基于 MCES 的网络编程接口研究*)

易发胜　夏梦芹　叶娅兰　曾家智
(电子科技大学计算机学院　成都 610054)

摘　要　本文对目前网络编程接口的局限性进行了分析,发现其已经难以适应网络应用的需求和网络技术的最新发展。微通信元系统构架(MCES)是一种易于从 TCP/IP 过渡的服务元网络体系结构[6]的构架,为了应用微通信元系统构架提供的高级网络服务,根据当前网络编程接口方面的研究情况,提出了新的网络 API:GSocket。GSocket 采用合理的设计,完全兼容传统的网络编程方法和接口函数,能自动适应不同的网络系统,支持 QoS 等多种高级网络服务功能,还具有良好的可扩展性。

关键词　网络编程,套接字,QoS 支持,终端体系结构

The Study of General Network API of MCES

YI Fa-Sheng　XIA Meng-Qin　YE Ya-Lan　ZENG Jia-Zhi
(Department of Computer Science, UEST of China, Chengdu 610054)

Abstract　This paper analyzes the limitation of current network API, and shows that these APIs has not suited for the request of network application and development of network technology. Micro Communication-Element System(MCES) can be used to migrate from the TCP/IP protocol stack to the Service Unit based Network Architecture(SUNA), In order to apply advanced services of MCES, according to the research of current network program interface, this paper put forward a new network API: GSocket. GSocket adopts reasonable design, has complete compatibility with conventional network program mode and interface function, and fits automatically different network system, supports lots of advanced network service function such as QoS, possesses nice expansibility also.

Keywords　Network program, Socket, Qos support, End system archtecture

1　引言

网络 API 是操作系统提供的网络应用程序使用网络功能的用户接口。目前广泛使用的网络 API 是基于 TCP/IP 网络的,我们称之为 socket,已经成为事实上的网络编程标准。TCP/IP 提供了可靠的数据流传输(TCP)和不可靠的数据报(UDP)服务,也相应地提供了这两种服务的网络编程接口。

TCP/IP 是 20 世纪 70 年代发展起来的一种网络系统,基于层次体系结构,主要应用在文本数据的传输方面,因此根据 TCP 协议和 UDP 协议提供的网络服务能够满足当时的网络应用的需要。随着网络应用的日益普及,适用的范围不断扩大,对网络的需求也越来越高,特别是网络多媒体应用对网络提供了更高要求,比如 IP 电话、多媒体会议等,不仅要求网络提供一定流量数据传输率,还要求网络对数据的传输时间也提供一定保证,这些要求统称为服务质量(QoS)。网络的这些要求推动了网络技术不断发展,基于 TCP/IP 体系开发了 MPLS[1],RSVP[2],RTP 和 RTCP[3]等协议来处理服务质量要求,还开发了帧中继、ATM[4]等保证服务质量的网络系统。

总的来说,基于层次网络体系结构的网络系统扩展性差、效率低、功能具有冗余。近年来,出现了对非层次的计算机网络体系结构的研究,如基于角色的网络体系结构[5]、服务元网络体系结构[6]等,它们在提供的网络服务上都增加了对 QoS 的良好支持,并对网络的效率和性能以及可扩展性都有很好的改进。服务元网络体系结构是一种模块化结构,模块是服务元。服务元是能够提供服务而又隐藏内部细节的最小实体(硬软件)。服务元之间没有严格层次要求,这样能很好地适应不同的网络应用需求,具有很好的可扩展性,也提高了网络处理效率,并功能不会冗余。

微通信元系统 MCES 是服务元网络体系结构的一个实现,易于从 TCP/IP 过渡而来[6],同时,允许用户定义服务元来扩展网络功能,这大大方便了网络应用的升级。但目前网络 API 并没有很好地适应这种网络应用和网络技术的快速发展。比如传统 Socket 没有提供对 QoS 的接口支持,无法提出或使用 QoS 等功能需求。网络 API 是操作系统提供给编程人员使用网络资源的界面,我们希望网络 API 具有如下特性:1、使用简便,并完全兼容已经存在的 BSD Socket;BSD Socket 已经成为事实上的标准,很多的编程人员已经熟悉,所以必须在兼容的基础上扩展。2、具有可扩展性,能够方便提出对网络的访问控制要求;包含已经实现和将来可能实现的网络服务功能。3、应能适应不同网络系统的实现,并具有高效的处理性能。

近年来,有许多基于 BSD 套接字的改进和研究工作。其中文[7]初步研究了应对各种网络服务的网络编程接口解决方案,提出了 Sockets++概念,采用面向对象的方法实现套接字的管理,并支持各种网络服务,除了传统的 TCP/IP 的套接字功能,它还支持对 QoS 的控制和组播通信等,并考虑了

*)国家自然科学基金资助项目,编号:69871005。**易发胜**　讲师,博士研究生,从事新型网络体系结构研究。**夏梦芹**　博士研究生,从事新型网络体系结构研究。**叶娅兰**　博士研究生,从事新型网络体系结构。**曾家智**　教授,博导,从事计算机网络与通信研究。

平台独立性、接口简化、支撑多种网络协议等问题。但是它作为 TIP 工程[8]的一部分，突出采用面向对象的方法，导致效率比较低，同时没有考虑将来的功能扩展。另一种 QoSockets 方案[9]，它的主要工作是提出了一种在应用程序和传输层协议之间传输和搜集 QoS 信息的接口机制，包含相应的 QoS MIB 的实施，并提供了简化的具有 QoS 要求信息的接口。但对 QoS 以外的因素考虑较少。还有一种 QSockets[10] 机制，也是在 BSD 套接字上的一种支持 QoS 的扩展，它的特点在于让应用程序通过接口能够支持对流量调度的选择控制，达到更好的端系统 QoS 控制，并允许开发者对每个报文进行 QoS 控制。同前面一样，这种机制也只是考虑 QoS 流量和带宽控制方面，并只能在 Linux 环境下，没有考虑网络编程的其他要求。Winsock2[11] 适用于 Windows 操作系统的 socket 接口 API，提供了一系列的支持 QoS 的 API，但是这些 API 是独立的并且依赖于 Windows 操作系统的，并不具有扩展性。

我们这里在研究基于 MCES 架构的实现机制上，提出了一种新的可扩展套接字 GSocket(general socket)接口机制。它借鉴了前面对 QoS 的编程控制研究结果，着重考虑如何适应新网络应用需求和适应新网络技术发展方面的问题，并尽量兼容传统的套接字编程。本文第 2 节讨论了 GSocket 功能特点和体系结构；第 3 节说明了 GSocket 的实现机制；第 4 节说明应用程序开发者如何使用这种网络 API 开发具有 QoS 的网络应用；第 5 节对其处理性能进行了分析。最后对文章作了总结。

2 GSocket 的体系结构

网络应用程序、网络 API 以及主机网络系统的实现共同组成网络体系的端系统。网络 API 的功能和性能直接影响网络的应用推广和传输性能。相对于 TCP/IP 而言，基于 MCES 的网络系统能够提供灵活的网络服务，在实施 MCES 模型的研究中，我们提出了一种 GSocket 的接口机制。如图 1 所示。

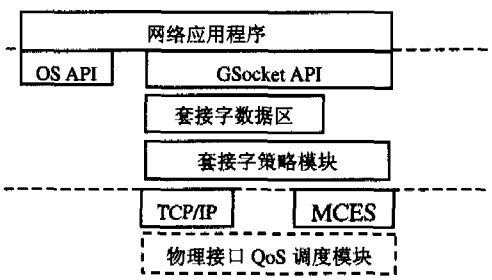

图 1 端系统 GSocket 体系结构

图中，上面的虚线是操作系统和应用程序之间的界面，GSocket 作为操作系统提供的一种 API 而存在。在两条虚线之间，是套接字接口的实现模块，它主要包含两个部分，即套接字数据区和套接字的策略模块。套接字数据区维护了一个套接字的各种信息，包含网络端系统保存的控制和状态信息。而套接字策略模块根据网络应用程序的要求，按照数据区的信息，要求虚线下面的网络系统提供合适的网络服务。此模块的适当设计，将保证 GSocket API 能够选择恰当的网络服务，并具有各种适应性要求，比如可扩展性、QoS，甚至组播和安全性等网络编程要求。在下面的虚线以下的部分是集成在操作系统中的 MCES，它和 TCP/IP 等网络系统一起，可以提供多种用户需要的网络服务。对于某些提供 QoS 要求的网络系统，在物理接口还需要 QoS 调度模块，便于进行数据报文的缓冲区队列调度。这样的设计简化了网络编程用户对网络系统实现的关心，用户只需要按照网络编程接口提出网络服务的要求就可以了。

3 GSocket 的实现

从网络应用需求和网络服务的分析着手，结合上述的体系结构，分析如何实现 GSocket 设计目标。

3.1 网络应用对系统的要求分析

应用程序要从主机(端系统)申请希望的网络服务，需要网络系统满足如下条件：

A. 网络系统支持如此服务。如要传输可靠数据流，需要实现类似 TCP 的协议；要传输 QoS 要求信息，则需要实现 RSVP 之类的协议以及相应的调度策略等。

B. 本地系统具有足够资源。包括本地系统有足够缓冲区，主机接口有可用带宽等。

C. 通信网络具有足够的资源。特别是对于具有 QoS 要求的网络应用，网络必须提供足够的带宽和缓冲区等资源。

D. 对方主机具有足够资源。同本地系统需要足够资源类似，对方主机也必须保证具有足够缓冲区和带宽资源。

不同的网络服务对这四个条件的要求也不一样。如无连接数据报服务要求最低，几乎不需要做过多的检查；而需要保证传输可靠的数据流服务则需要先建立 TCP 连接，让本地主机和目的主机准备好足够的缓冲区去启动滑动窗口机制传输数据。

3.2 网络提供的服务分析

仔细分析网络系统可以提供的各种服务，发现应用程序希望网络提供的服务包括四个方面，即顺序要求、可靠要求、迟延要求和带宽要求。此外对于不同的服务类型还可以选择是否使用组播和安全要求。不同的网络系统能够同时满足上面的一种或者多种服务要求，比如 TCP 能够满足顺序要求和可靠要求，而微通信元系统能够提供上述各种要求，但是可靠要求和迟延要求往往不能同时满足。我们将这四种服务进行仔细组合之后，暂时制定五个服务类型，如表 1 所示：

表 1

类型	含义	服务包含功能	提供服务的系统举例
0	数据报服务	无	MCES,UDP
1	数据流服务	顺序、可靠	MCES,TCP
2	紧急数据流	顺序、可靠、带宽	MCES
3	实时数据	顺序、带宽、迟延	MCES
4	可靠数据报	可靠	MCES

其中，每种类型由于具体参数定义的不同，又有很多小的类别，比如类型 3，对带宽和迟延可以有多种定义，会有 ABR，VBR，UBR 等多种应用类型。根据网络应用和技术发展，我们还能组合更多的服务类型。

3.3 套接字数据区的意义

在 GSocket 体系结构中的套接字数据区是网络应用程序和网络系统联系的桥梁。各套接字有独立的套接字数据区，保存一个网络应用程序对网络的服务要求(如 QoS 参数、服务类型等)、通信过程中的状态信息、收发数据的缓冲区以及其它网络信息。应用程序通过网络 API 设置、查询或者修改数据区中某些参数；而网络系统也通过查询某些参数来决定自己的行为，并将网络服务过程中的状态信息记录在这个数

据区中，便于应用程序了解网络状况。此外，这个数据区也是套接字策略模块进行决策的依据。

3.4 套接字策略模块

套接字策略模块负责根据用户的要求，选择适当的网络服务。它检查用户的要求是否能够得到满足，配合 QoS 调度模块和微通信元系统的返回结果，告诉应用程序能否得到希望的网络服务。

每个网络系统对自己能够提供的服务类型在操作系统中进行注册，并提供统一的接口来传递参数建立网络连接以及收发数据。对于层次网络体系结构如 TCP/IP 网络系统，比较简单，只需要最上层的服务即 UDP 和 TCP，类似 0、1 这两种网络服务类型。而对无层次的 MCES，系统注册各个网络通信元，各个通信元分别实现不同的功能，然后组合起来提供某种网络服务。

设已经注册的各个通信元的功能分别为 $A_1, A_2, \cdots, A_m$，则能够实现的网络服务是 $F_n = \sum_{i=1}^{n} A_i$，对于不同的网络服务，策略模块选择不同的通信元集合，并且按照不同的顺序来处理网络数据。这给用户带来了很大方便，在使用新网络服务时，或者扩展网络功能时，应用程序可以自动适用。

3.5 GSocket API 的设计

网络 API 根据其作用可以分为两类，第一类是设置参数或查询状态之类的，它们不产生网络数据的传输操作；第二类就是网络系统的传输相关的。在传统的 socket 中，socket，bind，listen 等函数属于第一类，而 send，recv 等函数属于第二类。此外 connect，close 等函数则和服务类型有关，在 UDP 服务中属于第一类；在 TCP 服务中，属于第二类。

在 GSocket 的 API 设计过程中，考虑到和传统的 socket 兼容的问题，传统的套接字 API 全部使用，用于实现 0 类和 1 类网络服务。在 GSocket 的实际实现中，第一类 API 用于操作套接字数据区。这个数据区在网络应用程序使用 socket 函数打开一个套接字的时候产生，并自动置初值。传统套接字的一些接口函数会影响其中一些成员变量的值，为了让 GSocket 支持更多更高级的网络特性，需要增加一些系统调用对表示 QoS 或者体现安全的成员变量进行设置或者查询操作。如果我们不用 socket 提供的高级服务，就不需要使用这些新的 socket 函数；这样既保证了和过去的套接字编程兼容，又具有可扩展性。随着网络技术的发展，这个数据区的数据成员可以根据实际网络系统提供的服务需要增加新的成员，同时也增加相应的操作接口函数操作这些成员。对于应用程序来说，不需要使用更加高级的网络服务，就不需要改动程序，这样保持和以前的网络程序兼容。

在第二类 API 中，根据传统的编程习惯，使用 connect 函数来建立 TCP 连接，因此我们在 GSocket 中，也让应用程序调用 connect 函数来检查应用程序需要的所有条件是否满足。这个函数启动策略模块，首先检查在网络系统中是否能提供希望的网络服务，然后调用本地接口调度模块检查本地系统是否有足够的带宽和缓冲区；之后，调用相关的网络系统接口建立网络系统的通信连接，比如微通信元系统的建立虚电路的过程，TCP 的建立连接的三次握手过程，或者 RSVP 的预留资源过程等。所有需要的参数都来自套接字数据区。完成这个过程之后，connect 返回成功，应用程序就可以实现以后的数据传输操作了。

其它第二类 API 会配合策略模块的判断去调用相应网络系统接口得到相应的服务。为了充分利用高级网络服务的特点，也会增加一些新的 API，更好地控制高级网络功能。

4 GSocket 的使用

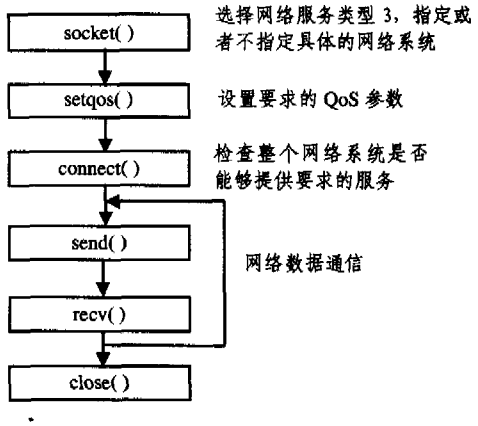

图 2 客户端编程流程

GSocket 充分考虑已经广泛应用的 BSD Socket 的编程习惯，提供给用户的编程接口要求完全兼容过去的编程流程。如果只是类似 0、1 两种类型的网络服务，将不用更改网络程序。对于其他类型的网络服务，则可以模仿 TCP 数据流的网络编程模型，只是在调用 connect 函数之前，需要调用增加的 GSocket 接口函数设置相关的 QoS 参数或者其他参数。图 2 是 MCES 中对类型 3 的一个客户端编程流程。

如图 2 所示，由于 GSocket 内部进行了许多适当的处理，在应用程序编程接口只需要提出网络服务的要求即可，保持传统的网络编程习惯，大大简化用户的编程学习要求。

5 GSocket 的运行性能分析

GSocket 的实现采用了一个智能策略控制模块，使网络应用程序具有很好的适应性和可扩展性能，在统一的接口界面下，能够很好地适应网络应用和网络技术的发展，这为我们开发新的网络体系结构提供了一个很好的编程接口。在服务元项目中我们用 C 语言在 Linux 系统中实现了 GSocket 模型，在 TCP/IP 和 MCES 上面能够很好工作。为了分析这样设计的实际运行性能，我们利用 TCP/IP 协议栈和 0 类服务，同传统 BSD Socket 的 UDP 服务进行对比。这里主要对比了发送时间和吞吐量两个指标。

如图 3 所示，GSocket 的各项指标相对于 BSD Socket 来说，有些微弱差距。这主要是由于在 GSocket 实现过程中，有一个策略模块的处理，稍微增加了系统处理时间，导致发送时间稍微增加，但是这种差距很小，而且还可以通过优化处理进一步减小。而对于传输数据吞吐量，差别更小，这是因为数据拷贝传输的时间大于接口函数的处理时间，从而使两者的数据吞吐量基本保持一致。

可以看到，GSocket 对 0 类网络服务的处理性能非常接近 BSD socket。同样，针对 1 类服务和 TCP 服务的对比测试也有类似结果。因此我们可以得出这样的结论，虽然 GSocket 进行了更多的策略处理来简化网络编程要求和提高支持各种网络服务的灵活性和扩展性，但其性能并没有明显下降。

结论 传统的 BSD socket 难以适应网络应用程序的发展需要。这篇文章在基于服务元网络体系的微通信元系统上对网络 API 进行了有益的探索，提出一种最新设计的网络

API:GSocket。GSocket在不同的网络系统的顶端实现了一个智能的策略控制模块,使网络应用程序具有很好的适应性和可扩展性能,在统一接口界面下,能够很好地适应网络应用和网络技术的发展。这为我们开发新的网络体系结构提供了一个很好的编程接口。经过比较测试其模型的运行性能,发现其运行性能很接近目前使用的BSD socket的性能。

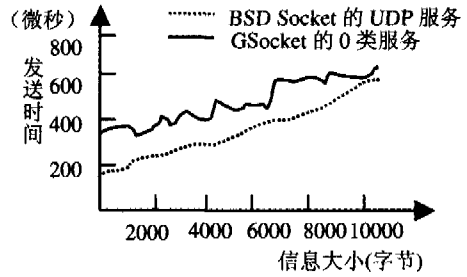

图3 0类服务与UDP的发送时间对比

下一步需要做的工作是根据实际的需要定义更多的网络服务类型,并实现其模型进行性能测试和比较,同时优化设计套接字策略模块,进一步提高GSocket的处理性能。

参考文献

1 Rosen E, Viswanathan A, Callon R. Multi- protocol Label Switching Architecture. RFC3031, Jan. 2001
2 Braden R, Zhang L, Estrin D, et al. Resource ReServation Protocol (RSVP) Version 1 Functional Specification. RFC2205, September 1997
3 Schulzrinne H, Casner S, Frederick R, et al. RTP: a Trans- port Protocol for Real-Time Applications. RFC 1889, 1989
4 De Prycker M. Asynchronous Transfer Mode: Solution for Broadband ISDN, 3rd. Prentice-Hall, Englewood Cli. s, NJ, 1995
5 Braden B, Faber T, Handley M. From Protocol Stack to Protocol Heap - Role-Based Architecture. First Workshop on Hot Topics in Networking, Oct. 2002
6 曾家智,等. 服务元网络体系结构和微通信元系统构架. 电子学报 2004,32(5):745~749
9 Bocking S. Sockets++: a uniform application programming interface for basic level communication services. Communications Magazine, IEEE, 1996, 34(12): 114~123
10 Bocking S. TIP's Performance Quality of Service. IEEE Commun. Mag. 1993, 33(8)
11 Florissi P G S, Yemini Y, Florissi D. QoSockets: a new extension to the sockets API for end-to-end application QoS management. Computer Networks 2001, 35(1): 57~76
12 Hasan Abbasi, et al. A Quality-of-Service Enhanced Socket API in GNU/Linux. http://www. linuxdevices. com/articles/ AT8639420091. html
13 Microsoft. Windows Socket 2 specication. URL: http://msdn. microsoft. com /library/ psdk/winsock/ wsapiref_2qr6. htm

(上接第49页)

击能力差。PAST提出了一个通过智能卡来实施配额的方案,这需要可信第三方和智能卡硬件[7]。Data trading要求节点间交换相同数量的实际数据[8]和文[9]一样要求对称存储关系。SHARP要求在离线状态下建立起节点间公钥的信任关系[10]。文[11]讨论了存储资源和带宽限制机制,但要求有PKI支持。Fileteller[12]使用Micropayments[13]进行文件存储的收费,但Micropayments要求分布式事务,对P2P备份系统并不适用[14]。

Samsara[4]与P-Promise最接近,其基本思想是通过引入存储声明(storage claim)来将不具备对称存储关系的环境转换为具有对称存储关系的环境来实现公平的存储资源共享。P-Promise和Samsara相比,特点和不同主要有:(1)思想和原理不同,P-Promise借鉴人类社会的承诺与诺言兑现机制,而Samsara是通过存储声明将非对称存储关系转换为对称存储关系,因此,其必然要求参与各方在资源能力方面是对称的,从而限制其适用范围;(2)P-Promise的承诺证书可以描述存储资源以外的资源,如处理器周期、信息等,可以由应用定义,而Samsara的存储声明只适用于存储资源;(3)P-Promise可以适用于资源和处理能力都比较有限的移动设备,比如资源丰富的桌面计算机可以代替移动设备进行承诺并兑现,移动设备就可以使用P2P环境中丰富的资源,而Samsara不能。

结论 P-Promise借鉴人类社会个体间相互承诺并通过兑现承诺来实现公平交往的机制,通过定义承诺证书和协议原语在P2P环境中建立公平的资源共享环并实现P2P公平资源共享。承诺证书可以描述各种资源,所以P-Promise可用于不同资源的共享。公平资源共享环的建立过程就是对等实体承诺并不断兑现承诺的过程,因此,P-Promise不需要可信第三方、认证的身份、货币支付和对称存储关系等有违P2P初衷的条件。分析和实验表明,P-Promise和信任管理机制一起使用,两者相互补充,具有良好的稳定性和抗攻击能力。下一步将把P-Promise应用到存储资源之外的资源共享中。

参考文献

1 Barkai D. An Introduction to Peer-to-Peer Computing. Intel Developer Update Magazine, 2000
2 Milojicic D S, Kalogeraki V, Lukose L, et al. Peer-to-Peer Computing: [Technical report, HPL-2002-57R1]. Hewlett-Packard Company, 2002
3 Adar E, Huberman B A. Free riding on Gnutella. First Monday, 2000, 5(10)
4 Cox L P, Nobel B D. Samsara: Honor Among Thieves in Peer-to-Peer Storage. 19th ACM Symposium on Operating Systems Principles, New York, 2003
5 Adya A, Bolosky W J, Castro M, et al. FARSITE: Federated, available, and reliable storage for an incompletely trusted environment. 5th Symposium on Operating Systems Design and Implementation, Boston, MA, 2002
6 Dabek F, Kaashoek M F, Karger D, Morris R, Stoica I. Wide-area cooperative storage with CFS. 18th ACM Symposium on Operating Systems Principles, Canada, 2001
7 Rowstron A, Druschel P. Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility. 18th ACM Symposium on Operating Systems Principles, Banff, Canada, 2001
8 Cooper B F, Garcia-Molina H. Peer-to-peer resource trading in a reliable distributed system. First International Workshop on Peer-to-Peer Systems, Cambridge, MA, 2002
9 Lillibridge M, Elnikety S, Birrell A, Burrows M, Isard M. A cooperative Internet backup scheme. USENIX Annual Technical Conference, Texas, 2003
10 Fu Y, Chase J, Chun B, Schwab S, Vahdat A. SHARP: An architecture for secure resource peering. 19th ACM Symposium on Operating Systems Principles, New York, 2003
11 Tsuen-Wan "Johnny" Ngan, Nandi A, Singh A, et al. Designing Incentives-Compatible Peer-to-Peer Systems. 2nd Bertinoro Workshop on Future Directions in Distributed Computing, Bertinoro, Italy, 2004
12 Ioannidis J, Ioannidis S, Keromytis A D, et al. Fileteller: Paying and getting paid for the storage. 6th Annual Conference on Financial Cryptography, Bermuda, 2002
13 Blaze M, Ioannidis J, Keromytis A. Offline micropayments without trusted hardware. 5th Annual Conference on Financial Cryptography, BWI, 2001
14 Cox L P, Murray C D, Noble B D. Pastiche: Making backup cheap and easy. 5th Symposium on Operating Systems Design and Implementation, Boston, MA, 2002