面向连接的P2P即时通信中继策略研究*)

段翰聪　卢显良　宋　杰
(电子科技大学计算机学院　成都610054)

摘　要　P2P即时通信模型要求所有对等节点都能进行双向通信，但网络安全技术方案的实施，使得整个P2P通信域有了内网和外网的区分，并且外网节点不能向内网主动发起连接，故P2P双向通信受到限制。本文分析并设计了一种高效的通信机制—小服务器节点(Small Service Peer)避免由第三方的通信中继服务所导致的延时。提出了内网中继判定算法，以优化处于同一内网节点之间通信的性能。试验表明，采用本文提出小服务器对等节点的通信机制和内网中继判定算法，在不改变下层网络结构和配置的条件下，可以显著降低P2P应用系统对中继服务器的依赖，并提高即时通信系统实时性。

关键词　内网，外网，小服务节点，中继判定算法

Research of Connections-Oriented Relay Policy of P2P Instant Messages

DUAN Han-Cong　LU Xian-Liang　SONG Jie
(Department of Computer Science of UEST of China, Chengdu 610054)

Abstract　It is required that all peers may initialize communication in P2P Instant Message Systems. With the deploying of network security equipment, there are two types of P2P communication domains, which are the outer network and inner network. So, the peers in outer network can not initial the inbound connection to the inner ones, and the communication between the outer and inner peers is blocked. Analyst and design a high performance communication mechanisms—Small Service Peer, which avoids the delay brought by relay service of the third side. Propose the relay judgment algorithm of inner network, which is used to optimize the communication between the peers in the same inner network. The experiments show that the Small Service Peer and the relay judgment algorithm of inner network can decrease the dependence of P2P application system to the relay server and improve the real time performance while not changing the under layer network

Keywords　Inner network, Outer network, Small service peer, Relay judgment algorithm

1　引言

在P2P系统中每个对等节点是平等的，通信是双向的，P2P通信域中任何一个对等节点都可以向其它节点发起连接请求，故不同于客户/服务器模式，即只由客户发起连接请求，服务器做出应答，服务器并不主动发起网络连接的模式。另外一方面，从网络安全的角度考虑，采取了如防火墙等网络安全技术(如包过滤、代理服务、状态监测)，同时又因为网络地址资源的日益紧张，促使了网络地址(端口)转换(NAT/NAPT)设备的大量使用，形成了许多使用私有地址的局域网段，采用上述两种技术，划分出了内网(防火墙或NAT设备保护的网络段)和外网，并限制了网络访问方式，只有合法的网络端口/地址能够进出防火墙或NAT设备，同时也限制了外网段的节点主动发起的网络连接，使得外网的主动连接通信被阻挡。P2P通信的本质要求就是要进行双向通信，但网络安全实施以及NAT的使用与之发生了矛盾，如何让内网节点和外网节点，以及处于不同或相同内网中的节点之间进行高效率的通信，是本文研究的重点。对此问题，有不同的解决办法，如文[1,2]中提出的面向连接通信协议(TCP)的NAT和防火墙通信机制在复杂的广域网环境中并不适用，原因有：一是因为OS的TCP实现差异，该机制并不适用于Windows平台；二为其穿透双NAT的TCP通信方法具有的延时限制，但由于在网络通信延时不确定故同样不适用；三为对于应用层级Web代理服务器并不支持。

通常针对NAT和防火墙的面向连接通信简单方法如下：

方法1：改变防火墙或NAT设备的设置，打开某些特定端口，使得某些外网节点能对内网某端口进行连接。

方法2：采用PUSH[3]技术，使得内网节点可以接收第三方的操作命令，改变通信发起方向，从而内网外网节点直接通信。

方法3：由通信中继服务器提供内网和外网节点或内网节点之间的中继。

方法4：将所有P2P通信结点建立一种有层次的拓扑结构，如果不能直接双向通信，则由父节点提供子节点的通信中继。

第1种方法通常会与网络安全策略相冲突，一般不会采用。第2种方法在特定情况下可行(例如在一个会话中一对节点通信中，至少有一个节点在外网)，但是通信连接可能会被防火墙阻挡。第3种方法适用于一对通信节点至少有一个节点处于内网之中，需要第三方通信中继服务器(RELAY SERVER)进行通信中继，如果通信负载较大，也会有第2种方法的问题。第4种方法因为通信拓扑为层次结构，如果不能直接到达某个节点，则经过父节点进行中继，会存在安全问题。

本文针对此问题的解决方法是采取超文本协议(HTTP)为底层通信协议，不用更改任何网络底层结构，避免了方法1带来的安全问题；结合即时通信特点，当一对通信节点至少有一个在外网的条件下，将外网节点配置为小服务器通信节点

*)受国家信息产业部电子发展基金支持(编号：信部运[2004])。段翰聪　博士研究生，主要研究方向：计算机网络、操作系统。卢显良　教授，博士生导师，主要研究方向：计算机网络、操作系统。宋杰　博士研究生，主要研究方向：计算机网络、操作系统。

以接受内网节点的通信连接(利用 HTTP),将方法2,3合并在一个节点中来实现,避免了第三方中继服务带来的延时;即使两个通信节点同处于内网时,对方法3进行优化,通过中继判定算法(Relay Judgment Algorithm),使得同处于一个内网或私网地址网段的节点直接通信,避免不必要的通信服务器中继,只有在不同内网的节点通信时才依靠第三方通信服务器中继。

P2P 即时通信包括消息、音频、视频、实时图像的传送等应用,其多媒体应用特性要求系统具有较高实时性,在腾讯公司产品 QICQ[3]中,其多媒体通信数据均采用 UDP 包发送(当网络模式设置为代理模式,则所有数据均通过服务器中转,则与方法3相同);微软公司产品 MSN[4]中,除消息数据采用 TCP+HTTP(消息全由服务器中继)之外,其它跟 QICQ 类似,它们与本文的基于面向连接的 TCP 即时通信不同,故本文提出的方法不与以上两种应用采用的方法作比较,所以在后面的章节中,将针对性的将本文提出的方法与此节中的方法2、3进行对比。

根据上面提出的方法,本文第2节是 P2P 即时通信结构模型与网络拓扑;第3节是内网判定算法;第4节是 P2P 即时通信场景分析与设计;第5节是性能测试与对比;最后是结束语。

2 P2P 即时通信结构模型与网络拓扑

2.1 P2P 即时通信结构模型

P2P 即时通信[5]应用其系统构架分为三个部分:登录控制服务器 LCS(Login Control Server)、中继服务器 RS(Relay Server)、通信节点 P(Peers)。登录服务器提供节点的登录控制,对等点登录信息与通信地址的映射,记账处理等服务。中继服务器为内网节点提供通信中继服务。通信节点登录控制服务器获取另外一方通信节点的地址信息,与之进行通信。

一个典型的 P2P 即时通信会话(Session),通常会经历三个阶段,第一是登录并身份认证,向登录控制服务器表明本用户已经上线,登记通信节点的 IP 地址和端口号,并定时向服务器发送心跳报文,通告本用户在线,确保用户 ID 和本次会话通信地址信息绑定有效;第二是获得通信目的节点的通信地址并发起通信连接;第三是如果连接成功,通信会话开始。所以,对于如何使处于防火墙或 NAT 设备后的内网节点 P2P 进行 TCP 通信,第二步是关键。

2.2 网络拓扑

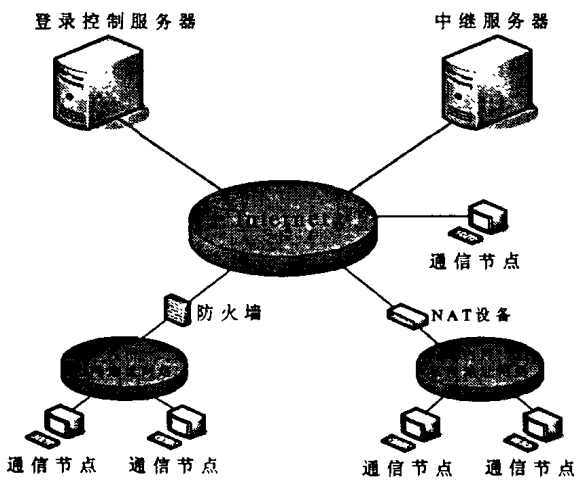

图1 P2P 即时通信网络拓扑

从图1拓扑图可见,P2P 即时通信中任何一对通信都可能发生在图中任何两个节点之间,故存在几种通信场景:场景1为一对通信节点同时处于外网;场景2为一个处于公网一个处于内网;场景3为两个节点同处于内网之中。在后面的小节中会对三个场景的通信机制逐个进行讨论。

2.3 网络协议支持

将所有 P2P 通信协议都建立在超文本协议[6](HTTP)之上,从而有利于 P2P 通信穿透防火墙或 NAT 设备,原因是:1)一般情况下,防火墙会打开 HTTP 的默认80端口,使得用户可以浏览网页,意味着允许内网节点(具有合法 IP 地址)可以通过80端口向外发起主动连接;2)可以直接使用代理服务器提供的 Web 代理(应用层)、SOCKS[7]代理(传输层)服务,使得即使内网中私有地址节点也能通过代理服务器主动向外发起连接。对于 NAT 设备,所有的上层网络议对其是透明的,它只是作 IP 地址和端口的转换。

3 内网判定算法

该算法用来判断某一节点是否处于内网之中。登录控制服务器在外网具有固定 IP 地址,任何通信节点登录时都会登录控制服务器发送 HTTP 报文,此报文内容包含了本节点的发起本次连接的 IP 地址 Addr 和端口号 Port(如果是内网节点,则此地址信息为内网地址和端口),服务器收到 HTTP 报文后,向该节点的 Addr 和 Port 发起连接,如果连接成功,则表明通信节点处于外网,否则处于内网。

4 P2P 即时通信场景分析与性能优化

本节首先给出系统可伸缩性评价指标,这个指标将在第5节得到测试和验证,然后着重讨论分析三个通信场景以及其中的通信机制。

4.1 系统可伸缩性评价指标

系统可伸缩性体现在当系统并发通信量逐渐增加,以至于到达高并发节点通信时,系统是否能满足性能要求,而不是达到系统饱和点而使得系统崩溃。

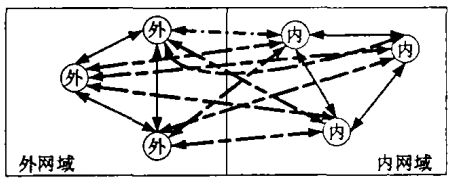

图2 内网外网通信示意图

因为要求所有 P2P 通信域中的节点都能进行通信,故可以构造一个全连接的网络模型如图2所示,则每个可能发生通信就为无向完全图的一条边。通过第3节的内网判定算法,可知某个通信节点是处于内网或是外网,从而节点性质得到确认。根据节点性质划分三个通信场景:外网节点之间的通信(场景1)、外网和内网节点之间的通信(场景2)、内网节点之间的通信(场景3)。如图,集合 $A=\{a_1,a_2,a_3,\cdots,a_m \mid a_i$为处于外网的节点,且 $i\leqslant m\}$,集合 $D=\{d_1,d_2,d_3,\cdots,d_n \mid d_i$为处于外网的节点,且 $i\leqslant n\}$。m 和 n 分别为外网节点和内网节点的数目,设外网节点向外网节点主动发起通信需中的概率为 B_o,外网节点主动向内网节点发起通信需中继的概率为 B_{oi},内网节点主动向外网节点发起通信需中继的概率为 B_{io},内网节点主动向内网节点发起通信需中继的概率为 B_i。从图2分析可得如下概率:

外网节点之间通信的概率为:

$$K_1=\frac{C_m^2}{C_{m+n}^2}=\frac{m(m-1)}{(m+n)(m+n-1)}$$

外网与内网之间通信的概率：

$$K_2=\frac{C_n^1C_m^1}{C_{m+n}^2}=\frac{n\cdot m}{(m+n)(m+n-1)/2}$$

则由外网节点主动向内网节点发起通信的概率为 K_{2o}，内网节点主动向外网节点发起通信的概率为 K_{2i}，且 $K_{2o}=K_{2i}=K_2/2$。

内网之间通信的概率为：

$$K_3=\frac{C_n^2}{C_{m+n}^2}=\frac{n(n-1)}{(m+n)(m+n-1)}$$

设 B=｛节点发起通信需中继的事件｝，P_1=｛外网节点之间发起通信需中继的事件｝，P_2=｛外网节点主动向内网节点发起通信需中继的事件｝，P_3=｛内网节点主动向外网节点发起通信需中继的事件｝，P_4=｛内网节点之间发起通信需中继的事件｝，由贝叶斯公式可得：

在场景1条件下，所有节点中继通信中，外网节点之间通信需中继所占的概率：

$$P(P_1|B)=\frac{P(P_1)P(B|P_1)}{P(P_1)P(B|P_1)+P(P_2)P(B|P_2)+P(P_3)P(B|P_3)+P(P_4)P(B|P_4)}=\frac{B_oK_1}{B_oK_1+B_{oo}K_2+B_{io}K_3+B_iK_4}=\frac{B_o(m^2-m)}{B_o(m^2-m)+B_{oo}mn+B_{io}mn+B_i(n^2-n)}\quad(1)$$

同理可得在场景2条件下，在所有节点中继通信中，外网向内网节点发起通信需中继所占的概率：

$$P(P_2|B)=\frac{P(P_2)P(B|P_2)}{P(P_1)P(B|P_1)+P(P_2)P(B|P_2)+P(P_3)P(B|P_3)+P(P_4)P(B|P_4)}=\frac{B_{oo}mn}{B_o(m^2-m)+B_{oo}mn+B_{io}mn+B_i(n^2-n)}\quad(2)$$

在所有节点中继通信中，内网向外网节点发起连接且需要中继所占的概率：

$$P(P_3|B)=\frac{B_{io}mn}{B_o(m^2-m)+B_{oo}mn+B_{io}mn+B_i(n^2-n)}\quad(3)$$

在场景3条件下，所有节点中继通信中，内网节点之间需要中继的通信所占概率：

$$P(P_4|B)=\frac{B_i(n^2-n)}{B_o(m^2-m)+B_{oo}mn+B_{oo}mn+B_i(n^2-n)}\quad(4)$$

定义1(中继影响因子(R_d)) 某个通信场景下的中继通信操作数量占整个系统中继通信操作的百分比。

显然，场景1中的中继影响因子 $R_{do}=P(P_1|B)$，场景2中的第一种情况中继影响因子 $R_{doo}=P(P_2|B)$，场景2中第二种情况的中继影响因子 $R_{dio}=P(P_3|B)$，场景3中的中继影响因子 $R_{di}=P(P_4|B)$。

定义2 中继判断因子(A)=$\begin{cases}1 & R_d\neq 0\\ 0 & R_d=0\end{cases}$，应用于计算系统中继依赖度。

定义3(中继依赖度(K_{rd})) 系统整体通信会话中，必须依赖中继通信操作才能进行 P2P 通信的会话数占总体会话数的比例。

显然，中继依赖度越低则系统的可伸缩性越强，系统整体的实时性也较高(没有中继操作引入的延时)。所以系统采取不同的中继策略，在复杂网络通信环境中将会有不同的中继依赖度，一般来说，中继依赖度越高则系统对通信服务器依赖越强，故可伸缩性较低。

一个 P2P 通信域中的 $K_{rd}=N_{relay}/N_{total}$，$N_{relay}$为需要中继的会话总数，$N_{total}$为通信域中全部会话数$=n+m$，为三个场景下的需中继的会话总数之和。所以存在：

$$K_{rd}=\frac{A_oN_o+A_{oo}N_{oo}+A_{io}N_{io}+A_{ii}N_{ii}}{N_{total}}\quad(5)$$

4.2 场景1：处于外网的节点之间的 P2P 通信

由图3可见，外网节点由于没有通信地址、端口等限制，任何一方都可以直接主动发起对另一方的通信连接，这是 P2P 即时通信三个典型通信场景中最理想的一个。只要双方登录时将自身 IP 地址和端口号提交给登录服务器，节点从服务器上获得对方的地址信息，就可以直接进行双向通信。

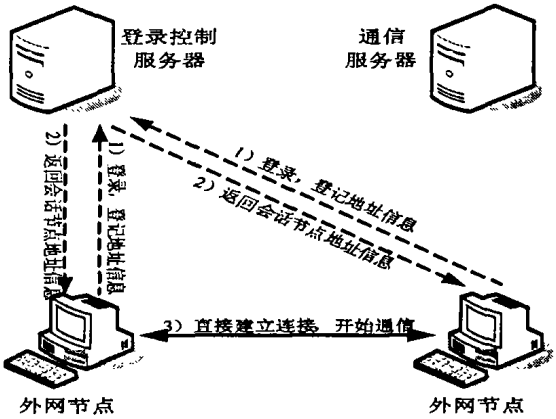

图3 通信场景1

可见，此场景下的 P2P 通信最为单纯、简单，本文的方法与 QICQ、MSN 相同，均采取直接通信无需第三方应用层中继，故 $B_o=0$，场景1中的中继影响因子 $R_{do}=P(P_1|B)=0$，式(5)中，$A_o=0$，系统的中继依赖度 K_{rd}与此场景发生的通信无关。

4.3 场景2：处于外网的节点与处于内网的节点之间的 P2P 通信

由于有一方处于内网中，P2P 双向通信受到限制，外网节点主动连接由于防火墙的通信限制被拦截，或者由于 NAT 的动态地址端口映射机制被 NAT 丢弃。

对此问题，第1节方法3配置一个具有固定合法 IP 地址的通信服务器，为内网节点提供通信中继。但是这种方案存在着缺陷，首先是系统可伸缩性受到影响，每个通信服务器都有通信负载的上限，一旦内网节点通信超过服务器负载上限，容易造成通信服务器崩溃[8]；其次造成了通信瓶颈，服务器的带宽同样有限，高并发的网络流量将会使信道拥塞，由此引起数据掉包会引起更多的重发数据，从而导致拥塞进一步加剧；此外，每次通信服务器的数据转发操作都会产生处理延时。上述缺陷都会导致 P2P 即时通信系统不稳定性和低实时性，从而大大影响有较高实时性要求的 P2P 通信应用。可见，改变通信中继策略，降低此场景下的中继依赖度 K_{rd}可显著改善通信性能。方法2采用 PUSH 技术，改变了连接的发起方向由内网向外网节点发起，但是这个 TCP 连接有可能被防火墙阻挡。

4.3.1 外网的节点主动向内网节点发起通信

本文提出的算法：外网节点的主动连接通常会被 NAT/防火墙等设备阻塞，我们可将外网节点配置成为一个小服务器对等节点(Small Service Peer)，除发送自身数据之外，同时接受 HTTP 协议的网络连接。内网节点会周期性的向登录控制服务器发送心跳报文，如果一旦有外网向内网发起的连接请求，在下一个内网节点的心跳报文到达时，则服务器会将此请求的外网节点 IP 地址和服务端口号80转发给内网节点，由内网节点利用 HTTP 协议向外网节点80端口发起连接。从图4可知，此算法避免了通信服务器中继，使得 $B_{oi}=0$，故中继影响因子 $R_{doi}=P(P_2|B)=0$。

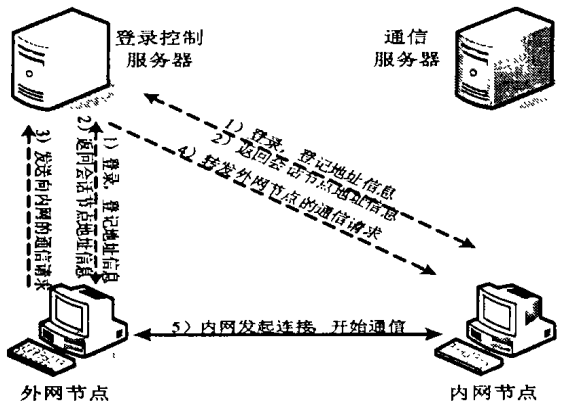

图4 通信场景2(外网节点发起连接)

4.3.2 *内网的节点主动向外网节点发起通信* 如果通信由内网节点发起，则相对比较简单，因为几乎所有 NAT、防火墙(代理)设备都允许从内网发起的 HTTP 协议80端口连接。直接由内网节点利用 HTTP 向外网节点的80端口发起连接。同样没有通信服务器中继，使得 $B_{io}=0$，故中继影响因子 $R_{dio}=P(P_3|B)=0$。

4.4 场景3：处于内网的节点之间的 P2P 通信

即使会话中双方都处于内网中，还存在两种可能：第一种情况为会话双方处于同一个内网中，节点之间且可以直接通信，第二种情况为会话双方处于不同的内网中，必须依靠中继才能通信。但是在第1节方法2、3中，它并没有对场景3进行这两种情况的划分，从而使得会话中所有数据通信均需要服务器中继，使得响应时间加大；P2P 系统的可伸缩性和实时性受到较大影响。但是，划分出场景3中的两种情况，我们还需要一种判定算法－内网中继判定算法，使得处于同一个内网中的节点之间的通信无需中继，降低系统的中继影响因子，提高系统的可伸缩性和实时性。

4.4.1 *不同内网域之间的通信概率* 设在 P2P 通信域中，内网域为 N，其节点总数为 N_p。且分布在 K 个内网域中，$K\leqslant N$。N_i为第 i 个内网域，$N=\{N_1,N_2,N_3,\cdots,N_i,\cdots,N_k\}$。

场景3中发生在不同内网域之间的通信概率为：

$$P_{id}=\Big(\sum_{\substack{i\neq j\\ 1\leqslant i,j\leqslant k}}^{k}(C_{N_i}^1C_{N_j}^1)\Big)/C_{N_p}^2 \qquad (6)$$

从式(6)可知，内网域划分越小，则 k 越大，不同内网域之间的通信概率 P_{id}越大。

4.4.2 *内网中继判定算法* 此算法需结合第3节的内网判定算法，首先判断当前会话的节点是内网节点。具体算法如图5。

```
Bool IRA(){
If (IPA(Peer1)==true&&IPA(Peer2)==true){
  ExchangePublicKey(Peer1,Peer2,SessionID);
  PAP1=Private Addr(Peer1);//假定 Peer1 为发起方
  PAP2=PrivateAddr(Peer2);
  If(Compeer(PAP1,PublicKey_Peer2,PAP2)==true)
    Retrun true;
  Else
    Return false;
}
```

图5 内网中继判定算法

具体说明如下：

第一步：根据第3节的内网判定算法 IPA(Peer)判断两个节点 Peer1和 Peer2是否为内网节点。因为此算法应用前提为会话中的节点必须同为内网节点。

第二步：因为内网地址可能包含私网地址，存在地址重复、冲突的可能，从系统安全性出发，需要对当前会话 ID(全局唯一)用对方的公匙 PublicKey_Peer 进行加密，故首先应由服务器交换会话双方公匙 ExchangePublicKey(Peer1，Peer2)。

第三步：从内网节点向登录控制服务器发出的登录控制报文中取出各自节点的内网地址和侦听端口号。

第四步：由通信发起方的内网节点 Peer1先向 Peer2发起 HTTP 连接，如果 Peer2应答建立连接，Peer2将会话 ID 数据用 Peer1的公匙加密后发给 Peer1，Peer1收到后用自己的私匙解密，并与从服务器获得的会话 ID 相比较，如果相同则表明会话双方处于同一内网中，且可以直接通信。当然，也存在其它几种结果：1)连接失败，表明不在同一内网中。2)连接成功，但会话 ID 与服务器获得的 ID 不同，表明有其它节点 Peer3冒充并不在同一内网中的节点 Peer2，但被发现。

4.4.3 *处于相同内网中的节点通信* 如图6，如果通过内网中继判定算法判断出会话双方节点可以直接通信，则由通信发起方节点向对方的内网地址及端口发起连接。当成功建立连接后，双方开始数据通信。可见，此类通信方式无需通信服务器中继，但在方法3中，将此类本可以直接通信的节点一律通过中继来进行，所以造成系统中继依赖度较高。

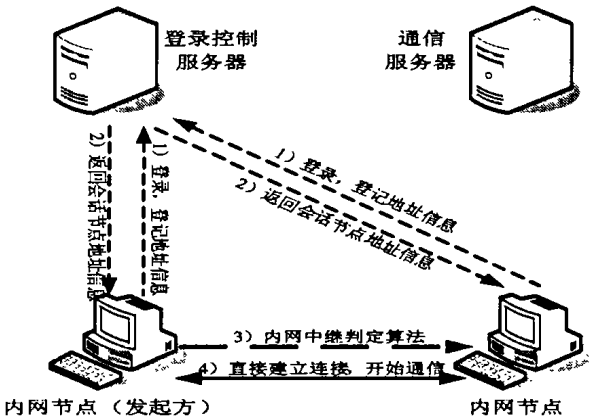

图6 通信场景3(相同内网节点之间通信)

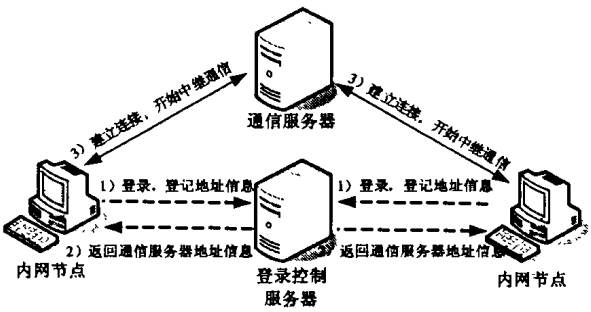

图7 通信场景3(不同内网节点通信)

4.4.4 *处于不同内网中的节点通信* 从图7可见，通过上节的内网中继判定算法，如果判断出会话中双方节点不处于同一个内网，则表明两个节点之间的通信必须通过通信服务器中继。从图6可知，会话双方节点将数据发送到通信服务器，同时如果检测到服务器上有发向自己的数据则取回。因为需要通信中继，如果此种类型的通信占系统总体通信的比例较大，也将影响系统的中继影响因子和系统可伸缩性。这一点我们将在第5节中得到验证。在此类通信中，$B_i=P_{id}$(不同内网域之间的通信概率)，$R_{di}=p(p_4|B)=\dfrac{P_{id}(n^2-n)}{B_o(m^2-m)+B_{oi}mn+B_{io}mn+P_{id}(n^2-n)}$，通过上面几个小节的分析可知，$B_{oi}=B_{io}=0$，$R_{di}=1$，即此场景3的第二种情况的

中继影响因子最大，所有的通信中继行为只发生与此，由式(5)，$A_o=A_{oi}=A_{io}=0$，$A_{ii}=1$故系统的中继依赖度 $k_{rd}=\frac{1\cdot N_{ii}}{N_{total}}$

$=P_{rd}$，$P_{rd}=\frac{\sum_{1\le i,j\le k,i\ne j}^{k}(C_{N_i}^1 C_{N_j}^1)}{C_{N_p}^2}$，即系统中继依赖度 K_{rd} 只与不同内网节点通信占所有节点的通信比例相关。

5 性能测试与对比

测试环境：登录控制、通信服务器采用奔4 1.7G、512M内存、100M网卡，服务器负载模拟采用10台PC，配置为：赛扬1.2G、256M内存、100M网卡。

测试方法：依据第2节的方法2、3和本文提出的中继策略分别实现[10]了三个面向连接的即时通信系统(包含登录控制服务器、通信服务器)，客户端实现为LPEclient程序(负载发生器)，模拟产生会话请求。因为在LAN上测试，为反映出Real World的情况，需要通过仿真软件(FreeBSD 5.1下的Dummynet)模拟广域网网络条件[9](网络带宽为10Mbytes/s，延时为20ms)，使得任何到通信服务器的RTT延时＞40ms。将第4节提出的高效P2P中继策略与第1节的方法2、3进行对比测试。

测试项目：通信服务器中依赖度(K_{rd})测试、系统可伸缩性(S)测试。

5.1 通信服务器中继依赖度(K_{rd})测试

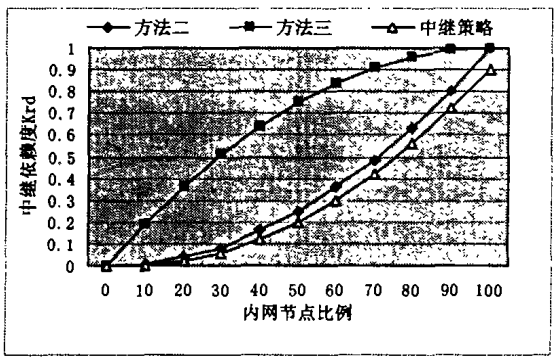

图8 中继策略对比

设定外网初始节点数为1000，内网初始节点数为0；为改变内外网节点比例，内网节点按100个节点数递增，外网节点数按100个节点数递减；内网节点每100个节点划分到同一个子网，使得存在最多10个不同的内网子网，内网节点与外网节点之间的通信使用http代理服务器。从图8可知，随着内网节点比例的增大，三种方法中方法3中继依赖度最高，采用本文提出的中继策略表现出现出了较低的中继依赖度。

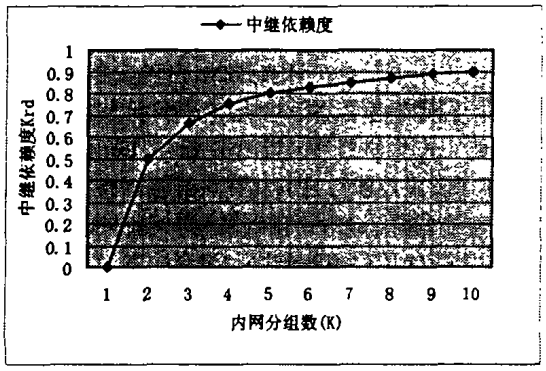

图9 不同内网子网数 K_{rd}

为观测中继判定算法的作用，将1000个节点全部设置为内网节点，并且内网子网数目分别按1，2，…，10逐渐增加，且相同子网中的节点可以直接通信。从图9可见，当分组数增加时，组的粒度越小，则中继依赖度越高，组的粒度越大，则中继依赖度越低。

5.2 系统可伸缩性(S)测试

本测试项目为验证方法2、3和本文的中继策略对即时通信系统可伸缩性的影响。测试方法为系统从600个节点开始，按200个节点逐渐增加节点数，并且保持内外网节点数比例各为50%，且内网节点中固定分为8个子网，每个子网中的节点数目是内网节点总数1/8。

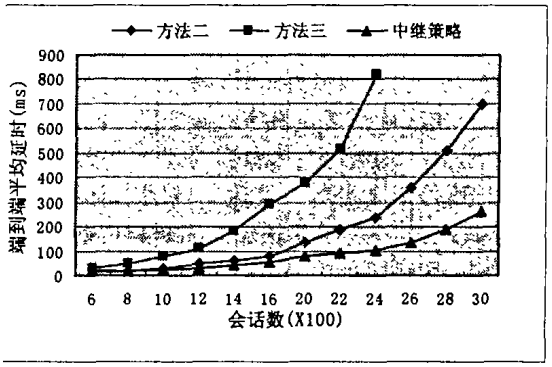

图10 系统可伸缩性测试

从图10可见，方法3由于只要会话中存在内网节点，就需中继，故其中继依赖度较高，在系统负载为2400个会话左右时，其端到端延时超过800ms，并且中继服务器已达到饱和点。方法2在2400个时，其延时增加速率加大，在3100个会话时，端到端延时已经达到700ms。采用本文的中继策略，由于中继服务器负载较小，故端到端延时变化曲线较为平稳，在3100会话时，其端到端延时为260ms左右，体现出在高负载下时，系统有更优的可伸缩性，并且端到端延时得到较大改善，系统实时性也得到提高。

结束语 本文首先讨论了一个典型的即时通信模型，并针对即时通信中NAT、防火墙问题，提出了内网判定算法和即时通信系统可伸缩性指标，划分并分析了三个通信场景，特别是对内网中的节点通信给出了小服务节点和中继判定算法，通过对比试验证明，本文提出的中继策略有效地提高了即时通信系统可伸缩性和实时性。

参考文献

1 Gnutella Protocol Specification, version 0.4. available at: http://www.clip2.com/GnutellaPro-tocol04.pdf, 20.03.2002
2 Peer-to-Peer Working Group. Bidirectional Peer-to-Peer Communication with Interposing Firewalls and NATs. NAT/Firewall Traversal Whitepaper, Revision 0.095, Aug. 2001. http://www.p2pwg.org/tech/nat/
3 http://www.tencent.com
4 http://www.msn.com
5 Milojicic D S, Kalogeraki V, Lukose R, et al. Peer-to-Peer Computing. HP Laboratories Palo Alto HPL-2002-57 March 8, 2002
6 HTTP 1.1. http://www.zvon.org/tmRFC/RFC2616/Output/index.html
7 Socks 5. http://www.zvon.org/tmRFC/RFC1928/Output/index.html
8 C10K Problem. http://www.kegel.com/c10k.html
9 Banga G, Druschel P. Measuring the Capacity of a Web Server [Z]. USENIX Symposium on Internet Technologies and Systems, 1997
10 Stevens W R. Unix Network Programming Networking APIs: Socket and XTI(Second Edition)[M/CD]. Prentince Hall International, Inc. 1998