多协议网络监控器的设计与应用*

于秀峰 鞠九滨

(吉林大学计算机系,长春 130023)

摘要 本文介绍基于网络连接的多协议网络监控器的设计与实现方法,并给出了用其对一个实际的多协议网络监控的实例.

关键词 协议,网际网,网络,监控器,连接.

随着网络规模的不断扩大和网络复杂程度的不断提高,网络的管理越来越困难.如何充分发挥网络的功能和性能,关键在于网络管理的好坏.网络管理包括信息的收集、分析和统计、网络安全、网络配置和网络控制等多方面.通常,信息收集、分析和统计是实现其它管理的基础,为实现网络管理提供可靠的信息和数据.

有些网络监控器只是捕获数据链路层信息,不对高层数据进行分析,处理的信息量比较小,所需时间较少,实时性强,但只限于数据链路层,所以网络管理的功能范围比较小[1-3],主要作为一般设计和管理的工具.而在大规模网际网环境中实时捕获连接信息及应用层信息,以网络连接为单位的网络监控器还很少[4].

本监控器是在由 DECnet 和 SUNnet 两种网络组成的异构网际网环境中研制的,主要监控 IP、DECnet 和 LAT 三种协议.监控程序运行于 1 台 80486 微机上,使用 DOS 5.0 单用户操作系统.本系统实现了实时接收、分析统计及输出,是一个基于网络连接的多协议网络监控器.

1 监控器的设计与实现

1.1 监控器的设计与实现

目的是设计一个实时的多协议网络监控器,监控所有连接.在网络通信中,大部分活动是通过网络连接实现的.连接的建立、维护和释放都是通过通信双方的包交换来完成的.因此,截获网络中的所有包,通过分析包头控制信息,可以分析出连接的全过程.通过对连接的分析,可以区分网络通信中各种用户活动和目的,最终实现网络全面管理和控制.

把网卡的接收匹配模式设置成接收所有包模式.当网上有包时,监控器主机网卡产生串行口中断,包的内容自动传到网卡 2K 缓存中的高地址区,然后再通过 I/O 方式或 DMA 方

* 本文 1992-07-14 收到,1993-10-18 定稿

本课题是国家自然科学基金资助项目.作者**于秀峰**,1964 年生,讲师,主要研究领域为计算机网络和分布式系统.**鞠九滨**,1935 年生,教授,主要研究领域为计算机网络、分布式系统和并行处理.

本文通讯联系人:于秀峰,长春 130023,吉林大学计算机系

© 中国科学院软件研究所 http://www.jos.org.cn

式将包内容传到内存中.如内存空间很大,主机处理速度足够快,收到的所有包都可存于内存中,并且在内存中为每个连接保持一个连接记录,对包头进行全面分析统计.

但主机的处理速度和内存空间都是有一定限制的.当网上数据传送速度很快而处理机速度相对较慢时,如任务划分不当,则可能使大量数据包丢失;另外,也不可能在内存中为每个连接保持一个连接记录以及保存所有包内容,因此,要保证包的实时接收.当有包到达时,马上启动包接收程序,最大限度地降低丢包的可能性,保证监控的完整和正确.另外,某些信息要写入外存,用于进一步分析,这一部分是影响系统性能的关键.

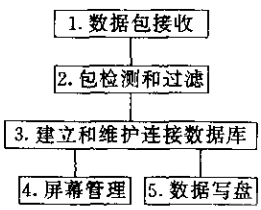

图1

任务划分见图1.

DOS操作系统中的多任务操作不是用多进程而是用多个中断程序来实现的,各中断程序的优先级不同.本监控程序主要由以下几部分组成:

1.包接收中断处理程序:占用0x0b中断向量,完成包接收,检测和过滤等任务.

2.时钟中断处理程序:控制监控时间.当达到监控时限时,监控程序结束标志置位.

3.键盘中断处理程序:在监控过程中,可用热键终止程序执行,进入屏幕管理窗口.

4.前台监控主程序:完成监控器的主要工作.主要包括建立和维护连接数据库、屏幕管理和数据写盘.其中建立和维护连接数据库是本监控器的核心.

1.2 建立、维护连接数据库

在网络中,不同结点的两个进程为实现某种通信目的建立的一个"虚拟通信信道",称为一个连接.此"信道"为这两个进程所专有,通过此"信道",能收到此连接的所有数据包.网络连接一般由结点地址和进程口号构成的连接关键字来唯一标识.结点地址标识主机位置,进程口号标识结点中确定进程.在网络中,一个连接是唯一确定的.

为实现连接的全面分析统计,每个连接保持一个连接记录.连接记录的数据结构如下:

```
struct connection_record{
  char source_connection_key[6];                    /* 源连接关键字 */
  char target_connection_key[6];                    /* 目的连接关键字 */
  char packet_type[2];                              /* 协议类型 */
  char control_byte;                                /* 连接状态控制字节 */
  unsigned long source_to_target_packet_number;     /* 连接启动方到目的方的包数 */
  unsigned long target_to_source_packet_number;     /* 连接目的方到启动方的包数 */
  char high_protocol;                               /* 高层协议类型 */
  unsigned long total_byte_number;                  /* 此连接传的字节数 */
  float connection_start_time;                      /* 启动连接时间 */
  float connection_end_time;                        /* 关闭连接时间 */
  unsigned long connection_head_packet_pointer;     /* 此连接第一个包指针 */
  unsigned long current_pointer;                    /* 当前包指针 */
  char ini_to_tar_ident[2];                         /* 源到目的结点的标志位 */
  char tar_to_ini_ident[2];                         /* 目的到源结点的标志位 */
  Int direction;          /* 上一包方向. 0:启动结点->目的结点;1:目的结点->启动结点 */
```

© 中国科学院软件研究所 http://www.jos.org.cn

```
    char sequence[4];                                  /*包的顺序号*/
    char acknowledge[4];                               /*包的应答号*/
    int byte_number[2];                                /*包有效字节数*/
    unsigned long source_to_target_byte_number;        /*启动结点->目的结点的字节数*/
    unsigned long target_to_source_byte_number;        /*目的结点->启动结点的字节数*/
    struct connection_record *next_record;             /*指向下一连接记录的指针*/
    int connection_code;                               /*连接记录序号*/
};
```

1.2.1　连接记录定位

当一个数据包到达时，可以通过连接关键字的唯一性来确定此包属于哪个连接. 通过这种方式，可以保证通信双方在整个通信过程中能够收到对方的所有数据包.

当新的数据包到达时，取出连接关键字. 若此连接记录存在且连接没有关闭，则更新此记录；否则，建立一新的连接记录.

记录定位采用一个比较简单的杂凑函数 H(k)来实现.

H(k)=k mod m 其中 k=SOURCE KEY+TARGET KEY，m=1997

m 值可根据内存空间来调整，以减少定位冲突. 冲突时，用一链表把记录连接起来.

1.2.2　连接状态

每个连接记录中有一控制字节(control_byte)来记录此连接的状态，包括连接建立、释放和维护. 当有包到达时，按上面 3 种类型进行连接记录处理.

1.2.3　重发包的识别：

在我们的系统中通过包序号跟踪的方法来识别重发包.

设开始建立连接时，序号为 n，用 pack_number 记，包号的上下界分别用 high_bound，low_bound 记，链表 point 包含 2 项：Num，point. 算法如下：

```
a) pack_number=n;high_bound=0,low_bound=0
b) curr_number<-所收包的顺序号.
    if (high_bound>low_bound) then goto c)
    if (curr_number<=pack_number) then
      此包为重发包，记录此包.
    if (curr_number==pack_number+1) then pack_number++
    else
      {
        high_bound=curr_number;low_bound=pack_nmber;
      }
    goto d)
c) if (curr_number<=low_bound) then
      此包为重发包，记录此包.
    if (curr_number>high_bound) then
      {
        把 high_bound 送入链表；high_bound=curr_number;
      }
    if (curr_number>low_bound) && (curr_number<high_bound)
    then
      {
        在链表中查是否有和 curr_number 相等项. 如果有，则此包为重发包，记录此包；否则把 curr_number 加入
链表；
```

© 中国科学院软件研究所 http://www.jos.org.cn

```
    }
  if (curr_number==high_bound) then
    此包为重发包,记录此包.
  if (high_bound==low_bound) then 将链表置为空;
d) 返回接收下一包.
```

监控结束时,上下界不等,且链表中的项并不连续,那么监控过程中丢过包.

2 应 用

2.1 工具及其用途

1. 系统执行完之后,把包头和连接记录转换成文本方式,用户可以通过本系统的菜单选择或其它编辑软件来阅读这些内容.

2. 可以以连接或结点为单位,把此连接或与所选结点相关的所有包抽取出来以文本方式存入一文件中. 这样,可以跟踪某一具体连接的全过程,给出此类连接的完整数据,是协议分析和实现进一步网络管理的良好工具.

我们已经使用本监控器对 DEC 网中的 LAT 协议进行了分析,并在此基础上,在 SUN 工作站网上实现了 LAT 协议.

在目前的网络操作系统中,以连接为单位进行监控并给出详细连接信息和包信息的还不多. 特别是在微机上进行网际网监控器的设计是个新的尝试,为微机用户提供了网络连接分析和协议分析的良好工具,为实现网络的进一步管理提供了方便.

2.2 运行结果

1. 屏幕管理窗口

当监控结束,进入屏幕管理窗口之后,可以通过菜单选择来抽取某连接或与某结点相关的所有包,把它们存于文本文件中,并显示数字和直方图等统计分析信息.

2. 三种协议分析实例

分别通过三种协议进行远程登录,执行同一远程命令,比较三者的总包数、总字节数、总时间. 主机有 2 台 DECstation 3100(jidaa,jidab)和 1 台 286 微机. 登录方式如下:

TCP 协议:	DECnet 协议:	LAT 协议:
jidab>rlogin jidaa	jidab>dlogin jidaa	c>sethost jidaa

执行命令:jidaa>find /usr -name "*" -print

统计结果见表:

次 数	统计项	TCP 协议	DECnet 协议	LAT 协议
第 1 次	时间(秒)	52.99	68.63	180.30
	总字节数	247456	358571	400458
	总包数	574	4548	3708
第 2 次	时间(秒)	49.79	62.84	175.01
	总字节数	231835	339728	385803
	总包数	406	4327	3862

三种协议的平均比率(TCP:DECnet:LAT)为:

时间比=51.93:65.735:177.655=1:1.2658:3.457

字节数比=239645.5:349149.5:393126.5=1:1.467:1.641

包数比=490:4437.5:3785=1:9.056:7.724

© 中国科学院软件研究所 http://www.jos.org.cn

从统计结果来看，完成同一功能，TCP 协议所需时间比较少，所传包数也比其它两种协议少得多，占用网络带宽较低. 当然，这里还有机器性能的影响. 在监控中我们发现，虽然 DECnet 和 LAT 协议的数据包长也可以达 1514 字节，但一般很少有长包出现. 而 TCP 在传送中一般尽量打大包. 由于每个包中都要包含包头控制信息，因此，小包传送包头控制信息所占带宽要大大增加. 传送正常数据的包和应答包数的增多，使网络的开销扩大.

2.3 性　能

影响系统性能的主要环节是连接记录定位和磁盘操作. 为减少记录定位时间，我们设计了一个冲突较少的杂凑函数，通过连接关键字取模来直接定位连接记录. 从统计结果来看，1000 个网络连接的冲突率在 2%以下. 为提高读写盘速度，本系统采用了高速缓存技术，DMA 输入输出技术，效果较好. 由于受到以上因素的影响，再加上监控器主机的时钟较低(20MHz)，使本系统可处理的网络带宽降低，在 400KB/S 以下，当网络带宽超过此值时，监控器可能丢包. 运行环境中有 70 余个结点，2 个终端服务器带 16 个终端，每个终端可登录到任何一个服务结点上. 另外，有 10 台 SUN4/20 无盘结点共享一个文件服务器(1.3GB 硬盘).

2.4 相关工作比较

(1) VMS 的 MONITOR DECNET

MONITOR DECNET 是 VMS 5.4 操作系统提供的适时网络监控程序. 执行此程序时，在命令执行终端上实时显示 DEC 网的各种统计信息. 此程序不保存包和连接内容.

(2) SunOS 的 eithefind

eithefind 是 SUNOS 4.1.2 上功能较强的网络监控程序. 执行此程序，可以把某些包的包头信息在屏幕上显示或存入一文本文件中，同时给出详细的统计信息. 包显示或保存的顺序由包到达的先后顺序决定. 它不能对整个网络通信情况进行监控，不能分析 LAT、DECnet 协议. 由于监控不是以连接为单位，在网上同时存在多个连接时，要抽取某一具体连接的所有包是很困难的. 因此，协议分析、连接分析不是很方便.

(3)文献[4]的工作

我们研制的监控器的主要思想参考了这一工作，但所选的研究环境是 SUN 网络和 DEC 网络组成的网际网环境，监控协议有 IP，DECnet，LAT 等. 监控器选在 TEMPO 80486 微机上，操作系统是 DOS 5.0 单用户操作系统，多个任务是通过多个中断处理程序实现的.

文献[4]的研制环境为 DECnet 网络环境，监控协议是 DECnet 协议. 监控器选在 VAX 小型机上，操作系统是 VMS 多用户操作系统. 多个任务是通过多个进程实现的. 由于运行监控程序时要占去较多的系统时间，因此，对系统中的其它进程的执行有较大影响.

从实现角度看，在多用户分时系统上多个任务可以由多个进程来完成，系统的整体结构比较好. 各进程之间的环境切换，时间片的划分以及优先级的划分不需用户过多干涉.

下一步我们将借助这个工具，在微机上设计实现一个全分布的网络控制管理系统模型，提高通信中的数据安全性. 由于控制在通信的第三方，所以不会影响网络的性能.

参考文献

1　The sniffer: operation and reference manual. Network General Corporation, Sunnyvale, California, 1986.

© 中国科学院软件研究所 http://www.jos.org.cn

2 Shoch J F, Hupp J A. Performance of the ethernet local network. Communications ACM, 1980, **23(12)**: 711—721.

3 Soha M. A distributed approach to LAN monitoring using intelligent high performance monitors. IEEE Network Magazine, 1987, **1(3)**: 13—21.

4 Sudama R, Chiu Dah—ming. Experiences of designing a sophisticated network monitor. Software Practice and Experience, 1990, **20(6)**: 555—570.

DESIGN AND APPLICATION OF MULTI—PROTOCOL NETWORK MONITOR

Yu Xiufeng Ju Jiubin

(*Department of Computer Science, Jilin University, Changchun* 130023)

Abstract This paper describes the design and application of a multi—protocol network monitor based on connection, and introduces some examples of using the monitor in a real multi—protocol network.

Key words Protocol, internet, network, monitor, connection.

© 中国科学院软件研究所 http://www.jos.org.cn