在 Linux 下建立 Windows NT 的终端的技术研究*

刘发贵, 杨　勉
(华南理工大学 计算机学院, 广东 广州 510640)

摘　要: 研究了在 Linux 上建立 Windows NT 终端的方法。在 Linux 下建立 Windows NT 的终端涉及到以下技术: Windows NT 中的终端服务; 客户端的 X Window 环境; 客户端与服务器的通信使用到的协议——RDP 协议; 客户端应用的实现。结合以上技术, 可以实现在 Linux 下建立 Windows NT 的终端。

关键词: Linux 终端; 终端服务; RDP; X Window
中图法分类号: TP316.8　　**文献标识码**: A　　**文章编号**: 1001-3695(2005)03-0087-03

Study on Developing Linux Terminals Utilizing Windows NT's Terminal Services

LIU Fa-gui, YANG Mian
(School of Computer Science & Technology, South China University of Technology, Guangzhou Guangdong 510640, China)

Abstract: Elucidate how to develop Linux terminals which utilize Windows NT's terminal services. The technology involved includes: Windows NT's terminal services; X Window GUI environment on the client side; Remote Desktop Protocol(RDP); implementation of the client side application.

Key words: Linux Terminal; Terminal Services; RDP(Remote Desktop Protocol); X Window

随着 Linux 技术的发展和日益广泛的应用, 以 Windows NT 操作系统为服务器, 以 Linux 操作系统作为终端的工作模式逐渐得到人们的重视。这种方式下, 用户在终端操作时所面对的界面是很熟悉的 Windows 界面, 而终端实际使用的是 Linux 操作系统。目前, 以 Windows NT 的终端服务作为服务器, 同时以 Windows 终端作为客户端的应用已很普遍, 但是以 Linux 终端作为客户端的应用却很少。本文要讨论的是客户端的平台采用 Linux 操作系统, 开发客户端的程序。使用 Linux 作为客户端, 是基于以下考虑: ①Linux 是一种飞速发展的操作系统, 而且使用的范围在逐渐扩大, 采用 Linux 作为客户端来访问 Windows, 从某种程度上实现了它们之间的互连与资源共享; ②Linux 是一种开放的资源, 如果出于将技术产品化的考虑, 选用 Linux 可以降低成本[1]。

1　Windows NT 的终端服务

Windows NT 服务器上的终端服务是 Windows NT 操作系统提供的一个重要的服务, 它使 Windows NT 扩大了接收用户的范围。当客户端运行符合 RDP 协议的客户端程序后, 无论客户端的操作系统是什么, 对用户而言看到的都是熟悉的 Windows 的操作界面。本文在 Linux 下建立 Windows NT 的终端, 服务器方应用的就是 Windows NT 的终端服务。

Windows NT 的终端服务主要由终端服务子系统、Win32 核心模式子系统(Win32k.sys)、显示驱动[2]组成。

终端服务的整个体系结构中, 核心是终端服务子系统。终端服务子系统在系统启动的时候即被运行, 它管理客户端的连接。终端服务子系统始终处于运行状态。终端服务子系统追踪所有的客户连接、创建和关闭连接上下文。它将用户连接看成一个会话, 并给每个连接赋一个唯一的会话标志符, 终端服务保留会话标志符 0 作为控制台的会话标志符, 其他的会话标志符都被认为是从客户端发出的连接[2]。

会话的组成包括 Win32 核模式子系统(Win32k.sys)和一个显示驱动。其中, Win32 核模式子系统作用是管理 Windows NT 的图形 GUI 环境。在标准状态下, 一个控制台的显示驱动是标准显卡驱动, 但由于 Win32k 与终端服务客户端的连接并不使用控制台的显示, 而只是传送画图的命令到客户端上, 所以, 终端服务器上使用的是一个供非控制台会话使用的虚拟的显示驱动。对 Win32k 而言, 它就像一个真实的显示驱动。它与真实的驱动的不同点在于, 真实的驱动将图形写进显卡显存; 而虚拟驱动将图形写进代表该客户端的缓冲中, 然后由服务器的通信栈将缓冲的内容传到客户端[2]。

2　Linux 的 X Window 和 RDP 协议

要将服务器上的虚拟显示驱动中的图形显示在 Linux 客户端上, 就涉及到了数据在网络中的传输和 Linux 上的图形显示。下面就分别介绍了这两种技术。其中, Linux 的 X Window 是终端显示的图形技术; RDP 协议是实现建立在 Linux 上 Windows NT 终端的传输协议。

2.1　Linux 的 X Window

X Window 是 Linux 的图形系统, 它是一个 C/S 架构的图形 GUI。这个 C/S 架构中的 Server 是指整个管理桌面的 X Window 系统, 称为 X Server; C/S 架构中的 Client 指的是在 X Window 下执行的应用程序, 它需要请求 X Server 为其服务。

收稿日期: 2004-04-04; 修返日期: 2004-06-26
基金项目: 国家"863"计划资助项目(2003AA1Z2170)

X Server 与 Client 的通信是一个双向的过程。Client 程序首先与 X Server 之间建立一个连接。在建立连接后, Client 程序对 X Server 的通信是: Client 应用程序会通过 Linux 下的图形库 Xlib, 向 X Server 发出请求。

X Server 对 Client 的通信是: 使用者按下按键或是移动鼠标, 或者 Client 发出了请求, X Server 就会产生事件, 然后 X Server 就通过连接通知 Client 程序事件产生了, Client 程序按照事件的不同作出不同的反应[3]。

Client 和 X Server 不一定要在同一部电脑上执行; Client 和 X Server 可以是接在网络上的两台电脑 。它们之间的通信可以通过网络来传输。

2.2 RDP 协议

终端服务器与其客户机之间的传输协议是 Remote Desktop Protocol(RDP) 。RDP 是以 International Telecommunications Union 's (ITU 's) 的 T. 120 协议为基础的协议。T. 120 协议的一个连接(Connection) 就支持 64 000 个通信通道(Communication Channel) , 所以数据能分为多个逻辑流。例如, 数据分为键盘输入流和图形显示流。终端服务使用的是单通道的连接, 键盘、鼠标和显示数据都共用这个通道[4]。

RDP 是一个应用层协议。针对其对数据包传输可靠性的严格要求, RDP 在运输层使用点对点的 TCP 协议进行传输。每一个 RDP 包都包括在此协议上构建的四个层次: ① ISO 运输协议; ② MCS(Multipoint Communications Services, 多点通信服务) 协议, 它为多个端点同时的输入/输出提供了可靠的传输服务; ③一个用于加密的安全层; ④内部的 RDP 包, 它包含了与具体应用相关的信息。

下面显示了一个 RDP 包的结构(其中对每一层的重要内容都做了一些注解) :

ISO 首部
长度: 7 个字节
包括: ISO 版本号, 长度以及 ISO PDU 的类型代码等。
MCS 首部
长度: 8 个字节
包括: MCS PDU 的类型代码, MCS 用户 ID, 长度以及用于传输的 MCS 信道类型—目前 RDP 只使用全局信道。
SEC 首部
如果启用了加密功能, 长度为 12 个字节; 否则, 如果已获得了一个认证则长度为 0; 否则包含一个 4 个字节的 SEC 传输常数。
RDP 首部
长度: 6 个字节
包括: 长度, RDP PDU 的类型代码等。
RDP 数据
长度可变。

3 在 Linux 下建立 Windows NT 的终端的技术

在利用 Windows NT 的终端服务建立 Linux 终端的这种模式下, Linux 的终端不仅要处理从服务器传输到本地的图形数据的显示, 还要将本地发生的事件向服务器报告。

3.1 报文格式

在终端服务器和客户端连接的初始阶段, 它们之间要建立初始的连接, 需要互发一些控制报文来达到同步。首先是客户端向终端服务器发请求建立连接、请求登录的报文; 服务器对报文作出回答; 在此之后, 两者还要互发一些同步、控制、协商字体的报文。这样初始的连接就建立了, 进入了具体传输数据报文的阶段。

数据报文是 RDP 中最为复杂的一类报文。为了使数据能很好的传输, RDP 设计了不同的 PDU 来封装不同的信息。从较大的方面来分, 就有八类。而每类又可以具体细分为若干子类, 有些子类还可以向下细分更小的子类。对于每个最小的子类都需要不同格式的报文格式来封装或者需要不同的方法处理。较大的八类报文用一个枚举来表示如下:

```
enum RDP_DATA_PDU_TYPE{
    RDP_DATA_PDU_UPDATE ,
    RDP_DATA_PDU_CONTROL ,
    RDP_DATA_PDU_INPUT,
    RDP_DATA_PDU_SYNCHRONISE,
    RDP_DATA_PDU_BELL,
    RDP_DATA_PDU_LOGON,
    RDP_DATA_PDU_FONT2
};
```

其中, 有四种是供终端服务的终端用的, 而另外四种是供终端服务的服务器用的。

供终端用的四种数据包如下:

RDP_DATA_PDU_SYNCHRONISE 和 RDP_DATA_PDU_CONTROL, 在连接的初始阶段终端向服务器发包是已经用到。

RDP_DATA_PDU_INPUT 用来包装鼠标和键盘的输入。

RDP_DATA_PDU_FONT2 用来包装字体。

供服务器用的四种数据包如下:

RDP_DATA_PDU_UPDATE 用来包装各种更新, 如颜色、字体的更新、调色板的更新等。

RDP_DATA_PDU_POINTER 用来包装鼠标的信息。

RDP_DATA_PDU_BELL 用来包装键响铃, 如告警响铃。

RDP_DATA_PDU_LOGON 用来包装登录事件。

3.2 主要事件

引起终端向终端服务的服务器发送报文的事件只有两类: 鼠标和键盘, 具体来说有五个事件, 即按下按键、释放按键、点击鼠标、释放鼠标和移动鼠标。

引起终端服务的服务器向终端发送报文的事件是屏幕的更新, 缓冲的更新。屏幕的更新包括线、矩形、鼠标位置、鼠标颜色的变化等; 缓冲的更新包括位图、颜色、字体的缓冲的更新等。就客户端而言, 只需要用一个窗口进行图形显示, 在窗口中看到的 Window 桌面事实上只是远程的终端服务器上桌面一个图像。所以即使在客户端运行服务器上的程序, 使窗口的显示看上去很复杂, 如何去排列和显示窗口是服务器方的工作, 而客户方的显示器上只是显示了一个服务器的图像而已。

3.3 虚通道

在终端的实现时, 不同的功能模块用不同的虚通道实现。在使用虚通道前, 要对虚通道调用的进程进行注册, 所以每个虚通道都可以实现其独立的功能。虚通道的数据结构如下:

```
typedef struct _VCHANNEL
{
    uint16 mcs_id;
    char name[ 8] ;
    uint32 flags;
    struct stream in;
    void ( * process) ( STREAM) ;
} VCHANNEL;
```

RDP5 中的声音传输是用一个虚通道来实现的。不同的虚通道对应了不同的数据处理。声音虚通道具体执行两个任务: 接收声音数据和操纵数字音频设备。其中, 操纵数字音频设备包括打开、关闭、设置音频属性、写入声音数据、调节音量。

总体来说, Linux 上的声音播放是通过 OSS——跨平台的音频接口实现的。在 Linux 系统中, 所有的设备都被统一成文件, 通过对文件的访问方式(首先 open, 然后 read/write, 同时可以使用 ioctl 读取/设置参数, 最后 close) 来访问设备。

3.4　两个 C/S 结构

在本地, 图形系统是 C/S 结构的 GUI。其中, X Server 是 Server, 应用程序是 Client, 本地的 X Server 与 Client 的连接会产生一个文件描述符 X Socket。应用程序和远程的终端服务的服务器连接又构成另外一个 C/S 结构, 它们之间的通信是 TCP 连接建立的, 这个连接可以用另外一个文件描述符 Socket 来表示。所以, 应用程序同时是这两个 C/S 结构的 Client[5]。

Client 既要处理本地的事件, 也要处理与终端服务的服务器的通信。例如, 当用户点击鼠标会被本地 X Server 捕捉到, 它就会通知 Client 对鼠标点击作出反应, 作为终端服务的客户端, Client 要做的是确定当前鼠标的事件是按下鼠标和鼠标的当前位置等, 然后将这些信息按照协议的格式打包, 发给终端服务的服务器, 供它去处理, 而它自己则等待终端服务的服务器通过网络发来的数据包告诉它, 显示器上该如何显示。

Socket 和 X Socket 这两个文件描述符构成了一个文件描述符集合。应用程序不断检查文件描述符集合中的每个文件描述符是否可读写。如果 X Socket 是可读写的, 说明本地产生了一些事件(鼠标或者键盘), X Server 捕捉到了这些事件, 这时应用程序就开始处理终端的一些事件; 如果 Socket 是可读写的, 说明远程的 Server 发来了 RDP 包, 这时应用程序就接收 RDP 包, 并分析包中的 PDU 类型, 在终端上进行相应的更新。其系统结构如图 1 所示。

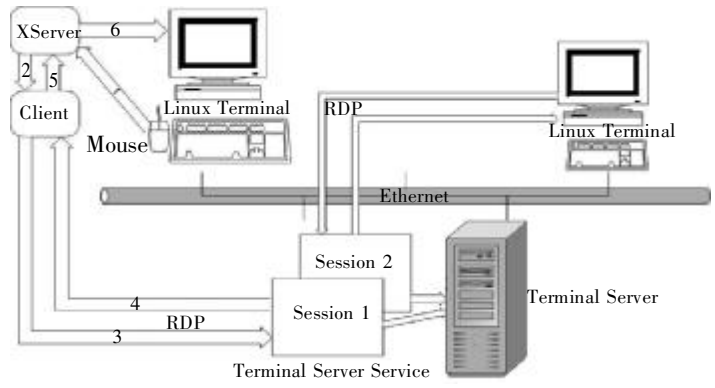

图 1 中 1 ～6 编号的箭头, 按照 1, 2, 3, 4, 5, 6 的顺序, 描述了终端对鼠标的响应、通信及显示过程

图 1　在 Linux 下建立 Windows NT 终端的系统结构图

4　结束语

综上所述, 利用客户方的图形显示技术, 遵循 RDP 协议开发 Linux 终端的客户端程序, 可以实现在 Linux 下使用 Windows NT 的终端服务。采用以上技术建立的 Linux 终端, 简化了网络安全管理的, 节约了硬件的升级成本, 实现环保, 达到很好的社会经济效益。而且, 随着 Linux 技术的发展和嵌入式技术的发展, 它会具有越来越广阔的应用前景。

参考文献:

[1] imon Lok, *et al*. A Graphical User Interface Toolkit Approach to Thin Client Computing[DB/OL]. ACM 1-58113-449-5/02/0005, 2002.

[2] Mark Russionovich, Windows & .NET Magazine. Inside Microsoft Terminal Server[J/OL]. http://www.winntmag.com/Articles/Index.cfm? IssueID=54&ArticleID=3594, 1998.

[3] Kurt Wall, *et al*. Linux Programming Unleashed Second Edition [M]. Beijing: Tsinghua University Press, 2001.442-445.

[4] Microsoft Corporation. Remote Desktop Protocol (RDP) Features and Performance White Paper[EB/OL]. http://www.microsoft.com/windows2000/techinfo/howitworks/terminal/rdpfandp.asp, 2000.

[5] Tom Swan. 深入学习: GNU C++ for Linux 编程技术[M]. 邱仲潘, 等. 北京: 电子工业出版社, 2000.523-600.

作者简介:

刘发贵(1963-), 女, 湖南娄底人, 副教授, 博士, 主要研究方向为操作系统、嵌入式软件环境、电子商务; 杨勉(1978-), 女, 湖北枝江人, 硕士生, 主要研究方向为操作系统、嵌入式软件环境。

(上接第 86 页)

表 1　两种操作所需时间的测试结果

		Failover	Failback			Failover	Failback
工作节点 1	1	9s	5s	工作节点 2	1	8s	4s
	2	9s	5s		2	8s	3s
	3	8s	3s		3	8s	3s
工作节点 3	1	9s	4s	工作节点 4	1	9s	3s
	2	9s	4s		2	8s	4s
	3	9s	4s		3	8s	4s

这个集群的 VRFS 层, 有一个 Cache 模块可以缓冲 6s 左右的电影数据, 所以如果节点只是因为网络故障失效, 在缓冲的作用下, Failover 时, 画面的停顿会减少至 2s ～3s。在这个系统中, 节点失效的主要原因是网络故障。另外 Failback 时, 在缓冲的作用下, 画面只有微小的抖动。这样的结果对终端用户来说是可以接受的。

5　总结

通过硬件冗余或软件的方法都可以从很大程度上提高系统的可用性。硬件冗余主要是通过在系统中维护多个冗余部件如硬盘、网线等来保证工作部件失效时可以继续使用冗余部件来提供服务; 而软件的方法是通过软件对集群中的多台机器的运行状态进行监测, 在某台机器失效时启动备用机器接管失效机器的工作来继续提供服务。

我们实现的集群解决方案, 保证集群系统可用拥有很高的可用性能力。具备热切换的功能, 对外的服务保持通畅。节点与节点之间通过高速网络连接, 相互之间可以发现对方的故障, 并且能根据故障作出相应的处理, 以实现容错的能力。

可以进行带电维护和升级。在保证对外提供的服务正常的前提下, 可以对一个节点进行停机维护, 或者是对硬件、软件进行升级。维护完成之后, 节点可以重新加入集群, 整个过程不影响集群对外提供的服务。

参考文献:

[1] ajkumar Buyya. High Performance Cluster Computing: Architectures and Systems[C]. volume 1, Prentice Hall PTR, 1999.68-82, 327-350.

[2] Rajkumar Buyya. High Performance Cluster Computing: Programming and Applications[C]. volume 2, Prentice Hall PTR, 1999.1-18.

[3] Advanced Computer, Network Corporation, RAID-5: Independent Data Disks with Distributed Parity Blocks[EB/OL]. http://www.acnc.com/04_01_05.html, 1999.

作者简介:

王明伟, 硕士研究生, 男, 主要研究方向并行计算、集群、视频服务器、嵌入式操作系统; 尹康凯, 硕士生, 主要研究方向包括操作系统、并行计算、集群、视频服务器系统; 李善平, 教授, 博士生导师, 主要研究领域操作系统、并行计算、集群、嵌入式系统、CIMS。