ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.19, No.12, December 2008, pp.3179−3195
DOI: 10.3724/SP.J.1001.2008.03179

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

基于服务的网络体系结构的设计和实现*

易发胜[1+], 陈贵海[2], 刘 明[1], 龚海刚[1], 曾家智[1]

[1](电子科技大学 计算机科学和工程学院,四川 成都 610054)

[2](南京大学 计算机软件与新技术国家重点实验室,江苏 南京 210093)

Design and Implementation of Service-Based Network Architecture

YI Fa-Sheng[1+], CHEN Gui-Hai[2], LIU Ming[1], GONG Hai-Gang[1], ZENG Jia-Zhi[1]

[1](School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 610054, China)

[2](State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093, China)

+ Corresponding author: E-mail: yifs@163.com

Yi FS, Chen GH, Liu M, Gong HG, Zeng JZ. Design and implementation of service-based network architecture. *Journal of Software*, 2008,19(12):3179−3195. http://www.jos.org.cn/1000-9825/19/3179.htm

Abstract: The state of the art of network architecture is presented. Based on a micro-communication element structure (MCES), a service-based network architecture prototype is implemented, in which service units are scheduled and assembled logically. While supporting QoS and ensuring network security more effectively, the prototype can be compatible with all existing TCP/IP applications by an improved socket design. The experimental results show that the prototype of service-based network architecture is feasible, and possesses good transmission efficiently and scalability.

Key words: network architecture; MCES; QoS; network security

摘 要: 对网络体系结构的研究现状进行了介绍.基于 MCES(micro-communication element structure)架构设计了一种基于服务的网络体系结构原型.在原型系统中,实现了服务元的合理组合和调度,并从体系结构上更好地支持 QoS 和网络安全.同时,通过改进的套接字机制实现了向后兼容现有 TCP/IP 网络应用程序,满足了用户不断增长的网络服务要求.基于 Linux 的实验结果显示,该体系结构组成的系统是合理的、可行的,并具有很好的传输效率和可扩展性.

关键词: 网络体系结构;微通信元架构;服务质量;网络安全

中图法分类号: TP393 **文献标识码**: A

网络体系结构体现了网络系统的设计思想.当前网络体系结构形成于 20 世纪 70 年代,其层次网络体系结构较好地解决了复杂通信协议的设计问题.基于尽力而为思想实现的 TCP/IP 四层网络系统具有很高的资源利

* Supported by the National Natural Science Foundation of China under Grant Nos.60573131, 60673142 (国家自然科学基金); the National Basic Research Program of China under Grant No.2006CB303000 (国家重点基础研究发展计划(973)); the Youth Research Foundation of University of Electronic Science and Technology of China under Grant No.JX05031 (电子科技大学青年研究基金); the Jiangsu High-Tech Research Project of China under Grant No.BG2007039 (江苏省高技术研究项目)

Received 2007-05-28; Accepted 2007-11-20

用率和健壮性,同时,分层设计使各种网络技术能够很快地得到广泛应用.

随着网络应用的日益广泛,特别是网络多媒体应用和网上电子商务的发展,当前的 Internet 越来越难以满足网络应用的要求,最初设计时没有特别考虑的服务质量(QoS)和网络安全问题已经成为新兴网络应用的障碍.近年来,许多研究围绕着这两个问题的解决而展开.总的来说,这些研究分为两个方面,一方面是在现有的 Internet 上提供保证 QoS 的机制,比如综合服务模型[1]、区分服务模型[2]、IPSec 等,这些方法的特点是改进现有的尽力而为机制,开发新协议和增加子层来保证网络传输的时间特性和安全性要求.但是,目前还没有得到满意的效果,而且协议的增加导致了处理的复杂性,TCP/IP 协议栈变得日益庞大,层次网络体系结构的局限性日益突出.另一方面是开发新的协议栈或者网络体系结构,以更好地保证 QoS 和网络安全的要求.如已经提出的 ATM,采用虚电路方式来保证 QoS 要求,但却由于技术复杂而没有得到广泛应用.当前投入应用的网络系统都是层次体系结构,其特点是分层管理,容易实施,但是可扩展性差,难以适应不断发展的网络要求.因此近年来,很多目光投向网络体系结构方面,提出了采用非层次体系结构的思想.非层次网络体系结构研究始于 Clark 和 Tennenhouse 在 1990 年提出的面向网络协议处理性能优化的应用级组帧(ALF)[3]的思想,后来又出现了一体化层次处理[4]、定制网络服务的可编程网络模型[5]和面向对象网络体系结构[6]等,这些努力都试图打破层次体系的固有概念以灵活组织网络服务,但仍然是以 TCP/IP 层次体系为基础的改进.而基于角色的网络体系结构[7]和服务元网络体系结构[8]完全去掉了层次网络体系结构的思想,为解决现有网络问题提出了新的思路.但是很多非层次网络的探索目前仅局限于研究领域,缺乏原型系统,没有投入实际应用,无法从实践上证明其设计思想是可行的.

本文就是在非层次的服务元网络体系结构的原理上设计、实现了一个非层次体系结构的网络原型.该原型网络基于微通信元的架构(micro-communication element structure,简称 MCES),提出了服务元和服务设计方法以及非层次网络功能的调度模型,实现了当前因特网类似基本网络功能.并通过实验展示了该结构的可行性和合理性以及良好的传输性能和效率.

1 网络体系结构的相关研究

网络体系结构是计算机网络的总体设计思想[9],用来指导计算机通信机制的设计和通信协议的实现.目前层次网络体系结构盛行,因为它具有如下优点:便于抽象,每一层都抽象成黑匣子,内部结构不可见,有利于人们交流理解,有助于标准化;便于模块化,有利于分工和协作开发;灵活性好,便于各个层次独立改进.

体系结构具有一定的相对稳定性和设计指导性,长期以来,因特网的改进都是在其网络体系结构的框架内进行,并有一个专门的机构 IAB 来保证其体系结构的稳定性.最初 TCP/IP 网络协议的整体功能无疑是有限的,与当初相比,从体系结构方面来看,当前的 Internet 有如下几方面的改进:(1) 增加子层或者选项.如让 TCP 支持 SACK,让 IP 支持源路由等选项;为了支持 QoS 或者网络安全的需要,增加了 MPLS,IPSec,SSL 等子层.(2) 各层修改或者增加新协议.如为了解决 UDP 没有拥塞控制机制,可能导致网络崩溃的问题,开发了 DCCP 作为下一代用户数据报协议;为了更加灵活地支持各种数据流的要求,开发了 SCTP 作为下一代 TCP 协议等,这些都是对相关层次协议进行更新或者补充,维护了整体结构的稳定性.(3) 提出新的网络处理框架思想.如综合服务模型[1]、区分服务模型[2]和主动网络技术[10]的思想;这些技术主要体现在网络层,但是也需要其他层次的协作,要相应修改或者增加各个层次的有关协议,从而导致 TCP/IP 协议栈变得日益复杂.

层次网络体系结构虽然便于实现,但随着网络技术的发展,网络协议不断增加,导致其网络功能冗余,效率降低且改进越来越困难.通常,因特网每增加一个新的应用需求,就需要增加或修改系列协议来满足这个要求,这使得 TCP/IP 协议栈越来越庞大,目前已有 100 多个协议.因此,层次网络体系结构已经越来越不适应新的网络需求,难以承担下一代高性能网络体系结构的重任.

从 20 世纪 90 年代开始,提高网络性能的研究已经从各个层次协议的修补转变到研究新型网络体系结构方面,以便更好地适应网络通信的新特点.根据前面的分析,新型网络体系结构应该具有如下几个方面特性:(1) 高效的网络处理,基本上没有冗余功能;(2) 便于保证各种 QoS 要求;(3) 支持网络安全需要;(4) 良好的可扩展性.

网络体系结构方面的研究已经成为下一代网络应用研究的一个重要内容,能否充分满足上面的要求,将是

一个新型网络体系结构能否成功的关键.应用级组帧(ALF)[3]和一体化层次处理[4]认为,传统的 TCP/UDP 并不能很好地满足每一个特定的网络应用需要,应用程序应该涉及数据传输处理过程,让应用层来完成各种复杂的数据处理.虽然这种方案增加了灵活性,但也增加了应用程序的实现难度,难以推广.面向对象的网络体系结构[6]以更加灵活的方式来满足不同网络应用的服务要求,将网络功能分解为不同的网络功能模块(即对象),系统模型将各个服务对象按照水平方向分为各个平面,在垂直方向分为各个功能层次,不同的层次完成不同的通信子功能,因此它仍然是一个基于层次思想的体系结构.目前,面向对象的 MCS 的两个公开问题是模块化准则和协议正确性的自动验证问题.基于角色网络体系结构(RBA)[7]则完全去掉了协议层的概念,取而代之的是称为角色(role)的功能单元来组成通信系统.协议模块称为一个角色(role).角色是对一个通信模块的功能性描述.角色并未按层次来进行组织,因而角色之间的交互作用将比传统的协议层次要丰富得多.一个报文中所有的数据,包括载荷,都是被分为 RSH(特定角色报头)的角色数据.目前,基于角色的研究还停留在概念上,需要进一步的探索.

服务元网络体系结构(SUNA)[8]作为最新的非层次体系结构综合了上述研究的优点,提出以服务元(SU)来定义网络的基本功能单元.在上述研究中,对象的功能定义得太细,组合处理过于复杂;而角色的功能过于复杂,实现困难.文献[8]比较详细地定义了服务元的特征.其关键点在于服务元功能独立,没有交叉.具有特定通信要求的网络应用根据功能选择不同的 SU 组成特定的服务元团队来实现网络服务.

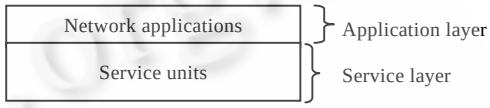

Fig.1 SUNA node module

图 1 SUNA 节点模型

SUNA 中定义网络系统由若干节点构成.网络上的主机和路由器统称为节点.SUNA 系统一般节点模型如图 1 所示.一个节点模型分为两层:应用层和服务层.应用层是服务的接受者.服务层是 SU 集合,它除了向本节点应用层提供服务以外,还与其他节点的 SU 合作向整个网络提供服务.

文献[11]根据 SUNA 的思想,提出了一种实现框架——微通信元系统架构(MCES),详细地阐明了 MCES 架构的构建原则、节点模型以及若干服务元的定义.MCES 架构根据现有 Internet 系统结构的特点,将整个网络分为主机部分和网络核心部分.图 2 是 MCES 架构的系统模型.图 2 用不同形状的图形表示不同类别的 SU.虽然 SU 作为网络的基本功能单元并没有规定其层次,但由于功能和作用的限制,某些 SU 只能在特殊地方起作用,如进行网络接口处理的 SU 只能在链路相接的地方.

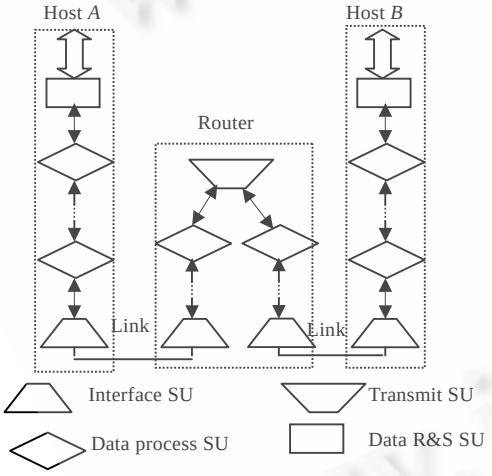

Fig.2 MCES module

图 2 MCES 系统模型

作为一个框架,MCES 并没有详细定义 SU 及其如何提供网络服务的方法.本文利用 MCES 架构思想设计、实现了一种基于服务的网络系统原型.该原型从体系结构上对当前网络面临的两个重要问题——服务质量和网络安全进行了研究.进一步研究了如何利用 SU 组合成网络服务,让各种数据流得到合适的网络服务处理的过程.处理不同类型数据流的 SU 按需组织,SU 之间没有固定的位置关系.我们将这种按需组织网络服务的网络系统称为基于服务的网络系统(SBNS).

2 SBNS 中的服务元组织与调度

2.1 服务元及其服务组合

在非层次体系结构中,如何组织功能单元组成需要的网络服务是一个关键.这个问题在层次网络体系结构中并不存在,因为所有的网络功能都严格按照层次关系进行了组织,不需要临时按照网络功能重新组合.

(1) 服务元的定义

要定义服务元,需要对网络功能有系统的认识.文献[12]中描述了网络的功能为顺序性、可靠性和时间特性组成的三维模型.实际上,网络安全应该是当前网络的基本功能之一;而顺序性和可靠性可以合并,它们具有共同的目标.因此,在 SBNS 中把网络基本功能定义为 4 种,即可靠性、时间特性(QoS)、安全性及其他(路由转发等).

定义(服务元). 服务元是一个独立功能单元,只对网络提供服务而不需要得到其他服务元的服务.

服务元具有独立性和网络功能完备性.可以用二元组表示服务元的属性,即〈Process,Function〉.其中,Process 定义了服务元的处理特征,Function 定义了服务元的功能特征,分别用一个字节表示.具体而言,对于 Process 属性,每位表示一个服务元具有的操作属性,即支持发送、支持接收、有状态、有特征首部、置于底部、置于顶部,置于中间、保留.这些属性对于一个服务元是否处于一个服务的适当位置具有重要意义.比如一个 NIC(网络接口)服务元需要将置于底部位置位、支持发送位、支持接收位和有特征首部位都置 1,表示这个服务元只能用于与链路相接的处理位置,并且有自己的特征首部,支持发送和接收处理.

0	1	2	3	4	5	6	7

0 0 : Reliability

0 1 : Time (QoS)

1 0 : Security

1 1 : Other (routing information)

Fig.3 Function attribution

图 3 Function 属性

对于 Function,其各部分含义如图 3 所示.

前 2 位表示大功能类,分别是可靠性、时间特性、安全性及其他;后 6 位表示这个大类中的子功能类.如安全性大类中可有认证、数字签名、加解密等不同功能服务元.

为了便于识别和管理,服务元的属性码由网络系统统一管理,便于不同服务元被各网络系统正确使用.

(2) 微服务的定义

单独的服务元不能提供完善的网络功能,比如发送或者接收一个网络报文的处理.

定义(微服务). 多个服务元一起协作完成一个完整的发送或者接收网络信息的功能称为微服务.

一个微服务能够组织若干服务元,按照一定的次序,对服务数据单元(SDU)进行处理,然后送给下一个节点或者网络应用程序.它有两种基本类型,即发送微服务和接收微服务.路由器转发处理是特殊的接收微服务.

(3) 组合服务

网络服务实际上是在不同的节点上对各种报文进行变换、转发、复制或者删除等操作.令 P 表示网络报文的集合,S 为网络微服务的集合,则一个节点 M 上的网络服务 E_M 可以表示为一个二元关系:

$$E_M=\{\langle x,y\rangle|x,y\in P\wedge x\ S\ y\} \tag{1}$$

在一个源节点 S 中,SDU 接受一个发送微服务 S_s,即 xS_sy;在经过系列路由器节点 $r_1,r_2,...,r_n$ 时,报文接受一系列转发微服务 S_r,即 $yS_{r1}z_1,z_1S_{r2}z_2...z_{n-1}S_{rn}z_n$;在目的节点,报文得到接收微服务 S_d,即 $zS_dw.x,y,z,w\in P;S_s,S_r,S_d\in S$.当一个报文从源主机传送到目的主机时,将受到各个节点的微服务合成服务,即

$$xS_sy\wedge yS_{r1}z_1\wedge z_1S_{r2}z_2\wedge...\wedge z_{n-1}S_{rn}z_n\wedge zS_dw,$$

令 $S_r=S_{r1},S_{r2},...,S_{rn},S_p$ 表示报文的传输服务关系,则有

$$S_p=\{\langle x,w\rangle|\exists y,z(xS_sy\wedge yS_rz\wedge zS_dw)\} \tag{2}$$

式(2)表示一个报文接受的网络传输服务,包括源主机的发送微服务、各路由器的转发微服务和目的主机的接收微服务,这些微服务综合起来满足网络传输的总服务集合 Service.其中,y,z 表示具体链路传输报文形态.对于数据传输,通常要求 $x=w$.当收发双方的链路相同时,$S_d=S_s^{-1}$,表明目的主机的接收微服务为源主机的发送微服务的逆微服务.当收发双方链路不同时,逆微服务的不同仅体现在有关链路操作方面的服务元不一样.

每个微服务由若干个服务元组合而成.以源节点的发送为例,假设组成发送微服务的服务元分别为 $F_1,F_2,...,F_n$,则 $S_s=F_1,F_2,...,F_n$;如前所述,同一类型的报文接受同样的微服务,报文经过一个微服务的处理,实际上是经过一系列固定服务元的处理.按照函数的定义,一个发送微服务的处理可以用一个函数关系来表示:

$$S_s(p)=f_1(f_2(...f_n(p))) \tag{3}$$

式(3)中,p 表示节点中需要网络服务的报文,S_s,f_1,f_2,...,f_n 分别表示发送微服务及其各服务元的功能函数.各个服务元对 p 加工的顺序决定了 f_1,f_2,...,f_n 的嵌套顺序.在实际处理过程中,某些服务元的处理有一定的位置要求,如 NIC 服务元一般是发送处理最后作为 f_1 存在.根据前面分析,一个报文的目的节点对应接收微服务处理应该是

$$Sd(p)=f_n^{-1}(f_{n-1}^{-1}(f_{n-2}^{-1}(\cdots f_1^{-1}(p)))) \tag{4}$$

易证,微服务是双射的.收发微服务特征提供了简单的方法来组合微服务,即接收方可以根据发送方的服务元序列进行逆处理而得到接收处理微服务.当然,有些服务元可能综合了几个服务元的功能,如 $F_m=F_i\cdot F_j$,则需要进行对应功能分解推理.

微服务的实现主要采取两种办法:一是系统缺省定义一系列微服务,能够满足网络系统提供的基本功能;二是用户通过定制服务元并定制微服务,满足用户的特殊要求.

为了正确使用微服务,同样需要对微服务属性进行描述.微服务属性有如下内容:

- 功能代码:描述微服务可以实现的功能,分为两部分:一是功能类别,主要有建立连接、关闭连接、发送数据、接收数据等;二是描述功能的细节,即微服务对网络功能维度的支持.
- 运行状态代码:描述微服务在什么时候运行,主要有初始态和建立连接态.
- 服务元向量:实现此功能需要的 SU 标识符,按顺序对 SDU 进行处理.

2.2 服务元管理器(SUM)

在 SBNS 中,最终的网络服务由各个彼此独立的服务元完成.虽然服务元已经按照微服务的形式进行了有效的组织,但是还有很多服务元的管理工作需要处理,我们用 SUM 来统一管理所有的微服务和服务元.

SUM 的主要功能如下:

1) 服务元的注册和注销.各个服务元作为独立单元模块,可以动态卸载和加载,提供了很好的灵活性.

2) 检查微服务合理性.微服务由一系列有序服务元构成,各服务元是否存在、是否处于合适位置等都需要检查.

3) 微服务的注册、注销和修改,便于用户动态调整微服务的内容,主要是为了满足用户要求定制的微服务.

4) 调用微服务和服务元.进行 SDU 调度处理,完成要求的网络服务功能.

SBNS 中网络功能的增加,完全取决于系统中不同功能的 SU 数量和由此构造的各种微服务的多少.如果一个设备网络功能简单(如手持网络终端),则只需要注册基本功能的服务元,定制基本要求的微服务;若需要比较复杂的网络功能(如网络服务器),则注册要求功能的各服务元,并定制相关的服务.并且,今后还可以不断开发新的服务元,提供新的网络功能.因此,SBNS 在网络功能方面体现了良好的可扩展性.

2.3 SUM调度模型

SUM 调度服务实现模型如图 4 所示.虚线部分表示主机节点特有的 Socket 和应用部分.在 SBNS 中,SUM 实际上是网络系统管理器,所有网络系统服务都需要通过 SUM 来实现.主机应用程序通过套接字 API 调用 SUM 来得到要求的服务,而路由器节点和主机的接收处理服务则由接收到的报文驱动 SUM 提供网络服务.对于接收服务请求,需要先通过与接口有关的预处理.这是由于 SUM 需要先通过分析报文得到基本首部的虚电路号和报文类型等信息,然后才能得到相关处理服务.功能由各节点的接收预处理服务模块完成.

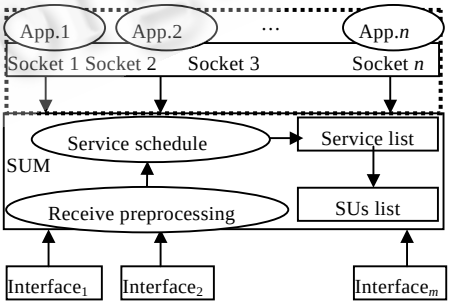

Fig.4 SUs schedule model

图 4 服务元调度模型

SUM 是保证 SBNS 服务质量的关键.SBNS 中所有发送和接收的数据包都是 SUM 统一调用适当微服务来进行的,这样便于 SUM 采用合理调度策略控制 QoS.而在层次体系结构中,不同数据流各自独立选择协议栈处理,难以进行统一调度.虽

然有研究[13−15]在应用层或网络接口处增加了报文调度处理以提供必要的 QoS,但在处理上缺乏一致性,而且难以与流量控制和拥塞控制相结合.

2.4 SUM调度分析

在层次网络体系结构中,主机应用程序难以对各个数据流进行有效的调度处理.相对来说,路由器由于是接收驱动处理,在处理调度上要方便得多,因此,当前 Internet 上对路由器进行 QoS 控制研究比主机要多.随着网络应用的不断扩展,特别是网络服务器的负担越来越重,主机节点的 QoS 控制将是一个重要的问题,虽然有大量的研究[20−22]是专门针对主机的 QoS 资源管理的,但在 TCP/IP 体系下仍然没有找到非常有效的方法.而 SUM 的服务处理调度模块对各个节点的各种服务要求都可以进行合理调度,对于具有 QoS 要求的网络应用具有重要的意义.

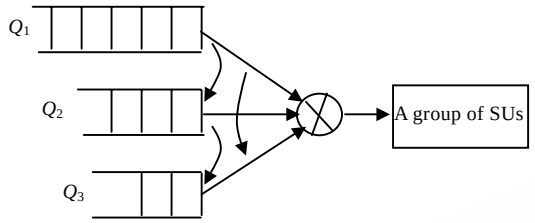

Fig.5 SUM scheduler

图 5 SUM 调度器

当前,Internet 主机节点的服务调度实际上是一种 FCFS 调度,在 CPU 资源紧张的服务器主机中很难保证关键网络应用的 QoS.SBNS 采用了一种基于资源预留的动态优先级(SRBDP)调度方法,如图 5 所示.

SUM 假定一个网络数据流的发送必须满足如下条件:本地带宽允许(本地接口管理)、网络带宽足够(拥塞控制),接收方同意(流量控制)、预留资源满足(本地调度).故 SUM 对要处理的数据流组成 3 个队列.Q_1 表示在预留带宽范围内的数据类型,具有最高优先级;Q_2 表示已经用完预留带宽资源,但是可以使用剩余带宽的流量类型,处理优先级次之;Q_3 表示已经用完预留带宽资源,并且其经过的路径已经没有多余的带宽,但是作为时间敏感数据又必须发送出的数据流量,其优先级最低.这样做的基本思想是让预留带宽资源的数据流优先得到保证.为了使各个队列中的数据处理均匀而且公平,SUM 采用队列内循环处理机制.这样保证在一个循环周期的各个数据流都能够发出数据.对于数据量比较小的或者申请带宽资源不多的套接字,会因为数据处理完成或者令牌不够而提前退出队列.相应地,其他数据流可以得到更多的服务时间.SUM 还维护了一个等待队列 Q_0,对于某些数据流,如果处理它们的服务元含有流量控制或者拥塞控制功能,则要求发送方阻塞发送时,SUM 将其放入等待队列,直到阻塞条件取消,继续置于相应的发送队列发送.

在有大量数据需要处理(如服务器或者核心路由器上)时,将产生报文等候处理的时延.另外,不当的处理过程会造成短时间内数据突发,导致网络拥塞,影响数据流的传输特性.SUM 的调度处理有效地减缓了这两个影响 QoS 的不利后果.下面通过对比常见的 FCFS、纯优先级(PQ)调度来分析 SRBDP 调度算法性能.

假设共有 n 个数据流在等候处理,每个数据流有 d_i 个字节的数据需要处理,节点的报文处理速度为 r byte/s.同时还定义 N 为一个 SDU 的大小.表 1 显示了 3 种方式的对比.

SRBDP 中的 n_1 表示处于 Q_1 队列中的数据流个数.表 1 显示 SRBDP 响应时间在高优先级时,可能比纯优先级调度要长,但在低优先级时,相应时间要短很多,但都比 FCFS 要好.同时,可以有效地保证资源要求,并且尽量减轻数据流的突发.由于结构上的优势,SUM 还很容易实现其他更加公平、有效的调度算法,这对改善网络节点,特别是主机节点的 QoS 要求具有明显的作用.

Table 1 Contrast among three scheduling methods

表 1 3 种调度方式对比

Scheduling	FCFS	PQ	SRBDP
Shortest response time (t)	$\sum_{i=1}^{n} d_i / r$	$N/r \sim \sum_{i=1}^{n} d_i / r$	$n_1 \cdot N/r \sim \sum_{i=1}^{n_1} d_i / r + (n-n_1)N/r$
Effect factor	Order	Priority	Requested bandwidth
Burst	Yes	Yes	Less

2.5 SBNS实施模型及分析

网络系统是复杂的系统软件.虽然 SBNS 系统力图简化网络处理,提高效率,但仍然具有比较复杂的软件结构.进程代数[16]是用来解决复杂并发系统通信问题的代数方法,具有严密定义的形式化语义,能够将软件行为与操作语义很好地联系起来,进行抽象分析和验证.这里,利用进程代数对 SBNS 的实现模型进行了描述和分析.

为了能够合理地描述 SBNS 的实现模型,首先对 SBNS 的处理过程进行抽象,如图 6 所示.一个网络系统就是数据流的服务处理系统,在主机上是收发处理,在路由器上是转发处理.图 6 中,3 种类型数据为整个系统的输入,其中Δ_1 表示主机发送的数据,Δ_2 表示接口收到的数据,Δ_3 表示处理过程中产生的数据.所有不同来源的数据经过不同的预处理 P_1,P_2 或者 P_3 处理后,进入缓冲区 Buff 等待 SUM 调度处理.输出可能是主机的一个套接字,也可能是路由器的另一个接口.

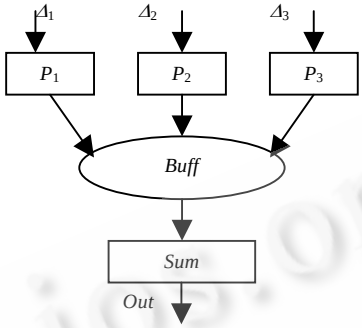

Fig.6 SBNS implement abstraction

图 6 SBNS 实施抽象

首先描述 SBNS 的系统实现模型:

$$System \overset{\text{def}}{=} (P_1 \parallel P_2 \parallel P_3)^{\infty}_{\{get,add,rel\}} Buff^{\infty}_{k\{get,del,rel\}}(Sum),$$

式中∞表示协作行为,{*get*,*add*,*rel*}和{*get*,*del*,*rel*}表示需要协同操作的动作.因为缓冲区资源分别被各个预处理和 SUM 所访问,属于临界资源,需要同步机制.P_1,P_2 或者 P_3 之间使用||表示需要并发执行.

P_1 表示主机发送的数据处理,即

$$P_1 \overset{\text{def}}{=} \Sigma_{d\in\Delta_1} packet_i(d).write_i(d)P_1.$$

式中 *packet*(*d*)表示对数据进行分包处理,而

$$write_i(d) \overset{\text{def}}{=} (get_i, g).(add, d).(rel, r)$$

表示 $write_i(d)$将分包的数据置于 *buff* 中的合适队列中等候 SUM 的调度和处理.

P_2 属于接收报文的预处理,先进行组包处理,然后按照一定的策略将接收到的报文写入缓冲区即可.

$$P_2 \overset{\text{def}}{=} \Sigma_{d\in\Delta_2} Nic_i(d).write_i(d)P_2.$$

Nic( )表示根据接口组包的处理.

P_3 处理的是超时重发或者某些服务元根据处理协议而要求产生的报文,比如拥塞报告、可靠确认等.这些报文已经产生,故

$$P_3 \overset{\text{def}}{=} \Sigma_{d\in\Delta_3} write_i(d)P_3.$$

Buff 缓冲区是 3 个不同类型的数据共享,由 SUM 读出处理.为了公平,每次缓冲区记住最近的访问者,避免一个访问者长久占用缓冲区.其定义如下:

$$Buff_i \overset{\text{def}}{=} \sum_{\substack{j=1\\ j\neq i}}^{3} (get_j, T).(use, T).(rel, T).Buff_j.$$

如前所述,*Sum* 是 SBNS 的重要组件,包含配置管理、调度和服务处理等功能,故我们定义:

$$Sum \overset{\text{def}}{=} (Config_1 \| (Schedule.MicroSrv)^{\infty}_{\{operate\}} SU_k.$$

Sum 中的配置组件 *config* 实现服务元的加载、移出、微服务的定制及数据流的服务映射等处理,它与 *Sum* 中的其他组件并发操作,操作时需要同步服务元的集合 *SU*,故

$$Config_i \overset{\text{def}}{=} \Sigma_{s\in\Omega,m\in\Phi}(operate_t(s,m)).(load_i(s)+remove_i(s)+customize_i(m)+map_i(m)).Confeg_i.$$

式中Ω和Φ分别表示服务元和微服务的集合.运算符"+"表示选择.因为服务元和微服务的各种操作不能并发进行.在 *Config* 组件的处理中,服务元的加载、移出是人工干预的,而微服务的定制可以人为地也可以根据服务元和微服务的属性以及相关规则实现自动配置.映射处理是将某些类型的报文处理映射到某个特定的微服务和调度策略上.在主机发送数据时,通过调用 *connect* 套接字函数(见第 4.3 节)可以建立映射关系;而在路由器上则是通过建立连接报文建立这种映射关系,这类似于主动网络中通过报文定制服务的处理.*Config* 组件极大地增强了 SBNS 系统的灵活性和可动态配置的特点,更容易支持扩展的网络服务.

调度组件 *Schedule* 从待发的数据报文Δ的缓冲区中选择一个合适的报文进行网络服务:

$$Schedule_i \overset{\text{def}}{=} \Sigma_{d\in\Delta}(Controler||read_i(d)).Schedule_i.$$

$$read_i(d) \overset{\text{def}}{=} (get_i,g).(del,d).(rel,r).$$

Schedule 中 *Controler* 根据报文及其处理数据流的情况排定调度器应该处理的报文.如前所述,控制器组件综合考虑报文申请的资源、网络资源(拥塞控制)、接收方资源(流量控制)和本地接口带宽等限制,动态决定各报文处理优先级,同时,在主机中对所有数据流采用统一控制机制,克服了传统层次体系的不足,可以更好地保障 QoS.

MicroSrv 组件是使用 SBNS 中定制的各种微服务对报文进行处理,其定义如下:

$$MicroSrv_i \overset{\text{def}}{=} \Sigma_{d\in\Delta}(Ms_1(d)+Ms_2(d)+\ldots+Ms_j(d)).MicroSrv_{i.}.$$

Ms_j 是各个具体的微服务,一个报文一次只能是一个微服务处理,因此对每个报文的处理,*MicroSrv* 组件选择一个合适的 Ms_j 进行处理.一个 Ms_j 是一系列有序的服务元的组合处理,因此,

$$Ms_j \overset{\text{def}}{=} \Sigma_{d\in\Delta}(Su_1(d),Su_2(d),\ldots,Su_i(d)).$$

在 SBNS 系统中允许存在一系列相似的微服务和服务元.假设 $Ms_k \overset{\leftrightarrow}{=} Ms_k'$,令

$$MicroSrv_i' \overset{\text{def}}{=} \Sigma_{d\in\Delta}(Ms_1(d)+\ldots+Ms_k(d)+Ms_k'(d)+\ldots+Ms_j(d)).MicroSrv_i'.$$

由进程代数定理可知,$MicroSrv_i$ 和 $MicroSrv_i'$ 也是互模拟的,即 $MicroSrv_k \overset{\leftrightarrow}{=} MicroSrv_k'$.

与此类似,假设 $Su_k \overset{\leftrightarrow}{=} Su_k'$,则

$$\begin{aligned} Ms_j' &\overset{\text{def}}{=} \Sigma_{d\in\Delta}(Su_1(d).Su_2(d),\ldots,(Su_k(d)+Su_k'(d)),\ldots,Su_i(d)) \\ &=(\Sigma_{d\in\Delta}(Su_1(d).Su_2(d).\ \ldots\ .Su_k(d).\ \ldots\ .Su_i(d)))+(\Sigma d\in\Delta(Su_1(d).Su_2(d).\ \ldots\ .Su_k'(d).\ \ldots\ .Su_i(d))) \\ &=Ms_j+Ms_j'', \end{aligned}$$

显然,Ms_j 和 Ms_j'' 是互模拟的,因此 $Ms_j' \overset{\leftrightarrow}{=} Ms_{j.}$.

这表明,在SBNS系统中可以同时存在多个功能相似的服务元和微服务,这对改善某些协议算法具有重要意义.比如在有线和无线环境下,拥塞控制的机制差别很大,我们可以设计互模拟的服务元和微服务,分别处理不同环境下的数据流,达到最好的网络性能.这也从另外一个方面显示了 SBNS 系统的可扩展性特点.

3 SBNS 的系统设计

为了充分发挥 SBNS 的优势,我们在网络管理机制、报文格式和套接字结构等方面采取了有效的设计措施.

3.1 增强网络管理机制

最初 Internet 强调网络互通,缺乏考虑的网络 QoS 和网络安全日益成为突出的问题.因此,新型网络系统的设计必须提供有效的网络管理机制,才能满足日益增长的 QoS 和网络安全的需要.采用类似虚电路服务的有管理的网络服务是提供 QoS 和网络安全的有效方案.通过这种机制,不仅可以协商网络通信需要的带宽、缓冲区和 CPU 资源,还可以进行访问控制、服务能力检查以及服务资格检查等,极大地提高了网络管理能力.其中,必要的访问控制提高了网络使用的安全性.在 SBNS 系统中,我们要求在网络边缘和目的主机都进行安全控制和资源检查(或分配),确保系统访问的安全性.这个过程称为建立通信连接.

用一个集合 *Service*={*R*,*S*,*N*,*Q*}表示建立通信连接的各种要求.其中 *R* 表示通信资源,如链路带宽、缓冲区和 CPU 资源.*S* 表示网络服务要求,如可靠服务、实时服务等.*N* 表示访问控制,一般在边缘路由器上检查用户是否可用传输网络.*Q* 表示目的主机服务资源的访问控制.各节点检查相关的要求,符合要求就传递到下一个节点继续检验,否则,将禁止建立通信连接的信息反馈给源主机.一般每个节点都需要对 *R*,*S* 的要求进行检查.而边缘路由器可能还需要检查 *N*,目的主机可能还需要检查 *Q*.

SBNS 中针对每个网络应用数据流采用端到端的虚电路机制.建立通信连接的过程用 4 次握手的双向虚电路机制,如图 7 所示.

源主机 *S* 首先向目的主机 *D* 发起一个通信请求①,根据通信要求将适当服务置于集合 *Service* 中.边缘路由器 *R*1 通过访问控制后根据网络状况采取源选径方式选择一条合适的路径,此后每个经过的节点检查自己关心的网络资源要求是否满足.如果满足,则建立一个虚状态并传递请求到下一站.否则,返回一个否认报文给 *S*,撤销以前建立的状态.*D* 收到 *S* 的请求,若一切条件满足,同样向 *S* 发起一个通信请求②,建立回传数据的通道.此时,建立的双向数据通道都是虚的,并没有真正预留网络资源,只是表明要求的资源可用.当 *S* 和 *D* 分别向对方传递第一个数据时,才真正获得要求的资源.如图 7 中的③④所示.

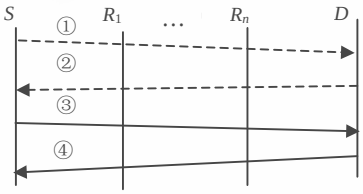

Fig.7 Creating communication connection

图 7 建立通信连接过程

3.2 安全、高效的报文格式

报文格式决定了网络系统的工作方式.SBNS 中采用两大类报文类型,即控制和数据报文,便于网络系统更加合理地组织服务元提供相应的服务,也利于控制数据流的 QoS 和安全要求.

(1) 基本首部

所有的报文都有统一的基本首部,如图 8 所示.

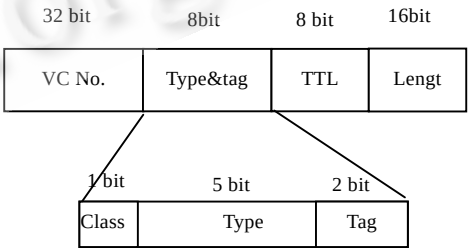

Fig.8 SDU base header

图 8 SDU 基本首部

在图 8 中,基本首部包含 6 个字节.其中虚电路号 4 字节,一字节防止循环的 TTL,两个字节报文长度,另加一个字节类型标记.虚电路号既是源端到目的端的虚电路标识符,又是网络节点寻址和处理的索引.与数据报方式的地址索引相比,大大节省了报文首部空间,同时减少了网络处理压力.虚电路号反映了某个源-目的地址对之间的报文,这条路径上可能存在多个报文类型,需要不同的服务来进行收发处理,因此,利用一个 6 位的类型域进行区别,低位为 1 时表示是控制报文,低位为 0 表示数据报文.随后 5 位表示报文的具体类型.

报文类型对应唯一的发送和接收处理服务.根据文献[12]的研究,基于报文类型的处理将大幅度地提高处理效率.网络通信过程中的大量报文是数据报文,报文首部的虚电路号让所有路由节点快速进行转发处理;而报文类型域实现了主机服务的快速索引处理,大大提高了网络传输处理效率.

最后两位是标志位,这些标记的产生类似于 RFC2859[18]中的规定,便于路由器在转发过程丢弃报文时进行区别对待.其中,00 表示在预留资源范围内报文,01 表示在拥塞控制许可范围内报文,11 表示超过允许的报文.在 Internet 中标记在边缘路由器上,而在 SBNS 中,标记过程在主机上,不仅减轻了边缘路由器的负担,而且主机更容易知道哪些报文是可丢弃的.

(2) 控制首部

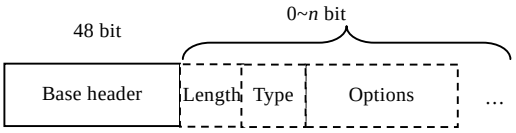

Fig.9 Building and release connection packet format

图 9 建立和释放通信连接报文

根据 SBNS 的管理需要,网络通信之前需要源和目的主机以及与网络传输系统之间协商各种服务要求,因此要建立和释放通信连接报文,其格式如图 9 所示.

如前所述,建立通信连接的过程实际上是对网络资源和网络服务的申请过程.不同的网络数据流可以有不同的网络要求,因此报文采用选项的形式,每个选项由长度、类型和协商首部构成.各节点找到自己需要的协商选项进行处理.SBNS 中协商选项主要有如下几种:

- 地址选项:包含源主机地址、目的主机地址、源端口和目的端口.
- QoS 选项:要求网络传输系统的 QoS 支持,主要有预留带宽、优先级、延迟要求等.
- 加解密选项:可以定义指明某些选项采用了要求的加密处理,确保关键选项的安全性.
- 网络访问控制选项:主要用于用户认证,进行网络接入控制.
- 目的主机访问控制选项:用于目的端验证用户是否具有访问资源的资格.
- 服务选项:用于定义用户要求的数据传输服务是否能够满足.

建立通信连接的各选项对于保证系统支持 QoS 和网络安全具有重要意义.除了比较明确的 QoS 协商和安全认证以外,服务选项也是保证网络安全的措施之一.采用服务选项,用户可以自定义数据类型和服务的映射关系,即使其他用户捕获到传输的数据报文,也很难进行合适处理.

释放通信连接报文一般无选项,由基本首部虚电路号和数据类型可以合理地释放资源.

(3) 数据报文格式

数据报文如图 10 所示.前面是若干可选的服务元首部,紧接着是传输数据.服务元首部根据服务所使用服务元功能的不同而不同.数据报文类型决定了处理这个数据报文的服务元序列.基本首部的前 40 位(虚电路号加类型标记域)相当于一个数据流标识,决定了一个数据流的传输和处理方法,也决定了这个数据的 QoS、网络安全和可靠性等网络服务特性.SBNS 较好地实现了数据流驱动,从体系结构上能够更好地支持各种网络服务的需要.同时,SBNS 在传输数据时只有一个标识符,没有更多的指示说明其处理过程,相对于目前基于数据报的处理机理,更具有结构上的安全性.

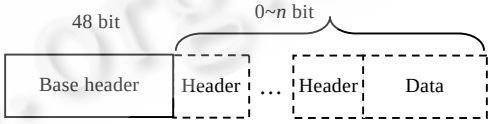

Fig.10 SDU data packet format

图 10 SDU 数据报文

3.3 改进的套接字结构

一个节点网络服务由若干微服务群来完成,图 11 显示了主机节点中一个网络应用到服务元的映射过程.

Fig.11 Map of user request to SUs

图 11 用户网络服务请求到服务元的映射

Socket 是操作系统提供的网络 API 统称,最早用于 TCP/IP 在 UNIX 中提供的网络编程接口,目前已经是事实上的网络编程标准.本系统允许用户定义服务元来扩展网络功能,极大地方便了网络应用的升级.但目前 Socket 并没有很好地适应这种技术的快速发展.在 SBNS 中,Socket 既是网络应用和网络系统的界面,又是用户请求服务到网络提供服务的映射组织系统.构建合适的 Socket 是 SBNS 系统中主机节点的一个重要内容.

近年来,为了支持 QoS,有许多基于 BSD 套接字的改进研究.如 Sockets++[19],QoSockets[20],QSockets[21] 以及 Windows 扩展的 Socket 接口 Winsock2[22]等,但这些研究除了改进对 QoS 的支持以外,在面向更一般的网络服务方面显得不足.而对于 SBNS 提供端到端数据流的处理来说,更缺乏有力的支持.GSocket[23]机制充分考虑了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,并尽量兼容传统套接字编程.按照 GSocket 思想,扩展了套接字的功能,如图 12 所示.

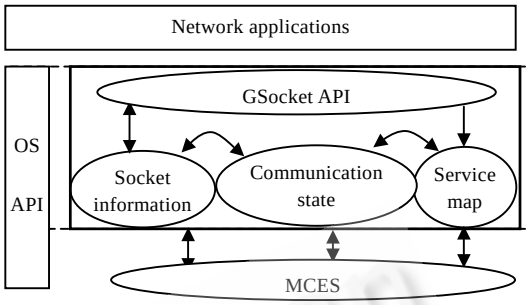

Fig.12 GSocket implement model
图 12 GSocket 的实现模型

GSocket 包含 API 接口处理、套接字信息模块、通信状态模块和服务映射模块.其中 API 接口处理、套接字信息模块兼容了现有网络系统,也便于兼容现有应用程序.服务映射模块(SM)将用户服务请求映射到系列微服务中去.应用程序通过扩展的 Socket( )函数提出网络服务请求传递给 SM,SM 检查系统要求的微服务群是否存在,并检查相应的服务元是否存在,将结果存放在套接字有关信息中.通信状态模块(CS)负责跟踪本套接字通信过程,记录当前通信状态,保证系统调用合适的服务来传递 SDU.每个套接字具有建立连接、数据传输和释放连接 3 个状态,同时,不同服务元也可能保存有自己的内部状态信息,如保障可靠通信的滑动窗口服务元需要保序的 SDU.为适应这些要求,CS 建立有 SU 按需设置状态机制.

4 SBNS 的原型实现

为了检验 SBNS 的性能和可行性,我们在 Linux 系统上实现了 SBNS 原型.原型设计从功能上借鉴了 TCP/IP 的经验,实现了当前因特网的主机节点和路由器节点的类似功能,但是从体系上完全是一个非层次的新结构.同时,根据 SBNS 便于调度和组合的特性,还提供了一些高级的网络服务.原型系统提供了如下几种网络服务功能:

- 不可靠服务.提供类似于 UDP 的不可靠数据传输服务.
- 媒体流服务.提供类似 DCCP 的带拥塞控制的不可靠服务,充分利用 SBNS 便于服务组合的优势.
- 可靠传输服务.提供类似于 TCP 的可靠传输服务.
- 紧急数据服务.提供高优先级的可靠传输服务,保证带宽和避免丢失.利用 SBNS 便于调度优势.

4.1 服务元的设计

为了实现上述服务功能,原型系统的服务元分为 4 类,即虚电路管理、传输管理、路由管理和基本收发管理.原型系统实现的基本服务元见表 2.

Table 2 Base SUs of prototype system
表 2 原型系统的基本服务元

Name	Function
CLSU	Creating communication link
RLSU	Releasing communication link
KLSU	Maintaining communication link
RSSU	Reliable transportation
CCSU	Congestion control
TMSU	Flow control
NICSU	Network interface process
CRSU	SDU forward process
SRSU	Source routing
RRSU	Reserving resource, such as bandwidth

其中,虚电路管理包括 CLSU,RLSU,MVCSU.采用建立连接的通信模式是现代网络通信的要求,实现了资源的可管理性.传输管理借鉴了 TCP 的实现,但将 TCP 的可靠传输、拥塞控制、流量控制分别组成服务元,可以提供更加灵活的类似于 DCCP 或者 SCTP 的功能.基本收发管理包含 NICSU,CRSU.这两个服务元是所有节点进行网络通信必不可少的.在路由管理服务元中,SRSU 用于建立连接时在边缘路由器进行选路,而 RRSU 用于建立连接时的资源预留处理,实现类似 RSVP 协议的功能.在原型网络中,为了简化设计,采用静态路由控制,从而没有实现自适应路由处理的有关功能.

4.2 微服务的设计

微服务决定了一个节点中 SDU 的完整处理过程.原型系统的基本微服务见表 3.表中微服务可以分为两部分,其中一部分用于主机节点,如前面 6 个微服务;另一部分用于路由器节点,如后面 4 个微服务.由于路由器上的报文需要转发,而主机节点是起止节点,因此不同节点上,同样的微服务所组成的服务元不同.

Table 3 Base micro-services of prototype system

表 3 原型系统的基本微服务

Name	SU's sequence	Function
Flow transportation	(CCSU, CRSU, NICSU)	Transport media flow
Base transportation	(CRSU, NICSU)	Base data transportation
Reliable transportation	(RSSU, TMSU, CCSU, CRSU, NICSU)	Transport reliable data flow
Creating VC	(CLSU, CRSU, NICSU)	Creating communication path
Releasing VC	(RLSU, CRSU, NICSU)	Releasing communication path
Maintaining VC	(MVCSU, CRSU, NICSU)	Process a VC maintenance
Keeping VC	(NICSU, CRSU, MVCSU, NICSU)	Keeping a VC at router
Base forwarding	(NICSU, CRSU, NICSU)	Base forwarding at router
Routing VC	(NICSU, CRSU, SRSU, RRSU, NICSU)	Routing and creating VC at edge router
Reserving VC	(NICSU, CRSU, RRSU, NICSU)	Reserving resource at core router

主机节点没有对应 UDP 的不可靠数据传输服务,实际上,基本传输微服务对应了类 UDP 服务.因为 UDP 服务没有任何的服务要求,建立连接通道以后,通过基本的收发处理就可以实现最简单的数据通信.另外,所有的接收处理和路由器处理都是以基本接收服务为基础的,这是因为,一个数据链路收到的 bit 流必须通过相应链路的 NICSU 处理才能变成可以识别的报文,并且通过 CRSU 的处理,才能知道下一步需要的服务(服务元).

4.3 网络服务的实现

微服务决定了节点的服务元处理序列,也决定了特定类型的报文 SDU 及其处理和调度方式.这样,不同网络节点上的若干个微服务就可以提供一个网络应用要求的网络服务.表 4 列出了原型系统实现的网络服务在不同的节点上要求的微服务.

Table 4 Network services of prototype system

表 4 原型系统的网络服务

Network service	Micro_services at host node	Micro_services at router node
Unreliable service	Creating VC base transportation maintaining VC releasing VC	Routing VC base forwarding keeping VC releasing VC
Media flow service	Creating VC flow transportation maintaining VC releasing VC	Routing VC, reserving VC base forwarding keeping VC releasing VC
Reliable service	Creating VC reliable transportation maintaining VC releasing VC	Routing VC base forwarding keeping VC releasing VC
Urgent service	Creating VC reliable transportation maintaining VC releasing VC	Routing VC, reserving VC base forwarding keeping VC releasing VC

基本的不可靠和可靠服务完全模拟 UDP 和 TCP 的服务功能,因此没有预留资源.而媒体流和紧急数据服务分别对应不可靠和可靠的数据传输,同时预留了必要的资源并设置高优先级,确保数据及时、可靠地到达.

5 实验性能分析与评价

我们建立了相应的原型实验环境来对上面的设计进行性能分析及评价.原型节点采用 Linux2.4.20 内核.根据前面的设计方案,扩展了其套接字处理,并将服务元管理器作为一个独立模块加载到内核中以支持设计的原型系统,这样可以让 TCP/IP 系统和原型系统共存于一个网络节点中,便于对比.在原型实验环境中,使用了 5 台通过 100Mbps Ethernet 相连的 PC 机,均是 1.5G Intel Pentium 4 CPU,内存 512M.其中 3 台用作主机节点,另外两台计算机置于中间,用作路由器,其缓冲区 20KB,构成一个简单的网络,如图 13 所示.主机 *S* 和 *D* 分别作为数据传输的发送方和接收方,主机 *H* 按照要求提供背景数据流到主机 *D*.所有链路通过以太网直连,为了进行带宽限制, 我们使用了 Linux 的链路整形模块,以使各个节点输出的带宽满足设定的要求.

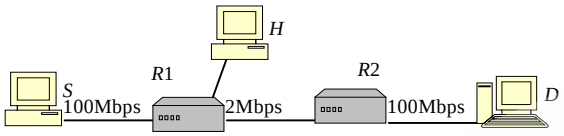

Fig.13 Network structure for experiment

图 13 实验环境网络结构

为了便于与 TCP/IP 系统进行对比,我们对开源的视频流播放软件 VLC 和文件传输软件 FTP 进行必要的修改(主要是修改 Socket 函数选择不同的网络系统),分别运行在 *S* 和 *D* 之间,以便对不可靠数据传输和可靠数据传输服务进行对比;同时自己编写了测试应用软件运行于 *H* 和 *D* 之间,提供要求的背景流量,便于对比考察不同环境下各个软件的使用效果和传输性能.

5.1 不可靠数据传输

使用网络来进行视频流的播放变得越来越普及.在传统的 TCP/IP 系统中,只能依靠增加带宽来满足视频流播放的各项要求;一旦带宽紧张,毫无控制的视频流播放的质量就变得很差.在 SBNS 系统中提供了两种不可靠传输服务,基本不可靠服务是最简单、处理最快的 SBNS 服务,类似于 UDP,而带有拥塞控制的视频流服务则提供了更好的网络控制能力,实际上只是在基本不可靠服务的基础上增加了一个拥塞控制服务元.

VLC 是一个应用广泛的开源跨平台视频播放软件.SBNS 系统中使用了 VLC0.8.2.首先,利用 SBNS 系统基本不可靠网络服务,便于与 TCP/IP 的 UDP 进行类比.让主机 *S* 作为 VLC 服务器,主机 *D* 作为播放客户机,播放速度 1Mbps 的视频流文件 30MB.主机 *H* 运行自己编写的程序,间歇性地向主机 *D* 发送不可靠数据报文作为背景流量,每间隔 1s 发送 300ms 长,速度为 2Mbps 的数据量.在主机 *S* 和主机 *D* 记录发送和接收的数据量,经过比较得出报文在传输过程中的丢失量.数据丢失量是一个随着时间变化的增函数,其结果如图 14 所示.

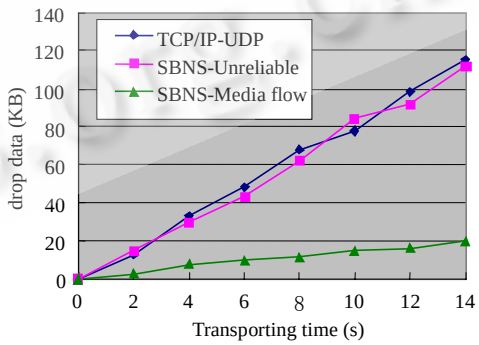

Fig.14 Lost data comparison on unreliable transportation

图 14 不可靠传输丢失数据比较

然后,使用 SBNS 系统的视频流服务.用 DCCP 的 CCID2 思想设计了一种类 TCP 的基于窗口机制的拥塞控制服务元.和前面一样进行测试得到传输过程中的数据丢失量.从图 14 中很明显地可以发现,VLC 使用 SBNS 使用基本的不可靠服务,其报文传输丢失率和 TCP/IP 很相近;但是使用了具有拥塞控制的视频流服务以后,数据丢失量大为降低,不仅有效地提高了信道传输效率,而且从播放画面上看,视觉效果也流畅多了.实际上,SBNS 系统可以方便更换拥塞控制策略,配合预留带宽和优先级控制机制,还可以方便地进一步改进视频流播放效果,这些工作将在后续的工作中深入研究.

5.2 可靠数据传输分析

文件传输服务是基本网络应用.我们构建了一个文件传输环境,主机 *S* 和 *D* 分别运行 FTP 服务器和客户机,测试过程中总是从服务器下载一个文件到客户端.其中服务器采用 vsftpd2.0.4,而客户端采用 gftp2.0.16.另外,在主机 *H* 上运行编写的受控可靠数据流软件提供背景流量,流向主机 *D*,其速度在 10s 内由 0~2Mbps 变化,但保持 1Mbps 的平均速率.考察 *S* 和 *D* 之间传输不同大小文件传输时间以及在主机 *S* 上实际传输数据量大小.

首先,使用 SBNS 的基本可靠传输服务,以便同 TCP/IP 的 TCP 数据流式服务进行比较.然后采用 SBNS 提供的紧急数据服务,并通过扩展的 setsockopt 函数设置了 1Mbps 的预留带宽,并将其数据优先级从普通的 0 提高到 1,分别比较了传输系列不同大小文件的效果,结果如图 15 所示.

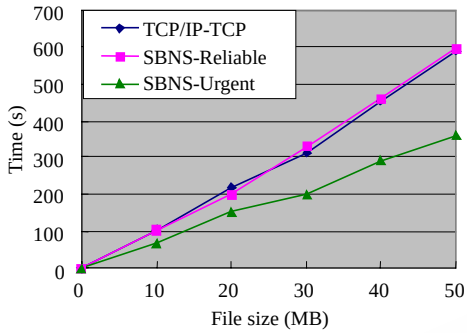

Fig.15 Time comparison on reliable transportation

图 15 可靠传输文件时间比较

从图 15 可以看到,SBNS 中的普通可靠服务和 TCP 数据流式服务具有相似的结果,因为它们都采用了类似处理(SBNS 中的可靠传输服务元、拥塞控制服务元、流量控制服务元都是从 TCP 借鉴的算法). FTP 采用了紧急数据服务后,由于 SBNS 从结构上能够很好地支持带宽预留和优先级控制,能够很好地避免背景流量的干扰,有力地保障了可靠数据传输的带宽要求和时延要求,对于未来网络的一些特殊应用具有重要意义.

5.3 系统处理性能

网络节点大量的数据处理要求系统具有良好的数据处理能力,为此,在主机节点上编写了专用网络文件传输程序,分别通过本地回环和双机直连两种情况使用.为便于对比,只用 SBNS 的基本可靠和不可靠数据传输两种服务,并与 TCP/IP 下的 TCP 和 UDP 服务对比,传输 1MB 和 100MB 文件,其传输时间统计结果见表 5.

Table 5 File transporting time on various conditions (s)

表 5 不同条件下文件传输时间(s)

File size		1MB		100MB	
		Unliable	Reliable	Unreliable	Reliable
SBNS	Loopback	0.035	0.035	27.82	2.002
	Two hosts	0.118	0.121	8.732	31.22
TCP/IP	Loopback	0.024	0.033	28.07	1.654
	Two hosts	0.098	0.114	9.675	30.32

由于采用同样的系统链路和拓扑结构,传输时间的差异反映了两个系统的处理时间差异.从表 5 中可以看出,在使用不可靠数据传输服务时,传输小文件时 SBNS 系统花费的时间稍微多些,这是因为 SBNS 下不可靠数据流也需要建立网络连接,在传输文件很小时,建立连接的时间影响了整体传输时间,而且随着链路传输时间的增加,体现得更加明显.随着传输文件的增大,建立连接的时间比例相对减小,主机节点的处理速度和链路传输速度影响了整体传输时间.在传输 100M 的文件时,SBNS 系统的表现要好些.其中本地回环效果更好,这主要是因为在本地回环时没有链路的带宽限制,完全是主机的处理速度限制了传输时间.在可靠数据传输的情况下,SBNS 系统的表现稍差.这主要是由于 SBNS 系统采用 3 个服务元使用送耦合的机制来处理可靠数据传输的问题,比较而言,TCP 将 3 种处理紧密结合,部分地提高了处理效率.

此外,通过两台主机节点直连传输 30MB 文件,分别进行可靠数据传输服务和不可靠数据传输服务,在发送方出口检查链路实际输出的数据量;比较不同系统下产生的实际链路数据量减去 30MB 后得到链路负载增量,结果如图 16 所示.图 16 表明,SBNS 系统产生的链路数据增量比 TCP/IP 要小,这是因为在 SBNS 系统的报文格式中,采用报文类型驱动数据流处理,冗余的首部信息大为减少,有效负载所占比例大,提高了网络带宽利用率.

5.4 SUM管理性能

随着网络功能的增强,主机和路由器中将会有大量的服务元和微服务存在.为了说明 SBNS 中的 SUM 是否会随着服务元/微服务的增加而降低性能,这里对 SUM 的管理性能进行了测试.首先,我们启动系统后,只装入基本不可靠服务所需要的服务元/微服务,根据如上实验中的程序传输 100MB 的数据 4 次,记录传输过程中 CPU 的使用率;然后,我们把所有的服务元都动态加载,并订制表 4 中的所有微服务,分别用不可靠服务和可靠服务传输同样的 100MB 数据 4 次,记录传输过程中 CPU 的使用率,得到如图 17 所示的结果.

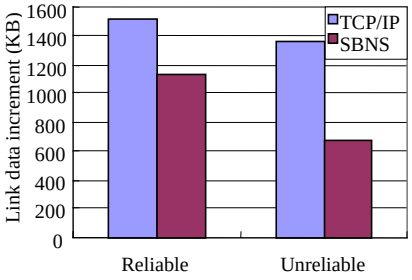

Fig.16 Increment comparison on data transportation

图 16 数据传输链路增量比较

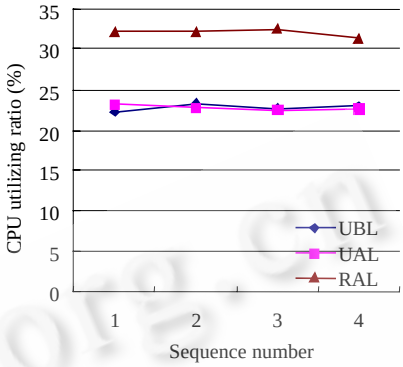

Fig.17 CUP utilizing ratio comparison on data transportation

图 17 PU 使用率的比较

图 17 中 UBL 表示所有其他服务元加载前的不可靠传输,UAL 表示加载后的不可靠传输,RAL 表示加载后的可靠传输.从图中很明显可以看到,在有大量和少量服务元/微服务时,使用同样的网络服务 CPU 的使用率非常接近(考虑系统其他进程对 CPU 使用率的干扰),但是使用不同的网络服务,CPU 的使用率有较大的差异.实际上,SUM 对各个服务元的使用并不需要遍历服务元表,在订制微服务时,已将各个服务元的入口组合在一起,因此无论服务元/微服务数量多少,使用同样的网络服务对 CPU 的使用率都是类似的.不同的网络服务由于使用了不同的调度策略和服务元序列,因此具有不同的 CPU 使用率.

5.5 分析与讨论

(1) 可扩展性和协议耦合问题

通过前面的测试发现,在最基本的可靠数据传输和不可靠数据传输方面,SBNS 系统和 TCP/IP 系统具有相似的效果.其中,在最基本的不可靠数据传输服务方面,SBNS 系统性能上稍微优异些;而在基本的可靠数据服务方面,TCP/IP 的 TCP 服务综合表现稍微好些.实际上,对于前者,SBNS 仅仅使用最基本的网络传输功能,结合类型处理的优势,减少了 TCP/IP 系统下 UDP 的一些冗余处理,如 UDP 的校验和 IP 首部校验等.而对于后者,SBNS 系统分离了 TCP 中的 3 个功能作为分离的服务元,便于更加灵活地订制网络应用要求的服务,实际上是以灵活性替代了部分性能,提高了系统灵活性和功能扩展性,便于更加灵活地提供新的网络服务.另外,原型实验网络还完全没有优化,通过对服务元和服务元管理器的优化调度处理,将会进一步改进系统的性能.

(2) 服务质量和分组调度问题

当前的 TCP/IP 系统面临的重要问题是提供 QoS 能力,QoS 需要网络系统的处理与网络资源的管理和数据分组调度相结合.对于层次体系结构来说,由于不同层次的相对独立性,QoS 需要的这种协调处理变得很难,虽然目前提出了区分服务和综合服务框架,但是实现起来还存在诸多问题.

SBNS 系统提供的灵活性较好地解决了网络系统和网络资源管理、数据分组调度的协调问题.由于无层次的概念,可以针对特定类型分组(某种微服务对应分组类型)定制网络系统对这种分组流的处理序列,从而提供要求的服务.比如,可将某种调度算法定义为一种特殊的服务元组合到微服务适当位置提供必要的 QoS 控制.

(3) SBNS 系统安全性分析

层次体系结构的 TCP/IP 为了提供网络安全保障,在各个层次提供了相应的安全协议.这让网络系统变得复杂且不能很好地实现安全管理目标.SBNS 系统从 3 个层面提供了网络系统的安全性.

首先建立连接提高了网络的安全管理性.SBNS 系统的网络通信要求实现建立通信连接,让沿途的路由器节点和接收方主机进行访问控制,有效减轻了网络随意攻击的可能性;其次是尽量减少首部信息.通过端到端建立连接以后,收发双方通过报文类型来索引服务元处理序列,有效防止了第三者截获报文分析内容的可行性;最后通过在不同的节点设计不同目标的保证通信安全的服务元来加强网络传输信息的安全性.比如,在主机和边缘路由器可以设计接入控制服务元,对用户的建立连接请求进行用户认证;在发送方和接受方主机设计加、解密服务元进行保密通信.与 TCP/IP 系统不同,这样的服务元只要一份,而不用像在 TCP/IP 的各个层次中设计功能相似的多个安全功能部件.它们在微服务的不同位置,将提供不同层面的安全功能.

6 结束语及未来工作

网络应用的普及对网络系统的要求越来越高,TCP/IP 网络虽然在不断改进,但还是落后于网络应用的发展要求.本文从体系结构方面分析了当前网络体系结构存在的问题,总结了近年来网络体系结构方面的研究成果,研究开发了非层次体系结构的 SBNS 及其网络原型.定义了 SBNS 中的服务元、微服务、报文格式等具体内容,并提出了 SUM 来统一管理各服务元组合提供的网络服务.通过分析和实验验证表明该设计是可行的,它占用 CPU 资源少,报文传输效率高,扩展灵活,更易于支持 QoS 和网络安全要求,克服了当前层次结构面临的很多问题,也有效解决了其他非层次体系结构研究中出现的功能组合复杂的问题,具有较高的实用价值.

目前,美国已推出 GENI 项目探索新的互联网架构,新型网络体系结构的研究已得到广泛重视.SBNS 是适应下一代网络要求的有益探索.当然,对于 SBNS 系统,很多细节还有待进一步研究.比如如何智能地组织网络服务,在大规模网络中如何提高传输性能,支持各种 QoS 和网络安全要求等.此外,还需要开发针对 SBNS 的仿真环境,以进一步验证此结构各方面性能.这些都是下一步研究的重点.

References:

[1] Braden R, Clark D, Shenker S. Integrated services in the internet architecture: An overview. RFC1633, 1994.

[2] Blake S, Black D, Carlson M, Davies E, Wang Z, Weiss W. An architecture for differentiated services. RFC2475, 1998.

[3] Clark DD, Tennenhouse DL. Architectural considerations for a new generation of protocols. ACM SIGCOMM Computer Communication Review, 1990,20(4):200−208.

[4] Braun T, Diot C. Protocol implementation using integrated layer processing. ACM SIGCOMM Computer Communication Review, 1995,25(4):151−161.

[5] Lazar AA. Programming telecommunication network. IEEE Network, 1997,11(5):8−18.

[6] Böcking S. Object-Orient network protocol. Proc. of the INFOCOM'97, 1997,4(3):1245−1252.

[7] Braden R, Faber T, Handley M. From protocol stack to protocol heap-role-based architecture. ACM SIGCOMM Computer Communication Review, 2003,33(2):17−22.

[8] Zeng JZ, Xu J, Wu Y, Li YC, Xu N. Service unit based network architecture and its micro-communication element system. Acta Electronica Sinca, 2004,32(5):745−749 (in Chinese with English abstract).

[9] Clark DD. The design philosophy of the DARPA Internet protocols. ACM SIGCOMM Computer Communication Review, 1988, 18(4):106−114.

[10] O'Malley SW, Peterson LL, Peterson L. A dynamic network architecture. ACM Trans. on Computer Systems, 1992, 10(2):110−143.

[11] Zheng P, Zeng JZ, Zhang M, Zhao JD. Micro-Communication Element System. In: Parallel and Distributed Computing: Applications and Technologies. Berlin: Springer-Verlag, 2004. 424−428.

[12] Chrisment I, Kaplan D, Diot C. An ALF communication architecture: Design and automated implementation. IEEE Journal on Selected Areas in Communications, 1998,16(3):332−344.

[13] Abdelzaher TF, Shin KG, Bhatti N. User-Level QoS-adaptive resource management in server end-systems. IEEE Trans. on Computers, 2003,52(5):678−685.

[14] Yau DKY, Lam SS. Migrating sockets-end system support for networking with quality of service guarantees. IEEE Trans. on Networking, 1998,6(6):700−716.

[15] Wang CY, Tham CK, Jiang Y. A framework of integrating network QoS and end system QoS. In: Proc. of the IEEE Int'l Conf. on Communications (ICC 2002). New York: IEEE Press, 2002. 1225−1229.

[16] Milner R. Communication and Concurrency. Prentice Hall, 1989.

[17] Chandranmenon GP, Varghese G. Trading packet headers for packet processing. IEEE Trans. on Networking, 1996,14(2):141−151.

[18] Fang W, Seddigh N, Nandy B. A time sliding window three colour marker (TSWTCM). RFC2859, 2000.

[19] Böcking S. Sockets++: A uniform application programming interface for basic level communication services. Communications Magazine, 1996,34(12):114−123.

[20] Florissi PGS, 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.

[21] Abbasi H, Poellabauer C, Schwan K, Losik G. A quality-of-service enhanced socket API in GNU/Linux. http://www.linuxdevices.com/articles/AT8639420091.html

[22] Microsoft. Windows Socket2 information. http://www.sockets.com/winsock2.htm

[23] Yi FS, Xia MQ, Ye YL, Zeng JZ. The study of general network API of MCSE. Computer Science, 2006,33(3):58−61 (in Chinese with English abstract).

附中文参考文献:

[8] 曾家智,徐洁,吴跃,李毅超,胥能.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.

[23] 易发胜,夏梦芹,叶娅兰,曾家智.基于 MCES 的网络编程接口研究.计算机科学,2006,33(3):58−61.

易发胜(1968－),男,湖北来凤人,博士生,讲师,主要研究领域为计算机网络体系结构.

陈贵海(1963－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为分布式与并行计算.

刘明(1972－),男,博士,讲师,主要研究领域为网络体系结构,无线传感器网络及其 QoS 的研究.

龚海刚(1975－),男,博士,讲师,主要研究领域为分布式计算,网络体系结构.

曾家智(1939－),男,教授,博士生导师,主要研究领域为新型网络体系结构.