



中华人民共和国通信行业标准

YD/T 2415-2012

基于 shim6 的 IPv6 站点多归属技术要求

Technical requirements of multihoming shim protocol for IPv6

2012-12-28 发布

2013-03-01 实施

中华人民共和国工业和信息化部 发布

目 次

前 言.....IV

1 范围.....1

2 规范性引用文件.....1

3 术语、定义和缩略语.....1

 3.1 术语和定义.....1

 3.2 缩略语.....5

4 shim6协议综述.....5

 4.1 概述.....5

 4.2 shim子层的位置.....5

 4.3 总体操作.....6

 4.4 上下文标签.....8

 4.5 上下文分叉.....8

 4.6 API扩展.....8

 4.7 shim6安全.....8

 4.8 shim控制报文的概述.....9

 4.9 扩展头顺序.....9

5 报文格式.....10

 5.1 概述.....10

 5.2 共同的shim6报文格式.....10

 5.3 shim6净荷扩展头格式.....10

 5.4 共同的shim6控制头.....11

 5.5 I1报文格式.....12

 5.6 R1报文格式.....13

 5.7 I2报文格式.....14

 5.8 R2报文格式.....15

 5.9 R1bis报文格式.....17

 5.10 I2bis报文格式.....18

 5.11 更新请求报文格式.....19

 5.12 更新确认报文格式.....21

 5.13 保活（Keepalive）报文格式.....22

 5.14 探测（Probe）报文格式.....22

 5.15 出错报文格式.....26

 5.16 选项格式.....27

6 一台主机的概念模型.....34

6.1 概述.....34

6.2 概念数据结构.....34

6.3 上下文的状态机状态.....35

7 建立ULID对的上下文.....36

7.1 概述.....36

7.2 上下文标签的惟一性.....36

7.3 定位符验证.....36

7.4 普通上下文的建立.....37

7.5 同时的上下文建立.....37

7.6 上下文恢复.....38

7.7 上下文的混淆.....39

7.8 发送I1消息.....40

7.9 重传I1消息.....40

7.10 接收I1消息.....40

7.11 发送R1消息.....41

7.12 接收R1消息和发送I2消息.....41

7.13 重传I2消息.....42

7.14 接收I2消息.....42

7.15 发送R2消息.....43

7.16 对上下文混淆的匹配.....43

7.17 接收R2消息.....43

7.18 发送R1bis消息.....44

7.19 接收R1bis消息和发送I2bis消息.....45

7.20 重传I2bis消息.....45

7.21 接收I2bis消息和发送R2消息.....45

8 处理ICMP出错消息.....46

9 ULID对上下文的拆除.....48

10 更新对端.....48

10.1 概述.....48

10.2 发送更新请求消息.....48

10.3 重传更新请求消息.....48

10.4 当重传时出现更加新的信息.....49

10.5 接收更新请求消息.....49

10.6 接收更新请求确认消息.....50

11 发送ULP净荷.....50

11.1 概述.....50

11.2 在一次交换后发送ULP净荷.....51

12 接收分组.....51

12.1 概述.....51

12.2 接收没有扩展头的净荷.....51

12.3 接收shim6净荷扩展头.....51

12.4 接收shim控制消息.....52

12.5 上下文查找.....52

13 初始联系.....53

14 失败检测以及定位符探测.....53

14.1 概述.....53

14.2 失败检测以及定位符探测协议概览.....54

14.3 协议行为.....56

15 协议常量.....59

16 其他地方的影响.....60

16.1 拥塞控制考虑.....60

16.2 中间设备(Middle-Boxes)考虑.....60

16.3 操作和管理考虑.....61

16.4 其他方面的考虑.....61

16.5 失败检测以及定位符探测的操作考虑.....62

17 安全考虑.....63

17.1 概述.....63

17.2 与IPSec交互.....63

17.3 其他的威胁.....64

17.4 失败检测以及定位符探测安全考虑.....65

18 IANA考虑.....65

参考文献.....68

前 言

本标准按照按照GB/T 1.1—2009给出的规则起草。

本标准技术内容与IETF RFC5533 (shim6: Level 3 Multihoming Shim Protocol for IPv6) 和IETF RFC5534 (Failure Detection and Locator Pair Exploration Protocol for IPv6 Multihoming) 保持一致, 对应关系如下:

- 第3章术语分别来自IETF RFC5533 2.1和IETF RFC5534 3;
- 第4章中, 4.2对应于IETF RFC5533 1.6, 4.3对应于IETF RFC5533 4, 4.4~4.9分别对应于IETF RFC5533 4.1~4.6;
- 第5章对应于IETF RFC5533 5, 其中5.13和5.14分别增加了IETF RFC5534 5.1和5.2的内容, 5.16增加IETF RFC5534 5.3的内容作为5.16.9;
- 第6章~第13章对应于IETF RFC5533 6~13;
- 第14章中, 14.1对应于IETF RFC5534 1, 14.2对应于IETF RFC5534 4, 14.3对应于IETF RFC5534 6;
- 第15章对应于IETF RFC5533 14, 增加了IETF RFC5534 7的内容;
- 第16章对应于IETF RFC5533 15, 增加了IETF RFC5534 9作为16.5;
- 第17章对应于IETF RFC5533 16, 增加了IETF RFC5534 8作为17.4;
- 第18章对应于IETF RFC5533 17。

本标准由中国通信标准化协会提出并归口。

本标准起草单位: 华为技术有限公司、工业和信息化部电信研究院、清华大学。

本标准起草人: 崔 勇、董 江、郭大勇。

基于 shim6 的 IPv6 站点多归属技术要求

1 范围

本标准规定了基于shim6的IPv6站点多归属的技术要求,规定了shim6的特征,设计目标和操作原理,包括消息格式,协议处理规则,以及如何在两个通信节点间检测到失败,同时还规定了一个探测协议。如果一个失败发生了且能找到另一对可操作的接口和(或)地址,使用该探测协议可用于选择相同的节点之间的另一对可操作的接口和(或)地址。

本标准适用于支持IPv6站点多归属协议的主机和网络设备。

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,仅所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。

- IETF RFC3972 加密生成地址 (Cryptographically Generated Addresses)
- IETF RFC3484 IPv6地址缺省选择方法 (Default Address Selection for Internet Protocol version 6)
- IETF RFC4086 用于安全的随机性需求 (Randomness Requirements for Security)
- IETF RFC4218 IPv6多归属方案面对的威胁 (Threats Relating to IPv6 Multihoming Solutions)
- IETF RFC4861 IPv6邻居协议 (Neighbor Discovery for IP version 6)
- IETF RFC4862 IPv5无状态地址自动配置 (IPv6 Stateless Address Autoconfiguration)
- IETF RFC5535 哈希基地址 (Hash-Based Addresses)

3 术语、定义和缩略语

下列术语、定义和缩略语适用于本文件。

3.1 术语和定义

3.1.1

上层协议 Upper-Layer Protocol

在IP层之上的协议。传输层协议,例如TCP和UDP;控制协议,例如ICMP;路由协议,例如OSPF;还有通过IP进行“隧道”(即封装)的网络层或低层协议,比如网络分组交换(IPX), AppleTalk, 或IP它本身。

3.1.2

接口 Interface

一个节点对一个链接的连接处。

3.1.3

地址 Address

一个IP层的名字,既包含了拓扑的含义,又表示为对一个接口的惟一标识符。长度为128 bits。本标准仅在既不表示定位符又不表示标识符的时候使用“地址”这一术语。

3.1.4

定位符 Locator

一个IP层的一个接口或一组接口的拓扑名字。长为128 bits。数据包穿越网络时定位符装载于IP的地址字段。

3.1.5

标识符 Identifier

IP层端点提供的一个IP层的名字。传输端点名字是传输协议的功能，并且其应包括IP标识符加上一个端口号。

注：本标准并没有定义一个新的格式的IP层标识符，而是仍把IP地址的标识和定位两重语义进行分离。

3.1.6

上层标识符 Upper-Layer Identifier

被上层协议所使用，用作与对端进行通信，实际上是一个IP地址。长为128 bits。ULID是用于伪头部校验和的计算以及ULP中连接的标识。为了确保负载扩展，对同一主机的不同通信（例如，不同的连接）或许要使用不同的ULID。

由于ULID只是节点的IP定位符/地址中的一个，故不必一个独立的命名空间和分配机制。

3.1.7

地址域 Address Field

在IPv6头部中的源和目的地址域。如目前IPv6中所定义的，这些域中装载着“地址”。如果标识符和定位符分离开了，对于在网络中的分组这些域将包含定位符。

3.1.8

域名全称 Fully Qualified Domain Name

主机的域名全称。

3.1.9

ULID对上下文 ULID-Pair Context

由多归属shim在一对上层标识符之间维护的状态。该上下文由以下几部分定义：一个用于通信双方的上下文标签(context tag)，一个ULID对，以及一个分叉实例标识符(forked instance identifier)，具体含义在下面有描述。

3.1.10

上下文标签 Context Tag

上下文的每个端点都会为该上下文分配一个上下文标签。这是用于惟一关联收到的控制分组以及属于该上下文的shim6净荷扩展头。

3.1.11

当前定位符对 Current Locator Pair

上下文的每个端点都有一个当前定位符对，用于向对端发送分组。但两个端点可能使用不同的当前定位符对。

3.1.12

默认上下文 Default Context

在发送端, shim使用ULID对(由ULP传下来的)来查找该对的上下文。因此一般情况下, 一个主机的一个ULID对最多能有一个上下文, 称之为“默认上下文”。

3.1.13

上下文分叉 Context Forking

一种机制, 使得那些知道有多个定位符的ULPs来对同一ULID对使用不同的上下文, 这样就使得相同的ULID可以对不同的通信使用不同的定位符对。上下文分叉使得同一ULID对产生的不仅是默认上下文。

3.1.14

分叉实例标识符 Forked Instance Identifier

为了能够处理上下文分叉的情况, 一个上下文由一个ULID对和一个分叉的上下文标识符来定义。默认的上下文FII值为0。

3.1.15

初始联系 Initial Contact

当ULP决定通过发送和接收ULP分组来与对端发起通信时, 用来描述shim之前的通信。通常, 这不会引起shim中的操作, 因为shim能够延迟上下文的建立, 直到一些任意的、靠后的机会出现。

3.1.16

基于哈希的地址 Hash-Based Addresses

IETF RFC5535定义的一种IPv6格式的地址, 其接口ID是来自所有分配到主机的前缀的加密哈希。

3.1.17

加密生成地址 Cryptographically Generated Addresses

IETF RFC3972定义的一种IPv6格式的地址, 其接口ID是来自公有键的加密哈希。

3.1.18

CGA参数数据结构 CGA Parameter Data Structure

一种CGA和HBA交互的消息, 为了通知对端接口ID是如何计算的。(参见[2]和[3])。

3.1.19

可用的地址 Available Addresses

如果下面所有条件都满足了则称一个地址为可用的:

- 该地址已经分配给该节点的一个接口;
- 分配给该地址的前缀的有效生存时间还没有过期。

注: 有效生存时间参见IETF RFC4861的第4.6.2小节[27]。

——在 IETF RFC4862 中该地址是不确定的。换句话说, 该地址分配完成后通信能够开始。

注: 按最佳化重复地址侦测(DAD)(参见[26])的观点, 这明确的使得一个地址成为最优, 尽管只要有替代者, 执行可能更喜欢使用其他地址。

——该地址是一个全球单播或惟一的本地地址(参见[13])。即它不是一个 IPv6 站点本地或链路本地地址。

根据链路本地地址, 该节点不能确定所给的地址在哪个链路有用。

——该地址和接口是否允许使用由主管部门的政策决定。

对可用地址的发现和监管的机制已经超出了shim6的范围。shim6的执行一定要能够使用IPv6邻居发现（参见[27]）、地址自动配置（参见[28]）以及DHCP（参见[25]）（当应用了DHCP）所提供的信息。这些信息包括新地址的可达性以及存在的地址的状态改变（例如当一个地址变得无效）。

3.1.20

本地的可使用地址 Locally Operational Addresses

当一个可用地址的用处被认为可能是本地的则被称作本地可用地址。换句话说，当接口启用，适合这个地址的一个默认路由（如果需要）是可达的，且对该地址没有其他的消息点成为无用的。

对本地可用地址的发现和监管的机制超出了shim6协议的范围。shim6的执行一定要能够使用IPv6邻居发现（参见[27]）所提供的信息。一些执行也可能使用附加的、链路层的特定机制。

注1：在确定一个地址是可用的这一问题中，有一部分是确保在经过了链路层连接性改变，我们仍能链接到相同的IP子网。类如[32]的机制可用作确保这一点。

注2：理论上说，如果存在合适的协议，节点也有可能知道对一个特定的源前缀的路由失败。已经制定了在这方面的一些协议（例如[29] 和 [34]），但都还没成为标准。

3.1.21

可用地址对 Operational Address Pairs

一对本地可用地址被称为单向可用地址对，当分组使用第一个地址作为源地址而使用第二个地址作为目的地址来发送到目的端。

shim6的执行必须支持通过使用确切可达性测试和强制双向通信（FBD）来发现可用地址对，这将在本标准的后面描述。shim6的未来的扩展可能定义其他的机制。这种机制的一些观点如下，但没有在本标准中完全描述：

——来自上层协议主动反馈。例如，TCP 能告诉 IP 层它正在取得进展。在某些情况下，这和当上层提供双向连接（参见[27]）的信息时能避免 IPv6 邻居不可达探测是相似的。在单向连接的情况下，该上层协议使用另一个地址对返回回应，但表明该发送的报文使用的是收到的第一个地址对。

——来自上层协议的被动反馈。可以想象上层协议提供了对多归属层问题的一个说明。例如，TCP 可说明在路劲中有阻塞或缺乏连接因为它没有得到 ACK。

——ICMP 出错报文。鉴于欺骗 ICMP 报文的缓解，但应当注意不要盲目相信这些。一种方法是使用 ICMP 出错报文仅作为一个示意，用于示意实行一个确切的可达性测试或把一个地址对移到要被探测的地址对表的较低位置，但不要在有问题指示时使用这些报文打断正在进行的通信。

注：当这一ICMP报文验证在实行时情形可能不一样，如Gont在[33]所解释的。这些验证能确保（几乎）只有路径耦合攻击者可以伪造该报文。

3.1.22

主地址对 Primary Address Pair

主地址对是由上层协议使用shim6层交互时使用的地址组成的。使用主地址对意味着该通信和常规的非shim6的通信是兼容的，且不需要上下文标签在。

3.1.23

当前地址对 Current Address Pair

shim6需要避免在多路径上同时发送属于相同传输连接的分组。这是因为传输协议一般使用的拥塞控制是基于单路径的概念的。尽管路由也可以引进路径的变化以及传输协议有方法来处理这些，但频繁的变化会引起问题。

注1：IETF RFC5533发布时，在多路径上的有效拥塞控制仍是一个研究话题。shim6并没有试着去同时使用多路径。

注2：流控制传输协议(SCTP)以及未来的多路径传输协议有可能需要与shim6交互，最少要确保它们不会出乎意料的使用shim6。

由于这些原因，有必要选择一个特定的地址对作为当前地址对，一直要用到出现问题，至少在同一段上。

理论上为不同的传输段或shim6上下文支持多个当前地址对是可能的。但本标准尚不支持。

注：当前地址对不需要在任何时候都可操作。如果没有流发送，我们可能不知道当前的地址对是否可操作。不过假设先前工作的地址对也能为新通信继续可操作是有意义的。

3.2 缩略语

下列缩略语适用于本文件：

CGA	Cryptographically Generated Addresses	加密生成地址
FQDN	Fully Qualified Domain Name	域名全称
FII	Forked Instance Identifier	分叉实例标识符
HBA	Hash-Based Addresses	基于哈希的地址
PDS	Parameter Data Structure	参数数据结构
ULID	Upper-Layer Identifier	上层标识符
ULP	Upper-Layer Protocol	上层协议

4 shim6 协议综述

4.1 概述

本标准定义了shim6协议，该协议是在第三层插入了一个shim子层，这样IPv6便可使用多归属技术，使之具有故障切换和负载均衡的特性（参见[10]），而不用考虑多归属的站点是否有一个在全球IPv6路由表里申明的IPv6 PI地址前缀。在站点中的主机若拥有多个IPv6 PA地址前缀则可以使用在本标准中定义的shim6协议来与对端主机建立状态，该状态可以在前一定位符对失效的时候来切换到另一对不同的定位符对上从而达到故障切换的目的。

shim6协议是一个站点级的多归属解决方案，即它允许一个有多连接连接到Internet的站点，在其中某些连接中断的情况下，仍能保持已经存在的通信不中断。但是shim6的处理是在主机端来实现的，而不是在整个站点范围内的机制上。

4.2 shim 子层的位置

本标准是在IP层中添加了多归属的shim子层，即在ULPs之下以使ULP具有独立性，如图1所示。多归属的shim子层的行为像是和扩展头相关，这个扩展头可以放置在分组中任意的与路由相关的头部之后（例如逐跳选项）。但是当定位符对就是ULID对的时候，在扩展头中没有需要装载的数据，此时为空。

让分片报头在多归属shim层之上使得重组更为健壮，因为会有打破的多路径路由会导致使用不同的路径，因此有可能不同的源定位符会用于不同的分片。因此多归属的shim子层放置在IP端系统层（负责分片和重组）和IP路由子层（选择下一跳和接口，用于发送数据包）之间。

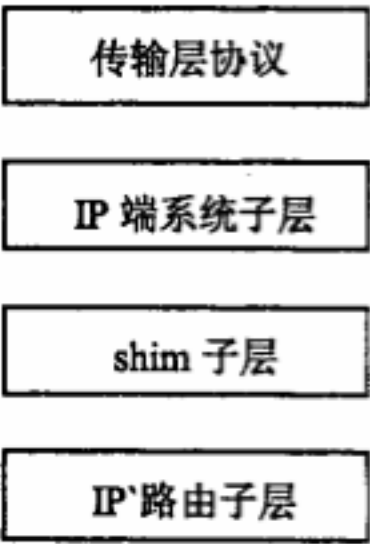


图1 协议栈

应用和上层协议使用由shim6子层提供的由不同定位符映射而来的ULIDs。shim6子层维护着一种状态，称之为ULID对的上下文，每一ULID对（即这种状态应用于所有的ULID对的ULID连接）都要执行这种映射。该映射持续的作用在发送方和接收方，这样ULPs所见的分组好像是通过ULIDs来进行端到端的发送的。即使分组传输会通过IP地址域包含定位符的网络，即使那些定位符或许会被传送shim6子层改动，这一特性也会得到维持。

这一上下文环境状态在每一远端ULID上得以维护，即在每一个对端主机上，而不是在任何更细的粒度上。特别地，该上下文环境状态独立于ULPs以及任何ULP的连接。但这一分叉的能力可使得知道shim6的ULPs对一个ULID对一次使用不止一个定位符对。

如图2所示的持续映射的结果使它对ULPs没有任何影响。特别地，它对伪头部校验和以及连接标识都没有影响。

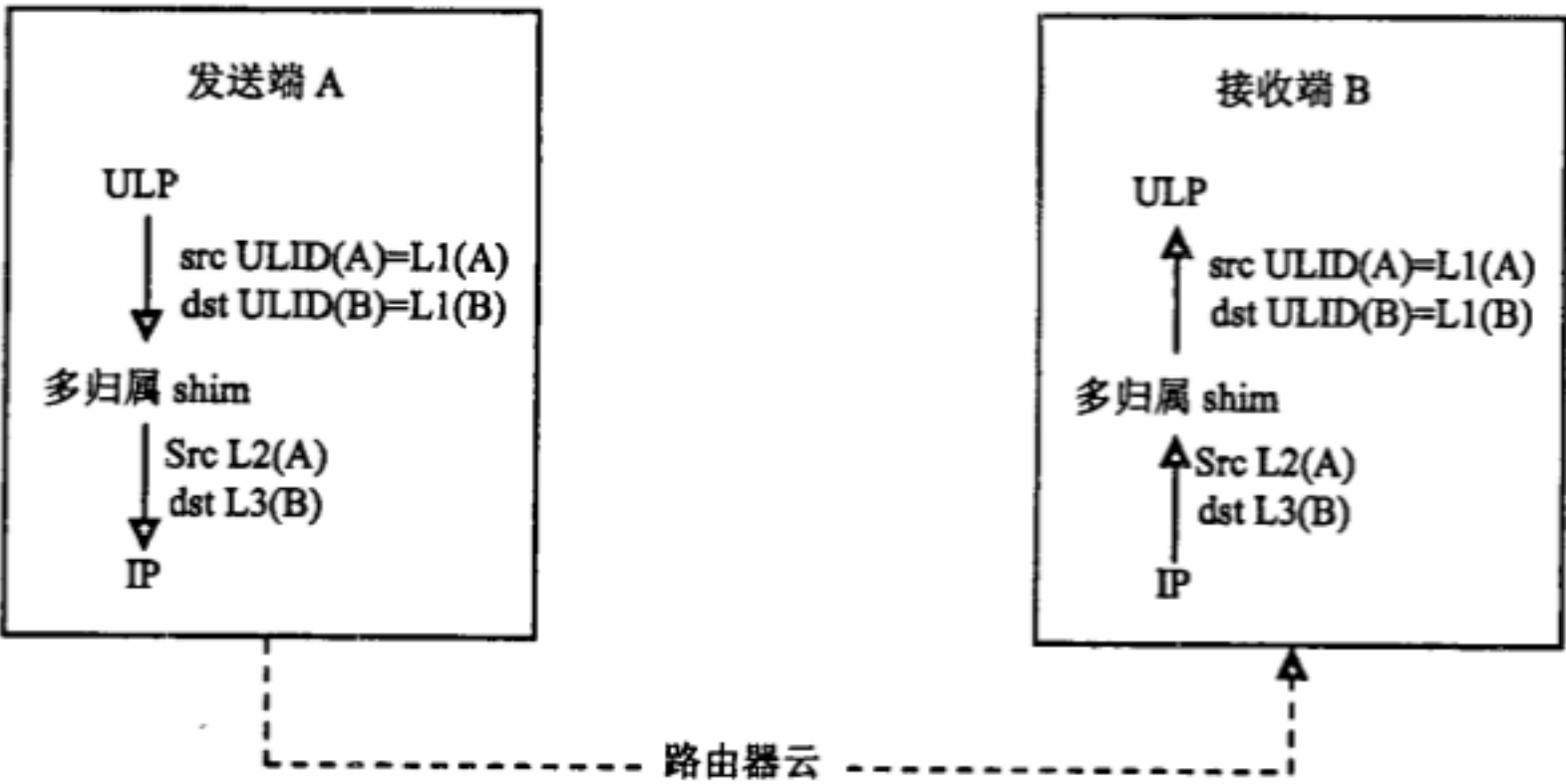


图2 使用改变的定位符来映射

从概念上讲，可以认为这种方法好像使ULIDs和定位符在每一数据包中都存在，就好像一个头部压缩机制的应用，这使得一旦压缩状态建立好了，就不需要在分组中载入ULIDs了。为了使接收者重新创建一个具有正确ULIDs的分组，有必要在数据包中包含一些“压缩标记”。这有助于在分组中的定位符对不足以惟一确定上下文时，指明使用正确的上下文来进行解压缩。

在shim6层和其他的协议之间会有各种不同类型的交互。那些交互会被这些方面影响：在其他的协议中使用这些地址，以及shim6对这些使用的映射的影响。

注：[18]中详细分析了不同协议间的交互，包括流控制传输协议（SCTP），移动IP（MIP），和主机标识协议（HIP）。此外，一些应用程序可能需要与shim6子层更为丰富的互动。为了达到这一要求，已定义了一个API（参见[22]）来允许更多的控制和信息交互，以用于那些需要它的应用。

4.3 总体操作

shim6协议可以在几个情况下运行，下面进行说明：

——主机 A 上的一个应用程序想要通过一些上层协议来与主机 B 上的应用通信。这导致了主机 A 上的 ULP 向主机 B 发送分组。我们称之为初始联系。假设 IP 地址是由默认地址选择（参见[6]）以及它的扩展（参见[8]）来决定的，在这个时候 shim 还不会有任何动作。Shim 的上下文建立能够被延迟到晚些再开始。

——一些在 A 或 B（或两者）上的启发动作会决定开支 shim6 的代价是否值得，使用 shim6 会使得主机到主机的通信健壮而不会出现定位符失败。例如，这种启发可能是超过了 50 个的分组收发，或是当激发分组交换时计时器超时。这就会导致 shim 的初始的 4 次上下文建立交互。该启发是为了避免当只有很少的分组在两台主机间交换时还进行 shim 的上下文建立。作为该交互的结果，A 和 B 双方都会知道一个对端定位符的列表。如果上下文建立交互失败，发起端就会知道对端不支持 shim6，并且会继续使用标准的（非 shim6 的）规范。

——对上层协议的分组来说，通信在没有任何改变中继续。特别地，上层协议的分组中不会加入 shim6 的扩展头，因为上层标识符对和定位符对是相同的。另外，可能会在 shim 子层之间为了（不）可达探测而进行消息的交互。

——在某些时刻，可能会出现一些失败。取决于可达性检测的方法，可能会有一些来自上层协议的建议，或者是 shim 的（不）可达性检测可能会发现存在一些问题。在这个时候，通信的一方或双方需要探测不同的可替代的定位符对，直到找到一对，然后选择使用那个定位符对。

——一旦找到了一对可替代的定位符对，shim 将重写发送的分组，并使用 shim6 净荷扩展头来标记该分组，在其中包含了接收方的上下文标记。接收者将使用该上下文标记来找到上下文状态，它会在分组传送给 ULP 之前指明将哪个地址填入 IPv6 头。结果就是，尽管 IP 的路由体系结构使用的是不同的定位符来发送的，从 ULP 的角度来看，分组端到端传送没有发生修改。

——Shim 的（不）可达性探测将会像监管初始的定位符对一样去监管新的定位符对，这样随后发生的失败将会被检测到。

——除了基于端到端观测的失败检测外，一个端点仍可能知道一个或多个它的定位符无法工作。例如，网络接口可能已经失效（在第二层），或是一个 IPv6 地址或许已经被弃用。在这些情况下，主机能够告知对端尝试这些地址是没有意义的。这引发了一些与错误处理相似的东西，并且必须要找到新的可用的定位符对。该协议也能够表述其他形式的定位符的优先符。在任意优先符中的改变都能够告知对方，这将使对端记住新的优先符。在优先符中的改变或许会随意的使得对端想要使用一个不同的定位符对。在这种情况下，对端在一次失败后遵守相同的定位符选择规程（例如通过验证使用替代的定位符使得它的对端确实还可达）。

——当 shim 认为该上下文状态不再会使用时，它便会对该状态进行垃圾回收；状态回收之前没有必要与对端主机进行协调。当已没有该上下文状态时，会有一个已定义的恢复消息用于标识，这可用于发现和恢复过早的垃圾回收，以及发现和恢复对端的状态丢失（由于碰撞或重启）。

注：shim6 中 ULP 的分组能够完全不作修改的被装载，只要 ULID 对仍是定位符对。当选择了不同的定位符对时，该分组就会通过 shim6 扩展头来“加标签”，这样接收者仍能够决定该上下文属于谁。这是在（扩展）头被端点子层和 ULPs 处理之前，由包含 8byte 的 shim6 扩展头来完成的。如果过了一段时间初始的 ULIDs 又被选为有效的定位符对，通过 shim6 的扩展头来标记分组就没有必要了。

4.4 上下文标签

在两台主机之间的上下文实际上就是在两个ULIDs之间的上下文。上下文是通过一对上下文标签来定义的。每个终端都需要分配一个上下文标签，当上下文建立好后，大多数的控制报文都要包含由报文的接收者分配的上下文标签。因此<对端ULID, 本地ULID, 本地上下文标签> (<peer ULID, local ULID, local Context Tag>) 能最小惟一标识一个上下文。但由于shim6的净荷扩展头没有查看分组中的定位符就解封装了，所以接收者需要分配一个对它的所有上下文而言惟一的上下文标签。上下文标签是一个47bit长的数(在一个8byte的扩展头中能适合的最大长)，当保持1bit来区分shim6信号报文和数据包中的shim6头时，允许两者使用相同的协议号。

用于在对端检测一个上下文状态丢失的机制假定了接收者能够告知分组需要定位符重写，即使它已失去所有状态(例如，由于在重新启动后崩溃)。这是因为经过一个主机转换(rehoming)事件，需要接收端重写的分组装载了shim6的净荷扩展头。

4.5 上下文分叉

已经可以确定对未来的ULPs——特别是未来的传输层协议——而言，对不同的通信能够控制使用不同的定位符对是非常重要的。例如，主机A和主机B可能同时使用VoIP流和ftp流来进行通信，那些通信或许会因使用不同的定位符对而受益。但基本的shim6机制只使用一个当前定位符对来应用于所有上下文；因此，一个单一的上下文不能够满足这点。

正是因为这个原因，shim6协议支持上下文分叉这一概念。这是一种机制，通过它ULP能够指定(使用一些还没定义的API)一个上下文，例如ULID对<A1,B2>，被分叉为两个上下文。在这种情况下，分叉后的上下文的FII值是非零的，而默认上下文的FII是0。

FII是一个32bit的标识符，在本协议中并没有其他的语义，只用于标识上下文的元组的一部分。例如，一台主机会为任给的ULID对分配FII来作为序列号。

在shim6协议处理分叉的上下文中，没有其他的特别需要考虑的地方。

应该注意的是，这种分叉并没有允许A通知B这种流可以反方向分叉。所定义的shim6分叉机制仅应用于发送ULP分组。如果一些ULP想要双向分叉，这取决于ULP对此进行设置，然后指明shim在双方使用分叉的上下文。

4.6 API 扩展

shim6讨论了几种API扩展，但对它们的确切的定义超出了本标准的范围。最简单的一个就是增加一个套接字选项，这样就能使流绕过shim(不用建立任何的状态，也不需其他流创建的状态)。这可以是一个IPV6_DONTSHIM套接字选项。这种选项对协议是有利的，比如DNS，在那里应用有其自己的故障切换机制(对DNS而言的多名字系统记录)以及使用shim会潜在增加额外延迟且毫无益处。

注：确切的API扩展在[22]中定义。

4.7 shim6 安全

该机制的保护是通过使用一些技术的结合：

——HBA 技术(参见[3])用于确认定位符对来阻止攻击者将分组流重定向到其他地方。

——在一个新的定位符用于目的地之前，需要可达性探测+恢复(见第14章)，这样用于阻挡第三方洪泛攻击。

——第一条报文不会在应答方创建任何状态。实质上，在应答方创建状态之前需要进行三次握手。

这意味着一个基于状态的 Dos 攻击（试着去耗竭接收方的所有内存）至少会留下攻击者所使用的 IPv6 地址。

——上下文建立报文使用随机数来阻挡重放攻击(replay attacks)，且阻止路径分离攻击者对建立的干扰。

——每个 shim6 协议的控制报文，自过去的上下文建立以来，装载着分配给特定上下文的上下文标签。这说明攻击者在能够欺骗任意的 shim6 的控制报文之前，需要探测到那个上下文标签。这种探测很可能需要潜在的攻击者一直沿着路径去找到上下文标签的值。结果就是通过这种技术 shim6 协议能够不受路径分离攻击。

4.8 shim 控制报文的概述

shim6上下文的建立是通过使用四种报文（I1、R1、I2和R2）来完成的。一般的，它们的发送就是依次从发起方到应答方按照那种顺序。若两端同时尝试建立上下文状态（对同一ULID对），那么它们的I1报文可能会交叉，这导致了一个R2报文的立即返回（这些报文的名称借用了HIP（参见[19]）。）

还定义了R1bis和I2bis报文；它们用于当一个上下文丢失时的恢复。当一个shim6控制报文或shim6净荷扩展头到达时，在接收端没有匹配的上下文状态就会触发一个R1bis的发送。当接收到这个报文，它会导致使用I2bis和R2报文来重新建立shim6上下文状态。

对端的定位符列表通常作为上下文建立交互的一部分来相互交换。但定位符集可能是动态的。因此有更新请求和更新确认报文以及定位符列表选项。

即使定位符表是固定的，主机仍可能会确定一些参数选择发生了变化。例如，它可能会确定有一个本地的明显的失败，这意味着一些定位符不能够再使用了。这使用了一个在更新请求报文中的定位符的优先符选项。用于（不）可达性检测的机制称为强制的双向通信（FBD）。FBD使用一个Keepalive报文，当一台主机已经收到它对端的分组但它的ULP又还没有向其对端发送任何的分组时，就会发送keepalive报文。

另外，当上下文已建立，随后又出现失败，需要有一个方法去探测定位符对可以高效的找到一对可用的。

以上的Probe和Keepalive报文是假设我们已经建立了ULID对上下文。但是通信可能会在初始联系（即，当应用或传输协议尝试建立一些通信）时失败。这是通过ULP中的一些机制来尝试不同的地址对来解决的（参见[6]和[8]）。

注：在IETF RFC5534的未来版本中以及随着ULP和shim之间接口的更为丰富，shim可能会帮助在初始联系时优化发现可用定位符的方法。可留于进一步研究。

4.9 扩展头顺序

因为shim是放置在IP端系统层和IP路由层之间的，故shim头会在端系统扩展头（分段头，目的选项头，AH，ESP）之前，而又在路由相关的头（逐跳扩展头，路由头，以及一个目的选项头，它先于路由头）之后。当使用了隧道，不管是IP-in-IP隧道还是移动IPv6使用的特殊格式的隧道（与本地地址选项和路由头的类型2），都会有一个选择：shim6是应用在隧道里还是隧道外，这会影响到shim6头的位置。

在多数情况下，IP-in-IP隧道用于路由技术；因此把它应用在定位符上是有意义的，这意味着发送方会在任何IP-in-IP封装后插入shim6头。这在当路由器应用IP-in-IP封装时自然发生的。因此，分组会有：

——外部 IP 头（Outer IP header）；

- 内部 IP 头 (Inner IP header) ;
- shim6 扩展头 (如果需要) ;
- ULP。

但shim也能用于创建“shim的隧道”(“shimmed tunnels”), 其中使用shim的IP-in-IP隧道能够在不同标识符间选择隧道端点地址。这种情况下分组有:

- 外部 IP 头 (Outer IP header) ;
- shim6 扩展头 (如果需要) ;
- 内部 IP 头 (Inner IP header) ;
- ULP。

在任何情况下, 接收端的行为都定义好了; 一个接收者按序处理扩展头。但移动IPv6和shim6之间的精确交互有待进一步研究; 或许使移动IPv6在定位符上操作也是有意义的, 意味着shim能够置于MIPv6机制的顶端。

5 报文格式

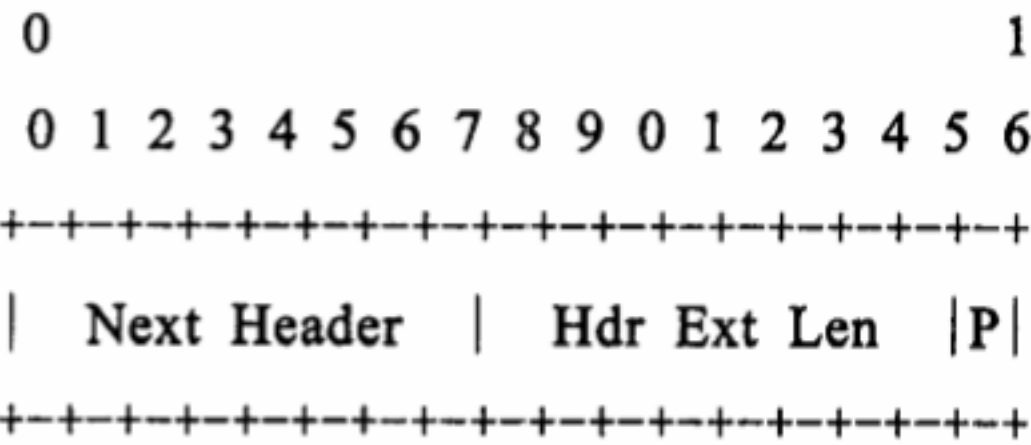
5.1 概述

shim6报文都是使用一个新的IP协议号(140)来装载的。shim6报文有一个共同的头部, 有一些固定的域, 后面跟着特定类型域。

shim6报文是作为IPv6的扩展头结构, 因为shim6净荷扩展头用于经过定位符选择后装载ULP分组。shim6控制报文使用相同的扩展头格式, 使单一的“协议号”需要被允许才能按序通过防火墙。

5.2 共同的 shim6 报文格式

shim6净荷扩展头和控制报文的shim6头部的前17bit是相同的, 其格式如下:



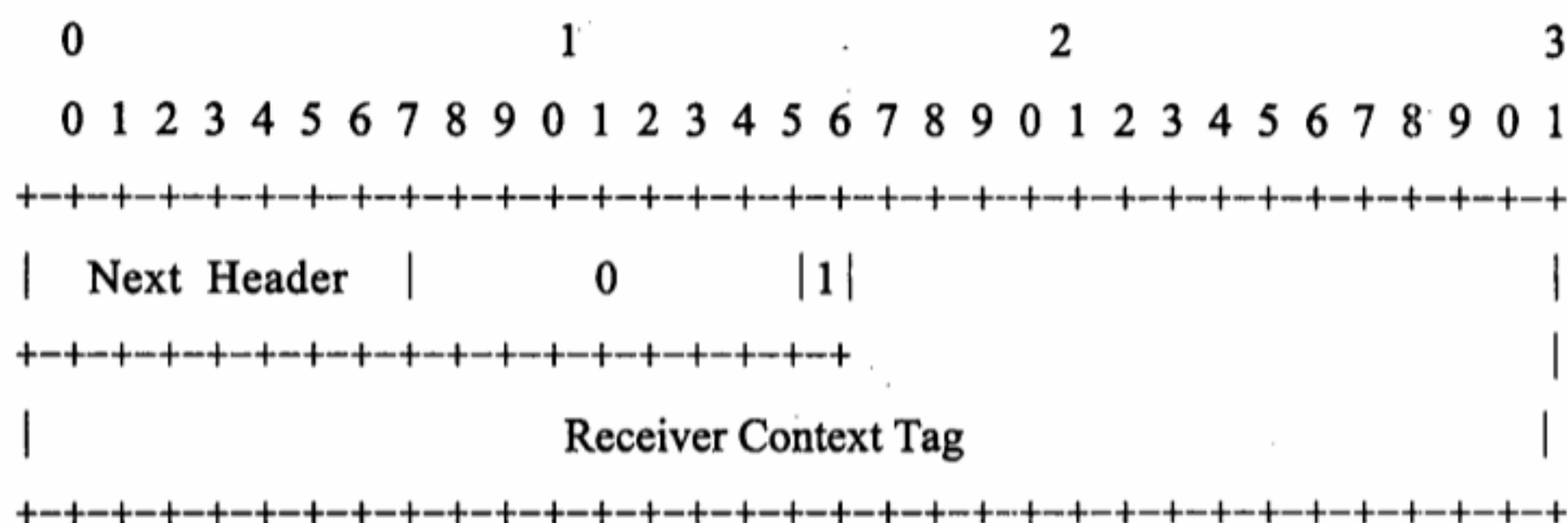
字段:

- Next Header: 这个头后面的净荷。
- Hdr Ext Len: 8位无符号整数。表示shim6头的长度, 以8byte为单位, 不包括头8byte。
- P: 一个单比特位, 用于区分shim6净荷扩展头和控制报文。

shim6的信令分组不会大于1280byte, 这包括IPv6头以及IPv6头和shim6头之间的任何的中间头。满足这一要求的一种方法就是, 如果包括部分定位符地址信息在内的分组大于1280byte, 那么忽略掉这部分。另一选择是进行选项工程, 把要传送的信息划分为不同的shim6报文。一个执行可能加强管理限制, 以避免过大的shim6分组, 例如限制使用的定位符的数量。

5.3 shim6 净荷扩展头格式

shim6净荷扩展头用于装载ULP包, 接收者在向ULP传送前必须要替代在IPv6头中的源和/或目的域的内容。因此当使用的定位符对和ULID对不一样时需要使用扩展头。shim6净荷扩展头的格式如下:



字段:

Next Header: 这个头后面的净荷。

Hdr Ext Len: 0（因为这个头就8个字节）。

P: 设置为1。一个单比特位，用于区分shim6净荷扩展头和控制报文。

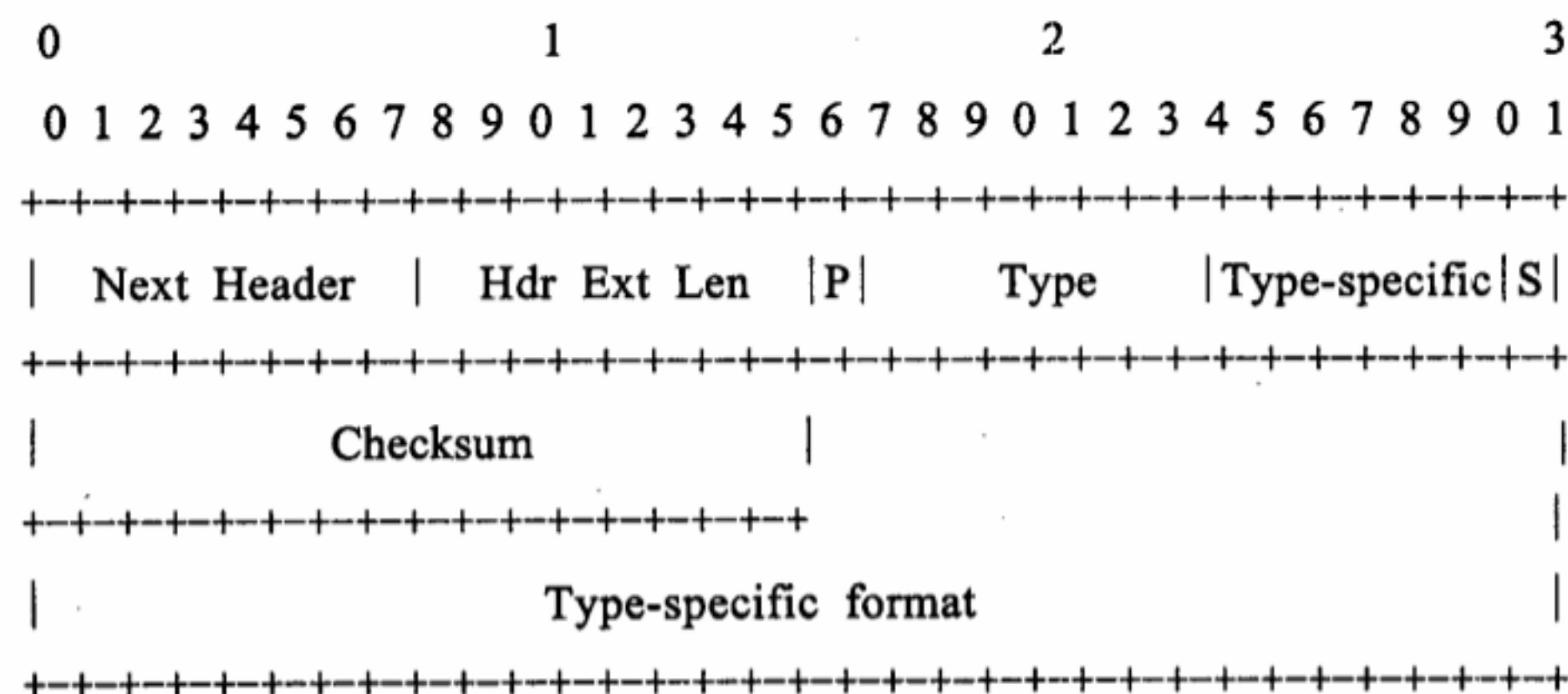
Receiver Context Tag: 47位无符号数。由接收者分配，用于标识该上下文。

5.4 共同的 shim6 控制头

头的共同部分有一个下一头字段和一个头扩展长度字段，是与其他IPv6扩展头一致的，即使对控制报文下一个头值始终是"NO NEXT HEADER"。

shim6头必须是8byte的整数倍；因此，最小大小为8byte。

共同的shim6控制报文头如下:



字段:

Next Header: 8位的选择符。通常设为NO_NXT_HDR (59)。

Hdr Ext Len: 8位无符号整数。以8byte为单位的shim6头长度，不包括头8byte。

P: 设置为0。一个单比特位，用于区分shim6净荷扩展头和控制头。

Type: 7位无符号数。定义了下面表1中所示的精确的报文。在一个丢失的上下文中，类型码0~63不会触发R1bis报文，而64~127会触发R1bis。

S: 单一的一位设成0来允许shim6和HIP有共同的头格式，但仍会区分shim6和HIP的报文。

校验和 (checksum) :

16位无符号数。该校验和是16位1的补，即整个shim6头部报文的1的补，从shim6下一头字段开始，结束由Hdr Ext Len标识。因此有净荷在shim6头的后部，该净荷

没有包括在shim6校验和中。注意，不像类似ICMPv6的协议，该校验和中没有伪头校验和部分；这提供了不更改校验和的定位符灵活性。

特定类型（Type-specific）：
报文的一部分，对不同的报文类型有所不同。

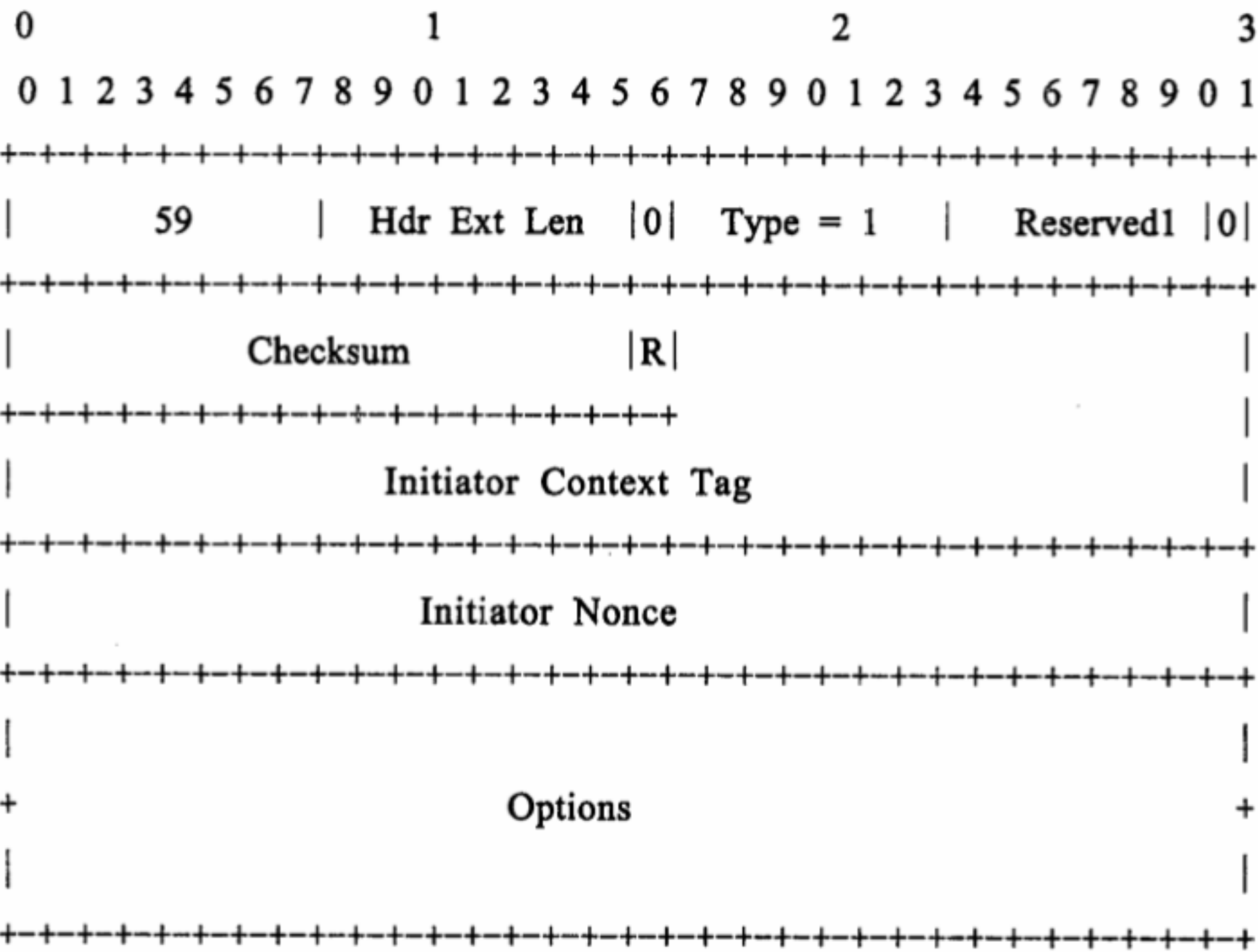
shim6协议的报文类型见表1。

表1 shim6 协议的报文类型

类型值	报文
1	I1 (发起放的第一个建立报文)
2	R1 (应答方的第一个建立报文)
3	I2 (发起方的第二个建立报文)
4	R2 (应答方的第二个建立报文)
5	R1bis (对相关的不存在该上下文的回应)
6	I2bis (对R1bis的回应)
64	更新请求
65	更新确认
66	保活
67	探测报文
68	出错报文

5.5 I1 报文格式

I1报文是上下文建立交互的第一条报文，其格式如下：



字段：
Next Header: NO_NXT_HDR (59)。

Hdr Ext Len: 至少值为1，因为当没有选项时头长度为16byte。

类型: 1。

保留1 (Reserved1): 7位字段。以备后用。传输时值为0.在接收时一定要忽略。

R: 1bit字段。以备后用。传输时值为0.在接收时一定要忽略。

发起者上下文标签 (Initiator Context Tag) :

47位字段。发起者所有的上下文标签，分配给该上下文。

发起者随机数 (Initiator Nonce) :

32位无符号整数。发起者选择的一个随机数，应答方要在R1报文中返回之。

以下的选项是为该报文定义的:

ULID pair: 当在IPv6头中的IPv6源和目的地址不能和ULID对匹配时，则必须要包含此选项。一个例子就是从一个丢失的上下文中恢复。

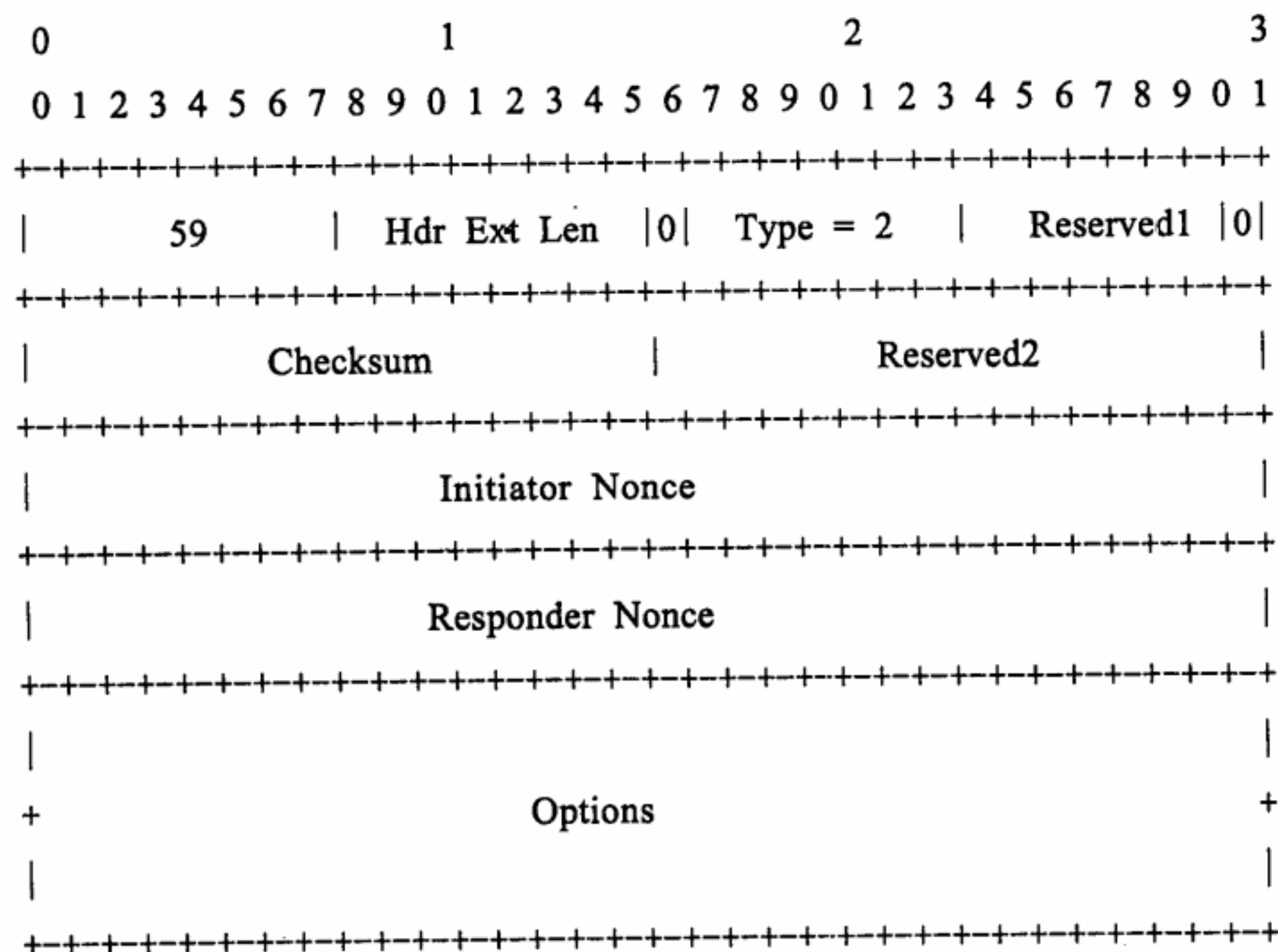
分叉实例标识符 (FII) :

当为相同的ULID对创建了另一个上下文实例时，则必须要包含FII，用来区分新的和已经存在的实例。

未来的协议扩展可能会定义该报文的附加选项。在选项格式中的C-bit定义了一个执行如何处理这种新选项，具体见5.16。

5.6 R1 报文格式

R1报文是上下文建立交互中的第二条报文。应答方用它来作为I1报文的回应，不用给发起方创建特定的状态。R1报文格式如下：



字段:

Next Header: NO_NXT_HDR (59).

Hdr Ext Len: 至少值为1，因为当没有选项时头长度为16byte。

类型：2。

保留1 (Reserved1)：7位字段。以备后用。传输时值为0。在接收时一定要忽略。

保留2 (Reserved2)：16位字段。以备后用。传输时值为0。在接收时一定要忽略。

发起者随机数 (Initiator Nonce)：

47位字段。从I1报文中拷贝。

应答方随机数 (Responder Nonce)：

32位的无符号整数。应答方选择的一个随机数，发起者要在I2报文中返回之。

以下的选项是为该报文定义的：

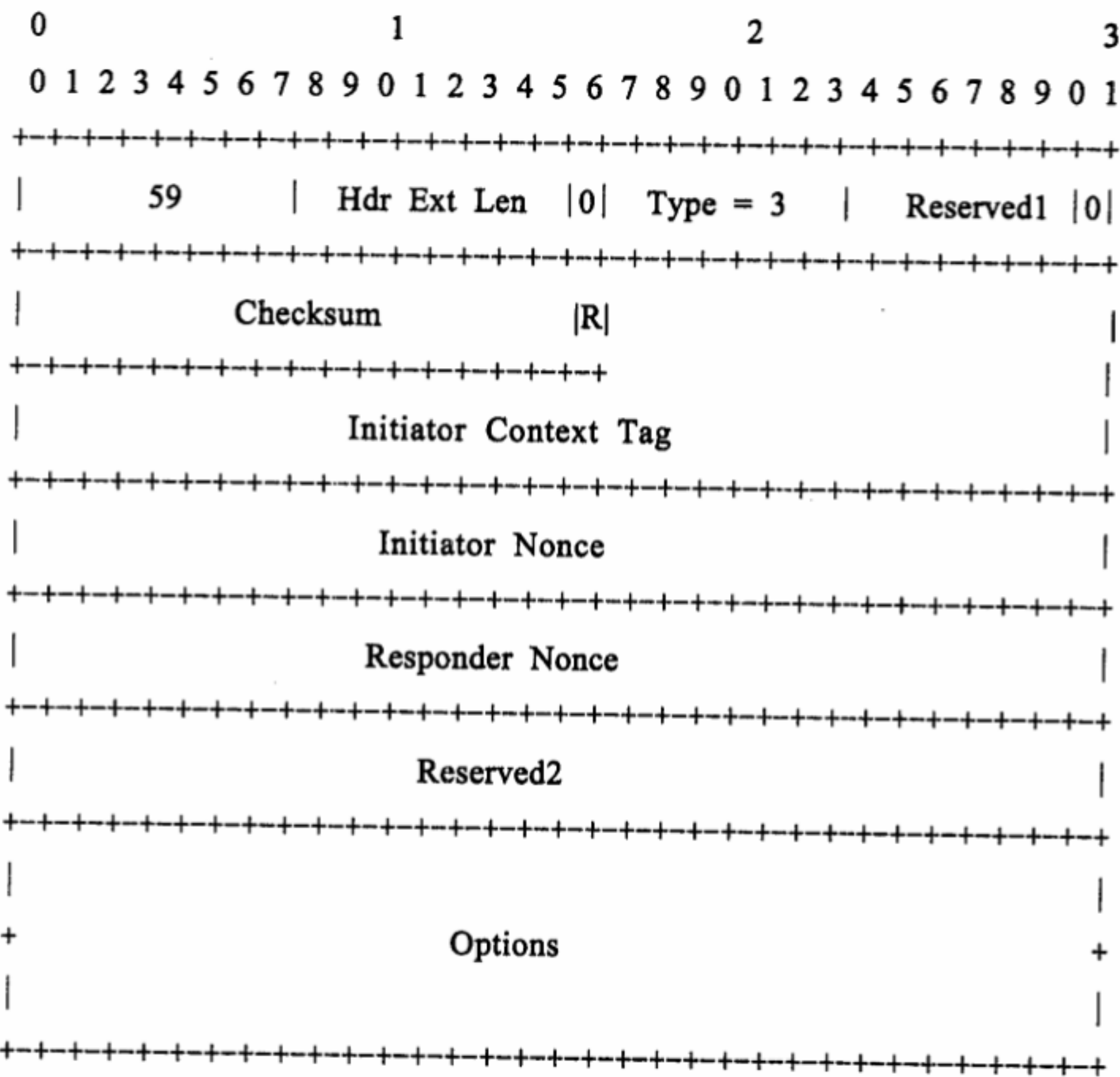
应答方验证符 (Responder Validator)：

可变长的选项。R1报文必须包含该选项。一般它会包含一个由应答方生成的哈希值，应答方用它和应答方随机数一起来验证一个I2报文的发送确实用于应答一个R1报文，且在I2报文中的参数和在I1报文中的是一样的。

未来的协议扩展可能会定义该报文的附加选项。在选项格式中的C-bit定义了一个执行如何处理这种新选项，具体见5.16。

5.7 I2 报文格式

I2报文是上下文建立交互中的第三条报文。发送者发送它作为核对过发送者随机数等之后对R1报文的回应。I2报文格式如下：



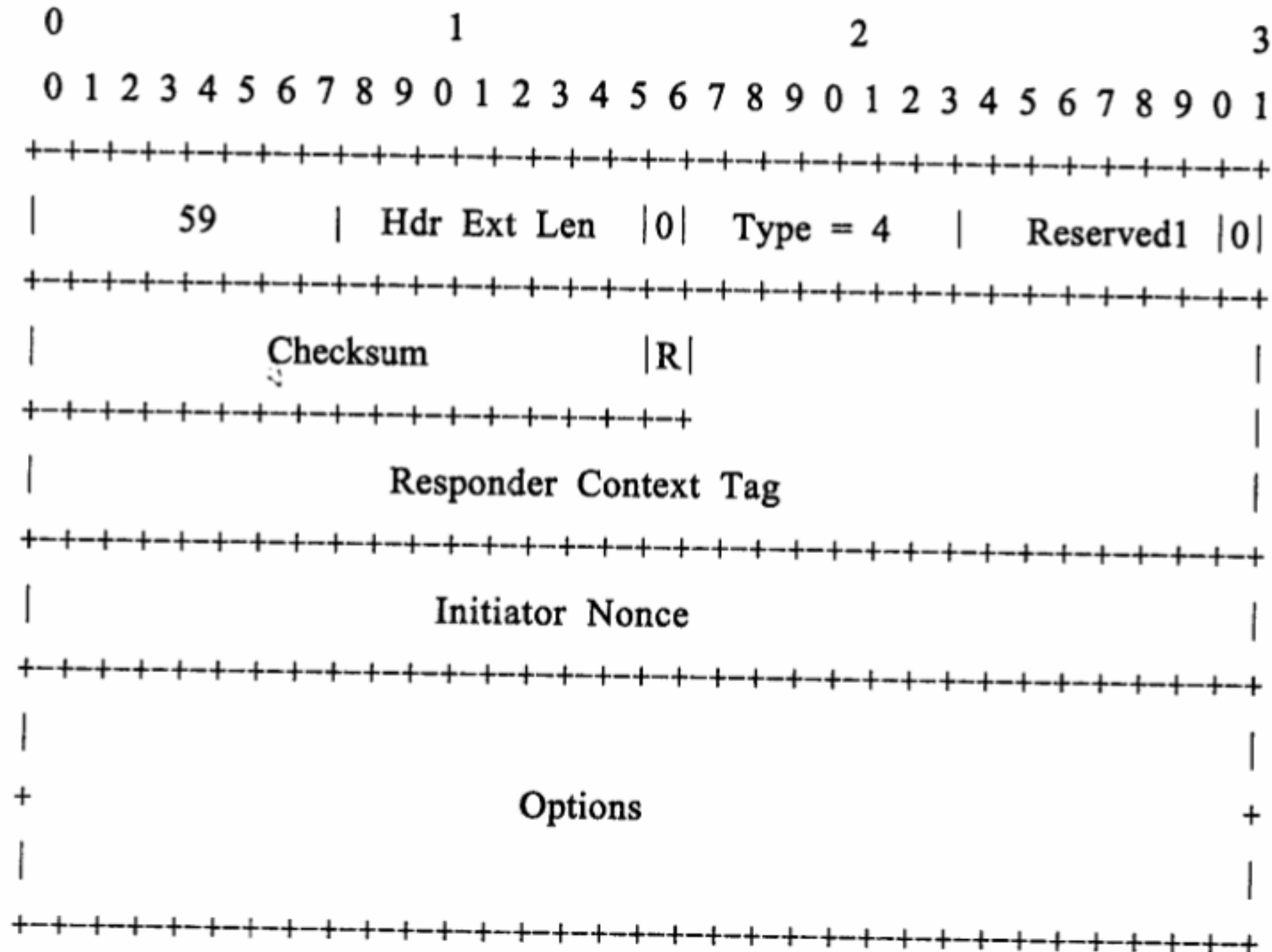
字段：

- Next Header:** NO_NXT_HDR (59)。
- Hdr Ext Len:** 至少值为2，因为当没有选项时头部为24byte。
- 类型:** 3。
- 保留1 (Reserved1):** 7位字段。以备后用。传输时值为0.在接收时一定要忽略。
- R:** 1bit字段。以备后用。传输时值为0.在接收时一定要忽略。
- 发起者上下文标签 (Initiator Context Tag):**
47位字段。发起者所有的上下文标签，分配给该上下文。
- 发起者随机数 (Initiator Nonce):**
32位无符号整数。发起者选择的一个随机数，应答方要在R2报文中返回之。
- 应答方随机数 (Responder Nonce):**
32位的无符号整数。从R1报文中拷贝的。
- 保留2 (Reserved2):** 32位字段。以备后用。传输时值为0。在接收时一定要忽略(需要让选项在8byte整数倍边界开始)。
- 以下的选项是为该报文定义的:
- 接收方验证符 (Responder Validator):**
可变长的选项。I2报文必须包含该选项，且必须是要通过拷贝在R1报文中接收到的接收方验证符选项来产生。
- ULID pair:** 当在IPv6头中的IPv6源和目的地址不能和ULID对匹配时，则必须要包含此选项。一个例子就是从一个丢失的上下文中恢复。
- 分叉实例标识符(FII):** 当一个已存在的上下文的另一个使用相同ULID对的实例创建时，则必须要包含FII，用来区分新的用例和已经存在的。
- 定位符列表 (Locator List):**
当发起方想要立即告诉接收方它的定位符列表时发送。当它发送时，用于确认定位符表的HBA/CGA信息也必须被包含。
- 定位符优先级 (Locator Preferences):**
当定位符没有相同优先级时随意发送。
- CGA参数数据结构 (CGA Parameter Data Structure):**
当定位符列表包含在I2报文中，该选项一定需要被包含以使得接收方能够确认定位符列表。
- CGA签名 (CGA Signature):**
当一些在列表中的定位符使用CGA（而不是HBA）来确认时，该选项必须要被包含在I2报文中。

未来的协议扩展可能会定义该报文的附加选项。在选项格式中的C-bit定义了一个执行如何处理这种新选项，具体见5.16。

5.8 R2 报文格式

R2报文是上下文建立交互中的第四条报文。接收方发送它作为对I2报文的回复。R2报文也用于当双方主机同时发送I1使I1交叉的情况。R2报文格式如下:



字段:

- Next Header: NO_NXT_HDR (59)。
- Hdr Ext Len: 至少值为1，因为当没有选项时头长度为16byte。
- 类型: 4。
- 保留1 (Reserved1): 7位字段。以备后用。传输时值为0.在接收时一定要忽略。
- R: 1bit字段。以备后用。传输时值为0.在接收时一定要忽略。
- 接收方上下文标签 (Responder Context Tag):

47位字段。接收方分配给该上下文的上下文标签。

发送方随机数(Initiator Nonce):

32位无符号整数。从I2中拷贝而来。

以下的选项是为该报文定义的:

定位符列表 (Locator List):

当发起方立即想要告诉接收方它的定位符列表时随意发送。当它发送时，用于确认定位符表的HBA/CGA信息也必须被包含。

定位符优先级 (Locator Preferences):

当定位符没有相同优先级时随意发送。

CGA参数数据结构 (CGA Parameter Data Structure):

当定位符消息包含在I2报文中，该选项一定需要被包含以使得接收方能够确认定位符列表。

CGA签名 (CGA Signature):

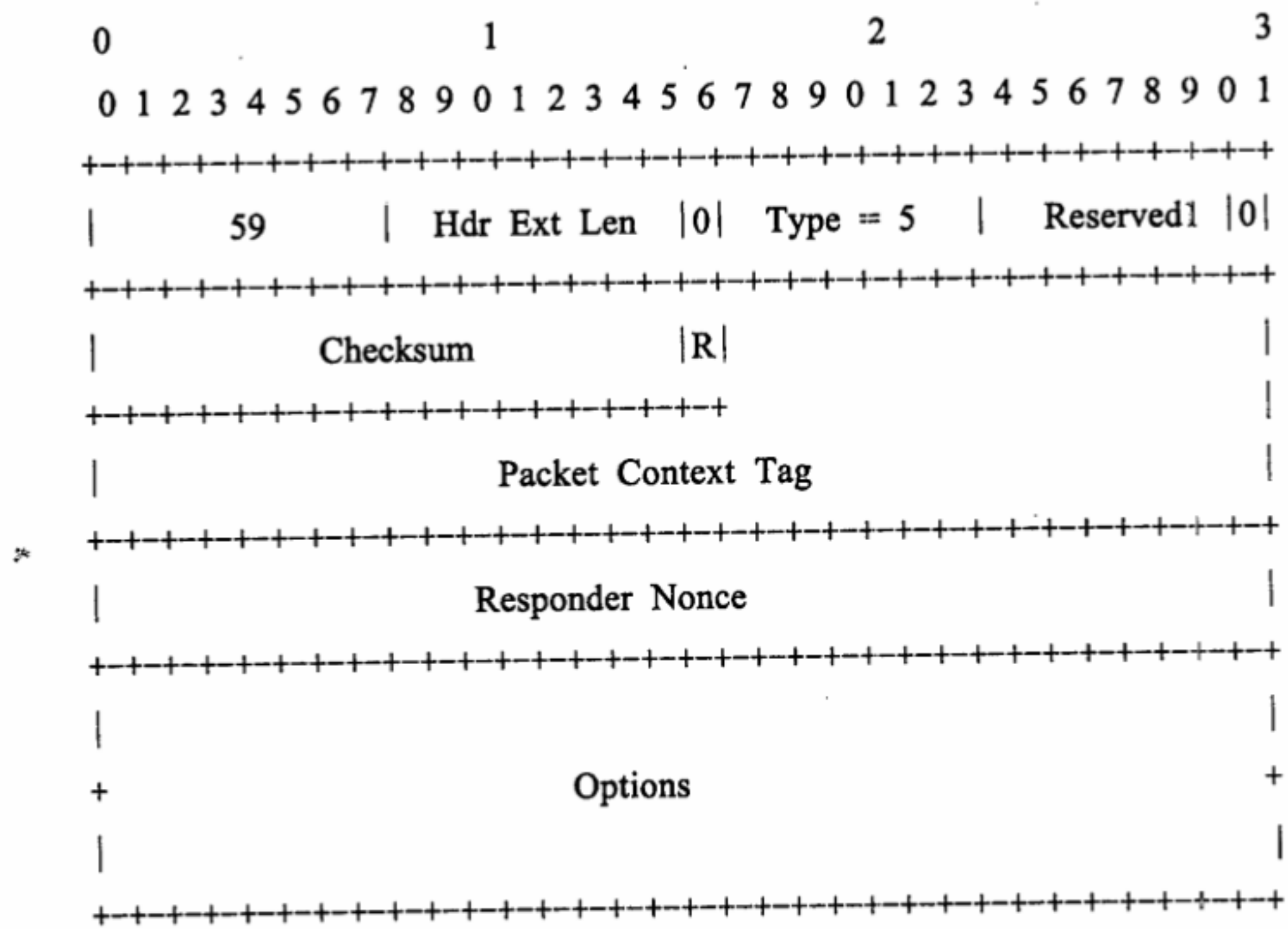
当一些在列表中的定位符使用CGA (而不是HBA) 来确认时，该选项必须要被包含在I2报文中。

未来的协议扩展可能会定义该报文的附加选项。在选项格式中的C-bit定义了一个执行如何处理这种新选项，具体见5.16。

5.9 R1bis 报文格式

如果主机受到一个带有shim6净荷扩展头或shim6控制报文的分组且类型号为64~127（比如一个更新或探测报文），且该主机没有收到的标签号对应的上下文状态，那么它将产生一个R1bis报文。

这个报文允许涉及不存在的状态的分组的发送者使用简化的上下文建立交互来重新建立上下文。取决于所收到的R1bis，接收方能够通过直接发送I2bis报文来处理重建丢失的上下文。R1bis报文格式如下：



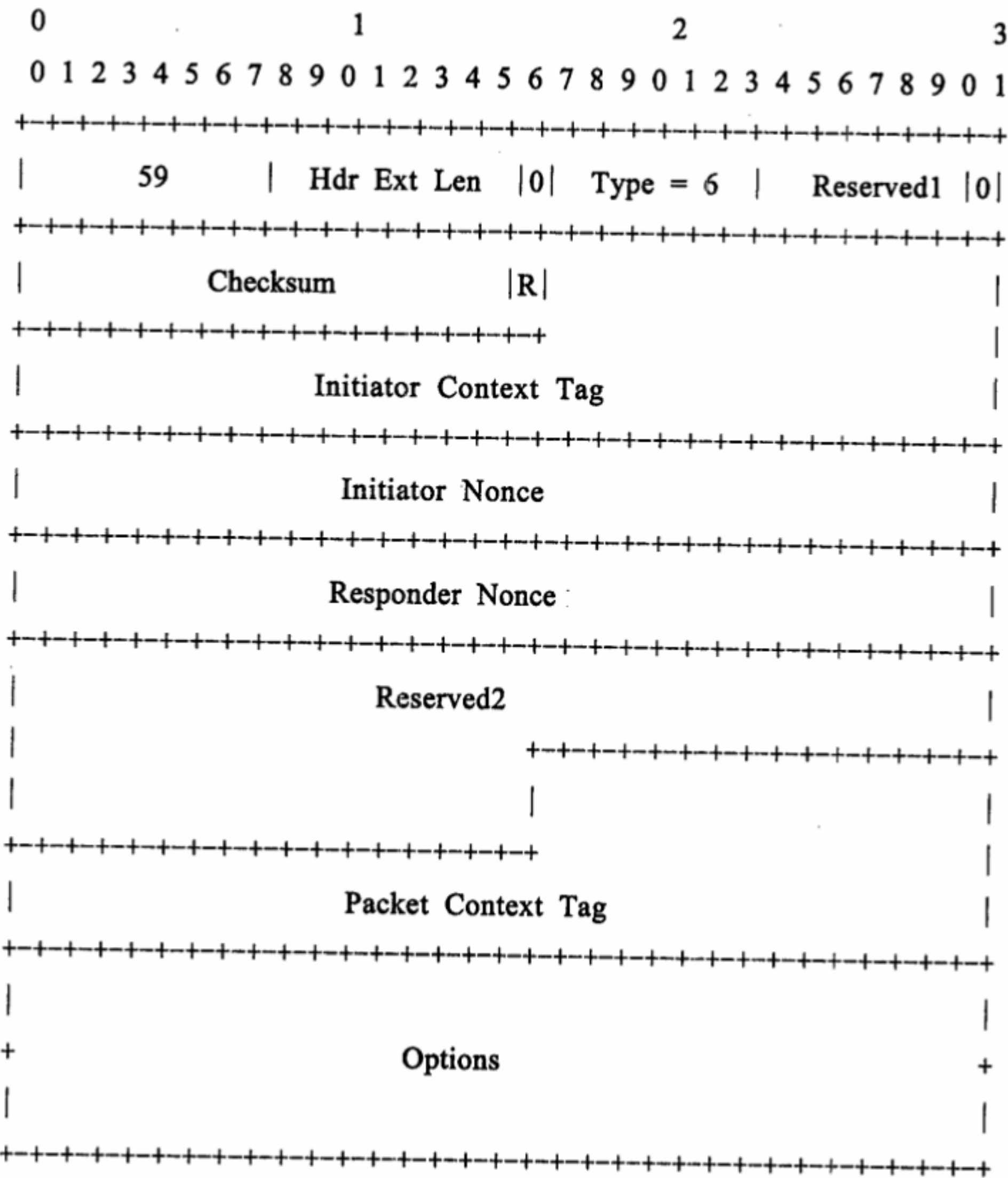
- 字段：
- Next Header: NO_NXT_HDR (59).
 - Hdr Ext Len: 至少值为1，因为当没有选项时头长度为16byte。
 - 类型: 5。
 - 保留1 (Reserved1) : 7位字段。以备后用。传输时值为0.在接收时一定要忽略。
 - R: 1bit字段。以备后用。传输时值为0.在接收时一定要忽略。
 - 分组上下文标签 (Packet Context Tag) : 47位无符号整数。该上下文标签包含在接收到的触发生成R1bis的分组里。
 - 接收方随机数 (Responder Nonce) : 32位无符号随机数。由接收方选的一个数字，发起方将会会在I2bis中返回。
- 以下的选项是为该报文定义的：
- 应答方验证符 (Responder Validator) : 可变长的选项。一般它会包含一个由应答方生成的哈希值，应答方用它和应答方随机数一起来验证一个I2bis报文的发送确实用于应答一个

R1bis报文。

未来的协议扩展可能会定义该报文的附加选项。在选项格式中的C-bit定义了一个执行如何处理这种新选项，具体见5.16。

5.10 I2bis 报文格式

I2bis报文是在上下文恢复交互中的第三条报文。它的发送是在确认R1bis报文涉及到存在的上下文等之后，对R1bis的回应。I2bis报文格式如下：



字段：

- Next Header: NO_NXT_HDR (59)。
- Hdr Ext Len: 至少值为3，因为当没有选项时头长度为32byte。
- 类型: 6。
- 保留1 (Reserved1) : 7位字段。以备后用。传输时值为0.在接收时一定要忽略。
- R: 1bit字段。以备后用。传输时值为0.在接收时一定要忽略。
- 发起者上下文标签 (Initiator Context Tag) : 47位字段。发起者所有的上下文标签，分配给该上下文。

发起者随机数 (Initiator Nonce) :

32位无符号整数。发起者选择的一个随机数, 应答方要在R2报文中返回之。

应答方随机数 (Responder Nonce) :

32位的无符号整数。从R1bis报文中拷贝的。

保留2 (Reserved2) : 32位字段。以备后用。传输时值为0。在接收时一定要忽略 (注意17位是不够的, 因为选项需要在8byte整数倍边界开始)。

分组上下文标签 (Packet Context Tag) :

47位无符号整数。从收到的R1bis中拷贝。

以下的选项是为该报文定义的:

应答方验证符 (Responder Validator) :

可变长的选项。仅是在R1bis中应答方验证符的一个拷贝。

ULID pair:

当在IPv6头中的IPv6源和目的地址不能和ULID对匹配时, 则必须要包含此选项。

分叉实例标识符:

当一个已存在的上下文的另一个使用相同ULID对的实例创建时, 则必须要包含FII, 用来区分新的用例和已经存在的。

定位符列表 (Locator List) :

当发起方立即想要告诉接收方它的定位符列表时随意发送。当它发送时, 用于确认定位符表的HBA/CGA信息也必须被包含。

定位符优先级 (Locator Preferences) :

当定位符没有相同优先级时随意发送。

CGA参数数据结构 (CGA Parameter Data Structure) :

当定位符消息包含在I2报文中, 该选项一定需要被包含以使得接收方能够确认定位符列表。

CGA签名 (CGA Signature) :

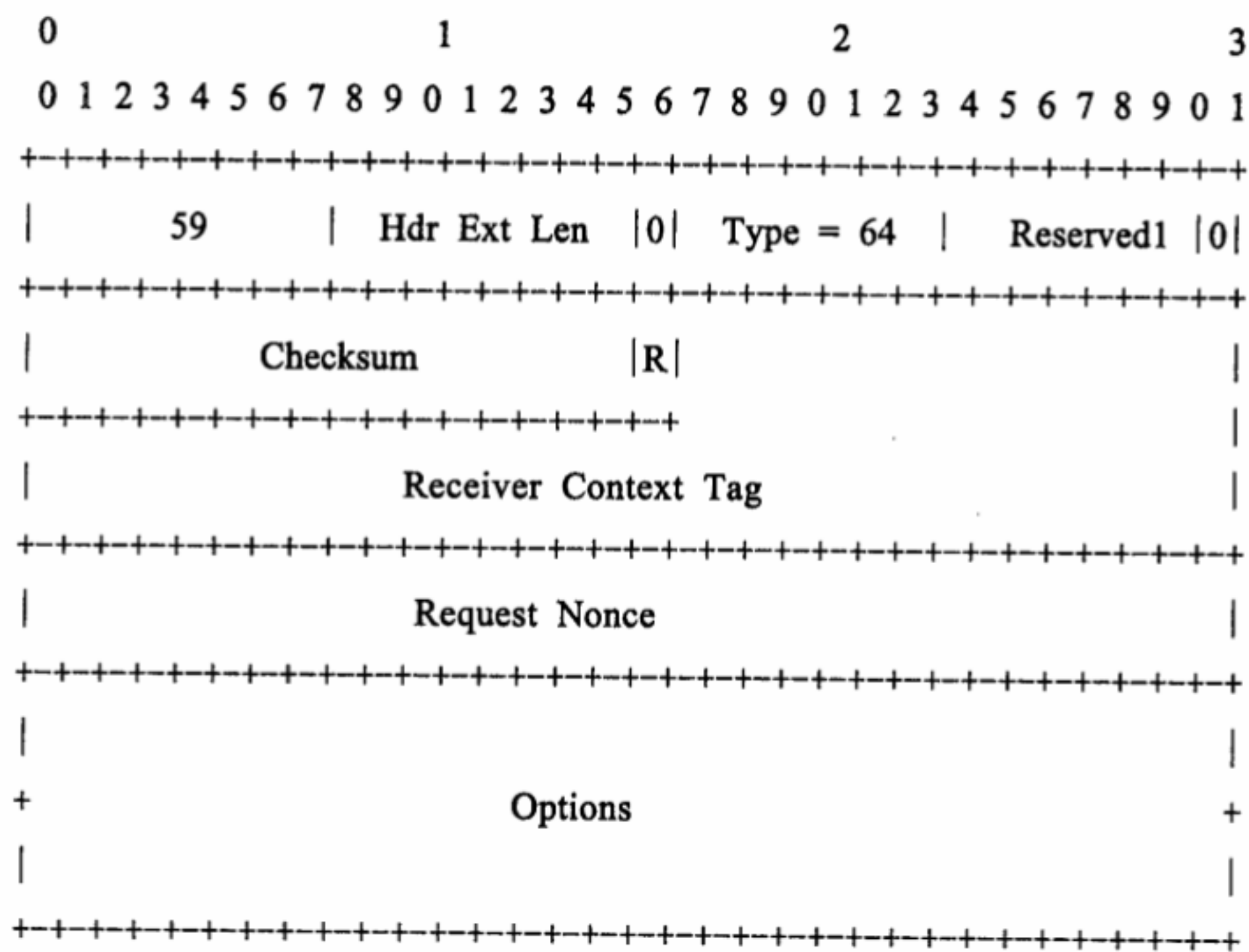
当一些在列表中的定位符使用CGA (而不是HBA) 来确认时, 该选项必须要被包含在I2报文中。

未来的协议扩展可能会定义该报文的附加选项。在选项格式中的C-bit定义了一个执行如何处理这种新选项, 具体见5.16。

5.11 更新请求报文格式

更新请求报文用于更新定位符列表, 或定位符优先级或两者。当定位符列表进行了更新, 该报文也包含了保护它的所需的HBA/CGA选项。用于阻止路径分离攻击者生成假的更新信息的基本完整性检查就是该报文中的上下文标签。

更新消息包含的选项 (定位符表和定位符优先级), 会完全替代各自原有的定位符表和优先级。因此, 没有仅仅去向定位符表发送一个累加备份的机制。更新请求报文格式如下:



字段:

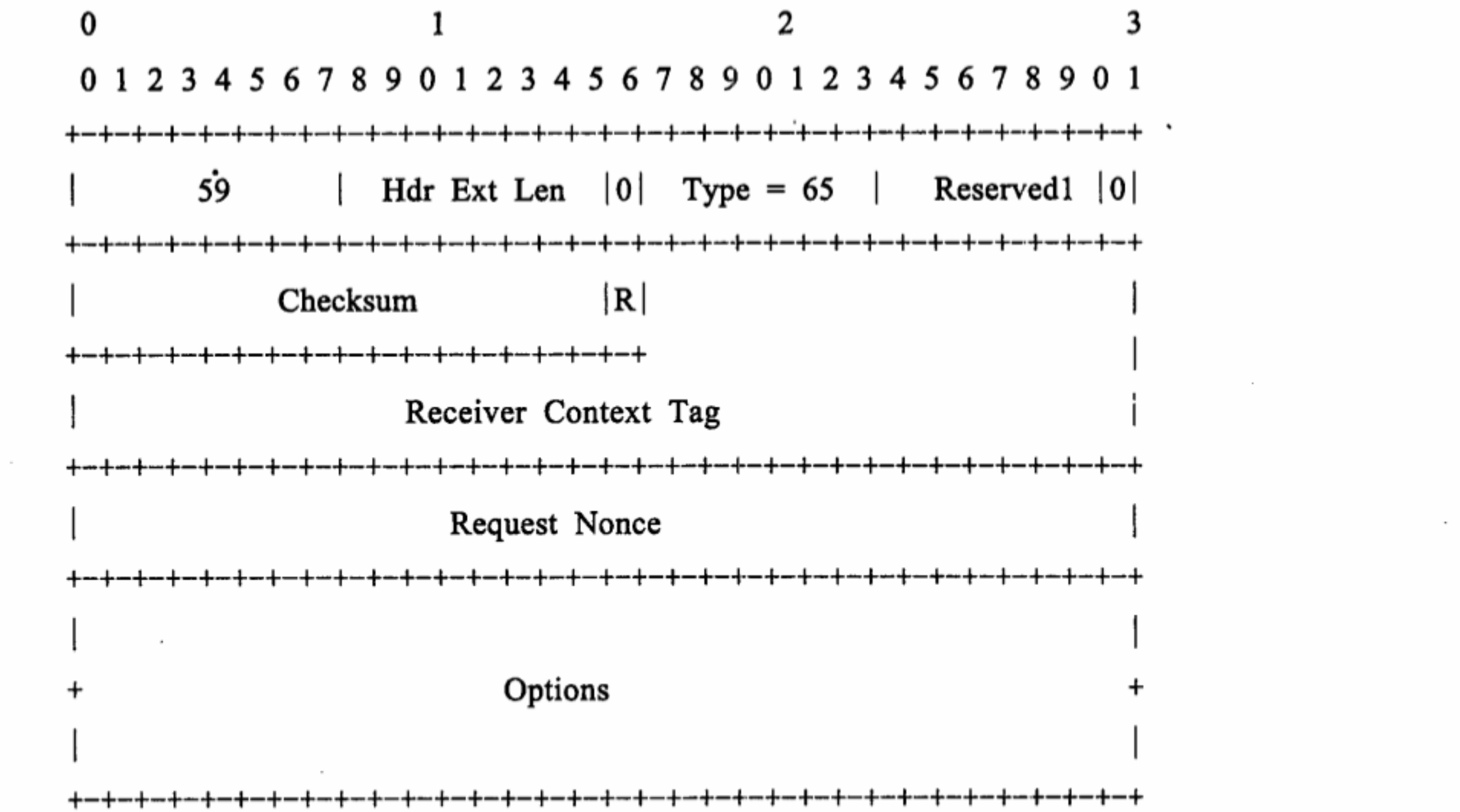
- Next Header: NO_NXT_HDR (59).
- Hdr Ext Len: 至少值为1, 因为当没有选项时头长度为16byte.
- 类型: 64.
- 保留1 (Reserved1): 7位字段。以备后用。传输时值为0.在接收时一定要忽略。
- R: 1bit字段。以备后用。传输时值为0.在接收时一定要忽略。
- 接收方上下文标签 (Receiver Context Tag): 47位字段。接收方分配给该上下文的上下文标签。
- 请求随机数 (Request Nonce): 32位无符号整数。发起者选择的一个随机数, 对端要在更新确认报文中返回。
- 以下的选项是为该报文定义的:
- 定位符列表 (Locator List): 发送者的 (新的) 定位符表。可能定位符没有改变而仅是优先级发生了改变。
- 定位符优先级 (Locator Preferences): 当定位符没有相同优先级时随意发送。
- CGA参数数据结构 (CGA Parameter Data Structure): 当I2/I2bis/R2报文包含定位符表而不包含PDS时, 需要CGA PDS, 这样接收方才能确认定位符列表。
- CGA签名 (CGA Signature): 当一些在列表中的定位符使用CGA (而不是HBA) 来确认时, 该选项必

须要被包含。

未来的协议扩展可能会定义该消息的附加选项。在选项格式中的C-bit定义了一个执行如何处理这种新选项，具体见5.16。

5.12 更新确认报文格式

该报文的发送是用于应答更新请求报文。这意味着更新请求已经收到，在更新请求中的任何的新定位符现在能够用作分组的源定位符。但并不意味着（新的）定位符已经核实可以作为目的，因为主机可能会延迟定位符的核实直到需要使用一个定位符去作为目的。更新确认报文格式如下：

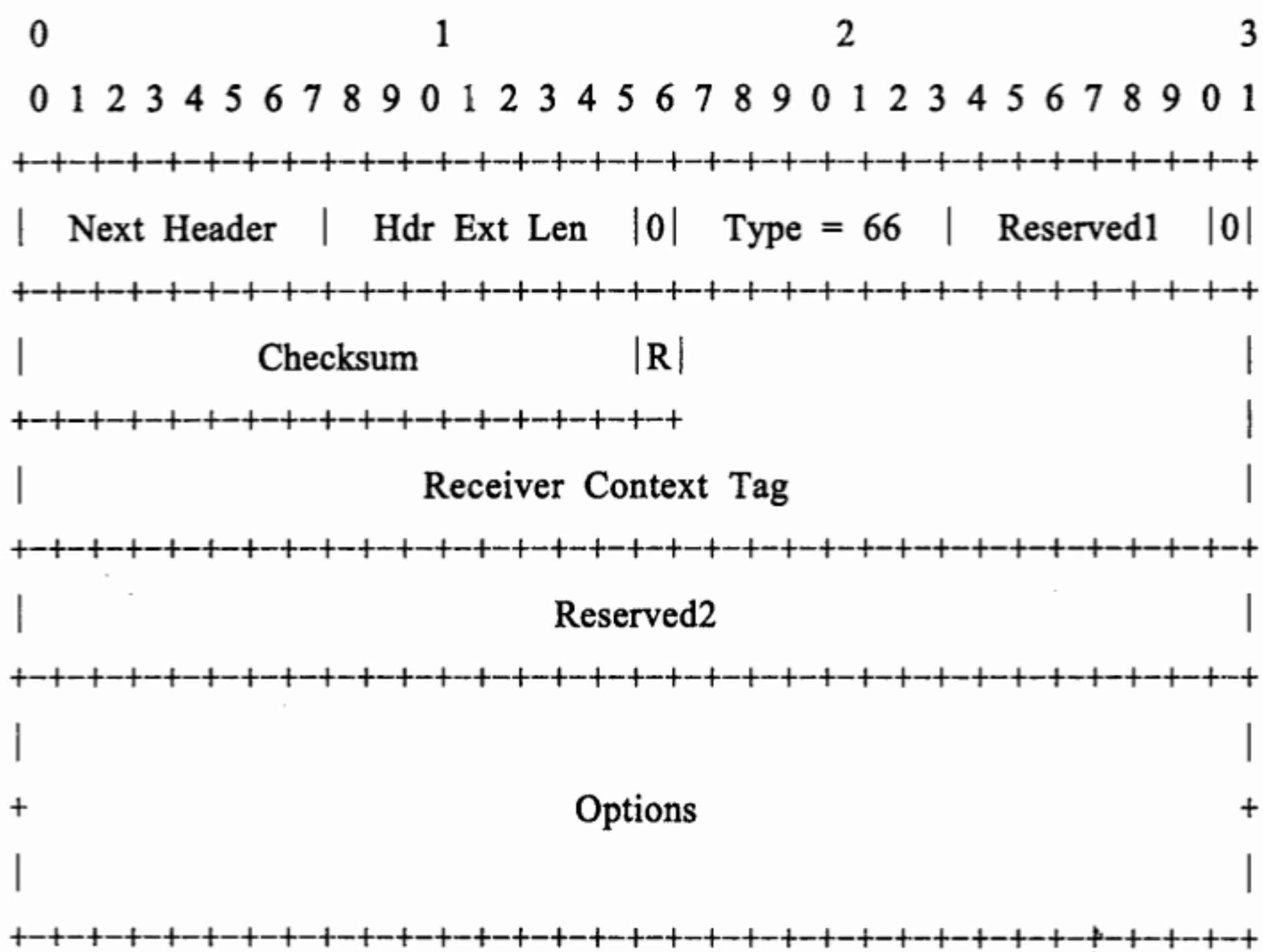


- 字段：
- Next Header: NO_NXT_HDR (59)。
 - Hdr Ext Len: 至少值为1，因为当没有选项时头长度为16byte。
 - 类型: 65。
 - 保留1 (Reserved1): 7位字段。以备后用。传输时值为0.在接收时一定要忽略。
 - R: 1bit字段。以备后用。传输时值为0.在接收时一定要忽略。
 - 接收方上下文标签 (Receiver Context Tag): 47位字段。接收方分配给该上下文的上下文标签。
 - 请求随机数 (Request Nonce): 32位无符号整数。从更新请求报文总拷贝而来。
- 该报文目前没有定义选项。
- 未来的协议扩展可能会定义该消息的附加选项。在选项格式中的C-bit定义了一个执行如何处理这种新选项，具体见5.16。

5.13 保活（Keepalive）报文格式

该报文用于确定当对端以某一上下文发送ULP分组，它也会在反方向收到一些分组。当ULP发送双向的流则不需要额外的分组。但对一个单向的ULP流模式，shim需要在收到ULP分组时返回一些Keepalive消息。

Keepalive报文的格式如下：



Next Header, Hdr Ext Len, 0, 0, Checksum的定义见5.4。

- 类型（Type）： 这一字段标识了Keepalive报文且必须被设为66（Keepalive）。
- 保留1（Reserved1）： 一个7bit的字段，留作后用。在传输时设为0，在接收端必须被忽略。
- R： 一个1bit的字段，留作后用。在传输时设为0，在接收端必须被忽略。
- 接收端上下文标签（Receiver Context Tag）： 一个47位字段，用于接收者为该上下文分配的上下文标签。
- 保留2（Reserved2）： 一个32bit的字段，留作后用。在传输时设为0，在接收端必须被忽略。
- 选项： 这个字段可能包含一个或多个shim6选项。但目前这些定义的选项对Keepalive报文都是没有用的。该选项字段仅用于以后的扩展需求。

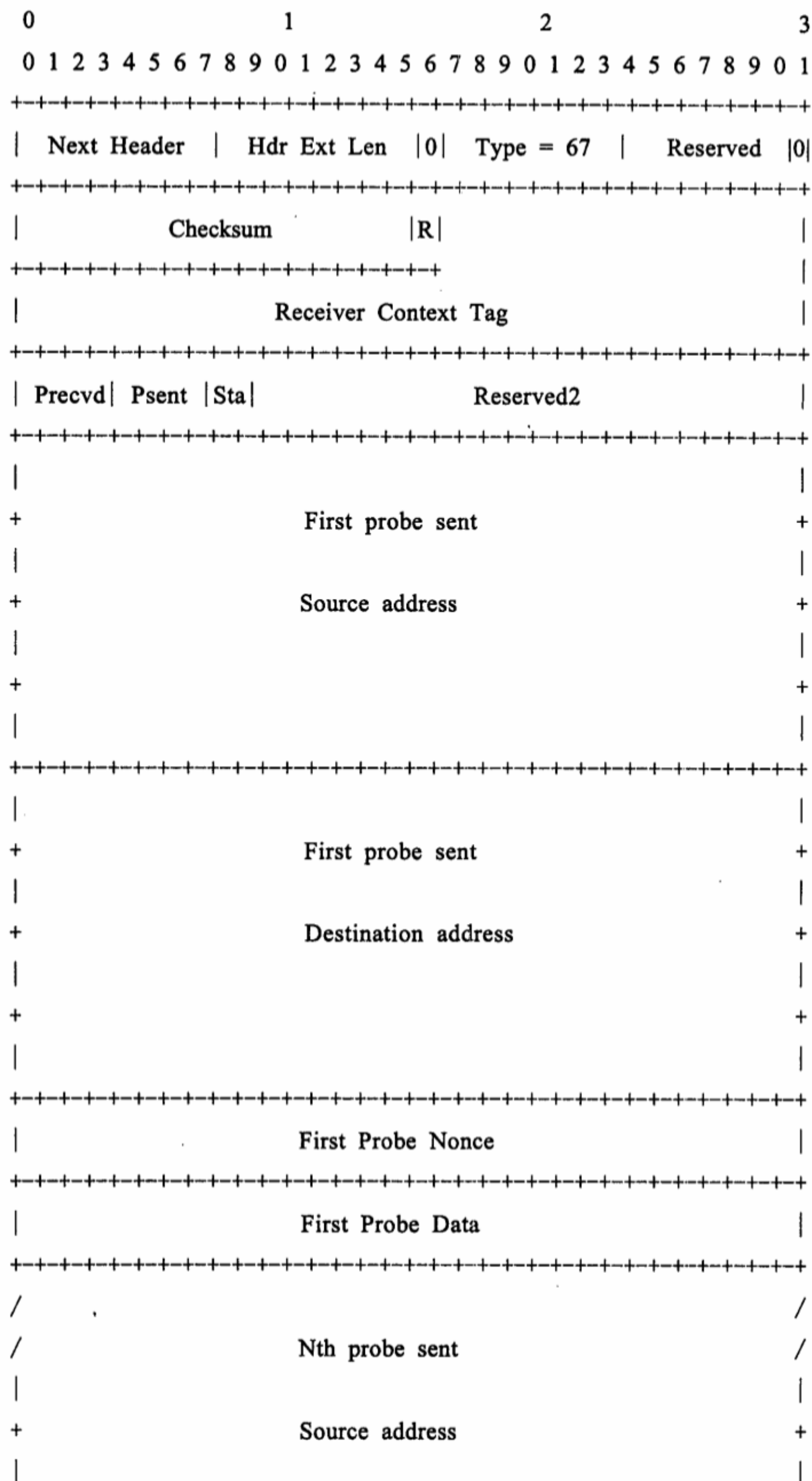
一个有效的信息符合上面的格式，有一个与接收端知道的上下文相匹配的接收端上下文标签，是一个有效的shim6控制报文，如在12.3所定义的，且有一个状态机状态是ESTABLISHED的shim6上下文。接受端通过检查它的选项以及执行为该选项指定的任何操作来处理一个有效报文。

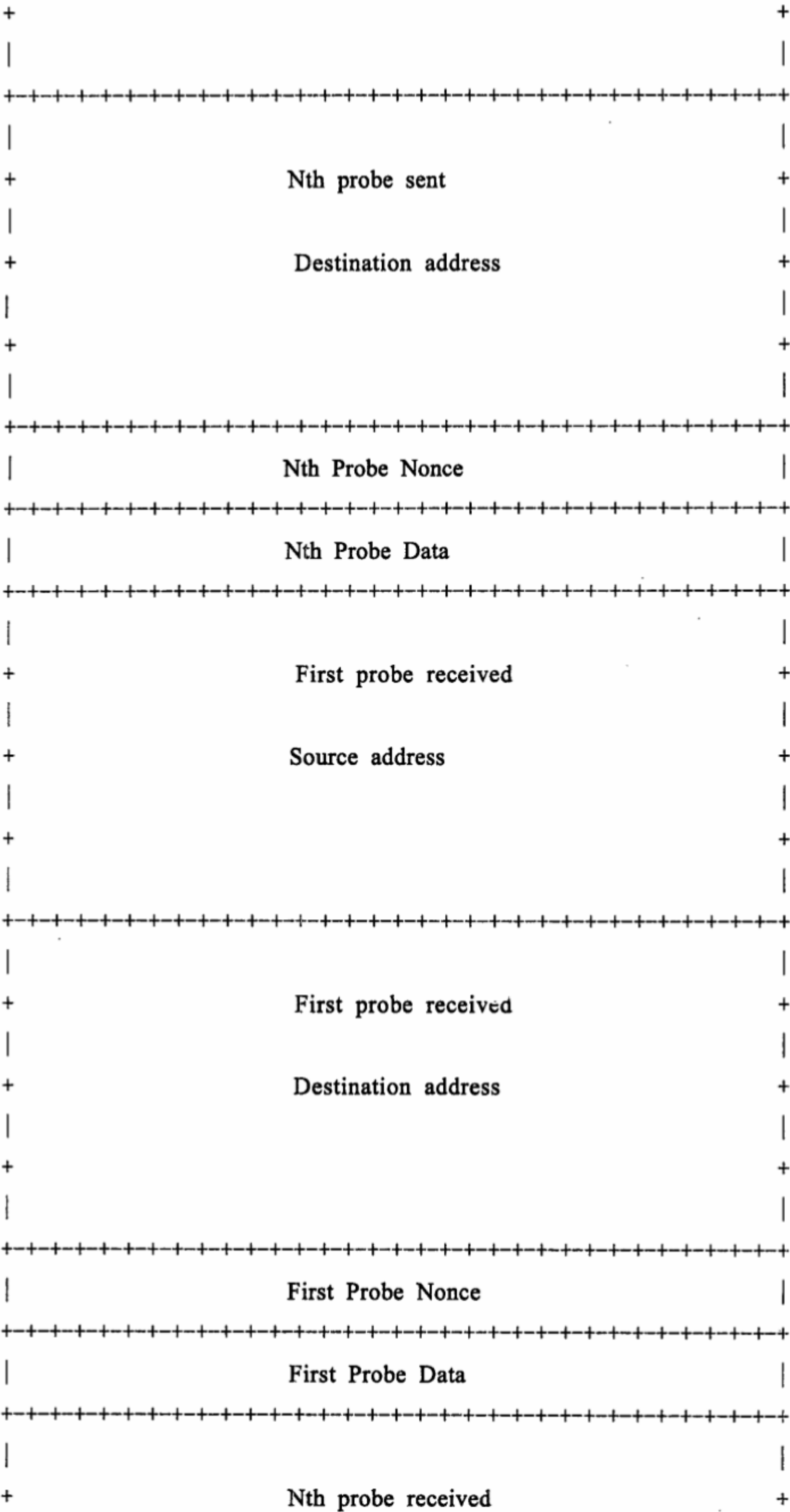
对该报文的处理规则详见14.3。

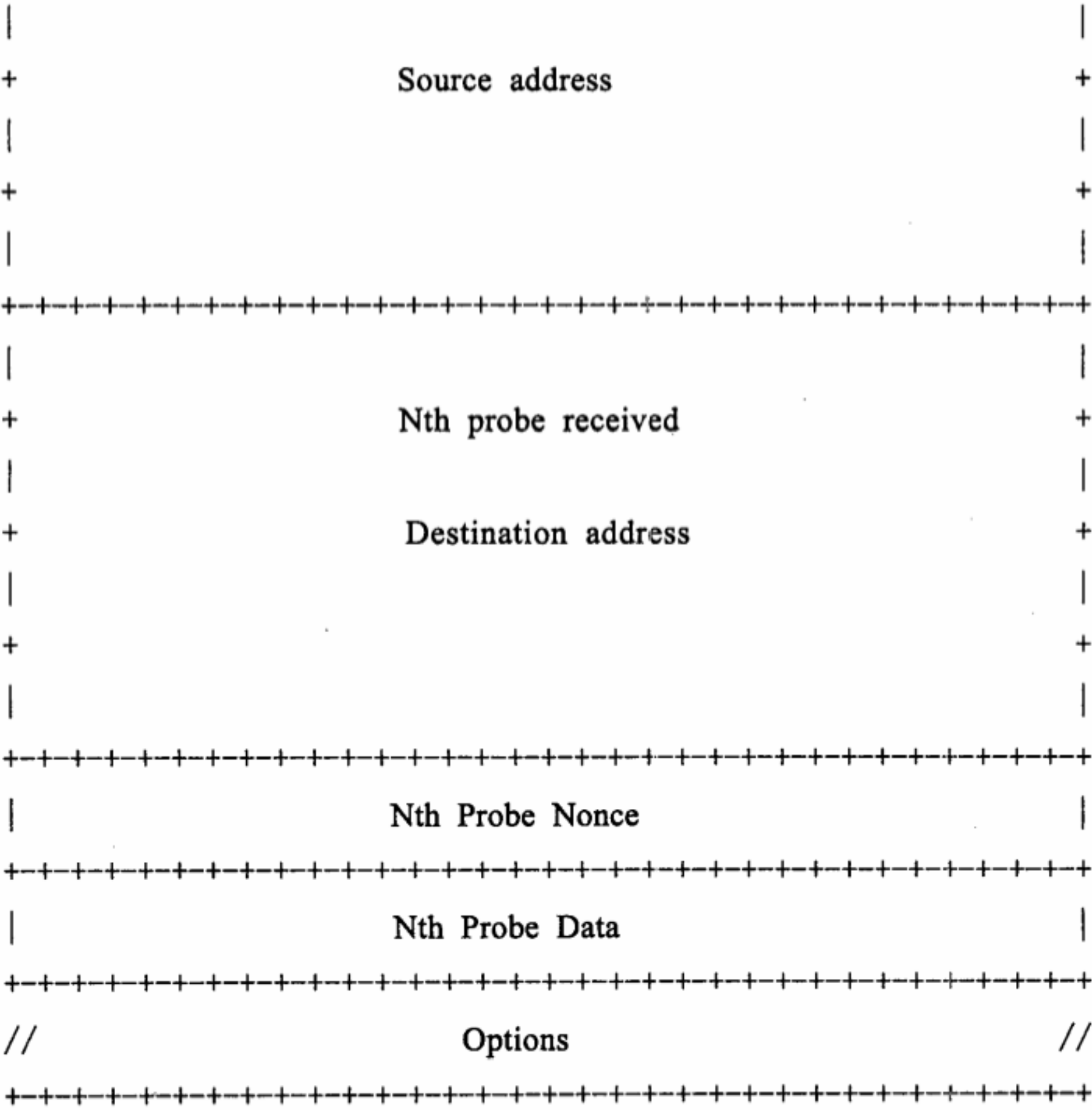
5.14 探测（Probe）报文格式

这一机制的目的是测试在一般情况下的定位符对是否工作。特别地，这一机制能够处理当一对定位符能从A到B工作，另一对定位符能从B到A工作，但没有在双向工作的定位符对的情况。该协议的机制是，A向B发送探测报文，B会观测它收到了哪些定位符对，并用探测报文向A报告。

该报文进行REAP探测。它的格式如下:







Next Header, Hdr Ext Len, 0, 0, Checksum见5.4。

- 类型 (Type) :

这一字段标识了探测报文且必须被设为67 (Probe) 。
- 保留 (Reserved) :

这是一个7bit的字段, 留作后用。在传输时设为0, 在接收端必须被忽略。
- R:

这是一个1bit的字段, 留作后用。在传输时设为0, 在接收端必须被忽略。
- 接收端上下文标签 (Receiver Context Tag) :

这是一个47位字段, 用于接收者为该上下文分配的上下文标签。
- Psent:

这是一个4位字段, 指明包含在该探测报文中的发送探测数。探测字段的第一部分涉及当前报文且必须被呈现出, 所以这一字段的最小值是1。附加的发送探测字段是早些发送的探测相同字段的拷贝, 且可能被执行的每一个任意逻辑使用包括或被忽略。
- Precv:

这是一个4位字段, 指明包含探测报文中的收到的探测数。接收探测字段是早些时候收到的探测的相同字段 (由于最近转变到状态Exploring而到达) 的拷贝。当一个发送方状态是InboundOk, 它必须包含最少一个到达的探测的该字段的拷贝。一个发送方可能包含额外的这些收到的在任意状态的探测字段作为被执行使用的每个任意的逻辑。探测源, 探测目的, 探测随机数, 以及探测数据字段可能被重复, 取决于Psent和Precv的值。
- Sta (State):

这个2位的状态字段用于通知对方发送者的当前状态。它有3个合法值:

- 0(Operational):

意味着发送方 (a) 相信它自己的通信没有问题且 (b) 相信接收的通信也没有问题;
- 1(Exploring):

意味着发送方与接收方的通信有问题, 即使它期望有一些流, 它也不能从对方看到;
- 2(InboundOk):

意味着发送方相信它的通信没有问题, 即, 它最少能看到来自接收方的分组, 但接收方要么有一个问题, 要么没有向发送方确认该问题已解决。
- 保留2 (Reserved2) :

这是一个32bit的字段, 留作后用。在传输时设为0, 在接收端必须被忽略。
- 探测源 (Probe Source) :

这个128位的字段包含用于发送探测的IPv6的源地址。
- 探测目的 (Probe Destination) :

这个128位的字段包含用于发送探测的IPv6的目的地址。
- 探测随机数 (Probe Nonce) :

这是一个32位的字段, 由发送方初始化, 使得发送方能用一个值来确定一个接收探测和哪个发送探测关联。我们非常建议随机数字段要适当的难以猜测, 这样甚至耦合攻击者也不能推断出下一个将要用的随机数值。这个值应当使用一个随机数生成器来生成, 该生成器要有IETF RFC4086中提出的好的随机性。
- 探测数据:

这是一个32位字段, 没有固定意义。探测数据字段没有改变的被拷贝回。未来的标志可能定义这个字段的用处。
- 选项 (Options) :

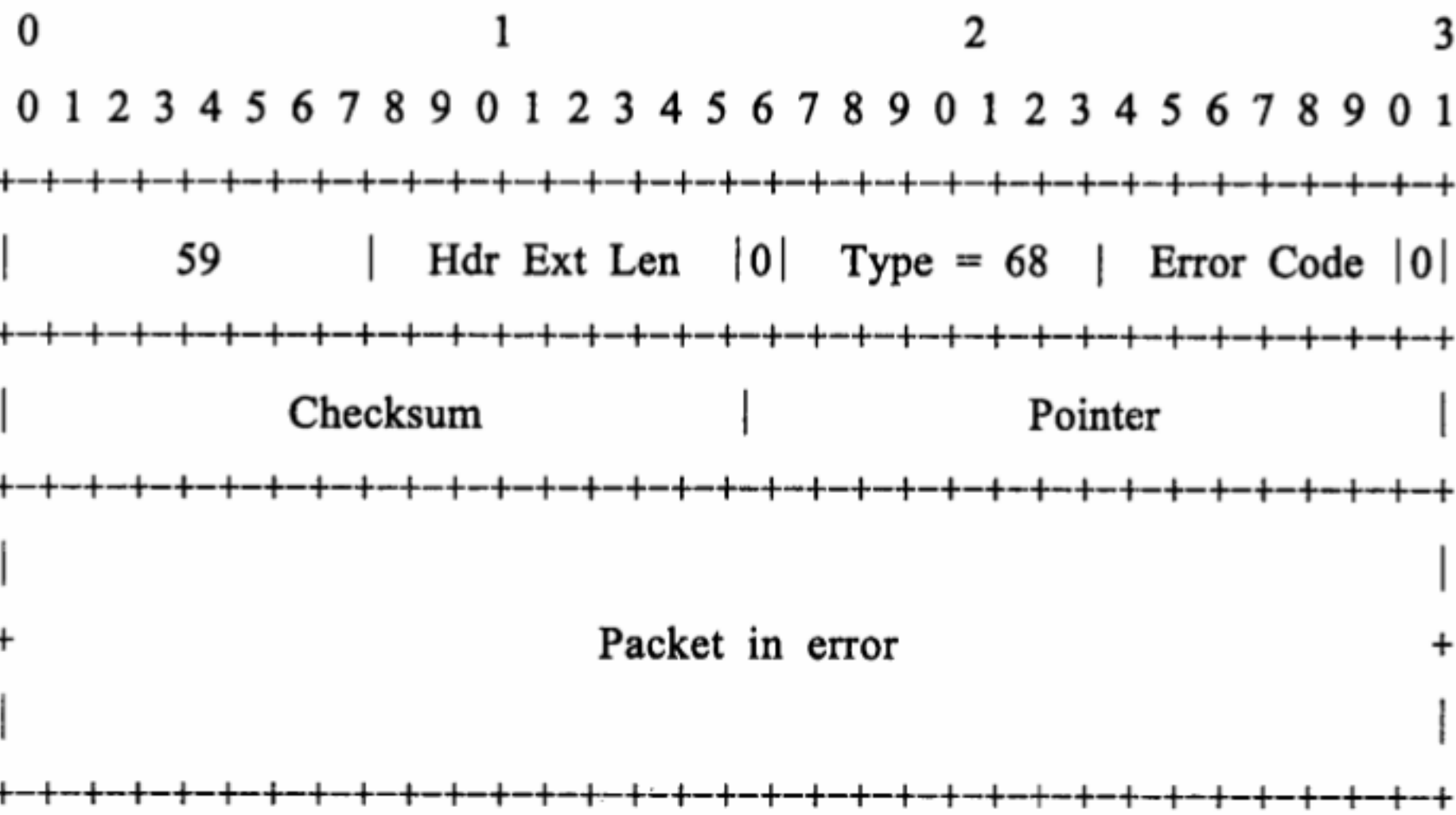
用于未来扩展。

5.15 出错报文格式

当接收方接收到的shim6消息包含不能恰当处理的错误信息时, 就会产生出错消息。

在shim6节点接收到一个包含了对shim6协议来说是错误的信息且接收者不支持的情况下, 它会发送出错消息给该shim6消息的源。出错消息不需进行确认。

另外, 在本节中定义的shim6出错消息能用于表示shim6执行的问题。为了达到这一点, 预留了一个范围的错误码类型。特别地, 执行可能会生成含有那个范围内出错码的shim6出错消息, 而不是在调试过程中默默地丢失掉那种shim6包。出错报文格式如下:



字段:

Next Header: NO_NXT_HDR (59)。

Hdr Ext Len: 至少值为1, 因为当没有选项时头长度为16byte。取决于特定的错误数据。

类型: 68。

错误码: 7bit字段, 描述产生了错误消息的错误。详见下面的错误码表。

指针: 16bit字段。标识在探测出了错误的分组中的字节偏量。

出错的分组 (Packet in error) :

在出错消息分组没有超出最小的IPv6MTU下, 尽可能多的唤起分组。

表2中定义了错误码。

表2 错误码

码值	描述
0	未知的shim6报文类型
1	关键选项未识别
2	定位符验证方法失败 (指向不符合验证方法的字节的指针)
3	不同步的定位符列表生成数
4	在按定位符优先级的多个定位符中的错误
120~127	预留作为调试的选项

5.16 选项格式

5.16.1 概述

选项的格式和当前HIP选项格式 (参见[19]) 类似。但我们并没有故意去跟踪HIP选项格式的任何改变, 也没有刻意去为选项类型值使用相同的名字空间。但使用相同的格式有可能会使得shim6引入HIP能力更容易一些, 作为shim6的一个扩展这是有帮助的。

所有的TLV参数有一个长度 (包括类型和长度字段), 即是8byte的整数倍。如果需要, 一定要在参数的末尾填充一些字符用于确保总长度为8byte的整数倍。这一规则确保数据正确对齐。如果填充了字符, 长度域一定不能包含填充部分。任何添加的字段都必须被发送者归零, 且它们的值不应被接收者检测。

因此, 长度字段指示内容字段的长度 (以字节为单位)。该TLV参数的总长度 (包括类型, 长度, 内容, 和填充符) 按下列公式与字段长度有关:

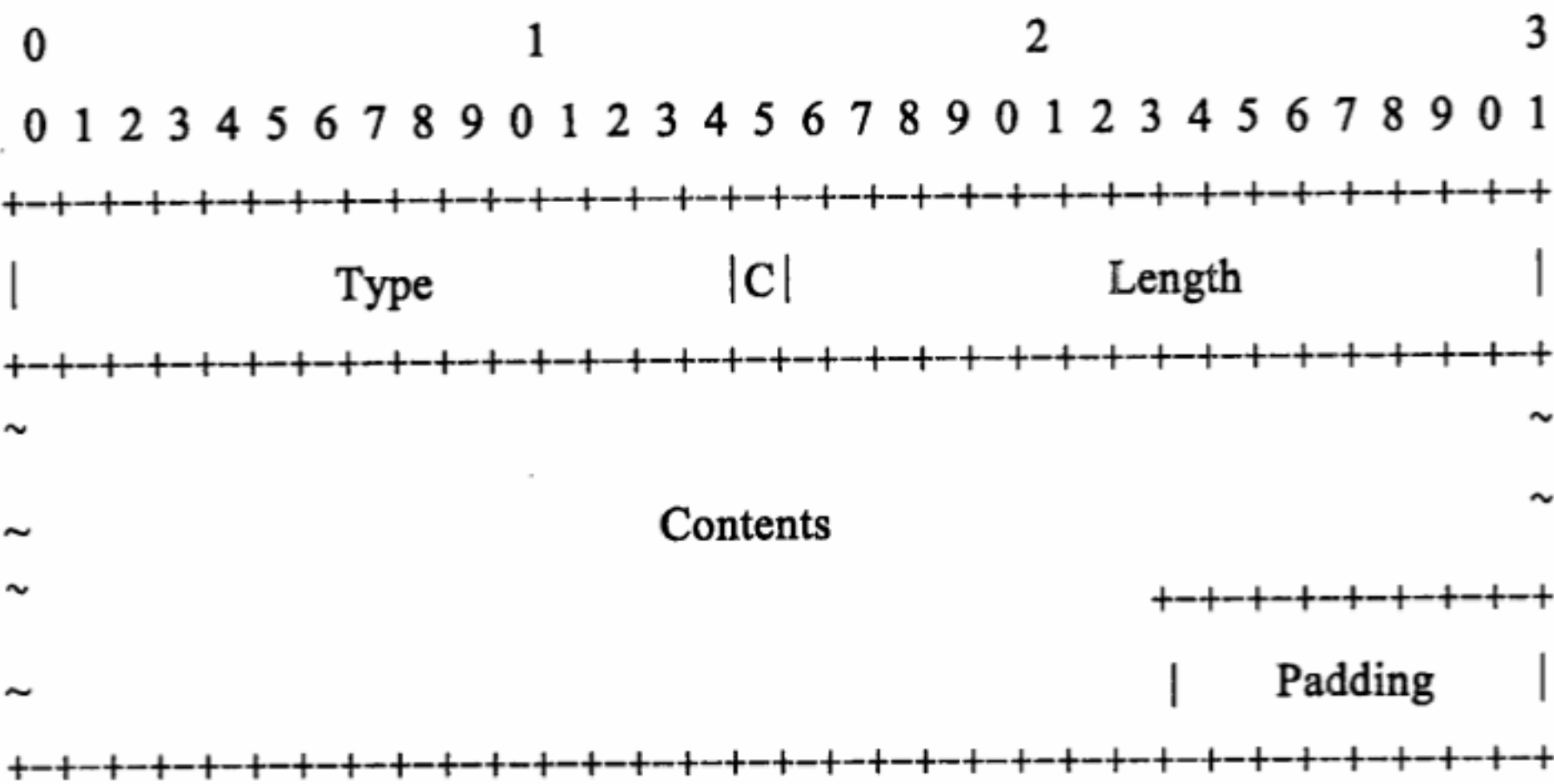
总长度 = 11 + 长度 - (长度 + 3) 模 8

选项的总长度是允许4byte选项头和选项本身的最小8byte整数倍。所需要的总的填充符可计算如下:

填充符 = 7 - ((长度 + 3) 模 8)

且:

总长度 = 4 + 长度 + 填充符



字段:

- 类型 (Type): 该类型选项的15位的标识符。在本标准中选项的定义见表3。
- C: 紧急的 (critical)。如果该参数是紧急的且必须被接收方识别则置为1; 否则为0。一个执行可能把C-bit视为类型字段的一部分, 通过把该定义中的类型值乘以2。
- 长度 (length): 内容的长度, 字节单位。
- 内容 (Contents): 指定的参数, 由字节定义。
- 填充符 (Padding): 0byte~7byte, 如果需要则添加。

表3 选项类型

类型	选项名
1	应答方验证符选项
2	定位符表选项
3	定位符优先级选项
4	CGA参数数据结构选项
5	CGA签名选项e
6	ULID对选项
7	分叉实例标识符选项
10	keepalive超时选项

未来的协议扩展可能会为shim6消息定义更多的选项。在该选项格式中的C-bit定义了一个新的选项如何被一个执行来处理。

如果一台主机收到一个它无法理解的选项 (一个在本协议中一些未来的扩展中定义的选项) 或者对以上的不同消息类型没有作为一个有效选项列出来, 则选项中的紧急位将决定输出。

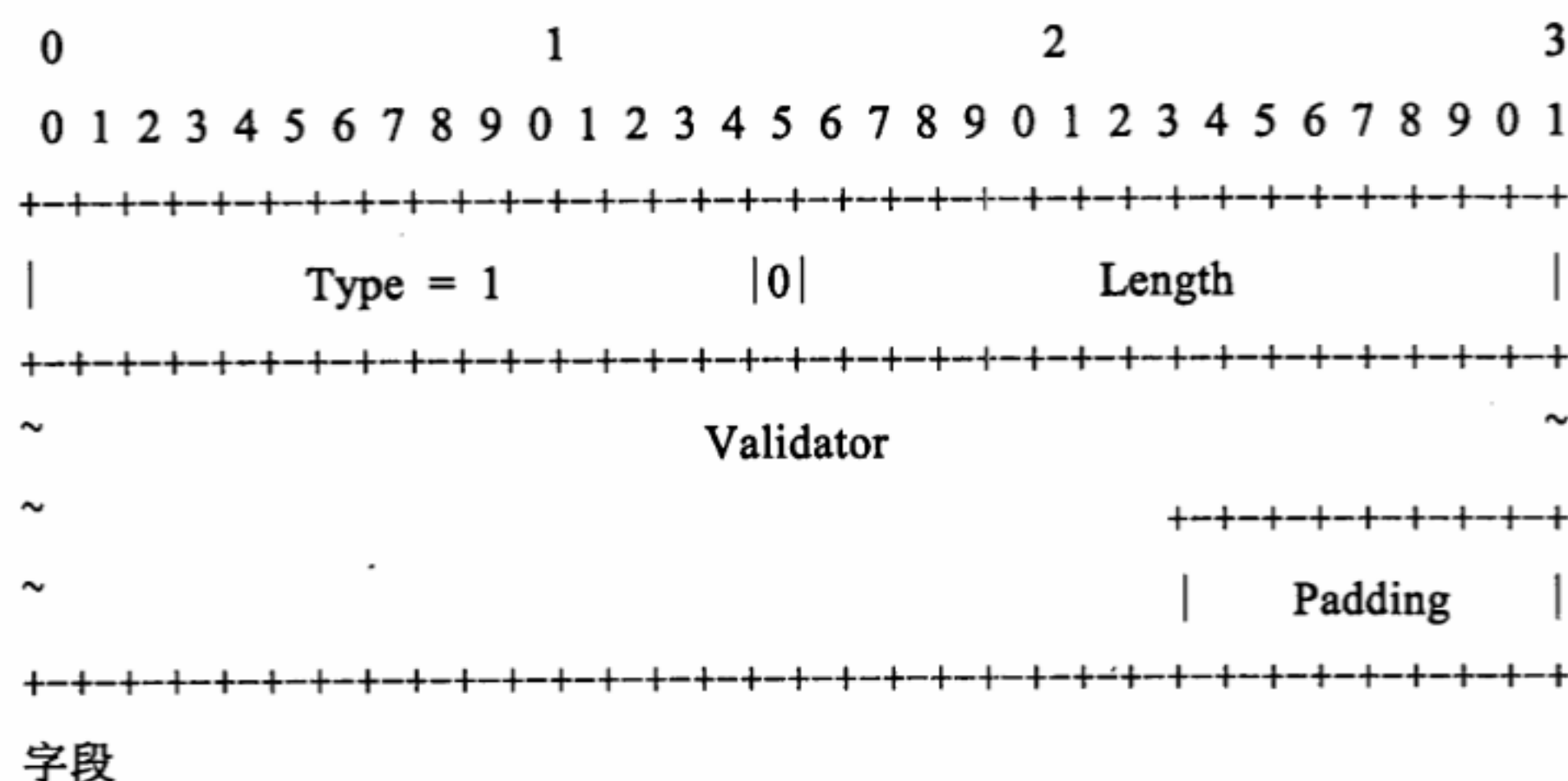
- 若 C = 0, 则该选项被忽略掉, 处理余下的消息。
- 若 C = 1, 则主机应当返回一个 shim6 出错消息, 出错码 = 1, 指针字段引用选项类型字段的头一个字节。当 C = 1 时该消息余下部分一定不能被处理。

5.16.2 应答方验证符选项格式

应答方能够准确选择用什么输入来计算验证，以及使用什么单项函数（如MD5或SHA1），只要应答方能够检测从I2或I2bis中接收到的验证符确实是同一个：

- 计算，
- 计算特殊的上下文，且
- 不是一个重复的I2/I2bis消息。

一些关于如何生成验证符的建议在7.10和7.18.2中有提到。应答方验证符选项格式如下：



验证符 (Validator)： 可变长度的内容，它的解释对应答方来说是本地的。

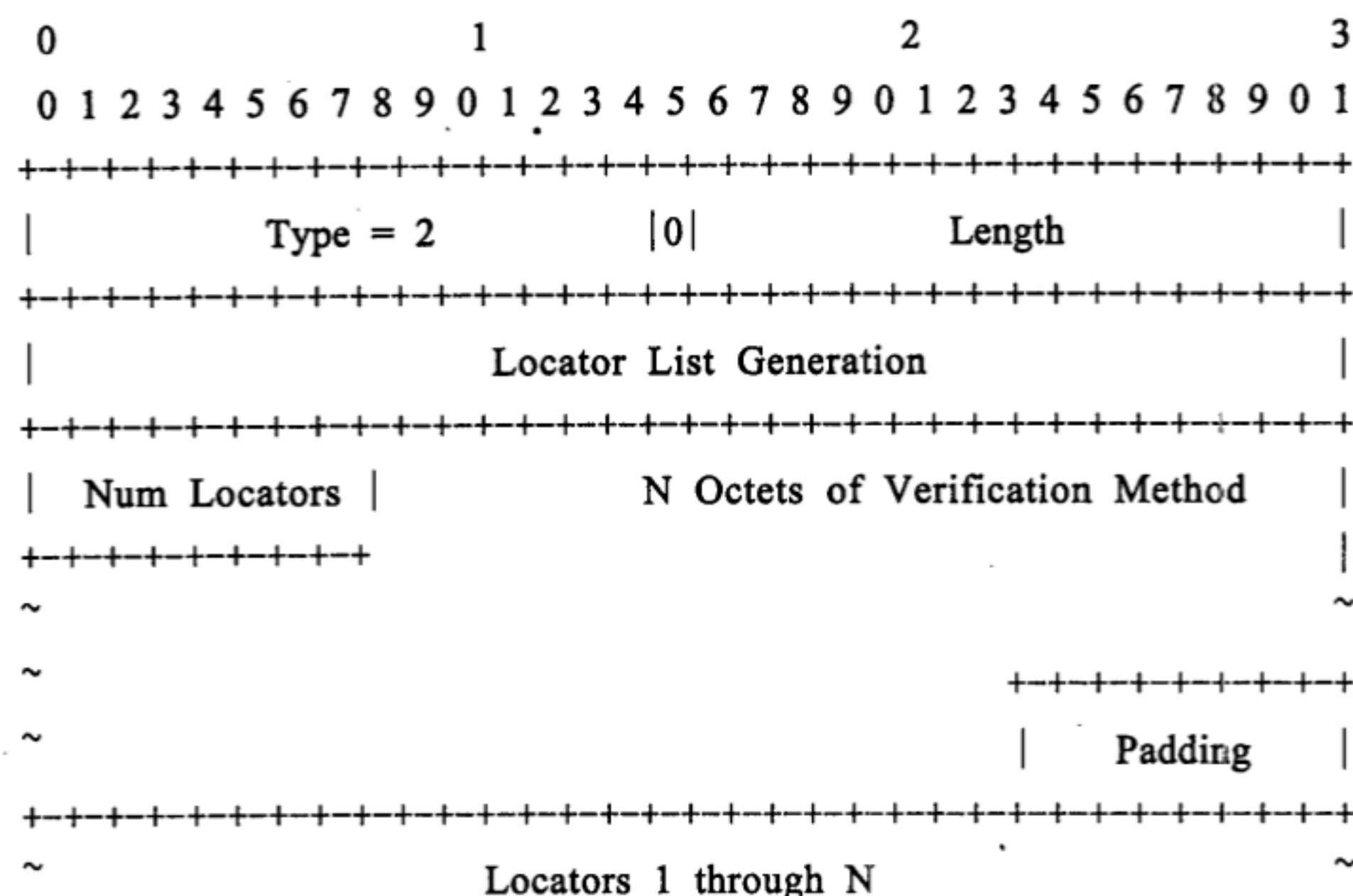
填充 (Padding) : 0byte~7byte, 如果需要则加上, 详见5.16.1。

5.16.3 定位符表选项格式

定位符表选项用于装载发送者的所有定位符。注意定位符的顺序是重要的，因为定位符的优先级选项是通过使用表中的索引来指定定位符。

注意我们装载选项中的所有定位符，即使其中的一些能由CGA参数数据结构来自动生成。

定位符表选项格式如下:



+++++

字段:

定位符表生成 (Locator List Generation) :

32位无符号整数。表示一个生成数, 对每一个新的定位符表其值增1。这是用于确认定位符优先级中的索引时指向定位符表的正确版本。

定位符数 (Num Locators) :

8位无符号整数。在该选项中包含的定位符数。我们在下面称该值为“N”。

验证方法 (Verification Method) :

Nbyte。第i个字节指定了第i个定位符的验证方法。

填充 (padding) : 0byte~7byte, 如果需要则添加, 这样定位符从8byte整数倍边界开始。注意对这个选项, 没有必要在后面填充, 因为定位符长度是8byte的整数倍。内部的填充包含在长度字段中。

定位符 (Locators) : N 128-bit定位符。

表4给出了验证方法的定义。

表4 验证方法

值	方法
0	保留
1	HBA
2	CGA
3~200	待分配
201~254	实验用
255	保留

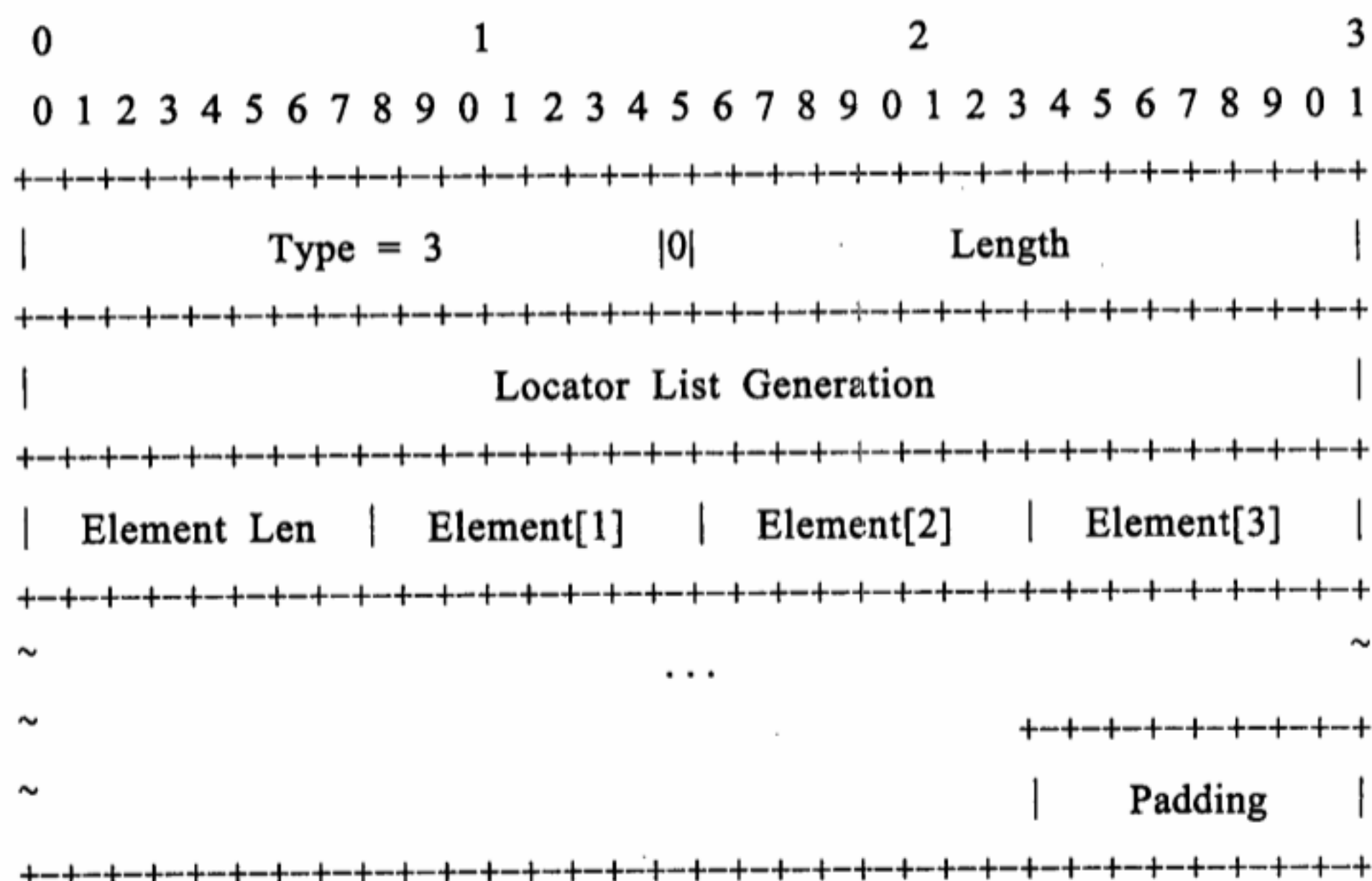
5.16.4 定位符优先级选项格式

定位符优先级选项能用一些标记来指明是否知道一个定位符能工作。另外, 发送方可以包含优先级的概念。把“优先级 (preferences)”定义成优先级和权重 (priority and weight) 的结合是有意义的, DNS SRV记录也有类似的信息。优先级 (priority) 可以提供一个排列定位符的方法, 且在一个给定的优先级 (priority) 内, 权重可以提供一个负载共享的方法。SRV如何定义优先级 (priority) 和权重的交互参见 [4]。

我们需要的优先级的最小概念是能够指明一个定位符“死”了。我们能够对每个定位符使用一个字节的标记来处理。

我们可以对每个定位符装一个更大的“元素”来扩展。本标准目前也定义了2byte和3byte元素, 且如果需要我们可以通过应用更大的元素来增加更多的信息。

定位符并没有包含在定位符优先级列表中。相反, 指示定位符的第一个元素在定位符选项里的第一个元素中。在这个选项和定位符表选项中的生成数用于验证它们指明的是相同的定位符表版本。



描述了元素长 = 1 的情况。

字段:

定位符表生成 (Locator List Generation) :

32位无符号整数。表示一个生成数,用于确定应用于定位符表哪个元素。

元素长 (Element Len)：8位无符号数。每个元素长以字节为单位。该指定定义了当长度为1、2或3的情况。

元素[i] (Element[i]) : 有一定数量的由元素长度字段定义的字节字段。为在使用的定位符表选项的第i个定位符提供优先符。

填充: 0byte~7byte, 若需要则添加, 见5.16.1。

若元素长度是1，则元素仅由一字节标记字段组成。当前对标记集的定义：

BROKEN: 0x01

TRANSIENT: 0x02

BROKEN标记的目的是通知对端一个已给的定位符不再工作。TRANSIENT的目的是当shim6由移动IP组成，且当我们可能有更多稳定本地定位符和不太稳定的定位符护理时，允许区分更稳定地址和不太稳定地址。

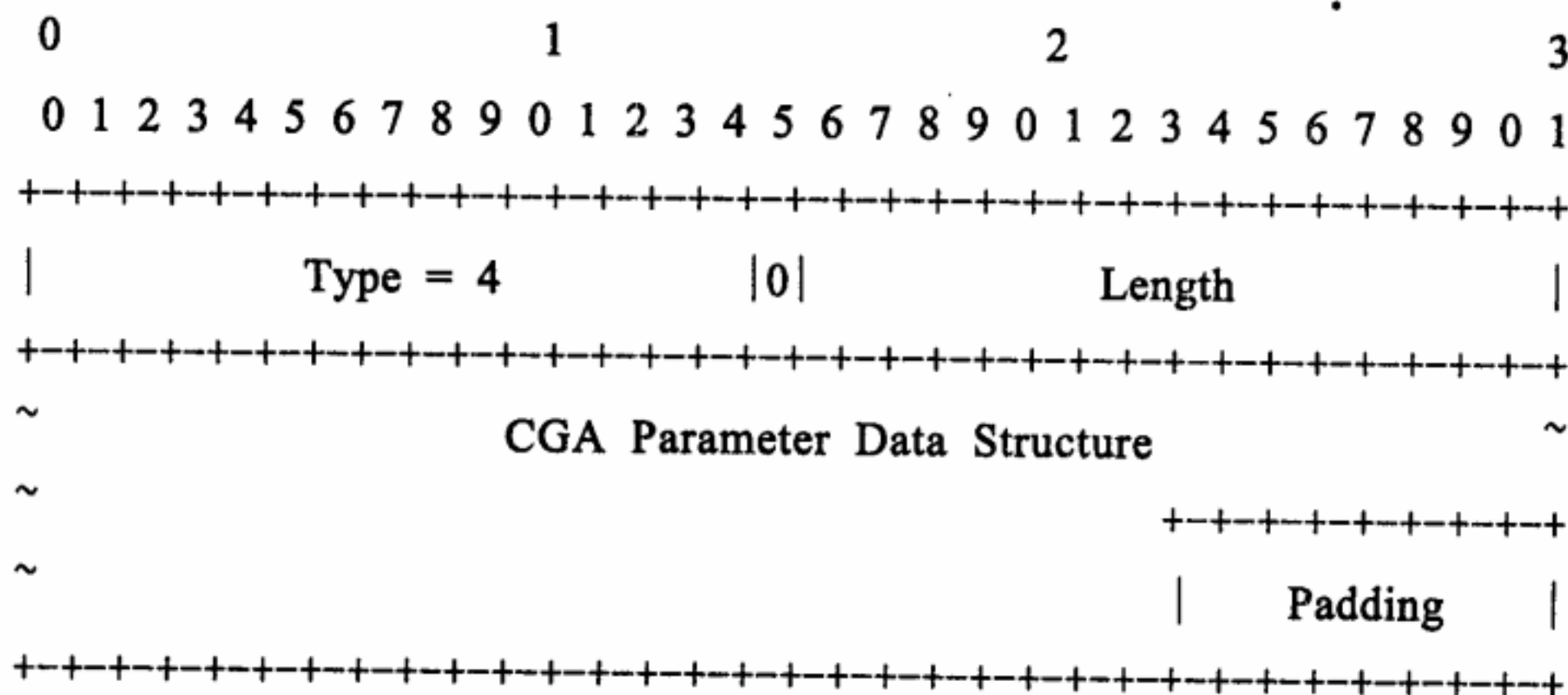
若元素长度是2，则元素由一字节标记字段，后面跟着一字节优先级字段组成。优先级字段的语义和DNS SRV记录中的优先级字段相同。

若元素长度是3，则元素的组成是一字节标记字段，后跟一字节优先级字段，后跟一字节权重字段。权重字段和在DNS SRV记录中的权重字段有相同语义。

本标准并没有定义元素长度超过3时的格式，除开必须定义此格式，这样前3byte和上面的情况是一样的，即一字节标记字段，后跟一字节优先级字段，后跟一字节权重字段。

5.16.5 CGA 参数数据结构选项格式

该选项包含了CGA参数数据结构（PDS）。当HBA是用于验证定位符时，PDS包含了PDS强制字段，以及其他PDS可能有的与shim6无关的扩展，此外还包含HBA多前缀扩展。当CGA是用于验证定位符时，除了PDS选项，主机也需要包含CGA签名选项格式的签名。CGA参数数据格式如下：



字段:

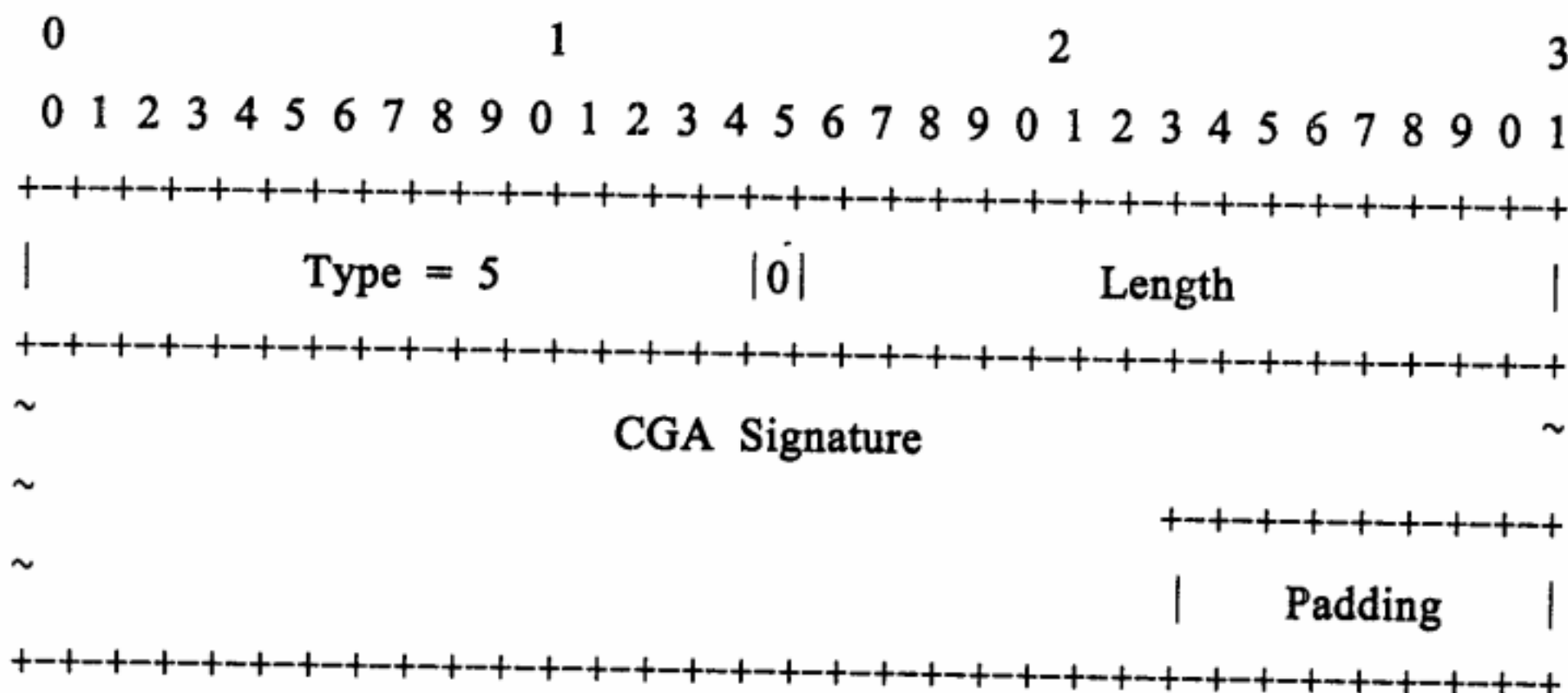
CGA参数数据结构 (CGA Parameter Data Structure) :

可变长内容，在IETF RFC5535和IETF RFC3972规定。

填充: 0byte~7byte, 若需要则添加, 详见5.16.1。

5.16.6 CGA 签名选项格式

当CGA用于验证在定位符表选项中的一个或多个定位符时，有关信息将需要包含这个选项。



字段:

CGA签名 (CGA Signature):

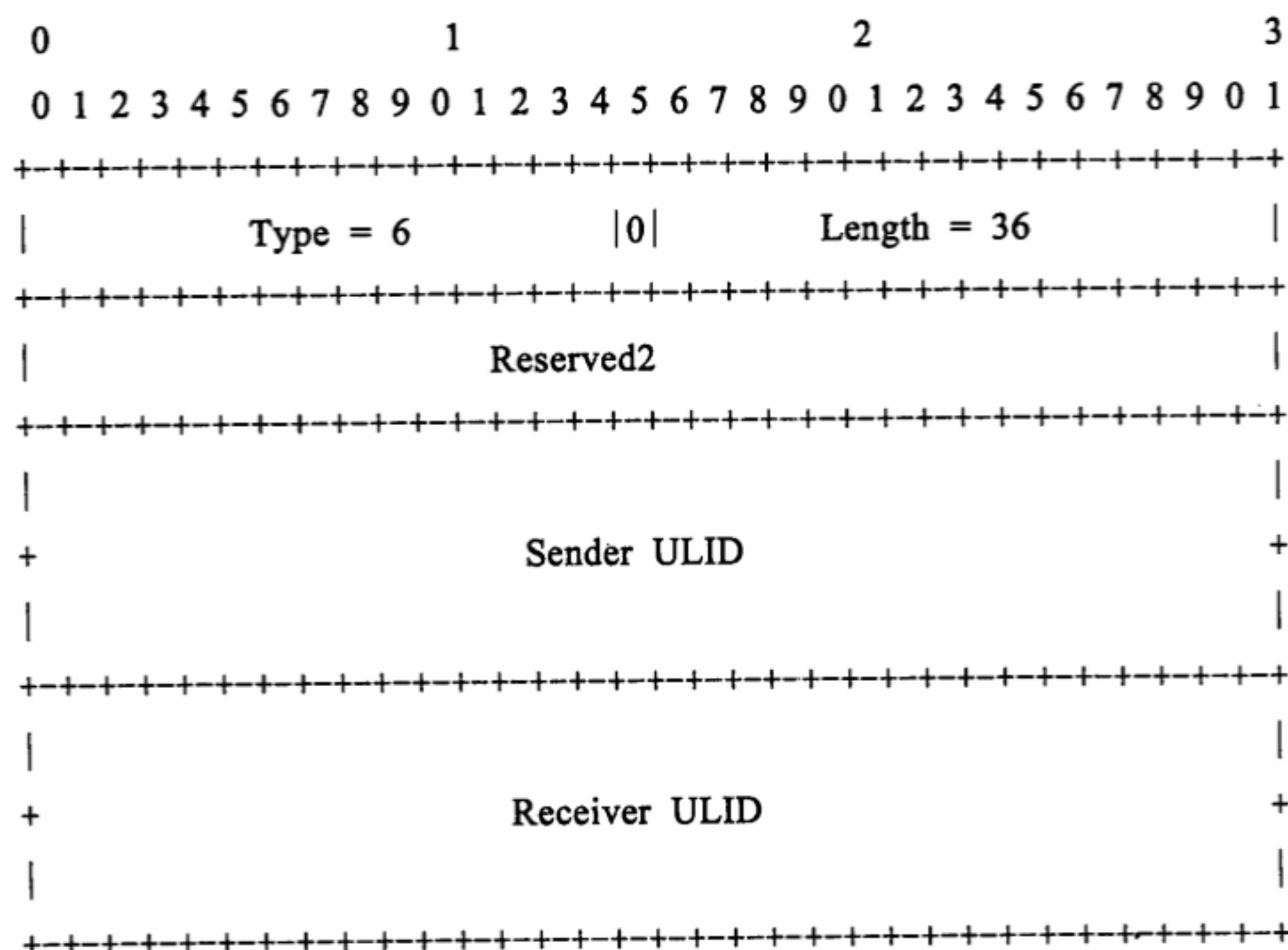
一个可变长的字段，包含了一个PKCS#1 v1.5签名，其构成是由发送者在以下字节序上的私钥：

- a) shim6的128位CGA消息类型标签值: 0x4A 30 5662 4858 574B 3655 416F 506A 6D48 (标签值已由IETF RFC5533随机生成);
- b) 对应定位符表的定位符表生成数;
- c) 在相应的定位符表选项中的定位符子集, 定位符表的规范方法由CGA设置。定位符的排序必须和他们在定位符表选项中的相同。

填充: 0byte~7byte, 若需要则添加, 见5.16.1。

5.16.7 ULID 选项格式

I1, I2和I2bis消息必须包含ULID对；一般地，这是在IPv6的源和目的字段。若上下文的ULID和在用来装载I1/I2/I2bis 消息的IPv6分组中源、目的地址域中的地址不同，I1/I2/I2bis消息必须要包含ULID对选项。ULID对选项格式如下：



字段:

保留2 (Reserved2) :

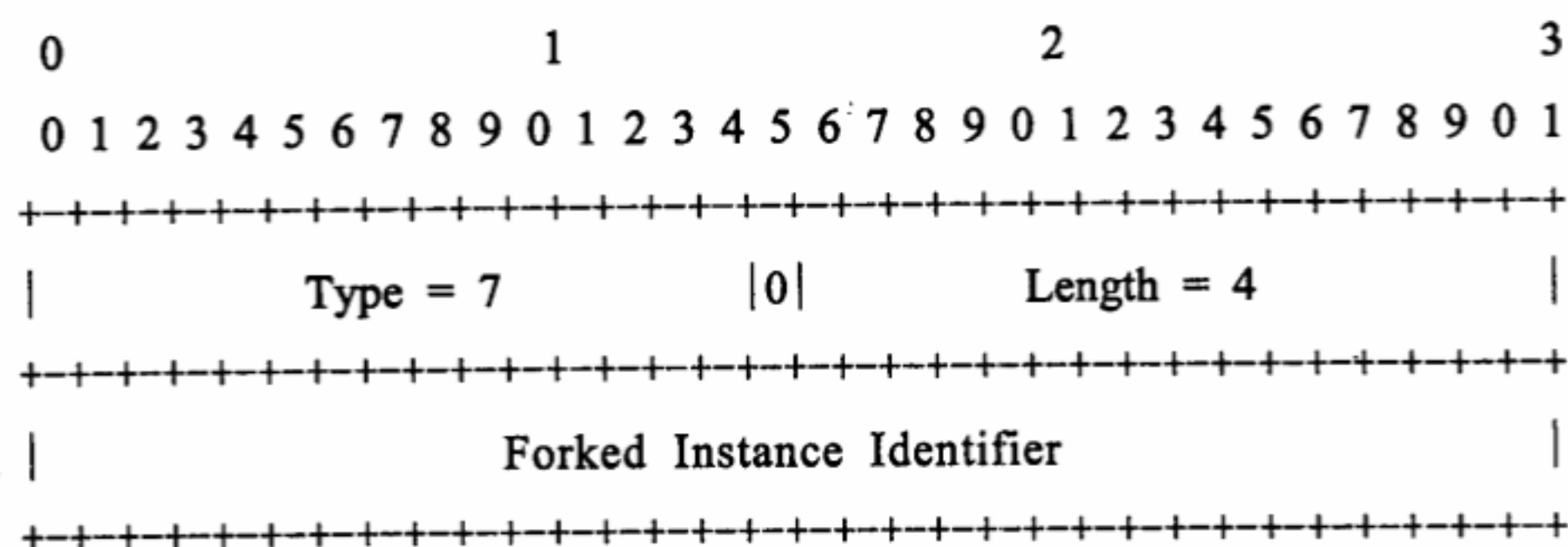
16位字段。以备后用。传输时值为0。在接收时一定要忽略。（需要使ULID在8byte的整数倍边界开始。）

发送者ULID (Sender ULID): 一个128位IPv6地址。

接收者ULID (Receiver ULID): 一个128位IPv6地址。

5.16.8 分叉实例标识符选项格式

分叉实例标识符选项格式如下:

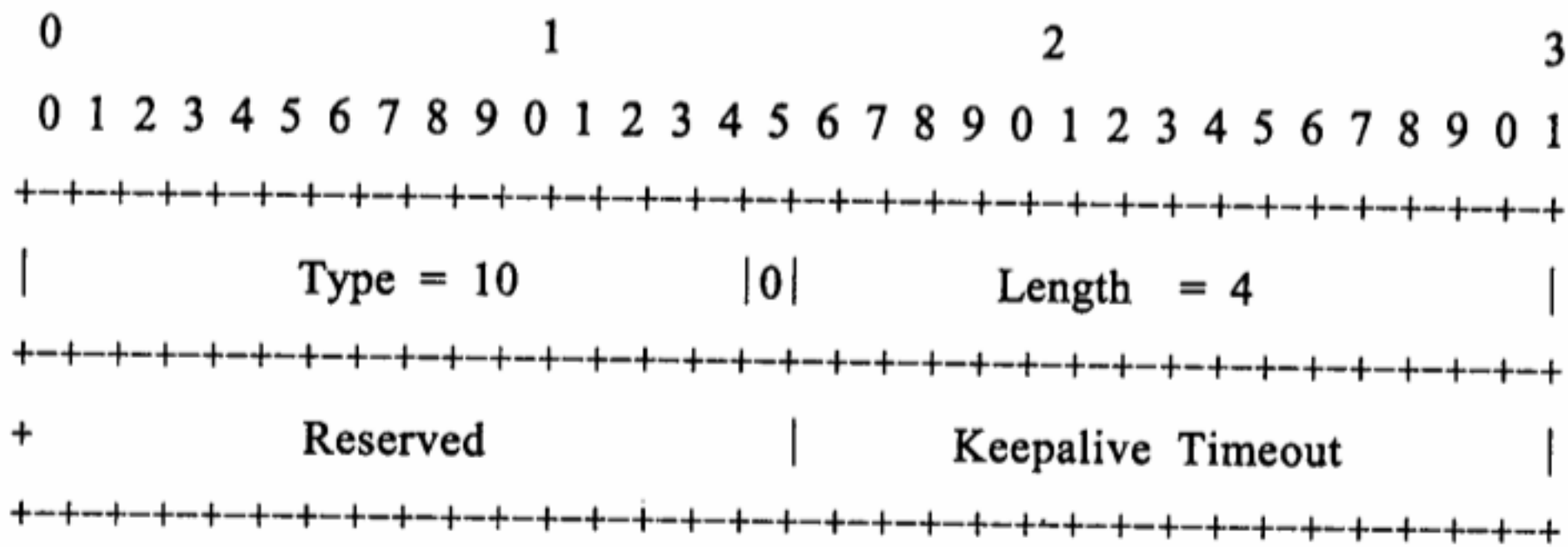


字段:

分叉实例标识符 (Forked Instance Identifier)： 包含了特殊的分叉实例的32位字段。

5.16.9 keepalive 超时选项格式

shim6上下文的双方都不能通知对端它想让对端使用来作为Keepalive Timeout值的值。如果该节点使用了一个非默认的Send Timeout值，它必须在下面的选项中告知对端这个值是Keepalive Timeout值。这个选项可能会在I2, I2bis, R2,或UPDATE报文中被发送。这个选项应当在给出的shim6联系中只发送一次。如果一个节点收到了这个选项，它应当为该对端更新它的Keepalive Timeout值。keepalive超时选项格式如下：



字段：

- 类型（Type）：这一字段标识了该选项且必须被设为10（Keepalive Timeout）。
- 长度（Length）：这个字段必须被设为4。
- 保留（Reserved）：这是一个16bit的字段，留作后用。在传输时设为0，在接收端必须被忽略。
- 保活超时（Keepalive Timeout）：
以秒为单位，为对端建议Keepalive Timeout值。

6 一台主机的概念模型

6.1 概述

本章描述了主机实现shim6所需数据结构的一种可能组织模型。但本标准并没有限定内部实现必须采用该数据模型，只要该实现的外部表现与协议规定一致即可。

6.2 概念数据结构

shim6协议的关键概念数据结构是ULID对上下文。该数据结构包含如下信息：

- 上下文的状态，见 6.3；
- 对端 ULID：ULID(peer)；
- 本地 ULID：ULID(local)；
- 分叉实例标识符：FII，对默认上下文（即没有分叉）来说其值为 0；
- 有优先序的对端定位符表：Ls(peer)；
- 最近收到的且确认的对端定位符表生成数；
- 对每个对端定位符，其使用的验证方法（从定位表选项中）；
- 对每个对端定位符，一个标记用于规定是否用 HBA 或 CGA 验证，以及一个比特位用于规定是否已经探测了定位符以验证在那个位置 ULID 是现成的；
- 当前对端定位符是当发送分组时用于当目的地址的定位符：Lp(peer)；
- 本地定位符集和优先序：Ls(local)；
- 大多数最近发送定位符表选项的生成数；

- 当前本地定位符是当发送分组时用作源地址的定位符: Lp(local);
- 用于传送控制消息和 shim6 净荷扩展头的上下文标签; 由对端分配: CT(peer);
- 在收到的控制消息和 shim6 净荷扩展头中所期望的上下文; 由本地主机分配: CT(local);
- 在上下文建立和更新消息的过程中的消息重传计时器;
- 根据一个执行如何来决定一个上下文是否还在被使用, 这或许需要跟踪最后一次使用该上下文来收发分组的情况;
- 为定位符对提供的可达性状态, 详见第 14 章;
- 在探索过程中, 在第 14 章中定义的已经发送或和接收的探测消息的信息;
- 在上下文建立阶段, 发起者随机数, 应答者随机数, 应答者验证符, 和涉及到不同分组发送(I1,I2,R2)的计时器, 详细定义见第 7 章。

6.3 上下文的状态机状态

用来描述shim6协议的状态机状态见表5。

表5 用于描述 shim6 的状态机状态

STATE	Explanation
IDLE	状态机开始
I1-SENT	初始化上下文建立交互
I2-SENT	等待完成上下文建立交互
I2BIS-SENT	潜在上下文丢失检测
ESTABLISHED	SHIM上下文已建立
E-FAILED	上下文建立交互失败
NO-SUPPORT	ICMP没有识别下一头类型 (type 4, code 1) 已接收, 说明shim6不支持

此外, 在每个上述的状态机状态中存储的状态信息见表6。

表6 状态机状态中存储的信息

状态	信息
IDLE	None
I1-SENT	ULID(peer), ULID(local), [FII], CT(local), INIT Nonce, Lp(local), Lp(peer), Ls(local)
I2-SENT	ULID(peer), ULID(local), [FII], CT(local),

表6（续）

状态	信息
	INIT Nonce, RESP Nonce, Lp(local), Lp(peer), Ls(local), Responder Validator
ESTABLISHED	ULID(peer), ULID(local), [FII], CT(local), CT(peer), Lp(local), Lp(peer), Ls(local), Ls(peer), INIT Nonce?(to receive late R2)
I2BIS-SENT	ULID(peer), ULID(local), [FII], CT(local), CT(peer), Lp(local), Lp(peer), Ls(local), Ls(peer), CT(R1bis), RESP Nonce, INIT Nonce, Responder Validator
E-FAILED	ULID(peer), ULID(local)
NO-SUPPORT	ULID(peer), ULID(local)

7 建立 ULID 对的上下文

7.1 概述

ULID对上下文的建立是用过四次握手来实现的，这可以避免应答方在收到第一个包后就建立状态。作为交互的一部分，每个终端分配一个上下文标签并且和对端共享这个上下文标签和它的定位符集。

在某些情况下，不一定要四次握手，例如，当双方在同一时刻尝试建立上下文，或是从一个被垃圾回收的上下文中恢复，或在某一方主机中丢失。

7.2 上下文标签的惟一性

作为建立一个新的上下文的一部分，每台主机必须分配一个惟一的上下文标签。由于shim6净荷扩展头的解复用仅依靠上下文标签的值（而不是用定位符），上下文标签必须对每个上下文是惟一的。路径分离攻击者很难猜到上下文标签这一点是重要的。因此，如果一个执行使用上下文标签中的结构来优化查找，上下文标签最少有30位必须是无结构的且是由随机或伪随机位组成。

此外，为了尽量减少上下文标签重用，主机应该通过非结构化标记的名字空间来随机循环，该空间是留作随机分配上下文标签值的（例如，遵从[12]中描述的方法）。

7.3 定位符验证

和第10章处理定位符更新一样，在上下文建立过程中，对端定位符可能需要被验证。

定位符验证有两个独立的方面。一方面，要验证定位符是和ULID绑定的，即“拥有”一个ULID的主机同时也是声明了该定位符“拥有权”的主机。shim6协议使用HBA或CGA技术来进行这一验证。另一方面，要验证主机在所声明的定位符下确实是可达的。这一验证不仅是确保通信能够进行下去，同时也是为了阻挡第三方的洪范攻击（参见[14]）。不同方面的定位符验证在不同的时间发生，因为第一个

验证是在拥有不确定的源定位符的对端接收分组之前需要执行，而后一个验证仅是在分组发送给定位符之前需要。

在一台主机能够使用（和ULID不同的）定位符作为源定位符之前，它必须知道对端会接受用那个源定位符作为上下文一部分的分组。因此在主机需要通过发送更新确认消息或R2消息来确认该新定位符之前，先应当执行HBA/CGA验证。

在一台主机能够使用（和ULID不同的）定位符作为目的定位符之前，若没有在收到定位符集后执行HBA/CGA验证，则必须先执行。此外，它必须验证该ULID确实存在于那个定位符。这个验证是通过一个返回可路由测试来作为探测子协议的一部分来进行的。

如果主机不支持在定位符表选项中的验证方法，或该验证方法和CGA参数数据结构不一致（例如，参数数据结构不包含多前缀扩展，验证方法使用HBA），则主机必须忽略定位符表和它所包含的消息。主机应当生成shim6出错消息，出错码为2，且指针指向未发现了一致验证方法的字节。

7.4 普通上下文的建立

普通的上下文建立由四次消息交互组成，按照I1，R1，I2，R2的顺序，如图3所示。

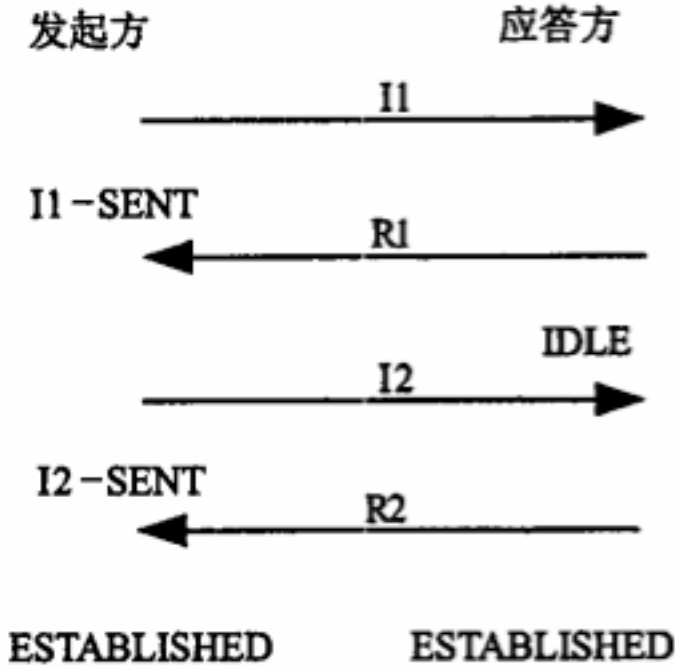


图3 普通的上下文建立

7.5 同时的上下文建立

当双方终端尝试对同一ULID对发起一个上下文，则我们可能以I1消息的交叉结束。或者，由于在收到I1后没有建立状态，一台主机可能在发送了一个R1消息后发送一个I1。

由于一台主机记得它已经发送了一个I1，所以它能在收到对端（为了相同的ULID对）发来的一个I1后回应一个R2，导致了如图4所示的消息交互。这种方式的交互是需要的，因为例如当R2消息丢失致使I1消息重传，这是一种正确的响应。

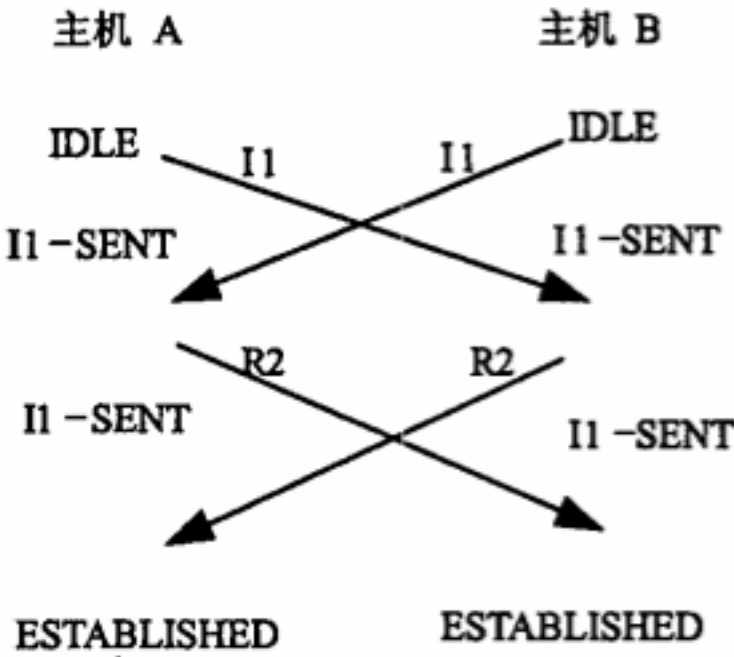


图4 交叉的 I1 消息

如果一台主机已收到I1且发送了R1，它没有状态去记住这一点。因此，如果该主机的ULP向下发分组，这可能引发该主机自己再发送一个I1。因此，当一端发送一个I1，另一端发送I2，如图5所示。

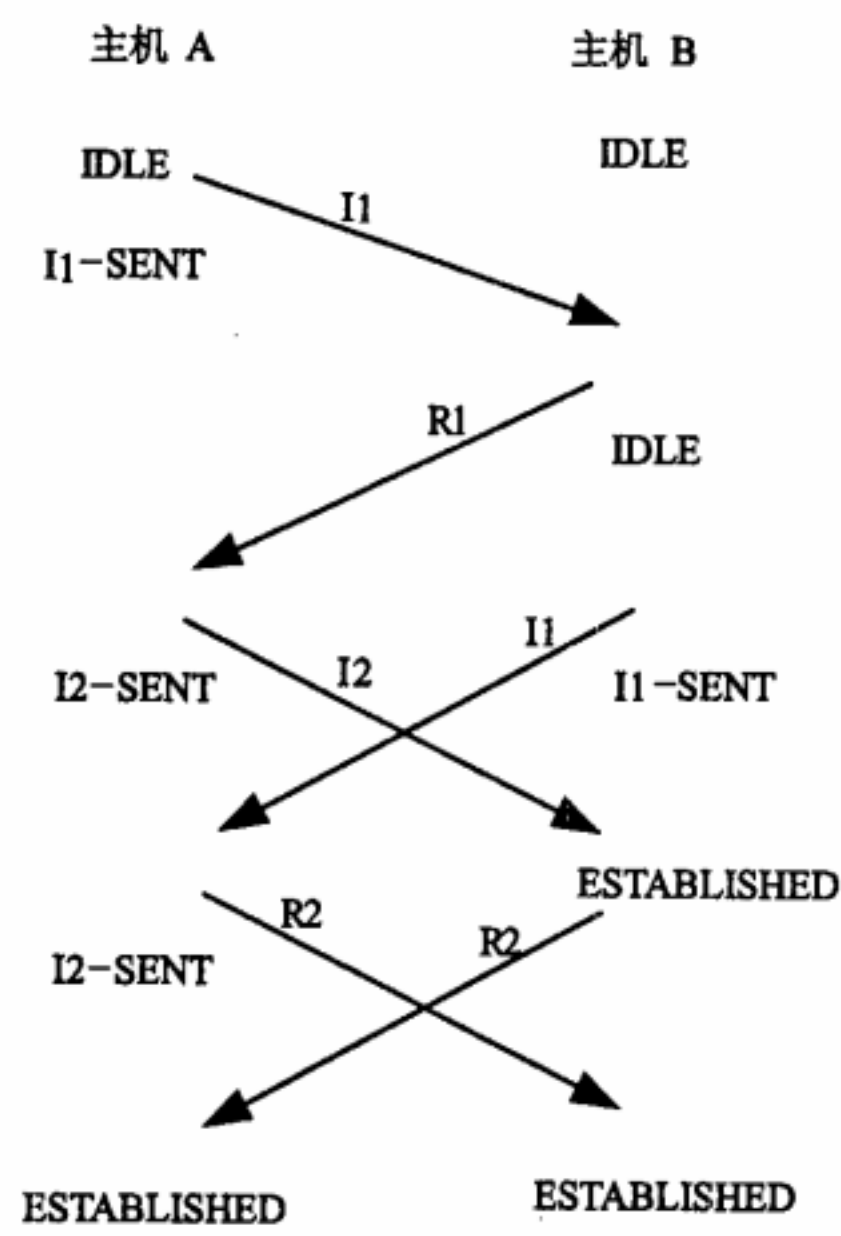


图5 交叉的 I2 和 I1

7.6 上下文恢复

由于垃圾回收，我们可以以一端拥有和使用该上下文状态，而另一端没有任何状态结束。在丢失了上下文状态的一端，我们在能使用它之前需要先恢复状态。

这一需求的产生可以是以下的情况：

——通信使用 ULID 对作为定位符对来工作，但出现了问题，且保留了上下文状态的主机决定去探测替代定位符对；

——通信的运行使用的定位符对不是 ULID 对；因此，保留了上下文状态的一端发送的 ULP 分组使用 shim6 净荷扩展头；

——保留了上下文状态的主机发送一控制消息（例如一个更新请求消息）。

在所有的情况下，结果都是没有上下文的一端收到了一个shim消息，且该端没有此上下文标签的上下文。

我们可以通过让没有上下文状态的节点发送一个R1bis消息来恢复，然后使用I2bis和R2消息来完成该回复，如图6所示。

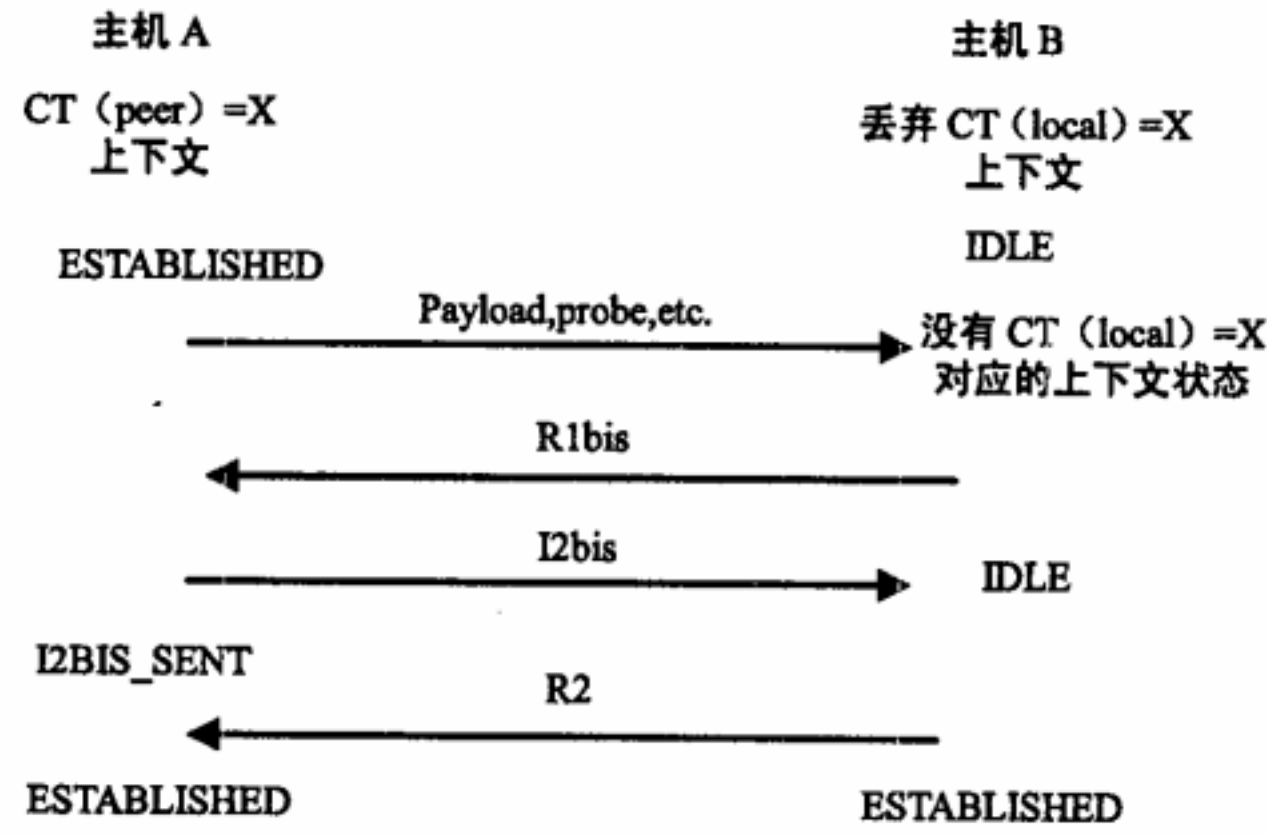


图6 上下文在接收端丢失

如果一端已进行垃圾回收或丢失了上下文状态，它可能通过发送一个I1去试着创建一个新的上下文状态（对相同的ULID对）。在这种情况下，对端（仍然有该上下文状态）将会以一个R1消息回复，且完整的4次握手将会重复一次，如图7所示。

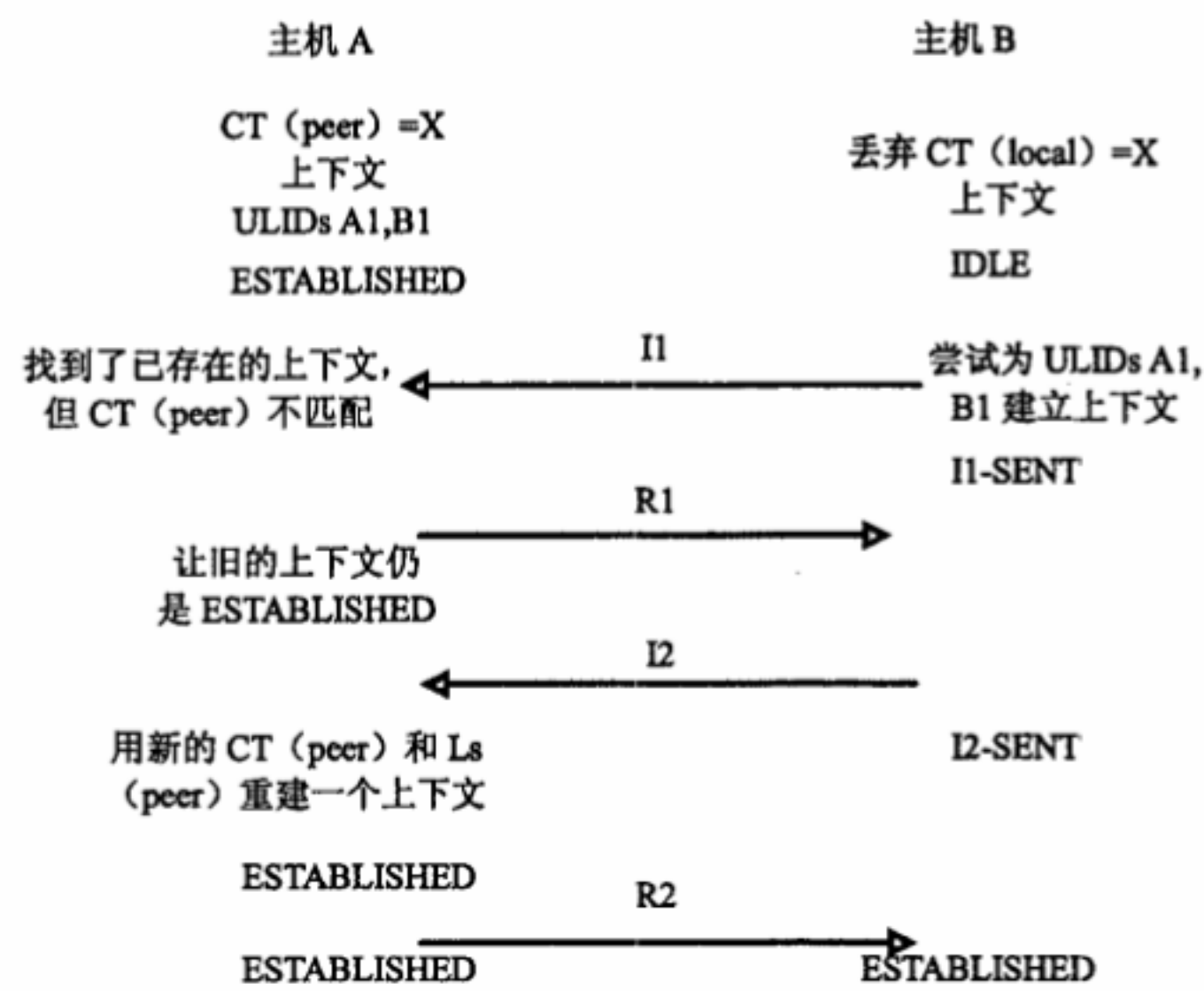


图7 上下文在发送端丢失

7.7 上下文的混淆

由于每一端都可能对上下文状态进行垃圾回收，我们会出现一种情况，一端仍保留该上下文状态并试着去使用，而另一端已经丢失了该状态。我们在7.6节的恢复中已经讨论了这种情况。但由于相同的原因，当一台主机为ULID对<A1, B1>保留上下文标签X来作为CT (peer)，对端可能在相同主机间为了另一ULID对（如<A3,B1>）而结束分配那个上下文标签来作为CT (local)。在这种情况下，我们不能使用恢复机制，因为这需要为两个ULID对使用独立的上下文标签。

这种类型的“混淆”可看成两种情况（假设A保留状态而B丢到了）：

- B 决定为 ULID 对<A3,B1>创建上下文，分配 X 作为它的上下文标签，且向 A 发送了 I1。
- A 决定为 ULID 对<A3,B1>创建上下文，且通过向 B 发送 I1 开始交互。当 B 收到 I2 消息时，它分配 X 作为这个上下文的上下文标签。

在两种情况下，A能够发现问题：B给ULID对<A3,B1>分配了X作为上下文标签，尽管A仍然为ULID对<A1,B1>留着X作为CT(peer)。因此，A会知道B一定已丢失了<A1,B1>的上下文。

当收到I2/I2bis/R2时会发现该混淆，因为我们需要那些消息一定要在定位符表选项中包含足够大的定位符集，这样对端能够通过对比在Ls(peer)中是否有一些相同的定位符来决定是否要两个上下文有相同的主机来作为对端。

使用该上下文标签的旧的上下文一定要被删除；它不能再用于发送分组。因此，A需要强制删除<A1,B1,X>的上下文状态，这样它能就收<A3,B1,X>的新的上下文。一个执行有可能重建一个上下文来替代删除的那个 – 本例中就是<A1, B1>。正常的I1, R1, I2, R2建立交互会为那个上下文选择一个惟一的上下文标签。这一重建时可选的，但当ULP使用那些上下文被删除的ULID进行通信时，这还是有帮助的。

I1消息有一个复制的上下文标签不应当导致旧上下文状态的删除；这个操作要延迟到直到收到I2消息。

7.8 发送 I1 消息

当shim层决定为一ULID对建立一个上下文，它通过为它的终端分配和初始化上下文状态来开始。作为这一过程的一部分，它为该上下文分配了一个上下文标签，该标签不被任何的其他上下文用作CT(local)。在使用一个新的API以及ULP请求一个分叉上下文的情况下，FII值会被设为非0。否则FII值为0。接下来发起者可以发送一个I1消息，并把上下文的状态机设为I1-SENT。I1消息必须包含ULID对，一般情况下，是在IPv6的源和目的字段。但当该上下文的ULID对和该I1消息的定位符对不相同，则I1消息必须要包含ULID选项。此外，若一个FII值是非0的，I1消息必须包含一个上下文实例标识符选项，其中会包括相应的值。

7.9 重传 I1 消息

如果经过了I1_TIMEOUT的时间后，主机没有收到作为对I1消息的回复的R1或R2消息，则它需要重传I1消息。该重传应当使用一个具有二进制指数退避算法的计时器，以避免由于太多主机进行I1重发而造成网络拥塞。此外，实际超时的值应在0.5和1.5随机选择，以避免自同步。

如果经过了I1_RETRIES_MAX时长的重传而没有答复，则最可能的是对端没有shim6协议（或对端有防火墙阻止了该协议）。在这种情况下，该主机记住不要为那个ULID建立一个上下文是有意义的。但任何此类的消极缓存应该最多保留NO_R1_HOLDDOWN_TIME的时长，这是为了在shim试着建立该上下文而该主机完全不可达的情况下，稍后的上下文建立。

如果主机收到了一个ICMP错误“无法识别的下一头”类型（类型4，编码1）且包含的包是它刚刚发送的I1消息，则我们更可以确认对端ULID确实没有使用shim6。同样，在这样的情况下，主机应当记住不要尝试再去为那个ULID建立上下文。这一否定缓存应保存最多ICMP_HOLDDOWN_TIME的时长，比前一种情况会长一些。

7.10 接收 I1 消息

若I1消息不符合在12.4节定义的以及所有下面列出的正确性检查，则主机必须丢弃该I1消息：

——Hdr Ext Len 字段最少为 1，即长度最少为 16byte。

根据收到的I1消息，主机抽取消息中的ULID对以及FII。如果没有ULID对选项，则ULID对从IPv6头的源和目的字段获得。如果消息中没有FII选项，FII值为0。

接下来，主机查询与该ULID对和FII匹配的上下文。

如果没有找到（即该状态机状态是IDLE），则主机以下一节指明的R1消息回复。

若这个上下文存在，且状态机状态是ESTABLISHED，主机检查发起者的定位符是否包含在Ls(peer)中。（若I1消息中没有ULID对选项，则没有必要执行这一检查）

若这个上下文状态存在于ESTABLISHED STATE，且该定位符不在定位符集中，则主机以下一节指明的R1消息回复。这完成了I1的处理，上下文的状态机状态不改变。

若这个上下文状态存在于ESTABLISHED STATE，且该定位符确实在定位符集中，则主机使用I1消息中包含的CT为该上下文比较CT(peer)：

——若上下文标签匹配，则这可能意味着 R2 消息丢失了，这个 I1 消息是重传的。在这种情况下，主机回复一个 R2 消息，该 R2 消息中包含了该上下文可用的消息；

——若该上下文标签没有匹配，则可能意味着发起者已经丢失了对该上下文的信息，且尝试为相同的 ULID 对建立一新的上下文。在这种情况下，主机以下一节指明的 R1 消息回复。这完成了 I1 的处理，

上下文的状态机状态不改变。

若该状态存在于其他的状态机状态中(I1-SENT, I2-SENT, I2BIS-SENT), 我们处在同时上下文建立的情况下, 详见7.5节。在这种情况下, 主机保持CT(peer)不变, 且回复一个R2消息。这完成了I1的处理, 上下文的状态机状态不改变。

7.11 发送 R1 消息

7.11.1 概述

当主机需要发送R1消息来作为对I1消息的回应时, 它拷贝I1消息中的发起者随机数到R1消息里, 生成一个应答者随机数, 并且以下面小节介绍的来计算应答者验证符。在这种情况下主机没有创建状态。

注: 用于生成R1回复消息的信息既包含在I1消息中, 又是一个全局信息, 即不与特定请求的上下文相关(S和应答者随机数值)。

当主机需要发送R2消息来作为对I1消息的回应时, 它拷贝I1消息中的发起者随机数到R2消息里, 否则遵循形成一个R2消息的正常规则(见7.15)。

7.11.2 生成 R1 验证符

如5.16.2小节所述, 验证符生成机制是一个本地的选择, 因为对验证符的生成和验证都是由同一节点来完成, 即应答者。当为了提供所需要的保护, 验证符需要通过填写5.16.2中规定的条件来生成。应答者正确产生验证符的一个方法就是为应答方随机数维护一个单一密码(S), 以及运行计数器(C), 该应答方随机数是在固定的周期递增的(这使应答方能独立于它所使用的上下文来验证应答方随机数的存在时间)。

当验证符生成了, 且包含在将要发送的作为对I1消息应答的R1消息中, 应答者能够进行如下的操作来生成验证符值:

第一, 应答者使用当前计数器C的值来作为应答者随机数。

第二, 它使用以下信息(多行索引)作为单项函数的输入:

- 密码 S;
- 应答者随机数;
- 来自 I1 消息的发起者上下文标签;
- 来自 I1 消息的 ULID;
- 来自 I1 消息的定位符(仅仅在他们与 ULID 不同时需要);
- 若 FII 选项也包含在 I1 消息中, 则也需要。

第三, 它使用哈希函数的输出来作为包含在R1消息中的定位符值。

7.12 接收 R1 消息和发送 I2 消息

若R1消息不符合在12.4节定义的以及所有下面列出的正确性检查, 则主机必须丢弃该R1消息:

- Hdr Ext Len 字段最少为 1, 即长度最少为 16byte。

根据收到的R1消息, 主机从该消息中抽取发起者随机数和定位符对(后者来自IPv6头的源和目的字段)。接下来主机查询匹配发起者随机数的上下文, 以及定位符各自包含在Ls(peer) 和 Ls(local)的地方。若没有找到合适的上下文, 则R1消息被丢弃。

若找到符合的上下文, 则主机查看状态机状态:

- 如果状态机状态是 I1-SENT, 则它发送下面指明的 I2 消息;

——如果是其他的状态机状态(I2-SENT, I2BIS-SENT, ESTABLISHED), 则主机已经发送了 I2 消息, 这个 R1 可能是对重传的 I1 的回复, 所以该 R1 必须被丢弃。

当主机发送I2消息时, 它包括在R1消息中的应答者验证符选项。I2消息必须包含ULID对 – 一般是在IPv6的源和目的字段。如果I1消息中包含了一个ULID对选项, 则I2消息中也必须包含该ULID对选项。此外, 如果该上下文的FII值为非0, I2消息必须包含一个FII选项来装载FII值。另外, I2消息包含一个发起者随机数。该数不需要和前面的I1消息中包含的发起者随机数一样。

I2消息可能也要包含发起者的定位符表。如果是这种情况, 则它必须也要包含CGA参数数据结构。如果是用CGA (而不是HBA) 来验证在该定位符表中的一个或更多的定位符, 则发起者必须也要包含一个CGA签名选项来包含签名。

当I2消息已发送, 状态机状态设为I2-SENT。

7.13 重传 I2 消息

如果发起者在发送了I2消息后, 经I2_TIMEOUT的时间还没有收到R2消息, 它可能使用二进制指数退避和随机的计时器来重传I2消息。应答者验证符可能有一个有限的时长 – 即对端可能拒绝比VALIDATOR_MIN_LIFETIME存在的久的验证符选项以避免重放攻击。在发起者决定不重传I2的情况下, 或发起者在重传I2消息过了I2_RETRIES_MAX的时间仍没有收到R2的情况下, 发起者应当退回到重传I1消息。

7.14 接收 I2 消息

若I2消息不符合在12.4定义的以及所有下面列出的正确性检查, 则主机必须丢弃该I2消息:

——Hdr Ext Len 字段最少为 2, 即长度最少为 24byte。

根据收到的I2消息, 主机从消息中抽取ULID对和FII。如果没有ULID对选项, 则从IPv6头的源和目的字段获得ULID对。如果没有FII选项, 则FII值为0。

接下来, 主机验证应答者随机数是一个最近的 (该数不比VALIDATOR_MIN_LIFETIME大则应当认作是最近的) 且应答者验证符选项与主机已经为ULID、定位符、应答方随机数、发起者随机数和FII而计算的验证符匹配。

如果该消息包含了一个CGA参数数据结构 (PDS), 则主机必须验证在该消息中包含的确切的PDS是否与ULID(peer)相关。

若上面的任一项验证失败了, 则主机直接丢弃该消息; 且它已完成了对I2的处理。

如果上面所有的验证都成功了, 则主机开始为发起者查找一个上下文状态。主机使用获得的ULID对和FII来查找上下文。如果不存在, 则该 (不存在的) 上下文的状态机状态视为IDLE; 因此, 行动取决于该状态机状态, 如下:

——若状态机状态时 IDLE (即上下文不存在), 主机分配一个上下文标签(CT(local)), 为该上下文创建上下文状态, 且把它的状态机状态置为 ESTABLISHED。它在上下文中记录 CT(peer)和对端的定位符集以及它自己的定位符集。它应当在这个时间点对对端的定位符集进行 HBA/CGA 验证, 见 7.3。然后主机按下面指明的返回一个 R2 消息。

——如果状态机状态是 I1-SENT, 则主机验证源定位符是否在 Ls(peer)中或在 I2 消息的定位符表中, 且为此特定的定位符的 HBA/CGA 验证是否成功。

- 如果不是这种情况, 则消息直接丢弃, 且上下文状态机状态不变。

- 如果是这种情况,则主机使用 I2 消息中的数据来更新上下文信息(CT(peer), Ls(peer)),且主机必须按下面指定的返回一个 R2 消息。注意在更新 Ls(peer)信息之前,主机应当在这个时间点对对端的定位符集进行 HBA/CGA 验证,见 7.3。主机状态机状态置为 ESTABLISHED。

——如果状态机状态为 ESTABLISHED, I2-SENT, 或 I2BIS-SENT,则主机验证源定位符是否包含在 Ls(peer)或在 I2 消息包含的定位符表中,且对这个指定的定位符的 HBA/CGA 验证是否成功。

- 如果不是这种情况,则消息直接丢弃,且上下文状态机状态不变。
- 如果是这种情况,则主机使用 I2 消息中的数据来更新上下文信息(CT(peer), Ls(peer)),且主机必须按 7.15 指定的返回一个 R2 消息。注意在更新 Ls(peer)信息之前,主机应当在这个时间点对对端的定位符集进行 HBA/CGA 验证,见 7.3。主机状态机状态保持不变。

7.15 发送 R2 消息

在主机发送R2消息之前,它必须查找一个可能的上下文混淆,即对同一个shim6对端出现使用一个CT(peer)标识多个上下文的现象,见7.16。

当主机需要发送R2消息时,主机形成该消息以及它的上下文标签,且从触发消息(I2, I2bis, or I1)中拷贝发起者随机数。此外,它可能包含替代的定位符和必需的选项,这样对端才能验证它们。特别地,R2消息可能包含应答方的定位符表和PDS选项。如果CGA(而不是HBA)用作验证定位符表,则应答方也要标记该消息的关键部分以及包含一个CGA签名选项用于包含签名。

R2消息从不会重传。如果R2消息丢失了,则发起方会重传I2/I2bis或是I1消息。任一个重传都会导致应答方发现该上下文状态且以一个R2消息来应答。

7.16 对上下文混淆的匹配

当主机收到I2, I2bis, 或R2,它必须查找可能的上下文混淆,即对同一个shim6对端出现使用一个CT(peer)标识多个上下文的现象。当主机收到上面提到的消息时有可能会出现这种情况,因为它们使用一个新的CT(peer)来创建一个新的上下文。当对一个存在的上下文更新CT(peer)也可能出现这个问题。

主机采用CT(peer)作为新创建或更新的上下文,且寻找符合以下条件的其他上下文:

——状态机状态是 ESTABLISHED 或 I2BIS-SENT。

——有相同的 CT(peer)。

——有一个 Ls(peer),它最少有一个定位符是和新创建或更新的上下文相同的。

如果找到了这样的上下文,则主机检测它的 ULID 对或 FII 是否和新创建或更新的上下文的不同:

——如果有一个或两个都不同,则对端是为新创建或更新的上下文重新使用了上下文标签,而 ULID 对或 FII 不同,这说明对端已经丢失了原有的上下文。在这种情况下,我们处于了上下文混淆的情况;且主机一定不能用旧的上下文来发送任何分组。它可能刚丢弃旧的上下文(比较对端已经丢弃了),或它可能试着通过发送一个新的 I1 消息以及把状态机状态改为 I1-SENT 来重建旧的上下文。不管怎样,一旦发现了这种情形,主机一定不能在 ESTABLISHED 状态机状态存有两个有着重叠的 Ls(peer)和相同的上下文标签的上下文,因为这可能导致对端出现解复用问题;

——如果两个都相同,则这个上下文实际上就是新创建或更新的上下文;因此没有混淆。

7.17 接收 R2 消息

若R2消息不符合在12.4定义的以及所有下面列出的正确性检查,则主机必须丢弃该R2消息:

——Hdr Ext Len 字段最少为 1,即长度最少为 16byte。

根据收到的R2消息，主机从该消息中抽取发起者随机数和定位符对（后者来自IPv6头的源和目的字段）。接下来，主机查询匹配发起者随机数的上下文，以及以及定位符各自包含在Lp(peer) 和 Lp(local) 的地方。基于状态机状态：

——如果没有找到这样的上下文，即状态机状态是 IDLE，则直接丢弃该消息。

——如果状态机状态是 I1-SENT, I2-SENT, 或 I2BIS-SENT 则主机进行下面的步骤。如果该消息中包含了 CGA 参数数据结构 (PDS)，则主机必须验证在该消息中包含的确切的 PDS 是否与 ULID(peer) 相关，见 7.3。如果验证失败，则该消息直接丢弃。如果验证成功，则主机在该上下文状态中记录 R2 消息的信息；它记录对端的定位符集和 CT(peer)。主机应当这个时间点对对端的定位符集进行 HBA/CGA 验证，见 7.3。主机的状态机状态设为 ESTABLISHED。

——如果状态机状态是 ESTABLISHED，则 R2 消息直接忽略（因为这就像是对一个重传的 I2 消息的回复）。

在主机完成对R2的处理之前，它必须查找可能的上下文混淆，即对同一个shim6对端出现使用一个CT(peer)标识多个上下文的现象，见7.16。

7.18 发送 R1bis 消息

7.18.1 概述

根据收到的shim6净荷扩展头，若接收者没有当前shim6上下文，接收端回复一个R1bis消息以能够快速重建丢失的shim6上下文。

同样，主机根据收到的消息类型在64-127之间的控制消息也会回复一个R1bis（即排除上下文建立消息，比如I1, R1, R1bis, I2, I2bis, R2,以及未来的扩展），控制消息指的是上下文不存在。

假设所有触发了生成R1bis消息的分组包含一个定位符对（在IPv6头的地址字段）和一个上下文标签。

根据收到的上面描述的任意分组，主机会回复一个包含以下信息的R1bis：

- 应答方随机数是一个由应答方选择的数，发起者会在 I2bis 消息中返回；
- 分组上下文标签是出发了 R1bis 消息生成的所收到的分组中包含的上下文标签；
- 还包含了应答方验证符选项，其中有下一小节介绍的方法计算的验证符。

7.18.2 生成 R1bis 验证符

应答者正确产生验证符的一个方法就是为应答方随机数维护一个单一密码(S)，以及运行计数器(C)，该应答方随机数是在固定的周期递增的（这使得应答方能独立于它所使用的上下文来验证应答方随机数的存在时间）。

当验证符生成了，且包含在Rbis1消息中 – 即作为对一个特定的控制分组或包含了shim6净荷扩展头消息的分组的应答 -- 应答者能够进行如下的操作来生成验证符值：

第一，应答者使用当前计数器C的值来作为应答者随机数；

第二，它使用以下信息（多行索引）作为单项函数的输入：

- 密码 S；
- 应答者随机数；
- 包含在接收的分组中的接受者上下文标签；
- 来自接收的分组的定位符；

第三，它使用哈希函数的输出来作为定位符串。

7.19 接收 R1bis 消息和发送 I2bis 消息

若R1bis消息不符合在12.4定义的以及所有下面列出的正确性检查，则主机必须丢弃该R1bis消息：

——Hdr Ext Len 字段最少为 1，即长度最少为 16byte。

根据收到的R1bis消息，主机从该消息中抽取发起者随机数和定位符对（后者来自IPv6头的源和目的字段）。接下来，主机查询匹配发起者随机数的上下文，以及定位符各自包含在Lp(peer) 和 Lp(local) 的地方。

——如果没有找到这种上下文，即状态机状态是 IDLE，则 R1bis 消息直接被丢弃。

——若状态机状态是 I1-SENT, I2-SENT, or I2BIS-SENT 则 R1bis 消息直接被丢弃。

——若状态机状态是 ESTABLISHED，则说明对端已经丢失了该上下文，且其目的是试着重建该上下文。为此，主机保持在该上下文状态中的 CT(peer)不变，过渡到 I2BIS-SENT 状态机状态，且发送一个 I2bis 消息，其中包含已计算的应答方验证符选项、分组上下文标签、以及在收到的 R1bis 中的应答方随机数。这个 I2bis 消息的发送使用的是 R1bis 消息总的定位符对。若这个定位符对与为该上下文定义的 ULID 对不同，则 ULID 选项必须要包含在该 I2bis 中。此外，如果该上下文的 FII 值为非 0，I2bis 消息必须包含一个 FII 选项来装载 FII 值。该 I2bis 消息可能也包含了一个定位符列表。如果是这种情况，则它必须也要包含 CGA 参数数据结构。如果是用 CGA（而不是 HBA）来验证在该定位符表中的一个或更多的定位符，则发起者必须也要包含一个 CGA 签名选项来包含签名。

7.20 重传 I2bis 消息

如果发起者在发送了I2bis消息后，经I2bis_TIMEOUT的时间还没有收到R2消息，它可能使用二进制指数退避和随机的计时器来重传I2bis消息。应答者验证符可能有一个有限的时长 – 即对端可能拒绝比 VALIDATOR_MIN_LIFETIME存在的久的验证符选项以避免重放攻击。在发起者决定不重传I2bis的情况下，或发起者在重传I2bis消息过了I2_RETRIES_MAX的时间仍没有收到R2的情况下，发起者应当退回到重传I1消息。

7.21 接收 I2bis 消息和发送 R2 消息

若I2bis消息不符合在12.4定义的以及所有下面列出的正确性检查，则主机必须丢弃该I2bis消息：

——Hdr Ext Len 字段最少为 3，即长度最少为 32byte。

根据收到的I2bis消息，主机从该消息中抽取ULID对和FII。如果没有ULID对选项，则从IPv6头的源和目的地址字段获得ULID对。如果该消息没有FII选项，则FII值被设为0。

接下来，主机验证应答者随机数是一个最近的（该数不比VALIDATOR_MIN_LIFETIME大则应当认作是最近的）且应答者验证符选项与主机已经为ULID、定位符、应答方随机数、发起者随机数和FII计算的验证符匹配。

如果该消息包含了一个CGA参数数据结构（PDS），则主机必须验证在该消息中包含的确切的PDS是否与ULID(peer)相关。

若上面的任一项验证失败了，则主机直接丢弃该消息；且它已完成了对I2bis的处理。

如果上面所有的验证都成功了，则主机开始为发起者查找一个上下文状态。主机使用获得的ULID对和FII来查找上下文。如果不存在，则该（不存在的）上下文的状态机状态视为IDLE；因此，行动取决于该状态机状态，如下：

——若状态机状态时 IDLE（即上下文不存在），主机分配一个上下文标签(CT(local))，为该上下文

创建上下文状态，且把它的状态机状态置为 ESTABLISHED。主机不应当在 I2bis 消息中为 CT(local)使用分组上下文标签；相反，它应当随机选择一个新的上下文标签，就像对 I2 消息的处理一样。它在上下文中记录 CT(peer)和对端的定位符集以及它自己的定位符集。它应当在这个时间点对对端的定位符集进行 HBA/CGA 验证，见 7.3。然后主机按 7.15 指明的返回一个 R2 消息。

——如果状态机状态是 I1-SENT，则主机验证源定位符是否在 Ls(peer)中或在 I2bis 消息的定位符表中，且为此特定的定位符的 HBA/CGA 验证是否成功。

- 如果不是这种情况，则消息直接丢弃，且上下文状态机状态不变。
- 如果是这种情况，则主机使用 I2 消息中的数据来更新上下文信息(CT(peer), Ls(peer))，且主机必须按下面指定的返回一个 R2 消息。注意在更新 Ls(peer)信息之前，主机应当在这个时间点对对端的定位符集进行 HBA/CGA 验证，见 7.3。主机状态机状态置为 ESTABLISHED。

——如果状态机状态为 ESTABLISHED, I2-SENT, 或 I2BIS-SENT,则主机验证下面的两个条件是否最少满足一个：i) 若源定位符是否包含在 Ls(peer)或，ii)在 I2bis 消息包含的定位符表中，且对这个指定的定位符的 HBA/CGA 验证是否成功。

- 如果不是这种情况，则消息直接丢弃，且上下文状态机状态不变。
- 如果至少满足上述两条件中的一个，则主机使用 I2bis 消息中的数据来更新上下文信息(CT(peer), Ls(peer))，且主机必须按 7.15 指定的返回一个 R2 消息。注意在更新 Ls(peer)信息之前，主机应当在这个时间点对对端的定位符集进行 HBA/CGA 验证，见 7.3。主机状态机状态保持不变。

8 处理 ICMP 出错消息

在路径中以及目的端的路由器可能产生ICMP出错消息。在一些情况下， shim6能够采取行动并解决导致错误的问题。在其他情况下， shim6不能解决问题，且有一点很重要：这些分组使它备份到ULPs，这样它们便能够做合适的处理。

对主机本身而言该机制是完全本地的，在这个意义上，这是一个执行的问题。但如何正确发送ICMP错误给主机的ULP这一问题是重要的；因此，本章说明了这个问题。

所有的ICMP消息必须要在所有情况下都能传给ULP，除了当shim6成功的作用于该消息（例如，选择了一个新路径）。应当有一个配置选项来无条件的传送所有的ICMP消息（包括被shim6作用过的）给ULP。

根据那些建议，以下的ICMP错误消息应答由shim6层来处理而不传给ULP：

——ICMP 错误的不可达目的地，它的编码：

- 0（没有向目的端的路由）；
- 1（与目的端的通信被管理禁止）；
- 2（超出源地址范围）；
- 3（地址不可达）；
- 5（源地址在入口/出口策略失败）；
- 6（拒绝向目的端路由）。

——ICMP 超时错误。

——ICMP 参数问题错误，导致了该错误的参数是一个 shim6 参数。

以下的ICMP错误消息报告不能被shim6层定址的问题，且应当传送给ULP（如下描述）：

- ICMP 分组太大错误；
- ICMP 目的端不可达，代码 4（端口不可达）；
- ICMP 参数问题（如果导致了该问题的参数不是一个 shim6 参数）。

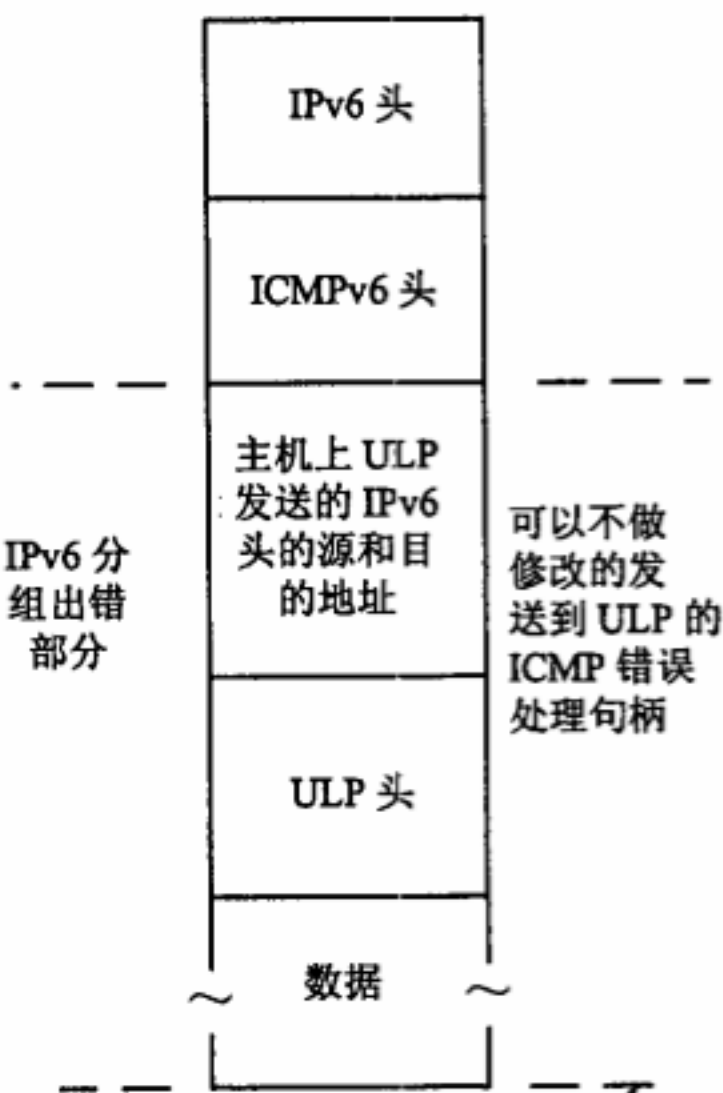


图8 没有 shim6 净荷扩展头的 ICMP 错误处理

当ULP分组没有shim6净荷扩展头发送——即当初始定位符与ULIDs相等且能工作——这说明没有新的关注点；一个执行的用于发送这些错误给ULP的存在的机制会起作用。如图8所示。

但当在传输侧的shim6插入了shim6净荷扩展头，且用一些其他的定位符替代IP地址字段的ULIDs时，则一个返回的ICMP错误将有一个“出错的包”，这个包不是ULP发送的。因此，执行在传输ICMP出错给ULP之前，要对“出错的包”应用反向映射，包括了ICMP扩展（参见[24]）。如图9所示。

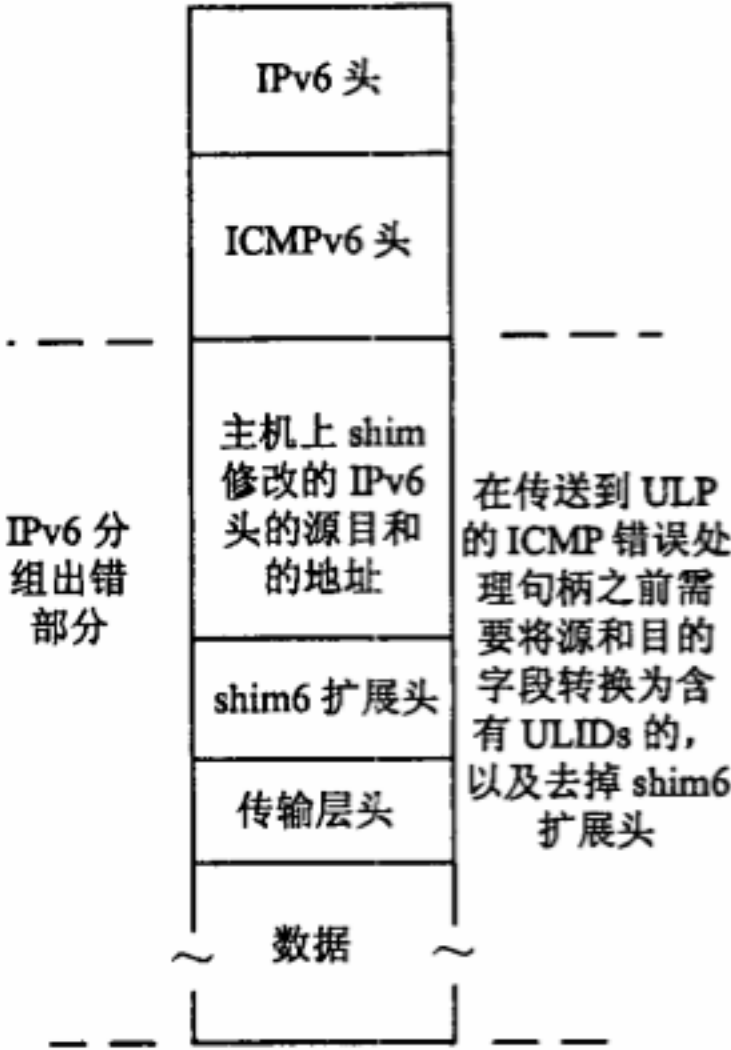


图9 有 shim6 净荷扩展头的 ICMP 错误的处理

注意这个映射和当从对端接收有shim6净荷扩展头的分组是不同的，因为，在那种情况下，该分组包含CT(local)。但ICMP出错有一个包含了净荷扩展头的“出错的分组”，它有CT(peer)。这是因为它们打算被对端接收。不管怎样，因为当被对端接收时<源定位符，目的定位符，CT(peer)>必须是惟一的，本地主机应当也仅能找到一个匹配该元组的上下文。

如果该ICMP错误时“分组过大”，报告的MTU必须被调整为小于8byte，因为当发送分组时shim会增加8byte。

在“出错的分组”已经插入了原始的ULIDs之后，这个shim6净荷扩展头可以被移除。结果就是一个传给ULP的“出错的分组”看起来好像没有存在过shim。

9 ULID 对上下文的拆除

每台主机都能单方面地决定何时拆除ULID-pair上下文。建议当主机知道有一些上层协议有可能使用该上下文时，主机不要拆除该上下文。例如，如果没有一个开放的连在ULID(peer)的套接字，一个执行可能知道这一点。但可能会出现消息不是很便利到达shim层，例如，对不连接它们的套接字的UDP应用或对保留一些高层状态的应用，穿过（TCP）连接和UDP分组。

因此我们建议执行应当通过观察使用该上下文进行收发的流的数目来最小化过早的拆除，并仅当一个上下文看起来不会再用到才查出它。一个合理的方法是当最后一条使用该上下文收发的消息已经过了5min后才考虑拆除。

注：使用ULID对作为定位符对的分组，以及不需要shim6层进行地址重写的分组都认为是使用了相关的shim6上下文的分组。

由于没有明确的、协调的上下文状态拆除，所以存在对上下文标签的潜在的问题。一个终端可能删除了该状态且可能为其他的通信重用该上下文标签，而对端可能过一会儿又试着使用旧的上下文（没有删除的）。本协议有从此恢复的机制，不论该状态的删除是彻底的和偶然的（例如，主机崩溃和重启）或是对看起来不再使用的shim状态进行垃圾回收都能工作。但主机应当试着去应用 2^{47} 个上下文标签值来尝试随机循环，以最小化上下文标签的重用。

10 更新对端

10.1 概述

更新请求和确认都用于更新定位符表（仅当CGA用于验证定位符时可能）以及更新于每个定位符相关的优先序。

10.2 发送更新请求消息

当一台主机的定位符集有一个变动，它能够通过发送一个更新请求通知对方。当一台主机在对其定位符集的优先序上有变动，它也能够通知对方。更新请求消息可以仅包含定位符表选项（来传送新的定位符集），或仅包含优先序选项，或同时包含两者。

若主机发送了一个新的定位符表，主机选择一个新的、随机的、本地的生成数，在上下文中记录这一点，且把它放在定位符表选项中。任何定位符优先序选项，不管是用相同更新请求还是用一些未来的更新请求来发送，都会使用那个生成数来确保该优先序应用在正确版本的定位符表上。

主机为每个更新选择了一个随机的请求随机数且为任何的更新请求重传保持相同的随机数。该随机数用于匹配该请求的确认。

更新请求消息也能包含一个CGA参数数据结构（如果CGA PDS先前没有改变，这一点是需要的）。如果CGA（且不是HBA）是用于验证一个或多个包含在该定位符表中的定位符，则更新请求消息也一定要包含一个CGA签名选项以包含该签名。

10.3 重传更新请求消息

如果发起者在发送了I2消息后, 经I2_TIMEOUT的时间还没有收到R2消息, 它可能使用二进制指数退避和随机的计时器来重传I2消息。应答者验证符可能有一个有限的时长 – 即对端可能拒绝比VALIDATOR_MIN_LIFETIME存在的久的验证符选项以避免重放攻击。在发起者决定不重传I2的情况下, 或发起者在重传I2消息过了I2_RETRIES_MAX的时间仍没有收到R2的情况下, 发起者应当退回到重传I1消息。

如果主机在经过了UPDATE_TIMEOUT的时间还没有收到作为对更新请求消息的响应的更新确认R2消息, 则它需要重传更新请求消息。该重传应当使用一个二进制指数退避重传计时器来避免由于过多的主机进行更新请求重传导致对网络产生了拥塞。同样, 确切的超时值应当在0.5到1.5标称值之间随机选择以避免自同步。

10.4 当重传时出现更加新的信息

在任何时候最多只能有一个未完成的更新请求消息。因此直到一个, 比如说, 新定位符列表的更新被确认了, 任何更加新的定位符表或定位符优先序才能发送。但当有一个更加新的信息而旧的信息还没有得到确认时, 主机不是去等待一个确认, 而是抛弃先前的更新而创建一个新的更新请求(使用一个新的请求随机数), 该请求和没有被确认的信息一样去包含新的信息。

例如, 如果原始定位符表仅是(A1, A2), 如果一个包含定位符表(A1, A3)的更新请求还未完成, 且主机决定应当也把A4加入定位符表而把A1标为BROKEN, 则需要:

- 为该新更新请求选择一个新的随机的请求随机数;
- 为该新的定位符表选择一个新的随机生成数;
- 形成新的定位符表: (A1, A3, A4);
- 形成一个定位符优先序选项, 使用新的生成数且为第一个定位符使用 BROKEN 标记;
- 发送该更新请求并启动一个重传计时器。

任何没有匹配当前请求随机数的更新确认(例如一个对已丢弃的更新请求的确认)会被直接忽略。

10.5 接收更新请求消息

若更新请求消息不符合在12.4定义的以及所有下面列出的正确性检查, 则主机必须丢弃该消息:

- Hdr Ext Len 字段最少为 1, 即长度最少为 16byte。

根据收到的更新请求消息, 主机从消息中抽取上下文标签。它然后查找匹配该上下文标签的上下文。如果没有找到, 它发送一个R1bis消息, 见7.18。

由于上下文标签可以重用, 主机必须验证IPv6源地址字段是Ls(peer)的一部分且IPv6目的地址字段是Ls(local)的一部分。如果不是这种情况, 更新请求的发起方有一个旧的上下文, 正好能匹配这个上下文的CT(local)。在这种情况下, 主机必须发送一个R1bis消息, 否则忽略掉该更新请求消息。

如果该消息包含一个CGA参数数据结构(PDS), 则主机必须验证包含在分组中的确切的PDS是否与ULID(peer)相关。如果这一验证失败, 该消息直接丢弃。

接下来, 根据上下文的状态机状态:

- 若是 ESTABLISHED, 继续处理消息;
- 若是 I1-SENT, 丢弃该消息, 且保持在 I1-SENT;
- 若是 I2-SENT, 发送 I2 且继续处理该消息;
- 若是 I2BIS-SENT, 发送 I2bis 且继续处理该消息。

为装载在更新请求消息中的定位符进行验证的问题在7.3有介绍。如果不能验证定位符表，这个程序应当发哦那个一个shim6出错消息，出错码为2。不管怎样，如果它不能被验证，则对更新请求没有后续处理。

一旦任何包含在更新请求中的定位符表选项已被验证，在上下文中的对端生成数更新为在定位符表选项中的那个。

如果更新请求消息包含了一个定位符优先序选项，在优先序选项中的生成数与在上下文中的对端生成数做比较。如果它们不匹配，则主机生产一个shim6出错消息，出错码为3，且指针字段指向在定位符优先序选项中的定位符表生成数的第一个字节。此外，如果在定位符优先序中的元素数目与在Ls(peer)中的定位符数不匹配，，则发送一个shim6出错消息，出错码为4，指针字段指向在定位符优先序选项中的长度字段的第一个字节。在两种失败情况下，对更新请求消息没有后续处理。

如果生成数匹配了，定位符优先序被记录在该上下文中。

一旦该定位符表选项（若存在）已经被验证，且所有新定位符表或定位符优先序已经被记录，主机发送一个更新确认消息，从请求中拷贝随机数且使用CT(peer)作为接收者上下文标签。

任何新的定位符（或更可能是新定位符优先序）可能导致主机想要为该上下文选择一个不同的定位符对 – 例如，如果定位符优先序选项列出了当前的Lp(peer)为BROKEN。主机使用在第14章中描述的可达性探测程序来验证在改变Lp(peer)前新的定位符是可达的。

10.6 接收更新请求确认消息

若更新确认消息不符合在12.4定义的以及所有下面列出的正确性检查，则主机必须丢弃该消息：

——Hdr Ext Len 字段最少为 1，即长度最少为 16byte。

根据收到的更新确认消息，主机从消息中抽取上下文标签和请求随机数。它然后查找匹配该上下文标签的上下文。如果没有找到，它发送一个R1bis消息，见7.18。

由于上下文标签可以重用，主机必须验证IPv6源地址字段是Ls(peer)的一部分且IPv6目的地址字段是Ls(local)的一部分。如果不是这种情况，更新请求的发起方有一个陈旧的上下文，正好能匹配这个上下文的CT(local)。在这种情况下，主机必须发送一个R1bis消息，否则忽略掉该更新请求消息。

接下来，根据上下文的状态机状态：

- 若是 ESTABLISHED，继续处理消息；
- 若是 I1-SENT，丢弃该消息，且保持在 I1-SENT；
- 若是 I2-SENT，发送 I2 且继续处理该消息；
- 若是 I2BIS-SENT，发送 I2bis 且继续处理该消息。

如果该请求随机数与最后一个为该上下文发送的更新请求的随机数不匹配，则该更新确认直接被忽略。如果该随机数匹配，则该更新已经完成且重置更新重传计时器。

11 发送 ULP 净荷

11.1 概述

若在发送端的ULID对没有上下文状态，则对ULP分组如何发送没有影响。如果主机使用一些启发式方法决定什么时候来进行一个延迟的上下文建立，则主机可能需要做一些计数（计算收和发的分组的数）即使在没有ULID对上下文之前。

如果上下文的状态机状态不是ESTABLISHED 或 I2BIS-SENT, 则对ULP分组如何发送也没有影响。仅仅当状态机状态是ESTABLISHED和I2BIS-SENT时, 主机才有CT(peer)和Ls(peer)集。

如果一个ULID对有一个ULID对上下文, 则发送者需要验证是否上下文使用ULID来作为定位符——即, 是否 $Lp(peer) = ULID(peer)$ 以及 $Lp(local) = ULID(local)$ 。

如果是这种情况, 则shim可以不修改的发送分组。如果不是, 则需要使用在11.2中的采取的方式。

还有一些与(不)可达性检测有关的维护, 分组发送是否使用源定位符。这些的细节超出了本标准讨论范围, 在第14章中有详细描述。

11.2 在一次交换后发送 ULP 净荷

当发送分组时, 如果ULID对有一个ULID对上下文, 且如果ULID对不再用作定位符对, 则发送方需要变换分组。除了要用定位符对来替代IPv6的源和目的字段外, 还要加一个8byte的头来使得接收方能够找到上下文以及对变换的分组进行恢复。

如果一个失败导致了一个交换, 且随后上下文交换回到使用ULID对和定位符对来发送, 则再没有必要让发送方去做任何的分组转换; 因此, 没有必要包含一个8byte扩展头。

替代IP地址字段, IPv6源地址字段设为 $Lp(local)$ 且目的地址字段设为 $Lp(peer)$ 。注意, 这一定不能导致任何的对ULP校验和的重计算, 因为ULP校验和是端到端装载的, 且ULP伪头部包含维护端到端特性的ULID。

发送方忽略任何的ULP可能已经包含的“路由子层扩展头”; 因此, 它忽略任何的逐跳扩展头、路由头、路由头之前的目的选项头。在这些头的后面将加上shim6的扩展头。这可能是在一个分片头、一个目的选项头、一个ESP或AH头、或一个ULP头之前。

插入的shim6净荷扩展头包含了对端的上下文标签。这需要前面的扩展头的下一报头值, 因为此扩展报头会有shim6的下一报头值。

12 接收分组

12.1 概述

通信的接收端能够接收与shim6上下文关联的分组, 其可能包含也可能不包含shim6扩展头。在ULID对被用作定位符对的情况下, 收到的分组不会有shim6扩展头, 且shim6子层会按下面描述的来处理。如果收到的分组没有装载shim6扩展头, 按照正常的IPv6接收侧分组处理, 接收端按序处理(扩展)头部。如果发现了一个shim6扩展头, 它将在那个头部看“P”字段。如果该位是0, 则分组必须被传给shim6净荷处理以重写。否则该分组被传给shim6控制处理。

12.2 接收没有扩展头的净荷

接收方抽取IPv6源和目的地址, 且使用这些来找一个ULID对上下文, 这样IPv6地址字段匹配 $ULID(local)$ 和 $ULID(peer)$ 。如果找到了这样的上下文, 该上下文似乎不是静态的, 这点应当被记住以避免把该上下文拆除了, 以及第14章中描述的可达性探测目的。主机继续正常的IP分组处理。

12.3 接收 shim6 净荷扩展头

接收端从shim6扩展头中抽取上下文标签, 且使用它来查找ULID对上下文。如果没有找到上下文, 接收端应当生成一个R1bis消息(详见7.18)

接下来, 根据上下文的状态机状态:

- 如果是 ESTABLISHED，继续处理消息；
- 如果是 I1-SENT，丢弃该消息且状态仍为 I1-SENT；
- 如果是 I2-SENT，发送 I2 且继续处理该消息；
- 如果是 I2BIS-SENT，发送 I2bis 且继续处理该消息。

根据所在处理的上下文，接收端现在能使用上下文中的ULID来替代IP地址字段。最后，shim6净荷扩展头从分组中删除（这样ULP不会被它混淆），且在前一报头中的下一报头值被设为该净荷的实际的协议号。接下来分组可以传送给下一报头值所定义的协议（这可能是一些与IP端系统层相关的函数或一个ULP）。

如果主机使用了一些启发式的方法来决定何时进行一个延迟的上下文建立，则主机可能需要为那些没有shim6扩展头的分组以及没有上下文状态的进行一些计数（计算收发分组数目）。但这一需求取决于该执行选择了什么样的启发式算法。

12.4 接收 shim 控制消息

一个shim控制消息有验证的校验和字段。Shim头长度字段也需要和IPv6分组长度进行验证以确保该shim消息没有申明结束穿过IPv6分组的末尾。最后，它验证IPv6的源和目的地址都不是组播地址或一个未定义地址。如果任一项验证失败，则该分组直接丢弃。

然后该消息根据shim消息类型进行发送。每一消息类型根据文档中的定义进行处理。如果该分组包含一个接收方不知道的消息类型，则生成和返回一个shim6出错消息，出错码为0。指针字段设为指向该shim消息类型的第一个字节。

所有的控制消息能够包含任意的C=0的选项。如果在该消息中有C=1的选项主机是不知道的，则主机一定要发送一个shim6出错消息，出错码为1，指针字段指向选项类型的第一个字节。

12.5 上下文查找

我们假设每个shim上下文有它自己的上下文状态机。我们假设调度者传送进入的分组到它所属的状态机。这里，我们描述调度者使用的规则，以使得能够传送分组到正确的shim上下文状态机。

每个定义的上下文（即由ULID对和FII（默认为0）定义或由CT(local)定义）都有一个状态机。但相信的查找规则更加复杂，特别是在上下文建立过程。

显然，如果所需的上下文没有建立，它的状态机状态是IDLE。

在上下文建立过程，上下文如下定义：

- I1 分组：交付给与 ULID 对和 FII 相关的上下文。
- I2 分组：交付给与 ULID 对和 FII 相关的上下文。
- R1 分组：交付给包含定位符对和发起者随机数的分组（R1 不包含 ULID 对或 CT(local)）。如果没有这种含有定位符对和发起者随机数的上下文，则直接丢弃。
- R2 分组：交付给包含定位符对和发起者随机数的分组（R2 不包含 ULID 对或 CT(local)）。如果没有这种含有定位符对和发起者随机数的上下文，则直接丢弃。
- R1bis 分组：交付给有定位符对以及其 CT(peer)与在 R1bis 分组中上下文标签相等的上下文。
- I2bis 分组：交付给与 ULID 对和 FII 相关的上下文。
- shim6 净荷扩展头：交付给其 CT(local)与在该分组中的接收者上下文标签相等的上下文。
- 其他控制消息(Update, Keepalive, Probe)：交付给其 CT(peer)与在该分组中的接收者上下文标签

相等的上下文。验证 IPv6 源地址字段是 Ls(peer)的一部分, 以及 IPv6 目的地址字段是 Ls(local)的一部分。如果不是, 发送一个 R1bis 消息。

——shim6 出错消息和 ICMP 错误, 该错误包含了 shim6 扩展头或其他“出错分组”中的 shim 控制分组: 使用如下的“出错分组”来调度: 交付给其 CT(peer)与接收者上下文标签相等的上下文, 其中 Lp(local)是 IPv6 的源地址且 Lp(peer)是 IPv6 的目的地址。

此外, 当一个ULP分组从ULP传下来, 在发送端的shim需要能够找到上下文状态。在那种情况下, 查找键就是ULID对以及FII=0。如果我们有一个允许ULP来做上下文分叉的ULP API, 则很可能ULP会把FII传下来。

13 初始联系

初始联系是指在两个ULID之间的非shim的通信。此时shim操作没有开始。

不管shim最终是否用到(例如, 对端可能不支持shim6), 即使是有一个或多个IP地址失败, 可以建立初始联系这一点是非常可取的。

采用应用和传输层协议来重试不同的源和目的地址建立连接, 可按照IETF RFC3484中“IPv6选择默认地址”及其修订草案[8]规范要求, 以使得它能尝试不同的源地址以及目的地址。

这一方法的实行会潜在的导致长期超时。例如, 考虑在套接字API的一个执行, 使用getaddrinfo()检索所有目标地址, 然后试图用bind()和connect()尝试所有的源和目标地址的组合, 在尝试下一个之前要等待每一个组合的TCP超时。

但是, 若执行在一些新的connect-by-name() API中封装这些以及使用非阻塞连接调用, 它可能或通过现有的组合以更迅速的方法来循环, 直到找到可用的源和目的定位符。因此, 在这一领域的问题是执行和当前套接字API的问题, 而不是协议定义的问题。其实, 与为TCP和其他面向连接的传输提供一个易于使用的connect-by-name() API是容易, 为UDP在API提供一个相似的能力是困难的, 这是因为协议本身不提供任何“成功”反馈。但即使UDP问题也是API和执行中的一个问题。

14 失败检测以及定位符探测

14.1 概述

shim6协议扩展了IPv6使之能够支持多归属。这是一个IP层的机制, 使得多归属对应用是不可见的。shim6解决方案的一部分包括: 当在两个通信节点之间的当前使用的一对地址(或接口)失效, 要及时发现, 并且当此发生时要选择另一对可用的。我们称前者为“失败检测”(failure detection), 后者为“定位符对探测”(locator pair exploration)。

本部分描述了该机制以及协议报文来实现失败检测和定位符对探测。shim6协议的这一部分称为可达性协议(REAP)。

失败检测要尽可能的轻量化。要观测双向的净荷数据流, 且在由于通信空闲而使得在没有流的情况下, 失败检测也是空闲的且不生成任何的分组。当净荷流在双向流动时, 也没有必要去发送失败检测分组。仅当只有一个方向有流时, 失败检测机制才在另一个方向生成Keepalive报文。这样只要是仅有发出的流而没有返回的流或Keepalive, 则一定是有失败, 这样定位符探测要开始为双方寻找可用的地址对。

在这些描述中, 我们认为地址和定位符是同义的。shim6协议的其他部分确定同一个节点使用的不同定位符是属于一起的。即, REAP并不确保提到的节点以一个合法的定位符结束。

REAP是专门设计用于shim6的，因此它适合于在主机上运行的环境，使用广泛变换的路径，且不知道应用上下文。因此，REAP试着尽可能的进行自配置以及不引人注目。特别地，除了一定需要以及使用指数退避来防止拥塞，它避免发送任何分组。其缺点是，它不能提供相同粒度的发现问题的机制，该机制有更多的应用上下文和能力去协调和配置参数。

注：IETF RFC5534的未来版本可能会考虑扩展这些方面的能力，例如，通过从双向转发检测(BFD)协议（参见[31]）中继承一些机制。

14.2 失败检测以及定位符探测协议概览

14.2.1 概述

这一部分讨论了可达性检查和完全可达性探测机制的设计，且给出了REAP协议的概念。

如果两个节点都有两个或更多的地址，探测它们之间的全部通信选项集是一个代价很大的操作，因为随着地址数的增加，要探测的结合数增加的非常快。例如，双方都有两个地址，则有四种可能的地址对。由于我们无法假设在一个方向的可达性就意味着在另一方也可达，因此双向的结合总数是8（结合= $n_A * n_B * 2$ ）。

在多归属中一个重要的发现是失败相对较少，所以一个几秒前工作的可用对很可能仍然可用。因此，有一个轻量化的确保存在的可达性的协议，并仅在可疑的失败是才触发稍大的探测机制，这一点是有意义的。

14.2.2 失败检测

失败检测由三部分组成：跟踪本地信息，跟踪远端状况，以及最后验证可达性。跟踪本地信息由使用例如本地路由的可达性信息作为一个输入来组成。节点应当使用在3.19和3.20列出的技术去跟踪本地情况。也需要从对端跟踪远程地址信息。例如，如果在当前地址对的对端地址不再本地可用，则需要转发那个信息的机制。在shim6协议中的更新请求报文用于这一目的。最后，当本地和远程信息显示那个通信是可能的且有上层分组要发送，需要可达性验证来确保对端确实有可用地址对。

一项称为强制双向检测（FBD）的技术用于可达性验证。在一个shim6上下文中的当前使用地址对的可达性是由确保无论何时只要在一个方向上有净荷流在另一个方向也有这一点来确定的。这也能是数据流，或若没有其他的流也可能是传输层确认或是REAP可达性Keepalive。这样就再不可能仅在一个方向有流；所以只要有净荷流发出但又没有返回分组，则一定是出现了失败，此时完全探测机制要启动。

一个更为详细的对当前可达性评价机制的描述：

a) 为了避免其他端认为这里有一个可达性失败，一个运行失败检测机制的节点有必要在没有其他流时生成周期性的 Keepalive。

b) FBD 通过若节点从对端收到分组但自身没有发送任何分组则产生 REAPKeepalive 来工作。Keepalive 在一定的时间间隔发送，这样当对端在 Send Timeout 的时间没有收到任何分组则认为有可达性问题。节点在 I2, I2bis, R2,或 UPDATE 报文中的 Keepalive Timeout 选项（5.16.9）与对端交流 Send Timeout 值。对端则把这个值设为它的 Keepalive Timeout 值。

c) Keepalive 发送的时间间隔成为 Keepalive 间隔（Keepalive Interval）。建议的 Keepalive 间隔方法是在二分之一到三分之一的 Keepalive Timeout 间隔发送 Keepalive，这样生产了多个 Keepalive 且在它超时之前有时间到达对端。

d) 无论什么时候生成了一个发出的净荷分组, 一个计时器会启动以反映对端应当从净荷分组生成返回流的需求。超时值设为 Send Timeout 值。

e) 为了这个规范的目的, “净荷分组”指任何的属于 shim6 上下文一部分的分组, 包括上层协议分组和 shim6 协议报文, 除本规范中定义的。对于后者报文, 第 14.3 节指明了当一个报文收发事会发生什么。

f) 无论什么时候收到进入的净荷分组, 与从对端返回的流相关的计时器要停止, 同时另一个计时器启动以反映这个节点需要生成返回流。这个超时值设为 Keepalive Timeout 值。

g) 这两个计时器是互斥的。换句话说, 要么该节点希望看到基于本节点早些时候所发送流的来自对端的流, 要么该节点希望基于对端早些时候所发送流回应对端 (否则该节点在 idle 状态)。

h) 收到 REAP Keepalive 报文导致停止了与从对端返回流相关的计时器。

i) 在一个上下文的最后一个收到的净荷分组后 Keepalive 间隔到时, 如果这个上下文自从该净荷分组收到后没有其他分组发送, 一个 REAP Keepalive 报文会为该上下文生成且发送给对端。一个节点可能比 Keepalive 间隔要短就发送 Keepalive, 但应当注意避免发送 Keepalive 速率太快。REAP Keepalive 报文应当在 Keepalive 间隔持续发送, 直到收到对端的该 shim6 上下文的一个净荷分组或 Keepalive Timeout 超时才停止。如果一个或多个净荷分组在 Keepalive 间隔内发送则不需要发送 Keepalive。

j) 在传输了一个净荷分组而没有对该上下文的返回流, 若 Send Timeout 超时, 则启动一个完全可达性探测。

第15章提供了这些超时值的一些建议默认值。实际的值为了防止同步应当是随机的。对shim6协议部署的经验是必要的, 以确定哪些值是最合适的。

14.2.3 完全可达性探测

如上一节所述, 当前使用地址对可能变得无效, 或是通过地址中的一个变得不可用或不可操作, 或是通过地址对自己宣传不可操作。一个探测处理试着找到另一可用对以恢复通信。

使这个过程困难的是需要支持单向可用地址对。通过一个简单的请求 - 应答协议不足以探测到地址对。相反, 第一次发现问题的一方开始一个处理, 它通过向对端发送报文来按序尝试每一个不同的地址对。这些报文装载对端之间的连接状态信息, 例如发送方最近是否从对端看到任何流。当对端收到指出一个问题的报文, 它通过开启它自己的一个另一方向的并行的探测来辅助该处理, 再发送关于目前收到的净荷流的信息或信令报文。

具体地, 当A决定它需要探测替代的地址对去B, 它将按序初始化探测报文集, 直到它从B得到一个探测报文说明 (a) B已收到A的报文中的一个, 显然的 (b) B的探测报文返回给A。B使用相同的算法, 但是从收到A的第一个探测报文开始该流程的。

由于换了新的地址对, 网络路径遍历很可能改变, 所以应当通知上层协议。这能成为ULP根据路径的改变进行适应的一个信号, 这样例如, 如果ULP是TCP, 它可以启动一个缓慢的启动过程。但有可能新路径选择的环境已经导致了足够的分组丢失以触发慢启动。

REAP设计用于支持失败恢复, 即使在仅有单向可用地址对的情况。但由于在第17章讨论的安全考虑, 探测流程一般能够在已建立的会话上运行。具体地尽管在连接建立过程REAP在理论上能够探测, 但shim6协议对它的使用不允许这一点。

14.2.4 探测顺序

探测流程假定了一种用于选择地址对用于测试的能力。REAP使用的选择流程总体如下:

——作为启动该流程的输入，节点了解它自己的地址，且通过 shim6 协议报文来被告知对端是什么地址。通过结合这两种信息，一个可能的地址对表就建立了。

——通过使用标准 ipv6 地址选择规则，该表经删除不合适的组合而得到修整，不合适的组合例如与使用全球单播地址的对端联系时试着使用链路本地地址。

——同样，标准 ipv6 地址选择规则为这些地址对提供了一个基本的优先级顺序。

——本地偏好可能用于对表的顺序做一些附加的调整。对本地偏好设置的机制没有被指明但能包括，例如，为使用一个接口而不是另一个而设置偏好的配置。

——结果就是该节点有着按优先级的地址对表可以尝试。但这个表可能仍然很长，若双方都有很多地址则可能有非常多的组合。REAP 按序应用这些对，但使用一个退避方法来避免“信令风暴”。这确保探测流程是相对保守的或“安全的”。缺点是若双方有很多地址则找到一个可工作的地址对要花一些时间。

更具体的流程如下：节点首先查询IETF RFC3484默认地址选择规则来确定从本地的观点什么样的地址组合是允许的，因为这减少了搜索空间。IETF RFC3484也提供了在不同地址对间的优先顺序，这可能也能够使搜索更快（更多的机制可能在将来定义，在测试开始前形成一个初始的顺序地址对，参见[35]）。节点也可能使用本地信息（例如已知服务参数的质量或接口类型）来确定更喜好什么地址，并先尝试带有这类地址的地址对。shim6协议也在它的报文中装载偏好选项。

从这些可能的候选地址对集里，节点应当试着测试它们所有直到找到可用对，且如果需要的话重试该流程。但所有的节点一定要顺序的进行这一流程且使用指数退避。为了当发生停用时（特别是整个站点）避免“信令风暴”，这一顺序流程是需要的。但它也限制了能够实际用于多归属的地址的数目，考虑到如果切换到一个新地址对用时太长，传输层和应用层协议将失败。

第15章介绍了与探索流程相关的计时器的默认值。初始探测超时值（0.5s）指明了初始尝试发送探测之间的间隔；初始探测数（4）指明了在需要使用指数退避程序之前可以发送多少初始探测。如果没有响应，这一流程增加了每个探测之间的时间。一般地，每个增加都加倍了该时间，但本标准并未授权特别的增加。

在使用指数退避之前以固定速率发送4个分组的根本原因是避免过快的发送这些分组。如果没有这些，第三和第四个探测之间间隔0.5s意味着第一和第二个探测之间的时间必须是0.125s，这使得留给应答第一个分组的时间非常少。同样，这意味着头四个分组在0.875s内发送而不是2s，如果大量的shim6上下文在经历失败后同时发送探测则增加了拥塞的可能。

最大探测超时（60s）指明了一个限制，探测间隔的增加不能超过它。如果探测流程到达了间隔，它将继续以这一速率发送，直到触发一个合适的回应或shim6上下文被垃圾回收，因为使用有问题的shim6上下文的上层协议不能试着发送分组。到达最大探测超时也可能作为一个对垃圾回收流程的暗示：该上下文不再有用。

14.3 协议行为

14.3.1 概述

REAP节点所需的运转情况在下面以状态机的形式描述。一个执行的外部可观察的运转情况必须符合本状态机，但该执行也不必确切的使用一个状态机。与下面的描述相混合，我们也以表格形式提供一个状态机描述。但这种格式仅是信息性的。

在给出的对端的上下文中，节点将是三种状态的一种：Operational, Exploring, 或 InboundOK。在Operational状态，基本地址对是可用的。在Exploring状态，这个节点在超过了一个发送计时器的周期没有看到来自对端的任何流。最后，在InboundOK状态，这个节点看到来自对端的流，但对端可能没有看到该节点的流，所以探测流程需要继续。

该节点也要维持发送计时器（Send Timeout）和Keepalive计时器（Keepalive Timeout）。发送计时器反映了当此节点发送一个净荷分组的需求，应当在Send Timeout超时前有一些返回流。Keepalive计时器反映当此节点收到一个净荷分组的需求，应当向对端有一个相似的反应。Keepalive计时器仅在Operational状态使用，且Send Timer在Operational和InboundOK状态使用。在Exploring状态没有计时器。如14.2.2小节所解释，这两个计时器是互斥的。即两个只有一个能在运行，或都没有运行。

14.3.2 收到净荷分组

根据在Operational状态收到的净荷分组，节点启动Keepalive计时器，如果它还没有运行，且如果发送计时器在运行则停止。

如果节点在Exploring状态，它转变到InboundOK状态，发送一个探测报文，并启动发送计时器。它使用最近探测报文的还没有被对端知道的信息填充Psent和相应的探测源地址，探测目的地址，探测随机数，以及探测数据字段。它也使用已从对端看到的最近探测报文的信息填充Precvd和相应的探测源地址，探测目的地址，探测随机数，以及探测数据字段。当发生一个探测报文，状态字段必须在发送该探测后被设为匹配发送方概念上状态的值。在这种情况下，该节点设置状态字段为2（InboundOk）。用于发送该探测报文的IP源和目的地址按照14.2.4规定选择。

在InboundOK状态，若发送计时器在运行则该节点把它停止，但不做任何事。

收到shim6控制报文而不是Keepalive和探测报文，则像收到净荷分组一样对待。

当Keepalive计时器在运行时，节点应当发送Keepalive报文给对端，报文中有Keepalive间隔。概念上，一个独立的计时器是用于区分Keepalive报文之间和整体Keepalive超时间隔之间的间隔。但这个独立的计时器没有在表格或图形状态机中体现。当发送时，该Keepalive报文按5.13规定所组成。它是由当前地址对发送。

在下面的表中，"START", "RESTART", 和 "STOP"标识各自地启动、重启和停止Keepalive计时器或发送计时器。"GOTO"标识转到另一状态。"SEND"标识发送一个报文，而"-"表示不采取行动。

Operational	Exploring	InboundOk
STOP Send	SEND Probe InboundOk	STOP Send
START Keepalive	START Send	
	GOTO InboundOk	

14.3.3 发出净荷分组

根据在Operational状态发送一个净荷分组，若Keepalive计时器在运行则停止，若发送计时器没有工作则启动。在Exploring状态没有影响，而在InboundOK状态该节点仅当发送计时器没运行则启动。（发送shim6控制报文同样对待。）

Operational	Exploring	InboundOk
-------------	-----------	-----------

START Send	-	START Send
STOP Keepalive		

14.3.4 保活超时 (Keepalive Timeout)

根据Keepalive计时器超时，节点发送一个最后的Keepalive报文。这仅会在Operational状态发生。Keepalive报文由5.13节描述的所组成。它使用当前地址对发送。

Operational	Exploring	InboundOk
<hr/>		
SEND Keepalive	-	-

14.3.5 发送超时 (Send Timeout)

根据发送计时器超时，节点进入探测状态且发送一个探测报文。该探测报文由14.3.2小节描述的所组成，且状态字段设为1 (Exploring) 。

Operational	Exploring	InboundOk
<hr/>		
SEND Probe Exploring	-	SEND Probe Exploring
GOTO Exploring		GOTO Exploring

14.3.6 重传

当在Exploring状态，该节点继续重传它的探测报文到不同（或相同）地址，见14.2.4。InboundOK状态使用相似的流程，除了根据这个重传，发送计时器如果还没有运行要启动。

探测报文按14.3.2描述的组成，且根据发送方处于什么状态要把状态字段设为1 (Exploring) 或2 (InboundOK) 。

Operational	Exploring	InboundOk
<hr/>		
	SEND Probe Exploring	SEND Probe InboundOk
		START Send

14.3.7 接收 Keepalive 报文

根据在Operational状态收到一个Keepalive报文，如果发送计时器在运行则该节点停止它。如果该节点在Exploring状态，它转到InboundOK状态，发送一个探测报文，且启动一个发送计时器。该探测报文按14.3.2描述的组成。

在InboundOK状态，若发送计时器在运行则停止。

Operational	Exploring	InboundOk
<hr/>		
STOP Send	SEND Probe InboundOk	STOP Send
	START Send	
	GOTO InboundOk	

14.3.8 接收探测报文 State=Exploring

根据收到一个探测消息，状态是Exploring，则节点进入InboundOK状态，发送一个探测报文，见14.3.2小节所述，Keepalive计时器在运行则停止，重启发送计时器。

Operational	Exploring	InboundOk
SEND Probe InboundOk	SEND Probe InboundOk	SEND Probe InboundOk
STOP Keepalive	START Send	RESTART Send
RESTART Send	GOTO InboundOk	
GOTO InboundOk		

14.3.9 接收探测报文 State=InboundOk

根据收到探测报文，状态是InboundOK，该节点发送一个探测报文，重启发送计时器，如果Keepalive计时器在运行则停止，且转到Operational状态。根据在我们刚收到报文中接收探测的报告，为这个连接选择一个新的当前地址对。如果没有接收的探测已被报告，当前地址对不变。

探测报文按14.3.2所述的组成，且状态字段设为0（Operational）。

Operational	Exploring	InboundOk
SEND Probe Operational	SEND Probe Operational	SEND Probe Operational
RESTART Send	RESTART Send	RESTART Send
STOP Keepalive	GOTO Operational	GOTO Operational

14.3.10 接收探测报文 State=Operational

根据收到一个探测报文，状态是Operational，若发送计时器在运行则该节点停止它，若Keepalive计时器没有运行则启动，且转到Operational状态。该探测报文由14.3.2小节所说的组成，且状态字段设为0（Operational）。

注意：当双方协商一致的情况下，则可以终止探测流程。

Operational	Exploring	InboundOk
STOP Send	STOP Send	STOP Send
START Keepalive	START Keepalive	START Keepalive
	GOTO Operational	GOTO Operational

一个新的可用对被确认前，可达性检查和探测流程对净荷通信没有影响。在此之前，净荷分组继续发送到先前使用的地址。

15 协议常量

本协议使用下列常量：

- I1_RETRIES_MAX = 4;
- I1_TIMEOUT = 4 seconds;
- NO_R1_HOLDDOWN_TIME = 1 min;
- ICMP_HOLDDOWN_TIME = 10 min;
- I2_TIMEOUT = 4 seconds;
- I2_RETRIES_MAX = 2;

- I2bis_TIMEOUT = 4 seconds;
- I2bis_RETRIES_MAX = 2;
- VALIDATOR_MIN_LIFETIME = 30 seconds;
- UPDATE_TIMEOUT = 4 seconds;
- MAX_UPDATE_TIMEOUT = 120 seconds。

重传计时器(I1_TIMEOUT, I2_TIMEOUT, UPDATE_TIMEOUT)服从二进制指数退避, 且在0.5倍到1.5倍的标称值范围内随机选择。这样就消除了许多的主机在同一时间独立的实行shim操作可能会出现同步的危险。

随机是在二进制指数退避后执行的。因此, 第一次重传将基于在 $[0.5 \times 4, 1.5 \times 4]$ 秒的均匀分布随机数来发生; 第二次重传在第一次之后的 $[0.5 \times 8, 1.5 \times 8]$ 秒, 以此类推。

失败检测以及定位符探测部分还定义了下列协议常量:

- Initial Probe Timeout = 0.5 seconds;
- Number of Initial Probes = 4 probes。

且这些变量有如下的默认值:

- Send Timeout = 15 seconds;
- Keepalive Timeout = X 秒, X 是对端通信的 Keepalive 超时选项中的发送超时 15s, 若对端没有发送 Keepalive 超时选项;

- Keepalive Interval = Y 秒, Y 是 Keepalive 超时值的三分之一到二分之一 (见 14.2.2)。

节点可能选择发送超时的替代值且在保活超时选项中告知对端。一个很小的发送超时值可能影响在有一个很大的往返延迟的路径上交换保活的能力。同样地, 它可能导致shim6对暂时的失败作出反应比需要的更多。因此建议替代的发送超时值不要小于10s。选择一个比上面建议的更大的值也是可能的, 但发送超时和REAP在通信路径中发现和改成错误的能力是有关系的。不管怎样, 为了shim6能有用, 应当在上层放弃之前发现和修复问题。因此, 建议发送超时最大为100s (默认TCP R2超时, 参见[36])。

注: 不希望发送超时或其他值在基于经历的往返时间来估量。信令交换基于指数退避交换。保活流程发送分组仅在相对少见的所有流都是单向的情况下。

16 其他地方的影响

16.1 拥塞控制考虑

当目前在shim6上下文用于交换分组的定位符对变得不可达了, shim6层将通过替代的定位符对来转移该通信, 在大多数情况下会导致通过一个替代的网络路径重定向分组流。在这种情况下, 我们建议shim6按照[20]中的提议, 且通知上层这种路径的改变, 目的是让上层的拥塞控制机制作出相应的反应。

16.2 中间设备(Middle-Boxes)考虑

属于一个shim6上下文的载有shim6净荷头的数据分组包含有替代的定位符, 而非ULID在IPv6头的源和目的地址字段。另一方面, 包含在该通信的对端的上层通过shim6层来操作ULID对, 而不是去操作在IPv6头中包含的定位符对。应当注意的是, shim6层不修改数据分组, 但由于不论定位符对是否改变, 呈现在上层的是一个不变的ULID对, 上层头部 (例如TCP, UDP, ICMP, ESP, 等等) 和IPv6头部之间的关系改变了。特别地, 当shim6扩展头出现在分组中, 若那些数据分组是TCP, UDP, 或 ICMP分组, 用于校验和计算的伪头部将包含该ULID对, 而不是包含在数据分组中的定位符对。

有可能一些防火墙或其他中间设备会试着验证在传输的分组的上层完整性校验的正确性。如果它们是基于包含在IPv6头的确切的源和目的地址而没有考虑shim6上下文信息（特别是若没有用shim6上下文使用的ULID对去替换定位符对）来验证，这一验证可能会失败。那些中间设备需要更新以能够编译shim6净荷头以及发现下一头。建议当包含IPv6头的地址来进行计算，出现明显的上层正确性校验错误时，防火墙和其他中间设备也不要轻易丢弃装载了shim6净荷头的分组，除非能够确定该shim6上下文的ULID对与该数据分组相关且使用这个ULID对来进行正确性校验的验证。

在TCP, UDP, 和 ICMP校验和的特殊情况，建议当使用IPv6头的地址来进行伪头计算时，出现明显错误的校验和，防火墙和其他中间设备也不要轻易丢弃装载了shim6净荷头的TCP, UDP, 和ICMP分组，除非它们能够确定该shim6上下文的ULID对与该数据分组相关且使用这个ULID对来决定该校验和一定要在一个通过shim6来进行地址重写的分组中出现。

此外，现在只通过有限的流（例如，发出的TCP连接）的防火墙很可能阻止shim6协议。这意味着即使支持shim6的主机在通信，I1消息仍会丢失；因此，这些主机不会发现他们的对端是支持shim6的。这实际上是一件好事，因为如果主机试着去建立ULID对上下文，防火墙很可能会丢弃经过一次失败后发送的“不同的”分组（那些使用shim6净荷扩展头，TCP报文在里面的）。因此，修改了的可以通过shim6消息的有状态的防火墙也能经过修改去通过shim6净荷扩展头，这样shim6能使用替代的定位符去从失败中恢复。这可能意味着防火墙需要通过观察shim6控制交互来跟踪在用的定位符集。这种防火墙可能甚至想它们自己来通过HBA/CGA验证对定位符进行验证，它们可以不修改任何通过它们的shim6分组来实现。

16.3 操作和管理考虑

本节考虑与操作和管理shim6协议相关的一些方面。

shim6协议的部署：shim6协议是一个基于主机的解决方案。因此，为了部署，使用shim6协议的主机协议栈需要进行更新以支持它。这是一个增量的部署，因为它不需要为该部署设定一个标识日期，仅仅是单一的主机更新。如果shim6在一个站点进行部署，那些主机会逐渐进行更新以支持该方案。此外，为了支持shim6协议，仅有端点主机需要进行更新而路由器不需要进行改变。但应当注意，为了从shim6协议中受益，通信双方都应支持该协议，即双方都应更新以支持shim6。不过shim6协议使用的是延迟的上下文建立方法，即允许主机先进行常规的IPv6通信，此后如果双方都支持shim6，再建立shim6上下文以使用shim6协议。这对部署有着重要的好处，因为在不对通信带来问题的前提下，shim6主机可以完美的和非shim6主机通信。

对shim6节点配置：shim6协议本身不需要任何特别的配置去提供它的基本特性。shim6协议是设计用于为上层提供一个默认的服务，即应当满足一般的应用。shim6应当能够自动使用提前定义好的启发机制去触发建立shim6上下文以保护长期的通信。当然，如果一些应用需要一些特殊的调节，这可能需要额外的配置。类似的考虑也应用在一个站点试着去进行一些格式的流工程，通过使用不同定位符的不同优先序。

地址和前缀配置：shim6协议假设在一个多归属的站点将会有多个可用的前缀。这种配置能增加一个网络中的操作工作。但应当注意在一个站点有多个前缀以及多个地址分配给一个接口的能力是IPv6的能力，这已经超出shim6的情况，并且这希望能广泛使用。因此，虽然这是shim6的情况，我们考虑这一配置的影响超出了shim6的特别情况且必须为一般的IPv6情况解决。然而shim6也假设shim6主机解决了CGA/HBA的使用。这意味着shim6主机应当使用HBA/CGA生成机制来配置地址（其他情况参见[18]）。

16.4 其他方面的考虑

一般的shim6方法以及指定的建议解决方案有其他方面的考虑，包括：

——进行转介(referrals)或回调(callbacks)的应用使用 IP 地址作为‘标识符’仍能使用有限的方法来正常工作，参见[17]。但为了让这种应用能够对冗余利用多个定位符，该应用需要修改使用主机域名全称为‘标识符’，或是它们需要通过所有的定位符作为‘标识符’，即该‘标识符’从应用的角度看成为了 IP 地址集而不只是一个 IP 地址。

——信令协议为 QoS 或为其他的包括在网络路径中有设备查看 IP 地址和端口号（或查看 IP 地址和流标签）的东西，该信令协议需要被主机调用当由于一个失败导致定位符的改变。在那个时间点，那些协议需要通知设备一个新的 IP 地址对将用于这个流。注意，尽管这个协议不像一些早些的协议，它仍没有装载流标签作为一个上下文标签；路径内的设备需要知道新定位符的使用，尽管流标签没有变。

——MTU 影响。通过在最近观察的路径 MTU 基础上计算一个最小，我们所使用的路径 MTU 机制对在 Internet 中不同的分组使用不同的路径是健壮的。当 shim6 由于故障从一个定位符对切换到另一定位符对，这意味这分组要穿过 Internet 中不同的路径；因此，该路径 MTU 可能会很不同。为了处理在该 MTU 中的这种改变，推荐使用分组层路径 MTU 发现（Packetization Layer Path MTU Discovery）（参见[23]）。

事实上在一次定位符选择后 shim 添加一个 8byte 的 shim6 净荷扩展头在 ULP 分组上仍影响为该 ULP 的可用路径 MTU。在这种情况下，对发送主机 MTU 改变时本地的；因此向 ULP 传达这个改变是一个执行任务。通过传达该信息给传输层，它可以适应和相应的减小最大段大小(MSS)。

16.5 失败检测以及定位符探测的操作考虑

当没有失败时，失败检测机制（以及一般的shim6）轻量化了：当shim6上下文空闲时，或当有双向的流时不发送保活。所以在正常的TCP或TCP-链接操作，当一个会话从活跃到空闲将仅有一或两个保活。

仅当有失败在有意义的失败检测流，特别是在一个有许多活跃会话且被多个节点共享的链接失败的情况下。当此发生了，会发送一个保活，且接下来是一系列的探测。这在每个活跃的（流生成的）上下文上发生，它们在该失败后都将在15秒内超时。这使得流到达高峰，shim6在失败后每个上下文每秒生成一个分组。假定地，在那些上下文上运行的对话最少发送那么多流且很有可能更多，但如果备份路径大大低于失败路径带宽，这可能导致暂时的拥塞。

但注意到多归属使用BGP的情况，如果该失败足够的快，TCP没有进入慢开始，在失败路径上的完整净荷数据流切换到备份路径，且若备份路径能力更低，则将会有更多的拥塞。

尽管失败检测探测没有像那样进行拥塞控制，但指数退避确保发送的分组数很快会降下来且最终达到每分钟每个上下文一个，这即使是在最低带宽链路上也足够保守了。

注：第15章规定了一些协议参数，这些参数有可能会发生调整，以及在本标准中没有授权的其他一些方面，可能会影响上面提到性质。希望本标准的未来修订能在从不同的环境获得足够的部署经验后，提供更多的信息。

执行可能提供方法去监管它们的性能以及对问题发出警报。但它们的标准是未来规范的目标。总的来说，shim6对小型站点和节点最为合适，且希望在这种部署上的监管需求相对温和。不管怎样，在节点与一个管理系统相关的地方，建议检测的失败和故障切换事件通过异步通知来报告给管理系统。同样地，在节点上日志机制是有用的，这些事件应当被记录在事件日志中。

在一个主机转换（rehomeing）事件后，shim6为信令和数据包净荷封装使用相同的头。这样，两种类型的分组一起同进退，所以可达性探测或保活能成功传送而净荷分组不能的情况大量的避免了：要么所有的shim6分组能通过，要么都不能，且没有shim6状态被协商。甚至在一些分组通过而其他的不能的情况，shim6将一般按计划的工作或提供不比没有shim6更糟的服务，除了可能生成少量信令流。

有时净荷分组（且有可能是封装shim6头的净荷分组）不能通过，但信令和保活能够。当在一条路径上有一个路径MTU发现黑洞时这种情况会出现。如果仅在某些时候有大分组发送，则可达性探测仅被开启且REAP将可能选择另一路径，这有可能或可能不被PMTUD黑洞影响。

17 安全考虑

17.1 概述

本标准满足了IETF RFC4218描述的以下要求：

——IETF RFC5535 HBA和IETF RFC3972 CGA技术，用于验证定位符以阻止一个攻击者重定向分组流到其他地方，也阻止IETF RFC4218中的4.1.1、4.1.2、4.1.3以及4.2描述的威胁。这两种技术提供了相似等级的保护但也提供了不同的计算代价的不同功能。HBA机制依靠生成所有的多归属主机的地址作为一个不变的内部边界IPv6地址集，称为HBA集。在这种方法中，地址将可用的前缀集的加密单项哈希并入到接口标识符的一部分。其结果是，在所有可用地址间的绑定被地址本身编码，提供了劫持保护。任何使用了shim6协议的对端节点能通过一个简单哈希计算有效验证被提议的用于继续通信的替代地址是与初始地址绑定的。在一个基于CGA的方法中，用作ULID的地址是一个CGA，它包含了一个公共密匙的哈希值在它的接口标识符。结果是在ULID和相关密匙对间有一个安全绑定。这使得每个对端可使用相应的私有密匙去标志传达定位符集信息的shim6消息。在这种情况下的信任链如下：用于通信的ULID安全的绑定密匙对因为他包含了公共密匙的哈希值，且定位符集通过签名与公共密匙绑定。HBA和CGA两种机制的任一种都提供了IETF RFC4218中4.1.2所述时移攻击的保护，因为ULID安全的绑定一个定位符集，仅能被该ULID对的所有者定义。在生成CGA中对RSA密匙可接受的最小密匙长度必须最少为1024位。任何执行都应遵循用于决定恰当的密匙长度的谨慎的加密方法。

——IETF RFC4218 中 4.3 描述的第 3 方洪范攻击通过要求在接受一新定位符用作分组的目的地之前，shim6 对端实行一个成功的可达性探测 + 回复交换来进行阻止。

——第一条消息不再应答方创建任何的状态。实际上，在应答方创建任何状态之前需要一个 3 次交互。这意味着基于状态的 Dos 攻击（试着用光应答方的所有内存）最少需要攻击者去创建状态，消耗他自己的资源；它也提供攻击者使用的 IPv6 地址。

——上下文建立消息使用随机数来阻止 IETF RFC4218 中 4.1.4 描述的重放攻击，也阻止路径分离攻击者干涉建立。

——shim6 协议的每一控制消息，传输上下文建立，都装载分配给特定上下文的上下文标签。这意味着一个攻击者再欺骗任何 shim6 控制消息之前需要发现上下文标签（IETF RFC4218 中 4.4 描述）。要发现上下文标签需要攻击者沿着路径以察觉上下文标签值。结果是，通过这一技术，shim6 协议不受路径分离攻击者攻击。

17.2 与 IPsec 交互

shim6有两种处理数据分组的模式。如果该ULID对也是被使用的定位符对，则数据分组不被shim6修改。在这种情况下，与IPsec的交互与没有shim6时是完全一样的。

如果ULID对和目前的定位符对不同，则shim6将拿出那个数据分组，把IP源和目的地址字段的ULID换为当前定位符对，且添加shim6扩展有着相应的上下文标签。在这种情况下，如4.2提到的，shim6从概念上像隧道机制一样工作，进入的头包含ULID而出的头包含定位符。主要的区别是进入的头被“压缩”，且添加一个压缩标签（即上下文标签）以在接收端解压该进入的头部。

在这种情况下，IPSec和shim6的交互与IPSec和隧道机制之间的交互相似。当分组有上层协议生成，它传给IP层，包含ULID在IP源和目的字段。IPSec接下来应用到这个分组。然后改分组传给shim6子层，shim6子层封装收到的分组，包含一个新的IP头，该头包含定位符对在IP源和目的字段。这个新的IP分组按序传给IPS处理，正如隧道的情况。这可以看做shim6子层在IPSec之间，且IPSec策略同时应用在ULID和定位符。

当在shim6子层已经处理过该分组后（即该分组在IP源和目的地址字段装载定位符）IPSec再处理该分组，shim6子层可能已经添加了shim6扩展头。在那种情况下，IPSec需要忽略shim6扩展头去为下一层的协议（例如，TCP, UDP, SCTP）找选择符。

当在另一端收到一个分组，它基于扩展头的顺序处理。因此，如果一个ESP或AH头在shim6头之前，那决定了顺序。shim6引入了策略检测的需要，类似隧道所做的，当shim6收到一个分组且那个分组的ULID对与定位符对不同。

17.3 其他的威胁

本协议中的一些其他的威胁：

——迟来到该路径的攻击者（即上下文已建立）能使用 R1bis 来导致对端重建上下文，且在那时可以观测到所有的交互。但这看起来不能为攻击者开启新的门，因为这类攻击者能观察到上下文标签，且一旦知道就可以使用它们发送假消息。

——在路径上的攻击者为了能找到上下文标签，能在它已经离开该路径后生成一个 R1bis 消息。为了让这个分组有效，它需要有一个属于上下文的源定位符；因此在攻击者的新位置和通信对端间不会有“太多”入口过滤。但这似乎没有那么严重，因为一旦 R1bis 导致了上下文重建，将使用一对新的上下文标签，攻击者是不知道的。如果这仍值得关心，我们需要一个 2 次握手，“你真的丢失了改状态？”来回答出错消息。

——有可能攻击者会尝试随机的 47 位上下文标签，看看他们是否可能会导致两个主机之间的通信中断。特别地，在净荷分组的情况下，这一攻击的影响和那些发送假源地址的分组的攻击者类似。在控制分组的情况下，还不足以发现正确的上下文标签 – 还需要附加信息（例如，随机数，合适的源地址；参照以前的 R1bis 情况）。如果一个 47 位标签（是在 8byte 扩展头中最长的匹配）不足够，可能要在 shim6 控制消息中使用更大的标签，且使用 shim6 净荷扩展头的低序 47 位。

——当使用了 shim6 净荷扩展头，猜出 47 位随机上下文标签的攻击者能用任意源地址把分组注入上下文中。因此，如果在攻击者和它的目标之间有入口过滤，这可能使得攻击者也通过入口过滤。但除了要猜出 47 位上下文标签，攻击者还需要找到一个上下文，在这里，在接收者用 ULID 替代定位符之后，ULP 校验和才是正确的。但即使这样对像 TCP 一类的 ULP 来说还是不顾偶的，因为 TCP 端口号和序列号必须与一个存在的连接匹配。因此，尽管关闭路径攻击者注入分组这个问题与现在的入口过滤不同，路径分离攻击者仍然很难去猜测。如果应用了 IPsec,那么这个问题就不存在了。

——在 R1 和 R1bis 分组中的验证符的生成是几个输入参数的哈希值。大多数的输入是由发送方明确决定的，只有密码值 S 对发送方是不知的，由此产生的保护被认为是足够的，因为对攻击者来说，通过发送一个 I1 包仅获得一个新的验证符比进行所有的为确定密码 S 的计算要容易。但我们建议主机应定期更换密码 S。

17.4 失败检测以及定位符探测安全考虑

攻击者可能从底层以及从网络来模仿多种表象以让对端混淆哪个地址是或不是可用的。例如，攻击者可能模仿ICMP出错报文导致流会到别处或甚至出现连接断开。攻击者也可能模仿与网络附件相关的信息，路由发现，以及地址分配试图去使主机相信它们有Internet连接而实际上没有。

这可能导致使用不希望的地址或甚至拒绝服务。

此协议不给自己提供任何的保护以把自己从协议栈的其他部分中标识。无保护的标识不应当被看作对连接问题的证实。但甚至从它实行它自己的测试优先于选择一个新地址对的意义无保护标识，REAP对不正确的信息有着很弱的阻挡。但拒绝服务漏洞存在和在路径耦合攻击者漏洞一样。

这些漏洞的一些方面可以通过使用定义在协议栈的其他部分的技术来缓和，例如恰当的处理ICMP错误（参见[33]），链路层安全，或使用SEND（参见[37]）来保护IPv6路由器和邻居发现。

shim6协议的其他部分确保我们交换的地址集确实属于一起。REAP本身不提供这种确保。相似地，REAP提供一些保护防止第三方洪范攻击（参见[30]）；当REAP在运行，它的探测随机数可以用作一个返回可路由检测，该申明地址确实希望收到流。但这需要和另一机制一起完成，以确保该申明地址也是正确节点。shim6通过实行绑定所有操作到上下文标签来做到这一点。

在这个规范中的保活机制很容易模仿。能够看到shim6上下文标签的路径耦合攻击者能够在每个发送超时间隔发送一个模仿的保活报文，以阻止两个shim6节点自己发送保活。这一漏洞仅与包含在单向通行的节点相关。这个攻击的结果是该节点毫无必要的进入了探测阶段，但它们应当能够确认连接，除非攻击者能够阻止探测阶段完成。由于上下文标签是47位随机数，路径分离攻击者可能不能够生成模仿的结果。

为了防御模仿的保活报文，一个执行shim6和IPsec的节点如果有好的理由去假设另一端将要发送IPsec保护的返回流，可能忽略进入的REAP保活。换句话说，如果一个节点在发送TCP净荷数据，它能够有理由地期望收到TCP ACKs。如果没有IPsec保护的ACKs返回但无保护的保活却返回了，这可能是一个攻击者试着隐藏断开的连接的结果。

探测阶段对在路径上的攻击者来说是脆弱的。路径分离攻击者将发现很难猜测上下文标签或正确的探测标识符。由于IPsec在shim6层之上操作，不太可能用IPsec保护探测阶段不受耦合路径攻击者攻击。这和保护shim6其他控制报文相似。有已到位的机制来阻止把通信重定向到错误的地址，但耦合路径攻击者能够导致拒绝服务，把通信移到不常用地址对，等等。

最后，探测本身能导致许多分组要发送。结果就是，它可能在洪范攻击中用作分组扩大的工具。这需要使用REAP的协议已有机制防止这一点。例如，shim6上下文仅在相关的大量分组已经被交换才创建，这减少了使用shim6和REAP扩大攻击的吸引力。但这种保护一般在连接建立阶段没有。当连接建立成功需要探测，它的使用可能导致扩大漏洞。结果就是，shim6在连接建立阶段不支持使用REAP。

18 IANA 考虑

IANA允许shim6协议的新的IP协议号值（140）。

IANA在CGA的扩展类型标签中为shim6协议记录一个CGA消息类型，注册表值为0x4A30 5662 4858 574B 3655 416F 506A 6D48。

IANA用四个部分建立一个shim6参数注册表：shim6类型注册、shim6选项注册、shim6错误代码注册和shim6验证方法注册。

shim6类型注册表的初始内容见表7。

表7 shim6 类型注册表

类型值	报文
0	预留
1	I1 (发起方的第一个建立报文)
2	R1 (应答方的第一个建立报文)
3	I2 (发起方的第二个建立报文)
4	R2 (应答方的第二个建立报文)
5	R1bis (对相关的不存在上下文的回应)
6	I2bis (对R1bis报文的回应)
7-59	已分配的使用标准动作
60-63	用于试验
64	更新请求
65	更新确认
66	保活
67	探测报文
68	出错报文
69-123	已分配的使用标准动作
124-127	用于试验

shim6选项注册表的初始内容见表8。

表8 选项注册表

类型	选项名
0	预留
1	应答方验证符
2	定位符列表
3	定位符优先顺序
4	CGA参数数据结构
5	CGA签名
6	ULID对
7	分叉实例标识符
8-9	已分配的使用标准动作
10	保活超时选项
11-16383	已分配的使用标准动作
16384-32767	用于试验

shim6错误代码注册表的初始内容见表9。

表9 错误代码注册表

代码值	描述
0	未知的shim6报文类型
1	关键选项未识别
2	定位符验证方法失败
3	定位符列表生成数不同步
4	定位符数目出错
5-19	已分配的使用标准动作
120-127	预留作为调试用

shim6验证方法注册表的初始内容见表10。

表10 验证方法注册表

值	验证方法
0	预留
1	CGA
2	HBA
3-200	已分配的使用标准动作
201-254	用于试验
255	预留

参 考 文 献

- [1] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [2] Aura, T., "Cryptographically Generated Addresses (CGA)", RFC 3972, March 2005.
- [3] Bagnulo, M., "Hash-Based Addresses (HBA)", RFC 5535, June 2009.
- [4] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, February 2000.
- [5] Ferguson, P. and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing", BCP 38, RFC 2827, May 2000.
- [6] Draves, R., "Default Address Selection for Internet Protocol version 6 (IPv6)", RFC 3484, February 2003.
- [7] Nordmark, E., "Multihoming without IP Identifiers", Work in Progress, July 2004.
- [8] Bagnulo, M., "Updating RFC 3484 for multihoming support", Work in Progress, November 2007.
- [9] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.
- [10] Abley, J., Black, B., and V. Gill, "Goals for IPv6 Site-Multihoming Architectures", RFC 3582, August 2003.
- [11] Rajahalme, J., Conta, A., Carpenter, B., and S. Deering, "IPv6 Flow Label Specification", RFC 3697, March 2004.
- [12] Eastlake, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, June 2005.
- [13] Hinden, R. and B. Haberman, "Unique Local IPv6 Unicast Addresses", RFC 4193, October 2005.
- [14] Nordmark, E. and T. Li, "Threats Relating to IPv6 Multihoming Solutions", RFC 4218, October 2005.
- [15] Huitema, C., "Ingress filtering compatibility for IPv6 multihomed sites", Work in Progress, September 2005.
- [16] Bagnulo, M. and E. Nordmark, "SHIM - MIPv6 Interaction", Work in Progress, July 2005.
- [17] Nordmark, E., "shim6-Application Referral Issues", Work in Progress, July 2005.
- [18] Bagnulo, M. and J. Abley, "Applicability Statement for the Level 3 Multihoming Shim Protocol (shim6)", Work in Progress, July 2007.
- [19] Moskowitz, R., Nikander, P., Jokela, P., and T. Henderson, "Host Identity Protocol", RFC 5201, April 2008.
- [20] Schuetz, S., Koutsianas, N., Eggert, L., Eddy, W., Swami, Y., and K. Le, "TCP Response to Lower-Layer Connectivity-Change Indications", Work in Progress, February 2008.
- [21] Williams, N. and M. Richardson, "Better-Than-Nothing Security: An Unauthenticated Mode of IPsec", RFC 5386, November 2008.

- [22] Komu, M., Bagnulo, M., Slavov, K., and S. Sugimoto, "Socket Application Program Interface (API) for Multihoming Shim", Work in Progress, November 2008.
- [23] Mathis, M. and J. Heffner, "Packetization Layer Path MTU Discovery", RFC 4821, March 2007.
- [24] Bonica, R., Gan, D., Tappan, D., and C. Pignataro, "Extended ICMP to Support Multi-Part Messages", RFC 4884, April 2007.
- [25] Droms, R., Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, July 2003.
- [26] Moore, N., "Optimistic Duplicate Address Detection (DAD) for IPv6", RFC 4429, April 2006.
- [27] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, September 2007.
- [28] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, September 2007.
- [29] Bagnulo, M., "Address selection in multihomed environments", Work in Progress, October 2005.
- [30] Aura, T., Roe, M., and J. Arkko, "Security of Internet Location Management", Proceedings of the 18th Annual Computer Security Applications Conference, Las Vegas, Nevada, USA, December 2002.
- [31] Katz, D. and D. Ward, "Bidirectional Forwarding Detection", Work in Progress, February 2009.
- [32] Krishnan, S. and G. Daley, "Simple procedures for Detecting Network Attachment in IPv6", Work in Progress, February 2009.
- [33] Gont, F., "ICMP attacks against TCP", Work in Progress, October 2008.
- [34] Huitema, C., "Address selection in multihomed environments", Work in Progress, October 2004.
- [35] Bagnulo, M., "Default Locator-pair selection algorithm for the shim6 protocol", Work in Progress, October 2008.
- [36] Braden, R., "Requirements for Internet Hosts Communication Layers", STD 3, RFC 1122, October 1989.
- [37] Arkko, J., Kempf, J., Zill, B., and P. Nikander, "SEcure Neighbor Discovery (SEND)", RFC 3971, March 2005.
- [38] Stewart, R., "Stream Control Transmission Protocol", RFC 4960, September 2007.
- [39] Nikander, P., Henderson, T., Vogt, C., and J. Arkko, "End-Host Mobility and Multihoming with the Host Identity Protocol", RFC 5206, April 2008.

中华人民共和国
通信行业标准
基于 shim6 的 IPv6 站点多归属技术要求
YD/T 2415-2012

*

人民邮电出版社出版发行
北京市崇文区夕照寺街 14 号 A 座
邮政编码: 100061
宝隆元(北京)印刷技术有限公司印刷
版权所有 不得翻印

*

开本: 880 × 1230 1/16 2013 年 3 月第 1 版
印张: 5 2013 年 3 月北京第 1 次印刷
字数: 133 千字

15115 • 31

定价: 60 元

本书如有印装质量问题, 请与本社联系 电话: (010)67114922