FALPS:大规模 P2P 系统网络仿真平台

武广柱[1,2],王劲林[2]
WU Guang-zhu[1,2],WANG Jin-lin[2]

1.中国科学院 声学研究所,北京 100080
2.中国科学院 研究生院,北京 100080
1.Institute of Acoustics,Chinese Academy of Sciences,Beijing 100080,China
2.Graduate University of Chinese Academy of Sciences,Beijing 100080,China
E-mail:wugz@dsp.ac.cn

WU Guang-zhu,WANG Jin-lin.FALPS fast accurate large-scale Peer-to-Peer simulator.Computer Engineering and Applications,2008,44(11):9-12.

Abstract: Simulating large scale Peer-to-Peer(P2P) networks efficiently is still challenging.In this paper,FALPS,a discrete event simulator for large scale network simulation,is designed.BackboneNet model is used in the simulator.Based on this model,the authors present three strategies to make FALPS efficient:event combination,piece discarding and event queue size control.Results from performance experiments show that except for its extra height speed and accuracy,FALPS reduces the memory consumption significantly.FALPS can be used to simulate a P2P system of 100 000 nodes in 1 664 s,only consuming about 60 MB memory.
Key words: network simulation;packet level;flow level;Peer-to-Peer(P2P)

摘 要:随着 P2P 技术的发展,传统的网络仿真平台已经不能满足研究需要。如何设计支持大规模 P2P 内容分发仿真的平台是亟待解决的问题。通过建立 BackboneNet 模型,并采取了"事件"合并、非尾片段丢弃、"事件队列"大小控制三个关键算法设计了一种用于大规模 P2P 内容分发系统的包级离散事件驱动网络仿真平台 FALPS。该平台具有内存消耗低、速度快、精确度高的特点,可用于仿真具有 105 数量级节点规模的 P2P 系统。
关键词:网络仿真;包级;流级;Peer-to-Peer(P2P)
文章编号:1002-8331(2008)11-0009-04 **文献标识码:**A **中图分类号:**TP301

1 前言

网络通信的研究需要仿真平台的支持以评估研究结果。传统的包级(packet-level)网络仿真平台如 NS-2[1]等可以精细的模拟网络通信细节,被广泛应用。然而,随着网络通信技术特别是 P2P 技术的发展,这些仿真平台已经无法满足大规模应用层仿真的需求。为了提高仿真速度,很多研究者从应用分布式计算、设计高效的数据结构、建立新的仿真模型等不同方面对大规模高性能网络仿真平台进行了研究。除速度是大规模仿真的瓶颈外,内存占用也是一个非常严峻的问题。就仿真平台的路由表来讲,按照通常的设计,空间复杂度为 $O(N^2)$,其中 N 是节点个数。如果仿真一个 10 万节点的网络系统,在路由表内部数据类型为 int32 的情况下,单路由表将消耗 37 G 的内存。为此,一些基于哈希的路由算法(NS-2)、基于前缀的路由算法(SSFNet[2])、基于 NIx Vector 的路由算法(GTNetS[3])等被相继提出,但内存消耗仍然严重。

本文设计了一种用于大规模 P2P 系统仿真的平台 FALPS(Fast Accurate Large-scale Peer-to-Peer Simulator)。FALPS 采用了包级仿真模型(packet-level model),省略了对应用层仿真所不关心的网络底层通信细节的模拟,在建立 BackboneNet 和抽象门模型的基础上,通过"事件"合并、非尾片段丢弃、"事件队列"大小控制三个算法极大的降低了内存消耗并提高了仿真速度。FALPS 可在仅消耗几十兆内存情况下仿真具有 105 数量级节点规模的 P2P 系统,并且具有非常高的精度和速度。

本文组织如下:第 2 章概括了本领域的相关研究;第 3 章分析了包级仿真平台的速度瓶颈和内存消耗情况,并给出 FALPS 的设计细节;第 4 章为平台的测试结果并同 GtNetS 仿真平台做了比较;第 5 章是结束语。

2 相关工作

网络仿真平台按照驱动方式可以分为离散事件驱动和离散时间驱动两种。现有的绝大多数平台都是离散事件驱动仿真平台。"事件"代表了某抽象时间发生的动作,它被仿真器预先

基金项目:国家高技术研究发展计划(863)(the National High-Tech Research and Development Plan of China under Grant No.2005AA1032);中国下一代互联网示范项目(the China Next Generation Internet(CNGI)under Grant No.CNGI-04-15-2A)。
作者简介:武广柱(1979-),男,博士生,主要研究领域为宽带多媒体通信,嵌入式系统;王劲林(1964-),男,博士生导师,主任研究员,主要研究领域宽带多媒体通信。
收稿日期:2007-11-28 **修回日期:**2008-01-21

插入“事件队列”,并在其发生时刻被仿真器执行。离散事件驱动平台假设系统状态随“事件”的发生而离散变化,“事件”和“事件”之间系统状态保持不变。按仿真模型划分,网络仿真平台又可分为包级(packet-level)仿真平台和流级(flow-level)仿真平台两种。包级仿真具有精确的优点,而流级仿真具有速度高的优点。但文献[4]指出,随着网络规模的进一步扩大,“波纹效应”的存在往往使流级仿真平台的性能迅速衰减,甚至会出现低于包级仿真平台的情况。

多数传统仿真平台的最初设计目标并不是支持大规模应用层仿真,此类平台不适合用于 P2P 系统的仿真。下面介绍几种典型的特别是适合大规模 P2P 系统仿真的平台。

2.1 NS-2

NS-2 是用 C++语言开发的开源包级离散事件驱动网络仿真工具。它按照 OSI 七层模型进行设计,是最为广泛使用的网络仿真平台。NS-2 适合需要精细结果的小规模网络仿真。PNDS[5]对 NS-2 进行了分布式扩展,使得 NS-2 可以有限的适应较大规模的仿真应用。NDP2PSim[1]对 NS-2 用于 P2P 仿真做了研究。

2.2 Narses[6]

Narses 是一种用 Java 开发的开源流级离散事件驱动分布式网络仿真平台。其设计目标是在保证一定精度的前提下提高仿真速度并降低内存消耗以支持大规模应用层仿真。Narses 忽略底层细节,对网络模型做了一些假设:骨干网带宽是足够大的,节点间的通信带宽仅受限于整个逻辑链接的第一条物理链路。平台测试了 600 个节点传送 10 000 个固定 200 k 大小的流。与 NS-2 相比,虽然有平均 8%的精度损失,但获得了 45 倍的速度提升,内存消耗也仅是 NS-2 的 28%。然而,随着网络规模的扩大和流大小的增加,误差可能会超过 8%。

2.3 P2PSim[7]

P2PSim 是 MIT 用 C++开发的开源包级离散事件驱动网络仿真平台,用于仿真结构化 P2P 网络,支持 Chord、Accordion、Koorde、Kelips、Tapestry、Kademlia 等 P2P 拓扑结构。P2PSim 同样将网络底层通信细节做了抽象。具有 3 000 节点的 Chord 网络已经利用该平台做过仿真。P2PSim 的设计目标是仿真仅需要节点间控制信令的 P2P 覆盖层拓扑组织。因控制信令非常小,其传输时间主要由网络延迟而非带宽决定,因此 P2PSim 忽略了对节点间带宽的模拟。需要考虑带宽因素的 BT、Cool-Streaming 等 P2P 网络共享和媒体分发系统的仿真无法应用此类平台。

2.4 Omnet++[8]

Omnet++是一种用 C++开发的开源离散事件模拟工具。Omnet++具有独特的网络拓扑语言 NED,可用于通信网络模型仿真、协议仿真、硬件体系结构仿真、复杂软件系统性能评估等。已经有实验[9]证明,Omnet++可以用于仿真拥有 1 000 个节点规模的 Swarming 网络。

2.5 GPS[10]

GPS 是用 Java 开发的开源消息级(message-level)离散事件驱动网络仿真平台。它采用了动态链路级网络带宽模型来调节各个逻辑链接的带宽占用。这种模型的提高了仿真速度,并且保证具有较高的精度。已经在 GPS 上做过具有 512 个节点的 BT 网络分发系统的仿真测试。然而,某一逻辑链接的带宽发生改变,很可能需要调整经由与本逻辑链接相关的物理链路段的各个逻辑链接的带宽,而这种改变又会导致更多逻辑链接带宽占用的改变,这样,在网络规模较大情况下同样会出现“波纹效应”。GPS 在具有数千数量级节点的网络系统仿真上的性能表现有待进于一步研究证实。

2.6 GTNetS

GTNetS 是一种用 C++开发的开源包级离散事件驱动网络仿真平台。在保证精度的情况下,设计者着重考虑了减少事件数量和节省内存消耗,以满足大规模网络仿真的需要。GTNetS 提出了 FIFO+Abstract Packet Queue+Time Buckets 的算法,大大减少了事件数量,提高了仿真速度。在节省路由表内存占用上,GTNetS 利用了边缘节点流必定流经与其直接相连的路由器的特点并结合 NIx—vector 技术使得内存占用和仿真节点数量基本呈线性关系。

3 FALPS

FALPS 是基于离散事件驱动模型的包级网络仿真平台。在此类平台中,直接相连的两节点间传送数据所需时间由两部分组成:传输延迟(transmission delay)和传播延迟(propagation delay)。传输延迟过后链路将处于空闲状态,源节点可以发送下一个数据包,但目的节点需要再经过一段传播延迟时间才能收到数据包。因此,一个数据包的发送至少对应两个“事件”:一个是源节点发送完毕,一个是目的节点收到数据。仿真平台中的节点有开/关两个状态,对应于与其相直接连接的物理链路的繁忙/空闲状态。节点将应用层传输来的流(flow)依据 MSS(Max Segment Size)分成若干片段(segment),并将其按照一定的顺序存入发送缓冲区。当与节点相连的链路处于空闲状态时,如果节点发送缓冲中有待发送的片段,节点将进入关闭状态并从发送缓冲区取出一个片段后产生“发送完成”和“收到片段”两个“事件”插入到仿真平台的“事件队列”中。仿真器将从“事件队列”中按照发生时间的先后顺序依次取出“事件”,并执行“事件”规定的动作:“发送完成”将导致发送节点再次进入空闲态从而可以发送下一个片段,“收到片段”将导致对端节点或转发该片段或将该片段传送给应用层或做丢弃处理。节点在开启/关闭的不断交替中将流发送到对端。

从上述分析可以看出,“事件队列”必须具有发生时间的有序性,因此队列不能采用插入/删除复杂度为 O(1)的链表来设计,而通常采用具有 O(lg$N$)复杂度的 map<time,event>的数据结构,N 为“事件队列”的大小。

仿真平台本身的运行时间主要同四个因素有关:流的数量 N_f(Number of flows),每个流的分片数 S_f(Segments per flow),每个流从源节点到目的节点的的平均跳数 H_f(Hops per flow),每仿真一个片段的发送接收动作所需要的时间 T_s(Time per segment)。运行时间 T 可以由下式表示:

$$T\approx(N_f*S_f*H_f)*T_s$$

从上式可以看出,减小表达式右边的任何一个变量,都可以提高仿真速度。N_f 是由具体的仿真应用决定的。在流大小一定的情况下,S_f 和 MSS 的大小成反比,增大 MSS 可以减小 S_f 从而可以提高仿真速度,但如果 MSS 取值过大将给仿真结果带来较大误差。H_f 是路由跳数,由网络拓扑决定,但对于 P2P 应用层仿真,可以通过合理抽象将 H_f 降低,从而使得仿真时间缩短。T_s 主要耗费在具有 O(lg$N$)复杂度的“事件”插入/删除操作上,因此,减小“事件队列”的大小 N,可以提高平台的仿真

速度。

基于上述分析,FALPS 采用了关键算法,有效地提升了仿真速度并降低了内存消耗:将骨干网络抽象为 BackboneNet 来减小 H_f,同时大大降低了路由表的内存消耗;严格控制节点向"事件队列"的插入操作,以减小 T_s;通过抽象门的事件合并、延迟插入、片段丢弃进一步减小 T_s 和降低内存消耗。

在 FALPS 中,每个节点都有两个通道链接到 BackboneNet:一个是上行通道,它连接到 BackboneNet 的抽象接收门上;一个是下行通道,它通过 BackboneNet 的抽象发送门和 BackboneNet 相连,如图 1 所示。

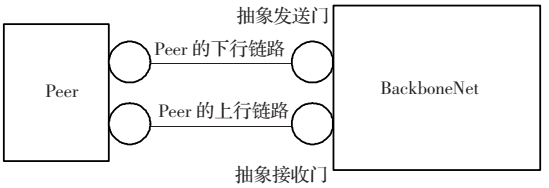

图 1 FALPS 结构

3.1 BackboneNet 模型和"事件"合并

同 Narses 相似,本文将和边缘节点直接相连的物理链路以外的路由传输链路抽象为 BackboneNet。BackboneNet 是一个骨干网络的抽象,它具有足够大的带宽。片段在 BackboneNet 内部的传输延迟可以忽略,其传送时间仅由传播延迟决定。Narses 还假设了节点的通信带宽仅受限于整个逻辑链路的第一个物理链接。与之不同,FALPS 不忽略发送门到目的节点的链路带宽和传输延迟,这大大提高了 FALPS 的仿真精度。

BackboneNet 的构成可以先通过 GT—ITM 生成一个网络拓扑,并记录边缘节点链接到哪个路由节点。这些和边缘节点直接相连的路由节点将被抽象为抽象门(抽象接收门和抽象发送门),计算各抽象门间的网络延迟便构成了 BackboneNet。两节点在 BackboneNet 中的延迟就是其所依附的抽象门间的延迟。

节点发送的片段到达 BackboneNet 的抽象接收门后,经过一定的延迟后被送到和目的节点相连的抽象发送门,抽象发送门将片段转发到目的节点。整个过程中,仿真器需要 5 次"事件"的插入操作:A:发送节点发送完毕;B:接收门收到片段;C:发送门收到片段;D:发送门转发完毕;E.目的节点收到片段。其中 A 和 B 之所以必须是两个"事件"是因为他们发生的时间不同,B 和 C 之间不需要接收门发送完毕"事件"是因为 BackboneNet 内部忽略传输延迟。"事件"C 的发生时间可以由下式表示:

$$T_C=\text{SegmentSize/Bandwidth}(A,B)+\text{Delay}(A,B)+\text{Delay}(B,C)$$

$$\underbrace{\qquad\qquad\text{传输延迟}\qquad\qquad}_{\uparrow\ Event_A}\ \underbrace{\text{传播延迟}}_{\uparrow\ Event_B}\ \underbrace{\text{传播延迟}}_{\uparrow\ Event_C}$$

然而仔细分析 BackboneNet 模型后发现,A 和 B 可以合并为一个"事件"。片段在 B 和 C 间的传输不存在排队问题也不争用带宽,因此可以将表达式后面两项合并而不影响仿真结果。合并的意义是使得节点到 BackboneNet 的上行链接的传播延迟被归并到了 BackboneNet 内部的传播延迟,即:

$$T_C=\text{SegmentSize/Bandwidth}(A,B)+\text{Delay}(A,B)+\text{Delay}(B,C)$$

$$\underbrace{\qquad\qquad\text{传输延迟}\qquad\qquad}_{\uparrow\ Event_A\&Event_B}\ \underbrace{\qquad\text{传播延迟}\qquad}_{\uparrow\ Event_C}$$

同一发送门的 D"事件"和 E"事件"的发生时间差是一个常数(目的节点下行链接的传播延迟)。从 C 到 D 的时间差为片段的大小和目的节点下行带宽的比值,一旦片段大小固定,则不同片段的 C"事件"到 E"事件"的时间差将是一个固定值。因此,在分段大小相同的条件下,事件 D 和 E 又可以合并。

事件合并减少了对"事件队列"的插入操作次数从而使得 T_s 降低,提高了仿真速度。

3.2 抽象门和事件的延迟插入算法

FALPS 在抽象接收门上采用了事件插入延迟算法来使同一时刻"事件队列"的规模变小。节点发送的片段到达 BackboneNet 的抽象接收门后并不立刻向"事件队列"插入上节所述的 C"事件",而是采取策略,尽量推迟这一插入操作。到达同一个接收门的片段可能属于发往不同目的节点的流,这就意味着它们所要到达的抽象发送门不同。而同一接收门和不同发送门间的延迟时间往往不同,后到达接收门的片段可能先到达其发送门,因此,不能在抽象接收门上直接使用 O(1)复杂度的 FIFO 队列。FALPS 使用了 FIFO 队列来延迟插入操作,但并不是将所有到达的片段都放入缓冲,只有那些 C"事件"发生时刻晚于其前驱片段 C"事件"发生时刻的片段才被插入 FIFO 缓冲。这样,凡是插入到 FIFO 缓冲的片段的 C"事件"发生时刻具有严格的时间顺序,故可将一个片段的 C"事件"插入操作延迟到其前驱片段的 C"事件"发生之后。这种尽量构成一个 FIFO 队列的算法,既能确保仿真器的正确状态,又能尽量减小"事件队列"的规模,提高了仿真运行速度。

```
//接收门收到片段,插入延迟算法
PROCEDURE AB_Event(thisSegment)
timeHappen_this=C_Event happen time of thisSegment;
IF(fifo buffer is empty)THEN
  IF(a C_Event has been scheduled)THEN
    timeHappen_that=C_Event happen time of the scheduled segment;
    IF(timeHappen_that
    <timeHappen_this)THEN
      fifo.push(timeHappen_this ,thisSegment);
    ELSE ScheduleC_Event(timeHappen_this,thisSegment);
    END IF
  END IF
ELSE
  preSegment=fifo.back();
  timeHappen_that=C_Event happen time of the preSegment;
  IF(timeHappen_that<timeHappen_this)THEN
  fifo.push(timeHappen_this,
  thisSegment);
ELSE ScheduleC_Event(timeHappen_this,thisSegment);
  END IF
END IF
END AB_Event
//片段到达发送门
PROCEDURE C_Event(thisSegment) IF (fifo buffer is not empty)THEN
    nextSegment=fifo.front().event;
    nextTime= fifo.front().time;
    fifo.pop(); ScheduleC_Event(nextTime,
    nextSegment);
END IF
SendgateRecvdNewSegment(thisSegment);
END C_Event
```

FALPS 在抽象发送门上采用了非尾片段丢弃策略。FALPS 忽略 DropTail 等丢包模型,所有到达抽象发送门的片段都将被

发送到目的节点。一个抽象门可能会同时向一个节点转发来自不同源节点的流,这些流将按照其各自片段到达的顺序在发送门内形成一个发送缓冲队列。如果将这一缓冲中各片段的“事件”都插入到“事件队列”,则同样会造成“事件队列”的急剧膨胀。当发送门向目的节点发送片段时,链路将被占用,各片段严格按照先入先出的顺序通过发送缓冲区。因此,缓冲中的任何一个片段到达目的节点的时间一定晚于它在缓冲中的前驱片段的到达时间。这样,可以在一个片段发生 *D*“事件”以后再插入其后继节点的 *D*“事件”。FALPS 并不将所有需要转发的片段都插入缓冲队列,而仅仅将一个流的最后一个片段插入。一个片段到达发送门后,首先检查其前驱片段将要被转发完毕或者已经被转发完毕的时刻,如果此时刻先于现在的抽象时刻,则表示其前驱片段早已被发送完毕,现在可以立刻发送刚到达的片段;否则,需要根据前驱片段发送出去的时刻计算本片段何时将要被转发完毕,并记录这一时刻以供后继片段利用。如果片段不是一个流的末尾片段,则仅仅记录其被转发完毕的时刻,而只有一个流的末片段才有必要向“事件队列”插入一个目的节点的 *E*“事件”。这相当于节点将整个流组装完整后才通知应用层。当一个末片段到达发送门后,是插入目的节点的 *E*“事件”还是放入发送缓冲,要看本发送门的发送缓冲是否已经有一个 *E*“事件”在“事件队列”中了。如果已经有了,则 *E*“事件”发生后会去检查并处理发送缓冲中的下一个片段,而无需提前插入。这种处理方法大大降低了总的“事件”数量和发送门缓冲队列的大小,并且使得一个发送门某时刻最多在“事件队列”有一个 *E*“事件”。

```
//发送门收到片段,转发到目的节
PROCEDURE SendgateRecvdNewSegment(thisSegment)
  IF(lastIdleTime of this gate <=now)THEN
  lastIdleTime=now+segmentSize/bandwidth;
ELSE
  lastIdleTime=lastIdleTime+segmentSize/bandwidth;
END IF
  IF(segment is the last one of a flow)THEN
   IF(a DE_Event has been scheduled)THEN
     thisSegment.outBufTime=lastIdleTime;
     FifoBuf.insert(thisSegment);
ELSE
    reachTime=thisSegment.outBufTime
    +delayFromBackboneToDst;
    ScheduleDE_Event(reachTime,thisSegment);
  END IF
END IF
END SendgateRecvdNewSegment
//发送门转发完毕,目的节点收到流
PROCEDURE DE_Event(thisSegment)IF(FifoBuf is not empty)
THEN
    nextSegment=FifoBuf.pop();
    reachTime=nextSegment.outBufTime+delayFromBackboneToDst;
    ScheduleDE_Event(reachTime,nextSegment);
  END IF
  destationNode.recvd();
END DE_Evnet
```

4 实验结果

在 BackboneNet 的内部延迟为 0 时,BackboneNet 将成为一个 switch,整个网络将构成一个以 switch 为中心的星型拓扑结构。在实验中,取用了此拓扑结构,对比平台选取了支持大规模仿真应用并具有高精度的 GtNets。实验仿真多个客户端节点从一个服务器下载一个大小为 10 M 的文件,各节点到 switch 的传输延迟均为 10 ms,节点每发出一个下载请求,服务器向其发送一个大小为 256 k 的流,直到整个文件下载完毕。MSS 大小为 1 k。实验用的机器 CPU 为 Pentium 4,主频 2.66 GHz,操作系统为 Windows XP。共进行了 5 次对比实验,分别仿真一个具有 1 000、2 000、3 000、4 000、5 000 个节点的星型拓扑网络,以比较 FALPS 和 GtNets 在内存占用、仿真速度上的表现。本文也将 FALPS 得出的仿真结果和 GtNetS 的结果做了比较(图 2、3)。

在网络规模为 5 000 节点时,FALPS 的平均内存占用仅为 GtNets 的 1.8%;平均仿真时间仅为 GtNets 的 13.3%。各次实验,FALPS 的仿真结果和 GTNetS 一致。

此外,PALPS 还用 1 664 s 的时间仿真了具有 10 万节点的系统,内存占用仅为 60 272 KB。

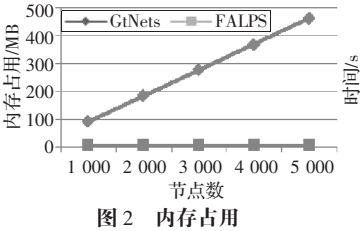

图 2 内存占用

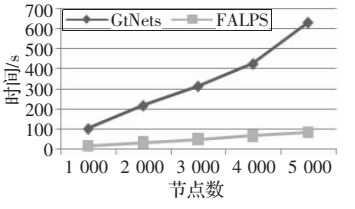

图 3 仿真时间

5 结束语

本文介绍了一种用于大规模 P2P 系统仿真的离散事件驱动网络仿真平台 PALPS。PALPS 通过 BackboneNet 模型、抽象门、“事件”延迟插入以及转发门非尾片段丢弃算法在保证非常高的精度的前提下,大大节降低了内存消耗并提高了仿真速度。PALPS 单机可支持 10 万节点规模的 P2P 系统仿真。

参考文献:

[1] The Network Simulator NS-2[EB/OL].(2006-10).http://www.isi.edu/nsnam/ns/.

[2] Scalable simulation framework net[EB/OL].(2006-10).http://www.ssfnet.org.

[3] Riley G F.Large-scale network simulations with GTNetS[C]//Proc of the Simulation Conference 2003,2003:676-684.

[4] Liu B,Figueiredo D R,Yang G,et al.A study of networks simulation efficiency:fluid simulation vs packet-level simulation[C]//INFOCOM 2001,2001:1244-1253.

[5] PNDS-parallel/distributed NS[EB/OL].(2006-10).http://www.cc.gatech.edu/computing/compass/pdns.

[6] Narses Simulator[EB/OL].(2006-10).http://sourceforge.net/projects/narses/.

[7] P2PSim[EB/OL].(2006-10).http://pdos.csail.mit.edu/p2psim/.

[8] Omnet++[EB/OL].(2006-10).http://www.omnetpp.org.

[9] Omnet++ P2P swarming protoco[EB/OL].(2006-10).http://me55enger.net/swarm/.

[10] Yang W,Abu-Ghazaleh N.GPS:a general peer-to-peer simulator and its use for modeling bittorrent [C]//MASCOTs'05,13th IEEE International Symposium,2005:425-432.

[11] 吴堃,戴菡叶,保留陆,等.基于 NS2 的 P2P 网络模拟平台研究[J].系统仿真学报,2006,8:2152-2157.