第6卷第12期
1995年12月

软 件 学 报
JOURNAL OF SOFTWARE

Vol. 6, No. 12
Dec. 1995

基于STREAMS的X.25分组交换网访问软件的设计与实现*

杨家海 李 军 吴建平 史美林

(清华大学计算机系,北京 100084)

摘要 随着远程通信需求的不断增长和我国公用分组交换数据网的扩容,实现微机UNIX环境下的分组交换网访问协议软件,使我国成千上万的微机用户实现远程通信已迫在眉睫.本文在简单介绍了基于STREAMS机制的通信软件开发的一般思路之后,着重叙述了基于STREAMS机制的X.25分组交换网协议访问软件的设计与实现.并对基于STREAMS机制的通信软件开发方法作了简单评价.

关键词 X.25协议,通信软件,流机制,服务原语.

X.25协议是国际电报电话咨询委员会(CCITT)制定的"公用数据网中通过电路连接的分组式数据终端设备(DTE)和数据电路终结设备(DCE)之间的接口"的协议[1].它目前已被世界上许多国家广泛接受并普遍运用,成为国际公认的组建分组交换网基础的标准化协议[2].X.25协议定义了3层协议——物理层、链路层和分组层——分别对应于ISO/OSI的7层参考模型的下3层[3].

物理层(Physical Level):描述了DTE/DCE接口特性,即建立、保持和拆除DTE和DCE之间的物理链路的机械、电器、功能和规程特性.

链路层(Link Level):描述了通过DTE/DCE接口之间链路交换数据的链路访问规程,其中单链路规程SLP(Single Link Procedure)提供了可靠的数据链路服务,多链路规程MLP(MultiLink Procedure)用于通过多条并行的单数据链路来交换数据.

分组层(Packet Level):描述了在DTE/DCE接口上交换控制信息和用户数据的分组层规程,定义了基本的分组结构和各种数据分组的格式,规定了虚呼叫和永久虚电路业务的规程,以及任选用户功能的规程和格式.

X.25协议全面规定了DCE的性能,还为DTE规定了必要条件的最小集合.X.25协议一端为DCE,另一端则是用户主机DTE,X.25协议实际上就是分组交换数据网PSDN

* 本文1994-06-21收到,1994-11-16定稿

作者**杨家海**,1966年生,讲师,主要研究领域为计算机网络及应用,协议一致性测试,高速网.**李军**,女,1968年生,工程师,主要研究领域为计算机网络,软件工程.**吴建平**,1953年生,教授,主要研究领域为计算机网络,协议一致性测试理论,高速网测试.**史美林**,1938年生,教授,博士导师,主要研究领域为计算机网络,协议一致性测试,CSCW(计算机支持的协同工作).

本文通讯联系人:杨家海,北京100084,清华大学计算机系

(Packet Switching Data Network)向用户提供服务的接口协议，就是说，用户要与 PSDN 通信，就必须实现 X.25 协议. 由于各国建立了许多提供数据分组交换传输业务的公用数据网络，因此 X.25 协议被广泛地采用，形成了许多 X.25 协议的网络产品[2,4].

然而，目前国内尚没有成熟的基于 X.25 协议的网络产品. 随着远程通信需求的不断增长和我国公用分组交换数据网的扩容，实现微机 UNIX 环境下的分组交换网访问协议软件，使我国成千上万的微机用户实现远程通信已迫在眉睫. 本文在简单介绍了 STREAMS 机制的工作原理和特点之后，着重叙述了基于 STREAMS 机制的 X.25 分组交换网协议访问软件的设计与实现.

1　基于 STREAMS 机制的通信软件开发方法

STREAMS 是 UNIX 系统为开发通信服务而提供的一套通用、灵活的开发工具，它能支持多种服务的实现，包括从完整的网络协议包到单个的设备驱动程序的实现. STREAMS 定义了用于系统内核的字符输入/输出、以及内核与 UNIX 系统其它部分之间的标准接口.

STREAMS 的这些机制不仅简单灵活，而且是开放的，包括了一系列的系统调用、内核资源及内核例程[5].

STREAMS 所提供的标准接口和机制使高性能的网络服务及其构件的开发能够实现模块化，并且可移植，也容易集成. STREAMS 提供的是一个框架，而不是局限于任何一个特殊的网络结构，它的用户接口与字符 I/O 的用户接口是完全向上兼容的.

1.1　STREAMS 的组成

STREAMS 是一种全双工数据处理和传输的通道，用于实现内核空间的一个 STREAMS 驱动程序与用户空间的一个进程之间的数据加工和传输. 在核内，一个流由流首、驱动程序以及在它们之间的可选择的零个或多个模块组成，如图 1 所示. 流首是流与进程的接口，主要功能是执行与 STREAMS 有关的系统调用；驱动程序可以是一个提供外部 I/O 设备服务的设备驱动程序，也可以是一个软件驱动程序，通常指一个虚拟设备的驱动程序或有特殊功能的驱动程序，如自环式驱动程序(Loop Driver)或多路器(Multiplexor)；模块位于流首和驱动程序之间，由一组用于处理数据、状态及控制信息的核心例程和数据结构组成，用于完成对流首和驱动程序之间的数据流的处理，STREAMS 的模块可由用户进程进行动态的内部连接，即根据不同的需要进行插入和删除[5,6].

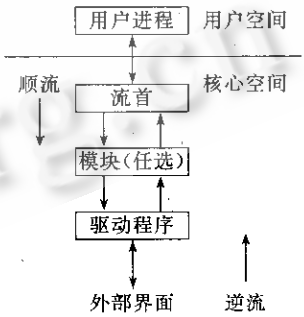

图1 STREAMS基本结构

STREAMS 使用队列结构，以保存与压入的模块或打开的 STREAMS 设备有关的信息. 队列(Queue)是一种数据结构，它包含状态、指向处理数据的若干指针和用于管理流的若干指针. 队列总是成对分配的，一个用于读端，另一个用于写端，每个流首、模块及驱动程序都有一对相应的队列.

数据在 STREAMS 中以消息的形式在流首和驱动程序之间进行传输. 消息(Message)是一组数据结构，它们用于在用户进程、模块和驱动程序之间传递数据、状态和控制信息，从流首到驱动程序的消息叫顺流(downstream)；反之，从驱动程序到流首的消息叫逆流

(upstream). 流首负责传输用户进程数据区与 STREAMS 内核数据区的数据,要从用户进程发送给驱动程序的数据,先形成消息,然后顺流传输,一直到达驱动程序;当包含数据的消息逆流到达流首时,流首则将消息中有关的数据复制到用户缓冲区中.

STREAMS 还通过特殊的伪设备驱动程序支持流的多路复用. 使用连接设备,用户可以动态地连接、维护和拆除多路连接的流配置,此多路转接配置可以在核心中通过交互连接多个流来创建.

1.2 基于 STREAMS 的通信软件开发方法

基于 STREAMS 的通信软件包,实际上就是一系列可动态配置的模块或驱动程序. 在开发基于 STREAMS 的通信软件包的时候,只要合理地划分模块,并依据 STREAMS 机制提供的框架,将通信软件的所有功能在相关的模块或驱动程序中加以实现,即可完成.

STREAMS 的每一个模块或驱动程序都由一组规定的例程组成:

(1)open:打开一个流模块或驱动程序. 驱动程序通过 open 系统调用来进行,模块则通过 ioctl 的 I_PUSH 来实现.

(2)close:关闭一个流模块或驱动程序. 驱动程序通过 close 系统调用来进行,模块则通过 ioctl 的 I_POP 来实现.

(3)put:接收来自流中上一队列的消息,并根据此消息的优先级,或直接传送到下一队列,或把它放在自己的队列上,等待 service 进行处理. put 例程可根据读、写队列的不同而分为 wput 和 rput.

(4)service:处理 put 例程放在自己队列上的消息,它允许对消息的延迟处理. service 例程也可根据读、写队列的不同而分为 wsrv 和 rsrv.

总之,STREAMS 为开发 UNIX 系统通信服务提供了灵活的、可复用和可移植的一组工具,它允许方便地创建提供标准服务的模块,以及在一个流中管理这些模块的能力,并可在用户层动态地选择并交互连接模块.

2 系统功能和结构

通过分析 X.25 协议,我们发现,虽然每个协议层其具体的功能有所不同,但其外部特性和基本结构大体都是相同的,每个协议层的外部特性都如图 2 所示.

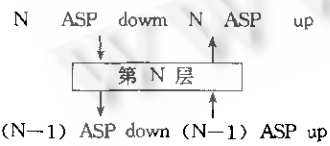

图2 第N层协议的外部特性

这里,N ASP down 表示从第 N+1 层传给第 N 层的访问服务原语(Access Service Primitive,简称 ASP),N ASP up 表示从第 N 层向第 N+1 层传送的访问服务原语,(N−1) ASP down 表示从第 N 层传给第 N−1 层的访问服务原语,(N−1) ASP up 表示从第 N−1 层向第 N 层传送的访问服务原语.

协议层次的这种特性与 STREAMS 的队列结构极其相似,ASP down 可看做是 down stream,ASP up 可看做是 up stream;而每个协议层的内部功能都是由一个状态机来进行描述,即根据接收到的数据和当前所处的状态,而采取相应的动作,并转移到下一个状态,这可通过 STREAMS 的模块或者驱动程序(或多路复用器)来实现.

X.25 协议软件实现下述功能：

(1)X.25 分组层功能：提供 X.25 协议定义的两种虚电路服务，即呼叫虚电路和永久虚电路服务，以及用户任选功能.

(2)X.25 多链路功能：提供多链路的控制和管理，可以根据用户的分配策略，将许多分组分配给几条链路，用以提高吞吐量；也可以同时将一个分组分配给多条链路，用以提高可靠性.

(3)X.25 单链路功能：实现 LAPB 规程，提供可靠的、有流量控制的链路数据传输.

(4)X.25 物理层功能：通过驱动同步通信板提供物理线路的通信.

根据上述功能描述，可将 X.25 协议软件包分成 4 个模块：

* X.25 PLP 模块：实现分组层功能；
* X.25 MLP 模块：实现多链路功能；
* X.25 SLP 模块：实现单链路功能；
* X.25 Ph 模块：实现物理层功能.

它们在 STREAMS 的框架下，具有如图 3 所示的结构.

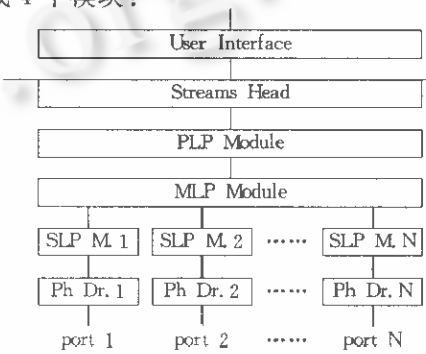

图3　基本结构图

User Interface：用户接口，为 X.25 软件包的用户提供统一的接口，使用户能透明地使用 X.25 软件包提供的服务接口，它是由核外一组提供给用户的函数组成的函数库.

PLP Module：分组层模块，实现 X.25 分组层功能，提供虚电路服务，它是一个多路器(Multiplexor)，用于连接多个用户进程.

MLP Module：多链路模块，实现 MLP 功能，提供多链路控制，它也是一个多路器，用于连接多个单链路.

SLP Module：单链路模块，实现 SLP 功能，提供可靠的、有流量控制的数据链路服务，它是一个流模块.

PhL Driver：物理层驱动程序，用于驱动同步通信板，提供物理层服务，它是一个流驱动程序.

3　状态机模块设计

协议状态机的设计和实现是整个 X.25 软件实现的核心. 前面也叙述过，尽管每个协议层具体的功能有所不同，但其外部特性和基本结构是大体相同的. 本节在描述一个与具体协议层次无关的抽象状态机模型的设计之后，再以 SLP 模块的设计为例加以说明.

图 4 给出了实现一个状态机所涉及的主要函数及其相互之间的调用关系，下面对此作一简要介绍.

aspup 和 aspdown 主要负责 ASP 的传送(向上层或下层)；wstamach 和 rstamach 是状态机的核心，完成通信的主要规程，它们可以通过调用 aspup，向上层发送规程产生的 ASP. 另一方面，若要向下发送(无论是 wstamach 还是 rstamach) PDU，则调用 tx—pdu 函数来发送此 PDU(frame/packet)，tx—pdu 首先调用 encode 对要发送的 PDU 进行编码，然后根

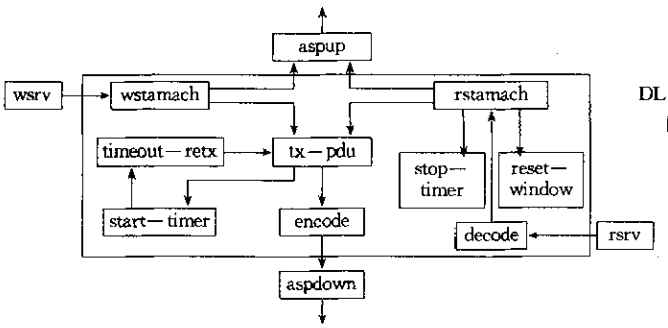

图4 状态机模块结构

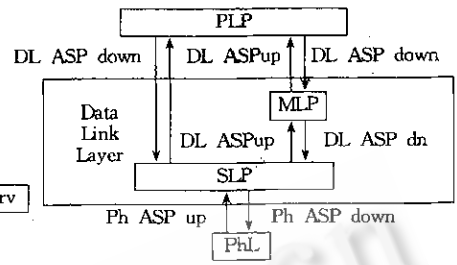

图5 SLP与其它协议的关系

据所发送的PDU类型调用start—timer,启动相应的时钟,最后由aspdown通过ASP将该PDU发送到下层;反之,如果接到下层来的PDU,则由rstamach调用decode对此PDU进行解码,然后根据状态机采取相应的动作.正常发送完指定的PDU时(即在协议定义的时间范围内接收到对方的响应帧/分组),可通过调用stop—timer来停止时钟和调用reset—window来重置窗口;当时钟发生超时时,则调用timeout—retx,该函数根据超时时钟及当前状态进行重发或其它相应处理.

3.1 SLP模块的设计

前面讨论了任一层协议实现的基本模型,下面,我们在这个模型的基础上,简单讨论一下SLP模块的设计.

单链路规程SLP是X.25协议中有关数据链路层的访问规程,前面我们已经说过,SLP的基本功能是提供一个可靠的、有纠错能力和流量控制的数据链路服务.它在X.25协议族中与其它层协议的关系如图5所示.SLP下层是物理层PhL,通过物理层服务原语PhL ASP向上提供物理层服务;SLP的上面可以是MLP,也可以直接与分组层PLP相连,但无论与谁相连,SLP均通过数据链路层服务原语DL ASP向上提供数据链路服务.

根据基本抽象模型的定义和协议文本(LAPB)的描述,SLP模块可分为下述子模块:状态机子模块,编码/解码子模块,ASP发送子模块,错误处理子模块,时钟操作和超时处理子模块等.其中,状态机子模块是SLP协议设计与实现的核心,它实现链路层协议的所有通信规程,包括主要的函数:slp—wstamach,slp—rstamach,slp—txframe,slp—processI,slp—processS等;编码/解码子模块由slp—encode,slp—decode等函数组成;ASP发送子模块由slp—aspdown和slp—aspup函数组成;时钟操作和超时处理子模块由slp—start—timer,slp—stop—timer,slp—timeout—retx以及slp—retxI等函数组成;错误处理子模块负责对检测到的错误进行处理,根据不同需要采取不同的处理方法,由slp—error函数来实现.

在STREAMS机制下,SLP一方面通过slp—wstamach接收流中上一个模块发送来的消息,并根据接收到的数据链路服务原语和当前SLP所处的状态作相应的处理,如果是本地规程,则调用slp—aspup向上层发送相应的回答;否则,通过slp—encode进行编码,并通过slp—aspdown向下一个模块发送出去.另一方面,SLP通过slp—rstamach接收下一层模块(来自通信对方)的信息,经过slp—decode的解码,再根据接收的帧是信息帧还是监控帧,分别调用slp—processI或slp—processS进行处理.

4 结束语

本文论述了基于 STREAMS 机制的通信软件开发的一般思路，并具体描述了在 STREAMS 框架下分组交换网访问协议软件的设计与实现. 在基本抽象模型的基础上，只要根据具体协议层的功能和特点，对基本抽象模型作适当的具体化，即可很快地完成一个协议层模块的设计，并使整个网络软件包具有统一的设计风格.

流(STREAMS)机制对开发 UNIX 系统程序是非常方便灵活的，结构清晰，易于移植和扩充. 基于流机制的 X.25 分组交换网访问协议软件的设计与实现进一步证明了流机制的这些优点.

整个 X.25 分组交换网访问协议软件在 AST/486 高档微机、AT&T UNIX SVR4.0 系统上实现. 目前已经完成所有软件的开发和调试工作，并经过初步的测试，可以正常工作.

今后，我们将进一步完善该软件，并使之逐步走向产品化，实用化.

参考文献

1 CCITT. Recommendation X.25 interface between data terminal equipment (DTE) and data circuit—terminating equipment (DCE) for terminals operating in the packet mode and connected to public data networks by dedicated circuit. 1988.

2 F Burg. Standardizing the user side of the X.25 interface. Comput Comm., Oct. 1982.

3 Liba Svobopova. Implementing OSI systems. IEEE Journal on Selected Areas in Communications, Sep. 1989, 7: 1115—1130.

4 Drukarch et al. X.25: the universal packet network interface. 5th Internation Conference on Computer Communications, 1980. 649—657.

5 UNIX System V/386 Release 4: STREAMS Programmer's Guide, 1990.

6 UNIX 系统 V/386 第 4 版集成软件开发指南. 电子工业出版社，1992.

THE DESIGN AND IMPLEMENTATION OF STREAMS—BASED X.25 PACKET SWITCHING NETWORK ACCESSING SOFTWARE

Yang Jiahai　Li Jun　Wu Jianping　Shi Meilin

(*Department of Computer Sciences, Tsinghua University, Beijing* 100084)

Abstract　After brief discussion of communication software development based on STREAMS mechanism, the design and implementation of X.25 accessing software based on STREAMS mechanism is presented, with which millions of microcomputer users can communicate each other remotely.

Key words　X.25 protocol, communication software, STREAMS mechanism, service primitive.