基于 Linux 的 NAT 技术研究

袁小珂
(中国民航飞行学院 计算机学院,四川 广汉 618307)

摘 要:为缓解 IPv4 模式下的 IP 地址短缺问题,文中以 Linux 为系统平台,深入分析了网络地址转换(NAT)的基本原理和实现技术,然后对其扩展应用进行了理论和实际的探讨,最后将排队模型引入到 NAT 的映射过程中,并进行了相应理论分析。给出了一种 NAT 的形式化模型和关于 NAT 映射的函数,实现了一种基于端口映射方式的具体算法。研究结果表明,网络地址转换为解决 IP 地址短缺提供了有效的解决方案,但其应用并非仅限于此。网络地址转换技术可广泛适用于虚拟服务器、负载均衡以及多路传送等多种不同应用领域。

关键词:网络地址转换;地址映射;负载均衡;虚拟服务器;排队模型
中图分类号:TP309 **文献标识码**:A **文章编号**:1673-629X(2011)11-0243-04

Research of NAT Techniques Based on Linux

YUAN Xiao-ke
(Computer School of Civil Aviation Flight University of China, Guanghan 618307, China)

Abstract: To remit the shortage of IP address under IPv4 mode, using Linux as the platform, it firstly introduces the basic theory and classic techniques of NAT. Then it puts emphasis on its expand techniques in theory and in practice, and a queuing model is introduced to NAT mapping process. The related theory is analyzed in the end. An NAT formal models and an NAT mapping functions are given, as well as a specific algorithm about the way realization based on port mapping. Research results show that NAT technology provides an effective solution to solve the shortage of IP address, but its application is not only that. It can be widely used for many different applications such as virtual server, load balancing, multi-channel transmission, etc.

Key words: NAT; address mapping; load balancing; virtual server; queuing model

0 引 言

计算机网络技术的快速发展导致了网络地址的日益紧缺,IP 地址的匮乏已成为限制其发展的瓶颈。从目前来看,使用 IPv6 是从根本上解决 IP 地址短缺的唯一办法;但由于 IPv4 网络的广泛采用,短时间内 IPv6 网络还无法完全取代 IPv4 网络,因此很长时间内两者将共存,而 IPv4 将逐渐向 IPv6 过渡。由此无类型域间路由(CIDR)、网络地址转换(NAT)、代理服务器(Proxy)等各种过渡方案应运而生。文中将基于 Linux 平台对 NAT 技术进行深入分析。

1 NAT 基本技术研究

网络地址转换(Network Address Translation, NAT)是指将一个 IP 地址转换为另一个 IP 地址的过程,它是一种基本的以延缓 IP 地址耗用速度为目的的过渡性方案[1~4]。

1.1 NAT 的基本原理

定义 1 设 N 为某网络,M 为 N 中 IP 报文的集合,若 $\forall m_i \in M$, $\exists m_i$ 满足源地址或目的地址为本网络(或本网络内部节点),则称 N 为桩域(Stub Domain)。

由于无报文需途经桩域进行中转,而且桩域与外部网络的连接接口有限(一个或几个),所以 NAT 模块可以方便地基于桩域来实现。桩域与外部网络的连接通常采用路由器连接,因此 NAT 模块可以安装在路由器与外部网络连接接口的位置。可以使用为私有网络保留的地址(如 192.168.X.X)作为桩域内部的地址,这些保留地址可以被其它桩域复用。一方面,在桩域内两个内部节点之间的通信可以使用保留的私有地址来完成;另一方面,内部节点与公共网络节点通信时则先通过 NAT 模块将内部保留地址映射成外部可见的地址,然后再进行数据通信。如图 1 所示。

1.2 NAT 实现过程的形式化分析

形式化方法(Formal Methods)是计算机网络研究

收稿日期:2011-04-22;**修回日期**:2011-07-29
基金项目:国家自然科学基金(60879022);民航局科研基地 863 项目(2007KF003);中国民航飞行学院青年基金(Q2009-28)
作者简介:袁小珂(1976-),男,四川盐亭人,讲师,硕士,主要从事网络与信息安全方面的研究。

中广泛采用的一种技术，这里采用面向模型的形式化方法来对 NAT 过程的实现进行分析。

定义 2 设 $F(A \leftrightarrow B)$ 表示网络通信中任意节点 A、B 之间的特定逻辑关系，则称 F 为流(Flow)。

流 F 可以有不同的具体存在形式，如在一个 UDP 会话过程中，相关 UDP 数据报前后之间有相互的逻辑关系，该会话就构成一个流；当流是一个 TCP 连接时，可以描述如下：

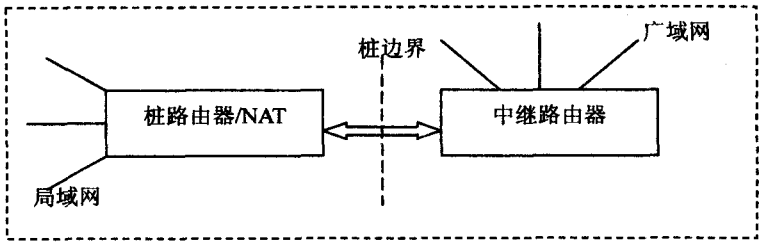

图 1 NAT 模块位置

F=((源 IP 地址，源端口号)，(目的 IP 地址，目的端口号))

基于以上关于流的定义，考虑内网节点 A 与外网节点 B 之间的一个 TCP 会话过程。如图 2 所示，假定桩域内节点 A 通过路由器 R 访问外网中提供网页访问服务的节点机器 B。

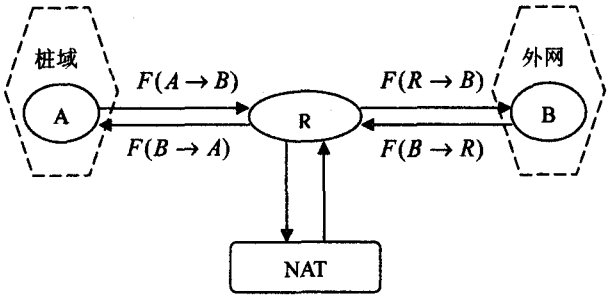

图 2 形式化 NAT 模型

具体过程如下：

Step 1: $F(A \rightarrow B) \Leftrightarrow$((Address A,port A),(Address B,80))

Step 2: $F(A \rightarrow B) \overset{NAT}{\Rightarrow} F(R \rightarrow B) \Leftrightarrow$((Address R,port R),(Address B,80))

Step 3: $F(B \rightarrow R) \Leftrightarrow$((Address B,80),(Address R,port R))

Step4: $F(B \rightarrow R) \overset{NAT}{\Rightarrow} F(B \rightarrow A) \Leftrightarrow$((Address B,80),(Address A,port A))

分析上述 NAT 实现，其关键问题是如何将桩域内的私有 IP 地址映射成外网可见的全局 IP 地址。

定义 3 假定桩域内的私有 IP 地址数目为 x；桩域与外网的边界点处的公有 IP 地址数目为 y，则有关于 NAT 映射的函数：

$$f_{ant}(x,y)=\begin{cases}1 & \text{当 } x=y \text{ 且 } x,y \text{ 均为} \geqslant 1 \text{ 的正整数}\\ x/y & \text{当 } x \geqslant 1 \text{ 且 } x \geqslant y, x,y \text{ 均为} \geqslant 1 \text{ 的正整数}\\ x & \text{当 } x \geqslant 1 \text{ 且 } y=1, x,y \text{ 均为正整数}\end{cases} \quad (1)$$

从该映射函数可知：

1) 当 $f_{nat}(x,y)$ 为 1 时，该映射为静态地址转换(Static Address Translation)；

2) 当 $f_{nat}(x,y)$ 为 x/y 时，该映射为动态地址转换(Dynamic Address Translation)；

3) 当 $f_{nat}(x,y)$ 为 x 时，该映射为端口转换(Masquerading)，且该映射为动态地址转换的一个特例。

1.3 动态端口映射的数据结构和实现

端口占用 16 位，范围从 0 ~ 65535。其中 0 ~ 1023 为熟知端口，这些端口被常用服务所占用[5,6]。在 Linux 中为避免采用熟知端口而带来的交互问题，通常使用 61000 ~ 65095 范围内的端口，该范围可以使用的并发连接有 4096 个，通过 linux/include/net/ip_masq.h 文件可进行修改。

为完成地址转换，需要设定如下基本数据结构：

```
typedef struct _Rule {
  UINT    nSer;//序号
  ULONG   lHostAddr;//连接的主机地址;
  UINT    nHostPort;
          //连接的主机端口号;
  UINT    nTranslatePort;//
          转换后的端口号;
  CHAR    ProtocolType;
          //连接的协议类型;
} Rule ;
```

分析 Linux 内核可知网络地址转换是基于内核中提供的 Netfilter 子系统来实现的[7]。利用内核代码提供的 5 个 Hook 点注册相应的钩子函数来实现一定的功能并编译成模块，以动态形式加载到内核中，Netfilter 框架流程如图 3 所示。

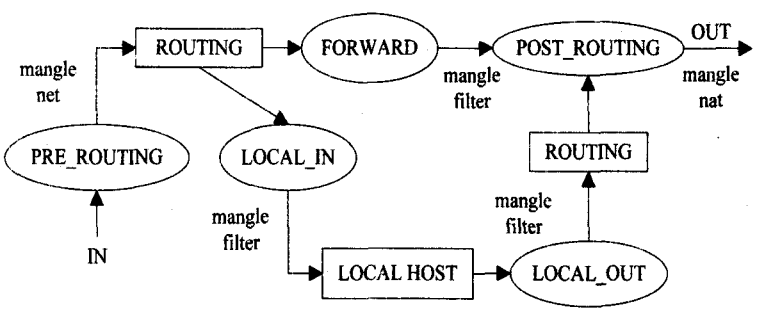

图 3 Netfilter 框架流程

为有效实现映射,NAT 要为每一连接建立相应的进程表项,通过维护查询端口映射表对不同流向的报文分别进行报头的转换。转换算法决定了转换过程的实际效率,可选取哈希表的方式来实现映射算法。如下的哈希函数 usIndex = ((ulIpAddr&0x000000FF) << 4) + (usSrcPort&0x000F);对一个 C 类地址取其低 8 位,再取其低 4 位,可将由源地址和源端口及协议所组成的半相关散列对应到整个哈希表中。采取哈希表结构后,还需要处理哈希冲突的情况,可以采用开放定址法进行冲突处理。

定义 4　假设哈希表长度为 Len,增量序列为 d_i,则哈希过程为:

$$H_i = (H(\text{key}) + d_i) \text{ MOD Len} \quad (2)$$

式中,$i = 1,2,\cdots,n(n \leqslant \text{Len} - 1)$;$d_i \in [1^2, -1^2, 2^2, -2^2, 3^2, \cdots, \pm n^2]$,其中 $n \leqslant \text{Len}/2$。

2　NAT 扩展应用研究

以上所讨论的 NAT 技术及实现都以缓解 IP 地址匮乏为目的,下面对 NAT 的扩展应用进行研究,这些应用与上面的讨论并没有直接的关系,因此这些应用并不会因为 IPv6 等长期方案的采用而变得过时。

2.1　虚拟服务器

将一个全局 IP 地址定义为虚拟服务器,而把桩域内的主机分别与不同的保留 IP 地址相联系。例如,当客户机向一个作为 WEB 服务器使用的虚拟服务器发出 TCP 连接请求时,NAT 模块就根据特定算法改变 TCP 报文首部所包含流的目的 IP 地址,将该 TCP 请求重定向到桩域内的某台实际服务器上,因此客户机实际连接了该实际服务器,而对于外部连接来说该 NAT 过程是透明的。利用虚拟服务器可以使服务更加可靠,在其它实际服务器正常运行时,对某台内部实际服务器的维护可通过在高层协议中对连接重定向来实现,而这并不影响客户机的连接请求。

在虚拟服务器的应用中服务的可靠性是该应用的关键。按照概率论的知识,有以下结论:

定义 5　假定实际服务器的数目为 $m(m \in N)$,某实际服务器 SR_k 故障的概率为 p_k,NAT 模块的故障率为 p_t,显然各概率相互独立且都满足 Poisson 分布,即

$$P(X = \xi) = \frac{\lambda^{\xi}}{\xi!} e^{-\lambda} \quad (3)$$

由此可计算出系统中虚拟服务器 SV 正常工作的概率 $p_{\text{virtual\_on}}$ 满足以下公式:

$$p_{\text{virtual\_on}} = (1 - p_t) \times \left(1 - \prod_{k=1}^{m} p_k\right) \quad (4)$$

而虚拟服务器 SV 的故障率 p_{virtual} 为:

$$p_{\text{virtual}} = 1 - p_{\text{virtual\_on}} \quad (5)$$

根据公式(3)~(5)可知,在各设备故障率不变的情况下,实际服务器数目越多,虚拟服务器的故障率越小,即系统的可靠性越高。在实际应用中,就需要在系统成本和系统可靠性方面综合考虑。

2.2　负载均衡

负载均衡技术基于现有网络结构提供了一种透明扩展服务器带宽和提高网络数据处理能力的方式。负载均衡策略的优劣及其实现的难易程度主要有两个关键因素:负载均衡算法以及对网络状况的检测方式和能力。为了使得负载分配相对均衡,负载均衡策略应当考虑服务请求的类型、实际服务器的处理能力等问题。针对一次服务请求,请求和响应的数据报文都需要通过负载调度器,当实际服务器的数目在 10~20 台时,负载调度器将成为整个系统的性能瓶颈。大多数网络服务请求报文较短而响应报文较长,若在负载调度器中只负责调度请求而响应直接返回给客户,将极大地提高整个系统的吞吐量。

可以基于高层协议来实现负载均衡算法。Linux 中的 squid 是一个带缓存功能的支持多种协议专用代理程序,可以处理 HTTP、SSL、FTP、GOPHER 和 SNMP 等多种协议,并对 SSL 和 URL 数据作缓冲处理。例如,缓冲器上 squid 能根据 http 访问的特点决定是否进行缓冲以及怎样实现缓冲,对 cgi 程序返回的动态页面,squid 可以被设置成不对其进行缓冲;squid 还可以配置成透明代理的工作模式。

在 Linux 平台下,要实现负载平衡集群系统,可以利用 LVS(Linux Virtual Server)来完成[8,9]。例如 RedHat 的 Piranha、TurboLinux 公司的 Turbo Cluster 等商业的集群产品,都是基于 LVS 核心代码的。LVS 有三种负载平衡方式,NAT(Network Address Translation),DR(Direct Routing),IP Tunneling;其中,DR 是最为常用的方式。

2.3　多路传送

上述虚拟服务器和负载均衡的实现都是基于主机进行的,也可以基于网络本身来实现上述应用。创建一个由多个实际连接构成的虚拟网络连接,假设有多个 ISP 供应商,则桩域中与公共网络互连的各主机都有多个分别由 ISP 提供的全局 IP 地址。当其中一个 ISP 连接发生故障时可以使用另外的 ISP 连接,这样可满足可靠性的要求;为实现线路负载均衡,可以按照特定的策略选择各主机使用不同的 ISP 连接从而实现对网络负载的分配。

如图 4 所示,其中 Net A 和 Net B 分别代表两个不同的 ISP 提供商。当一台内网主机发起一次目的地为公网某主机的 TCP 连接时,该请求以透明方式进行相应处理,即请求首先被发送至 NAT 路由器,NAT 路由

器选择一条实际的 ISP 连接线路并将该请求报文流的源 IP 地址改为该 ISP 提供的实际 IP 地址，然后将报文发送出去；同样，从外网返回的报文也从相同的 ISP 线路回到 NAT 路由器。在此过程中，线路的选择算法显然决定了效率。

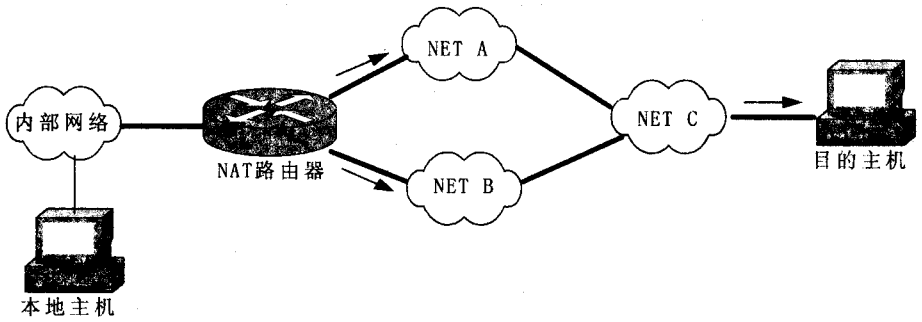

图 4 NAT 的多路传送实现

可以引入排队论中的相应排队模型对选择算法进行分析[10~12]。基本排队模型包括以下参数：到 NAT 路由器的 m 个向外访问的连接数据流，$w(w \leqslant m)$ 个服务台(ISP 的最大数目)，N ($N \leqslant w$) 个可以提供服务的 ISP。其基本处理流程为：一个数据流到达后，若所有线路都忙则数据流被直接丢弃；若忙的 ISP 数小于 N 则连接立即得到服务(并同时维护连接映射数据表)；若忙的 ISP 数等于 N 则连接进入队列排队，按先来先服务(FCFS)的算法处理。排队模型算法同样适用于负载均衡和虚拟服务器的应用。

在排队的实际应用中，通常有 Erlang-C、Erlang-B 和 Erlang-A 三种排队模型。其中，Erlang-C 模型应用最为普遍。

定义 6 在 Erlang-C 模型中，设 λ 为连接数据流到达率，μ 为服务率，N 为 ISP 数目；假定(1) 连接数据流到达满足固定速率 λ 的 Poisson 分布；(2) 对连接数据流的服务服从服务率 μ 的指数分布；则 Erlang - C 对应的排队模型为 $M/M/N/\infty$，W 为任意连接数据流的等待时间；且网络的负载为 $\alpha = \lambda/\mu$。

在 $\alpha \leqslant N$ 的情况下排队等待时间的具体分布：

$$P(W > t) = \begin{cases} C(N,a)e^{-(N\mu-\lambda)t} & a \leqslant N \\ 1 & a > N \end{cases} \quad (6)$$

式中，$C(N,a) = \dfrac{a^N}{(N-1)!\,(N-a)}\left[\sum_{j=0}^{N-1}\dfrac{a^j}{j!} + \dfrac{a^N}{(N-1)!\,(N-a)}\right]$。

假定在系统中排队等待时间服从均值 $(N\mu - \lambda) - 1$，由此可得到关于排队平均等待时间的公式为：

$$E[\text{wait for } M/M/N] = P\{W > 0\} \cdot E[W \mid W > 0] = C(N,a) \cdot \left(\frac{1}{N}\right)\left(\frac{1}{\mu}\right)\left(\frac{1}{1-\rho}\right) \quad (7)$$

式中，$\rho = \dfrac{\lambda}{N\mu}$。

按照上述公式(6)和(7)，可以根据相应参数计算出排队系统的平均等待时间。并由计算出的系统排队平均等待时间对实际应用的性能进行定量分析，确定合理的 ISP 数目。

3 结束语

网络地址转换是为解决 IP 地址短缺而提出的一个过渡性方案，但并非仅仅是一个过渡性的方案。使用网络地址转换过程中，由于路由器必须对每个数据报文进行检查和进行操作，因此会加大运行负载，增加处理时延。另外，网络地址转换使得对数据报文的跟踪变得困难，这会导致一定的安全隐患。文中将排队模型理论引入到 NAT 技术的应用中，具有一定的现实意义。更进一步的工作是研究网络地址转换与各种特定协议如何融合的相关问题。

参考文献：

[1] Egevang K, Francis P. The IP Network Address Translator (NAT)[S]. RFC 1631,1994.

[2] 刘 刚. 网络地址转换技术的研究 [J]. 网络安全技术与应用,2008(1):24-25.

[3] 邹 航. NAT 网络地址转换技术分析 [J]. 重庆工学院学报 (自然科学版),2007,21(4):89-91.

[4] 何宝宏. 浅析 NAT 的类型[J]. 电信网技术,2004(8):427-430.

[5] Forouzan B A. Data Communications and Networking[M]. 北京:机械工业出版社,2003:198-203.

[6] Comer D E. Internetworking with TCP/IP[M]. [s. l.]:Prentice-Hall Internationl, Inc. ,1995.

[7] 徐千洋. Linux C 函数库参考手册 [M]. 北京:中国青年出版社,2002.

[8] 王 南, 孙保锁, 王月平. P2PSIP 系统中 NAT 穿越方案的研究与设计[J]. 计算机技术与发展,2009,19(10):66-69.

[9] 朱 宁,许晓东. 基于 PC 架构 Linux NAT 性能优化[J]. 计算机工程与应用,2004(20):168-170.

[10] 徐 群, 祝永志. 集群系统中的负载均衡问题的研究[J]. 计算机技术与发展,2009,19(8):129-132.

[11] 危 达,李建华,李 曼,等. 服务请求排队算法[J]. 计算机系统应用,2009(8):51-52.

[12] Takagi H. Time-dependent analysis of M/G/1 vacation models with exhaustive service [J]. Queuing Systems,1990(6):369-390.