P2P 网络 Gnutella 协议中防火墙穿越的研究与实现

检测技术与自动化装置

硕 士 生：朱志伟

指导教师：吕志民 教授

摘 要

计算机对等联网 P2P 技术是目前新一代网络技术研究的活跃领域，它在网络资源利用率、消除服务器瓶颈等多方面都有明显的优势。而 Gnutella 协议就是一个基于 P2P 网络的文件搜索，数据交换协议。

本文通过分析了 Gnutella 协议的包结构，路由规则以及文件的传输，针对协议本身只能处理单方节点被防火墙屏蔽的问题的不足，提出了一个解决方案，通过一个代理节点进行文件的代理传输，使协议能够有效穿越屏蔽了双方节点的防火墙。为此，本文对协议进行了必要的扩展，增加了动态确定代理节点的消息，设计了这些消息的结构，给出其路由规则，并为文件的代理转发新增加了必要的 HTTP 消息。

本文在 Linux 平台上对协议扩展进行了实现，最后，对系统的实验结果进行了介绍和分析，并指出本系统的缺点和不足以及下一步的研究工作。

关键词： P2P，Gnutella，防火墙，穿越，代理

Research and Implementation of firewall traversing in P2P network Gnutella protocol

Detection Technique and Automatic Device

Name: Zhu Zhi Wei

Supervisor: Professor Lǚ Zhiming

ABSTRACT

With the high resource using rate and the advantage of eliminating the server's chock point, Peer-to-Peer (P2P) Network becoming the activity field in the new generation's network research. And Gnutella protocol is one of P2P network protocol for file searching and data exchange.

This article first analyzes the protocol's messages structure, routing rule, file transfer. For the protocol can only traverse one way firewall, this paper brings forward a solution to solve this problem by using a proxy node to transfer the data. For this, this paper make some extension for the protocol, adding some new message for dynamic confirm the proxy node and design the structure of these messages, giving out their routing rules. For file transferring, this paper also adds some new HTTP message.

The application for this proposal is based on Linux. At last, this paper presents and analyzes the result of this proposal, pointing out some deficiency of this system and the next research work.

Key Words: P2P, Gnutella, firewall, traverse, proxy

第1章 引 言

1.1 课题研究的背景

一直以来，在 Internet 上占据着统治地位的是 C/S(Client/Server) 结构。该结构是以一台机器作为服务器提供服务，其他机器作为客户机向服务器提出服务请求，服务器响应请求。这样的应用必须要设置一个拥有强大处理能力和大带宽的高性能计算机，配合高档的服务器软件的服务器，信息通过服务器才可以传递。信息一般是先集中上传到服务器保存，然后再分别下载（如网站），或是信息按服务器上专有规则处理后才可在网络上传递流动（如邮件）。C/S 模式适合一对多、强对弱的社会关系形式，如政府对个人、对企业，大企业对小企业，学校对学生，企业对职工等等关系。

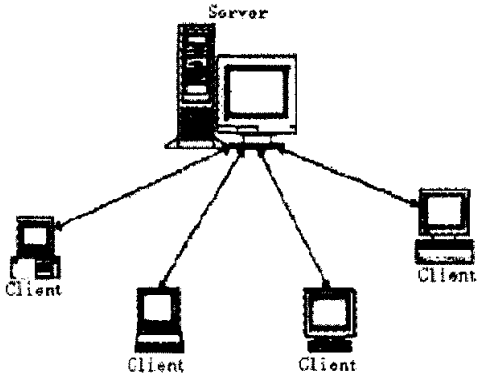

图 1.1 C/S 模式的网络结构

但是，自从 1999 年的 Napster(一个提供可以在网络上交换 MP3 的软件)的出现，把人们的目光又吸引到网络的另一种结构：P2P（peer-to-peer network）网络。对等网络打破了传统的 C/S 模式，对等网络中每个节点的地位都是相同的，每个节点既充当服务器，为其他节点提供服务，同时也充当客户机，享用其他节点提供的服务。在 P2P 技术的推动下，互联网的存储模式将由现在的“内容位于中心”模式转变为“内容位于边缘”模式。[1]

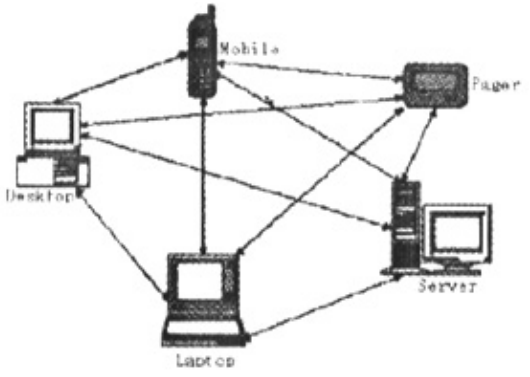

图 1.2 P2P 模式的网络结构

正是由于 P2P 网络结构相对于 C/S 结构的种种优点，P2P 技术正在受到人们的越来越广泛的关注，特别是基于 P2P 网络的文件交换协议 Gnutella 协议。许多著名的公司和企业也先后投入到对 P2P 技术的研究之中，其中包括微软, SUN, IBM 和惠普等。

1.2 课题研究的现状

正是 P2P 网络具有自己独特的分布式的网络结构，它在以下几方面得到广泛应用：[2]

(1) 对等计算

采用 P2P 技术的对等计算，是把网络中的众多计算机暂时不用的计算能力连结起来，使用积累的能力执行超级计算机的任务。使它们发挥出只有超级计算机才具有的巨大功能。从本质而言，对等计算就是网络上 CPU 资源的共享。

(2) 协同工作

P2P 技术的出现，使得互联网上任意两台 PC 都可建立实时的联系，可以建立一个安全、共享的虚拟空间，人们可以进行各种各样的活动，这些活动可以是同时进行，也可以交互进行。在协同工作的过程中，P2P 体现出了一些良好的素质。

(3) 搜索引擎

强大的搜索引擎功能是 P2P 软件引以为自豪的一个功能。P2P 技术使用户能够深度搜索文档，而且这种搜索无需通过 Web 服务器，也可以不受信息文档格式和宿主设备的限制，可达到传统目录式搜索引擎无可比拟的深度。P2P 为互联网的信息搜索提供了全新使用感受和信息搜索提供了全新的解决之道。著名的搜索引擎公司 Google 也宣称要采用 P2P 技术来改进其搜索引擎。

(4) 文件交换和文件共享

在传统的 Web 方式中，要实现文件交换需要服务器的大力参与，通过将文件上传到某个特定的网站，用户再到某个网站搜索需要的文件，然后下载。电子邮件显然方便地解决了个人间文件传递问题，却没法解决大范围的交换，这就是 Web 的重要缺陷，Napster[3]就是在情况下出现，抓住人们对 MP3 喜欢的需求，Napster 的 MP3 交换直接引发了网络的 P2P 技术革命。文件共享是 P2P 软件最实质的使用宗旨。

(5) 企业应用

应用 P2P 技术的互联网产品正在迅速开辟出一块新的互联网应用市场，例如 ICQ 类的即时信息工具不仅创立了一个巨大市场。在 P2P 网络上开展电子商务这另是人们根据 P2P 本身的特点就能直接想到的，而且这种模式尤其适于用户之间的商品买卖。P2P 在电子商务领域有光明的前景，因为这项技术是直接联系买主与卖主方面，不需要第三者参与。

在文件交换和文件共享方面，目前最为流行的两种 P2P 网络模型结构分别是基于集中式目录结构和基于分布式定位方法的对等网络模型，它们分别以 Napster 和 Gnutella 为代表。其中 Napster 模型因为中央服务器的存在被认为并非纯粹的 P2P 系统，而取消了中央服务器的 Gnutella 模型则被作为纯粹的 P2P 系统的代表。

Gnutella 协议实现的是纯粹的 P2P 模式，它能智能地发现其邻居节点，主要用于文件的搜索和交换上。在因特网上的任何一台机器，只要运行一个基于 Gnutella 协议的 P2P 软件，就可以成为 Gnutella 网络中的一个节点。

但是，Gnutella 协议正处于不断完善的阶段，并且出于协议的简单性方面的考虑，Gnutella 协议有许多方面都还没有给出完善的解决方案，许许多多研究人员与研究机构都在为完善该协议做出努力。譬如在安全方面，由于协议本身的简约性，Gnutella 网络很容易遭到分布式拒绝服务攻击（DDoS），国外有研究人员提出在两台客户机文件下载前，互相传递多一对 Query/QueryHit 数据包作验证，可以有效避免利用 IP 欺骗造成的分布式攻击[4]；SUN 公司甚至推出了一个全面的开发平台 JXTA，为 P2P 网络的开发，安全保证提供完善的平台[5]。

在用户的交互上，处于不同 NAT 之后的两台机器为进行通信，必须解决地址的映射问题，有研究人员提出了使用 UDP 进行 NAT 的穿越[6]。协议本身对单机被防火墙屏蔽有自己的解决方案，而当两台机器都处于防火墙后，协议对两台机器间的文件下载就无能为力了。通过代理机可以解决这个问题，但是，一般的代理机制会造成通讯效率的低下，如何高效地穿越防火墙，是 Gnutella 协议得以广泛应用的保证。国内的一位研究人员提出了一种新的 P2P 网络代理协议，能够高效地穿越防火墙，扩充了 P2P 网络的连接能力[7]。但是，在其提出的方法中并没有给出明确的路由规则与代理节点的选择方案。

1.3 课题研究的意义和目标

另一方面，在各个企业，学校的内部网中，出于安全原因，都会在内部网与外部网的网关中设置防火墙。这样，统一由网关来设置防火墙，能有效防止外部的恶意侵入，又能减轻内部网络主机的设置负担，不用每一台主机都配置防火墙，减低了安全成本，将安全性集中于一点。但是，统一的策略也带来了新的问题，主要是限制了有用的网络服务。防火墙为了提高被保护网络的安全性，限制或关闭了很多有用但存在安全缺陷的网络服务。譬如，内部网中主机所开的 ftp 服务往往就被防火墙屏蔽掉，造成网内主机与外部通信的极大不便。同样，Gnutella 协议也会遭遇同样的问题。为能最大限度的利用 Internet 上的资源，提高用户交互的方便性，Gnutella 协议中对防火墙的穿越是值得考虑的问题。

本课题就是针对 P2P 网络上的 Gnutella 协议中的防火墙穿越方案的不足，提出一个有效的穿越方案。该协议本身对单机被防火墙屏蔽有自己的解决方案，

而当两台机器都处于防火墙后，协议对两台机器间的文件下载就无能为力了。本文提出了一个代理方案，对协议进行扩展，增加了在 Gnutella 网络上动态确立文件下载的代理机的协商消息，以及这些消息的路由规则。目标在于使两台处于不同防火墙屏蔽的机器能够动态确立一台代理机，使这两台机器能够有效进行文件传输，实现 P2P 网络的最大交互。

1.4 论文的结构概况

第一章 介绍了课题研究的背景、现状、意义和目标。

第二章 介绍了 P2P 网络相对于 C/S 网络的优点，对当前的 P2P 网络的几种基本结构进行了介绍和比较，以及 P2P 的关键技术等。

第三章 详细介绍 P2P 网络中主要用于文件共享的协议 Gnutella 协议，描述了它的协议包格式，路由规则，文件下载，协议本身所有的单方防火墙穿越方案。

第四章 给出了本文提出的双方防火墙穿越方案，给出了对协议进行扩展的数据包格式，路由规则，以及方案的具体实现。

第五章 给出了在实验室环境下搭建的小型网络，对该网络进行防火墙策略配置，测试本代理方案的实现结果。并且分析了该实现的特点和不足,提出下一步工作展望。

第2章 P2P 网络结构

2.1 P2P 的基本概念

P2P 是 Peer to Peer 的简写，译为“对等网络”。P2P 系统中没有客户机和服务器之分，每个主机既是客户机也是服务器。它使互联网上的个人电脑相互平等地连接，在进行信息交换时无须通过任何服务器和公司。P2P 使用户感觉到在进行文件交换时象以往登录网络一样方便。这项技术主要应用于企业内部的局域网或局域网之外的任何特定人群间的对等交流。[8]

2.2 P2P 网络结构的优点

目前最流行的网络计算模式是 C/S 模式，该结构具有如下特点：[9]

（1） 集中计算方式，信息和数据都保存在服务器端。只有服务器端具有控制能力，客户端基本上只是一个高性能的 I/O 设备。

（2） 服务器及网络的带宽决定了网络的性能。每台服务器所能提供的信息数量受到自身存储空间的限制，而任意时刻它所能支持的客户端访问数量则既受到自身处理能力的限制也受到服务器所在网络吞吐能力的限制。

（3） URL 用来表示信息资源的地址，但是 URL 很少能直接体现所定位的信息的内容，甚至不能直接链接到具体的内容上。

（4） 被发布信息的分布与生存期十分稳定。服务器只发布机器所有者想公之于众的信息，这些信息将会在该服务器上稳定地保存一段时间，并且该服务器通常也不间断地运行在网络上。

（5） 被发布信息的存贮与管理比较集中、规范。互联网上所有可以公开访问的信息基本上都保存在服务器上，服务器根据适当的算法和规则管理本地信息，应答客户端的访问请求或进行计算。

相对 C/S 网络而言，P2P 模式有以下一些主要优点：

（1） P2P 模式最主要的优点就是资源的高度利用率。在 P2P 网络上，闲散资源有机会得到利用，所有节点的资源总和构成了整个网络的资源，整个网络可以被用作具有海量存储能力和巨大计算处理能力的超级计算机。C/S 模式下，纵然客户端有大量的闲置资源，也无法被利用。

（2） 随着节点的增加,C/S 模式下，服务器的负载就越来越重，形成了系统的瓶颈，一旦服务器崩溃，整个网络也随之瘫痪。而在 P2P 网络中，每个对等体都是一个活动的参与者，每个对等点都向网络贡献一些资源，如存储空间、CPU 周期等。所以，对等点越多，网络的性能越好，网络随着规模的增大而越发稳固。

（3） 基于内容的寻址方式处于一个更高的语义层次,因为用户在搜索时只需指定具有实际意义的信息标识而不是物理地址，每个标识对应着包含这类信息的节点的集合。这将创造一个更加精炼的信息仓库和一个更加统一的资源标识方法。

（4） 网络设备间直接流动，高速及时，降低中转服务成本。C/S 模式下的互联网是完全依赖于中心点——服务器的，没有服务器，网络就没有任何意义。而 P2P 网络中，即使只有一个对等点存在，网络也是活动的，节点所有者可以随意地将自己的信息发布到网络上。

表 1 是对 C/S 和 P2P 模式在若干方面的比较[10]。

表 1 P2P 模式与 C/S 模式比较

	P2P	C/S
数据发布	好	差
数据接收	中	好
数据安全性	差	好
数据更新	好	差
数据质量(价值)	中	好
数据覆盖率和数量(价值)	差	好
数据成本控制	好	差
数据管理方便性	差	好

2.3 P2P 网络结构

目前，P2P 网络主要存在三种网络结构[11]

(1) 集中式 P2P

集中式 P2P 形式有一个中心服务器来负责记录共享信息以及回答对这些信息的查询。每一个对等实体对它将要共享的信息以及进行的通信负责，根据需要下载它所需要的其它对等实体上的信息。这种形式具有中心化的特点，但是它不同于传统意义上的 Client/Server 模式。传统意义上的 Client/Server 模式采用的是一种垄断的手段，所有资料都存放在服务器上，客户机只能被动地从服务器上读取信息，并且客户机之间不具有交互能力。而集中式 P2P 形式则是，所有网上提供的资料都分别存放在提供该资料的客户机上，服务器上只保留索引信息，此外服务器与对等实体以及对等实体之间都具有交互能力。

采用集中式 P2P 形式的软件被称为第一代 P2P，其代表性软件为 Napster。Napster 的系统结构图见图 2。这种形式有一个中央服务器，为用户提供共享和搜索文件服务。这就要求有一个连续运转的服务器，并且一旦该服务器被关闭，整个网络就会停止运行。此外，这样的服务器必须能够处理大量的用户连接，拥有足够的内存和磁盘空间来维护和搜索文件列表。

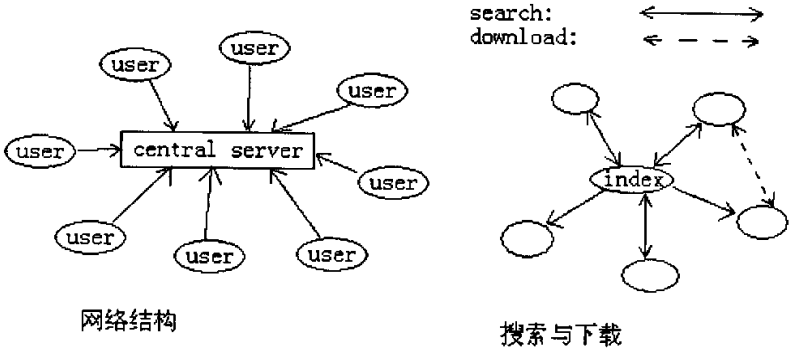

图 2.1 集中式 P2P 模式（Napster）

(2) 分布式 P2P

分布式 P2P 形式是一种纯 P2P 模式。这种形式不需要有中心服务器和中心路由器，其上的每一个 Peer 都作为对等实体，地位是完全平等的。每一个 Peer 既可

以作为客户机又可以作为服务器，并且它们与相邻的 Peer 有相同的能力。具体地说，它有两种路由结构，一种是分布式的目录结构，另一种是直接消息传递。像 Gnutella [12-14]就是这种形式的代表。

采用分布式 P2P 形式的软件被称为第二代 P2P。其代表性软件有 Gnutella, Bearshare, Limewire 等。Gnutella 的系统结构图见图 3。这些软件都是基于相同的运作模式，即没有中央服务器而是运用了分布式P2P技术，网络上的每一个结点都连接一些其他的结点。尽管 Gnutella 很好地解决了中心化的问题，但是搜索请求要经过整个网络或者至少是一个很大的范围才能得到结果，正因为如此，这种模式占用很多带宽，而且需要花费很长时间才能有返回结果。同时，这种纯分布式的P2P模式很难被企业所利用，因为它缺少对网络上的用户结点数以及他们提供的资源的一个总体把握。针对以上提出的分布式 P2P 模式的不利因素，提出了混合 P2P 形式，使 P2P 模式的系统结构更加合理和有效率。

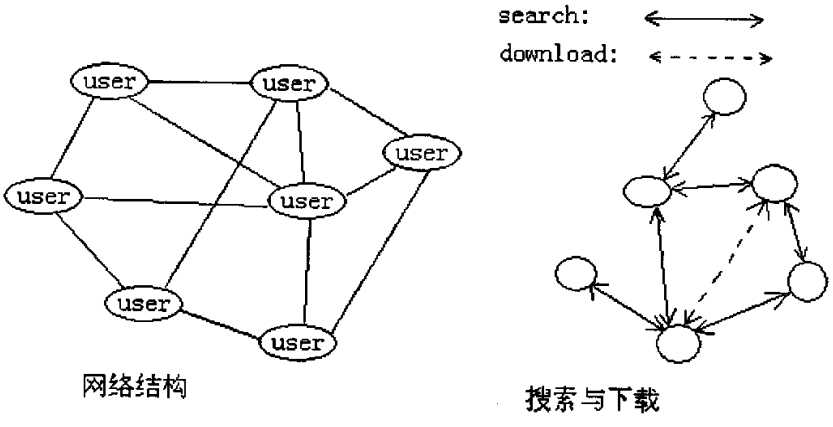

图 2.2 分布式 P2P 网络模式（Gnutella）

(3) 混合式 P2P

集中式 P2P 形式有利于网络资源的快速检索，以及只要服务器能力足够强大就可以无限扩展，但是其中心化的模式容易遭到直接的攻击；分布式P2P形式解决了抗攻击问题，但是又缺乏快速搜索和可扩展性。混合 P2P 形式结合了集中式和分布式 P2P 形式的优点，在设计思想和处理能力上都得到近一步优化。它在分布式模式基础上，将用户结点按能力进行分类，使某些结点担任特殊的任务。其系统结构图见图 2.3

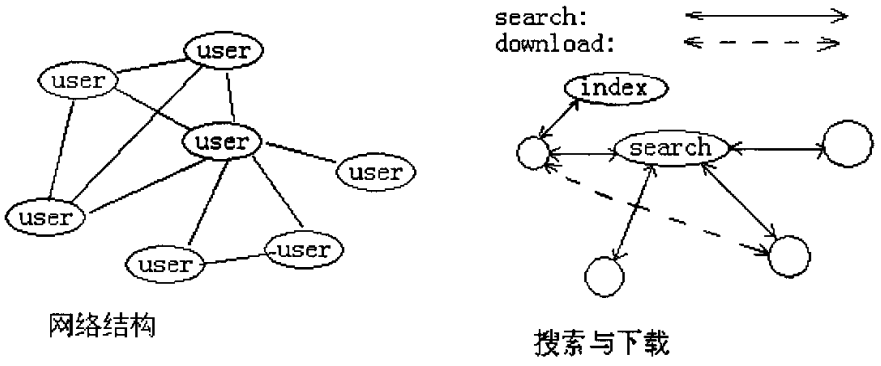

图 2.3 混合式 P2P 模式

2.4 P2P 的关键技术

P2P 网络中，主要考虑到以下几个问题：[15]

（1） 拓扑一致性和资源定位。对于互联网上众多计算机，P2P 应用比其他应用要更多考虑低端 PC 的互联，它们不具备服务器那样强的联网能力，同时对于以往的 P2P 应用技术，现在的硬件环境以及更为复杂，这样在通信基础方面，P2P 必须提供在现有硬件逻辑和底层通信协议上的端到端定位（寻址）和握手技术，建立稳定的连接。涉及的技术有 IP 地址解析，NAT 路由及防火墙。

P2P 系统需要解决的一个重要问题使：在一个缺少集中化服务器的动态环境下，各个节点能够维持一致的网络拓扑信息。由于 P2P 网络节点的加入和离开非常频繁，传统路由扩散的方法难以解决这一问题，所以需要一个高效的一致性信息维护机制实现一些功能。例如，当网络拓扑变化时快速恢复网络的稳定性问题需要解决，但多个节点的并发加入和离开使得解决这一问题更具挑战性。另外，用户从大量分散的节点中找到需要的资源和服务也是一个挑战。

（2） 互操作性，数据描述和交换的协议。在应用层面上，如果两个 Peer 分别代表两家不同的公司，而且它们已经通过互联网建立连接，那么一方的信息就必须为另一方所识别，所以当前互联网上关于数据描述和交换的协议，如 XML, SOAP, UDDI 等都是一个完善的 P2P 软件所要考虑的。

（3） 安全加密。P2P 中的安全问题直接决定了 P2P 能否被大规模进行商用，

除了 Freenet 强调 P2P 中的匿名问题之外，大多数系统并没有对 P2P 中的安全问题做太多的工作。P2P 中的安全问题包括信息的加密，用户身份的认证，恶意节点的识别和应对等等。值得注意的是，在 P2P 的分布式环境下，针对单个服务器的拒绝服务攻击将不再有效。有通信就要有安全保障，加密技术是必须要考虑的。

（4） QoS 问题。P2P 网络的 QoS 问题包括两个方面：

1. 信息获取的 QoS 问题，用户需要的信息可能在多个节点同时存放，如何选择一个处理能力强，负载轻，带宽高的节点需要用户考虑。

2. 用户可能共享出无用或者违法信息，造成信息垃圾充斥网络，因此，网络应该控制用户共享的信息，提高用户获得有用信息的效率。

（5）其他问题。其他需考虑的有如何设置中心服务器，如何控制网络规模，改善查询性能等。

2.5 P2P 的搜索算法

在 P2P 系统中，一个用户要共享另一个用户计算机上的资源，不论是文件、存储空间或是计算资源，一个关键的问题是要找到资源所在的目的主机，因此，内容的查询是很多 P2P 系统的核心。从目前的研究现状来看，主要存在 3 种有关内容查询的算法[16]：

(1)集中索引算法(Central index) 以 Napster 系统为代表。在 Napster 系统中，用户都与一个中央服务器相连接，中央服务器上保存了共享文件的索引。由中央服务器对收到的用户请求进行匹配查找，直到找到保存了所需文件的目的用户。然后，由发起请求的用户与目的用户直接进行文件交换。这种算法的不足在于依赖一个集中式的结构，将会影响系统的可扩展性。

(2)洪水消息算法(Flooded requests) 代表系统为 Gnutella。每一个用户消息都将被广播给与该用户直接相连的若干其他用户，这些用户收到消息后，也同样地将消息广播给各自连接的用户，以此类推，直到请求被应答，消息的 TTL 值减少为 0 或超过了最大的广播次数(通常在 5 和 9 之间)。这种算法存在的不足在于需要较大的网络带宽，因此也会影响可扩展性。

(3)文件路由算法(Document routing) 代表系统为 Freenet。算法的特点

是采用基于哈希函数的映射。系统中的每一个用户都有一个随机的 ID 序列号;系统中的每一个文件也有一个 ID 序列号,这个序列号是根据文件的内容和他的名字,经过哈希函数映射得来的。文件发布时,每一个用户都把文件转发到拥有与文件的 ID 最相近的 ID 值的用户上去,直到最接近文件 ID 的用户就是该用户本身。转发过程中每经过的一个用户都将保持该文件的副本。索取文件时,每个用户都将请求消息转发给一个拥有与所需文件 ID 最相近的 ID 的用户,直到文件或文件的一个拷贝被发现为止。Tapestry, Pastry, Chord, CAN 都是采用这种方法的 P2P 系统。这种算法的优势在于可扩展性较好,不足在于他可能导致整个网络分裂成若干彼此不相连的子网络,形成所谓的孤岛,他的查询也要比洪水消息算法等麻烦些。

第3章 P2P 网络 Gnutella 协议分析

3.1 Gnutella 协议概述

Gnutella 起源于 Nullsoft 所做的一个项目，Nullsoft 公司隶属于 AOL，该公司以用 Winamp 和 Shoutcast 的形式带给人们各种听觉乐趣为主要业务。

Gnutella 是一种完全的分布式信息共享技术，是无等级结构的网络。每一个 Gnutella 节点在作为服务器端的同时也是客户端，每个 Gnutella 节点仅知道与它直接相连的 Gnutella 节点，如果其他 Gnutella 节点不响应该节点的 ping 或回复查询，这台节点就不会看到其它 Gnutella 节点。这是一种令人惊奇的匿名机制。它是一种提供信息共享功能的软件，与传统的 Yahoo、sina 等搜索引擎相比，Gnutella 具有更方便的功能。当运行一个 Gnutella 软件并加入到 Gnutella 网络中时，不但可以将希望与他人共享的文件放在网络当中，与他人共享，也可以方便地定义要寻找的文件。哪些文件在何时可以共享，在何时停止共享都可方便地进行控制和管理。

Gnutella 客户端基本上就是一个具有文件共享功能的微型搜索引擎，当在 Gnutella 网络中进行搜索时，如果 Gnutella 网络中有与搜索信息相匹配的内容，将会自动获得该内容的相关信息，并可进行下载和相关处理。

协议最初版本为 0.4 版，目前最新版本为 0.6 版。并且，协议仍处于不断的改进当中，各个对协议的实现软件都对协议进行了不同程度的改进，完善，提出了很多的扩充建议。当然， 正如协议中所要求的：客户端软件允许扩展或者甚至改变协议的一部分（例如对消息进行压缩或编码），但是客户端软件必须总是保持与依照本协议定义的客户端相兼容。例如：假如一个客户端想压缩 Gnutella 协议消息，它必须首先保证和它连接的远端客户能够解压该消息流（在握手阶段协商），否则就只能保持原消息流不被压缩。客户端允许不和某台远程客户连接，假使该远程客户不支持一定特性，但是必须保证 Gnutella 网络不会分化为各个单独的网络。出于某种特别目的而将网络分化当然是允许的，但它自然也就不叫 Gnutella 网络了，而是另外一种网络了。

3.2 Gnutella 协议体系

Gnutella 协议定义了一个纯 P2P 网络的客户机间进行网络通讯的方式。其中包括定义了客户机间的握手规则，客户机进行数据通讯的描述符号集，客户机相互交互的一些路由规则，客户机间文件的上传下载等内容。

3.2.1 Gnutella 协议握手（Handshaking）规则

一个 Gnutella 节点通过与网络中的另外一个 Gnutella 节点建立连接，使其连接到 Gnutella 网络。一旦一个 Gnutella 节点发现了网络中的另一个 Gnutella 节点的地址，它将和这个 Gnutella 节点创建一个 TCP/IP 连接，并且会发送一个格式为：

“GNUTELLA CONNECT/<protocol version string>\n\n”

的 Gnutella 连接请求数据串。

希望接收连接请求的 Gnutella 节点会用下面这个字符串做出回应：

”GNUTELLA OK\n\n”

如果不是采用这个字符串做出回应，那么任何其它的响应都将表明这个 Gnutella 节点不想接收这个请求而建立连接。Gnutella 节点拒绝从外部发来的连接请求的原因有多种，如：它为从外部发来的连接准备的缓冲池已被占用，或者不支持那个发出请求的 Gnutella 节点的版本等。

3.2.2 Gnutella 协议的消息结构[17]

Gnutella 协议定义了 servent(Gnutella 协议中规定的一种名称，相当于节点)

之间在网络中通信的方式。表 3.1 所示的这些描述符在 Gnutella 协议中都做了完整的定义。

表 3.1 Gnutella 中的消息描述符定义

描述符	描述
Ping	用来动态发现网络上的Gnutella主机
Pong	Ping的响应，包括一个连接中的Gnutella servent的地址和关于它在网络中的相关数据
Query	寻找分布网络的主要机制。如果发现和Servent的本地数据匹配，那么收到一个query描述符的这个servent将用一个queryHit进行响应
Query hit	用于响应query请求。这个描述符提供给接受者足够的信息让他知道存在匹配query数据
Client Push request	允许一个在防火墙后面的servent和网络中的其他Gnutella节点交换数据

如果有一个 Gnutella 节点已经成功连接到 Gnutella 网络中，它和其他 Gnutella 节点之间的通信则通过发送和接收 Gnutella 描述符来实现。每个描述符在数据包中的格式说明如下：

（1）Gnutella 的数据包格式

Gnutella 的数据报头格式如下：

消息ID	功能ID	TTL剩余	跳数	数据长度
0~15	16	17	18	19~22

图 3.1 Gnutella 数据报头格式

其中，

*消息 ID：一个 16 位的唯一在网络上标识一个新建消息的字符串(GUID) 。[18]

*功能 ID：定义了这个数据包的消息类型，ID 值与功能名称的对应关系如下所示：

功能名称	ping	pong	Push请求	搜索	搜索结果
ID值	0x00	0x01	0x40	0x80	0x81

图 3.2 Gnutella 数据报头 ID 值

* TTL 剩余：该信息包在被丢弃前，还剩余的跳数，当 TTL=0 时，

信息包将被丢弃。

*跳数：信息包已发生的跳数，当信息包从一个 Gnutella 节点转发到另一个 Gnutella 节点的时候，TTL 剩余值和跳数值必须满足条件：TTL(0)＝TTL(i)＋hops(i),这里的 i>=0

*数据长度：数据长度在 Gnutella 数据报头的最后，定义了紧随其后的 Gnutella 数据长度。对一个在输入数据流中想找到下一个描述符开始位置的 Gnutella 节点来说，数据长度字段是唯一的方法。Gnutella 协议不提供一个明显的字符串或者任何描述同步的方式。因此，Gnutella 节点应当严格确保这个数据长度字段对每个收到的信息包都有效。如果一个 Gnutella 节点和它的输入数据流不同步，那么将抛弃与这个连接相关的数据。

（2）ping 数据包格式

ping(ID 值为 0x00)的长度为 0，如果一个 ping 被一个功能 ID 字段为 0x00 的描述符包头所回应，那么它的数据长度是 0x00000000。

一个Gnutella节点使用ping功能来动态检测网络中的其他Gnutella节点。一个收到 ping 的 Gnutella 节点会返回一个 pong 信息包来回应，这个回应数据包中含有一个正在网络中的 Gnutella 节点的地址及该节点提供的共享数据的相关信息。

Gnutella 协议尽量将 ping 引起的流量减少到最少，故现有的 Gnutella 协议仍不推荐 Gnutella 节点以较快的频率发送 ping 包。

（3） pong 数据包格式

pong 数据包仅被用来回应从外部进来的 ping 请求。pong(ID 值为 0x01)数据包格式如下：

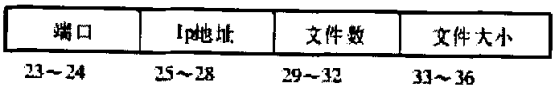

图 3.3 pong 数据包格式

*主机端口：回应主机的 TCP 端口。

*主机 IP 地址：回应主机的 IP 地址。

*共享文件数：该主机上共享文件的数量

*共享文件的大小：该主机上所有共享文件的大小(单位为 kB)。

（4）Push 请求数据包格式

当一个拥有资源的 Gnutella 节点在防火墙后面，需要该资源的 Gnutella 节点无法直接与该节点连接时，一个 Push 请求信息将被发送，来请求这个拥有资源的 Gnutella 节点从防火墙内部向外(向那个需要该资源的 Gnutella 节点)主动发起连接，并完成数据传送的任务。Push 请求数据包的格式 如下：

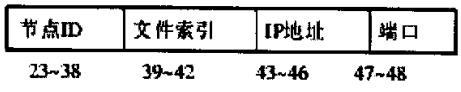

图 3.4 push 数据包格式

*节点 ID：16 个字节的字符串，唯一标识网络中的 Gnutella 节点(被请求 push 文件的 Gnutella 节点)的 ID 值。发出 Push 请求的 Gnutella 节点应当设置这个字段为在相应的“搜索结果”数据中返回的“节点标志”信息。

*文件索引：文件索引标识了发出 Push 请求的 Gnutella 节点定义的需要从目标 Gnutella 节点上被 Push 的文件。

* IP 地址：发出 Push 请求的 Gnutella 节点的 IP 地址。

*端口：发出 Push 请求的 Gnutella 节点的守候端口号。

（5） 搜索数据包（Query）格式

搜索(ID 值为 0x80)数据包的格式如下：

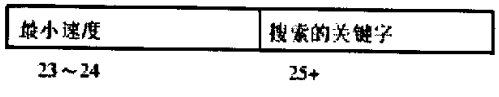

图 3.5 Query 数据包格式

*最小速度(kbps)：定义了 Gnutella 节点信息传输的最小速度。如果一个 Gnutella 节点能以大于 n kbps 的速度通信，那么这个收到含有

最小速度字段的搜索数据包并在本地有该搜寻信息的 Gnutella 节点会返回一个搜索结果响应。

*搜索的关键字：这个字符串的最大长度被绑定在这个 Gnutella 数据包头的数据长度字段中。

（6） 搜索结果数据包格式

搜索结果(ID 值为 0x81)数据包格式如下：

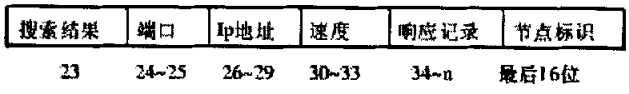

图 3.6 QueryHit 数据包格式

*搜索结果数目：在结果集合中的搜索结果的数目。

*端口：回应搜索请求的主机的守候端口号。

* IP 地址：回应搜索请求的主机的 IP 地址。

*速度：回应搜索请求的主机的速度(kbps)。

*响应记录：一个响应搜索请求的集合，结构如下：

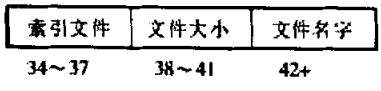

图 3.7 QueryHit 数据包中的响应记录集合格式

*文件索引：一个数字，由回应主机分配，它被用来唯一标识匹配查询的文件。

*文件大小：索引文件的大小(bytes).

*文件名字：匹配的文件名字，仅仅是文件的名字，不包含路径信息。

*节点标识：唯一标识响应搜索请求的 Gnutella 节点的一个 16 字节字符串。

3.2.3 Gnutella 协议消息的路由规则

为了实现信息查询、共享和交换及带宽的有效利用，Gnutella 节点遵循如

下这些路由协议：

a. Pong 描述符仅能沿着转发输入 Ping 描述符的同样路径被发送。该规则保证了只有那些路由了 Ping 描述符的对等机才能看见响应的 Pong 描述符。一旦一台对等机收到了一个 Pong 描述符，而其 Descriptor_ID=n，但却未看见带着 Descriptor_ID=n 的 Ping 描述符，则就会将该 Pong 描述符从网络中删除。

b. 同样道理,QueryHit 描述符仅能沿着转发输入 Query 描述符的同样路径被发送。该规则保证了只有那些路由了 Query 描述符的对等机才能看见响应的 QueryHit 描述符。一旦一台对等机收到了一个 QueryHit 描述符，而其 Descriptor_ID=n，但却未看见带着 Descriptor_ID=n 的 Query 描述符，则就会将该 QueryHit 描述符从网络中删除。

c. Push 描述符仅能沿着转发输入 QueryHit 描述符的同样路径被发送。该规则保证了只有那些路由了 QueryHit 描述符的对等机才能看见 Push 描述符。一旦一台对等机收到了一个 Push 描述符，而其 Servent Identifier=n，但却未看见带着 Servent Identifier=n 的 QueryHit 描述符，则就会将该 Push 描述符从网络中删除。

d. 一台对等机除了从输入 Ping 和 Pong 描述符方向直接来的对等机之外，将会向所有与之直接相连的对等机转发输入的 Ping 和 Query 描述符。

e. 而且在进行这种转发之前，将会修改相关的 TTL 和 Hops 字段，将 TTL 值减 1，Hops 值加 1。一旦该 TTL 值变成了 0 ，则将会把该描述符从网络中删除。

f. 如果一台对等机收到了与它原来曾经收到过的 Descriptor ID 和 PayLoad Descriptor 一样的描述符的话，应该避免对该描述符的转发。[19]

3.2.4 Gnutella 协议的文件下载

（1） 正常情况下

当 Gnutella 节点收到一个搜索结果信息，它将依据搜索结果的响应记录的描述文件直接进行下载初始化。在源 Gnutella 节点和目标 Gnutella 节点之间建立一个直接连接，实现文件下载。下载文件所使用的协议是 HTTP 协议[20]。初始

化下载的 Gnutella 节点发送一个请求字符串给目标 Gnutella 节点，格式如下：

```
GET /get/<File Index>/<File Name>/ HTTP/1.0\r\n
Connection: Keep-Alive\r\n
Range: bytes=0-\r\n
User-Agent: Gnutella\r\n3
\r\n
```

收到这个下载请求的 Gnutella 节点使用符合 HTTP1.0 的包头进行响应，响应信息为：

```
HTTP 200 OK\r\n
Server: Gnutella\r\n
Content-type: application/binary\r\n
Content-length: file size
\r\n
```

文件数据紧随其后，其大小为 HTTP 响应中 content-length 指定的字节数目。Gnutella 协议为 HTTP 排列参数提供支持，便于实现断点续传。

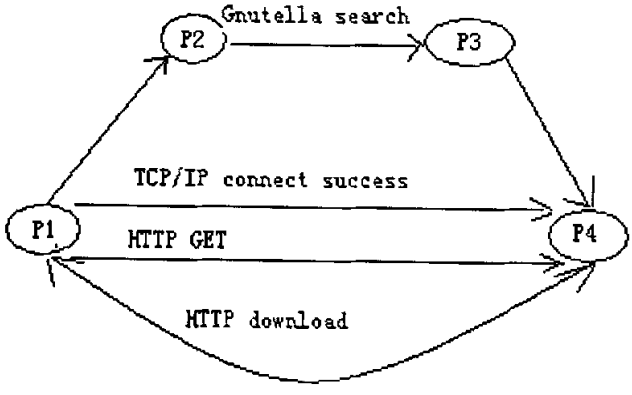

图 3.8 无防火墙屏蔽的情况

如图 3.1 所示，在双机都没有被防火墙屏蔽的情况下，当 P1 在 Gnutella 网络上通过 Query 消息查询到 P4 上有它想要的文件时，它就直接发送一个 TCP/IP 连接请求到 P4（直接在 TCP/IP 网络上发送，而非通过 Gnutella 网络），由于 P4

并未被屏蔽，该连接能成功建立。则 P1 再向 P4 发送一个 HTTP GET 文件下载请求，P1 和 P4 之间能够成功进行上传下载了。

（2） 单机屏蔽防火墙后面的 Gnutella 节点

如果在两个 Gnutella 节点之间不能直接建立连接时，试图下载文件的 Gnutella 节点将给发送搜索结果的 Gnutella 节点路由一个 Push 请求，来请求共享文件的 Gnutella 节点“推出(Push)”这个文件。作为 Push 请求目标的 Gnutella 节点(通过“Push 请求数据”中的“节点 ID”字段来确认)应该试图和那个发送 Push 请求的 Gnutella 节点(通过“Push 请求数据”中的“IP 地址”和“端口”来确认)建立一条新的 TCP/IP 连接。如果不能建立直接的连接，那么发出 Push 请求的 Gnutella 节点很可能也在防火墙后面。这种两个 Gnutella 节点都在防火墙后面的情况，文件传输将不能进行。[21]

如果从防火墙后的 Gnutella 节点到请求 Push 的 Gnutella 节点的直接连接被建立，防火墙后面的 Gnutella 节点将立刻发送下面的数据：

```
GIV <file index>:<servent identifier>/<file name>\n\n
```

这里<file index>和<servent identifier>指的是来自收到的 Push 请求中的文件索引(file index)和“节点 ID”字段的值，<file name>是本地文件表中那个文件的名字，它的文件索引号就是<file index>. 那个收到 GIV 请求包头的 Gnutella 节点(如 Push 请求者)从包头中提出<file index>和<file name>字段，建造一个如下格式的 HTTP GET 请求：

```
GET /get/<file index>/<file name>/ HTTP/1.0\r\n
Connection: Keep-Alive\r\n
Range: bytes=0-\r\n
User-Agent: Gnutella\r\n3
 \r\n
```

其下载过程的其余部分同前所述。

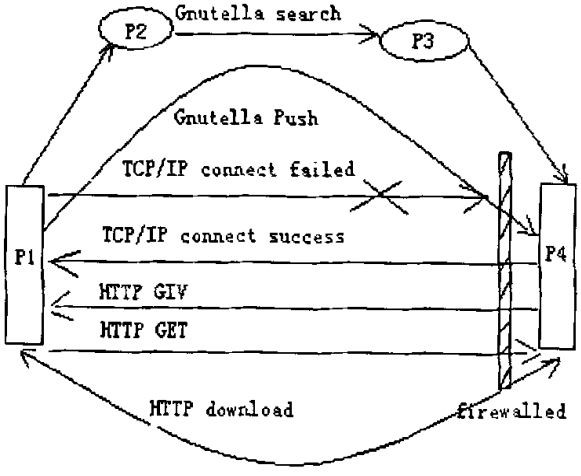

图 3.9 单机被防火墙屏蔽的情况

如图 3.2 所示，在有一台机器被防火墙屏蔽的情况下，当 P1 经过 P2，P3 搜寻到 P4 有它想要的文件时，P1 向 P4 直接发出一个 TCP/IP 连接请求，由于 P4 被防火墙屏蔽，该连接请求失败。P1 采取另一策略，在已建立的 Gnutella 链路上发出一个 Push 消息给 P4，P4 接收到 Push 消息后，由它主动向 P1 发起一个 TCP/IP 连接请求，由于 P1 并未被屏蔽，该连接能够成功建立，因而 P4 能在该 TCP/IP 链路上向 P1 发出一个 HTTP GIV 上传通告；P1 收到该上传通告后，再在本链路上发回一个 HTTP GET 下载请求，两机能够正常上传和下载了。

（3） 双机屏蔽防火墙后面的 Gnutella 节点

如果在两个 Gnutella 节点都处于不同的防火墙屏蔽之后，则两个节点间不可能直接建立一个 TCP/IP 连接，协议本身的 Push 解决方案也无能为力了，两台客户机间不能上传，下载文件，这样就大大限制了协议应用范围，是应该要解决的。

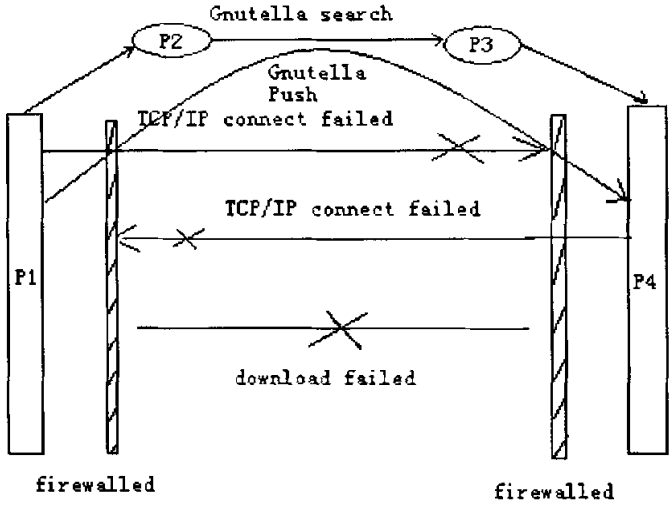

图 3.10 双机被防火墙屏蔽的情况

这是双机都被防火墙屏蔽的情况：当 P1 在 Gnutella 网络上通过 P2，P3 搜寻到 P4 有共享出 P1 所需的文件时，P1 向 P4 发出一个 TCP/IP 直接连接请求，自然，由于 P4 被屏蔽，该连接请求失败。于是 P1 在原 Gnutella 链路上向 P4 发出一个 Push 消息，请求 P4 向它“推”出所求文件。P4 收到 Push 包后，也尝试向 P1 发出 TCP/IP 直接连接请求。同样，由于 P1 也给屏蔽，该连接也失败，于是，双方的主动请求连接都归于失败，该下载尝试以失败告终了.

第4章 Gnutella 协议的一个双机防火墙穿越方案

根据上一章的分析，当只有单方机器被屏蔽，Gnutella 协议有自己完善的解决方案。但是，对于双方机器都被防火墙屏蔽的情况，协议本身并没有提出一个解决方案，这样，下载就只能告失败，用户只能眼睁睁的看着搜寻到的文件，却对之无能为力，不能下载.

并且，相对于 C/S 结构，在 P2P 网络上出现被防火墙屏蔽的情况更是普遍：由于 C/S 结构中，作为 Server 端的机器往往是一个强大的服务器，其主要功能在于提供给其他客户端进行文件的上传，下载以及其它等功能，因而，它一般不会被防火墙简单的屏蔽。而就 P2P 结构而言，处于 P2P 网络中的机器更多的是个人计算机，有大量的处于企业，教育网等局域网内的机器参与其中，它们往往会被所处网关的防火墙简单屏蔽。因而，解决了双机都被屏蔽的问题，必然能使本协议更为广泛，无阻的得到应用.

为此，本文提出了一个方案，动态借助 Gnutella 网络上的一台客户机作为代理，给出详细的设计思想，扩展的消息包，路由规则，文件的传输过程等。并且，在附录 1 中，本文给出了一份规范的协议扩展的 proposal。

4.1 方案的总体设计思想和可行性

4.1.1 代理机的选择策略

由于在 Gnutella 协议中，文件的下载是脱离了 Gnutella 网络的（所谓的 out-of-network）,它是借助了 HTTP 协议中的上传下载功能来进行文件下载的。因而，我们可以有三种代理机选择策略：

Ⅰ.脱离 Gnutella 网络，在基本的 TCP/IP 网络上以一台确定的机器作为代理机，公布其 IP 或域名，其他被双向屏蔽的机器都借助于它来进行文件传输。

Ⅱ.在 Gnutella 网络上，以一台确定的机器作为代理机，公布其 IP 或域名，其他被双向屏蔽的机器都借助于它来进行文件传输。

III. 在被屏蔽的双机的本 Gnutella 链路上，动态协商确定一个客户机作为代理机来代理文件的传输。

对比以上三种策略，第一，二种策略都要求有一台确定的机器来作为代理机。这样，其地位就相当于一台代理服务器了，它必须保持在线，有强大的处理能力，足够的带宽来代理文件的上传下载，这样就违背了 Gnutella 协议的纯 P2P 的特性了。

那么，在第三种策略中，由于被屏蔽的双机已经能够成功的建立 Gnutella 连接，进行文件的搜索与响应，那么在此链路上就必然存在至少一台机器，未被双方防火墙所屏蔽。

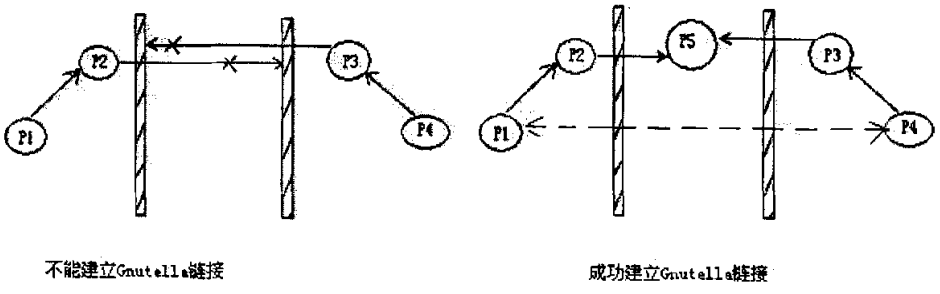

图 4.1 Gnutella 链路建立的对比图

由以上的对比图可见，两台分别被不同的防火墙屏蔽的机器（P1，P4）倘若能成功建立链路，必定有不被防火墙所屏蔽的某一台或几台机器作为中间桥梁。只要我们能通过新增一些消息来协商定某一台机器作为代理机，就能够代理文件的传输了。

4.1.2 代理机选择的协商过程

明确了代理机的选择策略及其可行性后，可以进行代理机选择的协商了。为此，为协议加入两个协商消息：DownloadProxyRequest，DownloadProxyAck；请求下载的机器向共享文件的机器发送请求 DownloadProxyRequest，沿途节点如果乐意作为代理机，就将自己的 IP 和代理 port 填入 DownloadProxyRequest 中。考

虑到在 P2P 网络中，通信协议的控制消息还是占很大比例的[22]，因而，对扩展的消息所携带的信息也是要作一定的限制，以免协议占用网络过多的带宽。

但是，对于有多台机器乐意作为代理机的情况，有两个问题要考虑：

Ⅰ.如何判断该节点为合格的代理节点，该节点并未被某一方的防火墙所屏蔽。

Ⅱ.为提高效率，减少网络带宽的占用，应该尽量减少协商过程中的消息交换。

考虑下图：

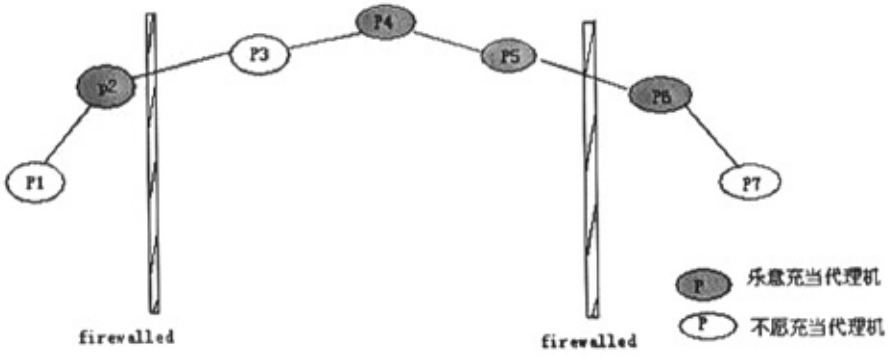

图 4.2 Gnutella 网络示例图

P1 发出 DownloadProxyRequest 请求代理，向 P7 下载文件。延途经过的节点有：P2，P3，P4，P5，P6。其中，P2，P3，P5，P6 乐意充当代理机，它们将自己的信息（IP，Port）填入 DownloadProxyRequest。

从图可见，在乐意充当代理机的节点中，只有 P4 和 P5 能够成为合格的代理机，P2 和 P6 都因被某一方的防火墙所屏蔽而不能成为有效的代理。

为判断中间节点是否能够成为合格的代理机，并且减少判断次序，我们只需从正向抽取 IP 和 Port，依次作 TCP/IP 连接尝试,只有该尝试失败才尝试下一个；第一个成功连接的节点确定未合格代理节点，并且将其信息填入 DownloadProxyAck 中响应给 P1。P1 再成功连接 P4 后就可以进行文件的代理传输了。

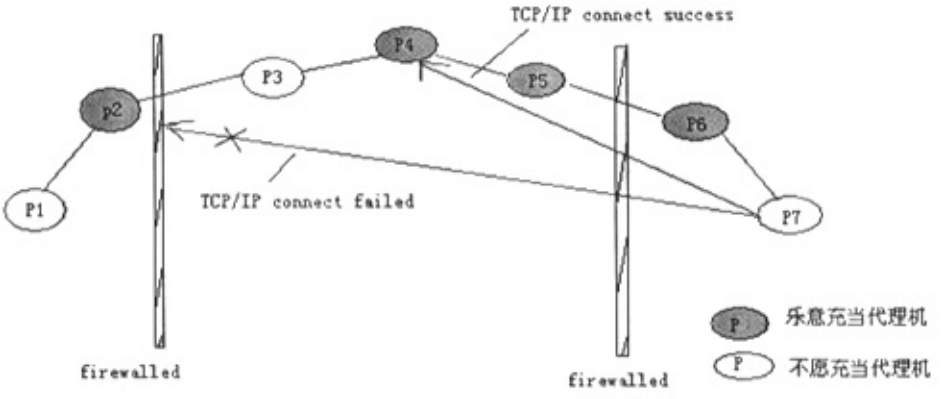

图 4.3 反向代理测试图

但是，考虑下图情况，如果在中间节点中只有 P2，P6 乐意充当代理机，而未被双方防火墙所屏蔽的 P3，P4，P5 都不愿充当代理机，则会出现以下情况：P7 依次与 P2，P6 进行 TCP/IP 连接尝试，首个成功连接的节点为 P6，将其确立为合格代理机，并将其信息填入 DownloadProxyAck 响应给 P1。但是，由于 P6 也给屏蔽，P1 与 P6 的 TCP/IP 连接失败，因而，P6 也是不合格的代理机。文件的代理传送最终是失败的。

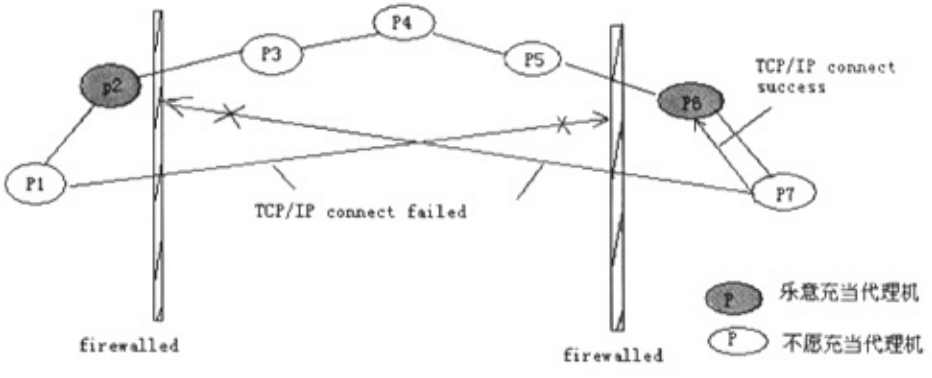

图 4.4 代理测试失败

完整的判断流程如下图所示：

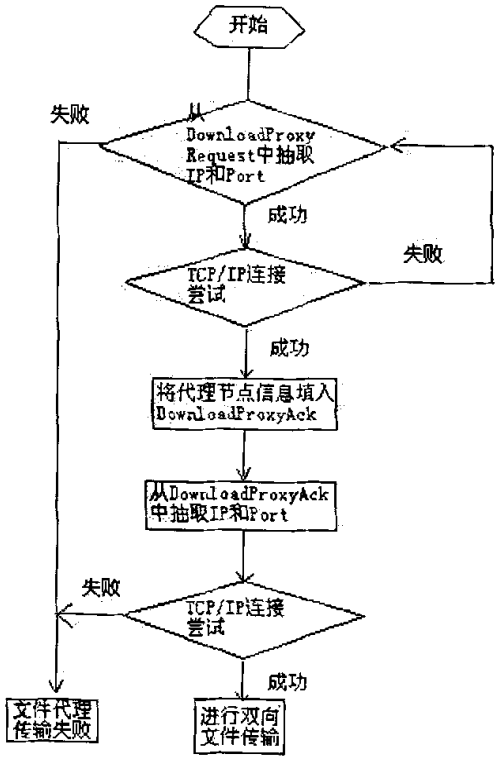

图 4.5 代理测试流程图

并且，为方便代理机的文件转发，新增了两个 HTTP 消息：FWGIV(ForWardGIV), FWGET(ForWardGET)。

4.1.3 总体结构

以下是协议扩展后，客户机交互的总的过程图：

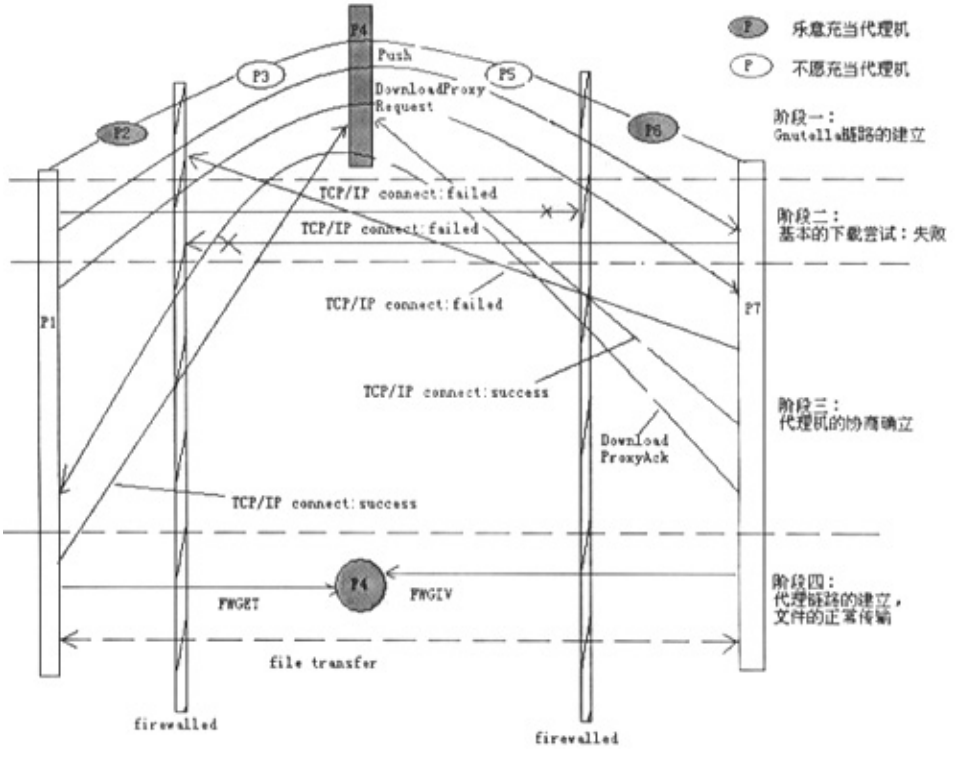

图 4.6 扩展后的协议流程

分为四个阶段：

阶段一：Gnutella 链路的建立。P1 和 P7 通过 P2，P3，P4，P5, P6 建立了一台 Gnutella 链路，并且 P1 通过 Query 消息以及 P7 响应的 QueryHit 消息得知 P7 有满足其搜寻请求的文件共享。

阶段二：基本的下载尝试，该尝试最后以失败告终。P1 得知 P7 有合适的文件共享，于是向 P7 进行直接的 TCP/IP 连接请求，由于 P7 被屏蔽，该请求失败。P1 再尝试协议本身对单方防火墙屏蔽的解决策略：通过原 Gnutella 链路（通过 P2,P3,P4,P5,P6）,向 P7 发出一个‘Push’消息，请求 P7 将文件推向 P1。P7 收到 P1 的 Push 消息后，尝试向 P1 进行直接的 TCP/IP 连接请求，以图能向 P1“推”出文件。但是，由于 P1 也被屏蔽，该尝试自然也失败。

阶段三：代理机的协商确立阶段。由于双向的直接 TCP/IP 连接请求都失败，只能借助于代理机进行文件传输了。代理机的协商，借助于自定义的两个消息：DownloadProxyRequest，DownloadProxyAck。P1 延原 Gnutella 链路发送代理请求

消息 DownloadProxyRequest，延途的机器如果愿意作为代理机，就向该请求消息内填入自己的 IP 和代理端口 Port(如上图的 P2，P4，P6)；如果不愿意当代理机，就只延原路转发消息（如上图的 P3，P5）。当 P7 收到 DownloadProxyRequest 后，就将各沿途节点的 IP 和 Port 信息抽取出来。再依次向这些节点尝试 TCP/IP 连接，只要有一个连接能够成功，后面的机器都不必再作尝试，可确立此机为代理机，将其 IP 和 Port 填入 DownloadProxyAck 中，延原途响应给 P1。

阶段四：代理链路的建立阶段。P7 发送完 DownloadProxyAck 消息后，就可以开始发送一个新增的 HTTP 消息 FWGIV 给代理机 P4，表示要上传转发的文件；而 P1 收到 DownloadProxyAck 后，抽取出 P4 的 IP 和 Port，与之成功进行 TCP/IP 连接，然后再发送一个新增的 HTTP 消息 FWGET 给代理机 P4，表示要下载转发的文件。

于是，该文件就通过代理机 P4 在 P1 和 P7 之间成功传输了。

4.2 对 Gnutella 协议扩充的具体实现

4.2.1 协议握手阶段（Handshaking Sequence）

为保证协议扩展的可用，必须在建立 Gnutella 连接时的握手阶段交换信息，互相通知是否支持该扩展。如果对方节点不支持该扩展，则不应该向其发送自定义的消息：DownloadProxyRequest，DownloadProxyAck。在一条 Gnutella 链路中，如有任意一个中间节点不支持该扩展，本代理扩展都告失败，因为链路中有某节点不能转发 DownloadProxyRequest 消息。

为此，我们定义，支持该协议扩展的节点在握手阶段都必须交换自定义的消息头：

"DownloadProxy: 1.0<cr><lf>"

以下是一个双方机器都支持本协议扩展的例子，双方机器在握手阶段所进行的交互：

Servent A	Servent B
GNUTELLA CONNECT/0.6<cr><lf>	
User-Agent: Dtella<cr><lf>	
<cr><lf>	
	GNUTELLA/0.6 200 OK<cr><lf>
	User-Agent: Dtella<cr><lf>
	DownloadProxy: 1.0<cr><lf>
	<cr><lf>
GNUTELLA/0.6 200 OK<cr><lf>	
DownloadProxy: 1.0<cr><lf>	
<cr><lf>	
[binary messages]	[binary messages]

如果对方机器支持本协议，则作以下设定：

```
s->setProxySupported(true)
```

如果不支持，则：

```
s->setProxySupported(false);
```

以后，在进行协议交互时，通过以下语句进行判断相邻的节点是否支持本协议扩展：

```
_servants[i]->getProxySupported()
```

只有返回真值，表示支持协议扩展，本机才会向相邻节点发送扩展的消息。

4.2.2 对 Gnutella 协议进行扩充的消息的结构

为保持所以对协议的扩展的一致性，我们将客户自定义的新消息统一封装于一个普通的 Gnutella 消息内，其消息 ID 值为 0x31。[23]则 Gnutella 的消息 ID

值如下列表：

消息名	Ping	Pong	Push	Query	QueryHit	Vendor-Specific
ID 值	0x00	0x01	0x40	0x80	0x81	0x31

客户自定义的消息结构如下（Vendor－Specific message)：

Vendor ID	Sub-selector	Version number
0-3	4	5

*Vendor ID:客户的 ID，是一个 4 自己的 ASCII 码，注意区分大小写。该码值与 QueryHit 中的码值一致。不同客户端软件对协议的扩充，以此码来区分，避免不同客户端软件间扩充的冲突。

*Sub－selector：用以表示该 Vendor ID 的扩充消息的类型。对于每个客户端软件，以此来区分本身对协议的扩充的消息的类型。在本扩充建议中，定义了两个消息类型：

0x01 = DownloadProxyRequest

0x02 = DownloadProxyAck

*Version number：表示扩充建议的版本。

紧跟这个消息头，就是本文所扩充的两个消息。

（1） DownloadProxyRequest Message (0x01)

该消息格式如下：

File Index	File Size	Node Set	Servent Identifier
0－3	4－7	8－	Last 16

*File Index：文件索引。该值与 QueryHit 消息中的值一样。用以识别要代理传输那个文件。

*File Size：文件大小。中间节点可以借助此值来决定是否要代理传输此文件，一般不代理太大的文件。

*Node Set：节点信息集合。只要某中间节点乐意充当代理机，就向此集合填入它自己的 IP 和 Port 值。该集合的每一个组成有以下结构：

0－1 Port：代理节点填入自己的代理端口

2－5 IP Address: 代理节点填入自己的 IP

*Servent Identifier: 该值设为所获取的 QuerHit 消息中的值所一致，用来将 DownloadProxyRequest 消息路由回原发送 QueryHit 消息的机器。

完整的 DownloadProxyRequest 消息结构如下：

Bytes:	Description:	
0-15	Message ID/GUID (Globally Unique ID)	
16	Payload Type (0x31 Vendor-Specific Message)	
17	TTL (Time To Live)	
18	Hops	
19-22	Payload Length	
23-26	Vendor ID ("DTEL" for my proposal)	
27	Sub-Selector(0x01 DownloadProxyRequest)	
28	Version Number(1.0 fro this proposal)	
29-32	File Index	
33-36	File Size	
37-	Node Set	
	0-1	Port
	2-5	IP Address
Last16	Servent Identifier	

以下是在具体实现中，DownloadProxyRequest消息的数据结构：

```
//message header
  std::string  _id;
  Q_UINT8      _ttl;
  Q_UINT8      _hops;
  std::string  _data;

//vendor message header
```

```
std::string  _vendor_id;          //0-3
Q_UINT8      _version_number;     //5

//DownloadProxyRequest message header
Q_UINT32                 _index;      //6-9
Q_UINT32                 _filesize;   //10-11 in byte
std::vector<Address>     _node_address;
std::string              _servent_id;
                                      //last 16: servent id
```

（2） DownloadProxyAck Message (0x02)

该消息结构如下：

Port	IP Address
0－1	2－5

*Port：由上传机确定的代理机的代理端口

*IP Address：由上传机确定的代理机的 IP 地址

本消息（DownloadProxyAck）的 Message ID 必须设定为与其所响应的 DownloadProxyRequest 消息的 Message ID。这样，可以以 Message ID 将本消息路由回发送 DownloadProxyRequest 消息的机器

完整的 DownloadProxyAck 消息结构如下：

Bytes:	Description:
0-15	Message ID/GUID (Globally Unique ID)
16	Payload Type (0x31 Vendor-Specific Message)
17	TTL (Time To Live)
18	Hops
19-22	Payload Length
23-26	Vendor ID ("DTEL" for my proposal)
27	Sub-Selector(0x02 DownloadProxyRequest)

28	Version Number(1.0 fro this proposal)
29-30	Port
31-34	IP Address

以下是在具体实现中，DownloadProxyAck消息的数据结构：

```
//message header
  std::string  _id;
  Q_UINT8      _ttl;
  Q_UINT8      _hops;
  std::string  _data;

//vendor message header
  std::string  _vendor_id;
  Q_UINT8      _version_number;

//DownloadProxyAck message header
  Address      _address;
```

4.2.3 对 Gnutella 协议进行扩充的路由规则

为使增加的两个消息能够有效传输，在原协议的路由规则的基础上给出以下路由规则：

a. DownloadProxyRequest：沿着从外部发来的搜索结果（QueryHit）相同的路径被发送。这样可以确保只有发送和转发搜索结果响应的 Gnutella 节点能看到该消息。

为了保证此路由规则，在实现中，必须保存每个进来的 QueryHit 消息的 Servent Identifier：

```
//Add queryhit servent id.
  s->_queryhit_id_set.insert(q->_id_result);
```

在进行本消息转发前，通过以下语句判断转发的路径

(_servants[i]->_queryhit_id_set.find(dpr._servent_id) != _servants[i]->_queryhit_id_set.end())

b. DownloadProxyAck: 沿着从外部发来的代理请求（DownloadProxyRequest）相同的路径被发送。这样可以确保只有发送过代理请求的 Gnutella 节点能看到代理测试结果。

为了保证此路由规则，在实现中，必须保存每个进来的 DownloadProxyRequest 消息的 Message ID:

s->_dprfw_id_set.insert(dpr->_id);

在进行本消息转发前，通过以下语句判断转发的路径

(_servants[i]->_dprfw_id_set.find(dpa._id) != _servants[i]->_dprfw_id_set.end()))

c. 在每个 Gnutella 节点将描述符转发到与它直接相连的 Gnutella 节点前，会将数据包头中的 TTL 字段－1，同时将跳数字段＋1。如果包头中的 TTL－1 后等于 0，那么该信息包将不再被转发到任何其他 Gnutella 节点上。

d. 接收到与曾经收到过的 Gnutella 数据包头及 ID 相同的信息包的 Gnutella 节点将避免把这些信息包再转发给其他 Gnutella 节点。这样可防止重复发送，浪费带宽。

4.2.4 文件的传输过程

为区别一般的上传、下载，在基本的 HTTP 上传、下载消息的基础上，增加两个 HTTP 消息 FWGIV, FWGET, 用来代理文件的上传和下载。

当上传端确定代理机后，首先在 Gnutella 链路上响应一个 DownloadProxyAck 消息给下载端，然后发送一个 HTTP 消息 FWGIV 给代理机，表

示要上传文件：

FWGIV <File Index>: <Servant Identifier>/<File Name><lf><lf>

其中，<File Index>和<Servent Identifier>都是从 DownloadProxyRequest 中抽取出来的信息。<File Name>是本地共享文件表中索引为 File Index 的文件的文件名。代理机收到该 HTTP 消息后，将响应一个 HTTP GET 消息，将文件下载到它所开辟的一个缓冲区，等待下载方的请求：

GET /get/<File Index>/<File Name> HTTP/1.1<cr><lf>

下载方在收到DownloadProxyAck后，也向代理机发出一个TCP/IP连接请求，当成功连接后，发送一个 HTTP FWGET 消息给代理机，请求转发文件：

FWGET /get/<File Index>/<File Name> HTTP/1.1<cr><lf>

代理机收到此消息后，比较 FWGET 和 FWGIV 的<File Index>/<File Name>对，如果匹配，则将缓冲区暂存的文件传到下载机中。该操作可以同时进行，不必等到上传方将文件都传送完毕载进行。

过程如下图所示，FC 是被防火墙所屏蔽的下载端，FS 是被防火墙所屏蔽的上传端，DP 是所确定的代理机：

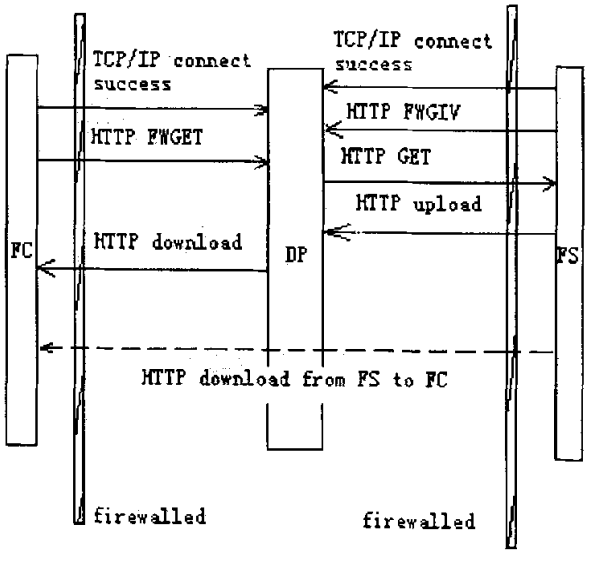

图 4.7 文件代理传输过程

第5章 系统运行情况和改进方法

本文是在 Linux[24,25]平台上实现对协议的扩展的。借助 KDE 的 Qt 库，以 c++ 语言进行开发。在实现上，本文借助了 Linux 上的一个开源软件 Qtella[26]，该软件实现了基本的 Gnutella 协议功能。在其基础上，本文设计，实现了 DownloadProxyRequest，DownloadProxyAck 类，实现了其路由算法，实现了文件的转发。最后，对该扩展作了测试。

5.1 模拟环境搭建

为测试协议扩展的可行性，在实验室的环境下，搭建了如下的一个小规模的 Gnutella 网络：

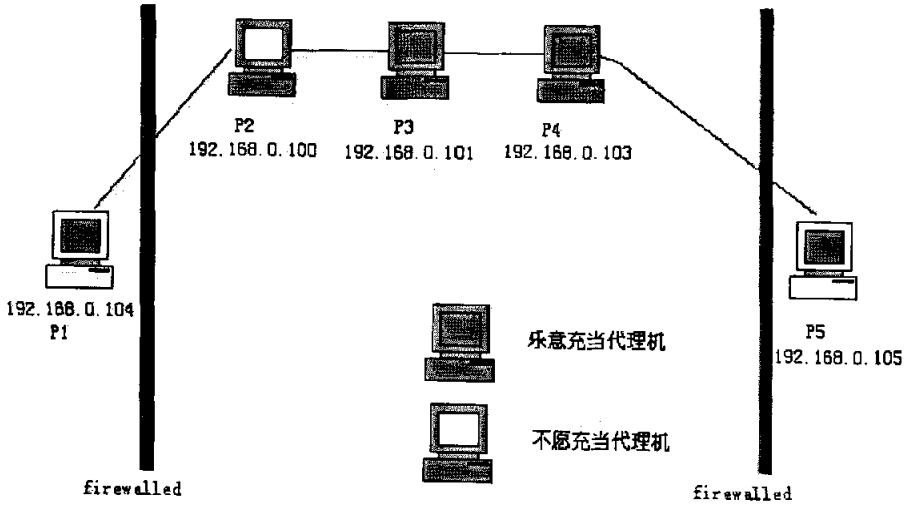

图 5.1 模拟环境网络图

防火墙的配置，使用 Linux 上的 iptables，采取以下的策略[27]：

[root@seasw dyingwood]# iptables -A INPUT -p tcp --syn -j REJECT

或者

[root@seasw dyingwood]# iptables -A INPUT -p tcp --tcp-flags SYN, RST,

ACK SYN.

该命令匹配SYN位置位，ACK，RST位清零的TCP包。这些包用来发出一个TCP链接请求。这样的一个防火墙规则将拒绝每一个TCP主动链接，回应一个icmp-port-unreachable；但是允许TCP链接响应，即允许由己方发起的TCP链接请求。这样，可以屏蔽所有进入的链接，但是允许所有发出的链接。

5.2 系统的运行情况

5.2.1 握手阶段

在握手阶段，相邻节点的两台机器必须交换信息，表明是否支持本协议扩展。以下是用 ethereal 截取下来的 TCP stream，该 TCP 流是两台支持本协议扩展的机器的握手：

表 5.1 握手信息

GNUTELLA CONNECT/0.6
User-Agent: Qtella 0.6.5
Remote-IP: 192.168.0.100
GNUTELLA/0.6 200 OK
User-Agent: Qtella 0.6.5
Remote-IP: 192.168.0.108
DownloadProxy: 1.0
GNUTELLA/0.6 200 OK
DownloadProxy: 1.0

5.2.2 扩展消息的交互阶段

在本实验网络中，P5 为屏蔽的下载方，P1 为被屏蔽的上传方，P2 为不愿充当代理的中间节点，P3，P4 为乐意充当代理的中间节点，最后将会确立 P3 为代理节点。

当 P5 尝试了协议本身的 Push 请求对方“推”出文件失败后，将会首先发送一个 DownloadProxyRequest 消息，尝试确定代理机；在收到响应的 DownloadProxyAck，判断得知该消息是发往本机的（通过比较 Message ID），从中抽取出代理机的 IP 和 Port 后，可以开始进行 HTTP 下载文件了：

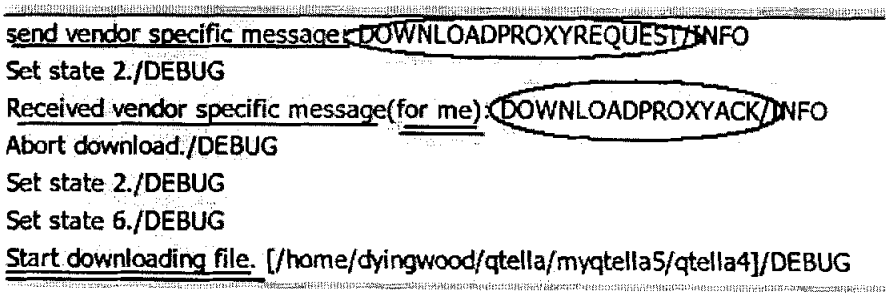

图 5.2 下载方 P5 的消息过程

P1 作为上传端，在收到 P5 发送过来的 DownloadProxyRequest 消息后，判断得知该消息的目的地是本机（通过比较 Servent ID）后，从消息中抽取出乐意充当代理机的节点的 IP 和 Port，在尝试确定代理节点后，响应一个 DownloadProxyAck 给 P5，开始上传文件给代理机：

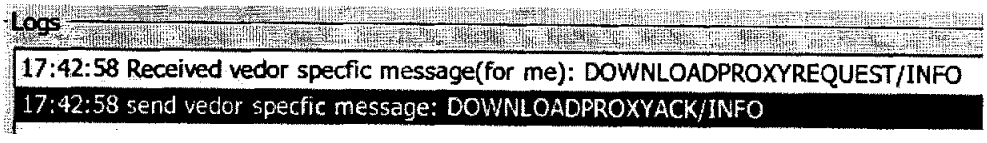

图 5.3 P1 的消息过程

作为中间节点，负责对 DownloadProxyRequest 和 DownloadProxyAck 进行转发。中间节点是分别通过保留的 Servent ID 和 Message ID 和收到消息的 Servent ID，Message ID 作比较得知该消息并非发往本机的。P2 不乐意充当代理机，它只负责转发收到的 DownloadProxyRequest 和 DownloadProxyAck 消息：

```
17:42:58 Received vedor specfic message(not for me): DOWNLOADPROXYREQUEST/INFO
17:42:58 I don't want to be a download proxy, just forward DOWNLOADPROXYREQUEST/INFO
17:43:00 Received vedor specfic message(not for me): DOWNLOADPROXYACK/INFO
17:43:00 forward vedor specfic message: DOWNLOADPROXYACK/INFO
```

图 5.4 P2 的消息过程

而中间节点 P3，P4 都是乐意充当代理机的，它们会将自己的 IP 和端口信息填入 DownloadProxyRequest 消息，再进行转发：

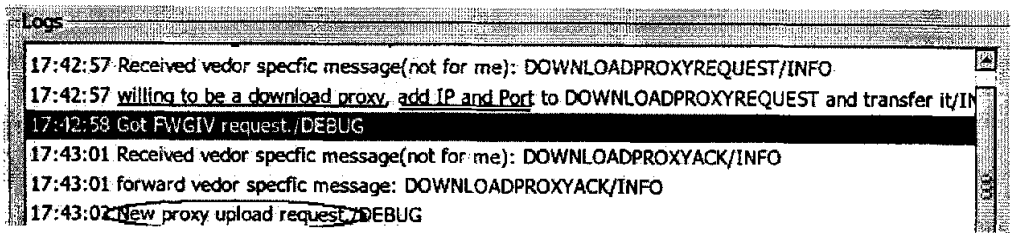

图 5.5 P3 的消息过程

上图是节点 P3，该节点被确认为代理节点，收到上传方的 FWGIV 消息接受其上传的文件，并且开启一个上传请求向下载方，向其转发收到的文件。

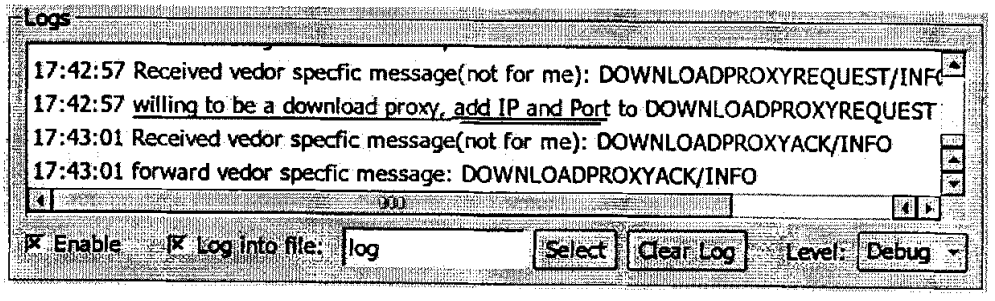

图 5.6 P4 的消息过程

上图是 P4 节点，虽然它乐意充当代理机，填入了自己的 IP 和 Port 信息，但却没有最后给确认为代理节点。其整个过程都是对消息进行转发而已。

5.2.3 文件的代理传输阶段

在确认了 P3 作为代理节点后，就可以进行文件的代理传输了。

这是上传端 P1 和代理节点 P4 的 HTTP 交互和文件上传的 TCP 流：

图 5.7 文件上传消息流

从中可见在收到本文扩展的 HTTP FWGIV 消息后，代理机另发出一个 HTTP GET 消息，请求文件下载。上传方在作了 HTTP OK 响应后，开始上传文件流了。

从上传方 P1 的控制台也可以看到同样的信息显示：

图 5.8 控制台显示的文件的上传

同样，下载端 P5 首先传送了一个本文扩展的 HTTP FWGET 消息请求下载后，代理节点在做了必要的判断后，确认其为合法的下载方，就响应一个 HTTP OK 消

息，然后开始代理上传文件了。其流程如下图所示：

图 5.9 文件下载消息流

最后，我们可以在被屏蔽的下载方看到，文件全部下载完毕，防火墙障碍不复存在了：

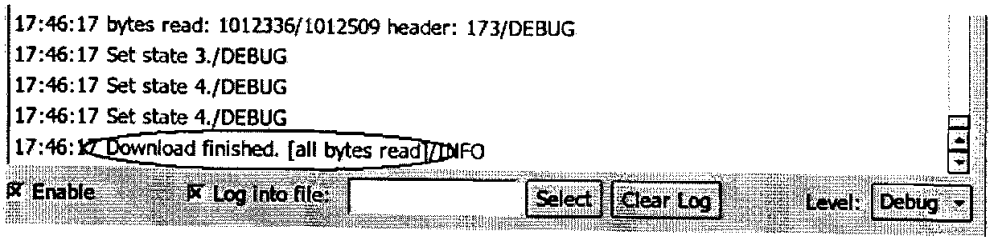

图 5.10 文件传输完毕

5.3 系统的不足和与下一步改进的展望

本方案能有效解决双机被防火墙的屏蔽的问题，但是它也存在一些缺点和不足，需要在下一步研究中继续改进：

i．相对 C/S 结构而言，P2P 结构有更大的不稳定性。由于不是确定专门的机器作为代理机，我们动态确定的代理机随时都有可能关闭机器和关闭客户端软件，脱离本网络，从而使文件的代理传输中断。相似的情况，在中间节点传送 Push 消息时也会出现[28]。在进一步完善协议扩展时，应该对代理的中断作一定处理，

可以就此中断文件的传输，也可以另确认新的代理节点，进行断点续传。

ii. 引入了代理节点，等于引入了更多的不安全因素。文件通过代理节点的传输，文件的信息可能会被恶意节点所窃取；恶意的代理节点也可能借代理传输文件的机会，在文件内嵌入其他的一些恶意程序，向下载方发送。因此，完善的方案应该在考虑效率的情况下，对文件的作必要的加密，完整的校验。

iii. 在有多台机器乐意充当代理节点的情况下，从效率上出发，可以对不同的节点作判断，选择更为稳定，长期在线的节点，也可以选择带宽更大的节点作为代理节点。这就要求在作代理节点的协商时，DownloadProxyRequest 消息必须携带更充分的信息，而不仅仅仅是其 IP 和 Port 信息。

以上的种种问题，很大程度上是由于目前的协议扩展过于简单，在进一步完善中，这些问题都是可以解决的。

结束语

Gnutella 是一种能够智能发现节点、完全分布式的 P2P 网络通信协议。其完全分布式的优点使 Gnutella 网络受到人们的极大关注。但是，协议本身对于双方节点都被防火墙所屏蔽的情况没有一个解决方案，造成检索到的许多数据都无法传输，影响了协议的通用性。

针对这个情况，本文提出了一个解决方案，通过在本 Gnutella 链路上动态确定一台代理机，代理数据的传输，达到穿越屏蔽了双方节点的防火墙的目的。为此，本文为 Gnutella 协议增加了动态确定代理机的协商消息，设计了这些消息的包结构，路由规则，也为代理机的代理数据传输而增加了新的 HTTP 消息。并且，本文在 Linux 平台上对本协议扩展作了实现，验证，证实了本方案是可行的。

由于时间比较短，对系统的实现考虑得还不是很全面，因此具体的实现还比较粗糙，由于本人在知识面、经验及能力等方面所存在的局限性，错误和不足之处在所难免，恳请各位专家批评指正。

参考文献

1. 黄道颖，李祖鹏，庄 雷，黄建华，张安琳， 分布式 Peer-to-Peer 网络 Gnutella 模型研究， 计算机工程与应用，2003 年，第 5 期

2. 熊 江，胡仲华，P2P 技术及其应用，重庆三峡学院学报，第 19 卷，第 3 期，2003 年

3. Napster: http://www.Napster.com/

4. Demetrios Zeinalipour-Yazti，Exploiting the Security Weaknesses of the Gnutella Protocol，Department of Computer Science ,University of California Riverside

5. http://www.jxta.org

6. 李 河,王树明,《P2P网络中使用UDP穿越NAT的方法研究》吉林大学学报(信息科学版) 第 21 卷，第 3 期，2003 年 8 月

7. 沈 波，张宏科， 一种新的P2P网络代理协议， 北方交通大学学报，第 28 卷，第 3 期，2004 年 6 月

8. 姜国华，顾君忠， 一种 P2P 系统索引结构生成算法， 计算机工程与应用，2004.2

9. 陈 姝，方滨兴，周勇林，P2P 技术的研究与应用， 计算机工程与应用，2002.13

10. 吕 向 辰 ， P2P 技 术 与 应 用 http://www2.ccw.com.cn/01/0128/d/0128d06_1.asp

11. 王 丹，魏 红， P2P 模式的系统结构研究， 沈阳航空工业学院学报，第 20 期，第 2 卷，2003 年 6 月

12. Vincent Berk and George Cybenko, File Sharing Protocols: A Tutorial on Gnutella, 2001-03-06

13. Gnutella Protocol. http://www.capnbry.net/gnutella/protocol.php

14. Gnutella Forums. http://www.gnutellaforums.com/

15. 汤 晟，吴朝晖， P2P－对等网络的未来， 计算机应用，第 1 期，2004 年

16. 杨再晗，陈建二，王建新， P2P 计算研究现状及关键技术， 现代电子技术，2004 年第 1 期总第 168 期.

17. 刘宝旭，孙笑庆， 崔 石，陆建强， Gnutella 协议及数据包结构分析，计算机工程，2004 年 1 月第 30 卷，第 2 期

18. Networking Working Group, UUIDs and GUIDs, IETF Internet-Draft, 2002

19. 黄道颖，陈 新，张安琳，张 尧，黄建华， P2P 网络 Gnutella 模型中搜索消息的路由机制及改进研究， 计算机工程与应用，2003 年，第 25 期

20. Network Working Group, Hypertext Transfer Protocol, RFC 2616. 1999, 6

21. Gnutella&Firewalls, http://koeln.ccc.de/archiv/hackschiffseiten/information/firewalls.html. 2000, 5

22. MateiR, LanF. Mapping the Gnutella Network [J]. IEEE Internet Computing, 2002, 2:50-57.

23. Raphael Manfredi, Framework for Vendor-specific Messages, 2003, 5

24. 李善平，刘文峰，李程远，王焕龙，王伟波编著，Linux 内核 2.4 版源代码分析大全，机械工业出版社，2002，1

25. Neil Matthew 等著，叶小虎，龙浩等译，Linux 高级编程，机械工业出版社，2002，1

26. Qtella, www.qtella.com

27. Douglas E. Comer 著，林瑶，蒋慧，杜蔚轩等译，用 TCP/IP 进行网际互联（第一卷），电子工业出版社，2004，2

28. Jason Thomas， Andrew Mickish，Susheel M. Daswani ，Push Proxy，http://sourceforge.net/projects/gtk-gnutella/

附 录 1:Download Proxy Proposal

Download Proxy

Document Version: 1.0

Protocol Version: 0.6

Feb 26, 2005

ZhiWei Zhu (Dyingwood@tom.com)

1 Introduction

1.1 Background

Gnutella servents exchange Query and QueryHit messages over Gnutella network for finding files. But files are download out-of-network, file data is never transferred over the Gnutella network. The transfer files to one another using HTTP. Once a servent receives a QueryHit message, it may initiate the direct download of one of the files described by the message's Result Set. A direct connection between the source and target servent is established in order to perform the data transfer.

This scheme works so long as the servent hosting the file (the server) is accessible by the servent downloading (the downloader) the file. But it is not always possible to establish a direct connection to a Gnutella servent in an attempt to initiate a file download. The servent may be behind a firewall that does not permit incoming connections to its Gnutella port. And increasingly it is the case that servers are behind a firewall. If there is only the server is firewalled, the original Gnutella protocol has a workaround called Push. With Push, the downloader constructs and sends a Gnutella Push packet on the GNET containing the down loader's address and port, as well as the server's ID. The server opens a connection back to the downloader, sends a special header, and then waits for the downloader to begin a HTTP conversation on this open connection.

If the servent that issued the Push request is itself behind a firewall, then the direct connection can also not be established; file transfer cannot take place by the Push request described in Gnutella protocol. It means that Gnutella protocol cannot transfer files between two firewalled servents.

This Download Proxy proposal outlines a technique to do away with these problems, therefore a firewalled downloader attempting to get content from a firewalled server.

1.2 Requirements

The key words MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL in this document are to be interpreted as described in RFC 2119.

1.3 Terminology

To understand this document, we use a standard set of terminology throughout. A Gnutella servent that is sharing files and that cannot be directly reached by the bulk of nodes on the GNET (due to firewall) is called the Firewalled Server or FS. A Gnutella servent that wishes to download from the FS and that cannot be directly reached by the bulk of nodes on the GNET (due to firewall) is called the Firewalled Downloader or FD. A Gnutella servent that is used to proxy download data is called the Download Proxy Server or DP. Any Gnutella servent that is along the QueryHit message route is called Route Node or RN.

2 Basic Operations

The basic operational steps of the Download Proxy proposal are:

1. The FC sends a Query message and receives a QueryHit Message from FS.
2. If the direct download and push download are both fail, FC sends a vendor message DownloadProxyRequest message along the QueryHit route

to FS.

3. If any RN is willing to be a download proxy, it adds its address and proxy port to the DownloadProxyRequest message.
4. FS try to make backwards a direct TCP connection to all the RNs listing in the DownloadProxyReques message's Node Set. The last successfully connected RN will be choosing as a DP.
5. FS sends a vendor specific DownloadProxyAck message to FC. This message contains DP's address and proxy port. Each RN checks whether this message's IP Address field is the same as his. If not, it means he will not be the proxy; he can cancel his proxy socket.
6. FS make another HTTP GIV to DP to upload the file.
7. When FC receiving the DownloadProxyAck message, it sends a GET request to DP to get the file.
8. DP forwards the file data receiving from DS to DC.

3 Vendor Specific Messages

According to the proposal "Framework for Vendor-specific Messages", vendor specific messages are sent encapsulated within a regular Gnutella message bearing the message code 0x31.

The Vendor-specific Message has the following fields:

Bytes:	Description:
0-3	Vendor ID. Case sensitive sequence of 4 ASCII characters, as with query hits.
4	Sub-selector. Indicates the message type for this vendor ID. In this proposal, there is two types of vendor specific message: 0x01 = DownloadProxyRequest 0x02 = DownloadProxyAck
5	Version number. A little-endian unsigned 16-bit integer. Versions traditionally start with 1, but a vendor may choose to start at 0 if he

so wishes. (In this proposal, set to 1 as the version is 0.1)

Immediately following the Vendor-specific Message header is a payload consisting of one of the following messages.

3.1 DownloadProxyRequest Message (0x01)

The message has the following fields

Bytes: Description:

0-3 File Index. The same as the QueryHit message. Tell the FS which file should be proxy transfer.

4-7 File Size. The size (in bytes) of the file wanted to be transfer. RNs can determine whether to proxy according to the file size.

8- Node Set. A set of nodes between FS and FC. This set contains Number of willing to be downloading Nodes elements, each with the following structure:

Bytes: Description:

0-1 Port. If the node routed this message is willing to be a proxy, it sets this to its proxy port.

2-5 IP Address. If the node routed this message is willing to be a proxy, it sets this to its IP Address.

Last 16 Servent Identifier. This field MUST be set to the Servent Identifier returned in the corresponding QueryHit message. This is used to route the DownloadProxyRequest message to the sender of the QueryHit message.

The Message_Id field in the Message Header of the DownloadProxyRequest message SHOULD not contain the same value as that of the associated QueryHit message, but SHOULD contain a new value generated by the servant's Message_Id generation algorithm.

DownloadProxyRequest messages are forwarded to the originator of the Query Hits message using the Servant Identifier value.

3.2 DownloadProxyAck Message (0x02)

The message has the following fields

Bytes: Description:

0-1 Port. The port of DP using for download proxy.

2-5 IP Address. The IP Address of DP.

The Message ID of a DownloadProxyAck message MUST be the Message ID of the DownloadProxyRequest message it is sent in reply to. Servant MUSTfirst make sure that the remote host supports these two message types. This can be done using handshaking headers.

4 Handshaking Sequence

After the TCP/IP connection is created, a handshaking sequence is initiated, and we MUST add our vendor-specific headers to the sequence to indicate that this proposal is supported.

Servents supported this proposal MUST exchange a "DownloadProxy: 1.0<cr><lf>" (1.0 is the version number) Private-Data header or with higher version numbers when this proposal improved.

Here is a sample interaction between two servents supported download proxy.

Servent A	Servent B
GNUTELLA CONNECT/0.6<cr><lf>	
User-Agent: Dtella<cr><lf>	
<cr><lf>	
	GNUTELLA/0.6 200 OK<cr><lf>
	User-Agent: Dtella<cr><lf>

DownloadProxy: 1.0<cr><lf>

<cr><lf>

GNUTELLA/0.6 200 OK<cr><lf>

DownloadProxy: 1.0<cr><lf>

<cr><lf>

[binary messages] [binary messages]

If one node of RNs doesn't exchange this header, this download proxy proposal to tunnel bi-firewalled servents is failed.

5 File Transfer

The files are downloaded out-of-network, and download protocol is HTTP. Once the download proxy servent is identified, FS should immediately send the following to DP:

FWGIV <File Index>: <Servant Identifier>/<File Name><lf><lf>

Where <File Index> and <Servant Identifier> are the values of the File Index and Servant Identifier fields respectively from the DownloadProxyRequest received, and <File Name> is the name of the file in the local file table whose file index number is <File Index>. The File Name MAY be url/uri encoded. DP then responses with an http GET message to FS to download the file to its buffer:

GET /get/<File Index>/<File Name> HTTP/1.1<cr><lf>

FC also makes a connection to the same port of DP and send an http FWGET message:

FWGET /get/<File Index>/<File Name> HTTP/1.1<cr><lf>

When DP receives the http FWGET message from FC, it compares the <File Index>/<File Name> pair with which was in the FWGIV message received from FS. If they are matching, then DP transfers the file from buffer to FC.

References

==========

Jason Thomas (jason@revolutionarystuff.com),

Andrew Mickish (mickish@freepeers.com),

Susheel M. Daswani (sdaswani@limepeer.com)

Push Proxy

6/25/2003

Raphael Manfredi <Raphael_Manfredi@pobox.com>, "Framework for

Vendor-specific Messages"

1/5/2003

Jason Thomas <jason@jasonthomas.com>,

"Gnutella Generic Extension Protocol (GGEP)"

2/4/2002

Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels"

<ftp://ftp.isi.edu/in-notes/rfc2119.txt>, March 1997

后 记

首先，我应该感谢我的导师吕志民老师对我的细心指导，使我能顺利完成论文。吕老师丰富的科研经验使我获益良多，无论在科研上，工作上，生活上，他都将会是我的榜样。

最衷心的感谢，要致以我的父母，家人，是他们对我的一贯支持，给予我的环境，使我的学业能得以完成。而我身边的同学，多年来互相交流，互相促进，也使我不断成长，在此一并谢过。

感谢这两年来，在我身边，给我关怀支持的朋友；对不在身边，离开了我的朋友，我都应该表示感谢，祝愿。

原创性声明

本人郑重声明：所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的作品成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

学位论文作者签名：朱志伟

日期：2005年 6 月 7 日