P2P 系统的性能优化:研究综述*)

陈小明[1] 李舟军[2]

(国防科技大学计算机学院 长沙 410073)[1] (北京航空航天大学计算机学院 北京 100083)[2]

摘 要 Peer-to-Peer (P2P) 计算模型在利用和管理日益增长的分布式信息和资源方面是一种非常成功的计算模型。但是这种计算模式也有一些缺点,特别是 decentralized unstructured P2P 系统,它将随机选择邻居和 blind flooding search 机制联合起来使用,使系统的网络开销急速地增加,严重地影响了系统的性能。本文介绍了减少 decentralized unstructured P2P 系统网络开销的各类方法,对它们的优缺点进行了详细的比较,并根据比较的结果对今后的发展趋势进行了展望。

关键词 P2P 计算,网络开销,缓存,性能优化

A Survey: The Performance Optimization in P2P System

CHEN Xiao-Ming[1] LI Zhou-Jun[2]

(School of Computer, National University of Defence Technology, Changsha 410073)[1]

(School of Computer Science & Engineering, Beihang University, Beijing 100083)[2]

Abstract Peer-to-Peer (P2P) computing model is a successful computing model in utilizing and managing the growing information and resources that are distributed over the Internet. However, this computing model has some limitations. Especially, decentralized unstructured P2P systems uses randomly choosing neighbor and blind flooding search in combination, which dramatically aggravates the network overhead thus seriously affects the performance of such systems. In this paper we introduce and compare various types of existing methods for alleviating the network overhead of decentralized unstructured P2P systems. Based on the comparison, we predict the directions of the researching in the performance optimization of decentralized unstructured P2P systems.

Keywords P2P computing, Network overhead, Buffer, Performance optimization

1 前言

Peer-to-Peer(P2P) 计算模式作为一种共享和管理网络中海量信息资源的分布式计算模式,近年来已经越来越多的受到人们的关注。Peer-to-Peer 最主要的思想是所有的节点之间的地位关系完全是对等的,没有 client 和 server 之分,也就是说每个节点既做 client,也做 server,既向别人提供服务,也从别人那里获取服务。实际上,Peer-to-Peer 系统的最原始的目的是文件共享,例如:Napster[1] 和 Gnutella[2] 就是节点先集中可共享的文件,然后发布出来供用户使用。这种模式使得位于网络边缘的用户可以共享和访问这些文件,这也是自 1999 年 Napster[1] 出现以来,P2P 计算模式变得异常火热的最主要原因。虽然这种计算模式最初只是应用于文件共享,但是研究者们很快意识到了 Peer-to-Peer 计算模式在别的方面的应用潜力,所以在后来的一段时期内 Peer-to-Peer 计算模式很快被应用到各种分布式应用中,例如:共享 CPU 资源、分布式存储、分布协作环境。因而在 P2P 的第一轮研究浪潮中主要是研究 Peer-to-Peer 计算模式在不同的领域和应用中可能带来的好处,而现在正在进行的第二轮研究浪潮主要是关注怎样增强 P2P 应用的服务质量。

现在 Peer-to-Peer 系统有三种不同的体系结构,分别是:centralized、decentralized structured 和 decentralized unstructured[3]。Centralized P2P 系统由两类节点构成,一类是一般的节点;另一类是具有超级计算能力的节点,它能起到协调和控制系统中第一类 peer 在系统中的行为的作用。例如:Napster 就是这样的 P2P 系统,虽然在系统中节点之间可以直接交换文件,但是在系统中必须有中央服务器为所有的节点来维护系统中所有共享文件的地址索引信息。然而这种体系结构的系统会引起单点失效,并且容易受到拒绝服务(Dos)攻击。decentralized structured 和 decentralized unstructured P2P 系统就是由一组节点构成,这些节点叫做 peer,它们之间相互直接进行文件共享和交换,不需要任何中央协调节点。decentralized structured P2P 系统是基于分布式哈希表的,所以共享数据的位置和 overlay 拓扑有紧密的关系,例如:Chord[4]、Pastry[5]、Tapestry[6] 和 CAN[7]。

实际上,现在 decentralized unstructured P2P 系统使用得非常多,在这种结构的系统中文件所在的位置是随机的,与 overlay 拓扑结构没有任何关系。所以 decentralized unstructured P2P 系统都使用 blind flooding 机制来查找系统中可以使用的资源,例如:Gnutella 和 KaZaA[11]。Blind flooding 机制就是广播一条查询消息出去直到满足系统规范的标准,如生命周期(TTL)结束或者是找到目标资源。当某个 peer 收到这条消息并且发现自己能够提供这种资源时,就发送一条响应消息,按照原路返回给发出查询消息的源节点。其实这

*)国家自然科学基金项目(90104026,60473057)。**陈小明** 硕士研究生,主要研究方向为 P2P 计算及其性能优化;**李舟军** 博士、教授、博士生导师,主要研究方向为进程代数理论、安全协议的形式化验证、Web 服务与 P2P 计算。

种 blind flooding 机制主要是为了确保查询消息能够在短时间内发送给尽量多的 peer，以便能找到所需要的资源。这同时也带来一个严重的问题——消耗网络带宽。

文[12]对 Gnutella 这种 decentralized unstructured P2P 系统进行了 7 个多月的追踪测试，发现随着系统的规模的不断扩大，最终 blind flooding 机制产生的消息量占整个系统产生的消息量的 91%；当 Gnutella 只有 50000 个节点并且其中 95%的节点所产生消息的 TTL 小于 7 时，每个月产生的消息量 330TB。2001 年以来出现了很多这一方面的测试，例如在文[8]和[9]的研究中都发现 P2P 系统的通讯量占整个 Internet 通讯量的很大一部分，文[10]对校园网的测试研究中发现，P2P 系统消耗了整个带宽的 43%，而 WWW 只消耗整个带宽的 14%。所以我们可以看出 P2P 系统大量消耗网络资源是一个日益严重的问题，本文试图对减少网络开销以优化 decentralized unstructured P2P 系统的性能方面的研究现状进行总体综述。

2 P2P 系统性能优化的研究

现在已经有很多研究试图减少基于 blind flooding 查询算法的 decentralized unstructured P2P 系统中的冗余消息。一般来说，现存的方法可以分为三类：forwarding-based、cache-based、overlay optimization。

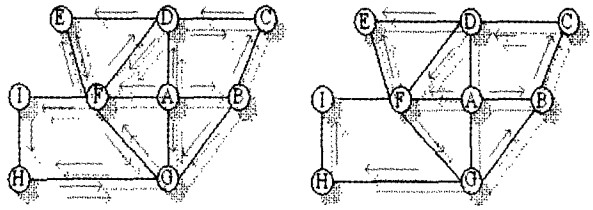
2.1 forwarding-based 类方法

在 forwarding-based 类方法中，peer 只须在这些逻辑邻居中选择一个子集转发就可以了，而不像原来的 blind flooding 那样，peer 必须向除作为消息来源的逻辑邻居以外的所有其它逻辑邻居转发它收到的查询消息。在这类方法中最具有代表性的有两种：Directed BFS[13]、混合周期泛洪算法(HPF)[14]。

Directed BFS 是较早被用来减少网络开销以提高查询效率的方法。它的主要思想是：每个 peer 转播查询消息时只选择逻辑邻居中它认为能以最快的速度返回结果的一个最小子集，其中通过什么样的方法选择这个子集是关键，因为它直接影响着查询的效率和网络开销。在 Directed BFS 中为了让 peer 的选择更加合理，每个 peer 都维护着所有逻辑邻居上的某种信息，例如以前收到的查询结果或者是它与邻居间的网络延迟，通过这些信息 peer 可以得到一些启发性的选择逻辑邻居子集的规则，一般包括以下几种：一、选择那些在以前的查询过程中返回结果最多的逻辑邻居；二、选择逻辑邻居子集，这些逻辑邻居返回的响应信息从响应源节点到这个节点的跳步数最少；三、选择与它连接时间比较长的节点；四、选择那些拥有的消息队列比较短的节点。通过 Directed BFS 可以急剧的减少接受这条查询消息的节点，从而达到减少网络开销的目的。

HPF 是所有 forwarding-based 类方法中性能最好的方法，它减少的网络开销比 Directed BFS 高 20%左右。HPF 方法由两部分组成，分别是：peer 所维护的关于逻辑邻居的各种信息和周期泛洪(PF)。第一部分决定 peer 转发查询信息时选择邻居的标准，第二部分决定要选择的邻居的数量。在 HPF 方法中 peer 所维护的信息有多种，而不像 Directed BFS 中所维护的那样只有一种。这样在 HPF 中 peer 选择邻居是同时综合多重因素进行计算得到的，使选择更加合理。PF 使用了一个周期函数控制 peer 每次选择要转发查询消息的邻居的个数。这个函数的形式是：$h=f(n, TTL)$，h 表示要选择的逻辑邻居的个数，n 表示 peer 现在的逻辑邻居数，TTL 表示查询消息的生命周期。由于这是一个周期函数，所以这种查询机制被分成几个阶段，在整个查询过程中不断重复这些阶段，函数的周期就是这些不同阶段的个数。对于同一个 overlay 拓扑，不同的周期可能会得到不同的性能，所以周期的选择也很关键。下面我们说明一个周期为 2 的 PF 函数的例子：

$$f(n,TTL)=\begin{cases}\left\lceil \frac{1}{2}n \right\rceil & \text{if TTL is odd}\\ \left\lceil \frac{1}{3}n \right\rceil & \text{if TTL is even}\end{cases}$$



(a) blind flooding　　(b) PF

图 1　blind flooding 和 PF 的比较

我们用两幅图来说明在这个函数下 PF 的工作过程。在 blind flooding 中每一个 peer 都必须向除作为消息来源的逻辑邻居以外的所有其它逻辑邻居转发这则查询消息。而在 PF 中则不同，我们假设每一条查询消息的初始 TTL 是七，则在图 1(b)中 peer A 只能选择两个邻居作为它的查询消息的转发节点(因为 A 只有四个邻居，并且 TTL=7 是奇数)。A 选择 F 和 B，消息分别到达 F 和 B 时，消息的 TTL 变为 6，是偶数，所以 F 只能选择两个邻居转发这条消息，而 B 只能选择一个邻居转发，依次这样下去直到 TTL 等于零或者找到目标资源。

2.2 cache-based 类方法

cache-based 也是一类优化 P2P 系统网络开销非常有效的方法。现在存在的缓存策略有两种：一种是中央服务器缓存形式，另一种是本地缓存形式。不管是哪一种形式都可以缓存数据索引或者数据本身。中央服务器缓存方式主要是用于 centralized P2P 系统，这台中央服务器提供所有 peer 上可共享文件的索引。也有一种 decentralized unstructured P2P 系统——KaZaA 使用这种缓存策略，但是它做了以下改变：只让每一个索引服务器缓存少量的 peer 上的共享文件索引，这样这种服务器可以由计算能力较强一点的 peer 来承担。

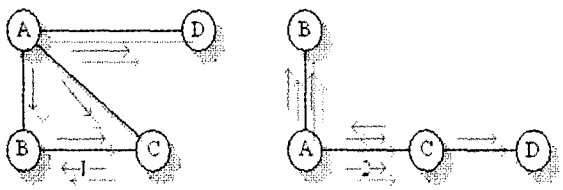
本地缓存是 decentralized unstructured P2P 系统最常用的一种方式，其主要思想是每个 peer 都缓存少量的相关的信息。在文[13]中就提出了一种本地缓存的方法——Local Indices。这种方法的核心思想是：每个 peer 都必须缓存距它 r 跳以内的所有 peer 上的共享数据索引，这个 r 必须在 P2P 系统的半径范围内(当 r=0 时就是指这个 peer 本身)，当查询消息到达它时，它能以距它 r 跳内的所有 peer 的身份来处理这个查询请求。由于 peer 能以距它 r 跳以内的所有 peer 的身份来处理查询请求，所以查询消息的 flooding 策略在这种系统中相应的做了一些改变。下面我们就用整个缓冲策略的工作过程来说明这一切。既然 peer 可以代表某些节点处理查询请求，那么首先在消息发送之前应该有一个策略 P——规定哪些节点收到消息后处理，哪些节点收到消息后不用处理直接转发。例如：P={1,5}，那么在第一跳和第五跳收到

查询消息的节点都会处理这个查询请求，处于第一跳和第五跳之间的节点只需简单的转发这条消息就可以了(注：这个策略非常适合 r=2，TTL=7 的情况)。为了保证缓存数据的有效性，这种方法必须考虑一个非常关键的问题：节点的加入和退出。在这个方法中规定，节点加入时必须发出 TTL=r 的加入消息，并且消息中包含这个节点上的所有关于可以共享的数据的索引信息，而且每一个收到这种消息的节点必须给这个节点直接回一个信息，中间也包含它的所有共享数据的索引信息。同时规定一个超时来管理节点的各种离开情况，如果这个节点有一定的次数没有相应 ping 信息，那么缓存索引节点就会清空关于这个已离开的节点的信息。在这个方法的基础上，文[15]、[16]的作者通过调查本地的查询消息，进一步提出了缓存查询字符串和所经过的查询结果的方法。

现在对于缓存文件内容也有一些研究，基本上只是处于试验阶段，例如：文[17]在 KazaA 上建立一种非常理想化的文件内容缓存仿真系统。它的思想来源是 Web 应用中的缓存思想，但是由于 P2P 系统的对象比 Web 系统中的对象数据量大得多，所以在这个仿真系统中缓存的是每一块大小为 32kB 的数据片断，并且决定是否缓存的因素不是 Web 系统中的对象的点击率而是数据片断的点击率。

2.3 overlay optimization 类方法

从产生冗余消息的源头来讲主要有两个方面：第一个方面是 flooding-based search，第二个方面是物理网络拓扑和 P2P overlay 的不匹配。Overlay optimization 就是使 P2P overlay 更好地匹配物理网络拓扑，在文[18,19,20,21]中分别提出了 Adaptive Connection Establishment(ACE)、Two Hop Away Neighbor Comparison and Selection(THANCS)、Scalable Bipartite Overlay(SBO)和 Location-aware Topology Matching(LTM)四种优化方法。在文[20]中还全面地分析了由于拓扑不匹配和 flooding-based search 产生的冗余消息，并且把它们分为两类：overlay 连接上的多余消息，如图 2(a)中的消息 1；物理连接上的多余消息，如图 2(b)中的消息 2(注：图 2 中(a)图代表 P2P 拓扑，(b)图代表(a)图所对应的物理网络拓扑)。



(a) Overlay topology　　(b) Physic topology

图 2　冗余消息类型

2.3.1 ACE

ACE 的核心思想就是将通信开销大的逻辑邻居用通信开销小的 peer 替换掉。它的工作过程分三个阶段：第一个阶段是每个 peer 都用探测消息探测它与逻辑邻居之间的通信开销，使用一张耗费表将这些测试数据记录下来，然后邻居之间相互交换这张耗费表；第二个阶段是利用获得的这些数据在每个节点和它的逻辑邻居之间建立一颗最小生成树，这时节点如果要 flooding 查询消息，它就会选择这个最小生成树中的直接逻辑邻居，而其余的邻居则作为 non-flooding 邻居记录下来；第三个阶段，根据第二阶段分的 flooding 和 non-flooding 邻居的情况，以及耗费表中的数据，将物理距离上远的邻居用物理距离近的 peer 代替。

2.3.2 SBO

SBO 方法的核心思想是在 overlay 形成之初尽量减少环形结构的形成，并在 overlay 形成后对邻居关系进行调整，尽量使物理距离近的 peer 具有邻居关系。它主要分四个阶段：第一个阶段，节点加入系统时首先从红、白两种颜色中随机选择一种颜色来标记自己，而且在选择邻居时必需选择与自身的颜色不同的 peer 作为邻居；第二阶段，每个选择白颜色标记的 peer 探测它与所有红色邻居之间的距离，然后把这些信息告诉给每一个红色的邻居；第三个阶段，所有红色邻居根据得到的信息计算出消息广播的最有效路径；第四个阶段，所有不在最有效路径的白色 peer 去掉这个计算路径的红色邻居并选择更好的红色 peer 来当它的邻居。

2.3.3 LTM 和 THANCS

LTM 和 THANCS 的核心思想都是将 overlay 拓扑中物理距离远的边去掉，建立物理距离近的边。在 LTM 中首先规定了一种 TTL=2 的探测消息记为 $d(i,S,v)$，i 表示消息 ID，S 表示消息源节点，v 表示消息的生命周期 TTL。要求所有的 peer 中的时钟要同步。所有探测情况可以分为四类，以下用四个例子加以说明。图 3 中(a)图代表第一类，(b)图代表第二类，(c)图代表第三类，(d)图代表第四类。

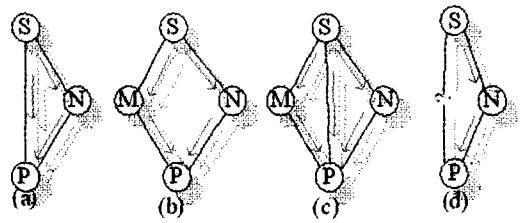


图 3　探测情况分类

图 3(a)中 P 将收到探测消息 d(i,S,1)和 d(i,S,0)，并且从(a)中我们可以清楚地知道 d(i,S,1)是从路径 SP 过来的，d(i,S,0)是从路径 SNP 过来的，通过这些消息可以分别计算出 SP、SN、NP 之间的物理距离。然后将它们三个比较，如果 SP 或 NP 物理距离最远，将最远的放入 will-cut 表中，所有放入这个表中的连接在一定的时间内都会被剪断，并且在这段时间内不会有 flooding 消息经过它；如果 SN 最远不做任何动作。在(b)中 P 将连续几次收到 d(i,S,0)这种探测消息，通过这些消息 P 可以计算出 SM、SN、MP、NP 的物理距离，然后对它们进行比较，如果 MP 或者 NP 是最远的，与前面一样将最远的连接放入 will-cut 表中，否则不作任何处理。在实际系统中 P 可能会收到很多 d(i,S,0)这类消息，那么它就将这些随机的配对进行处理。对于图(c)将随机选择一个 d(i,S,0)与 d(i,S,1)进行配对，然后与处理(a)情况一样处理。在(d)中由于 S 与 P 之间没有逻辑连接，所以 P 收到 d(i,S,0)后，可能在很长一段时间内都收不到 d(i,S,1)，P 将直接发送一条探测消息给 S，以探测 SP 之间的距离，把它与 SN、NP 进行比较，如果 SP 不是三个中最远的，就在 S 和 P 之间建立一个逻辑连接。THANCS 对 LTM 做了进一步的改进，peer 在加入系统之初就判断它自己到所有邻居的距离，并存储在系统中，然后使用消息捎带的方法将这些信息分给它的每一个邻居共享，这样使每一个 peer 都有像图 3 中 P 节点一样的信息来判断是剪掉连接还是增加连接，只是 peer 判断时要分清哪些节点既是它的邻居也是它邻居的邻居，例如：图 3(a)、

(b)中 P 和 S 的关系。

3 各类方法的性能评估与比较

对于各种方法,由于它们出发的角度不同,所以在优化性能方面都有自己的长处,下面我们用一张表格来说明它们的性能。在这张表格中我们用四种因素来评价我们上面所介绍的每一种方法的性能,它们分别是:查询范围、网络开销降低程度、查询效率提高程度、优化所用网络开销。查询范围是指某个 peer 广播一个查询消息所能到达的范围,这里都以 flooding search 的范围为基准。网络开销降低程度是指能够使 P2P 系统中的消息量减少的程度。查询效率提高程度是指缩短查询响应时间的程度。优化所用网络开销是指算法优化 P2P 系统时使网络负载增加的程度。对网络开销降低程度、查询效率提高程度分别用五个等级表示:*、**、***、****、*****,星号越多性能越好。而对优化所用网络开销用三个等级表示,星号越多表示开销越大,还有"—"表示基本上没有开销。从图中我们可以看出 forwarding-based 类方法会缩小查询范围,可能在某些时间影响查询效果,而且在性能优化程度上要远远落后于后面的两类方法。Cache-based 类方法开销很大,而且性能不是很理想。如果它们能够和 overlay optimization 类方法结合起来,还是能够得到很理想的结果,虽然优化所用网络开销大一点,但是一般来说这种开销相对于我们所减少的 P2P 系统中搜索所带来的开销来说是很小的一部分。特别是 ACE 和 Local Indices 的结合,比他们两个原来的性能要好得多。所以我们从这张表中可以看出,以后在优化方面的研究应该是怎样将三类方法结合起来使用,使它们能够相互取长补短,发挥更好的优化性能。

表 1 性能比较

方法名	是否缩小查询范围	网络开销减少程度	查询效率提高程度	优化所用网络开销
Directed BF	是	*	*	—
HPF	是	*	*	—
Local Indices	否	* *	* *	* * *
ACE	否	* *	* *	* *
SBO	否	* * * *	* * *	*
LTM	否	* * * *	* * *	* *
THANCS	否	* * * *	* * *	*
ACE+Local Indices	否	* * * *	* * * *	* * *
SBO+Local Indices	否	* * * * *	* * * * *	* *
THANCS+Local Indices	否	* * * * *	* * * * *	* *

总结 本文概括性地介绍了当前国际上在 decentralized structured P2P 系统性能优化研究方面的发展状况和研究的热点,系统的介绍了现在解决 P2P 系统大量消耗网络资源的方法,并对这些方法做出了评价和比较。

参 考 文 献

1 Napster. http://www.napster.com, 2005

2 Gnutella. http://www.gnutella.com, 2005

3 Lv Q, Cao P, Cohen E, et al. Search and Replication in Unstructured Peer-to-Peer Networks. Proc 16th ACM Int'l Conf Supercomputing, 2002

4 Stoica I, Morris R, Karger D, et al. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. Proc ACM SIGCOMM, 2001

5 Rowstron A, Druschel P. Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems. Proc Int'l Conf Distributed Systems Platforms, 2001

6 Zhao B Y, Kubiatowicz J D, Joseph A D. Tapestry: An Infrastructure for Fault-Resilient Wide-Area Location and Routing: [Technical Report]. UCB//CSD-01-1141. Univ of California, Berkeley, 2001

7 Ratnasamy S, Francis P, Handley M, et al. A Scalable Content-Addressable Network. Proc ACM SIGCOMM, 2001

8 Sen S, Wang J. Analyzing Peer-to-Peer Traffic across Large Networks. Proc ACM SIGCOMM Internet Measurement Workshop, 2002

9 Saroiu S, Gummadi K P, Dunn R J, et al. An Analysis of Internet Content Delivery Systems. Proc Fifth Symp Operating Systems Design and Implementation, 2002

10 Saroiu S, Gummadi K P, Dunn R J, et al. An analysis of internet content delivery systems. Proceedings of the Fifth Symposium on Operating Systems Design and Implementation (OSDI 2002), Boston, MA, December 2002

11 KaZaA. http://www.kazaa.com, 2005

12 Ripeanu M, Iamnitchi A, Foster I. Mapping the Gnutella Network. IEEE Internet Computing, 2002

13 Yang B, Garcia-Molina H. Efficient Search in Peer-to-Peer Networks. Proc 22nd Int'l Conf Distributed Computing Systems (ICDCS), 2002

14 Zhuang Z, Liu Y, Xiao L, et al. Hybrid Periodical Flooding in Unstructured Peer-to-Peer Networks. Proc Int'l Conf Parallel Processing, 2003

15 Markatos E P. Tracing A Large-Scale Peer-to-Peer System: An Hour in the Life of Gnutella. Proc Second IEEE/ACM Int'l Symp Cluster Computing and the Grid, 2002

16 Sripanidkulchai K. The Popularity of Gnutella Queries and Its Implications on Scalability. http://www.cs.cmu.edu/~kunwadee/research/p2p/gnutella.html, 2006.1

17 Saroiu S, Gummadi K P, Dunn R J, et al. An Analysis of Internet Content Delivery Systems. Proc Fifth Symp Operating Systems Design and Implementation, 2002

18 Liu Y, Zhuang Z, Xiao L, et al. A Distributed Approach to Solving Overlay Mismatch problem. Proc 24th International Conference on Distributed Computing Systems, 2004

19 Liu Y, Esfahanian A-H, Xiao L, et al. Approaching Optimal Peer-to-Peer Overlays. Proceedings of the 13th Annual Meeting of the IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS), 2005

20 Liu Y, Xiao L, Ni L M. Building a Scalable Bipartite P2P Overlay Network. Proc 18th International Parallel and Distributed Processing Symposium, 2004

21 Liu Y, Xiao L, Liu X, et al. Location Awareness in Unstructured Peer-to-Peer Systems. IEEE Transactions on Parallel and Distributed Systems (TPDS), 2005