of MESSAGE_ID_NACK is described in further detail below.

MESSAGE_ID_NACK and MESSAGE_ID_NACK TLVs may be sent in any NTLP message that has an IP destination address matching the generator of the associated MESSAGE_ID. This means that they will not typically be included in non hop-by-hop messages. When no appropriate message is available, one or more ACKs or NACKs should be sent on their own.

Implementation should limit the amount of time that an acknowledgment is delayed in order to be piggy-backed or sent on its own. Different limits may be used for MESSAGE_ID_ACK and MESSAGE_ID_NACK. MESSAGE_ID_ACK is used to detect link transmission losses. If an ACK object is delayed too long, the corresponding message will be retransmitted. To avoid retransmission, ACKs should be delayed a minimal amount of time. A delay time equal to the link transit time may be used. MESSAGE_ID_NACK may be delayed an independent andlonger time, although additional delay increases the amount of time a message is not acted upon.

5.5. Summary Refreshes

Summary refreshes enable the refreshing of NTLP and NSLP state without the transmission of entire reservation request messages. The benefits are that it reduces the amount of information that must be transmitted and processed in order to maintain state synchronization. Importantly, it preserves NTLP's ability to handle non-NSIS next hops and to adjust to changes in routing.

The summary refresh mechanism builds on MESSAGE_ID. Only state that was previously advertised in applications encapsulated in NTLP messages containing MESSAGE_ID TLVs can be refreshed via a summary refresh.

The summary refresh uses the TLVs and the ACK message previously defined as part of the MESSAGE_ID functionality, and a new TLV that is specifically in support of summary refresh. This TLV is the MESSAGE_ID_ LIST.

The MESSAGE_ID_LIST TLV is used to refresh all NTLP and NSLP reservation state and path-related state. It is made up of a list of Message_Identifier fields that were originally advertised in MESSAGE_ID TLVs. An NTLP node receiving a message with a MESSAGE_ID_LIST TLV matches each listed Message_Identifier field with

installed NTLP state, and requests the associated NSLP to do the same. If matching state cannot be found then the sender is notified with a refresh NACK. A refresh NACK is sent via the MESSAGE_ID_NACK TLV. As described previously, the rules for sending a MESSAGE_ID_NACK are the same as for sending a MESSAGE_ID_ACK. This includes sending MESSAGE_ID_NACKs piggy-backed in an unrelated NTLP message or in NTLP ACK messages.

5.5.1. MESSAGE_ID_LIST usage

Unlike RFC 2961, no special message format is used. If a MESSAGE_ID_LIST TLV appears in an NTLP message, the message is processed according to procedures described above. Messages containing the MESSAGE_ID_LIST TLV are normally sent end-to-end in a forward direction with a destination IP address equal to the address of the original reservation, or trigger, request. The destination IP address may be set to the NTLP next hop when the next hop is known to be NTLP-capable.

The source IP address in a message containing a MESSAGE_ID_LIST TLV is the address of the node that generated the message. The source IP address must mach the address associated with the MESSAGE_ID TLVs when they were included in a standard NTLP message. The source address associated with a MESSAGE_ID depends on the contents of the message. For messages with a HOP TLV, the address is found in the HOP object. For other messages, the address is taken from the source address in the IP header.

Only one MESSAGE_ID_LIST TLV is allowed per NTLP message.

5.5.2. Summary Refresh Message Processing

A MESSAGE_ID_LIST TLV may be used to refresh reservation and path state. If summary refresh is used, the generation of a standard refresh message should be suppressed. A state's refresh interval is not affected by the use of summary refresh.

When generating summary refreshes, a node should refresh as much state as possible by including the information from as many MESSAGE_ID TLVs as possible in the same MESSAGE_ID_LIST. Only the information from MESSAGE_ID objects that describe the source and destination IP address restrictions, as described above, may be included in the same MESSAGE_ID_LIST.

Only state that was previously advertised in NTLP messages containing MESSAGE_ID TLVs can be refreshed using summary refresh. The

use of summary refreshes must not result in state being timed out at the NTLP next hop. The period at which state is refreshed using summary refresh may be shorter than the period that would be used using normal idempotent messaging, but it must not be longer.

The particular approach used to trigger summary refreshes is implementation specific. Some possibilities are based on each state's refresh period, or on a per-interface basis.

When generating a summary refresh, there are two methods for identifying which path state may be refreshed in a specific message. In both cases the previously mentioned refresh interval and source IP address restrictions must be followed. The primary method is to include only those sessions which share the same destination IP address in the same MESSAGE_ID_LIST TLV.

The secondary, multicast-based method described in RFC 2961 is not supported.

5.5.3. Summary Refresh NACKs

NACKs are used to indicate that a received Message_Identifier field carried in a MESSAGE_ID_LIST does not match any installed state. This may occur for a humber of reasons including, for example, a route change. A summary refresh NACK is encoded in a MESSAGE_ID_NACK TLV. When generating a summary refresh NACK, the Epoch and Message_Identifier fields of the the MESSAGE_ID_NACK must have the same value as was received.

Received MESSAGE_ID_NACK TLVs indicate that the TLV generator does not have any installed state matching the Message_Identifier included in TLV. Upon recieving a MESSAGE_ID_NACK, the receiver performs a lookup on uninstalled state based on the Epoch and Message_Identifier values contained in the TLV. If matching state is found then the receiver must transmit the matching state via a standard forward or backward NTLP message. If the receiver cannot identify any isntalled state, then no action is required.

5.5.4. Preserving Soft State

In RFC 2961, summary refreshes are used in place of the periodic sending of full representations of installed state. While this provides scaling benefits and protects against common network events such as packet loss or routing changes, it does not provide exactly the same error recovery properties. An example error that could potentially be recovered from via standard messages but not

with summary refresh is internal corruption of state. This section recommends two methods that can be used to better preserve NTLP's soft state error recovery mechanism. Both use existing protocol messages.

The first uses a checksum or other algorithm to detect a previously unnoticed change in internal state. This mechanism does not protect against internal state corruption. It just covers the case where a trigger message should have been sent, but was not. When sending a message, a node should run a checksum or MAC over the internal state and store the result. The choice of algorithm is an administrative decision. Periodically it should rerun the algorithm and compare the new result with the stored result; if the values differ, than a corresponding standard, full message should be sent and the new value should be stored. The recomputationperiod should be set based on the computational resources of the node and the reliability requirements of the network.

The second mechanism is to periodically send full, standard NTLP with complete NSLP payloads as appropriate. Since this mechanism uses standard refresh mechanisms it can recover from the same errors as NTLP without summary refresh. When doing this, the period that standard refresh messages should are sent must be longer than the interval between summary refresh messages in order to gain the benefits of summary refresh. When sending a standard refresh a corresponding summary refresh should not be sent during the same refresh period. The frequency of generation of standard refresh messages relative to summary refresh messages should be configurable by the network administrator.

5.5.5. Exponential Back-off Procedures

This section is taken from RFC 2961 and is based on [Pan]. Implementations must use the described procedures or the equivalent.

5.5.5.1. Outline of Operation

The following is one possible mechanism for exponential back-off retransmission of an unacknowledged NTLP message. When sending such a message a node inserts a MESSAGE_ID TLV with the ACK_Desired flag set. The sending node will retransmit the message until a message acknowledgment is received or the message has been transmitted a maximum number of times. Upon reception, a receiving node acknowledges the arrival of the message by sending back a message acknowledgment (one containing a corresponding MESSAGE_ID_ACK TLV). When the sending node receives the acknowledgment,

retransmission of the message is stopped. The interval between retransmission is governed by a rapid retransmission timer. The rapid retransmission timer starts at a small interval and increases exponentially until it reaches a threshold.

5.5.5.2. Time parameters

The described procedures make use of the following time parameters. All parameters are per-interface.

Rapid retransmission interval Rf:
Rf is the initial retransmission interval for unacknowledged messages. After sending the message for the first time, the sending node will schedule a retransmission after Rf seconds. The value of Rf could be as small as the round trip time (RTT) between a sending and receiving node, if known.

Rapid retry limit Rl:
Rl is the maximum number of times a message will be transmitted without being acknowledged.

Increment value Delta:
Delta governs the speed with which the sender increases the retransmission interval. The ratio of two successive retransmission intervals is (1 + Delta).

Suggested default values are an initial retransmission timeout (Rf) of 500ms, a power of 2 exponential back-off (Delta = 1), and a retry limit (Rl) of 3.

5.5.5.3. Retransmission Algorithm

After a sending node transmits a message containing a MESSAGE_ID TLV with the ACK_Desired flag set, it should immediately schedule a retransmission after Rf seconds. If a corresponding MESSAGE_ID_ACK TLV is received earlier than Rf seconds, then retransmission should be canceled. Otherwise, it will retransmit the message after (1 + Delta) * Rf seconds. The staged retransmission will continue until either an appropriate MESSAGE_ID_ACK TLV is received, or the rapid retry limit, Rl, has been reached.

A sending node can use the following algorithm when transmitting a message containing a MESSAGE_ID TLV with the ACK_Desired flag set:

    Prior to initial transmission initialize: Rk = Rf and Rn = 0

```
while (Rn++ < Rl)  {
    transmit the message;
    wake up after Rk seconds;
    Rk = Rk * (1 + Delta);
}
/* acknowledged or no reply from receiver for too long: */ do any
       needed clean up; exit;
```

6. NSLP Interface

7. NAT Interactions

NTLP uses IP addresses for routing, both end-to-end and hop-by-hop. Given the applications which NTLP will be transporting, it is highly likely that those applications will be using payload-embedded addresses and there will be some interactions. The use of a NAT NSLP together with other NSLPs can mitigate this, but there will be problems transiting non-NSIS-capable NATs.

When an NSIS entity receives an NTLP message travelling in the forward direction, it writes the address in the IPv4_HOP or IPv6_HOP, as appropriate, from the packet into local per-session state and replaces the HOP data in the message with the address of the outgoing interface. When the entity is a NAT, it will write the translated-to address. Note that while it is usually the case that payload integrity protection breaks in the presence of NATs if embedded addresses are being rewritten, this is not substantially different from the rewriting of the HOP field which occurs within NTLP anyway.

However, if an NTLP message crosses a non-NSIS-capable NAT, several problems may occur. The first is that if the message is being dropped in a raw IP packet, the NAT may simply drop the packet because it doesn't know how to treat it. Another is that the address in the HOP field will be incorrect. NTLP and the applications it carries cannot be expected to function properly across non-participating NATs.

8. Proxy Considerations

9. Security Considerations

[we need to determine exactly what we're protecting and why - ed].

10. IANA Considerations

Protocol number
UDP port number
TLVs

11. Multicast Considerations

12. Acknowledgments

Large sections of this document were lifted wholesale from RFC 2961, and credit belongs with its authors: Lou Berger, Der-Hwa Gan, George Swallow, Ping Pan, Franco Tommasi, and Simone Molendini.

13. Normative References

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

[Hancock] Hancock, R. et al. "Next Steps in Signaling: Framework," work in progress, draft-ietf-nsis-fw-02.txt, March 2003.

[Pan] Pan, P. and H. Schulzrinne. "Staged Refresh Timers for RSVP," Global Internet '97, Phoenix, AZ, November 1997. http://www.cs.columbia.edu/~pingpan/papers/timergi.pdf

[RFC2205] Braden, R. et al. "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification," RFC 2205, September 1997. [RFC2711] Partridge, C. and A. Jackson. "IPv6 Router Alert Option," RFC 2711, October 1999.

[RFC2961] Berger, L. et al. "RSVP Refresh Overhead Reduction Extensions," RFC 2961, April 2001.

14. Informative References

[RFC1633] Braden, R., Clark, D., and S. Shenker. "Integrated Services in the Internet Architecture," RFC 1633, June 1994.

[Ylonen] Ylonen, T. et al. "SSH Protocol Architecture," work in progress, September 2002.

15. Author's Address

Melinda Shore
Cisco Systems

809 Hayts Road
Ithaca, NY 14850
USA
mshore@cisco.com

16. TO DO

need better Flow ID - minimize collision likelihood
error codes
proxy issues, inc. early proxy terminating path
nslp headers
Should message_id be folded into the ntlp header?
timer object
MESSAGE_ID_NACK processing
Maybe directionality bit should be hop-by-hop bit?
fragmentation
security
IANA considerations
fill in terminology (definitions)
multicast considerations
simple example NSLP (discovery?)

17. Full Copyright Statement