ICS 33.180.01
M 33

中华人民共和国国家标准

GB/T 21645.9—2012

自动交换光网络(ASON)技术要求 第9部分:外部网络-网络接口(E-NNI)

Technical requirements for automatically switched optical network (ASON)—Part 9:External network to network interface(E-NNI)

2012-12-31 发布　　2013-06-01 实施

中华人民共和国国家质量监督检验检疫总局
中国国家标准化管理委员会　发布

目　　次

前　言

GB/T 21645《自动交换光网络(ASON)技术要求》分为以下几个部分：

——第1部分：体系结构与总体要求

——第2部分：术语和定义

——第3部分：数据通信网(DCN)

——第4部分：信令技术

——第5部分：用户网络接口(UNI)

——第6部分：管理平面

——第7部分：自动发现

——第8部分：路由

——第9部分：外部网络-网络接口(E-NNI)

本部分为GB/T 21645的第9部分。

本部分按照GB/T 1.1—2009给出的规则起草。

本部分主要技术内容参考了OIF E-NNI接口的信令和路由规定，包括OIF E-NNI1.0、OIF E-NNI2.0、OIF E-NNI-OSPF-01.0，此外还参考了IETF基于GMPLS的域间接口规定和协议，包括IETF RFC4726、IETF RFC4874、IETF RFC5151、IETF RFC5152、IETF RFC4974、IETF RFC5298等。

本部分各章节与上述标准的对应关系如下：

——第4、5章对应了OIF E-NNI1.0第3～9章，OIF E-NNI2.0第6～9章的内容；

——第6章对应了OIF E-NNI2.0第10章的内容；

——第7章对应了OIF E-NNI2.0第12章的内容；

——第8章对应了OIF E-NNI2.0第13章的内容；

——第9章对应了OIF E-NNI-OSPF-01.0第3章的内容；

——第10章对应了OIF E-NNI-OSPF-01.0第9章的内容；

——第11章对应了OIF E-NNI-OSPF-01.0第10章的内容；

——第12章对应了RFC4726第2～5章的内容；

——第13章对应了OIF E-NNI2.0第11章的内容；

——附录A.2.1对应了IETF RFC5151的内容；

——附录A.2.2对应了IETF RFC4874的内容；

——附录A.2.3对应了IETF RFC4974的内容；

——附录B.2对应了IETF RFC4655的内容；

——附录B.3对应了IETF RFC5441的内容；

——附录B.4对应了IETF RFC5520的内容。

本部分由中华人民共和国工业和信息化部提出。

本部分由中国通信标准化协会归口。

本部分起草单位：工业和信息化部电信研究院、中国电信集团公司、华为技术有限公司、中兴通讯股份有限公司、武汉邮电科学研究院、上海贝尔股份有限公司、中国移动通信集团公司。

本部分主要起草人：张国颖、荆瑞泉、蔡军州、徐云斌、汤瑞、柯明、朱冰、许宗幸、李晗。

自动交换光网络(ASON)技术要求 第9部分:外部网络-网络接口(E-NNI)

1 范围

GB/T 21645的本部分规定了用于运营商内ASON网络的E-NNI接口的技术要求。主要包括基于OIF的E-NNI接口规范,如:E-NNI信令功能、支持的业务、信令参考配置、基于RSVP-TE的信令协议、E-NNI路由架构、基于OSPF-TE的路由协议扩展等;以及基于IETF GMPLS的E-NNI规范,如:IETF GMPLS的域间信令架构、域间信令协议扩展、域间路由要求等。本部分不包括自动发现、策略等内容。

本部分适用于基于SDH和OTN的自动交换光网络(ASON)中E-NNI接口。

2 规范性引用文件

下列文件对于本文件的应用是必不可少的。凡是注日期的引用文件,仅注日期的版本适用于本文件。凡是不注日期的引用文件,其最新版本(包括所有的修改单)适用于本文件。

GB/T 21645.1—2008 自动交换光网络(ASON)技术要求 第1部分:体系结构与总体要求

GB/T 21645.2—2010 自动交换光网络(ASON)技术要求 第2部分:术语和定义

GB/T 21645.3—2009 自动交换光网络(ASON)技术要求 第3部分:数据通信网(DCN)

GB/T 21645.5—2012 自动交换光网络(ASON)技术要求 第5部分:用户-网络接口(UNI)

GB/T 21645.8—2012 自动交换光网络(ASON)技术要求 第8部分:路由

ITU-T G.7713—2006 分布式呼叫和连接管理[Distributed Connection Management (DCM)]

ITU-T G.7713.2 采用GMPLS RSVP-TE的DCM信令[DCM Signalling Mechanism Using GMPLS RSVP-TE (DCM GMPLS RSVP-TE)]

ITU-T G.7715.1 链路状态路由协议的要求(ASON Routing Architecture and Requirements for Link State Protocols)

ITU-T G.8080 自动交换光网络结构[Architecture of the Automatic Switched Optical Network (ASON)]

OIF UNI1.0 UNI1.0信令规范[User Network Interface (UNI) 1.0 Signaling Specification]

OIF E-NNI-Sig-01.0 运营商内E-NNI1.0信令规范(OIF Implementation Agreement OIF-E-NNI-Sig-01.0-Intra-Carrier E-NNI Signaling Specification)

OIF SEP-02.1 UNI和E-NNI的安全性扩展规范(Addendum to the Security Extension for UNI and NNI)

OIF SLG-01.0 OIF采用SysLog的控制平面日志和审查功能(OIF Implementation Agreement OIF Control Plane Logging and Auditing with Syslog)

IETF RFC2205 资源预留协议(RSVP) 版本1 功能规范(Resource ReSerVation Protocol (RSVP)-Version 1 Functional Specification)

IETF RFC2328 OSPF版本2(OSPF Version 2)

IETF RFC2961 RSVP刷新开销减少扩展(RSVP Refresh Overhead Reduction Extensions)

IETF RFC3209 RSVP-TE:RSVP支持LSP隧道的扩展(RSVP-TE: Extensions to RSVP for

LSP Tunnels)

IETF RFC3471 通用标记交换协议 信令功能描述[Generalized Multi-Protocol Label Switching (GMPLS) Signaling Functional Description]

IETF RFC3473 通用标记交换信令 RSVP-TE 扩展[Generalized Multi-Protocol Label Switching (GMPLS) Signaling Resource ReserVation Protocol-Traffic Engineering (RSVP-TE) Extensions]

IETF RFC3474 用于 ASON 的 GMPLS RSVP-TE 的使用和扩展 IANA 分配文件[Documentation of IANA assignments for Generalized MultiProtocol Label Switching (GMPLS) Resource Reservation Protocol-Traffic Engineering (RSVP-TE) Usage and Extensions for Automatically Switched Optical Network (ASON)]

IETF RFC3477 用于无编号信令 RSVP-TE 信令[Signalling Unnumbered Links in Resource ReSerVation Protocol-Traffic Engineering (RSVP-TE)]

IETF RFC3630 OSPF 流量工程扩展版本 2[Traffic Engineering (TE) Extensions to OSPF Version 2]

IETF RFC4203 支持通用标记交换协议的 OSPF 扩展[OSPF Extensions in Support of Generalized Multi-Protocol Label Switching (GMPLS)]

IETF RFC4328 用于 G.709 OTN 网络控制的 GMPLS 信令扩展[Generalized Multi-Protocol Label Switching (GMPLS) Signaling Extensions for G.709 Optical Transport Networks Control]

IETF RFC4606 用于 SONET&SDH 控制的 GMPLS 信令扩展[Generalized Multi-Protocol Label Switching (GMPLS) Extensions for Synchronous Optical Network (SONET) and Synchronous Digital Hierarchy (SDH) Control]

IETF RFC4655 PCE 架构[A Path Computation Element (PCE)-Based Architecture]

IETF RFC4726 MPLS-TE 的域间体系架构(A Framework for Inter-Domain Multiprotocol Label Switching Traffic Engineering)

IETF RFC4874 排斥路由-RSVP-TE 协议扩展[Exclude Routes-Extension to Resource ReserVation Protocol-Traffic Engineering (RSVP-TE)]

IETF RFC4974 支持呼叫的 MPLS (GMPLS) RSVP-TE 信令扩展[Generalized MPLS (GMPLS) RSVP-TE Signaling Extensions in Support of Calls]

IETF RFC5151 域间 MPLS 和 GMPLS 流量工程 RSVP-TE 扩展[Inter-Domain MPLS and GMPLS Traffic Engineering—Resource Reservation Protocol-Traffic Engineering (RSVP-TE) Extensions]

IETF RFC5298 域间 LSP 恢复分析[Analysis of Inter-Domain Label Switched Path (LSP) Recovery]

IETF RFC5441 PCE 后向递归多域路径计算[A Backward-Recursive PCE-Based Computation (BRPC) Procedure to Compute Shortest Constrained Inter-Domain Traffic Engineering Label Switched Paths]

IETF RFC5520 PCE 域间路径保密(Preserving Topology Confidentiality in Inter-Domain Path Computation Using a Path-Key-Based Mechanism)

3 术语和定义、缩略语

3.1 术语和定义

GB/T 21645.2—2010 界定的以及下列术语和定义适用于本文件。

3.1.1

域间链路　inter-domain link

一条链路的两端位于一个特定路由层次内的不同路由区。

3.1.2

域内链路　intra-domain link

一条链路的两端位于某个特定路由层次中的相同路由区内。

3.1.3

层次/等级　level

路由区和包含它的路由区或它所包含的路由区。在同一个路由层次深度的路由区被认为处于相同路由等级。

3.1.4

节点标识　node ID

标识传送拓扑图中的一个节点。节点还可以表示一个路由区或一个子网。

3.1.5

协议控制器　protocol controller

提供抽象接口参数与接口上实际承载协议消息映射的功能元件。

3.1.6

路由控制器　routing controller;RC

提供路由服务接口,负责路由信息的协调和发布的功能元件。

3.1.7

路由控制器标识　RC ID

唯一标识一个 RC 实例。该标识符可能用于数据库同步功能。

3.1.8

路由控制器协议控制器标识　RC PC ID

唯一标识一个 RC 的协议控制器。每个协议控制器获取一或多个控制平面元件的原语,并把接口复用到一个协议实例中。

3.1.9

RC PC SCN 地址　RC PC SCN address

RC 通过协议控制器所关联 IP SCN 的 SCN 地址。一个 RC 可以有多个 PC 关联到 SCN,分别支持特定的协议的过程和格式。本部分中该地址标识 RC 的 OSPF PC。

3.1.10

路由控制域　routing control domain;RCD

一个传送域是根据运营商策略建立的规则所组合起来的一组传送资源。一个 RCD 是一类传送域,是为了传送资源发布的目的所组成的一个 RC 联邦。

3.1.11

流量工程链路　TE link

一个具有流量工程属性的逻辑链路。TE 链路是逻辑的概念,因为它可以把特定物理资源信息进行组合/映射,用于基于约束的最短路径算法的路由计算。

3.2　缩略语

下列缩略语适用于本文件。

AGC:接入组容器(Access Group Container)

AS:自治系统(Autonomous System)

ASBR：自治系统边界路由器(Autonomous System Boundary Router)
ASON：自动交换光网络(Automatically Switched Optical Network)
BRPC：后向递归的 PCE 路径计算(Backward-Recursive PCE-Based Computation)
CC：连接控制器(Connection Controller)
CCC：呼叫方/被叫方呼叫控制器(Calling/Called Party Call Controller)
CD：控制域(Control Domain)
CI：特征信息(Characteristic Information)
CPS：机密路径信息(Confidential Path Segment)
CSPF：基于约束的最短路径优先(Constraint-based Shortest Path First)
DCN：数据通信网(Data Communication Network)
ECC：嵌入控制通道(Embedded Control Channels)
EMS：网元管理系统(Element Management System)
E-NNI：外部网络-网络接口(Exterior Network-Network Interface)
eNNI-D：下游 E-NNI 逻辑控制平面实体(E-NNI Downstream)
eNNI-U：上游 E-NNI 逻辑控制平面实体(E-NNI Upstream)
ERO：显式路由对象(Explicit Route Object)
FRR：快速重路由(Fast ReRoute)
GMPLS：通用多协议标签交换(Generalized MPLS)
GRE：通用路由封装(Generic Routing Encapsulation)
IETF：因特网工程任务组(Internet Engineering Task Force)
ID：标识(Identifier)
ITU-T：国际电信联盟-电信(International Telecommunications Union-Telecommunications)
I-NNI：内部网络-网络接口(Interior Network-Network Interface)
LC：链路连接(Link Connection)
LSA：链路状态通告(Link State Advertisement)
LSP：标签交换路径(Label Switched Path)
MCN：管理通信网(Management Communications Network)
MPLS：多协议标记交换(Multi-protocol Label Switching)
NCC：网络呼叫控制器(Network Call Controller)
NMS：网络管理系统(Network Management System)
NNI：网络-网络接口(Network-to-network interface)
Node ID：节点标识(Node Identifier)
NCC：网络呼叫控制器(Network Call Controller)
NE：网元(Network Element)
OIF：光因特网论坛(Optical Internet Forum)
OSPF：开放最短路径优先(Open Shortest Path First)
OTN：光传送网(Optical Transport Network)
PC：协议控制器(Protocol Controller)
PCC：路径计算请求客户端(Path Computation Client)
PCE：路径计算单元(Path Computation Element)
PCReq：路径计算请求(Path Computation Request)
PCRep：路径计算响应(Path Computation Respond)
PDU：协议数据单元(Protocol Data Unit)

RA:路由域(Routing Area)
RC:路由控制器(Routing Controller)
RCD:路由控制域(Routing Control Domain)
RRO:记录路由对象(Record Route Object)
RSVP:资源预留协议(Resource Reservation Protocol)
RSVP-TE:RSVP 流量工程扩展(RSVP Traffic Engineering)
SRLG:共享风险链路组(Shared Risk Link Group)
SC:交换连接(Switched Connection)
SC PC ID:信令控制器协议控制器标识(Signaling Controller Protocol Controller Identifier)
SCN:信令通信网(Signaling Communications Network)
SDH:同步数字体系(Synchronous Digital Hierarchy)
SONET:同步光网络(Synchronous Optical Network)
SN:子网(Subnetwork)
SNC:子网连接(Sub-Network Connection)
SNP:子网点(Sub-Network Point)
SNPP:子网点池(Sub-Network Point Pool)
SNPP Link:子网点池链路(Sub-Network Point Pool Link)
SPC:软永久连接(Soft Permanent Connection)
SPF:最短路径优先(Shortest Path First)
TE:流量工程(Traffic Engineering)
TLV:类型/长度/值(Type/Length/Value)
TNA:传送网络分配地址(Transport Network Assigned Address)
TNE:传送网络单元(Transport Network Element)
TRI:传送资源标识(Transport Resource Identifier)
TTL:生存时间(Time To Live)
UNI:用户-网络接口(User-network Interface)
UNI-C:用户-网络接口客户侧(Client Side of a UNI)
UNI-N:用户-网络接口网络侧(Network Side of a UNI)
VLAN:虚拟局域网(Virtual Local Area Network)
VSPT:虚拟最短路径树(Virtual Shortest Path Tree)

4 E-NNI 信令功能定义

4.1 E-NNI 提供的信令功能

自动交换光网络(ASON)体系架构支持按照管理、策略、传送网络的内在差异等因素,将网络分割成不同的控制域,用来区分不同行政上或者管理上的职责、信任关系、地址方案、基础设施能力、生存性技术、控制功能的分布等。控制域之间的参考点称为 E-NNI,表示支持多域连接建立的业务划分点。通过 E-NNI 参考点交换的域间信息具有公共的语义,而控制域内允许具有不同的语义表示。

E-NNI 信令主要完成跨不同控制域的连接控制。E-NNI 信令与 UNI、NNI 信令协议一起实现端到端传送业务连接的建立。从信令角度来看,运营商内和运营商间 E-NNI 没有区别,其主要区别体现在策略、路由、编址和安全等方面。

图 1 给出了一个多运营商、多设备厂商传送网络的控制域划分的示例。

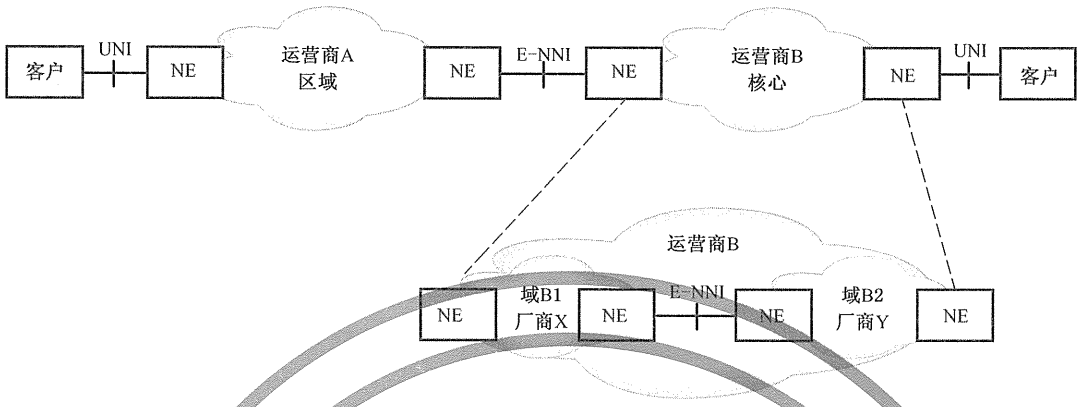


图 1　控制平面划分控制域的示例

E-NNI 信令应支持以下功能：

a）　E-NNI 接口应支持 RSVP-TE 信令协议。

b）　提供跨不同控制域的端到端呼叫(业务)的建立、修改和删除。其中，呼叫修改主要通过增加、删除已建呼叫的连接，或通过修改已建连接的带宽来实现。

c）　提供跨一个或多个域的端到端业务所需的连接建立、删除。

d）　E-NNI 信令应支持 SDH，OTN 和以太网连接业务：

1）　SDH 连接业务：其中 VC4 连接为必选支持，VC12、VC3、连续级联 VC-4-Xc，(X＝4，16，64，256)、虚级联 VC-3/4-Xv，(X＝1，2，…，256)为可选支持；

2）　OTN 连接业务：其中 ODU_k(k＝1，2)、OCh 连接为必选支持，ODU_k(k＝0，3，4)以及级联业务为可选支持；

3）　以太网连接业务：支持以太网业务适配到服务层网络传输，E-NNI 接口不支持以太网交换。

e）　本部分规定的 E-NNI 信令仅支持单层网络(即单层的客户和服务层关系)，E-NNI 支持以太网业务交换和多层网络的信令规范待研究。

本部分规定的基于 OIF 的 E-NNI 信令符合 OIF E-NNI2.0 的要求，OIF E-NNI1.0 和 2.0 的兼容性说明参见附录 D。基于 IETF 的域间信令见附录 A。

4.2　E-NNI 支持的连接类型

4.2.1　连接类型

GB/T 21645.1—2008 定义了 ASON 支持的 3 种连接类型：

a）　PC(永久连接)：PC 是由管理系统指配的连接类型。

b）　SC(交换连接)：SC 是由终端用户发出请求，在终端用户之间以信令方式建立的连接。

c）　SPC(软永久连接)：SPC 由网络边缘的 PC 和网络内部的 SC 一起构成端到端的连接。另外，端到端业务的一侧用户-网络部分可能由网络管理系统配置，称为 SPC/SC 混合业务。

E-NNI 信令应支持跨越多个控制域提供 SPC、SC 业务或 SPC/SC 混合业务，如图 2 所示。

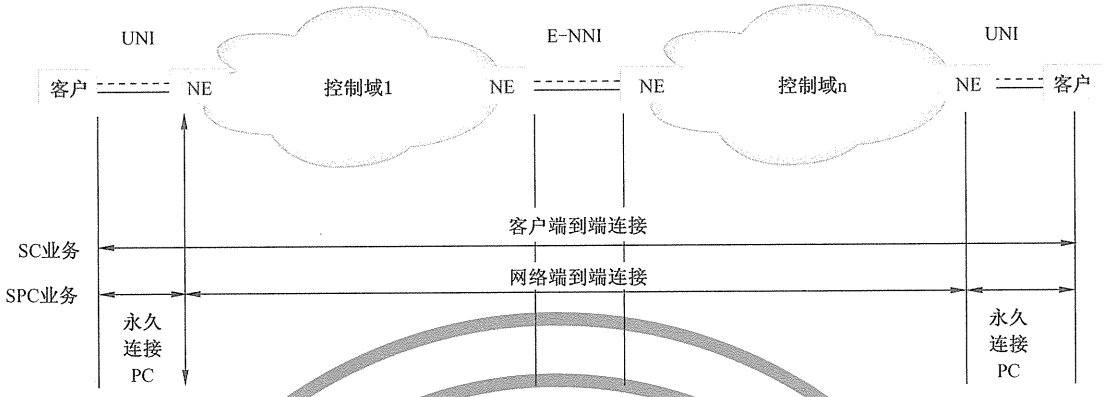


图 2　跨多个控制域的 SC 和 SPC 业务

4.2.2　支持交换连接

当经过一个或多个 E-NNI 接口提供端到端 SC 时，源呼叫方的呼叫参数应在 E-NNI 接口被保存和传递。例如，E-NNI 信令消息中应传递业务相关属性（例如与路由多样性相关的属性）。业务可能在不同域内具有不同的实现方式。此外，在 E-NNI 信令消息中应支持传递显式路由信息。

4.2.3　支持软交换连接

通过一个或多个 E-NNI 接口提供软永久连接 SPC 时，源端网络呼叫参数应在 E-NNI 接口被保存和传递。例如，在 E-NNI 信令消息中应传递业务相关属性（例如与路由多样性相关的属性）。同样，业务可能在不同域内具有不同的实现方式。此外，在 E-NNI 信令消息中应支持传递显式路由信息。

4.2.4　支持混合交换连接/软永久连接业务

一条连接的两个端点中的一端支持信令，另一端不支持信令，这样的连接称为混合交换连接/软永久连接。源端信令控制器不能确定连接的宿端被配置为软永久连接操作，仅宿端网络信令控制器能够根据出口配置确定出口接口类型。

通过一个或多个 E-NNI 接口提供混合 SC/SPC 连接时，源端网络呼叫参数应在 E-NNI 接口被保存和传递。在 E-NNI 信令消息中应传递业务相关属性（例如与路由多样性相关的属性）。同样，业务可能在不同域内具有不同的实现方式。此外，在 E-NNI 信令消息中应支持传递显式路由信息。

4.3　E-NNI 呼叫和连接分离功能

呼叫和连接分离是 ASON 的基本要求，这保证了业务请求本身与网络实现手段的分离。由于呼叫是端到端的业务关联，呼叫状态不仅保存在终端点，还应保存在策略应用点（E-NNI 接口）中。

呼叫和连接分离可以采用以下两种信令方式实现：

——方法一：呼叫操作和连接操作完全分离，即采用独立的呼叫和连接消息；

——方法二：呼叫和连接操作逻辑分离，即采用已有的连接协议完成呼叫控制。

本部分规定的基于 OIF 的 E-NNI 信令采用方法二，即呼叫建立和删除与连接建立和删除的信令同时进行，呼叫控制信息被承载在连接控制信息中。本部分附录 A 规定的基于 IETF 的域间信令采用方法一，即呼叫和连接控制采用独立的信令消息。

图 3 给出跨越多域的呼叫和连接分离的示例。跨越多个控制域的呼叫由呼叫控制点之间的多个呼

叫段组成。连接控制仅限于单个呼叫段内部(例如由于每个域的生存性方案不同,一个业务可以在不同域中采用不同的保护恢复实现)。为支持一个呼叫段,可以建立一个或多个连接。在一个端到端呼叫中,每个呼叫段关联的连接数量可以不同。在图3示例中,UNI呼叫段包含1个链路连接,而控制域1呼叫段包含2个子网连接。

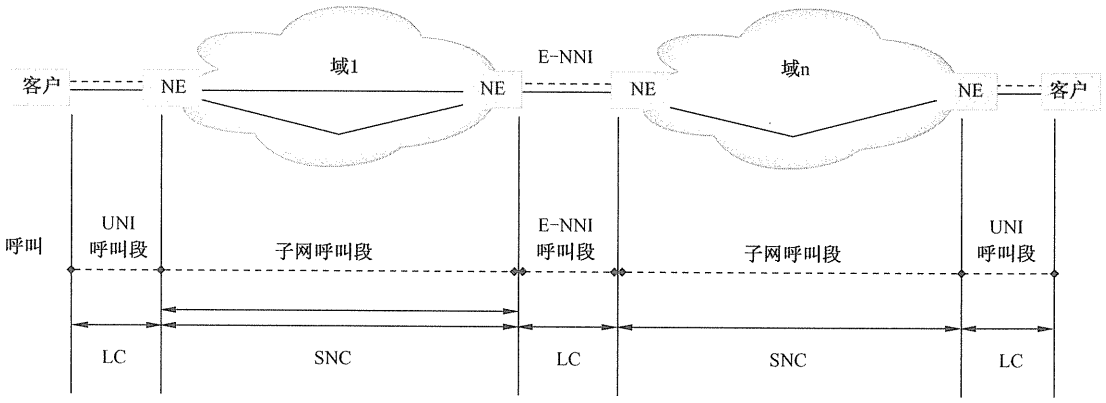


图3 经过多信令控制域的呼叫和连接分离

呼叫和连接分离的架构,允许不同的域灵活选择信令,例如UNI和E-NNI呼叫段使用GMPLS RSVP-TE信令,而I-NNI呼叫段使用其他信令协议。

5 E-NNI信令参考配置

5.1 E-NNI支持的控制平面元件

E-NNI信令接口主要包含以下控制平面元件:

a) 连接控制器(CC):连接控制器元件互相协作完成连接建立。

b) 主叫/被叫方呼叫控制器(CCC-a/CCC-z)和网络呼叫控制器(NCC):主叫、被叫方呼叫控制器负责控制呼叫的建立、释放和修改。呼叫控制器与业务划分点关联,即CCC-a/CCC-z与UNI-C关联,NCC与UNI-N和E-NNI关联。

c) 协议控制器(PC):协议控制器负责把控制元件(例如CC、CCC、NCC)的参数映射到具体的协议消息中。

图4描述了跨域的呼叫控制器之间的交互关系。

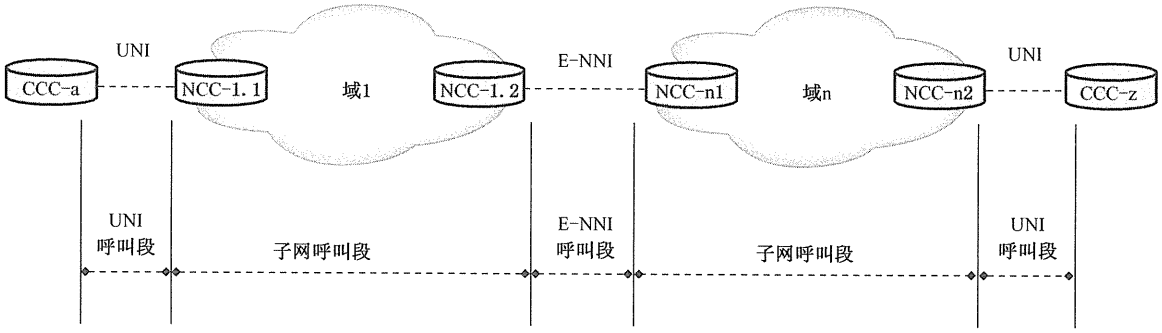


图4 跨域多个控制域的呼叫控制器交互

主叫方呼叫控制器(CCC-a)与被叫方呼叫控制器(CCC-z)通过中间的一或多个网络呼叫控制器

NCC 通信。

网络边界(即 UNI-N)和控制域之间的网关提供 NCC 功能,NCC 执行的功能由 UNI 和 E-NNI 参考点上的策略定义。网络呼叫控制器(NCC)负责以下功能:

a) 在每个控制域内负责关联 SNC 和呼叫(NCC-1 入口和出口负责控制域 1,NCC-n 入口和出口负责控制域 n);

b) 与 CCC-a 和 CCC-z 交互(入口 NCC-1 与 CCC-a 交互,出口 NCC-n 与 CCC-z 交互)来关联链路连接(LC)和呼叫;

c) 在 E-NNI 控制域边界与对等 NCC 交互,来关联 LC 和呼叫(出口 NCC-1 和入口 NCC-n)。

连接控制器 CC 负责建立与每个呼叫段关联的连接。

5.2 E-NNI 信令参考配置

E-NNI 参考配置如图 5 所示,E-NNI 位于两个控制域之间,上游协议控制器(eNNI-U)发送呼叫请求,下游协议控制器(eNNI-D)接收请求。上游/下游协议控制器包括网络呼叫控制器(NCC)和连接控制器(CC)功能。

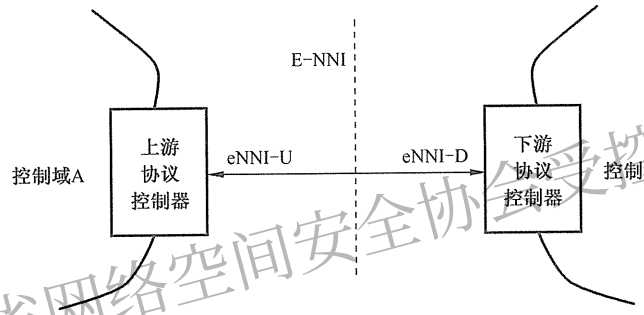


图 5 E-NNI 参考配置

两个控制域之间可以由不支持 ASON 的传送网相连,如图 6 所示。这时,协议控制器将固定连接子网看作一个简单的链路,该链路可能有一定的限制或属性(如只有部分时隙可用)。

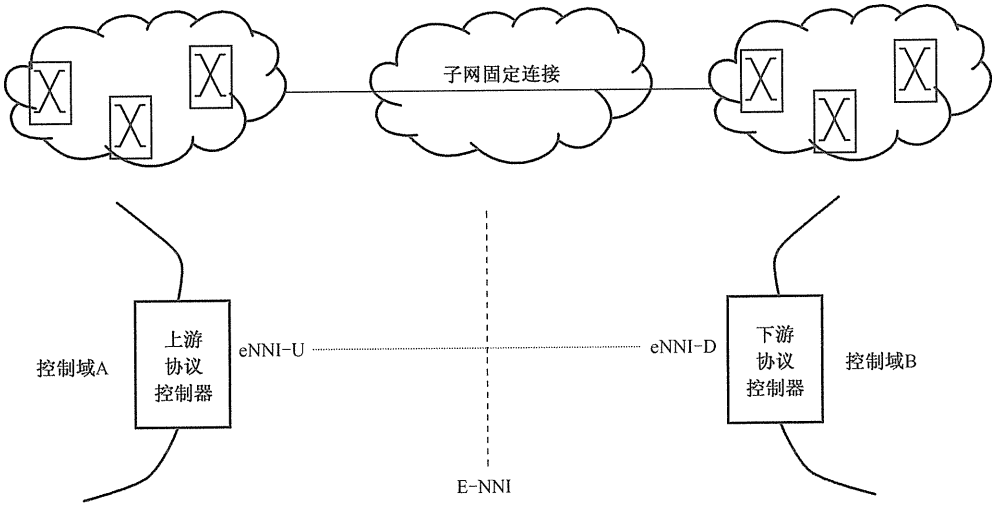


图 6 经过固定连接子网的 E-NNI 参考配置

5.3 E-NNI 信令调用模型

与 UNI 信令调用模型相似，E-NNI 信令调用模型包含直接调用和间接调用两种模型。对于直接调用模型，NCC PC 集成在网元内部；对于非直接调用模型，NCC PC 不与 NE 集成，并可以支持一个或者多个网元的 E-NNI 功能。图 7 给出了调用模型的示例。

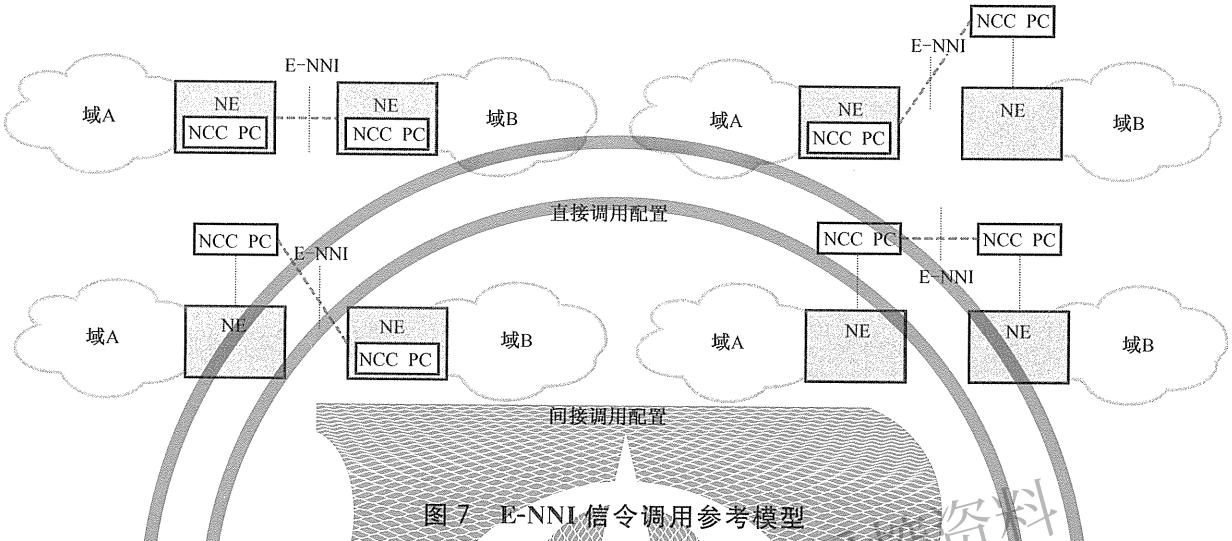


图 7 E-NNI 信令调用参考模型

5.4 用于信令的标识符

E-NNI 信令相关的标识符包括 3 类：传送资源名称、信令协议控制器标识符和 SCN 地址。

5.4.1 传送资源名称

传送资源标识(TRI)用于 ASON 控制平面元件标识 ITU-T G.805 的传送平面资源，包括传送网络分配地址(TNA)和子网点池(SNPP)标识。

a) TNA 地址用于标识位于 UNI 参考点上的传送资源。在 E-NNI 参考点上，传送资源标识标识网络控制域之间的资源，如图 8 所示。CCC 和 NCC 使用 UNI 和 E-NNI 的 TRI 标识，为呼叫控制消息提供源端和宿端信息。

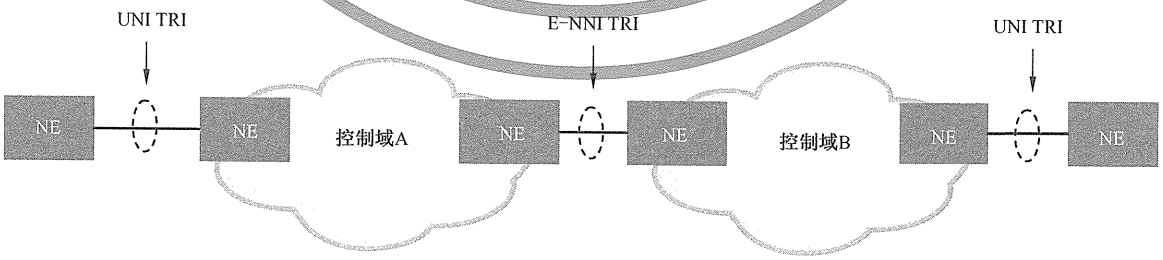


图 8 ASON 传送资源标识

b) SNPP 表示用于路由的一组子网点集合。一条 SNPP 链路表示了位于不同子网或路由域的 SNPP 的关联关系。SNPP 链路标识用于区分这些链路。SNPP 链路的两端通过自动发现或者人工配置的方法进行关联。SNPP ID 标识了这些 SNPP 链路的端点，如图 9 所示。

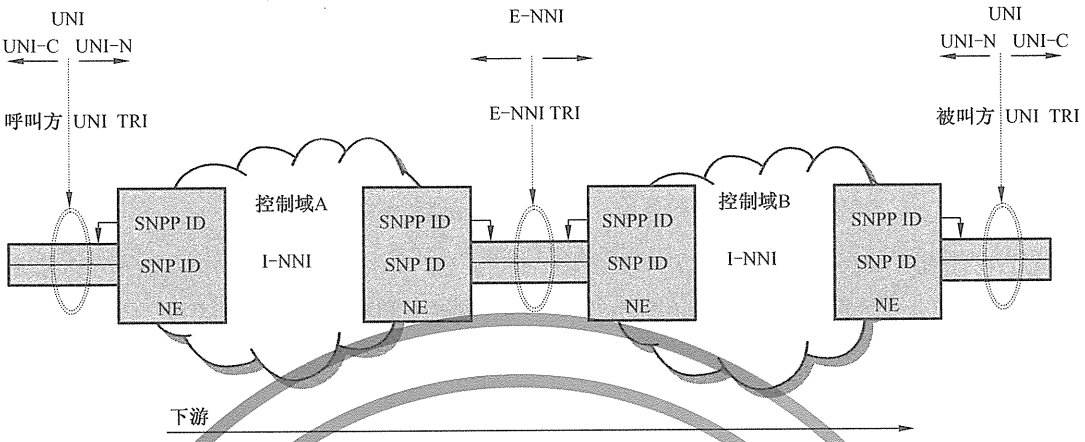


图 9 SNP 和 SNPP ID 示例

UNI/E-NNI 传送资源标识（TRI）与 SNPP 链路标识可以具有 1：N 或者 N：1 的关系。这种关系允许传送资源使用别名（对于相同的资源给予多个名字），也允许接入相同资源的一组链路使用同一个 TRI 来标识。图 10 给出了每个 E-NNI 传送资源对应一个或多个 SNPP 链路的例子。

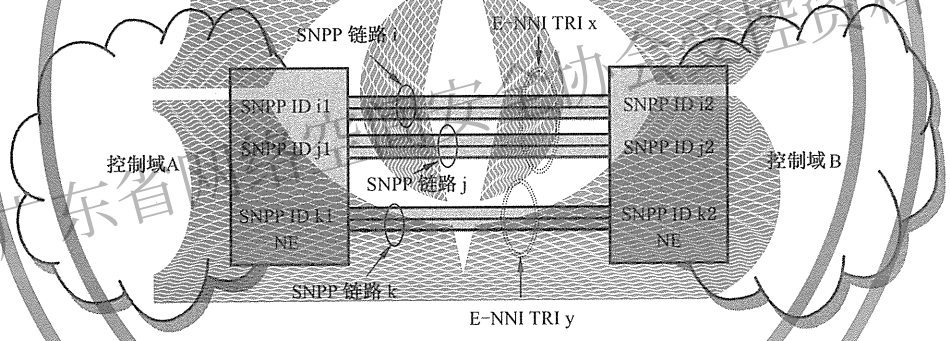


图 10 SNPP ID 和 E-NNI TRI 的关系

表 1 给出 ITU-T、OIF UNI 2.0、OIF E-NNI 2.0 在 E-NNI 中使用的 TRI 的关系和映射。

表 1 传送资源标识符映射表

ITU-T		OIF		说 明
G.8080/G.7713	范围	OIF UNI 2.0	OIF E-NNI 信令 2.0	
SNPP ID	本地	节点 ID(Node ID)/逻辑端口 ID(Logical Port ID)	Node ID/ Logical Port ID	G.7713—2006 的表 7-3 在下游节点的一条链路可以支持多个 SNPP ID
[E-NNI 上游上下文] SNPP ID	本地	N/A	Node ID/源 Logical Port ID	G.7713 未规范
[E-NNI 下游上下文] SNPP ID	本地	N/A	Node ID/宿 Logical Port ID	G.7713 未规范

表 1（续）

ITU-T		OIF		说　明
G.8080/G.7713	范围	OIF UNI 2.0	OIF E-NNI 信令 2.0	
SNP ID	本地	通用标签	通用标签	通用标签应从一个逻辑端口 ID 获得
[E-NNI 上游上下文] SNP ID	本地	N/A	源端通用标签	
[E-NNI 下游上下文] SNP ID	本地	N/A	宿端通用标签	
[UNI-N] 目的 SNP ID	远端	SPC 宿端通用标签	N/A	范围是 SNPP（可能不在 UNI TRI 范围内）或者 UNI TRI
UNI 传送资源 ID（TRI）	端到端	TNA 名字	N/A	E-NNI 行为：端到端或透明承载
呼叫方 UNI TRI		源 TNA		G.7713—2006 的表 7-1
被叫方 UNI TRI		宿 TNA		G.7713—2006 的表 7-1
E-NNI 传送资源 ID	端到端	N/A	TNA	当 E-NNI 链路的任何一端不希望暴露内部路由地址给对方时，可以使用 E-NNI TRI

UNI 和 E-NNI 中 TRI、SNPP 和 SNP ID 的术语如图 11 所示。

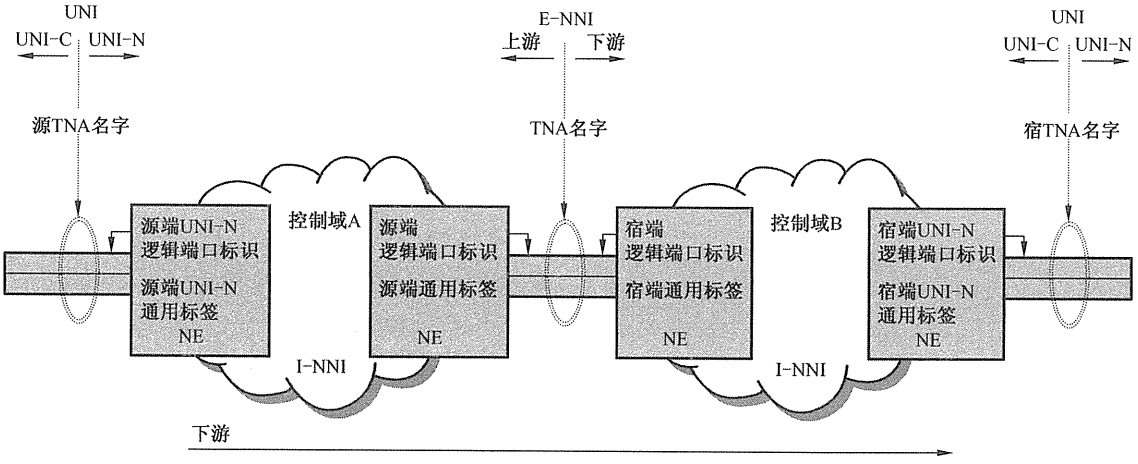


图 11　OIF UNI 2.0 和 OIF E-NNI 2.0 术语的示例

5.4.2 信令协议控制器标识符

在 E-NNI 中，上游节点 NCC PC(eNNI-U)标识符称为发起方 NCC PC ID，下游节点 NCC PC (eNNI-D)标识符称为终结方 NCC PC ID。为了能够标明一个特定的 E-NNI 信令通道，信令 PC(eNNI-U 和 eNNI-D)的标识符应是唯一的。本部分要求信令 PC 使用 IPv4 或 IPv6 作为标识符。

PC 用于特定协议的通信，NCC 和 CCC PC 应具有标识符，允许它们与对端保持呼叫信令关系。在 E-NNI 的本地上下文中，上游节点 NCC PC 标识称为发起方 NCC PC ID，下游方向的 NCC PC 标识称为终结方 NCC PC ID，如图 12 所示。这些标识符应唯一标识一个特定 E-NNI 信道。

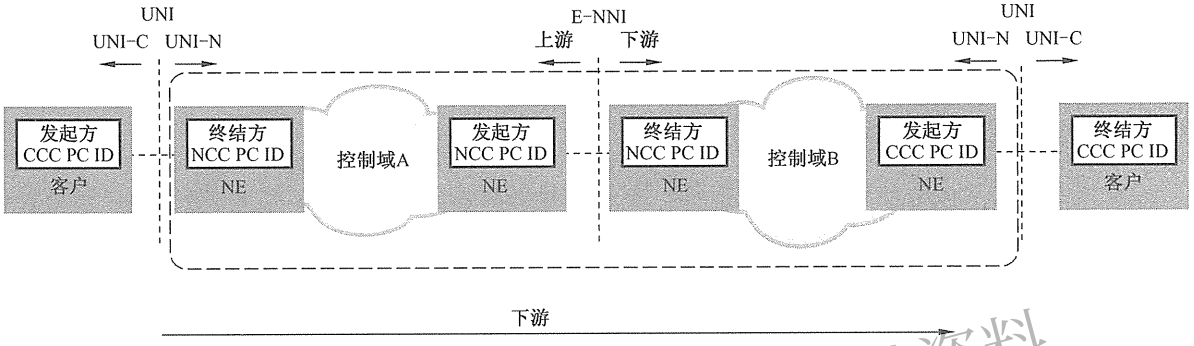


图 12 上游和下游 NCC PC ID

表 2 给出了 UNI-N 和 E-NNI 信令 PC ID 的关系。

表 2 信令协议控制器标识关系(下游上下文)

ITU-T		OIF	
G.8080/G.7713	范围	OIF UNI 2.0	OIF E-NNI 2.0
NCC PC ID	全局	UNI-N SC PC ID	ENNI SC PC ID
发起方 NCC PC ID	全局	位于宿 UNI 的宿端 UNI-N SC PC ID	eNNI-U SC PC ID
终结方 NCC PC ID	全局	位于源 UNI 的源端 UNI-N SC PC ID	eNNI-D SC PC ID
CC PC ID	全局	UNI SC PC ID	eNNI SC PC ID
发起方 CC PC ID	全局	位于宿 UNI 的宿端 UNI-N SC PC ID	与 eNNI-U SC PC ID 共享
终结方 CC PC ID	全局	位于源 UNI-N 的 UNI-N SC PC ID	与 eNNI-D SC PC ID 共享

SC PC ID 的全局范围使用 SC PC ID 提供运营商域内的呼叫 ID。对于跨运营商的呼叫，用于呼叫 ID 的 SC PC ID，还应包含运营商 ID。

在 OIF E-NNI 2.0 中，NCC PC ID 和 CC PC ID 共享相同的标识符，为了实现呼叫和控制的完全分离，需要不同的标识符，有待进一步研究。

图 13 描述了 UNI 和 E-NNI 中的相关术语。

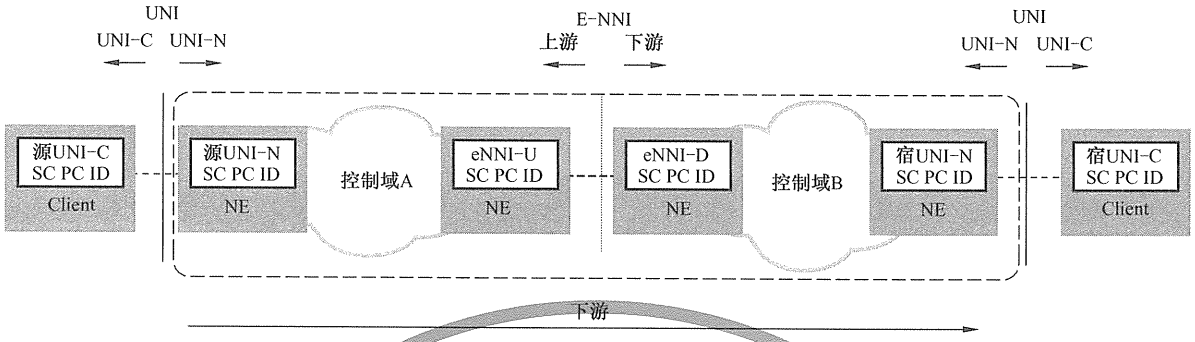


图 13 OIF UNI 2.0 和 OIF E-NNI 2.0 中的信令 PC ID

5.4.3 信令协议控制器 SCN 地址

SCN 用于控制平面组件之间的相互通信。实现控制平面通信功能的 PC 需要使用 SCN 地址。信令 PC SCN 地址表示信令 PC 与 SCN 的连接点，如图 14 所示。因此信令 PC SCN 地址基于承载信令消息的 SCN 拓扑，而不是传送平面或控制平面的拓扑。

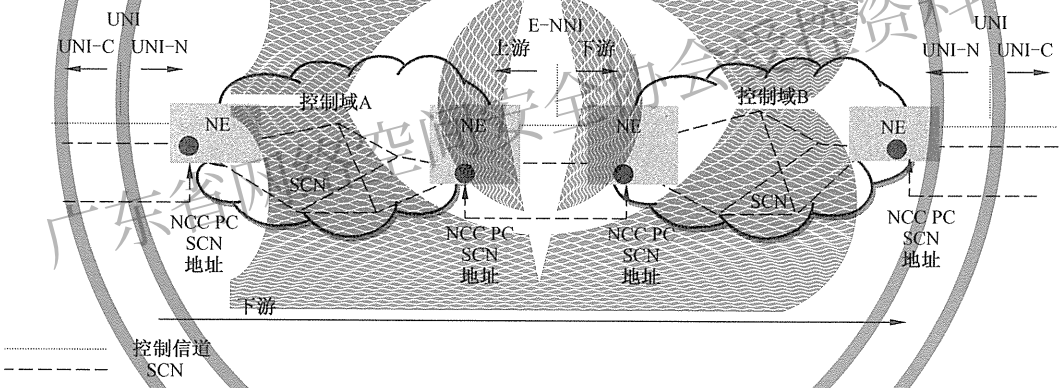


图 14 NCC PC SCN 地址

虽然协议控制器 ID 与协议控制器 SCN 地址不同，但它们可以分配相同的值以便于管理。表 3 给出了 ITU-T、OIF UNI、OIF E-NNI 中信令 PC SCN 地址的关系。

表 3 信令协议控制器 SCN 地址关系

ITU-T		OIF UNI 2.0	OIF E-NNI 2.0	说 明
G.8080/ G.7713	范围			
NCC PC SCN 地址	本地	UNI-N SC PC SCN 地址	eNNI SC PC SCN 地址	
发起方 NCC PC SCN 地址	本地	宿端 UNI-N SC PC SCN 地址(位于宿 UNI)	eNNI-U SC PC SCN 地址	
发起方 NCC PC SCN 地址	本地	源 UNI-N SC PC SCN 地址(位于源 UNI)	eNNI-D SC PC SCN 地址	

表 3（续）

ITU-T		OIF UNI 2.0	OIF E-NNI 2.0	说　明
G.8080/G.7713	范围			
CC PC SCN 地址	本地	与 UNI SC PC SCN 地址共享	与 NCC PC SCN 地址共享	为了支持呼叫和链接完全分离，可能需要独立的地址
发起方 CC PC SCN 地址	本地	源 UNI：与源 UNI-C SC PC SCN 地址共享。宿 UNI：与宿 UNI-N SC PC SCN 地址共享	与发起方 NCC PC SCN 地址共享	为了支持呼叫和链接完全分离，可能需要独立的地址
终结方 CC PC SCN 地址	本地	源 UNI：与源 UNI-C SC PC SCN 地址共享。宿 UNI：与宿 UNI-N SC PC SCN 地址共享	与终结方 NCC PC SCN 地址共享	为了支持呼叫和链接完全分离，可能需要独立的地址

5.5　信令通信网要求

E-NNI 接口应使用可靠的信令通信网来传送控制信息。本部分规定的 E-NNI 接口应支持 GB/T 21645.3—2009 定义的 DCN 体系结构和功能要求。

根据是运营商内或运营商间的 E-NNI，E-NNI 的 SCN 可以有不同的选项，例如：

a）　不同 SCN 路由区相互独立，在路由区之间人工配置可达地址；

b）　为不同的 SCN 构建一个公共的父 SCN 路由区，由 0 层路由器进行 SCN 之间的互联。

图 15 给出了 E-NNI SCN 结构的示例。

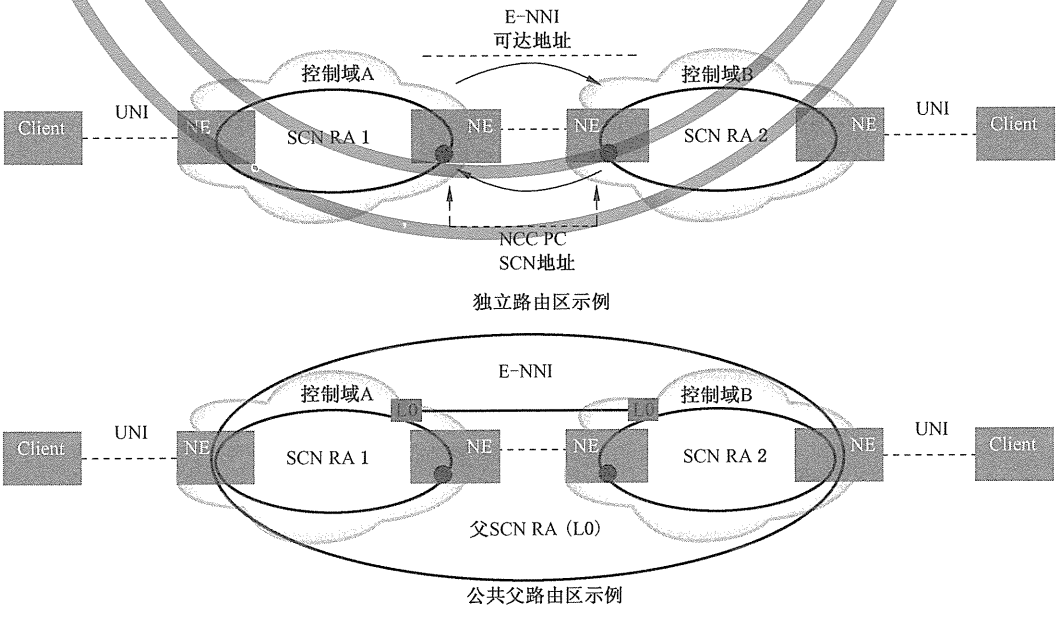


图 15　E-NNI SCN 结构示例

6 E-NNI 信令抽象消息和属性

6.1 抽象消息和错误编码

6.1.1 抽象消息类型

E-NNI 信令用于在控制域接口之间交换消息，进行连接管理。本章给出 E-NNI 信令需要使用的抽象消息，即不依赖于所使用的具体协议。本章规定的 E-NNI 信令抽象消息和属性主要依据 OIF E-NNI 2.0 规范。

E-NNI 抽象呼叫消息属性包含在连接消息中。表 4 给出了 E-NNI 支持的抽象呼叫消息，这些消息分为 5 类，包括连接建立消息、连接释放消息、连接查询消息、通知消息和修改消息。抽象呼叫消息不适用于采用 OIF E-NNI 1.0 规范的实现。

表 4 E-NNI 抽象呼叫消息

抽象消息	所属的信令消息
呼叫建立请求	属于呼叫的第一个连接的连接建立请求消息
呼叫建立指示	属于呼叫的第一个连接的连接建立指示消息
呼叫建立确认	属于呼叫的第一个连接的连接建立确认消息
呼叫释放请求	属于呼叫的最后一个连接的连接释放请求消息
呼叫释放指示	属于呼叫的最后一个连接的连接释放指示消息
呼叫查询请求	属于呼叫中每一个连接的连接查询请求消息
呼叫查询指示	对于呼叫的每一个连接，此呼叫信息搭载在连接查询请求指示信息中
呼叫修改请求	如果现存连接的带宽变化引起呼叫的修改，呼叫修改请求可以暗含在连接修改请求中。 如果呼叫的修改由于连接的添加引起，呼叫修改请求可以暗含在连接建立请求中。 如果呼叫的修改由于移除连接引起，呼叫修改请求可以暗含在连接释放请求中
呼叫修改指示	如果现存连接的带宽变化引起呼叫的修改，呼叫修改指示可以暗含在连接修改指示中。 如果呼叫的修改由于连接的添加引起，呼叫修改指示可以暗含在连接建立指示中。 如果呼叫的修改由于移除连接引起，呼叫修改指示可以暗含在连接释放指示中
呼叫修改确认	如果现存连接的带宽变化引起呼叫的修改，呼叫修改确认可以暗含在连接修改确认中。 如果呼叫的修改由于连接的添加引起，呼叫修改确认可以暗含在连接建立确认中。 如果呼叫的修改由于移除连接引起，呼叫修改确认可以暗含在连接释放确认中

表 5 在 ITU-T G.7713 的基础上，给出了 E-NNI 支持的抽象连接消息，这些消息分为 5 类，包括连接建立消息、连接释放消息、连接查询消息、连接通知消息和连接修改消息。其中连接修改消息不适用于采用 OIF E-NNI 1.0 规范的实现。

表 5　E-NNI 信令抽象消息

抽象消息			版本
连接建立消息	ConnectionSetupRequest	连接建立请求	OIF E-NNI 1.0/OIF E-NNI 2.0
	ConnectionSetupIndication	连接建立指示	OIF E-NNI 1.0/OIF E-NNI 2.0
	ConnectionSetupConfirm（可选）	连接建立确认(可选)	OIF E-NNI 1.0/OIF E-NNI 2.0
连接释放消息	ConnectionReleaseRequest	连接释放请求	OIF E-NNI 1.0/OIF E-NNI 2.0
	ConnectionReleaseIndication	连接释放指示	OIF E-NNI 1.0/OIF E-NNI 2.0
连接查询消息	ConnectionQueryRequest	连接查询请求	OIF E-NNI 1.0/OIF E-NNI 2.0
	ConnectionQueryIndication	连接查询指示	OIF E-NNI 1.0/OIF E-NNI 2.0
连接通知消息	ConnectionNotify	连接通知	OIF E-NNI 1.0/OIF E-NNI 2.0
连接修改消息	ConnectionModifyRequest	连接修改请求	OIF E-NNI 2.0
	ConnectionModifyIndication	连接修改指示	OIF E-NNI 2.0
	ConnectionModifyConfirm	连接修改确认	OIF E-NNI 2.0

6.1.2　连接建立消息

6.1.2.1　连接建立请求

eNNI-U 向 eNNI-D 发送 ConnectionSetupRequest 消息请求建立连接。此消息提供在 E-NNI 上建立连接的信息，还提供下游 CC 完成端到端连接所使用的信息。此消息为全局范围，即消息被端到端传递用于指示业务请求。此消息不承载错误代码。表 6 给出了抽象消息和内容。

表 6　连接建立请求(ConnectionSetupRequest)抽象消息

消息：ConnectionSetupRequest 范围：全局 方向：eNNI-U→eNNI-D			
属　　性	类型	范围	呼叫/连接
源 TNA	强制	全局	呼叫
目的 TNA	强制	全局	呼叫
发起方 NCC PC ID	强制	局部	连接
终结方 NCC PC ID	强制	局部	连接
连接名称	强制	局部	连接
呼叫名称	强制	全局	呼叫
SNP ID[a]	可选	局部	连接
SNPP ID	强制	局部	连接
方向性	可选	全局	连接
目的 SNP ID[b]	可选	局部[c]	呼叫
路由	可选	全局[d]	

表 6（续）

消息：ConnectionSetupRequest 范围：全局 方向：eNNI-U→eNNI-D			
属　　性	类型	范围	呼叫/连接
业务级别	强制[e]	局部[f]	呼叫
合同号	可选	全局	呼叫
编码类型	强制	全局	呼叫/连接
交换类型	强制	全局	呼叫/连接
SDH、OTN 或 以太网流量参数	强制	全局	呼叫/连接
通用负荷标识	可选	全局	呼叫

[a] SNP 和 SNPP ID 为分配给本地链路的资源的标识符。它们是一系列 ID 的列表。对一个双向连接请求，SNP 和 SNPP ID 列表包含上游和下游的 ID 信息。SNP ID 和目的 SNP ID 作为可选信息。

[b] 宿 SNP ID 指连接或呼叫的最终地址的 SNP ID，可以是一个 UNI-C 或者 UNI-N 的 SNP。

[c] 此属性只在 SPC 连接的目的节点有意义，在所有中间节点透传此 ID。

[d] 此属性提供跨多控制域的路由信息，并可提供具体节点路由（如果策略允许）。因此，此路由信息可以跨多个E-NNI。

[e] 业务级别是一个强制属性。如果此属性不存在，需要使用一个默认的业务级别属性。

[f] 此属性在单个运营商内唯一。它允许在不同的控制域之间转换等价的业务级别信息。

6.1.2.2　连接建立指示

此消息由 eNNI-D 发送给 eNNI-U，用于响应连接建立请求。此消息提供在 E-NNI 上建立连接的信息，还提供可能用于 eNNI-U 完成端到端请求的附加信息，例如使用为请求分配的 SNP。当建立被拒绝时，此消息包括有关拒绝的错误代码。此消息为全局消息，即此消息被端到端传送。表 7 显示了此消息和内容。

表 7　连接建立指示（ConnectionSetupIndication）抽象消息

消息：ConnectionSetupIndication 范围：全局 方向：eNNI-D→eNNI-U			
属　　性	类型	范围	呼叫/连接
发起方 NCC PC ID	强制	局部	连接
终结方 NCC PC ID	强制	局部	连接
连接名称	强制	局部	连接
呼叫名称	强制	全局	呼叫
SNP ID	强制	局部	连接
连接状态	强制	局部	连接
方向性	可选	全局	连接
错误代码	可选	局部/全局	连接
是否支持带宽修改	强制	全局	连接

此消息包含如下错误代码：

a） 被叫方忙；

b） 未认证发送者(策略错误)；

c） 未认证接收者(策略错误)；

d） 无效连接 ID；

e） 无效呼叫 ID；

f） 无效 SNP；

g） 不可用 SNP；

h） 无效 SNPP；

i） 不可用 SNPP；

j） 不可用方向性；

k） 无效 SPC SNP；

l） 无效路由；

m） 不可用业务级别。

6.1.2.3 连接建立确认

此消息为可选消息，用于 eNNI-U 向 eNNI-D 发送建立指示的响应。如果在连接建立指示中请求了确认消息，此消息是一个强制消息。此消息提供了在 E-NNI 上，对端到端连接建立的最终确认。当建立被拒绝时，此消息包含有关拒绝的错误代码。此消息为全局范围，即此消息被端到端传送。表 8 显示了此消息和内容。

表 8 连接建立确认(ConnectionSetupConfirm)抽象消息

消息：ConnectionSetupConfirm 范围：全局 方向：eNNI-U→eNNI-D			
属 性	类型	范围	呼叫/连接
发起方 NCC PC ID	强制	局部	连接
终结方 NCC PC ID	强制	局部	连接
连接名称	强制	局部	连接
连接状态	强制	局部	连接
呼叫名称	强制	全局	呼叫
错误代码	可选	局部	连接

此消息包括如下错误代码：

a） 未认证发送者(策略错误)；

b） 未认证接收者(策略错误)；

c） 无效连接 ID；

d） 无效呼叫 ID；

e） 无效 SNP；

f） 不可用 SNP。

6.1.3 连接释放消息

6.1.3.1 连接释放请求

此消息由 eNNI-D 或 eNNI-U 发送，用于向相关协议控制器(eNNI-D 或 eNNI-U)发起连接释放请求。此消息提供了用于释放端到端连接的 E-NNI 部分的信息，以及可能用于 eNNI-U 或 eNNI-D 完成端到端请求的附加信息。此消息不携带错误编码。此消息为全局消息。表 9 显示了此消息的内容。

表 9 连接释放请求(ConnectionReleaseRequest)抽象消息属性

消息：ConnectionReleaseRequest 范围：全局 方向：eNNI-D→eNNI-U 或 eNNI-U→eNNI-D			
属　　性	类型	范围	呼叫/连接
发起方 NCC PC ID	强制	局部	连接
终结方 NCC PC ID	强制	局部	连接
连接名称	强制	局部	连接
呼叫名称	强制	全局	呼叫

释放请求可以由一条连接的中间节点产生，使用连接通知消息用来通知源或目的节点发起释放请求。在此情况下，中间 CC 发送请求触发源或目的 CC 发起释放请求，并可能包含附加信息指示此请求是中间节点产生的。此消息不包括错误代码。

6.1.3.2 连接释放指示

此消息由 eNNI-D 或 eNNI-U 发送给相关协议控制器(eNNI-U 或 eNNI-D)，用于对连接释放请求的响应。此消息提供了端到端连接中 E-NNI 部分的最终释放确认。此消息为全局消息，消息被端到端地传递。表 10 为此消息内容。

表 10 连接释放指示(ConnectionReleaseIndication)抽象消息属性

消息：ConnectionSetupIndication 范围：全局 方向：eNNI-D→eNNI-U 或 eNNI-U→eNNI-D			
属　　性	类型	范围	呼叫/连接
发起方 NCC PC ID	强制	局部	连接
终结方 NCC PC ID	强制	局部	连接
连接名称	强制	局部	连接
连接状态	强制	局部	连接
呼叫名称	强制	全局	呼叫
错误代码	可选	局部	连接

此消息包括如下错误代码：

a) 未认证发送者(策略错误)；

b) 未认证接收者(策略错误)；

c) 无效连接 ID；

d) 无效呼叫 ID。

6.1.4 连接查询消息

6.1.4.1 连接查询请求

此消息用于 eNNI-U 或 eNNI-D 向相应协议控制器(eNNI-D 或 eNNI-U)发起查询请求。此消息用于请求同一呼叫中的一或多个已建连接的连接状态信息。如果没有指定连接或者呼叫名称，则返回所有由 eNNI-D 或者 eNNI-U 管理的连接信息。如果只指定了呼叫名称，则返回此呼叫所包含的所有连接信息。此消息为局部消息，表 11 为此消息内容。

表 11 连接查询请求(ConnectionQueryRequest)抽象消息属性

消息：ConnectionQueryRequest 范围：局部 方向：eNNI-D→eNNI-U 或 eNNI-U→eNNI-D			
属 性	类型	范围	呼叫/连接
发起方 NCC PC ID	可选	局部	连接
终结方 NCC PC ID	可选	局部	连接
连接名称	可选	局部	连接
呼叫名称	可选	全局	呼叫

6.1.4.2 连接查询指示

此消息由 eNNI-D 或 eNNI-U 发送，用于响应相关协议控制器(eNNI-D 或 eNNI-U)的查询请求。此消息提供同一呼叫内已建的一或多个连接的描述信息。如果请求多个连接的状态，对每个连接都需要提供以下内容：

——连接名称；

——连接详细信息。

此消息为局部消息。表 12 为此消息内容。

表 12 连接查询指示(ConnectionQueryIndication)抽象消息属性

消息：ConnectionQueryIndication 范围：局部 方向：eNNI-D→eNNI-U 或 eNNI-U→eNNI-D			
属 性	类型	范围	呼叫/连接
发起方 NCC PC ID	强制	局部	连接
终结方 NCC PC ID	强制	局部	连接
连接名称	强制	局部	连接

表 12（续）

消息：ConnectionQueryIndication 范围：局部 方向：eNNI-D→eNNI-U 或 eNNI-U→eNNI-D			
属　　性	类型	范围	呼叫/连接
呼叫名称	强制	全局	呼叫
错误代码	可选	局部	连接
连接详细信息如下：			
连接状态	强制	局部	连接
源 TNA 名称	可选	局部	连接
目的 TNA 名称	可选	局部	连接
本地逻辑端口标识符	可选	局部	连接
本地通用标签	可选	局部	连接
合约 ID	可选	局部	连接
编码类型	可选	局部	连接
交换类型	可选	局部	连接
SDH、OTN 或以太网业务参数	可选	局部	连接
方向	可选	局部	连接
路由	可选	局部	连接
通用静荷标识符	可选	局部	连接
业务级别	可选	局部	连接
分级	可选	局部	连接

“连接详细信息”属性可包含此连接的任何属性，如使用的 SNP、使用的特定恢复机制等，“连接详细信息”属性所包含的信息类型参见属性定义部分。此属性可以包含一个或多个连接的状态信息。例如，此属性可包括所使用的 SNP，特定的恢复机制等。

此消息包括如下错误代码：

a） 未认证发送者（策略错误）；

b） 未认证接收者（策略错误）；

c） 无效/未知连接 ID；

d） 无效/未知呼叫 ID。

6.1.5 连接通知消息

此消息由 eNNI-U 或 eNNI-D 发送，用于通知相关协议控制器（eNNI-D 或 eNNI-U）一个或多个连接的状态。此消息通知需要删除连接的源和目的地址，用于触发连接释放请求。此消息是局部消息，表 13显示了此消息的内容。

表 13　连接通知(ConnectionNotification)抽象消息属性

消息:ConnectionNotification 范围:局部 方向:eNNI-D→eNNI-U 或 eNNI-U→eNNI-D			
属性	类型	范围	呼叫/连接
发起方 NCC PC ID	强制	局部	连接
终结方 NCC PC ID	强制	局部	连接
连接名称(列表)	强制	局部	连接
呼叫名称	强制	全局	呼叫
错误代码	可选	局部	连接
连接状态	可选	局部	连接

此消息包括如下错误代码:

a)　影响业务的缺陷(导致连接失效);

b)　不影响业务的缺陷(不导致连接失效)。

6.1.6　连接修改消息

6.1.6.1　连接修改请求

此消息由 eNNI-U 发送给 eNNI-D,请求对已建立连接的修改。E-NNI 2.0 支持连接带宽修改功能(在 UNI 2.0 中,带宽和标签相关属性可以被修改。例如对于以太网 EVPL 业务,表示 CE-VLAN 标识的通用标签可以被修改。而 E-NNI 仅需要支持带宽修改)。此消息不承载错误代码。表 14 为此消息内容。

表 14　连接修改请求(ConnectionModifyRequest)抽象消息属性

消息:ConnectionModifyRequest 范围:全局 方向:eNNI-D→eNNI-U				
属性	类型	范围	呼叫/连接	是否可修改
源 TNA	强制	全局	呼叫	否
目的 TNA	强制	全局	呼叫	否
发起方 NCCPCID	强制	局部	呼叫	否
终结方 NCCPCID	强制	局部	呼叫	否
连接名称	强制	局部	连接	否
呼叫名称	强制	全局	呼叫	否
SNP ID	可选	局部	连接	是
SNPP ID	强制	局部	连接	否
目的 SNP ID	可选	局部	连接	是
显式路由	可选	全局	连接	是
SDH、OTN 或以太网流量参数	强制	全局	呼叫/连接	是

6.1.6.2 连接修改指示

此消息由 eNNI-D 发送给 eNNI-U，用于确认 eNNI-U 发起的连接修改请求。OIF E-NNI 2.0 支持连接带宽修改指示消息。表 15 为此消息内容。

表 15 连接修改指示(ConnectionModifyIndication)抽象消息属性

消息：ConnectionModifyIndication 范围：全局 方向：eNNI-D→eNNI-U			
属性	类型	范围	呼叫/连接
发起方 NCC PC ID	强制	局部	呼叫
终结方 NCC PC ID	强制	局部	呼叫
连接名称	强制	局部	连接
呼叫名称	强制	全局	呼叫
SNP ID	可选	局部	连接
SNPP ID	强制	局部	连接
连接状态	强制	局部	连接
错误代码	可选	局部	呼叫/连接

6.1.6.3 连接修改确认

此消息用于 eNNI-U 向 eNNI-D 发送修改指示的响应。此消息提供了在 E-NNI 上，对端到端连接修改的最终确认。此消息为全局范围，即此消息被端到端传送。OIF E-NNI 2.0 支持连接修改确认消息。表 16 显示了此消息和内容。

表 16 连接修改确认(ConnectionModifyConfirm)抽象消息属性

消息：ConnectionModifyConfirm 范围：全局 方向：eNNI-U→eNNI-D			
属性	类型	范围	呼叫/连接
发起方 NCC PC ID	强制	局部	呼叫
终结方 NCC PC ID	强制	局部	呼叫
连接名称	强制	局部	连接
连接状态	强制	局部	连接
呼叫名称	强制	全局	呼叫
错误代码	可选	局部	呼叫/连接

6.2 抽象属性

6.2.1 抽象属性类型

表 17 提供了支持 E-NNI 信令的抽象属性列表。

表 17　E-NNI 信令抽象属性

抽象属性	
标识属性	源 TNA
	目的 TNA
	源协议控制器地址
	目的协议控制器地址
	连接名称
	呼叫名称
	SNP ID
	SNPP ID
	目的 SNP ID
业务属性	方向
	编码类型
	交换类型
	SDH、OTN 或以太网业务参数
路由属性	显式路由
策略属性	业务级别
其他属性	连接状态
	错误编码

6.2.2　与标识相关的属性

与标识相关的属性如下：

a)　源 TNA

此属性代表发起业务请求的源端用户 TNA 地址。此属性由源控制域验证，承载在跨 E-NNI 的信令消息中。关于 TNA 地址的具体要求见 GB/T 21645.5—2012。

b)　目的 TNA

此属性代表连接目的端用户 TNA 地址，在出口控制域中可以用来确定到达用户的显式路由。

c)　发起方 NCC PC ID

此属性表示上游节点 NCC PC 的标识符。

d)　终结方 NCC PC ID

此属性表示下游节点 NCC PC 的标识符。

e)　连接名

此属性代表某连接的名称，本地唯一。由发起请求的协议控制器分配此连接名。

f)　呼叫名

此属性为某个呼叫的名称，全局变量。此名在 UNI 呼叫边界点由网络分配；E-NNI 不参与分配，仅进行呼叫名称验证。此名一直保留到呼叫被删除。

g)　SNP ID

此属性表示用于支持跨 E-NNI 业务请求的传送平面链路连接资源（即子网点，用于建立子网连接，

从 SNPP 中选取)。eNNI-U 可以选择使用一个 SNP ID,而 eNNI-D 可以选择另外的 SNP ID 来替代。

h) SNPP ID

此属性表示用于支持跨 E-NNI 业务请求的传送链路资源。SNPP ID 表示了一条 SNPP 链路的发送端(请求端)。由于 SNPP 链路的两端是互相关联的(通过自动发现或者人工配置),因此指定一侧的 SNPP 端点可以表示所要使用的 SNPP 链路。此属性可以采用不同的地址格式实现。例如对于 IP,可以采用编号或者无编号链路,编号或者无编号链路捆束。

i) 目的 SNP ID

此属性表示在目的端网络—用户连接中所使用的 SNP。此地址对目的端来说是局部唯一,并由外部代理(例如发起 SPC 请求的网管系统)提供给控制平面。对于 SPC,源和目的两端的用户—网络连接是指配的,因此需要指定目的 SNP,来保证控制平面将网络中的交换连接与目的端的指配连接正确连接起来。

6.2.3 与业务相关的属性

与业务相关的属性如下:

a) 方向性

此属性表示请求的业务是单向还是双向连接。

b) 编码类型

编码类型经过 UNI 接口时的信号编码格式。编码类型包括:

1) SDH;
2) ODUk 层;
3) OCh 层;
4) 以太网业务。

c) 交换类型

指示一条特定链路上的交换类型。交换类型包括:

1) 二层以太网;
2) TDM(SDH、OTN ODUk);
3) 波长(OTN OCh);
4) 数据通路交换能力(DCSC)。

d) 业务参数

业务参数包括 SDH、OTN 和以太网。

6.2.4 与路由相关的属性

显式路由:此属性表示连接所经过的一系列链路和子网。根据所交换的路由信息数量的不同,此属性可以包含详细的路由信息或抽象视图。不同的信令协议实现此属性的方式可以不同,如 RSVP-TE 使用 ERO 和 RRO 来表示显式路由。

6.2.5 与策略相关属性

业务级别:此属性包含业务级别消息。不同控制域之间对业务级别消息的解释应一致。

6.2.6 其他属性

其他属性如下:

a) 连接状态:表示连接状态。

b) 错误代码:错误代码用于表示连接动作所引起的错误,包括:

1) 呼叫方忙(Calling party busy);
2) 被叫方忙(Called party busy);
3) 未鉴权的发送者(策略错误)[Unauthorized sender(policy error)];
4) 未鉴权的接收者(策略错误)[Unauthorized receiver(policy error)];
5) 无效/未知连接 IDInvalid/unknown connection ID;
6) 无效/未知呼叫 IDInvalid/unknown Call ID;
7) 无效 SNP(Invalid SNP);
8) 不可用 SNP(Unavailable SNP);
9) 无效 SNPP(Invalid SNPP);
10) 不可用 SNPP(Unavailable SNPP);
11) 不可用网络资源(Unavailable network resource);
12) 不可用方向(Unavailable directionality);
13) 无效 SPC SNP(Invalid SPC SNP);
14) 无效路由(Invalid route);
15) 不可用业务等级业务影响缺陷(引起连接失效);
16) 无业务影响缺陷(无失效连接)。

7 E-NNI 信令流程

7.1 概述

本部分描述 E-NNI 接口的信令流程,包括 SC、SPC 正常和异常建立和释放的过程,该信令流程与具体协议无关,也与 UNI 和 I-NNI 接口的协议无关。在下述描述中,只表示 E-NNI 运行所需要的信息交换,并不描述 UNI 和 I-NNI 接口实际的信令流程。

7.2 正常操作

7.2.1 正常建立请求

当建立 SPC 时,假定用户与网络之间的连接已经配置完成,控制平面只是在网络中负责建立连接。网络部分连接建立的请求由网管系统发起并传送到控制域的入口节点,由 E-NNI 接口向下游传播,直到目的控制域的出口节点。在 E-NNI 接口建立跨域的连接需要 3 个消息交换过程(第 3 个消息是可选的)。注意第 3 个消息(建立确认)是可选消息,但是如果目的方请求就应包含此消息。信令流程如图 16 所示。

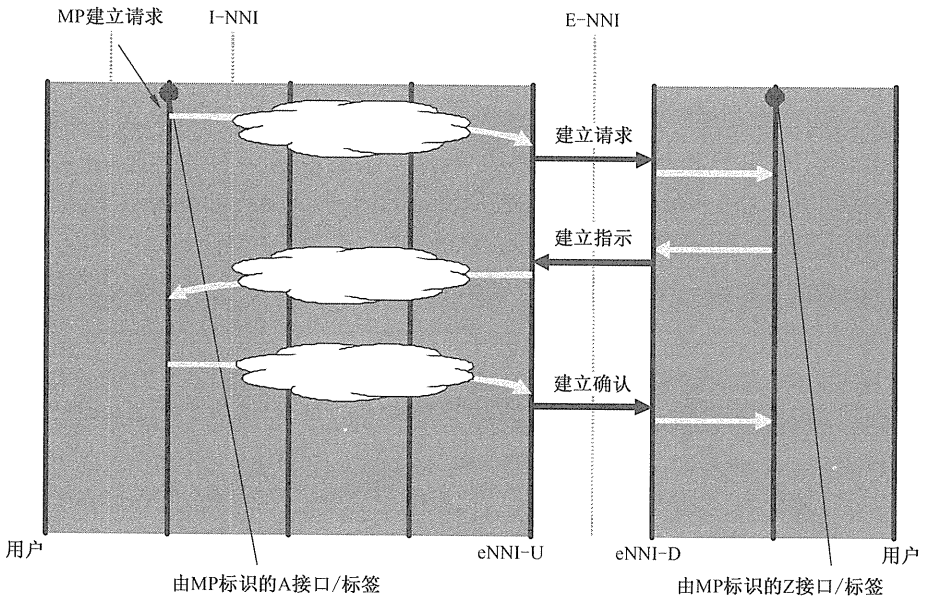


图 16　SPC 建立信令流程

当请求 SC 时，控制平面负责端到端连接的建立，由 UNI 发起连接建立请求。信令流程如图 17 所示。

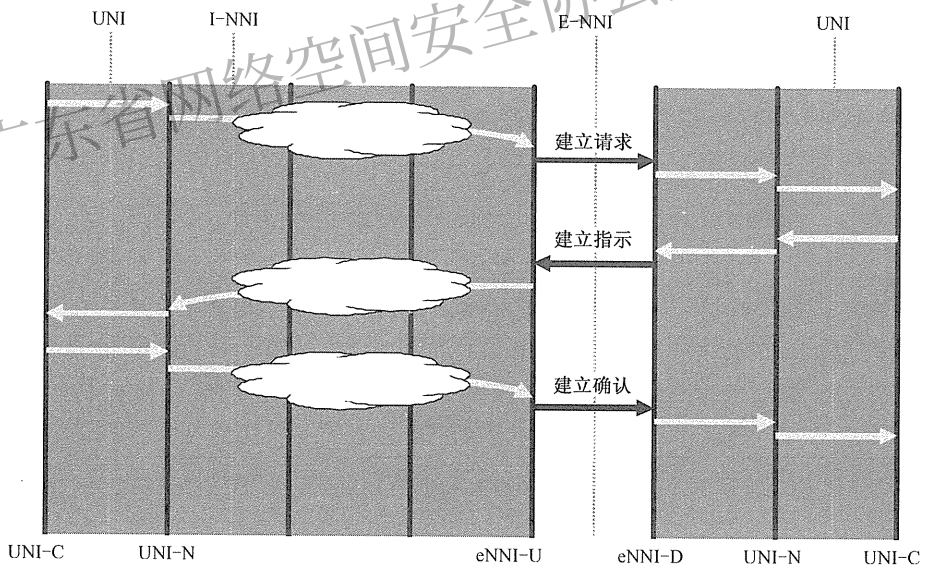


图 17　基本 SC 建立信令流程

7.2.2　正常释放请求

对于 SPC，由网管系统向控制平面发起网络部分连接释放的请求，并且与配置部分（用户与网络之间）连接的释放与否无关。释放网络部分的连接在 E-NNI 接口需要有 2 个消息交换过程。信令流程如图 18 所示。

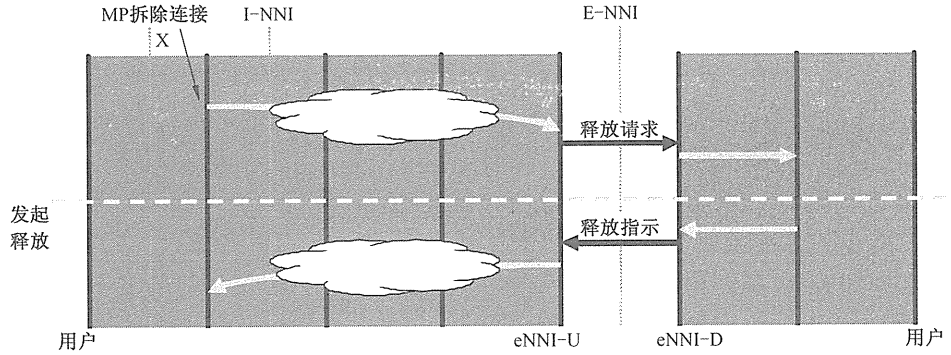


图 18　基本 SPC 释放

释放一个 SC，根据由谁发起释放请求可以有几个信令流程。释放 E-NNI 部分的网络连接在 E-NNI 接口需要有 2 个消息交换。图 19 和图 20 分别表示由源 UNI-C 和宿 UNI-C 发起释放请求的信令流程。

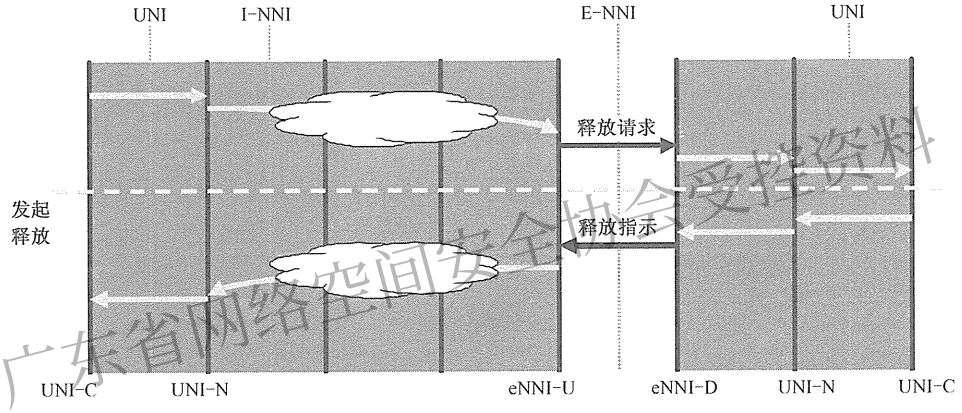


图 19　基本 SC 释放信令流程：源 UNI-C 发起

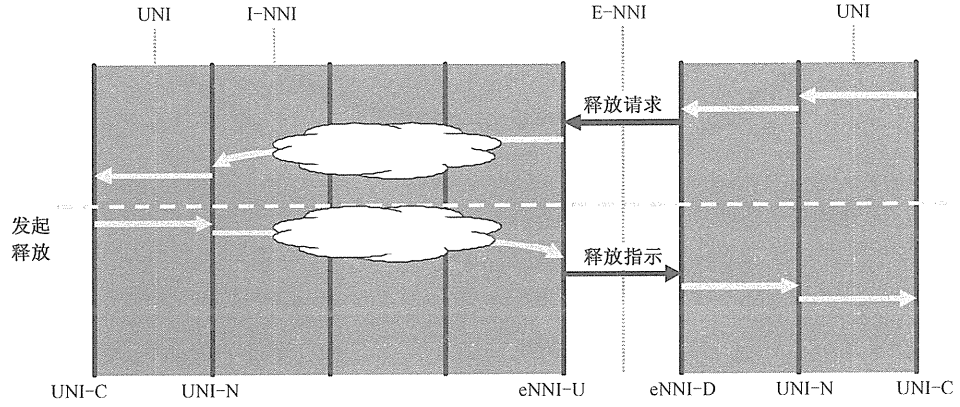


图 20　基本 SC 释放信令流程：宿 UNI-C 发起

由于网络故障或策略原因，网络中间节点也可以发起 SC 连接的释放请求。在这种情况下，网络中间控制平面节点(CC)通过发送释放通知发起连接释放。对于 E-NNI 2.0，eNNI-U 或 eNNI-D 发起的释放通知总是向上游方向发送。中间节点发起的连接释放信令流程如图 21～图 24 所示。

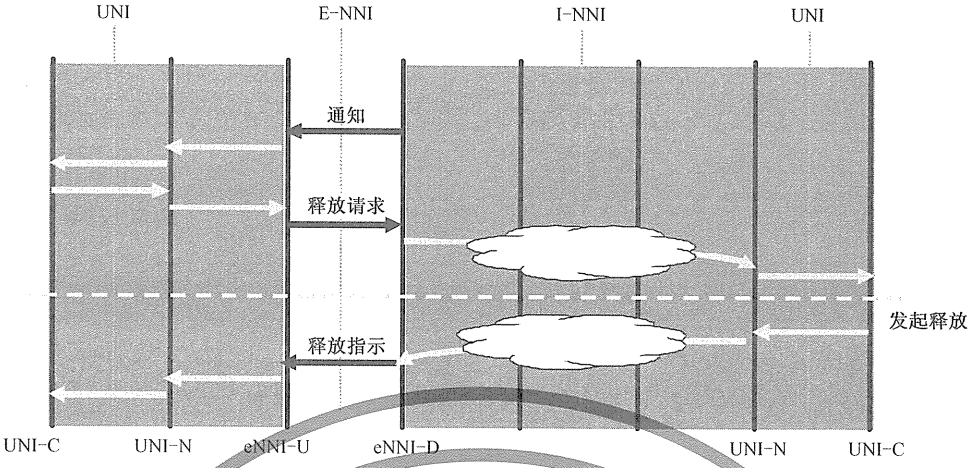


图 21　删除/释放：中间节点 eNNI-D 发起

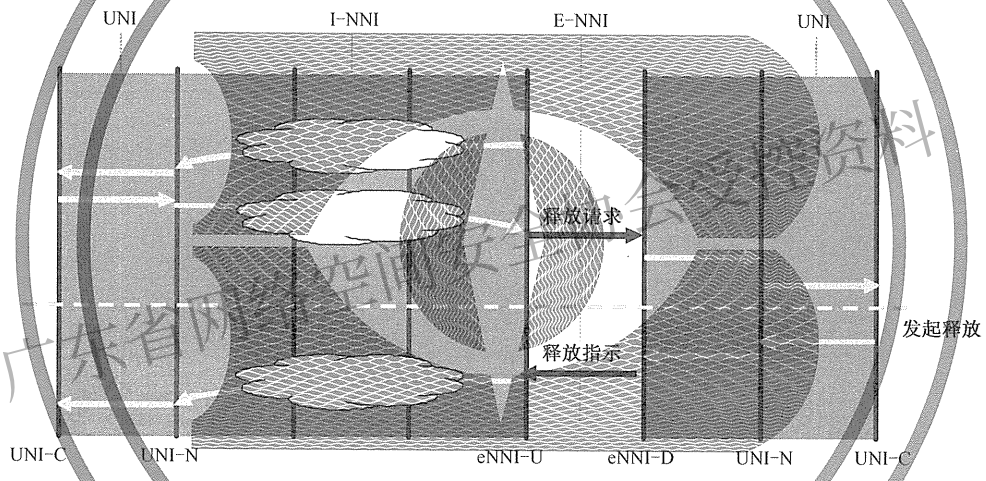


图 22　删除/释放：中间节点 eNNI-U 发起

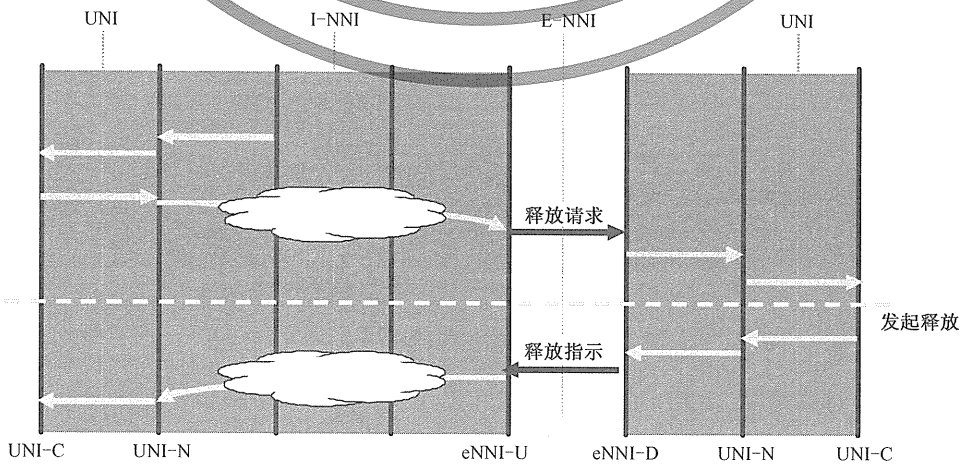


图 23　删除/释放：上游网络节点发起

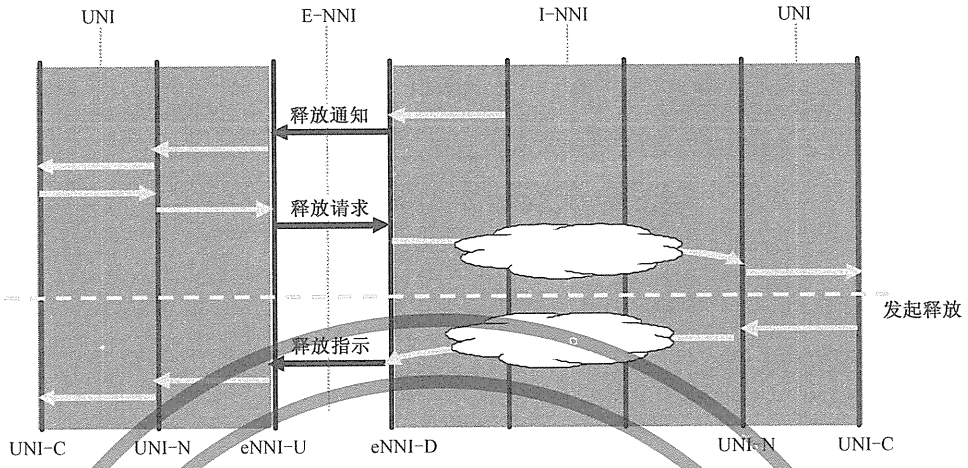


图 24　删除/释放：下游网络节点发起

网络也可能向下游方向发起释放通知，支持 OIF E-NNI 1.0 的 eNNI-U 或 eNNI-D 可能存在这种情况。此时，OIF E-NNI 2.0 接口应把释放通知转发到目的节点。一旦释放通知达到目的节点，则信令行为将与图 20 宿 UNI-C 发起的释放相同，该信令流程如图 25 所示。

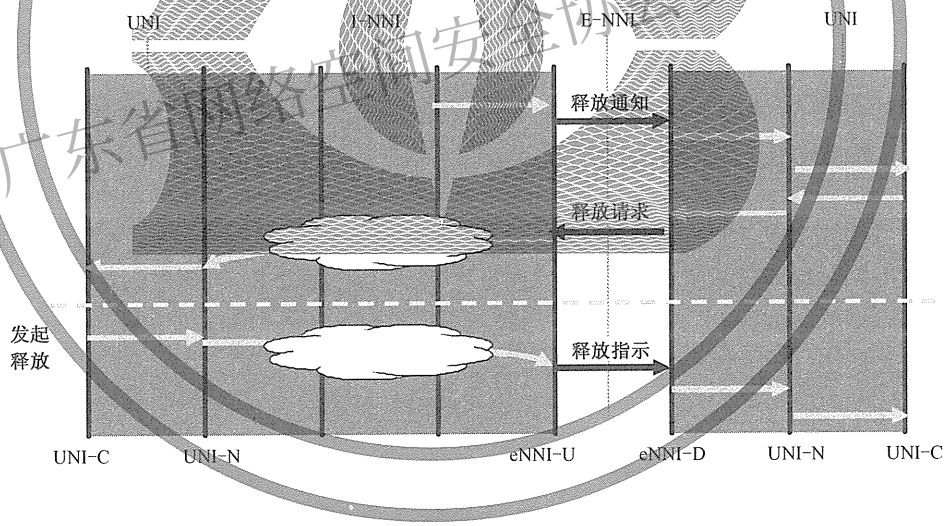


图 25　中间节点向下游方向发起的删除/释放

7.2.3　连接修改请求

OIF E-NNI 2.0 支持对已建呼叫和连接的修改。呼叫修改通过增加或删除已建呼叫中的连接来实现。连接修改通过修改已建连接的带宽来实现。UNI2.0 支持对 EVPL 业务 CE-VLAN 标识的修改，但这对于 E-NNI 是透明的。图 26 描述了连接修改的信令流程。

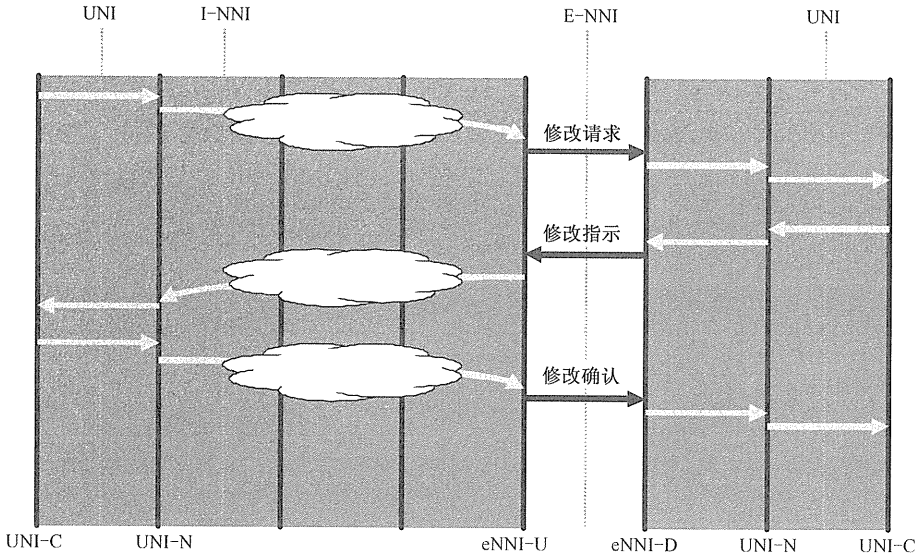


图 26　连接修改请求

7.3　异常情况操作

以下描述异常情况和故障处理时 E-NNI 接口的信令流程。需要注意的是，这里并不提供拒绝请求的原因（如资源冲突、策略决定等），而是描述拒绝请求以后的信令流程。

7.3.1　建立请求拒绝

一个请求可能在信令流程的不同阶段被拒绝。当决定拒绝一个请求后，应遵循一定的信令流程。当处理请求或指示消息时可能发生请求拒绝：

a）　在请求消息传送的过程中，其经过的任意节点都有可能拒绝该请求。决定拒绝的节点向源端发送一个释放消息，以便释放任何可能已经占用的资源。

b）　在指示消息传送的过程中，其经过的任意节点都有可能拒绝该请求。决定拒绝的节点发送一个包含错误消息的建立指示消息给源节点（NCC）；此外，向下游发送一个包含错误代码的确认消息，释放已经预留的资源。

图 27 表示建立请求被拒绝时的信令流程。

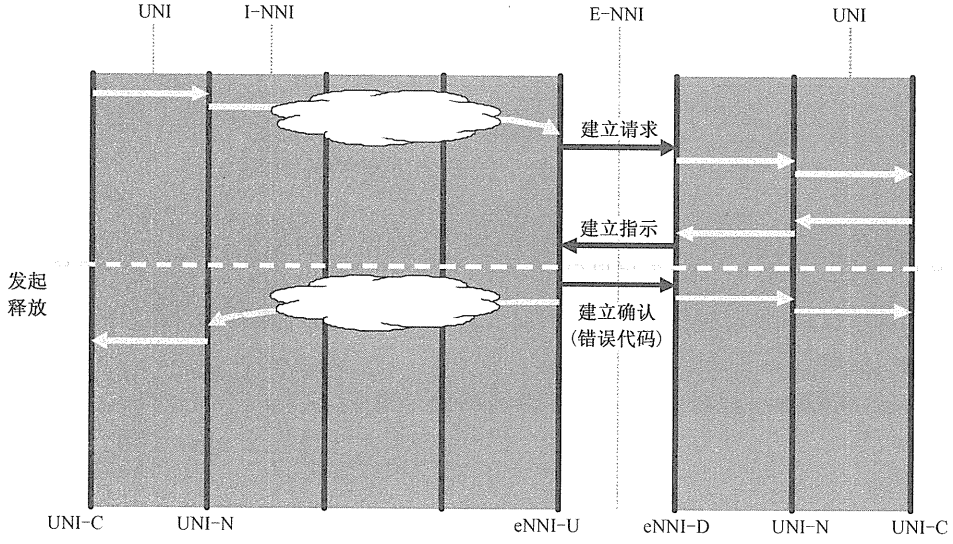


图 27　连接建立拒绝

7.3.2 释放请求拒绝

只有当释放请求无效时才会产生释放请求拒绝，如请求未认证。对于其他所有情况，释放请求应执行。如果在释放过程中产生了错误，导致连接没有释放，则应向外部控制器（如网管系统）发送错误消息。但是对该释放请求的响应应该是成功的，因为此时需要停止对用户的计费，并释放与该连接相关的资源。

因此，对于有效的释放请求，可以使用正常的释放请求信令流程；对于未经认证的释放请求，可以发送一个“未经认证释放请求”响应消息。对于因为错误而无法完成的释放请求，可以产生一个告警。

7.3.3 修改请求拒绝

呼叫修改包括添加一条新的连接或者修改现存的连接。对于现存连接的修改，采用“先建后拆”方式建立新的连接，然后删除原有的连接。在呼叫修改过程中如果出现故障，就会返回修改前的初始状态。

修改请求失败的原因包括无效的修改请求、分配额外带宽失败或者信令故障造成。E-NNI 节点拒绝修改请求后应发送连接建立指示消息（对添加一条新的连接引起的呼叫修改）或者连接修改指示（对修改现存连接引起的呼叫修改），同时携带错误编码，指示故障原因。

呼叫修改故障告警由源 UNI-N 或 UNI-C 发起。

7.3.4 信令通道失效

信令通道失效不应导致已经建立连接的释放。可以删除没有完成的建立请求（在失效时或是失效恢复后）。已经发起释放请求的连接应被释放（在失效时或是失效恢复后）。

7.3.5 控制平面节点失效

控制平面节点失效不应导致已经建立连接的释放。可以删除没有完成的建立请求（在失效时或是失效恢复后）。已经发起释放请求的连接应被释放（在失效时或是失效恢复后）。

7.3.6 传送平面资源失效

传送平面资源失效（例如链路或网元）可能导致已经建立连接的释放，这与连接类型和业务等级相关。例如当传送资源失效时，“无保护”连接可以被释放，而“保护”连接应根据业务等级进行恢复，并可以根据保护类型决定是否释放初始的连接。

8 RSVP-TE 扩展

8.1 RSVP-TE 概述

支持 E-NNI 信令的 RSVP-TE 扩展基于 IETF 中 RFC2205、RFC2961、RFC3209、RFC3473、RFC3474、RFC3476、RFC4328 和 RFC4606 规范的协议功能。关于 RSVP-TE 基本运行方式参见 IETF RFC3209 和 IETF RFC3474。以下描述与 E-NNI 相关的消息、对象和错误码。

在本部分中，当 eNNI-U（eNNI-D）发送一个 RSVP 消息时，应直接发送给其对端的 eNNI-D（eNNI-U），发送消息时将使用对等节点的 SCN 地址。节点应该使用 RSVP 消息的 IP 封装，任何 RSVP 消息中不应包含路由器告警的选项。在 IETF RFC2205 的 3.1.1 定义了 RSVP＜通用包头＞对象的格式，如表 18 所示。

表 18　E-NNIRSVP 消息的 IP 包头值

版本	4
头长	5
TOS	RFC 791 定义
总长	消息长度
标记	RFC 791 定义
分段偏移量	RFC 791 定义
TTL	≥1
协议	46
头校验	定义在 RFC 791
源地址	eNNI-U/eNNI-D 的 SC PC SCN IP 地址
目的地址	eNNI-D/eNNI-U 的 SC PC SCN IP 地址

RSVP 通用包头的标记域应设置为 1，表示支持消息绑定和消息刷新。因此 eNNI-U(eNNI-D)应支持处理来自邻居的绑定和刷新的消息，并且在发送消息时可以选择是否使用绑定或刷新消息。

8.2　消息和错误码

8.2.1　与抽象消息和错误码的对应关系

表 19 给出了 E-NNI 抽象消息和 RSVP-TE 消息之间的对应关系。

表 19　抽象消息和 RSVP-TE 消息之间的对应关系

抽象消息	RSVP-TE 消息
ConnectionSetupRequest	Path
ConnectionSetupIndication	Resv 或 PathErr(发生错误时)
ConnectionSetupConfirm	ResvConf 或 PathTear(发生错误时)
ConnectionReleaseRequest	Path 或 Resv(w/D&R bits) Path 或 Resv(w/A&R bits)(仅用于 UNI/OIF E-NNI 1.0)
ConnectionReleaseIndication	PathTear 或 PathErr(w/Path_State_Removed flag)
ConnectionQueryRequest	隐含
ConnectionQueryIndication	隐含
ConnectionNotification	Notify(w/D 比特设置)或 PathErr
ConnectionModifyRequest	Path
ConnectionModifyIndication	Resv，PathErr(发生错误时)
ConnectionModifyConfirm	ResvConf，PathTear(发生错误时)
Signaling Adjacency Maintenance	Hello

表 20 给出了 RSVP-TE 消息和 E-NNI 抽象消息之间的对应关系。

表 20　RSVP-TE 消息和抽象消息之间的对应关系

RSVP-TE 消息	抽象消息
Path	ConnectionSetupRequest ConnectionReleaseRequest ConnectionModifyRequest
Resv	ConnectionSetupIndication ConnectionReleaseRequest ConnectionModifyIndication
PathErr	ConnectionSetupIndication ConnectionReleaseIndication ConnectionNotification ConnectionModifyIndication
ResvConf	ConnectionSetupConfirm ConnectionModifyConfirm
PathTear	ConnectionSetupConfirm ConnectionReleaseIndication
Notify(w/D bit set)	ConnectionNotification
Hello	SignalingAdjacencyMaintenance

表 21 给出了 E-NNI 抽象错误码和 RSVP-TE 错误码及取值之间的对应关系。这些错误码由 IETF 的 RFC 2205、RFC 3209 和 RFC 1974 规定。

表 21　抽象错误码和 RSVP-TE 错误码及取值之间的对应关系

抽象错误	RSVP-TE 错误代码/错误值
呼叫方忙	ERROR_SPEC 24/5
被叫方忙	ERROR_SPEC 24/103
未认证发送方(策略错误)	ERROR_SPEC 2/100
未认证接收方(策略错误)	ERROR_SPEC 2/101
无效连接 ID	ERROR_SPEC 24/102
无效呼叫 ID	ERROR_SPEC 24/105
无效 SNP	ERROR_SPEC 24/6 or 24/11 or 24/12 or 24/14
不可用 SNP	ERROR_SPEC 24/6 or 24/11 or 24/12 or 24/14
无效 SNPP	ERROR_SPEC 24/104
不可用 SNPP	ERROR_SPEC 24/104
不可用方向性	ERROR_SPEC 24/6 or 24/11
无效 SPC SNP	ERROR_SPEC 24/106
无效路由	ERROR_SPEC 24/1,24/2,24/3,or 24/7
无效恢复	ERROR_SPEC 24/15 or 24/100

表 21（续）

抽象错误	RSVP-TE 错误代码/错误值
不可用恢复	ERROR_SPEC 24/15 or 24/100
不可用业务级别	ERROR_SPEC 24/101 or 2/{any}
影响业务缺陷	ERROR_SPEC 24/5,24/9,24/100,24/101 or 24/103
不影响业务缺陷	一般协议错误：一般 RSVP-TE 错误码/错误值

8.2.2 Hello 消息（消息类型＝20）

Hello 消息由 IETF RFC3209 规定，该消息用于建立邻接关系和通信失效检测。

Hello 消息的格式如下：

〈Hello message〉::=

〈Common Header〉

〈HELLO〉

〈RESTART_CAP〉

Hello 消息在邻接 E-NNI 信令实体之间周期性传送。重传间隔可以配置，缺省间隔时间为 5 s。

8.2.3 Path 消息（消息类型＝1）

Path 消息由 IETF RFC 2205 规定，并由 IETF 的 RFC2961、RFC3209 和 RFC3473 进行了扩展。

Path 消息的格式如下：

〈Path Message〉::=

〈Common Header〉

[[〈MESSAGE_ID_ACK〉|〈MESSAGE_ID_NACK〉]...]

〈MESSAGE_ID〉

〈SESSION〉〈RSVP_HOP〉

〈TIME_VALUES〉

〈GENERALIZED_LABEL_REQUEST〉

〈CALL_ID〉

[〈LABEL_SET〉...]

[〈SESSION_ATTRIBUTE〉]

[〈EXPLICIT_ROUTE〉]

〈NOTIFY_REQUEST〉

[〈ADMIN_STATUS〉]

〈Generalized UNI〉

[〈POLICY_DATA〉...]

〈sender descriptor〉

注：无论连接由 UNI 客户发起，增加或删除；还是作为 SPC 由入口域发起，或者携带源宿 TNA 的混合连接，都需要 GENERALIZED_UNI 对象及可选的 GENERALIZED_UNI 子对象，对于 SPC 需要使用该对象承载 SPC_LABEL。中间 NNI 节点不需要处理 GENERALIZED_UNI 对象，仅需要在连接的源和宿之间转发该对象。GENERALIZED_UNI 对象和 Sender Descriptor 对象的格式参见 OIF-UNI-02.0-RSVP 文档的 9.1.3 的“Path 消息”的规范。

例如对于 SONET/SDH 呼叫，格式如下：

〈sender descriptor〉::=
〈SENDER_TEMPLATE〉〈SENDER_TSPEC〉
[〈RECORD_ROUTE〉]
[〈UPSTREAM_LABEL〉]

建立单向连接的 Path 消息，不需要包含 UPSTREAM_LABEL 对象。

8.2.4 Resv 消息(消息类型=2)

Resv 消息由 RFC2205 规定，并由 IETF 的 RFC2961、RFC3209 和 RFC3473 进行了扩展。Resv 消息用于连接的建立以及呼叫/连接的修改。Resv 消息中可以包含 RESV_CONFIRM 对象用于请求 ResvConf 消息来确认连接的建立。只有将 Resv 消息中的 RESV_CONFIRM 对象删除，并在下一个 Resv 消息中插入新的 RESV_CONFIRM 对象，才会触发产生新的 ResvConf 消息。

E-NNI Resv 消息的格式如下：

〈Resv Message〉::=〈Common Header〉
[[〈MESSAGE_ID_ACK〉|〈MESSAGE_ID_NACK〉]...]
〈MESSAGE_ID〉
〈SESSION〉〈RSVP_HOP〉
〈TIME_VALUES〉
〈CALL_ID〉
〈NOTIFY_REQUEST〉
[〈ADMIN_STATUS〉]
[〈POLICY_DATA〉...]
[〈RESV_CONFIRM〉]
〈STYLE〉
〈FF flow descriptor〉|〈SE flow descriptor〉

〈FF flow descriptor〉::=
〈SONET/SDH_FLOWSPEC〉|〈G.709 FLOWSPEC〉|〈ETHFLOWSPEC〉
〈FILTER_SPEC〉
〈GENERALIZED_LABEL〉
[〈RECORD_ROUTE〉]

〈SE flow descriptor〉::=
〈SONET/SDH_FLOWSPEC〉|〈G.709 FLOWSPEC〉|〈ETHFLOWSPEC〉
〈FILTER_SPEC〉
〈GENERALIZED_LABEL〉
[〈RECORD_ROUTE〉]

8.2.5 ResvConf 消息(消息类型=7)

ResvConf 消息由 IETF RFC2205 规定，并由 IETF 的 RFC2961、RFC3209 和 RFC3473 进行了扩展。依据 IETF RFC3474，ResvConf 消息不包含呼叫名称(呼叫 ID)，因此与抽象连接建立确认消息不一致。但可以利用 SESSION 和流描述符实现 RescConf 与 Resv 状态的关联。

UNI-C 产生一个包含 RESV_CONFIRM 对象 ResvConf 消息来告知已收到 Resv 消息，ResvConf 消息由源 UNI-C 发往相关联的 UNI-N，以及宿 UNI-N 发往宿 UNI-C。ENNI 逐跳处理 ResvConf 消息，但 ResvConf 消息是端到端的，因此网络应能够在源 UNI-N 到宿 UNI-N 之间中继 ResvConf 消息。

ResvConf 消息的格式如下：

〈ResvConf message〉::=〈Common Header〉
[[〈MESSAGE_ID_ACK〉|〈MESSAGE_ID_NACK〉]...]
〈MESSAGE_ID〉
〈SESSION〉〈ERROR_SPEC〉
〈RESV_CONFIRM〉
〈STYLE〉
〈FLOW_SPEC〉|〈FILTER_SPEC〉

8.2.6 PathTear 消息(消息类型=5)

PathTear 消息由 IETF RFC2205 规定，并由 IETF 的 RFC2961、RFC3209 和 RFC3473 进行了扩展。ITU-TG.7713.2 做了进一步扩展以支持跨 E-NNI 的多种业务。

PathTear 消息格式如下：

〈Path Tear Message〉::=〈Common Header〉
[[〈MESSAGE_ID_ACK〉|〈MESSAGE_ID_NACK〉]...]
〈MESSAGE_ID〉
〈SESSION〉
〈CALL_ID〉
〈RSVP_HOP〉

8.2.7 PathErr 消息(消息类型=3)

PathErr 消息由 IETF RFC2205 规定，并由 IETF 的 RFC2961、RFC3209 和 RFC3473 进行了扩展。PathErr 消息用于报告错误或连接删除失败。

E-NNI PathErr 消息格式如下：

〈PathErr message〉::=〈Common Header〉
[[〈MESSAGE_ID_ACK〉|〈MESSAGE_ID_NACK〉]...]
〈MESSAGE_ID〉
〈SESSION〉
〈CALL_ID〉
〈ERROR_SPEC〉
[〈ACCEPTABLE_LABEL_SET〉]
[〈POLICY_DATA〉…]
〈sender descriptor〉

8.2.8 Notify 消息(消息类型=21)

Notify 消息由 IETF RFC3473 规定。

Notify 消息格式如下：

〈Notify message〉::=〈Common Header〉
[[〈MESSAGE_ID_ACK〉|〈MESSAGE_ID_NACK〉]...]
〈MESSAGE_ID〉
〈ERROR_SPEC〉
〈notify session lists〉/*1 或多 notify sessions*/

〈notify session lists〉::=[〈notify session lists〉]
〈upstream notify session〉|〈downstream notifysession〉
〈upstream notify session〉::=〈SESSION〉〈CALL_ID〉〈ADMIN_STATUS〉
[〈POLICY_DATA〉...]〈sender descriptor〉
〈downstream notify session〉::=〈SESSION〉〈CALL_ID〉〈ADMIN_STATUS〉
[〈POLICY_DATA〉...]〈flow descriptor list〉
〈flow descriptor list〉::=
〈FF flow descriptor〉|〈SE flow descriptor〉/*1 或多个 flow descriptor*/

在下游 notify session 中加入了 ADMIN_STATUS 对象，适用于以下两种情形：

a) UNI/ENNI 1.0 节点允许从上游或下游发起网络删除。如果 ENNI 2.0 接口的上游为 UNI/ENNI 1.0 节点，ENNI 2.0 接口可以接收来自 UNI/ENNI 1.0 节点的下游网络删除请求。

b) 网络 I-NNI 域发起下游正常删除。

注：在 OIF E-NNI 2.0 规定，为支持中间节点发起的删除，要求 Notify 消息中的 ADMIN_STATUS 为必选。

在 IETF 模型中允许向任何接收者发送 Notify 消息。在 OIF E-NNI 2.0 中规定仅在中间节点发起正常删除时使用 Notify 消息，因此 Notify 消息只能向与其相邻的上游或下游节点发送。虽然 Notify 消息逐跳处理，但也是一类端到端的消息，因此网络应转发来完成中间节点的删除消息流。

8.2.9 Srefresh 消息

该消息在 IETF RFC2961 中规定，ITU-T G.7713.2 做了进一步扩展以支持跨 E-NNI 的多种业务。

功能：

刷新 RSVP-TE 状态而不需要传输 Path 或 Resv 消息，减少维持呼叫和连接状态所需要传送和处理的信息。

格式：

〈Srefresh Message〉::=
〈Common Header〉
[〈INTEGRITY〉]
[[〈MESSAGE_ID_ACK〉|〈MESSAGE_ID_NACK〉]...]
[〈MESSAGE_ID〉]
〈srefresh list〉|〈source srefresh list〉

〈srefresh list〉::=
〈MESSAGE_IDLIST〉|〈MESSAGE_ID MCAST_LIST〉
[〈srefresh list〉]

〈source srefresh list〉::=
〈MESSAGE_ID SRC_LIST〉
[〈source srefresh list〉]

8.2.10 Ack 消息

该消息在 IETF RFC2961 中规定，ITU-T G.7713.2 做了进一步扩展以支持跨 E-NNI 的多种业务。

功能：

提供对已发送消息的确认。确认功能可以直接由 Ack 消息提供，也可以由其他消息间接提供，即当发送的消息有响应消息时(如 Resv 是 Path 的响应消息)。在后一种情况时，通过在响应消息中包含 MESSAGE_ID_ACK 对象提供 ACK 功能。

格式：

〈ACK Message〉::=

〈Common Header〉

[〈INTEGRITY〉]

〈MESSAGE_ID_ACK〉|〈MESSAGE_ID_NACK〉

[[〈MESSAGE_ID_ACK〉|〈MESSAGE_ID_NACK〉]...]

8.3 属性和对象

8.3.1 与抽象属性和对象的对应关系

表 22 给出了 E-NNI 接口抽象属性和 RSVP-TE 对象之间的对应关系。

表 22 抽象属性和 RSVP-TE 对象之间的对应关系

抽象属性	抽象属性名称	RSVP-TE 对象
Source TNA Name	源 TNA	GENERALIZED_UNI/Source_TNA
Destination TNA Name	目的 TNA	GENERALIZED_UNI/Destination_TNA
DEST SNP ID	DEST SNP ID	GENERALIZED_UNI/SPC_LABEL
Initiating NCC PC ID	发起 NCC PC ID	SENDER_TEMPLATE
Terminating NCC PC ID	终结 NCC PC ID	SESSION
Connection Name	连接名称	SESSION+SENDER_TEMPLATE
Connection Name	连接名称	SESSION+SENDER_TEMPLATE
Call Name	呼叫名称	CALL_ID
SNP ID	SNP ID	SENDER_TSPEC, RSVP_HOP, LABEL, GENERALIZED_UNI/EGRESS_LABEL
SNPP ID	SNPP ID	Source/destination TNA,RSVP_HOP,LABEL_SET
Directionality	方向性	UPSTREAM_LABEL 隐含
Explicit Route	显式路由	EXPLICIT_ROUTE,RECORD_ROUTE
Recovery	恢复	PROTECTION
Service Level	业务等级	GENERALIZED_UNI/DIVERSITY, GENERALIZED_UNI/SERVICE_LEVEL,POLICY_DATA, SESSION_ATTRIBUTE
Contract ID	合同 ID	POLICY_DATA
Encoding Type	编码类型	GENERALIZED_LABEL_REQUEST/LSP_ENC_TYPE

表 22（续）

抽象属性	抽象属性名称	RSVP-TE 对象
Switching Type	交换类型	GENERALIZED_LABEL_REQUEST/SWITCHING_TYPE
SONET/SDH, OTN or Ethernet Traffic Parameters	SONET/SDH，OTN 或以太网流量参数	SONET/SDH_SENDER_TSPEC， SONET/SDH_FLOWSPEC G.709 TSPEC，G.709FLOWSPEC ETHERNET SENDER_TSPEC，ETHERNETFLOWSPEC
Generalized Payload Identifier	通用净荷标识符	GENERALIZED_LABEL_REQUEST/G-PID
Connection Status	连接状态	ADMIN_STATUS

表 23 列出了支持 E-NNI 接口信令功能所需要的 RSVP-TE 对象，以及与 E-NNI 相关的编码点。需要注意的是，该表只列出了与 E-NNI 相关的编码点，而不是所有的编码点。另外，除了在这里特别给出的对象格式以外，其他的对象格式参见相关引用文件。

表 23 RSVP-TEE-NNI 对象汇总

RSVP-TE 对象	Class-Num/C-type[/Type/[Sub-type]]	参考
ACCEPTABLE_LABEL_SET	130/{同 label_set}[a]	IETFRFC3473，IETF RFC4328
ADMIN_STATUS[b]	196/1	IETF RFC3473
CALL_ID	230/〈1,2〉	IETF RFC3474
ERROR_SPEC	6/3/{同 RSVP_HOP}	IETF RFC2205，IETF RFC3209，IETF RFC3471，IETF RFC3473 参考 8.3.2
EXPLICIT_ROUTE	20/1/〈3,4〉[c]	IETF RFC3209，IETF RFC3473，IETF RFC3477 参考 8.3.3
FILTER_SPEC	10/{同 SENDER_TEMPLATE}	IETF RFC2205，IETF RFC3209，IETF RFC3473
SONET/SDH_FLOWSPEC	9/4	IETF RFC4606
G.709 FLOWSPEC	9/5	IETF RFC4328
ETHERNET FLOWSPEC	9/6	ETH_PARAM
GENERALIZED_UNI/DESTINATION_TNA	229/1/2/〈1,2,3〉	OIF-UNI-02.0
GENERALIZED_UNI/DIVERSITY	229/1/3/1	OIF-UNI-02.0
GENERALIZED_UNI/EGRESS_LABEL	229/1/4/1	OIF-UNI-02.0
GENERALIZED_UNI/SERVICE_LEVEL	229/1/5/1	OIF-UNI-02.0

表 23（续）

RSVP-TE 对象	Class-Num/C-type[/Type/[Sub-type]]	参考
GENERALIZED_UNI/SOURCE_TNA	229/1/1/〈1,2,3〉	OIF-UNI-02.0
GENERALIZED_UNI/SPC_LABEL[d]	229/4/2	IETF RFC3474
HELLO_REQUEST/HELLO_ACK	22/〈1,2〉	IETF RFC3209,IETF RFC3473
RSVP_LABEL (GENERALIZED_LABEL)[e]	16/2	IETF RFC3473,IETF RFC4328
GENERALIZED_LABEL_REQUEST	19/4	IETF RFC3473,IETF RFC4328
LABEL_SET[f]	36/1	IETF RFC3473,IETF RFC4328
MESSAGE_ID	23/1	IETF RFC2961
MESSAGE_ID_ACK/MESSAGE_ID_NACK	24/〈1,2〉[g]	IETF RFC2961
MESSAGE_ID_LIST	25/1	IETF RFC2961
NOTIFY_REQUEST	195/1	IETF RFC3473
POLICY_DATA	14/1	IETF RFC2205
PROTECTION	37/1	IETF RFC3473
RECORD_ROUTE	21/{同 ERO}	IETF RFC3209,IETF RFC3473,IETF RFC3477
RECOVERY_LABEL	34/{同 RSVP_LABEL}	IETF RFC3473,IETF RFC4328
RESTART_CAP	131/1	IETF RFC3473
RESV_CONFIRM	15/1	IETF RFC2205 参考 10.3.5
RSVP_HOP	3/3/〈3,4,5〉[h]	IETF RFC2205,IETF RFC3471,IETF RFC3473 参考 10.3.6
SENDER_TEMPLATE	11/7	IETF RFC2205,IETF RFC3209,IETF RFC3473 参考 10.3.7
SONET/SDH_TSPEC	12/4	IETF RFC4606
G.709 TSPEC	12/5	IETF RFC4328
ETHERNET TSPEC	12/6	ETH_PARAM
SESSION	1/15	IETF RFC2205,IETF RFC3209, 参考 10.3.8
SESSION_ATTRIBUTE	207/〈1,7〉	IETF RFC3209

表 23（续）

RSVP-TE 对象	Class-Num/C-type[/Type/[Sub-type]]	参考
STYLE[i]	8/1	IETF RFC2205
SUGGESTED_LABEL	129/{同 RSVP_LABEL}	IETF RFC3473
TIME_VALUES[j]	5/1	IETF RFC2205
UPSTREAM_LABEL	35/{同 RSVP LABEL}	IETF RFC3473,IETF RFC4328

[a] { }里面的文字是说明。

[b] ADMIN_STATUS 缺失与收到值均设为 0 的对象等效。

[c] 〈…〉表示该对象包含的不同的 C-type 或 sub-type。

[d] SPC_Label 子对象中的端口标识符在目的端 UNI-N 中指定的逻辑端口标识符，EGRESS_LABEL 子对象中的端口标识符是在目的端 UNI-C 中指定的逻辑端口标识符。

[e] LABEL 的格式依赖于 LABEL_REQUEST 对象定义的信号类型。

[f] 一个 LABEL_SET 对象包含一个“子信道(sub-channel)”的列表，每个子信道表示一个 label。子信道的类型从标记类型字段获得，每个 LABEL_SET 对象只能包含一种标记类型。对标记的解释依赖于链路的类型，因此每个子信道不需要自己的信头。

[g] MESSAGE_ID_NACK 是 MESSAGE_ID_ACK 的一个 sub-type。

[h] 为支持 OIF E-NNI 1.0 后向兼容性，RSVP_HOP 应支持类型 4 和 5 但本应产生该值。

[i] OIF E-NNI 1.0 只使用“fixed filter”。OIF E-NNI 2.0 同时使用“fixed filter”和“shared explict”方式。

[j] 该值应与 Srefresh 间隔相协调，以确保对状态信息的刷新。

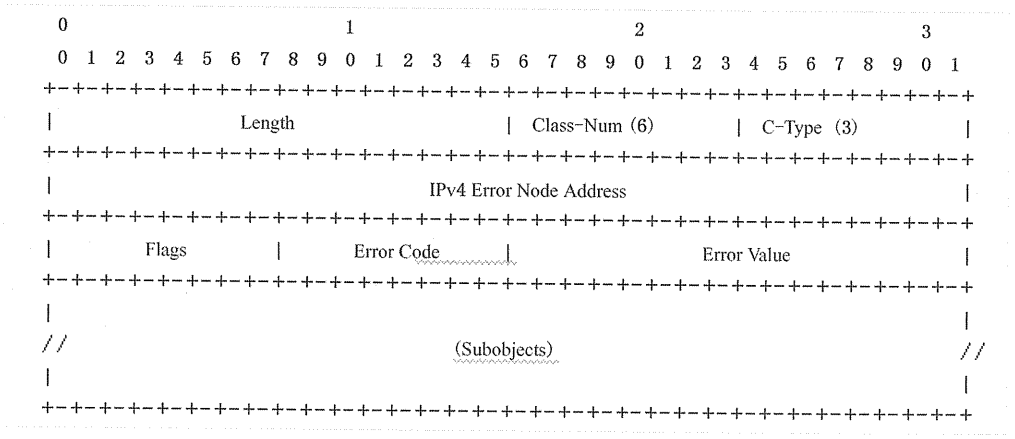
8.3.2 ERROR_SPEC

IPv4 IF_ID_ERROR_SPEC(Class＝6,C-Type＝1)和 IPv6 IF_ID_ERROR_SPEC(Class＝6,C-Type＝2(OIF E-NNI 1.0)/4(OIF E-NNI 2.0))在 IETF RFC3473 中定义。应支持 IPv4 IF_ID_ERROR_SPEC。在 E-NNI 信令中，报告错误的节点地址应设置为该节点的 SC PC ID(eNNI-U 或 eNNI-D 标识符)。在正常删除时，应该设置 Path_State_Remove 标志，并使用 Error code 0(确认)和 Error Value 0。

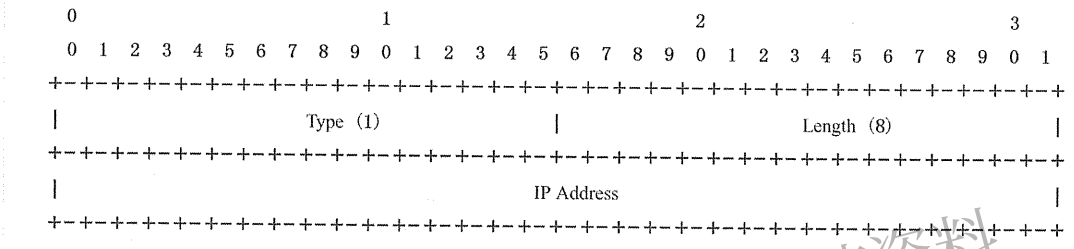
当 Error code 不为 0 时，依据 RFC3471，IPv4 IF_ID_ERROR_SPEC 应包含一个 IPv4(类型为 1)或 IPv4 IF_INDEX(类型为 3)的子对象，其中的内容主要用于路由广播，用于广播的标识符也是在 E-NNI 2.0 中规定广播路由和信令的 ERO 对象。IF_INDEX 子对象与 ERO 中的跳数相关联。除了类型 1 和 3 的子对象外，还可依据 RFC4920 使用更多的对象。

类型为 1 的对象报告整个传输节点的故障。IPv4 地址就是故障节点的 ID。类型为 3 的子对象还携带了一个接口 ID，指定故障节点的某个链路故障。

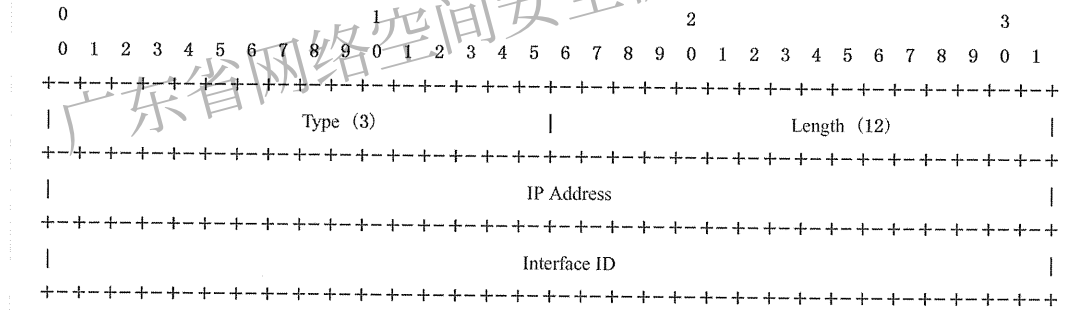
依据 RFC3473，IF_ID ERROR_SPEC，Class 为 6，C-Type 为 3 的子对象描述如下：



依据 RFC3473,IF_IDERROR_SPEC,类型为 1 的子对象描述如下:

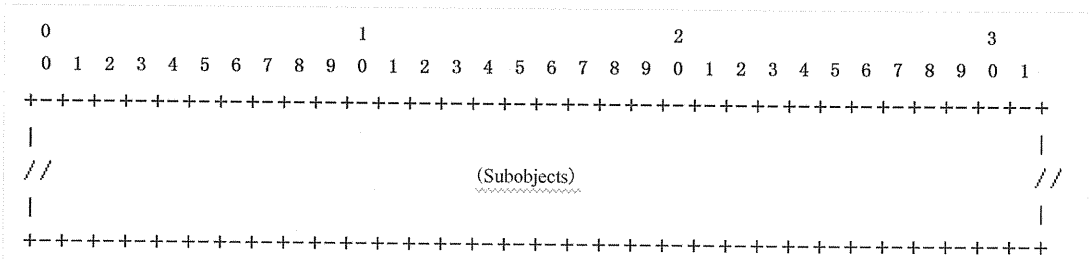


依据 RFC3473,IF_IDERROR_SPEC,类型为 3 的子对象描述如下:



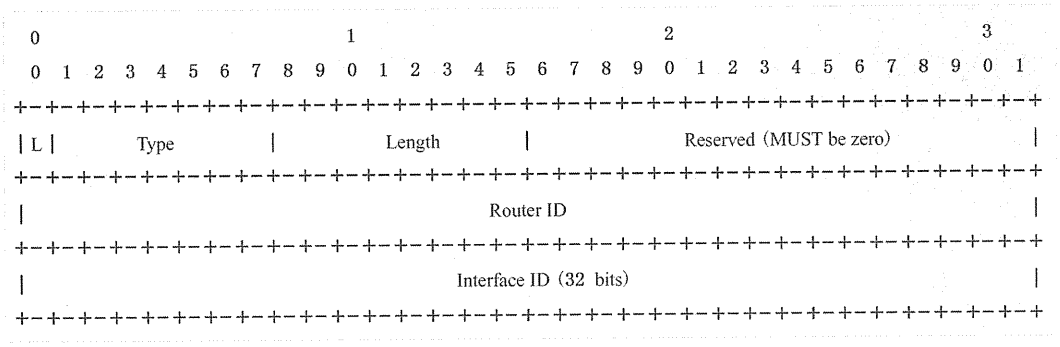
8.3.3 EXPLICIT_ROUTE

EXPLICIT_ROUTE 对象(Class=20,C_Type=1)格式如下,参见 IETFRFC3209:



ERO 中的子对象应用来选择连接通过的传送链路。E-NNI 应支持子对象 IETF RFC3473 中的 Type 3(Label)和 IETF RFC3477 中的 Type4(Unnumbered interface ID)。对子对象 Type1,2,32 的支持需要进一步研究。OIF E-NNI2.0 把 ASON SNPPid 映射为〈RA ID,NodeID,IfIndex〉,因此 OIF E-

NNI2.0 不需要支持 ERO hop 类型 1,2,32。



在上面描述的 Type 4(Unnumbered interface ID)的子对象中,所有 ERO 子对象中的 L 比特都不设置,因为在分级路由中认为经过 E-NNI 的 ERO 属于高一等级的路由域。路由器 ID 应设置为传送节点 ID,即是一传送平面名称。接口 ID 是由传送节点分配给链路的标识符,可以表示一条单一链路或绑定链路。节点 ID 和接口 ID 都是指上游节点。

节点 ID 可以使用 IP 地址或名称,对名称的支持需要进一步研究。图 28 给出了 Path 消息中的 ERO 具体内容的示例。

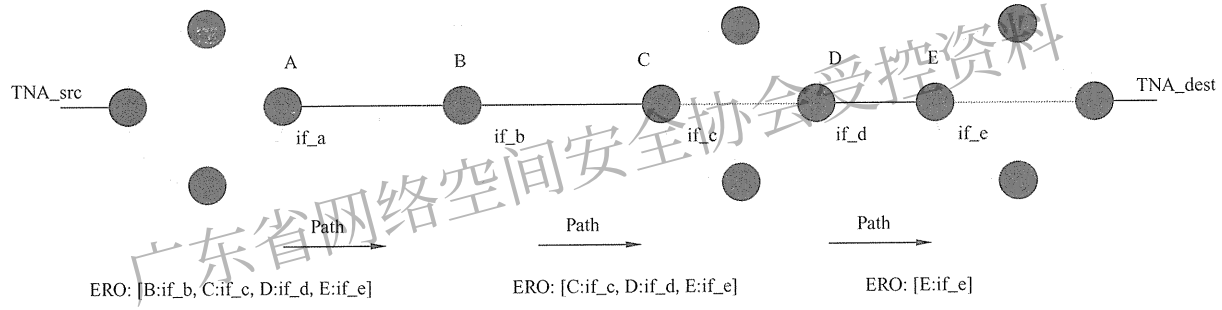


图 28 ERO 示例

8.3.4 RECORD_ROUTE

记录路由对象由 IETF RFC3209 定义。对象包含了一系列长度可变的子对象。IETF 为 RRO 定义的两个子对象(见表 24),可用于 E-NNI 2.0 信令。

表 24 RRO 子对象

子对象	类型值	内容
Label	0x03	RFC3209 规定了应用,并由 RFC3473 作进一步扩展
Unnumbered IPv4	0x04	RFC3477 规定了应用,NodeID 用于 RouterID

所有其他的 RRO 子对象都不能用于 OIF E-NNI2.0 信令。

对 RRO 对象的语法设计稍作改变即可将其用于显式路由对象。因此,RRO 对象可以用于识别上下游链路终端。

8.3.5 Generalized UNI 对象

该对象用来确定呼叫方和被叫方标识符,以及其他的呼叫属性。可以由用户通过控制平面请求

(UNI 信令),或由管理平面在网络侧发起请求。被叫方呼叫控制器使用该呼叫属性,目的网络呼叫控制器也可以使用该呼叫属性对呼叫进行验证。E-NNI 信令只对该对象进行传送,而不能改变其内容。当在运营商之间使用 E-NNI 信令时,E-NNI 信令可以对大部分子对象进行传送,而对剩余的一些子对象重新赋值。例如,当从一个域进入另一个域时,改变服务级别的值。

GENERALIZED_UNI 的内容是一系列可变长度的子对象。考虑未来的兼容性,Type 和 Sub-Type 的值根据 RFC2205 中 RSVP 对象的规定来赋值。对于 Type 和 Sub-Type 值的处理分别与对 RSVP 的 Class-Num 和 C-Type 的处理一致。如果 Type 或 SubType 值不可识别而需发送一个错误消息时,节点应该使用 error code"未知 Class-Number"或"未知 Class-Number,未知的 C-Type Class-Number",并将 GENERALIZED_UNI 对象的 Class-Number 和 C-Type 设为相应的值。

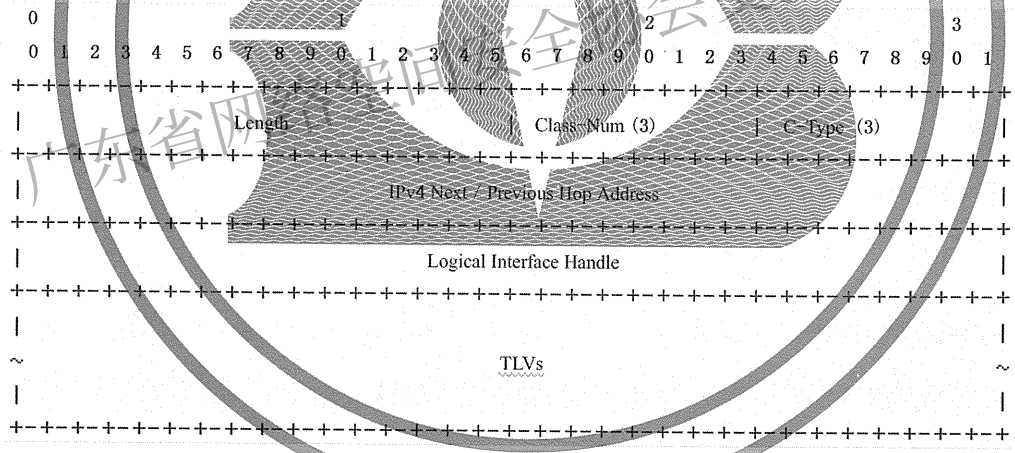
注:E-NNI2.0 规定,当 label 值不需要时,EGRESS_LABEL(Type 4 子对象)或者 SPC_LABEL 的值将设为 0。特定的出口标签类型不指定是否在出口处激活信令协议。

8.3.6 RESV_CONFIRM

IPv4 RESV_CONFIRM(Class=15,C-Type=1)和 IPv6 RESV_CONFIRM(Class=15,C-Type=2)在 IETF RFC2205 中定义,应支持 IPv4 RESV_CONFIRM。在 E-NNI 信令中,接收方地址应设置为下游 SC PC ID(即 eNNI-D 标识符)。

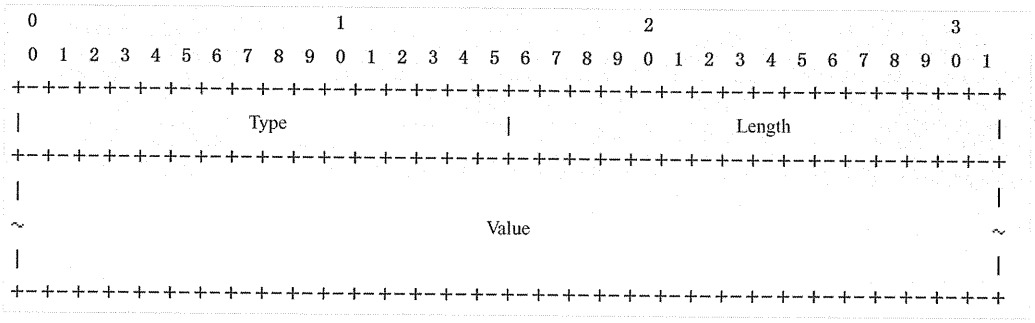
8.3.7 RSVP_HOP

IF_ID RSVP_HOP 对象的格式如下,参见 IETF RFC3473:



IPv4 Next/Previous Hop Address 应该设置为与该链路对应的 SC PC ID(eNNI-U 或 eNNI-D 标识符)。在 Path 消息中,该对象包含上游域的 SC PC ID,在 Resv 消息中包含下游域 SC PC ID。

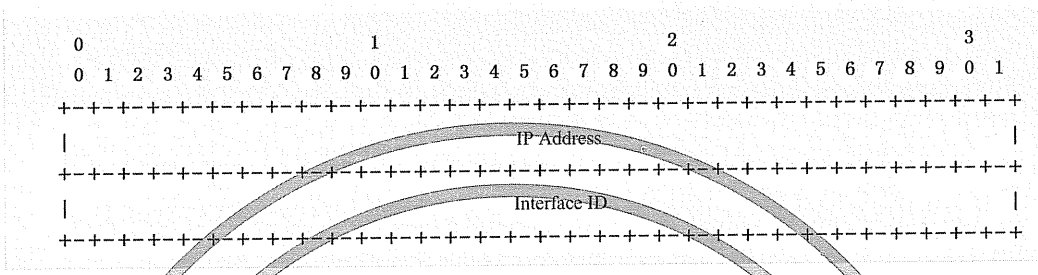
TLV 的格式如下,参见 IETF RFC 3471:



应支持 Type 3:

Type	Length	Format	Description
3	12	见下	If_INDEX (Interface Index)

在传送网中的应用不支持 Type 1 和 2TLV，需要进一步研究。Type3 的 Value 字段格式如下：



应使用 IF_ID_RSVP_HOP 对象来选择用于分配连接资源的传送链路，IP 地址字段应设置为与该链路对应的节点 ID。

a) 对于单向 LSP，应指出下游数据链路；

b) 对于双向 LSP，应指出上下游的数据链路。特别是当双向 LSP 经过绑定链路时，需要分别指出上游数据链路和下游数据链路。当需要两个 RSVP_HOP 子对象时，类型为 3 的子对象按如下方法使用：

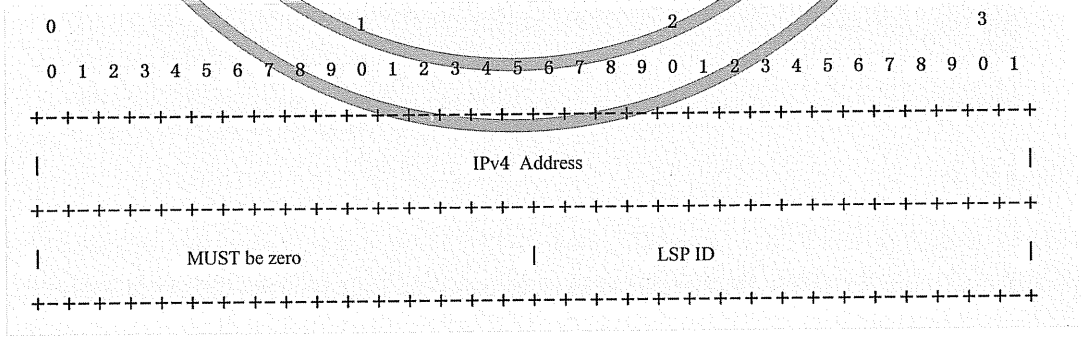
1) 第一个子对象应用于表示下游数据链路；

2) 第二个子对象应用于表示上游数据链路。

接口 ID 表示单一链路或一绑定链路的组成链路。eNNI-U 和 eNNI-D 维护接口 ID 的映射，即本地和远端的接口 ID 可以不一样。对非 IP 地址格式的节点 ID 的支持需要进一步研究。

8.3.8 SENDER_TEMPLATE

LSP_TUNNEL_IPv4 对象(Class＝11，C-Type＝7)在 IETF RFC3209 中定义。应支持 LSP_TUNNEL_IPv4，且应在 E-NNI 的 SENDER_TEMPLATE 和 FILTER_SPEC 中使用。LSP_TUNNEL_IPv4 对象的格式如下，参见 IETF RFC3209：



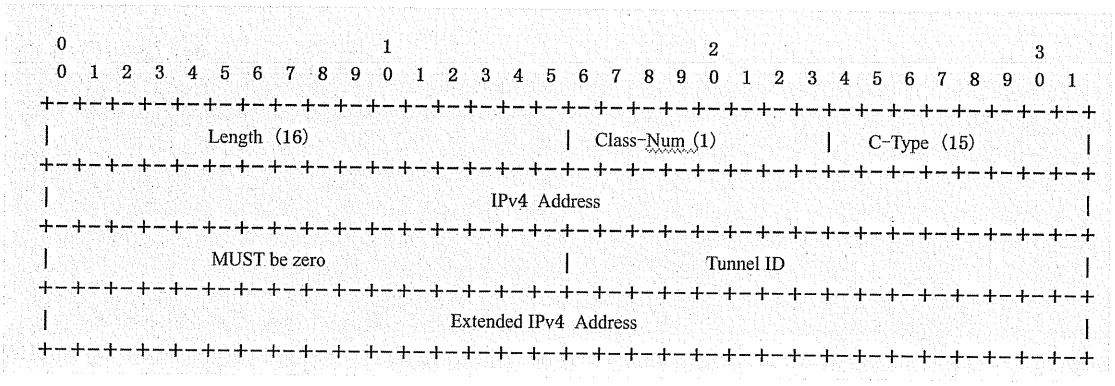
IPv4 Address：应设置为上游 SC PC ID(eNNI-U)。

LSP ID：用于 SENDER_TEMPLATE 和 FILTER_SPEC 的 16-bit 标识符。

LSP_TUNNEL_IPv4_SENDER_TEMPLATE 和 E-NNI_IPv4_SESSION 对象的组合应能够在 E-NNI 唯一标识一条连接，并在连接期间保持不变。只有当应用 make-before-break 步骤修改连接带宽时，SP_TUNNEL_IPv4_SENDER_TEMPLATE_LSPID 才会改变。一个无法识别的连接 ID 将产生一个错误消息，错误码＝“Routing Problem：Invalid/Unknown ConnectionID”。

8.3.9 SESSION

E-NNI_IPv4 SESSION 对象(Class＝1,C-Type＝15)和 E-NNI_IPv6 SESSION 对象(C-Type＝16)在 ITU-TG.7713.2 中定义。应支持 E-NNI_IPv4 SESSION,其格式如下:



IPv4 Address:应设置为下游信令控制 ID(eNNI-D)。

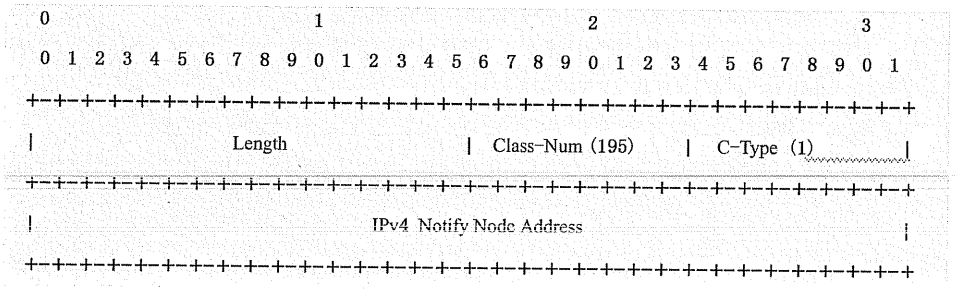
Tunnel ID:由 Path 消息的发送方分配的一个 16-bit 标识符,在连接期间保持不变。

Extended IPv4 address:应设置为上游信令控制 ID(eNNI-U)。

LSP_TUNNEL_IPv4_SENDER_TEMPLATE 和 E-NNI_IPv4_SESSION 对象的组合应能够在 E-NNI 唯一标识一条连接,并在连接期间保持不变。只有当应用 make-before-break 步骤修改连接带宽时,SP_TUNNEL_IPv4_SENDER_TEMPLATE_LSPID 才会改变。一个无法识别的连接 ID 将产生一个错误消息,错误码＝“Routing Problem:Invalid/Unknown ConnectionID”。

8.3.10 NOTIFY_REQUEST

Notify Node Address 域包含了产生该对象的 E-NNI 节点的 SC PC ID。其格式如下:



8.3.11 CALL ID

CALL_ID 的格式由 IETF RFC3474 规定,并在 OIF-UNI-02.0-RSVP 中有进一步的解释。为了保证穿过多个域后 CALL ID 的唯一性,源 LSR 的地址应设为产生 Call ID 的源节点的 SCPCID。

8.4 RSVP-TE 信令流程

8.4.1 连接建立

用于 E-NNI 的 RSVP-TE 协议应遵循第 7 章描述的 E-NNI 信令流程。图 29 描述了在 E-NNI 接口之间建立连接的信令流程。收到来自网络的连接请求时,eNNI-U 向 eNNI-D 发送 Path 消息。eNNI-D 继续向下游发送建立请求。eNNI-D 收到网络的连接建立回应时,向 eNNI-U 发送 Resv 消息。

出口节点可以通过连接建立指示请求一个可选的确认消息。如果 eNNI-U 收到了网络连接的确认，则向 eNNI-D 回送一个 ResvConf 消息。eNNI-D 继续向出口转发确认信息。

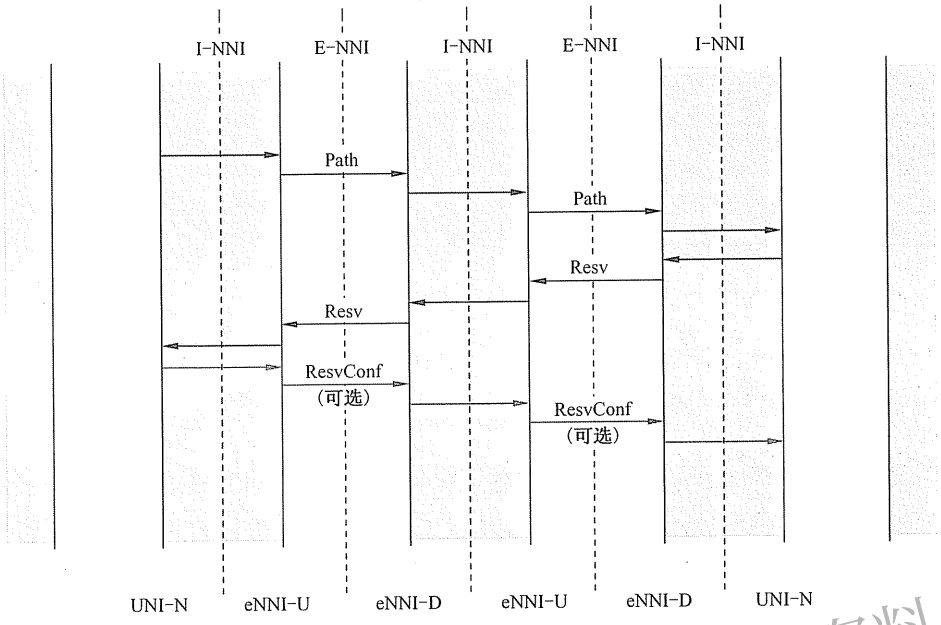


图 29　使用 RSVP-TE 建立基本连接

连接建立可能由于策略控制、资源不足或目的节点不可达等原因建立失败。eNNI-D 建立连接失败或收到连接建立失败的指示时，应删除自己的 Path 状态并向 eNNI-U 发送删除 Path 状态的 PathErr 消息。eNNI-U 向入口节点转发连接建立失败指示。图 30 描述了连接建立失败时的信令流程。

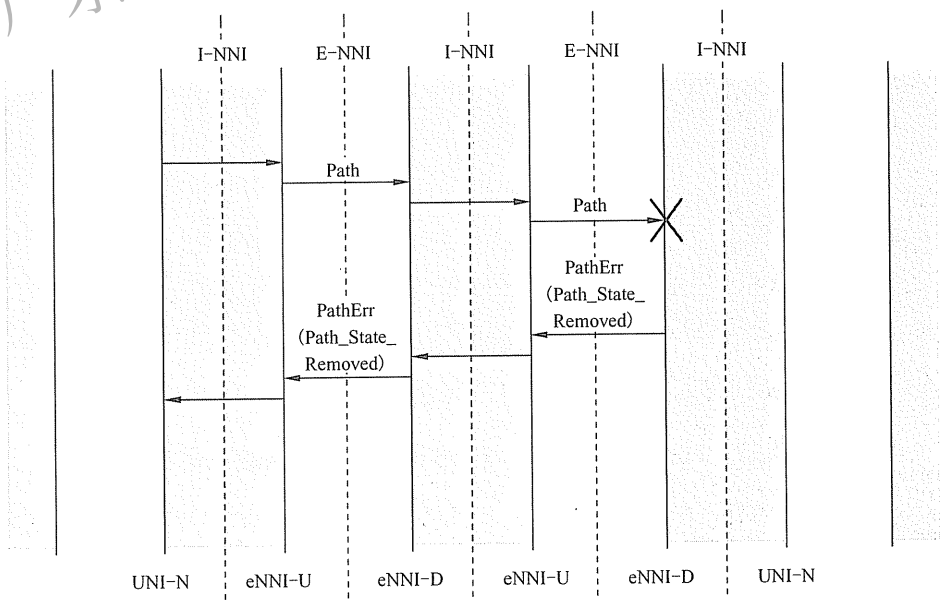


图 30　连接建立失败

连接建立时也可能在指示(Resv)过程中失败，见图 31。例如，可能由于两个连接同时建立时标签分配失败。这种情况下，eNNI 删除自己的 Path State 并向上游方向发送删除 Path State 的 PathErr 消息，向下游方向发送删除 Path State 的 PathTear 消息。

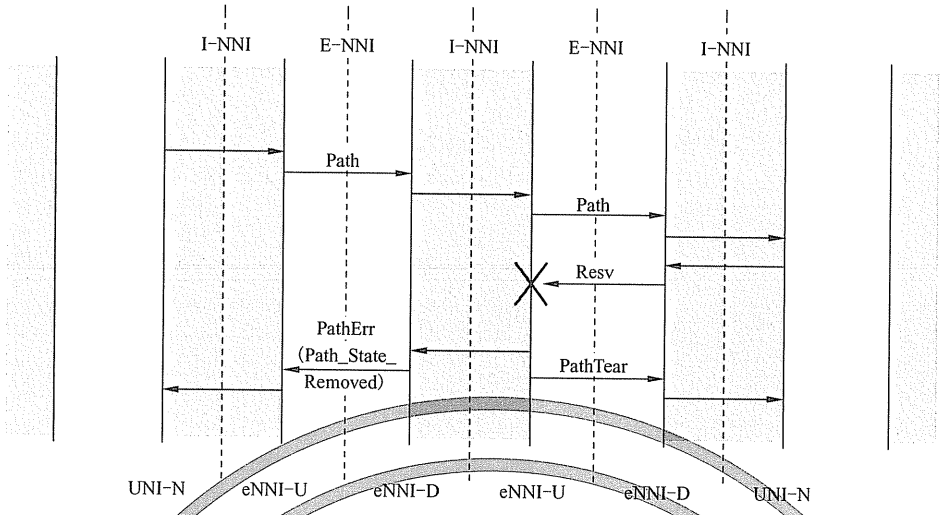


图 31 指示过程中连接建立失败

如果 PathErr 消息中 Path_State_Removed 标记没有设置，源 UNI-C 或者 UNI-N 将会显式删除连接，见图 32。当删除请求到达 eNNI 时，eNNI-U 向 eNNI-D 发送 PathTear 消息。如果 PathTear 消息中携带了 MESSAGE_ID，并且 Ack_Desired flag 已设置，收到该消息的节点如果不满足任何 Path State 则接收到该消息后抛弃。

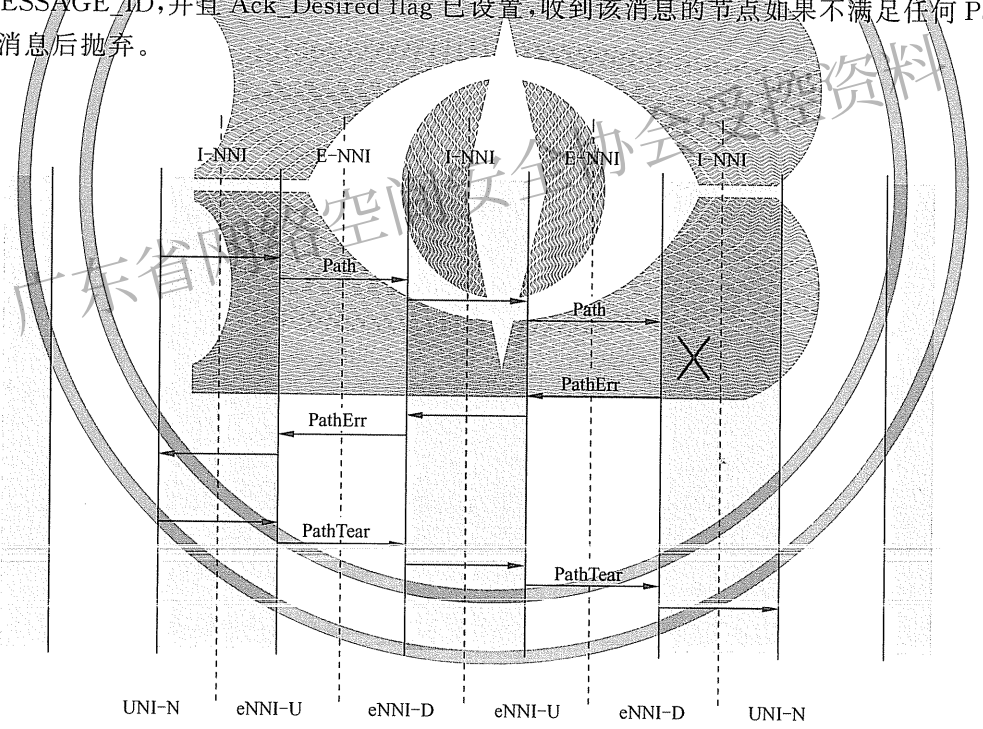


图 32 Path_State_Removed 标记未设置时的建立失败

8.4.2 连接修改

8.4.2.1 连接修改方式

连接的修改有两种方式：

a) 增加或删除已有呼叫的一个连接，这种情况下无需连接修改流程，而是发送一个新的连接建立或删除请求。增加的连接可以使用与已有连接相同或不同的路由。使用不同路由的情况仅适用于对已有呼叫增加连接。

b) 对呼叫的已有连接进行修改。

8.4.2.2 对呼叫增加或删除连接的修改

对已存在的呼叫增加或删除连接可以用于修改呼叫的带宽。每条连接都有自己的 RSVP 状态，因此增减连接失败不会对呼叫的其他的连接产生影响。

图 33 显示了对已存在的呼叫成功增加一条连接。呼叫与连接同时建立，接下来如果源想要修改该呼叫如增加另外一条连接，可以产生一个新的 Path 消息，携带与已有连接相同的 CALL_ID。该 CALL_ID 用来指示在特定的呼叫中增加一条新的连接，CALL_ID 也用于将 E-NNI 节点间的不同连接相关联。当收到呼叫修改请求时，eNNI-U 会产生一个具有相同 CALL_ID 的 Path 消息。该 Path 消息包含一个不同的连接标识符(TUNNEL_ID)和一个新的 MESSAGE_ID。如果 E-NNI 节点没有对应该 CALL_ID 的呼叫状态，那么 E-NNI 则会将它作为一个新的呼叫而不是呼叫修改来处理。

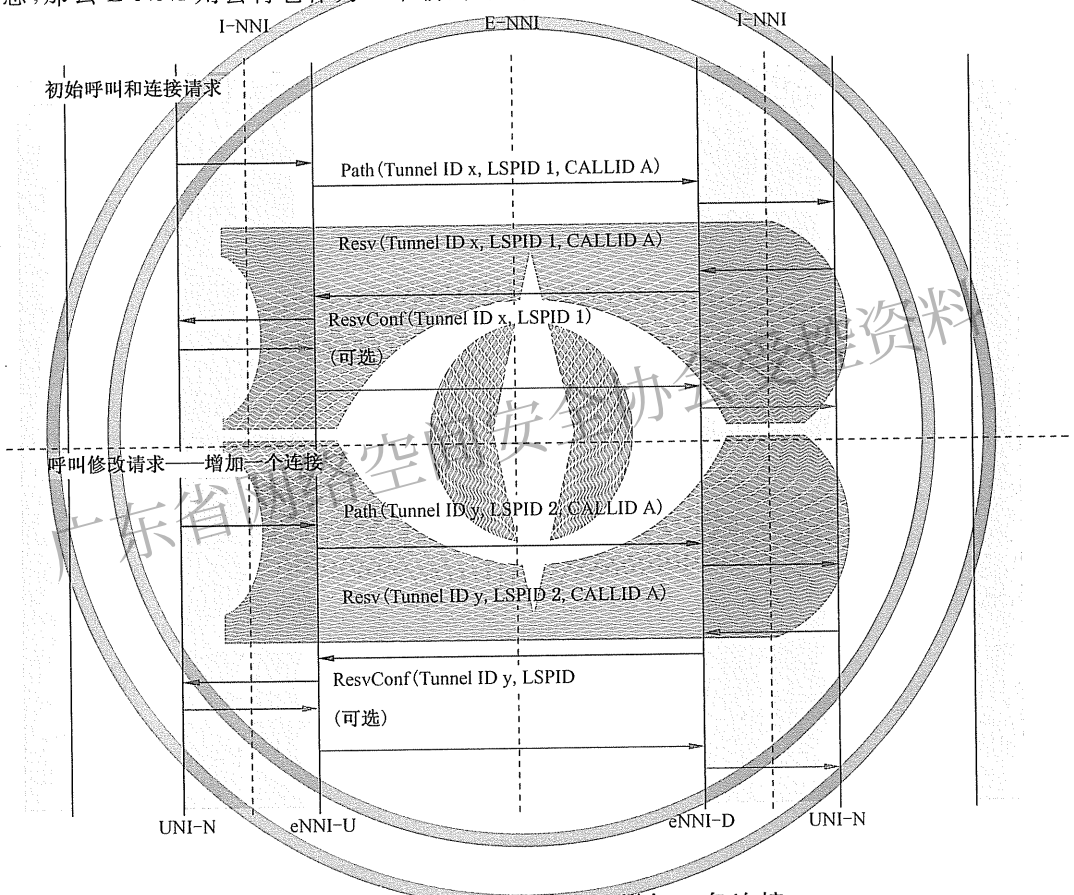


图 33 呼叫修改成功——增加一条连接

连接删除消息在 8.4.3 中描述。呼叫中的连接可以由源、宿，或者网络中的 E-NNI 节点删除。连接删除的过程相互独立。不支持没有连接的呼叫。呼叫的最后一条连接删除也意味着呼叫状态的删除。

8.4.2.3 对呼叫中已有连接的修改

E-NNI 信令可支持对呼叫中以下服务参数进行无损修改：

a) 以太网 SENDER_TSPEC/FLOWSPEC(CIR，CBS，EIR，orEBS)或 CE-VLANID 映射信息中带宽域；

b) SONET 或 OTN SENDER_TSPEC/FLOWSPEC 复用字段。

为了支持连接修改，原始连接应以共享显示预留的方式建立原始的连接，以允许两个或多个 RSVP 路由状态共享相同的资源（如带宽或 CE-VLANID 信息）。使用 E-NNI2.0 中规定进行服务参数无损修改时应使用原始 Resv 消息中的 SESSION_ATTRIBUTE 请求 SE 预留方式。E-NNI2.0 节点不能转发无损修改请求时应产生一个错误指示，携带 error code“Traffic Control Error：ServiceUnsupported”。

图 34 显示了对服务参数的无损连接修改。可以使用相同的消息流对已存在连接的带宽或 CE-VLANID 映射信息的修改。

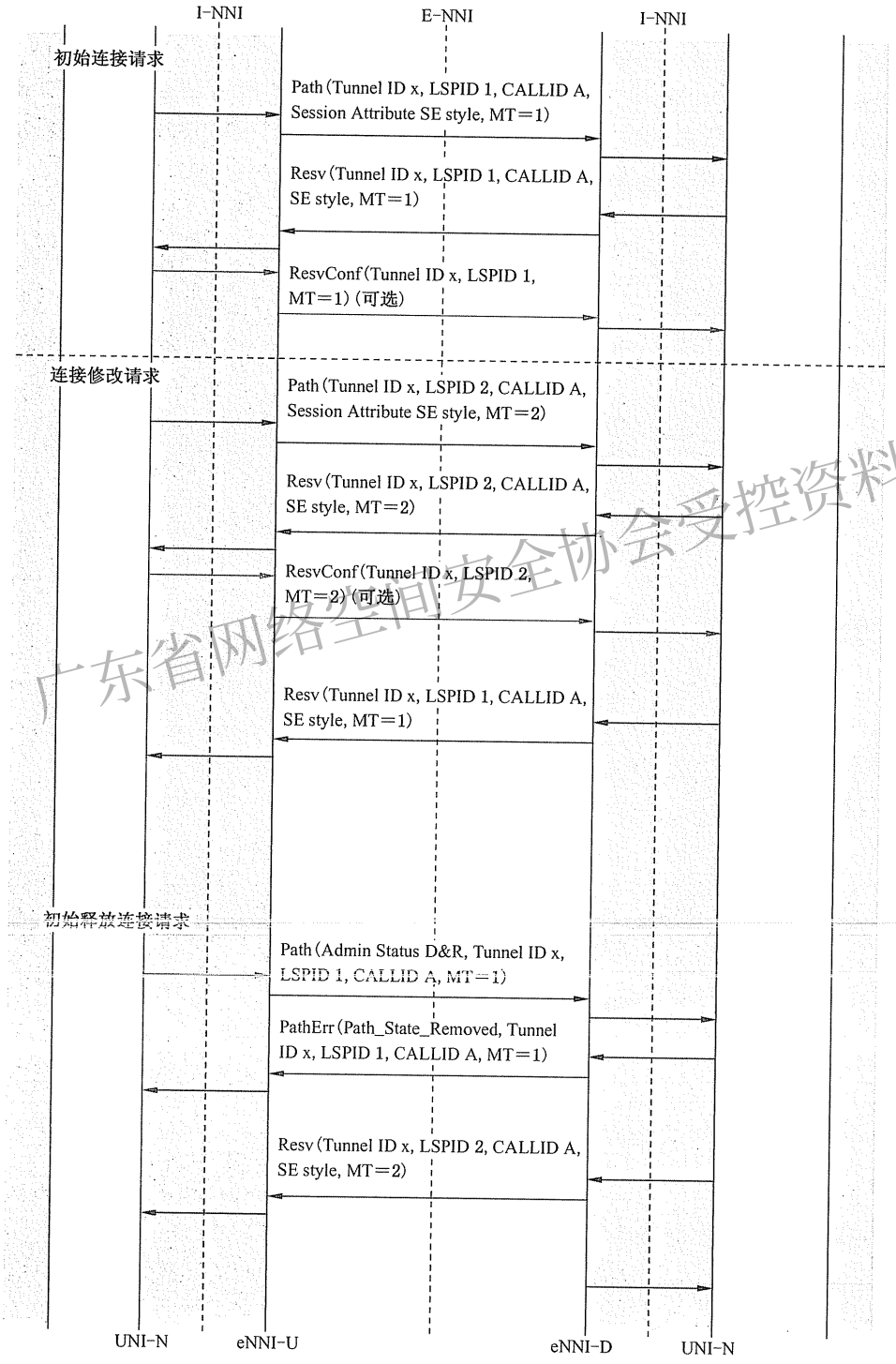


图 34　连接修改成功

建立呼叫及相应的连接时，eNNI-U 向 eNNI-D 发送 Path 消息请求建立呼叫和连接。Path 请求包含了一个 SESSION_ATTRIBUTE 对象，其中 SEStyle request 标记已设。如果连接建立请求中没有 CALL_ID，eNNI-U 会在 Path 消息中指定一个唯一的 CALL_ID 对象。eNNI-U 也会为连接指定一个本地唯一的 SESSION 对象和 LSPID。

如果没有呼叫状态与 eNNI-U 接收到的连接修改请求中的 CALL ID 相关联，eNNI-U 应发送错误指示“invalid/unknown call ID”。同样，如果没有相应的连接状态，eNNI-U 应发送错误指示“invalid/unknown connection ID”。eNNI-U 可以进行连接修改时，eNNI-U 会产生一个新的与原始连接具有相同 SESSION 对象，不同 LSP_ID 的 Path 消息。这使得新的连接与原始连接可以共享相同的连接资源。

当接收到连接修改指示时，eNNI-D 反向发送一个新的 Resv 消息，其中包含 FILTER_SPEC 及与 Path 消息对应的标签。

eNNI-U 接收到来自网络的修改确认请求时应发送一个新的 ResvConf 消息。此时网络中应该出现对应原始连接的删除请求，eNNI-U 收到该请求后应该产生一个 Path 消息，并设置 ADMIN_STATUS 对象中的 Delete 和 Reflect(D&R)比特。

在收到 PATH_STATE_REMOVED 标记已设的 PathErr 消息后，源连接 Path 状态的清除将会导致 Resv 状态的清除，继而与被删除状态相关的 Resv 停止刷新。

对连接带宽修改的失败结果是发送 Path 消息请求更多的带宽，并且不能影响到其他的 Path 消息，或 RSVP 状态。图 35 显示了增加连接带宽失败的情形。

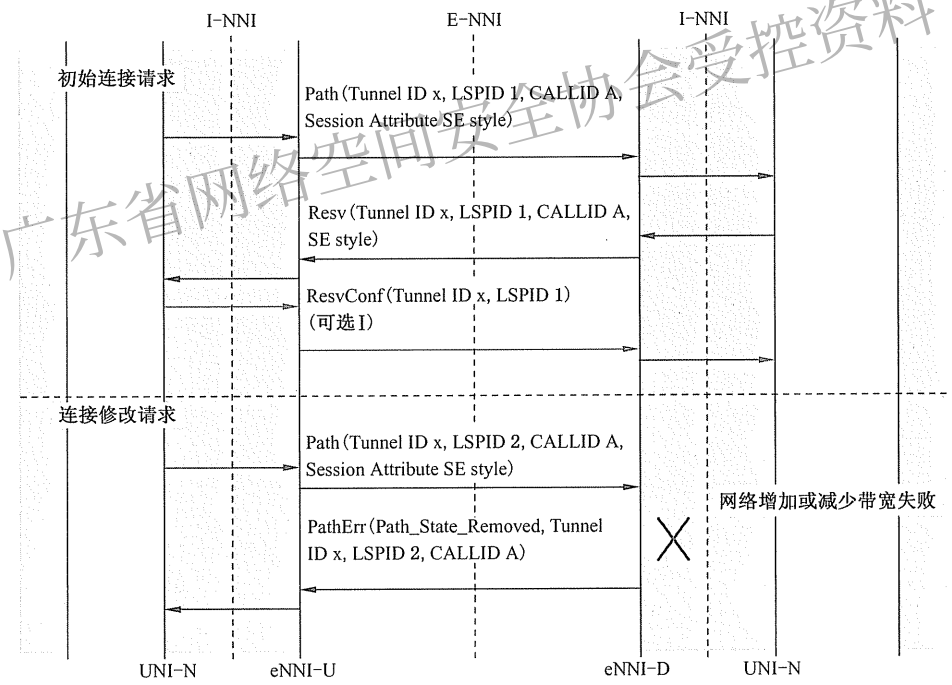


图 35　连接修改失败

这种情况下，ResvConf 消息不是必需的，增加带宽可以通过同一个消息流来实现。连接带宽增加失败会导致发送 PathErr 消息，该操作不能影响到其他的 Path 消息，或 RSVP 状态。

8.4.3　连接删除

8.4.3.1　源或宿发起的正常删除

RSVP 允许使用 PathTear 消息或 ResvTear 与 PathTear 消息的结合实现连接删除。节点收到

PathTear 消息后，删除连接状态并转发该消息。在光网络中连接的删除（如删除交叉连接）可能导致下游节点认为连接出现故障（如 LOS 等）从而触发网络的保护倒换。

为避免该情况，应该遵循以下的正常删除连接的流程。在实际删除之前，通过设置 Path 或 Resv 消息中 ADMIN_STATUS 对象的 D-bit 向连接路径上的各节点发出通知。IETF RFC3473 描述了该流程，见图 36 和图 37。

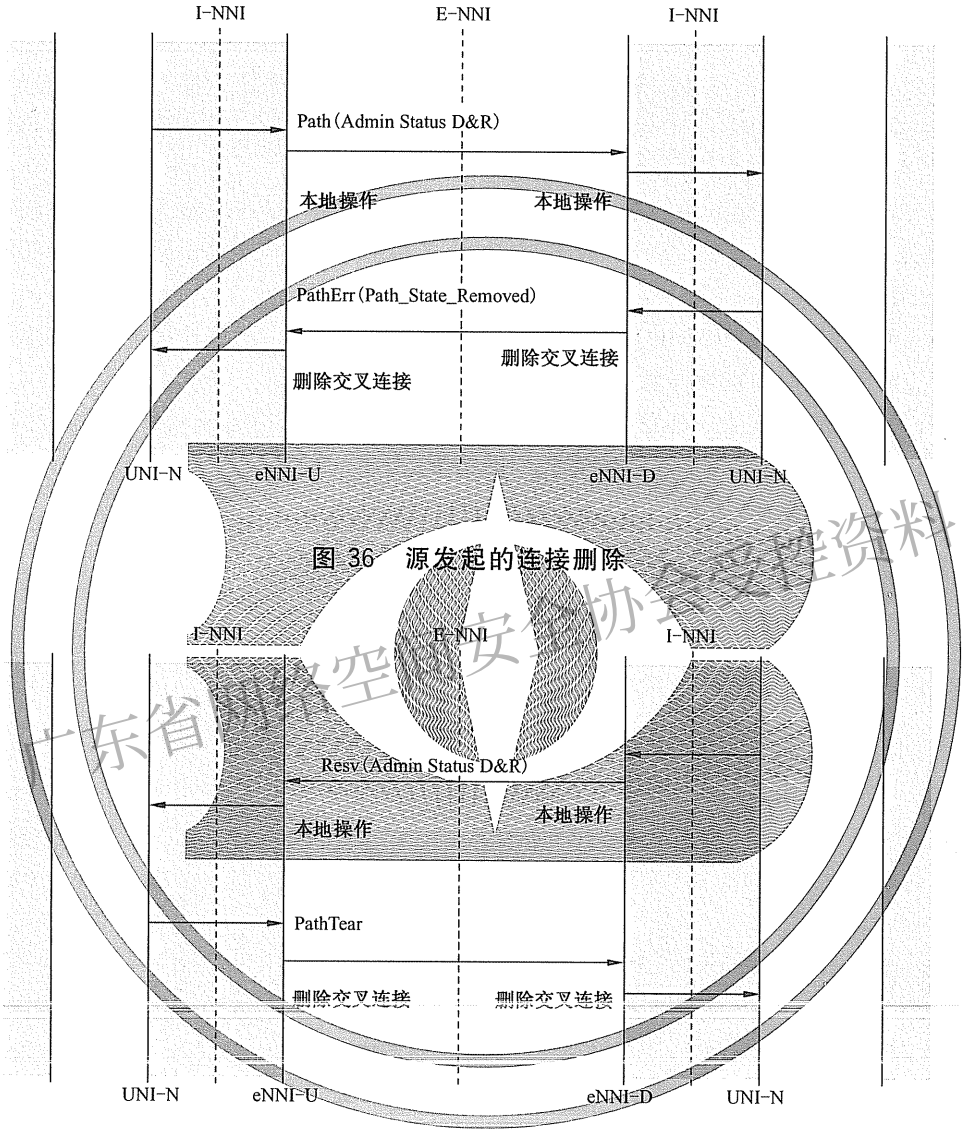


图 37　宿发起的连接删除

8.4.3.2　网络发起的连接正常删除

OIF E-NNI1.0 和 OIF E-NNI2.0 规定要求支持从 E-NNI 接口或从网络节点发起的连接正常删除。在 OIF E-NNI1.0 中信令规定，正常删除通知采用 A&R 比特置位的 Path 或 Resv 消息实现。在 OIF E-NNI2.0 中，为了保持与 IETFRFC3473 的一致性，改用 Notify 消息发起正常删除。

8.4.3.3　支持 Notify 消息

OIF E-NNI2.0 要求应支持 Notify 消息，该消息仅用于从 E-NNI 接口或网络节点发起连接删除。

eNNI-U 节点应在向 eNNI-D 发送的 Path 对象中包含 NOTIFY_REQUEST 对象。同样，eNNI-D 节点应在向 eNNI-U 发送的 Resv 对象中包含 NOTIFY_REQUEST 对象。NOTIFY_REQUEST 对象中的 Notify Node Address 域，应设置为发送节点的 SCPCID。

当 E-NNI 接口需要产生一个 Notify 消息时，它根据接收到的 Path 或 Resv 消息中的 Notify Node Address，将把消息传递给相应的节点。对于每个 E-NNI 连接段，需要设置相关会话的对象。

如果一个 E-NNI 节点没有收到 NOTIFY_REQUEST 对象，它不应产生 Notify 消息。此外，如果 E-NNI 节点收到的 Notify Node Address 域其邻居的 SCPCID 不符，它不应产生 Notify 消息。

8.4.3.4 网络发起的正常删除

E-NNI 接口应支持转发网络发起的正常删除通知的能力。此外，E-NNI 接口可以支持产生一个正常删除通知的能力。在 OIF E-NNI2.0 规范中，网络发起的正常删除采用 Notify 消息，替代采用 A&R 比特置位的 Path 或 Resv 消息的实现机制。

eNNI-D 节点向 eNNI-U 发送 Notify 消息来发起一个正常删除通知。正常删除通知消息包含 D 比特置位的 ADMIN_STATUS 对象。eNNI-D 收到网络发出的正常删除通知消息后，也将发送正常删除通知消息。

eNNI-U 节点可以发起正常删除通知，或者在收到正常删除通知后，在网络中转发正常删除通知消息。

在 OIF E-NNI2.0 中规定，正常删除通知应向上游发送到源节点。收到删除通知后，源节点将发起正常的正常删除过程。如图 38 所示。

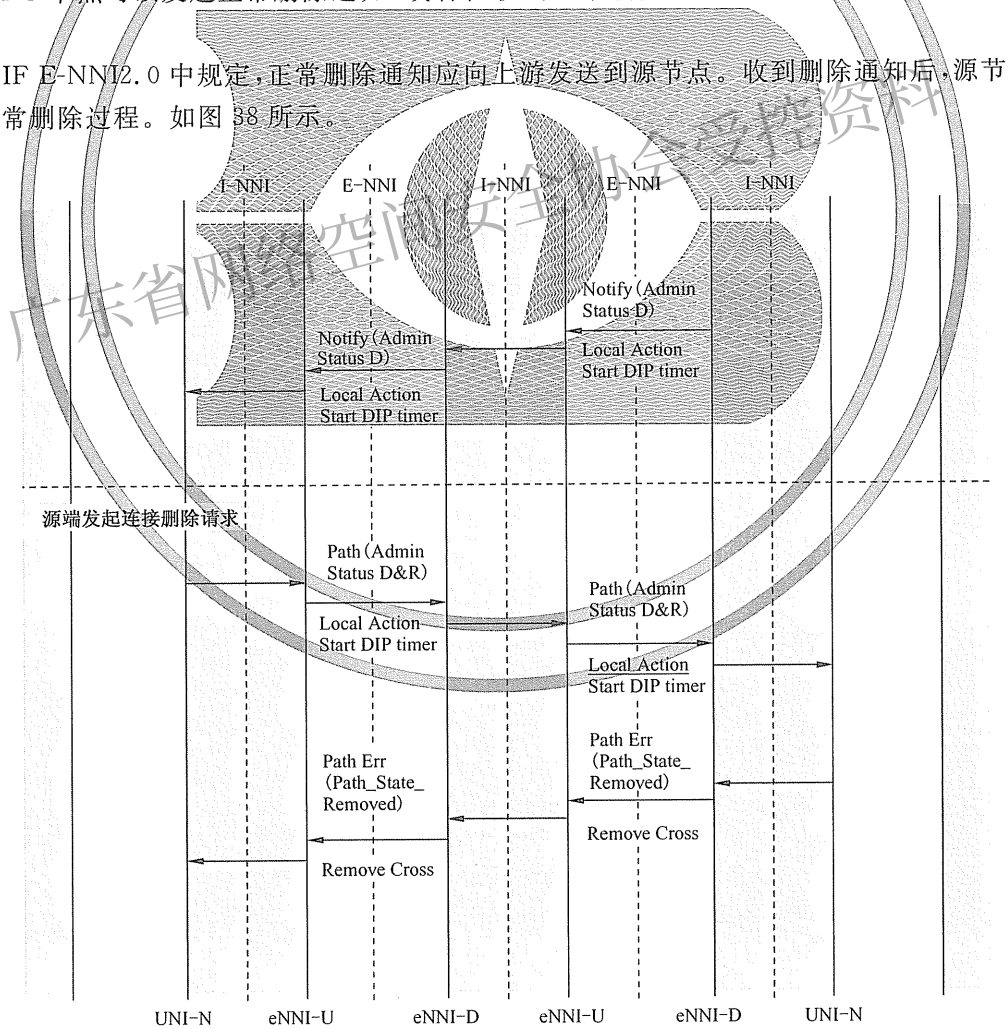


图 38 网络发起的正常删除：eNNI-D 发起连接删除

8.4.3.5 正常删除过程中 OIF E-NNI2.0 与 OIF E-NNI1.0 的兼容性

如果 OIF E-NNI2.0 节点的邻居节点仅支持 OIF E-NNI1.0 信令，则不应发送 Notify 消息用于正常删除通知。OIF E-NNI2.0 节点应发送包含 A&R 比特置位的 ADMIN_STATUS 对象的 Path 或 Resv 消息，来启动网络正常删除。

OIF E-NNI2.0 节点通过手工配置或者自动发现确定其邻居 E-NNI 的版本。如果未接收到 NOTIFY_REQUEST 对象，或者虽然接收到 NOTIFY_REQUEST 对象但 Notify Node Address 不等于邻居 SC PCID，则 E-NNI 接口假设邻居节点运行 OIF E-NNI1.0 信令。

OIF E-NNI1.0 信令还允许向下游发送正常删除通知。这时，如果 eNNI-D 采用 OIF E-NNI2.0 信令，eNNI-U 应使用包含 ADMIN_STATUS 对象的 Notify 消息，向下游 eNNI-D 发送正常删除通知。如果 eNNI-D 采用 OIF E-NNI1.0 信令，则应发送包含 A&R 比特置位的 ADMIN_STATUS 对象的 Path 消息，来转发正常删除通知。

8.4.3.6 强制删除

E-NNI 接口应支持发起连接的强制删除。强制删除主要用于以下情况：

——内部网络失效，强制网络拆除连接；

——ADMIN_STATUS 对象的“删除进行中”计时器超时。

eNNI-U 节点通过删除其 RSVP 状态并拆除交叉连接来发起强制删除。然后，eNNI-U 节点向下游 eNNI-D 发送 PathTear 消息，同时向上游发送强制删除信令。eNNI-D 收到 PathTear 消息后，删除其 RSVP 状态，并拆除交叉。eNNI-D 节点继续向下游转发强制删除信令。

图 39 和图 40 描述了 eNNI-D 节点发起的强制删除。eNNI-D 删除其 RSVP 状态和交叉连接，然后向 eNNI-U 发送包含“Path_State_Removed”标记置位的 PathErr 消息，同时向下游发送强制删除信令。eNNI-U 收到 PathErr 消息后，将删除其 RSVP 状态并拆除交叉连接。eNNI-U 将向上游转发强制删除信令。

网络也有可能发起强制删除信令。当 eNNI-U 收到上游网络发出的强制删除信令时，它将删除 RSVP 状态和交叉连接，并向 eNNI-D 发送 PathTear 消息。同样，如果 eNNI-D 收到下游网络发出的强制删除信令，它也将删除其 RSVP 状态和交叉连接，并向 eNNI-U 发送包含“Path_State_Removed”标记置位的 PathErr 消息。

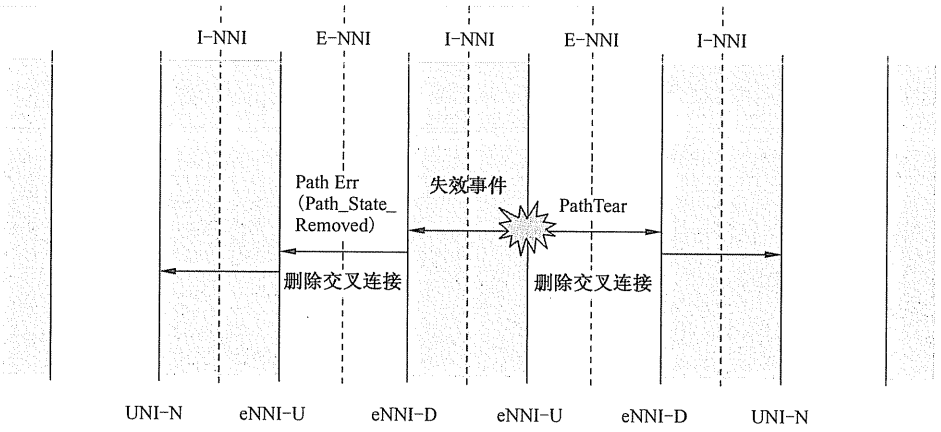


图 39 eNNI-U 发起强制删除

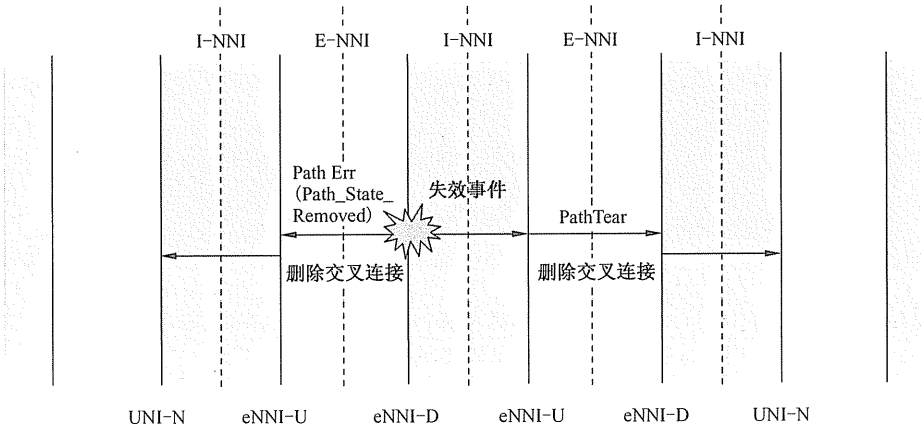


图 40　eNNI-D 发起强制删除

8.4.4　其他 RSVP-TE 消息

RSVP-TE 定义了 ACK 消息，且只在 eNNI-U 和 eNNI-D 之间传送。ACK 消息可以用来对发送的消息进行直接确认，确认功能也可以通过该消息的响应消息中的 MESSAGE_ID_ACK 来间接实现。图 41 描述了连接建立时使用 ACK 消息的信令流程。

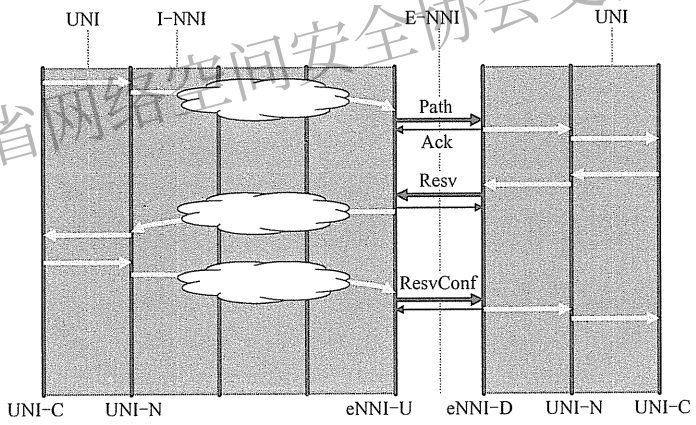


图 41　使用 RSVP-TE 建立基本 SC

RSVP-TE 还定义了 Hello 和 Srefresh 消息。

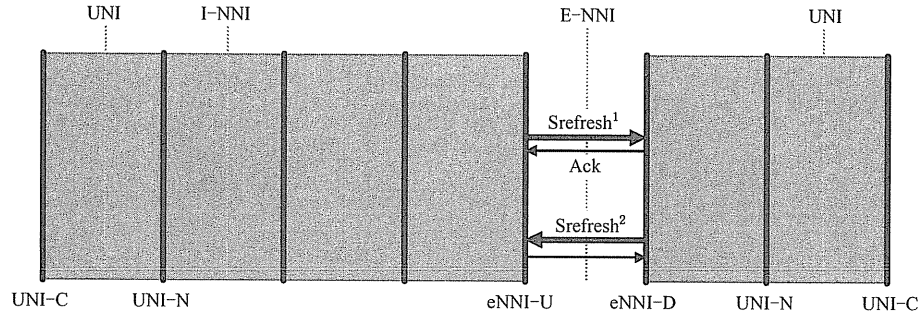
Hello 消息用于：

——保证 RSVP 会话激活(使用请求和确认对象)；

——通过交换 recovery 和 restart timers，发起重启动进程。

Srefresh 消息用于刷新 RSVP-TE 状态而不需要传输 Path 或 Resv 消息，可以减少维持呼叫和连接状态所需要传送和处理的信息。

图 42 描述了使用 Srefresh 消息刷新 Path 和 Resv 状态的例子。



注 1：Srefresh 消息可以用于从 eNNI-U 向 eNNI-D 刷新所有连接的 Path 和 Resv 状态信息。

注 2：Srefresh 可以用于从 eNNI-D 向 eNNI-U 刷新所有连接的 Path 和 Resv 状态信息。

图 42　基本 Srefresh 信令流程

需要注意的是，Srefresh 消息不能再对其他 Srefresh 消息进行汇总。

8.5　RSVP 控制平面失效

8.5.1　RSVP-TE 信令通道失效

信令通道和控制协议实体的失效不应引起已建立连接的删除。IETF RFC3473 描述了控制状态失效（但没有丢失转发状态）的处理机制，使用了 Hello 消息中的 RESTART_CAP 对象。此外，节点应支持 IETF RFC3473 中第 9 章规定的故障处理流程。

RSVP-TE 需要进行消息交换来同步已建立连接的状态。当发生信令通道失效时，为了防止状态信息超时，需要执行“自刷新”程序。当失效恢复后，相邻的控制实体开始交换 Hello 消息，并使用 Hello 消息触发状态同步程序，见图 43。这样可以确保已经建立连接状态的一致。受到信令通道失效影响的节点应遵循以下本地行为：

a）　检测到信令通道失效的节点应向管理系统请求进一步的行为（如维持自刷新或释放部分连接等）。缺省行为是进入呼叫和连接状态的自刷新模式。

b）　当一个节点检测到无法对部分连接的状态与邻居进行同步时，应向管理系统请求进一步的行为。缺省行为是保留连接，除非有明确的指示要求释放连接。

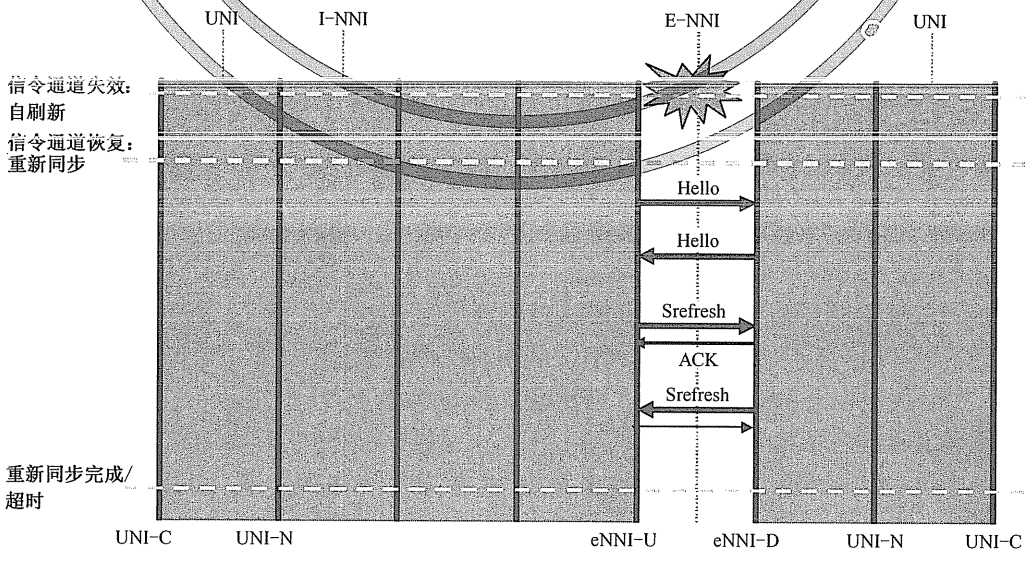


图 43　从信令通道失效恢复

在控制邻接被成功建立以前，节点不应接受任何呼叫请求。从不能发出任何 Hello 消息的节点收到的 Path 消息，可以用于重新触发 Hello 进程。对于未预期的或错误的 Hello 消息，节点通过设置 Hello Request 或 HelloAck 中的 Dst_Instance 为 0 来进行响应，表示收到的消息未被接受。

8.5.2 RSVP-TE 控制平面失效

RSVP-TE 需要进行消息交换来同步已建立连接的状态。当发生信令通道失效时，为了防止状态信息超时，需要执行“自刷新”程序。当失效恢复后，恢复节点应尝试从其本地永久存储器中的信息重建已经建立连接的状态信息。之后，相邻的控制实体开始交换 Hello 消息，并使用 Hello 消息触发状态同步程序。这样可以确保已经建立连接状态的一致。因此，受到控制平面节点失效影响的节点应遵循以下本地行为：

——控制平面节点应提供永久保存呼叫和连接状态信息的能力。尽管重启机制允许相邻控制平面节点自动恢复呼叫和连接状态，当永久存储器提供本地状态的恢复时，也可以用这种机制来验证邻居状态。在这种情况下，依据 RFC3473，如果在 Hello 同步过程中，重启节点检测到邻居节点不支持状态恢复，并且重启节点维持其状态，重启节点应该立刻认为恢复过程已经完成。

——当一个节点检测到无法对部分连接的状态与邻居进行同步时，应向管理系统请求进一步的行为。缺省行为是保留连接和状态，除非有明确的指示要求释放连接。

——失效恢复后的节点可能无法从本地永久存储器中恢复邻居转发邻接信息。在这种情况下，节点应向外部控制系统（如管理系统）请求恢复转发邻接的信息。

9 基于 OSPF 的 E-NNI 路由架构

9.1 概述

本部分规定的 E-NNI 路由满足 GB/T 21645.1 和 GB/T 21645.8—2012 中的要求，可以支持单层路由结构或多层路由结构。在每一个路由层次中，路由协议独立运行，不影响其他路由层次中路由协议的运行。某一路由层次的路由信息，可传送给其上层或下层的路由层次，这种层次路由机制允保证了路由协议在大规模网络中的扩展性。

图 44 表示由多个路由控制域（RCD）组成的 ASON 网络，RCD 之间通过 E-NNI 接口连接。

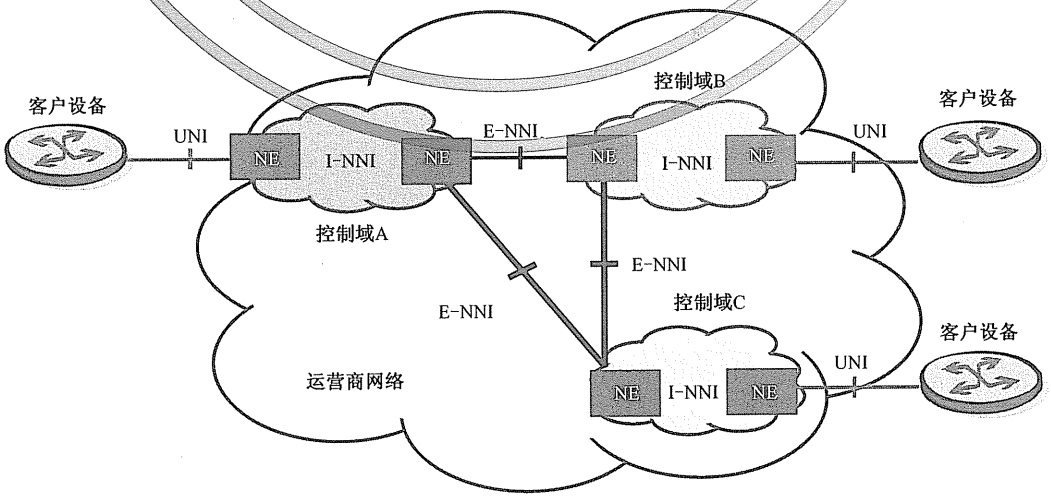


图 44 由多个路由控制域（RCD）组成的控制平面网络

本部分规定的 E-NNI 路由协议是基于 IETF GMPLS OSPF 协议的扩展。在每个路由等级存在一个单一的 OSPF 域，即路由区(RA)。E-NNI 路由通过层次结构来实现扩展性机制，而不需要使用 OSPF 的多域运行模式，OSPF 多域运行模式的应用有待进一步研究。本部分规定了基于 OSPF 的 E-NNI 路由单级路由的协议扩展，对多等级路由的协议支持待研究。

本部分规定的 E-NNI 路由不包括在 DCN 中使用 OSPF 的内容，因此不包含 OSPF(1～4)类的 LSA，以及用于 IP 路由的路径计算。只参与 E-NNI 路由的 RC 不允许向邻接 RC 发送与光网络无关的 LSA，如用于 IP 网络的 Type 1(Router LSAs)和 Type 3(Summary Network LSAs)。

9.2 路由信息的传送

本部分假定存在满足 GB/T 21645.3—2009 中的 DCN，在不同的路由控制器(RC)之间提供控制信息的传送。

在 ASON 中，构成邻接关系的 RC 节点，在控制平面一般不是物理相邻的，而且在 SCN 拓扑中也不是一跳路由。E-NNI 路由建立邻接关系方式存在以下两种方式：

a) 点到多点方式

在 OSPF 点到多点方式中，RC 之间的邻接关系由人工配置，不使用 OSPF Hello 消息进行发现。为了在大于一跳的 RC 之间创建 OSPF 邻接关系，E-NNI OSPF PDU 的 IP 包的 TTL 初始值应大于 1，并建议设置为 255。此外，OSPF Hello PDU 的网络掩码应设置为 0x00000000，从而允许非直接邻居的节点建立邻接关系。

b) 隧道方式

隧道是用于非邻接节点的一个常用技术。E-NNI 使用的隧道方式是利用隧道技术为非相邻 RC 之间建立 SCN 直联链路。为了避免非法的路由和业务成环，需要额外的管理机制来保证隧道仅用于 E-NNI 消息。隧道方式要求在 RC 之间建立适合的隧道，并使用这些隧道作为 E-NNI 相关的 OSPF-TE 实例的接口。

本部分推荐使用点到多点方式建立路由邻接关系。

9.3 路由域拓扑的抽象

ASON 路由结构允许对路由域拓扑进行抽象，拓扑抽象可以减少域间路由协议承载的信息。采用路由层次和拓扑抽象后，对外发布的拓扑是域内路由区实际拓扑的抽象视图。

拓扑抽象视图表示为一些链路和相关的链路代价，主要用于进行跨路由区的通道计算。拓扑抽象视图根据路由区内的情况、所使用的提供附加路由信息的工具(例如路由计算单元 PCE)的不同，会对路由性能产生不同的影响。如果域内的可用带宽相对于平均的业务请求较大，节点层次的抽象对于计算的路由质量将不会有很大的负面影响。

E-NNI 路由支持两种抽象方式：

a) 单节点抽象：是指将一个路由域抽象为一个节点。

b) 多节点抽象：是指将一个路由域抽象为多个边界节点，以及连接边界节点的抽象链路。

本部分要求 E-NNI 应支持采用多节点抽象方式。

此外，域间路由计算也可以通过查询路径计算单元(PCE)实现，这种方式能够得到比 E-NNI 路由协议发布更精确的路由信息。PCE 可以通过多种方式支持，包括管理平面。PCE 的相关要求待研究。

9.4 安全性考虑

E-NNI 路由的安全性要求应符合 OIF SEP-02.1。

10 单级 OSPF E-NNI 路由

10.1 配置

10.1.1 基本配置

实现 OSPF 等级路由的前提条件是每个 RCD 应至少有一个代言者(speaker),即路由控制器(RC)。这个 RC 的工作范围是一个路由区(RA)。一个 RC 有一个 RC ID 和一个用于 OSPF 协议通信的 SCN 地址。

通过发现或配置,每个 RC 发现其在传送平面的邻居 RC,这些 RC 应在同一 RA 内。RCID 和对应的 SCN 地址可以通过发现或配置得到。本部分不包含发现过程。如果通过配置,可以人工决定与特定的邻居 RC 建立邻接关系,以便交换路由信息。

为了启动 OSPF RCD 的第一级路由层次,需要对一系列参数进行配置。图 45 表示一个具有 3 个 RCD 的 ASON 网络。

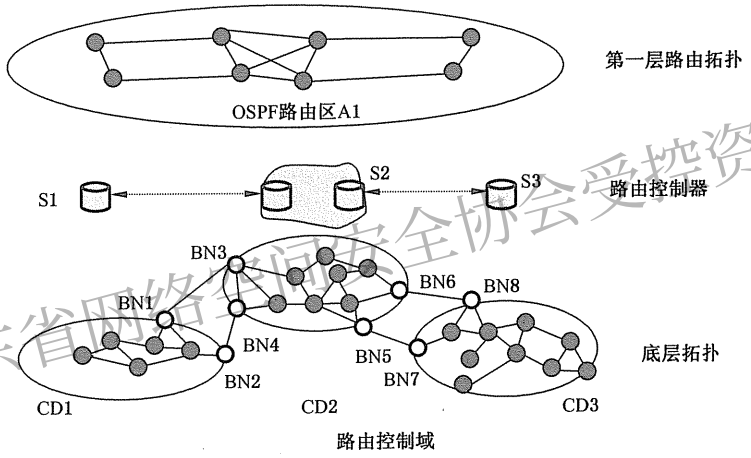


图 45 单级 OSPF-TE 运行示例

单级 E-NNI OSPF 的基本参考配置如下:

Level 1 area:0.0.0.0

IP TTL:255

OSPF Hello Message:

Subnet Mask:0.0.0.0.

HelloInterval:30 s

RouterDeadInterval:120 s

10.1.2 路由控制器

每个 RCD 应至少指定一个 RC,由 RC ID(OSPF Router ID)来识别。如图 45 中 CD2 中的 S2 是多个 RC 联邦,用于在 A1 中发布 CD2 的路由信息。S2 多个 RC 联邦的实现方式,不在本部分范围内。

10.1.3 邻接 RCD 中的路由控制器

对于一个 RCD 中的每个 RC,在邻接 RCD 中至少存在一个 RC,并具有以下信息:

a) 邻接 RC 的 RC ID;

b） 邻接 RC 的 SCN 地址。

例如，S2 的邻接 RCD 信息配置如表 25 所示。

表 25 RC S2 的邻接 RC 节点

邻居 RC	RC ID	SCN 地址
S1	S1 的 Router ID	S1 的 SCN 地址
S3	S3 的 Router ID	S3 的 SCN 地址

10.1.4 域间 TE 链路

在 RC 节点上可配置域间链路信息，域间 TE 链路反映了与邻接 RCD 互连的情况，以及从本地节点到远端节点出口方向的流量参数。

例如 S2 上总共配置了 4 条域间链路，如表 26 所示。

表 26 S2 上配置的域间链路

域间 TE 链路	本地边界节点	远端边界节点
BN3-BN1	BN3	BN1
BN4-BN2	BN4	BN2
BN5-BN7	BN5	BN7
BN6-BN8	BN6	BN8

10.1.5 域内链路

域内链路反映多节点抽象时边界节点之间的抽象 TE 链路，包含了从入口节点到出口节点单方向的流量参数。域内 TE 链路具有抽象的特性，它反映一个 RCD 拓扑的聚合。可以在 RC 上人工配置一条或多条域内链路，或者 RC 也可以根据域内路由信息自动推断出域内链路。域内 TE 链路的配置数量由该 RCD 决定。

在图 45 的例子中，S2 上总共配置了 12 条域内链路，如表 27 所示。

表 27 S2 上配置的域内 TE 链路

域内链路	本地边界节点	远端边界节点
BN3-BN4	BN3	BN4
BN4-BN3	BN4	BN3
BN3-BN5	BN3	BN5
BN5-BN3	BN5	BN3
BN3-BN6	BN3	BN6
BN6-BN3	BN6	BN3
BN4-BN5	BN4	BN5
BN5-BN4	BN5	BN4
BN4-BN6	BN4	BN6

表 27（续）

域内链路	本地边界节点	远端边界节点
BN6-BN4	BN6	BN4
BN5-BN6	BN5	BN6
BN6-BN5	BN6	BN5

10.1.6 可达 TNA 地址

连接到一个给定 RA 中的 NE 的 TNA 地址由上一等级 RA 中的 RC 进行发布。为了具有可扩展性，在发布之前可以对这些地址进行摘要。

例如，在 CD2 中的可达性 TNA 地址由 S2 广播，这些 TNA 地址可以在 S2 上进行配置，或者由内部路由信息推断出来。

10.2 运行

以上配置目的是启动 RCD 中 OSPF-TE 的第一级路由层次。配置完成后，具有邻接关系的 RC 通过交换 OSPF 消息形成第一级路由层次。RCD 边界节点之间并不建立邻接关系，除非它们也是其 RCD 的 RC。

属于同一路由层次的 RC 节点将在控制平面形成路由邻接。同时，每个 RC 将发布其相关 RCD 的域间链路和域内链路，以及可达 TNA 地址。

例如，图 45 中 RCS1、S2、S3 在控制平面上形成 OSPF 路由邻接。S2 发布表 26 所列的抽象链路，作为第一层次路由区（OSPF 路由区 A1）的部分拓扑。表 26 中所列的抽象链路包含了 CD2 到 CD1、CD2 到 CD3 的链路属性。而 CD1 到 CD2 以及 CD3 到 CD2 的相应链路属性将分别被 RCS1 和 S3 发布。表 27 中的链路表示了与 CD2 相关的路由信息，用于穿越 CD2 域连接的通道选择。此外，S2 将在整个 OSPF 路由区 A1 发布 CD2 的可达 TNA 地址。

路由协议的运行遵循 IETF RFC2328 和本部分第 12 章定义的程序。运营商应配置产生 LSA 的定时器，以免发生发送和接收的失配。

11 多级路由层次的运行结构

11.1 配置

11.1.1 多级路由层次

路由层次通过路由区的垂直堆叠形成。E-NNI 的多级路由层次符合 GB/T 21645.8—2012 中的需求。在一个路由区内，独立运行一种路由协议，每个路由区通过配置或者选举的方式产生至少一个 RC，这个 RC 在上级路由层次中代表了该路由区。通常在路由层次中第 N 层的 RC 和它对应的控制域路由实体之间存在一定的通信机制，保证双向的路由信息交换，即路由信息的上行和下行，这些信息交换在控制域内进行。

建立路由层次需要进行一定的配置。由运营商决定路由区的层次结构，即路由区的包含结构。路由层次结构通常反映了运营商网络的组织架构。

图 46 中，网络被分为 2 个路由层次。在层次 N 中，有 3 个独立的路由区 A1、A2 和 A3。在这些区域内部，不需要通过 RC 进行路由信息交换，RC 仅负责找到跨区域的路由。这 3 个路由区被上一级路由层次中的 A4 路由区所包含。在 A4 中，RCS8、S9 和 S10 分别发布 A1、A2 和 A3 的路由信息。

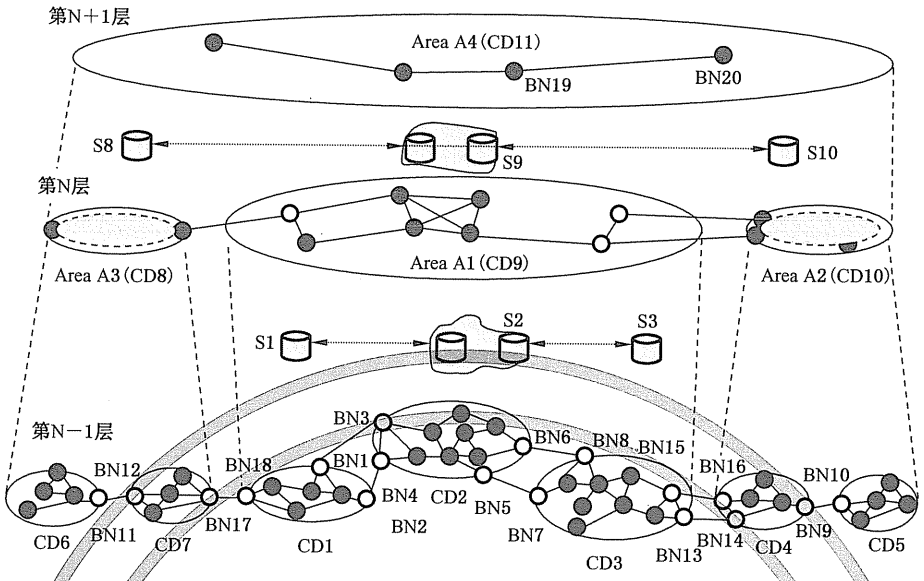


图 46 多级路由层次示例

11.1.2 路由控制器和路由区

位于路由层次中第 N 级的路由区 RAn，至少在第 N+1 层存在一个对应的 RC，直到路由层次的最高级。在路由层次模型中，每个层次中各个路由区的操作是独立的。在一个层次中运行链路状态路由所得到的路由信息，可以上行（最高层除外）和下行（最低层除外），或者在该层的 RC 上进行配置。

通过信息上行可以把一个路由区的路由信息发布给其他路由区，用于确定跨区域连接建立的路由。上行的路由信息在节点上被逐层处理。为了满足网络可扩展性要求，上行的信息可以进行一定的信息聚合和汇总。从层次 N 上行的路由信息，由 N+1 层的 RCn+1 进行发布。因此，上级 RC 不需要了解下级的标识（RCID，RAID）。信息上行还可以减少配置的压力，例如低层的 RC 可以自动聚合一些信息。

通过信息下行，可以把其他路由区的路由信息发布给本地路由区，使本地路由区可以计算跨越或者达到其他路由区的连接路由。LSA 所发布的路由信息，是从高层路由区发出的在本地路由区以外的路由信息，这些信息经过了聚合和汇总，可以用于连接选路过程。在本部分中，路由信息下行是可选的。

11.1.3 邻接 RCD 的路由控制器（每个 RC）

配置邻接 RCD 的 RC 的方法见 10.1.3。图 46 中，S9 是路由区 A1 中 RC 的联邦，S10 和 S8 分别是 A3 和 A2 的 RC。在 S9 上配置 S8 和 S10 的 RC ID 和 SCN 地址，同样在 S8 和 S10 上配置 S9 的 RC ID和 SCN 地址。

11.1.4 域间链路（每个 RC）

域间链路反映控制域之间的抽象 TE 链路。域间链路的配置可能不同于 10.1.4 的描述，因为路由层次中的更高级别的路由区，可能需要对边界节点和域间链路进行额外的聚合。

例如控制域 CD9 和 CD10 互连时，存在两条物理链路（BN13-BN14 和 BN15-BN16），边界节点和链路在高层路由层次中被聚合，S9 和 S10 的配置如表 28 和表 29 所示。

表 28　图 46 中 S9 上配置的域间链路

域间链路	本地边界节点	远端边界节点
BN19-BN20	BN19	BN20

表 29　图 46 中 S10 上配置的域间链路

域间链路	本地边界节点	远端边界节点
BN20-BN19	BN20	BN19

11.2　运行

11.2.1　控制平面邻接

对于在路由层次 N 的一个路由区 RAn，在路由层次 N+1 有一个对应的路由控制器 RCn+1。在图 46 中，在路由区 A4(即控制域 CD11)中，RC S8、S9、S10 分别代表控制域 CD9、CD10 和 CD8，并构成路由邻接用于交换路由区 A4 的 OSPF 路由信息。

11.2.2　拓扑聚合和上行

1 层以上的控制域拓扑可以被 RC 聚合，并且自动在上一级路由层次进行发布，或者通过配置实现。其他的拓扑聚合方法待研究。

a)　域间链路

如 10.1.3 所述，域间链路信息可在包含链路两个端点的路由层次的 RC 上配置完成。

b)　域内链路

在一个多等级路由层次结构中，域内链路可以采用 10.1.5 描述的方法进行配置，同时也可以自动发现和产生。当一个路由区对外发布至少两个边界节点时，域内拓扑可以通过计算虚拟域内链路的方式来进行抽象。域内链路一旦被聚合，将通过上级路由层次中对应该控制域的 RC 发布。

11.2.3　TNA 地址汇总和上行

在路由层次 N 中的路由区内，每个节点在区域内可以发布一个或者多个 TNA 地址。该路由区内的 RC 在向上级路由层次发布可达 TNA 地址之前，可以对所有可达的 TNA 地址进行汇总。

11.2.4　从层次 N 到 N－1 的路由信息下行

路由层次 N 的节点记录的路由信息，可以下行到 N－1 层的节点，采用如下所述的标准机制。用来减少下层 RC 信息存储需求的下行机制待研究。

路由信息的下行由层次 N 的 RC 执行，层次 N 的 RC 转发路由信息到相关的 RC 或者 N－1 层的 RC，这些路由信息被发布到该 RC 所属的整个路由区中。

下行的路由信息包括：

a)　包含域间链路的 LSA；

b)　包含域内链路的 LSA；

c)　包含可达 TNA 地址的 LSA。

N 层下行的 LSA，可被 N－1 层 RC 发布。同样的信息可能以同样方式继续下行到 N－2 层。

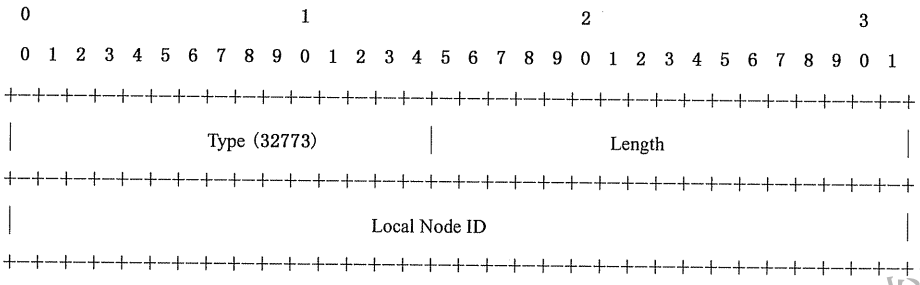
路由信息下行的目的是在低层路由层次中，向控制域的所有节点发布流量工程信息，甚至到0层的所有节点，这样端到端的路径选择就可以采用分布方式实现。

12 E-NNI路由的OSPF协议扩展

12.1 新增与扩展的Sub-TLV

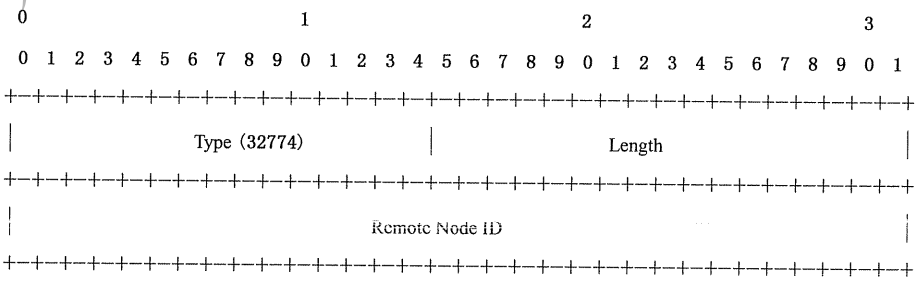
12.1.1 本地节点ID Sub-TLV

域间或域内TE链路使用本地节点ID Sub-TLV(Type 32773)来指示TE链路的本地末端。该Sub-TLV可以用来支持控制平面和传送平面的分离，以及拓扑提取。格式如下：



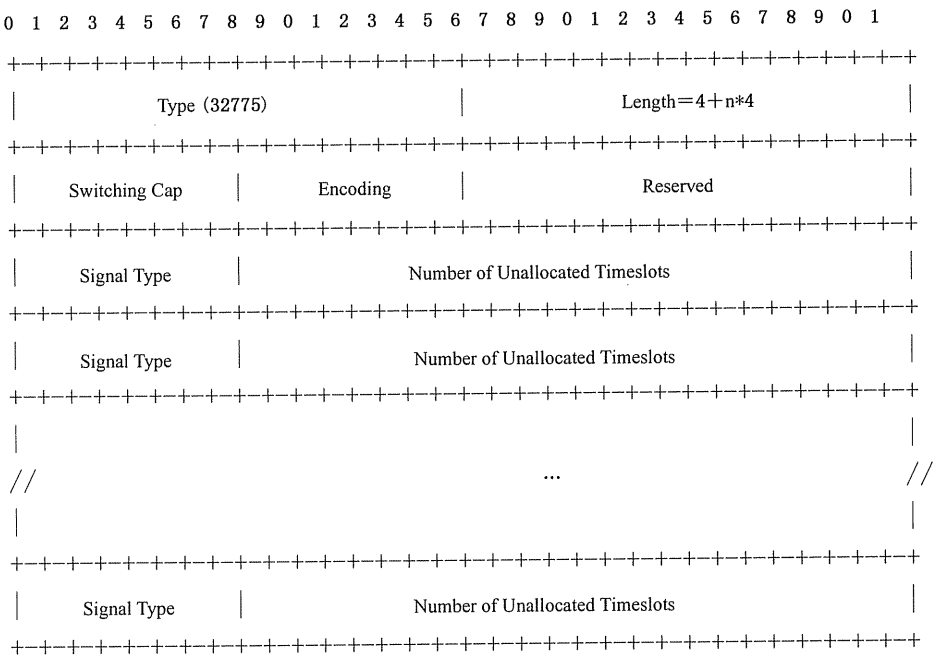
12.1.2 远端节点ID Sub-TLV

域间或域内TE链路使用远端节点ID Sub-TLV(Type 32773)来指示TE链路的远端末端。该Sub-TLV可以用来支持控制平面和传送平面的分离，以及拓扑提取。格式如下：



12.1.3 SDH接口交换能力描述符Sub-TLV

这个Sub-TLV[Type 32775，长度(4+n*4)字节]用来表示SDH的带宽信息，格式如下：



注：n 表示该链路支持的信号类型的数量，大于或等于 1。

根据 IETF RFC4203(GMPLS-OSPF)的定义，用于 SDH 接口的交换能力和编码字段应取下面的值：

交换能力(8 bit)：100(TDM)

编码(8 bits)：5(SDH)

保留(16 bits)：0

信号类型(8 bits)：如表 30 所示。

表 30 信号类型

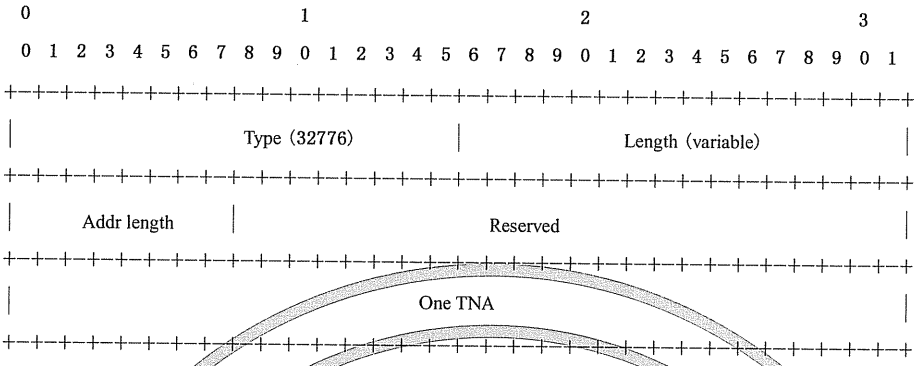
取值	信号类型
1	VT1.5 SPE/VC-11
2	VT2 SPE/VC-12
3	VT3 SPE
4	VT6 SPE/VC-2
5	STS-1 SPE/VC-3
6	STS-3c SPE/VC-4
21	STS-12c SPE/VC-4-4c
22	STS-48c SPE/VC-4-16c
23	STS-192c SPE/VC-4-64c

未分配时隙数量(24 bits)：表示一条 TE 链路具有的相同信号类型的未分配时隙数量。

12.1.4 TNA 地址 Sub-TLV

TNA 地址 Sub-TLV 表示一个 TNA 地址。TNA 地址有 3 种可能的格式：IPv4、IPv6 和 NSAP。TNA 地址 TLV 除了包括一个 TNA 地址 Sub-TLV 外，还可以包括至少一个节点 ID Sub-TLV。也可

以包括多个节点 ID Sub-TLV 和 TNA 地址 Sub-TLV。IPv4 TNA 地址 Sub-TLV(Type 32776,长度可变)的格式如下:



其他格式的 Sub-Type 的值如下:

a) IPv6 TNA-32778;

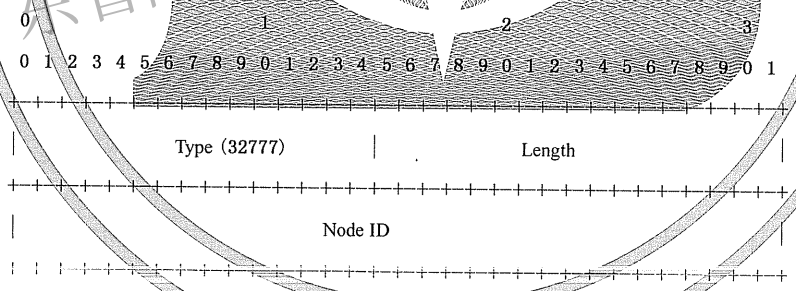
b) NSAP TNA-32779。

addr_length 规定 TNA 地址的长度(单位:bit)。

可以使用地址前缀汇聚 TNA 地址,此时 addr length 表示 TNA 地址前缀的长度。例如,地址前缀 192.10.3.0/24 可以表示为 TNA=193.10.3,addr_length=24。

12.1.5 节点 ID Sub-TLV

节点 ID Sub-TLV(Type 32777)包含在 TNA 地址 TLV 中,它包含拥有该 TNA 地址的节点。该 Sub-TLV 的格式如下:



12.2 不透明 TE LSA

12.2.1 不透明 TE LSA 类型

GMPLS-OSPF 定义了两类顶级 TLV,即节点级 TLV 和链路级 TLV。OIF ENNI-OSPF1.0 增加了一类新的顶级 TLV:地址 TLV(TNA 地址 TLV)。地址 TLV 的目的是在一个给定的路由域内广播 TNA 地址。

对于 E-NNI 路由,路由器地址 TLV 和 TE 链路 TLV 的广播是必选的。TNA 地址 TLV 的广播依赖运营商的网络。如果一个 RC 收到 TNA 地址 TLV,应存储并向相邻的 RC 发布。

根据 IETF RFC3630(OSPF-TE)的规定(如表 31 所示),路由器地址 TLV 应出现在由一个 RC 发出的一个 TE LSA 中。一个 RC 发出的一个 TELSA 中只能包含一个 TE 链路 TLV 和一个 TNA 地址 TLV。

表 31　不透明 TELSA

类型	顶级 TLV	语义	参考	范围
1	路由器地址 TLV	RC ID	IETF RFC3630	由 RA 中的任意 RC 发送。为与 IETF RFC3630 保持一致，为必选
2	TE 链路 TLV	点到点链路	IETF RFC3630	由任意发布一条或多条 TE 链路的 RC 发送。NNI 路由必选
3	TNA 地址 TLV	可达 TNA		由任意发布 TNA 可达性的 RC 发送。与运营商相关

12.2.2　RC 不透明 TE LSA

一个 RC 代表一个路由控制域(RCD)，ITU-TG.7715.1 描述了 RC 不透明 TE LSA 应包含的信息，详见表 32。其中顶级 TLV 是路由器地址 TLV，格式在 IETF RFC4203 中定义。

表 32　路由控制器信息

No	TLV	语义	参考	选择性
1	路由器地址顶级 TLV	顶级 TLV	IETF RFC3630	必选

12.2.3　TE 链路不透明 LSA

TE 链路不透明 LSA 用来表示一条域内或域间的 TE 链路。OSPF 分组头的 RC 字段携带广播的 RC ID。链路 ID Sub-TLV 携带域内/域间 TE 链路的对端 RC ID；或者对域间 TE 链路携带对端 RC ID，对域内 TE 链路携带未知值。当一个抽象 RC 发起的 TE 链路不透明 LSA 代表域内 TE 链路时，采用后一种情况。

TE 链路包含的信息如表 33 所示。

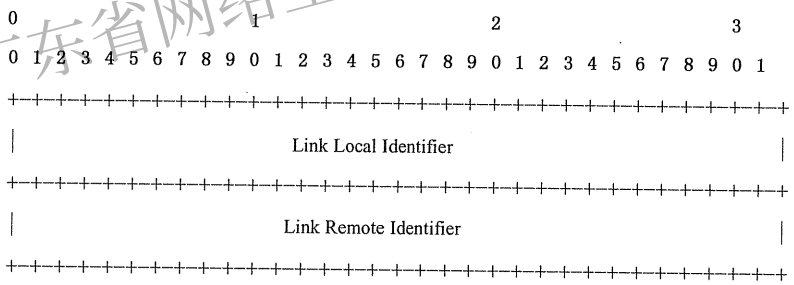
表 33　域内和域间 TE 链路信息

No	TLV	语义	参考	选择性(M 为必选)
1	链路顶级 TLV	顶级 TLV	IETF RFC3630	M
2	链路类型 Sub-TLV	点到点链路	IETF RFC3630	M
3	链路 ID Sub-TLV	对端 RC-ID (对于域间链路应设置为对端 RC-ID。对于域内链路，如果只有一个 RC，应设置为 0.0.0.0(未知)；如果有多个 RC，应设置为对端 RC-ID)	IETF RFC3630	M
4	链路成本 Sub-TLV	链路成本	IETF RFC3630	M(缺省值：1)
5	链路资源类型 Sub-TLV	链路资源类型	IETF RFC3630	M(缺省比特掩码：0…0)

表 33（续）

No	TLV	语义	参考	选择性(M 为必选)
6	链路本地/远端 ID Sub-TLV 中的本地 ID	本地接口 ID	IETF RFC4203	M
7	链路本地/远端 ID Sub-TLV 中的远端 ID	远端接口 ID	IETF RFC4203	M(如未知应设 0.0.0.0)
8	链路保护类型 Sub-TLV	链路保护类型	IETF RFC4203	O(缺省:无保护链路)
9	SRLG Sub-TLV	共享风险链路组	IETF RFC4203	O(缺省:链路 ID 是 SRLG)
10	本地节点 ID Sub-TLV	本地末端(如本地边界节点 ID)	12.1.1	M
11	远端节点 ID Sub-TLV	远端末端(如远端边界节点 ID)	13.1.2	M
12	SONET/SDH 接口交换能力描述符 TLV	描述 TE 链路的带宽信息	12.1.3	M
注：根据 IETF RFC3630，链路类型和链路 ID Sub-TLV 应且只能出现 1 次。远端节点 ID，链路成本和接口 ID Sub-TLV最多只能出现 1 次。				

本地 ID 和远端 ID 都是 IETF RFC4203(GMPLS-OSPF)定义的链路本地/远端 ID Sub-TLV(Type 11)的一部分，格式如下：



注 1：某些情况下，链路远端 ID 可能未知(值为 0)，此时该 LSA 不用于路径计算。

注 2：某些情况下，链路 ID 和远端节点 ID 可能包含重复信息，此时便用远端节点 ID。

12.2.4 可达 TNA 不透明 LSA

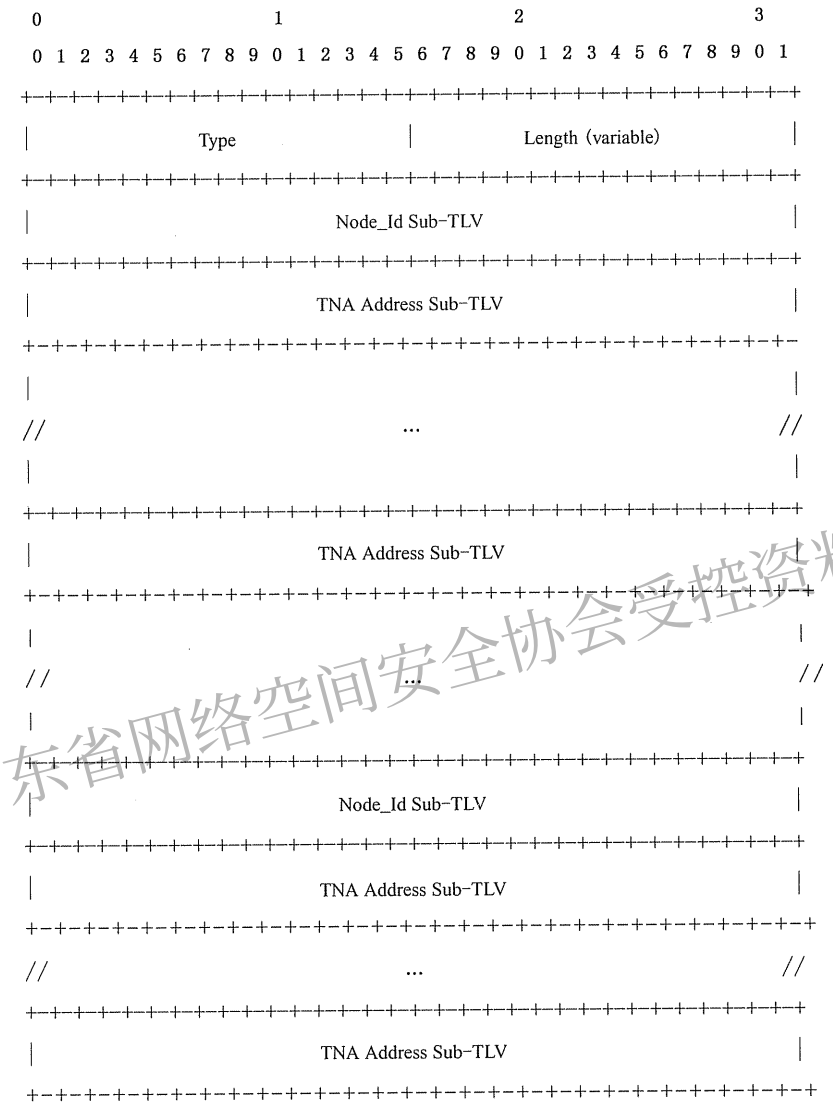
路由控制器信息见表 34。

表 34 路由控制器信息

No	TLV	语义	参考	选择性
1	可达 TNA 顶级 TLV	顶级 TLV	本节	M
2	节点 ID Sub-TLV	规定与该 TNA 相关的节点	12.1.5	O
3	TNA 地址 Sub-TLV	规定 TNA 地址	12.1.4	M

在单一路由域内的多 RCD 环境中，可以采用 TNA 地址顶级 TLV 交换连接端点的可达性信息。这个 TLV 源自类型 1 TELSA，遵循类型 10 不透明 LSA 的泛洪规则。该 TLV 由某个 RCD 的 RC 向其他 RCD 的 RC 广播。

TNA 地址 TLV(Type 32768，长度可变)的格式如下：



Node_ID Sub-TLV 表示紧随其后的 TNA 地址 TLV(s)中的 TNA 地址所附着的节点。Node_ID TLV 应出现在 TNA 地址 TLV 之前。后续的 Node_ID Sub-TLV 表示一个节点的 TNA 地址列表的结束，并表示紧随其后的 TNA Address TLV(s)所附着的节点。

一个顶级 TLV 对于相同的节点可能包含多个 Node_ID Sub-TLV。一个 RC 可能发送多个针对同一节点的 Node_ID Sub-TLV。

13 E-NNI 安全和日志(可选)

13.1 安全

由于光传送网络呼叫和连接承载高速数据并消耗大量网络资源，因此需要一定的安全措施保证传送网不受到危及控制平面的攻击，并避免未经鉴权使用网络资源或者获得网络配置信息等。

E-NNI的安全功能需求见GB/T 21645.5—2012中9.4。满足这些需求的安全机制实现细节见GB/T 21645.5—2012中9.5和9.6,其中描述了对经过控制平面接口交换信息的实体进行鉴权的机制、保证信息的完整性、信息的保密机制等可选机制。

13.2 日志

支持安全扩展的E-NNI实现还应支持OIF SLG-01.0规定的控制平面日志功能。日志功能包括配置、控制和安全的记录。不支持安全扩展的实现也可以支持日志功能。

附 录 A
（规范性附录）
IETF GMPLS 的域间信令技术

A.1 域间信令要求

A.1.1 域间信令模型

在 GMPLS 中，域表示在同一个地址管理空间或者路由计算控制下的一组网元，划分多域的目的包括管理、可扩展性、保护域的组建等。

IETF 跨域建立 TE LSP 存在三种信令机制，包括连续域间信令、拼接域间信令和嵌套域间信令：

a） 连续域间信令

在连续域间信令模型中，通过 IETF RFC 3209 和 IETF RFC 3473 中描述的 RSVP-TE 信令方法和过程建立一条端到端的跨越多个域的 LSP，在 LSP 经过的每个节点上，会话（Session）和 LSP ID 都是相同的，这与单域中的做法相同。

b） 拼接域间信令

拼接域间信令是把多个域内的 LSP 段（Segement）拼接成一条跨域的端到端的 LSP。LSP 段可以是单域内的一条 LSP，也可以是一条跨越多域的 LSP。在数据平面，通过拼接，形成一条端到端的 LSP。在控制平面中，LSP 段是作为独立的 TE LSP 通过信令建立的，不同的 LSP 段的会话（Session）是不相同的，并且与跨域的 LSP 的会话也没有关系。

TE LSP 段的建立可由前一条 TE LSP 段或者由管理操作触发。LSP 段也可以作为 TE 连路被管理和发布。

c） 嵌套域间信令

嵌套域间信令是把一条或多条 TE LSP 嵌套在一个嵌套 LSP 中。在一个 TE LSP 中承载另一个 LSP 的技术，称为 LSP 嵌套。嵌套 LSP 是 GMPLS 的基本功能，IETF RFC 4206 中详细规定了嵌套 LSP 的概念。嵌套 LSP（H-LSP）可以作为 TE 链路进行扩散。嵌套 LSP 可以用于支持域间 TE LSP，嵌套 LSP 可以在一个域内任意 LSR 之间建立连接，嵌套 LSP 的入口和出口可以是同一个域的一对边界节点，或者同一个域的一对域内节点。

建立嵌套 LSP 的信令触发条件包括：

1） 收到它承载的一条 TE LSP 的信令请求；

2） 管理操作，用于为嵌套 LSP 预配置需要经过的网络。

嵌套和被嵌套 LSP 的属性（包括带宽）的映射关系（继承规则），可以静态配置。对于按需的嵌套 LSP 可以根据嵌套 LSP 的属性动态继承。在动态情况下，从嵌套 LSP 属性的继承可以由本地或者域的策略规则完成。

嵌套 LSP 可以跨越多个域或者域的一些部分。在一条嵌套 LSP 经过多个域时，该 LSP 不能被发布为 TE 链路。路由协议不会通过嵌套 LSP 交换信息，LSP 不会用于创建路由器之间的路由邻接关系。

d） 混合方式

在建立一个端到端跨域 TE LSP 时，可以将上述信令方式混合使用。在采用混合方式时，可通过 RRO 对象报告整个路径上使用的各种信令方式。

建立跨域的端到端的 TE LSP，可以采用上述的一种或多种信令方式。信令方式的选择由首节点

的策略以及每个域边界节点的策略决定。通过对 LSP_ATTRIBUTES 对象进行扩展，携带相关的信息来识别信令方式。

A.1.2 域间呼叫模型

呼叫是端节点之间，也可能是关键传送节点（如网络边界节点）之间的一种关联关系，一个呼叫用来支持一个用户业务实例。呼叫本身不提供传送用户业务的连通性，仅用来构建后续一个或多个 LSP 之间的关联关系。呼叫是端节点之间的一种合约关系，用于简单、方便地管理一组提供端到端的数据传送的连接。呼叫的建立包括：策略验证、鉴权、网络安全认证以及端节点之间的能力协商等处理。

呼叫可以用来管理一组连接，这组连接共同为客户提供端到端的网络连接服务。一个呼叫可以与一个或多个连接关联，也可以不与任何连接关联；一个连接可以与一个呼叫关联，也可以不与任何呼叫关联，但不能与多个呼叫关联。因此，呼叫和连接之间要在逻辑上完全分离。呼叫的建立和维护可以独立于关联的 LSP 连接的建立和维护处理。

在多域场景下，同一条业务需要经过多个域，业务经过的每个域都需要处理该业务对应的呼叫。因此，在呼叫发起方发起呼叫时，可以在呼叫建立请求消息中显式指定各域需要处理该呼叫的节点及其顺序，并将呼叫建立请求消息发往下一个处理该呼叫的节点。

下一个处理该呼叫的节点处理该呼叫之后，可以根据消息中指定的处理该呼叫的节点信息继续转发，从而每个指定需要处理该呼叫的节点都可以接收到该消息，并进行呼叫处理。如果处理呼叫的全部节点同意建立该呼叫，则从呼叫目的地依次返回呼叫建立成功响应，呼叫发起方收到成功响应，呼叫建立成功。否则，只要有一个呼叫处理节点不允许建立该呼叫，则立即返回呼叫建立失败响应，呼叫发起方收到失败响应，呼叫建立失败。

A.1.3 域间保护恢复

域间保护恢复存在多种机制，IETF RFC4872 中规范了端到端恢复、本地保护（如 FRR）和区段恢复。跨域的端到端保护恢复机制，包括：

a) 单向 1+1 保护：建立两条端到端的相互分离的 LSP，一条为工作路径，一条为保护路径，在发端同时往两条 LSP 上发送业务，在收端从工作路径上选收业务。当工作路径发生故障时，只检测到故障的端点切换到保护路径上，从保护路径上接收业务，即只实施单端倒换。

b) 双向 1+1 保护：与单向 1+1 保护的区别是，一端检测到故障时，会通知另一端进行协商，两端都倒换到保护路径上，即实施双端倒换。

c) 1：1 保护：建立两条端到端的相互分离的 LSP，一条为工作路径，一条为保护路径。正常情况下，由工作路径传送业务，保护路径可传送额外业务。当工作路径故障时，两端都切换到保护路径上传送业务，额外业务中断。

d) 1：N 保护：由一条保护路径为 N（N>1）条工作路径提供保护，这些工作路径相互分离，同时与保护路径也是分离的。正常情况下，保护路径可以传送额外业务。在工作路径故障时，两端都切换到保护路径上传送业务，额外业务中断。

e) 预计划重路由（Pre-planned rerouting）：预先建立一条与工作路径分离的保护路径，为该保护路径预留好资源，但数据平面没有建立好，因此保护路径不能传送额外业务，其资源也可被其他 LSP 占用。在工作路径故障时，需要通过传递信令来将保护路径的传送平面建好，然后将业务切换到保护路径上，恢复业务。

f) 共享网状网恢复：由一条保护路径为多条相互分离的工作路径提供保护，保护路径与这些工作路径也是分离的。为保护路径事先预留好资源，但没有建立好数据平面。在工作路径故障时，需要通过传递信令来将保护路径的传送平面建好，然后将业务切换到保护路径上，恢复业务。

g) LSP 重路由：在工作路径故障后，建立一条端到端的新路径，然后将业务切换到新路径上，从而恢复业务。在重路由时，新路径可以重用老路径的资源。

IETF RFC4873 中定义了基于 GMPLS 的区段恢复(segment recovery)机制，可用于保护节点故障、链路故障、LSP 的多点故障。在区段恢复中，如果主、备路径存在松散跳，则需要关注主、备路径的分离问题。如果路径采用的是增量计算方式，则可以采用 IETF RFC4874 中描述的路由排斥机制来达到路径分离的目的；否则，需要采用同时计算两条分离路径的技术，如 IETF RFC4655 中的 PCE 技术。区段恢复能够在单域内为 LSP 提供较好的保护，如果是跨越多域且这些域由不同的厂家的设备组成，要提供跨域的区段恢复就比较困难了，那样会涉及复杂的信令互通等问题。

对于分组交换网络，还可以采用 MPLS-TE FRR 技术实现域内链路和节点、域间链路和域边界节点的保护。

基于 GMPLS 的域间保护恢复应满足以下要求：

a) 域间端到端保护恢复机制应至少支持 1+1 保护和 LSP 重路由。

b) 端到端保护恢复应满足工作和保护恢复路由分离的要求，应支持节点分离、链路分离、SRLG 分离和域分离。域分离是指选择的路径只有相同的域出口和域入口，可以提供更强的故障恢复能力。

c) 端到端保护恢复应支持返回功能，即在工作路径上的故障消失后，业务切换回工作路径，重新从工作路径上传送业务。如果是采用重路由方式进行的恢复，则在业务恢复到原路径后，应删除恢复路径，释放资源。在返回的过程中应尽量减少业务中断和重配置。

d) 跨域 LSP 可以在每个域内采用不同的保护恢复机制。例如域边界的保护由域间链路提供，域内通过区段恢复提供等。

A.1.4 跨域的路径重优化

重优化是把 TE LSP 从一个路由移到另一个路由。通常采用先建后拆的技术来保证业务损伤尽可能的小。

对于跨域 LSP 的优化可以提供两种方式：

a) 只在单个域内进行优化，这种优化不改变域边界节点；

b) 端到端的优化，即从首节点重新计算出一条新的路径到末节点，这种优化可能会改变 LSP 经过的域或者域边界节点。

优化可以通过操作人员的请求、定时器、网络资源的可用信息改变或者 LSP 的一些运作因素的改变(例如带宽)触发。

A.1.5 LSP 建立失败处理

当一个域间 LSP 在除了首域之外的其他域建立失败时，可以采用以下方法报告和重试 LSP：

a) 在失败的域进行重试。重试可以由域内的节点进行，或者可以返回到域边界节点进行。

b) 如果该失败无法在本域内解决(例如没有合适的替代路由，或者域策略不允许重路由，或者 PATH 消息禁止该操作)，应向上一个域或者首节点域报告该失败。后续的修复尝试可以由上游域执行，但是条件是应向上游提供足够的建立失败和修复尝试失败的信息(Crankback)。进一步尝试可以把失败信息作为路由计算的约束条件，或者在信令协议中作为排斥路由信息。

A.2 IETF GMPLS 的域间信令协议扩展

A.2.1 域间信令协议

建立跨域的端到端的 TE LSP，可以采用上一章所述的一种或多种信令方式，信令方式的选择由首

节点的策略以及每个域边界节点的策略决定。LSP 段和 H-LSP 可以预先配置，也可以由信令动态驱动建立。信令方式的识别可通过对 LSP_ATTRIBUTES 对象进行扩展，携带相关的信息来完成。

A.2.1.1 域边界节点的处理

跨域 LSP 采用哪种信令方式主要由经过的节点（一般是域边界节点）所支持的方式来决定，也依赖于首节点的信令参数。

域边界节点接收到跨域 LSP 建立请求时，需要做如下处理：

a) 进行本地策略校验：如果校验失败，应发出 PathErr 消息报错。

b) 检查信令方式：如果首节点限制了信令方式，则应按首节点指定的信令方式建立 LSP；如果首节点没有显示指定信令方式，则按本地的配置或策略选择相应的信令方式。如果不支持首节点指定的信令方式，则应发出 PathErr 消息报错。

c) 对 ERO 进行处理，具体请见 A.2.1.2。

d) 计算跨越本域的路径，即扩展 ERO。如果计算失败，应发出 PathErr 消息报错。

e) 当从本域或下一个域收到报错的 PathErr 消息时，域边界节点可以进行 Crankback（回溯）处理。如果不进行 Crankback 或者 Crankback 也失败，则域边界节点应继续进行 PathErr 处理，其处理过程应遵循 IETF RFC3209 和 IETF RFC3473 中的定义。具体见 A.2.1.3。

f) 进行 RRO 的处理，具体见 A.2.1.4。

A.2.1.2 跨域 ER 的处理

域边界节点接收到跨域 LSP 建立请求时，需要对其中的 ERO 进行处理，包括：

a) 如果有相关的 ERO 策略，则应进行策略处理。例如，一种策略出于保密性考虑，如果 ERO 中包含本域内的节点，则拒绝。如果策略处理失败，则应发出 PathErr 消息报错，或者根据本地配置的策略进行处理——将该 Path 消息丢弃或出于安全考虑，返回另外一个错误码。

b) 对嵌套 LSP 和拼接 LSP，在边界节点，查找 H-LSP 或触发建立一条新的 H-LSP，应遵循 IETF RFC 4206 中的定义的过程。

c) 如果 ERO 中子对象标识的 TE LINK 是 H-LSP 或 LSP 段，则不能使用连续信令方式，应使用嵌套或拼接方式。如果冲突，则应发出 PathErr 消息报错，或者根据本地配置的策略进行处理——将该 Path 消息丢弃或出于安全考虑，返回另外一个错误码。

d) 如果 ERO 中指定的下一跳是松散节点，则应计算到达下一跳的路径，扩展 ERO，计算得到的路径也可能包含松散节点。

e) 如果没有 ERO 子对象，则应该将目的节点作为下一跳的松散路由，按照 d）进行处理。

f) 遇到其他任何处理 ERO 失败的情况，则都应该按照 IETF RFC3209 和 IETF RFC3473 中的定义过程发出 PathErr 消息报错，域边界节点也可以根据本地配置的策略进行处理，如，将 Path 消息丢弃或出于安全考虑，返回另外一个错误码。

A.2.1.3 LSP 建立失败和回溯

在跨域 LSP 建立失败后，需要通过 PathErr 消息向首节点通告错误信息，这样 PathErr 消息就需要跨越多个域，就会涉及到域内信息的保密问题，这可以通过策略进行控制。域边界节点可根据本地的策略，对通告的错误信息进行控制，如对 PathErr 消息中的信息进行替换。LSP 建立失败的处理应满足：

a) 域边界节点不能阻止 PathErr 消息进一步向上游节点通告，除非按后面描述的方式进行重路由；

b) 除域边界节点外，其他节点不能修改 PathErr 消息的内容；

c) 域边界节点一般应不修改 PathErr 消息的内容，除非出于机密、安全考虑。

在收到建立 LSP 失败的 PathErr 消息后，如果本地策略和 Path 消息中的相关参数允许，则域边界节点可以暂缓将 PathErr 消息进一步向上游通告，进行回溯（Crankback），尝试重新建立 LSP。如果建立 LSP 成功，则域边界节点应丢掉暂缓通告的 PathErr 消息。如果尝试建立 LSP 仍然失败，则域边界节点应继续向首节点方向通告暂缓的 PathErr 消息，但可以根据本地的策略对 PathErr 消息的内容进行修改，如出于安全、保密考虑，改变其携带的错误码或者将报错的节点地址修改为域边界节点的地址。

A.2.1.4 跨域 RRO 的处理

RRO 的主要作用是进行环路检测和记录 LSP 经过的每一跳。但在跨域时，出于信息保密性，可能有些内部信息（如节点 ID）不允许跨域传输。边界节点可以根据本地的策略对 RRO 中携带的信息进行过滤或修改，如将经过本域的所有路径信息替换成本域的标识 ID（如 AS 号）或只让 RRO 中包含本域的边界节点信息。

A.2.1.5 Notify 消息的处理

Notify 消息是直接发向目的地，而不是采用 hop-by-hop 的方式进行传送，以加快对错误信息的通告。如果域边界节点需要看到该消息（如通过它来提供保护倒换），则可将 Path 和 Resv 消息中的 Notify Request 对象修改为携带域边界节点的地址，该处理过程应遵循 IETF RFC3473 中定义的规范。

出于安全性和保密的目的，域边界节点可根据本地策略对 Notify 消息进行过滤或修改。

A.2.1.6 控制下游节点选择信令方式

通过信令扩展，可以支持建立跨域 LSP 时控制下游节点选择信令方式。在 LSP_ATTRIBUTES 对象的 Attributes Flags TLV 中增加一个新的标志位 Contiguous LSP 标志，用于首节点指示是否需要采用端到端的连续信令方式建立跨域 LSP。如果设置了该标志位，则域边界节点应使用连续方式，不能选择嵌套或拼接方式。如果没有设置该标志位，则可以选择嵌套或拼接方式。除首节点外，其他任何节点都不能修改该标志位。

中间节点接收到设置了 Contiguous LSP 标志位的 Path 消息，如果该节点能识别该对象及该标志位，并且支持连续 LSP 方式，则应以连续信令方式进行处理。如果该节点是域边界节点或者该节点在 ERO 中增加了一个松散跳（loose hop），则它应在相应的 Resv 消息的 RRO 对象的 RRO Attributes 子对象中设置"Contiguous LSP signaled"标志位。

如果中间节点支持 LSP_ATTRIBUTES 对象，但不支持 Attributes Flags TLV 或支持 Attributes Flags TLV 但不支持 Contiguous LSP 标志位，则应毫无修改地转发该对象。

A.2.1.7 域内拓扑信息安全性

RSVP-TE Path Key 子对象（其格式应遵循 IETF RFC5553 中的定义）用于保证域内的拓扑信息不会扩散到其他域。对该子对象的处理包括：

a) ERO 中携带 Path Key Subobject（PKS）
   1) 如果发现 Path 消息携带的 ERO 中的第一个子对象是 PKS，则应发出 PathErr 消息报错。
   2) 如果发现 ERO 中的下一跳子对象是 PKS，则应能够通过 PKS 转换得到机密路径信息（CPS，即不希望暴露给其他域的路径）。如何从 PKS 转换得到 CPS，可以根据本地的策略和网络的实现方法采取合适的方式。在得到 CPS 后，相应的路径信息应插入到 ERO 中，替换掉 PKS。余下的处理应遵循 IETF RFC3209 中的规定。如果不能通过 PKS 转换得到相应的 CPS，则应产生 PathErr 消息报错。
   3) 如果节点不支持 PKS 子对象，则应发出 PathErr 消息报错。

b) RRO 中携带 PKS

当信令消息跨越域边界时，如果需要隐藏 LSP 经过本域的路径信息(CPS)，则可以将 RRO 中的该段路径信息使用 PKS 替换。如果是 Resv 消息，则 RRO 中的 PKS 应是对应的 Path 消息中的 ERO 中携带的那个 PKS。在 Path 消息中，PKS 应能够标识由哪个节点来转换 CPS，并且提供 Path Key，根据该 Path Key 可以得到 CPS 路径。

A.2.2 排斥路由约束扩展

为了在多域网络中建立两条分离路径(如保护恢复、双归属的情况)，需要扩展信令来支持排斥路由约束。排斥路由分两种类型：

a) 在 LSP 的整条路径上排除某些抽象节点或资源，这种排斥方式称为排斥路由表(Exclude Route List)。通过增加一个 RSVP 对象 EXCLUDE_ROUTE 对象(XRO)来携带排斥的信息。

b) 在一个 ERO 里的两个抽象节点之间排除某些抽象节点或资源，这种排斥方式称为显式排斥路由(Explicit Exclusion Route)。通过增加一个 ERO 子对象 Explicit Exclusion Route 子对象(EXRS)来携带两个节点间要排斥的信息。

A.2.2.1 EXCLUDE_ROUTE 对象(XRO)

通过 XRO 对象携带多个子对象来标识需要排斥的节点、端口、AS 或 SRLG。XRO 应能够支持：

a) 排斥节点、端口、AS、SRLG；

b) 排斥(应避开所指定的节点/端口/AS/SRLG)、尽量排斥(尽量避开所指定的节点/端口/AS/SRLG)。

对 XRO 的处理应遵循如下规则：

a) 收到 Path 消息时，检测本节点是否是 XRO 中应排除的对象。如果是，则应产生 PathErr 消息报错。

b) 如果 XRO 中含有 SRLG，则应检查本节点使用的资源的 SRLG 属性和 XRO 中强制排除的 SRLG 是否相同，如果相同，则应产生 PathErr 消息报错。

c) 校验子对象的一致性。发现不一致时，应产生 PathErr 消息报错。

d) ERO 和 XRO 中的子对象不应相互冲突，在冲突时应按如下原则处理：

1) 如果子对象在 XRO 中是应排斥属性，其又在 ERO(应包含)中，则应拒绝该 Path 消息，产生 PathErr 消息报错；

2) 如果子对象在 XRO 中是尽量排斥属性，其又在 ERO(应包含)中，则可以拒绝该 Path 消息，也可以忽略 XRO 中的该子对象，继续 LSP 的建立。

e) 当选择下一跳或扩展路由时，节点：

1) 不能选择被 XRO 中指定为应排斥的节点或抽象节点，应尽量不选择被 XRO 中指定为尽量排斥的节点或抽象节点；

2) 不能选择被 XRO 中指定为应排斥的 SRLG 所代表的链路、节点或资源，应尽量不选择被 XRO 中指定为尽量排斥的 SRLG 所代表的链路、节点或资源。

f) 如果 XRO 超大或过于复杂以至本节点不能解析和处理，则应产生 PathErr 消息报错，收到该错误信息的首节点应减少 XRO 的复杂性或绕开报错的节点。

g) 如果本节点不支持 XRO，则应毫无改变地转发该对象，不需要做其他的处理。

h) 如果本节点支持 XRO，但不支持里面的某个或某些子对象，则应忽略对这些子对象的处理。

i) 在转发 Path 消息时，节点还可做如下的处理：

1) 如果没有 XRO 对象，则可以根据需要，增加 XRO 对象；

2) 如果有 XRO 对象，但本节点确定下一跳不再需要这些信息，则可以删除该 XRO 对象；

3) 如果有 XRO 对象，本节点可以对该对象的内容进行修改，增加或删除子对象。

上述操作不能导致 XRO 和 ERO 发生冲突。

A.2.2.2 Explicit Exclusion Route 子对象(EXRS)

通过 EXRS 子对象来标识 ERO 中某两个抽象节点间要排除的节点(如显示节点、抽象节点、AS)或资源(如链路、标签)。EXRS 中的子对象和 XRO 中的一致，对应的处理也一致。

A.2.2.3 同时存在 XRO 和 EXRS 的处理

如果同时存在 XRO 和 EXRS，节点在扩展路由时，应按如下规则处理：

a) 应排除 XRO 和 EXRS 中列出的所有 SRLG、节点、链路和资源；

b) 如果某种排斥条件在 XRO 和 EXRS 中都存在，则应按排斥条件更严格的处理，如，XRO 中要求应排斥某个节点，而在 EXRS 中要求尽量排斥该节点，则在扩展路由时，应排斥该节点。

A.2.3 域间呼叫协议扩展

A.2.3.1 Notify 消息的扩展

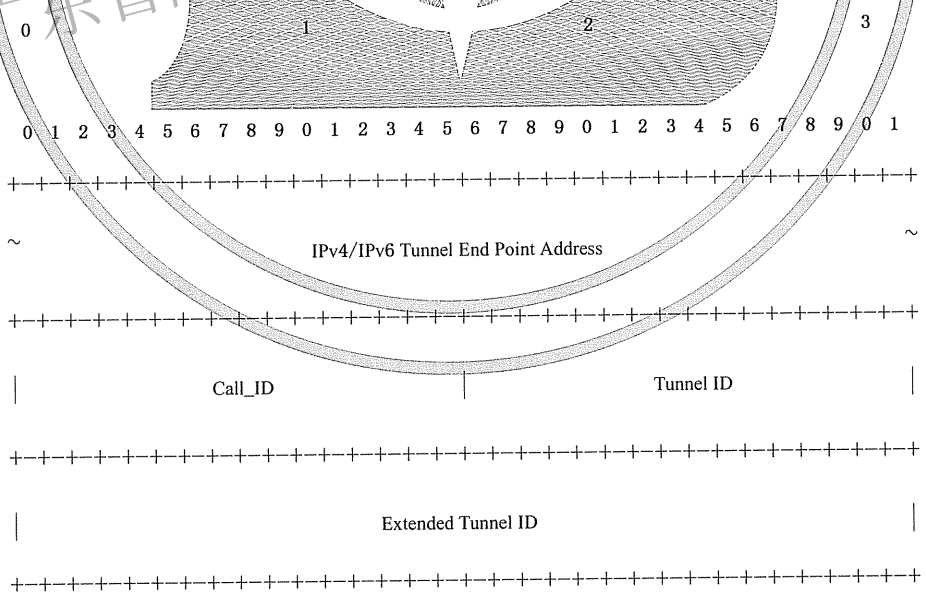
IETF RFC4974 规定了支持呼叫的信令扩展。呼叫的建立和拆除与连接的建立和拆除分离，GMPLS 不支持同时建立呼叫和连接。通过对 Notify 消息进行扩展，来支持呼叫的建立和拆除。

a) 扩展支持携带呼叫 ID(Call ID)

呼叫 ID 用于唯一标识一个呼叫，分为 Long Call ID(长呼叫 ID)和 Short Call ID(短呼叫 ID)。

Long Call ID 用于标识全局唯一的呼叫，通过 Notify 消息中的 SESSION_ATTRIBUTE 对象里的“Session Name”字段携带。

Short Call ID 用于标识连接(LSP)属于哪个呼叫，在 SESSION 对象中增加 Call_ID 字段携带，其格式如下：

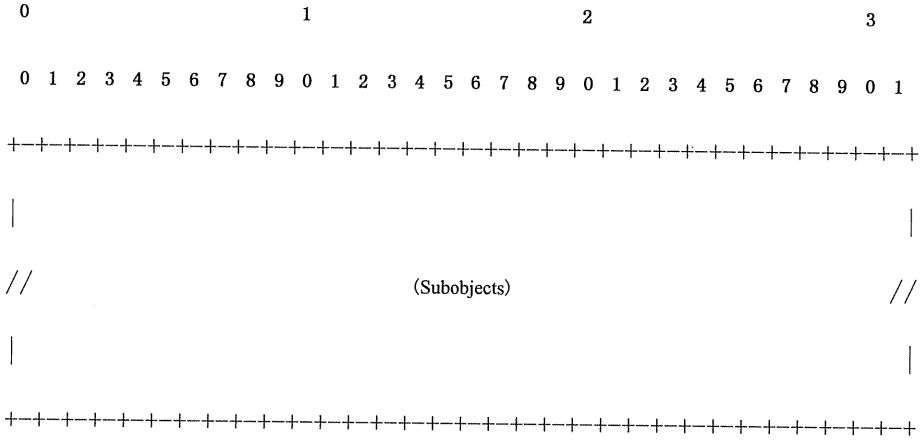


用于建立和管理呼叫的 Notify 消息中的 SESSION 对象里的 Call_ID 用于标识哪些 LSP 属于该呼叫，其值不能为 0；如果 LSP 属于某个呼叫，则该 LSP 中携带的 SESSION 对象里的 Call_ID 应与该呼叫的 SESSION 对象里的 Call_ID 一致；如果 LSP 不属于任何呼叫，则其携带的 SESSION 对象里的 Call_ID 字段应设置为 0。

b) 增加 LINK_CAPABILITY 对象

为了支持在呼叫建立时，交换连接呼叫发起节点(或呼叫终结节点)与网络的链路的能力(包括链路的IP地址、接口索引、最大预留带宽、接口的交换能力等)，增加LINK_CAPABILITY对象，其格式如下：

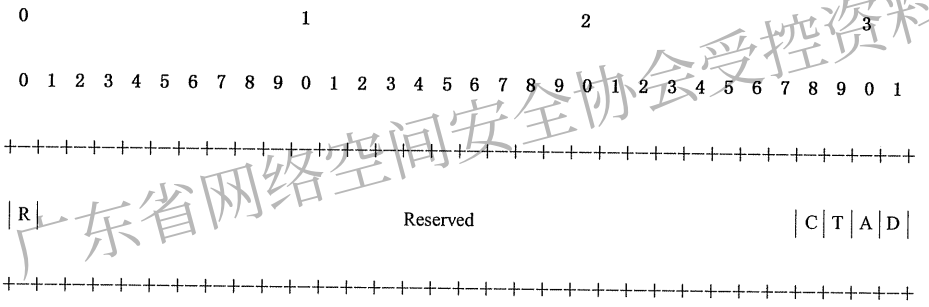
Class-Num=133(form 10bbbbbb)，c-Type=1



支持的子对象参见IETF RFC4974。

c) 在ADMIN_STATUS对象中增加呼叫管理位

在ADMIN_STATUS对象中增加呼叫管理比特位，用于标识携带该对象的Notify消息是用于呼叫的控制和管理的，其格式如下：



Reflect (R)：1 bit——其含义参见IETF RFC3471；

Testing (T)：1 bit——其含义参见IETF RFC3471；

Administratively down (A)：1 bit——其含义参见IETF RFC3471；

Deletion in progress (D)：1 bit——其含义参见IETF RFC3471；

Call Management (C)：1 bit——当消息用于呼叫的控制和管理时，设置该位。

d) 对Notify消息的扩展

在Notify消息中增加可选对象LINK_CAPABILITY，以支持交换链路的能力；在notify session中增加SESSION_ATTRIBUTE对象，用于携带Long Call ID；一个Notify消息中可通过携带〈notify session list〉来支持同时创建多个呼叫；在ADMIN_STATUS对象中增加呼叫管理比特位，用于标识该Notify消息是用于呼叫的控制和管理的；每个呼叫可以携带ERO/RRO对象，具体格式见IETF RFC3209、IETF RFC3473、IETF RFC3477，前者用于指定需要处理该呼叫的节点编号，后者沿途记录处理该呼叫的节点编号。扩展后的Notify消息格式如下：

〈Notify message〉::= 〈Common Header〉[〈INTEGRITY〉]
[[〈MESSAGE_ID_ACK〉|〈MESSAGE_ID_NACK〉]...]
[〈MESSAGE_ID〉]
〈ERROR_SPEC〉
〈notify session list〉

〈notify session list〉::=[〈notify session list〉]〈notify session〉

〈notify session〉 ::= 〈SESSION〉[〈ADMIN_STATUS〉]
[〈POLICY_DATA〉...]
[〈LINK_CAPABILITY〉]
[〈SESSION_ATTRIBUTE〉]
[〈ERO〉]
[〈RRO〉]
[〈sender descriptor〉 | 〈flow descriptor〉]

〈sender descriptor〉 ::= 见 IETF RFC3473

〈flow descriptor〉 ::= 见 IETF RFC3473

A.2.3.2 呼叫和连接的建立过程

呼叫和连接的建议过程如下：

a) 呼叫和连接的建立要求

1) 呼叫建立时，不需要与任何连接关联。在呼叫建立后，可以将连接加入到呼叫中；
2) 一个连接可以被加入到一个已经存在的呼叫，该呼叫可以是已经有一个或者多个连接与它关联，或者没有连接与它关联；
3) 连接建立时，可以不必与任何呼叫关联。

b) 呼叫的建立

如图 A.1 所示，建立呼叫时，Ingress(发起端)向 Egress(终结端)发出 Notify 消息，发起呼叫建立请求。Notify 消息中需要携带如下信息：

1) 将 ADMIN_STATUS 对象里的 R、C 比特置位，表示该 Notify 消息需要回送响应，且为呼叫管理消息；
2) 在 SESSION_ATTRIBUTE 对象中的 Session Name 里携带 Long Call ID；
3) 〈notify session list〉中应至少有一个 SESSION 对象(即至少创建一个呼叫)。在 SESSION 中携带非 0 的 Call_ID，其值将作为与该呼叫关联的所有 LSP 的 Short Call ID；Tunnel_ID 应设置为 0；Extended_Tunnel_ID 可以设置为 0 或者设置为 Ingress 节点的地址；
4) SENDER_TEMPLATE 对象中包含呼叫发起者的 IP 地址，LSP_ID 填为 0；
5) 在 LINK_CAPABILITY 对象中携带 Ingress 端链路能力及 Ingress 节点的本地策略(可选)；
6) ERROR_SPEC 对象中的错误码设置为 0。

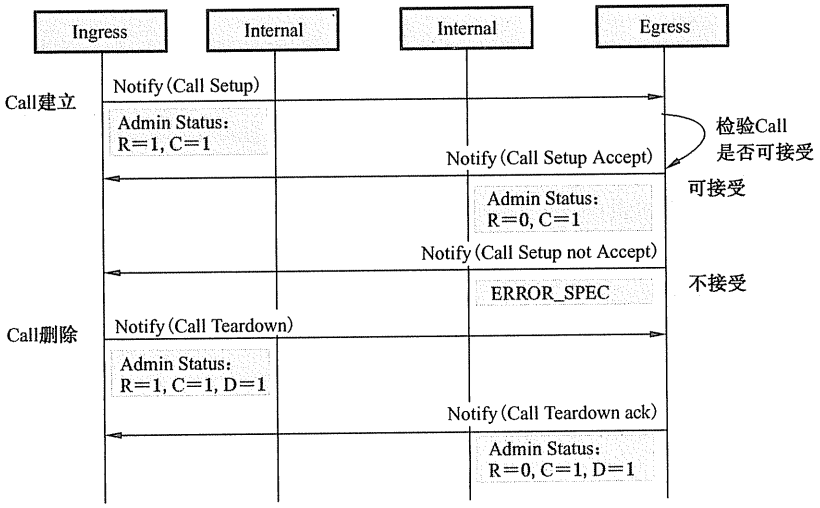


图 A.1 呼叫建立过程

Egress 节点收到 Notify 消息后，进行安全、认证和策略等校验。如果校验通过，则回送 Notify 消息接受呼叫的建立，该 Notify 消息中的 ADMIN_STATUS 对象里的 C 比特置位，其他比特位都置为零，且 Notify 消息中可选择地携带 Egress 节点的链路能力；如果校验不通过，则在 Notify 消息中回送相应的错误码。

为了确保 Notify 消息的可靠传送，需要在 Notify 消息中携带 Message ID，通过 Message ID 机制进行保证（具体参见 IETF RFC2961）。如果 Ingress 在一定的时间内没有收到 Message ID 确认，则要重发该 Notify 消息，如果重发超过了一定的次数（根据本地的策略配置）仍没有收到确认，则宣告呼叫建立失败，并发出呼叫删除请求。

如果虽然收到了 Message ID 确认，但没有收到进行呼叫响应的 Notify 消息，则 Ingress 根据配置的呼叫尝试次数，重新发出呼叫建立请求。在所有尝试失败后，宣告呼叫建立失败，并发出呼叫删除请求。

如果 Egress 节点根据本地的策略，不接受呼叫建立请求，则应在发出的 Notify 响应消息中携带相应的错误码。

c) 将一个连接加入到呼叫中。

可以将连接（LSP）加入到一个已建立好的呼叫中。由于属于同一呼叫的 LSP 在 SESSION 对象中具有相同的 Short Call ID，建立呼叫时，Notify 消息中携带的 SESSION 对象中的 CALL_ID 与相关联的 LSP 中的 Short Call ID 是一样的，因此关联的 LSP 在建立时，就自动加入到了呼叫中了。不与呼叫关联的 LSP 的 Short Call ID 应设置为 0；与呼叫关联的 LSP 的 Short Call ID 不能设置为 0。从 Ingress 和 Egress 节点都可将 LSP 加入到呼叫。

d) 连接的建立

连接的建立应该与呼叫无关。如果 LSP 不与呼叫关联，在建立时，SESSION 对象中的呼叫 ID 应设置为 0，SESSION_ATTRIBUTE 对象中的 Session Name 字段应作为 Session 的名称进行解释。用于 LSP 控制的消息（包括 Notify 消息）中携带的 ADMIN_STATUS 对象里的 C 比特应设置为 0。

e) 呼叫冲突处理

由于呼叫的两个端点可能使用相同的 Long Call ID 同时发起呼叫建立请求，从而产生冲突。如果发出了呼叫建立请求，在还没有收到呼叫建立响应前，收到了具有相同的 Long Call ID 的呼叫建立请求，则按如下规则处理：

1) 如果本节点的地址比收到的呼叫建立请求的源地址大，则应丢弃收到的呼叫建立请求消息，继续等待本节点发起的呼叫建立请求的响应；

2) 如果本节点的地址比收到的呼叫建立请求的源地址小，则应放弃本节点发起的呼叫建立请求，对收到的呼叫建立请求进行响应。

如果收到的呼叫建立请求的源、目的地址及 Long Call ID 都与本节点一个已经存在的呼叫的目的、源地址及 Long Call ID 相同，则应发出 Notify 消息，携带相应的错误码报错，已存在的呼叫应不受任何影响。

当具有相同的源、目的节点，但 Long Call ID 不同的两个呼叫使用了相同的 Short Call ID 同时发起建立请求时，也会产生冲突，应按照如下规则处理：

1) 如果本节点的地址比收到的呼叫建立请求的源地址大，则应发出 Notify 消息，携带相应的错误码报错；

2) 如果本节点的地址比收到的呼叫建立请求的源地址小，则接受呼叫建立请求，做相应的处理。随后，本节点会收到一条拒绝呼叫建立的消息[如 a）中描述的]，那时，本节点可以选择一个新的 Short Call ID，重新发起呼叫建立请求。

A.2.3.3 呼叫和连接的删除

对不属于任何呼叫的LSP的删除与常规的LSP删除一样，应遵循IETF RFC 3473中定义的过程。下面主要描述与呼叫相关的删除：

a) 从一个呼叫中删除一个连接。从一个呼叫中删除一条LSP，按照IETF RFC 3473中定义的过程删除该LSP即可。不能从一个呼叫中移出一条LSP，但不删除该LSP，因为不删除该LSP的话，就应将它的Short Call ID修改为0，而修改Short Call ID就应修改SESSION对象，但修改SESSION对象是不允许的。

b) 从一个呼叫中删除最后一条连接。在删除与呼叫相关联的最后一条LSP后，不一定就意味着需要将该呼叫也删除，因为呼叫可以没有相关联的LSP而存在，这种呼叫如何处理，参见下面c)的描述。

c) 删除没有关联连接的呼叫。与一个呼叫相关联的所有LSP都被删除后，可以删除该呼叫，也可以继续保留该呼叫以便以后使用，根据本地的策略而定。通过发送Notify消息来删除呼叫。在删除呼叫请求的Notify消息中，ADMIN_STATUS对象的R、D、C比特位设置为1，在Notify响应消息中，ADMIN_STATUS对象的D、C比特位设置为1，其他位都设置为0。在没有收到Message ID Ack或者没有收到Notify响应消息时，呼叫删除发起者应进行重试，在所有的重试都失败后，则可以认为该呼叫被删除，但该节点应至少在5个Notify刷新周期内不能将该呼叫使用的Long Call ID和Short Call ID重新用于其他的呼叫。

d) 删除还存在关联的LSP的呼叫。如果收到删除呼叫请求的Notify消息，发现该呼叫还存在与它关联的LSP，则应回送Notify响应消息报错，该呼叫不能被删除或改变。

e) 从Egress发起删除呼叫。可以从Ingress发起删除呼叫，也可以从Egress发起删除，删除过程与上面的描述一致。当Ingress和Egress同时发起删除同一个呼叫时，则收到的删除呼叫请求的Notify消息可以作为删除呼叫的响应消息处理，但对该请求的Notify消息，还应回Notify消息进行响应。在收到删除呼叫的Notify请求消息后，如果发现相应的呼叫已不存在，则也要回响应。

A.2.3.4 控制平面的生存性

A.2.3.4.1 呼叫的刷新。呼叫的Ingress和Egress应周期性地发送Notify消息，以对呼叫的状态进行维护。刷新周全由本地策略而定，建议刷新周期是与该呼叫相关联的LSP中刷新周期最短的周期值的两倍。在没有关联的LSP时，建议呼叫的刷新周期为不小于1 min。除LINK_CAPABILITY对象外，Notify刷新消息与第一次新建呼叫时的Notify消息一样。

A.2.3.4.2 呼叫刷新中断。当收不到Notify刷新请求消息时，本节点可以主动发出Notify刷新请求消息。如果收不到Notify刷新响应或者收不到Message ID Ack，则可以断定远端已不可达，此时，可以删除与呼叫相关联的所有LSP和该呼叫，根据本地策略而定。

A.2.3.4.3 节点重启。在Ingress或Egress发生重启后，如果没有保存状态，则可以通过从邻居节点和呼叫的远端节点重新学习，恢复LSP的状态和呼叫的状态。

附 录 B
（资料性附录）
IETF GMPLS 的域间路由技术

B.1 域间路由基本要求

基于 IETF GMPLS 的域间路由主要采用 PCE 技术。路径计算单元 PCE(Path Computation Element)是一种能够基于网络视图和限制条件完成网络路由计算的实体。PCE 实体可以位于网元内部或者位于网络之外的服务器中。路径计算请求者(PCC,Path Computation Client)向 PCE 发送路径计算请求,PCE 向 PCC 返回路径计算结果。路径计算可以由单个 PCE(集中架构)完成,也可以由多个 PCE 协同完成(分布式架构)。PCE 之间使用 PCEP(PCE 通讯协议)通信。

在多域的应用场景中,域之间相互不信任,因此域拓扑信息不会发往其他域。即每个域只有自己的拓扑信息,没有其他域的拓扑信息。此时,为了计算跨域的路径,需要各个域的 PCE 利用通讯协议,联合起来才能计算出端到端的跨域路径。

同时,由于域之间互不信任,各个域需要向其他域隐藏其域内路径信息。即,在 PCE 联合计算跨域路径的情况下,某个域的 PCE 计算出一条域内路径,不能将详细的路径信息发送给其他域的 PCE,需要通过某种保密机制来隐藏其路径信息。

以下各章内容以 IETF 相关标准为基础,给出 PCE 在跨域接口 ENNI 中的应用。在国内 PCE 相关标准的标准化之后,应遵循国内 PCE 标准。

B.2 PCE 多域应用场景

图 B.1 给出了一种 PCE 多域应用场景。每个域都有相应的 PCE 负责计算域内路径。PCE 功能可以部署在一个独立的服务器,也可以部署在某个计算能力较强的节点上。上述两种情况,PCE 都需要收集相应域的拓扑信息。这可以通过路由协议(如 OSPF-TE)实现,也可以通过网管注入网络拓扑的方式来实现。同时,PCE 采用相应的 CSPF 算法,可以计算域内的最优约束路径。多个 PCE 可以联合起来,利用 PCE 通讯协议,协同计算跨域最优路径。PCE 的多域应用还可以采用其他模式。

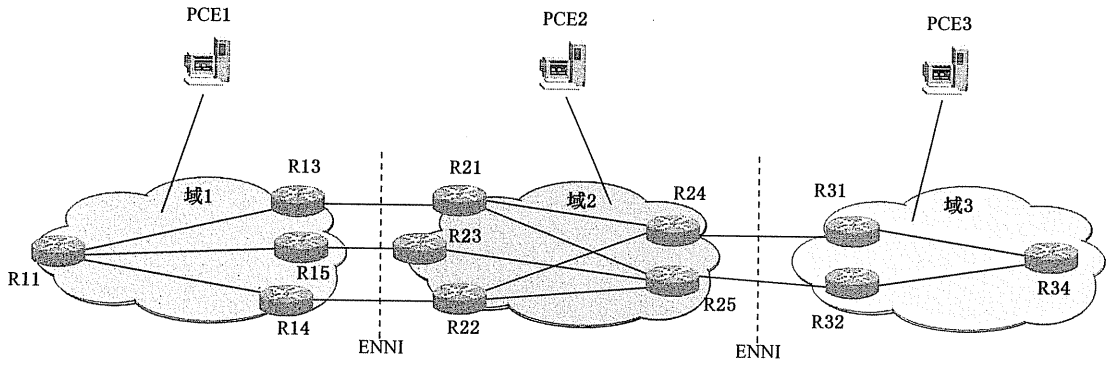


图 B.1 PCE 多域应用场景

B.3 PCE 联合计算跨域最优路径

可以采用后向递归路径计算(Backward-Recursive PCE-Based Computation BRPC)方法,由多 PCE 的交互计算得到跨域的最优路径。

在利用 BRPC 计算跨域最优路径之前,需要确定域序列(即源节点所属域和目的节点所属域之间经过的域)。域序列的确定,可以人工指定或通过其他方式发现。

确定域序列之后,PCC 在请求计算跨域路径时,将路径计算请求通过 PCReq 消息发送给本域的 PCE。本域的 PCE 再将路径计算请求转发给下一个域的 PCE。依次转发,直到目的地所属域的 PCE 接收到路径计算请求。目的地所属域的 PCE 计算潜在路径(与上一个域相邻的边界节点到目的节点的路径),生成虚拟最短路径树 VSPT(Virtual Shortest Path Tree),并利用 PCRep 消息发送给上一个域的 PCE。上一个域的 PCE 根据域拓扑信息,以及 VSPT,计算潜在的路径(与上一个域相邻的边界节点到目的节点的路径),生成新的 VSPT,并利用 PCRep 消息发送给上一个域的 PCE。其他 PCE 继续上述过程,直到源节点所属域的 PCE 收到 PCRep 消息,根据域拓扑信息,以及 VSPT 信息,计算得到源节点到目的节点的路径。

BRPC 的协议扩展遵循 IETF RFC5441。根据 IETF RFC5441,以图 B.2 为例说明 PCE 计算跨域最优路径的方法。

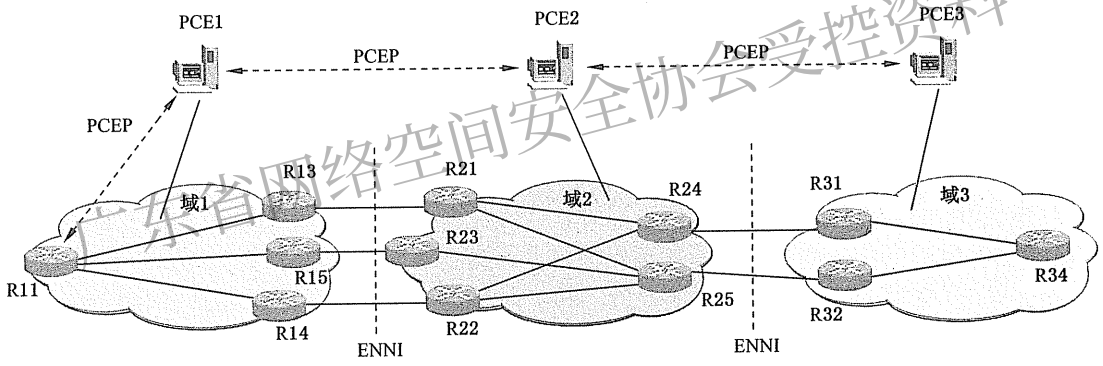


图 B.2 BRPC 路径计算

假设需要计算 R11 到 R34 的路径,过程如下:

a) R11 发送 PCReq 消息给 PCE1,请求计算 R11 到 R34 的路径,并指定域序列为域 1-域 2-域 3;

b) PCE1 发现目的节点不在域 1,转发 PCReq 消息给下一个域的 PCE(PCE2);

c) PCE1 发现目的节点不在域 2,转发 PCReq 消息给下一个域的 PCE(PCE3);

d) PCE3 发现目的节点在域 3,计算与域 2 相邻的边界节点到目的节点的路径,即 R31/R32 到 R34 的路径,分别为 R31-R34/R32-R34;

e) PCE3 将上述两条路径分别放入两个 ERO,并携带 METRIC 对象指示相应路径的代价,通过 PCRep 消息返回给 PCE2;

f) PCE2 根据 PCE3 返回的路径信息,以及本域拓扑信息,计算与域 1 相邻的边界节点到目的节点的路径,假设为 R21-R24-R31-R34/R23-R25-R32-R34/R22-R24-R31-R34;

g) PCE2 将上述三条路径分别放入 3 个 ERO,并携带 METRIC 对象指示相应路径的代价,通过 PCRep 消息返回给 PCE1;

h) PCE1 根据 PCE2 返回的路径信息,以及本域拓扑信息,计算源节点到目的节点的路径,假设为 R11-R13-R21-R24-R31-R34;

i) PCE1 返回上述路径给 R11。

上述过程中，VSPT 由一个或多个 ERO 对象组成，指示一个或多个边界节点到目的节点的路径。

B.4 域间路径保密

在多域路径计算的过程中，如果域之间不是互相信任的，则需要对其他域隐藏自己内部的路径信息，即需要路径保密。

在 PCE 采用上述 BRPC 过程实现跨域路径计算时，路径保密通过 PATH-KEY 的机制来实现。PCE 计算域内的一段路径，在返回路径结果给 PCC 或其他域的 PCE 之前，可以用一个 PATH-KEY 子对象替代详细的域内路径。信令建路过程中，到达入边界节点时，入边界节点再向相应的 PCE 请求获取 PATH-KEY 对应的域内路径。域间路径保密应遵循 IETF RFC5520。

根据 IETF RFC5520，以图 B.2 为例说明路径保密的实现过程。图 B.3 中包含两个互不信任的网络域。假设需要从入口节点建立一条 LSP 到出口节点。入口节点和出口节点在两个不同的 AS，两个 AS 分别由 PCE1 和 PCE2 负责各自 AS 内的路径计算。

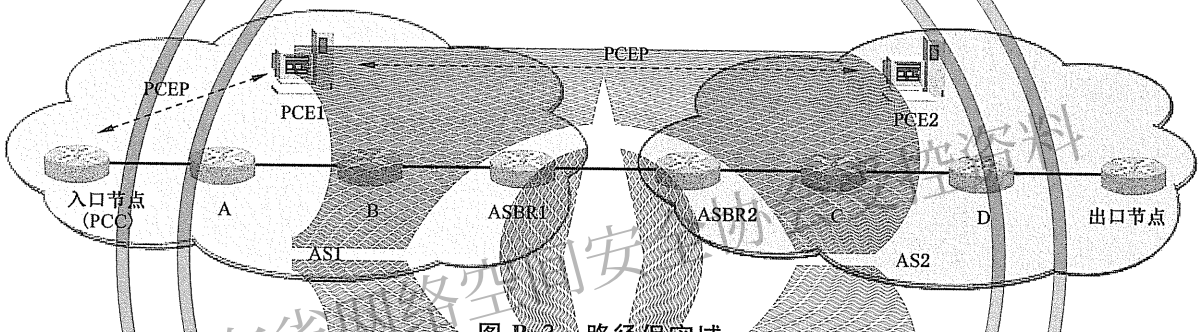


图 B.3 路径保密域

入口节点作为 PCC，向 PCE1 发送路径计算请求消息。PCE1 不能计算端到端的路径，因此请求 PCE2 计算。PCE2 计算 ASBR-2 到出口节点的路径(ASBR-2->C->D->出口)，并生成 Path Key 信息，将路径以(ASBR-2，PKS，出口)的形式返回给 PCE1。其中 PKS 表示 Path Key 子对象。PCE1 再计算端到端路径，得到路径(入口，A，B，ASBR-1，ASBR-2，PKS，出口)，并返回给 PCC。

PCC 得到路径，开始信令建立 LSP 的过程。信令在第一个 AS 的处理过程没什么特别的地方。当 Path 消息到达 ASBR-2 时，ERO 中下一跳是 PKS，因此需要对其进行扩展。ASBR-2 从 PKS 中获取 PCE2 的地址信息，并向其请求 Path Key 对应的路径。PCE-2 返回详细路径(ASBR-2->C->D->出口)给 ASBR-2，ASBR-2 继续正常的信令处理过程。

附 录 C
（资料性附录）
单级路由举例

本附录给出一个单级结构的 E-NNI 路由/信令的示例。本示例中，控制域拓扑发布使用一个 RC 节点表示一个控制域。实际中，一个控制域可以与多个 RC 关联。

本示例包含了以下内容：

a） 控制平面拓扑；

b） 数据平面拓扑；

c） 连接路由计算和 ERO 结构；

d） 域边界的呼叫过程。

C.1 控制域

本部分定义的 TCE 状态机可溯源自 IETF RFC1661“点到点协议”定义的链路控制协议（LCP）状态机。PPP 使用该状态机上来协商配置细节，包括一个终端点的能力，以提供分组层协议在一个点到点连接的运行。在使用多个不同互操作实现的系统中，该状态机被广泛应用了。本附录定义了源自 LCP 状态机的修改。

在如图 C.1 所示的路由结构中，一个 RA 被划分为一些低等级的 RA 和互连的 SNPP 链路。RA 的内部结构对 RA 内部可知，对外不可知。（即在 RA1 中，通过两条 SNPP 链路互连的 3 个子 RA 了解其拓扑，而对外是不透明的）。

图 C.1 ASON 路由结构

在某个路由层次，两个 RCD 通过一条 SNPP 互连（见图 C.2）。

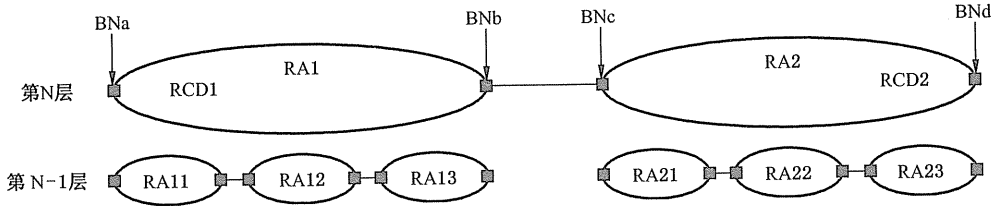


图 C.2 路由控制域

要发布穿越一个 RCD 的代价有多种方法，下面给出了两种方法：

a) 抽象节点：把一个 RCD 描述为一个单节点，不存在内在结构。E-NNI 路由协议中发布的第 N 层拓扑包含两个节点（AN1 和 AN2），以及 1 条域间链路（如图 C.3 所示）。

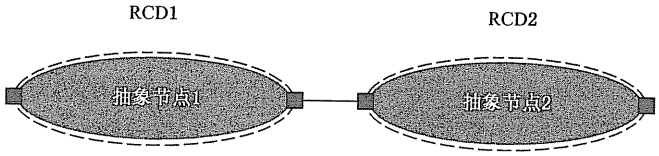


图 C.3 抽象节点表示

b) 抽象链路：把一个 RCD 用其边缘节点和域内部的抽象 SNPP 链路来表示。E-NNI 路由协议发布的第 N 层拓扑包含 4 个节点（BNa、BNb、BNc 和 BNd）和 3 条 SNPP 链路（如图 C.4 所示）。

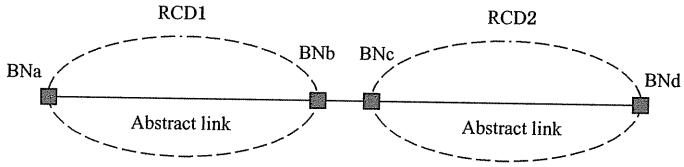


图 C.4 抽象链路表示

在图 C.5 中，在一个路由层次上有 4 个路由控制域，即 CD1、CD2、CD3 和 CD4。
在这个例子中，对于 CD1 和 CD2 采用抽象链路模型，对于 CD3 和 CD4 采用抽象节点模型。

C.2 控制平面

如图 C.5 所示，有 4 个 OSPF 节点 RC1、RC2、RC3 和 RC4，构成控制邻接关系，在图中用红色表示。这 4 个 OSPF 节点分别代表了 4 个控制域。同样，本例中使用一个 RC 代表一个域，也可以使用多个 RC 代表一个域。

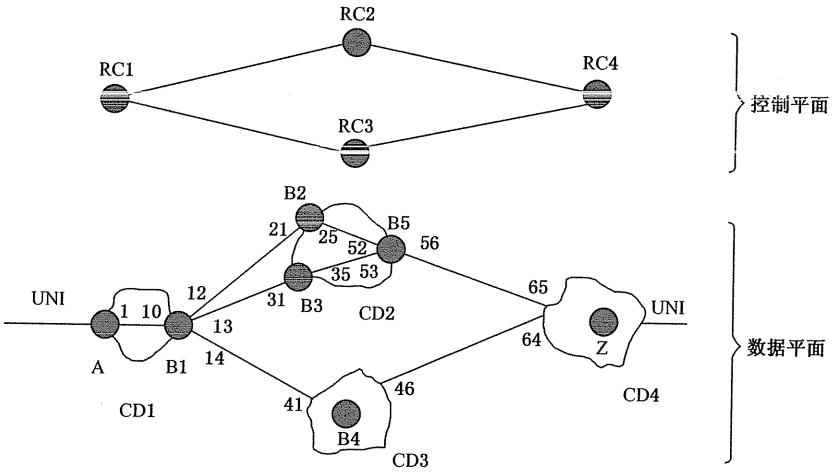


图 C.5 拓扑示例

C.3 数据平面

图 C.5 给出了这些控制域的数据平面和拓扑，拓扑中包含边界节点、域间链路和域内链路。图中标出了边界节点(B1、B2 等)和链路接口 ID(13、31 等)。

C.4 RC1 发布的链路

RC1 发布的链路如下：

a) B1->B2(域间链路)
   1) 发布路由器是 RC1；
   2) 本地节点 ID 子 TLV 包含 B1；
   3) 远程节点 ID 子 TLV 包含 B2；
   4) 本地接口 ID 子 TLV 包含 12；
   5) 远程接口 ID 子 TLV 包含 21；
   6) 链路 ID 子 TLV 包含 RC2。

b) B1->B3（域间链路）
   1) 发布路由器是 RC1；
   2) 本地节点 ID 子 TLV 包含 B1；
   3) 远程节点 ID 子 TLV 包含 B3；
   4) 本地接口 ID 子 TLV 包含 13；
   5) 远程接口 ID 子 TLV 包含 31；
   6) 链路 ID 子 TLV 包含 RC2。

c) B1->B4（域间链路）
   1) 发布路由器是 RC1；
   2) 本地节点 ID 子 TLV 包含 B1；
   3) 远程节点 ID 子 TLV 包含 B4；
   4) 本地接口 ID 子 TLV 包含 14；
   5) 远程接口 ID 子 TLV 包含 11；
   6) 链路 ID 子 TLV 包含 RC3。

d) A->B1（域内链路）
   1) 发布路由器是 RC1；
   2) 本地节点 ID 子 TLV 包含 A；
   3) 远程节点 ID 子 TLV 包含 B1；
   4) 本地接口 ID 子 TLV 包含 1；
   5) 远程接口 ID 子 TLV 包含 10；
   6) 链路 ID 子 TLV 被设置为 0.0.0.0。

e) B1->A（域内链路）
   1) 发布路由器是 RC1；
   2) 本地节点 ID 子 TLV 包含 B1；
   3) 远程节点 ID 子 TLV 包含 A；
   4) 本地接口 ID 子 TLV 包含 10；
   5) 远程接口 ID 子 TLV 包含 1；

6） 链路 ID 子 TLV 被设置为 0.0.0.0。

C.5 RC2 发布的链路

RC2 发布的链路如下：

a） B2->B1（域间链路）

1） 发布路由器是 RC2；
2） 本地节点 ID 子 TLV 包含 B2；
3） 远程节点 ID 子 TLV 包含 B1；
4） 本地接口 ID 子 TLV 包含 21；
5） 远程接口 ID 子 TLV 包含 12；
6） 链路 ID 子 TLV 包含 RC1。

b） B3->B1（域间链路）

1） 发布路由器是 RC2；
2） 本地节点 ID 子 TLV 包含 B3；
3） 远程节点 ID 子 TLV 包含 B1；
4） 本地接口 ID 子 TLV 包含 31；
5） 远程接口 ID 子 TLV 包含 13；
6） 链路 ID 子 TLV 包含 RC1。

c） B5->Z（域间链路）

1） 发布路由器是 RC2；
2） 本地节点 ID 子 TLV 包含 B5；
3） 远程节点 ID 子 TLV 包含 Z；
4） 本地接口 ID 子 TLV 包含 56；
5） 远程接口 ID 子 TLV 包含 65；
6） 链路 ID 子 TLV 包含 RC4。

d） B2->B5（域内链路）

1） 发布路由器是 RC2；
2） 本地节点 ID 子 TLV 包含 B2；
3） 远程节点 ID 子 TLV 包含 B5；
4） 本地接口 ID 子 TLV 包含 25；
5） 远程接口 ID 子 TLV 包含 52；
6） 链路 ID 子 TLV 被设置为 0.0.0.0。

e） B5->B2（域内链路）

1） 发布路由器是 RC2；
2） 本地节点 ID 子 TLV 包含 B5；
3） 远程节点 ID 子 TLV 包含 B2；
4） 本地接口 ID 子 TLV 包含 52；
5） 远程接口 ID 子 TLV 包含 25；
6） 链路 ID 子 TLV 被设置为 0.0.0.0。

f） B3->B5（域内链路）

1） 发布路由器是 RC2；
2） 本地节点 ID 子 TLV 包含 B3；

3） 远程节点 ID 子 TLV 包含 B5；
4） 本地接口 ID 子 TLV 包含 35；
5） 远程接口 ID 子 TLV 包含 53；
6） 链路 ID 子 TLV 被设置为 0.0.0.0。

g） B5->B3（域内链路）
1） 发布路由器是 RC2；
2） 本地节点 ID 子 TLV 包含 B5；
3） 远程节点 ID 子 TLV 包含 B3；
4） 本地接口 ID 子 TLV 包含 53；
5） 远程接口 ID 子 TLV 包含 35；
6） 链路 ID 子 TLV 被设置为 0.0.0.0。

C.6 RC3 发布的链路

RC3 发布的链路如下：

a） B4->B1（域间链路）
1） 发布路由器是 RC3；
2） 本地节点 ID 子 TLV 包含 B4；
3） 远程节点 ID 子 TLV 包含 B1；
4） 本地接口 ID 子 TLV 包含 41；
5） 远程接口 ID 子 TLV 包含 14；
6） 链路 ID 子 TLV 包含 RC1。

b） B4->Z（域间链路）
1） 发布路由器是 RC3；
2） 本地节点 ID 子 TLV 包含 B4；
3） 远程节点 ID 子 TLV 包含 Z；
4） 本地接口 ID 子 TLV 包含 46；
5） 远程接口 ID 子 TLV 包含 64；
6） 链路 ID 子 TLV 包含 RC4。

C.7 RC4 发布的链路

RC4 发布的链路如下：

a） Z->B5（域间链路）
1） 发布路由器是 RC4；
2） 本地节点 ID 子 TLV 包含 Z；
3） 远程节点 ID 子 TLV 包含 B5；
4） 本地接口 ID 子 TLV 包含 65；
5） 远程接口 ID 子 TLV 包含 56；
6） 链路 ID 子 TLV 包含 RC2。

b） Z->B4（域间链路）
1） 发布路由器是 RC4；
2） 本地节点 ID 子 TLV 包含 Z；

3） 远程节点 ID 子 TLV 包含 B4；
4） 本地接口 ID 子 TLV 包含 64；
5） 远程接口 ID 子 TLV 包含 46；
6） 链路 ID 子 TLV 包含 RC3。

C.8 UNI-N 的通道计算和 ERO

假设需要建立从 A 到 Z 的连接，源节点 A 知道 3 条可能路径：

a） A->B1->B2->B5->Z；
b） A->B1->B3->B5->Z；
c） A->B1->B4->Z。

假设选择路径 1，则 A 节点发出的 ERO 为：A:1->B1:12->B2:25->B5:56->Z。

假设选择路径 3，则 A 节点发出的 ERO 为：A:1->B1:14->B4:46->Z。

C.9 通道扩展

如果 CD 内部拓扑不对外广播，需要对接收到的 ERO 进行映射或者扩展，来适应 CD 内部的真实拓扑。例如如果 CD3 包含多个节点，ERO 列项{B4:46}在其内部被扩展来适应真实的入口边缘节点和到达目的 TNA 的内部路径。

附 录 D
（资料性附录）
OIF E-NNI 的兼容性

D.1 OIF E-NNI2.0 与 UNI 的兼容

OIF E-NNI2.0 信令应与 OIF UNI2.0 兼容，以支持呼叫和连接建立。

ASON 架构同时支持单层和多层场景。支持多层涉及到一个 CI 到另外一个 CI 的适配功能。OIF E-NNI2.0 仅限于在单层网络中的操作。

OIF E-NNI2.0 可以在网络服务层连接上建立以太网 UNI 业务，但不支持以太网承载接口。在 OIF E-NNI2.0 协议中，以太网信令只在客户层提供消息和对象的转发功能以支持 OIF UNI 2.0。

D.2 OIF E-NNI2.0 与 OIF E-NNI1.0 的兼容

OIF E-NNI1.0 的实现仅要求支持 SDH 业务，OIF E-NNI2.0 还支持以太网业务、OTN 业务。OIF E-NNI2.0 支持 OIF E-NNI1.0 规定的全部 RSVP 消息和信令过程。OIF E-NNI2.0 还支持以下 OIF UNI2.0 中规范的特征：

——呼叫控制；
——子 STS-1 速率连接；
——以太网业务传输；
——OTN 接口传输；
——扩展的安全性；
——呼叫修改。

OIF E-NNI 2.0 还修改了 OIF E-NNI1.0 中的部分信令过程，包括：

a） 网络发起的正常删除；
b） Node ID 和 SC PC ID 的地址分离；
c） 握手过程声明；
d） RSVP-HOP 的注释。

本附录包含了这些特征的后向兼容要求。OIF E-NNI 实现可以对邻居节点的版本进行人工配置。

a） 呼叫控制

呼叫控制使用 RSVP 的 CALL_ID 对象，该对象包含在 Path、PathErr、PathTear 和 Resv 消息中。CALL_ID 对象格式参见 IETF RFC3474，在 OIF UNI2.0 中包含了对该对象的特别声明。

CALL_ID 对象在 OIF E-NNI2.0 中为强制要求。OIF E-NNI1.0 规定要求 CALL_ID 对象为可选，因此发送给 OIF E-NNI1.0 接口的消息中如包含 CALL_ID 对象，也不应引起任何问题。此外，CALL_ID 的类型号为 11bbbbbb，根据 IETF RFC2205，任何信令实例收到这种类型的未知对象，应予以忽略并原样转发。

如果 OIF E-NNI2.0 信令实例收到了不带 CALL-ID 对象的 Path 消息，应在该消息中插入 CALL-ID 对象，并在所有与该呼叫相关联的消息中使用该对象。

b） 子 STS-1 速率连接

这个功能在 OIF E-NNI2.0 是可选的。子 STS-1 速率连接要求使用 IETF RFC4606 规定的 SONET/SDH SENDER_TSPEC 格式，这与 OIF E-NNI1.0 一致。为支持此功能不需要新的对象。但是要求路径上的所有 E-NNI 接口都应支持 OIF E-NNI2.0，且节点应支持子速率交换，才能提供这种业务。

c) 以太网业务传输

OIF E-NNI2.0 不支持经过以太网承载的业务接口，但支持以太网业务映射到 TDM 服务层传送，并由 OIF E-NNI2.0 信令控制。以太网业务需要一个新的 SENDER_TSPEC/FLOWSPEC 类型，以及两种新的标签格式来支持 OIF UNI2.0 规定的以太网专线和以太网虚拟专线业务。为了支持以太网业务请求，所有 E-NNI 节点都应支持 OIF E-NNI2.0。

d) OTN 接口传输

为支持 OTN 接口，需要支持 IETF REC4328 规定的 SENDER_TSPEC/FLOWSPEC 和新的标签格式。为支持 OTN 业务请求，所有 E-NNI 接口都应支持 OIF E-NNI2.0 和 OTN (G.709)接口 ODUk 层的交换。

e) 扩展的安全性

扩展的安全性对本部分没有直接的影响，对于信令实现并不引入任何后向兼容性。

f) 呼叫修改

为支持呼叫修改，所有 E-NNI 接口应支持 OIF E-NNI2.0 和可选的呼叫修改扩展机制。

呼叫修改的一种机制是增加或删除已建呼叫的连接。这种机制仅要求支持 CALL_ID 对象。另一种机制是通过修改已建连接的带宽，这种机制依赖于 make-before-break 的信令机制，需要使用 SESSION_ATTRIBUTE 对象中的 Shared-Explicit (SE)预留方式，如果 Resv 消息选择了该方式则可以使用 make-before-break 步骤。这种新的预留方式和对象的使用，参见 GB/T 21645.5—2012。

与 GB/T 21645.5—2012 中的 Resv 消息支持一个或两个 FILTER_SPEC 对象相反，OIF E-NNI2.0的呼叫修改机制建议使用两个 Resv 消息，每个携带单独的 FILTER_SPEC 对象。用于 Path 状态的 Resv 消息只包含与 Path 消息相关的 FILTER_SPEC 和 label 对象，并建议只在收到 PathErr 中 Path_State_Removed 标记设置后停止刷新与 Path 状态相关的 Resv 消息。

g) 网络发起的正常删除

OIF UNI2.0 规定使用 Notify 消息而不是 Path/Resv 消息中 ADMIN_STATUS 对象的 A+D 比特发起网络侧正常删除。网络侧删除包含从源 UNI-N 或宿 UNI-N 发起的删除。

源或宿 UNI-C 发起的 SC 业务正常删除以及源或宿 UNI-N 发起的 SPC 业务的正常删除与 OIF E-NNI1.0 保持一致。正常删除与 OIF E-NNI1.0 信令规定的兼容性见 8.4.3。

h) Node ID 和 SC PC ID 的地址分离

OIF E-NNI2.0 引入了 Node ID，SC PC ID 以及 SC PC SCN 地址标识符。在 OIF E-NNI1.0 中，所有这些标识符一起构成 Node ID。为后向兼容性，OIF E-NNI2.0 应该使用与之相同的 Node ID，SC PC ID，和 SC PC SCN 地址。标识符分离在 OIFE-NNI-Sig-01.0 的 6.1 中描述。

i) 握手过程声明

OIF E-NNI2.0 规定，应在接收新的 Path 消息前，交换 Hello 消息。为了与 OIF UNI1.0 实现进行互连，需满足以下要求：

——OIF E-NNI1.0 实现支持在接收新的 Path 消息前，交换 Hello 消息，或者；

——OIF E-NNI2.0 实现允许配置 Hello 消息行为来支持 OIF E-NNI1.0 实现。

OIF E-NNI2.0 要求 Hello 消息的 RESTART_CAP 对象中的 RESTART_TIME 的值为

0xFFFFFFFF,RECOVERY_TIME为非零值。当OIF E-NNI2.0与OIF E-NNI1.0互连时，如果RESTART_TIME值非0xFFFFFFFF,OIF E-NNI2.0实现可以忽略RESTART_TIME。如果恢复Hello邻接后，OIF E-NNI1.0发送的RECOVERY_TIME为0，OIF E-NNI2.0可以删除到OIF E-NNI1.0的连接。RFC3209相一致,OIF E-NNI2.0不允许在Hello消息中使用MESSAGE_ID_ACK或者MESSAGE_ID_NACK对象。收到包含MESSAGE_ID_ACK对象的Hello消息后,信令实例应该接收MESSAGE_ID_ACK对象而忽略MESSAGE_ID_NACK对象。

j) RSVP_HOP的注释

为支持OIF E-NNI1.0的后向兼容性,应支持类型为4和5的RSVP_HOP,但不应主动产生这种RSVP_HOP。

中华人民共和国
国家标准
自动交换光网络(ASON)技术要求
第9部分:外部网络-网络接口(E-NNI)
GB/T 21645.9—2012
*
中国标准出版社出版发行
北京市朝阳区和平里西街甲2号(100013)
北京市西城区三里河北街16号(100045)
网址 www.spc.net.cn
总编室:(010)64275323 发行中心:(010)51780235
读者服务部:(010)68523946
中国标准出版社秦皇岛印刷厂印刷
各地新华书店经销
*
开本 880×1230 1/16 印张 6.5 字数 185千字
2013年5月第一版 2013年5月第一次印刷
*
书号:155066·1-46522 定价 84.00 元

GB/T 21645.9-2012