维普资讯 http://www.cqvip.com

BitTorrent 协议 Choking/Unchoking 机制的度量和分析*)

程 伟[1] 吴产乐[1] 雷迎春[1,2] 姜 琦[1] 阳立堂[1]

(武汉大学计算机学院 武汉 430072)[1] (中国科学院计算技术研究所 北京 100080)[2]

摘 要 BitTorrent是一个用于内容分发的P2P协议,现在已经发展成为互联网的一项重要的应用。本文从性能的角度,度量BitTorrent的行为,解释BitTorrent协议的关键元素,分析BitTorrent是否是高效的。本文有以下贡献:①提出一种有效度量BitTorrent式的内容分发协议的方法。②确认BitTorrent的Choking/Unchoking机制存在缺陷,不是高效的。③设计ShareStorm协议,证明BitTorrent的缺陷可以避免。经初步验证,在下载完成时间这个最主要的性能指标上,ShareStorm比BitTorrent至少减少50+%。

关键词 P2P,BitTorrent,Choking/Unchoking机制,性能,效率

Measurement and Analysis of Choking/Unchoking in BitTorrent

CHENG Wei[1] WU Chan-le[1] LEI Ying-chun[1,2] JIANG Qi[1] YANG Li-tang[1]

(School of Computer, Wuhan University, Wuhan 430072, China)[1]

(Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100080, China)[2]

Abstract BitTorrent is a P2P file distribution protocol, which has become an important Internet application in recent years. In this paper, we measure the performance of BitTorrent, explain its key factors, and analyze its efficiency. Our contributions are as follows: 1) present an effective way to measure the performance of BitTorrent; 2) find out some flaws of BitTorrent's Choking/Unchoking which impair its efficiency; 3) design a ShareStorm Protocol which avoids the flaws above. Our preliminary experiments show that in terms of download time, ShareStorm outperforms BitTorrent by 50+%.

Keywords P2P, BitTorrent, Choking/Unchoking, Performance, Efficiency

1 引言

BitTorrent已经发展成为一种主要的P2P应用,并逐渐改变着互联网的使用。据报道,在2004年初,全球互联网流量的35%出自BitTorrent的贡献;而在2004年末,这个数字几乎增加了一倍,且在某些地区(如亚洲),BitTorrent流量最多可以占据全部带宽的80%[1]。相比于其它P2P应用(如Gnutella,KaZaa,Skype),BitTorrent的长处是分发大文件,特别是在大量用户同时下载相同文件的场景中,即flash crowd。为处理flash crowd这种互联网所特有的用户访问模式,BitTorrent采用swarming技术,把文件划分成许多piece块,并使每一个piece块被独立共享分发。这样,BitTorrent客户端(下文称为peer)不必从文件的发布方下载该文件的所有piece块,而是当从BitTorrent网络中下载一个完整的piece块后,就在彼此之间进行块交换(注:块交换操作使得不同的piece块被分布在不同的peers上),并通过不断地交换不同的piece块,实现从不同的peers上并行下载,快速获取所需文件。

在BitTorrent中,下载相同文件的peers通过BitTorrent协议自组成一个独立的覆盖网络,并通过这个网络,共享彼此的CPU、网络和存储等资源。所以,BitTorrent是自扩展的,其容量随peers数量的增加而增加:每一个peer既是client peer,又是server peer,服务负载被自然地分布在所有参与peers上。由于大量使用peers之间的带宽资源,BitTorrent被认为是一种适合大规模内容分发的方法,它使内容发布者可以廉价地分发带宽密集型内容给大量的同时用户。

本文从性能的角度,度量BitTorrent的行为,解释BitTorrent协议的关键元素,分析BitTorrent的Choking/Unchoking是否是高效的。本文有以下贡献:①提出一种有效度量BitTorrent式的内容分发协议的方法。②确认BitTorrent的Choking/Unchoking不是高效的,且有以下缺陷:Choking/Unchoking中,tit-for-tat的竞争特性客观上制约着互连peers之间的协作,不利于BitTorrent系统的整体性能最优化。③设计ShareStorm协议,证明BitTorrent的缺陷可以避免。经初步验证,在下载完成时间这个最主要的性能指标上,ShareStorm比BitTorrent至少减少50+%。

2 BitTorrent协议

BitTorrent是一个用于内容分发的P2P协议[2]。不同于其它的P2P协议,BitTorrent不要求所有peers参与构建一个公共的P2P覆盖网络,并基于这个P2P网络共享不同的文件,而是为每个被共享的文件建立一个独立的BitTorrent传输会话。下载相同文件的peers隶属于同一个BitTorrent会话,且这些peers按BitTorrent协议自组织成一个独立的P2P覆盖网络,实现协作下载。这种设计使得BitTorrent协议本身不支持任何文件搜索的功能,它的目标是快速、高效且公平地把文件分发到大量用户。

在BitTorrent协议中,一个Torrent文件标识一个BitTorrent传输会话。Torrent文件中的内容包括被分发文件的元数据信息(如文件名、文件大小、piece块的大小等)、每一个

*)基金项目:国防科大并行与分布处理国家重点实验室基金项目(51484030105JW1203)资助。程 伟 博士研究生,主要研究方向为计算机网络与通信、网格计算;吴产乐 教授,博导,主要研究领域为新一代网络通信及网格计算;雷迎春 副研究员,主要研究领域为计算机网络与通信。

维普资讯 http://www.cqvip.com

piece块的SHA1哈希值、Tracker服务器的IP地址信息等等。BitTorrent会话由中心的Tracker服务器维护，一个Tracker服务器可以同时维护若干数目的BitTorrent会话。对一个BitTorrent会话而言，Tracker上不存储任何被分发的数据，Tracker的作用是跟踪下载该文件的所有BitTorrent客户端的活动。

peers使用BitTorrent协议的Neighbor Selection机制、Choking/Unchoking机制、Piece Selection算法来执行piece块交换操作，下载所需文件。

Choking/Unchoking机制决定peer A是否向peer B提供上载服务，可以拒绝服务(即Choking)，也可以允许服务(即Unchoking)。实际上，Choking/Unchoking机制决定两个相连的peers是否共享彼此的资源。在BitTorrent协议中，Choking/Unchoking机制的核心是tit-for-tat算法。peer A周期执行tit-for-tat算法，对自己从邻居peers上下载数据的速率(最近一个周期内的平均速率)进行排序，选出下载速率最快，且对peer A上的内容感兴趣的4个peers，然后对这4个peers执行Unchoking操作、对其余peers执行Choking操作，即peer A周期性评价令自己最受益的4个peers，并向这些peers提供服务。

3 试验方法

为精细并客观地度量BitTorrent的各种行为，同时支持下一代P2P内容分发协议的研究和开发，我们必须设计一个可控的试验环境。该试验环境需要具备以下属性：①无论是商业的BitTorrent系统、实验的BitTorrent系统，还是其它的文件分发系统，都可以在其上方便地运行；②试验环境的规模可灵活伸缩，即参与的结点数可以按照试验的需求而变化；③试验环境可重复，即试验环境能按照我们的需要保持不变；④试验环境可定制，即试验环境能按照我们的需要而变化；⑤试验环境必须能近似真实地反映现实世界的行为，如客户端的操作系统环境和互联网环境。

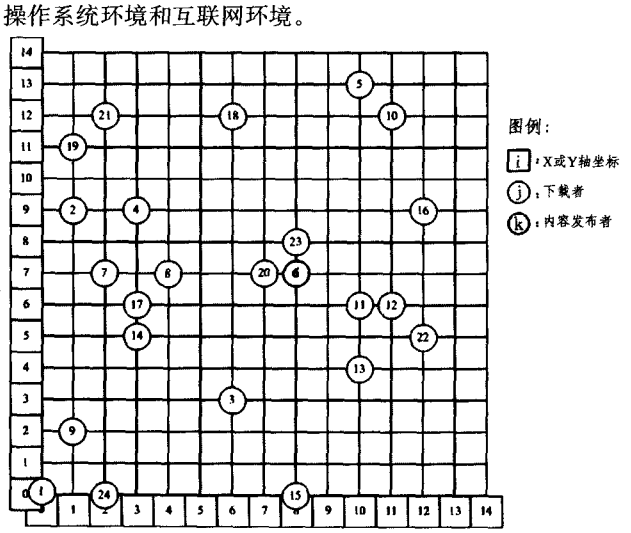

图1 网络拓扑图

3.1 可控的试验环境

对一个面向P2P应用的测试而言，可控的试验环境意味着我们能对参与的结点和连接结点的网络进行控制。由于要求支持任何一个商业或试验的BitTorrent式系统，且要真实反映被测试系统与客户端操作系统的交互，一个合适的选择是直接使用一台PC机来模拟一个参与结点。如果这样，那么接下来的问题就是如何把互联网的行为映射到连接PC的网络环境中。

表1 TCP传输吞吐率值与矩阵距离之间的对应关系

矩阵距离(d)	TCP的传输吞吐率(kB/s)
d=0	183.224
0<d≤1	106.023
1<d≤2	75.1365
2<d≤3	53.2975
3<d≤4	37.8566
4<d≤5	21.0028
5<d≤6	13.7747
6<d≤7	9.92721
7<d≤8	7.21612
d>8	5.3123

互联网的网络行为主要有带宽限制、拥塞、逐跳时延等等。从上层应用来看，这些网络行为将综合表现为TCP传输吞吐率。例如，如果端到端传输链路的带宽限制为1Mbps，又恰好有较多的拥塞，那么在应用层的表现就是TCP的平均传输吞吐率下降到低于1Mbps的某个值。因此，我们认为，只要能真实模拟结点之间的TCP传输吞吐率，就可以近似模拟复杂的互联网环境对BitTorrent式系统的传输影响，即近似反映被测试系统与互联网的交互。

已有的研究表明，TCP传输吞吐率和发送端与接收端之间的平均RTT值成反比[3]。直观地，端到端之间的传输距离长，平均RTT值大；距离短，平均RTT值小。这样，当我们把平均RTT值看作距离时，我们就有一个非正式的结论：TCP传输吞吐率和发送端与接收端之间的距离成反比。

因为任何两个结点之间的距离都可以表示为某个虚拟的距离，且为了试验的方便，我们使用矩阵来表示网络拓扑。如图1所示，矩阵上的每一个点表示一个可能的Site，每个Site中的结点个数可被任意地指定。显然，矩阵的大小可以任意地变化，且试验的规模决定矩阵的大小。

3.2 本文的试验环境

在本文所给出的测试中，能够使用的最多PC个数是96。如果我们规定一个Site中的结点数是4，那么共有24个Site需要分布在网络拓扑上。根据Site的数目，我们首先确定一个15*15的矩阵。然后我们使用随机的方法指定这24个Site在矩阵上的位置，如图1所示。

试验环境中，每台PC的基本配置是2.8GHz的Pentium-4、256M的内存、100Mbps的网卡、运行Windows XP SP2操作系统，PC机通过千兆交换机互连。虽然任一对PC机之间的数据传输能力可以达到100Mbps，不过，我们使用一个类似ModelNet[4]的带宽控制软件约束任何一对PC机之间的TCP传输吞吐率。

为试验的方便，我们设置每个PC的最大下载带宽为500kB/s，但是每个PC到其它PC的上载带宽和它们之间的矩阵距离有关，如表1所示。例如，Site_1(图2中左下角)中，同一个Site中的PC距离为0，所以，上载带宽设置为183.2kB/s(这也是系统中的最高值)；Site_1和Site_2之间由于矩阵距离较远，因此上载带宽设置为5.3kB/s；相对于Site_1和Site_2，Site_1和Site_3之间的矩阵距离较近，所以上载带宽为9.9kB/s。

表1中的数据是这样确定的：我们首先在实际的互联网中随机选择大约50个不同地理位置，不同ISP中的主机测试彼此的端到端RTT值，以及相应的数据传输值。然后，基于文献[3]中的公式，计算采样RTT值与理论TCP传输吞吐率的关系。最后，基于实际采样的数据传输值，对理论TCP传输吞吐率值进行折中，才得到表1中的数据。

维普资讯 http://www.cqvip.com

3.3 评价对象及评价指标

在本文所给出的测试中，我们选用的 BitTorrent 客户端程序是 BitComet-0.64。BitComet[5] 是 BitTorrent 协议的 C++实现，它是一种流行的、面向商业应用的 BitTorrent 客户端。我们认为，与 BitComet 的直接比较，会让比较数据更加客观、真实。BitComet 的配置选项使用缺省值，除以下选项外，最大上载连接数设置为 5，最多选择 15 个 peers 作为邻居。

我们用链路利用率作为评价 BitTorrent 效率的主要性能指标。在我们的试验环境中，一个 PC 到另一个 PC 的上载带宽是独享的，且设置的上载带宽值与相应的距离有关。按测试环境设置，每个 PC 有 15 个邻居，所以每个 PC 的上载带宽值等于该 PC 到所有邻居的上载带宽的总和。这样，每个 PC 的总上载带宽值很有可能超过 500kB/s，且彼此不同。不过，每个 PC 的下载带宽是共享的，其值都为 500kB/s。由于下载带宽比上载带宽更易成为瓶颈，且具有一致性，因此我们选用下载链路的利用率作为每个 peer 的链路利用率。

试验中，我们模拟 flash crowd 场景，即所有 peers 在 10s 内加入到会话，且除原始种子外，每个 peer 在下载完内容后，立即离开会话。在这种情况下，每个 peer 的下载链路利用率可计算为((文件大小/该 peer 的下载时间)/peer 的下载带宽)。其中，文件大小和 peer 的下载带宽为固定值。

4 评价

我们全新设计了一个称为 ShareStorm 的 P2P 内容分发协议。ShareStorm 在功能上类似于 BitTorrent，也是运行在用户端的 PC 机上，不需要改变 Internet 基础结构，而直接利用现有的 Internet 资源实现内容分发。相比于 BitTorrent，ShareStorm 具有以下特性：①Efficient。ShareStorm 显著地降低了内容被分发的时间，使最终用户具有更好的下载体验。②ISP-friendly。ShareStorm 显著地提高了内容被分发的效率，大幅度降低 Internet 运营商的运营成本，且 ShareStorm 允许 Internet 运营商动态调整指定区域的流量。本文的目标是证明 BitTorrent 不是高效的，所以我们不在本文描述 ShareStorm 的细节，而是以 ShareStorm 为参照，说明 BitTorrent 存在哪些影响系统性能的缺陷。

我们基于完全相同的硬件和软件环境，在上文所述的 flash crowd 场景中，使 peers 先后执行 BitTorrent 和 ShareStorm 协议，把一个 130MB 的文件从一个种子(位于图 1 中的 Site_6)分发给其他所有 peers。为限制每轮测试的时间长度不至于太长，我们设置了一个最长会话时间。BitTorrent 由于效率较低，平均有 3%的 peers 不能在最长会话时间内下载完内容，而 ShareStorm 不会有这种情况的发生。另外，由于整个测试活动长达两个多月(因为我们为 ShareStorm 的每个关键机制都设计了多种算法，在测试前我们并不知道哪种算法会更好)，我们经常发现少部分 PC 因为硬件故障不能在某天正常工作(注：每轮测试至少有 80 台 PC 参与)，因此，这些故障机器的测试数据有可能就没有记录。

4.1 性能

在我们给出的测试中，图 2、图 3 和表 2 描述了 BitTorrent 和 ShareStorm 的性能对比。图 2 的 X 轴是递增的会话时间，Y 轴是完成 peers 占整个 peers 数目的百分比。在图 2 中：①当会话时间为 1000s 时，若 peers 使用 BitTorrent 协议，则只有 10%的 peers 完成下载；若使用 ShareStorm 协议，则有 71%的 peers 完成下载。②当会话时间为 1352s 时，若使用 ShareStorm 协议，则 100%的 peers 完成下载；若 peers 使用 BitTorrent 协议，则只有 54%的 peers 完成下载。

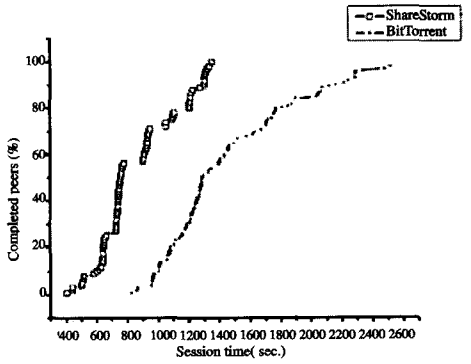

图 2 peers 的下载完成时间的累积分布

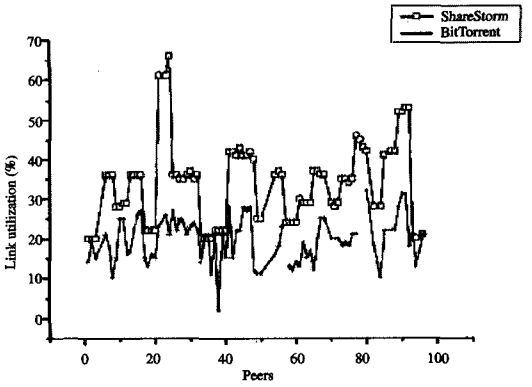

图 3 peers 的链路利用率

表 2 ShareStorm 与 BitTorrent 的平均下载完成时间的比较

累计完成的 peers 个数	ShareStorm(单位：秒)	BitTorrent(单位：秒)	加速比
5	399.4	935.6	0.573
25	563.44	1215.4	0.536
45	630.8444	1374.866667	0.541
65	708.6154	1537.446154	0.539
85	797.6627	1715.166667	0.535

在表 2 中，第一行，取完成下载时间最少的前 5 台，ShareStorm 的平均下载时间是 399.4s，BitTorrent 的平均下载时间为 935.6s；ShareStorm 相对于 BitTorrent，平均下载时间减少了 57.3%。第二行，取完成下载时间最少的前 25 台，ShareStorm 的平均下载时间是 563.44s，BitTorrent 的平均下载时间为 1215.4s；ShareStorm 相对于 BitTorrent，平均下载时间减少了 53.6%。因此，由表 2 的数据我们可以获知，在下载完成时间这个最主要的性能指标上，ShareStorm 比 BitTorrent 至少减少 50+%。

图 3 的 X 轴是按 peerID 递增的 peers，Y 轴是每个 peer 分别执行 ShareStorm 和 BitTorrent 协议时的平均链路利用率。BitTorrent 正是由于自身存在缺陷，使其不能充分使用网络资源，因此平均链路利用率才比 ShareStorm 低一半左右。同时，由下载链路利用率的计算公式我们也可以得知，每个 peer 的下载时间与下载链路利用率成反比。如果想缩短下载时间，必须提高下载链路利用率。

4.2 Choking/Unchoking 机制

本节从微观角度量化 Choking/Unchoking 机制对 BitTorrent 效率的影响。图 4 描述当分别执行 BitTorrent 和 ShareStorm 协议时每个 peer 的下载完成时间。图 4 中的 X 轴是按 peerID 递增的 peers，Y 轴表示 peers 的下载完成时间。由于属于同一个 Site 的 peers，其 peerID 是相邻的，因此

维普资讯 http://www.cqvip.com

我们观察到：①对于 BitTorrent，同属一个 Site 的 peers 的下载完成时间是相异的，而不是一致性的；②对于 ShareStorm，同属一个 Site 的 peers 在下载完成时间上具有较好的一致性。进一步地，我们用图 5 分别表示同一个 Site 中的 peers 在分别执行 BitTorrent 和 ShareStorm 时最快下载 peer 和最慢下载 peer 的完成时间之差。其中，图 5 中的 X 轴是 Site 的序号。从图 5 我们可以了解到，当执行 ShareStorm 时，每个 Site 的完成时间差值都很小，平均为 23.16s。而当执行 BitTorrent 时，仅有少数 Sites 的差值比较下（如 Site13 和 Site22），大部分 Sites 的完成时间差值都比较大，平均为 560.33s。

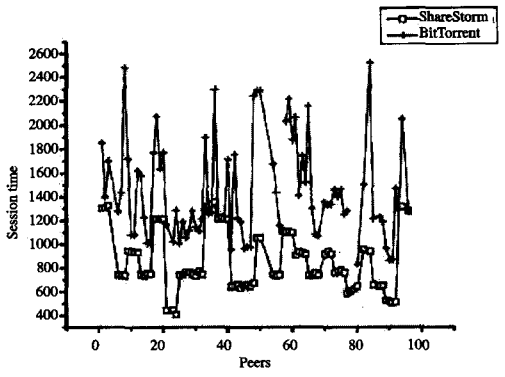

图 4　peers 的下载完成时间

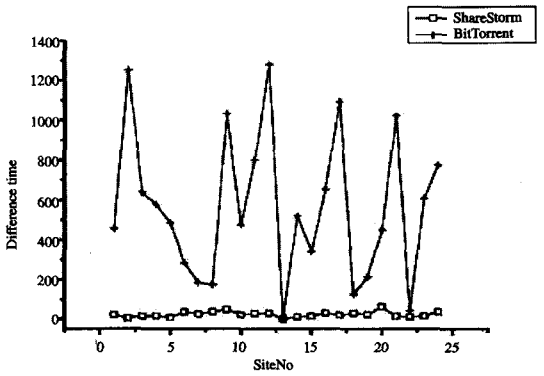

图 5　同一 site 中最快下载和最慢下载 peers 的完成时间差

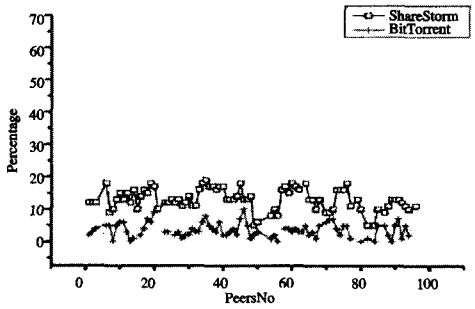

图 6　Set(0)的邻居分布

我们使用图 6 和图 7 来进一步描述上述现象。为下文叙述方便，我们首先给出 Set(i)的含义。给定一个 peer A，我们把 peer A 的所有邻居 peers 按相对于 peer A 的距离（这里的距离指矩阵距离）远近分成不同集合，并用 Set(i)表示。当 $i>0$ 时，Set(i)的含义是该集合中的元素与 peer A 的距离大于 $i-1$、小于等于 i。如 Set(1)表示该集合中的元素与 peer A 的距离大于 0、小于等于 1，Set(2)表示该集合中的元素与 peer A 的距离大于 1、小于等于 2。对于特殊的 Set(0)，Set(0)的含义是该集合中的元素与 peer A 的距离差等于 0。

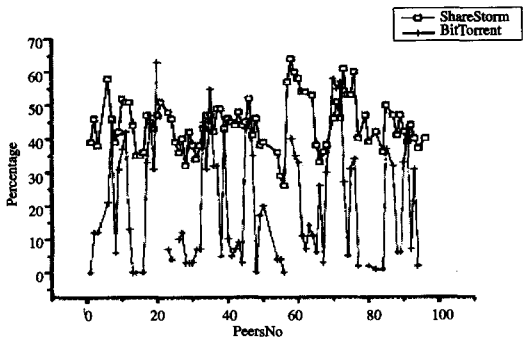

图 7　Set(0)的通信事件分布

图 6 描述当 peers 分别执行 BitTorrent 和 ShareStorm 时，每个 peer 的邻居属于 Set(0)的分布，即属于 Set(0)的邻居个数占所有邻居个数的百分比。图 7 描述当 peers 分别执行 BitTorrent 和 ShareStorm 时，每个 peer 发生在 Set(0)上的通信事件分布，即属于 Set(0)上的通信事件占所有通信事件的百分比。这里，一次通信事件是指本 peer 从其它 peer 下载一个数据块，或本 peer 向其他 peer 上载一个数据块。

观察图 6,7，当 peers 执行 ShareStorm 时，所有 peers 的 Set(0)通信事件比例的平均值为 44.37%，其中，peerID 为 56 的 peer，它的 Set(0)通信事件比例最低（26%）；peerID 为 58 的 peer，它的 Set(0)通信事件比例最高（64%）。而当 peers 执行 BitTorrent 时，所有 peers 的 Set(0)通信事件比例的平均值只有 20.24%，其中，peerID 为 56 的 peer，它的 Set(0)通信事件比例为 0；peerID 为 20 的 peer，它的 Set(0)通信事件比例最高（63%）。上述数据说明 BitTorrent 并不能很好地促进同一个 Site 内的 peers 之间的协作，BitTorrent 测试中，虽然部分 peers 的 Set(0)通信事件的比例较高（如 peerID 为 20 的 peer），但很多 peers 的 Set(0)通信事件的比例相对较低，约 44%的 peers 的 Set(0)通信事件比例低于 10%。

这个现象给我们的启示是，尽管两个 peers 之间有很好的通信带宽，但 BitTorrent 并不促进两个 peers 之间的协作。其原因，一方面是由于 BitTorrent 的 Neighbor Selection 使得 peers 建立 Set(0)连接的比例相对少，另一方面是因为 BitTorrent 的 Choking/Unchoking 机制不能很好地促进属于相同 Site 的 peers 之间的协作。

4.2.1　讨论

Choking/Unchoking 机制决定两个相连的 peers 是否协作。我们认为，该机制是 BitTorrent 系统是否正常运行的保障，同时它也是 BitTorrent 明显区别于其它文件分发系统，从而能经历残酷的市场选择而茁壮成长的重要因素。

Choking/Unchoking 机制由 tit-for-tat 算法和 optimistic unchoking 算法组成，其中核心是 tit-for-tat 算法。tit-for-tat 是一种简单、有效的激励机制，它促进 peers 之间的公平性，使每个 peer 的下载量和上载量是相关的，即上载带宽大的 peers，其下载带宽也相应大。因此，tit-for-tat 算法使 BitTorrent 有效地避免了 free-rider 问题[6]。free-rider 是自私的 peers，它们不对外或很少对外共享自己的资源，却拥有较高的下载速率。

事实证明，free-rider 的大量存在会破坏系统的健康运转，而 tit-for-tat 式的协作使 BitTorrent 系统成功地克服了 free-rider 问题。那些乐观认为 peers 是无私的，而没有使用任何激励机制的 P2P 系统，大多因为 free-rider 问题，只能是

维普资讯 http://www.cqvip.com

试验系统，不能被广泛应用。而激励机制为 credit/reputation 的系统，如 eMule，已被证明过于复杂和理想化[7]。

tit-for-tat 机制惩罚性地使 free-rider 得不到很好的下载服务，促使 peers 之间的资源共享。具体来说，tit-for-tat 机制使本 peer 优先向一些 peers 提供服务，因为这些 peers 向本 peer 提供的服务要多于其它 peers，或者说这些 peers 更乐于与本 peer 协作。在这里，服务的多少表现为，在指定的时间段内，本 peer 从这些 peers 上获得的数据要多于从其它 peers 上获得的数据。即，在 tit-for-tat 的作用下，本 peer 优先与那些正在给自己带来显著益处的 peers 进行协作，而不考虑自私的 peers。

tit-for-tat 算法的每次执行意味着一个新周期的开始，因此，tit-for-tat 对 peers 的评估也是周期性的，且评估所基于的速率是以周期为界的，是瞬时的。前一个周期，peer A 选择给自己带来最多益处的 4 个 peers 执行 Unchoking 操作，并不意味着下一个周期仍然是对这 4 个 peers 执行 Unchoking 操作，它们随时会被其它更有竞争力的 peers 所替代。这种优胜劣汰的驱动力使 BitTorrent 能很好地规避 peers 的自私行为，并强迫会话中的 peers 积极贡献自己的资源（包括带宽、CPU 和存储），努力寻求与其它 peers 的协作。

严格的 tit-for-tat 算法不利于新 peers 参与到会话中来，因为新 peers 上没有任何 piece 块，无法与其它老 peers 进行 piece 块的交换。为处理这种情况，BitTorrent 使用 optimistic unchoking 算法。optimistic unchoking 随机选择一个邻居 peer，而不考虑这个邻居 peer 对本 peer 是否有益。optimistic unchoking 的另一个用处是寻找具有更好上载能力的邻居。换言之，optimistic unchoking 的随机选择的特性能帮助本 peer 评估按 tit-for-tat 算法选中的那 4 个 peers 是否是最优的。因为，相比于 tit-for-tat 算法选择的 4 个 peers，optimistic unchoking 选择的那个 peer 有可能对本 peer 更有益。

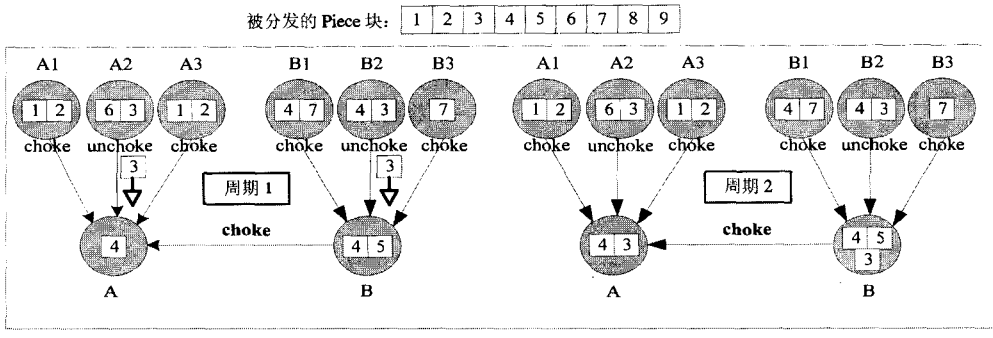

图 8　tit-for-tat 示例

虽然 tit-for-tat 算法使 BitTorrent 不再脆弱，但也有它的局限性。如为避免 free-rider 问题而引入的竞争机制，客观上制约着 peers 之间的协作。如图 8 中，peer A 的邻居有 A1，A2，A3 和 B，peer B 的邻居有 B1，B2，B3 和 A。在周期 1，A2 unchoking A，B2 unchoking B，所以 A 可以从 A2 下载 piece 块，B 可以从 B2 下载 piece 块。A 和 B 在执行 LRF 算法后，均选择 piece 3 下载。在周期 2，B 用 tit-for-tat 算法评估哪些 peers 对自己有益，A 因为其上的内容是 B 所不关心的，首先就被排除在外，即使 A 和 B 就在同一个局域网上。

从以上的简单例子我们可以看出，tit-for-tat 制约了 BitTorrent 的性能，每个 peer 以最大化自己所能获得的利益为目标，而不是考虑如何协作使得 BitTorrent 的整体性能最优。optimistic unchoking 因为随机性和较长的执行间隔，不足以缓解这个问题。

从上文的讨论来看，我们认为有必要设计新的 Choking/Unchoking 机制来避免现有 tit-for-tat 的局限性。在新的机制中，仍然要以 tit-for-tat 思想为骨架，不过要折中 tit-for-tat 的利益至上的特性，即要充分考虑其它一些辅助信息来促进 peers 之间的协作，使整个系统的性能接近最优。

需要注意的是，抛弃 tit-for-tat 机制，固然会给我们一个更加广阔的空间，设计出比 BitTorrent 性能更优的 peers 协作机制。但是，从过去的经验来看，这样的系统只能是试验系统，它只能证明文件分发系统的最优性能是怎样的，而不具备实际的可操作性。

5　相关研究

作为一个 P2P 内容分发协议，BitTorrent 已经成为互联网的一项重要应用，并改变着人们消费内容的习惯。许多研究团体和组织都在研究 BitTorrent 现象。以文献[8,9]为代表的研究团体使用模拟器或数学模型来分析 BitTorrent 的效率。这些文献假设连接 peers 的网络是理想或近似理想的，在这种前提下证明 BitTorrent 是高效的。我们认为这些文献的结论只是给出了 BitTorrent 性能的理论上限，并不能很好地预测 BitTorrent 系统的实际性能。

文献[10]用网络编码技术实现 P2P 内容分发，并基于模拟器测试，说明基于网络编码的系统要比 BitTorrent 系统的性能好 20%～30%。不过，我们研究网络编码技术的一点体会是，如果没有好的 Neighbor Selection，Choking/Unchoking 机制的支持，将很难在互联网环境中充分发挥网络编码技术的长处。我们认为这也是文献[10]的系统迟迟没在互联网上发布并被应用的重要原因。目前，我们正在实现一个采用网络编码技术的原型系统，并将结合到 ShareStorm 中。

文献[11,12]提出了类似 BitTorrent 的系统。这些系统与 BitTorrent 的显著区别是没有采用类似 TFT 的机制，而是要求所有 peers 是无私协作的。事实证明，这样的系统将不可避免地遭受到 free-rider 问题的困扰，从而破坏系统的健康运转，最终导致系统被人遗忘。ShareStorm 正是对这一现象的充分认识，因此其所有核心机制都处于 TFT 的框架之下。

除纯 P2P 内容分发系统外，还存在一些需要互联网基础结构支持的 P2P 内容分发系统，如文献[13,14]。这些系统一般要求在互联网中部署 Cache 服务器或 Index 服务器等设备，以辅助 P2P 的内容分发。这类系统的共同问题是：1）出现新的访问瓶颈，使系统新增可扩展性问题；2）成本问题；3）服务部署问题。

维普资讯 http://www.cqvip.com

结束语 本文从性能的角度，度量 BitTorrent 的行为，证明 BitTorrent 不是高效的，给出并分析 Choking/Unchoking 机制对 BitTorrent 效率的影响。在证明 BitTorrent 不是高效的过程中，我们设计了一种接近于互联网行为的、专门用于度量 P2P 内容分发协议的新方法，且全新设计了一个称为 ShareStorm 的协议，作为参照对象。通过我们提出的度量方法，我们证明 Share Storm 比 BITTorrent 具有更好的效率。

参 考 文 献

[1] ISP's battle against encrypted BitTorrent downloading. http://www.afterdawn.com/news/archive/7896.cfm

[2] Cohen B. Incentives build robustness in BitTorrent // Proceedings of the First Workshop on the Economics of Peer-to-Peer Systems. Berkeley, CA, June 2003

[3] Akan O B. On the Throughput Analysis of Rate-based and Window-based Congestion Control Schemes. Computer Networks Journal (Elsevier Science), 2003

[4] Vahdat A, Yocum K, Walsh K, et al. Scalability and Accuracy in a Large-scale Network Emulator // Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI). December 2002

[5] BitComet. http://www.bitcomet.com/

[6] Adar E, Huberman B A. Freeriding on Gnutella. First Monday, 2000, 5(10)

[7] Anagnostakis K G, Greenwald M B. Exchange-based incentive m-echanismsfor peer-to-peer file sharing // Proc. of IEEE Icdcs (March 2004)

[8] Bharambe A, Herley C, Padmanabhan V. Analyzing and improvingbittorrent performance // Proc. of IEEE Infocom. 2006

[9] Qiu D, Srikant R. Modeling and performance analysis of bittorrentlike peer-to-peer networks // Proc. ACM SIGCOMM'04. Portland, Oregon, USA, Aug. 30—Sept. 3, 2004

[10] Gkantsidis C, Rodriguez P. Network Coding for Large Scale ContentDistribution // IEEE INFOCOM 2005, Miami. March 2005

[11] Cherkasova L, Lee J. Fastreplica: Efficient large file distribution within content delivery networks // USENIX Symposium on Internet Technologies and Systems. 2003

[12] Levin D, Sherwood R, Bhattacharjee B. Fair File Swarming with FOX // International Workshop on Peer-to-Peer Systems (IPTPS). 2006

[13] Chun B G, Wu P, Weatherspoon H, et al. ChunkCast: An AnycastService for Large Scale Content Distribution. In IPTPS, 2006

[14] Park K, Pai V S. Scale and performance in the coblitz large-file distribution service. In NSDI, 2006

(上接第 68 页)

显然就存在(原来可能是合法信息流)，所以，这一类非法信息流只可能由原来的合法信息流或非法信息流转化而来，由合法信息流转化来的由步骤(3)生成，由非法信息流转化来的由步骤(5)生成。

注意：算法 2 省略了对合法信息流和每个 E_i，F_i 的更新计算，它们是非法信息流进一步更新的前提。

如果系统中用户的变化是删除用户，此时系统中信息流显然不可能增加，且原来合法的信息流不会转化为非法的信息流。非法信息流的更新相对简单。只需将被删除用户从原来非法的用户信息流的引发用户集和威胁用户集中剔除，剔除后仍非法的信息流和没有经过剔除的非法信息流构成了新的非法信息流集合。

如果在系统中添加新角色并赋予已有的用户，这相当于给系统中某些用户增加权限，系统中信息流一般会增加，并且，原来合法与非法的信息流都可能变为相反的信息流。此时，最简单也是最直接的信息流更新方法是，首先更新每个客体 o_i 的 E_i 和 F_i，然后由所有的 E_i 和 F_i 直接生成新的非法与合法信息流。

4 非法信息流的控制

RBAC 系统中的非法信息流可能导致信息的非授权泄漏，为保证客体信息的安全性，需要对这些非法信息流进行控制。控制非法信息流实际是控制非法信息流引发用户的"写"操作，这与用户的访问历史有关，是动态的。下面给出一个非法信息流控制算法。算法中假定用户的访问请求已满足 RBAC 系统基于角色的授权约束，即用户 u 与其请求权限 (op,o) 满足：$(u,(op,o))\in UP$。

输入：用户 u，u 请求的权限 (op,o)，系统非法信息流集合 H；

输出：授权访问或拒绝访问。

算法 3

(1)对每个非法信息流 $(i,j,X,Y)\in H$，设置访问历史记载项 $\tau(o_i,o_j)$，初始化 $\tau(o_i,o_j)=\phi$。

(2)若 (op,o) 满足 $\lambda((op,o))=-1$，且 $o=o_i$，则将 u 添加到每个 $\tau(o_i,o_j)$ 中，并对 u 的访问授权。

(3)若 (op,o) 满足 $\lambda((op,o))=1$，则检查每个 $\tau(o_i,o_j)$，如果存在某个 $\tau(o_i,o_j)$ 满足 $o_j=o$ 且 $u\in\tau(o_i,o_j)$，则拒绝 u 的访问请求。否则对 u 的访问授权。

(4)若 (op,o) 是其它类型的权限，都对 u 的访问授权。

算法为每个非法信息流 (i,j,X,Y) 生成一个历史记载项 $\tau(o_i,o_j)$。用于记载 X 中已执行过使信息流出 o_i 的操作的用户。这些用户被记载后请求的权限若能使信息流入 o_j 时，将拒绝请求，其它情况的访问请求都会获得授权。

结束语 RBAC 模型的访问授权基于的是用户角色所承担的职责，而不是对信息流的约束。因此，用户的授权访问可能引发不安全的信息流，从而导致信息的非授权泄漏。为保证 RBAC 模型系统的信息流安全性，在定义了非法的用户信息流概念的基础上，重点研究了非法信息流的检测、更新以及控制问题，给出了具体的检测、更新和控制算法。将这些算法纳入到 RBAC 模型系统的授权管理中可有效杜绝系统中非法信息流的发生，实现访问控制对信息机密性的保护。

参 考 文 献

[1] Sandhu R, Coyne E, Feinstein H et al. Role-based access control model [J]. IEEE comprter, 1996, 29(2): 38-47

[2] Ferraiolo D F, Sandhu R S, Gavrila S, et al. Proposed NIST standardfor role-based access control [J]. ACM transactions on information and system security, 2001, 3(4): 182-186

[3] Al kahtani M, Sandhu R. Induced role hierarchies with attribute-based RBAC [C] // proceedings of the 8th ACM symposium on access control models and technologies. Villa Gallia, 2003: 142-148

[4] Bell D E, LaPadula L J. Secure computer system: Unified eposition and MUL TICS interpretation. The MITRE Corporation, Technical Report: MTR-2997 Revision1. 1976

[5] 李斓，冯登国，徐震. RBAC 与 MAC 在多级关系数据库中的综合模型. 电子学报，2004，32(10)：1635-1639

[6] Izaki k, Tanaka K, Tskizawa M. Information flow control in role-based model for distributed objects // Proc of the 8th Int'l Conf. on Parallel and Distributed Systems. Los Alamitos, CA: IEEE Computer Society Press, 2001: 363-370