网络工作组
RFC：3489
种类：标准路线

J. Rosenberg
J. Weinberger
dynamicsoft
C. Huitema
微软
R. Mahy
思科
2003 年 3 月

STUN—简单地用 UDP 穿过 NAT

本文状态

本文指定互联网社区的一个互联网标准路线协议，并请求进行改进的讨论和建议。该协议的标准化状态请参考最新版的“互联网官方协议标准”（STD 1）。可以无限制地分发本文。

版权声明

摘要

简单地用 UDP 穿过 NAT（STUN）是个轻量级的协议。它允许应用发现它们与公共互联网之间存在的 NAT 和防火墙及其类型。它还为应用提供判断 NAT 给它们分配的公共网际协议（IP）地址。STUN 可工作在许多现存 NAT 上，并且不需要它们的做任何特别的行为。结果就是，它允许广泛的各类的应用穿过现存的 NAT 设施。

目录表

1. 适用性声明

本协议不关心与 NAT 有关的所有问题。它不允许向内穿过 NAT 的 TCP 连接请求。它允许向内穿过 NAT 的 UDP 数据包，但只可穿过现有 NAT 类型中的一个子集。特别是，STUN 不允许向内穿过对称（Symmetric）NAT（定义见下面）的 UDP 数据包。该类 NAT 在大型企业中很常见。STUN 的发现过程基于 NAT 对待 UDP 的假定；对于新型已部署的 NAT 设备，该假定被证明是无效的。无法使用 STUN 来获取碰巧位于同个 NAT 之后的通迅端点的地址。当 STUN 服务器不在公共共享地址域内时，STUN 将无法工作。关于 STUN 限制的更完整的讨论见 14 节。

2. 简介

网络地址转换（NAT），提供很多好处的同时，也带来了许多缺点。这些缺点中的最麻烦的是它中断了许多现有的 IP 应用，并且使用部署新的应用非常困难。已经开发出描述如果创建“NAT 友好”的协议的指导方针[8]，但是许多协议不能简单地使用该指导方针来创建。这类协议的例子包括几乎所有的 P2P 协议，如多媒体通迅、文件共享和游戏。

为了与该问题作斗解，已经在 NAT 中嵌入了应用层网关（ALG）。ALG 执行特定协议所需要的穿过 NAT 的应用层功能。一般情况下，这会引入重写应用层消息来包含转换后的地址，而不是消息发送者在其中插入的。ALG 有严重的限制，包括伸缩性、可靠性和部署新应用的速度。要解决这些问题，正在开发中间盒通迅（MIDCOM）协议[9]。MIDCOM 允许应用实体，如终端客户或一些类型的网络服务器（如会话发起协议（SIP）代理[10]）来控制 NAT（或防火墙），以获得 NAT 的捆绑和打开或关闭这些洞。通过这种方法，NAT 和应用可再一次分开，剔除需要在 NAT 中嵌入 ALG 的需求，并解决利用当前结构的限制。

非常不幸，MIDCOM 需要对现存的 NAT 和防火墙进行升级，应用组件也一样。对这些 NAT 和防火墙产品进行完全地升级将花去很长的时间，可能是几年。这部分由于 NAT 和防火墙的部署者与部署并使用应用的人是不同的。结果就是，许多情况下，升级这些设备的动机将非常小。作为例子，考虑一个航空港的互联网沙龙提供通过 NAT 的接入服务。一个用户连接到该 NAT 网络可能希望使用 P2P 服务，但是不能，因为 NAT 不支持。既然沙龙的管理员不提供该服务，它们也没有动机来升级它们的 NAT 设备来支持该服务，而不管是 ALG 还是 MIDCOM。

MIDCOM 协议的另一个问题是，它需要控制中间盒的代理知道这些中间盒的标识，并且与这些允许控制的有关系。在许多配置下，这将是不可能的。例如，许多线路接入提供者在他们的整个接入网络前使用 NAT。这个 NAT 可能会附加上由终端用户购买和操作的住宅 NAT。终端用户可能与线路接入网络中的 NAT 没有控制关系，且可能并不知道他们的存在。

许多现存的私有协议，如在线游戏的协议（如在 RFC 3027[11]中描述的游戏）和 VoIP，已经开发现这样的诡计。它允许他们穿过 NAT 进行操作而不需要改变这些 NAT。可尝试从本文获得一部分这些主意。本文将它们编辑成一个可互操作的协议，并可满足许多应用的需要。

这里描述的协议，用 UDP 简单穿过 NAT（STUN），允许 NAT 后的实体首先发现 NAT 的存在和其类型，接着知道 NAT 分配的地址捆绑。STUN 不需要修改 NAT，并可工作在应用实体和公共互联网前后间有任意数量的 NAT 的情况下。

3. 术语

在本文中，关键字“MUST”、“MUST NOT”、“REQUIRED”，“SHALL”，“SHALL NOT”，“SHOULD”，“SHOULD NOT”，“RECOMMENDED”，“MAY”，和“OPTIONAL”应按 BCP 14，RFC 2119[1]中描述的一样进行解释，并表明的顺从 STUN 的应用的所需的等级。

4. 定义

STUN 客户

STUN 客户（只引用为客户）是产生 STUN 请求的实体。STUN 客户能够在终端系统上执行，如用户的 PC 机，或能够在网络器件中运行，如会议服务器。

STUN 服务器

STUN 服务器（还是只引用为服务器）是接收 STUN 请求并发送 STUN 响应的实体。STUN 服务器通常连接到公共互联网。

5. NAT 种类

假设用户熟悉 NAT。观察发现 NAT 有各种对待 UDP 的实现。观察到的实现中的 4 程对待方式是：

完全锥形

完全锥形 NAT 将从同个内部 IP 地址和端口号发出的所有请求映射为相同的外部 IP 地址和端口号。此外，外部主机能够向内部主机发送数据包，即通过向映射过的外部地址发送数据包。

限制锥形

限制锥形 NAT 将从同个内部 IP 地址和端口号发出的所有请求映射为相同的外部 IP 地址和端口号。与完全锥形 NAT 不同的是，外部主机（IP 地址为 X）只能够向先前已经向 IP 地址 X 发送过数据包的内部主机发送数据包。

端口限制锥形

端口限制锥形 NAT 与限制锥形 NAT 一样，但是限制包括端口号。特别是，有源 IP 地址 X 和端口号 P 的外部主机只能够向先前已经向 IP 地址 X 和端口 P 发送过数据包的内部主机发送数据包。

对称

对称 NAT 将从相同的内部 IP 地址和端口号，以及指定的目的 IP 地址和端口号，发出的所有请求映射为相同的外部 IP 地址和端口号。如果同个主机用相同的源地址和端口给不同的目的地发送数据包，将使用不同的映射。此外，只有收到数据的外部主机才可以反过来向内部主机发送 UDP 数据包。

在许多情况下，决定 NAT 的类型是重要的。取决于应用相做什么，需要考虑特定的行为。

6. 操作概貌

本节只作描述。标准行为在 8 和 9 节中描述。

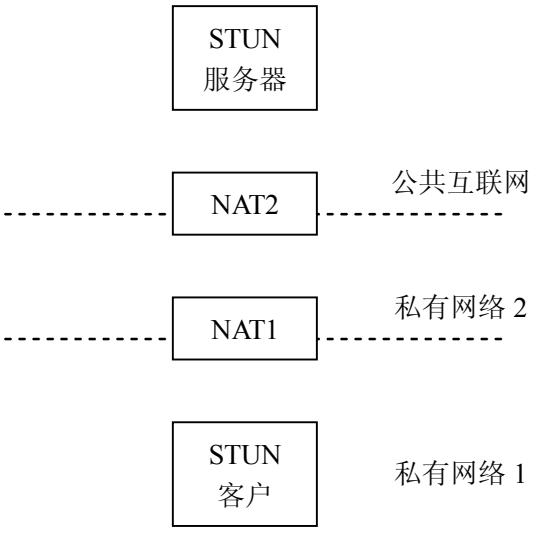

图 1：STUN 配置

图 1 显示了典型的 STUN 配置。STUN 客户连接到私有网络 1。该网络又通过 NAT1 连接到私我网络 2。私有网络 2 又通过 NAT2 连接到公共互联网。STUN 服务器位于公共互联网上。

STUN 是个简单的客户—服务器协议。客户向服务器发送请求，接着服务器返回响应。有两种类型的请求—捆绑请求，通过 UDP 发送，和共享私密请求，通过 TLS TCP[2]。共享私密请求叫服务器返回临时的用户名和密码。该用户名和密码用于后序的捆绑请求和捆绑响

应，其目的是认证和检验消息的完整性。

捆绑请求用来获得 NAT 分配的捆绑。客户发送捆绑请求给服务器，通过 UDP。服务器检验该请求的源 IP 地址和端口号，然后将它们拷贝到回送给该客户的响应中。该请求中还有一些参数允许客户请求响应发送到其它地方，或者请求服务器从不同的地址和端口发送该响应。提供了属性来保证消息的完整性并认证。

该诡计就是使用 STUN 来发现 NAT 的存在，并获知和使用它们分配的捆绑。

STUN 客户一般是嵌入到需要获得可用于接收数据的公共 IP 地址和端口号的应用程序中。例如，可能需要获得 IP 地址和端口号来接收实时传输协议（RTP）[12]流量。当应用程序开始时，应用程序中的 STUN 客户发送 STUN 共享私密请求给他的服务器，获得用户名和密码，接着向它发送捆绑请求。STUN 服务器能够从 DNS SRV 记录[3]发现，并且通常它假定客户被配置了该域以发现 STUN 服务器。通常，这将会应用程序正使用的服务器的提供者的域（这样的提供者被驱动部署 STUN 服务器，以允许他的客户可穿过 NAT 来使用他的应用程序）。当然，客户能够通过其它途径来判断 STUN 的地址或域名。STUN 服务器甚至可以嵌入到终端系统中。

STUN 捆绑请求用于发现 NAT 的存在，并发现 NAT 生成的公共 IP 地址和端口号映射。捆绑请求通过 UDP 发送给 STUN 服务器。当捆绑请求到达 STUN 服务器，它可能已经在 STUN 客户和 STUN 服务器间通过了一个或更多个 NAT。其结果是，服务器收到的请求的源地址将会是由离服务器最近的 NAT 创建的映射过的地址。STUN 服务器拷贝该源 IP 地址和端口号到 STUN 的捆绑响应中，并回送给 STUN 请求的源 IP 地址和端口号。对于上面的所有 NAT 类型，该响应将会到达 STUN 客户。

当 STUN 客户收到 STUN 捆绑响应，它将数据包中的 IP 地址和端口号与请求发送时他绑定的本地 IP 地址和端口号相比较。如果这些不匹配，则 STUN 客户位于一个或更多个 NAT 之后。在完全锥形 NAT 的情况下，STUN 响应体内的 IP 地址和端口号是公共的，并且能够被公共互联网上的任何主机用来向发送该 STUN 请求的应用程序发送数据包。应用程序只需监听从 STUN 请求发送地来的 IP 地址和端口号。任何公共互联网上的主机发送到 STUN 获知的公共地址和端口号的数据包将会被该应用程序收到。

当希，主机可能不在完全锥形 NAT 之后。其实，他也不知道他位于其后的 NAT 的类型。要作出判断，客户使用附加的 STUN 捆绑请求。确切的过程是灵活的，但一般工作如下。客户将发送第二个 STUN 捆绑请求，这时到一个不同的 IP 地址，但是从相同的源 IP 地址和端口。如果响应中的 IP 地址和端口号与第一次的响应不同，客户就知道他位于对称 NAT 之后。发判断是否在完全锥形 NAT 之后，客户可发送设置标志的 STUN 捆绑请示。该标志使 STUN 服务器从与请求接收到的不同的 IP 地址和端口号发送响应。换句话说，如果客户使用为 X/Y 的源 IP 地址/端口号向 IP 地址/端口号 A/B 发送捆绑请求，STUN 服务器将使用源 IP 地址/端口号 C/D 发送捆绑响应给 X/Y。如果客户收到该响应，他就知道他在完全锥形 NAT 之后。

STUN 还允许客户使服务器从与收到请求相同的 IP 地址，但用不同的端口号发送捆绑响应。这可用来检测客户是否在端口限制锥形 NAT 或只在限制锥形 NAT 之后。

要指出的是，图 1 中的配置不是唯一许可的配置。STUN 服务器可在任何地方，包括在另一个客户中。唯一的要求是，STUN 服务器可被客户访问，并且如果客户要尝试获取公共可路由的地址，则服务器应在公共互联网上。

7. 消息概貌

STUN 消息是用 TLV（类型—长度—值）编码的，使用大端字节序（网络字节序）。所有的 STUN 消息开始于 STUN 头，接着是 STUN 负载。负载是一系列的 STUN 属性，它的集合由消息的类型决定。STUN 头包含 STUN 消息类型，事务 ID 和长度。消息类型可以是捆绑请求，捆绑响应，捆绑错误响应，共享私密请求，共享私密响应，或共享私密错误响应。事务 ID 用来关联请求和响应。长度指示 STUN 负载的总长度，不包括头。允许 STUN 运行在 TCP 上。共享私密请求经常通过 TCP 发送（其实，是使用在 TCP 上的 TLS）。

定义了几个 STUN 属性。第一个是 MAPPED-ADDRESS 属性，这是 IP 地址和端口号。它经常放在捆绑响应中，并且表示服务器看见的捆绑请求的源 IP 地址和端口号。还有一个 RESPONSE-ADDRESS 属性，可以出现在捆绑请求中，且表示捆绑响应发送的目的地。它是可选的，如果不存在，捆绑响应应发送给捆绑请求的源 IP 地址和端口号。

第 3 个是 CHANGE-REQUEST 属性，且它包含两个标志来控制用来发送响应的 IP 地址和端口号。这些标志称为“改变 IP”和“改变端口号”标志。CHANGE-REQUEST 属性只允许在捆绑请求中。“改变 IP”和“改变端口号”标志对于判断客户是否在限制锥形 NAT 或限制端口锥形 NAT 之后是有用的。它们命令服务器从不同的源 IP 地址和端口号发送捆绑响应。CHANGE-REQUEST 属性在捆绑请求中是可选的。

第 4 个属性是 CHANGED-ADDRESS 属性。它存在于捆绑响应中。它表明，如果客户请求了“改变 IP”和“改变端口号”行为，客户应该使用的源 IP 地址和端口号。

第 5 个属性是 SOURCE-ADDRESS 属性。它只存在于捆绑响应中。它表明响应发送的源 IP 地址和端口号。用于检测两次 NAT 的配置。

第 6 个属性是 USERNAME 属性。它存在于共享私密响应中。它为客户提供一个临时的用户名和密码（在 PASSWORD 属性中编码）。USERNAME 还存在于捆绑请求中，用作共享私密的索引，并用来对捆绑请求进行完整性保护。第 7 个属性，PASSWORD，只存在于共享私密响应消息中。第 8 个属性是 MESSAGE-INTEGRITY 属性。它包含检查捆绑请求和捆绑响应的完整性的消息。

第 9 个属性是 ERROR-CODE 属性。它存在于捆绑错误响应和共享私密错误响应中。它指示所发生的错误。第 10 个属性是 UNKNOWN-ATTRIBUTES 属性，它存在于捆绑错误响应或共享私密错误响应中。它表明请求的命令属性未知。第 11 个属性是 REFLECTED-FROM 属性，存在于捆绑响应中。表明捆绑请求发送者的 IP 地址和端口号，用于阻止并跟踪某类拒绝服务器攻击。

8. 服务器行为

服务器行为取决于请求是捆绑请求，还是共享私密请求。

8.1 捆绑请求

STUN 服务器 MUST 准备接好收捆绑请求，在 4 个地址/端口号组合上，（A1，P1）、（A2，P1）、（A1，P2）和（A2，P2）。（A1，P1）代表主要的地址和端口号，并且它们是通过如下的客户发现过程获得的。一般 P1 是端口号 3478，缺省的 STUN 端口。A2 和 P2 是任意的。用 CHANGE-ADDRESS 属性，服务器公告 A2 和 P2，如下所述。

RECOMMENDED，服务器检查捆绑请求的 MESSAGE-INTEGRITY 属性。如果不存在，并且服务器需要检查请求的完整性，它生成捆绑错误响应，并用响应号 401 设置 ERROR-CODE 属性。如果 MESSAGE-INTEGRITY 属性存在，服务器计算请求的在 11.2.8 节中所述的 HMAC，键的使用取决于共享私密机制。如果使用 STUN 共享私密请求，键 MUST 与请求中的 USERNAME 属性相关。如果 USERNAME 属性不存在，服务器 MUST 生成捆绑错误响应。捆绑错误响应 MUST 包括响应号为 432 的 ERROR-CODE 属性。如果 USERNAME 存在，但服务器不认识该 USERNAME 的共享私密（例如，因为它超时了），服务器 MUST 生成捆绑错误响应。捆绑错误响应 MUST 包括响应号为 430 的 ERROR-CODE 属性。如果服务器知道该共享私密，但所计算的 HMAC 与请求的不同，服务器 MUST 生成捆包含响应号为 431 的 ERROR-CODE 属性的绑错误响应。捆绑错误响应发送给请求到达的 IP 地址和端口号，并且从捆绑请求发送的目的地的 IP 地址和端口号发送。

假设消息完整性检查通过了，过程继续。服务器 MUST 检查请求中的任何属性的值，看是否是它不认识的小于或等于 0x7fff。如果遇到了，服务器 MUST 生成捆绑错误响应，并且它 MUST 包括响应号为 420 的 ERROR-CODE 属性。

该响应 MUST 包含 UNKNOWN-ATTRIBUTES 属性，并列出不理解的值小于或等于 0x7fff 的属性。捆绑错误响应发送给捆绑请求到达的 IP 地址和端口号，且从捆绑请求发送的目的地的 IP 地址和端口号发送。

假设请求依然正确，服务器 MUST 生成单个捆绑响应。捆绑响应 MUST 包含与捆绑请求中包含的相同的事务 ID。消息头中的长度 MUST 包含消息的总的字节长度，除了头。捆绑响应 MUST 有“捆绑响应”的消息类型。

服务器 MUST 在捆绑响应中加入 MAPPED-ADDRESS 属性。该属性的 IP 地址成员 MUST 设置为观察到的捆绑请求的源 IP 地址。该域的端口号成员 MUST 设置为观察到的捆绑请求的源端口号。

如果捆绑请求中缺少 RESPONSE-ADDRESS 属性，则捆绑响应的目的地址和端口号 MUST 与捆绑请求的源地址和端口号相同。否则，捆绑响应的目的地址和端口号 MUST 具有 RESPONSE-ADDRESS 属性中的 IP 地址和端口号的值。

捆绑响应的源地址和端口号取决于 CHANGE-REQUEST 属性的值，且和捆绑请求收到的地址和端口号相关。在表 1 中总结。

用 Da 表示捆绑请求的目的 IP 地址（将为 A1 或 A2），Dp 表示捆绑请求的目的端口号（将为 P1 或 P2）。用 Ca 表示另一个地址，因此如果 Da 是 A1，则 Da 为 A2。如果 Da 是 A2，则 Ca 为 A1。相似地，用 Cp 表示另一个端口号，因此如果 Dp 是 P1，则 Cp 是 P2。如果 Dp 是 P2，则 Cp 为 P1。如果设置了捆绑请求中的 CHANGE-REQUEST 属性的“改变端口号”标志，且“改变 IP”标示没有设置，则捆绑响应的源 IP 地址 MUST 是 Da，且捆绑响应的源端口号 MUST 是 Cp。如果捆绑请求中的“改变 IP”标志设置了，且“改变端口号”标志没有设置，则捆绑响应的源 IP 地址 MUST 是 Ca，且捆绑响应的源端口号 MUST 是 Dp。当两个标志都设置时，捆绑响应的源 IP 地址 MUST 是 Ca，且捆绑响应的源端口号 MUST 是 Cp。如果两个标志都没有设置，或者整个 CHANGE-REQUEST 都没有，则捆绑响应的源 IP 地址 MUST 是 Da，且捆绑响应的源端口号 MUST 是 Dp。

标志	源地址	源端口号	CHANGED-ADDRESS
无	Da	Dp	Ca:Cp
改变 IP	Ca	Dp	Ca:Cp
改变端口号	Da	Cp	Ca:Cp
改变 IP 且改变端口号	Ca	Cp	Ca:Cp

表 1：标志对数据包源和 CHANGED-ADDRESS 的影响

服务器 MUST 在捆绑响应内加入 SOURCE-ADDRESS 属性，包含用于发送捆绑响应的源地址和端口号。

服务器 MUST 在捆绑响应中加入 CHANGED-ADDRESS 属性。包含源 IP 地址和端口号，将用于客户在捆绑请求中设置“改变”和“改变端口号”标志的情况下。如表 1 中所总结的，它们分别是 Ca 和 Cp，而不管 CHANGE-REQUEST 标志的值如何。

如果捆绑请求包含 USERNAME 和 MESSAGE-INTEGRITY 两个属性，则服务器 MUST 在捆绑响应内加入 MESSAGE-INTEGRITY 属性。该属性包含响应的 HMAC[13]，如 11.2.8 节中所述。键的使用取决于共享私密机制。如果使用了 STUN 的共享私密请求，则键 MUST 与出现在捆绑请求中的 USERNAME 相关联。

如果捆绑请求包含 RESPONSE-ADDRESS 属性，服务器 MUST 在响应中加入 REFLECTED-FROM 属性。如果捆绑请求使用从共享私密请求获得的用户名进行认证，则 REFLECTED-FROM 属性 MUST 包含共享私密请求到达的源 IP 地址和端口号。如果请求中的用户名不是使用共享私密分配的，则 REFLECTED-FROM 属性 MUST 包含获得该用户名的实体的源地址和端口号，最好能够验证用于分配用户名的机制。如果请求中没有用户名，且服务器愿意处理该请求，则 REFLECTED-FROM 属性 SHOULD 包含请求发出的源 IP 地址和端口号。

服务器 SHOULD NOT 重传响应。可靠性通过客户周期性地重发请求来达到。每个请求都会触发服务器进行响应。

8.2 共享私密请求

共享私密请求经常从 TLS 连接上收到。当服务器收到客户要求建立 TLS 连接的请求时，它 MUST 用 TLS 处理，并 SHOULD 对站点用证书进行认证。SHOULD 应该使用 TLS 的密码套 TLS_RSA_WITH_AES_128_CBC_SHA [4]。客户的 TLS 认证 MUST NOT 完成，因为服务器不为客户分配任何资源，且计算负担可能会成为攻击的源泉。

如果服务器收到共享私密请求，它 MUST 验证从 TLS 连接上到达的该请求。如果不是通过 TLS 收到的请求，它 MUST 生成共享私密错误响应，且它 MUST 包括响应号为 433 的 ERROR-CODE 属性。错误响应的目标取决于收到请求的传输方式。如果通过 TCP 收到共享私密请求，则共享私密错误响应通过收到请求的相同连接发送。如果通过 UDP 收到共享私密请求，则共享私密错误响应发送回请求送出的源 IP 地址和端口号。

服务器 MUST 检查请求中的任何属性，保证没有其不理解的小于或等于 0x7fff 的值。如果有任何错误发生，服务器 MUST 生成共享私密错误响应，且它 MUST 包括响应号为 420 的 ERROR-CODE 属性。响应 MUST 包含 UNKNOWN-ATTRIBUTES 属性，列出它不理解的值小于或等于 0x7fff 的属性。共享私密错误响应通过 TLS 连接发送。

所有的共享私密错误响应 MUS 包含与共享私密请求中相同的事务 ID。该消息头中的长度域 MUST 包含消息的总的字节数，除了头。共享私密错误响应 MUST 有“共享私密错误响应”（0x0112）的消息类型。

假设请求构造成确，服务器创建共享私密响应。共享私密响应 MUST 包含与共享私密请求中相的事务 ID。消息头中的长度域 MUST 包含消息的总的字节长度，除了头。共享私密响应 MUST 有“共享私密响应”的消息类型。共享私密响应 MUST 包含 USERNAME 属性和 PASSWORD 属性。USERNAME 属性作为密码的索引，基中包含了 PASSWORD 属性。服务器可以选择使用任何机制来生成用户名。然而，用户名 MUST 必须至少在 10 分钟内有效。认证意味着服务器能够计算出该用户名的密码。每个用户名 MUST 有一个密码。换句话说，服务器不能在 10 分钟后为相同的用户名分配不同的密码。服务器 MUST 分每个不同的共享私密请求的分发不同的用户名。在这种情况下，不同的暗示不同的事务 ID。RECOMMENDED，服务器声明 10 分钟后用户名无效。它 MUST 在 30 分钟后使用用户名无效。PASSWORD 包含与该用户名绑定的密码。密码 MUST 有至少 128 位。服务器为两个不同的用户名分配相同的密码的可能性 MUST 非常非常小，前且密码 MUST 是不能猜测的。换句话说，它们 MUST 是用户名的密码学的随机函数。

这些要求依然能够使用无状态服务器来满足，通过智能地计算 USERNAME 和 PASSWORD。一个方法是构造 USERNAME 为：

USERNAME = <prefix,rounded-time,clientIP,hmac>

“prefix”是一些随机字符串（每个共享私密请求不同），“rounded-time”当前的时间按 20 分钟取模，“clientIP”是共享私密请求送出的源 IP 地址，“hmac”是“prefix”、“rounded-time”和“clientIP”的使用服务器私有密匙的 HMAC[13]。

然后，密码的计算如下：

password = <hmac(USERNAME,anotherprivatekey)>

用该结构，将出现在捆绑请求中的用户名自己就包含共享私密请求送出的源 IP 地址。这就允许服务器满足在 8.1 节中指出的要求，构造 REFLECTED-FROM 属性。服务器可以验证用户名没有使用用户中的“hmac”被篡改。

共享私密响应通过与收到请求的相同的 TLS 连接发送。服务器 SHOULD 保持连接打开状态，并让客户送闭它。

9. 客户行为

客户的行为是非常明确的。他的任何是发现 STUN 服务器，获得共享私密，形成捆绑请示，处理请求的可靠性，并处理捆绑响应。

9.1 发现

一般情况下，客户会配置 STUN 服务器提供者的域名。该域名被解析为 IP 地址和使用在 RFC 2782[3]中指定的 SRV 过程的端口号。

特别地，服务器名是“stun”。协议是发送捆绑请求的“udp”，或发送共享私密请求的“tcp”。RFC 2782 过程接收判断联系的服务器。RFC 2782 清楚地说明 SRV 记录集如何排序的细节，然后尝试。然而，它只说明客户应该“尝试连接到（协议，地址，服务器）”，而没有给出任何关于失败后所发生的事件的细节。这些细节在这时为 STUN 描述。

对于 STUN 请求，如果出现某种类型的传输失败（通常因为 UDP 的故障 ICMP 错误或 TCP 的连接失败），则失败发生。如果传输由于超时失败，失败也会发生。在第一将发送请求的 9.5 秒后，这就会发生，不管是共享私密请求，还是捆绑请求。关于捆绑请求的事务超时的细节见 9.3 节。如果失败发生，客户 SHOULD 创建新的请求，它与前面的相同，但有不同的事务 ID 和 MESSAGE-INTEGRITY 属性（HMAC 也会因为事务 ID 的改变而改变）。该请求发送给 RFC 2782 指定的表中的下个元素。

STUN 请求的缺省端口号 3478，用于 TCP 和 UDP。管理员 SHOULD 在他们的 SRV 记录中使用这些端口，但 MAY 使用其它的。

如果没有发现 SRV 记录，客户执行域名的记录查询。结果会是 IP 地址表，每项都能用缺省的端口号进行联系。

这可能会要求防火墙管理打开 STUN 端口，以使企业中的主机能够访问新的应用。他们是否会这样做是个很好的问题。

9.2 获得共享私密

如同 12 节中讨论的一样，可能会有几种对 STUN 系统进行的攻击。许多可通过请求和

响应的完整性检查来阻止。要提供完整性检查，STUN 在客户与服务器间使用共享私密，在捆绑请求和捆绑响应中使用 HMAC 的密匙资源。STUN 允许通过任何方式来获得共享私密（例如，Kerberos[14]）。然而，它 MUST 至少有随机性的 128 位。为了保证互操作性，该规范在基于 TLS 的机制中描述。本节中描述的该机制 MUST 被客户和服务器都实现。

首先，客户获得他将与之建立 TCP 连接的 IP 地址和端口号。这可通过 9.1 节中的发现过程来完成。客户打开到该地址和端口的连接，紧跟着开始 TLS 协商[2]。客户 MUST 验证服务器的标识。要这样做，它应该按 RFC 2818[5]的 3.1 节中定义的标识过程来做。该过程假设客户正参照 URI。关于使用该规范的目的，客户将 9.1 节中使用的域名或 IP 地址作为该主机的参照的 URI 的一部份。

一旦打开了连接，客户就发送共享私密请求。该请求没有属性，只有头。头中的事务 IDMUST 满足捆绑请求中的事务 ID 的需求要点，在下面的 9.3 节中描述。服务器生成响应，可能是共享私密响应或共享私密错误响应。

如果响应是共享私密错误响应，服务器检查 ERROR-CODE 属性中的响应号。这些响应号的解释与 9.4 节中的捆绑错误响应处理相同。

如果客户收到共享私密响应，其属性类型大于 0x7fff，则 MUST 乎略该属性。如果客户收以共享私密响应，其属性类型小于或等于 0x7fff，则乎略该响应。

如果响应是共享私密响应，它将包含短效的用户名和密码，并分别编码在 USERNAME 和 PASSWORD 中。

客户 MAY 在该连接上生成多个共享私密请求，且他 MAY 在收到先前的共享私密请求的共享私密响应前发出。客户 SHOULD 在完成获得用户名和密码后尽快关闭该连接。

9.3 节描述这些密码如何用于提供捆绑请求的完整性保护，8.1 节描述如何在捆绑响应中使用它。

9.3 形成捆绑请求

客户形成的捆绑请求要按照 11 节中定义的语法规则。任何两个请求都不应该是位等，且不应该从相同的 IP 地址和端口发送给相同的服务器，MUST 携带不同的事务 ID。事务 ID MUST 唯一且随机分布于 0 到 2^{128-1} 之间。需要大的范围，因为事务 ID 作为随机的形式，帮助避免重新从服务器收到以前处理过的响应。请求的消息类型 MUST 是“捆绑请求”。

RESPONSE-ADDRESS 属性在捆绑请求中是可选的。如果客户希望响应发送到不同的 IP 地址和端口号，而不是发送请求的地方，则使用它。这用于判断客户是否位于防火墙之后和对于有分离控制和数据成员的应用是非常有用的。细节见 10.3 节。CHANGE-REQUEST 属性也是可选的。是否存在取决于应用尝试完成的事。用法的一些例子见 10 节。

客户 SHOULD 在捆绑请求中加入 MESSAGE-INTEGRITY 和 USERNAME 属性。MESSAGE-INTEGRITY 属性包含 HMAC[13]。用户名的值和 MESSAGE-INTEGRITY 属性

中使用中的键取决于共享私密机制。如果 STUN 使用共享私密请求，USERNAME 必然是从最近 9 分钟内的共享私密响应中获取的有效的用户名。HMAC 共享私密是从相同的共享私密响应中获取的 PASSWORD 属性的值。

一旦形成，客户发送捆绑请求。可靠性通过客户重传来完成。客户 SHOULD 开始用 100ms 的间隔重传，每次重传间隔加倍，直到 1.6s。间隔 1.6s 的重传继续，直到收到响应或总共已经发送了 9 次请求。如果发送请求的 1.6s 内没有收到响应，客户 SHOULD 认为传输已经失败。换句话说，请求的发送时间为 0ms，100ms，300ms，700ms，1500ms，3100ms，4700ms，6300ms 和 7900ms。在 9500ms，如果没有收到响应，客户认为传输已经失败。

9.4 处理捆绑响应

响应可以是捆绑响应或捆绑错误响应。捆绑错误响应常常在请求发送的源地址和端口号收到。捆绑响应将会在请求中的 RESPONSE-ADDRESS 属性的地址和端口号收到。如果没有，则捆绑响应将在请求的发送的源地址和端口号收到。

如果响应是捆绑错误响应，客户应检查响应中的 ERROR-CODE 属性的响应号。对于 400 响应号，客户 SHOULD 向用户显示原因语句。对于 420 响应号，客户 SHOULD 重新尝试请求，这次省去响应的 UNKNOWN-ATTRIBUTES 属性中列出的任何属性。对于 430 响应号，客户 SHOULD 获取新的共享私密，并用新的事务重新尝试捆绑请求。对于 401 和 430 响应号，如果客户如同错误指示的一样缺少 USERNAME 升 MESSAGE-INTEGRITY 属性，他 SHOULD 用这些属性再次尝试。对于 431 响应号，客户 SHOULD 警告用户，且 MAY 在获取新的用户名和密码后再次尝试。对于 500 响应号，客户 MAY 等待数秒，然后重新尝试该请求。对于 600 响应号，客户 MUST NOT 重新尝试该请示，且 SHOULD 向用户显示原因语句。400 至 499 之间的未知属性按 400 处理，500 至 599 之间的未知属性按 500 处理，600 至 699 之间的未知属性按 600 处理。任何 100 和 399 之间的响应 MUST 使请求重传中止，但其它则乎略。

如果客户收到响应的属性类型大于 0x7fff，则属性 MUST 乎略。如果客户收到的响应的属性类型小于或等于 0x7fff，则请求重传 MUST 停止，但整个响应也就乎略。

如果响应是捆绑响应，客户 SHOULD 检查响应的 MESSAGE-INTEGRITY 属性。如果不存在，则客户应在请求中加入 MESSAGE-INTEGIRTY 属性，他 MUST 放弃该响应。如果存在，客户计算响应的 HMAC，如 11.2.8 节所述。使用的键取决于共享私密机制。如果使用 STUN 共享私密请求，键 MUST 与请求中用于计算 MESSAGE-INTEGRITY 属性的相同。如果计算出的 HMAC 与响应中的不同，则客户 MUST 放弃该响应，且 SHOULD 警告用户可能受到了攻击。如果计算出的 HMAC 与响应中的匹配，则过程继续。

收到捆绑请求的响应（不管是捆绑错误响应或捆绑响应），将中止该请求的重传。然而，客户 MUST 在第一次响应后继续监听捆绑请求的响应 10 秒钟。如果在些期间他收到任何消息类型不同的响应（例如，捆绑响应和捆绑错误响应）或不同的 MAPPED-ADDRESS 属性，这表明可能受到攻击。客户 MUST NOT 使用收到的任何这些响应中的 MAPPED-ADDRESS（不管是第一次还是以后的），且 SHOULD 警告用户。

此外，如果客户收到的捆绑响应次数超过他发送的捆绑请求数的两倍，他 MUST NOT 使用任何这些响应的 MAPPED-ADDRESS，且 SHOULD 警告用户可能有攻击。

如果捆绑响应经过认证，且由于可能的攻击而没的放弃 MAPPED-ADDRESS，则客户 MAY 使用该 MAPPED-ADDRESS 和 SOURCE-ADDRESS 属性。

10. 用例

8 和 9 节中的规则准确地描述客户和服务器间如何交互发送请求和获得响应。然而，它们没有规定如何使用 STUN 协议来完成有用的任务。这是由客户来处理的。这里，我们提供一些应用 STUN 的有用的场景。

10.1 发现过程

在该声场景中，用户运行多媒体应用程序，需要决定应用下面的哪个场景：

- 在公开的互联网上
- 阻止 UDP 的防火墙
- 允许发出 UDP 和发回的给请求源的响应的防火墙（如同对称型 NAT，但没有传输。我们称这是对称型 UDP 防火墙）。
- 完全锥型 NAT
- 对称型 NAT
- 限制锥型或端口限制锥型 NAT

应用这 6 种场景中的哪个，可能通过图 2 中描述的流程图判断出。该图只引用了捆绑请求的序列；当然，共享私密请求将需要用来认证该序列中使用的每个捆绑请求。

该流程使用 3 个测试。在测试 I 中，客户发送 STUN 捆绑请求给服务器，没有设置 CHANGE-REQUEST 属性中的任何标志，且没有 RESPONSE-ADDRESS 属性。这引起服务器回发响应到发出请求的地址和端口。在测试 II 中，客户发送捆绑请求，并设置了 CHANGE-REQUEST 属性集中的“改变 IP”和“改变端口”标志。在测试 III 中，客户发送捆绑响应，只设置“改变端口”标志。

客户首先从测试 I 开始。如果该测试没有响应，客户就知道他不能使用 UDP 连接。如果测试产生响应，则客户检查 MAPPED-ADDRESS 属性。如果其中的地址和端口与用来发送请求的套接口的本地 IP 地址和端口相同，则客户知道他没有通过 NAT。然后执行测试 II。

如果收到响应，则客户知道他与互联网之间可互访（或者，至少，他在表现如同完全锥型 NAT 且没有进行转换的防火墙后）。如果没有响应，则客户知道他在对称型 UDP 防火墙

之后。

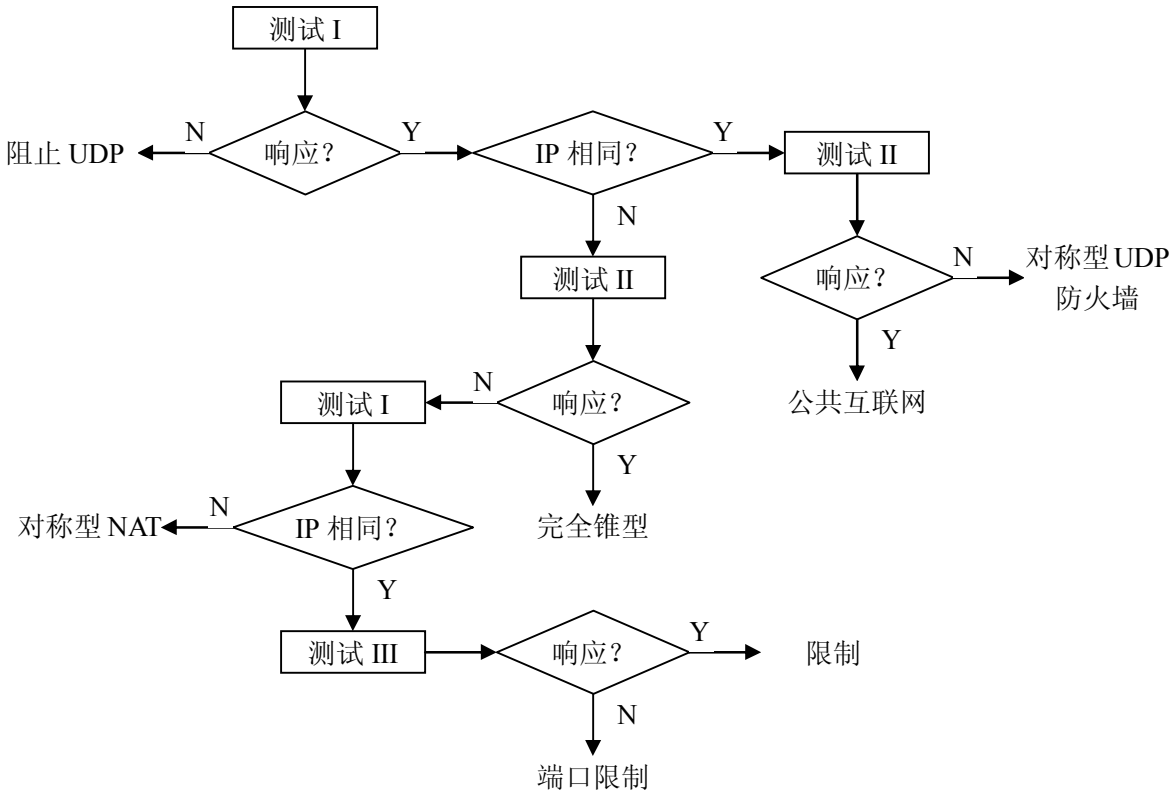

图 2：类型发现过程的流程

在套接口的 IP 地址和端口与测试 I 响应的 MAPPED-ADDRESS 属性中的不同的情况下，客户就知道他在 NAT 后。他执行测试 II。如果收到响应，则客户知道他在完全锥型 NAT 后。如果没有响应，他再次执行测试 I，但这次，使测试 I 响应的 CHANGED-ADDRESS 属性中的地址和端口。如果 MAPPED-ADDRESS 中返回的 IP 地址和端口与第一次测试 I 中的不同，客户就知道他在对称型 NAT 后。如果地址和端口相同，则客户要么在限制 NAT 后，要么在端口限制 NAT 后。要判断出到底是哪种，客户进行测试 III。如果收到响应，则他在限制 NAT 后，如果没有响应，则他在端口限制 NAT 后。

该过程产生关于客户应用程序的操作条件的实际信息。在客户与互联网间存在多级 NAT 的情况下，发现的类型将是客户与互联网间限制最多的 NAT 的类型。NAT 的类型接照限制排序从高到低是对称型，端口限制锥型，限制锥型和完全锥型。

通常情况下，客户会周期性地重做该发现过程，以检测改变，或搜索易变的结果。要注意的重要事情是，当发现过程重做时，一般不应该使用前面发现过程中使用的相同的本地地址和端口。如果重用相同的本地地址和端口，且前面测试的捆绑依然存在，则测试结果就会无效。为后序的测试使用不同的本地地址和端口可解决该问题。另一个可选的方案是等待足够长的时间来确保旧的捆绑已经过期（半个小时应该足够了）。

10.2 捆绑生命期发现

STUN 还可用于发现 NAT 创建的捆绑的生命期。在许多情况下，客户将需要刷新捆绑，要么通过新的 STUN 请求，要么是应用包，以使应用继续使用该捆绑。通过发现捆绑生命期，客户能够判断出需要刷新的频率。

要判断捆绑的生命期，客户首先要从特定的套接口 X 向服务器发送捆绑请求。这会使 NAT 建立捆绑。服务器的响应包含 MAPPED-ADDRESS 属性，提供 NAT 上的公共地址和端口。分别称为 Pa 和 Pp。客户接着启动长为 T 秒的定时器。如果时间到了，客户向服务器发送另一个捆绑请求，使用相同的目的地址和端口，但从不同的套接口 Y。该请求包含 RESPONSE-ADDRESS 地址属性，发送给（Pa，Pp）。这会使 NAT 创建新的捆绑，并引起 STUN 服务器发送与还存在的旧捆绑相匹配的捆绑响应。如果客户在套接口 X 上收到捆绑响应，他就知道捆绑还没有过期。如果客户在套接口 Y（如果旧的捆绑过期就可能，且 NAT 为新的捆绑分配相同的公共地址和端口）上收到捆绑响应，或都跟本没收到响应，他就知道捆绑已经过期。

客户能够通过 T 秒来进行二叉查找算法来发现捆绑生命期的值，即最终获得的超过 T 的任何时间就没有响应，但小于 T 的任何时间就会有响应的值。

该发现过程需要相当长的时间，且有时候它通常是在设备启动后在后台运行的。

运行该过程客户每次能够得到非固定的结果是可能的。使如，如果 NAT 要重启，或由于某种原因需要复位，则该过程发现的生命期可能比实际的更短。由于这个原因，鼓励实现多个运行该测试，且准备好得到非固定的结果。

10.3 获取捆绑

再次考虑 VoIP 电话的情况。当它启动时使上面的发现过程来发现它所处的环境。现在，它想建立一个呼叫。作为该发现过程的一部份，它知道它在完全锥型 NAT 后。

更进一步考虑该电话由两个独立的逻辑组件组成，一个组件处理信号，一个媒体组件处理音频、视频和 RTP[12]。都在同个 NAT 后。因为控制和媒体的分割，我们希望最小化它们之间的通迅。实际上，它们甚至可能没有运行在同个主机上。

为了建立语音呼叫，电话需要获得 IP 地址和端口，将它放到呼叫设置消息中作为收到音频的目的地。

要获得地址，控制组件向服务器发送共享私密请求，获得共享私密，接着发送捆绑请求给服务器。捆绑请求中没有 CHANGE-REQUEST 属性，也没有 RESPONSE-ADDRESS 属性。捆绑响应包含映射过的地址。控制组件接着形成第 2 个捆绑请求。该请求包含 RESPONSE-ADDRESS，设置为从前面的捆绑响应中知道的映射过的地址。将该捆绑请求传递给媒体组件，同时发给 STUN 服务器的 IP 地址和端口。媒体组件发送该捆绑请求。该请求到达 STUN 服务器，它回发捆绑响应给控制组件。控制组件收到该响应，这样就已经知道将回路给发送请求的媒体组件的 IP 地址和端口。

客户就能够收到从任何地址到该映射地址的媒体。

在抑制静音的情况下，客户可能周期性地收不到媒体流。在这种情况下，UDP 捆绑将超时（通常 NAT 的 UDP 捆绑都很短；一般是 30 秒）。要解决这个问题，应用程序可能通过周期性地重传请求来保持捆绑更新。

某个媒体会话的两个参考方都在同个 NAT 后是可能的。在该情况下，两方将重复上述过程，且都获得公共地址捆绑。当一方向另一方发送媒体时，媒体寻路到 NAT，然后掉头回到企业中，在这时它转换为接收方的私有地址。这不是特别有效，然而不幸的是，在许多商业 NAT 中都没办法工作。在这种情况下，客户可能需要使用私有地址重试。

11. 协议细节

本节表述 STUN 消息的详细编码。

STUN 是个请求—响应协议。客户发送请求，服务器发送响应。有 2 种请求，捆绑请求，和共享私密请求。捆绑请求的响应可以是捆绑响应或捆绑错误响应。共享私密请求的响应可以是共享私密响应或共享私密错误响应。

STUN 消息编码为多个二进制域。所有的整数域用网络字节序传输，即，从离符号最近的字节（8 位组）开始。该字节序通常称为大端。传输顺序在 RFC 791[6]的附录 B 中详细描述。除非说明，数字常数是十进制（基数为 10）。

11.1 消息头

所有的 STUN 消息包含 20 字节的头。

字节

0	1	2	3
STUN 消息类型		消息长度	
事务 ID			

消息类型许可的值如下：

0x0001：捆绑请求
0x0101：捆绑响应
0x0111：捆绑错误响应
0x0002：共享私密请求
0x0102：共享私密响应
0x0112：共享私密错误响应

消息的长度是消息大小的字节数，不包括 20 字节的头部。

事务 ID 是 128 位的标识符。它还用人随机请求和响应的调节剂。与请求相应的所有响应都有与其相同的标识符。

11.2 消息属性

在头之后是 0 个或多个属性。每个属性进行 TLV 编码（译者注：TLV 即类型—长度—值英文缩写），类型 16 位，长度 16 位，接着是可变的值：

字节

0	1	2	3
类型		长度	
值			
……			

类型定义如下：

0x0001：MAPPED-ADDRESS
0x0002：RESPONSE-ADDRESS
0x0003：CHANGE-REQUEST
0x0004：SOURCE-ADDRESS
0x0005：CHANGED-ADDRESS
0x0006：USERNAME
0x0007：PASSWORD
0x0008：MESSAGE-INTEGRITY
0x0009：ERROR-CODE
0x000a：UNKNOWN-ATTRIBUTES
0x000b：REFLECTED-FROM

为了允许将来对本规范进行修订时需要增加新的属性，将属性空间分为可选部分和强制部分。值超过 0x7fff 的属性是可选的，即是说，客户或服务器即使不认识该属性也能够处理的该消息。值小于或等于 0x7fff 的属性是强制理解的，即，除非理解该属性，否则客户或服务器就不能处理该消息。

MESSAGE-INTEGRITY 属性 MUST 是消息中的最后一个属性。任何已知的，但不希望出现在消息中的属性（例如，请求中的 MAPPED-ADDRESS）MUST 乎略。

表 2 表示某个消息中可在出现的属性。M 表示强制在消息中包括该属性，C 意思是其（出现）条件基于消息的其它某方面，N/A 意思是该消息类型不使用该属性。

属性	捆绑请求	捆绑响应	捆绑错误响应	共享私密请求	共享私密响应	共享私密错误

						响应
MAPPED-ADDRESS	N/A	M	N/A	N/A	N/A	N/A
RESPONSE-ADDRESS	O	N/A	N/A	N/A	N/A	N/A
CHANGE-REQUEST	O	N/A	N/A	N/A	N/A	N/A
SOURCE-ADDRESS	N/A	M	N/A	N/A	N/A	N/A
CHANGED-ADDRESS	N/A	M	N/A	N/A	N/A	N/A
USERNAME	O	N/A	N/A	N/A	M	N/A
PASSWORD	N/A	N/A	N/A	N/A	M	N/A
MESSAGE-INTEGRITY	O	O	N/A	N/A	N/A	N/A
ERROR-CODE	N/A	N/A	M	N/A	N/A	M
UNKNOWN-ATTRIBUTES	N/A	N/A	C	N/A	N/A	C
REFLECTED-FROM	N/A	C	N/A	N/A	N/A	N/A

表 2：属性总结

长度提示值元素的长度值，表示为字节的无符号整数。

11.2.1 MAPPED-ADDRESS

MAPPED-ADDRESS 属性表示映射过的 IP 地址和端口。它包括 8 位的地址族，16 位的端口号，接着是长度固定的值，用来表示 IP 地址。

字节

0	1	2	3
XXXXXXXX	地址族	端口号	
地址			

端口号为网络字节序，表示映射的端口号。地址族通常是 0x01，与 IPv4 相对应。MAPPED-ADDRESS 的前 8 位乎略，作为本地边界对齐参数。IPv4 地址是 32 位的。

11.2.2 RESPONSE-ADDRESS

RESPONSE-ADDRESS 属性表示捆绑响应应该响应的目的地。它的语法与 MAPPED-ADDRESS 相同。

11.2.3 CHANGED-ADDRESS

如果捆绑请求的 CNANGE-REQUEST 属性中的“改变 IP”和“改变端口”标志设置了，则 CHANGED-ADDRESS 属性表示响应发出的 IP 地址和端口号。属性通常出现在捆绑响应中，独立于标志的值。它的语法与 MAPPED-ADDRESS 相同。

11.2.4 CHANGE-REQUEST

客户使 CHANGE-REQUEST 属性来请求服务器使用不同的地址和/或端口号来发送响

应。属性是 32 位长的，即使只用 2 位（A 和 B）：

字节

0	1	2	3
00000000	00000000	00000000	00000AB0

标志意义是：

A：这是“改变 IP”标志。如果为真，它请求服务器将捆绑响应发送到不同的 IP 地址，而不是收到捆绑请求的地址。

B：这是“改变端口”标导。如果为真，它请求服务器将捆绑响应发送到不同的端口，而不是收到捆绑请求的端口。

11.2.5 SOURCE-ADDRESS

SOURCE-ADDRESS 属性出现在捆绑响应中。它表示服务器发送响应的源 IP 地址和端口。它的语法与 MAPPED-ADDRESS 相同。

11.2.6 USERNAME

USERNAME 属性用于消息的完整性检查。它的意思是在消息完整性检查中标识共享私密。USERNAME 通常出现在共享私密响应中，与 PASSWORD 一起。当使消息完整性检查时，可有选择地出现在捆绑请求中。

USERNAME 的值是变长且不透明的。它的长度 MUST 是 4 的倍数，以保证属性与字边界对齐。

11.2.7 PASSWORD

PASSWORD 属性用在共享私密响应中。它通常出现在共享私密响应中，与 USERNAME 一起。

PASSWORD 的值是变长的，用作共享私密。它的长度 MUST 是 4 的倍数（以字节为单位），以保证属性与字边界对齐。

11.2.8 MESSAGE-INTEGRITY

MESSAGE-INTEGRITY 属性包含 STUN 消息的 HMAC-SHA1[13]。它可以出现在捆绑请求或捆绑响应中。由于使用 SHA1 散列算法，HMAC 将会是 20 个字节。用作 HMAC 输入的文本是 STUN 消息，包括头部，直到且包括 MESSAGE-INTEGRITY 属性前面的属性。文本填充 0 以使其 64 字节对齐。其结果是，MESSAGE-INTEGRITY 属性 MUST 是任何 STUN 消息的最后一个属性。用作 HMAC 输入的键取决于其内容。

11.2.9 ERROR-CODE

ERROR-CODE 属性出现在捆绑错误响应或共享私密错误响应中。它的数值范围从 100 到 699，加上以 UTF-8 编码的文本原因语句，且固定于它的代码分配，SIP[10]的语义和 HTT[15]。原因语句用于用户提示，且可以是表示原因代码的任何适当的语句。原因语句的长度 MUST 是 4 的倍数（以字节为单位）。如果需要，可以通过在文本的末尾加入空格来完成。所定义的响应号的缺省原因语句表述如下。

为有助于处理，从余下的代码中对错误代码（百位数）的分类进行单独编码。

字节

0	1	2	3
00000000	00000000	00000 分类	数字
原因语句（可变）			

分类表示响应号的百位数。其值 MUST 在 1 至 6 之间。数字表示响应号的 100 的模数，其值 MUST 在 0 至 99 之间。

下面的响应号，与它们缺省的原因语句（括号中）一起，目前定义为：

400 (Bad Request)：请求变形了。客户在修改先前的尝试前不应该重试该请求。

401 (Unauthorized)：捆绑请求没有包含 MESSAGE-INTEGRITY 属性。

420 (Unknown Attribute)：服务器不认识请求中的强制属性。

430 (Stale Credentials)：捆绑请求没有包含 MESSAGE-INTEGRITY 属性，但它使用过期的共享私密。客户应该获得新的共享私密并再次重试。

431 (Integrity Check Failure)：捆绑请求包含 MESSAGE-INTEGRITY 属性，但 HMAC 验证失败。这可能是潜在攻击的表现，或者客户端实现错误。

432 (Missing Username)：捆绑请求包含 MESSAGE-INTEGERITY 属性，但没有 USERNAME 属性。完整性检查中两项都必须存在。

433 (Use TLS)：共享私密请求已经通过 TLS 发送，但没有在 TLS 上收到。

500 (Server Error)：服务器遇到临时错误。客户应该再次尝试。

600 (Global Failure)：服务器拒绝完成请求。客户不应该重试。

11.2.10 UNKNOWN-ATTRIBUTES

UNKNOWN-ATTRIBUTES 属性只存在于其 ERROR-CODE 属性中的响应号为 420 的捆

绑错误响应或共享私密错误响应中。

属性包含 16 位值的表，每个表示服务器不认识的属性类型。如果未知属性数量是奇数，则表中的属性之一 MUST 重复，以使表的总长度是 4 字节的倍数。

字节

0	1	2	3
属性 1 类型		属性 2 类型	
属性 3 类型		属性 4 类型	
……		……	

11.2.11 REFLECTED-FROM

REFLECTED-FROM 属性只存在于其对应的捆绑请求包含 RESPONSE-ADDRESS 属性的捆绑响应中。属性包含请求发出的源的身份（按照 IP 地址）。它的目的是提供跟踪能力，这样 STUN 就不能被用作 DOS 攻击的反射器。

其语法与 MAPPED-ADDRESS 属性相同。

12. 安全考虑

12.1 用 STUN 攻击

一般说来，用 STUN 进行攻击可分类为拒绝服务攻击和窃听攻击。拒绝服务攻击可用来对付 STUN 服务器自己或对付其它使用 STUN 协议的元素。

STUN 服务器通过共享私密请求机制来创建状态。要防止被通信量所淹没，当新的请求到达时，STUN 服务器 SHOULD 通过丢弃存在的连接（例如，基于最久未使用（LRU）策略）来限制它保持打开的同时在线的 TLS 连接数。同样地，在服务器存储共享私密的情况下，它 SHOULD 它将存储的共享私密数量。

有巨大兴趣的攻击是使用 STUN 服务器和客户来发起对其它实体的 DOS 攻击，包括客户自己。

许多的攻击需要攻击者生成合法 STUN 请求的响应，以便向客户提供伪造的 MAPPED-ADDRESS。可使这种技术发起的攻击包括：

12.1.1 攻击 I：对同一目标的 DDOS 攻击

在这种情况下，攻击者给大量的客户提供相同的伪造 MAPPED-ADDRESS，该地址指向攻击目标。这将欺骗所有的 STUN 客户，使他们认为他们的地址是该目标地址。接着客户分发该地址以在上面接收通信（例如，在 SIP 或 H.323 消息中）。然而，所有的通信都聚焦在攻击目标。该攻击能够进行实质性的放大，特别是当用在使 STUN 来允许多媒体应用的客户上时。

12.1.2 攻击 II：客户静言

在这种攻击中，攻击者寻求拒绝客户访问 STUN 提供的服务（例如，客户使用 STUN 来允许基于 SIP 的多媒体通信）。要达到该目的，攻击提供给客户一个伪造的 MAPPED-ADDRESS。它提供的 MAPPED-ADDRESS 是没法路由的 IP 地址。其结果是，当客户分发该 MAPPED-ADDRESS 时，它不会收到任何它所期望收到的包。

该开发对攻击者而言不是非常有趣。它冲击单个客户，经常不是期望的目标。此外，任何能够发起该攻击的攻击者也能够用某种段对该客户进行拒绝服务（攻击），如阻止客户收到从 STUN 服务器的任何响应，或者甚至对 DHCP 服务器。

12.1.3 攻击 III：假设客户的标识

该攻击与攻击 II 相似。然而，伪造的 MAPPED-ADDRESS 指向攻击者自己。这允许攻击者收到目标是客户的流量。

12.1.4 攻击 IV：窃听

在这种攻击中，攻击者强制客户使用路由到它自己的 MAPPED-ADDRESS。它接着转发任何它收到的包给该客户。这种攻击将允许攻击者观察发送给客户的所有的包。然而，为了发起攻击，攻击者必须已经能够观察从客户到 STUN 服务器的包。在许多情况下（如当攻击从接入网络发起时），这就意味着攻击者已经观察到发送给客户的包。作为结果，该攻击只对攻击者在从客户到 STUN 服务器的路径上观察流量有用，但通常不是在包向客户路由的路径上。

12.2 发起攻击

需要重点注意的是这种性质的攻击（通过注入含有伪造 MAPPED-ADDRESS 的响应）需要攻击者有能力窃听从客户发送到服务器的请求（或者扮演该攻击的 MITM）。这是因为 STUN 请求包含事务标识符，由客户选择，是 128 位幂的随机数。服务器用含有该值的响应来回应，且客户乎略没有匹配该事务 ID 的任何响应。因此，攻击者为了提供客户会收受的伪造响应，攻击者需要知道请求中的该事务 ID。随机性的大数，与客户发送请求的需要知晓相组合，将预防攻击采用猜测事务 ID 的方法。

由于上面所有的攻击取决于这一点——注入有伪造 MAPPED-ADDRESS 的响应——阻止攻击就通过阻止这一个操作来达到。要阻止它，我们需要了解它能够完全成的各种方式。有几种是：

12.2.1 方法 I：危害合法的 STUN 服务器

在这种攻击中，攻击者通过病毒或特洛伊木马来危害合法的 STUN 服务器。大概，这将允许攻击者接管 STUN 服务器，且控制它生成的响应的类型。

对 STUN 服务器的危害还能够通过发现开放的端口来发起。开放端口的知识为在这些端口上的 DoS 攻击（或当穿越的 NAT 是完全锥型的 NAT 的情况下的 DDoS 攻击）创造了机会。通过使用端口探测来发现打开的端口已经是非常轻微的事了，所以这并不是主要的威胁。

12.2.2 方法 II：DNS 攻击

使用 DNS SRV 记录来发现 STUN 服务器。如果攻击者能够危害 DNS，它就能够注入映射到由攻击者运行的 STUN 服务器域名的 IP 地址的伪造记录。这就允许它注入伪造的响应来发起任何上面的攻击。

12.2.3 方法 III：欺骗路由器或 NAT

与危害 STUN 服务器相对，攻击者能够通过危害从客户到 STUN 服务器的路径上的路由器或 NAT 来引起 STUN 服务器生成有错误 MAPPED-ADDRESS 的响应。当 STU 请求通过欺骗的路由器或 NAT，它就重写包的源地址为希望的 MAPPED-ADDRESS 的源地址。该地址不能是任意的。如果攻击者在公共因特网上（就是在它与 STUN 服务器间没有 NAT），且攻击者没有修改 STUN 请求，则该地址就有从服务器发送到该地址的属性，将路由通过该危害路由器。这是因为服务器将顺送响应给请求的源地址。用修改过的源地址，唯一能够到达客户的方式是危害路由器是否转发它们。如果攻击者在公共因特网上，但它们不能修改 STUN 请求，他们可以在请求中插入 RESPONSE-ADDRESS 属性，包含 STUN 请求的实际源地址。这会使服务器发送请求给客户，而独立于 STUN 服务器所见的源地址。当转发 STUN 请求时，这就给攻击者伪造任意源地址的能力。

如果攻击者在私有网络上（就是，在它和 STUN 服务器间有 NAT），则攻击者将不能强制服务器在响应中生成任意的 MAPPED-ADDRESS。它们只能够强制 STUN 服务器生成路由到私有网络的 MAPPED-ADDRESS。这是因为在攻击者和 STUN 服务器间的 NAT 将会重写 STUN 请求的源地址，映射它为公共地址，即路由私有网络。因为这个原因，攻击者只能够强制服务器生成伪造的路由到私有网络的映射地址。非常不幸，低质量的 NAT 将会映射已经分配的公共地址为另一个公共地址（而不是内部私有地址），这样攻击者就能够强制 STUN 请求中的源地址为任意公共地址。NAT 的这种行为非常少见。

12.2.4 方法 IV：MITM

作为方法 III 的代替者，如果攻击者能够在客户到服务器的路径上放置一个元素，该元素能够扮演中介者。在该情况下，它能够截取 STUN 请求，并直接用 MAPPED-ADDRESS 域的希望值来生成 STUN 响应。同样，它能够转发 STUN 请求给服务器（在可能修改后），接收响应，并转发给客户。当转发请求和响应时，该攻击会受到在 12.2.3 节中描述的 MAPPED-ADDRESS 相同的限制。

12.2.5 方法 V：响应注射和 DoS

通过这种方法，攻击者不需要成为 MITM（与方法 III 和 IV 中一样）。它只需要窃听载有 STUN 请求的网络数据段的能力。在多路访问网络如以太网或非保护 802.11 中这非常容

易办到。要注入伪造的响应，攻击者监听网络中的 STUN 请求。当它发现一个，它同时对 STUN 服务器发起 DoS 攻击，并生成它自己的含有希望的 MAPPED-ADDRESS 值的 STUN 响应。该攻击者生成的 STUN 响应将到达客户，并且对服务器的 DoS 攻击的目标是阻止服务器的合法响应到达客户。有争议的是，攻击者能够办到而不需要对服务器发起 DoS 攻击，只要伪造响应打败回送给客户的真实响应，且客户使用第一个响应并乎略第二个（即使它们是不同的）。

12.2.6 方法 VI：复制

该方法与方法 V 相似。攻击者监听网络中的 STUN 请求。当它发现了，它就生成它自己的 STUN 请求给服务器。该 STUN 请求与它发现的相同，但有哄骗的源 IP 地址。该哄骗的地址与攻击者希望放在 STUN 响应的 MAPPED-ADDRESS 中的地址相同。实际上，攻击者生成大量这样的包。STUN 服务器将会收到原始的请求和大量复制的伪造请求。它给所有的请求生成响应。如果该洪流足够大以使路由器或其它设备拥塞，就有可能使真实的响应丢失（与许多伪造的一样），但是网络造成只有伪造的响应被 STUN 客户所收到。该响应是全部一样的且所有包含的 MAPPED-ADDRESS 都是攻击者希望客户使用的。

复制包的洪流不是必须的（即，只有发送一个伪造请求），只要伪造响应打败回送给客户的真实响应，且客户使用第一个响应而乎略第二个（即使它们是不同的）。

要注意，在该方法中，对 STUN 服务器或 IP 网络发起 DoS 攻击来阻止有效响应的发送或接收是有问题的。攻击者需要 STUN 服务器能够处理它自己的请求。由于从客户发送请求的周期性重传，这只留下一个非常小的机会。攻击者必需在客户发送实际请求后立即开始 DoS 攻击，以使正确的响应被丢弃，然后停止 DoS 攻击以发送它自己的请求，所有的要在客户的下次重传开始前完成。由于重传的关闭空间（100ms 到几秒），这是非常难办到的。

除了 DoS 攻击，还有其它办法来阻止客户的实际请求到达服务器。例如，控制第 2 层可能会达成。

非常幸运，方法 IV 面临与在第 12.2.3 节中描述的相同的限制，即限制攻击者能够使 STUN 服务器生成的 MAPPED-ADDRESS 的范围。

12.3 对策

STUN 提供应对上面所述方法的机制，而且额外的非 STUN 技术同样能够应用。

首先，RECOMMENDED STUN 客户的网络实现入口源过滤（RFC 2827[7]）。这对于 NAT 自己是非常重要的。如 12.2.3 节所解释的，不进行这种检测的 NAT 能够被用做 DDoS 攻击的“反射器”。许多 NAT 都执行这种作为缺省操作模式的检测。我们强烈建议人们购卖允许该能力并激活的 NAT。

其次，RECOMMENDED STUN 服务器运行在只用行 STUN 的主机上，且禁止除 STUN 端口外的所有的 UDP 和 TCP 端口。这会阻止病毒和特洛伊木马感染 STUN 服务器，以阻止它们的危害。这帮助减轻方法 I（12.2.1 节）

再次，要阻止 12.2.2 节、9.2 节中的 DNS 攻击，推荐客户在 DNS 查询中验证服务器提供的证书的名称。

最后，所有上面的攻击取决于客户获得它从 STUN 了解的映射地址，且在应用层协议中使用它。如果这些协议提供加密和消息完整性，窃听和完整性企图攻击就可能被阻止。因此，应用程序在应用层协议中使用 STUN 地址 SHOULD 使用完整性和加密，即使如果该协议没有指定一个 SHOULD 的级别强度。例如，多媒体应用程序使用 STUN 地址来收到 RTP 通信量次使用安全 RTP[16]。

上面的三项技术是非 STUN 机制。STUN 自己提供了几种对策。

方法 IV（12.2.4 节），当本地产生响应，且 V（12.2.5 节）需要攻击者生成伪造的响应。攻击可使用 STUN 提供的消息完整性机制来阻止，在 8.1 节中描述。

方法 III（12.2.3 节）IV（12.2.4 节）和 VI（12.2.6），当使用转发技术，然而就是不能通过服务器标记来阻止的。当攻击者能够修改请求（插入路由到客户的 RESPONSE-ADDRESS）时，所有的方法都是有效的。非常幸运，这种修改可使用用 9.3 节中描述的消息完整性技术来阻止。然而，当攻击者不修改除了 STUN 请求的源地址外的任何域时这些方法依然有用。不幸，这种事情不能通过加密技术来保护，因为这种改变是 STUN 自己都需要寻求检测和报告的。这是 NAT 与生俱来的弱点，而不是 STUN 固有的。要有助于减轻这些攻击，9.4 节提供了客户遵循的几种探索方法。客户发现不一致或额外的响应，所有的都有上面描述的攻击的迹象。然而，这些探索方法只是—探索，且不能保证会阻止攻击。这些探索似乎如同我们今天知道如何发起攻击一样阻止攻击。实现者应该继续发表关于未来可能需要的新探索的信息。这种信息将发布于 IETF MIDCOM 邮件列表中，midcom@ietf.org。

12.4 其余的威胁

上面列出的对策都不能能够阻止 12.2.3 节中描述的攻击，如果攻击者位于网络路径上的适当的位置。特别是，考虑这种情况，攻击者希望说服客户 C 它的地址是 V。攻击者需要有在 A 和服务器间的路径上（为了修改请求）和在服务器与 V 间的网络元素，这样它就能够转发响应给 C。更进一步，如果在攻击者与服务器间有 NAT，则 V 必须也在同个 NAT 后。在这种情况下，攻击者既能够获得所有应用层通信量，也能够安装 12.1.1 节中描述的 DDOS 攻击。注意任何存在于正确拓扑关系中的主机都可被 DDOS。它不需要使用 STUN。

13. IANA 考虑

STUN 不能够扩展。该协议的修改通过该规范的标准路线修订来进行。这样，不需要注册 IANA。任何将来的扩展将建立任何需要的注册。

14. IAB 考虑

IAB 已经研究了问题“单方面的自主地址修复”，这是客户通过合作协议反射机制（RFC

3424[17]）试图判断在 NAT 另一边的另一域的它的地址的通用过程。STUN 是执行这种功能的协议的一个例子。IAB 已经要求为该目的开发的任何协议要说明特定考虑的集合。该节满足这些要求。

14.1 问题定义

从 RFC 3424[17]，任何 UNSAF 提议必须提供：

精确明确的定义，UNSAF 提议将解决的限制范围的问题。短期修补不应该通用化来解决其它问题；这就是为什么“短期修补通常什么都不是”。

STUN 要解决的明确的问题是：

o 为客户检测在它和在公共因特网上的服务提供商运行的服务器间存在的一个或多个 NAT 提供办法。这种检测的目的是判断可能必要的额外的步骤以能够收到从特定提供者来的服务。

o 为客户检测在它和另一个客户间存在的一个或多个 NAT 提供办法。这里的第二个客户可被第一个访问，但不知道第二个客户是否位于公共因特网上。

o 为客户获得非对称 NAT 分配的公共因特网地址提供办法。专门的目的是收到从其它主机发来的目标是该地址的 UDP 通信量。

STUN 不寻址 TCP，不管是向内或向外，而且不寻址向外的 UDP 通信。

14.2 退出策略

从[17]，任何 UNSAF 提议必须提供：

描述退出策略/转换计划。更好的短期修补将自然地更少使用，因为适当的技术已经部署。

STUN 有它自己内建的退出策略。该策略是检测操作，它先于实际的 UNSAF 地址修补操作而执行。该发现操作，在 10.1 节中描述，试图发现在客户和服务提供商网络间的任何 NAT 的存在和类型。然而，检测 NAT 的特定类型可能是脆弱的，发现存在 NAT 是相当强健的。由于通过部署 IPv6 使 NAT 逐渐被淘汰，该发现操作将立即返回，其结果是不存在 NAT，且不需要更进一步操作。相反，该发现操作自己能够用于帮助刺激部署 IPv6；如果用户检测到他们与公共因特网间存在 NAT，他们能够呼叫他们的接入提供商并抱怨。

STUN 还可帮助促进引入中间盒。由于中间盒兼容的 NAT 部署，应用程序将，不使用 STUN（也位于应用层），首先分配中间盒绑定的地址。然而，都知道的中间盒的限制是只有当代理知道它的通信量将流过中间盒时它才有用。一旦已经通过这些中间盒分配了绑定，STUN 检测过程能够证实没有额外的中间盒在公共因特网与客户的路径上。如果是这种情况，应用程序可继续操作使用从中间盒分配的地址绑定。如果不是这种情况，STUN 提供了

自地址修补机制来通过余下的不兼容的中间盒。这样，STUN 提供了帮助转换到完全中间盒兼容的网络上的方法。

14.3 STUN 引入的脆弱性

从[17]，任何 UNSAF 提议必须提供：

可能导致系统更“脆弱”的明确问题的讨论。例如，在多网络层使用数据的方式将建立更多的附属品；增加调试的挑战性；并使移植更困难。

STUN 通过几种方式来引入系统中的脆弱性：

o 发现过程假设存在一定种类的基于它们对待 UDP 的方式的设备。可能部署了其它类型的 NAT，它不适作这些模型之一。因此，将来的 NAT 可能不能正确地被 STUN 检测到。STUN 客户（不是服务器）将需要修改以适应它。

o STUN 获取绑定的用法不能工作于所有 NAT 类型。它只能在使用完全锥型 NAT 的任何应用程序上工作。对于限制锥型和端口限制锥型 NAT，它将在一些应用程序中工作，这取决于应用程序。通常应用程序将需要特定的过程。对于对称型 NAT，该绑定获取的也将是不可使用的地址。这些特定类型 NAT 上的严厉的附属品使该协议更脆弱。

o STUN 假设服务器存在于公共因特网上。如果该服务器们于另一个私有地址域中，则用户将能够或不能使用它发现的地址与其他用户通迅。没有办法来检测这种情况。

o NAT 分配的绑定需要持续地更新。由于这些绑定的超时参数正是由实现指定的，该更新的间隔时间将不能轻松地判断。当绑定不活动，不能用于接收流量时，但却等待着接收到来的消息，则该绑定的更新将是不需的，只会消耗网络带宽。

o 使用 STUN 服务器作为附加的网络元素引入了另外的潜在安全攻击的点。这些攻击的大部分会被 STUN 提供的安全检测阻止，但不是全部。

o 使用 STUN 服务器作为另外的网络元素引入另外的失败点。如果客户不能定位 STUN 服务器，或如果服务器由于失败而不可用，则应用将不起作用。

o 使用 STUN 来发现地址绑定将导致应用延迟的增加。例如，跨 IP 的语音应用将发现呼叫建立延迟的增加等于与 STUN 服务器的至少一个的 RTT。

o 发现绑定的生存期可能会错误。它假定所有绑定存在相同的生存期。如果 NAT 使用动态绑定生存期来处理过载或者如果 NAT 在发现过程中自己重启，则这就是不可能的。

o STUN 具有一些在网络固有操作拓扑上的约束。如果客户 A 从 STUN 服务器 X 获取地址，并发送给客户 B，B 可能不能使用该 IP 地址向 A 发送。如果下列任何情况发生，则该地址就不能使用：

STUN 服务器不在客户 A 和 B 共同的通用祖先（拓扑上的）的地址域中。例如，考虑客户 A 和 B，都有住宅 NAT 设备。两个设备都连接到它们的线缆操作器，但两客户有不同的提供商。每个提供商在他们网络前端都有 NAT，连接到公共因特网。如果 A 使用的 STUN 服务器位于 A 的线缆操作器网络中，则通过它获得的地址将不能被 B 使用。STUN 服务器必须在属于两者的通用祖先的网络中——在这种情况下，就是公共因特网。

STUN 服务器在两个客户的通用祖先的地址域中，但是两个客户都在同个 NAT 后，连接到该地址域。例如，如果前而例子中的两个客户有相同的线缆操作器，该线缆操作器有单个 NAT 连接它们的网络到公共因特网，且 STUN 服务器在公共因特网上，则 A 获得的地址将不能被 B 使用。这是因为一些 NAT 交不接受内部包发送到公共 IP 地址，该地址又映射回内部网络。要解决这个问题，需要引入额外的协议机制或配置参数来检测这种情况。

o 很显然，STUN 引入了不能去除的潜在的安全威胁。本规范描述了能够用来减轻该问题的探索，但它可能不能解决获得 STUN 努力完成的东西。这些安全问题完全在 12 节中描述。

14.4 长期解决方法的需求

从[17]，任何 UNSAF 提议必须提供：

参与长期方法的需求，可行的技术解决方案——有助于发现正确的长期解决方法。

我们的 STUN 经历引起如下的对长期解决 NAT 问题的需求：

请求捆绑和控制 NAT 中需要公开的其它资源。STUN 我许多脆弱性从它猜测 NAT 的参数继承而来，而不是告诉 NAT 使用什么参数。

控制需要“在带内”。有太多的情景，客户将不知道中间盒在其前而什么位置。取而代之，控制该盒需要发生在带内，按与数据传自身传输相同的路径传输。这保证操作正确的中间盒集。这只对第一方控制有用；第三方控制最好使用中通框架来处理。

控制需要限制。用户将需要通过 NAT 来通迅，而它在管理控制之外。为了使供应商部署能够被不同域中的用户控制的 NAT，则这种控制的范围需要极度地限制——一般是，分配控制包到来的接触地址的绑定。

简单最重要。控制协议将需要在非常简单的客户中实现。该服务器将需要支持极高的负载。协议将需要极度强健的，作为主机应用协议的先导。所以，简单是关键。

14.5 与现存 NAPT 盒的问题

从[17]，任何 UNSAF 提议必须提供：

讨论与现存的已部署的 NA[P]T 的实践问题的明显影响和经历报告。

与 STUN 包含特性的几个实践问题证明——当新 NAT 类型部署时推翻协议。幸运地，

当前这还不是问题，因为大多数部署的 NAT 是 STUN 假设的类型。STUN 的主机用途发现是在 VoIP，帮助分配地址来接收 RTP[12]流量。在该应用中，RTP 流量自己提供周期的保活消息。然而，对 RTP 有几个实践问题。首先，RTP 假设 RTCP 流量在主于 RTP 流量的端口上。该成对的特性将不能通过 NAT 来保证，在为 NAT 不能直接控制。其结果是，可能不能正确地收到 RTCP 流量。SDP 的协议扩展已经提议通过允许客户标明不同的 RTCP[18]端口来减轻该问题。然而，一段时间内将会存在互操作的问题。

对于 VoIP，静音抑制能够引起 RTP 包传输的间隙。这能够导致在呼叫的过程是丢失绑定，如果该静单抑制超过绑定超时。可通过经常发送静音包来保持绑定存活。然而，结果是额外的脆弱性；适当的操作取决于使用的静单抑制算法，使用舒服的杂音编码，静单周期的持续期，和 NAT 中的绑定生存期。

14.6 结束语

STUN 的问题不是 STUN 设计的问题。STUN 的问题是由于 NAT 缺少标准化的行为和控制。缺少标准化的结果已经存在于激增的设备中，它们的行为非常不可预知，极度变化，且不可控制。STUN 在该敌对的环境中已尽其所能。最终，解决办法是使用环境更少敌对，并向 NAT 引入控制和标准化行为。然而，直到这些发生的时该，STUN 为被迫在可怕的条件下操作提供了好的短期解决方法。

15. 感谢

作者非常感谢 Cedric Aoun，Pete Cordell，Cullen Jennings，Bob Penfield 和 Chris Sullivan 的注解，和 Baruch Sterman 和 Alan Hawrylyshen 的首次实现。感谢 Leslie Daigle，Allison Mankin，Eric Rescorla 和 Henning Schulzrinne 的在本文中加入 IESG 和 IAB。

16. 参考标准

[1] Bradner, S., "Key words for use in RFCs to indicate requirement levels", BCP 14, RFC 2119, March 1997.

[2] Dierks, T. and C. Allen, "The TLS protocol Version 1.0", RFC 2246, January 1999.

[3] Gulbrandsen, A., Vixie, P. and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, February 2000.

[4] Chown, P., "Advanced Encryption Standard (AES) Ciphersuites for Transport Layer Security (TLS)", RFC 3268, June 2002.

[5] Rescorla, E., "HTTP over TLS", RFC 2818, May 2000.

[6] Postel, J., "Internet Protocol", STD 5, RFC 791, September 1981.

[7] 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.

17. 参考信息

[8] Senie, D., "Network Address Translator (NAT)-Friendly Application Design Guidelines", RFC 3235, January 2002.

[9] Srisuresh, P., Kuthan, J., Rosenberg, J., Molitor, A. and A. Rayhan, "Middlebox Communication Architecture and Framework", RFC 3303, August 2002.

[10] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M. and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[11] Holdrege, M. and P. Srisuresh, "Protocol Complications with the IP Network Address Translator", RFC 3027, January 2001.

[12] Schulzrinne, H., Casner, S., Frederick, R. and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", RFC 1889, January 1996.

[13] Krawczyk, H., Bellare, M. and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, February 1997.

[14] Kohl, J. and C. Neuman, "The kerberos Network Authentication Service (V5)", RFC 1510, September 1993.

[15] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P. and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[16] Baugher M., et al., "The secure real-time transport protocol", Work in Progress.

[17] Daigle, L., Editor, "IAB Considerations for UNilateral Self-Address Fixing (UNSAF) Across Network Address Translation", RFC 3424, November 2002.

[18] Huitema, C., "RTCP attribute in SDP", Work in Progress.

18. 作者地址

Jonathan Rosenberg
dynamicsoft
72 Eagle Rock Avenue
First Floor
East Hanover, NJ 07936

EMail: jdrosen@dynamicsoft.com

Joel Weinberger
dynamicsoft
72 Eagle Rock Avenue
First Floor
East Hanover, NJ 07936

EMail: jweinberger@dynamicsoft.com

Christian Huitema
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052-6399

EMail: huitema@microsoft.com

Rohan Mahy
Cisco Systems
101 Cooper St
Santa Cruz, CA 95060

EMail: rohan@cisco.com

19. 完整的版权声明

特别感谢

互联网协会目前为 RFC 编辑功能提供资助。