Internet Engineering Task Force
INTERNET-DRAFT
Expires October 2003

L. Westberg
A. Bader
D. Partain
V. Rexhepi

Ericsson

G. Karagiannis

University of Twente

April 2003

A Proposal for RSVPv2-NSLP
draft-westberg-proposal-for-rsvpv2-nslp-00.txt
Document Version: $Revision: 2.1 $

Status of this memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts. Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet- Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

Distribution of this memo is unlimited.

Copyright Notice

Abstract

The Resource ReSerVation Protocol (RSVPv1) has been on the standards track within the IETF for a number of years. During that time, the level of vendor support and deployment has been relatively slow, despite the demand for technologies offering services with different levels of quality of service to their customers. This memo seeks to initiate a dialog about the design of a new version of RSVPv1, called RSVPv2, that meet the requirements formulated by IETF NSIS working group. It also outlines the motivation for using RSVP2 as "next step in signaling".

The RSVPv2 framework uses the layer-split architecture separating signaling application and transport functions. This concept was adopted by NSIS WG and the two layers are called NSLP NSIS Signaling Layer Protocol (NSLP) and NSIS Transport Layer Protocol (NTLP). This draft provides design guidelines and specifications for the development of the RSVPv2 NSLP part.

RSVP2-NSLP offers increased modularity and contains less mandatory objects compared to RSVPv1, which allow lightweight implementation and flexible application. The new protocol is extended with PHR and PDR objects that makes it possible to use the protocol in different part of multi-domain networks and use the protocol in DiffServ environment.

Table of Contents

1. Introduction

A number of different QoS solutions have been developed by the IETF, amongst them IntServ and its signaling protocol, RSVPv1, defined in [RFC2205]. RSVPv1 [RFC2205] is a resource reservation signaling protocol that was designed to be applied in an end-to-end communication path. It can be used by an application to make its QoS requirements known and reserve resources in all the network nodes in the path.

RSVPv1 has not enjoyed the level of deployment that might have been expected. This is due to issues such as design constraints as it is optimized for multicast, etc. [RFC2475, RFC3175, etc]. This memo seeks to initiate a dialog about the design of a new version of RSVPv1, which we call RSVPv2. The goal of the RSVPv2 framework would be to rectify the issues that have been identified with RSVPv1 and provide an evolutionary path forward.

The RSVPv2 framework uses the concept introduced in [BrLi01] that splits signaling protocol into two layers:

(1) a common lower level protocol that performs transport-layer and soft-state functions. This common lower level is called CSTP ("Common Signaling Transport Protocol").

(2) a set of upper-level signaling functions that are specific to particular signaling applications. These upper-level signaling tasks and functions are accomplished by a set of ULSPs ("User Layer Signaling Protocols).

The CSTP together with the set of ULSPs will implement the Internet Signaling Protocol Suite (ISPS). The NSIS working group adopted this concept denoting the two protocols as NSIS Transport Layer Protocol (NTLP) and NSIS Signaling Layer Protocol (NSLP), respectively [Hanc03].

This memo outlines the motivation for using RSVPv2-NSLP as NSIS Signaling Layer Protocol. It provides a design guideline and specification for RSVPv2-NSLP. Note that in order to be able to communicate with NTLP, RSVPv2-NSLP needs to use an NSLP Identifier that has to be assigned by the NSIS WG. RSVPv2-NSLP specified in this draft is able to interwork with NTLP specified in [WeKa03].

1.1. Definitions/Terminology

Interdomain traffic:

Traffic that passes from one NSIS domain to another ([identical to [Hanc03]).

Intra-domain NSIS signaling is where the NSIS signaling messages are originated, processed and terminated within the same NSIS domain.

NSIS Domain (ND) (identical to [Hanc03]):

Administrative domain where an NSIS protocol signals for a resource or set of resources.

NSIS Entity (NE) (identical to [Hanc03]):

the function within a node which implements an NSIS protocol. In the case of path-coupled signaling, the NE will always be on the data path.

NSIS Forwarder (NF) (identical to [Hanc03]):

NSIS Entity between a NI and NR which may interact with local resource management function (RMF). It also propagates NSIS signaling further through the network.

NF Edge nodes:

NF Nodes that are located at the boundary of an administrative domain, e.g., Diffserv. This node is a NTLP stateful node.

NF Interior node:

All the nodes that are part of an administrative domain, e.g., Diffserv, and are not NF edge nodes. An interior node can be either a NTLP stateful node or a NTLP stateless node.

NF Ingress node:

An NF edge node that handles the traffic as it enters the domain. This node is a NTLP stateful node.

NF Egress node:

An NF edge node that handles the traffic as it leaves the domain. This node is a NTLP stateful node.

NSIS Initiator (NI) (identical to [Hanc03]):

NSIS Entity that initiates NSIS signaling for a network resource.

NSIS Responder (NR) (identical to [Hanc03]):

NSIS Entity that terminates NSIS signaling and can optionally interact with applications as well.

NSIS Signaling Layer Protocol (NSLP) (identical to [Hanc03]):

generic term for an NSIS protocol component that supports a specific signaling application.

NSIS Transport Layer Protocol (NTLP) (identical to [Hanc03]):
placeholder name for the NSIS protocol component that will support lower layer (signaling application independent) functions.

NTLP aware node:

a node that implements and supports the NTLP protocol.

NTLP stateful node:

a NTLP aware node that maintains a NTLP transport layer state.

NTLP stateless node:

a NTLP aware node that does not maintain a NTLP transport layer state.

NE NTLP stateful

NE entity that is NTLP stateful.

NE NTLP stateless

NE entity that is NTLP stateless.

NF NTLP stateful

NF entity that is NTLP stateful.

NF NTLP stateless

NF entity that is NTLP stateless.

Resource Management Function (RMF) (identical to [Hanc03]):

an abstract concept,
representing the management of resources in a domain or a node.

End Host:

QoS-aware end terminal, either fixed or mobile, i.e. running QoS-aware applications. This node is a NTLP stateful node and it can be considered as a NI or a NR.

RSVPv2 NSLP:

an NSLP type that can be a part of the RSVPv2 framework.

NSLP intra-domain:

a domain that supports NSIS intra-domain signaling.

Classifier - an entity which selects packets based on the content of packet headers according to defined rules.

DS behavior aggregate (identical to [RFC2475]):

A collection of packets with the same DS codepoint crossing a link in a particular direction.

DS-compliant (identical to [RFC2475]):

Enabled to support differentiated services functions and behaviors as defined in [RFC2474], this document, and other differentiated services documents; usually used in reference to a node or device.

Interdomain traffic - Traffic that passes from one NSIS domain to another

Intra-domain NSIS signaling is where the NSIS signaling messages are originated, processed and terminated within the same NSIS domain.

NSIS Forwarder (NF) - NSIS Entity on the path between a NI and NR which may interact with local resource management function (RMF) The NSIS Forwarder also propagates NSIS signaling further through the network (identical to [Hanc03]). Note that NF can be also considered as a RSVPv2 forwarder.

NSIS Initiator (NI) - NSIS Entity that initiates NSIS signaling for a network resource (identical to [Hanc03]). Note that NI can be also considered as a RSVPv2 initiator.

NSIS Responder (NR) - NSIS Entity that terminates NSIS signaling and can optionally interact with applications as well (identical to [Hanc03]). Note that NR can be also considered as a RSVPv2 responder.

Path-coupled signaling - a mode of signaling where the signaling messages follow a path that is tied to the data messages (see [Hanc03]).

Path-decoupled signaling - signaling with independent data and signaling paths (see [Hanc03]).

Per Hop Behavior (PHB) (identical to [RFC2475]):

The externally observable forwarding behavior applied at a DS-compliant node to a DS behavior aggregate.

Per Hop Reservation (PHR):

The per-hop resource reservation in a Diffserv domain, extending the Diffserv PHB, e.g., the bandwidth allocated to an AF PHB (see RFC2597]), with resource reservation. It is implemented at both the interior nodes and the edge nodes.

Per Hop Reservation (PHR) protocol:

A type of protocol that is used to perform a per hop reservation. A PHR protocol part is used in all nodes in the Diffserv domain (both edge and interior nodes) on a hop by hop basis.

Per Domain Behavior (PDB)(similar to [NiKa01]):

Describes the behavior experienced by a particular set of packets as they cross a DS domain. A PDB is characterized

by specific metrics that quantify the treatment that a set of packets with a particular DSCP (or set of DSCPs) will receive as it crosses a DS domain.

Per Domain Reservation (PDR):

The resource reservation functionality in the complete Diffserv domain.

Per Domain Reservation (PDR) protocol:

A type of signaling protocol used to perform a per domain reservation signaling.
A PDR protocol part is used by NF(edge) nodes (NF(ingress) and NF(egress)),
but not by the NF(interior) nodes.

Resource - something of value in a network infrastructure to which rules or policy criteria are first applied before access is granted. Examples of resources include the buffers in a router and bandwidth on an interface

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2. Motivation for RSVPv2

Embarking on the adventure of creating RSVPv2 is not to be done lightly. A great deal of effort was put into the design of the IntServ model and its signaling protocol, RSVPv1. As such, there must be a clear need for the evolution of the QoS signaling part of RSVPv1. This section tries to provide that motivation.

We believe that this work can be accomplished by examining the design constraints placed upon the development of RSVPv1 and eliminate these constraints in RSVPv2 design.

2.1. Limitation of RSVPv1 design

RSVPv1 is well-designed for the applications for which it was intended and worked hard to provide a modular protocol within the constraints of its intended use. We see value in questioning the applications chosen, thereby improving the protocol.

This section outlines some of the design considerations that went into the design of RSVPv1, which in turn led to decisions that make it difficult to use RSVPv1 beyond its originally-intended scope.

2.1.1. Designed for Multicast Applications

One of the most important design requirement for RSVPv1 was support for multicast applications. RFC 1633 [RFC1633] states, "There are a number of requirements to be met by the design of a reservation setup protocol. It should be fundamentally designed for a multicast environment...."

Multicast support introduces a level of complexity into the protocol that is not needed in support of unicast applications. For example, RSVPv1's state maintenance is complex as it needs to support dynamic membership changes in the multicast groups, such as reservation state
merging and maintenance.

Our working assumption is that RSVPv2 should be optimized for unicast
rather than multicast and that relaxing this design constraint will in turn greatly simplify the protocol.

2.1.2. Least Common Denominator Not Small Enough

Rightfully so, RSVPv1 put a great deal of effort into creating a modular protocol with the ability to use those pieces that apply in a particular setting. However, this modularity was created with the backdrop of multicast applications. This means that, while modular to some degree, even the "least common denominator" of objects that must be carried is too heavy in some networking contexts. That is, while flexible, RSVPv1 does not allow for more lightweight implementations if fewer features are needed in certain parts of the network.

The fact that the least common denominator is too heavy means that:

* Some objects are always carried in RSVPv1 messages that are not applicable in some settings.

* There is an expectation of the same level of protocol functionality throughout the network(s). Clearly, different parts of the network need different levels of functionality, a differentiation not supported by RSVPv1.

On May 20, 2002, Bob Braden, one of the creators of RSVPv1, wrote the following on the NSIS working group's mailing list [NSIS-ML1]:

> "...RSVP may have had the modularity wrong.... RSVP design and specification may have talked too much about int-serv specific things like Tspecs. We should instead have defined RSVP strictly in terms of transporting opaque QoS objects upstream and downstream...."

Our working assumption is that RSVPv2 can be created with even more modularity to enable its use in most (if not all) networking contexts. In particular, we believe that RSVPv2 can be made more suitable for use in the different parts of the network where requirements on the signaling protocol differ greatly.

2.1.3. Sender-initiated versus Receiver-initiated Signalling

RSVPv1 is receiver-oriented, which is to say that the receiver of a data flow initiates and maintains the resource reservation used for that flow. This choice was made despite the fact that sender-initiated reservations are "perhaps the most obvious choice" since sender-initiated reservations "scale poorly for large, dynamic multicast delivery trees and for heterogeneous receivers" (Section 5.1.3, RFC 1633 [RFC1633]). These two problems were solved by using receiver-initiated reservations.

Our working assumption is that relaxation of the requirement for multicast support will also allow for sender-initiated reservations without introducing scalability problems.

2.1.4. Designed for End-host to End-host Communication

RSVPv1 was primarily designed with signaling between end-host systems in mind. This communication implies a certain set of requirements on the entities involved and on the kinds of information that they need to signal.

In recent work (particularly in NSIS), it has become clear that there are in fact several different kinds of signaling conversations that may be needed in different parts of the network. Each of these kinds of signaling implies a different -- and potentially conflicting -- set of requirements on the signaling protocol. For example, the signaling requirements for the conversation between the end-host and the network may indeed need more complexity than RSVPv1 whereas the signaling needs in a DiffServ-capable access network would require significantly less.

Our working assumption is that RSVPv2 must be designed to allow an appropriate set of objects to be defined for the various "interfaces" (e.g., host-to-network, edge-to-edge, end-to-end) used in various parts of the network while not including any mandatory objects that are not applicable in all parts of the network.

2.2. Different Network Signalling Requirements/Needs and RSVP

As previously mentioned, RSVPv1 put a great deal of effort into creating a modular protocol with the ability to use those pieces that apply in a particular setting. However, while flexible, RSVPv1 does not allow for more lightweight implementations if fewer features are needed in certain network scenarios.

This section provides a (non-exhaustive) list of scenarios where there seems to be a need for new tools, either because the need for optimization is sufficiently strong or the scenario was not considered in the design of RSVPv1.

* Networks with semi-permanent trunk aggregation: In such networks the transmission links are not expensive and semi-permanent trunk aggregation can be applied.

* Networks with trusted end hosts: In these networks the security requirements are less important. Such networks are

the networks used between PSTN (Public Switched Telephone Networks) gateways and backbone routers [PAN-SSP].

* Networks with untrusted mobile end hosts: In these networks the security requirements between the first hop access router and the untrusted mobile end host are very significant. Such networks are the networks that use wireless LAN (WLAN) access [RFC2002].

* Networks that have to support fast and frequent mobility procedures (e.g., handover), where the transmission links are expensive, and the majority of the traffic is unicast. Cellular radio access networks are examples of such networks. [RAN-ISSUE].

3. Design Goals and General Features for RSVPv2-NSLP

This section briefly outlines some of the guiding principles behind the design of RSVPv2-NSLP. Moreover, the RSVPv2 NSLP general features are described. These design goals and features are in line with some of the NSIS requirements described in [Bru03] and [Hanc03].

3.1. Increased Layer Modularity and Extendibility

The essential design goal for RSVPv2 framework is to preserve the flexibility of RSVPv1 while at the same time further expanding its modularity. It can be fulfilled by using the NTLP-NSLP layer-split architecture.

The RSVPv2-NSLP protocol can be considered as an NSLP that will use a subset of the transport layer functions provided by the NTLP (see for example [WeKa03]) such as:

* Support of Path-Coupled (Path-Directed) Signaling;

* Soft state support: This feature ensures that a state will be removed if it is not periodically refreshed or explicitly removed.

* Adaptation to load sharing. Load sharing allows NF interior nodes to take advantage of multiple routes to the same

destination by sending via some or all of these available routes. The NTLP protocol level has to adapt to load sharing once it is used.

* The NTLP signaling protocol should be able to exchange local information between NSIS Forwarders located within one single administrative domain. Local information might, for example, be IP addresses, severe congestion notification, notification of successful or erroneous processing of signaling messages.

3.2. Increased Object Modularity

The purpose of the object modularity is to increase processing efficiency of RSVPv2 NTLP messages by only including those objects relevant in a particular part of the network.

RSVPv1 uses flexible object definitions that are opaque to RSVPv1 for transporting and maintaining traffic and policy control parameters. This type of object definition has certain advantages in terms of flexibility, but one of its main disadvantages is that each RSVPv1 message may contain up to fourteen classes of attribute objects. Even if some of the RSVPv1 objects are not needed in a scenario they will have to be included in RSVPv1 messages and considered as mandatory objects.

In order to achieve modularity, the RSVPv2-NSLP object structure will need to have less (possibly no) "mandatory" functionality and allow a more open object structure.

This open object structure can be solved by enhancing the RSVPv1 object structure and by introducing a concept of "profiles". A profile is a specification of which RSVPv1 objects are needed for a certain network scenario (see Section 2.2 above). In this way, the RSVPv1 messages will only carry the RSVPv1 objects that are required and specified by each profile. The profile concept makes use of profile identifiers to separate different profiles used in RSVP aware nodes.

3.3. Hierarchical Object Structure

RSVPv1, even in its simplest form, still uses objects and features that are not needed in all routers (nodes) used in a network

scenario. For example, in a network scenario with WLAN access, the QoS signaling protocol used between the access router and the untrusted mobile end host requires strong security procedures. However, the QoS signaling protocol used in the same network scenario between the same access router and another router will not require the same security procedures. Another example is a network with semi-permanent trunk aggregation, where the edges of such a network have to provide aggregator/deaggregator features, e.g., maintenance of both per micro-flow and per aggregated flow reservation states, while the interior nodes require only simpler functionality, e.g., maintenance of per aggregated flow reservation states.

The RSVPv2 framework will endeavor to improve this by providing a hierarchical structure and positioning of the RSVPv2 NSLP objects within RSVPv2 messages for each networking scenario. Each profile used for a network scenario will have to specify how the objects are structured into the RSVPv2 NSLP message and how they should be processed by a router. The objects that will be processed by all routers used in a network scenario will be placed as the first ones in the object sequence of the RSVPv2 NSLP message. Objects that will be processed only in specific routers can be placed later in the sequence.

3.4. Global and Local Objects

NSLP RSVPv2's object space will consist of globally-understood objects ("global objects") and locally-understood objects ("local objects"). The purpose of this division is to provide additional flexibility in defining the objects carried by the RSVPv2 protocol such that only those objects that are applicable in a particular setting are used.

The appropriate fora for defining these objects and how to manage the object space is obviously still a very open question.

3.5. Local information exchange

The signaling protocol MUST be able to exchange local information between NSIS Forwarders located within one single administrative domain. Local information might, for example, be IP addresses, severe congestion notification, notification of successful or erroneous processing of signaling messages.

In some cases, the NSIS signaling protocol MAY carry identification of the NSIS Forwarders located at the boundaries of a domain. However, the identification of edge should not be visible to the end host (NSIS Initiator) and only applies within one administrative domain.

3.6. Object Re-use

Obviously, whenever it is appropriate, RSVPv2 will re-use objects that are defined for RSVPv1.

3.7. Reduced Focus on Multicast

Given the complexity that multicast support introduces to QoS signalling and the fact that the vast majority of the traffic in typical IP networks is point-to-point unicast transport, RSVPv2 will be optimized to operate as a sender-initiated protocol for unicast data flows.

This should not be interpreted to mean that multicast support is not important and should not be supported. Given the increased modularity of RSVPv2 framework, it is entirely possible that appropriate objects will be defined in support of multicast.

3.8. Primarily Sender-initiated Signalling

Given a reduced focus on multicast, the "obvious" choice of sender-initiated signalling seems to be applicable to the NSLP RSVPv2. The receiver-initiated reservations will undoubtedly still be needed in some network scenarios, so the RSVPv2 framework will need to handle such reservations as well. However, this feature will be optional.

3.9. Low latency in setup

The RSVPv2 framework SHOULD allow for low latency setup of reservations in scenarios, where reservations are in a short time scale (e.g. handover in mobile environments), or where human interaction is immediately concerned (e.g., voice communication setup delay).

3.10. Highest possible network utilization

There are networking environments that require high network utilization for various reasons, and the signaling protocol SHOULD do its best ability support high resource utilization while maintaining appropriate QoS.

In networks where resources are very expensive (as is the case for many wireless networks), efficient network utilization is of critical financial importance. On the other hand there are other parts of the network where high utilization is not required.

3.11. Uni / bi-directional reservation

Both unidirectional as well as bi-direction reservations SHOULD be possible. With bi-directional reservations we mean here reservations having the same end-points. But the path in the two directions does not need to be the same. The goal of a bi-directional reservation is mainly an optimization in terms of setup delay. There is no requirements on constrains such as use the same data path etc.

3.12. End-to-end

When used end-to-end (see also [Hanc03]), the RSVPv2 NSLP protocol is initiated by an end host and is terminated by another end host. In this context, RSVPv2 NSLP can be applied as needed within all of the RSVPv2 NSLP domains between the end hosts. In the end-to-end path, RSVPv2 NSLP may be used both for intra-domain RSVPv2 NSLP signaling, as well as for inter-domain signaling.

3.13. Edge-to-edge

In this scenario (see also [Hanc03]) the RSVPv2 NSLP protocol is initiated by an edge node of a RSVPv2 NSLP domain and is terminated by another edge node of the same (or possibly different) RSVPv2 NSIS domain. RSVPv2 NSLP can be applied either within one single RSVPv2 NSLP domain, which is denoted as edge-to-edge in a single domain, or within a concatenated number of RSVPv2 NSLP domains, which is denoted as edge-to-edge in a multi-domain. When an appropriate security trust relation exists between two or more concatenated RSVPv2 NSLP domains, these concatenated RSVPv2 NSLP domains are considered, in terms of RSVPv2 NSLP, to be a single, larger RSVPv2 NSLP domain.

3.14. End-to-edge

In this scenario (see also [Hanc03]) the RSVPv2 NSLP protocol is either initiated by an end host and is terminated by an edge node or is initiated by an edge node and is terminated by an end host. In the path-coupled case, the edge node may be a proxy that is located on a boundary node of a RSVPv2 NSLP domain.

4. Overview of the RSVPv2-NSLP Framework

The RSVPv2 protocol can be considered as an NSLP that will use a subset of the transport layer functions provided by the NTLP protocol level (see for example, [WeKa03]). The RSVPv2 protocol can be used for End-to-End, Edge-to-Edge, and End-to-Edge scenarios. In the End-to-End scenario the both NSIS end nodes are functioning as NSIS Initiators (NI) and NSIS Responders (NR). In the Edge-to-Edge scenario, both NSIS edge nodes are functioning as NI, NR and NSIS Forwarders (NF). In the End-to-Edge scenario the NSIS end host is functioning as a NI or NR and the edge node is functioning as a NI, NR and NF.

The NSLP can consist of one protocol level or it can be separated into more than one hierarchical levels.

Figure 1 shows the NSIS protocol that consists of one NTLP level and one NSLP level.

```
   |-----|   |-------|   |-------|   |-------|   |-------|   |-----|
   |NSLP |<->| NSLP  |<->| NSLP  |<->| NSLP  |<->| NSLP  |<->| NSLP|
   |     |<->|       |<->|       |<->|       |<->|       |<->|     |
   |     |   |       |   |       |   |       |   |       |   |     |
    -----     -------     -------     -------     -------     -----
   |NTLP |<->| NTLP  |<->| NTLP  |<->| NTLP  |<->| NTLP  |<->| NTLP|
   |     |<->|       |<->|       |<->|       |<->|       |<->|     |
   |-----|   |-------|   |-------|   |-------|   |-------|   |-----|
     NI         NF          NF          NF          NF         NR
```

Figure 1: One level used for RSVPv2 NSLP signaling

The NSLP depicted in Figure 1 includes a set of upper-level signaling functions that are specific to particular signaling applications.

Such functions could, for example, be end to end resource reservation signaling, security, policy, billing, etc.

In Figure 2 the NSIS protocol is shown, which consists of one NTLP level and two NSLP hierarchical levels. However, the approach is quite general to more NSLP hierarchical levels: the important issue is the use of NSLP at more than one level at all.

This type of hierarchical level separation can for example, be applied for intra-domain signaling in order to maximize the scalability in an NSIS intra-domain.

The lowest hierarchical level in Figure 2 represents the NTLP level protocol. Note that in this the NF nodes are usually considered to be NTLP stateful nodes. This holds also for the NF nodes used at the boundary of a domain, i.e., the NF edge nodes. However, as described in [WeKa03], the NF interior nodes of a domain can be considered to be NF stateful nodes (see Figure 1) or, when processing optimization is required, the NF interior nodes can be NF stateless nodes (see Figure 2). The NF stateful nodes are NF NTLP aware nodes that maintain a NTLP state by using the NTLP soft state principle and are able to process and modify the application level information (NSLP) that is transported by the NTLP protocol. The NF NTLP stateless nodes are NF NTLP aware nodes that do not maintain a NTLP state, but they are able to process and modify the application level information (NSLP) that is transported by the NSLP protocol. The RSVPv2 NSLP framework depicted in Figure 2 is separated in two levels:

* the intra-domain level (located above the NTLP level), that is composed by two protocol parts the Per Domain Reservation (PDR) protocol part and the Per Hop Reservation (PHR) level. Note that these two protocol parts are simialr to the two protocols (PDR and PHR) that are described in the Resource management in Diffserv (RMD) scheme [RMD-frame].

  In order to maximize the scalability in the RSVPv2 intra-domain the complexity imposed by the combination of the RSVPv2 NSLP and NTLP has to be moved as much as possible away from the interior nodes. Therefore, the RSVPv2 NTLP separates the problem of a complex reservation within a domain from a simple reservation within a node. This is accomplished by specifying two types of resource reservation protocol parts into the RSVPv2 NSLP intra-doamin.
  The first resource reservation protocol part type is denoted as Per

Hop Reservation (PHR) that enables the signaling of the resources to be reserved per traffic class (e.g., Diffserv Per Hop Behavior (PHB) class) in each node within a domain. This protocol type is optimized to reduce the requirements placed on the functionality of the NF interior nodes of the domain. For example, the nodes that implement this protocol type do not have per flow responsibilities. This protocol can be either reservation-based or measurement-based. In the reservation-based PHR, each node keeps only one reservation state per each supported traffic class. In the measurement-based PHR no reservation states are installed and the resource availability is checked by measuring traffic (user) data load. In the NF interior nodes there is no NTLP state and there is no PDR functionality. Note that these NF interior nodes are NTLP stateless nodes.

The second protocol type is denoted as Per Domain Reservation (PDR) and is responsible for the resource reservation signaling on the NF edge nodes. The PDR is used by NF edge nodes (ingress and egress) but not by the interior nodes. This protocol introduces strict and complex requirements on the functionality implemented on the edge nodes. An example of such functionality is the mapping of the "global" traffic parameters signalled by the e2e service level (see Figure 2) to "local" parameters that are useful to the intra-domain scheme. Note that in the NF edge nodes (NF ingress and NF egress) a NTLP state is maintained and both PDR and PHR functionalities are active.

* the e2e service level is located above the PDR/PHR level and includes a set of upper-level signaling functions that are specific to particular signaling applications. Such functions could, for example, be end to end resource reservation signaling, security, policy, billing, etc.

The interface between the RSVPv2 NSLP and NTLP can be based on an API (Application Program Interface) and for the time being, is out of scope of this memo.

As shown in Figure 2, the two NSLP hierarchical levels might be applied on different NSIS entities.

This architecture for NSIS (e.g., RSVPv2) signaling can be provided by using:

*) a single end-to-end NSIS (e.g., RSVPv2) protocol that supports both NLSP hierarchical levels
*) two independent NSIS (e.g., RSVPv2) protocols: the e2e service level is supported by an end-to-end NSIS protocol, and the PDR/PHR level is supported by another edge-to-edge NSIS (e.g., RSVPv2) protocol.

```
|------|   |-------|                           |-------|   |------|
| e2e  |<--| e2e   |<-------------------------->| e2e   |<->|  e2e |
service|<->|service|                           |service|<->|service
|      |   |-------|                           |-------|   |------|
|      |   |       |                           |       |   |      |
|      |   |-------|   |-------|   |-------|   |-------|   |      |
|      |   |PDR/PHR|<->|  PHR  |<->|  PHR  |<->|PDR/PHR|   |      |NSLP
|      |   |       |   |       |   |       |   |       |   |      | ^
|      |   |-------|   |-------|   |-------|   |-------|   |      | |
-----------------------------------------------------------------------
|      |   |       |                           |       |   |      | |
|------|   |-------|   |-------|   |-------|   |-------|   |------| V
| level|<->| level |<->| level |<->| level |<->| level |<->|level |NTLP
| NTLP |<->| NTLP  |<->| NTLP  |<->| NTLP  |<->| NTLP  |<->|NTLP  |
|st.ful|   |st.ful |   |st.less|   |st.less|   |st.full|   |st.ful|
|------|   |-------|   |-------|   |-------|   |-------|   |------|
   NI         NF          NF          NF          NF         NR
             (edge)     (interior)  (interior)   (edge)
```

NTLP st.ful : NTLP stateful
NTLP st.less : NTLP stateless

Figure 2: Two levels used for the RSVPv2 NSLP

The hierarchical level separation can be provided by supporting a hierarchical object structure. In other words, the NSIS protocol objects should be structured and positioned within the NSIS messages in a hierarchical way, i.e., first the "NTLP level" objects, then the "PDR/PHR" objects and finally the "e2e service" objects.

4.1. RSVPv2 NSLP protocol features provided by the intra-domain level

The RSVPv2 NSLP protocol functions provided by the intra-domain level are composed by the protocol functions provided by the PDR and PHR protocol parts (similar to [RMD-frame], [RODA], [RIMA]).

4.1.1. PDR protocol part functions

The RSVPv2 NSLP PHR and PDR protocol parts that implement the RSVPv2 NSLP intra-domain level are listed below.

A PDR protocol part implements all or a subset of the following functions:

* Admission control and/or resource reservation signaling within a domain (i.e., on the edge nodes).

* Mapping of external QoS request provided by the e2e service level to a traffic class identifier, e.g., Diffserv Code Point (DSCP).

* Modification of an already installed RSVPv2-NSLP reservation state.

* Notification of the NF ingress node IP address to the NF egress node.

* Notification of resource availability in all the nodes located in the communication path from the NF ingress to the NF egress nodes.

* Severe congestion handling. Due to a route change or a link failure, a severe congestion situation may occur. The NF egress node is notified by PHR when such a severe congestion situation occurs. Using PDR, the egress node notifies the NF ingress node about this severe congestion situation. The NF ingress node resolves this situation by using a predefined policy, e.g., refusing new incoming flows and terminating a portion of the affected flows.

* Uni / bi-directional reservation. Both unidirectional as well as bi-direction reservations SHOULD be possible

* Notification that lost signalling messages (containing PHR and PDR information) occurred in the communication path from the ingress to the egress nodes.

4.1.2. PHR protocol part functions

A RSVPv2-NSLP PHR protocol part implements all or a subset of the following functions:

* Admission control and/or resource reservation signaling within a node.

* Management of one reservation state (i.e., PHR state) per traffic class by using a combination of the reservation soft state and explicit release signaling principles (see e.g., [RODA]). Note that the PHR state is maintained by using the NTLP soft state principle.

  Each NF node in the communication path from an NF ingress node to an NF egress node keeps only one reservation state per traffic class.

  The reservation signaling is done in terms of resource units, which may be based on a single parameter, such as bandwidth, or on more sophisticated parameters. These resources are requested dynamically per traffic class (e.g., per DSCP) and reserved on demand on all nodes in the communication path from an NF ingress node to an NF egress node. This concept is denoted as reservation based "PHR".

* Measurement of the user traffic load (see e.g., [RIMA]). This PHR function is used to check the availability of resources before flows are admitted and without installing any reservation state. That is, the resource management function that is used is actually a Measurement Based Admission Control (MBAC) algorithm, which performs measurements on the traffic (user) data load. The main advantage of this PHR group is that the PHR functionality that is executed at the edge and interior nodes will not have to maintain any reservation states. This concept is denoted as measurement based "PHR".

* Stores a pre-configured threshold value on maximal allowable traffic load (or resource units) per traffic class, e.g., PHB. When the resource management function (RMF) that is used in combination with this PHR protocol function maintains a reservation state per traffic class it also has to maintain a threshold for each traffic class (e.g., PHB) that specifies the maximum number of reservable resource units. This threshold could, for example, be statically configured.
  When the resource management function (RMF) that is used in combination with this PHR protocol function is an MBAC algorithm it also has to maintain one state per traffic class that stores the measured user traffic load associated to the traffic class, e.g., PHB and another state per traffic class, e.g., PHB that stores the

maximum allowable traffic load per traffic class, e.g., PHB.

* Severe congestion notification. This situation occurs as a result of route changes or a link failure. The PHR has to notify the NF edges about the occurrence of this situation.

  Once detected the severe congestion should be signalled to the NF(edges).
  As previously mentioned, the NF(egress) node will first be notified, after which the NF(egress) will notify the NF(ingress) node using the NSLP PDR functionality.

  Below is a list of several notification methods that can be used:

  * Greedy marking: all user data packets which pass through a severe congested interior node and are associated with a certain traffic class, e.g., DSCP, will be remarked into a another traffic class, e.g., a domain specific (DSCP)

  * Proportional marking: this method is similar to the previous method, with the difference that the number of the remarked packets is proportional to the detected overload

  * PHR message marking: only PHR objects that pass through a severely congested interior node will be marked. The marking is done by setting a special flag in the "PHR" object, i.e., "S" (see [RODA]).

The last method can only be applied on the reservation-based "PHR" concept, while the other two can be applied on both "PHR" concept types. A comparison between different severe congestion solutions is given in [CsTa02]. Note that in the RMD NSLP the PHR and PDR protocol parts have to be generated and discarded at the edge nodes (ingress and egress nodes) and not at the end hosts.

4.2. RSVPv2 NSLP protocol features provided by the e2e service level

The e2e service level protocol features that are used by this NSLP should satisfy all or a subset of the application signaling requirements provided in [Bru03]. The detailed description of these features will be included in the next updated versions of this draft.

5. RSVPv2 NSLP specification

RSVPv2 NSLP is considered in this draft to be primarily optimised for unicast and sender initiated signaling. This section provides a first step in the RSVPv2 NSLP specification.

5.1. RSVPv2 NSLP Object Classes structure

As described in [WeKa03] the NTLP message format consists of a common header, followed by a body consisting of a number of variable-length, typed transport layer "objects". The application layer (NSLP) "objects" are placed always after the transport layer "objects". Note that the application layer (NSLP) "objects" are opaque and transparent to NTLP. The NTLP message format is depicted in Figure 3.

```
       0             1              2             3
+-------------+-------------+-------------+-------------+
|                                                       |
+                   Common Header                       +
|                                                       |
+-------------+-------------+-------------+-------------+
|                                                       |
//       (Transport layer objects content)              //
|                                                       |
+-------------+-------------+-------------+-------------+
|                                                       |
//   Application layer (RSVPv2 NSLP) objects content)   //
|                                                       |
+-------------+-------------+-------------+-------------+
```

Figure 3: NTLP message format

The Application layer (RSVPv2 NSLP) depicted in Figure 3 contains RSVPv2 NSLP messages. The RSVPv2 NSLP messages and their meaning is introduced in Table 1. Furthermore, the same table identifies the NTLP message that will transport a NSLP message.

Table 1: NSLP messages

Meaning of the NSLP message	NSLP Message Type	NTLP Message Type
Initiation	NslpPathInit	PATH
Initiation	NslpResvInit	RESV
Modification	NslpPathMod	PATH
Modification	NslpResvMod	RESV
Refresh	NslpPathRef	PATH
Refresh	NslpResvRef	RESV
Path Tear down	NslpPathTear	PATHTEAR
Resv Tear down	NslpResvTear	RESVTEAR
Path Error report	NslpPathErr	PATHERROR
Resv Error report	NslpResvErr	RESVERROR
Resv Confirm	NslpResvConfirm	RESVCONFIRM

In order to have a flexible and modular RSVPv2 NSLP object class structure, we propose a grouping of signalling information into RSVPv2 NSLP object classes, called RSVPv2 NSLP Object_Classes. These will contain objects that are defined globally and/or locally. A locally defined object will allow signalling of information relevant to nodes belonging to a certain domain, while the globally defined objects will be used anywhere on the Internet. The globally defined objects are denoted as "e2e service objects" and the locally defined objects are denoted as ""PDR/PHR" objects.

In the RSVPv2 NSLP structure the following RSVPv2 NSLP Object_Classes are defined:

* Service_Class

  This object class carries the information related to the service desired from the network, i.e. QoS. This class includes all information related to the requested/expected network service. The resource reservation is related to the QoS request as well as to the response on this QoS request. This object class is flexible in order to support different kinds of QoS requests for different kinds of networking scenarios such as a end-to-edge (proxy) scenario, bi-directional reservations, receiver-initiated, etc.

* Session_ID_Class

  This object class is common for the NTLP and NSLP. In [Weka03] this object class is denoted as Session object. This class includes information related to the identification of NTLP states. This object will contain a session identifier.

  The session identifier has to identify a NTLP state and has to remain unchanged for the complete duration of a data flow. Moreover, the Session_ID_Class identifier has to be associated with the flow ID information included in the Flow_Specification_Class object. In other words, for the duration of a data flow, the session identifier remains the same while the flow ID information associated with the same data flow might change. For example, in a mobile IP scenario, during handover the IP address of a mobile node might change, causing a change in the flow ID of an ongoing data flow. However, the session identifier associated with that data flow should not change.

* Flow_Specification_Class

  This object class specifies the relation of the addressing (IP address/mask/port) to the reservation and if/how the reservation is shared between many addresses. In general, Flow_Specification contains information that identifies a particular data flow for which the specific service is requested from the network. For example, a flow ID consisting of a combination of source IP address, destination IP address, Source port, Destination port, Protocol number will be typical information belonging to the Flow_Specification_Class. This class should also contain an NSLP identifier, which identifies the NSLP type.

* Security_class

  This object class includes information related to the protection, authorization and authentication of the information in the message. This object class is optional.

* Error_message_class

  This class includes information related to the errors that

occur during reservation state processing. This object class can be considered as common for the NTLP and NSLP. In [Weka03] this object class is denoted as Error_Spec object.

5.1.1. RSVPv2 NSLP Message Structure

The exact object structure and the object sequence will have to be defined for each network scenario by a pre-defined "profile" (see Section 3.2). A profile can be either standardized or it could be an agreement between two or more participants.

Based on the above defined RSVPv2 object-class structure the format of the RSVPv2 NSLP messages may be as follows:

```
<NslpPathInit> | <NslpPathMod> | <NslpResvInit> | <NslpResvMod> |
<NslpPathTear> | <NslpResvTear> | <NslpResvConfirm> ::=
                                     <Service_Class>
                                     <Session_ID_Class>
                                     <Flow_Specification_Class>
                                     [<Security_class>]


 <NslpPathErr> | <NslpResvErr> ::=
                             <Service_Class>
                             <Session_ID_Class>
                             <Flow_Specification_Class>
                             [<Security_class>]
                             <Error_message_class>
```

5.2. RSVPv2-NSLP Objects in RSVPv2-NSLP Object_Classes

This section presents a generic method of mapping globally and locally defined RSVPv2 NSLP objects into RSVPv2 NSLP classes. Based on the definitions of the RSVPv2 NSLP object classes, an RSVPv2 NSLP Object_Class might contain globally and locally defined objects. Below is shown a possible way of mapping globally and locally defined objects into the RSVPv2 NSLP Object_Classes. The locally defined

objects are the "PHR" (Per Hop Reservation) and "PDR" (Per Domain Reservation). These objects are used for intra-domain signaling and are described in more detail in the Appendix.

```
 Service_Class:
            [<PHR>]
            [<PDR>]
            <any globally defined e2e service objects>

 Flow_Specification_Class:
   <any globally defined e2e service Flow_Specification objects>

 Session_ID_Class:
   <any globally defined e2e service session ID objects>

 Security_Class:
   <any globally defined e2e service Security objects>

 Error_Message_Class:
   <any globally defined e2e service Error_Message objects>
```

where:

```
  [] is optional for unicast and multicast support and
     sender-initiated and receiver-initiated approach
```

5.2.1. Example of mapping of RSVPv1 [RFC2205] objects in RSVPv2 object_classes"

This section gives an example of mapping the RSVPv1 objects into the RSVPv2 object_classes when RSVPv1 is optimized for unicast and sender initiated signaling.

If RSVPv1 is to be optimized for unicast and sender initiated signaling certain changes in the mandatory usage of RSVPv1 objects have to be provided. Based on the RSVPv2 object-class structure an example of a possible mapping of current RSVPv1 objects in RSVPv2 NSLP object structure is given.

The mandatory objects that will be needed in an sender-initiated NSLP RSVPv2 optimized for unicast are:

* SESSION

  It contains the IP destination address (DestAddress), the IP protocol id, and some form of generalized destination port, to define a specific session for the other objects that follow. This object contains information that is used to define the flow ID.

* SENDER_TSPEC

  Defines the traffic characteristics of a sender's data flow. Required in a Path message. This object is used to specify the QoS service required by the sender.

* SENDER_TEMPLATE

  Contains a sender IP address and perhaps some additional de-multiplexing information to identify a sender. Required in a Path message. This object contains information that is used to define the flow ID.

* TIME_VALUES

  Contains the value for the refresh period R used by the creator of the message. Required in every Path and Resv message.

* ERROR_SPEC

  Specifies an error in a PathErr, ResvErr, or a confirmation in a ResvConf message.

* POLICY_DATA

  Carries information that will allow a local policy module to decide whether an associated reservation is administratively permitted. May appear in Path, Resv, PathErr, or ResvErr message. The use of POLICY_DATA objects is not fully specified at this time; a future document will fill this gap.

* INTEGRITY

  Carries cryptographic data to authenticate the originating node and to verify the contents of this RSVPv1 message. The use of the INTEGRITY object is described in [RFC2747].

Based on the definitions of the RSVPv2 object classes, some of the RSVPv1 objects (see [RFC2205]) can be re-used in a RSVPv2 NSLP object-class structure. During the RSVPv2 NSLP design phase the RSVPv1 objects may be changed or removed completely and also some other objects may be defined as well. The goal is to reuse as much as possible of RSVPv1 objects. Based on the description of RSVPv2 NSLP object classes and the current RSVPv1 objects the mapping of RSVPv1 objects into the RSVPv2 NSLP object-class structure is rather simple. This mapping is given below and it is done for all RSVPv1 objects. Note that the Service_Class contains the PHR and PDR objects that are locally defined objects and are used for intra-domain signaling.

```
 Service_Class:
            [<PHR>]
            [<PDR>]
            <SENDER_TSPEC>
            {<ADSPEC>}
            [FLOWSPEC]
            {<RESV_CONFIRM>}
            [<POLICY_DATA>]

 Flow_Specification_Class:
            <SESSION>
            <SENDER_TEMPLATE>
            <TIME_VALUES>
            <NSLP_ID>
            {<FILTER_SPEC>}
            {<STYLE>}
            {<SCOPE>}

 Session_ID_Class:
            <SESSION>
            <NSLP_ID>

 Security_Class:
            [<INTEGRITY>]

 Error_Message_Class:
            <ERROR_SPEC>
```

where:

{} is mandatory only for multicast support and receiver-initiated approach

[] is optional for unicast and multicast support and
sender-initiated and receiver-initiated approach
<NSLP_ID> is a new object that identifies the ID of the NSLP
protocol level.

5.2.2. PDR/PHR objects

The PHR and PDR objects are locally defined objects that are used for intra-domain signaling. The information contained in these objects is similar to the information contained in the PHR and PDR messages described in [RMD-frame] and [RODA].

The PDR and PHR information is encapsulated into two different NSLP RSVPv2 object. The Appendix provides an example of PHR and PDR object specifications

5.3. RSVPv2-NSLP functionality on nodes used for inter-domain signaling

This section describes the RSVPV2-NSLP functionality on the different nodes used for inter-domain signaling. These nodes are NI (NSIS Initiator), NF (NSIS Forwarder) and NR (NSIS Responder). Note that this functionality is used in the examples provided in Section 6.

5.3.1. NI (NSIS Initiator) functionality

The NI (NSIS Initiator) functionality can be characterized as unidirectional and bi-directional reservation functionality.

5.3.1.1. Unidirectional functionality

The "e2e service" functionality of the NI(sender), after creating an NSLP reservation state, it generates an NslpPathInit message (see Section 5.1.1). The flow ID, the ID of the NSLP protocol and the time values can be included in the Flow_Specificaton_Class (e.g., <Session>, <Sender_template>, <Time_Values>, <NSLP_ID> objects). The session ID and the ID of the NSLP protocol can be included in the Session_ID_Class (e.g.,<Session> and <NSLP_ID> objects). The information that is related tothe service desired from the network, i.e., requested QoS, can be included into the Service_Class object class (e.g., <Sender_Tspec> and <Flowspec> objects). Moreover, the

Service_Class object specifies the directionality of the reservation, i.e., in this case uni-directional. The NslpPathInit is encapsulated into a NTLP PATH message (see [RFC2205]) and sent towards the NR(receiver).

The NI(sender) can receive a NslpResvInit message that is encapsulated into a RESV message. This message is associated with a NslpPathInit message that is sent earlier and that is used for a uni-directional reservation. The "e2e service" functionality of the NslpResvInit message specifies that the reservation initiated by the NslpPathInit message was successful. In this case the NI(sender), after processing the NslpResvInit message, it can start transmitting traffic user data. The "e2e service" functionality of the NI(sender) can receive a NslpPathErr message that is encapsulated into a PATHERROR message, that is associated with a NslpPathInit message sent earlier, and which is used for a uni-directional reservation. The NslpPathErr message can specify that the reservation initiated by the NslpPathInit message was unsuccessful. In this case the NI(sender), after processing the NslpPathError message, it has to delete the reservation state.

The RSVPv2-NSLP refresh procedure is a pure NTLP refresh procedure, meaning that a refresh NTLP PATH message that is periodically sent through all the NTLP stateful nodes located between NI (sender) and NR (receiver). If a NTLP state in a NTLP stateful is not refreshed on time then the NTLP functionality at this node informs the RSVPv2-NSLP state that the refresh procedure is unsuccessful. Note that the refresh NTLP PATH message may optionally carry a NslpPathRef message. In this case the information carried by the NslpPathRef message is similar to the information carried by the NslpPathInit message, (see Section 5.1.1).

The NI(sender) can receive a NslpResvRef message that is encapsulated into a RESV message. This message is associated with a NslpPathRef message that is sent earlier and that is used for a uni-directional reservation. The "e2e service" functionality of the NslpResvRef message specifies that the reservation initiated by the NslpPathRef message was successful.

The RSVPv2-NSLP "e2e service" functionality of the NI(sender) can inform the NTLP functionality of the same node to start a tear down procedure for the specific flow. A NTLP PATHTEAR message is created that is sent towards the NR (receiver). This message will tear down all the NTLP and RSVPv2-NSLP states that are associated with the

Session_ID_Class of all NTLP stateful nodes that process the NTLP PATHTEAR message. Note that the NTLP PATHTEAR message may optionally carry a NSLPPathTear message. In this case the information carried by the NslpPathTear message is similar to the information carried by the NslpPathInit message, (see Section 5.1.1).

The RSVPv2-NSLp protocol supports the modification of a reservation procedure. The "e2e service" functionality includes the request for modification of the reservation into a NslpPathMod message. This NSLP message is encapsulated into a NTLP PATH message and it is sent hop-by-hop towards the NR(receiver). The flow ID of the flow that has to be modified is included in the Flow_Specificaton_Class. The information that has to be modified is included into the Service_Class object class. (e.g., <Sender_Tspec> and <Flowspec> objects).

The NI(sender) receives a NslpResvMod message that is encapsulated into a NTLP RESV message (see Section 5.1.1). This message is associated with a NslpPathMod message that is sent earlier and that is used for a uni-directional reservation. The "e2e service" functionality of the NslpResvMod message specifies that the modification of the reservation requested by the NslpPathMod message was successful. In this case the NI(sender), after processing the NslpResvMod message, it can adjust the transmitted traffic user data to the modified reservation.

The "e2e service" functionality of the NI(sender) can receive a NslpPathErr message that is associated with a NslpPathMod message sent earlier. The NslpPathErr message can specify that the modification procedure initiated by the NslpPathMod message was not successful. In this case the "e2e functionality" of the NI (sender) will identify the modification type of the NslpPathErr message. If the modification procedure required a higher amount of reservation, then the reservation asociated to the modified flow will have to be reset to the reservation or to the amount (or type) of reservation that was stored before the modification procedure started.

5.3.1.2. Bidirectional functionality

The bi-directional reservation functionality supported by the NI(sender) is similar to a combination of two unidirectional reservation functionalities that are accomplished in opposite directions. Such a unidirectional reservation functionality is

described in Section 5.3.1.1. The main differences of the bi-directional reservation functionality with the combination of two unidirectional reservation functionalities accomplished in opposite directions are as follows:

* the reservation state specifies a bi-directional reservation
* the Service_Class object specifies that the reservation is bi-directional
* the NI(sender) does not receive the NslpResvInit, NslpResvRef and NslpResvMod messages
* the success of the reservation procedure is reported to the NI(sender) using the NslpPathInit that is sent hop-by-hop from NR(receiver) towards the NI(sender)
* the success of the refresh procedure is reported to the NI(sender) using the NslpPathRef that is sent hop-by-hop from NR(receiver) towards the NI(sender)
* the success of the refresh procedure is reported to the NI(sender) using the NslpPathMod that is sent hop-by-hop from NR(receiver) towards the NI(sender)

5.3.2. NF (NSIS Forwarder) functionality

The NF (NSIS Forwarder) functionality can be characterized as unidirectional and bi-directional reservation functionality.

5.3.2.1. Unidirectional functionality

The NslpPathInit is encapsulated into a NTLP PATH message (see [RFC2205]). The NTLP PATH message is processed by all NTLP stateful NF nodes that is passing through, up to the NR (receiver). Each node that processes the NTLP PATH message will create a NTLP state and will activate the RSVPv2-NSLP "e2e service" functionality by using the transported NslpPathInit information and it will create an RSVPv2-NSLP reservation state. This RSVPv2-NSLP reservation state will be associated to a flow ID. Note that the NTLP states have to store back-ward routing information, which are used by NTLP messages that are transported hop-by-hop in the backward direction towards the NI(sender). The NslpResvInit which is encapsulated into a RESV message will only be processed by the RSVPv2-NSLP "e2e service" functionality at each NF hop that is passing by and that is supporting the "e2e service" functionality.

The used RSVPv2-NSLP refresh procedure is a pure NTLP refresh

procedure, meaning that a refresh NTLP PATH message is periodically sent through all the NTLP stateful nodes located between NI (sender) and NR (receiver). If a NTLP state in a NTLP stateful is not refreshed on time then the NTLP functionality at this node informs the RSVPv2-NSLP state that the refresh procedure is unsuccessful. Note that the refresh NTLP PATH message may optionally carry a NSLPPathRef message.

The NTLP RESV message used during the refresh procedure is processed at each NF hop towards the NI (sender). This message will be used to report information related to how the NTLP PATH message has been processed along the path. Note that the refresh NTLP RESV message may optionally carry a NSLPResvRef message. The NslpPathRef and NslpResvRef messages are processed by the RSVPv2-NSLP "e2e service" functionality at each NF hop that are passing by and that is supporting the "e2e service" functionality.

The NF node processes a NTLP PATHTEAR message that is tearing down all the NTLP and RSVPv2-NSLP states that are associated with the Session_ID_Class of all NTLP stateful nodes that process the NTLP PATHTEAR message. Note that the NTLP PATHTEAR message may optionally carry a NslpPathTear message. The NslpPathTear message will be processed by the "e2e service" functionality.

When one of the NF nodes is not able to satisfy a NslpPathInit request the RSVPv2-NSLP "e2e service" functionality of this particular NF(router) will generate an NslpPathErr to report to NI(sender) that the NslpPathInit request could not be satisfied. This NslpPathErr message will be encapsulated into a NTLP PATHERROR message and it will be sent hop-by-hop towards the NI(sender). This message will be processed hop-by-hop by the RSVPv2-NSLP "e2e service" functionality. Each NF (router) that processes this NslpPathError message will have to to delete its associated reservation state. Note that similar to [RFC2205] the NslpPathErr could be created due to other errors in the router. The type of this error must be included into the Error_Message_Class (see Section 5.1.1). Note that the reservation state is only deleted when the NSlpPathErr message is associated to a NslpPathInit message.

When one of the NF nodes is not able to satisfy a NslpPathMod request the RSVPv2-NSLP "e2e service" functionality of this particular NF(router) will generate an NslpPathErr to report to NI(sender) that the NslpPathMod request could not be satisfied. This message will be encapsulated into a NTLP PATHERROR message and it will be sent

towards the NI(sender). The NslpPathMod message will not be forwarded further. The "e2e functionality" of the NF intermediate nodes will identify the modification type of the NslpPathErr message. If the modification procedure required a higher amount of reservation, then the nodes that modified the reservation will have to reset the reservation to the amount (or type) of reservation that was stored before the modification procedure started.

Each NTLP stateful node can process a NslpPathMod message that is carried by a modification NTLP PATH message. The RSVPv2-NSLP functionality identifies the flow that has to be modified by using its flow ID information carried by the NslpPathMod message. By using the information contained in the Service_Class, the RSVPv2-NSLP functionality is modifying the service information stored into the RSVPv2-NSLP state. The "e2e service" functionality of each NF node has to process a NslpResvMod message which is used to report information related to how the NslpPathMod message has been processed along the path. This NslpResvMod message is encapsulated into a NTLP RESV message and sent towards the NI(sender).

5.3.2.2. Bidirectional functionality

The bi-directional reservation functionality supported by the NF(router) is similar to a combination of two unidirectional reservation functionalities that are accomplished in opposite directions. Such a unidirectional reservation functionality is described in Section 5.3.2.1. The main differences of the bi-directional reservation functionality with the combination of two unidirectional reservation functionalities accomplished in opposite directions are as follows:

* the reservation state specifies a bi-directional reservation
* the Service_Class object specifies that the reservation is bi-directional
* the NF(router) does not process the NslpResvInit, NslpResvRef and NslpResvMod messages
* the success of the reservation procedure is reported to the NI(sender) using the NslpPathInit that is sent hop-by-hop from NR(receiver) towards the NI(sender)
* the success of the refresh procedure is reported to the NI(sender) using the NslpPathRef that is sent hop-by-hop from NR(receiver) towards the NI(sender)
* the success of the refresh procedure is reported to the NI(sender) using the NslpPathMod that is sent hop-by-hop from NR(receiver)

towards the NI(sender)

5.3.3. NR (NSIS Responder) functionality

The NR (NSIS Responder) functionality can be characterized as unidirectional and bi-directional reservation functionality.

The NR(receiver) can receive a NslpPathInit message that is encapsulated into a NTLP PATH message. The NR (receiver) processes the NTLP PATH message, creates a NTLP state and activates the RSVPv2-NSLP "e2e service" functionality by using the transported NslpPathInit information and it will create an RSVPv2-NSLP reservation state. This RSVPv2-NSLP reservation state will be associated to a flow ID. Note that the NTLP states have to store back-ward routing information. The "e2e service" functionality creates a NslpResvInit message that is used to report information related to how the NslpPathInit has been processed along the path. This NslpResvInit will be encapsulated into a RESV message and it will be sent on a hop-by-hop basis in the backward direction towards the NI(sender).

The RSVPv2-NSLP refresh procedure supported by the NR(receiver) is a pure NTLP refresh procedure, meaning that a refresh NTLP PATH message is periodically sent through all the NTLP stateful nodes located between NI (sender) and NR (receiver). If a NTLP state in a NTLP stateful is not refreshed on time then the NTLP functionality at this node informs the RSVPv2-NSLP state that the refresh procedure is unsuccesful. Note that the refresh NTLP PATH message may optionally carry a NSLPPathRef message. The NR (receiver) that receives a refresh NTLP PATH message will create a refresh NTLP RESV message that will be sent towards the NI (sender). This message will be used to report information related to how the NTLP PATH message has been processed along the path. Note that the refresh NTLP RESV message may optionally carry a NSLPResvRef message.

A NTLP PATHTEAR message can be received by the NR (receiver). This message will tear down all the NTLP and RSVPv2-NSLP states that are associated with the Session_ID_Class of the NR (receiver) that process the NTLP PATHTEAR message. Note that the NTLP PATHTEAR message may optionally carry a NSLPPathTear message.

When the NR(receiver) is not able to satisfy a NslpPathInit request the RSVPv2-NSLP "e2e service" functionality of the NR(receiver) will generate an NslpPathErr to report to NI(sender) that the NslpPathInit

request could not be satisfied. This NslpPathErr message will be encapsulated into a NTLP PATHERROR message and it will be sent hop-by-hop towards the NI(sender). Note that similar to [RFC2205] the NslpPathErr could be created due to other errors in the router. The type of this error must be included into the Error_Message_Class (see Section 5.1.1). When the NSlpPathErr message is associated to a NslpPathInit message then its associated reservation state will be deleted.

The NR (receiver) can receive the NslpPathMod message which is carried by the modification NTLP PATH message. The RSVPv2-NSLP functionality identifies the flow that has to be modified by using its flow ID information carried by the NslpPathMod message. By using the information contained in the Service_Class, the RSVPv2-NSLP functionality is modifying the service information stored into the RSVPv2-NSLP state. Subsequently the RSVPv2-NSLP "e2e service" functionality at the NR(receiver) creates a NslpResvMod message that will be used to report information related to how the NslpPathMod message has been processed along the path. This NslpResvMod message will be encapsulated into a NTLP RESV message and sent hop-by-hop towards the NI(sender).

When the NR(receiver) is not able to satisfy a NslpPathMod request the RSVPv2-NSLP "e2e service" functionality of this node will generate an NslpPathErr to report to NI(sender) that the NslpPathMod request could not be satisfied. This message will be encapsulated into a NTLP PATHERROR message and it will be sent towards the NI(sender). In this case the "e2e functionality" of the NR (receiver) will identify the modification type of the NslpPathErr message. If the modification procedure required a higher amount of reservation, then the reservation asociated to the modified flow will have to be reseted to the reservation or to the amount (or type) of reservation that was stored before the modification procedure started.

5.3.3.1. Bidirectional functionality

The bi-directional reservation functionality supported by the NR(receiver) is similar to a combination of two unidirectional reservation functionalities that are accomplished in opposite directions. Such a unidirectional reservation functionality is described in Section 5.3.3.1. The main differences of the bi-directional reservation functionality with the combination of two unidirectional reservation functionalities accomplished in opposite

directions are as follows:

* the reservation state specifies a bi-directional reservation
* the Service_Class object specifies that the reservation is bi-directional
* the NR(receiver) does not process the NslpResvInit, NslpResvRef and NslpResvMod messages
* the success of the reservation procedure is reported to the NI(sender) using the NslpPathInit that is sent hop-by-hop from NR(receiver) towards the NI(sender)
* the success of the refresh procedure is reported to the NI(sender) using the NslpPathRef that is sent hop-by-hop from NR(receiver) towards the NI(sender)
* the success of the refresh procedure is reported to the NI(sender) using the NslpPathMod that is sent hop-by-hop from NR(receiver) towards the NI(sender)

5.4. RSVPv2-NSLP functionality on nodes used for intra-domain signaling

This section describes the RSVPV2 functionality on the different nodes used for intra-domain signaling. These nodes are NF (ingress), NF (interior) and NF (egress). These intra-domain signaling procedures are using the NSIS protocol which consists of one NTLP level and two NSLP hierarchical levels (see Figure 2).

Intra-domain signaling is where the RSVPv2-NSLP signaling messages are originated, processed and terminated within the same domain. RSVPv2-NSLP is considered in this section to be optimized for unicast and sender-initiated protocol.

The Intra-domain signaling procedures are mainly using RSVPv2-NSLP PHR/PDR objects, (see Section 5.2.2) that are originated, processed and terminated within the same domain. Note that this functionality is used in the examples provided in Section 7.

5.4.1. NI (NSIS Initiator) functionality

The NI (NSIS Initiator) functionality can be characterized as unidirectional and bi-directional reservation functionality.

5.4.1.1. Unidirectional functionality

The unidirectional functionality supported by the NI(sender) used in this type of scenarios is identical to the functionality supported by the NI(sender) used in the inter-domain signaling scenario (see Section 5.3.1.1).

5.4.1.2. Bidirectional functionality

The bi-directional functionality supported by the NI(sender) used in this type of scenarios is identical to the functionality supported by the NI(sender) used in the inter-domain signaling scenario (see Section 5.3.1.2).

5.4.2. Functionality of NF (NSIS Forwarder) located outside NSIS intra-domain

The functionality of the NF (NSIS Forwarder) located outside the NSIS intra-domain can be characterized as unidirectional and as bi-directional reservation functionality.

5.4.2.1. Unidirectional functionality

The unidirectional functionality supported by the NF located outside the NSIS intra-domain and used in this type of scenarios is identical to the functionality supported by the NF(router) used in the inter-domain signaling scenario (see Section 5.3.2.1).

5.4.2.2. Bidirectional functionality

The bi-directional functionality supported by the NF located outside the NSIS intra-domain and used in this type of scenarios is identical to the functionality supported by the NF(router) used in the inter-domain signaling scenario (see Section 5.3.2.2).

5.4.3. NF (ingress) functionality

The NF (ingress) functionality can be characterized as unidirectional and bi-directional reservation functionality.

5.4.3.1. Unidirectional functionality

When an NslpPathInit arrives at the ingress node of a domain, i.e., NF(ingress), the RSVPv2-NSLP "e2e service" functionality creates a RSVPv2-NSLP Path reservation state. Subsequently, the RSVPv2-NSLP "PDR" protocol functionality is activated (see Figure 2) classifying the flow (i.e., Flow_Specification_Class) that is associated with the NslpPathInit message into an appropriate traffic class, e.g., Diffserv class. The RSVPv2-NSLP PDR functionality uses the RSVPv2-NSLP path state created by the NslpPathInit message and it introduces additional information that can be used to associate the PHR and PDR objects with the flow that created the RSVPv2-NSLP Path reservation state in the NF(ingress) node. The RSVPv2-NSLP PDR functionality is subsequently using the Service_Class (e.g., <SENDER_Tspec> object) and translates the requested bandwidth parameter into a number of resource units. If the QoS request is satisfied locally, then the ingress node will generate a reservation request PHR object denoted as "PHR_Resource_Request" and a reservation request PDR object denoted as "PDR_Reservation_Request", (see Section 5.2.2). The PDR object MAY contain information such as the IP address of the NF(ingress) node and the per-flow specification ID. These PHR and PDR objects are locally defined objects which are included into the Service_Class object class carried by the NslpPathInit message. The NslpPathInit message is encapsulated into a NTLP PATH message and is sent towards the NR(receiver). Note that the "PDR/PHR" functionality of the NF(ingress) node should temporarily store the TTL value, included in the IP header of any message, in the PDR state associated to the NTLP PATH message. The variable that temporarily stores the TTL value is denoted in this text as PDR_TTL_I.

The NF(ingress) can receive a NslpResvInit message that is encapsulated into a RESV message. This message is associated with a NslpPathInit message that is sent earlier and that is used for a uni-directional reservation. The "e2e service" functionality by extracting the Service_Class object from the NslpResvInit message, it can deduce that the reservation was successful. Moreover, the RSVPv2-NSLP "PDR" functionality of the NF(ingress) node is extracting the "PDR_Reservation_Report" PDR object from the Service_Class object class of the NslpResvInit message. If the initial reservation request was successful the RSVP-NSLP functionality encapsulates the NslpResvInit message into the NTLP RESV message and it is sent towards the NI(sender).

The intra-domain RSVPv2-NSLP refresh procedure is a combination of a

NTLP and a RSVPv2-NSLP "PHR/PDR" procedure. When a refresh NTLP PATH message is received by a NF(ingress) node the NTLP functionality will activate the RSVPv2-NSLP "PHR/PDR" functionality that is carried by the NslpPathRef message. By using the Flow_Specification_Class object class the "PDR" functionality can identify the RSVPv2-NSLP path state. The RSVPv2-NSLP "PDR" functionality of the NF(ingress) node will generate a refresh request PHR object denoted as "PHR_Refresh_Update" and a refresh request PDR object denoted as "PDR_Refresh_Request", (see Section 5.2.2). The PDR object MAY contain information such as the IP address of the NF(ingress) node and the per-flow specification ID. These PHR and PDR objects are locally defined objects which are included into the Service_Class object class carried by the NslpPathRef message.

The NF(ingress) can receive a NslpResvRef message that is encapsulated into a RESV message. This message is associated with a NslpPathRef message that is sent earlier and that is used for a uni-directional reservation. The "e2e service" functionality by extracting the Service_Class object from the NslpResvRef message, it can deduce that the refresh procedure was successful. Moreover, the RSVPv2-NSLP "PDR" functionality of the NF(ingress) node is extracting the "PDR_Refresh_Report" PDR object from the Service_Class object class of the NslpResvRef message. If the refresh procedure was successful the RSVP-NSLP functionality encapsulates the NslpResvRef message into the NTLP RESV message and it is sent towards the NI(sender).

The NTLP functionality of the NF(ingress) can receive a NTLP PATHTEAR message sent by the NI(sender). The NTLP PATHTEAR message may optionally carry a NslpPathTear message. The NTLP functionality activates the RSVPv2-NSLP "PDR/PHR" functionality, that is related to the Session_ID_Class class object, and that is using the "PDR" object. The RSVPv2-NSLP "PDR" functionality of the NF(ingress) node will generate a release request "PHR" object denoted as "PHR_Release_Request" and a release request PDR object denoted as "PDR_Release_Request", (see Section 5.2.2). The PDR object may contain information such as the IP address of the NF(ingress) node and the per-flow specification ID. These PHR and PDR objects are locally defined objects which are included into the Service_Class object class carried by a NslpPathTear message. All the RSVPv2-NSLP and NTLP reservations, in the NF(ingress) node that are associated to the Session_ID_Class object class will be released.

During an unsuccessful procedure, the NTLP functionality of the

NF(ingress) node can receive the a PATHERROR message that will

activate the RSVPv2-NSLP "PDR" functionality. Due to the "M" marked "PDR_Reservation_Report" object the "PDR" functionality will activate the RSVPv2-NSLP "e2e service". The RSVPv2-NSLP "e2e service" functionality of the NF(ingress) node will generate a NslpPathErr message that will be sent hop-by-hop to the NI(sender) and will be encapsulated into a NTLP PATHERROR message. This message will inform the NI(sender) that the reservation request was not successful. Simultaneously, the NF(ingress) node will start a partial explicit release procedure, for releasing the unnecessarily reserved RSVP-NSLP resources in some NF(interior) nodes for the rejected flow. In this case, the RSVP-NSLP "PDR" functionality of the NF(ingress) node will generate a "PHR_Release_Request" object, and it will include the amount of the requested resources specified the PDR state. Moreover, the RSVPv2-NSLP "PDR" functionality will create the "PDR_Reservation_Request" PDR object. The RSVPv2-NSLP "PDR" functionality of the NF(ingress) node can calculate the number of NF(interior) nodes that processed and reserved RSVPv2-NSLP resources. This number can be calculated by subtracting the value included in the PDR_TTL field that was included in the received "PDR_Reservation_Report" PDR object from the value included in the PDR_TTL_I variable that has been stored into the RSVPv2-NSLP state when the initial NslpPathInit message has been sent towards the NF(egress) node. This calculated value will be included in the TTL - IP header field of the NTLP PATHTEAR message which is generated by the NF(ingress) node and which transports the "PHR_Resource_Release" object. The "PHR_Release_Request" and "PDR_Release_Request" objects are included into a NslpPathTear message. The NslpPathTear message is transported by a NTLP PATHTEAR message.

A NTLP PATH message that encapsulates the NslpPathMod message can be received by the NTLP functionality of the NF(ingress) node. The NTLP functionality activates the RSVP-NSLP "PDR/PHR" functionality, which is associated with the Session_ID_Class object class.

When the modification request requires an increase on the number of reserved resources stored in the RSVPv2-NSLP state, then the RSVPv2-NSLP "PHR" functionality of the NF(ingress) node will have to subtract the old and already reserved number of resources from the number of resources included in the new modification request. The result of this subtraction should be introduced within a "PHR_Resource_Request" PHR object as the requested resources value. Furthermore, the number of resources that were reserved for a certain flow in the RSVPv2-NSLP state should also be replaced with the number of resources included in the modification request.

The RSVPv2-NSLP "PDR" functionality will create a

"PDR_Modification_Request" PDR object. These two objects will be included into the Service_Class of the NslpPathMod message. The NslpPathMod message is encapsulated into a modification NTLP PATH message and is sent towards the NF(egress) node.

When the modification request requires a decrease on the number of reserved resources stored in the RSVPv2-NSLP path state, then the RSVPv2-NSLP "PHR" functionality of the NF(ingress) node will have to subtract the number of resources included in the new modification request from the old and already reserved number of resources. The result of this subtraction should be introduced in an RSVPv2-NSLP "PHR_Release_Request" PHR object. Furthermore, the number of resources that were reserved in the RSVPv2-NSLP path state for a certain flow should also be replaced with the number of resources included in the modification request. The RSVPv2-NSLP "PDR" functionality will create a "PDR_Modification_Request" PDR object. These two objects will be encapsulated into a modification NTLP PATH message. This message will be sent towards the NF(egress) node.

The NF(ingress) can receive a NslpResvMod message that is encapsulated into a RESV message. This message is associated with a NslpPathMod message that is sent earlier and that is used for a uni-directional reservation. The "e2e service" functionality by extracting the Service_Class object from the NslpResvMod message, it can deduce that the modification procedure was successful. Moreover, the RSVPv2-NSLP "PDR" functionality of the NF(ingress) node is extracting the "PDR_Refresh_Report" PDR object from the Service_Class object class of the NslpResvRef message. If the modification procedure was successful the RSVP-NSLP functionality encapsulates the NslpResvMod message into the NTLP RESV message and it is sent towards the NI(sender).

If the modification procedure is not successful, the NTLP functionality of the NF(ingress) node can receive a PATHERROR message. This message carries a NslpPathErr message. The "e2e functionality" of the NF (ingress) will identify the modification type of the NslpPathErr message. The NslpPathErr message could either carry a "PDR_Modification_Report" or not. When the NslpPathErr message carries a "PDR_Modification_Report", the RSVP-NSLP "PDR" functionality will detect the "M" marked "PDR_Modification_Report" object and it will activate the RSVPv2-NSLP "e2e service". When the NslpPathErr message does not carry a "PDR_Modification_Report" message, the RSVPv2-NSLP "e2e service" is directly activated. The RSVPv2-NSLP "e2e service" functionality of

the NF(ingress) node will generate a NslpPathErr message that will be

sent hop-by-hop to the NI(sender) and will be encapsulated into a NTLP PATHERROR message. This message will inform the NI(sender) that the modification request was not successful. If the modification procedure required a higher amount of reservation, then the NF(ingress) node has to start a partial explicit release, for releasing the unnecessarily reserved RSVP-NSLP resources in some NF(interior) nodes for the modified flow. The number of unnecessarily reserved resources is found by the RSVPv2-NSLP "PHR" functionality that subtracts the old and already reserved number of resources from the number of resources included in the new modification request. The partial explicit release procedure is further accomplished in the same as the partial explicit release procedure used during the unsuccessful reservation procedure.

The NTLP signaling messages and subsequently the "PHR" and "PDR" objects might be dropped, for example due to route or link failure. The "PHR" objects that need to be sent reliable are:

PHR_Resource_Request
PHR_Refresh_Update

The reliable delivery of the "PHR_Resource_Request" object is provided by using the functionality provided by the RSVPv2-NSLP "PDR" functionality located in the NF(ingress) node. The RSVPv2-NSLP "PDR" functionality of the NF(ingress) node sends the "PHR_Resource_Request" object towards the NF(egress) node and it starts a timer. If the reply, e.g., "PDR_Reservation_Report" object, does not arrive in a predefined time it assumes that the "PHR_Resource_Request" object is lost. The reliable deliver of the "PHR_Refresh_Update" object is provided in a similar way. A timer at the NF(ingress) node is started when the "PHR_Refresh_Update" is sent towards the NF(egress) node. If the reply, e.g., "PDR_Refresh_Report" object, does not arrive in a predefined time it assumes that the "PHR_Refresh_Update" object is lost.

During a severe congestion situation, the NF(ingress) node can receive the PDR_Congestion_Report object. This object is included into a NslpPathErr message that is carried by a NTLP PATHERROR. The RSVPv2-NSLP PDR functionality of the NF(ingress) node is extracting the Pdrop blocking probability from the PDR_Congestion_Report message. Depending on the used policy the NF(ingress) node might terminate the flow, i.e., for a higher blocking probability there is a higher chance that the flow is terminated. If a flow needs to be terminated, then for this flow, the NF(ingress) node will generate a "PHR_Release_Request" object that will be included into the

Service_Class of the NslpPathTear message. This message will be

transported by a NTLP PATHTEAR message towards the NF(egress). Furthermore, the RSVPv2-NSLP "e2e service" functionality in the NF(ingress) node will create a NslpPathErr that will be encapsulated into a NTLP PATHERROR that will be sent towards the NI(sender) to notify that an error occurred.

5.4.3.2. Bidirectional functionality

The bi-directional reservation functionality supported by the NF(ingress) is similar to a combination of two unidirectional reservation functionalities that are accomplished in opposite directions. Such a unidirectional reservation functionality is described in Section 5.4.3.1. The main differences of the bi-directional reservation functionality with the combination of two unidirectional reservation functionalities accomplished in opposite directions are as follows:

* the reservation state specifies a bi-directional reservation
* the Service_Class object specifies that the reservation is bi-directional
* the NF(ingress) does not receive the NslpResvInit, NslpResvRef and NslpResvMod messages
* the success of the reservation procedure is reported to the NI(sender) using the NslpPathInit that is sent hop-by-hop from NR(receiver) towards the NI(sender)
* the success of the refresh procedure is reported to the NI(sender) using the NslpPathRef that is sent hop-by-hop from NR(receiver) towards the NI(sender)
* the success of the refresh procedure is reported to the NI(sender) using the NslpPathMod that is sent hop-by-hop from NR(receiver) towards the NI(sender)
* the PDR_Reservation_Report object used to report a successful reservation procedure is carried by the NslpPathInit that is sent hop-by-hop from NF(egress) towards the NF(ingress)
* the PDR_Refresh_Report object used to report a successful refresh procedure is carried by the NslpPathInit that is sent hop-by-hop from NF(egress) towards the NF(ingress)
* the PDR_Modification_Report object used to report a successful modification procedure is carried by the NslpPathInit that is sent hop-by-hop from NF(egress) towards the NF(ingress)
* the NF(egress) node can initiate an explicit partial release procedure towards the NF(ingress) node.

5.4.4. NF (interior) functionality

The NF (interior) functionality can be characterized as unidirectional and bi-directional reservation functionality.

5.4.4.1. Unidirectional functionality

The initiation NTLP PATH message is processed by all NTLP stateless NF(interior) nodes that is passing through, up to the NF (egress). Each stateless NF(interior) node that processes the NTLP PATH message it will not create a NTLP state but it will activate the RSVPv2-NSLP functionality by using the transported NslpPathInit message. The RSVPv2-NSLP "PHR" functionality of these NF(interior) nodes will use the information included in the PHR object ("PHR_Resource_Request") and it will identify the ID of the traffic class, e.g., Diffserv class. If there is enough bandwidth capacity, it will reserve the requested resources. The NF(interior) node reserves the requested resources by e.g., adding the requested amount to the total amount of reserved resources for that traffic class, e.g., Diffserv class.

It is possible that one of the NTLP stateless NF(interior) is not able to satisfy the request carried by the "PHR_Resource_Request" PHR object. The RSVPv2-NSLP "PHR" functionality of this NF(interior) node will mark the "M" field of the "PHR_Resource_Request" object. The RSVPv2-NSLP "PHR" functionality will also include the number of previous NF(interior) nodes that successfully processed the RSVPv2-NSLP "PHR_Resource_Request" PHR object (see Appendix). This number can, for example, be identified by the TTL (Time-To-Live) value included in the IP header of the received NTLP PATH message. Note that each time that an IP packet passes a node, its TTL value is decreased by one. In particular, the NF(interior) node that is not admitting the reservation request initiated by the "PHR_Resource_Request" PHR object will copy the TTL value included in the IP header of the received NTLP PATH message that carries the "PHR_Resource_Request" object into the "PDR_TTL" field of "PDR_Reservation_Request" PDR object. Moreover, the "T" field of the "PHR" object (see Appendix) is set to "1". These "PHR" and "PDR" objects are included in the NslpPathInit message. The NslpPathInit message is encapsulated into a NTLP PATH message. This NslpPathInit message is sent towards the NF(egress) node, which will be transported by a NTLP PATH message. Any NF(interior) node receiving a PATH message will activate the RSVPv2-NSLP "PHR" functionality. If the "PHR_Resource_Request" PHR object is "M" marked, then the RSVPv2-NSLP "PHR" functionality will not further process the "PHR"

object.

The refresh NTLP PATH message is processed by all NTLP stateless NF(interior) nodes that is passing through, up to the NF (egress). Each node that processes the refresh NTLP PATH message it will refresh the NTLP state associated with the session ID, i.e., information included into the Session_ID_Class object class. The NTLP level functionality of the NTLP stateless NF(interior) nodes receiving the refresh NTLP PATH message will activate the RSVPv2-NSLP "PHR" functionality. The RSVPv2-NSLP "PHR" functionality of these NF(interior) nodes will use the information included in the PHR object ("PHR_Refresh_Request") and it will identify the ID of the traffic class, e.g., Diffserv class. This object will refresh the requested resources included in the "PHR_Refresh_Update" object.

The NTLP PATHTEAR message is processed by all NTLP stateless NF(interior) nodes that is passing through, up to the NF (egress). Each node that processes the PATHTEAR message will activate the RSVPv2-NSLP "PHR" functionality by using the transported RSVPv2-NSLP "PHR" object. The NTLP functionality in the NTLP stateless NF(interior) node that receives the PATHTEAR message will pass the NslpPathTear message to the RSVPv2-NSLP functionality. The NslpPathTear message contains the "PHR_Release_Request" and "PDR_Release_Request" PHR and PDR objects, respectively. The RSVPv2-NSLP "PHR" functionality of this NF(interior) node will use the information included in the PHR object ("PHR_Release_Request") and it will identify the ID of the traffic class, e.g., Diffserv class. This object will subtract the requested resources included in the "PHR_Release_Request" object from the total reserved amount of resources stored in the traffic class state. Moreover, its TTL value of the NTLP PATHTEAR message is decremented by one. If this value becomes zero, the "PHR_Resource_Release" object reached an NF(interior) node that marked the "PHR_Resource_Request" object during an unsuccessful procedure and the NTLP PATHTEAR message will be dropped. Otherwise, the NTLP PATHTEAR message propagates towards the NR(receiver).

Each stateless NF(interior) node that receives the modification NTLP PATH message will activate the RSVPv2-NSLP "PHR" functionality. The RSVPv2-NSLP "PHR" functionality of each stateless NF(interior)node processes the "PHR_Resource_Request" and "PHR_Release_Request" objects included in the modification NTLP PATH message as typical "PHR_Resource_Request" and "PHR_Release_Request" objects, respectively.

After detecting the severe congestion situation, the RSVPv2-NSLP "PHR" functionality of the NF(interior) node will notify the NF(egress) node by using remarking of user data bytes that pass through the node. Proportionally to the detected overload the NF(interior) node will remark a number of user data bytes which are passing through a severe congested interior node and are associated with a certain traffic class, e.g., DSCP, into a domain specific DSCP.

5.4.4.2. Bidirectional functionality

The bi-directional reservation functionality supported by the NF(interior) is similar to a combination of two unidirectional reservation functionalities that are accomplished in opposite directions. Such a unidirectional reservation functionality is described in Section 5.4.4.1. The main differences of the bi-directional reservation functionality with the combination of two unidirectional reservation functionalities accomplished in opposite directions are as follows:

* the PDR_Reservation_Report object used to report a successful reservation procedure is carried by the NslpPathInit that is sent hop-by-hop from NF(egress) towards the NF(ingress)
* the PDR_Refresh_Report object used to report a successful refresh procedure is carried by the NslpPathInit that is sent hop-by-hop from NF(egress) towards the NF(ingress)
* the PDR_Modification_Report object used to report a successful modification procedure is carried by the NslpPathInit that is sent hop-by-hop from NF(egress) towards the NF(ingress)
* the NF(egress) node can initiate an explicit partial release procedure towards the NF(ingress) node.

5.4.5. NF (egress) functionality

The NF (egress) functionality can be characterized as unidirectional and bi-directional reservation functionality.

5.4.5.1. Unidirectional functionality

The behavior of the NF(egress) node on admission or rejection of the NslpPathInit message that contains the "PHR_Resource_Request" object is the same as in the NF(interior) nodes. After processing the "PHR_Resource_Request" object, the RSVPv2-NSLP functionality of the NF(egress) node uses the "PDR_Reservation_Request" object and creates/identifies the flow specification ID and the state associated with it. Subsequently, the RSVPv2-NSLP "e2e service" functionality is activated and by using the information contained in the Flow_Specification_Class it will create an RSVPv2-NSLP Path reservation. If the request is admitted, the RSVPv2-NSLP "PDR" functionality of the NF(egress) node will report the successful reservation to the RSVPv2-NSLP "PDR" functionality of the NF(ingress) node by using a "PDR_Reservation_Report" PDR object. This object is temporarilty stored until a NslpresvInit message arrives that is carried by a NTLP RESV message, that was sent by the NR(receiver) and that is associated with an earlier processed NslpPathInit message. This "PDR_Reservation_Report" PDR object will be included into the Service_Class object class of the NslpResvInit message. The NTLP PATH message is forwarded towards the NR (receiver). Note that this NTLP PATH message will not include the "PDR/PHR" object information. If the "PHR_Resource_Request" PHR object is "M" marked, then the RSVPv2-NSLP "PHR" functionality will activate the RSVPv2-NSLP "PDR" functionality which will create and "M" mark the "PDR_Reservation_Report" object. Moreover, if the "T" field value included in the "PHR" object is "1" then the PDR_TTL value that was included by the NF(interior) node into the "PDR_Reservation_Request" object will be copied into the PDR_TTL value of the "PDR_Reservation_Report" object. The "PDR" object will be included in an NslpPathErr message. The NslpPathErr message will be encapsulated into a NTLP PATHERROR message. The NslpPathError message will only be processed by the NF(ingress) node.

When the NF(egress) node receives a NslpPathError message which is carried by a PATHERROR message will have to identify the error type. If the NSlpPathErr message is associated to a NslpPathInit message then the NslpPathErr will have to be encapsulated into a NTLP PATHERROR message and sent towards the NI(sender). Moreover, its associated RSVPv2-NSLP state has to be deleted. The NTLP PATHERROR message will be processed within the NSIS intra-domain only by the NF(ingress) node. If the NslpPathErr message is associated to a NslpPathMod request, then then the NslpPathErr will have to be encapsulated into a NTLP PATHERROR message and sent towards the

NI(sender). Moreover, if the modification procedure required a higher

amount of reservation, then the nodes that modified the reservation will have to reset the reservation to the amount (or type) of reservation that was stored before the modification procedure started.

The NF(egress) node can receive a NslpResvInit message, carried by a NTLP RESV message which was sent by the NR(receiver) and is associated with an earlier processed NslpPathInit message. The RSVPv2-NSLP "PDR" functionality of the NF(egress) node will report the successful reservation to the RSVPv2-NSLP "PDR" functionality of the NF(ingress) node by using a "PDR_Reservation_Report" PDR object. This object will be included into the Service_Class object class of the NslpResvInit message. Note that this message is processed in a NSIS intra-domain only by the NF(egress) and NF(ingress) nodes. The NF(interior)nodes are not processing this message.

The NF(egress) node can receive a refresh NTLP PATH message. The NF(egress) node that processes the refresh NTLP PATH message it will refresh the NTLP state associated with the session ID included into the Session_ID_Class object class. Furthermore, it will activate the RSVPv2-NSLP "PHR" functionality by using the transported RSVPv2-NSLP "PHR" object. The behavior of the RSVPv2-NSLP "PHR" functionality in the NF(egress) node is similar to the RSVPv2-NSLP "PHR" functionality provided in the NF(interior) nodes. If the refresh is admitted, the RSVPv2-NSLP "PDR" functionality of the NF(egress) node will report the successful refresh procedure to the RSVPv2-NSLP "PDR" functionality of the NF(ingress) node by using a "PDR_Refresh_Report" PDR object. This object is temporarilty stored until a NslpResvRef message arrives that is carried by a NTLP RESV message, that was sent by the NR(receiver) and that is associated with an earlier processed NslpPathRef message. This "PDR_Refresh_Report" PDR object will be included into the Service_Class object class of the NslpResvRef message. The NTLP PATH message is forwarded towards the NR (receiver). Note that this NTLP PATH message will not include the "PDR/PHR" object information.

The NF(egress) node can receive a NslpResvRef message, carried by a NTLP RESV message which was sent by the NR(receiver) and is associated with an earlier processed NslpPathRef message. The RSVPv2-NSLP "PDR" functionality of the NF(egress) node will report the successful refresh PDR/PHR procedure to the RSVPv2-NSLP "PDR" functionality of the NF(ingress) node by using a "PDR_Refresh_Report" PDR object. This object will be included into the Service_Class object class of the NslpResvRef message. Note that this message is

processed in a NSIS intra-domain only by the NF(egress) and

NF(ingress) nodes. The NF(interior) nodes are not processing this message.

The NF(egress) node that processes the PATHTEAR message it will activate the RSVPv2-NSLP "PHR" functionality by using the transported "PHR" object. The behavior of the RSVPv2-NSLP "PHR" functionality in the NF(egress) node is similar to the RSVPv2-NSLP "PHR" functionality provided in the NF(interior) nodes. Furthermore, the NTLP state is released and the PATHTEAR message is forwarded towards the NR (receiver). Note that this PATHTEAR message will not include the "PDR/PHR" objects.

The behavior of the NF(egress) node related to the modification procedure is the same as in the NF(interior) nodes. After receiving the modification NTLP PATH message the RSVPv2-NSLP is processing either the "PHR_Resource_Request" or "PHR_Release_Request" object. After that the RSVPv2-NSLP functionality of the NF(egress) node uses the "PDR_Modification_Request" object and identifies the flow specification ID and the RSVPv2-NSLP state associated with it. Subsequently, the RSVPv2-NSLP "e2e service" functionality is activated and by using the information contained in the Flow_Specification_Class it will modify the reservation stored into the RSVPv2-NSLP path state. If the modification is admitted, the RSVPv2-NSLP "PDR" functionality of the NF(egress) node will report the successful modification procedure to the RSVPv2-NSLP "PDR" functionality of the NF(ingress) node by using a "PDR_Modification_Report" PDR object. This object is temporarily stored until a NslpResvMod message arrives that is carried by a NTLP RESV message, which was sent by the NR(receiver) and that is associated with an earlier processed NslpPathMod message. This "PDR_Modification_Report" PDR object will be included into the Service_Class object class of arriving NslpResvMod message. The modification NTLP PATH message is forwarded towards the NR (receiver). Note that this NTLP PATH message will not include the "PDR/PHR" object information.

The NF(egress) node can receive a NslpResvMod message, carried by a NTLP RESV message which was sent by the NR(receiver) and is associated with an earlier processed NslpPathMod message. The RSVPv2-NSLP "PDR" functionality of the NF(egress) node will report the successful modification procedure PDR/PHR procedure to the RSVPv2-NSLP "PDR" functionality of the NF(ingress) node by using a "PDR_Modification_Report" PDR object. This object will be included into the Service_Class object class of the NslpResvMod message. Note

that this message is processed in a NSIS intra-domain only by the

NF(egress) and NF(ingress) nodes. The NF(interior) nodes are not processing this message.

During a severe congestion situation marked data packets arrive at the NF(egress) node. When the marked packets arrive at the NF(egress) node, the NF(egress) node will generate a "PDR_Congestion_Report" object and send it to the NF(ingress) node containing the over-allocation volume of the flow in question, e.g., a blocking probability. The "PDR_Congestion_Report" PDR object should be included into a NslpPathErr and transported by a NTLP PATHERROR message. For each flow ID, the RSVPv2-NSLP PDR functionality at the NF(egress) node will count the number of marked bytes (# marked bytes) and the number of unmarked bytes (#unmarked bytes). Based on this information the RSVPv2-NSLP PDR functionality at the NF(egress) node will have to calculate the blocking estimation of data. The NF(egress) node will actually calculate the blocking probability (Pdrop), which will be used by an NF(ingress) node to block this particular flow. The blocking probability is calculated as the ratio between the dropped bytes and the maximum number of bytes that can be supported by the interior node:

Pdrop = (# marked bytes)/(# marked bytes + # unmarked bytes)

This blocking probability will be included in the "PDR_Congestion_Report" object that will be sent to the NF(ingress).

5.4.5.2. Bidirectional functionality

The bi-directional reservation functionality supported by the NF(egress) is similar to a combination of two unidirectional reservation functionalities that are accomplished in opposite directions. Such a unidirectional reservation functionality is described in Section 5.4.5.1. The main differences of the bi-directional reservation functionality with the combination of two unidirectional reservation functionalities accomplished in opposite directions are as follows:

* the reservation state specifies a bi-directional reservation
* the Service_Class object specifies that the reservation is bi-directional
* the NF(egress) does not process the NslpResvInit, NslpResvRef and NslpResvMod messages
* the success of the reservation procedure is reported to the NI(sender)

using the NslpPathInit that is sent hop-by-hop from NR(receiver) towards the NI(sender)
* the success of the refresh procedure is reported to the NI(sender) using the NslpPathRef that is sent hop-by-hop from NR(receiver) towards the NI(sender)
* the success of the refresh procedure is reported to the NI(sender) using the NslpPathMod that is sent hop-by-hop from NR(receiver) towards the NI(sender)
* the PDR_Reservation_Report object used to report a successful reservation procedure is carried by the NslpPathInit that is sent hop-by-hop from NF(egress) towards the NF(ingress)
* the PDR_Refresh_Report object used to report a successful refresh procedure is carried by the NslpPathInit that is sent hop-by-hop from NF(egress) towards the NF(ingress)
* the PDR_Modification_Report object used to report a successful modification procedure is carried by the NslpPathInit that is sent hop-by-hop from NF(egress) towards the NF(ingress)
* the NF(egress) node can initiate an explicit partial release procedure towards the NF(ingress) node.

5.4.6. NR (NSIS Responder) functionality

The NR (NSIS Responder) functionality can be characterized as unidirectional and bi-directional reservation functionality.

5.4.6.1. Unidirectional functionality

The unidirectional functionality supported by the NR(receiver) used in this type of scenarios is identical to the functionality supported by the NR(receiver) used in the inter-domain signaling scenario (see Section 5.3.3.1).

5.4.6.2. Bidirectional functionality

The bi-directional functionality supported by the NR(receiver) used in this type of scenarios is identical to the functionality supported by the NR(receiver) used in the inter-domain signaling scenario (see Section 5.3.3.2).

6. Example of RSVPv2-NSLP Inter-domain signaling procedures

This section gives a brief description of the main flow diagram used by the RSVPv2-NSLP protocol for inter-domain signaling procedures. RSVPv2-NSLP is considered in this section to be optimized for unicast and sender-initiated protocol. This means that the NslpPathInit initiates and activates a reservation in each node that is passing through. The Inter-domain signaling procedures are mainly using globally defined objects, i.e., e2e service objects, see Figure 1.

6.1. Normal operation for uni-directional reservation

This section presents examples of RSVPv2-NSLP inter-domain signaling procedures for RSVPv2-NSLP normal operation, i.e., successful reservation and operation without failures. In this example only the uni-directional feature is considered and it is assummed that no intra-domain signaling procedures are used.

Figure 4 shows the main flow diagram used by the RSVPv2-NSLP protocol. The NI(sender), after creating an NSLP reservation state, generates an NslpPathInit. The flow ID, the ID of the NSLP protocol and the time values can be included in the Flow_Specificaton_Class (e.g., <Session>, <Sender_template>, <Time_Values>, <NSLP_ID> objects). The session ID and the ID of the NSLP protocol can be included in the Session_ID_Class (e.g., <Session> and <NSLP_ID> objects). The information that is related to the service desired from the network, i.e., requested QoS, can be included into the Service_Class object class (e.g., <Sender_Tspec> and <Flowspec> objects).

The NslpPathInit is encapsulated into a NTLP PATH message (see [RFC2205]). The NTLP PATH message is processed by all NTLP stateful nodes that is passing through, up to the NR (receiver). Each node that processes the NTLP PATH message will create a NTLP state and will activate the RSVPv2-NSLP "e2e service" functionality by using the transported NslpPathInit information and it will create an RSVPv2-NSLP reservation state. This RSVPv2-NSLP reservation state will be associated to a flow ID. Note that the NTLP states have to store back-ward routing information, which are used by NTLP messages that are transported hop-by-hop in the backward direction towards the NI(sender).

When the NR(receiver) receives NslpPathInit the RSVPv2-NSLP "e2e service" functionality creates an NslpResvInit message that is used

to report information related to how the NslpPathInit has been processed along the path. This NslpResvInit will be encapsulated into a RESV message and it will only be processed by the RSVPv2-NSLP "e2e service" functionality at each hop that is passing by and that is supporting the "e2e service" functionality.

After the successful reception of the NslpResvInit message the NI(sender) can start transmitting traffic user data.

Figure 4 also shows how the refresh procedure is performed. The RSVPv2-NSLP refresh procedure is a pure NTLP refresh procedure, meaning that a refresh NTLP PATH message that is periodically sent through all the NTLP stateful nodes located between NI (sender) and NR (receiver). If a NTLP state in a NTLP stateful is not refreshed on time then the NTLP functionality at this node informs the RSVPv2-NSLP state that the refresh procedure is unsuccesful. Note that the refresh NTLP PATH message may optionally carry a NSLPPathRef message. NR (receiver) will create a refresh NTLP RESV message that will be sent towards the NI (sender). This message will be used to report information related to how the NTLP PATH message has been processed along the path. Note that the refresh NTLP RESV message may optionally carry a NSLPResvRef message.

```
 NI (sender)          NF (router)          NF (router)       NR (receiver)
NTLP stateful       NTLP stateful         NTLP stateful     NTLP stateful
       PATH(NslpPathInit)  |                    |                    |
      |------------------->|                    |                    |
      |                    |                    |                    |
      |                    |PATH(NslpPathInit)  |                    |
      |                    |------------------->| PATH(NslpPathInit) |
      |                    |                    |------------------->|
      |                    |                    | RESV(NslpResvInit) |
      |                    | RESV(NslpResvInit) |<-------------------|
      |                    |<-------------------|                    |
      |RESV(NslpResvInit)  |                    |                    |
      |<-------------------|                    |                    |
      |                    | Traffic(user) Data |                    |
      |------------------->|------------------->|------------------->|
      |                    |                    |                    |
      |PATH([NslpPathRef]) |                    |                    |
      |------------------->|                    |                    |
      |                    |                    |                    |
      |                    |PATH([NslpPathRef]) |                    |
      |                    |------------------->|PATH([NslpPathRef]) |
      |                    |                    |------------------->|
      |                    |                    | RESV([NslpResvRef])|
      |                    | RESV([NslpResvRef])|<-------------------|
      |                    |<-------------------|                    |
      |RESV([NslpResvRef])|                    |                    |
      |<-------------------|                    |                    |
```

Figure 4: Inter-domain signaling normal operation for successful reservation

Figure 5 depicts the RSVPv2-NSLP tearing down procedure. In Figure 5 The RSVPv2-NSLP "e2e service" functionality of the NI(sender) informs the NTLP functionality of the same node to start a tear down procedure for the specific flow. A NTLP PATHTEAR message is created that is sent towards the NR (receiver). This message will tear down all the NTLP and RSVPv2-NSLP states that are associated with the Session_ID_Class of all NTLP stateful nodes that process the NTLP PATHTEAR message. Note that the NTLP PATHTEAR message may optionally carry a NSLPPathTear message.

```
 NI (sender)          NF (router)          NF (router)       NR (receiver)
NTLP stateful        NTLP stateful        NTLP stateful      NTLP stateful
         |PATHTEAR([NslpPathTear])                |                    |
         |------------------->|                  |                    |
         |                    |PATHTEAR([NslpPathTear])               |
         |                    |------------------->|PATHTEAR([NslpPathTear])
         |                    |                  |------------------->|
```

Figure 5: Inter-domain signaling normal operation for tearing down a reservation initiated by NI (sender)

Figure 6 shows the main flow diagram used by the RSVPv2-NSLP protocol in case of an unsuccessful reservation assuming that no intra-domain signaling procedures are used. In this situation only the uni-directional feature is considered. In this situation the NslpPathInit and NTLP PATH messages are created and transmitted in the same way as during the successful reservation. The main difference is related to the fact that one of the NF(routers) is not able to satisfy the NslpPathInit request. In this situation this RSVPv2-NSLP "e2e service" functionality of this particular NF(router) will generate an NslpPathErr to report to NI(sender) that the NslpPath request could not be satisfied. This NslpPathErr message will be encapsulated into a NTLP PATHERROR message and it will be sent hop-by-hop towards the NI(sender). This message will be processed hop-by-hop by the RSVPv2-NSLP "e2e service" functionality.

```
NI (sender)         NF (router)          NF (router)       NR (receiver)
NTLP stateful       NTLP stateful       NTLP stateful      NTLP stateful
      PATH(NslpPathInit)  |                    |                    |
     |------------------->|                    |                    |
     |                    |                    |                    |
     |                    | PATH(NslpPathInit) |                    |
     |                    |------------------->|                    |
     |                    |                    |                    |
     |                PATHERROR(NslpPathErr)   |                    |
     |                    |<-------------------|                    |
     |PATHERROR(NslpPathErr)                   |                    |
     |<-------------------|                    |                    |
```

Figure 6: Inter-domain signaling normal operation for unsuccessful reservation

Figure 7 shows the main flow diagram used by the RSVPv2-NSLP protocol in case of a modification of a reservation procedures assuming that no intra-domain signaling procedures are used. In this situation only the uni-directional feature is considered. The modification of the reservation is included in a new NslpPathMod message. This NSLP message is encapsulated into a NTLP PATH message and it is sent hop-by-hop towards the NR(receiver). The flow ID of the flow that has to be modified is included in the Flow_Specificaton_Class. The information that has to be modified that is included into the Service_Class object class. (e.g., <Sender_Tspec> and <Flowspec> objects).

The NslpPathMod information is read by each NTLP stateful node that processes the NTLP PATH message. The RSVPv2-NSLP functionality identifies the flow that has to be modified by using its flow ID information carried by the NslpPathMod message. By using the information contained in the Service_Class, the RSVPv2-NSLP functionality is modifying the service information stored into the RSVPv2-NSLP state.

Subsequently the RSVPv2-NSLP "e2e service" functionality at the NR(receiver) will create an NslpResvMod that will be used to report information related to how the NslpPathMod message has been processed along the path. This NslpResvMod message will be encapsulated into a NTLP RESV message and sent hop-by-hop towards the NI(sender).

When a NSIS node is not able to satisfy a NslpPathMod request the RSVPv2-NSLP "e2e service" functionality of this node will generate an NslpPathErr to report to NI(sender) that the NslpPathMod request could not be satisfied. This message will be encapsulated into a NTLP PATHERROR message and it will be sent towards the NI(sender). In this case the "e2e functionality" of any NSIS node will identify the modification type of the NslpPathErr message. If the modification procedure required a higher amount of reservation, then the reservation asociated to the modified flow will have to be reseted to the reservation or to the amount (or type) of reservation that was stored before the modification procedure started.

```
NI (sender)          NF (router)         NF (router)      NR (receiver)
NTLP stateful       NTLP stateful       NTLP stateful     NTLP stateful
    PATH(NslpPathMod)   |                    |                    |
   |------------------->|                    |                    |
   |                    | PATH(NslpPathMod)  |                    |
   |                    |------------------->|PATH(NslpPathMod)   |
   |                    |                    |------------------->|
   |                    |                    |  RESV(NslpResvMod) |
   |                    | RESV(NslpResvMod)  |<-------------------|
   |                    |<-------------------|                    |
   | RESV(NslpResvMod)  |                    |                    |
   |<-------------------|                    |                    |
```

Figure 7: Inter-domain signaling normal operation for modification of reservation

6.2. Normal operation for bi-directional reservation

This section gives one example of inter-domain signaling for a successful and one example of inter-domain signaling for an unsuccessful bi-directional reservation. Figure 8 shows the flow diagram of inter-domain signaling used by the RSVPv2-NSLP protocol in case of a successful bi-directional reservation.

The bi-directional successful reservation is similar to a combination of two unidirectional successful reservations that are accomplished in opposite directions. The main differences of the bi-directional successful reservation procedure with the combination of two unidirectional successful reservations accomplished in opposite directions are as follows:

* the reservation state specifies a bi-directional reservation
* the Service_Class object specifies that the reservation is bi-directional
* the NSIS aware nodes do not receive the NslpResvInit, NslpResvRef and NslpResvMod messages
* the success of the reservation procedure is reported to the NI(sender) using the NslpPathInit that is sent hop-by-hop from NR(receiver) towards the NI(sender)
* the success of the refresh procedure is reported to the NI(sender) using the NslpPathRef that is sent hop-by-hop from NR(receiver) towards the NI(sender)
* the success of the refresh procedure is reported to the NI(sender)

using the NslpPathMod that is sent hop-by-hop from NR(receiver) towards the NI(sender)

Figure 9 shows the flow diagrams of inter-domain signaling used by the RSVPv2-NSLP protocol in case of a unsuccessful bi-directional reservation. The bi-directional unsuccessful reservation is similar to a combination of two unidirectional unsuccessful reservations that are accomplished in opposite directions. The main differences of the bi-directional unsuccessful procedure with the combination of two unidirectional successful reservations accomplished in opposite directions are as follows:

* the reservation state specifies a bi-directional reservation
* the Service_Class object specifies that the reservation is bi-directional
* the NSIS aware nodes do not process the NslpResvInit, NslpResvRef and NslpResvMod messages

```
NI (sender)          NF (router)          NF (router)       NR (receiver)
NTLP stateful       NTLP stateful         NTLP stateful     NTLP stateful
       PATH(NslpPathInit)  |                    |                    |
      |------------------->|                    |                    |
      |                    |           PATH(NslpPathInit)            |
      |                    |----------------------------------------->|
      |                    |                    |                    |
      |                    |                    |  PATH(NslpPathInit)|
      |                    |                    |<-------------------|
      |             PATH(NslpPathInit)          |                    |
      |<----------------------------------------|                    |
      |                    |                    |                    |
      |                    |                    |                    |
      |                    | Traffic(user) Data |                    |
      |------------------->|----------------------------------------->|
      |                    |                    |                    |
      |                    | Traffic(user) Data |                    |
      |<----------------------------------------|<-------------------|
      |                    |                    |                    |
      |                    |                    |                    |
      |PATH([NslpPathRef]) |                    |                    |
      |------------------->|                    |                    |
      |                    |PATH([NslpPathRef]) |                    |
      |                    |----------------------------------------->|
      |                    |                    |                    |
      |                    |                    | PATH([NslpPathRef])|
      |           PATH([NslpPathRef])           |<-------------------|
      |<----------------------------------------|                    |
      |                    |                    |                    |
```

Figure 8: Inter-domain signaling for bi-directional
reservation in case of a successful reservation

```
NI (sender)          NF (router)          NF (router)       NR (receiver)
NTLP stateful       NTLP stateful         NTLP stateful     NTLP stateful
       PATH(NslpPathInit)  |                    |                    |
      |------------------->|                    |                    |
      |                    |           PATH(NslpPathInit)            |
      |                    |----------------------------------------->|
      |                    |                    |                    |
      |                    |                    |  PATH(NslpPathInit)|
      |                    |                    |<-------------------|
      |                    |                    |                    |
      |                    |                  PATHERROR(NslpPathErr)|
      |                    |                    |------------------->|
      |                    |         PATHERROR(NslpPathErr)          |
      |                    |<----------------------------------------|
      | PATHERROR(NslpPathErr)                  |                    |
      |<-------------------|                    |                    |
```

Figure 9: Inter-domain signaling for bi-directional reservation in case of an unsuccessful reservation

7. Example of RSVPv2-NSLP Intra-domain signaling procedures

This section gives a brief description of the main flow diagram used by the RSVPv2-NSLP protocol for intra-domain signaling procedures. These intra-domain signaling procedures are using the NSIS protocol which consists of one NTLP level and two NSLP hierarchical levels (see Figure 2).

Intra-domain signaling is where the RSVPv2-NSLP signaling messages are originated, processed and terminated within the same domain. RSVPv2-NSLP is considered in this section to be optimized for unicast and sender-initiated protocol.

The Intra-domain signaling procedures are mainly using RSVPv2-NSLP PHR/PDR objects, (see Section 5.2.2) that are originated, processed and terminated within the same domain.

7.1. Normal operation for uni-directional reservation

This section presents examples of RSVPv2 intra-signaling procedures for RSVPv2-NSLP normal operation, i.e., operation without failures.

Figure 10 shows the main flow diagram intra-domain signaling used by the RSVPv2-NSLP protocol in case of a successful reservation. In this situation only the uni-directional feature is considered. Note that the same figure shows how the RSVPv2-NSLP inter-domain and intra-domain signaling can inter-operate. When an NslpPathInit arrives at the ingress node of a domain, i.e., NF(ingress) (see Figure 10), the RSVPv2-NSLP "e2e service" functionality creates a RSVPv2-NSLP Path reservation state. Subsequently, the RSVPv2-NSLP "PDR" protocol functionality is activated (see Figure 2) classifying the flow (i.e., Flow_Specification_Class) that is associated with the NslpPathInit message into an appropriate traffic class, e.g., Diffserv class. The RSVPv2-NSLP PDR functionality uses the RSVPv2-NSLP path state created by the NslpPathInit message and it introduces additional information that can be used to associate the PHR and PDR objects with the flow that created the RSVPv2-NSLP Path reservation state in the NF(ingress) node. The RSVPv2-NSLP PDR functionality is subsequently using the Service_Class (e.g., <SENDER_Tspec> object) and translates the requested bandwidth parameter into a number of resource units. If the QoS request is satisfied locally, then the ingress node will generate a reservation request PHR object denoted as "PHR_Resource_Request" and a reservation request PDR object denoted as "PDR_Reservation_Request", (see Section 5.2.2). The PDR object MAY contain information such as the IP address of the NF(ingress) node and the per-flow specification ID. These PHR and PDR objects are locally defined objects which are included into the Service_Class object class carried by the NslpPathInit message. The NslpPathInit message is encapsulated into a NTLP PATH message. The NTLP PATH message is processed by all NTLP stateless NF(interior) nodes that is passing through, up to the NF (egress). Each stateless NF(interior) node that processes the NTLP PATH message it will not create a state but it will activate the RSVPv2-NSLP functionality by using the transported NslpPathInit message. The RSVPv2-NSLP "PHR" functionality of these NF(interior) nodes will use the information included in the PHR object ("PHR_Resource_Request") and it will identify the ID of the traffic class, e.g., Diffserv class. If there is enough bandwidth capacity, it will reserve the requested resources. The NF(interior) node reserves the requested resources by e.g., adding the requested amount to the total amount of reserved resources for that traffic class, e.g., Diffserv class.

The behavior of the NF(egress) node on admission or rejection of the NslpPathInit message that contains the "PHR_Resource_Request" object is the same as in the NF(interior) nodes. After processing the

"PHR_Resource_Request" object, the RSVPv2-NSLP functionality of the

NF(egress) node uses the "PDR_Reservation_Request" object and creates/identifies the flow specification ID and the state associated with it. Subsequently, the RSVPv2-NSLP "e2e service" functionality is activated and by using the information contained in the Flow_Specification_Class it will create an RSVPv2-NSLP Path reservation. The NTLP PATH message is forwarded towards the NR (receiver), and it will be processed by all NTLP stateful nodes that is passing through as an inter-domain signaling procedure, see Section 6. Note that this NTLP PATH message will not include the "PDR/PHR" object information.

When the NR(receiver) receives the NTLP PATH message, similar to the procedure used in Section 6, it will create a NTLP state and it will activate the RSVPv2-NSLP "e2e service" functionality by using the NslpPathInit message. It will create a RSVPv2-NSLP path reservation state which will be identified by using the information contained in the Flow_Specification_Class object class.

Subsequently the RSVPv2-NSLP "e2e service" functionality will create an NslpResvInit message that will be used to report information related to how the NslpPathInit has been processed along the path. This NslpResvInit will be encapsulated into a RESV message and it will only be processed by the RSVPv2-NSLP "e2e service" functionality at each hop that is passing by and that is supporting the "e2e service" functionality. Note that this message is processed in a domain only by the NF(egress) and NF(ingress) nodes. The NF(interior) nodes are not processing this message. Moreover, the RSVPv2-NSLP "PDR" functionality of the NF(egress) node will report the successful reservation to the RSVPv2-NSLP "PDR" functionality of the NF(ingress) node by using a "PDR_Reservation_Report" PDR object. This object will be included into the Service_Class object class of the NslpResvInit message.

After the successful reception of the NslpResvInit message, the NI(sender) can start transmitting traffic user data. Figure 10 also shows how the refresh procedure is performed.

The inter-domain RSVPv2-NSLP refresh procedure is a pure NTLP refresh procedure, see Section 6. However, the intra-domain RSVPv2-NSLP refresh procedure is a combination of a NTLP and a RSVPv2-NSLP "PHR/PDR" procedure. When a refresh NTLP PATH message is received by a NF(ingress) node the NTLP functionality will activate the RSVPv2-NSLP "PHR/PDR" functionality that is carried by the NslpPathRef message. By using the Flow_Specification_Class object class the "PDR" functionality can identify the RSVPv2-NSLP path

state. The RSVPv2-NSLP "PDR" functionality of the NF(ingress) node will generate a refresh request PHR object denoted as "PHR_Refresh_Update" and a refresh request PDR object denoted as "PDR_Refresh_Request", (see Section 5.2.2). The PDR object MAY contain information such as the IP address of the NF(ingress) node and the per-flow specification ID. These PHR and PDR objects are locally defined objects which are included into the Service_Class object class carried by the NslpPathRef message.

The refresh NTLP PATH message is processed by all NTLP stateless NF(interior) nodes that is passing through, up to the NF (egress). Each node that processes the refresh NTLP PATH message it will refresh the NTLP state associated with the session ID, i.e., information included into the Session_ID_Class object class. The NTLP level functionality of the NTLP stateless NF(interior) nodes receiving the refresh NTLP PATH message will activate the RSVPv2-NSLP "PHR" functionality.

The RSVPv2-NSLP "PHR" functionality of these NF(interior) nodes will use the information included in the PHR object ("PHR_Refresh_Request") and it will identify the ID of the traffic class, e.g., Diffserv class. This object will refresh the requested resources included in the "PHR_Refresh_Update" object.

The NF(egress) node that processes the refresh NTLP PATH message it will refresh the NTLP state associated with the session ID included into the Session_ID_Class object class. Furthermore, it will activate the RSVPv2-NSLP "PHR" functionality by using the transported RSVPv2-NSLP "PHR" object.

The behavior of the RSVPv2-NSLP "PHR" functionality in the NF(egress) node is similar to the RSVPv2-NSLP "PHR" functionality provided in the NF(interior) nodes.

Subsequently, the refresh NTLP PATH message is forwarded towards the NR (receiver), and it will be processed by all NTLP stateful nodes that is passing through as an inter-domain signaling procedure, see Section 6. Note that this refresh NTLP PATH message will not include the "PDR/PHR" objects.

When the NF(responder) receives the refresh NTLP PATH message, it will refresh the NTLP state and it will invoke the RSVPv2-NSLP "e2e service" functionality.

If a NTLP state in a NTLP stateful is not refreshed on time then the

NTLP functionality at this node informs the RSVPv2-NSLP state that the refresh procedure is unsuccessful.

Note that the refresh NTLP PATH message may optionally carry a NSLPPathRef message. NR (receiver) will create a NTLP RESV message that will be sent towards the NI (sender). This message will be used to report information related to how the NTLP PATH message has been processed along the path. Note that the refresh NTLP RESV message may optionally carry a NSLPResvRef message. Note that this message is processed in a domain only by the NF(egress) and NF(ingress) nodes. The NF(interior) nodes are not processing this message. Moreover, the RSVPv2-NSLP "PDR" functionality of the NF(egress) node will report the successful refresh PDR/PHR procedure to the RSVPv2-NSLP "PDR" functionality of the NF(ingress) node by using a "PDR_Refresh_Report" PDR object. This object will be included into the Service_Class object class of the NslpResvRef message.

```
NF (ingress)     NF (interior)          NF (interior)     NF (egress)
NTLP stateful    NTLP stateless         NTLP stateless    NTLP stateful
    |                    |                   |                    |
PATH(NslpPathInit)       |                   |                    |
--->|PATH(NslpPathInit): |                   |                    |
    |PHR_Resource_Request|                   |                    |
    |PDR_ResReq          |PATH(NslpPathInit): |                    |
    |------------------->|PHR_Resource_Request|                    |
    |                    |PDR_ResReq          |PATH(NslpPathInit): |
    |                    |------------------->|PHR_Resource_Request|
    |                    |                    |PDR_ResReq          |
    |                    |                    |------------------->|
    |                    |                    |         PATH(NslpPathInit)
    |                    |                    |                    |----->
    |                    |                    |         RESV(NslpResvInit)
    |                    |                    |                    |<------
    |                    |RESV (NslpResvInit) |                    |
    |                    |PDR_Reservation_Report                   |
    |<------------------------------------------------------------|
RESV(NslpResvInit)       |                    |                    |
<---|                    |                    |                    |
    |                    | Traffic(user) Data |                    |
--->|------------------->|------------------->|------------------->|--->
    |                    |                    |                    |
PATH([NslpPathRef])      |                    |                    |
--->|                    |                    |                    |
    PATH(NslpPathRef):   |                    |                    |
    |PHR_Refresh_Update  |                    |                    |
    |PDR_RefReq          |PATH(NslpPathRef):  |                    |
    |------------------->|PHR_Refresh_Update  |                    |
    |                    |PDR_RefReq          |PATH(NslpPathRef):
    |                    |------------------->|PHR_Refresh_Update  |
    |                    |                    |PDR_RefReq          |
    |                    |                    |------------------->|
    |                    |                    |         PATH([NslpPathRef])
    |                    |                    |                    |----->
    |                    |                    |         RESV([NslpResvRef])
    |                    |                    |                    |<------
    |                    |RESV(NslpResvRef):  |                    |
    |                    |PDR_Refresh_Report  |                    |
    |<------------------------------------------------------------|
RESV ([NslpResvRef])     |                    |                    |
<---|                    |                    |                    |
```

(PDR_ResReq) - represents the "PDR_Reservation_Request" PDR object.

This PDR object is processed only by the

NF(ingress) and NF(egress) nodes.

(PDR_RefReq) - represents the PDR_Refresh_Request PDR object
This PDR object is processed only by the NF(ingress) and NF(egress) nodes.

Figure 10: Intra-domain signaling normal operation for successful reservation

Figure 11 depicts the intra-domain RSVPv2-NSLP tearing down procedure. A NTLP PATHTEAR message, is received by the NTLP functionality in the NF(ingress) node. Note that the NTLP PATHTEAR message may optionally carry a NSLPPathTear message. The NTLP functionality activates the RSVPv2-NSLP "PDR/PHR" functionality, that is related to the Session_ID_Class class object, and that is using the "PDR" object. The RSVPv2-NSLP "PDR" functionality of the NF(ingress) node will generate a release request "PHR" object denoted as "PHR_Release_Request" and a release request PDR object denoted as "PDR_Release_Request", (see Section 5.2.2). The PDR object may contain information such as the IP address of the NF(ingress) node and the per-flow specification ID.

These PHR and PDR objects are locally defined objects which are included into the Service_Class object class carried by a NslpPathTear message. All the RSVPv2-NSLP and NTLP reservations, in the NF(ingress) node that are associated to the Session_ID_Class object class will be released. The NTLP PATHTEAR message is processed by all NTLP stateless NF(interior) nodes that is passing through, up to the NF (egress). Each node that processes the PATHTEAR message will activate the RSVPv2-NSLP "PHR" functionality by using the transported RSVPv2-NSLP "PHR" object. The NTLP functionality in the NTLP stateless NF(interior) node that receives the PATHTEAR message will pass the NslpPathTear message to the RSVPv2-NSLP functionality. The NslpPathTear message contains the "PHR_Release_Request" and "PDR_Release_Request" PHR and PDR objects, respectively.

The RSVPv2-NSLP "PHR" functionality of this NF(interior) node will use the information included in the PHR object ("PHR_Release_Request") and it will identify the ID of the traffic class, e.g., Diffserv class. This object will subtract the requested resources included in the "PHR_Release_Request" object from the total reserved amount of resources stored in the traffic class state.

The NF(egress) node that processes the PATHTEAR message it will it

will activate the RSVPv2-NSLP "PHR" functionality by using the transported "PHR" object.

The behavior of the RSVPv2-NSLP "PHR" functionality in the NF(egress) node is similar to the RSVPv2-NSLP "PHR" functionality provided in the NF(interior) nodes. Furthermore, the NTLP state is released and the PATHTEAR message is forwarded towards the NR (receiver), and it will be processed by all NTLP stateful nodes that is passing through as an inter-domain signaling procedure, see Section 6. Note that this PATHTEAR message will not include the "PDR/PHR" objects.

```
NF (ingress)     NF (interior)          NF (interior)     NF (egress)
  NTLP stateful  NTLP stateless         NTLP stateless    NTLP stateful
   |                    |                    |                    |
   |                    | Traffic(user) Data |                    |
-->|------------------->|------------------->|------------------->|--->
   |                    |                    |                    |
PATHTEAR([NslpPathTear])|                    |                    |
-->|                    |                    |                    |
   PATHTEAR(NslpPathTear):                   |                    |
   |PHR_Release_Request |                    |                    |
   |PDR_RelReq          |PATHTEAR(NslpPathTear):                  |
   |------------------->|PHR_Release_Request                      |
   |                    |PDR_RelReq   PATHTEAR(NslpPathTear):
   |                    |------------------->|PHR_Release_Request |
   |                    |                    |PDR_RelReq          |
   |                    |                    |------------------->|
   |                    |                    |     PATHTEAR([NslpPathTear])
   |                    |                    |                    |----->
```

(PDR_RelReq) - represents the PDR_Release_Request object. This object is processed only by the NF(ingress) and NF(egress) nodes.

Figure 11: Intra-domain signaling normal operation for explicit release

Figure 12 shows the main intra-domain flow diagram used by the RSVPv2-NSLP protocol in case of an unsuccessful reservation. In this situation only the uni-directional feature is considered. In this situation the RSVPv2-NSLP and NTLP messages are created and transmitted in the same way as during the successful reservation. The main difference is related to the fact that one of the NTLP stateless NF(interior) is not able to satisfy the request carried by the "PHR_Resource_Request" PHR object. The RSVPv2-NSLP "PHR" functionality of this NF(interior) node will mark the "M" field of

the "PHR_Resource_Request" object. The RSVPv2-NSLP "PHR" functionality will also include the number of previous NF(interior) nodes that successfully processed the RSVPv2-NSLP "PHR_Resource_Request" PHR object (see Appendix). This number can, for example, be identified by the TTL (Time-To-Live) value included in the IP header of the received NTLP PATH message. Note that each time that an IP packet passes a node, its TTL value is decreased by one. Furthermore, note that the NF(ingress) node should temporarily store the TTL value included in the IP header of any message in the PDR state associated to the NTLP PATH message. In case of an unsuccessful reservation, this information, that we denote as PDR_TTT_I will be used in combination with the value included in the PDR_TTL field (see Appendix) of a receiving "PDR" reporting object. The PDR_TTL field is generated and sent by a NF(interior) node that could not successfully process the "PHR" object, e.g., admit the requested "PHR" reservation. The NF(Ingress) node using this information can calculate how many NF(interior) nodes, before the NF(interior) node, rejected the "PHR_Resource_Request" object.

In particular, the NF(interior) node that is not admitting the reservation request initiated by the "PHR_Resource_Request" PHR object will copy the TTL value included in the IP header of the received NTLP PATH message that carries the "PHR_Resource_Request" object into the "PDR_TTL" field of "PDR_Reservation_Request" PDR object. Moreover, the "T" field of the "PHR" object (see Appendix) is set to "1". These "PHR" and "PDR" objects are included in the NslpPathInit message. The NslpPathInit message is encapsulated into a NTLP PATH message.

This NslpPathInit message is sent towards the NF(egress) node, which will be transported by a NTLP PATH message. Any NF(interior) node receiving a PATH message will activate the RSVPv2-NSLP "PHR" functionality. If the "PHR_Resource_Request" PHR object is "M" marked, then the RSVPv2-NSLP "PHR" functionality will not further process the "PHR" object. The NF(egress) node receiving a NTLP PATH message will activate the RSVPv2-NSLP "PHR" functionality. If the "PHR_Resource_Request" PHR object is "M" marked, then the RSVPv2-NSLP "PHR" functionality will activate the RSVPv2-NSLP "PDR" functionality which will create and "M" mark the "PDR_Reservation_Report" object. Moreover, if the "T" field value included in the "PHR" object is "1" then the PDR_TTL value that was included by the NF(interior) node into the "PDR_Reservation_Request" object will be copied into the PDR_TTL value of the "PDR_Reservation_Report" object. The "PDR" object will be included in an NslpPathErr message. The NslpPathErr

message will be encapsulated into a NTLP PATHERROR message. The

NslpPathError message will only be processed by the NF(ingress) node. The NTLP functionality of the NF(ingress) node that receives the PATHERROR message will activate the RSVPv2-NSLP "PDR" functionality. Due to the "M" marked "PDR_Reservation_Report" object the "PDR" functionality will activate the RSVPv2-NSLP "e2e service". The RSVPv2-NSLP "e2e service" functionality of the NF(ingress) node will generate a NslpPathErr message that will be sent hop-by-hop to the NI(sender) and will be encapsulated into a NTLP PATHERROR message. This message will inform the NI(sender) that the reservation request was not successful.

Simultaneously, the NF(ingress) node will start a partial explicit release procedure, for releasing the unnecessarily reserved RSVP-NSLP resources in some interior nodes for the rejected flow. In this case, the RSVP-NSLP "PDR" functionality of the NF(ingress) node will generate a "PHR_Release_Request" object, and it will include the amount of the requested resources specified the PDR state. Moreover, the RSVPv2-NSLP "PDR" functionality will create the "PDR_Reservation_Request" PDR object.

The RSVPv2-NSLP "PDR" functionality of the NF(ingress) node can calculate the number of NF(interior) nodes that processed and reserved RSVPv2-NSLP resources.

This number can be calculated by subtracting the value included in the PDR_TTL field that was included in the received "PDR_Reservation_Report" PDR object from the value included in the PDR_TTL_I variable that has been stored into the RSVPv2-NSLP state. This calculated value will be included in the TTL - IP header field of the NTLP PATHTEAR message which is generated by the NF(ingress) node and which transports the "PHR_Resource_Release" object. The "PHR_Release_Request" and "PDR_Release_Request" objects are included into a NslpPathTear message. The NslpPathTear message is transported by a NTLP PATHTEAR message.

The RSVPv2-NSLP "PHR" functionality of any node that receives a "PHR_Resource_Release" object must identify the traffic class, e.g., DSCP and release the requested resources associated with it. This can be achieved by e.g., subtracting the amount of requested resources, included in the "PHR_Release_Request" object, from the total reserved amount of resources stored in the traffic class state. Moreover, its TTL value of the NTLP PATHTEAR message is decremented by one. When this value becomes zero, the "PHR_Resource_Release" object reached the interior node that marked the "PHR_Resource_Request" object and it will be dropped. This means that

this PHR object will not release any resources in this node.

```
NF (ingress)     NF (interior)         NF (interior)       NF (egress)
NTLP stateful    NTLP stateless        NTLP stateless    NTLP stateful
    |                    |                   |                    |
PATH(NslpPathInit)       |                   |                    |
--->|PATH(NslpPathInit): |                   |                    |
    |PHR_Resource_Request|                   |                    |
    |PDR_ResReq          |PATH(NslpPathInit): |                    |
    |------------------->|PHR_Resource_Request|                    |
    |                    |PDR_ResReq          |PATH(NslpPathInit): |
    |                    |------------------->M PHR_Resource_Request (M
    |                    |                    M            marked)
    |                    |                    M PDR_ResReq         |
    |                    |                    M------------------->|
    |                    |PATHERROR(NslpPathErr):                  |
    |                    |PDR_Reservation_Report                   |
    |<-----------------------------------------------------------|
PATHERROR(NslpPathErr)   |                   |                    |
<---|                    |                   |                    |
    |PATHTEAR(NslpPathTear):                 |                    |
    |PHR_Resource_Release|                   |                    |
    |PDR_RelReq          |                   |                    |
    |------------------->|                   |                    |
```

(PDR_ResReq*) - represents the "PDR_Reservation_Request" PDR object. This PDR object is processed only by the NF(ingress) and NF(egress) nodes.

(PDR_RelReq*) - represents the PDR_Release_Request object. This object is processed only by the NF(ingress) and NF(egress) nodes.

Figure 12: Intra-domain signaling normal operation for unsuccessful reservation

Figure 11 and Figure 12 show the intra-domain main flow diagram used by the RSVPv2-NSLP protocol for a modification of a reservation procedure. In this situation only the uni-directional feature is considered.

The request for modification of the reservation is included into the NslpPathMod message. A NTLP PATH message that encapsulates the NslpPathMod message is received by the NTLP functionality of the NF(ingress) node. The NTLP functionality activates the RSVP-NSLP "PDR/PHR" functionality, which is associated with the

Session_ID_Class object class. When the modification request requires an increase on the number of reserved resources stored in the RSVPv2-NSLP state (see Figure 13), then the RSVPv2-NSLP "PHR" functionality of the NF(ingress) node will have to subtract the old and already reserved number of resources from the number of resources included in the new modification request. The result of this subtraction should be introduced within a "PHR_Resource_Request" PHR object as the requested resources value. Furthermore, the number of resources that were reserved for a certain flow in the RSVPv2-NSLP state should also be replaced with the number of resources included in the modification request.

The RSVPv2-NSLP "PDR" functionality will create a "PDR_Modification_Request" PDR object. These two objects will be included into the Service_Class of the NslpPathMod message.

The RSVPv2-NSLP "PHR" functionality of each stateless NF(interior) node processes the PHR object as a "PHR_Resource_Request" object. Each stateless NF(interior) node that receives the modification NTLP PATH message will activate the RSVPv2-NSLP "PHR" functionality. If a RSVPv2-NSLP "PHR" functionality of any node is not able to reserve the number of requested resources, then the "PHR_Resource_Request" PHR object will be marked. In this situation the RSVPv2-NSLP PHR and PDR protocol functionality associated with an unsuccessful reservation procedure will be applied for this case (see Figure 12).

The behavior of the NF(egress) node related to the modification procedure is the same as in the NF(interior) nodes. After processing the "PHR_Resource_Request" object, the RSVPv2-NSLP functionality of the NF(egress) node uses the "PDR_Modification_Request" object and identifies the flow specification ID and the RSVPv2-NSLP state associated with it. Subsequently, the RSVPv2-NSLP "e2e service" functionality is activated and by using the information contained in the Flow_Specification_Class it will modify the reservation stored into the RSVPv2-NSLP path state.

The NTLP PATH message is forwarded towards the NR (receiver), and it will be processed by all NTLP stateful nodes that is passing through as an inter-domain signaling procedure, see Section 6. Note that this NTLP PATH message will not include the "PDR/PHR" object information.

When the NR(receiver) receives the NTLP PATH message, similar to the procedure used in Section 6, it will create a it will activate the

RSVPv2-NSLP "e2e service" functionality by using the NslpPathMod message. By using the information contained in the Flow_Specification_Class it will modify the reservation stored into the RSVPv2-NSLP path state.

Subsequently the RSVPv2-NSLP "e2e service" functionality will create an NslpResvMod message that will be used to report information related to how the NslpPathMod has been processed along the path. This NslpResvMod will be encapsulated into a RESV message and it will only be processed by the RSVPv2-NSLP "e2e service" functionality at each hop that is passing by and that is supporting the "e2e service" functionality. Note that this message is processed in a domain only by the NF(egress) and NF(ingress) nodes. The NF(interior) nodes are not processing this message. Moreover, the RSVPv2-NSLP "PDR" functionality of the NF(egress) node will report the successful reservation to the RSVPv2-NSLP "PDR" functionality of the NF(ingress) node by using a "PDR_Modification_Report" PDR object. This object will be included into the Service_Class object class of the NslpResvMod message.

```
NF (ingress)    NF (interior)          NF (interior)        NF (egress)
NTLP stateful   NTLP stateless         NTLP stateless       NTLP stateful
    |                   |                   |                   |
PATH(NslpPathMod)       |                   |                   |
--->|PATH(NslpPathMod):  |                   |                   |
    |PHR_Resource_Request|                   |                   |
    |PDR_ModReq          |PATH(NslpPathMod):  |                   |
    |------------------->|PHR_Resource_Request|                   |
    |                    |PDR_ModReq          |PATH(NslpPathMod): |
    |                    |------------------->|PHR_Resource_Request|
    |                    |                    |PDR_ModReq          |
    |                    |                    |------------------->|
    |                    |                    |         PATH(NslpPathMod)
    |                    |                    |                   |----->
    |                    |                    |         RESV(NslpResvMod)
    |                    |                    |                   |<------
    |                    |RESV (NslpResvMod)  |                   |
    |                    |PDR_Modification_Report                 |
    |<------------------------------------------------------------|
RESV(NslpResvMod)       |                   |                   |
<---|                   |                   |                   |
```

(PDR_ModReq*) - represents the "PDR_Modification_Request" PDR object. This PDR object is processed only by the NF(ingress) and NF(egress) nodes.

Figure 13: Intra-domain signaling normal operation for modification of reservation when new number of resources is higher than number of already reserved resources

When the modification request requires a decrease on the number of reserved resources stored in the RSVPv2-NSLP path state (see Figure 14), then the RSVPv2-NSLP "PHR" functionality of the NF(ingress) node will have to subtract the number of resources included in the new modification request from the old and already reserved number of resources. The result of this subtraction should be introduced in an RSVPv2-NSLP "PHR_Release_Request" PHR object. Furthermore, the number of resources that were reserved in the RSVPv2-NSLP path state for a certain flow should also be replaced with the number of resources included in the modification request. The RSVPv2-NSLP "PDR" functionality will create a "PDR_Modification_Request" PDR object. These two objects will be encapsulated into a modification

NTLP PATH message. This message will be sent towards the NF(egress) node. The RSVPv2-NSLP "PHR" functionality of each NF node processes the PHR object as a "PHR_Resource_Release" object. Each NF(interior) node that receives the modification NTLP PATH message will activate the RSVPv2-NSLP "PHR" functionality. The RSVPv2-NSLP "PHR" functionality of these NF(interior) nodes will use the information included in the PHR object ("PHR_Release_Request") and it will identify the ID of the traffic class, e.g., Diffserv class. This object will subtract the requested resources included in the "PHR_Release_Request" object from the total reserved amount of resources stored in the traffic class state.

The behavior of the NF(egress) node related to the modification procedure is the same as in the NF(interior) nodes. After processing the "PHR_Resource_Request" object, the RSVPv2-NSLP functionality of the NF(egress) node uses the "PDR_Modification_Request" object and identifies the flow specification ID and the RSVPv2-NSLP state associated with it. Subsequently, the RSVPv2-NSLP "e2e service" functionality is activated and by using the information contained in the Flow_Specification_Class, it will modify the reservation stored into the RSVPv2-NSLP path state.

The rest part of the modification procedure depicted in Figure 14 is identical to the one described earlier and depicted in Figure 13.

```
NF (ingress)    NF (interior)          NF (interior)        NF (egress)
NTLP stateful   NTLP stateless         NTLP stateless       NTLP stateful
    |                    |                   |                   |
PATH(NslpPathMod)        |                   |                   |
--->|PATH(NslpPathMod):  |                   |                   |
    |PHR_Resource_Release|                   |                   |
    |PDR_ModReq          |PATH(NslpPathMod):  |                   |
    |------------------->|PHR_Resource_Release|                   |
    |                    |PDR_ModReq          |PATH(NslpPathMod):  |
    |                    |------------------->|PHR_Resource_Release|
    |                    |                    |PDR_ModReq          |
    |                    |                    |------------------->|
    |                    |                    |         PATH(NslpPathMod)
    |                    |                    |                   |----->
    |                    |                    |         RESV(NslpResvMod)
    |                    |                    |                   |<------
    |                    |RESV (NslpResvMod)  |                   |
    |                    |PDR_Modification_Report                 |
    |<------------------------------------------------------------|
RESV(NslpResvMod)        |                   |                   |
<---|                    |                   |                   |
```

(PDR_ModReq*) - represents the PDR_Modification_Request object. This object is processed only by the NF(ingress) and NF(egress) nodes.

Figure 14: Modification of reserved resources when new number of resources is lower than number of already reserved resources

7.2. Example of Fault Handling Operation

Fault Handling Operation refers to the situations when there are errors in the network, such as loss of NTLP messages route change, link failure, etc. Two typical situations will be described: the loss of the PHR signalling messages and severe congestion. The fault handling operation described here is in general independent from the type of the example scenarios, thus it can be applied in both cases.

7.2.1. Loss of NTLP signalling messages

The NTLP signaling messages and subsequently the "PHR" and "PDR" objects might be dropped, for example due to route or link failure. The loss of the "PHR" objects is especially problematic for the reservation-based "PHR" concept, see e.g., [RODA], since the dropped signalling messages might have reserved resources in some NF(interior) nodes in the communication path that will now not be used. This does not present a problem for the measurement-based "PHR" concept, see e.g., [RIMA], since there are no reservation states. The "PHR" objects that need to be sent reliable are:

PHR_Resource_Request
PHR_Refresh_Update

The reliable delivery of the "PHR_Resource_Request" object is provided by using the functionality provided by the RSVPv2-NSLP "PDR" functionality located in the NF(ingress) node. The RSVPv2-NSLP "PDR" functionality of the NF(ingress) node sends the "PHR_Resource_Request" object towards the NF(egress) node and it starts a timer. If the reply, e.g., "PDR_Reservation_Report" object, does not arrive in a predefined time it assumes that the "PHR_Resource_Request" object is lost. The reliable deliver of the "PHR_Refresh_Update" object is provided in a similar way. A timer at the NF(ingress) node is started when the "PHR_Refresh_Update" is sent towards the NF(egress) node. If the reply, e.g., "PDR_Refresh_Report" object, does not arrive in a predefined time it assumes that the "PHR_Refresh_Update" object is lost.

7.2.2. Severe Congestion Handling operation

Severe congestion can be detected by any NF(interior) node by using different methods. Moreover, the severe congestion situation can be notified by any NF(interior) node to NF(egress) nodes by using three approaches, i.e., "Greedy Marking", "Proportional Marking" and "PHR message marking". The "PHR message marking" can only be applied on the reservation-based "PHR" concept, while the other two methods can be applied on both "PHR" concept types.

In this section the "Proportional Marking" severe congestion notification methods is used.

7.2.2.1. Proportional marking

Using this severe congestion notification method, after detecting the severe congestion situation, the NF(interior) node will notify the NF(egress) node by using remarking of user data packets that pass through the node (see Figure 15). Proportionally to the detected overload the NF(interior) node will remark a number of user data packets which are passing through a severe congested interior node and are associated with a certain traffic class, e.g., DSCP, into a domain specific DSCP.

When the marked packets arrive at the NF(egress) node, the NF(egress) node will generate a "PDR_Congestion_Report" object and send it to the NF(ingress) node containing the over-allocation volume of the flow in question, e.g., a blocking probability. The "PDR_Congestion_Report" PDR object should be included into a NslpPathErr and transported by a NTLP PATHERROR message. For each flow ID, the RSVPv2-NSLP PDR functionality at the NF(egress) node will count the number of marked bytes (# marked bytes) and the number of unmarked bytes (#unmarked bytes).

Based on this information the RSVPv2-NSLP PDR functionality at the NF(egress) node will have to calculate the blocking estimation of data. The NF(egress) node will actually calculate the blocking probability (Pdrop), which will be used by an NF(ingress) node to block this particular flow.

The blocking probability is calculated as the ratio between the dropped bytes and the maximum number of bytes that can be supported by the interior node:

Pdrop = (# marked bytes)/(# marked bytes + # unmarked bytes)

This blocking probability will be included in the "PDR_Congestion_Report" object that will be sent to the NF(ingress).

The RSVPv2-NSLP PDR functionality of the NF(ingress) node after receiving the PDR_Congestion_Report object and based on the Pdrop blocking probability, and depending on the used policy, might terminate the flow, i.e., for a higher blocking probability there is a higher chance that the flow is terminated.

If a flow needs to be terminated, then for this flow, the ingress node will generate a "PHR_Release_Request" object that will be included into the Service_Class of the NslpPathTear message. This

message will be transported by a NTLP PATHTEAR message towards the NF(egress). Furthermore, the RSVPv2-NSLP "e2e service" functionality in the NF(ingress) node will create a NslpPathErr that will be encapsulated into a PathError that will be sent towards the NI(sender) to notify that an error occured.

```
NF (ingress)   NF (interior)          NF (interior)      NF (egress)
NTLP stateful  NTLP stateless        NTLP stateless     NTLP stateful
(sent)
 user|(sent) user data   |                    |                    |
 data|                   |                    |                    |
---->|------------------>| (sent) user data   |  user data         |
     |                   |------------------->S(# marked bytes)    |
     |                   |                    S------------------->|
     |                   |                    S(# unmarked bytes)  |
     |                   |                    S------------------->|
     |                   |PATHERROR(NslpPathErr):                  |
     |                 PDR_Congestion_Report ("S" marked + Pdrop) |
     |<------------------|--------------------|--------------------|
Terminate                |                    |                    |
flow?|                   |                    |                    |
Yes PATHTEAR(NslpPathTear):                   |                    |
     |PHR_Release_Request|                    |                    |
     |PDR_RelReq         |PATHTEAR(NslpPathTear):                  |
     |------------------>|PHR_Release_Request                      |
     |                   |PDR_RelReq          | PATHTEAR(NslpPathTear):
     |                   |------------------->|PHR_Release_Request |
     |                   |                    |PDR_RelReq          |
     |                   |                    |------------------->|
     |                   |                    |                    |
     |                   |                    |                    |
PathErr(NslpPathErr)     |                    |                    |
<----|                   |                    |                    |
```

Figure 15: Intra-domain Severe Congestion handling Operation:
with proportional marking

(PDR_RelReq*) - represents the PDR_Release_Request object. This object is processed only by the NF(ingress) and NF(egress) nodes.

7.3. Example of Adaptation to load sharing operation

This procedure has to be based on the solutions provided by the NTLP protocol level on this issue. These NTLP protocol level solutions are not yet defined. Therefore, this procedure will be specified in a future version of this draft.

7.4. Normal operation for bi-directional reservation

There are situations where, for example, the inter-domain signaling has to support in addition to the unidirectional reservations also bi-directional reservations. This section gives one example of inter-domain signaling for a successful and one example of inter-domain signaling for an unsuccessful bi-directional reservation.

Figure 16 shows the flow diagram of intra-domain signaling used by the RSVPv2-NSLP protocol and the way how the RSVPv2-NSLP inter-domain and intra-domain signaling inter-operates in case of a successful bi-directional reservation.

The bi-directional successful reservation is similar to a combination of two unidirectional successful reservations that are accomplished in opposite directions. The main differences of the bi-directional successful reservation procedure with the combination of two unidirectional successful reservations accomplished in opposite directions are as follows:

* the reservation state specifies a bi-directional reservation
* the Service_Class object specifies that the reservation is bi-directional
* the NSIS aware nodes do not process the NslpResvInit, NslpResvRef and NslpResvMod messages
* the success of the reservation procedure is reported to the NI(sender) using the NslpPathInit that is sent hop-by-hop from NR(receiver) towards the NI(sender)
* the success of the refresh procedure is reported to the NI(sender) using the NslpPathRef that is sent hop-by-hop from NR(receiver) towards the NI(sender)
* the success of the refresh procedure is reported to the NI(sender) using the NslpPathMod that is sent hop-by-hop from NR(receiver) towards the NI(sender)
* the PDR_Reservation_Report object used to report a successful reservation procedure is carried by the NslpPathInit that is sent hop-by-hop from NF(egress) towards the NF(ingress)

* the PDR_Refresh_Report object used to report a successful refresh procedure is carried by the NslpPathInit that is sent hop-by-hop from NF(egress) towards the NF(ingress)
* the PDR_Modification_Report object used to report a successful modification procedure is carried by the NslpPathInit that is sent hop-by-hop from NF(egress) towards the NF(ingress)
* the NF(egress) node can initiate an explicit partial release procedure towards the NF(ingress) node.

Figure 17 and Figure 18 show the flow diagrams of intra-domain signaling used by the RSVPv2-NSLP protocol and the way how the RSVPv2-NSLP inter-domain and intra-domain signaling inter-operates in case of a unsuccessful bi-directional reservation.

The bi-directional unsuccessful reservation is similar to a combination of two unidirectional unsuccessful reservations that are accomplished in opposite directions. The main differences of the bi-directional unsuccessful procedure with the combination of two unidirectional successful reservations accomplished in opposite directions are as follows:

* the reservation state specifies a bi-directional reservation
* the Service_Class object specifies that the reservation is bi-directional
* the NSIS aware nodes do not process the NslpResvInit, NslpResvRef and NslpResvMod messages
* the PDR_Reservation_Report object used to report a successful reservation procedure is carried by the NslpPathInit that is sent hop-by-hop from NF(egress) towards the NF(ingress)
* the PDR_Refresh_Report object used to report a successful refresh procedure is carried by the NslpPathInit that is sent hop-by-hop from NF(egress) towards the NF(ingress)
* the PDR_Modification_Report object used to report a successful modification procedure is carried by the NslpPathInit that is sent hop-by-hop from NF(egress) towards the NF(ingress)
* the NF(egress) node can initiate an explicit partial release procedure towards the NF(ingress) node.

```
NF (ingress)     NF (interior)          NF (interior)     NF (egress)
NTLP stateful    NTLP stateless         NTLP stateless    NTLP stateful
    |                    |                    |                    |
PATH(NslpPathInit)       |                    |                    |
--->|                    |                    |                    |
    |PATH(NslpPathInit): |                    |                    |
    |PHR_Resource_Request|                    |                    |
    |PDR_ResReq          |      PATH(NslpPathInit):                |
    |------------------->|      PHR_Resource_Request               |
    |                    |      PDR_ResReq    |                    |
    |                    |----------------------------------------->|
    |                    |                    |           PATH(NslpPathInit)
    |                    |                    |                    |-->
    |                    |           PATH(NslpPathInit):           |
    |                    |             PDR_Reservation_Report      |
    |                    |             PHR_Resource_Request PATH(NslpPathInit)
    |        PATH(NslpPathInit):       PDR_ResReq                 |<---
    |        PDR_Reservation_Report           |                    |
    |        PHR_Resource_Request             |<-------------------|
    |        PDR_ResReq  |                    |                    |
    |<-----------------------------------------|                    |
    |                    |                    |                    |
    |                    |                    |                    |
    |                    | Traffic(user) Data |                    |
 ---|------------------->|----------------------------------------->|--->
    |                    | Traffic(user) Data |                    |
 <--|<-----------------------------------------|<-------------------|----
    |                    |                    |                    |
  PATH(NslpPathRef)      |                    |                    |
--->|                    |                    |                    |
    |PATH([NslpPathRef]):|                    |                    |
    |PHR_Refresh_Update  |                    |                    |
    |PDR_Refreq          |           PATH([NslpPathRef]):          |
    |------------------->|           PHR_Refresh_Update            |
    |                    |           PDR_Refreq                    |
    |                    |----------------------------------------->|
    |                    |                    |     PATH([NslpPathRef])
    |                    |            PATH([NslpPathRef]):         |--->
    |                    |               PDR_Refresh_Report        |
    |                    |               PHR_Refresh_Update PATH([NslpPathRef])
    |                    |               PDR_Refreq                |<---
    |           PATH([NslpPathRef]):          |<-------------------|
    |           PHR_Refresh_Report            |                    |
    |           PHR_Resource_Request          |                    |
```

```
|            PDR_ResReq                     |                     |
```

```
    |<----------------------------------------|                    |
 PATH([NslpPathRef])     |                    |                    |
 <--|                    |                    |                    |
```

(PDR_ResReq) - represents the "PDR_Reservation_Request" PDR object.
This PDR object is processed only by the
NF(ingress) and NF(egress) nodes.

(PDR_RefReq) - represents the PDR_Refresh_Request PDR object
This PDR object is processed only by the NF(ingress) and
NF(egress) nodes.

Figure 16: Intra-domain signaling normal operation for successful
bi-directional reservation

```
NF (ingress)     NF (interior)          NF (interior)     NF (egress)
NTLP stateful    NTLP stateless         NTLP stateless    NTLP stateful
    |                    |                    |                    |
PATH(NslpPathInit)       |                    |                    |
--->|                    |                    |                    |
    |PATH(NslpPathInit): |                    |                    |
    |PHR_Resource_Request|                    |                    |
    |PDR_ResReq          |      PATH(NslpPathInit):                |
    |------------------->M      PHR_Resource_Request                |
    |                    M      PDR_ResReq    |                    |
    |                    M---------------------------------------->|
    |  PATHERROR(NslpPathErr):                |                    |
    |   PDR_Reservation_Report                |                    |
    |<------------------------------------------------------------|
PATHERROR(NslpPathErr)   M                    |                    |
 <--|                    M                    |                    |
    |PATHTEAR(NslpPathTear):                  |                    |
    |PHR_Release_Request M                    |                    |
    |PDR_RelReq          M                    |                    |
    |------------------->M                    |                    |
```

(PDR_ResReq) - represents the "PDR_Reservation_Request" PDR object. This PDR object is processed only by the NF(ingress) and NF(egress) nodes.

(PDR_RefReq) - represents the PDR_Refresh_Request PDR object This PDR object is processed only by the NF(ingress) and NF(egress) nodes.

(PDR_RelReq) - represents the PDR_Release_Request PDR object This PDR object is processed only by the NF(ingress) and NF(egress) nodes.

Figure 17: Intra-domain signaling normal operation for unsuccessful bi-directional reservation (rejection on path NF(ingress) towards NF(egress))

```
NF (ingress)     NF (interior)          NF (interior)     NF (egress)
NTLP stateful    NTLP stateless         NTLP stateless    NTLP stateful
    |                    |                    |                    |
PATH(NslpPathInit)       |                    |                    |
--->|                    |                    |                    |
    |PATH(NslpPathInit): |                    |                    |
    |PHR_Resource_Request|                    |                    |
    |PDR_ResReq          |      PATH(NslpPathInit):                |
    |------------------->|      PHR_Resource_Request               |
    |                    |      PDR_ResReq    |                    |
    |                    |----------------------------------------->|
    |                    |                    |           PATH(NslpPathInit)
    |                    |                    |                    |-->
    |                    |           PATH(NslpPathInit):           |
    |                    |             PHR_Reservation_Report      |
    |        PATH(NslpPathInit):       PHR_Resource_Request PATH(NslpPathInit)
    |        PDR_Reservation_Report     PDR_ResReq                  |<---
    |        PHR_Resource_Request             M<------------------|
    |        PDR_ResReq  |                    M                    |
    |<----------------------------------------M                    |
    |  PATHERROR(NslpPathErr):                M                    |
    |   PDR_Reservation_Report                M                    |
    |------------------------------------------------------------>|
    |                    |                    M
PATHERROR(NslpPathErr)
    |                    |                    M                    |-->
    |                    |                    M
PATHTEAR(NslpPathTear):
    |                    |                    M             PHR_Release_Request
    |                    |                    M             PDR_Relreq
    |                    |                    M<------------------|
    |  PATHERROR(NslpPathErr):                M                    |
    |   PDR_Reservation_Report                M                    |
    |<------------------------------------------------------------|
PATHERROR(NslpPathErr)   |                    M                    |
 <--|                    |                    M                    |
    |PATHTEAR(NslpPathTear):                  M                    |
    |PHR_Release_Request |                    M                    |
    |PDR_RelReq          |      PATHTEAR(NslpPathTear):            |
    |------------------->|      PHR_Release_Request                |
    |                    |      PDR_RelReq    M                    |
    |                    |----------------------------------------->|
   (PDR_ResReq) - represents the "PDR_Reservation_Request" PDR object.
                     This PDR object is processed only by the
```

NF(ingress) and NF(egress) nodes.

(PDR_RefReq) - represents the PDR_Refresh_Request PDR object

This PDR object is processed only by the NF(ingress) and NF(egress) nodes.

(PDR_RelReq) - represents the PDR_Release_Request PDR object
This PDR object is processed only by the NF(ingress) and NF(egress) nodes.

Figure 18: Intra-domain signaling normal operation for unsuccessful bi-directional reservation (rejection on path NF(egress) towards NF(ingress))

8. Appendix - Examples of PHR and PDR object specifications

This appendix provides examples of how PHR and PDR objects could be specified.

8.1. PHR objects

RSVPv2 should support both IP versions, i.e., IPv4 and IPv6. The format of the PHR object that is based on both IPv4 and IPv6 versions is depicted in Figure 19. On top of the PHR specific information of the PHR object, the three (standard) RSVPv1 object fields are used, i.e., Length, Class-Num and C-type.

```
 0                                                               31
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        Length(bytes)          | Class-Num     |   C-Type       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        Unused                 |P-LEN| P-ID  |S|M|  C  |T|Unused|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Requested Resources        |   Delta T     |   Shared %     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                                 |
|                                                                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 19: PHR object format based on IPv4 or IPv6

Length
(in octets): 16-bit field containing the total object length in octets. It must always be a multiple of 4 and at least 4 octets.

Class-Num: 8-bit field identifying the object class. Each object class has a name.

C-Type: 8-bit field identifying the object type, unique within Class-Num. In this case a C-Type ID should be assigned to the PHR object.

P-LEN (PHR length) 3-bit field. This specifies the length in octets of the specific PHR information data, without including the "Variable length" field.

The value 0 specifies that this IP option field contains only data in the "Variable length" field.
This data MUST begin on the next 32-bit word boundary after the P-LEN field (after the first "unused" field). In this case, the sender MUST set the "S", "M", "C", and "unused" fields to 0. The P-ID MUST have the value 1.

If a receiver receives a packet with a P-LEN value of 0, it MUST ignore the values in the "S", "M", "C", and "unused" fields.

P-ID (PHR type) 4-bit field. This specifies the PHR type. For the RODA PHR, the value MUST be 1.

S (Severe Congestion) 1-bit field. The sender MUST set the "S" field to 0. This field is set to 1 by an NF(interior) or NF(edge) node when a severe congestion situation occurs.

M (Marked) 1-bit field. The sender MUST set the "M" field to 0. This field is set to 1 by an NF(interior) or NF(edge) node when the node cannot satisfy the "Requested Resources" value.

C (Object type) 3-bit field. This field specifies the type of the PHR object.

C	Description
0	Reserved
1	"PHR_Resource_Request"
2	"PHR_Refresh_Update"

3 "PHR_Release_Request"
4-7 Unused

"PHR_Resource_Request": initiate or update the traffic class reservation state on all nodes located on the communication path between the NF(ingress) and NF(egress) nodes according to an external SAPU Path request.

"PHR_Refresh_Update": refresh the traffic class reservation soft state on all nodes located on the communication path between the NF(ingress) and NF(egress) nodes according to a resource reservation request that was successfully processed by the RSVPv2-NSLP PHR functionality during a previous refresh period.

"PHR_Release_Request": explicitly release, by subtraction, the reserved resources for a particular flow from a traffic class reservation state.

T (TTL active)	1-bit field. The NF(ingress) node MUST set the "T" field to 0. This field MAY be set to "1" by a node when the node will have to include the TTL value from the header of the IP packet into the "PDR_TTL" field of the PDR object.
U	A 3-bit field that is currently unused. Reserved for future PHR object extensions.
Requested Resources	16-bit field. This field specifies the requested number of units of resources to be reserved by a node. The unit is not necessarily a simple bandwidth value. It may be defined in terms of any resource unit (e.g., effective bandwidth) to support statistical multiplexing at message level.
Delta T	8 bit field. The value of this field MAY be set by any NF(ingress) node into (only) "PHR_Resource_Release" objects. It specifies a percentage that represents the ratio between a time lag, say T_lag, and the length of the refresh period, say T_period. Where, T_lag represents the difference between the departure time of the

	previous sent "PHR_Refresh_Update" object and the departure time of the "PHR_Resource_Release" object. T_period represents the length of the refresh period. This information MAY be used by any node during an explicit release procedure.
Shared % (Shared percentage)	8 bit field. This value MAY be used to specify if a load sharing situation occurred on a communication path or not. The ingress node sets this value to 100. If load sharing occurred in a node then the node will divide the shared percentage value to the number of equal cost paths.
Variable length	this field is currently unused. Reserved for future PHR object extensions.

8.2. PDR objects

The format of the PDR object that is based on the IPv4 version is depicted in Figure 20. On top of the PDR specific information of the PDR object, the three (standard) RSVPv1 object fields are used, i.e., Length, Class-Num and C-type.

```
 0                                                               31
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        Length(bytes)          | Class-Num     |   C-Type      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| PDRID |MsgType|S|M|B| Unused  |F-Type |EP-Type|   PDR-TTL     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Reverse Requested Resources  |   Shared %    |Dropping rate %|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Ingress (Egress) Address (IPv4)                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|                     Flow-ID (length varies)                   |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|                  Variable length field                        |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 20: PDR object format based on IPv4

Length (in octets): 16-bit field containing the total object length in octets. It must always be a multiple of 4 and at least 4 octets.

Class-Num: 8-bit field identifying the object class. Each object class has a name.

C-Type: 8-bit field identifying the object type, unique within Class-Num. In this case a C-Type ID should be assigned to the the PHR object.

PDRID: 4-bit field identifying the ID of the PDR object. It is zero for an experimental protocol.

MsgType: 4-bit field identifying the type of PDR object. See below for a table of recognized values.

MsgType	Description	Sent with PHR object
0	reserved	
1	PDR_Reservation_Request	PHR_Resource_Request
2	PDR_Refresh_Request	PHR_Refresh_Update
3	PDR_Release_Request	PHR_Resource_Release

```
4     PDR_Reservation_Report
5     PDR_Refresh_Report
6     PDR_Release_Report
7     PDR_Request_Info         PHR_Resource_Request OR
                               PHR_Refresh_Update   OR
                               PHR_Resource_Release OR
                               PHR_Modification_Request
8     PDR_Congestion_Report
9     PDR_Modification_Request
10    PDR_Modification_Report
11-16 unused
```

"PDR_Reservation_Request": generated by the NF(ingress) node in order to initiate or update the RSVPv2-NSLP PDR state

in the NF(egress) node

"PDR_Refresh_Request": generated by the NF(ingress) node and sent to the NF(egress) node to refresh, in case needed, the RSVPv2-NSLP PDR states located in the NF(egress) node

"PDR_Modification_Request": generated and sent by the NF(ingress) node to the NF(egress) node to modify the PDR states located in the NF(egress) node

"PDR_Release_Request": generated and sent by the NF(ingress) node to the NF(egress) node to release the flows explicitly

"PDR_Request_Info": an object that can be used as a common "PDR_Reservation_Request", "PDR_Refresh_Request", "PDR_Release_Request" and "PDR_Modification_Request"

"PDR_Reservation_Report": generated and sent by the NF(egress) node to the NF(ingress) node to report that a "PHR_Resource_Request" PHR object and a "PDR_Reservation_Request" PDR object has been received and that the request has been admitted or rejected

"PDR_Refresh_Report": generated and sent by the NF(egress) node in case needed, to the NF(ingress) node to report that a "PHR_Refresh_Update" PHR object and a "PDR_Refresh_Request" PDR object have been received and have been processed

"PDR_Congestion_Report": generated and sent by the NF(egress) node to the NF(ingress) node and used for severe congestion notification. They are only used when either the "greedy marking" or "proportional marking" severe congestion notification procedures are applied.

"PDR_Modification_Report": generated and sent by the NF(egress) node to NF(ingress) node to report that the combination of either the "PHR_Resource_Request" PHR object and the "PDR_Modification_Request" PDR object or the "PHR_Release_Request" PHR object and the "PDR_Modification_Request" have been received and processed

PDRID: 4-bit field. ID of the PDR object. It is zero for an experimental protocol.

S (Severe Congestion): 1-bit field. specifies if a severe congestion situation occured. It can also carry the "S" flag of the "PHR_Resource_Request" or "PHR_Refresh_Update" PHR objects.
This flag only applies to "PDR_Reservation_Report", "PDR_Refresh_Report", "PDR_Congestion_Report" and "PDR_Modification_Report" objects.

M (Marked): 1-bit field. Carries the "M" value of the "PHR_Resource_Request" or "PHR_Refresh_Update" PHR objects.
This flag only applies to "PDR_Reservation_Report", "PDR_Refresh_Report", "PDR_Congestion_Report" and "PDR_Modification_Report" objects.

B (Bi-directional reservation): 1-bit field. specifies that the "PHR" objects should be used for bi-directional reservations in intra-domain signaling. Note that when the inter-domain signaling procedures are applied for bi-directional reservations it does not mean that the associated intra-domain signaling procedures should also use bi-directional reservations.

F-Type: (Flow Type) 4-bit field. The Flow-ID type identifier. Defined by the PDR protocol. It informs the NF(ingress) and NF(egress) nodes what kind of data is contained in the Flow-ID and its length. Every NF(edge) node should be configured to process the F-Types.

EP-Type: (External Protocol Type)	4-bit field. Identifies the used external protocol. Only useful when the intra-domain signaling procedures are used in combination with non-RSVPv2 inter-domain signaling procedures. It informs the NF(ingress) and NF(egress) nodes what type of external protocol (EP) data is contained in the Variable length field. Every edge node MUST be configured to process the EP-Type. If this field is 0000 then the Variable length field can be used for other purposes, i.e., future specifications.
PDR-TTL:	8-bit field. The TTL value introduced by a node that could not admit or process a "PHR_Resource_Request" object.
Reverse Requested Resources :	16 bits. This field only applies when the "B" flag is set to "1". It specifies the requested number of units of resources that have to be reserved by a node in the reverse direction when the intra-domain signaling procedures require a bi-directional reservation procedure. The unit is not necessarily a simple bandwidth value: it may be defined in terms of any resource unit (e.g., effective bandwidth) to support statistical multiplexing at packet level.
Shared % : (Shared percentage):	8-bit field. This value specifies if a load sharing situation occurred on a communication path or not. The NF(ingress) node sets this value to 100. If load sharing occurred in a node then the node will have to divide the shared percentage value to the number of equal cost paths.
Dropping rate %: (Dropping rate percentage) :	8-bit field: This value specifies the dropping rate percentage and is used during severe congestion. The ingress node will use this rate as a blocking probability, to terminate the particular flow.
Ingress (Egress) Address :	32-bit field. For the case that the PDR object is sent by NF(ingress) to NF(egress) this field represents the NF(ingress) IP address. In the other direction this field represents the NF(egress) IP address.
Flow-ID:	Length depends on F-Type. It specifies the flow ID used by the PDR state.
Variable :	variable length field. It can be used either for

length field including external protocol data or reserved for future PDR object extensions.

The format of the PDR object that is based on the IPv6 version is depicted in Figure 21. Note that the only difference between the PDR object format based on IPv4 and IPv6 versions is the Ingress (Egress) Address field, i.e., in IPv6 is this field 128 bits long, while in IPv4 is this field 32 bits long.

```
 0                                                               31
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        Length(bytes)          | Class-Num     |   C-Type      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| PDRID |MsgType|S|M|B| Unused  |F-Type |EP-Type|   PDR-TTL     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Reverse Requested Resources  |   Shared %    |Dropping rate %|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|                                                               |
|            Ingress (Egress) Address (IPv6)                    |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|                     Flow-ID (length varies)                   |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|                  Variable length field                        |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 21: PDR object format based on IPv6

9. References

[BrLi01] Braden, R., Lindell, B., "A Two-Level Architecture for Internet Signaling", Internet Draft (work in progress), 2001.

[Bru03] Brunner, M., "Requirements for QoS Signaling Protocols", draft-ietf-nsis-req-06.txt (work in progress), 2003

[CsTa02] Csaszar, A., Takacs, A., Szabo, R., Rexhepi, V.,

Karagiannis, G., "Severe Congestion Handling with Resource Management in Diffserv On Demand", submitted to Networking 2002, May 19-24 2002, Pisa - ITALY.

[Hanc03] Hancock, R., Freytsis, I., Karagiannis, G., Loughney, J., Van den Bosch, S., "Next Steps in Signaling: Framework", Internet Draft, 2003, Work in progress.

[RFC1633] Braden, R., Clark, D., Shenker, S., "Integrated Services in the Internet Architecture: an Overview", IETF RFC 1633, 1994.

[RFC2002] Perkins, C., Editor, "IP Mobility Support", RFC 2002, October 1996.

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

[RFC2747] F. Baker, B. Lindell, M.Talwar. "RSVP Cryptographic Authentication", IETF RFC, January 2000.

[RFC2475] Blake, S., Black, D., Carlson, M., Davies, E., Wang, Zh., Weiss, W., "An Architecture for Differentiated Services", IETF RFC 2475, 1998.

[RFC3175] Baker, F., Iturralde, C. Le Faucher, F., Davie, B., "Aggregation of RSVP for IPv4 and IPv6 Reservations", IETF RFC 3175, 2001.

[RIMA] Westberg, L., Heijenk, G., Karagiannis, G., Oosthoek, S., Partain, D., Rexhepi, V., Szabo, R., Wallentin, P., el Allali, H.,"Resource Management in Diffserv Measurement-based Admission Control PHR", Internet draft Work in progress, 2002.

[RODA] Westberg, L., Karagiannis, G., Kogel, de M., Partain, D., Oosthoek, S., Jacobsson, M., Rexhepi, V., "Resource Management in Diffserv On DemAnd (RODA) PHR", Internet Draft, Work in progress.

[RMD-frame] Westberg, L., Jacobsson, M., Karagiannis, G., Rexhepi, V., Partain, D., Oosthoek, S., Szabo, R., Wallentin, P.,

"Resource Management in Diffserv
Framework", Internet draft, Work in progress.

[NSIS-ML1] Braden, B., "Re: What's in the charter of NSIS", http://www1.ietf.org/mail-archive/working-groups/nsis/current/msg01368.html

[PAN-SSP] Pan, P., Murphy, J., "A Network Architecture for Simplified Signaling Protocol", IETF Internet Draft, draft-pan-signal-req-00.txt, Work in Progress, May 2002.

[RANISSUE] Partain, D., Karagiannis, G., Wallentin, P., Westberg, L.,"ResourceReservation Issues in Cellular Radio Access Networks", Internet Draft, draft-westberg-rmd-cellular-issues-01.txt, Work in Progress, June 2002.

[WeKa03] Westberg, L., Karagiannis, G., Rexhepi, V., "Using RSVPv1 as NTLP (NSIS Transport layer Protocol): suggestions for modifications on RFC2205", Internet Draft, Work in Progress, draft-westberg-nsis-rsvp-as-ntlp-01.txt, February 2003.

10. Authors' Addresses

Lars Westberg
Ericsson Research
Torshamnsgatan 23
SE-164 80 Stockholm
Sweden
EMail: Lars.Westberg@era.ericsson.se

Attila Bader
Traffic Lab
Ericsson Hungary Ltd.
Laborc u. 1
H-1037 Budapest
Hungary
EMail: Attila.Bader@eth.ericsson.se

Georgios Karagiannis
University of Twente
P.O. BOX 217
7500 AE Enschede
The Netherlands
EMail: karagian@cs.utwente.nl

David Partain
Ericsson Radio Systems AB
P.O. Box 1248
SE-581 12 Linkoping
Sweden
EMail: David.Partain@ericsson.com

Vlora Rexhepi
Ericsson EuroLab Netherlands B.V.
Institutenweg 25
P.O.Box 645 7500 AP Enschede
The Netherlands
EMail: Vlora.Rexhepi@eln.ericsson.se