基于负载均衡机制的防火墙技术研究与实现

专　　业：计算机应用技术
硕 士 生：赵　征
指导教师：马光思　教授

摘要

网络安全始终是计算机科学技术领域引人注目的重大研究课题。防火墙作为互联网络安全必需的基础设备，其技术在过去近十年里经历了不断的完善和更新。在对防火墙一直追求的安全、灵活、可用等性能指标中，高可用性始终是防火墙技术发展的主要方向。

本文作者以国家“十五”科技攻关项目：“银行信息系统安全保密平台” 的子课题 ——“基于负载均衡机制的防火墙技术研究”为背景，开展了大量的研发工作，在理论和实践方面都获得了很大收益。

结合实际课题，论文概要论述了防火墙的基本概念、特点、分类、关键技术及其发展。按照把负载均衡机制用于防火墙的策略，探讨了负载均衡的概念、传输链路聚合、交换技术，并按照防火墙系统的体系结构，详细介绍了 LVS 集群策略和各种均衡算法。

论文进而从设计目标出发，根据作者在项目研发中所取得的成果，全面阐述了设计思路，系统平台的构建，采用的核心技术，并以此为基础深入讨论了服务负载均衡模块和双机热备份模块的设计和实现方法，其中包括 DNAT 技术、LC(最少连接)算法、心跳检测技术、同步配置策略。

尽管加入服务负载均衡模块和双击热备份模块大大提高了防火墙系统的性能，但没改变防火墙单点接入的现状。为了更好地改善整个系统性能，在分布式防火墙和防火墙集群的基础上，文给出了防火墙集群系统的开发原型，指出了可能存在的问题、实现的难点及未来的研究方向。

键词： 防火墙；负载均衡；地址转换；双机热备份；防火墙集群

RESEARCH & IMPLEMENTATION OF LOADING BALANCE ON FIREWALL

Specialty: Computer Application Technology

Name: Zhao Zheng

Instructor: Prof. Ma Guangsi

Abstract

Network security was an important issue in Computer Science. Firewall, as the basic structure on the network security, have improved and updated in the recent ten years. In the characters of firewall, such as security, flexibility and availability, high availability is a target in the future of firewall.

Research of loading balance on firewall is a sub-project of researching and implementing on the security platform of banking information system, a key state-level technology development project in the Tenth Five Year Plan period. Based on this project, we do lots of research, thus benefit from both theory and practice.

In this paper, we gave the definition, character, class, key technology of firewall and its development. According to loading balance on firewall, we described loading balance's conception, transport link polymerization, exchange technology, especially LVS cluster and balance algorithms.

From the design object of this sub-project, we expound the design thinking, system structure and core technology. And based on this, we address the design and method of Servers Loading Balance module and Hot Standby module, including DNAT, LC algorithms, heartbeat inspection, and synchronization policy.

Although we have improved performance of firewall using modules of Servers Loading Balance and Hot Standby, we can't settle single entry point of firewall. In order to more improve firewall performance, after researching and analyzing distributed firewall and firewall cluster, we present the development prototype of firewall cluster system, difficulties.

At last, we summarize of our design and purpose the development direction of future firewall.

Keyword: firewall, loading balance, NAT, hot standby, firewall cluster

声　明

本人郑重声明我所呈交的论文是我个人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含本人或其他人在其它单位已申请学位或为其它用途使用过的成果。与我一同工作的同志对本研究所做的所有贡献均已在论文中作了明确的说明并表示了致谢。

申请学位论文与资料若有不实之处，本人承担一切相关责任。

论文作者签名：赵征　日期：2003.3

关于论文使用授权的说明

本人完全了解西安建筑科技大学有关保留、使用学位论文的规定，即：学校有权保留送交论文的复印件，允许论文被查阅和借阅；学校可以公布论文的全部或部分内容，可以采用影印、缩印或者其它复制手段保存论文。

（保密的论文在论文解密后应遵守此规定）

论文作者签名：赵征　导师签名：[illegible]　日期：3.18

注：请将此页附在论文首页。

1 绪论

1.1 研究背景

随着 Internet 在世界范围内的日益普及，网络资源的安全问题变得越来越重要。1988 年 11 月发生的“蠕虫”事件，使人们更加意识到计算机网络的安全问题同网络的其他技术一样应该作为今后计算机网络发展必须研究的重大课题。1996 年美国国防部宣布其计算机网络系统在该年内曾遭受到非法用户的攻击多达 25 万次，而且这些攻击中有 2/3 获得成功，尽管大多数攻击未造成损失，但毕竟是对计算机网络系统的潜在威胁[1]。

针对各种侵袭，采取什么手段进行防备？人们可以选择各种各样的安全模式或手段，范围从完全没有安全、“模棱两可的安全”、主机安全，到网络安全。相关安全设施中，防火墙作为非常有效的网络安全模型，对防止因特网的威胁传播到内部网络起到了非常重要的作用。

从第一台防火墙诞生至今，已经有十多年了。在这十多年中，防火墙经历了体系、结构和技术等多方面的变革。第一代防火墙技术几乎与路由器同时出现，采用了包过滤（Packet filter）技术。1989 年，贝尔实验室的 Dave Presotto 和 Howard Trickey 推出了第二代防火墙，即电路层防火墙，同时提出了第三代防火墙——应用层防火墙（代理防火墙）的初步结构。1992 年，USC 信息科学院的 BobBraden 开发出了基于动态包过滤（Dynamic packet filter）技术的第四代防火墙，后来演变为目前所说的状态检测（Stateful inspection）技术。1994 年，以色列的 CheckPoint 公司开发出了第一个基于这种技术的商业化的产品。

防火墙技术，作为一种网络安全技术在政府、金融、电信、军事等领域得到广泛应用。特别是金融领域，帐户的安全性不仅关系着客户利益，更关系着国家金融体系的稳定。自“八五”以来，我国政府和金融业自身日渐重视金融信息安全。“十五”期间，由中国人民银行连同国内各商业银行共同提出的“银行信息系统安全保密平台”被列为国家“十五”科技攻关项目。该项目由武汉人行承担并进行应用示范，亿阳信通作为合作单位参与开发。本人在亿阳信通实习期间有幸参与了此项目组的一些工作。

武汉人行信息专网，是基于 ATM 技术和 Internet 技术的宽带多媒体综合信息网络系统，所有的应用系统都建立在这个统一的公共通信网络平台。信息网除提供信息交换网络平台，同时也要为网络互联和信息交换提供相应的安全机制和安全平台。信息网的安全需求是全方位的、整体的，而网络安全是分层次的，需要在不同层面解决不同的安全问题。主要包括网络层安全、应用层安全以及安全检测和报警。

该项目中，网络层安全主要解决网络互联过程和网络通讯层安全问题，主要包括：（1）网络进出控制；（2）网络和链路层数据加密；（3）对外通信的管理控制和计费。在所构建安全网

络结构中，可将整个网络划分为非安全区、停火区和安全区。在非安全区中放置公开的不需要任何安全措施的信息；停火区为内部网用户提供对外访问的连接和管理，以及整个网络管理；安全区内为所有重要的服务器。由于内部和外部用户都能到达停火区，所以它提供对内和对外的各种服务，负责传递或代理内外相互之间的访问。并在停火区内部针对应用服务进行细粒度访问控制，对客户和服务器双方进行身份验证。同时对安全区中服务器的服务提供代理，实施第一层次的安全保护。这里要求在停火区和非安全区的出入口处设置防火墙。从而可以看出，网络层安全主要是通过防火墙产品解决划分网络结构，管理和控制内部与外部通信。通过防火墙上的 NAT 或 Proxy 功能可以实现内部用户对外部的间接访问。目前流行的防火墙也设定一定的流量记录和计费功能。

作为网络层安全的关键环节，项目对防火墙提出了更高的要求。对于停火区和安全区的集群服务器，不再采用 DNS 实现负载均衡。而是直接利用防火墙，作为集中式调度器，解决在多个服务器结点间均衡分配用户请求响应。同时，由于传统防火墙始终采用单点接入网络，其数据流量和计算强度之大，使得单一设备无法承担。一旦防火墙崩溃，整个网络将陷入瘫痪。为保证防火墙始终有效，对防火墙构建双机热备份模块。

为更好地实现项目要求，项目组提出“基于负载均衡机制的防火墙技术研究”课题并将它列为“银行信息系统安全保密平台”的子项目。

1.2 作者在系统开发中所承担的工作

对国家“十五”科技攻关项目——“银行信息系统安全保密平台”的子项目，作者主要负责承担研究任务有：（1）利用服务集群，实现防火墙上的服务负载均衡；（2）防火墙实现双机热备份，保证任何时间防火墙都能够正常工作；（3）为进一步提高防火墙的可用性，分析国外目前研究情况，从理论上给出防火墙集群模型。

1.3 本文组织结构

第一章为绪论，给出了课题的研究背景以及作者所承担的工作。第二章为防火墙的基础知识。主要介绍防火墙的概念、特点、技术发展现状和防火墙的关键技术。第三章为负载均衡机制。从防火墙中负载均衡机制的提出谈起，给出了负载均衡的概念、策略，特别强调 LVS 集群技术。第四章为基于负载均衡防火墙的设计与实现。在总体设计中给出需求分析、系统体系结构、关键技术和性能目标。紧接着分别详细介绍了服务负载均衡和双机热备份两个模块的设计和实现。第五章为进一步设想。为更好地解决防火墙单点失效问题，目前国外提出了两种方法——分布式防火墙和多台防火墙集群系统。根据实际情况，作者给出了一种集群系统的实现模型。

2 防火墙及其关键技术

2.1 防火墙的定义

防火墙是一种网络安全技术，它最初被定义为实施某些安全策略，保护可信网络，用以防止来自不可信网络攻击的设施。但随着人们意识到威胁不仅来自外部网络，还可能来自内部网络，并且技术上也有可能实施更多的解决方案时，现在的防火墙逐渐发展为在两个网络之间强制实施访问控制策略的一个系统或一组系统。

简单地说，防火墙就是用来保护可信网络不受非可信网络侵入的一种机制，并且允许在两个网络之间的通信。它应具有的功能大致包括：

- 拒绝未经授权的用户访问；
- 阻止未经授权的用户存取敏感数据；
- 允许合法用户不受妨碍地访问网络资源。

2.2 防火墙的体系结构

防火墙可以有不同的结构，如双重宿主主机体系结构、被屏蔽主机体系结构和被屏蔽子网体系结构[2,3,4]。

2.2.1 双重宿主主机体系结构

双重宿主主机是一个被取消路由功能的主机，与双重宿主主机相连的外部网络与内部网络之间在网络层是断开的。这样做的目的是使外部网络无法了解内部网络的拓扑。

双重宿主主机至少有两个网络接口。这种主机可以充当与这些接口相连的网络之间的路由器。它能从一个网络向另一个网络发送 IP 数据包，然而双重宿主主机的防火墙体系结构禁止这种发送。因此，IP 数据包并不是从一个网络直接发送到另一个网络。外部网络与双重宿主主机通信，内部网络也能与双重宿主主机通信。但是外部网络与内部网络不能直接通信，它们之间的通信必须经过双重宿主主机的过滤和控制。

双重宿主主机防火墙的体系结构相当简单，它位于内部网络和外部网络之间。图 2-1 显示了这种体系结构。

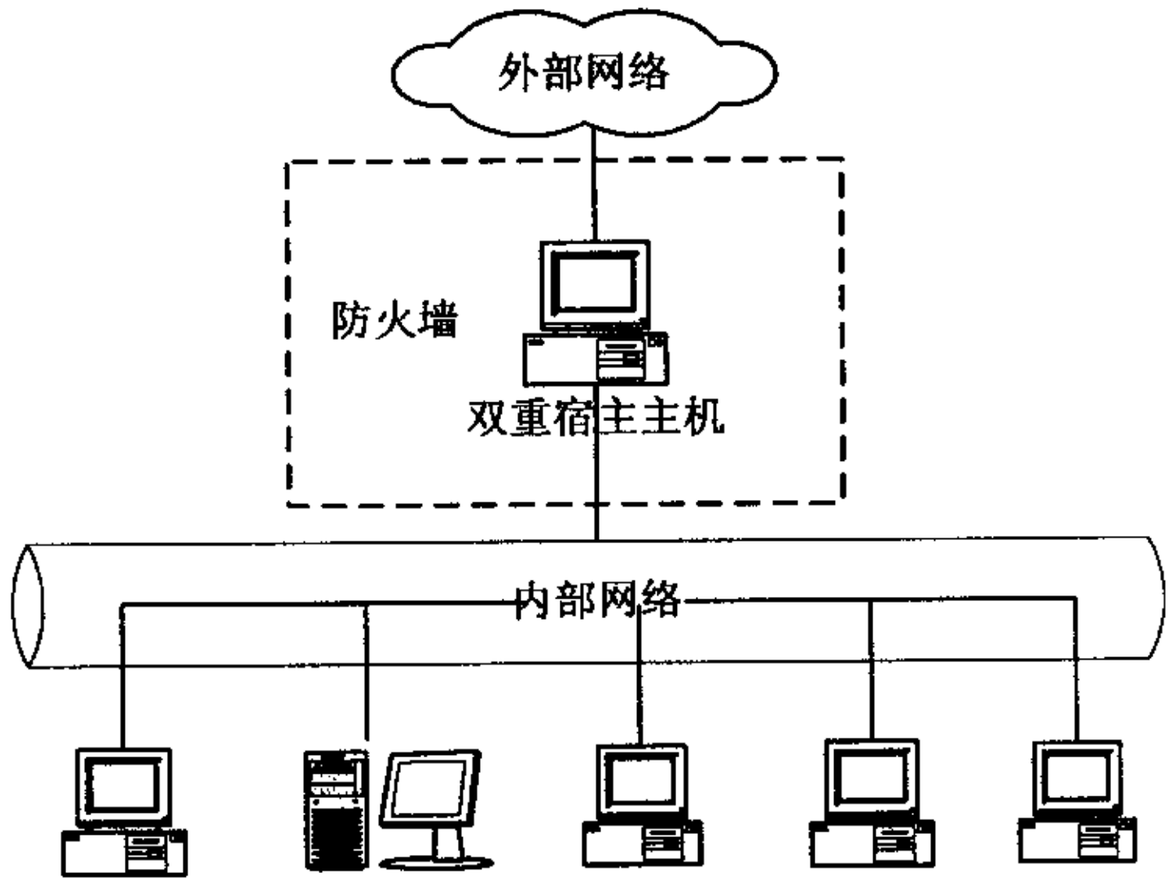


图2-1 双重宿主主机体系结构防火墙

双重宿主主机体系结构防火墙只能以代理的方式提供服务，可用于维护系统日志、硬件拷贝日志和远程日志。这对日后检查很有帮助，但不能帮助网络管理者确认内部网中哪些主机可能已被黑客入侵。

采用这种体系结构的致命弱点是：一旦入侵者侵入双重宿主主机并使其只具有路由功能，则任何网上用户均可随便访问内部网。例如，若双重宿主主机的操作系统是 UNIX，系统管理者可以修改核心变量 ipforwarding 来禁止 TCP/IP 转发。但熟知该操作系统的黑客设法取得系统特权后，就可以重设该变量使其具有路由功能。

2.2.2 屏蔽主机体系结构

屏蔽主机体系结构防火墙使用单独一个路由器把内部网络和外部网络隔开，提供服务的主机只连接内部网络。这种体系结构的防火墙是由路由器和堡垒主机组成，主要的安全由数据包过滤保证。图 2-2 给出这种防火墙的体系结构。

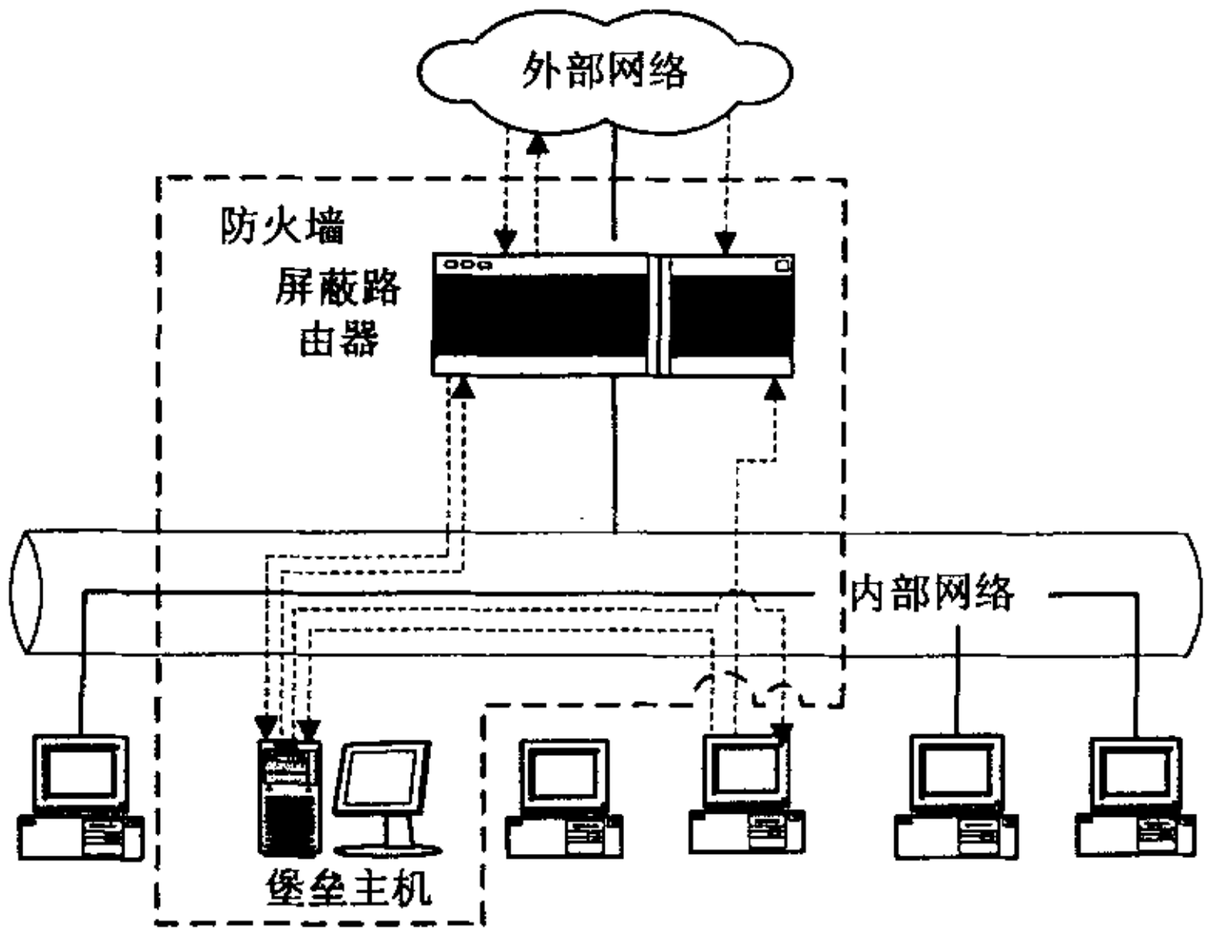


图2-2 屏蔽主机体系结构防火墙

这种体系结构涉及到堡垒主机。它是 Internet 上的主机能连接到的唯一的内部网络上的系统。任何外部系统要访问内部的系统或服务都必须先连接到这台主机上。因此堡垒主机需要保持更高等级的主机安全。一旦攻击者设法侵入到堡垒主机，而且在堡垒主机和其余的内部主机之间没有任何网络安全措施存在的情况下，路由器会出现一个单点失败。如果路由器被损害，整个网络将对侵袭者就全面开放。

2.2.3 被屏蔽子网体系结构

被屏蔽子网体系结构增加额外的安全层到被屏蔽的主机体系结构中，即通过添加虚拟的内部网络更进一步地把内部网络与外部网络隔开。

堡垒主机是用户网络上最容易受到侵袭的主体。通过虚拟的内部网络隔离堡垒主机，能减少堡垒主机被侵入的影响。可以说，它只给入侵者一些无关紧要的访问机会，但不是全部。在这种结构中，外部网络的侵袭者通过了第一道防火墙，他所看到的是一个虚拟的内部网络和堡垒主机，在这个虚拟的环境中，他看到的是知识假象。被屏蔽子网体系结构的最简单形式是，防火墙为两个屏蔽路由器，每个都连接到虚拟的内部网络即周边网络，一个位于周边网与内部网之间，另一个位于周边网与外部网之间，其结构如图 2-3 所示。

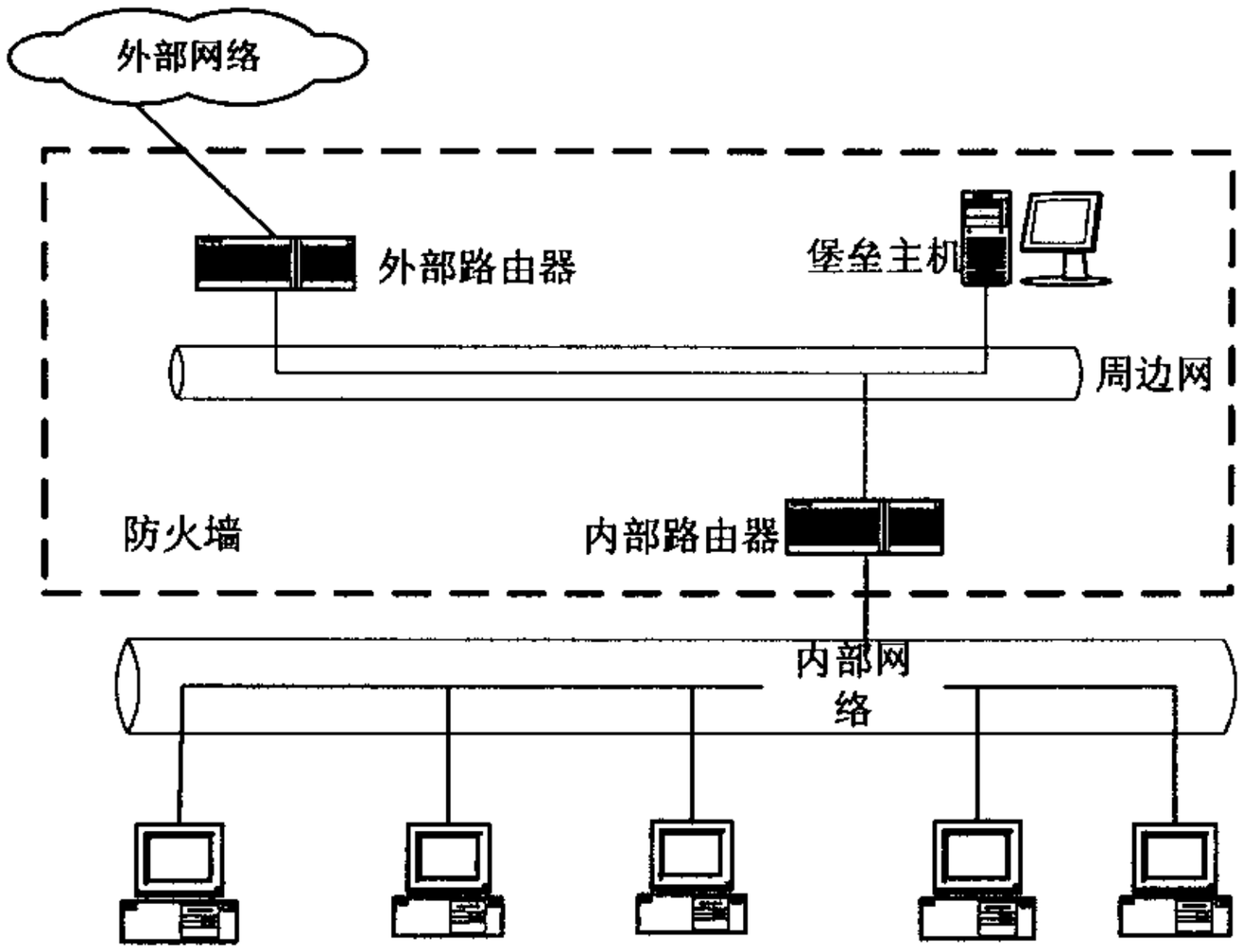


图2-3 屏蔽子网体系结构防火墙

如果攻击者试图完全破坏防火墙，他必须重新配置连接三个网的路由器。既切断连接，又不能把自己锁在外面，同时又不使自己被发现，这会使黑客的性能变得更困难。

2.3 防火墙关键技术及发展

防火墙技术随着信息技术的迅猛发展，也经历了由简单到复杂、功能不断丰富和强大的过程，本部分将按照防火墙技术的发展历程简要地介绍防火墙中所涉及的关键技术[1,2]。

2.3.1 包过滤

包过滤防火墙，又称筛选路由器。它一般有一个包检查模块，作用在网络层（IP 层），按照一定的安全策略（信息过滤规则）分析进出内部网络的所有信息，并按照一定授权通过。信息过滤规则是以所收到的数据包头信息为基础，比如 IP 数据包源地址、IP 数据包目的地址、封装协议类型（TCP、UDP、ICMP 等）、TCP/IP 源端口号、TCP/IP 目的端口号、ICMP 报文类型等。当一个数据包满足过滤规则，则允许该数据包通过，否则拒绝通过。它相当于数据包所要到达的网络物理上被断开，起到保护内部网络的作用。

包过滤防火墙具有速度快、逻辑简单、成本低、易于安装和使用等优点，并且网络性能和透明度都比较好。其中 CPU 用来处理包过滤的时间可以忽略不计，合法用户在进出网络时，根本感觉不到包过滤的存在。但包过滤技术也存在不足之处：（1）由于该技术的安全控制层次只限于数据包头部，对于恶意的拥塞攻击、内存覆盖攻击或病毒等高层次的攻击手段，则无能为力；（2）包过滤技术只能按照规则取舍数据包而不做日志，缺乏用户级授权；（3）包过滤配

置规则复杂，很难准确地设置包过滤器。

2.3.2 应用代理

代理技术与包过滤技术完全不同，包过滤是在网络层拦截所有的信息流，而代理技术是针对每一个特定应用都有一个程序。应用代理的原理是在应用层实现防火墙功能，即彻底隔断通信两端的直接通信，所有的通信都必须经过应用代理层转发，访问者任何时候都不能与服务器直接建立 TCP 连接，应用层的协议会话过程必须符合代理的安全策略要求。

代理服务器的优点对于用户而言，似乎是直接与外部网络相连的，代理服务器对用户透明。由于代理机制完全阻断了内部网络与外部网络的直接联系，保证了内部网络拓扑结构等重要信息被限制在代理网关内侧，不会外泄，从而减少了黑客攻击时所需的必要信息。另外，应用代理是有状态的，它能够利用扩展信息中由应用程序产生的信息和部分由通讯上下文产生的状态，具有部分信息处理的能力。然而，代理服务器的应用也受到诸多限制：（1）当一项新的应用加入时，如果代理服务程序不予支持，则此应用不能使用；（2）应用代理防火墙需要在一定程度上定制用户系统，这取决于所使用的应用程序，而有些应用程序可能根本不支持代理连接；（3）实现在应用层的应用代理，其性能也比较低。

2.3.3 状态检测

状态检测技术普遍被认为是新一代防火墙的核心技术。状态检测防火墙是由 Check Point 公司率先提出，又称为动态包过滤防火墙。与传统的包过滤技术相比，状态检测对新建的应用连接，检查预先设置的安全规则，允许符合规则的连接通过，并在内存中记录下该连接的相关信息，生成状态检测表。对于该连接的后续数据包，只要符合状态表，就可以通过。

这种方式的好处在于：（1）由于不需要对每个数据包进行规则检查，而是一个连接的后续数据包通过哈希算法，直接进行状态检查，从而使得性能得到了较大提高；（2）由于状态表是动态的，因而可以有选择地，动态地开通 1024 号以上的端口，使得安全性得到进一步地提高；（3）部分状态检测型防火墙还支持多种用户认证方式，提供了应用级的安全认证手段，增加了某些应用代理功能，使得安全控制粒度更为细致。它的缺点在于这种状态检测可能造成网络连接的某种迟滞，需要硬件性能的补充。

2.3.4 地址转换技术

网络地址转换（NAT）[5,6,7]作为防火墙的一个重要功能，它能将内部地址和外部地址进行转换，以使具备内部地址的计算机能访问外部网络。同样，外部网络访问防火墙拥有的某一外

部地址时，防火墙能将其转发到该地址映射的内部地址的计算机上。

NAT 从功能上讲分为源网络地址转换和目的网络地址转换，从实现方法上讲分为静态地址转换和动态地址转换。NAT 技术已经成熟，但是由网络地址转化发展而来的技术——负载均衡、地址伪装以及透明代理，成为大众所关注的研究热点。

2.3.5 其它防火墙技术

除上面介绍的防火墙技术外，一些新的技术正在防火墙产品中采用，主要有：

1．内容检查技术

内容检查技术提供对高层服务协议数据的监控能力，确保用户的安全。包括计算机病毒、恶意的 Java Applet 或 ActiveX 攻击、恶意电子邮件及不健康网页内容的过滤防护。

2．安全审计

绝对的安全是不可能的，因此必须对网络上发生的事件进行记载和分析，对某些保护网络的敏感信息访问保持不间断的记录，并通过各种类型的报表、报警等方式向系统管理人员进行报告。比如在防火墙的控制台上实时显示与安全有关的信息、对用户口令非法、非法访问进行动态跟踪等。

3．身份认证

目前，一般防火墙主要提供三种认证方法：

- 用户认证。防火墙设定可以访问内部网络资源的用户访问权限。
- 客户认证。防火墙提供特定用户端授权用户特定的服务权限。
- 会话认证。防火墙提供通信双方每次通信时透明的会话授权机制。

2.4 小结

本节给出防火墙的定义、主要体系结构和防火墙的关键技术。从以上对防火墙各阶段技术的回顾中也可以看出：包过滤防火墙在网络层提供访问控制，这使得它能够带来高性能和可扩展性，但却是安全性最差的一类防火墙。因为它只检查地址和端口，使得应用层协议的数据很容易被黑客攻破。应用代理防火墙弥补了包过滤的弱点，在应用层实现，能够做复杂的访问控制，并做精细的日志和审核，安全性提高，但应用代理防火墙对每一种协议都需要相应的代理程序，工作量增大，造成性能明显下降。而在此基础上的状态检测防火墙，克服前面两种技术的限制，对每一个连接，建立动态的状态表，不再需要对每个包进行规则检查，性能得到比较大地提高，并且能够从应用层提取与状态有关的信息做出安全决策，安全性能进一步提高[8]。

从安全性角度考虑，防火墙安全性不断提高，但其作为网络进出的唯一扼守点，始终没有改变，使得防火墙单点故障和网络性能瓶颈不能得到根本性改变，这就要求防火墙向高性能发展。它对提高防火墙的性能和可伸缩性方面都有很大作用。在基于负载均衡机制的防火墙技术研究中，主要应用了状态检测技术、网络地址转换和负载均衡技术。

3 负载均衡机制

3.1 防火墙中负载均衡机制的提出

当今计算机技术已进入以网络为中心的计算时代。从上世纪九十年代中期，万维网（World Wide Web）的出现以其简单操作方式将图文并茂的网上信息带给普通大众。到今天，经济、文化、生活都强烈地依赖于网络，网上业务量的发展远远超出了人们的估计。这促进 Internet 用户剧烈增长和 Internet 流量爆炸式地增长，特别是网络的核心部分，其数据流量和计算强度之大，使得单一设备根本无法承担。如何在完成同样功能的多个设备之间实现合理的业务量分配，改变一些设备过忙影响效率、甚或导致宕机，而别的设备却处于闲置，不能充分发挥作用的现象，就成了急需解决的问题。负载均衡机制因此应运而生。

防火墙，位于内部网和外部网的交界点上，所有客户的连接请求都必须经由防火墙才能访问内部网络的服务器。因而可以通过增加防火墙功能，建立服务器集群系统，将客户端的用户请求均衡地分配到集群系统的各个设备上。另外，在网络中，单台防火墙本身也会造成网络瓶颈。为了解决该问题，在并行多台防火墙方案中，提出了集群防火墙。这也是未来防火墙的发展方向。

3.2 负载均衡机制的基本概念

3.2.1 负载均衡机制的定义

负载均衡是一种策略，它使得多台服务器或多条链路共同承担一些繁重的计算或 I/O 任务，从而以较低成本消除网络瓶颈，提高网络的灵活性和可靠性。它处理的主要任务有：解决网络拥塞问题，就近提供服务，实现位置无关性；为用户提供更好的访问质量；提高服务器响应速度；提高服务器及其他资源的利用率；避免网络关键部位出现单点失效等[9]。

3.2.2 负载均衡机制的分类

负载均衡机制包括两种：一种是静态负载均衡，一种是动态负载均衡[9]。静态负载均衡采取一劳永逸的方式把任务分配给处理器，要求预先知道程序状态，设备性能，根据系统的先验知识做出决策，但是在运行时负载不能重新分配。相反，动态负载均衡很少假设任务执行时间或通信延迟等运行时的先验信息。由于系统根据当前的负载状况来调整任务划分，重分配进程，所以更好地动态使用了全系统的所有资源，提高了系统的性能。然而，负载是有代价的，它主要来自三

方面：（1）处理器间传播的负载信息；(2)任务转移前的任务选择的决定；（3）任务移植的通信延迟。因而，在实现动态负载均衡时要采取有效的策略降低负载分配的代价。

3.3 负载均衡的策略

对一个网络的负载均衡应用，可以从网络的不同层次入手，具体情况要看对网络瓶颈所在之处的具体分析，大体上不外乎从传输链路聚合、采用更高层网络交换技术和设置服务器集群策略三个角度实现[10]。

3.3.1 传输链路聚合

链路聚合技术，将多个线路的传输容量融合成一个单一的逻辑连接。当原有的线路满足不了需求，而单一线路的升级又太昂贵或难以实现时，就要采用多线路的解决方案了。目前有 4 种链路聚合技术可以将多条线路“捆绑”起来。同步 IMUX 系统工作在 T1/E1 的比特层，利用多个同步的 DS1 信道传输数据，来实现负载均衡。IMA 是另外一种多线路的反向多路复用技术，工作在信元级，能够运行在使用 ATM 路由器的平台上。用路由器来实现多线路是一种流行的链路聚合技术，路由器可以根据已知的目的地址的缓冲（cache）大小，将分组分配给各个平行的链路，也可以采用循环分配的方法来向线路分发分组。多重链路 PPP，又称 MP 或 MLP，是应用于使用 PPP 封装数据链路的路由器负载平衡技术。MP 可以将大的 PPP 数据包分解成小的数据段，再将其分发给平行的多个线路，还可以根据当前的链路利用率来动态地分配拨号线路。这样尽管速度很慢，因为数据包分段和附加的缓冲都增加时延，但可以在低速线路上运行得很好。

链路聚合系统增加了网络的复杂性，但也提高了网络的可靠性，使人们可以在服务器等关键 LAN 段的线路上采用冗余路由。对于 IP 系统，可以考虑采用 VRRP（虚拟路由冗余协议）。VRRP 可以生成一个虚拟缺省的网关地址，当主路由器无法接通时，备用路由器就会采用这个地址，使 LAN 通信得以继续。总之，当主要线路的性能必需提高而单条线路的升级又不可行时，采用链路聚合技术[10]不失为一种行之有效的方案。

3.3.2 交换技术

在 IP 层之上的交换一般可分为四层交换和 5~7 层交换。现在许多交换机提供第四层交换功能，其交换依据包括源/目的 IP 地址和源/目的端口号，可以将一个外部 IP 地址映射为多个内部 IP 地址，对每次 TCP 连接请求动态使用其中一个内部地址，达到负载均衡的目的。而五~七层交

换则为应用层交换，其交换依据包括 HTTP 的 URL、头字段等。通常又称为内容交换技术。

内容交换技术提供了一种对访问流量的高层控制方式。通常需要由服务器组、虚拟 IP 地址和内容规则构成。检查所有的 HTTP 报头，提取报头内的信息在内容规则库中匹配，然后根据负载均衡算法在服务器组中选定一个服务器，待服务器成功响应连接后，内容交换机就可获得客户与服务器的一个 TCP 的全部信息。图 3-1 给出内容交换设备的运行过程[11]。

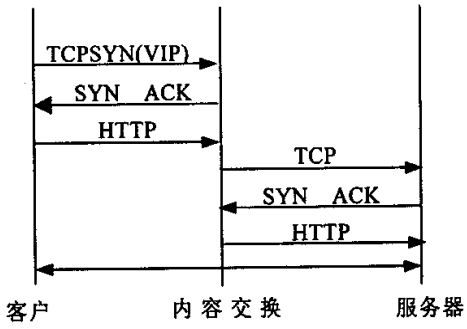


图 3-1 内容交换设备运行过程

目前，交换技术主要是以集成硬件的方式设计，实施成本高。特别是内容交换技术所能支持的标准和规则的数目有限，其采用的标准和规则的灵活性也有限。它还受到能够同时开启的 TCP 连接数量以及 TCP 连接的建立和断开比率的限制。而且内容交换技术还会占用大量的系统资源（包括内存占用和处理器占用）。另外，由于灵活性差，内容交换很难与其他网络设备集成，比如防火墙或者应用网关。所以，选用交换机解决网络负载均衡问题必须认真考虑投入与回报[12]。

3.3.3 集群策略

所谓集群系统是一种并行或分布式处理系统，它就像把很多连接在一起的独立计算机集成为一个单独的计算资源进行协同工作[13,14]。集群系统具有高性能、可扩展性、高吞吐量和易用性等特点，应用它可达到共享和高效利用资源的目的，并提供大型运算、均衡分配请求压力以及故障恢复的能力。

3.3.3.1 LVS 集群的通用体系结构

LVS 集群采用 IP 负载均衡技术和基于内容请求分发技术。调度器具有很好的吞吐率，将请求均衡地转移到不同的服务器上执行，且调度器自动屏蔽掉服务器的故障，从而将一组服务器构成一个高性能的、高可用的虚拟服务器。整个服务器集群的结构对客户是透明的，而且无需修改客户端和服务器端的程序。为此，在设计时需要考虑系统的透明性、可伸缩性、高可用性和易管

理性。一般来说，LVS 集群采用三层结构，其体系结构如图 3-2 所示，三层主要组成部分为：[13,14]

- 负载调度器（load balancer），它是整个集群对外的前端机，负责将客户请求发送到一组服务器上执行，而客户认为服务是来自同一个 IP 地址（常可称之为虚拟 IP 地址）。
- 服务器池（server pool），是真正执行客户请求的一组服务器，执行的服务有 WEB、MAIL、FTP 和 DNS 等。
- 共享存储（shared storage），它为服务器池提供一个共享的存储区，这样很容易使得服务器池拥有相同的内容，提供相同的服务。

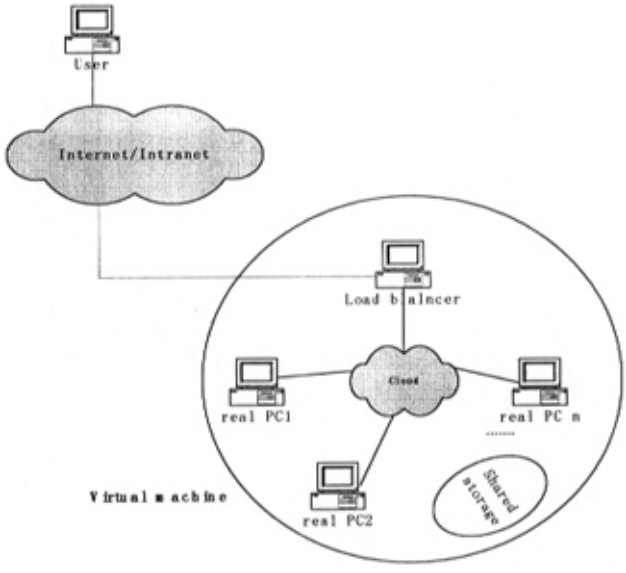


图 3-2　LVS 集群体系结

调度器是服务器集群系统的唯一入口点（Single Entry Point），它可以采用 IP 负载均衡技术、基于内容请求分发技术或二者相结合。在 IP 负载均衡技术中，需要服务器池拥有相同的内容提供相同的服务。当客户请求到达时，调度器只根据服务器负载情况和设定的调度算法从服务器池中选出一个服务器，将该请求转发到选出的服务器，并记录这个调度；当这个请求的其他报文到达，也会被转发到前面选出的服务器。在基于内容请求分发技术中，服务器可以提供不同的服务，当客户请求到达时，调度器可根据请求的内容选择服务器执行请求。因为所有的操作都是在 Linux 操作系统核心空间中完成的，其调度开销很小，所以它具有很高的吞吐率。

服务器池的结点数目是可变的。当整个系统收到的负载超过目前所有结点的处理能力时，可以在服务器池中增加服务器来满足不断增长的负载请求。对大多数网络服务来说，请求间不存在很强的相关性，请求可以在不同的结点上并行执行，所以整个系统的性能基本上可以随着服务器池的结点数目增加而线性增长。

共享存储通常是数据库、网络文件系统或者分布式文件系统。服务器结点需要动态更新的数

据一般存储在数据库系统中，同时数据库会保证并发访问时数据的一致性。静态的数据可以存储在网络文件系统（如NFS/CIFS）中，但网络文件系统的伸缩能力有限，一般来说，NFS/CIFS服务器只能支持3~6个繁忙的服务器结点。对于规模较大的集群系统，可以考虑用分布式文件系统，如AFS[15]、GFS[16,17]、Coda[18]和Intermezzo[19]等。分布式文件系统可为各服务器提供共享的存储区，它们访问分布式文件系统就像访问本地文件系统一样，同时分布式文件系统可提供良好的伸缩性和可用性。此外，当不同服务器上的应用程序同时读写访问分布式文件系统上同一资源时，应用程序的访问冲突需要消解才能使得资源处于一致状态。这需要一个分布式锁管理器（Distributed Lock Manager），它可能是由分布式文件系统内部提供的，也可能是外部提供的。开发者在写应用程序时，可以使用分布式锁管理器来保证应用程序在不同结点上并发访问的一致性。

负载调度器、服务器池和共享存储系统通过高速网络相连接，如100Mbps交换网络、Myrinet和Gigabit网络等。使用高速的网络，主要为避免当系统规模扩大时互联网络成为整个系统的瓶颈。

一般来说，调度器的可靠性较高，因为调度器上运行的程序较少而且大部分程序早已遍历过，但往往不能排除硬件老化、网络线路或者人为误操作等主要故障。为了避免调度器失效而导致整个系统瘫痪，需要把一个从调度器设立为主调度器的备份。两个心跳（Heartbeat）进程[6]分别在主、从调度器上运行，它们通过串口线和UDP等心跳线来相互定时地汇报各自的健康状况。当从调度器不能听得主调度器的心跳时，从调度器通过ARP欺骗（Gratuitous ARP）来接管集群对外的Virtual IP Address，同时接管主调度器的工作来提供负载调度服务。当主调度器恢复时，这里有两种方法，一是主调度器自动变成从调度器，二是从调度器释放Virtual IP Address，主调度器收回Virtual IP Address并提供负载调度服务。这里，多条心跳线可以使得因心跳线故障导致误判（即从调度器认为主调度器已经失效，其实主调度器还在正常工作）的概率降到最低。

通常，当主调度器失效时，主调度器上所有已建立连接的状态信息将丢失，已有的连接会中断。客户需要向重新连接，从调度器才会将新连接调度到各个服务器上，这对客户会造成一定的不便。为此，IPVS调度器在Linux内核中实现一种高效状态同步机制，将主调度器的状态信息及时地同步到从调度器。当从调度器接管时，绝大部分已建立的连接会持续下去。

3.3.3.2 IP负载均衡技术

在集群服务器策略里调度器通常是由IP负载均衡技术实现的。从图3-2中可以看到，一个集群由一个或两个路由节点和很多数量的机器组成。其中真实机器通过内部网络相连。路由节点同时连接着内部和外部网络。在同一时刻，仅有一个路由是激活的。激活的路由扮演的角色是将虚拟机器上的请求重定向到真实机器上。激活的路由节点通过相互交换“我活着的”（I am live）心

跳信息，动态地监视真实机器的健康情况和每个机器的工作量。如果有一个真实机器无效，工作的路由将停止向这台机器发送任务直到它恢复正常。在路由选择方式下，常见的实现方法有：网络地址转换（NAT）、隧道技术（Tunneling）、直接路由（Direct Routing）[14,20]。

(1) 网络地址转换（NAT）

通过网络地址转换，调度器重写请求报文的目标地址，根据预设的调度算法，将请求分派给后端的真实机器。真实机器的响应报文通过调度器时，报文的源地址被重写，再返回给客户，完成整个负载调度过程。这样做的优点是节约 IP 地址，能对内部进行伪装。缺点是效率低，因为返回给请求方的流量经过转换器。

(2) 隧道技术（Tunneling）

调度器把请求报文通过 IP 隧道转发至真实机器，而真实机器将响应直接返回给客户，所以调度器只处理请求报文。由于一般网络服务应答比请求报文大许多，采用这种技术后，集群系统的最大吞吐量可以提高 10 倍。

(3) 直接路由（DR）

直接路由通过改写请求报文的 MAC 地址，将请求发送到真实机器，而真实机器将响应直接返回。同隧道技术一样，直接路由技术可极大地提高集群系统的伸缩性。这种方法没有 IP 隧道的开销，对集群中的真实机器也没有必要支持 IP 隧道协议的要求，但是要求调度器与真实机器都有一块网卡连在同一物理网段上。

从上面集群的实现技术上可以看出服务集群系统具有以下优点：

- 性能：网络服务的工作负载通常是大量相互独立的任务，通过一组服务器分而治之，可以获得很高的整体性能。
- 性能/价格比：组成集群系统的 PC 服务器或 RISC 服务器和标准网络设备因为大规模生产降低成本，价格低，具有最高的性能/价格比。若整体性能随着结点数的增长而接近线形增长，该系统的性能/价格比更接近 PC 服务器。所以，这种松偶合结构比紧偶的多处理器具有更好的性能/价格比。
- 可伸缩性：集群系统中的结点数目可以增加，其伸缩性远超过单台超级计算机。
- 高可用性：在硬件和软件上都有冗余，通过检测软硬件故障，可将故障屏蔽，由存活结点提供服务，可实现高可用性。

3.4 负载均衡算法

从上一节可知，当某个服务请求找到一个虚拟服务器时，会通过某种负载均衡算法规则将请

求重定向到真实服务器上，因此，负载均衡算法的优劣很大程度上影响着集群的性能，针对不同的网络服务需求和服务器配置，常用的算法有以下几种[21,22]：

1．轮询（Round Robin）

“轮询”调度算法将所有物理服务器看作一个有向逻辑环，依次轮流调度逻辑环中的各个物理服务器。当各个物理服务器的计算能力不一样时，“轮询”算法将造成负载不平衡，因为性能较高的服务器分配不到较多的任务。

2．加权轮询（Weighted Round Robin）

“加权轮询”调度算法允许为每个物理器S_i指定一个整数权值W_i(缺省值为 1)，以标记服务器性能（性能较高的服务器具有较高的权值）。假设有n个物理服务器，定义调度周期为$W_i()$。该算法也是依次调度逻辑环上的物理服务器，但它同时保证在一个调度周期内，服务器$S_i()$被调度的次数为W_i。与动态调度算法相比，该算法的调度开销比较小，因此可支持更多的物理服务器；它的缺点是当任务的大小频繁变化时，有可能将大多数长任务调度到同一个物理服务器上，从而导致负载不平衡。

3．最少连接（Least Connections）

“最少连接”调度算法的特点是，物理服务器S_i上的活动连接数C_i越大，其当前的负载越大，假设有n个物理服务器，LC 算法每次都将调度满足条件的物理服务器S_i。显然，该算法是一个动态算法，它可保证不将大多数长任务调度到同一个物理服务器上；如果集群系统的真实服务器具有相近的系统性能，采用"最小连接"调度算法可以较好地均衡负载。其缺点是调度开销较大，特别是当物理服务器的数量很多时。

4．加权最少连接（Weighted Least Connections）

“加权最小连接”调度算法允许为每个物理服务器S_i指定一个整数权值W_i以标记服务器的性能（性能较高的服务器具有较高的权值）。假设有 n 个物理服务器，该算法每次都将调用满足条件$C_j / W_j = \min(C_i / W_i)(1 < i < n)$的物理服务器$S_j$。该算法与 LC 算法相比，增加了额外的除法运算，所以当各个物理服务器具有相同的性能时，最好使用 LC 算法。

5．基于局部性的最少链接（Locality-Based Least Connections）

“基于局部性的最少链接” 调度算法是针对目标 IP 地址的负载均衡，目前主要用于 Cache 集群系统。该算法根据请求的目标 IP 地址找出该目标 IP 地址最近使用的服务器，若该服务器是可用的且没有超载，则将请求发送到该服务器；若服务器不存在，或者该服务器超载且有服务器处于一半的工作负载，则用"最少链接"的原则选出一个可用的服务器，将请求发送到该服务器。

6．带复制的基于局部性最少链接（Locality-Based Least Connections with Replication）

“带复制的基于局部性最少链接”调度算法也是针对目标 IP 地址的负载均衡，目前主要用于 Cache 集群系统。它与 LBLC 算法的不同之处是它要维护从一个目标 IP 地址到一组服务器的

映射，而 LBLC 算法维护从一个目标 IP 地址到一台服务器的映射。该算法根据请求的目标 IP 地址找出该目标 IP 地址对应的服务器组，按"最小连接"原则从服务器组中选出一台服务器，若服务器没有超载，将请求发送到该服务器，若服务器超载；则按"最小连接"原则从这个集群中选出一台服务器，将该服务器加入到服务器组中，将请求发送到该服务器。同时，当该服务器组有一段时间没有被修改，将最忙的服务器从服务器组中删除，以降低复制的程度。

7. 目标地址散列（Destination Hashing）

“目标地址散列”调度算法根据请求的目标 IP 地址，作为散列键（Hash Key）从静态分配的散列表找出对应的服务器，若该服务器是可用的且未超载，则将请求发送到该服务器，否则返回空。

8. 源地址散列（Source Hashing）

“源地址散列”调度算法根据请求的源 IP 地址，作为散列键（Hash Key）从静态分配的散列表找出对应的服务器，若该服务器是可用的且未超载，则将请求发送到该服务器，否则返回空。

3.5 小结

本节论述了负载均衡机制的概念、分类、策略和算法。结合“基于负载均衡机制的防火墙”课题的要求，特别强调了 LVS 集群策略。详细分析了 LVS 群集的体系结构，给出集群策略里调度器的三种实现技术——网络地址转换、隧道技术、直接路由。基于集群策略，负载均衡在防火墙中的应用有两个方面：1）服务器集群；2）防火墙集群。

4 基于负载均衡的防火墙的设计与实现

4.1 总体设计思路

4.1.1 需求分析

今天，Internet 除了可以给大众带来大量的资讯外，也创造了许多全新的网络商业模型。信息服务提供者追求的目标是拥有大量的用户和高访问量。但随着系统使用户数的增多和影响范围的扩大，也出现了许多问题，造成系统服务中断或不稳定。出现的问题主要有如下几方面：

1．系统负载过重

大量的访问将使服务器变得不胜负荷，如果无法及时处理大量的用户服务请求，将出现服务中断的情况。过去，只能采用更强计算能力的服务器来替换原来的服务器。但是高档服务器的价格是随着服务器的性能呈现指数型上升。并且单台服务器的负载能力是有限的，不可能无限扩展。因此，采用多台廉价服务器组成负载分担系统模型日渐成为主流。

2．系统受到黑客攻击

随着一个好的应用系统的用户量增长，该应用系统的影响力也将大大增长。由于 Internet 的开放体系，系统就面临黑客攻击，结果或是系统中的重要数据被偷窃，或是系统被改得面目全非，要不就是被彻底摧毁。因而，如何合理地采用防火墙技术，隐藏服务器细节，就成了 Internet 应用供应商在建设网络中首要考虑的问题。

3．防火墙效率

防火墙安装于网络边界，处于内部网络的唯一出口处。通常是由单台计算机组成，完成所有监控任务。因此，如果因本身网络带宽不够，防火墙建立的连接数目有限，就无法满足过多用户的连接请求；另外，防火墙身兼认证、访问控制、完整性检查等多项任务，限制了防火墙的处理能力，会造成防火墙单点失效[23]。

4.1.2 设计中的几点考虑

1．提供负载分担：众所周知，网络负载均衡技术可提高诸如 Web 服务器、FTP 服务器和其他关键任务服务器上的 Internet 服务器程序的可用性和可伸缩性。因而，这里考虑采用服务器集群技术，通过防火墙上服务负载均衡模块将负载分配给多个服务器连成的集群。

2．解决防火墙单点失效：传统防火墙是单点将内部网接入外部网，容易成为网络瓶颈，

甚至会因防火墙故障造成网络中断。可以有多种解决途径：

- 采用高性能计算机作为防火墙，如千兆防火墙。但是，这种方式不能解决根本问题。例如，当外部主机采用流量攻击时，由于单台防火墙的连接数有限，很容易造成网络拥塞。
- 采用双机热备份。如图 4-1 所示。双机热备份系统是由两台防火墙及双机热备份软件模块组成。它采用主从工作方式，即一台防火墙主机处于活动工作状态，称为活动防火墙。另一台主机为备份机，处于热等待监控状态。活动防火墙出现故障时备用机将主动接管其工作，成为活动防火墙，并且命令原先的活动防火墙放弃占有的资源。尽管双机热备份对防火墙故障作了相应处理，仍然没有解决防火墙的单点接入问题[24]。
- 采用多台防火墙并行工作。提高防火墙系统的总带宽，同时实现负载均衡和容错，图 4-2。

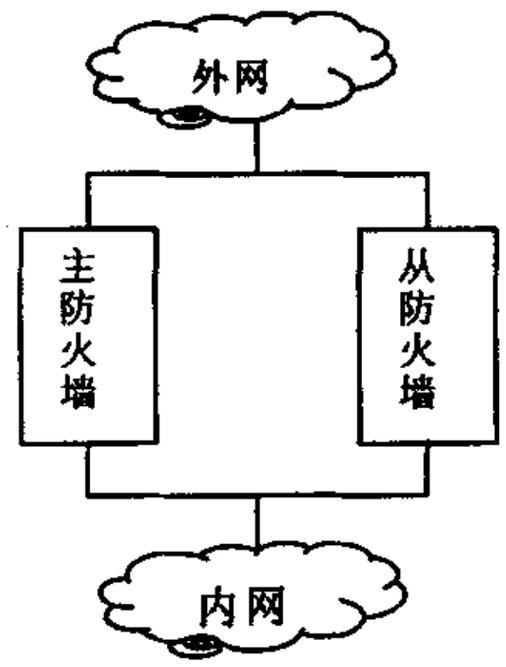


图 4-1 双机热备结构图

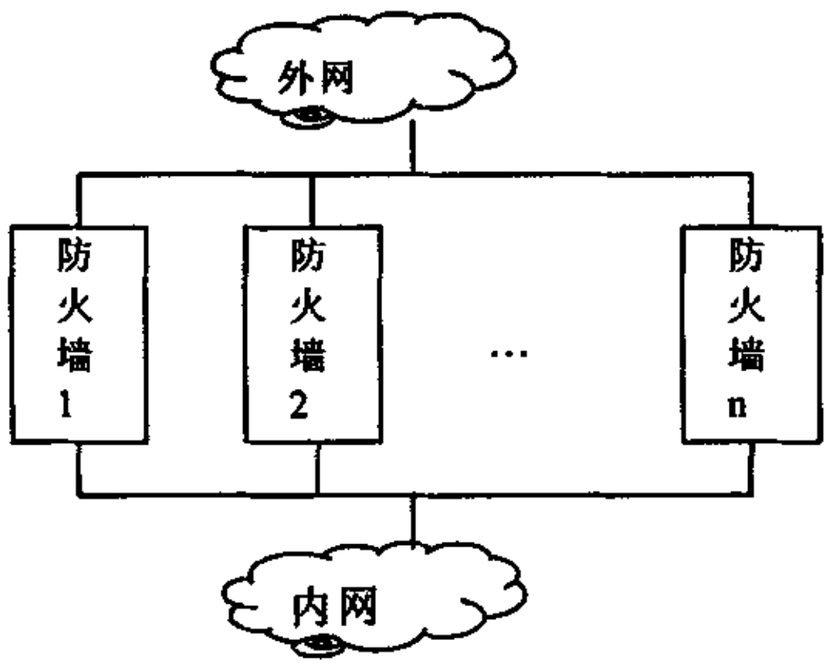


图 4-2 防火墙负载平衡结构图

考虑到系统开发的规模、周期和实验技术环境的局限性等因素，在设计中主要考虑以完成服务负载均衡和双机热备份的实现为目标。

4.1.3 体系结构图

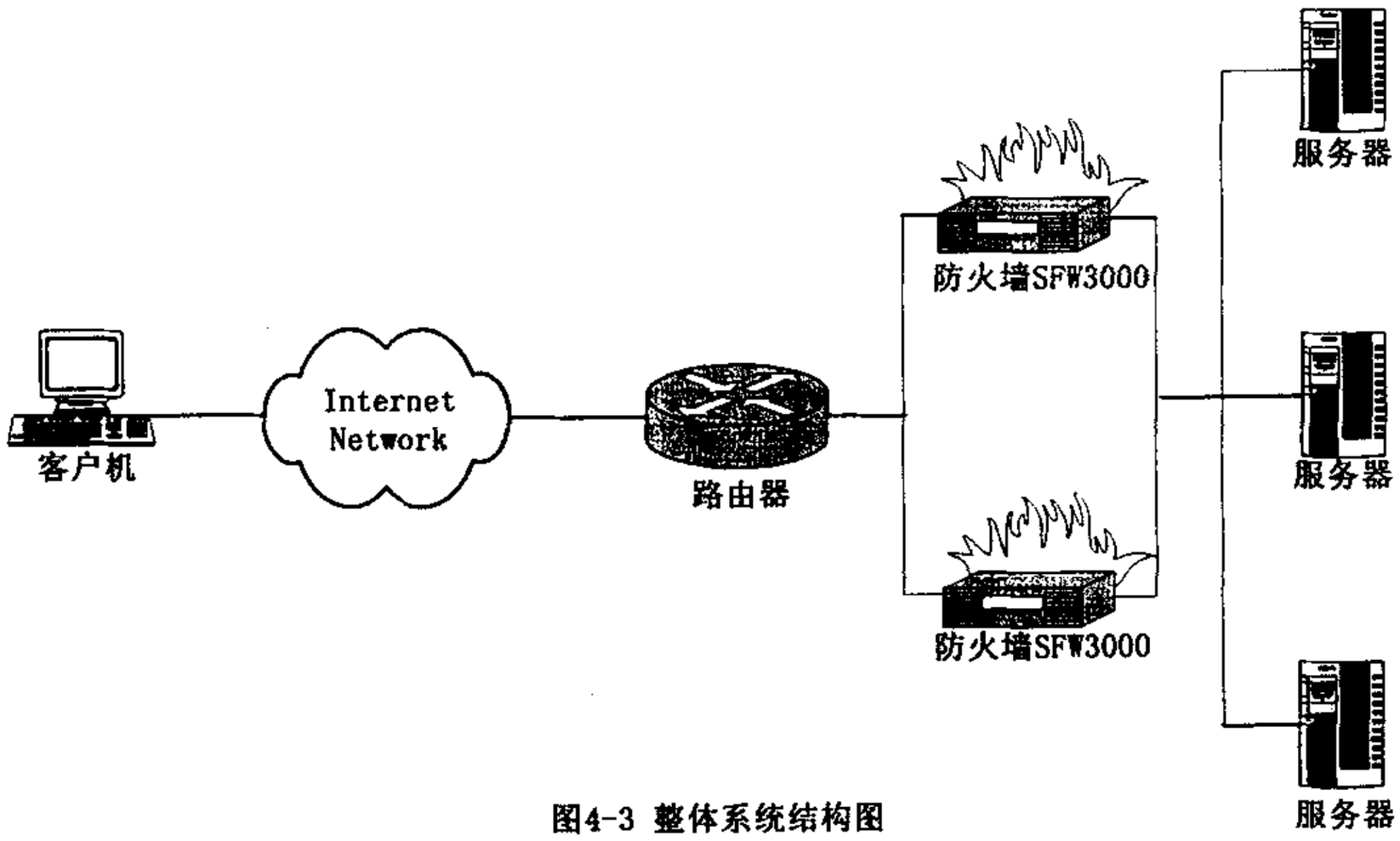


图4-3 整体系统结构图

如图 4-3 可知，为实现具有负载均衡机制的防火墙，首先利用 NAT 技术和集群技术，完成服务映射，实现了基于服务的负载均衡。而后设计防火墙双机热备份。将两台防火墙同时接入到用户网络中，当正在工作的主机发生故障时，备份防火墙可在几秒之内切换到工作状态。

4.1.4 系统运行环境

- 系统运行平台 — 防火墙 BOCO．SFW-3000
- 开发环境 — linux redhat7.2；开发工具 — 编辑软件，GNU C，gdb；开发方法— 结构化设计
- 系统软件和支持软件 — 防火墙 BOCO. SFW-3000 基本系统，主要运行设备型号 — 亿阳网警防火墙 3000
- 系统移植范围 — 仅限于防火墙 BOCO．SFW-3000 及其衍生系统

4.2 系统平台的构建及其关键技术

本系统的运行平台选用亿阳网警防火墙 BOCO．SFW-3000。它采用状态检测包过滤技术。只对新建连接进行规则匹配，可大大加快防火墙的过滤速度，由于只允许处于状态建立

完毕的包通过防火墙，提高了防火墙安全保证功能。并能够根据应用层的内容检测来建立相关状态，分析已知的相应应用层内容，建立相关连接状态，利用相关连接状态动态地决定后继用户相关包的命运，透明地加强了安全保证。

下文详细论述了构建该平台内核的关键技术。它最主要的特点是建立了 Netfilter 抽象框架，定义了5个钩子函数。基于Netfilter框架，防火墙对数据包的处理，不再沿用早先的Ipchains技术，而采用 Iptables 技术，并引入连接跟踪技术实现状态检测的防火墙。

4.2.1 Netfilter

4.2.1.1 Netfilter 框架

Netfilter 是本防火墙基于 Linux2.4.16 实现的框架。它提供了一个抽象、通用化的框架，该框架定义一个子功能实现的就是包过滤子系统。Netfilter 由一系列基于协议栈的钩子组成，这些钩子都对应某一具体的协议。如图 4-4 给出 Ipv4 标准上 Netfilter[25,26,27]。

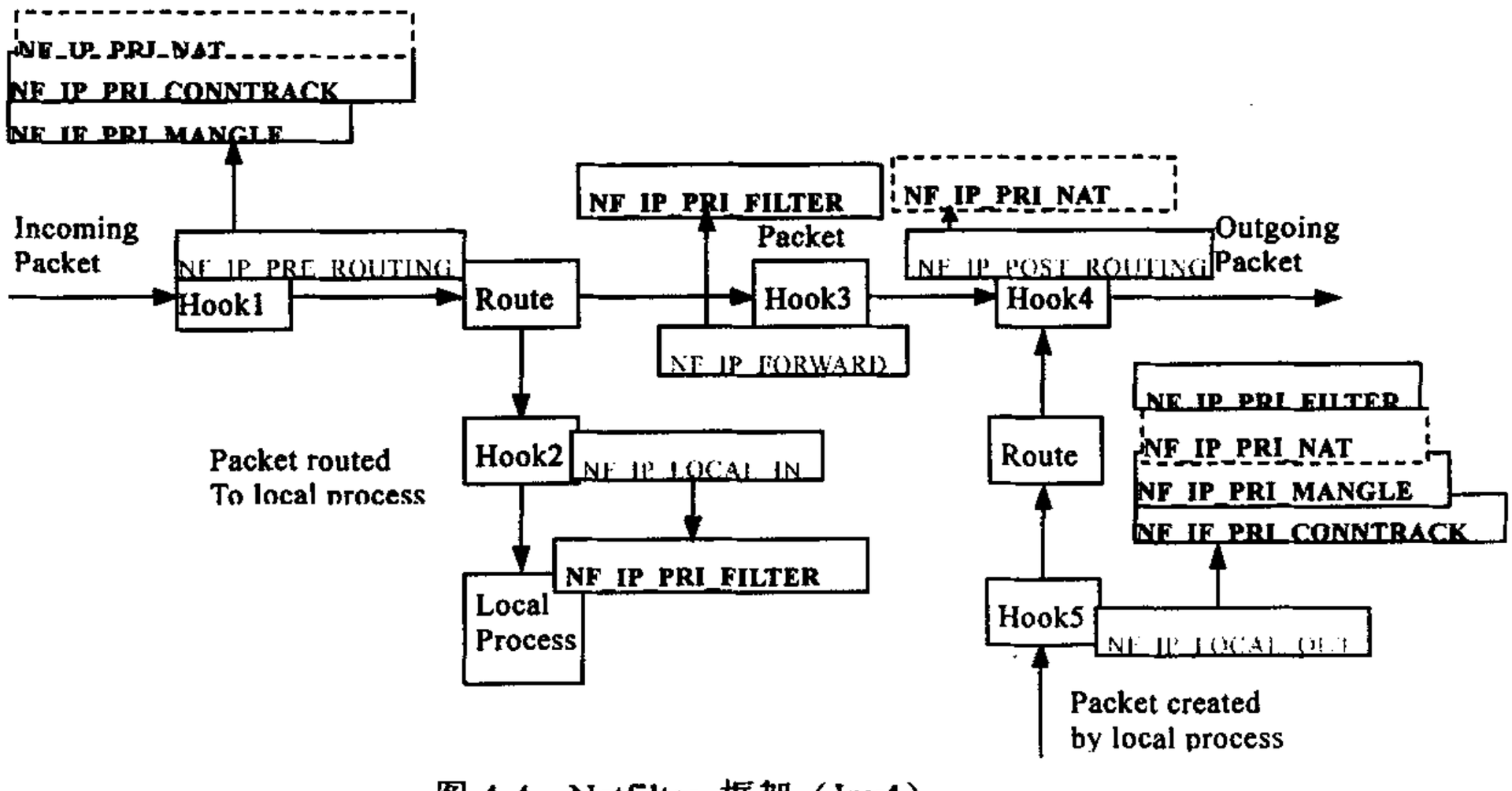


图 4-4　Netfilter 框架（Ipv4）

- NF_IP_PRE_ROUTING：刚刚进入网络层的数据包通过此点（刚刚进行完版本号，校验 和等检测），源地址转换在此点进行；ip_input.c 中 IP_rcv 调用。
- NF_IP_LOCAL_IN：经路由查找后，送往本机的通过此检查点,INPUT 包过滤在此点进行；ip_local_deliver 中调用。

- NF_IP_FORWARD：要转发的包通过此检测点，FORWORD 包过滤在此点进行；
- NF_IP_POST_ROUTING：所有马上便要通过网络设备出去的包通过此检测点，内置的目的地址转换功能在此点进行；
- NF_IP_LOCAL_OUT：本机进程发出的包通过此检测点，OUTPUT 包过滤在此点进行。

这些点是已经在内核中定义好的，内核模块能够注册在这些 HOOK 点进行的处理，可使用 nf_register_hook 函数指定。在数据包经过这些钩子函数时被调用，从而模块可以修改这些数据包，并向 netfilter 返回如下值：

NF_ACCEPT：继续正常传输数据包

NF_DROP：丢弃该数据包，不再传输

NF_STOLEN：模块接管该数据包，不要继续传输该数据包

NF_QUEUE：对该数据包进行排队(通常用于将数据包给用户空间的进程进行处理)

NF_REPEAT: 再次调用该钩子函数

在附录 1 中给出了一个调用例子。

4.2.1 2 Netfilter 框架下规则表结构

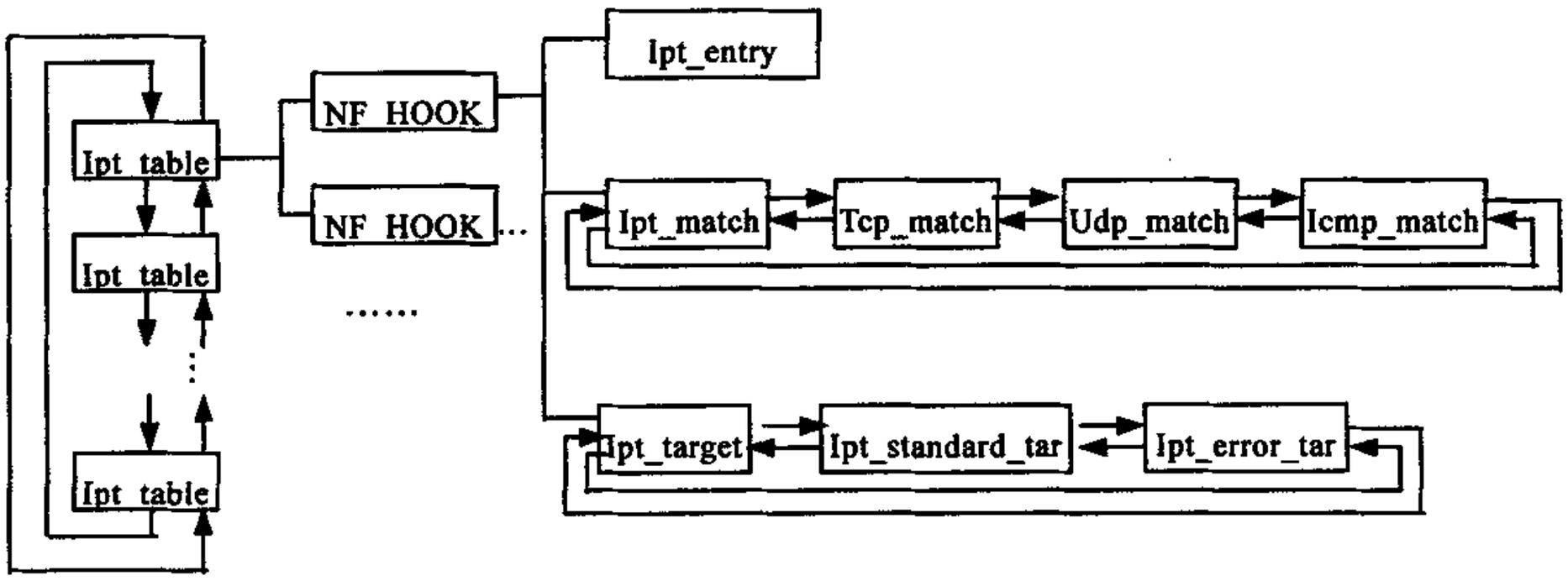


图 4-5　netfilter 框架下规则表的组织结构

以 iptables 组织防火墙规则。如图 4-5。第一级是 table 级，每个防火墙可以有多个 table；第二级是 hook 级，每一个 table 都有一个 hook 集合，每个 hook 都有一个防火墙规则链；第三级基本规则级的规则包括三部分，IP 匹配模板、匹配器和匹配靶。并且这三部分的每一个都可以包括同类型的多个部分规则。如下给出规则级的基本工作过程[28]。

1．它采用表驱动方法，表驱动函数为 ipt_do_table()。驱动表，又称 IP 表，用 ipt_table 结构描述。IP 表是位置无关的数据块，它由表头（ipt_table_info）和变长的规则链组成。表头包

含 IP 表的尺寸，规则数量和不同过滤编号对应规则项的起始位置。规则链是由多个变长的规则项组成，每个规则项可分为匹配链和匹配靶两部分，匹配链由 IP 匹配模板（ipt_entry）和可选的多个扩展匹配操作组成。匹配项由匹配结构（ipt_entry_match）和变长参数区组成，匹配结构包含指向匹配器 (ipt_match)模块的指针。匹配靶则由靶结构(ipt_entry_target)和变长参数区组成，靶结构包含指向靶处理(ipt_target)模块的指针。

2．Ipt_do_table()对输入的 IP 包首先沿着 IP 表规则项的链头 IP 模板进行匹配。当规则头匹配成功时，如果存在后继扩展匹配链，则进行链式扩展匹配。当所有的匹配都成功后，则运行相应的规则靶，如果规则靶函数返回 IPT_CONTINUE，则继续表中下一个规则项匹配，否则过滤过程结束并返回。如果靶函数为空，则说明它是一个标准靶(ipt_standard_target)，标准靶的参数区具有一个判别字（verdict），其值非负则表示它是一个转移靶指向另一个规则项。如果判别字为负且等于 IPT_RETURN，则说明标准靶是转移返回靶，否则将判别字取反减一作为过滤器的返回值中止过滤过程。

4.2.1.3 Netfilter 框架下数据包处理

内核模块可以注册一个新的规则表(table)，并要求数据包流经指定的规则表。这种数据包选择用于实现数据报过滤(filter_表)，网络地址转换(Nat_表)及数据包处理(mangle_表)。Linux2.4 内核提供的这三种数据包处理功能都基于 netfilter 的钩子函数和 IP 表。它们是独立的模块，相互之间是独立的。它们都完美的集成到由 Netfileter 提供的框架中。

1．包过滤

filter 表格不会对数据包进行修改，而只对数据包进行过滤。iptables 优于 ipchains 的一个方面就是它更为小巧和快速。它是通过钩子函数 NF_IP_LOCAL_IN, NF_IP_FORWARD 及 NF_IP_LOCAL_OUT 接入 netfilter 框架的。因此对于任何一个数据包只有一个地方对其进行过滤。这相对 ipchains 来说是一个巨大的改进，因为在 ipchains 中一个被转发的数据包会遍历三条链。

2．NAT

NAT 表监听三个 Netfilter 钩子函数：NF_IP_PRE_ROUTING、NF_IP_POST_ROUTING 及 NF_IP_LOCAL_OUT。 NF_IP_PRE_ROUTING 实现对需要转发的数据包的源地址进行地址转换而 NF_IP_POST_ROUTING 则对需要转发的数据报的目的地址进行地址转换。对于本地数据包的目的地址的转换则由 NF_IP_LOCAL_OUT 来实现。NAT 表格不同于 filter 表格，因为只有新连接的第一个数据包将遍历表格，而随后的数据包将根据第一个数据包的结果进行同样的转换处理。NAT 表格被用在源 NAT,目的 NAT。

3．数据包处理(Packet mangling)

mangle 表格在 NF_IP_PRE_ROUTING 和 NF_IP_LOCAL_OUT 钩子中进行注册。使用 mangle 表，可以实现对数据包的修改或给数据报附上一些外带数据。当前 mangle 表支持修改 TOS 位及设置 skb 的 nfmard 字段。

4.2.2 连接跟踪

4.2.2.1 连接跟踪的概念

连接跟踪指在内存表中记录某个连接的状态信息，如源、目的 IP 地址及其端口号，协议类型，连接状态，超时设定等。其目的是通知防火墙特定连接的状态。具有该功能的防火墙我们称之为全状态检测防火墙。连接跟踪，作为一个独立运行的模块，保证了动态包过滤及地址转换得以实现。

连接跟踪的工作原理是：检测每一个有效连接状态，并根据这些信息决定网络数据包是否能通过防火墙。该技术在协议底层截取并分析这些数据包。通过将当前数据包及其状态信息和其前一时刻的数据包及其状态检测进行比较，从中得到数据包的控制信息，达到网络保护的目的。

与连接跟踪相关的数据包具有四个不同的状态。分别是 new, established, related 和 invalid。[28][29]

- New：指在建立新连接后尚未接收到并远程主机的通信流， 或一个双向连接通信都没有数据包时的状态。
- Established：一旦发现连接的两个方向上都有通信流，则其与之相关的附加包都被看成 established。它要求一个主机发送一个包，之后获得另一台主机的应答。New 状态在收到应答包后，防火墙将状态变为 established。或当在一个包后紧接着生成 ICMP 信息，则 ICMP 出错信息、重定向等也被认为是 established。
- Related：指启动的新连接与当前现有连接相关时的状态。也就是首先必须要有一个 established 连接，其次该连接生成一个新的连接并能够被连接跟踪模块理解。它可以调整组成多重连接协议，如 FTP。FTP-data 连接可理解为 related FTP-control port。
- Invalid：该状态指不能归入以上三种类别的包。如果系统内存用完，或因没有应答已知连接而产生 ICMP 错误消息时产生的状态。这种包不会自动废弃，需要插入适当规则，设置链策略才能正确予以处理。

4.2.2.2 TCP 协议的连接跟踪

TCP 连接总是以三次握手开始。整个会话过程以 SYN 包开始，接下来是 SYN/ACK 包，最后发 ACK 包完成整个会话连接的建立，此后才能发送数据。那么连接跟踪怎样获取该点呢？

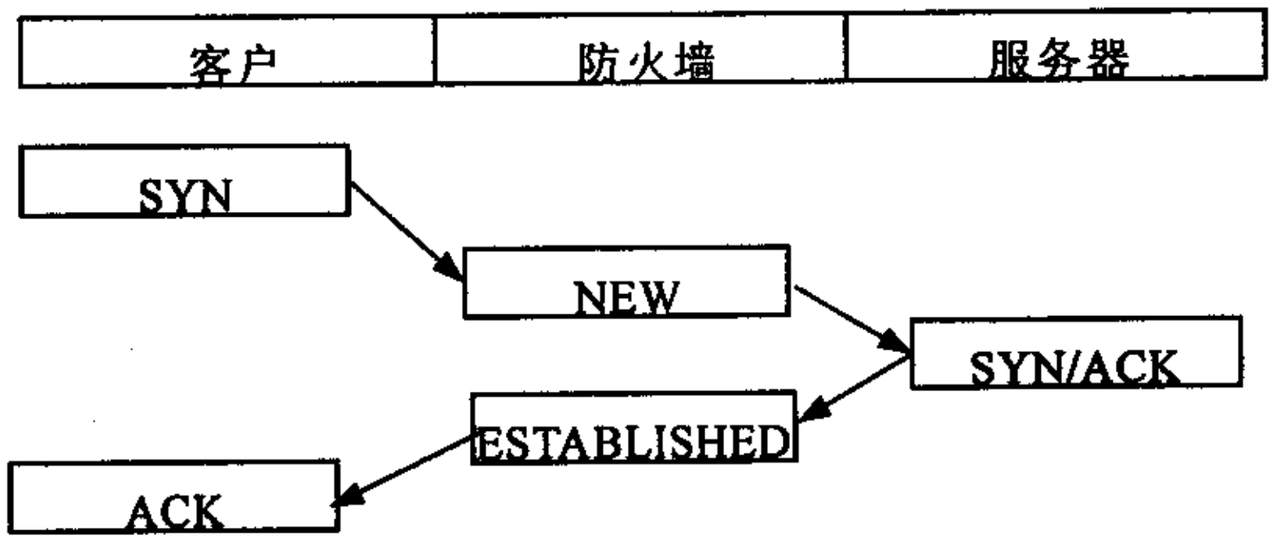


图 4-6 建立 TCP 连接

图 4-6 给出了在连接的不同状态下通信流的状态。当使用 SYN 包建立连接时，防火墙首先将该数据包和规则库比较。一旦通过这个数据连接请求，便在连接跟踪信息表中记录一个状态为 new 的连接，并设置连接的时间溢出值。然后防火墙等待一个确认连接的数据包。一旦接收到这个确认，防火墙即会修改该连接的时间溢出值。对于返回确认连接的数据包，防火墙需要对其包类型进行判断，看是否有 SYN/ACK 标志。

当关闭一个 TCP 连接，按照图 4-7（a），（b）的步骤对连接跟踪表进行操作。当通信双方关闭连接时，连接跟踪信息应该被维护一段时间。连接跟踪模块监测到一个 FIN 包时，就通知系统减少时间溢出值。如果在此周期内没有关于该连接跟踪表项的数据包交换，该连接就被删除。如果有数据包交换，恢复原周期值。如果连接双方继续通信，该连接状态（Established）会继续以该周期维持下去。

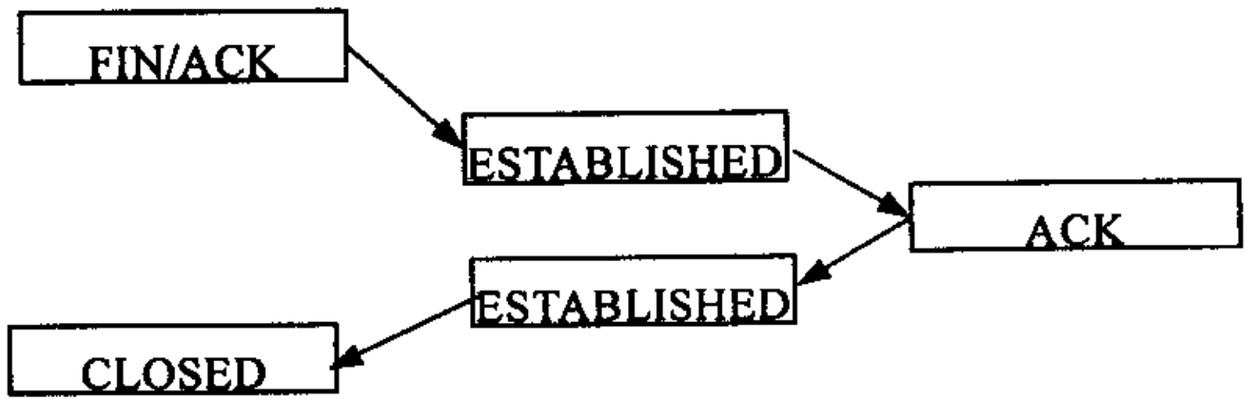


图 4-7 (a)　关闭 TCP 连接

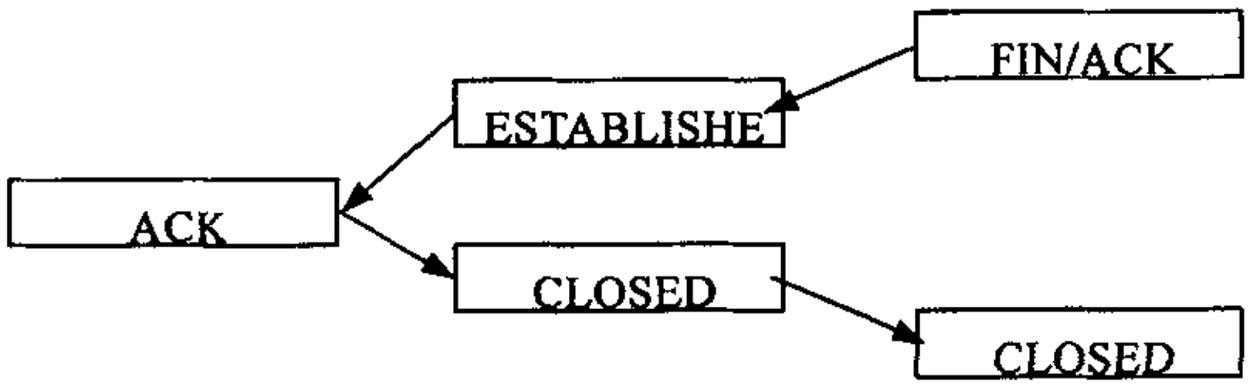


图 4-7 (b) 关闭 TCP 连接

对于 TCP 协议的连接跟踪，该模块的数据结构为

```
struct ip_conntrack_protocol ip_conntrack_protocol_tcp
= {
        { NULL, NULL },
        IPPROTO_TCP,   // TCP 协议
         "tcp",              //协议名称
         //将传来的网络协议包的 TCP 头中的源、目的端口字段取出后填入结构替
ip_conntrack_tuple 中。
        tcp_pkt_to_tuple,
        //将结构体 ip_conntrack_tuple 中源、目的端口字段互换。
        tcp_invert_tuple,
        //输出这个“元组”中的每个协议部分
        tcp_print_tuple,
        //用于保存在由结构体 ip_conntrack 组成链表总的连接跟踪状态
        tcp_print_conntrack,
        //TCP 连接跟踪最主要的处理函数。抛弃过小的分片包，然后获取包的连接跟踪信息。
如果通过连接跟踪信息表的检查，则返回 NF_ACCEPT。
        tcp_packet,
        //建立新的 TCP 连接时，调用该处理函数。目的是将通过防火墙规则检查到的数据包的
连接状态，添加到结构体 ip_conntrack 所形成的链表中，并返回该连接的时间溢出值。
        tcp_new,
         NULL
    };
```

4.2.2.3 UDP 协议的连接跟踪

尽管 UDP 是无状态协议，可以通过引入 UDP 定义逻辑的连接概念，使得 UDP 具有方向性。如图 4-8，当客户端发起一个新的 UDP 连接 并且尚未得到响应，则在连接跟踪信息表上对 new 连接上加[unreplied]标记，。通常情况下设定 UDP 时间溢出值为 30 秒。当在此时间周期内收到上一个包应答，连接状态更改为 established，去掉[unreplied]标记。之后，重置时间溢出值。如果在同一对 socket 中有多个请求和应答发生，该连接会被看作数据报流并将时间溢出值改为 180 秒。几秒后该连接被标记[assured]。这样，即便在高负载，该连接下的数据包都不会被丢弃。如果当新连接到来时连接跟踪信息表已满，则标记[unreplied]的 new 连接首先被删掉[28]。

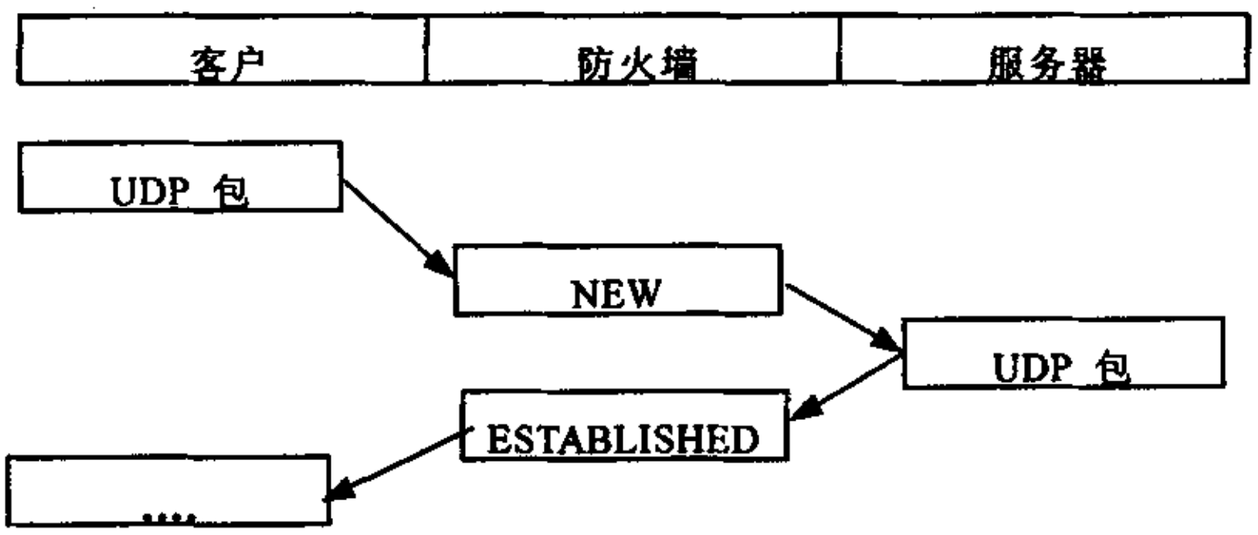


图 4-8 建立 UDP 连接

对于 UDP 协议的连接跟踪，该模块的数据结构为

```
struct ip_conntrack_protocol ip_conntrack_protocol_udp
= {
        { NULL, NULL },
        IPPROTO_UDP,        //UDP 协议
        "udp",              //协议名称
        //将 UDP 包转化为“元组”。从 UDP 头中取出源、目的端口赋到其元组中
        udp_pkt_to_tuple,
        //获取输入 UDP 数据包反向“元组”
        udp_invert_tuple,
        //按格式输出输入 UDP 数据包“元组”源、目的端口
        udp_print_tuple,
        //该函数目前为空，仅返回值 0
```

```
udp_print_conntrack,
//当一个 UDP 通信建立后，检查每个传入的 UDP 数据包状态，根据状态刷新时间溢出值
udp_packet,
//当一个新的 UDP 通信建立后，返回该通信的时间溢出值
udp_new,
 NULL
};
```

4.2.2.4 ICMP 协议的连接跟踪

在 iptables 用法中，只有四种 ICMP 状态类型（Echo 请求与应答，时间戳请求与应答，信息请求与应答，地址掩码请求与应答）能被归类为 new 和 established，其中请求定义为 new，应答定义为 established。对于非请求-应答类型的其它 ICMP 状态类型归为 related。下文将分别给出这两种情况的范例。

如图 4-9，以 Echo 请求与应答为例，发送方向接收方发送 echo 请求，防火墙将该连接跟踪信息状态设为 new。与 TCP 和 UDP 不同的是，需要在其跟踪信息中加入 ICMP 的类型、代码及标识符。接收方按照相同 ICMP 标识符发出 echo 应答后，防火墙将跟踪信息状态改为 established。所以发送方能够识别出应答并将它和 ICMP 请求连接。在 ICMP 应答之后，该连接将不再有任何合法数据流。正因如此，一旦该应答通过防火墙上的 Netfilter 结构，该连接跟踪就被删除。

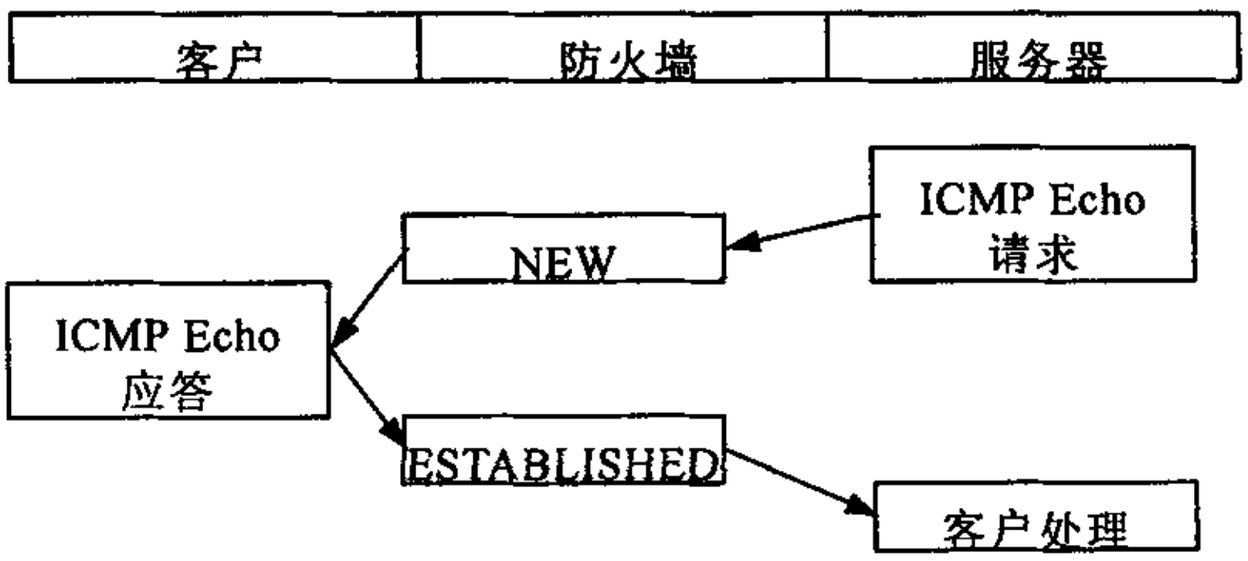


图 4-9 ICMP Echo 请求与应答

ICMP 另一个重要的作用是告诉主机特定的 UDP 和 TCP 连接发生了什么。这种情况下，ICMP 应答的状态为 related，如 ICMP 主机不可达，或 ICMP 网络不可达。图 4-10，向特定的地址发送 TCP SYN 包后，防火墙将该连接跟踪信息设为 new。由于该包不能到达目的网

络，因此路由器将网络不可达的 ICMP 差错信息返回。此时防火墙将连接跟踪信息状态改为 related。正因如此，客户才知晓中途失败。同时，防火墙确认该应答包为错误信息后将其连接跟踪删除。同样，对于 UDP 连接，防火墙将其应答 ICMP 差错信息的连接跟踪状态改为 related。图 4-11，发送一个 UDP 包时，防火墙将该 UDP 连接跟踪状态设为 new。若遇到网络被防火墙或路由器禁用，防火墙会收到 ICMP 网络禁用的消息。由于该应答与 UDP 连接相关，所以将它作为 related 状态包发给客户。客户收到该 ICMP 差错信息后中断该连接，同时防火墙也删除该 ICMP 连接跟踪。

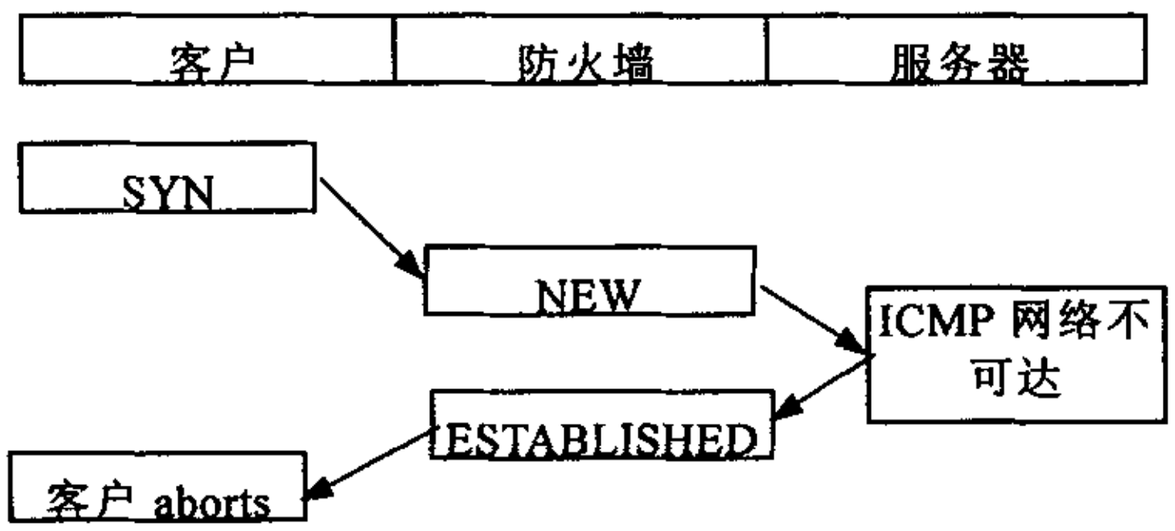


图 4-10 与 TCP 相关的 ICMP

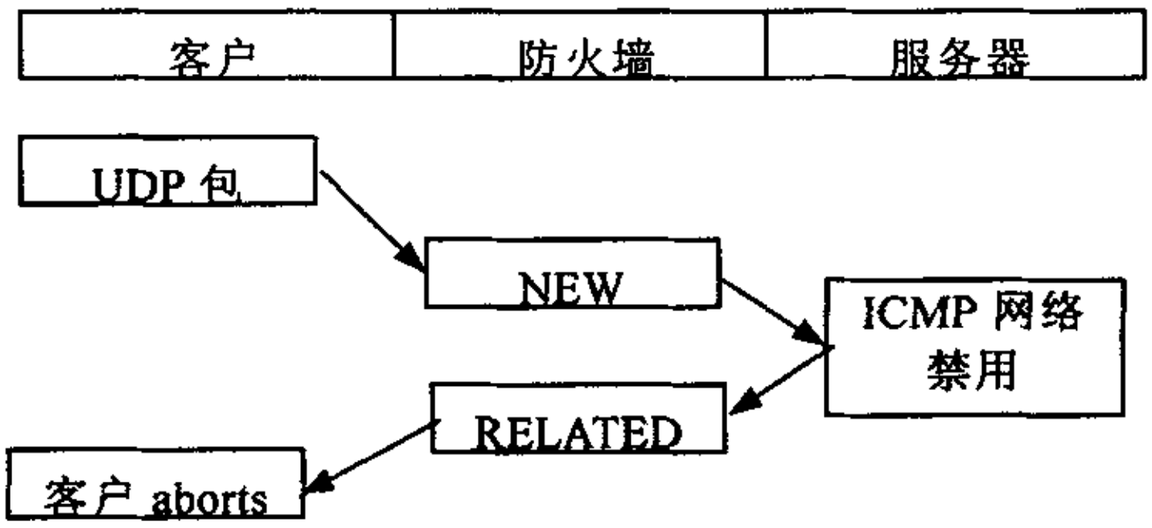


图 4-11 与 UDP 相关的 ICMP

对于 ICMP 协议的连接跟踪，该模块的数据结构为

```
struct ip_conntrack_protocol ip_conntrack_protocol_icmp
= {
    { NULL, NULL },
    IPPROTO_ICMP,       //ICMP 协议
    "icmp",             //协议名称
     //将 ICMP 包转化为“元组”。从 ICMP 包头中取出其类型、代码及标识符赋到其元组中
    icmp_pkt_to_tuple,
    //获取输入 ICMP 包的反向“元组”。先要检查输入 ICMP 包的类型是否属于安全类型。若
```

不是，返回 0 不做任何处理。若是，就转换为反向“元组”

icmp_invert_tuple,

//按格式输出输入 ICMP 数据包“元组”。ICMP 包类型、代码和标识符

icmp_print_tuple,

//该函数目前为空，仅返回值 0

icmp_print_conntrack,

//完成对 ICMP 包处理。先检查是否为回应 ICMP 包。若是则在连接跟踪表中去掉该连接信息。反之回应 ICMP 包，更新溢出时间值

icmp_packet,

//当一个新的 ICMP 通信建立时调用该函数。首先验证输入的 ICMP 包的类型是否为安全类型，如果是，返回一个时间溢出值

icmp_new,

NULL

};

4.3 服务负载均衡模块设计

服务负载均衡模块对大数据流，高业务量的网络处理中，提供了一个比较好的解决方案。这里，从 Linux 系统中的技术基础出发，并采用基于 NAT 的服务器集群方式的设计思路，给出模块的结构以及主要算法的实现。

4.3.1 功能需求

基于 LVS 系统，在防火墙上加入服务负载均衡模块，保证防火墙提供服务负载均衡的功能。具体地说，它必须具有以下功能：

- 服务转发。能接收来自外部网中基于 TCP/IP 的服务请求，并将它们转发到当前负载最轻的机器上执行。
- 动态负载均衡。根据连接到内部网中的实际服务器的连接数目，决定负载最轻的机器。
- 连接持续性。来自外部网的同一客户的所有请求必须转发到内部网中的同一台实际服务器上进行处理。

4.3.2 环境设置

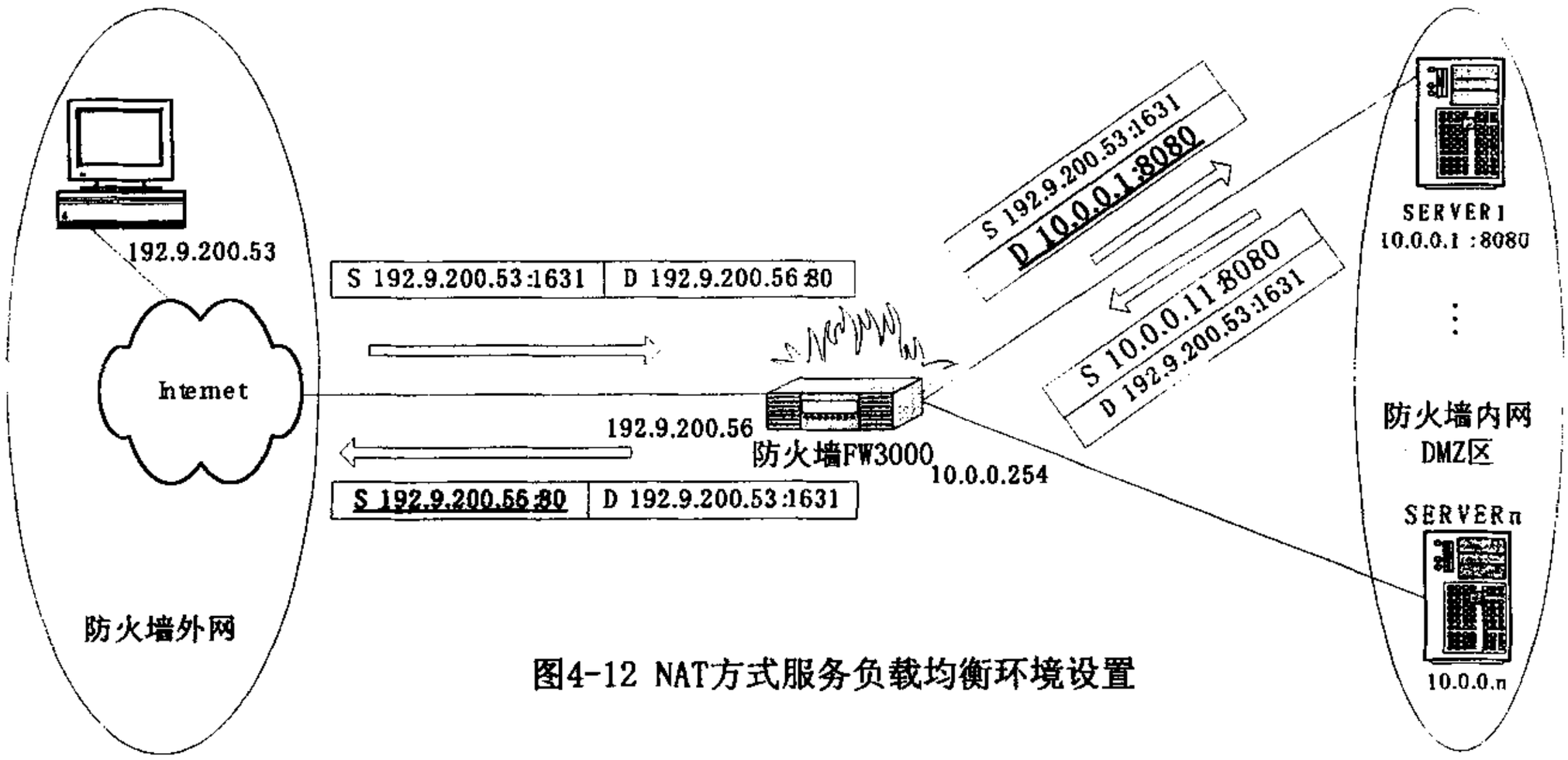


图4-12 NAT方式服务负载均衡环境设置

如图 4-12 所示，该集群系统由防火墙 BOCO．SFW-3000、server1、server2 以及 server n 等机器组成。其中防火墙 BOCO．SFW-3000 作为内部网和外部网的接口，能够接收外部网的用户请求，并将此用户请求发送到防火墙 DMZ 区 server1 到 server n 中的某台机器上（如 server1 ）。当 server1 处理完用户的请求以后，就将处理完的结果发送给防火墙 BOCO．SFW-3000，然后再由防火墙 BOCO．SFW-3000 将请求应答返回给外部网的用户。本文将防火墙上的 DNAT 模块称为服务负载平衡器，因为它承担了均衡负载的作用；将实际响应用户请求的 server1 等机器称为实际服务器。试验环境中外部网中的客户机器的 IP 地址为 192.9.200.53，防火墙有两个 IP 地址，一个是内部全局地址[30]（代表一个或多个内部 IP 到外部世界的合法 IP）192.9.200.56，一个是内部局部地址（指定内部网络的主机地址，全局唯一，但为私有地址）10.0.0.254，服务负载均衡器在防火墙 BOCO．SFW-3000 上运行。内部网中有 *n* 台实际服务器，它们的 IP 地址分别为 10.0.0.1、10.0.0.2……..10.0.0.n。将这些服务器的网关都设为 10.0.0.254，并且都增加了通往 192.9.200.0 网络的路由。服务负载均衡的目的就是将客户发向平衡器的请求如：telnet、 ftp 、www 等按照内部网机器当前负载的情况分发到各个实际服务器上。

4.3.3 设计原则

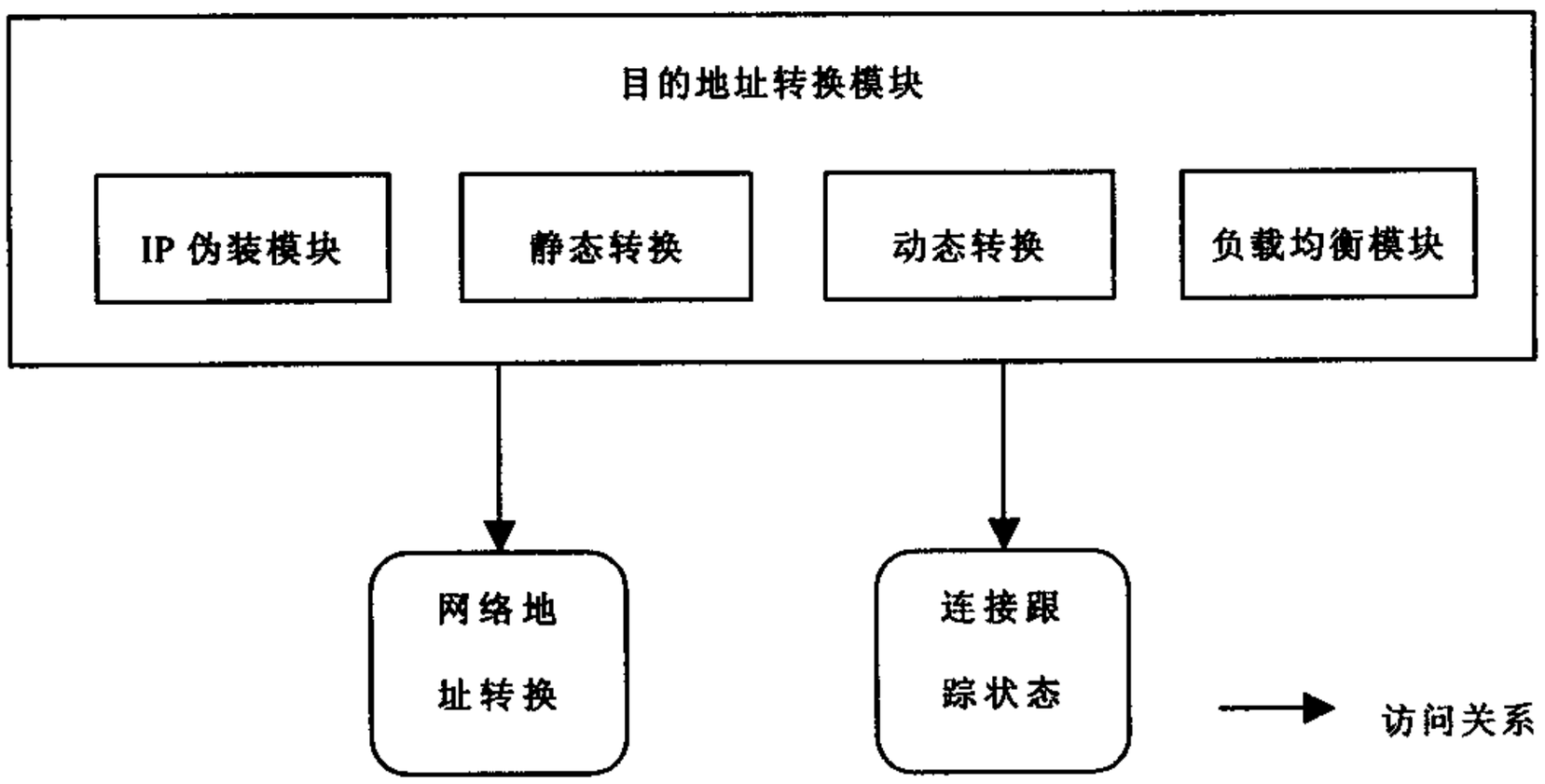
原则一：对能进行负载均衡的基本网络结构，设计要求其为对称结构。在对称结构中每

台服务器都处于等价地位，即都可以单独对外提供服务而无须其他服务器的辅助。

原则二：利用 NAT 实现 IP 级的服务负载均衡。以图 4-12 为例，当外部用户（IP 地址：192.9.200.53:80）向服务负载均衡器发送一个请求报文时，在均衡器的 IP 层利用 DNAT 对此请求报文的目的地址（IP 地址：192.9.200.56:1631）进行替换，将目的地址替换为内部网络中某个实际服务器（如 server1）的 IP 地址（10.0.0.1:8080），然后将此报文转发出去。当内部网络中的实际服务器（server1）将请求处理完以后，它将应答发回到服务负载均衡器，该平衡器再次利用 DNAT 将目地地址（192.9.200.56:1631）转换为发出请求的外部网用户的 IP 地址（192.9.200.53:80），然后将此报文转发到用户。并由连接跟踪技术提供的连接状态表和地址转换表决定是否进行服务负载均衡。

原则三：选取负载最轻的服务器的 IP 地址在 DNAT 模块中实现。

4.3.4 程序模块



服务负载均衡模块是基于 DNAT 的路由选择方式。如图 4-13 所示，它作为防火墙系统中 NAT 模块的组成部分，通过目的地址转换实现。在该过程中，需要访问连接跟踪状态表（由连接技术提供的）和网络地址转换状态表，以便找到匹配的表项。

4.3.5 DNAT 及其数据结构

4.3.5.1 DNAT 表

规则元素 / 对象 NA 类型	转换前			转换后		
	源地址名	目的地址名	服务名	源地址名	目的地址名	服务名
DNAT	任意	任意	任意	Orig	任意	Orig
服务 DNAT	任意	任意	TCP 对象 TCP 对象组 UDP 对象 UDP 对象组	Orig	任意	TCP 对象 UDP 对象

表 4-1　DNAT 表

1．转换前：只是一种 IP 包匹配条件。

源地址名：　用于匹配原 IP 包中的源地址。

目的地址名：用于匹配原 IP 包中的目的地址。

服务名：　用于匹配原 IP 包中的服务，类似于包过滤规则中的服务名，这里因用于服务负载均衡，所以将带有服务转换的 DNAT 中限定为 TCP 或 UDP 对象(或是只包括 TCP 或 UDP 对象的对象组)。

2．转换后：转换后三元素决定了地址转换的行为动作。其中 orig 表示保持不变，与转换前的相同。根据不同的动作要求，来配置相应的三元素内容

源地址名：　在 DNAT 中无效，保持原有源地址不变。

目的地址名：在所有的 DNAT 中有效，可以是任意地址对象。主机地址对象提供了静态目的地址转换功能；网络地址对象提供管理员指定的网络地址段内所有地址组成的地址池的 DNAT 转换(根据连接数目决定转换后地址池中的一个 IP 地址，提供负载均衡功能)。

服务名：　只在服务 DNAT 中有效，只能为 TCP 或 UDP 对象，且与转换前的服务名具有相同的协议类型

4.3.5.2 数据结构

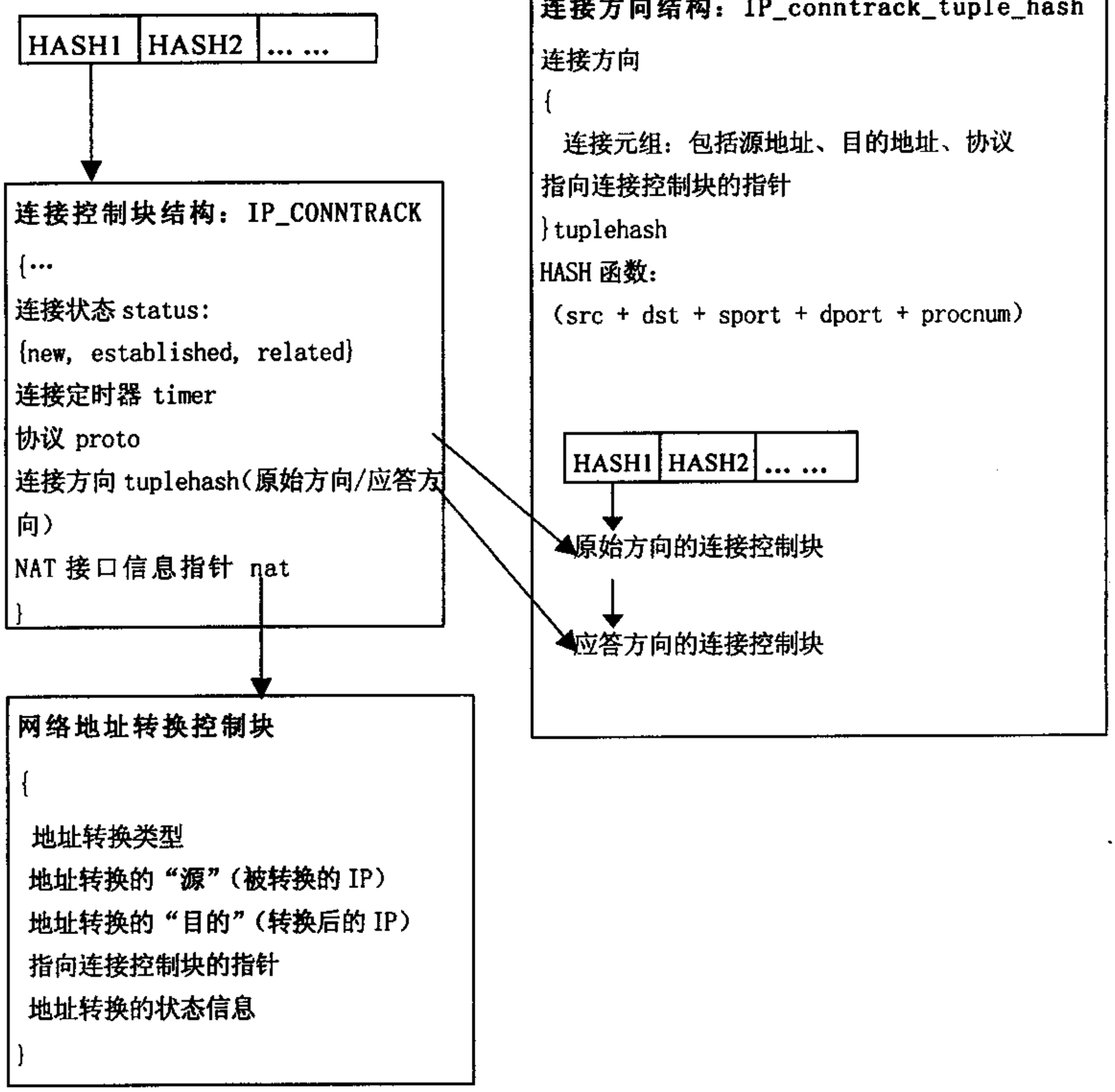


图 4-14 NAT 的数据结构图

如图 4-14 所示，网络地址转换（NAT）是通过 HASH 表组织。在此 HASH 表的每一个冲突链是由连接控制块构成。特别重要的是，在每一个连接控制块中包含 NAT 接口信息 *nat*，其中含有网络地址转换控制块。[31]

4.3.6 模块流程

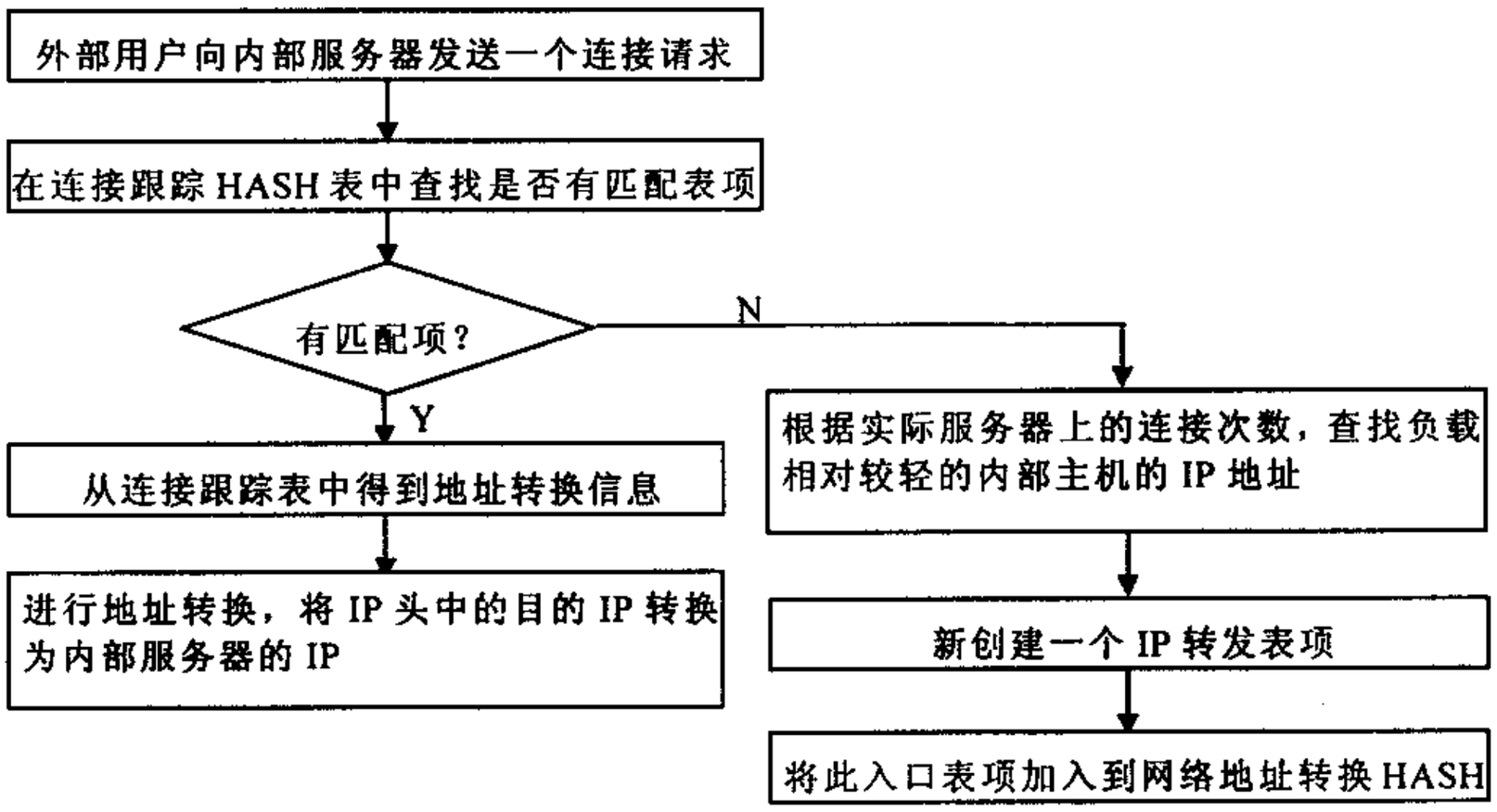


图 4-15 服务负载平衡流程图

4.3.7 实现负载均衡的算法

第三部分给出了八种负载均衡算法，其中轮循算法、加权轮循算法、最少连接次数算法和加权最少连接次数算法是目前比较常用的算法。由于后两种算法适用于动态网络地址转换，并依照设计原则一内部网中的服务器性能相同，因而在服务负载均衡模块中选用 LC 算法。如下，给出其功能函数[32]。

```
if 新连接包
  中心策略模块处理
  if 需 NAT 处理
    if 服务器的连接数目范围
      lc （最少连接次数）：=65535（不可达到的值）
      for 规则中的每个“目的”地址
cnt （计数器）= 0
for 在地址转换状态表中 每一项
  if（目的地址（目的转换地址）/源地址（源地址转换））=“目的”地址
      cnt ：= cnt + 1
  fi
endif
if cnt < lc
```

```
            okADDR（选中的地址）：＝ “目的”地址
        fi
    fi
fi
做 IP 包的地址转换
把地址转换信息写入连接跟踪表
    对连接跟踪表进行修改
    加入 IP 包被修改后可能的连接应答
fi
if  not 新连接包
    从连接跟踪表中得到地址转换信息
    进行地址转换
fi
```

4.4 双机热备份模块设计

4.4.1 功能需求

针对防火墙高可用性的要求，提出双机热备份(只支持两台)。当主机失效(Fail over)后，从机接管(take over)主机的工作。其中，防火墙主机失效定义：1）防火墙主机本身的故障；2）防火墙网口失效。如下给出各功能模块的需求。

1. 主机功能需求

- 主机配置双机热备份的基本参数。
- 主机定时向从机发送心跳。
- 主机可根据从机请求向从机发送同步资料。
- 主机在从机接管工作前根据需要复制从机配置（相同模式时）。
- 主机能够判断从机软件版本，版本不同不做双机热备份。
- 主机能够监测各网口状态，当某个网口故障时自动退出工作。
- 发送运行日志。
- 主机上的配置文件更改后，可以通知从机更改了哪些配置。

2. 从机功能需求

- 从机在指定基本的同步数据网口和 IP 配置后可以自动完成配置。
- 从机定时向主机发送心跳。

- 从机根据需要复制主机配置。
- 验证主机模式、从机处在不同模式时自动重启、同步模式。
- 从机能够判断主机软件版本，不同版本不做双机热备份。
- 在处于工作状态时，发送状态日志、告警日志。
- 监测到主机后能主动移交资源控制权。

3. 主从机状态接口

主从机通过以太网口互相传递 heartbeat 信息，也可以同时使用串口 ttyS1。

4.4.2 设计原则

双机热备份系统是目前最为常用的高可靠性设计方案，用于防止单点故障。

原则一：基于网络运行是否正常来判定防火墙是否正常。

一个节点的健康状况包括：

- 节点机的硬件设备是否运行正常
- 节点机的网络工作是否正常
- 节点机各项服务是否运行正常

因此，判定防火墙主机是否运行正常也同样从三方面进行，即：

- Network adapter events
- Network events
- Node events

对应防火墙 BOCO．SFW-3000，需要监测的 Node events 主要有：1）透明代理程序；2）日志/报警服务程序；3）管理中心配置程序。而 Network events 和 Network adapter events 对应于防火墙 BOCO．SFW-3000，都可以通过监测其“网络正常与否”来确定。

因此，防火墙 BOCO．SFW-3000 的双机热备份系统可以立足于“网络正常与否”、“服务正常与否”来判定。考虑到“服务正常与否”实施比较复杂且软件可通过反复测试排除故障，故防火墙 BOCO．SFW-3000 双机热备份现阶段仅基于“网络正常与否”来判定防火墙是否运行正常。

原则二：基于主/从模式，实现双机热备份。

主机工作时，从机处于实时等待状态，而不转发数据帧，而一旦探测主机“坏了”，从机则及时接替主机工作。主机修好后，重新启动后仍作为主机，从机仍作为从机。在任一时刻，主/从机只会有一台处于工作状态，不具有负载均衡功能。

从机在不工作时，将关闭其数据转发机制，并关闭其 STP 协议[①]。

原则三：双机热备份要同时适应路由模式和透明模式。

4.4.3 模块流程

从图 4-16 中可以看出，防火墙双机热备份从实现角度上分为三个模块：

- 防火墙双机热备份配置
- 防火墙双机热备份同步数据
- 防火墙双机热备份状态检测模块

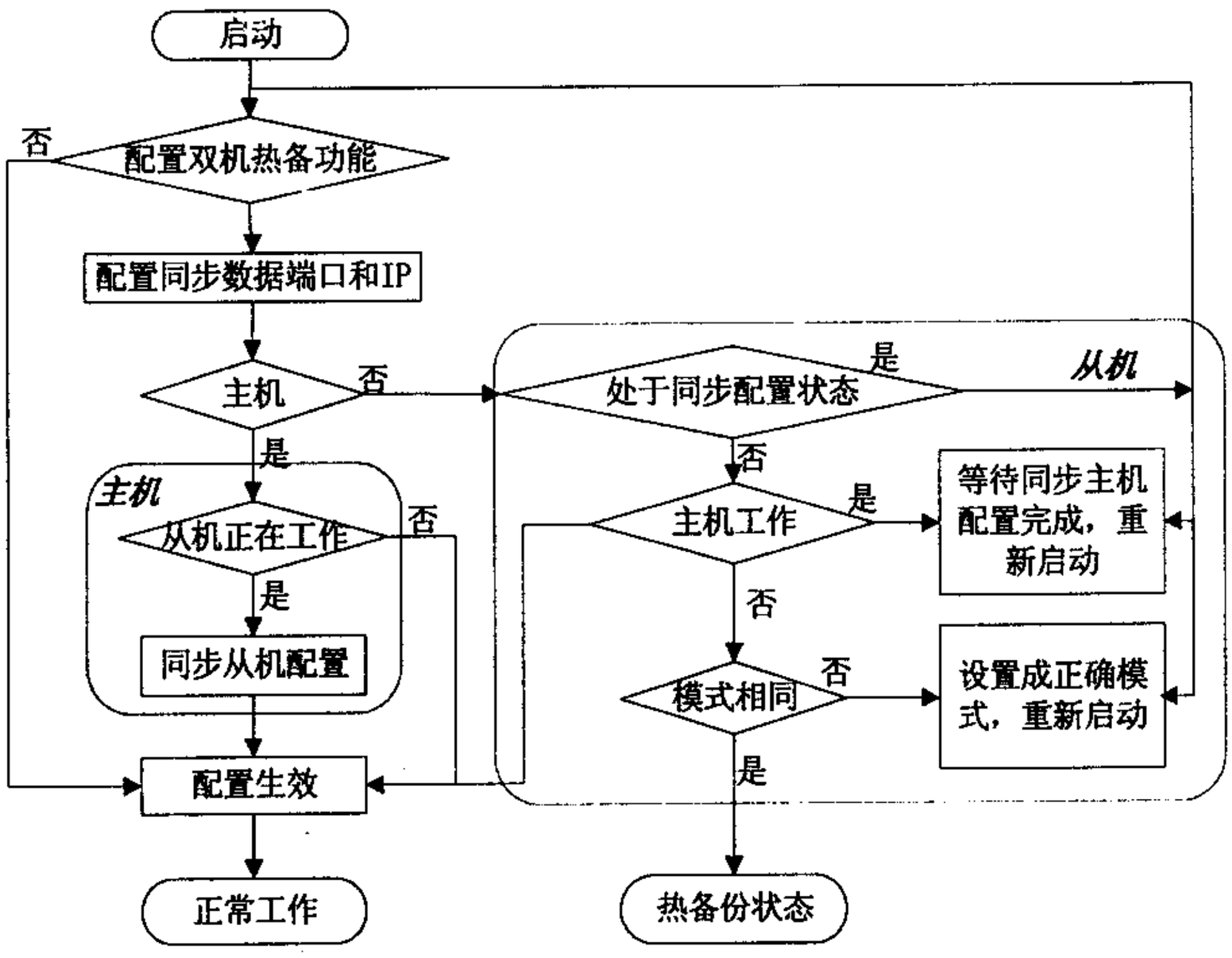


图4-16 双机热备份的工作流程图

4.4.4 防火墙主/从机状态探测模块

在冗余备份系统中，心跳检测技术是最常用的系统内节点 Health 指标检测方法。该技术指在一个节点上，通过一定方法，由网络或串行口等对临近主机节点的某些状态进行检测，以探知该节点的健康状况。

防火墙 BOCO．SFW-3000 的双机热备份基本沿用 OpenSource Heartbeat 原型代码，以下做简要说明。如图 4-17。

- 写进程：从管道（与控制进程相关联）中读取心跳信息，通过 udp 和 ttyS1（可选），

发送心跳。

- 读进程：分别从 udp 和 ttyS1（可选），接收心跳，将收到心跳消息写入与处理进程相关联的 pipe 中。
- 处理进程：处理心跳信息，检测、设置 heartbeat 状态，调用脚本程序进行设置，通过/var/lib/heartbeat/fifo 传递心跳信息到控制进程。
- 控制进程：从/var/lib/heartbeat/fifo 中读取并处理心跳信息，然后通过管道传递改写进程。

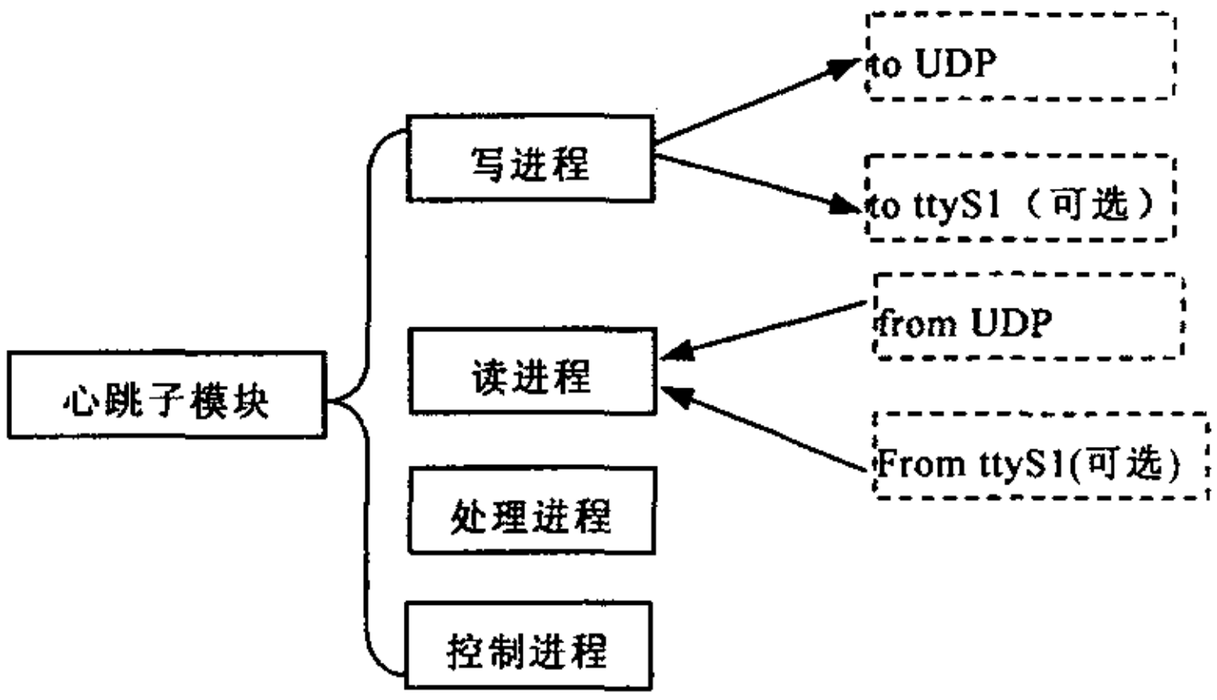


图 4-17 心跳模块结构

4.4.5 防火墙双机热备份数据同步模块

在防火墙双机热备份系统中，需要解决的一个重要问题是主从机之间的数据同步问题。因为，主机上的数据可能改变，有些信息为动态数据。为了保证备份机在接替主机后能按原先方式运行，就必须同步主机上的关键数据。

4.4.5.1 主要数据结构

1．Dirtyflag 变量

dirtyflag 是为主从机之间同步配置设置的变量，用于标识配置文件的修改状况。每一位对应一个配置文件。当配置文件更改时，对应的 dirtyflag 相应位也要被置位。

(1) Dirtyflag 的数据结构

```
union dirtyflag {
    long long llflag
    long      lflag[2];
```

}

(2) Dirtyflag 的操作过程

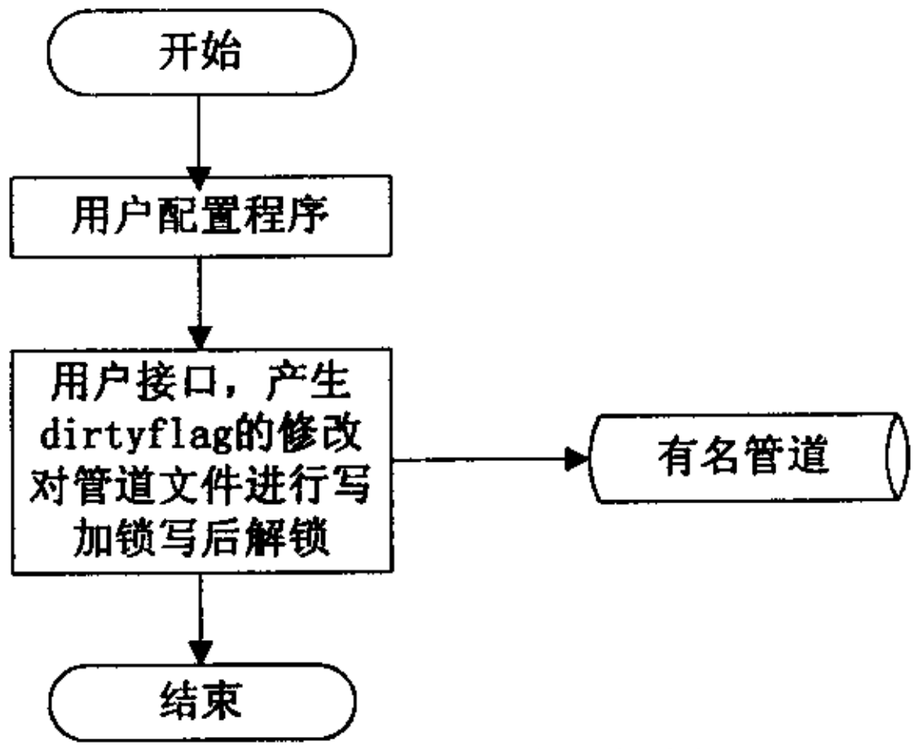


图4-18 dirtflag的操作过程

2. 同步配置结构

```
Struct conf_msg {
    int    size;                    // 内存块大小
    int    cmd;                     // 请求指令
    Long long   dirtyflag;                  // dirtyflag
    char       date[1];             // 可能是 hash 值，也可能是传递的文件，要根据
};// cmd 来决定
```

3. 数据同步消息

(1) 数据同步消息格式

交换数据类型	数据序列标识	版本	数据长度	校验	数据区

图 4-19 数据同步消息格式

对数据同步消息格式的各项作如下解释：

- 交换数据类型：包括数据信息，会话信息，错误信息，使协议结构化更好，并增加错误处理。
- 数据序列标识：表示了在系统注册了的同步数据的序列号，通过辨析这一数据段，可以告诉处理程序采用哪个注册处理函数来处理同步数据。
- 数据长度：表示“数据区”数据的长度，该字段使得数据、字段的定位更加容易。
- 版本：以后协议修改了，用户可以通过该字段，识别调用相应的处理程序。

(2) 数据同步消息数据结构

```
stuct syn_msg{
        int    msg_type;
        int    version;
        int    msg_flag;
        int    msg_len;
        char check[16];
        char data[0];
    }
```

(3) 数据同步消息注册结构

```
struct syn_register_info{
        struct node_info    *node;
        struct auth_info     *msg_auth;
        struct syn_msg *msg_buf;
        int    msg_tag;
        char msg_name[64];
        int    (*do_msg)(void*);
        int    (*get_msg)(void*,...);
        void (*show_msg)(void*);
        int    (*modify)(int type, int flag, char *buf, int num);
    } *msg_reg[MAX_MSG_NUM];
```

(4) 数据同步消息摘要算法注册结构体

```
struct auth_info {
        int auth_method;
        int (*do_auth)(char *string, char *digest, int strlen);
        void(*show_auth)(unsigned char *digest);
    }
```

4.4.5.2 配置同步

每次同步配置文件，先同步配置文件的 MD5 摘要，然后与本机保存的相应摘要对比，有变化再同步资料。

1. 第一次同步

主机工作，从机配置了第一次同步

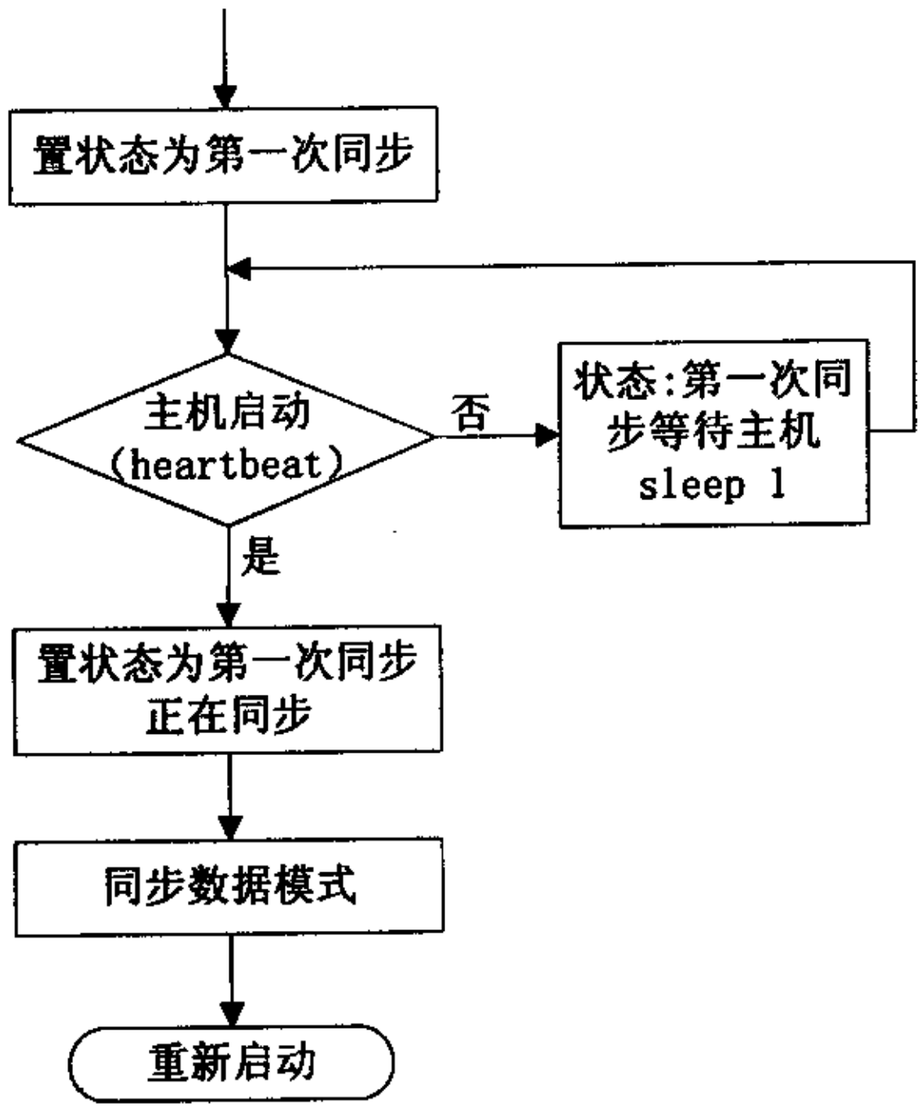


图4-20 从机配置第一次同步

2. 启动时同步

配置同步只在第一次同步且 firewall 重启时，才主动发送命令以获取工作在主机模式的防火墙配置文件的 hash，此时传送命令中 dirtyflag 各位全部为 1。其他情况下配置同步都是主机发起的。

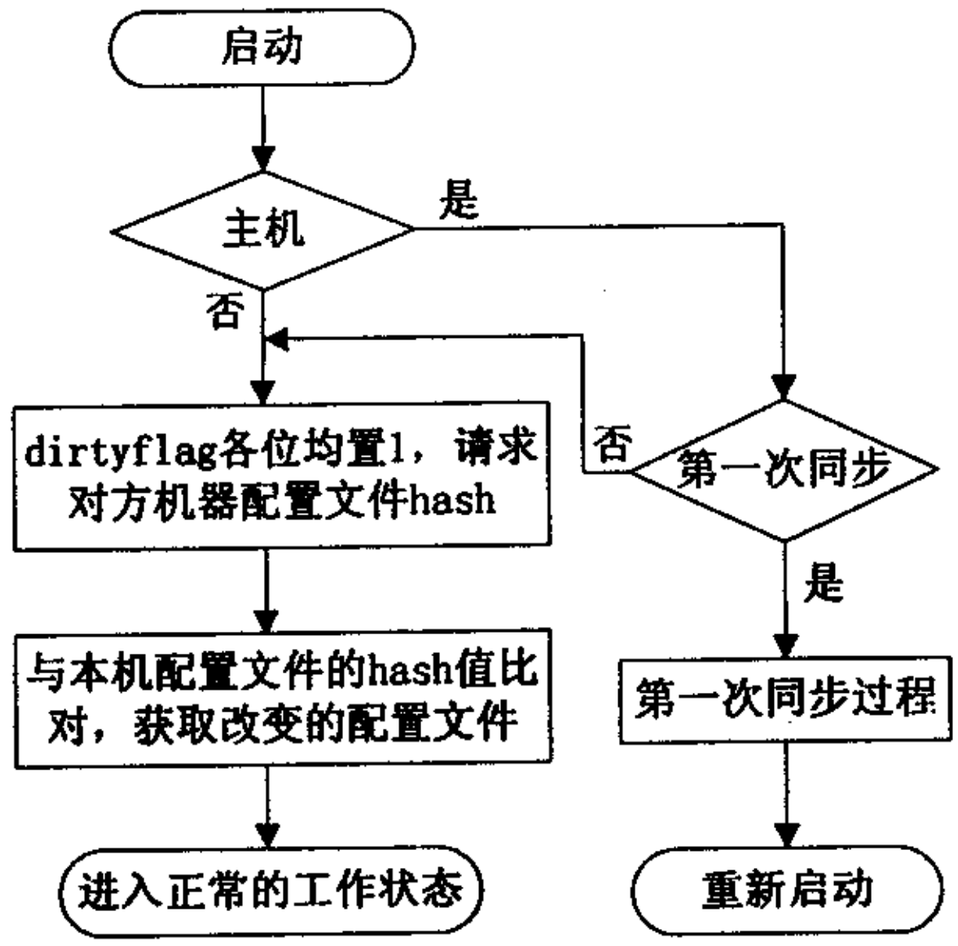
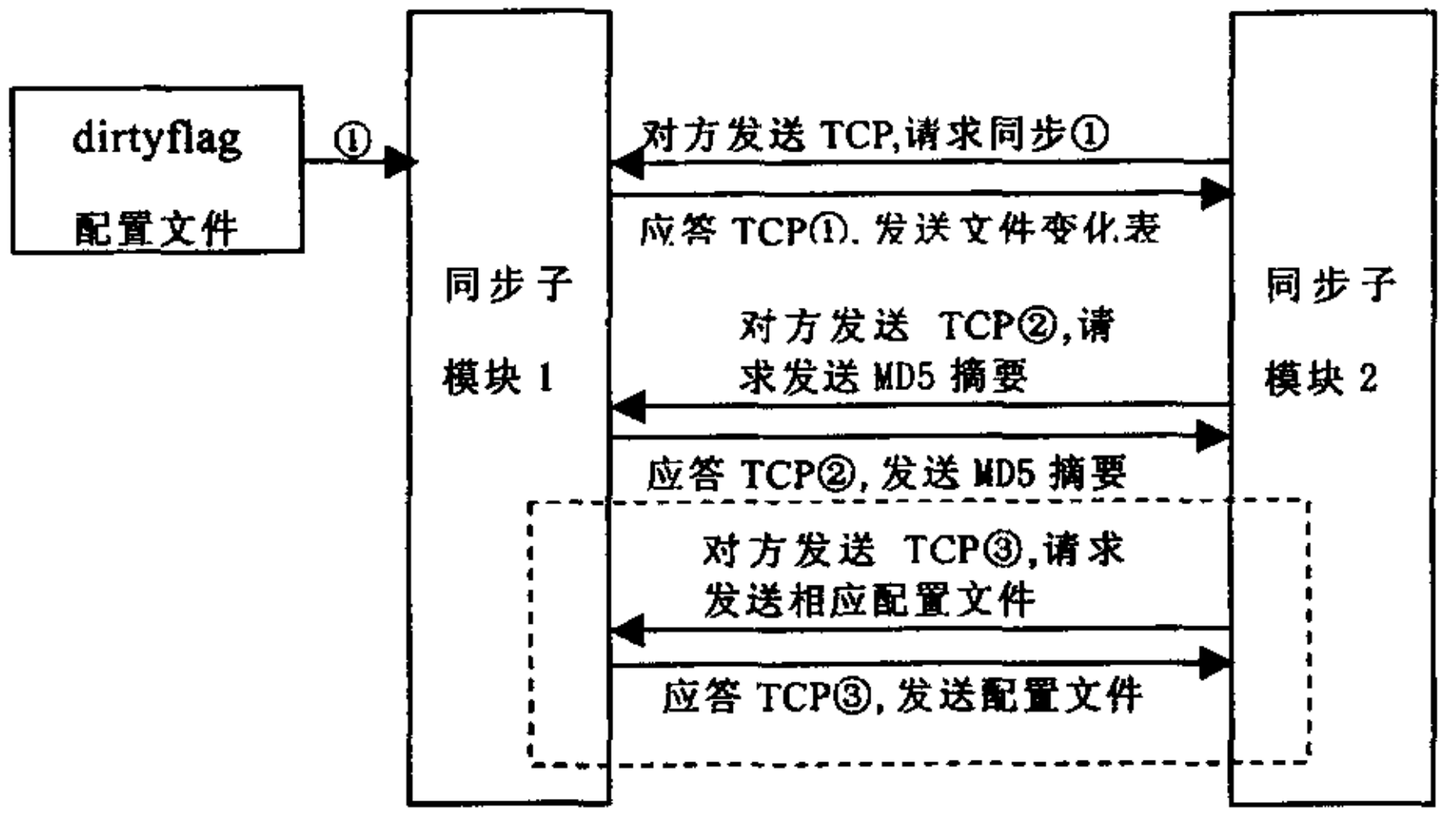


图4-21 配置同步

4.4.5.3 同步子模块的调用



表示将同步子模块1发送的配置文件的MD5摘要与对方防火墙的相应配置文件的 MD5 摘要进行比较。如果存在不同，则进行该框中的动作。同步子模块 2 收到文件后保存文件到本地磁盘，重新启动相应模块。

图 4-22　同步子模块的调用

1. 从机过程

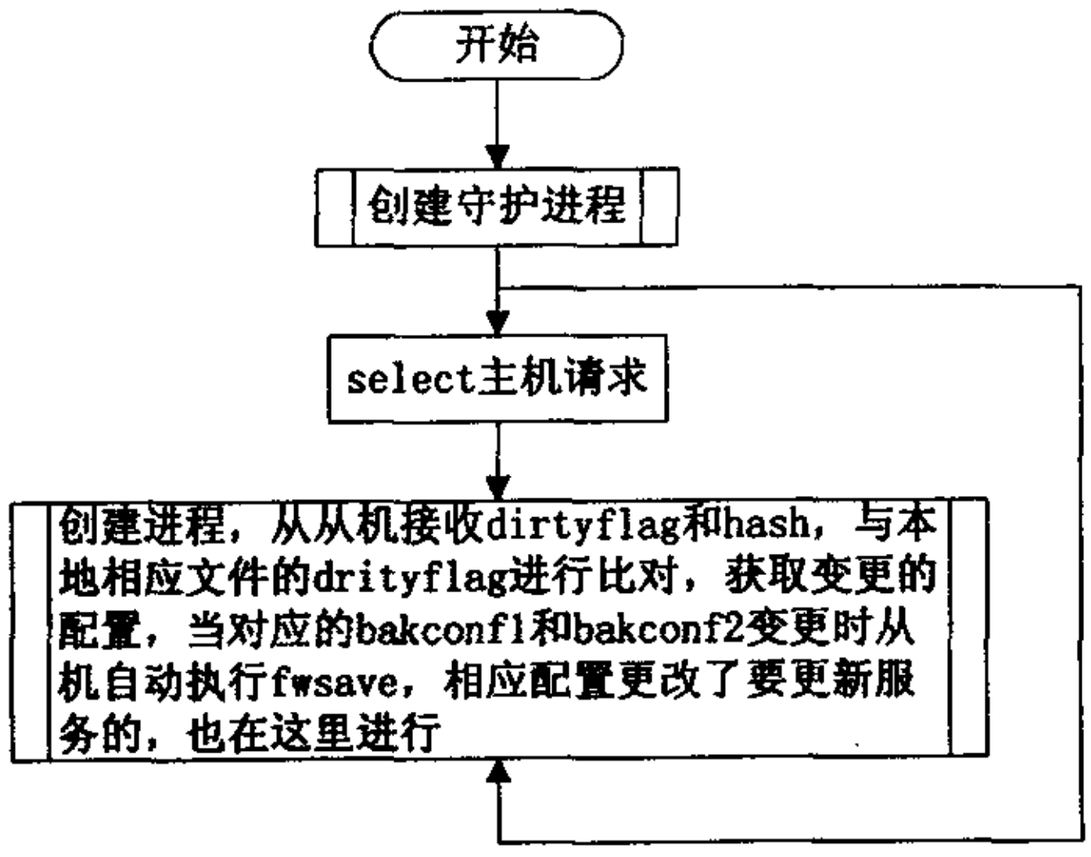


图4-23 同步时从机过程

2. 主机过程

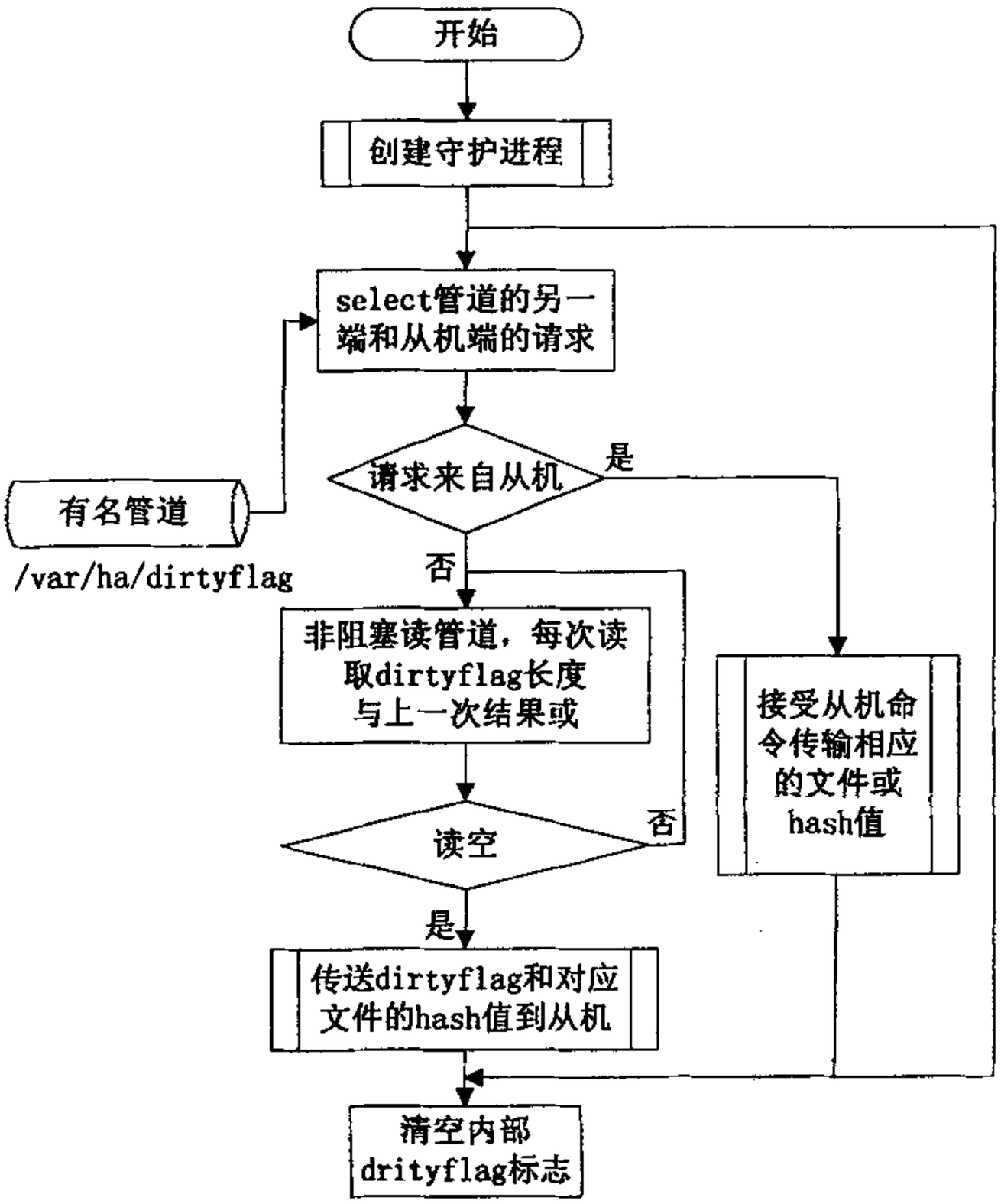


图4-24 同步时主机过程

4.4.6 防火墙双机热备份命令行配置模块

该模块是双机热备份的管理员的用户接口，只有使用双机热备份时才需要使用程序设置系统。需要说明的是该项设置一定要在防火墙初始化后执行，该模块包括以下功能：

- 设置双机热备份系统中本机节点名，IP 地址，在充当工作机时，该网卡需要绑定漂移地址。其中，节点名应与本机的 hostname 一致。当该防火墙为路由模式时，IP 地址是外部网口（eth1）的地址；透明模式时，IP 地址是本机桥（br0）的地址。路由模式下，还需要输入各网卡应绑定的漂移地址；透明模式时，则仅指定需要漂移的网卡即可，响应需要绑定的漂移地址在 Set HA Primary Node 确定。
- 指明系统的主节点名，设置当处于工作时需要漂移的 IP 地址。透明模式下，该漂移

地址主要供管理中心用；路由模式下，该 IP 地址供管理中心、路由寻径用。

- 指明系统心跳检测方法（缺省：eth2 UDP），并设置 Heartbeat 心跳检测使用的端口号。
- 指明 Heartbeat 系统心跳间隔
- 设置同步端口的 IP。用户不可更改同步端口号。
- 删除冗余系统中的节点。当系统中移去一个节点后，通过该项删除此选项。
- 显示双机热备份系统信息。
- 启动双机热备份模式，在该选项中选择“y”，则打开双机热备份模式。
- 设置主机死亡时间的判定门限。

4.5 小结

本节给出了目前比较流行的一种防火墙负载均衡的实现方法和过程。这里仅给出了服务负载模块和双机热备份模块的模块设计。由于负载均衡的效率同网络环境和集群服务器个数有关，而且目前尚未完成防火墙整体最后代码的编制，所以此处不给出相应的性能评估。但是从宏观上，该方法解决了因服务器负载过大或防火墙单点失效造成整个网络崩溃的问题。然而在实现过程中，要求热备中的两台防火墙以主从机方式工作，所以在任何情况下，仅有一台防火墙处于工作状态，因而没有从根本上解决防火墙的负载问题。进一步研究和设想在第 5 章讨论。

5 进一步研究和设想

高可用性的目标是为了消除网络中的单点失效，为此提出了多台防火墙共同工作，实现负载均衡的机制。为此，给出了国外目前流行的两种方法。并根据实际情况，特别强调了防火墙集群。此时，集群中的各台防火墙处在对等式工作状态，能够根据当前每台防火墙流量情况自动均衡负载。

5.1 分布式防火墙

5.1.1 分布式防火墙的定义

分布式防火墙技术是由 Steven M. Bellovin 于 1999 年提出的。其基本思想是由多个防火墙构成，安全策略被某种策略语言集中定义，系统管理工具将安全策略分布到每个防火墙上。通常，分布式防火墙包括三部分：（a）安全策略语言。它描述何种连接被允许或被禁止；(b)策略分布机制。它用来保证在传输过程中，安全策略的完整性；(c)认证和加密机制，如 IPSec。根据网络拓扑结构，传统防火墙通过 IP 地址和它上面的网络接口，如“内部网”、“外部网”和“停火区”，可识别主机。入侵者很容易进行地址欺骗。在分布式防火墙中，作为一种安全机制，IPSec 提供了加密证书。依据证书授权机器，无须考虑机器的物理位置。如图 5-1 所示，根据策略和 IPSec 中发送方的加密校验认证，由受保护的主机来决定接收或拒绝到来的包[33,34,35,36]。

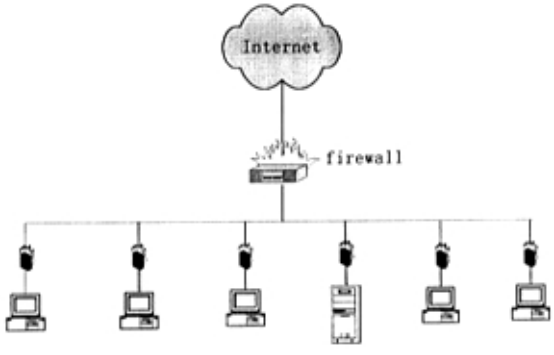


图 5-1 分布式防火墙的拓扑结构

5.1.2 分布式防火墙的优点

分布式防火墙具有如下优点：1）拓扑结构独立。分布式防火墙最重要的优点是它所保护的主机不受网络拓扑的限制。2）防止内部攻击。由于分布式防火墙消除了拓扑限制，那

么对于任意主机而言，内、外网之间就不存在差异了。只要启动防火墙，安全策略就会施加到输入、输出流上。同时，主机是由加密证书识别的，因而消除了欺骗。3）消除了单点失效。作为内外网的唯一连接点，传统防火墙不但造成单点失效，而且因为防火墙的处理速度限制了整个网络的性能。分布式防火墙将防火墙的审计、认证、访问控制、完整性检查等功能由多台计算机承担，实现负载均衡。在这种情况下，网络中有很多入口点，吞吐量不再受防火墙速度的限制。而且安全策略是由系统统一发送到各主机端，避免了在多台防火墙接入时，由于策略不一致造成的冲突。因此，网络性能得到了很大提高。

5.1.3 分布式防火墙的实现

分布式防火墙的实现方式有多种，一种可行的方案是基于 OpenBSD 系统来实现，因为其提供了 IPSec stack，SSL，KeyNote 等特性和库函数，具有良好的完整性。系统由三部分组成：内核扩展、策略设备和策略处理。如图 5-2 所示。

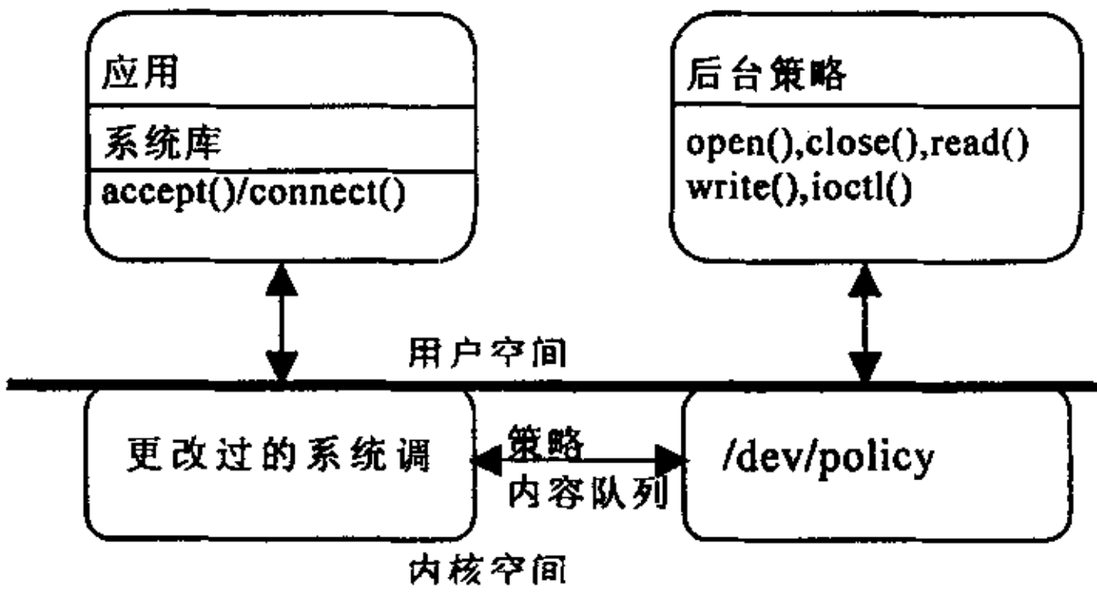


图 5-2 系统功能调用

1．内核扩展

以TCP连接控制为例，在UNIX操作系统中，TCP连接用两个系统调用connect()和accept()来完成，这两个系统调用可被任何用户访问，因此需要基于安全策略的过滤。过滤既可在用户空间也可在内核空间实现。用户级方式指将应用直接联系到提供所需安全机制的库中。它的好处是独立于操作系统，无须改变内核代码。但是，如果应用程序使用了已修改的库，这种方式就不能保证安全。内核方式要求修改操作系统。虽然会限制使用像 BSD 这样开放的操作系统，但是为应用层提供了透明的安全机制。策略内容是包含所有连接信息的一个数据结构。它包括用户 ID，证书，目的地址及端口等内容。策略内容按序号排列，等待后台策略程序决定是否接受它。

2．策略设备

为了使整个系统更加灵活，可以将后台策略处理放到应用层。采用伪设备驱动器（/dev/policy）作为策略设备可达到该目的，策略设备作为用户空间的策略处理和内核中以

修改的系统调用之间的通信路径。支持一般的操作，如 open()，close()，read()，write()和 ioctl()。当从设备读时，策略处理被阻断，直到提出了请求。Write()负责将策略处理决定返回到被阻断的调用并唤醒它。

3．策略处理

策略处理是用户级处理。它基于策略来决定是否接收或否认连接，由管理员和内核提供策略或由远程接收的证书指定，用伪设备完成策略处理和内核的通信。每一个连接关联的策略内容是用 KeyNote 库处理的。所需的认证也能通过查询远端策略池如 web server 或 LDAP server 来获取。远端策略池将策略内容中的 ID 作为密钥。

5.2 防火墙集群系统

从上文可以看出，构建分布式防火墙是相当复杂的。而且目前还不能实现在分布式防火墙上加载入侵检测模块。所以，利用集群系统实现多台防火墙并行工作是未来防火墙高可用性发展的目标。

5.2.1 防火墙集群系统的实现方式

在防火墙集群系统中，要求多台防火墙功能对称并且访问控制规则完全相同。按照负载均衡机制的分类，有两种实现方式。[23,35]

1．利用用户或子网实现静态负载均衡。对于存在用户概念的防火墙系统，内部主机通过认证访问外部网，因而可以人为地将用户平均分配给每台防火墙，要求某些用户固定从一台防火墙上访问外部网。如果防火墙系统不支持用户，也可将内部网络划分为多个子网。同样，可以规定某一子网的计算机固定通过一台防火墙访问外部。

2．利用均衡器实现动态负载均衡。在防火墙系统中加入均衡器后，可以根据防火墙当前的连接数、CPU 利用率、内存占用情况、固定时间内 I/O 次数等作为负载判定依据。均衡器可以分为硬件和软件。

- 硬件均衡器：在网络中无需修改防火墙集群环境，便可将硬件均衡器放置在外部网和防火墙集群间。当硬件均衡器监测和定向流量时，并不影响防火墙上的应用程序处理数据。由于是独立设备，容易升级、维护。但是硬件均衡器也有不足之处，如尽管均衡器处理复杂度比防火墙轻，也会产生单点失效，只能通过双机热备份解决。另外，硬件均衡器获取防火墙的性能信息是有限的。通常情况，很难得到防火墙的 CPU 利用率和内存占用情况。
- 软件均衡器：软件均衡器有两种装载方式：a) 仅装载到集群中的一台防火墙上，那

么这台防火墙在集群处在主机位置；b) 装载到集群中的每一台防火墙上，可以更进一步的分析防火墙的操作系统，获取更多有关防火墙的性能信息。软件均衡器在处理资源分配时，会降低防火墙的性能，最糟糕的情况是使得防火墙耗竭。另外，软件均衡器依赖防火墙操作系统实现，所以不易扩展、升级。

5.2.2 防火墙集群系统的开发原型

由于高可用性是未来防火墙发展目标，所以如何设计软件均衡器是下一步防火墙负载均衡模块的重点。

作者经过半年的研究，给出一种原型设想。

1. 设计原则

- 防火墙采用状态检测技术。
- 防火墙集群系统中所有防火墙是同构的，管理各防火墙的访问规则配置，验证规则也是一致的。
- 防火墙的负载均衡模块包括监控器和应答器两部分。
- 在防火墙集群系统中设定其中一台机器为主防火墙。在初始状态，由管理员直接设定主防火墙。进入工作状态，则由监控器根据每台防火墙的负载决定哪台防火墙为主机。
- 监控器始终工作在主防火墙上。每隔一段给定时间，当监控器询问集群中每台防火墙的负载情况时，相应防火墙的应答器就将当前负载返回给监控器。
- 在防火墙集群系统中，只允许主防火墙接收新连接，其余的防火墙仅保持已建立连接的传递工作。
- 将主防火墙上的连接跟踪表复制到集群中的每一台机器上。
- 用心跳检测技术定时检测集群中每一台机器的健康情况。
- 对监控器和应答器之间的包进行加密。

2. 工作状态转换图

如图 5-3，监控器端主要包括四个状态：启动、监控、状态保存和错误恢复。启动时监控器初始状态，负责读入系统配置文件。监控状态负责对各个防火墙的负载定期检测，如 cpu、系统 I/O 树等。状态保存是监控器定期保存各个防火墙的连接表。错误恢复是当监控器检测初某台防火墙出错时，对该防火墙上的连接根据保存的状态转移到最近的防火墙上。

应答器端包括两个状态：启动和状态返回。启动负责和监控器建立连接；状态返回和监控状态相联系，返回当前的系统状态。

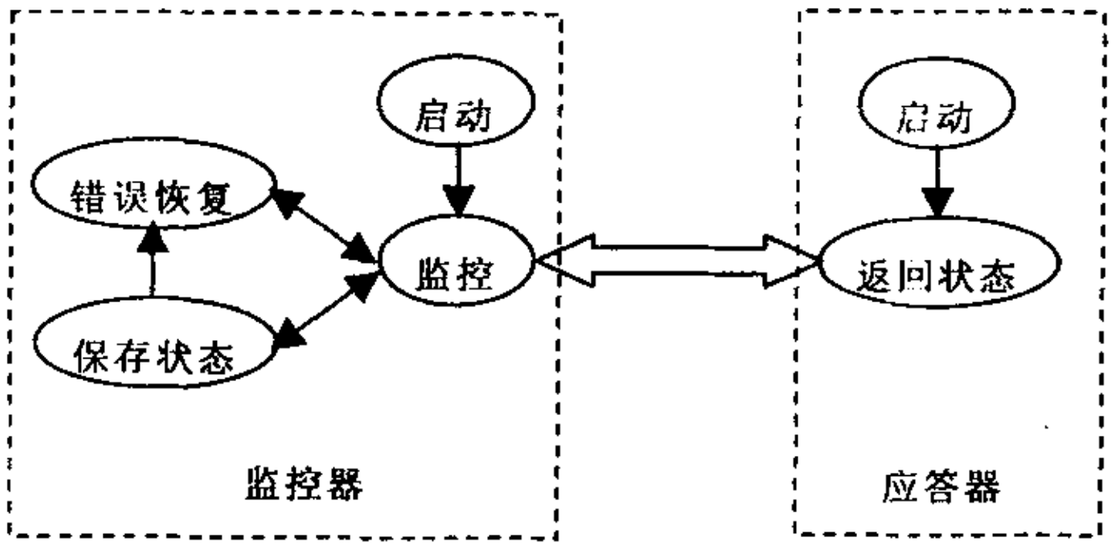


图 5-3 防火墙集群系统状态转换

3．防火墙集群的负载调度

（1）负载调度模块的设计思想

调度模块的主要任务是：

- 监控器向集群中的各台防火墙发送收集负载信息命令；
- 各台防火墙分别运行取 cpu 运行队列长度的程序；
- 各台防火墙上的应答器将各自 cpu 运行队列长度信息传给监控器；
- 监控器对各台机器的 cpu 运行队列长度进行比较并选出 cpu 运行队列长度最短的机器为负载最轻的机器；
- 根据负载，将原主防火墙的系统配置到负载最轻的机器上，并将该机器变为主防火墙。

（2）负载调度模块实现方法

利用 Server-Client 方法，监控器可以从应答器上收集到负载信息。将监控器设为 Client 方，收集应答器上的负载信息并进行处理；应答器设 Server 方，收集本机负载信息并传向应答器。如图 5-4，给出 Server-Client 流程。

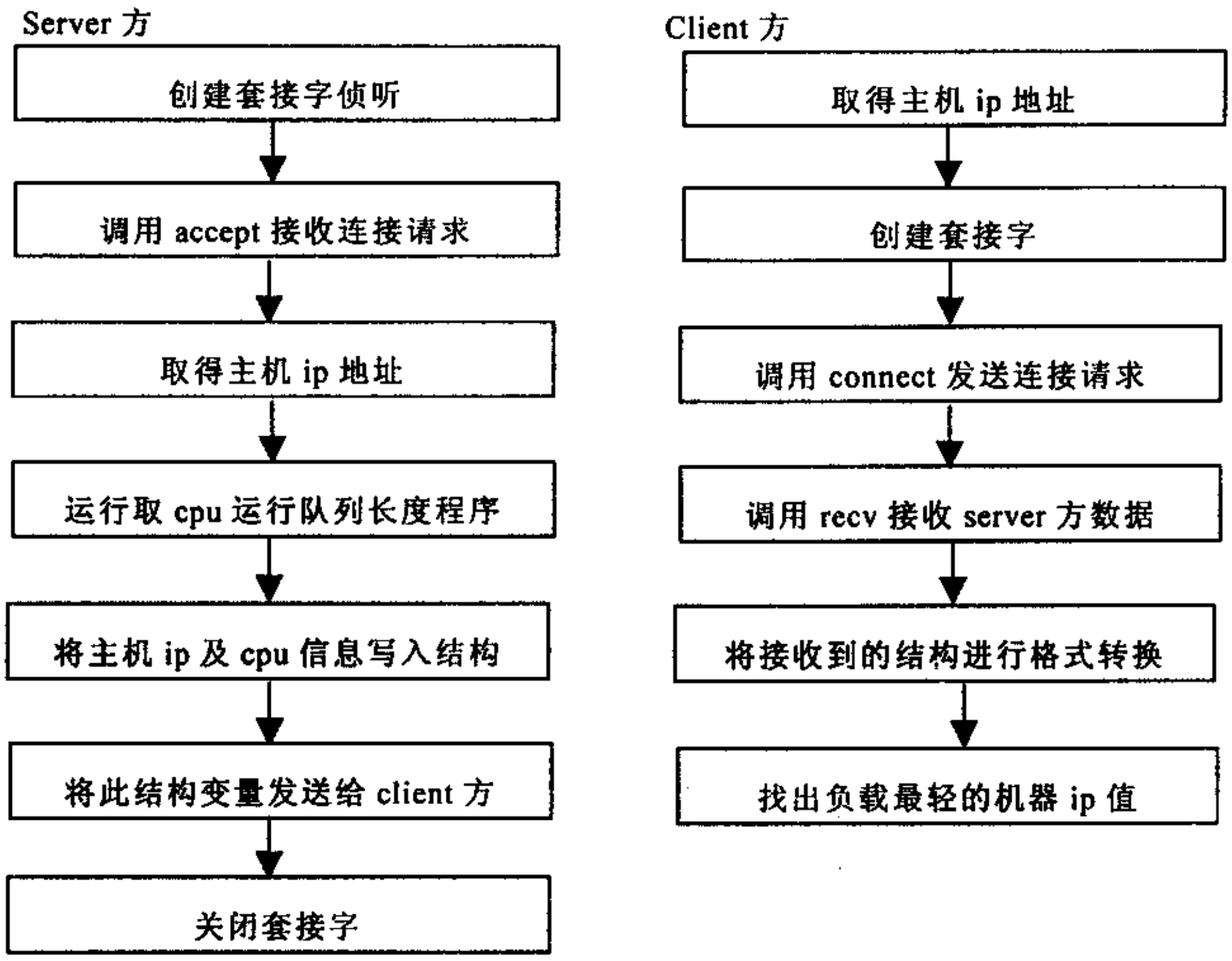


图 5-4　Server-Client 流程

（3）难点

连接跟踪表作为状态检测防火墙的核心，极大地提高了防火墙的性能。但是，在防火墙集群系统中，所有的机器都保持相同的连接跟踪表，它就要求集群系统能够复制连接跟踪表。然而，众所周知，通常连接跟踪表是庞大的，频繁地复制连接跟踪表，反而会降低系统的性能，造成更大的网络拥塞。为此，如何解决连接跟踪表的复制问题是防火墙集群系统的关键，也是未来实现高可用性防火墙的重点。

目前在防火墙市场上，仅有个别国外防火墙厂商通过软件均衡器实现了防火墙自身负载均衡。随着高性能成为未来防火墙追求的目标，国内厂商也逐步将此提上研究日程。本课题的研究为今后更进一步提高防火墙高可用性打下良好基础。

6 总结与展望

6.1 总结

本文以防火墙为技术核心，重点研究、设计与实现了基于负载均衡机制的防火墙的相关核心技术：状态检测技术，LVS 集群技术，地址转换与负载均衡，数据同步。提出了提高防火墙高可用性的解决方案：服务负载均衡模块、双机热备份模块和防火墙集群系统。其中，服务负载均衡模块和双机热备份模块已进入实际测试阶段。

6.2 展望

1．防火墙向高性能发展。由于安全与性能是一对矛盾，所以高性能是防火墙的未来发展目标。

2．向商务型防火墙方向发展。互连网络带来了产业革命，企业 IT 化正蓬勃发展，电子商务在企业中会扮演十分重要的角色。防火墙真正的大市场在于对电子商务的支持。

致谢

在此毕业论文完成之际，我首先要感谢我的导师马光思教授。在攻读硕士的两年半中，给予我教诲、关心和帮助。马老师渊博的知识、严谨的治学作风都跟我留下了深刻的印象。马老师倡导的自由的研究学风，以及他开朗豁达、平易近人的性格必将会对我将来的学习和工作产生深刻的影响。今后我只有努力的学习工作，才能无愧于马老师的教诲。

在课题的整个研究过程中，刘向武研究员、刘文清博士给了我无私的帮助。两位老师的严谨、广博使我受益匪浅，在此表示深深的谢意。

同时感谢北京亿阳信通股份有限公司安全事业部防火墙组的同事。他们在我实习期间给了我支持和帮助。

最后感谢一直关心、鼓励我的家人以及所有老师和朋友们。

参考文献

[1] 楚狂：网络安全与防火墙技术，人民邮电出版社，P1-4，P190-225，2000年4月

[2] 博嘉科技：《Linux 防火墙技术探秘》，国防工业出版社，P301-307，2002年10月

[3] Terry William Ogletree 著，李之棠等译： 防火墙原理与实施，电子工业出版社，2001.2

[4] 杨辉，吴昊：《防火墙-网络安全解决方案》，国防工业出版社，P33-36，2001年9月

[5] NAT 原理及其注意事项，http://www.net130.com/netbass/Route/r120602.htm

[6] Bradner,S.：The Internet Standards Process –Revision 3，BCP 9，RFC2026，October 1996

[7] Egevang, K. and P. Francis：The IP Network Address Translator，RFC1631，May 1994

[8] 黄登玺：基于负载均衡地防病毒防火墙地设计和实现，P1-4，2002年5月

[9] Jie Wu（美）.高传善等译：分布式系统设计[M]，机械工业出版社，2001.

[10] 刘皓：负载均衡技术概览，http://www.ccidnet.com/tech/network/2001/04/30/network/2001/04/30/58_2105.html，2001.

[11] 谢军安： Quidway S8016 的内容交换技术，华为技术报，第 148 期，2002.07.10

[12] LinuxAid：多层负载平衡之四， 2002.08.23 http://www.neweasier.com/article/2002-08-23/1030092170.html

[13] Rajkumar B： High Performance Cluster Computing. Architectures and Systems，Vol1，Prentice Hall PTR，2000

[14] 章文嵩： Linux 服务器集群系统（一），http://www-900.ibm.com/developerWorks/cn/linux/cluster/lvs/part1/index.htm

[15] J.H. Howard. An Overview of the Andrew File System. In Proceedings of the USENIX Winter Technical Conference, Dallas, TX, USA, February 1998.

[16] Kenneth W. Preslan, Andrew P. Barry, Jonathan E. Brassow, Grant M. Erickson, Erling Nygaard, Christopher J. Sabol, Steven R. Soltis, David C. Teigland, and Matthew T. O'Keefe: A 64-bit, Shared Disk File System for

Linux. In Proceeding of 16th IEEE Mass Storage Systems Symposium， San Diego， CA， USA. March 15-18,， 1999.

[17] Global File System Website， http://www.globalfilesystem.org.

[18] Coda File System Website， http://www.coda.cs.cmu.edu.

[19] InterMezzo File System Website，http://www.inter-mezzo.org

[20] 胡季敏，雷乃旺：使用动态负载均衡技术的 LINUX 高性能集群服务器研究，微电脑应用，Vol.17，No.4，2001

[21] 肖钧，庞丽萍： LINUX 虚拟服务器 WRR 调度算法的优化，华中科技大学学报，第二卷 第 2 期 2001 年 2 月

[22] Zhiruo Cao, Zheng Wang: Performance of Hashing-Based Schemes for Internet Load Balancing， INFOCOM 2000，Volume 1

[23] 吴昕，李之棠：并行防火墙研究，计算机工程与科学，2000 年第 22 卷，第 2 期

[24] 赵征，马光思：负载均衡在防火墙的应用，西安建筑科技大学学报， 2002 年第 4 期

[25] Rusty Russell：Linux 2.4 Packet Filtering HOWTO，
Mailing list netfilter@lists.samba.org，vision: 1.22，Date: 2001/11/20

[26] Rusty Russell： Linux netfilter Hacking HOWTO，
Mailing list netfilter@lists.samba.org，vision: 1.11，Date: 2001/10/31

[27] Paul "Rusty" Russel：a Module for netfilter in Linux Magazine，June 2000
http://www.linux-mag.com/2000-06/gear_01.html

[28] Iptables Tutorial 1.1.16，
http://iptables-tutotial.frozentux.net/chunkyhtml/userlandstats.html

[29] Connection tracking，
http://www.xinux.de/docs/sicherheit/firewall/iptables/connect-track

[30] hi xueyong：项目开发计划，2000 年 3 月 15 日

[31] 用 iptables 实现 NAT，http://LinuxAid.com.cn

[32] 蒙杨：高安全等级防火墙核心技术研究、设计与实现， 2001 年 6 月

[33] Bellovin S M.：Distributed Firewalls，login: magazine, special issue on security. November 1999

[34] Wei Li： Distributed Firewall，
http://www.cs.helsinki.fi/u/asokan/distsec/documents/li.ps.gz,2000

[35] 赵征： 分布式防火墙技术与实现， 2002 年全国开放式分布与并行计算学术会议论文集, 华中科技大学出版社，2002

[36] S.Loannidis and A.D.Keromytis.: Implementing a Distributed Firewall, http://www.cis.upenn.edu/，2000

[37] Daniel Wan ： Distributed Firewall，http://www.sans.org/rr/firewall/dist.php May 28，2001

注释

①生成树协议——STP　STP(Spanning Treep Protocol)能够提供路径冗余，使用 STP 可以使两个终端中只有一条有效路径。STP 在大的网络中定义了一个树，并且迫使一定的备份路径处于备用状态。如果生成树中的网络一部分不可达，或者 STP 值变化了，生成树算法会重新计算生成树拓扑，并且通过启动备份路径来重新建立连接。STP 操作对于终端来说是透明的，而不管终端连在 LAN 的某一部分或者多个部分。当创建网络时，网络中所有节点存在多条路径。生成树中的算法计算出最佳路径。因为每个 VLAN 是一个逻辑 LAN 部分，所以网管员能使 STP 一次工作在最多 64 个 VLAN 中。如果要配置超过 64 个 VLAN，网管员需要将其他 VLAN 的 STP 禁止，因为默认的 STP 可以支持 1-64 个 VLAN。

附录 1　Netfilter 框架

以 ip_iput.c 中的 ip_rcv()函数为例。它调用了定义在 NF_IP_PRE_ROUTING 点上的防火墙处理函数。可在 Netfilter.h 中给出了 NF_HOOK()函数的原始定义。而在 Netfilter.c 中，提供了处理 hook 点函数。对于 netfilter 所提供的框架，这些钩子函数都将去执行一个重要的流程，那就是 ipt_do_table()函数，该函数执行完成后，将返回通用的防火墙策略之一，如 NF_ACCEPT 等。

Ip_input.c

```
Ip_rcv()
    Return    NF_HOOK(PF_NET,NF_IP_PRE_ROUTING,SKB,    dev,    NULL,
ip_rev_finish)
```

Netfiter.h

```
/*如果在编译内核时没有配置 netfilter 时，则调用(okfn)(skb)，此例中即执行 in_rev
_finish 函数；否则进入 HOOK 点，执行 nf_register_hook()登记,实际上是进入
nf_hook_slow()，再由它执行登记的函数*/
#ifdef CONFIG_NETFILTER_DEBUG
#define NF_HOOK nf_hook_slow
#else
#define NF_HOOK(pf, hook, skb, indev, outdev, okfn)  \
(list_empty(&nf_hooks[(pf)][(hook)])                          \
 ? (okfn)(skb)                                  \
 : nf_hook_slow((pf), (hook), (skb), (indev), (outdev), (okfn)))
#endif


int nf_hook_slow(int pf, unsigned int hook, struct sk_buff *skb,
            struct net_device *indev, struct net_device *outdev,
            int (*okfn)(struct sk_buff *));
struct nf_hook_ops{
      struct list_head list;   /*初始化总为 {NULL, NULL} */
      nf_hookfn *hook;
      int pf;                  /* 总为 PF_INET*/
      int hooknum;             /* 所选择的 HOOK 点*/
      int priority;            /*挂多个处理函数，需要指定优先级*/
};
```

Netfiter.c

```
Struct list_head   nf_hooks[NPROTO][NF_MAX_HOOKS]
nf_register_hook( nf_hook_ops *reg);
nf_unregister_hook();
```

位于各 Hook 点上的防火墙处理函数，通过它们完成添加、删除工作

```
nf_iterate()   /*负责执行 Hook 上的函数,返回 NF_ACCEPT, NF_QUEUE 、
                NF_STOLEN、 NF_DROP, NF_REPEAT*/
nf_hook_slow(){    /*根据 nf_hook[]数组，开始处理 packet*/
       br_read_lock_bh(BR_NETPROTO_LOCK);  /*本地 CPU bh 屏蔽*/
         elem = &nf_hooks[pf][hook];                    /*找到对应的 hook 链表头*/
         verdict = nf_iterate(&nf_hooks[pf][hook], &skb, hook, indev, outdev, &elem,
okfn);
           ...
         /*根据返回的信号，执行相应的操作：*/
         NF_QUEUE: nf_queue(skb, elem, pf, hook, indev, outdev, okfn);
         NF_ACCEPT: okfn(skb)执行本来正常的的操作
         NF_DROP: kfree_skb(skb)释放此数据包;
           }
```

攻读硕士学位期间发表的论文

1 赵征，于波，马光思，分布式防火墙技术及实现，2002 年全国开放式分布与并行计算学术会议论文，华中科技大学出版社，2002

2 赵征，马光思，负载均衡机制在防火墙中的应用，西安建筑科技大学学报（自然科学版），2002 年第 4 期

3 赵征，用数组方法实现 Web 管理方式下的动态网页设计，华南金融电脑，2002 年第 10 期

4 赵征，OLAP 技术及其在金融领域中的应用，华南金融电脑，2002 年第 12 期

5 于波，赵征，唐世渭，OLAP 中的 CUBE 计算问题，计算机应用，2003 年第 1 期