基于 VNC 的远程桌面传输协议分析与研究①

朱永强[1,2], 汤 雄[1,2]

[1](成都网安科技发展有限公司, 成都 610092)
[2](电子科技大学 计算机科学与工程学院, 成都 610054)

摘 要: VNC 由于优秀的瘦客户性与良好的跨平台性, 在远程桌面同步领域得到了广泛的应用和关注. 针对 VNC 的整体架构, 对其进行了分析与研究, 重点研究了 VNC 服务端的桌面更新获取机制以及 VNC 所使用的 RFB 协议对图像的压缩处理算法, 并提出了对 VNC 协议进行进一步优化的方向.

关键词: VNC 协议; 远程桌面同步; Hook; Mirror driver; RFB 协议

Research and Analysis of Remote Desktop Transfer Protocol Based on VNC

ZHU Yong-Qiang[1,2], TANG Xiong[1,2]

[1](Chengdu Wang'an Technology Co.LTD , Chengdu 610092, China)
[2](School of Computer Science & Engineering, University of Electronic Science and Technology of China, Chengdu 610054, China)

Abstract: Due to the excellent performance of thin clients and cross-platform, VNC has been widely applied and drawn widespread attentions in the field of the Remote Desktop synchronization. In the paper, we analyze and study the overall architecture of VNC, especially the updating retrieval mechanism of the desktop of VNC server and image compression processing algorithms in RFB protocol utilized by VNC. Moreover, the direction for further optimization on VNC protocol is proposed.

Key words: VNC protocol; remote desktop synchronization; Hook; Mirror driver; RFB protocol

远程桌面同步是一种以网络信息推送为基础的服务提供方式, 用于异地之间展示远端用户的桌面图像信息. 远程桌面同步利用网络打破了地域与硬件的限制, 使异地办公与协作成为可能, 被广泛应用在远程协助与视频会议等领域.

应用程序的逻辑执行和显示分离是远程桌面同步的实现基础, 此概念最早由 Citrix 公司引入, 并通过 ICA(Citrix Independent Computing Architecture)技术实现了应用程序的执行和显示分离. 此技术下, 应用程序在服务端执行, 通过相应的传输协议与网络为客户端提供界面显示. ICA 协议中, 客户端只提供简单的桌面展示(瘦客户端概念), 所有的处理器运算[1,2], 都可以在服务端进行. 桌面图像使用远程桌面同步协议通过网络与客户端的显示进行关联, 使客户端的用户体验如同使用本地运算一样. 远程桌面同步协议中, AT&T 剑桥所提出的 VNC(Virtual Network Computing), 由于具有良好的瘦客户端性与跨平台性, 在世界范围得到了广泛的发展. VNC 架构主要由客户端 VNC Client、服务端 VNC Server 与连接二者的 RFB 协议组成, 本文研究并分析了 VNC 的各组成部分, 并重点对 VNC 的两个主要部分: vnc server 与 RFB(Remote Frame Buffer)协议进行了分析与讨论.

1 VNC基本架构与桌面更新策略

1.1 VNC 的基本架构

VNC 主要用于实现计算机间的远程共享和远程操作, 它借助网络传送鼠标、键盘的消息以及桌面的更新图像数据. VNC 使用服务端的处理机资源进行实际的计算, 再通过瘦客户端对用户进行桌面展示. 从组成上看, VNC 架构是由服务端 VNC Server、客户端

① 基金项目:科技型中小企业技术创新基金(10C26215122841)
收稿时间:2016-03-07;收到修改稿时间:2016-04-21 [doi: 10.15888/j.cnki.csa.005475]

VNC Client 以及用于二者通讯的 RFB 协议三部分组成. 其中 VNC Server 使用真实的处理器资源进行实际桌面渲染运算, 并获取实时的显示图像更新, VNC Client 将接收到的图像更新到本地显存上, 而 RFB 协议则负责进行二者的数据传输与交互. 由于 RFB 协议工作在帧缓存级别, 直接操作显示器绘制前显存上的像素数据, 因此具有非常好的平台移植性, 可以广泛应用于各种窗口系统, 如: Linux、Windows 和 Mac 系统.

1.2 VNC 的更新推送机制

VNC 中对于更新区域的通知主要有 Push 和 Poll 两种机制[3]. Push 机制是服务端每隔 10ms 检查有无更新, 如果有更新则主动把更新推送至客户端, Poll 机制则是客户端通过发送主动 framebufferupdate 来请求某一个区域的更新, 服务端处理该消息并向服务端发送相应的更新.

考虑到带宽的原因, 默认情况下, VNC 采用的是惰性更新的 Poll 机制, VNC 具有一定的自适应性, 可以根据 Client 端的处理能力和网络情况, 来决定更新推送的频率.

2 VNC的桌面更新区域获取方法研究

VNC 对于桌面区域的同步主要有三个步骤: 桌面更新区域获取、桌面更新区域传输与客户端渲染, 其中前两步是 VNC 对桌面进行远程同步的核心. VNC 通过 VNC Server 完成对桌面更新区域的获取, VNCServer 主要通过两种方式获取桌面图像的更新: 基于 Hook 消息的桌面更新获取与基于 Mirror Driver 的桌面更新获取. 以下将详细介绍 VNC 对于桌面更新区域的获取方法.

2.1 基于 Hook 的图像增量获取策略

钩子(Hook)是一种 Windows 的消息处理机制, 应用程序可以利用此机制设置子进程以监视指定发向某些窗口的消息. Hook 函数可以监视其他进程创建的窗口, 消息到达后, 可以在在目标窗口处理函数之前处理它. 通过 Hook 机制, 应用程序可以截获处理 window 的消息或特定事件.

Windows 操作系统中, 桌面的改变也会产生对应的消息, 因此 VNC 协议可以通过 Hook 来截获 Windows 平台 GUI 的更新指令, 以获得桌面的更新信息. 通过 Hook 函数, VNC 可以获取更新区域的坐标, 将其理解为二维像素矩阵并拷贝, 再使用 RFB 协议对此矩阵进行传输, 具体执行过程如下[4]:

1) 使用 Wm_hooks 监视并截获桌面更新产生的特定通知消息, 并将其转化为自定义消息发送给 WMHooksThread 线程, 此消息即包含更新区域的坐标.

2) WMHooksThread 线程记录桌面中发生更新的区域位置信息, 并调用 WMHooksThread::run() 函数来获得桌面区域改变的大小, 从而定位发生改变的区域并记录到 SimpleUpdateTracker new_changes 的队列中, 最后通知拷贝函数.

3) 通过 rfb::win32::WMHooks::processMessage 函数通知 rfb::win32::WMHooks::setUpdateTracker 函数, 将当前 SimpleUpdateTracker new_changes 队列中的更新区域内图像数据拷贝到 SDisplay 中.

4) 把 SDisplay 中记录的图像像素数据发送给 VNCServerST 对象, 至此, 完成了 VNCSever 对更新数据的记录与拷贝, 此后, VNC 协议被动等待更新推送请求, 在得到更新请求后, 通过 RFB 协议传输 VNCServerST 对象中的更新图像数据至客户端, 并在客户端最终绘制展示给用户.

图 1 给出了基于 Hook 机制的 VNC 协议工作流程.

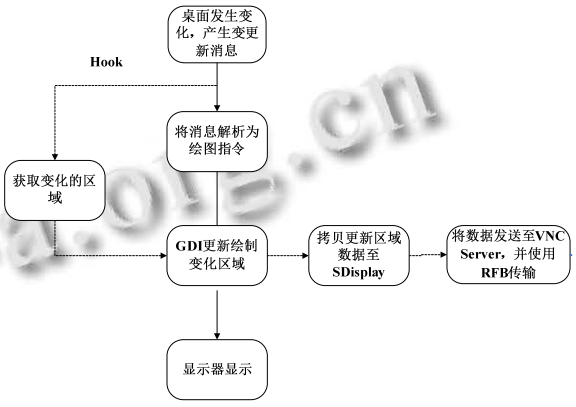

图 1 基于 Hook 的 VNC 协议的传输流程

2.2 基于 Mirror Driver 的图像增量获取策略

2.2.1 hook 机制获取桌面更新的局限性

基于 Hook 机制获取桌面更新的方式依赖的是桌面发生变化时所产生的消息,其最小更新单位为窗口的焦点区域, 但由于窗口内很少像素点发生变化, 都会产生窗口更新的消息, 因此此方法在传输中存在冗余数据, 同时, 桌面绘制指令解析为位图之后, 数据量也会产生一定膨胀, 也会产生额外的传输数据.

2.2.2 基于 Mirror Driver 的桌面更新区域获取

为解决 Hook 机制的不足, 在 Ultra VNC 及以上版本中使用了镜像驱动 Mirror Driver, 实现对桌面更新区域的获取.

Mirror Driver 重新实现了显卡驱动对于绘图指令的各种接口, 它可以像真实的显卡驱动一样, 获取到用于桌面绘制的绘图指令, 由于底层的绘图指令已经被精简为最小更新绘制指令, 因此可以有效的去除桌面同步传输的冗余数据, 并且相比于直接传输位图, 传输绘图指令可以更有效的对传输数据进行压缩.

在使用 Mirror Driver 的情况下, VNC 主要通过网络传输绘图指令, 而对于绘图指令中自带的位图数据, VNC 依旧使用 RFB 中的图像压缩算法, 对位图进行压缩后再传输.

基于 Mirror Driver 的桌面更新获取见图 2.

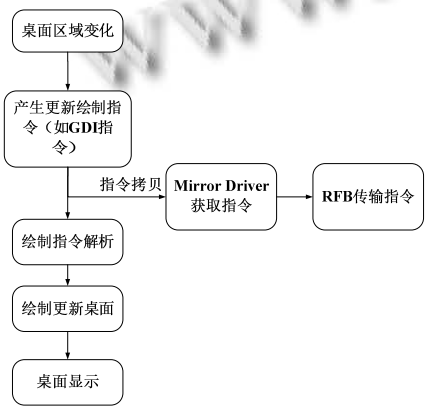

图 2　基于 Mirror Driver 的 VNC 协议的传输流程

3 基于RFB协议的图像压缩编码

RFB 协议负责对 VNC 更新区域获取的像素矩阵进行压缩并通络传输, 图像压缩算法是 RFB 协议的核心[5,6,7], 直接影响了远程桌面传输的延迟与带宽消耗. RFB 协议支持多种图像压缩算法, 包括 Raw、Copy Rectangle、RRE、CoRRE、Hextile 和 ZRLE 等, 以下简单介绍几种代表性的编码方式.

3.1 Raw 编码与 Copy Rectangle 编码

1) Raw 编码: Raw 编码是一种简单的编码方式, 它直接传送未经压缩的原始像素矩阵, 因此图像压缩率最低, 同等情况下的带宽消耗最大, 但由于算法没有编解码的额外计算开销,因此延迟与 CPU 消耗都较小.

2) Copy Rectangle 编码: Copy Rectangle 编码在客户端帧缓存中已拥有部分与更新像素相同的数据的环境中是非常有效的. 此时服务端只需向客户端发送此部分数据在客户端帧缓存中的位置坐标, 客户端则根据此位置坐标, 直接复制数据至更新区域并显示即可.

Copy Rectangle 编码并不独立的发送各个矩形分块, 对于各点像素值完全相同的一组矩形分块, 算法只发送第一个矩形的完整像素数据, 随后只需要发送此矩形更新后的左上角定位坐标(x,y). 因此算法只需编码第一个矩形分块的详细数据, 以后只更新此矩形的一系列定位坐标, 对于第一个矩形分块所使用的编码压缩算法则并不加以限制, 因此, Copy Rectangle 可和其它图像压缩方式结合使用.

3.2 RRE 编码与其扩展编码

RRE 编码充分的利用了图像的空间关联性, 即相邻像素的像素值往往相近或相等. 编码算法通过对具有相同或者相近的像素区域进行分块, 实现像素矩阵的压缩, 以 RRE 编码方式为基础, 先后出现了多种利用空间关联性的编码算法, 如 CoRRE 编码、Hextile 编码与 ZRLE 编码.

1) RRE 编码: RRE 编码将待传输矩阵依据空间关联性分成多个子矩形区域, 每个子区域由一个基本象素点和该区域的大小构成. 编码过后, 用于传输的数据流由一个背景象素值 Vb(图像区域内各象素的均值点)和 N 个子矩形区域构成, 每个子矩形的信息由 5 元素组<v, x, y, w, h>来表示. 其中 v 值为子矩形内的基本像素值, (x,y)是对应子矩形的坐标, 表示子矩形上-左的坐标值, (w,h) 表示子矩形的宽高, 将一个区域的像素值压缩为一个五元数组. 可见, RRE 编码是通过空间关联的方式, 在不影响或少影响图像视觉效果的前提下, 对图像进行了有损压缩处理.

图 3 为 RRE 编码方式对图像区域进行逐列划分子矩形区域的示意图, 其中 R1 至 R6 分别为具有相同背景像素值的子矩形分块.

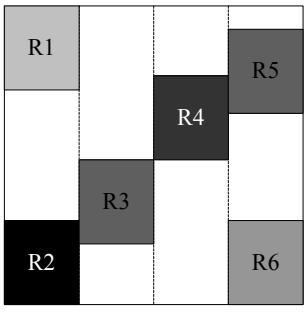

图 3　RRE 编码对于矩形区域的逐列划分示意图

图 3 只给出 RRE 对于矩形区域进行列分块的示意图, 实际应用中, RRE 编码会对矩阵块的行与列都进行分块, 其对于行分块的方法与列分块类似, 然后选取具有最佳压缩率的分块方式对矩阵块进行分块压缩并传输.

2) CoRRE 编码: CoRRE 编码是 RRE 编码的扩展形式, 其压缩性能要高于 RRE. CoRRE 通过限制 RRE 编码的窗体大小, 来降低编码压缩所用的字节, 从而达到比 RRE 更高的压缩率. 其假设每个子矩形的宽度和高度均小于 256, 以控制五元组中定位参数所占的字节位数, 从而节省表示每个矩形的参数值所需的最小空间. 实验证明, 当子矩形的大小为 48*48 时, CoRRE 编码的压缩率能达到最高值.

3) Hextile 编码: Hextile 编码是 CoRRE 编码的变形, 其压缩效率也高于 CoRRE 编码. 它将整个窗口划分为 16*16 块, 从而进一步压缩控制参数的空间, 可以只用 4 个 bits 来表示长度或宽度. 同时由于分块大小固定, 每一块的大小和位置不需要特别给出, 所有的数据流都按照一定的顺序排列. 解码时, 只需同样对数据流进行顺序解码即可. 与 RRE 编码类似, 算法中每个块数据流都有自己的背景象素点 Vb, 如果某块的 Vb 值与它前一个块相同的话, 可将此值缺省. 当块的所有子矩形象素值全部相同时, 则只需给出整个块的前景色象素值.

4) ZRLE 编码: ZRLE 编码是 Zlib 游程编码, 包含 zlib 压缩、分块、调色板和游程编码. ZRLE 编码传输的数据流是由 4byte 长度值和此长度值个字节的 zlib 压缩数据组成. 由于每个单一的 zlib 数据流都对应一个给定的 VNC 协议连接, 因此 ZRLE 矩形必须严格的按照顺序进行编码和解码. 没有压缩过的 zlib 数据类似于 Hexfile 编码, 其单位块大小为 64*64. 每一个块都包含一各字节的子编码类型标示位, 用来决定当前块的内部的具体编码方式. 通常情况下, ZRLE 编码也是和其它多种编码方式混合使用的.

4 结语

VNC Server 端通过 Hook 或者 Mirror Driver 的方式获取桌面区域更新的位图与指令信息, 并通过其 RFB 协议以 Push 或 Poll 方式传输这些更新数据, 由于基于 Hook 方式的 VNC 协议传输的数据为最底层的像素数据, 因此相比于基于 Mirror Driver 方式的 VNC, 具有更良好的瘦客户端特性, 但同时也会消耗更多的带宽.

对 VNC 协议的下一步研究与优化重点包括: 对于直接在显存上进行渲染的绘制方式(如 DirectX 的 D-Draw 方式), 由于基于 Hook 与基于 Mirror Driver 的 VNC 协议皆无法获取对应的桌面数据, 因此如何使 VNC 适应更多的绘制机制, 是对 VNC 进行优化与研究的重点; 当需同步的桌面区域为高分辨率、高变化率的流媒体图像时, VNC 的带宽消耗将十分惊人[8,9], 因此对于多媒体播放的场景, 可结合客户端硬件解码与流媒体重定技术, 在流媒体被播放器解码前, 即通过网络将视频流重定向至客户端并通过客户端硬件进行解码, 以在保证流媒体播放流畅度的前提下, 有效降低网络消耗; 同时, VNC 中 RFB 协议主要使用无损压缩算法对图像进行压缩, 因此可以在 RFB 协议中加入部分有损压缩算法(如: jpeg 压缩与降采样压缩), 在网络环境比较恶劣的情况下, 使用有损压缩, 在部分牺牲桌面图像质量的前提下, 尽可能的降低桌面同步的消耗带宽, 以适应更多的远程桌面同步环境需求.

参考文献

1 张跃冬,朱定局,宋振华等.一个面向分布式桌面计算环境的超级瘦客户端.计算机工程 2007,33(7):104–106.
2 刘胜辉,舒友村.虚拟网络计算在应用程序共享中的应用研究.微型机与应用,2009,(9):56–59.
3 吴筱桉.VNC 系统中 RFB 协议分析及视频播放性能改进[硕士学位论文].天津:南开大学,2008.
4 kfhzy, VNC 源码阅读--VNC 图像更新机制. http://blog.csdn.net/kfhzy/article/details/8053052.[2012-10-09].
5 熊俊杰.虚拟工作平台中的图形传输机制研究[硕士学位论文].武汉:华中科技大学,2008.
6 肖道举,刘洪峰,陈晓苏.面向远端屏幕监控的一种图像压缩传输方法.计算机工程与设计,2005,26(12):3356–3358.
7 梁飞蝶,李锦涛,史红周.虚拟网络计算(VNC)协议中的编码方法.计算机应用,2004,24(6):93–95.
8 左强翔,吴洁.一种基于分块采集和压缩技术的屏幕共享方案.计算机技术与发展,2008,18(4):206–209.
9 刘坚,余综.VNC 多媒体数据实时传输的研究与实现.计算机工程与设计,2012,33(7):2706–2710.