**中图分类号：TP311.5**

**论文编号：10006SY1106516**



硕 士 学 位 论 文

**云计算环境下桌面虚拟化协议SPICE的优化**

作者姓名 徐 浩

学科专业 计算机软件与理论

指导教师 兰雨晴 副教授

培养院系 计算机学院

**Optimization of Virtual Desktop Display protocol SPICE in Cloud Computing Environment**

A Dissertation Submitted for the Degree of Master

**Candidate：Xu Hao**

**Supervisor：Associate Prof. Lan Yuqing**

School of Computer Science and Engineering

Beihang University, Beijing, China

**中图分类号：TP311.5**

**论文编号：10006SY1106516**

硕 士 学 位 论 文

云计算环境下桌面虚拟化协议SPICE的

优化

作者姓名 徐 浩 申请学位级别 工学硕士

指导教师姓名 兰雨晴 职 称 副教授

学科专业 计算机软件与理论 研究方向 软件工程

学习时间自 年 月 日 起至 年 月 日止

论文提交日期 年 月 日 论文答辩日期 年 月 日

学位授予单位 北京航空航天大学 学位授予日期 年 月 日

关于学位论文的独创性声明

本人郑重声明：所呈交的论文是本人在指导教师指导下独立进行研究工作所取得的成果，论文中有关资料和数据是实事求是的。尽我所知，除文中已经加以标注和致谢外，本论文不包含其他人已经发表或撰写的研究成果，也不包含本人或他人为获得北京航空航天大学或其它教育机构的学位或学历证书而使用过的材料。与我一同工作的同志对研究所做的任何贡献均已在论文中作出了明确的说明。

若有不实之处，本人愿意承担相关法律责任。

学位论文作者签名：        日期： 年 月 日

学位论文使用授权书

本人完全同意北京航空航天大学有权使用本学位论文（包括但不限于其印刷版和电子版），使用方式包括但不限于：保留学位论文，按规定向国家有关部门（机构）送交学位论文，以学术交流为目的赠送和交换学位论文，允许学位论文被查阅、借阅和复印，将学位论文的全部或部分内容编入有关数据库进行检索，采用影印、缩印或其他复制手段保存学位论文。

保密学位论文在解密后的使用授权同上。

学位论文作者签名： 日期： 年 月 日

指导教师签名： 日期： 年 月 日

摘 要

近年来，云计算相关技术得到广泛发展，桌面虚拟化是典型的云计算应用。桌面虚拟化技术作为一种重要的云资源交付方式，将用户的桌面环境与远程终端解耦合，为用户提供了便捷的桌面环境，同时也为企业提供了集中的管理方式，减少企业的信息管理成本，提供数据的安全性。当前各大厂商分别推出了自己的桌面虚拟化解决方案，通过虚拟桌面显示协议为用户提供虚拟桌面环境，其中SPICE桌面虚拟化是一个开源的解决方案。

针对虚拟桌面的高清视频播放，目前并没有提供有效的解决方案，现有的解决方案都是由服务器进行视频解码并将解码后的视频帧发送用户终端进行显示，播放效果和本地视频播放有很大差距，同时对服务器以及网络带宽造成很大的压力。

论文首先介绍了桌面虚拟化相关的研究现状以及对虚拟桌面传输协议性能提升所做的改进与优化，然后详细分析了SPICE虚拟桌面架构及其关键技术，重点研究了SPICE协议的图像视频处理流程。在以上研究和分析的基础上，本论文主要研究成果如下：

1、针对网络带宽占用很高的问题，本文从视频压缩算法的角度进行改进，通过MJPEG2000算法对视频进行压缩，提高压缩效率；

2、本文通过增加窗口监听代理模块改进虚拟桌面的视频检测机制，提高视频检测的准确率；

3、为了提高视频播放的流畅性，本文在SPICE协议中引入流媒体视频传输机制，以流的方式传输视频，并在客户端进行视频的解码播放；

4、通过以上对SPICE协议视频播放功能进行改进和优化，实现了SPICEx协议。

论文的研究成果通过评估优化后的SPICEx协议视频播放的效果进行验证，分别对视频播放效果进行主客观质量评估。结果表明，SPICEx协议可以分别在低带宽和高带宽网络环境下提供良好的高清视频播放体验，有效地提升了虚拟桌面的用户体验。

关键词：桌面虚拟化，虚拟桌面显示协议，SPICE，MJPEG2000，流媒体

**Abstract**

In recent years, cloud computing technology has been widely developed, and desktop virtualization is a typical application of cloud computing. As an important delivery mode for cloud resources, desktop virtualization technology decouples the user's desktop environment and remote terminals, providing a convenient desktop environment for remote users, and meanwhile provides safe data and centralized management mode for enterprises, which reduces the total cost of enterprise information management. At present many manufacturers have introduced the desktop virtualization solution, and provide virtual desktop environment for users through the virtual desktop display protocols. The SPICE desktop virtualization is an open source solution.

However, there are still no efficient solutions for HD video playing. The current solutions rely on servers to decode videos and send video frames to user terminals to display. Video playing in virtual desktop solutions has a big disparity with local video playing, and meanwhile causes great pressure on the server and network bandwidth.

This paper firstly introduces the current research on desktop virtualization related and the improvement and optimization of the performance of virtual desktop protocol, and then analyzes in details the SPICE virtual desktop infrastructure and its key technology, mainly focus on image and video processing of the SPICE protocol. Based on the above research and analysis, the main research is as follows:

1. To address high bandwidth usage problems, this paper improves the video compression algorithm with the MJPEG2000 algorithm to compress the video, and the compression efficiency is improved;

2. The video detection mechanism is improved by adding an agent module monitoring video windows, increasing the detection rate;

3. In order to improve the fluency of video playing, the streaming media video transmission mechanism is introduced into the SPICEx protocol, where video data is transferred in flow mode, and videos is decoded directly on the client;

4. Finally, from the above three aspects this paper has improved and optimized the SPICE protocol in video playing, and implemented the SPICEx protocol.

To verify the research results of the optimized SPICEx protocol video playing, this paper evaluates the assessment respectively of both subjective and objective video quality. The results show that, the SPICEx protocol can provide a good HD video playing experience in both low and high bandwidth network environment respectively, and effectively improves the virtual desktop user experience.

**Key words**：desktop virtualization, virtual desktop display protocol, SPICE, MJPEG2000, streaming media

目 录

[第一章 绪论 1](#_Toc374701052)

[1.1研究背景 1](#_Toc374701053)

[1.2问题的提出 3](#_Toc374701054)

[1.3论文的研究内容及意义 4](#_Toc374701055)

[1.4论文的组织结构 5](#_Toc374701056)

[第二章 桌面虚拟化国内外研究现状 6](#_Toc374701057)

[2.1桌面虚拟化基础架构 6](#_Toc374701058)

[2.2虚拟操作系统基础架构 8](#_Toc374701059)

[2.3智能桌面虚拟化 9](#_Toc374701060)

[2.4虚拟桌面显示协议 11](#_Toc374701061)

[2.4.1 RDP协议 11](#_Toc374701062)

[2.4.2 ICA协议 13](#_Toc374701063)

[2.4.3 SPICE协议 14](#_Toc374701064)

[2.4.4 PCoIP协议 15](#_Toc374701065)

[2.4.5 RFB协议 16](#_Toc374701066)

[2.4.6 其他协议 17](#_Toc374701067)

[2.5虚拟桌面协议改进 17](#_Toc374701068)

[2.6本章小结 20](#_Toc374701069)

[第三章 SPICE虚拟桌面研究 21](#_Toc374701070)

[3.1 SPICE协议特性 21](#_Toc374701071)

[3.2 SPICE框架结构研究 22](#_Toc374701072)

[3.2.1 SPICE服务端结构 23](#_Toc374701073)

[3.2.2 SPICE客户端结构 24](#_Toc374701074)

[3.2.3 SPICE虚拟通道 25](#_Toc374701075)

[3.2.4 SPICE代理模块 26](#_Toc374701076)

[3.3 SPICE核心图像技术研究 27](#_Toc374701077)

[3.3.1 图像处理流程 27](#_Toc374701078)

[3.3.2 图形命令树 30](#_Toc374701079)

[3.3.3 智能压缩技术 30](#_Toc374701080)

[3.3.4 自适应图形处理 32](#_Toc374701081)

[3.4 本章小结 33](#_Toc374701082)

[第四章 SPICEx视频优化研究与实现 34](#_Toc374701083)

[4.1 相关技术基础 34](#_Toc374701084)

[4.1.1 MJPEG2000算法 34](#_Toc374701085)

[4.1.2 流媒体与流式传输协议 39](#_Toc374701086)

[4.1.3 GStreamer多媒体框架 41](#_Toc374701087)

[4.2 视频压缩算法改进 42](#_Toc374701088)

[4.3 视频区域检测增强 46](#_Toc374701089)

[4.4 流媒体传输机制实现 47](#_Toc374701090)

[4.4.1 流媒体服务端的整体设计 47](#_Toc374701091)

[4.4.2 GStreamer视频播放 50](#_Toc374701092)

[4.5 本章小结 52](#_Toc374701093)

[第五章 SPICEx视频优化验证 53](#_Toc374701094)

[5.1视频质量评估 53](#_Toc374701095)

[5.1.1主观质量评估方法介绍 53](#_Toc374701096)

[5.1.2客观质量评估方法介绍 54](#_Toc374701097)

[5.1.3SPICEx视频质量评估 55](#_Toc374701098)

[5.2测试方案 55](#_Toc374701099)

[5.2.1选取测试对象 55](#_Toc374701100)

[5.2.2测试环境配置 56](#_Toc374701101)

[5.2.3主观质量评估方案 57](#_Toc374701102)

[5.2.4客观质量评估方案 58](#_Toc374701103)

[5.3测试结果分析 59](#_Toc374701104)

[5.3.1主观质量评估结果 59](#_Toc374701105)

[5.3.2客观质量评估结果 62](#_Toc374701106)

[5.4本章小结 68](#_Toc374701107)

[结论 69](#_Toc374701108)

[论文成果 69](#_Toc374701109)

[进一步工作 70](#_Toc374701110)

[参考文献 71](#_Toc374701111)

[附录 75](#_Toc374701112)

[攻读硕士学位期间取得的学术成果 78](#_Toc374701113)

[致谢 79](#_Toc374701114)

**图 目**

[图1 RDP协议的层次架构 11](#_Toc374650084)

[图2 RemoteFX技术架构 12](#_Toc374650085)

[图3 ICA虚拟通道原理 13](#_Toc374650086)

[图4 混合虚拟桌面协议处理流程 18](#_Toc374650087)

[图5 基于调度的屏幕刷新模式 19](#_Toc374650088)

[图6 SPICE协议消息格式 22](#_Toc374650089)

[图7 SPICE框架结构图 22](#_Toc374650090)

[图8 SPICE协议VDI通信模型 24](#_Toc374650091)

[图9 SPICE客户端主要类图结构 25](#_Toc374650092)

[图10 Agent消息处理流程 26](#_Toc374650093)

[图11 SPICE图像处理流程 27](#_Toc374650094)

[图12 SPICE图像处理子系统 28](#_Toc374650095)

[图13 SPICE服务端图像处理 29](#_Toc374650096)

[图14 SPICE处理显示数据顺序图 31](#_Toc374650097)

[图15 SPICE图形命令优化示意图 31](#_Toc374650098)

[图16 JPEG2000图像编码解码过程 35](#_Toc374650099)

[图17 JPEG2000编码步骤 36](#_Toc374650100)

[图18 JPEG2000小波变换分层示意 37](#_Toc374650101)

[图19 Motion JPEG2000视频压缩流程 38](#_Toc374650102)

[图20 RTSP服务器与客户端交互过程 40](#_Toc374650103)

[图21 GStreamer单向视频通信过程 42](#_Toc374650104)

[图22 数字图像组成结构 43](#_Toc374650105)

[图23 SPICE的图像视频压缩 43](#_Toc374650106)

[图24 SPICEx的图像视频压缩 43](#_Toc374650107)

[图25 Live555转发模式顺序图 48](#_Toc374650108)

[图26 Media服务端主要类图结构 49](#_Toc374650109)

[图27 SPICEx协议涉及的主要类 50](#_Toc374650110)

[图28 测试环境示意图 57](#_Toc374650111)

[图29 测试视频1主观评估结果 60](#_Toc374650112)

[图30 测试视频2主观评估结果 60](#_Toc374650113)

[图31 测试视频3主观评估结果 61](#_Toc374650114)

[图32 测试视频1数据传输量 62](#_Toc374650115)

[图33 测试视频2数据传输量 62](#_Toc374650116)

[图34 测试视频3数据传输量 63](#_Toc374650117)

[图35 测试视频1视频质量 63](#_Toc374650118)

[图36 测试视频2视频质量 64](#_Toc374650119)

[图37 测试视频3视频质量 64](#_Toc374650120)

[图38 测试视频1的实时PSNR 65](#_Toc374650121)

[图39 测试视频1平均PSNR 65](#_Toc374650122)

[图40 测试视频2平均PSNR 66](#_Toc374650123)

[图41 测试视频3平均PSNR 66](#_Toc374650124)

[图42 平均网络带宽使用情况对比 67](#_Toc374650125)

[图43 平均CPU使用情况对比 67](#_Toc374650126)

[图44 平均内存使用情况对比 68](#_Toc374650127)

**表 目**

[表1 VDI与IDV架构对比 10](#_Toc374695010)

[表2 几种编码方式的性能和功能比较 34](#_Toc374695011)

[表3 视频主观质量的评分标准 53](#_Toc374695012)

[表4 测试视频1详细参数 55](#_Toc374695013)

[表5 测试视频2详细参数 56](#_Toc374695014)

[表6 测试视频3详细参数 56](#_Toc374695015)

[表7 测试环境配置 56](#_Toc374695016)

[表8 本地视频播放评估结果 75](#_Toc374695017)

[表9 100M带宽MJPEG视频播放评估结果 75](#_Toc374695018)

[表10 100M带宽MJPEG2000视频播放评估结果 76](#_Toc374695019)

[表11 100M带宽流媒体视频播放评估结果 77](#_Toc374695020)

# 第一章 绪论

本章简要介绍了论文的研究背景、论文研究问题的提出、论文研究的主要内容和论文的组织结构。

## 1.1研究背景

近年来，云计算(Cloud Computing)已成为一种新的IT资源交付和使用模式，提供了“无所不能的计算和存储”，对基于网络的、可配置的共享计算资源池能够方便的随需访问，打破了地域和空间的束缚[[[1]](#endnote-2)]。云计算将传统的以桌面为核心的任务处理转化为以网络为核心的任务处理，利用互联网中的计算系统来支持互联网各类应用。采用创新的计算模式使客户通过互联网随时获得近乎无限的计算能力和丰富多样的信息服务[[[2]](#endnote-3)]。云计算是基于互联网的相关服务的使用和交付模式，通常涉及通过互联网来提供动态易扩展且经常是虚拟化的资源。狭义的云计算指IT基础设施的交付和使用模式，指通过网络以按需、易扩展的方式获得所需资源；广义云计算指服务的交付和使用模式，指通过网络以按需、易扩展的方式获得所需服务，这种服务可以是IT和软件、互联网相关，也可是其他服务，意味着计算能力也可作为一种商品通过互联网进行流通[[[3]](#endnote-4)]。

云计算模式中的核心技术是虚拟化技术，虚拟化实现了IT资源的逻辑抽象和统一表示，在大规模数据中心管理和解决方案交付方面发挥着巨大的作用，是支撑云计算的重要技术基石。虚拟化技术是伴随着计算机技术的产生而出现的，随着虚拟化技术的普及，出现了数据中心部署和管理的方式。数据中心的虚拟化技术主要有服务器虚拟化、网络虚拟化、应用虚拟化、桌面虚拟化(Desktop Virtualization)等[[[4]](#endnote-5)]。桌面虚拟化作为云计算环境中的虚拟化技术之一，是云中资源交付的重要方式之一，通过封装向远程客户终端设备提供完整的桌面信息环境。桌面虚拟化技术将用户的桌面环境与终端设备分离，服务器上存储的是用户完整的桌面环境，用户可以通过不同的终端设备访问该桌面环境。桌面虚拟化技术将终端的资源迁移到数据中心进行集中管理。

桌面即服务(Desktop as a Service, DaaS)[[[5]](#endnote-6)]是一种交付和管理桌面的服务模型，也称为云计算桌面模式，DaaS为业务部门提供了快速部署和按需扩展其虚拟桌面环境的能力。虚拟桌面是典型的云计算应用，可以为用户提供部署在云端的远程计算机桌面服务，即通过在虚拟桌面服务器上运行用户所需的操作系统和应用软件，采用桌面交付协议将操作系统桌面视图以图像的方式传送到用户端设备上显示。同时，用户端的输入通过网络传递到服务器侧进行处理，并更新桌面视图内容。本质上看，虚拟桌面是一种将个人计算机桌面与物理计算机相隔离的技术。理想情况下，计算机桌面均由网络中的服务器提供而非本地桌面环境，所有程序的执行和数据存取都在数据中心服务器中完成，用户可以通过网络访问虚拟桌面并获得与使用本地计算机桌面相近的体验。但是当虚拟桌面需要传输的数据量非常大时，比如播放高清视频，对网络带宽、CPU负载都是一个挑战，严重影响用户体验，目前的解决办法是提高客户端的硬件配置，添加相应的硬件辅助完成视频解码。

在当前的桌面云计算解决方案中，虚拟桌面基础架构(Virtual Desktop Infrastructure, VDI)是主流的架构与部署方式。VDI基于底层的硬件资源和中间层的虚拟化软件，通过上层的桌面虚拟化应用软件，为终端用户提供虚拟桌面交付模式。VDI解决方案中所提供的桌面虚拟化不仅充分利用了服务器端的计算能力，而且提供了强大而灵活的用户PC管理能力，同时具有很强的可扩展性和显著的能源节约优势，解决了传统个人计算机管理模式中的很多问题。VDI解决方案的主要设计思想是通过虚拟化技术，把传统意义上的个人物理PC转换为运行在虚拟化服务器上的一个个虚拟桌面，通过提供对底层存储、网络等基础资源和上层虚拟桌面的集中管理和连接功能，为给终端用户提供灵活和良好的虚拟桌面使用体验。

桌面虚拟化是一种基于中心服务器的计算模型，建立在服务器虚拟化的基础之上，沿用了瘦客户端模型，所有的桌面虚拟机在数据中心进行托管并统一管理，同时用户能够获得完整的PC使用体验，用户可以根据自己的需要使用虚拟桌面服务[[[6]](#endnote-7)]。系统管理员只需要维护部署在中心服务器上的系统即可，不需要更新升级客户机上的系统。桌面虚拟化与传统的远程桌面的关键不同之处在于，远程桌面技术是接入到一个真正安装在物理机上的操作系统，如果要大面积作为企业应用技术，不仅需要一个个PC运行操作系统，还需要投入相应的很多终端，所以远程桌面仅作为远程控制和远程访问的一种方式。采用桌面虚拟化极大地降低了整体IT的采购成本和运维成本，提高系统的安全性和硬件系统的利用率。其中虚拟桌面显示协议是VDI解决方案的核心，其最终目的是为用户提供接近传统PC操作系统的桌面服务体验。当前主流的虚拟桌面技术厂商已经确定了各自主打的桌面显示协议，不同的协议在应用效果、用户体验方面各有特色。虚拟桌面显示协议工作在OSI七层网络协议架构中的表示层，和其他表示层协议一样，其主要工作是作为上层的应用程序和底层网络之间的翻译层。对于桌面交付协议，表示层的关键任务是对远程操作系统桌面输出以及对客户端设备输入的编/解码。

远程桌面技术可以说是桌面虚拟化技术的前身。内置在Windows中的远程桌面使用了远程桌面协议RDP(Remote Desktop Protocol)，用户可以从其他的电脑上远程登录、访问与使用目标桌面。在虚拟化技术日益成熟后，微软将Windows Server 2008上的Terminal Services重新定义为演示虚拟化技术。从技术演进发展的时间来看，2006年桌面虚拟化技术概念才开始形成，2007年出现了第一代VDI的方案，2008年开始出现比较完整的解决方案。目前桌面虚拟化对应的虚拟系统不直接与硬件打交道，而是通过一个中间管理层来进行资源的协调，使得虚拟系统不会独占硬件资源。桌面虚拟化技术进一步发展，会实现虚拟桌面的网络化和集中化，虚拟桌面操作系统将被存储在网络上，进行集中化的管理。这样，用户的访问环境和运行环境分离开，用户将可以通过任何设备对网络上的桌面进行操作与访问,桌面通过服务的形式提供给最终用户。

在VDI模式下，数千个桌面环境从用户终端迁移到数据中心，数据中心需要实时地将桌面图像返回到用户终端，同时还要处理用户的输入信息，这些对网络性能提出了很高的要求。在目前的带宽条件下，用户想获得与使用本地桌面相同的体验还是有一定困难的，尤其是视频、3D图像以及多媒体应用等。因此，目前各大厂商围绕虚拟桌面显示协议展开竞争，使用各种技术对显示协议进行优化，减少需要传输的数量，提升用户使用体验。

## 1.2问题的提出

传统的个人计算机在使用过程中存在着诸多问题，例如不能随时随地运行和访问计算机上的应用和数据，维护规模化的计算机成本很高，也不能保证数据的安全性。因此出现了VDI虚拟桌面基础架构，但是VDI只能在一定程度上解决上述问题，同时VDI本身也存在着固有的缺陷。VDI模式中终端对于服务器的压力非常大，同时VDI利用硬件仿真以及桌面传输协议，使得视频、Flash、IP语音(VoIP)以及其它计算或图形密集型应用不适用于该模式。目前VDI对于视频播放的解决方案一般是在服务端进行视频解码，然后将解码后的视频帧压缩后传输到远程用户终端进行渲染显示，对于要求高清无损画面的应用，比如医学影像，VDI并不能满足需求，因为视频数据在压缩时会有数据损失，同时在低带宽网络环境中会丢失数据帧。

SPICE桌面虚拟化是一个典型的开源VDI解决方案，其中独立计算环境简单协议(Simple Protocol for independent Computing Environment, SPICE)协议定义了服务器和用户终端之间数据传输的格式，通过无损压缩算法来确保桌面图像的质量，同时为了降低网络带宽占用量，使用有损压缩算法MJPEG进行视频数据压缩。在虚拟桌面运行的过程中，SPICE协议会进行视频检测，一旦检测到视频播放，就会切换到MJPEG视频压缩算法。视频检测通过监测某个区域发生变化的频率，如果一个区域更新频率超过特定值，则判定当前区域为视频区域，对于该区域使用有损视频压缩算法进行压缩。如果视频画面更新频率降低，则会切换到无损图像压缩算法。一次视频播放如果频繁发生压缩算法切换，会导致视频画面的停滞和抖动，严重影响用户体验。同时，SPICE虚拟桌面需要占用非常高的带宽才能保证一定的用户体验。

综上所述，本文将要解决的问题如下：

1. 视频播放窗口的检测机制改善，视频的检测机制会在很大程度上影响到视频播放的效果，因此良好的视频检测机制能够改善用户体验。比较好的实现方式是，在检测到视频播放时就不要再轻易地切换成图像压缩，直到视频播放完毕。
2. 改进SPICE协议的视频压缩算法，降低视频传输网络带宽，通过使用高效的视频压缩算法改进SPICE协议，降低视频播放时传输的数据量，提高在低带宽网络环境下虚拟桌面的性能。
3. SPICE协议中集成流媒体视频播放，提高网络状况较好时高清视频播放效果。

## 1.3论文的研究内容及意义

本论文的研究目标是优化现有的开源虚拟桌面显示协议SPICE，实现虚拟桌面显示协议SPICEx，从协议本身的软件实现上进行调优，为用户提供良好的视频播放体验，从而降低用户终端以及网络环境的配置要求，为企业级的桌面虚拟化解决方案降低设备采购成本，提供更灵活高效的访问方式。具体而言，包括：

1. 桌面虚拟化框架相关的虚拟化技术研究
2. 虚拟桌面显示协议的分析与研究
3. SPICE协议原理研究
4. JPEG2000压缩算法研究
5. 流媒体视频传输实现机制的研究
6. SPICE协议视频优化方法研究与实现

本论文的研究意义在于通过减少虚拟桌面视频播放的传输数据量，降低对网络带宽的要求，可以在无线网络环境下实现虚拟桌面，同时进一步提高虚拟桌面视频播放的质量，扩大了虚拟桌面的应用范围，具有较高的理论与实际意义。

## 1.4论文的组织结构

全文由六部分组成：

第一章：绪论。阐述论文的研究背景并提出论文研究所面临的主要问题，以及论文的主要内容和组织结构。

第二章：桌面虚拟化国内外研究现状。阐述了虚拟桌面基础架构、虚拟操作系统基础架构、智能桌面虚拟化以及虚拟桌面显示协议等方面的国内外研究发展现状。

第三章：SPICE虚拟桌面研究。首先总结了SPICE协议的特性，然后详细研究了SPICE的框架结构，接着重点分析了SPICE的核心图像技术。

第四章：SPICEx视频优化研究与实现。在对SPICE协议优化研究之前，总结了相关的技术基础，主要包括MJPEG2000压缩算法研究以及流媒体和流式传输协议研究，然后分别从视频压缩算法、视频区域检测机制和流媒体传输机制3个方面对SPICE协议进行改进，实现SPICEx虚拟桌面显示协议。

第五章：SPICEx视频优化验证。首先总结了视频质量的评估方法，然后描述了对SPICEx协议进行优化验证采取的实验测试方案，最后对测试的结果进行了分析与总结。

结论：总结论文的研究成果，并对进一步的研究工作进行探讨和展望。

# 第二章 桌面虚拟化国内外研究现状

本章对论文涉及到的桌面虚拟化问题进行详细的国内外现状分析，具体包括：桌面虚拟化基础架构、虚拟操作系统基础架构、智能桌面虚拟化、虚拟桌面显示协议以及对协议的改进。

## 2.1桌面虚拟化基础架构

桌面虚拟化基础架构VDI通过服务器虚拟化提供管理和安全优势，为整个桌面映像提供集中化的管理。基于VDI的虚拟桌面解决方案是在服务器端为每个用户提供专用的虚拟机并部署用户所需要的操作系统和各种应用，然后通过桌面显示协议将完整的虚拟机桌面交付给远程用户。目前有很多厂商提供基于VDI的虚拟桌面解决方案，主要有Citrix、VMware和微软，红帽等。在国内，包括IBM中国研发中心，TCL多媒体中心，美的，格力，中国石化，广东移动，中国海洋大学，西南财经大学，中国人民大学，泰康资产等也都已经采用桌面虚拟化技术来替代传统PC的使用,其发展势头有超过服务器虚拟化的趋势[[[7]](#endnote-8)]。

Citrix的桌面虚拟化解决方案主要包括XenDesktop、XenApp和XenServer。XenServer作为服务器虚拟化平台，用户的虚拟桌面运行于XenServer上，通过XenApp向用户交付应用，通过XenDesktop将虚拟桌面交付给用户。XenDesktop可以运行在其他服务器虚拟化平台上，如Hyper-V、VMware vSphere。XenDesktop使用高效的独立计算结构ICA (Independent Computing Architecture)协议，拥有较高的远程桌面访问的效率。Citrix进一步推出了HDX技术[[[8]](#endnote-9)]，支持本地设备即插即用、高清视频播放、VoIP双向语音等功能，增强了XenDesktop在低带宽网络环境下的虚拟桌面体验。

作为服务器虚拟化的主导者，VMware提供了桌面虚拟化产品VMware View，运行在服务器虚拟化平台VMware vSphere和VMware ESX之上，可以通过桌面系统模版和主虚拟镜像文件，创建基于管理策略的虚拟桌面并分配给连接的用户使用。最新的VMware View版本为用户提供了在本地系统运行受控虚拟桌面的功能，即使用户在离线的情况下，仍然可以操作分配得到的虚拟化桌面[[[9]](#endnote-10)]。

微软在桌面虚拟化领域提供了本地虚拟化和远程虚拟化两种解决方案，本地虚拟化如 MED-V[[[10]](#endnote-11)]允许管理员在任何基于Windows的桌面上创建、交付和管理公司VPC镜像，主要用于解决应用程序与操作系统的不兼容问题。而远程虚拟化方案基于Hyper-V服务器虚拟化平台，通过Windows Server RDS(Remote Desktop Service)远程桌面服务交付虚拟桌面，由 RemoteApp、App-V for RDS交付应用程序。该解决方案同样支持动态地根据需求使用配置好的虚拟镜像向用户提供虚拟桌面，在服务器端进行更新、管理[[[11]](#endnote-12)]。

红帽企业级虚拟化桌面是一个端到端的桌面虚拟化解决方案，为企业部署虚拟化桌面提供了所需的各个方面，提供跨平台的Windows和Linux虚拟桌面支持。红帽虚拟化解决方案建立在基于包含基于内核的虚拟机KVM(Kernel-based Virtual Machine)技术的红帽企业级Linux之上，有集中的管理控制台，拥有一套全面的管理工具，供管理员用于创建、监控和维护虚拟桌面。SPICE是红帽企业虚拟化桌面版使用的自适应远程桌面交付协议，用于将用户与虚拟桌面进行连接。SPICE是一个具有3层架构的协议，具体包括QXL驱动、SPICE客户端、QXL设备。SPICE最大特点就是其架构中增加了位于KVM中的QXL设备。这种架构使得协议实现与虚拟桌面下层的服务器虚拟化软/硬件基础设施产生较为紧密的依赖关系，不利于虚拟桌面的扩展。

斯坦福大学课题研究组成立的公司MokaFive目前主要业务是为用户提供桌面即服务平台的虚拟化桌面产品，研发出自有产品Mokafive Suite和Mokafive Service。MokaFive Suite是一款端到端的桌面解决方案，可以安置在桌面即服务平台里面，企业或服务的提供商可以通过MokaFive Suite为用户们提供和管理虚拟桌面。

桌面虚拟化技术在国内推广的进程也很迅速，人们对桌面虚拟化的认知度相比以前有了很大的提高，企业对桌面虚拟化所带来的诸多优势不再持有怀疑的态度，桌面虚拟化技术的优势如资源运用更加合理充分，在管理上的高效性等在此次事件中都得以充分体现。不过就目前国内的形式来看，真正自主开发实现桌面虚拟化解决方案的厂商还不是很多。目前国内提供桌面虚拟化技术解决方案厂商有联想、清华同方、深信服、威邦、京华科讯等，它们提供的桌面虚拟化产品采用的桌面显示协议是RDP协议或VDP协议或是基于开源协议进行的扩展。

典型的国内桌面虚拟化解决方案有联想公司的LVCC(Lenovo Virtual Client Cloud)联想虚拟云终端系统，该系统方案是由联想集团服务器事业部自主研发的虚拟云计算解决方案，对桌面/应用/用户数据进行统一管理、统一存储、统一计算，并向云终端用户交付Windows虚拟桌面/应用等云计算服务。LVCC是一套集服务器、终端硬件和操作系统、服务器存储、传输控制协议、管理中心于一体的整套解决方案，是目前市场整合度最高的桌面云方案产品。联想桌面云方案产品LVCC基于微软Windows Server 2008R2 VDI架构的远程桌面服务,采用最新的桌面虚拟化技术和管理平台，将桌面操作系统、应用和数据等，传统上由终端个人电脑处理的事务迁移到了后台数据中心[[[12]](#endnote-13)]。联想LVCC解决方案中的桌面交付协议采用的是联想自主研发的LXT协议，进一步促进了国内虚拟化技术的发展。

深信服桌面虚拟化aDesk方案[[[13]](#endnote-14)]为企业提供端到端的一站式解决方案，通过将用户桌面在数据中心集中化运行和管理，极大地降低了运维难度并提高了数据的安全性，同时实现了用户桌面在各种终端上的任意切换。深信服一站式桌面云解决方案包含虚拟机管理软件VMS、虚拟桌面控制器VDC及瘦终端aDesk三大组件。

威邦Vebula虚拟桌面产品是一套完善的桌面云VDI应用，能够在云中为用户提供远程的计算机桌面服务。服务供应商在数据中心的服务器上虚拟出用户所需的操作系统和应用软件，通过采用Vebula-Fit协议，将桌面视图以自适应网络硬件环境的视频流和图像的方式传送到用户终端设备上。通过Vebula-Fit协议，在服务器端可以根据服务器集群网络及存储环境的性能以及提供虚拟化主机的计算资源进行智能选择性适应。在用户终端根据用户应用和网络环境自适应地改变连接效果和性能[[[14]](#endnote-15)]。

DesktopCloud是京华科讯推出的桌面云软件系统，能够帮助客户以更大的灵活性，更低的成本和更快速的部署、运行和管理自有的私有云、公有云平台，能够按需交付计算资源，快速轻松地扩大和缩小云的规模。DesktopCloud允许用户跨越可大规模扩展的、异构的虚拟化基础架构，基于用户请求为一系列应用工作负载动态分配计算资源。全自动的云计算调度流程能够满足用户安全需求并且定制满足用户应用需求。京华科讯通过Free VDI协议传输虚拟桌面，Free VDI 协议通过网络多通道技术，将图像显示，键鼠，音视频数据分别进行处理，提高了数据安全性，以独有的压缩算法节省了网络的带宽，独特的高清视频支持，极大地提升了用户的桌面体验。

## 2.2虚拟操作系统基础架构

随着人们对虚拟桌面性能要求的不断提高，虚拟操作系统基础架构(Virtual OS Infrastructure, VOI)成为一个新的桌面虚拟化发展方向。VOI的概念由IBM的Steve Mills最早提出，VOI可以看作桌面虚拟化发展的一个新阶段。VOI桌面虚拟化技术不仅可实现基于服务端的虚拟操作系统、虚拟应用、用户配置的按需交付，也可基于客户端的本地缓存，这样就可利用终端资源处理图形密集型应用，比如音视频、3D图形应用、大型设计及工业软件等，也可支持各种计算机外设以适应复杂的应用环境以及未来的应用扩展。同时，VOI对网络和服务器的依赖性将大大降低，在网络中断或服务器故障后终端也可继续离线使用，数据可在云端集中存储，也可在本地加密存储，且终端应用数据不会因网络或服务端故障而丢失。从实际应用方面考虑，VOI具有更强和更符合用户环境的适应性。尤其是可脱离服务器和网络运行的特点，让VOI具有了更好的可用性和安全性以及更好的用户体验。VOI架构是一个新的桌面虚拟化发展方向，国内由北京和信创天VEsystem首次开发了基于VOI的虚拟桌面架构。

VOI模式以客户端虚拟容器技术为底层基础，其中虚拟容器就是将终端彻底虚拟化，建成一个虚拟的容器池，在这个容器池上，再部署操作系统，由于终端已经被虚拟化，所以从操作系统角度来看，它们面对的“硬件”已经是虚拟的，这为企业大规模部署虚拟机打下基础。因为客户端已经被虚拟化，所以，虚拟机即可以在服务器端也可以在终端运行，对于服务器端来说，就不需要进行太大的运算量，企业的数据中心规模可以得到缩减。也就是说，在客户端虚拟容器模式下，企业业务的计算需求是通过终端和服务器端来共同负担。

VOI从桌面应用交付提升到了操作系统的标准化与即时分发，与传统VDI设计的不同之处在于终端对本机系统资源的充分利用不再依靠于GPU虚拟化与CPU虚拟化技术，而是直接在I/O层实现对物理存储介质的数据重定向，以达到虚拟化的操作系统完全工作于本机物理硬件之上，从驱动程序、应用程序到各种设备均不存在远程端口映射关系，而是直接的内部址。

## 2.3智能桌面虚拟化

智能桌面虚拟化(Intelligent Desktop Virtualization, IDV)最早由Intel提出，可以理解为下一代桌面虚拟化或者桌面虚拟化的另一个发展分支。与VDI下所有桌面计算资源高度集中在数据中心、虚拟桌面发送到终端设备不同，IDV采取的是分布式方法来满足客户对虚拟桌面的高可靠性和高性能的要求，同时集中和简化管理和部署功能。IDV解决方案在确保用户尽享高性能、移动性和灵活性的同时，提供IT人员控制和保护桌面映像和设备的能力。

操作系统虚拟化(Operating Systems virtualization, OSV)是IDV的一个具体实现，其核心是由操作系统虚拟化发展而来，在不改变用户使用习惯、不改变PC性能、不增加额外硬件投入的情况下，实现PC桌面从分散转向集中管理，数据的集中保存和应用的集中下发。这种解决方案常用于一些需要重型应用的工作场合，如研发单位，军工企业的高安全性场合以及复杂应用的教育行业。OSV将操作系统和用户数据进行虚拟化，集中存储于中心服务器，宿主机可以是物理PC机，也可以是虚拟机。通过集中式桌面管理、本地运算的方式，可以管理成千上万台终端，投资成本仅相当于VDI的1/10，部署规模的越大，整体拥有成本则越小。OSV智能桌面虚拟化使得管理用户计算的整个系统变得更加智能，在最大化用户体验的同时给IT人员提供所需的管理功能。VDI与IDV架构对比如表1所示。

表1 VDI与IDV架构对比

|  |  |  |
| --- | --- | --- |
| 对比项 | VDI | IDV |
| 管理 | 集中管理 | 集中管理 |
| 运行 | 集中运行 | 本地运行 |
| 性能 | 不能保证一致性 | 接近本地系统 |
| 能否离线运行 | 否 | 是 |
| 终端安全性 | 高 | 高 |
| 数据中心成本 | 高 | 低 |
| 网络成本 | 高 | 低 |
| 存储成本 | 高 | 低 |

VMware收购Wanova并推出Mirage作为IDV产品，它运行Windows操作系统，并安装一个2MB客户端，虚拟机镜像保存到运行Windows 2008 R2的Mirage服务器上。Mirage不仅上传系统的完整图像，还能确定Mirage服务器上已经存在的文件和数据，然后只传送有差异的部分到用户终端，这个过程叫“三角洲计算”，这样可以节省带宽和存储。同时，Mirage客户端提供整个系统的硬软件清单，可以更容易地进行管理。如果某个系统出现了问题，也可以更加容易地进行灾难恢复。如果有人出差时遗失了一台笔记本电脑，IT管理员可以拿出一个新的，安装Windows和Mirage客户端，并通过局域网或广域网，将保存在Mirage服务器上的数据，同步到新的笔记本电脑上。它还可以恢复PC镜像到虚拟机。

## 2.4虚拟桌面显示协议

### 2.4.1 RDP协议

RDP协议是微软虚拟桌面产品中采用的显示协议。在应用过程中，在服务器侧用于生成远程桌面屏幕显示内容的图像设备接口GDI指令被RDP驱动截获，在服务器侧进行渲染，然后以光栅图像的形式传送到用户终端上输出。同时，用户终端上安装RDP协议的客户端把用户通过鼠标、键盘等设备输入的信息通过RDP重定向到服务器侧， 进而在服务器侧使用相应的驱动进行处理。RDP协议是在国际电信联盟ITU T.120协议族的基础上进行的扩展，通过建立多个独立的虚拟通道，承载不同的数据传输和设备通信，其总体架构[[[15]](#endnote-16)]如图1所示：



图1 RDP协议的层次架构

RDP协议每一层的作用如下：

1. 传输层：用于处理数据传输，管理连接过程，基于传输层，RDP协议能够提供多播服务，支持点到点和点到多点的连接；
2. 安全层：由加密和签名算法以及服务组成，防止未经认证的用户对RDP连接进行监控。RDP协议采用RC4算法进行加密，同时采用MD5和SHA-1组合算法进行签名；
3. 虚拟通道复用层：多个虚拟通道可以复用同一个RDP连接，虚拟通道具有可扩展性，每个虚拟通道内部可以增加新的内部属性，也可以开放给第三方使用；
4. 压缩层：利用压缩算法(微软的点对点压缩协议)针对各个虚拟通道的数据进行压缩。

随着用户对虚拟桌面体验的要求不断提高，微软在windows server 2008 R2的远程桌面服务Remote Desktop Service(RDS)产品中提出了RemoteFX技术，对RDP协议进行增强。RemoteFX技术通过提供虚拟3D显示适配器、智能编/解码和USB重定向等技术为用户提供良好的桌面体验，已经应用在微软的VDI和SBC虚拟桌面解决方案中。VDI解决方案对RemoteFX的应用能全面体现RemoteFX的技术特征，RemoteFX的技术架构如图2所示：



图2 RemoteFX技术架构

RemoteFX是与RDP7.1以及之后版本的RDP协议整合使用的，其中RDP协议为RemoteFX提供加密、认证、管理和设备支持等功能。RemoteFX需要与微软的服务器虚拟化技术Hyper-V集成，其图像处理组件分别运行在Hyper-V的父分区和子分区。父分区包括RemoteFX的管理组件，用于管理图像的处理过程。在子分区中运行的主要有虚拟GPU。GPU虚拟化是RemoteFX增强技术的核心，当虚拟机中的应用通过DirectX或GDI调用图像处理操作时，相关命令将传递给虚拟GPU，然后由虚拟GPU将命令从子分区传递给Hyper-V的父分区并在物理GPU上高效处理。最新的RDP协议是8.0版本，其新特性包括：

1. 通过RemoteFX for WAN功能，WAN网络上的用户可以使用智能与自适应 UDP传输、网络丢包率和恢复等高级技术获取快速流畅的体验；
2. RemoteFX 网络自动检测功能会自动检测网络特征并相应地优化用户体验；
3. RemoteFX 自适应图形功能可以提供丰富的图形体验，能够动态适应服务器负载、客户端访问设备负载和网络特征；
4. 对于所有媒体内容格式，RemoteFX 媒体流功能都能够让用户在 WAN 网络上获得流畅的媒体体验；
5. 通过 RemoteFX USB 重定向功能，即使远程桌面计算机未安装 RemoteFX vGPU，用户也能够使用带有RemoteApp应用程序和远程桌面的USB设备；
6. 特定场景下，RDP 8.0支持在远程桌面连接会话中运行其他远程桌面连接会话；
7. 通过性能计数器(RemoteFX 图形和 RemoteFX 网络计数器组)，管理员可以监视和解决用户体验问题。

### 2.4.2 ICA协议

ICA协议[[[16]](#endnote-17)]为桌面内容和外设数据在服务器和用户终端之间的传输提供了多种独立的虚拟通道，每个通道可以采用不同的交互时序、压缩算法、安全设置等。ICA虚拟通道是在服务器和用户终端之间建立双向连接，可用于传输声音、图像、打印数据、外设驱动等信息，其虚拟通道实现原理如图3所示：



图3 ICA虚拟通道原理

在ICA的基础上，Citrix在2009年发布了HDX(High Definition eXperience)技术对ICA协议进行了改进和增强，其目标是在桌面领域的多媒体、语音、视频和3D图形等内容为虚拟桌面提供更好的高清使用体验。HDX技术的核心内容如下：

1. HDX Plug-n-Play：本地虚拟桌面支持多种外设，并简化了设备的连接，包括USB设备、打印机、扫描仪、智能卡等外设；
2. HDX RichGraphics：充分利用服务器的处理能力，提供高分辨率图像的处理，优化图形密集型及富媒体应用的性能；
3. HDX MediaStream：将经过压缩处理的音频和视频发送到用户终端并在本地进行播放，提升多媒体的播放效果；
4. HDX RealTime：主要用于改善用户访问的实时性，支持双向音频，支持基于虚拟桌面的视频会议；
5. HDX Broadcast：针对不同的网络环境，利用压缩、缓存等技术提高访问远程桌面和应用的效果；
6. HDX WAN Optimization：优化广域网的访问性能和带宽消耗，提供自适应的加速能和QoS保证；
7. HDX Smart Access：支持用户在任何地点、任何设备上安全地访问虚拟桌面，支持SSO；
8. HDX Adaptive Orchestration：自适应调整HDX的配置，全面优化用户体验。

通过灵活地部署和应用各项HDX技术，能够全面、有效地优化虚拟桌面服务的交付效果，在各种网络条件下为用户提供更好的体验。

### 2.4.3 SPICE协议

Red Hat在KVM虚拟化的基础上采用SPICE作为桌面交付协议为用户提供VDI解决方案。SPICE是一个具有三层架构的解决方案：

1. QXL驱动：部署在服务器侧提供虚拟桌面服务的虚拟机中，用于接收操作系统和应用程序的图形命令，并将其转换为KVM的QXL图形设备命令；
2. SPICE客户端：部署在用户终端上的软件，负责显示虚拟桌面，同时接收终端外设的输入；
3. QXL设备：部署在KVM服务器虚拟化的Hypervisor中，用于处理各虚拟机发来的图形图像操作。

SPICE协议最大的特点是其架构中增加的位于Hypervisor中的QXL设备，本质上是KVM虚拟化平台中通过软件实现的PCI显示设备，利用循环队列等数据结构供虚拟化平台上的多个虚拟机共享实现了设备的虚拟化[[[17]](#endnote-18)]。但是这种架构使得SPICE协议紧密地依赖于服务器虚拟化软/硬件基础设施，SPICE必须与KVM虚拟化环境绑定。

SPICE协议能够自动判断和调整图像处理的位置，如果用户终端能够处理复杂的图像操作，就尽可能地传输图像处理命令而不是渲染后的图像内容，这样可以减少网络上传输的数据量。SPICE协议传输的内容主要包括两种命令流：一种是图形命令数据流，一种是代理命令数据流。图形命令数据流是从服务器端流向用户终端，将服务器端需要显示的图形图像信息传送到用户终端；代理命令数据流从用户终端流到服务器，主要传输虚拟机中部署的代理模块接收到的用户在终端进行的键盘、鼠标等的操作信息。SPICE协议支持多通道设置，利用不同的通道传输不同的内容。每个通道中的内容都可以通过相应的图形命令数据流或代理命令数据流进行传输。同时能够独立进行加密，支持不同的QoS。

### 2.4.4 PCoIP协议

PCoIP协议是Teradici在现有的标准IP网络的基础上研发的以显示压缩方式连接远程桌面的协议，支持高分辨率、全帧速的图像显示和媒体播放，同时还支持多屏幕显示设备、完整的USB外设和高质量的音频。VMware收购Teradici后在虚拟桌面产品VMware View中实现了利用服务器的通用处理器进行的基于软件的PCoIP协议处理。2012年1月，Teradici发布了PCoIP协议的专用板卡来降低服务器通用处理器的负载，实现性能加速，改进虚拟桌面的显示效果和应用体验。

提供虚拟桌面服务的虚拟机中，存在软件和硬件两种PCoIP协议的处理方式。硬件处理方式是在VMware服务器虚拟化平台对专用的PCoIP板卡进行虚拟化后由各个虚拟机共享，板卡的主要功能是处理图像编码。

VMware虚拟桌面产品的PCoIP协议主要特征有：

1. 服务器端渲染

PCoIP主要是在服务器进行图像的渲染，渲染完成后将加密后的像素通过网络传送到客户端，客户端只需要对像素进行解码并显示。服务器侧渲染降低了由客户端渲染导致的延迟问题。

1. 多种编码/解码方式

PCoIP协议对图像进行分析并进行元素分解，对图形、文本、图表、视频等内容进行区分，然后使用合适的编码/解码算法对相关像素进行压缩。智能图像分解和图像编码优化能够更有效地进行数据的传输，降低网络负载。

1. 动态适应网络状态

PCoIP协议能够根据网络状态自动调整图像质量。PCoIP传输的是像素，而不是传输数据文件，因此可以保证响应速度快、交互性强的用户体验。PCoIP协议的底层采用TCP协议和UDP协议，TCP协议主要是用于会话的建立和控制，而UDP则用于优化传输多媒体内容，能够降低对带宽的要求，优化交互体验。

### 2.4.5 RFB协议

RFB 协议全称为远程帧缓冲(Remote Frame Buffer)协议，是用于VNC(Virtual Network Computing)系统中的一个远程显示协议。RFB协议是一个简单的远程图形传输协议，不依赖于具体的图形接口，因此在所有的窗口系统(X11、Windows、Mac等)和应用程序中都可使用。RFB协议显示部分基于一条简单的显示原语“在给定的x，y位置上显示一个矩形像素数据”，通过一列矩形区域显示的更新来达到一次帧缓冲更新。RFB传输的是图像，是在服务器端将窗口在显存中画好之后将图像传给客户端，客户端只充当一个图像解码显示的角色。RFB协议中的编码方法主要包括Raw，Copy Rectangle、RRE、Hextile和ZRLE等方法[[[18]](#endnote-19)]。

采用Raw编码方式，屏幕上的像素点简单地从左到右扫描，然后发送到客户端。服务器端默认采用这种方式。Copy Rectangle编码使用客户端缓冲区中的数据，此时客户端已有了待传输的数据，只需将更新区域的坐标发送给客户端，然后客户端在缓冲区中相应位置拷贝即可。RRE(Rise-and-Run-Length)编码是将像素值相同的矩形区域作为一个整体传输，从而减少了传输的数据量。RRE编码首先传输的是一个背景色(屏幕上最常用的像素值)和一个矩形计数值N，然后是小矩形块，每个小矩形块的信息包括矩形的左上坐标、高度、宽度以及一个前景色值。在客户端显示时，先填充背景色，然后再在每个小矩形区域填充对应的前景色即可。RRE的解码效率较高，可以减轻客户端的处理负担。Hextile是RRE编码的变种，Hextile编码把整个屏幕分割成一个个16x16的小片，遵守从左到右，自上而下的顺序。每个小片采用的编码方式可以是Raw编码，也可以是RRE编码的变种。ZRLE(Zlib Run-Length Encoding)编码结合了zlib压缩、分片、调色板和Run-Length编码等技术，在传输时，首先传输的是—个4字节的长度值．紧接着是该长度的zlib压缩数据。每个RFB连接使用的是单一的zlib流对象，因此ZRLE编码的矩形必须严格地按照顺序进行编解码。

RFB协议的这些编码方式的特点是编码速度快，但压缩率很低，对于普通应用能够满足要求，但是对于视频等屏幕刷新频率很高的应用来说不能满足。

### 2.4.6 其他协议

Vebula-Fit协议在根据用户使用应用所需带宽的不同进行自适应，能更加有效地利用带宽。采用了分层渐进的方式在用户侧显示桌面图像，即首先传送给用户一个完整但是比较模糊的图像，进而在此基础上逐步精化。Vebula-Fit在视频传输过程中采用的是双向侦测平衡，从而自动选择将编/解码工作放在服务器或者用户终端进行，达到资源合理利用、平衡负载的目的。

Free VDI协议是京华科讯自主研发桌面虚拟化协议，协议通过网络多通道技术，将图像显示、键盘鼠标、音视频数据分别进行处理，提高了数据安全性，以独有的压缩算法节省了网络带宽的占用量。

## 2.5虚拟桌面协议改进

针对虚拟桌面显示协议的各种问题和不足，学术界及工业界都在不断地进行协议改进与优化。

混合桌面显示协议[[[19]](#endnote-20)]包括经典的虚拟桌面显示协议和一种实时桌面流协议。应用程序的图形输出经过GPU处理后通过基于H264/AVC的视频编码转换为视频流发送到用户终端，终端只需要解码视频流即可。

如图4所示，在图形库和设备驱动之间插入了一个驱动决策抽象层，用来接收图形库发来的绘图命令，然后决定转发给硬件显卡驱动还是虚拟的软件驱动。如果图形库发来的绘图命令需要GPU的支持，驱动决策层会检测命令的复杂度，然后做出决策。如果绘图命令转发到显卡驱动进行处理，GPU处理后的图像会被实时视频编码器编码成视频流发送到客户端。驱动决策层也会根据图像里的动作的数量来进行判断，同时，实时视频编码器也会向驱动决策层发送动作反馈。如果动作数量低于某个阈值，则由虚拟驱动进行处理。测试结果表明，实时视频流传输的数据量明显低于传统的虚拟桌面显示协议。视频质量在不限带宽的情况下能保证80%的质量，如果限制带宽3Mbit/s，则能保证70%的质量[19]。



图4 混合虚拟桌面协议处理流程

自适应协议[[[20]](#endnote-21)]针对虚拟桌面显示协议数据传输的上下游两个方向进行了协议优化，自适应终端、服务器、网络环境以及用户体验。下游方向，提出了基于调度的屏幕刷新模式，根据客户端需求和网络环境来调度屏幕刷新。上游方向，如果把用户产生的每个操作事件都单独封装起来发送到服务器端，则会产生大量的分组开销(Packetization Overhead)，因此提出在考虑到用户响应的情况下将用户的输入事件缓存起来，然后再一起打包发送，这样可以有效地降低分组开销。基于调度的屏幕刷新模式如图5所示。

客户端根据本地系统的运行状态(电量、处理速度、支持的解码方式、可用的带宽、延时等)，计算出调度值，以Scheduling Request消息的形式发送给服务器。除了最大更新频率，消息里还可以包含一些具体的刷新指标，比如刷新区域、编码模式等。服务端接收到该消息后需要重新计算调度值，然后服务端在检测到屏幕更新后，就基于新的调度值进行屏幕刷新。用户对文本编辑可以忍受的延迟是150ms但是对于游戏、视频类的应用能忍受的延迟则是80ms[[[21]](#endnote-22)]。通过动态适应客户端和服务端的处理能力以及网络环境，可以使用更少的带宽为用户提供可接受的QoE(Quality of Experience)。



图5 基于调度的屏幕刷新模式

RFB协议可以工作在任何可靠传输协议上，无论是字节流或基于消息的。一般来说，它使用TCP协议进行数据的传输[[[22]](#endnote-23)]。改进的RFB协议[[[23]](#endnote-24)]用UDP协议作为传输协议并在RFB头部增加了一些额外的控制字段。该协议主要适用于局域网环境系下，针对零客户端进行的改进。使用UDP进行数据传输，可以有效地简化硬件配置，可以降低数据冗余。但是UDP协议没有检错机制，在网络拥塞时数据包会丢失。针对这些问题，在RFB头部增加了一个序号字段，同时使用IEEE 802.1p定义的服务类别(CoS)来区别高优先级队列。为了简化RFB协议的解释过程，添加了一个额外的RFB命令字段。修改后的RFB头部如下所示：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| IPv4 Header | UDP Header | RFB Command (1 byte) | Sequence Number (1 byte) | RFB header |

Xiaofei Liao等人提出了轻量级虚拟桌面(Lightweight Virtual Desktop, LVD)，并提出了wRFB(window-based Remote Frame Buffer)协议[[[24]](#endnote-25)]，旨在高效地传输应用显示命令。wRFB协议支持同时运行不同操作系统中应用程序。在传统的VDI解决方案中，每个用户分配一个特定的桌面环境，可以是Windows、Linux或其他操作系统，如果用户需要运行其他平台上的应用，则必须要面对多个虚拟桌面并在多个桌面环境之间进行切换。wRFB协议可以从多个虚拟机中获取应用程序的用户界面，所有的图像融合后传输到同一个用户终端进行显示。

## 2.6本章小结

本章阐述了桌面虚拟化相关的国内外研究现状，总结了当前桌面虚拟化主要的发展方向，主要有桌面虚拟化基础架构VDI、虚拟操作系统基础架构VOI以及智能桌面虚拟化IDV。然后从企业的角度描述了当前的桌面虚拟化的发展现状，各虚拟化厂商推出的虚拟桌面产品以及产品特性。本章着重分析了当前主流的虚拟桌面显示协议，并总结了针对虚拟桌面显示协议所做的改进与优化。

# 第三章 SPICE虚拟桌面研究

SPICE协议是RedHat桌面虚拟化解决方案的核心部分，是专门针对远程显示传输而设计的协议，集成了很多传输协议优点，具有较高的传输性能。SPICE根据桌面传输内容通过不同的抽象通道进行数据传输，使得SPICE能够根据网络状况对传输的数据进行控制，并提供QoS控制。本章在总结了SPICE特性后，详细分析了SPICE虚拟桌面的框架结构及其核心技术。

## 3.1 SPICE协议特性

SPICE协议最早由Qumranet开发，是专门为桌面虚拟化解决方案设计的传输协议。Red Hat收购Qumranet之后，继续在KVM虚拟化的基础上采用SPICE作为桌面传输协议为用户提供VDI解决方案。SPICE是一个具有三层架构的协议，SPICE协议具有许多良好的特性：

1. 硬件设备的支持。传统的远程桌面协议运行在虚拟机中，因此不能直接访问服务器的硬件资源。SPICE协议本身工作在虚拟机服务器中，可以直接使用服务器的硬件资源，从而实现硬件加速；
2. 支持不同虚拟操作系统。SPICE协议通过虚拟设备接口与相应的虚拟设备进行交互，因此协议的实现与具体的操作系统无关，可以支持不同的虚拟操作系统；
3. 易于部署和管理。使用SPICE协议部署虚拟桌面时，只需要为虚拟机配置一个监听端口，不需要在虚拟机操作系统中进行任何配置；
4. 启发式决定任务处理端。SPICE协议根据用户终端的处理能力决定是否把CPU和GPU密集型任务交给终端进行处理。

SPICE协议定义了一组协议消息用于服务端与终端设备进行通信，这些消息不依赖于任何专用的传输层协议，SPICE协议不包含数据加密算法，可以灵活选用加密算法。SPICE的连接会话划分为多条虚拟通道，每一条通道对应一个远程设备，并且可以在运行时动态地添加、删除通道。如果需要添加对新设备的支持，只需要添加一条新的通道。SPICE协议消息以字节流的方式进行传输。SPICE协议通信过程包括连接建立和消息通信两个阶段，每条虚拟通道建立一条独立的socket。通道连接由用户终端发起，向服务端发送RedLinkMess消息，服务端返回RedLinkReply消息。如果通信正常，用户终端发送RedLinkReply中的公钥，服务端返回连接结果，通道连接建立完成。在建立通道后，各条虚拟通道通过发送消息与远程设备间进行通信。SPICE协议定义了所有通道通用的消息和各通道专用的消息。SPICE协议消息的具体格式如图6所示[[[25]](#endnote-26)]：



图6 SPICE协议消息格式

## 3.2 SPICE框架结构研究

基于SPICE协议的虚拟桌面解决方案主要包括客户端、服务端以及相关组件，如QXL设备、QXL驱动等，如图7所示。客户端运行在用户终端设备上，提供虚拟桌面环境。服务端运行在虚拟化服务器上，通过SPICE协议与客户端进行通信。



图7 SPICE框架结构图

SPICE服务端基于KVM/QEMU虚拟化环境，每个虚拟桌面环境是QEMU虚拟机中一个独立的进程。服务端通过虚拟设备接口与虚拟设备直接交互，不需经过虚拟操作系统。通过在Guest OS上安装QXL驱动可以提供更好的显示效果和增强的图像处理能力。

KVM虚拟机是基于Linux内核虚拟化，是Linux的一个模块，自Linux2.6.20之后集成在Linux的各个主要发行版本中。KVM使用Linux自身的调度器进行管理，核心源码很少。KVM虚拟化需要硬件的支持(如intel VT技术或者AMD V技术)，是基于硬件的完全虚拟化。但仅有KVM模块是不够的，因为用户无法直接控制内核模块，还必须有一个用户空间的工具。这个用户空间的工具，开发者选择了已经成型的开源虚拟化软件QEMU。QEMU可以虚拟不同的CPU，KVM借用了QEMU的一些组件，同时KVM的非内核部分是由QEMU实现的。

SPICE最大的特点是其架构中增加的位于QEMU中的QXL设备，本质上是KVM虚拟化平台中通过软件实现的PCI显示设备，利用循环队列等数据结构供虚拟化平台上的多个虚拟机共享实现了设备的虚拟化。但是这种架构使得SPICE紧密依赖于服务器虚拟化软/硬件基础设施，SPICE必须与KVM虚拟化环境绑定。在部署SPICE时，需要在KVM虚拟化环境QEMU中安装libspice库，这样KVM才能提供SPICE支持。

### 3.2.1 SPICE服务端结构

SPICE服务端通过虚拟设备接口VDI(Virtual Device Interface)接口与QEMU进行交互。VDI接口为应用程序提供了两个软件组件来发布虚拟设备接口，这两个软件组件分别称为后端(back-end)和前端(front-end)[[[26]](#endnote-27)]，软件组件可以通过VDI发布虚拟设备的操作接口，其他软件组件便可以通过VDI对虚拟设备进行访问和操作。

后端提供的接口可以被不同的前端使用而不需修改代码，同时后端也可以动态的切换前端，提高了程序的可用性和灵活性。后端接口在硬件仿真层，对外暴露了显示设备、键盘、鼠标、音频设备等的接口。前端接口通过后端接口获取音频、视频等输出信息，同时把鼠标和键盘输入时间推送到虚拟设备中。

前后端设备接口交互操作的初始化工作，是由后端设备接口完成的，它使用 VDI\_INIT信号将核心接口传送到前端，同时将前端接口初始化所需的参数也传递过来，前端根据后端传递的接口函数与虚拟设备进行通信。核心接口是一个包含数据成员和方法的结构体，提供连接其它接口的方法，使前端可以发现和访问后端提供的其它接口。

在SPICE虚拟桌面的整体框架中，SPICE服务端使用SPICE协议通过相应的虚拟通道与SPICE客户端进行通信，通过虚拟设备接口与QEMU虚拟设备进行交互。VDI的后端在QEMU中实现，QEMU提供了对不同虚拟设备的模拟，并提供统一的访问接口。SPICE服务端实现了VDI的前端接口，通过QEMU的后端接口访问虚拟设备，如图8所示。



图8 SPICE协议VDI通信模型

### 3.2.2 SPICE客户端结构

SPICE客户端的主要类图如图9所示，图中显示了SPICE客户端主要类以及类之间的继承关系。SPICE客户端使用C++实现，支持Windows和Linux平台。平台相关的数据结构及操作抽象成统一的类和接口，在不同的平台下有相应的实现；平台无关的部分则通过这些类和接口使用统一的代码实现。

Application类是客户端的核心类，包含RedClient、RedScreen、Monitor类的实例，用于处理应用程序的一般功能，如解析程序的命令行参数，执行主消息循环，响应和处理连接、断开、错误等事件，将鼠标输入重定向MouseHandler等。主消息循环中包含三个消息事件队列，主要处理两类消息，一种是与平台相关的消息，这部分消息的处理由EventSources实现；另一种为SPICE定义的事件，这些事件单独构成专门的处理类，派生自EventBase类、Event类，其中实现了接口函数response函数，此函数在消息循环中被调用，完成对不同事件的处理。

RedWindow类继承自RedDrawable和RedWindow\_p类，是平台相关的实现，主要用于实现客户端窗口的功能，如窗口的移动、隐藏、大小改变、标题等状态的设置和操作。RedDrawable 和RedWindow\_p也是平台相关的实现RedDrawable提供基础的像素渲染操作，RedWindow\_p 处理不同平台特殊的窗口数据及操作。

客户端通过RedScreen类提供远程虚拟桌面环境的显示，在显示的处理上，SPICE使用了分层的概念，它将虚拟桌面的显示根据不同显示逻辑分成不同的层，如光标显示层、信息层等。这种分层的方法具有较强的灵活性，不仅分离了不同的显示逻辑，使得各层在实现上互相独立，而且可以很方便的进行扩展。光标显示层位于其他显示层之上，单独绘制光标的形状和位置。ScreenLayer类封装了层的基本操作，如清除、更新、失效等，并附加到RedScreen上，RedScreen类封装屏幕操作的相关逻辑，通过ScreenLayer显示虚拟桌面的内容。



图9 SPICE客户端主要类图结构

### 3.2.3 SPICE虚拟通道

SPICE协议定义了用于访问、控制、接收远程设备输入以及返回输出结果的消息，不依赖于任何RPC标准或者具体的传输层，协议传输的数据加密独立于协议本身。SPICE协议用于客户端和服务端之间、客户端和Agent之间的通信。具体的通信操作通过不同的虚拟通道channel，每一种channel专门用来负责一种类型数据的传输与通信。服务端和客户端有对应的channel，每个channel使用专门的TCP连接，并有专门的线程进行处理，可以设置不同的优先级来达到不同的QoS。

SPICE协议定义了以下通道：

1. Main Channel：主连接通道，控制所有其他通道的创建、连接、销毁等，并处理控制、配置以及迁移消息等；
2. Display Channel：显示通道，处理图形图像渲染命令、视频流；
3. Inputs Channel：处理鼠标、键盘的输入消息；
4. Cursor Channel：控制鼠标设备的位置、形状及可见性；
5. Playback Channel：从服务端接收音频，在客户端进行播放；
6. Record Channel:在客户端捕获音频并发送到服务端。

每条通道都会注册自己的ChannelFactory，然后主连接通道通过ChannelFactory创建相应的通道。

### 3.2.4 SPICE代理模块

SPICE代理模块Agent应用程序运行在Guest OS上，主要负责处理与操作系统和平台相关的任务，比如Guest OS的桌面显示配置、屏幕分辨率设置、剪贴板支持等功能。Agent模块完成的功能与具体的操作系统相关，不同系统上有不同的实现。Windows中的Agent由两部分组成，包括系统服务和应用程序。系统服务从QEMU虚拟设备端口VDI Port中接收操作指令，并转发给Agent应用程序，应用程序使用Windows API完成这些指令的操作。客户端和服务端与Agent通过VDI Port设备与驱动进行交互，Agent消息处理流程如图10所示：



图10 Agent消息处理流程

Agent消息可以由客户端生成，也可以由服务端生成，或者由Agent程序生成。VDI Port驱动读写设备的输入输出命令。由客户端或服务端生成的Agent消息写入到VDI Port设备的输出命令环中。Agent生成的消息由驱动写入到设备的输入环中，这些消息会被服务端读取并保存在缓存中等待处理。

## 3.3 SPICE核心图像技术研究

### 3.3.1 图像处理流程

SPICE框架中对图像数据的处理主要包括Guest OS中的显示驱动程序、QEMU中的虚拟显示设备、服务端数据处理模块和客户端的图像渲染部分。虚拟显示设备支持标准VGA显示设备和QXL显示设备。QXL显示设备可以提高远程显示效果并增强Guest端的图形显示能力。SPICE图像处理的整体流程如图11所示[[[27]](#endnote-28)]。应用SPICE协议时，需要在KVM虚拟化环境的QEMU中安装libspice库，这样KVM才能成为SPICE服务器。提供虚拟桌面服务的虚拟机(即图中的Quest OS)上的应用程序向操作系统的图形引擎(GDI/X Engine)发出图形处理操作，图形引擎把绘图命令发送给QXL驱动，QXL驱动将操作系统的绘图命令转换为QXL命令后推送到QXL设备的图形命令循环队列中，libspice库从中获取绘图命令，添加到图形命令树上。图形命令树主要负责对QXL命令进行组织和优化，同时负责对视频流进行侦测。经过图形命令树优化的QXL命令放入到发送队列，由libspice库维护，并发送到SPICE客户端更新显示内容。



图11 SPICE图像处理流程

SPICE协议传输的是绘画指令，而不是绘画内容，以此来减少传输内容。SPICE对图像显示数据的处理主要是在服务端进行，处理过程主要涉及Guest OS中的QXL驱动程序、QEMU中的QXL虚拟设备、服务端的显示命令处理和客户端的图像数据渲染。Guest OS中安装相应的QXL显示驱动，SPICE利用QXL驱动和设备直接交互来增强Guest OS的图型显示能力，并通过将显示操作转换为QXL指令来提高远程传输显示数据的效率。服务端的图像处理使用专门的线程Red Worker处理，以达到高效的执行效率。服务端为每一个QXL设备接口创建一个分派器Red Dispatcher，分派器负责创建专门的工作线程Red Worker对图像命令进行处理。Red Dispatcher和Red Worker之间通过socket传输数据。Red Worker通过QXL设备接口与QXL设备交互，Red Dispatcher对Red Worker内部实现进行了封装，可以独立于QXL设备和服务端。Red Dispatcher实现了QXL Worker接口，用于将设备命令转换成Red Worker识别的消息格式。服务端Red Server的控制消息通过Red Dispatcher接口调用，由Red Dispatcher将控制消息转发给对应的Red Worker。SPICE图像处理子系统[[[28]](#endnote-29)]如图12所示：



图12 SPICE图像处理子系统

Red Worker是SPICE 服务端中进行图形图像处理的核心模块，主要负责的功能有以下几个：

1. 处理QXL设备命令，比如绘制、更新、光标等；
2. 处理从Red Dispatcher接收到的消息；
3. 负责显示通道、光标通道；
4. 对图像数据进行压缩；
5. 视频流的识别、编码、创建；
6. 管理缓存；
7. 优化图形命令，通过图形命令树移除被遮盖的部分；
8. Cairo和OpenGL的图像渲染；
9. 维护图像命令环。

显示数据包括图像视频数据，由Red Worker处理后发送到客户端，其中显示数据来源主要有三类：

1. Red Worker通过QXL Interface获取的QXL异步显示命令，主要是绘图命令；
2. QXL设备通过Red Dispatcher发送的同步QXL命令；
3. Red Server通过Red Dispatcher发送的控制命令，如创建显示通道和光标通道。

Guest OS内的虚拟桌面更新时，QXL驱动把绘图命令转换成QXL命令发送到QXL虚拟设备。QEMU监听到QXL设备更新后，调用Red Dispatcher向Red Worker发送消息。Red Worker读取QXL命令，转换成SPICE协议消息，通过相应的通道发送到客户端，由客户端做图像的渲染显示。SPICE对图像显示数据处理的基本流程如图13所示：



图13 SPICE服务端图像处理

SPICE对显示数据的处理主要分为4个阶段，分别是：

1. 初始化

QEMU初始化SPICE服务端，首先向QXL设备注册监听器，设置更新处理回调函数，监听QXL更新操作，然后向SPICE服务端注册QXL设备接口，同时添加相应的处理函数。

在注册VDI接口时，服务端为QXL设备创建一个Red Dispatcher，然后Red Dispatcher会创建一个Red Worker线程，该线程用于处理相应的控制信息和图形命令，并发送处理后的数据到客户端进行显示。

1. 客户端建立连接

由客户端发起连接请求，服务端响应连接请求，由Red Server调用Red Dispatcher接口，向Red Worker发送消息创建显示通道，用于处理控制消息和绘图命令。同时客户端也会创建显示通道，对接收的绘图命令进行处理。

1. 桌面显示区域更新

Guest OS中显示区域更新时，QXL驱动将系统的绘图指令转换成QXL命令写入到QXL虚拟设备中。QEMU监听到QXL设备命令更新后，通过Red Dispatcher向Red Worker发送消息，通知Red Worker对显示命令进行处理。

Red Worker收到Red Dispatcher发送的消息后，调用QXL设备接口，从QXL设备的命令环中读取绘图指令，通过图形命令树对QXL命令进行优化处理，将优化后的命令转换为SPICE协议消息，然后通过显示通道发送到客户端。客户端收到图像数据后，会根据当前客户端的处理能力采取不同的处理方式。

1. 客户端断开连接

服务端收到客户端断开连接的请求后，将Red Worker设置为休眠状态，同时关闭与客户端建立的连接通道。SPICE处理显示数据的顺序图如图14所示：

### 3.3.2 图形命令树

图形命令树包含了当前显示内容的所有图形命令。SPICE服务端使用图形命令树删除被覆盖的图形命令，并用来检测视频流。SPICE服务端从QXL虚拟设备中读取QXL图形命令，添加到命令树中。添加时，服务端根据当前命令与命令树中的其他命令所对应的显示区域进行比较，如果某些命令对应的显示区域被新的命令完全覆盖，并且没有其他命令依赖于这些命令，这些命令将会从命令树中删除，这样就减少了传输的命令数量。

图形处理命令树处理过程如图15所示。当命令环中出现的新的命令覆盖了之前命令的显示区域，图中命令E完全覆盖了命令B、C、D，此时命令B、C、D将被从命令树上删除。同时SPICE服务端维护命令发送队列，队列中的命令与命令树中保存的命令保持同步，命令树中的命令被删除时，也会删除发送队列中对应的命令，同时释放相关资源。

### 3.3.3 智能压缩技术

SPICE提供了3种图像压缩算法，这些算法可以在服务端启动时选择，也可以在运行过程中动态切换。SPICE根据不同的图像类型，启发式地选择不同的压缩算法，以达到最佳的压缩效果，降低传输图像数据所需的带宽需求。



图14 SPICE处理显示数据顺序图



图15 SPICE图形命令优化示意图

SPICE提供的图像压缩算法有Quic，LZ和GLZ。Quic是SPICE的专有算法，基于SFALIC算法[[[29]](#endnote-30)]实现，是一种基于编码预测的快速压缩算法，根据预测方程对图像数据的预测误差计算进行编码。LZ(LZSS)[[[30]](#endnote-31)]算法是另一个可选的压缩算法，LZ算法是对LZ77算法[[[31]](#endnote-32)]的改进，LZ77是一种基于字典模型的压缩算法，该算法描述了一种基于滑动窗口的缓存技术，该缓存用于保存最近刚刚处理的数据。LZ采用了一个大小为N的滑动窗口用于在文件中滑动。Quic和LZ算法都是局部压缩算法，独立地压缩每一张图片。Global LZ (GLZ)是SPICE另一个专有算法，在LZ算法的基础上采用一个基于历史机制的全局字典，使用一个宽窗体来合并图像数据流，优化算法对图像之中或图像之间的重复部分。这三种压缩算法都是无损压缩算法，可以避免一些重要的显示数据被破坏，SPICE服务端会根据图像数据的特性，启发式的在GLZ/LZ和Quic之间选择合适的压缩算法。理论上来说，人工合成的图像，使用GLZ/LZ算法会有更好的效果；对于真实的图像，比如照片，使用Quic算法压缩效果更好。对于视频流来说，每个视频帧都是一个独立的图像，如果使用无损压缩算法，将消耗大量的带宽资源，而且，视频内容对于显示来说不是非常关键的，因此SPICE使用有损压缩算法来处理视频流。SPICE服务端启发式地识别更新频率很高的区域，使用MJPEG算法将这些区域编码成视频流发送到客户端，这样可以节省大量的带宽资源，尤其是在广域网环境下，可以有效地提升SPICE虚拟桌面的多媒体性能。

### 3.3.4 自适应图形处理

图像的渲染和处理是虚拟桌面实现的重点。SPICE在设计中，提供了软件处理和硬件处理的支持。SPICE客户端采用了基于Cairo图形库的软件处理方式，通过CPU计算资源，提供2D图形数据的渲染能力。同时，SPICE客户端也支持基于GPU的硬件处理方式，在Windows平台使用GDI接口，在Linux平台使用OpenGL库。使用GPU进行图形处理，可以提供高性能的图形渲染处理能力，同时降低CPU利用率，使CPU可以处理更多的其他应用。

SPICE可以自适应的处理图形数据。如果客户端的处理能力较低，对图形的处理会尽可能的在服务端完成；如果客户端处理能力足够强，服务端直接将图形命令发送给客户端，由客户端进行渲染处理，这样可以降低服务端的运行负载，同时也减少网络传输的数据量。

## 3.4 本章小结

SPICE协议及其整体框架是基于KVM虚拟化环境的远程虚拟桌面解决方案。本章从协议特性、消息格式对SPICE协议进行详细介绍，分析了SPICE虚拟桌面框架的各个部分的具体实现，重点研究了SPICE协议在图像视频等多媒体方面的实现原理和方法。

# 第四章 SPICEx视频优化研究与实现

## 4.1 相关技术基础

### 4.1.1 MJPEG2000算法

JPEG2000[[[32]](#endnote-33)]是由Joint Photographic Experts Group(联合图像专家小组，JPEG)制订的新一代图像压缩标准。JPEG2000是针对静止图像提出的压缩标准，而运动JPEG2000标准则是在该标准的第三部分定义的，称为Motion JPEG2000(MJPEG2000)，是一种针对视频图像(即运动图像)的标准，将JPEG2000图像编码技术应用于视频[[[33]](#endnote-34)]。MJPEG2000继承了JPEG2000图像编码的特点。与现有的MPEG、H26X系列视频压缩标准不同，MJPEG2000只进行帧内编码，没有运动估值和帧间预测编码，生成的文件中包含多个JPEG2000压缩图像序列，与声音、元数据同步后存储为MJ2格式的文件。MJPEG2000的设计与现有JPEG2000、MPEG4兼容, 最大限度地实现了视频编码标准的互操作性[[[34]](#endnote-35)]。JPEG2000对于从高比特率到低比特率的所有比特率都具有超级压缩特性。与JPEG相比，JPEG2000有明显优越的压缩效率。JPEG在低比特率编码时会产生DCT固有的块状编码失真，而JPEG2000通过采用小波变换编码，失真以图像的迟钝形式显露出来，在视觉效果上相差很大。表2显示了集中编码方式的性能和功能的比较。

表2 几种编码方式的性能和功能比较

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 评价项目 | JPEG2000 | JPEG-LS | JPEG | MPEG-4 VTC |
| 可逆压缩性能 | +++ | ++++ | + |  |
| 不可逆压缩性能 | +++++ | + | +++ | ++++ |
| 渐进比特流 | ++++ |  | + | ++ |
| ROI编码 | +++ |  |  | + |
| 随机存取 | ++ |  |  |  |
| 低复杂性 | ++ | +++++ | +++++ | + |
| 容错性 | +++ | + | + | +++ |
| 非递归率控制 | +++ |  |  | + |
| 通用性 | +++ | +++ | ++ | ++ |

JPEG2000的核心算法是离散小波变换，通过JPEG2000进行图像数据的编码与解码的过程[[[35]](#endnote-36)]如图16所示：



图16 JPEG2000图像编码解码过程

小波变换的计算量大、计算过程复杂。Sweldens提出了小波提升算法(lifting scheme)，有效地解决传统的基于Mallat的塔式分解小波变换算法计算量大、对存储空间的要求高的问题，从算法方面提高了小波变换的实现效率。小波提升不依赖于傅里叶变换，其复杂度只有原来卷积方法的一半左右，因此成为计算离散小波变换的主流方法。第一代小波变换非常依赖傅里叶变换，小波提升就是为了构造第二代小波变换，同时已经证明了可以实现所有的第一代小波变换。小波提升有如下特点：

1. 继承了第一代小波的多分辨率的特性
2. 不依赖傅里叶变换
3. 原位(in-place)计算，不需要申请额外内存
4. 反变换很容易从正变换得到，只是改变了数据流的方向和正负号

正因为小波提升计算速度快、占用内存少、可以实现整数变换等特点，所以被JPEG2000推荐，是JPEG2000里面的核心算法。小波提升的核心就是预测算法和更新算法，通过预测和更新两个提升环节实现信号的高低频分离。由于信号有局部相关性，某一点的信号值可以通过其相邻的信号的值，通过适当的预测算子预测出来，同时预测出来的误差就是高频的信息，从而这个过程就是预测环节。预测环节下面得到的高频信息又通过更新算子来调整信号的下抽样来得到低频信息，这个过程就是更新环节，在整个的提升算法中，更新环节叫做primary lifting，而预测环节叫做dual lifting。通过预测算法可以得到高频信息，而通过更新算法可以得到正确的低频信息。提升样式可以实现原位计算和整数提升，并且变换的中间结果是交织排列的，其中原位计算和整数提升在硬件实现中很有价值。

JPEG2000已被证明能比之前的标准JPEG提供更高效的编码效率。JPEG2000编码具体过程[[[36]](#endnote-37)]如图17所示：



图17 JPEG2000编码步骤

JPEG2000编码步骤描述如下：

1. DC层进与分量变换。表示一个图像必需的信号成分成为分量。分量变换被定义为可逆变换和不可逆变换。可逆变换通过由整数值变换为整数值，不会产生运算误差；不可逆变换包含了基于实数运算的运算误差，是依赖于YCbCr变换的一种方式。
2. 图像分量填充块分割。分量变换后的各分量图像可以分割成填充块Tile。填充块是不具备重叠部分的任意大小空间域，填充块分割是在各分量之间同一相对位置进行的。填充分量相互独立编码构成了编码处理的基本单位。
3. 小波变换。小波变换滤波器被定义为实型9/7不可逆滤波器(9/7 irreversible filter)和整型5/3(5/3 reversible)两种。各填充分量通过小波变换被划分成多个分解层(decomposition level)。各分解层由子带构成，各子带系数表示了填充分量内适合采用小波变换的局部区域中水平和垂直方向的频率特性[[[37]](#endnote-38)]，如图18所示。子带的分裂是对低频进行均量折半递减，因此，相邻分解层之间的空间分辨率以2的幂系数增减。
4. 量化。对于一次子带分裂可生成低通和高通两个子带，形成编码对象的变换系数。由于在一次子带分裂后被抽取成1/2，所以变换系数的总和与原来采样数目相同。减少该子带系数精度的处理就是量化，量化是具有相同步长的标量量化，进行量化时的量化步长在子带内是固定的，同时可以对每个子带进行改变。在采用整型小波滤波器进行可逆编码时，可以不进行量化。
5. 系数位建模。各填充分量的量化子带系数被进一步分解为代码块(code block)，代码块内的子带系数其大小由MSB向LSB做位平面展开后，有三种编码扫描方式，分别是有效传播(significance propagation)，量级细化(magnitude refinement)，精化(cleanup)。对于各个编码扫描，还要考虑周边采样状态，将位平面上的0/1值分离成统计性质相似的多个群组。这个过程成为系数位建模(coefficient bit modeling)。



图18 JPEG2000小波变换分层示意

1. 算术编码。运用系数位建模得到的二进制判决代码串和上下文信息，再用熵编码对量化子带系数进行无损压缩，生成比特流。此熵编码采用了MQ算术编码，在每段上下文按二进制码串的统计性质适应算术编码的内部状态变数，可以获取很高的压缩效率。
2. 压缩代码的分层和打包。由各代码块内编码扫描生成的算术符号比特流，在考虑了图像质量的前提下，构成嵌入码，以层为单位进行成组处理。一个填充分量内的压缩图像数据处在一个分辨率等级，一个范围内来自一个分层的压图像压缩数据被归入一个包内，以包为单位构成压缩码流的基本单位。

MJPEG2000的基础是JPEG2000，其压缩过程需要对每一帧图像进行JPEG2000方式编码。MJPEG2000压缩算法有如下特性[[[38]](#endnote-39)]：

1. 帧内压缩，MJPEG2000采用帧内压缩编码，减少帧间数据的依赖性，提高动态响应和容错能力。由于码流内各帧的数据独立编码,每帧都是一幅单独JPEG2000码流，这样不但便于视频帧的插入、移动或删除，还可以修改一帧的数据。
2. 可缩放性，JPEG2000引入的EBCOT算法使得编码图像可以多次降低比特率并且在任意比特率下都保持最优的率失真性能。
3. 单一码流内同时提供有损压缩和无损压缩功能
4. MJPEG2000提供了注释功能,可以把注释内容集成在码流里

MJPEG2000对图像序列的每一帧都采用JPEG2000压缩算法进行编码，因此视频帧的压缩过程可以看成将多次单帧图像的JPEG2000压缩结果合成统一数据流，具体的流程[[[39]](#endnote-40)]如图19所示:



图19 Motion JPEG2000视频压缩流程

JPEG2000编码器通过选择各子带系数的量化步长或编码流的截断来实现码率控制，MJPEG2000可以直接利用JPEG2000的码率控制机制。调整各子带的量化步长可以控制压缩图像的码率，量化步长增大，则量化子带系数的动态范围减小，因而编码比特流的码长即码率相应减小。编码器可以计算出每个编码通道对码率的贡献，以及图像失真减小值。MJPEG2000视频编码与H.264视频编码性能比较[[[40]](#endnote-41)]如下：

* 1. 对于小尺寸图像，H.264帧内压缩效果较好，随着图像尺寸的增大，M JPEG2000表现出明显优越的压缩效果。对于大尺寸的图像，MJPEG2000压缩得到图像质量明显优于采用H.264帧内压缩的图像。
  2. 当压缩率很高时，H.264压缩效果要比MJPEG2000压缩效果略好。主要是由于H.264帧内压缩采用了性能优秀的帧内预测模式，可以极大减少存储图像所需数据量。MJPEG2000所使用的小波变换和优化截取的嵌入式块编码同样性能优秀，码率较高时MJPEG2000性能更好。
  3. 码率非常低时(小于0.5kbps)，MJPEG2000编码图像看起来非常模糊，缺乏高频信息，而H.264的帧内编码图像没有出现这种情况。

由上述比较可以看出MJPEG2000适合应用于高品质视频编码，这也是本文选择MJPEG2000进行视频压缩的原因。

### 4.1.2 流媒体与流式传输协议

流媒体技术，是指在网络上按时间先后次序传输和播放连续音视频数据流的技术。流媒体技术使得用户在播放前不需要下载整个文件，只需将一部分内容缓存，从而实现媒体数据流边传送边播放的功能。用户无需等到整个文件下载完毕，只需经过很短时间地点延时即可在用户的计算机上对压缩的多媒体文件解压后进行播放。流式传输方式将整个多媒体文件经过特殊的压缩方式分成一个个压缩包，由视频服务器向客户端连续、实时地传送。在采用流式传输方式的系统中，多媒体文件的剩余部分将在后台的服务器内同时进行下载。与单纯的下载方式相比，这种在下载的同时进行播放的流式传输方式不仅使启动延时大幅度地缩短，而且大大降低了系统缓存容量的需求。流式文件播放结束后，数据不会保存在硬盘上，这样也节省了磁盘空间。根据媒体形式的不同，流媒体有流式的音频、视频、动画、图像、文本等类型，其播放方式有单播、组播及点播与广播。

流媒体技术中采用实时流控制协议(Real Time Streaming Protocol，RTSP)[[[41]](#endnote-42)]控制数据的实时发送。RTSP协议主要是用于建立和控制单个或多个连续媒体的时间同步流。RTSP协议本身并不传输数据，主要用来对多媒体服务器实现网络远程控制。视频流数据传输用的协议是实时传输协议RTP(Realtime Transport Protocol，RTP)和实时传输控制协议(Realtime Transport Control Protocol, RTCP)[[[42]](#endnote-43)]。RTP是用于Internet上针对多媒体数据流的一种传输协议，其目的是提供时间信息和实现流同步。为了保证实时性，RTP通常采用UDP来传送数据。RTP报文中有时间戳项，用于接收方对乱序报文进行重组，并实现同步。RTCP和RTP一起提供流量和拥塞控制服务。在RTP会话期间，每个参与者周期性的传送RTCP数据包，其中包含已发送数据包的数量和丢失的数据包的数量等统计资料，因此服务器端可以利用这些信息动态的改变传输速率，以实现网络吞吐率和拥塞控制的折中。在基本的交互过程中，主要使用以下几种消息进行通信。

1. Option：客户端发送请求，告知服务端支持的方法。
2. Describe：客户端发送请求，服务端对其权限进行验证，并响应关于媒体的请求消息。
3. Setup：客户端发送建立连接请求。
4. Play：客户端选择播放实时视频时段。
5. Teardown：客户端发送关闭请求，服务端响应后释放播放连接。

流媒体服务器与客户端进行交互的示意过程如图20所示。在多点传送或单点传送的网络服务上，RTP提供端对端的网络传输功能，适合应用程序传输实时数据。RTP没有为实时服务提供资源预留的功能，也不能保证QoS。数据传输功能由RTCP来扩展 ，通过扩展对数据传输进行监测控制。



图20 RTSP服务器与客户端交互过程

流媒体视频传输的关键技术主要有：

1. 视音频的压缩/解压缩算法

从流式传输的角度来说，压缩/解压算法主要希望算法具有可伸缩性，即经过一次压缩编码后，数据在进行传输过程中能根据网络带宽的变化进行一定的调整，使客户端在获得较好图象质量的情况下保持连续性。在压缩的同时还要将文件分成压缩包，形成数据流，将原来的多媒体文件转化成具有流格式的流媒体。

1. 网络的服务质量QoS

流媒体数据在客户端的流畅播放，需要保证一定的网络时延和带宽，因此需要引入服务质量QoS控制。QoS控制机制可以分为基于网络和基于终端系统两种。基于网络的方法是由网络中的路由器、交换机等提供QoS支持，如路由器发生阻塞时不再是随机丢包，而是根据服务的优先级或包中信息的重要程度有选择地丢包等。基于终端系统的方法是由服务器和客户端采取QoS控制措施来提高视频质量，不需要网络参与。

1. 流服务器

流服务器根据客户端的反馈信息和网络拥塞情况自动调整发送媒体流的速率。为了提高流服务器的性能。可以从硬件上采用高带宽、高性能的服务器，同时可以在软件上采用合适的传输策略，如Batching和Patching处理技术等。

1. 网络优化与控制技术

有效的网络优化和控制技术可以在网络带宽有限的条件下提高流媒体的播放质量。通过采用智能流技术把不同速率的流编码合并到同一个文件。在传输的同时探测网络带宽的变化情况，智能流通过描述Internet上变化的带宽来发送高质量媒体并保证可靠性。

在实际应用中，为了解决单一媒体服务器易于导致超负荷的问题，通常可以使用分流技术在不同地点的媒体服务器之间传输数据，使得客户端可以就近访问媒体服务器，从而获得更好的访问质量，减少带宽使用。

### 4.1.3 GStreamer多媒体框架

GStreamer[[[43]](#endnote-44)]是一个创建媒体应用程序的框架，其主要用途是构建媒体播放器。GStreamer是基于插件的，具有任何符合规范的插件都可以方便的插入已定义的数据流管道的优点，因此，可以简化媒体播放器的开发流程，提高可移植性。GStreamer库中主要有以下元素：

* 1. 元件和插件：元件是GStreamer的核心。在插件的开发中，一个元件就是继承于GstElement的一个对象。按照功能可细分成三类：数据源元件(src)，只有输出端，它仅能用来产生供管道消费的数据，而不能对数据做任何处理；接收器元件(Sink)，只有输入端，它仅具有消费数据的能力，是整条媒体管道的终端；过滤器元件(Filter)，既有输入端又有输出端，它从输入端获得相应的数据，并在经过特殊处理之后传递给输出端。为了使GStreamer能够使用元件，必须将元件封装到一个插件中。一个插件是一块可以加载的代码，通常被称为共享对象文件或动态链接库，一个插件中可以包含一个或若干元件。GStreamer核心库函数是一个处理插件、数据流和媒体操作的框架，核心库中只有少量基本函数，其他所有的功能都由插件来实现。所有注册的插件的详细信息用一个XML文件来保存，这样可以查询并只在需要元件时才进行加载。
  2. 衬垫(Pads)：衬垫是元件与外界的连接通道，用来在元件间协商连接和数据流，可以看作元件间互相连接的“接口”，数据流通过这些接口流入流出元件。它具有特殊的数据处理能力，可以限制通过它的数据类型，当两个衬垫允许通过的数据类型兼容时才可以将它们连接起来。
  3. 数据、缓冲区和事件：GStreamer中的所有数据流被分割成块，从一个元件的源衬垫传到另一个元件的接收衬垫。当前存在两种数据类型：事件和缓冲区。事件包含两个相连的衬垫间的流的状态信息，只有事件被元件显式地支持时它们才会被发送，否则核心层将自动处理。缓冲区可以包含两个相连接的衬垫所能处理的任何数据。通常，一个缓冲区包含一块音频或视频数据块，该数据块从一个元件流向另一个元件。

RTP插件基本原理如图21所示，图中简要描述了双工工作时单向视频通信过程。其中发送视频数据流时，关键部件为元件Pay，负责将采样、编码以后的音频数据封装为GstBuffer 包格式交给系统处理后通过网络传输。接收视频数据流时，关键部件为元件Depay，完成系统通过网络接收后的GstBuffer数据包，解封装为音频数据，解码后播出播放。



图21 GStreamer单向视频通信过程

## 4.2 视频压缩算法改进

一个图像能包含多于一个的图像分量(最多214个分量)，每个分量都是相同大小的矩阵。每个分量的样本值都是整数，这些样本可以是有符号或无符号的1到38位的整数，样本的符号性质和位数都在分量数据前给出。所有分量都能有个统一的空间扩展信息，但这些分量代表着不同的颜色空间或辅助信息。比如，一个RGB彩色图像有三个图像分量，各表示红、绿、蓝颜色。在最简单的灰度图像中，只有亮度分量。数字图像组成结构如图22所示。

图像数据的分量变换在输入图像由RGB三种分量组成的情况下，通过线性变换方法变换成亮度和色差的YCbCr三种成分，通过这种变化可减少RGB信号之间的相关性，能比亮度成分更大地压缩色差成分，提高压特性。



图22 数字图像组成结构

SPICE的图像视频压缩机制如图23所示。本文实现的SPICEx采用MJPEG2000压缩算法对虚拟桌面的视频数据进行压缩，修改后的压缩机制如图24所示。



图23 SPICE的图像视频压缩



图24 SPICEx的图像视频压缩

通过无损压缩的可逆分量变换RCT对图像数据进行处理，可逆分量变换只能与5/3可逆小波变换组合使用。可逆分量变换的变换算法如下：

在解码器中，进行离散小波逆变换后，采用下列算法进行可逆分量逆变换(逆RCT)：

小波变换可以通过基于提升的滤波运算高效执行。基于提升的滤波由非常简单的滤波系数组成，奇数索引的系数值用偶数索引的系数值加权和来更新，偶数索引的系数值则用奇数索引的系数值加权和来更新，这样交替重复。基于提升的方法非常易于实现可逆变换，能比卷积运算方法以更少的运算次数实现小波变换。以一维信号X进行分解为例，对X进行第一级分解，低频信息在奇数上面，高频在偶数上面，进行第二级分解，对第一级的低频信息进行分解，分解的结果则是在奇数数据中的奇数位数为低频信息，偶数位数为高频信息。

小波提升算法的基本思想是通过由基本小波(lazy wavelet)逐步构建出一个具有更加良好性质的新小波，其实现步骤有3个：分解(split)、预测(predict)和更新(update)。分解是将数据分为偶数序列和奇数序列2个部分，预测是用分解的偶数序列预测奇数序列，得到的预测误差为变换的高频分量，更新是由预测误差来更新偶数序列，得到变换的低频分量。提升算法是原位计算，即进行小波变换时在原位计算各系数，计算的系数可以直接替代原始数据而不需要附加的存储空间。在JPEG2000中，5/3提升小波变换算法为：

(4.3)

逆变换的算法为：

(4.4)

在SPICEx协议中通过定义Jp2Encoder结构实现JPEG2000的编码，Jp2Encoder结构定义如下：

|  |
| --- |
| struct Jp2Encoder {  opj\_cio\_t \*cio; /\* compressed image saved in cio->buffer \*/  opj\_cinfo\_t \*cinfo;  opj\_bool b\_success;  opj\_cparameters\_t parameters;  opj\_event\_mgr\_t event\_mgr;  opj\_image\_cmptparm\_t cmptparm[3];  opj\_image\_t \*image;  OPJ\_COLOR\_SPACE color\_space;  int first\_frame;  int quality;  unsigned int bytes\_per\_pixel; /\* bytes per pixel of the input buffer \*/  void (\*pixel\_converter)(uint8\_t \*src, uint8\_t \*dest);  }; |

在SPICEx协议中，通过jp2\_encode函数进行JPEG2000图像压缩。在具体实现过程中，首先需要获得图像数据，将图像的RGB数据分别保存在opj\_image\_t相应的分量中，然后对opj\_image\_t进行JPEG2000编码。jp2\_encode函数实现的伪代码如下：

SPICEx 服务端检测到视频播放后，通过red\_marshall\_stream\_data函数对视频数据进行编码，编码的过程中每一帧图像都通过jp2\_encode函数进行JPEG2000压缩，其处理的部分代码如下：

|  |
| --- |
| static inline int red\_marshall\_stream\_data(RedChannelClient \*rcc,  SpiceMarshaller \*base\_marshaller, Drawable \*drawable) {  ……  **jp2\_encode**(&image->u.bitmap, width, height, dcc->send\_data.stream\_outbuf, &len, 50+ratio);  ……  agent->last\_send\_time = time\_now;  return TRUE;  } |

## 4.3 视频区域检测增强

视频区域的检测是一个关键技术。SPICE协议本身根据虚拟桌面区域更新的频率来判断是否在播放视频。判断是否有视频播放在red\_is\_stream\_start函数中完成，其定义如下：

|  |
| --- |
| #define RED\_STREAM\_FRAMES\_START\_CONDITION 20  #define RED\_STREAM\_GRADUAL\_FRAMES\_START\_CONDITION 0.2  static inline int red\_is\_stream\_start(Drawable \*drawable) {  return ((drawable->frames\_count >= RED\_STREAM\_FRAMES\_START\_CONDITION) &&(drawable->gradual\_frames\_count >= (RED\_STREAM\_GRADUAL\_FRAMES\_START\_CONDITION \* drawable->frames\_count)));  } |

只有当某个区域的更新频率大于特定值时才判断是视频播放。但是这种检测方法具有很大的局限性，并不是完全准确，而且在判断错误时会导致不同压缩算的频繁切换，引起桌面图像的抖动。

为了提高视频播放检测的准确性，采用了窗口监听机制，即在虚拟机内部运行一个后台程序检测当前桌面上运行的所有窗口，一旦发现有播放器在运行，即判断有视频在播放，同时将视频播放的控制功能由虚拟机转移到客户端，客户端直接控制视屏的播放进度。在检测到视频播放后，启动一个单独的线程PlayFunc，其中运行流媒体进程，为客户端请求流媒体做好准备。该窗口监听模块默认监听常见的播放器，也可以手动添加新的播放器类型到相应的配置文件中。监听模块获取播放器窗口的坐标信息，发送到服务端，服务端根据窗口坐标，再加上自身对视频区域的判断，可以更加精确的检测出视频播放区域。服务端在向客户端发送更新区域的同时，也发送播放窗口的坐标信息，这样客户端根据收到的信息，在对应的位置进行视频播放。

## 4.4 流媒体传输机制实现

### 4.4.1 流媒体服务端的整体设计

流媒体服务器负责处理视频数据，进行RTP打包处理，然后通过socket发送到客户端，同时创建RTCP处理线程，动态地计算出当前网络的发送速率，达到对网络拥塞控制的目的。流媒体服务器内运行RTSP服务线程，当有客户端建立连接时，服务器会创建响应客户端的RTP会话，并进行RTP封包。

实时流传输协议RTSP采用C/S模型，是一个基于文本的协议，允许服务端自行选择UDP或TCP协议来传输流媒体数据。RTSP协议是一个双向协议，客户端和服务端都可以发出请求。Live555支持RTSP/RTP/RTCP等协议，提供了单播和组播等模式，支持多种常见的媒体编码格式流，而且Live555本身是一个跨平台的开源项目，是实现RTSP协议的最精炼的开源代码，能够很方便地移植到各种系统中，Live555转发模式的顺序图如图25所示[[[44]](#endnote-45)]。客户端在RTSP服务器上注册，RTSPServer在收到客户端的DESCRIBE请求后，根据请求中的媒体名字查找ServerMediaSession列表。客户端发送SETUP请求后，通过一系列的CreateNew()函数创建新的数据源和RTPSink。客户端发送PLAY请求时，服务器开始进行数据传输。每当Source检测到数据时，就把数据发送给RTPSink。

在liveMedia库中有Source，Sink以及Filter等元素。Source是生产数据的元素，比如通过RTP读取数据。Sink是消费数据的元素，比如把接收到的数据存储到文件，这个文件就是一个Sink。数据流经过多个'source'和'sink'，下面是一个示例：

数据接收的终点是Sink类，MediaSink是所有Sink类的基类。Sink类实现对数据的处理是通过实现纯虚函数continuePlaying()，通常情况下continuePlaying调用fSource-> getNextFrame来为Source设置数据缓冲区，处理数据的回调函数等，fSource是MediaSink的类型为FramedSource\*的类成员。



图25 Live555转发模式顺序图

本文中的RTSP流媒体服务器采用了Live555开源协议栈。Live555开源库默认通过mediaServer目录中MediaServer实现音频视频文件的点播功能，本文中主要实现单播模式下实时流媒体的点播方式，需要改写Live555的源代码，可以通过继承相关类并重写其中的方法来实现对mkv格式的媒体文件的流媒体播放功能。在Live555开源库的基础上，添加MatroskaStreamParser类，用来解析mkv格式的视频文件，同时把数据发到接收端Sink。读取mkv视频文件后，按照mkv格式提取一帧数据，然后把数据发送给RTPSink，RTPSink根据RTP传输协议把该帧数据通过网络发送到播放器。在流媒体服务端，Medium是所有类的基类，根据不同功能其他类继承自Medium或其Medium的子类。MatroskaVideoSink接收流媒体数据，然后根据请求发送到客户端进行播放。SeverMediaSession维护当前客户端连接会话，在本文的实现中，一个Server只能对应一个播放客户端。Media服务端主要的类图结构如图26所示。



图26 Media服务端主要类图结构

Live555静态库采用的是事件驱动模式，视频主模块程序设计框架如下：

1. 建立基本任务，创建一个BasicUsageEnvironment对象用于处理信息输出、事件调度、延迟事件处理，创建方法如下：

TaskScheduler scheduler = BasicTaskScheduler::createNew();

UsageEnvironment\* env=BasicUsageEnvironment::createNew(\*scheduler);

TaskSecheduler 类是任务调度器，是整个Live555 的任务调度中心，主要负责任务的调度和执行；UsageEnvironment 代表了整个系统的运行环境，主要用于错误的输入和输出；

1. 创建MatroskaFile，并实现其中的createNew()方法，主要负责为 RTSP服务器提供源码流；
2. 创建RTSPserver对象；
3. 进入主循环doEventLoop()；
4. 创建RTSPClientSession对象。

本文中涉及到的主要类如图27所示。通过MatroskaStreamParser对Matroska视频文件进行解析，最后由MatroskaVideoRTPSink对每一帧的视频数据进行RTP打包，然后发送到客户端进行播放。其中MatroskaStreamParser类保存了视频的当前帧以及全部的视频帧数，通过seekStream( )函数可以随机访问指定的某一帧视频，doGetNextFrame( )函数读取下一视频帧。MatroskaVideoRTPSink类通过continuePlaying( )和stopPlaying( )等函数控制视频播放，同时保存当前缓冲区的大小，发送数据包的大小，当前视频的播放进度。



图27 SPICEx协议涉及的主要类

### 4.4.2 GStreamer视频播放

基于GStreamer的流媒体播放器的功能模块与普通的基于GStreamer的媒体播放器一样。为了支持流媒体传输，需要构造RTP/RTCP传输插件[[[45]](#endnote-46)]。RTP/RTCP传输插件是以元件的形式出现的。源文件元件负责读取源文件，然后转交给转换元件。转换元件负责将数据组织成GstBuffer的形式，再转交给RTP传输元件。RTP传输元件使用开源流媒体库Jrtplib对GstBuffer中的数据进行RTP封装，并调用RTPSession.SendPacket()函数，将流媒体数据传输给客户端。在会话的过程中,服务器端可以调用RTPSession类的SetDefaultPayloadType()、SetDefaultMark()以及SetDefaultTimeStampIncrement()来调整传输的参数，实现对流媒体传输的控制。

初始化基于GStreamer的RTSP流媒体播放器：

|  |
| --- |
| int gst\_rtsp\_init(char\* file)  {  ……  GstStateChangeReturn state\_ret;  /\* initialization \*/  gst\_init(NULL, NULL);  GMainLoop\* loop = g\_main\_loop\_new(NULL, FALSE);  main\_loop = loop;  build\_gstreamer\_pipeline(rtsp, loop);  sgst\_element\_set\_state((GstElement \*)play, GST\_STATE\_PLAYING);  ……  g\_main\_loop\_run(loop);  ……  return 0;  } |

创建管道，添加playbin元素，并设置playbin的uri属性：

|  |
| --- |
| static build\_gstreamer\_pipeline(const gchar \*uri, GMainLoop\* loop) {  play = gst\_pipeline\_new ("Mediaplayer");  bin = gst\_element\_factory\_make ("playbin", "bin");  gst\_bin\_add (GST\_BIN (play), bin);  ……  g\_object\_set (G\_OBJECT (bin), "uri", uri, NULL);  return TRUE;  } |

## 4.5 本章小结

本章首先介绍了SPICE协议视频优化相关的技术基础，主要包括JPEG2000压缩算法、流媒体视频传输以及GStreamer多媒体框架等，然后分别从三个角度说明了本文对SPICE协议所做的优化，详细描述了视频压缩算法的改进，虚拟桌面视频区域检测机制增强，基于Live555的流媒体视频传输机制。

# 第五章 SPICEx视频优化验证

本章首先概述用于视频评估的主要方法，然后针对SPICEx协议视频优化方法，提出了验证方案，最后从不同的方面验证优化后的SPICEx协议可以提供比原始SPICE协议性能良好的视频播放体验。

## 5.1视频质量评估

视频质量评估方法通常分为两类：主观质量评估(Subjective Quality Assessment, SQA)和客观质量评估方法(Objective Quality Assessment, OQA)。

### 5.1.1主观质量评估方法介绍

各种图像、视频系统的最终目的是为用户提供相应的服务，所以用户对视频质量的直观感受，即主观评估，是最为准确和重要的方法。主观质量评估是选择一批非专家类型的受测者，在一个受控的环境中，连续观看一系列的测试视频，然后采用统计方法对不同视频质量进行评分，最后计算出平均评分(Mean Option Score, MOS)，并对所得数据进行分析。受控环境包括观看距离、观测环境、测试视频的选择等。对于平均评分MOS值，通常采用5分制来表征视频质量的好坏，具体定义见表3。

表3 视频主观质量的评分标准

|  |  |  |
| --- | --- | --- |
| 数值评分 | 视频质量级别 | 视频失真级别 |
| 5 | 优(Perfect) | 没有察觉 |
| 4 | 良(Good) | 有察觉但不可厌 |
| 3 | 中(Fair) | 察觉且略微可厌 |
| 2 | 差(Poor) | 可厌但不令人反感 |
| 1 | 坏(Unusable) | 极可厌且令人反感 |

主观质量评估的优点是可以适用于各种视频系统，包括传统的模拟系统和压缩系统，可以得到平均评分MOS，能很好地应用于各种静态和运动图像系统中。不足之处主要有，必须提供严格的测试环境，考虑大量的影响因素和可能性，需要组织一定的人力资源作为受测者，实现起来步骤复杂，代价昂贵，实时性差。常用的主观评估方法有：

* 1. Double Stimulus Impairment Scale(DSIS)

受测者观看多个原始参考图像、失真图像组成的图像对，每次都是先显示原始参考图像，然后显示失真图像，对失真图像的整体失真程度进行打分。

* 1. Double Stimulus Continuous Quality Scale(DSCQS)

受测者观看多个原始图像与失真图像组成的图像对，且原始图像和失真图像的显示顺序是随机的，受测者参考其中的一个图像对另一个图像进行评分。

* 1. Single Stimulus Method(SSM)

随机组合若干原始参考图像和失真图像并独立显示每幅图像，受测者对其质量进行打分。可以选择重复或者不重复放映测试序列。

* 1. Just Noticeable Difference(JND)[[[46]](#endnote-47)]

JND是IEEE建议的主观质量评估方法，1个JND单位定义为，对于两个输入图像，大约有75%的受测者能分辨出差别，2个JND单位对应于93.75%的受测者可以分辨差别，3个对应于98.44%。JND方法更符合人的视觉特征。

### 5.1.2客观质量评估方法介绍

图像质量客观评估方法根据原始参考图像的存在与否分为全参考、部分参考和无参考三类。视频质量客观评估方法也可分为三类：

1. 全参考方法(Full Reference, FR)

全参考方法要求必须提供完整的原始视频信息和待测视频信息[[[47]](#endnote-48)]。通过一定的算法进行比较，对待测视频序列质量做出评判。

目前广泛应用的峰值信噪比(Peak Signal to Noise Ratio, PSNR)算法采用的就是全参考方法，计算公式为：

(5.1)

为视频信号的峰值，由于一般图像为256色，每个彩色分量用8为表示，所以一般取。MSE为均方误差(Mean Square Error, MSE)，其定义为：

(5.2)

其中K为视频序列的帧数，M x N为图像的大小，和分别代表原始参考图像和失真图像在第*k*帧视频序列点(m, n)处的像素值。这种基于数学统计值的客观评估方法计算简单，计算量小，能够较好地满足实时质量评估的需求并且具有明显的物理意义。

1. 部分参考方法(Reduced Reference, RR)

部分参考方法中没有原始视频的完整信息，只有按一定方法提取出来的原始视频特征数据信息，然后以相同的方法提取出待测视频数据，最后对比两者特征信息，对视频质量进行评判。

1. 无参考方法(No Reference, NR)

无参考方法没有任何的原始视频信息，通过对测试视频的处理和分析，提取视频序列中某些失真特征，然后根据各类失真特征判定视频质量。

### 5.1.3 SPICEx视频质量评估

虚拟桌面视频播放的性能由用户在终端的实际体验来评判[[[48]](#endnote-49)]，本文采用主观质量评估方法和客观质量评估方法相结合的验证方案，其中主观质量评估方法采用类似DSIS的方法进行评估，受测者通过观测原始视频和待测视频序列，分别给出视频观测的主观评分。客观质量评估采用了两种方法，慢镜头检测方法和基于数学统计的部分参考评估。

## 5.2测试方案

### 5.2.1选取测试对象

为了验证本文对SPICE协议视频播放优化的有效性，选取3段1080P高清测试视频，测试视频的详细信息如表4、表5和表6所示。

表4 测试视频1详细参数

|  |  |
| --- | --- |
| 视频流信息 | 音频流信息 |
| +编码格式: V\_MPEG4/ISO/AVC  +视频码率: 0 kbps  +视频帧率：30 fps  +分 辨 率: 1920 x 1080  +显示比率: 1.778 | +编码格式: A\_AAC/MPEG4/MAIN  +音频码率: 0 kbps  +声 道 数: 2 channels  +采 样 数: 48000 Hz  +音频位数: 0 bits |
| 视频大小：187MB 视频长度：00:04:15 | |

表5 测试视频2详细参数

|  |  |
| --- | --- |
| 视频流信息 | 音频流信息 |
| +编码格式: V\_MPEG4/ISO/AVC  +视频码率: 0 kbps  +视频帧率：24 fps  +分 辨 率: 1440 x 1080  +显示比率: 1.333 | +编码格式: A\_AAC/MPEG4/MAIN  +音频码率: 0 kbps  +声 道 数: 2 channels  +采 样 数: 44100 Hz  +音频位数: 0 bits |
| 视频大小：85MB 视频长度：00:01:58 | |

表6 测试视频3详细参数

|  |  |
| --- | --- |
| 视频流信息 | 音频流信息 |
| +编码格式: V\_MPEG4/ISO/AVC  +视频码率: 0 kbps  +视频帧率：30 fps  +分 辨 率: 1920 x 1080  +显示比率: 1.778 | +编码格式: A\_AAC/MPEG4/MAIN  +音频码率: 0 kbps  +声 道 数: 2 channels  +采 样 数: 48000 Hz  +音频位数: 0 bits |
| 视频大小：151MB 视频长度：00:03:37 | |

### 5.2.2测试环境配置

在测试环境的配置过程中，为了避免硬件配置对测试本身造成影响，所有的测试场景都在特定的测试环境中完成。对优化后的SPICEx协议，针对视频播放性能进行了测试和分析。本文采用的测试环境如表7所示：

表7 测试环境配置

|  |  |  |  |
| --- | --- | --- | --- |
|  | 硬件配置 | 操作系统 | 软件环境 |
| 服务器 | CPU Intel i7 2.2 GHz  内存4 GB  硬盘250 GB  网卡100BaseT NIC | 中标麒麟  NeoShineV6.0 | QEMU 0.14  SPICE server |
| 虚拟机 | 内存1GB | Windows7 32位 | QQ影音，WinAgent |

**表7 测试环境配置(续)**

|  |  |  |  |
| --- | --- | --- | --- |
| 客户端 | CPU Intel i3 2.2 GHz  内存2 GB  硬盘80 GB  网卡100BaseT NIC | 中标麒麟  NeoShineV6.0 | SPICE Client |
| 交换机 | 可以控制不同的网络带宽 | | |

测试过程在不同的网络带宽条件下进行，由交换机控制当前可用的最大带宽，测试的带宽分别为100M、10M和4M。整个测试环境如图28所示：



图28 测试环境示意图

### 5.2.3主观质量评估方案

通过20名受测者观看视频的直观感受，参考MOS判定标准在不同的测试场景下进行评分，最后计算出每种测试场景中视频质量的平均值。为了准确评估视频质量，本文分为4个测试场景进行主观质量评估。4个测试场景分别是：

1. 客户端本地播放

由于本地视频播放不涉及到网络带宽，因此无需交换机控制网络带宽。在该测试场景中，20名受测者分别独立地对3个原始视频片段进行评估。在评估每一段视频时，要求每一位受测者每隔10s就针对当前的视频片段做出评估，并给出相应的评分，最后给出整个视频的整体评分。综合所有受测者给出的评分，计算得到每一个测试视频的MOS评分，作为参考的基准。

1. 原始SPICE视频播放

本测试场景中，在测试服务器上搭建中标麒麟NeoShineV6.0操作系统，然后部署QEMU 0.14，在QEMU中配置libspice，客户机同样安装中标麒麟NeoShineV6.0操作系统，然后部署SPICE 客户端。服务器上启动QEMU，运行Windows7虚拟机，客户端通过交换机连接到虚拟桌面。连接成功后打开QQ影音播放器，分别播放3个测试视频片段，由同样的20名受测者对播放的视频进行主观质量评估。

在该测试场景中，为了验证网络带宽对视频播放质量的影响，整个评估过程分别在3种网络环境下进行，网络带宽分别限制为100M、10M和4M。

1. SPICE+MJPEG2000视频播放

本测试场景和原始SPICE视频播放类似，需要在QEMU虚拟机管理器中配置支持MJPEG2000编码的libspice，同时客户机上部署支持MJPEG2000解码的SPICE 客户端。同样在3种网络环境下进行视频质量评估。

1. SPICE+流媒体视频播放

本测试场景和原始SPICE视频播放类似，需要在Windows7虚拟机中部署一个Agent模块，该模块用于探测当前视频播放的窗口信息，同时启动一个流媒体服务器线程等待客户端发送连接请求。相应地，客户端需要配置能够解码流媒体视频的SPICE 客户端，然后连接到虚拟桌面播放测试视频。评估过程同原始SPICE视频播放。

### 5.2.4客观质量评估方案

为了能够更加准确地评估协议优化后的视频播放效果，客观质量评估采用了两种方法，分别是慢镜头检测(Slow Motion Benchmark)和基于数学统计的部分参考方法。其中慢镜头检测方法通过比较两种播放速率(Playback Rate)下视频流量来评估视频质量，1fps视频帧率下从服务端传输到客户端的数据量作为理想视频帧率的参考，在这种评估方法中采用的视频质量Video Quality(VQ)定义[[[49]](#endnote-50)]如下：

在不同的网络带宽条件下计算SPICE和SPICEx协议对应的视频质量VQ，通过对比得到SPICE协议和SPICEx协议视频播放客观质量评估结果。

SPICE协议以及基于MJPEG2000压缩算法的SPICEx协议在客户端接收的不是完整的图像数据，可能只是视频窗口的一部分，而且会在图像压缩算法和视频压缩算法之间进行切换，因此，在这种情况下的视频的播放已经没有帧的概念。因此，在采用基于数学统计值的部分参考方法的过程中，针对测试视频本文采用在客户端截取整个视频帧画面的方法来近似提取视频帧，这样可以更接近用户观看的效果。在实际的测试过程中，选取每秒视频的第一帧，即针对每一个测试视频片段，从第一秒开始每隔1s截取一帧视频画面。针对原始参考视频和失真视频对应的同一帧计算均方差MSE以及峰值信噪比PSNR。根据计算结果评估客户端视频播放的质量。

在客户端通过import工具进行视频画面的无损提取，保存成无压缩的bmp格式图片。在使用import的过程中，需要指定视频播放的窗口以及需要截取的窗口坐标和大小。通过时间同步，控制截取到的原始视频和测试视频图像是同一帧。针对原始视频和测试视频，采用相同的视频图像截取算法，这样可以保证获得的是同一帧图像。根据得到的原始参考图像和失真图像像素值，计算MSE、PSNR，并给出视频质量评估结果。客观质量评估过程同样在3种不同的网络条件下进行，得到的结果可以和主观质量评估的结果相互印证。

在评估视频质量的同时，收集虚拟桌面视频播放性能的影响因素，主要包括网络带宽、CPU利用率、内存利用率等。网络带宽的检测通过Linux网络流量实时监控工具nload实现，nload是Linux下的一款实时流量监控工具，可以显示系统内所有或者单个网卡的实时流量，可区分进入和流出的流量。CPU利用率通过计算CPU执行非系统空闲进程的时间与CPU总的执行时间的比值得到。具体实现中，通过读取/proc/stat文件来获取CPU用执行用户态和系统态的jiffies，然后除以总的jiffies来表示。内存使用率通过读取/proc/meminfo文件并计算两秒之间的差异获得。

## 5.3测试结果分析

### 5.3.1主观质量评估结果

20名受测者通过对原始视频和测试视频的直观观测，给出了不同网络带宽条件下的主观评估得分(附录A)，分别计算出测试视频的最终MOS得分。采用加权平均的方法进行计算，即受测者在观看过程中的评分赋予0.6的权值，对视频的整体评分赋予0.4的权值。针对3个测试视频的评估结果如图29、图30和图31所示：

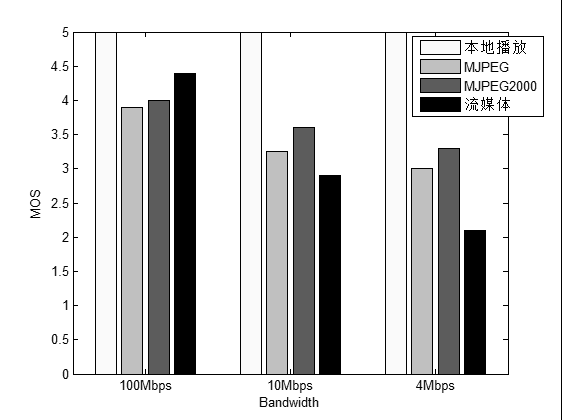


图29 测试视频1主观评估结果

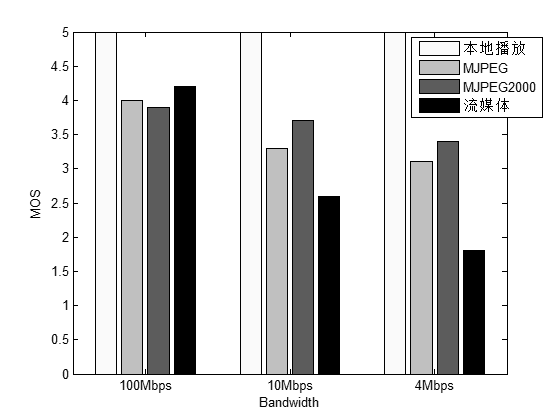


图30 测试视频2主观评估结果

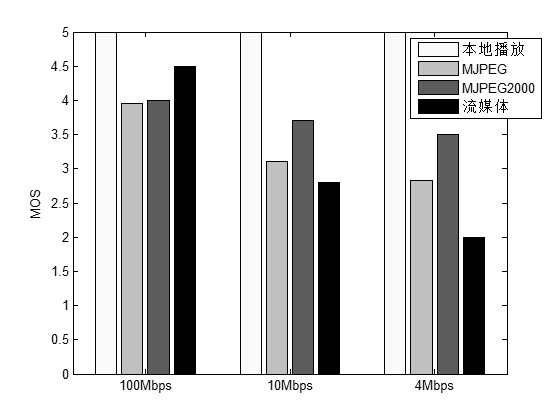


图31 测试视频3主观评估结果

从以上的评估结果可以看出，与本地视频播放相比，SPICE协议和SPICEx协议都有一定程度的视频画面失真。在网络状况较好的条件下，从20名受测者的直观体验来看，使用MJPEG2000压缩算法的SPICEx协议进行视频播放时，播放效果与SPICE协议基本相同，换句话说，通过修改视频压缩算法并没有在很大程度上提高用户的视频播放体验。对于流媒体视频播放，虽然也会有一定的画面失真，但总的来说其播放效果明显优于SPICE协议的MJPEG视频压缩和SPICEx协议的MJPEG2000视频压缩，在一定程度上提升了虚拟桌面用户视频播放体验。从网络带宽的角度来看，MJPEG2000视频压缩受网络带宽的影响最小，在低带宽网络条件下可以提供优于MJPEG视频压缩的播放效果。MJPEG视频压缩随着网络带宽的降低，视频播放效果会变差。受带宽影响最大的是通过流媒体进行视频播放，在带宽降低到4M时，播放效果已经降低到无法正常观看的程度。

从主观视频质量评估结果可以看出，流媒体的视频播放需要在网络条件相对较好的情况下才可以提供良好的用户体验，而MJPEG2000视频压缩适合于低带宽的网络环境。用户在使用的过程中可以根据当前的网络状况，把两者结合起来，以获得最优的视频播放效果。

### 5.3.2客观质量评估结果

5.3.2.1慢镜头检测的客观评估结果

在慢镜头检测评估的过程中，首先比较了不同帧率下服务端发送到客户端的数据量，使用iftop工具进行数据量统计。分别针对MJPEG，MJPEG2000和流媒体在理想帧率(30fps或者24fps)和1fps传输的数据量进行统计。图32、图33和图34分别是3段测试视频在不同网络带宽环境下的比较的结果。

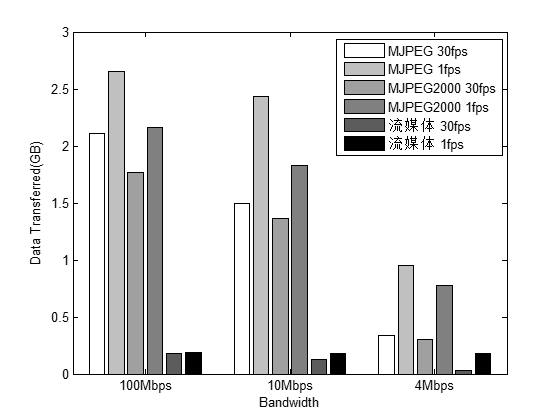


图32 测试视频1数据传输量

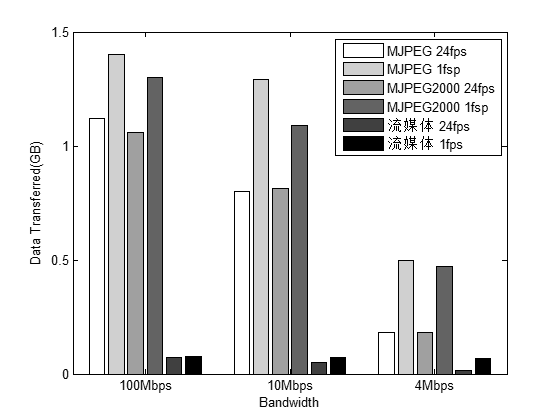


图33 测试视频2数据传输量

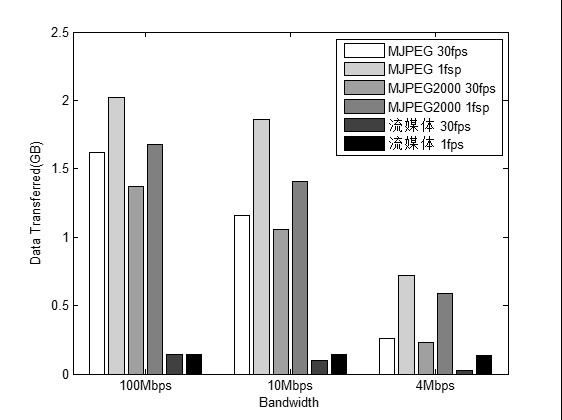


图34 测试视频3数据传输量

从图中可以看出，两种帧率下的MJPEG、MJPEG2000以及流媒体视频传输都会发生数据丢失，主要是因为视频数据包在网络的传输过程中会产生一定的延时，因此在视频播放过程中会跳过已经过时的帧，从而产生数据丢失。同时，网络带宽的变化也会对数据包的丢失产生较大的影响，相对于SPICE协议来说，基于MJPEG2000的SPICEx协议数据丢失率小，受带宽的影响较小。流媒体视频传输在低带宽环境下数据丢失率很高，受带宽限制最大。

视频质量VQ使用公式5.1进行计算，3组测试视频得到结果如图35、图36和图37所示：

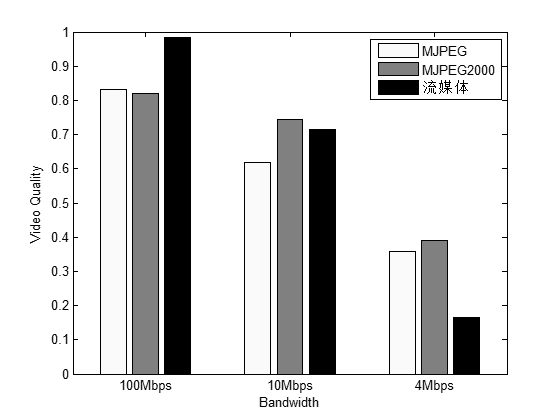


图35 测试视频1视频质量

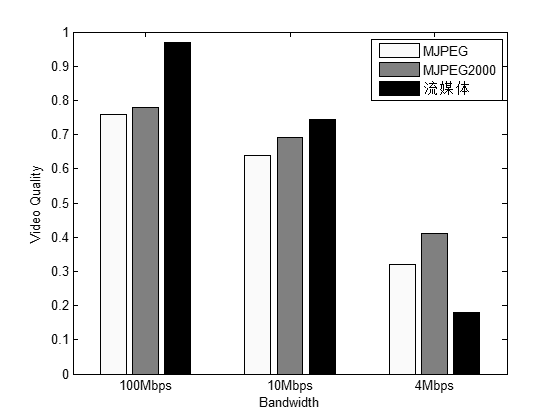


图36 测试视频2视频质量

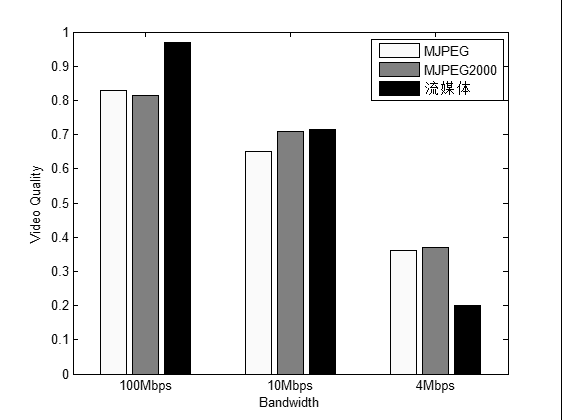
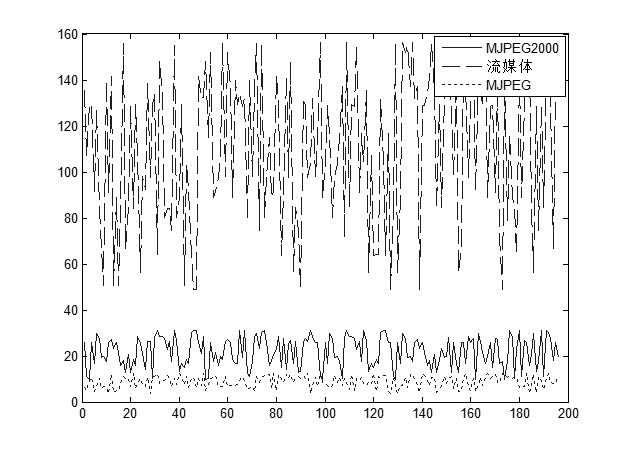


图37 测试视频3视频质量

从图中的比较可以看出，在100M的网络带宽下，SPICE协议和基于MJPEG2000的SPICEx协议可以保持超过80%的视频质量，SPICE协议略优于SPICEx协议。但是在10M和4M的网络环境中，SPICEx协议比SPICE协议的视频播放质量要好。在10M带宽条件下，SPICEx协议能够保持70%以上的视频质量，但是在4M时，只能维持不到40%的质量。对于流媒体视频传输，在100M带宽的网络环境中可以提供良好的视频播放效果，随着网络带宽的降低，流媒体视频播放质量急剧下降，在4M的环境中，仅能保持20%的视频质量，可见流媒体视频传输依赖于较高的网络带宽，不适用于低带宽的网络环境。

5.3.2.2基于数学统计的部分参考评估结果

在100M带宽的网络环境下，针对测试视频1评估得到的SPICE协议和SPICEx协议播放视频的PSNR如图38所示，从图中可以看出，流媒体视频播放效果最好，SPICE协议视频播放的PSNR最低，表明其画面失真程度最大。在不同网络带宽环境下计算得到的平均PSNR如图39所示。图40和图41是视频2和视频3在不同网络环境中得到的平均PSNR，可以看出基于数学统计的部分参考客观评估的结果与主观评估的结果是一致的。



Time(s)

PSNR

图38 测试视频1的实时PSNR

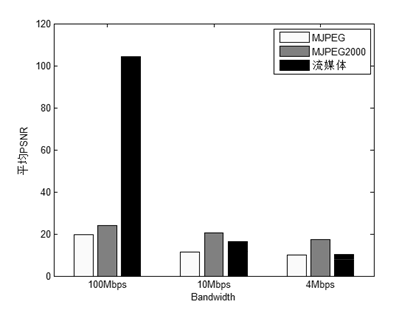


图39 测试视频1平均PSNR

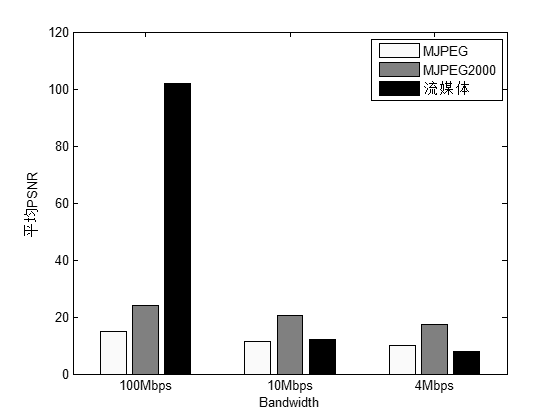


图40 测试视频2平均PSNR

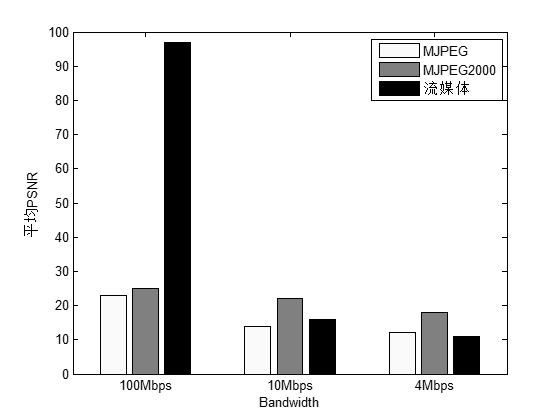


图41 测试视频3平均PSNR

5.3.2.3其他性能影响因素比较

在测试视频播放质量的同时，考虑了客户端的负载情况，从其他影响视频播放效果的角度对SPICEx协议进行评估。本文主要考虑了在播放视频过程中使用的网络带宽、CPU平均使用率和内存使用情况，评估的结果如图所示。

图42显示在100M带宽的网络环境中，SPICE协议占用的带宽最高，超过10MB/s，同样的网络环境下，MJPEG2000压缩算法仅使用了500KB/s的网络带宽，而流媒体视频传输为了保证视频播放的流畅性需要1MB/s以上的带宽。随着网络环境所提供带宽的降低，可以看出SPICE协议占用了网络环境的最大带宽。MJPEG2000算法维持在400KB/s到500KB/s之间，受网络环境影响最小。流媒体视频传输在10M的网络环境中平均占用1MB/s的带宽，但是在4M的环境下，处于满带宽工作状态。

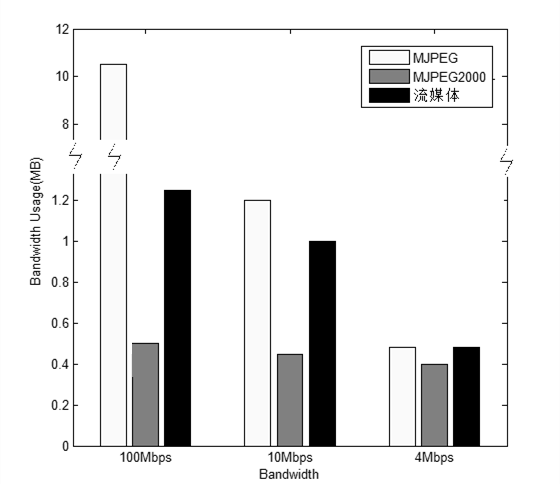


图42 平均网络带宽使用情况对比

从图43可以看出，MJPEG2000压缩算法对CPU的负载最大，主要是在JPEG2000解码的过程中需要消耗大量的CPU计算资源。流媒体视频播放消耗的CPU最少，此时客户端仅需要解码流媒体视频并进行本地播放，无需复杂的计算操作。SPICE协议基本保持在40%的CPU使用率。

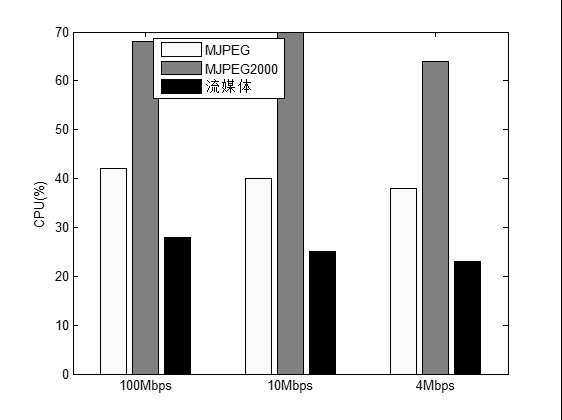


图43 平均CPU使用情况对比

图44显示了SPICE协议和SPICEx协议在播放视频过程中的内存消耗情况，MJPEG2000压缩算法的内存使用率略高于MJPEG和流媒体视频传输，同时可以看出内存使用情况不受网络带宽的影响。

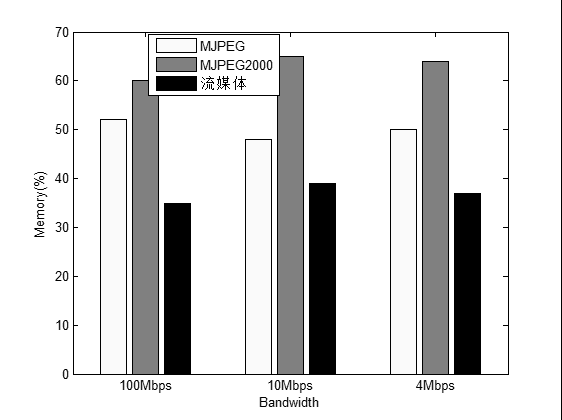


图44 平均内存使用情况对比

## 5.4本章小结

本章首先总结了当前常用的视频质量评估的主客观方法，并说明了本文采用的验证方案。然后详细描述了测试对象以及测试环境配置，主观质量评估和客观质量评估的具体方法步骤。最后对优化后的SPICEx协议与原始的SPICE协议进行了不同角度的对比和分析，实验验证得到SPICEx协议可以通过MJPEG2000压缩算法在低带宽的网络环境下提供较好视频播放效果，在网络环境较好时可以通过流媒体进行视频播放，并保持很好的播放效果。

# 结论

本章从论文成果和进一步工作两个方面对本文进行总结。

## 论文成果

论文以实现一个可以流畅播放高清视频的虚拟桌面显示协议SPICEx为目标，首先深入研究了开源的虚拟桌面显示协议SPICE的图像视频传输机制及其关键技术，然后分别从3个方面对SPICE协议进行改进与优化。论文取得的研究成果包括：改进SPICE协议的视频压缩算法、增强SPICE协议的视频检测机制、引入流媒体视频传输机制。

SPICE协议提供了3种图像压缩算法和一种视频压缩算法，这些压缩算法可以在服务器启动时指定，也可以在运行时动态地改变。SPICE会启发式的检测当前桌面的图像类型，针对不同的图像选用不同的压缩算，目的是最大限度的在用户终端上还原桌面环境。针对视频压缩算法对网络带宽要求很高且视频播放效果不理想的问题，本论文采用MJPEG2000压缩算法对SPICE协议进行改进，提高了图像压缩效率，有效地减少了网络中传输的数据量，降低了对网络环境带宽的要求。

为了提高视频播放检测的准确性，本论文采用了窗口监听机制，即在虚拟机内部运行一个后台程序检测当前桌面上运行的所有窗口，一旦发现有播放器在运行，即判断有视频在播放，同时将视频播放的控制功能由虚拟机转移到客户端，客户端直接控制视频的播放进度。在检测到视频播放后，启动一个单独的线程运行流媒体服务进程，为客户端请求流媒体做好准备。

高清视频的流畅播放一直是虚拟桌面解决方案难以克服的一个难题。为了提升用户观看1080P高清视频的体验，本论文引入了流媒体视频传输机制，通过扩展Live555开源流媒体库实现一个流媒体服务程序，客户端对流媒体视频进行解码的是GStreamer媒体库。通过GStreamer来构建流媒体应用多媒体框架，简化了音/视频应用程序的开发，目前已经能够被用来处理像MP3、Ogg、MPEG1、MPEG2、AVI、Quicktime等多种格式的多媒体数据。利用GStreamer，可以构建一系列的媒体处理模块，包括从简单的ogg播放功能到复杂的音频(混音)和视频(非线性编辑)的处理。应用程序可以透明的利用解码和过滤技术。GStreamer所具备的很多优点来源于其框架的模块化，GStreamer能够无缝的合并新的插件。

## 进一步工作

本文对开源的虚拟桌面显示协议SPICE进行了深入的研究与探讨，并实现了部分视频播放的改进与优化。由于时间和工作量的限制，本文的研究与SPICEx协议的实现还存在不足，进一步的工作将在以下几点展开：

1) 进一步改进SPICE协议的图像压缩算法，将多种图像压缩算法统一起来，避免不同压缩算法切换导致的虚拟桌面抖动问题。目前Motion JPEG2000压缩算法使用固定的压缩率，不能根据网络环境的变化自适应地调整压缩率，因此需要进一步改进该算法，在传输桌面环境的同时检测当前的网络状况，并自动调整压缩率来适应网络环境。

2) SPICEx协议中的流媒体视频传输需要虚拟机提供外部可访问的IP地址，这样客户端才能向虚拟机的流媒体服务程序请求视频流。SPICE协议的实现中没有对此进行要求，虚拟机的图像数据是通过SPICE 服务端作为中间媒介进行传输的。为了与SPICE协议保持兼容性，需要进一步改进SPICE 服务端，使其可以接收虚拟机中的流媒体数据，然后转发到客户端进行播放。

# 参考文献

# 附录

**附录A 主观视频质量评估部分数据**

**表8 本地视频播放评估结果**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 测试者 | 测试过程评分 | | | | | | | | | | | | | | | | | | | | | | 整体  评分 |
| 1 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| 2 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| 3 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| 4 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 4 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| 6 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| 7 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| 8 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| 9 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 4 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| 10 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| 11 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| 12 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 4 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| 13 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| 14 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 4 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| 15 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| 16 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| 17 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| 18 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| 19 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| 20 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |

**表9 100M带宽MJPEG视频播放评估结果**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 测试者 | 测试过程评分 | | | | | | | | | | | | | | | | | | | | | | 整体  评分 |
| 1 | 4 | 4 | 3 | 3 | 3 | 3 | 4 | 5 | 3 | 4 | 3 | 4 | 4 | 3 | 4 | 3 | 4 | 4 | 4 | 5 | 3 | 3 | 4 |
| 2 | 4 | 3 | 3 | 3 | 3 | 4 | 5 | 4 | 4 | 3 | 3 | 3 | 3 | 4 | 3 | 3 | 3 | 4 | 4 | 4 | 3 | 4 | 3 |
| 3 | 3 | 3 | 3 | 3 | 3 | 4 | 4 | 4 | 3 | 4 | 3 | 3 | 4 | 4 | 4 | 4 | 3 | 5 | 4 | 3 | 3 | 5 | 3 |
| 4 | 4 | 4 | 3 | 3 | 3 | 4 | 4 | 3 | 3 | 3 | 4 | 3 | 4 | 3 | 5 | 3 | 2 | 4 | 4 | 3 | 4 | 4 | 4 |
| 5 | 3 | 3 | 4 | 3 | 4 | 3 | 4 | 4 | 3 | 3 | 3 | 3 | 4 | 4 | 4 | 3 | 3 | 3 | 4 | 4 | 3 | 3 | 3 |
| 6 | 4 | 3 | 3 | 4 | 3 | 3 | 3 | 4 | 4 | 4 | 3 | 3 | 5 | 3 | 3 | 3 | 4 | 4 | 5 | 4 | 3 | 4 | 3 |
| 7 | 3 | 4 | 4 | 3 | 3 | 3 | 4 | 3 | 4 | 3 | 3 | 4 | 5 | 4 | 4 | 3 | 4 | 4 | 3 | 4 | 4 | 3 | 3 |
| 8 | 4 | 5 | 3 | 4 | 3 | 4 | 3 | 3 | 3 | 3 | 4 | 3 | 5 | 4 | 4 | 3 | 4 | 3 | 4 | 3 | 5 | 4 | 4 |
| 9 | 4 | 4 | 3 | 3 | 4 | 4 | 4 | 3 | 4 | 4 | 3 | 3 | 3 | 4 | 5 | 3 | 4 | 4 | 3 | 3 | 3 | 4 | 4 |
| 10 | 3 | 3 | 4 | 3 | 3 | 4 | 4 | 3 | 5 | 3 | 3 | 3 | 4 | 5 | 4 | 4 | 3 | 3 | 3 | 3 | 4 | 3 | 4 |

**表9 100M带宽MJPEG视频播放评估结果(续)**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 测试者 | 测试过程评分 | | | | | | | | | | | | | | | | | | | | | | 整体  评分 |
| 11 | 3 | 4 | 4 | 3 | 4 | 3 | 3 | 3 | 3 | 3 | 3 | 4 | 4 | 4 | 3 | 3 | 4 | 4 | 4 | 3 | 5 | 3 | 3 |
| 12 | 4 | 4 | 3 | 4 | 3 | 4 | 3 | 2 | 3 | 4 | 3 | 3 | 4 | 4 | 3 | 3 | 4 | 5 | 3 | 4 | 4 | 4 | 4 |
| 13 | 3 | 3 | 4 | 3 | 4 | 4 | 3 | 3 | 4 | 3 | 4 | 3 | 3 | 4 | 4 | 3 | 3 | 4 | 3 | 3 | 3 | 3 | 4 |
| 14 | 3 | 5 | 3 | 4 | 4 | 5 | 4 | 3 | 4 | 3 | 4 | 4 | 3 | 3 | 4 | 4 | 4 | 3 | 3 | 3 | 4 | 3 | 4 |
| 15 | 4 | 5 | 4 | 3 | 3 | 3 | 4 | 3 | 4 | 4 | 4 | 3 | 3 | 4 | 3 | 4 | 4 | 4 | 3 | 4 | 3 | 4 | 4 |
| 16 | 3 | 5 | 4 | 4 | 3 | 4 | 3 | 3 | 3 | 5 | 3 | 4 | 4 | 3 | 3 | 3 | 3 | 4 | 3 | 5 | 4 | 3 | 4 |
| 17 | 3 | 4 | 3 | 4 | 3 | 3 | 3 | 4 | 3 | 4 | 4 | 3 | 4 | 5 | 4 | 3 | 3 | 3 | 4 | 4 | 4 | 3 | 3 |
| 18 | 4 | 3 | 3 | 3 | 3 | 4 | 3 | 4 | 4 | 3 | 3 | 3 | 4 | 3 | 4 | 4 | 4 | 4 | 4 | 3 | 4 | 4 | 3 |
| 19 | 3 | 3 | 3 | 4 | 4 | 3 | 3 | 5 | 3 | 4 | 2 | 4 | 3 | 4 | 3 | 5 | 4 | 4 | 4 | 4 | 5 | 3 | 4 |
| 20 | 4 | 4 | 3 | 4 | 4 | 3 | 5 | 4 | 4 | 3 | 3 | 4 | 4 | 4 | 3 | 3 | 3 | 4 | 3 | 3 | 3 | 4 | 3 |

**表10 100M带宽MJPEG2000视频播放评估结果**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 测试者 | 测试过程评分 | | | | | | | | | | | | | | | | | | | | | | 整体  评分 |
| 1 | 4 | 4 | 3 | 4 | 3 | 4 | 4 | 3 | 4 | 3 | 3 | 4 | 4 | 3 | 3 | 5 | 4 | 4 | 5 | 4 | 5 | 4 | 4 |
| 2 | 4 | 3 | 4 | 3 | 5 | 3 | 4 | 3 | 5 | 4 | 3 | 3 | 4 | 3 | 4 | 3 | 4 | 3 | 4 | 4 | 4 | 4 | 4 |
| 3 | 3 | 4 | 4 | 4 | 4 | 3 | 5 | 3 | 4 | 5 | 3 | 3 | 3 | 3 | 4 | 4 | 3 | 3 | 4 | 4 | 3 | 3 | 3 |
| 4 | 4 | 4 | 3 | 5 | 3 | 2 | 4 | 4 | 4 | 4 | 3 | 3 | 4 | 3 | 4 | 3 | 4 | 3 | 3 | 4 | 3 | 4 | 4 |
| 5 | 5 | 4 | 4 | 4 | 3 | 3 | 4 | 3 | 4 | 3 | 3 | 4 | 3 | 4 | 3 | 3 | 3 | 3 | 4 | 4 | 4 | 3 | 3 |
| 6 | 4 | 5 | 3 | 3 | 3 | 4 | 4 | 3 | 3 | 4 | 4 | 3 | 3 | 3 | 3 | 4 | 3 | 3 | 4 | 5 | 4 | 4 | 4 |
| 7 | 4 | 5 | 4 | 4 | 3 | 4 | 4 | 4 | 4 | 3 | 3 | 3 | 4 | 4 | 3 | 3 | 3 | 4 | 3 | 3 | 4 | 3 | 5 |
| 8 | 3 | 5 | 4 | 4 | 3 | 4 | 3 | 5 | 3 | 4 | 4 | 3 | 5 | 3 | 4 | 3 | 4 | 3 | 3 | 4 | 3 | 4 | 3 |
| 9 | 4 | 3 | 4 | 5 | 3 | 4 | 4 | 3 | 4 | 4 | 3 | 4 | 4 | 3 | 4 | 4 | 3 | 3 | 3 | 3 | 3 | 4 | 3 |
| 10 | 5 | 4 | 5 | 4 | 4 | 3 | 3 | 4 | 4 | 4 | 3 | 3 | 3 | 5 | 5 | 3 | 3 | 4 | 4 | 3 | 4 | 3 | 4 |
| 11 | 3 | 4 | 4 | 3 | 4 | 4 | 4 | 5 | 3 | 4 | 3 | 4 | 5 | 4 | 3 | 3 | 4 | 4 | 3 | 4 | 4 | 3 | 3 |
| 12 | 5 | 4 | 4 | 5 | 3 | 4 | 5 | 4 | 3 | 4 | 4 | 5 | 4 | 3 | 4 | 4 | 3 | 3 | 2 | 3 | 4 | 4 | 4 |
| 13 | 4 | 3 | 4 | 4 | 3 | 3 | 4 | 3 | 3 | 3 | 3 | 4 | 3 | 4 | 4 | 3 | 4 | 3 | 5 | 4 | 3 | 3 | 4 |
| 14 | 4 | 3 | 3 | 4 | 4 | 4 | 3 | 4 | 4 | 3 | 4 | 4 | 5 | 3 | 5 | 4 | 4 | 4 | 3 | 3 | 3 | 3 | 4 |
| 15 | 5 | 3 | 4 | 3 | 4 | 4 | 4 | 3 | 4 | 4 | 3 | 3 | 5 | 4 | 3 | 4 | 4 | 3 | 3 | 3 | 4 | 4 | 4 |
| 16 | 3 | 4 | 3 | 4 | 4 | 3 | 4 | 4 | 3 | 3 | 4 | 3 | 5 | 4 | 4 | 5 | 3 | 4 | 3 | 3 | 5 | 3 | 4 |
| 17 | 5 | 4 | 5 | 4 | 3 | 3 | 3 | 4 | 3 | 3 | 4 | 3 | 4 | 3 | 3 | 4 | 4 | 3 | 4 | 4 | 4 | 3 | 3 |
| 18 | 4 | 4 | 3 | 4 | 4 | 4 | 4 | 4 | 3 | 4 | 3 | 3 | 3 | 3 | 4 | 3 | 3 | 3 | 4 | 4 | 3 | 4 | 3 |
| 19 | 5 | 3 | 4 | 3 | 5 | 4 | 4 | 5 | 3 | 3 | 4 | 4 | 3 | 3 | 3 | 4 | 4 | 4 | 5 | 4 | 4 | 3 | 4 |
| 20 | 4 | 4 | 4 | 3 | 3 | 3 | 4 | 3 | 5 | 4 | 4 | 4 | 4 | 3 | 4 | 3 | 3 | 4 | 5 | 4 | 3 | 4 | 4 |

**表11 100M带宽流媒体视频播放评估结果**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 测试者 | 测试过程评分 | | | | | | | | | | | | | | | | | | | | | | 整体  评分 |
| 1 | 4 | 5 | 4 | 5 | 5 | 4 | 5 | 5 | 5 | 4 | 5 | 5 | 4 | 3 | 3 | 4 | 5 | 5 | 5 | 4 | 5 | 5 | 4 |
| 2 | 5 | 5 | 4 | 5 | 5 | 4 | 4 | 4 | 5 | 4 | 4 | 3 | 5 | 4 | 5 | 4 | 5 | 4 | 5 | 5 | 5 | 4 | 5 |
| 3 | 5 | 4 | 5 | 5 | 4 | 5 | 5 | 5 | 4 | 5 | 3 | 4 | 3 | 4 | 5 | 4 | 5 | 4 | 4 | 4 | 5 | 4 | 5 |
| 4 | 4 | 5 | 5 | 5 | 5 | 5 | 4 | 5 | 4 | 4 | 3 | 5 | 4 | 3 | 5 | 5 | 5 | 5 | 5 | 4 | 4 | 4 | 4 |
| 5 | 5 | 5 | 5 | 5 | 3 | 4 | 4 | 4 | 4 | 4 | 3 | 4 | 5 | 5 | 3 | 5 | 5 | 5 | 4 | 5 | 4 | 5 | 4 |
| 6 | 5 | 5 | 4 | 4 | 5 | 4 | 5 | 3 | 4 | 5 | 5 | 3 | 4 | 5 | 5 | 4 | 3 | 5 | 4 | 5 | 4 | 4 | 5 |
| 7 | 4 | 5 | 5 | 4 | 5 | 5 | 5 | 4 | 4 | 5 | 4 | 4 | 5 | 5 | 5 | 4 | 4 | 5 | 5 | 5 | 5 | 4 | 5 |
| 8 | 5 | 5 | 5 | 5 | 5 | 4 | 5 | 5 | 3 | 4 | 4 | 3 | 5 | 5 | 4 | 5 | 4 | 5 | 5 | 5 | 5 | 5 | 4 |
| 9 | 4 | 5 | 4 | 5 | 5 | 5 | 5 | 5 | 4 | 5 | 5 | 5 | 4 | 5 | 4 | 4 | 4 | 4 | 4 | 4 | 5 | 4 | 4 |
| 10 | 5 | 4 | 5 | 4 | 4 | 4 | 5 | 4 | 4 | 4 | 4 | 5 | 3 | 5 | 5 | 5 | 5 | 4 | 4 | 5 | 5 | 3 | 4 |
| 11 | 5 | 4 | 4 | 5 | 4 | 5 | 5 | 5 | 3 | 4 | 4 | 4 | 5 | 4 | 5 | 5 | 5 | 5 | 5 | 4 | 4 | 3 | 5 |
| 12 | 5 | 5 | 4 | 5 | 5 | 4 | 5 | 4 | 5 | 4 | 5 | 5 | 4 | 5 | 4 | 5 | 4 | 4 | 4 | 4 | 4 | 5 | 5 |
| 13 | 4 | 5 | 4 | 5 | 4 | 5 | 4 | 5 | 4 | 4 | 4 | 4 | 5 | 5 | 4 | 5 | 5 | 4 | 5 | 4 | 4 | 5 | 5 |
| 14 | 5 | 4 | 4 | 5 | 5 | 4 | 5 | 4 | 5 | 4 | 4 | 4 | 4 | 5 | 5 | 4 | 5 | 5 | 5 | 5 | 4 | 5 | 5 |
| 15 | 4 | 5 | 4 | 5 | 4 | 4 | 4 | 5 | 4 | 5 | 5 | 5 | 4 | 5 | 4 | 5 | 4 | 5 | 5 | 4 | 4 | 4 | 4 |
| 16 | 5 | 4 | 4 | 4 | 5 | 5 | 4 | 4 | 4 | 4 | 4 | 4 | 5 | 4 | 4 | 5 | 4 | 4 | 5 | 5 | 5 | 5 | 5 |
| 17 | 4 | 4 | 5 | 4 | 5 | 5 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 5 | 3 | 4 | 4 | 5 | 5 | 5 | 4 | 5 | 5 |
| 18 | 5 | 4 | 5 | 4 | 5 | 4 | 5 | 4 | 3 | 4 | 5 | 4 | 5 | 4 | 4 | 3 | 5 | 5 | 5 | 5 | 4 | 5 | 4 |
| 19 | 5 | 5 | 4 | 5 | 5 | 4 | 4 | 5 | 4 | 4 | 5 | 4 | 5 | 4 | 4 | 5 | 4 | 4 | 5 | 5 | 5 | 4 | 5 |
| 20 | 5 | 4 | 4 | 5 | 5 | 5 | 5 | 5 | 5 | 4 | 5 | 5 | 5 | 5 | 4 | 5 | 5 | 5 | 4 | 5 | 4 | 4 | 5 |

# 攻读硕士学位期间取得的学术成果

**(1) 发表的主要学术论文**

[1] 徐浩, 兰雨晴, 刘晓辉. The research and implementation of performance test method for Linux process scheduling[J]. 北京航空航天大学第9届学术论坛

[2] 徐浩，兰雨晴. 基于SPICE协议的桌面虚拟化技术研究与改进方案[J]. 计算机工程与科学

# 致谢

在本文即将结束的时候，首先要感谢我的导师兰雨晴老师。兰老师对本文的选题、研究和撰写工作给予了大量的指导和中肯的教诲。兰老师扎实的理论功底、严谨的治学态度和兢兢业业的工作作风都使我受益匪浅。在课题的探讨过程中，兰老师给我提供了很多有建设性的意见和建议，帮助我拓展思路。兰老师不仅在科研学习上对我进行指导，而且在生活上对我关系和照顾，这种家人般的温暖让我的研究生生活多了更多的温馨和感动。兰老师在学业和生活上的指导和关怀，让我对自己有更加清楚的认识，及对未来有更好的规划。感谢兰老师对我的论文工作的悉心指导，让我深切得懂得了做学问来不得一丝马虎。在此我怀着无比感恩的心情，向兰老师致以衷心的感谢。

课题组是一个集体，没有课题组其他同学的大力配合，就不会有我今天的工作和研究成果。感谢实验室的夏庆新师兄、李红娟师姐和韩涛师姐，他们在我论文的研究和撰写过程中都给予了很大的帮助。特别要感谢夏庆新博士，在论文的开题过程中帮助我拓展思路，在课题的研究过程中提出了很多宝贵的意见和建议。李红娟师姐和韩涛师姐在生活上对我关心很多，在我课题研究遇到困难时给我很大的鼓励和动力，在此也要感谢她们。感谢实验室的王龙、和郑晓静同学，大家在毕业设计和开发技术方面的交流讨论使我的思路更加开阔，很多技术难题也迎刃而解，他们的鼓励和支持给了我莫大的帮助。此外，我还要感谢实验室的赵亮、张洸豪、王洋、伯为翰等，是他们让我的研究生活变得丰富多彩。

衷心地感谢默默支持我、理解我的父亲、母亲以及李梦然，是他们无微不至的关怀和不求回报的爱，让我能够更专心的投入到学习及科研过程中。没有他们的支持和鼓励，我很难有今天的成绩。在此，我衷心的祝愿他们身体健康，事事顺心。

最后，向所有评阅本论文的老师致以最真诚的谢意，谢谢各位老师的批评指正。

1. [] 虚拟化与云计算小组.虚拟化与云计算[M].北京:电子工业出版社, 2009:15-20 [↑](#endnote-ref-2)
2. [] K. Ye, X. Jiang, D. Huang, J. Chen, and B. Wang. Live Migration of Multiple Virtual Machines with Resource Reservation in Cloud Computing Environments[C]. IEEE International Conference on Cloud Computing, 2011 [↑](#endnote-ref-3)
3. [] 云计算[EB/OL], http://baike.baidu.com/view/1316082.htm, 2007-12-17 /2012-11-08 [↑](#endnote-ref-4)
4. [] IBM虚拟化与云计算小组.虚拟化与云计算[M].北京:电子工业出版社, 2010: 26-54 [↑](#endnote-ref-5)
5. [] 雷葆华,饶少阳.云计算解码[M].北京:电子工业出版社, 2012:24-31 [↑](#endnote-ref-6)
6. [] 董向军,张恩刚,张沛等.桌面虚拟化技术研究[J].中国信息界, 2010(4):50-52 [↑](#endnote-ref-7)
7. [] 桌面虚拟化[EB/OL], http://baike.baidu.com/view/1434493.htm, 2008-02-25 /2012-2-17 [↑](#endnote-ref-8)
8. [] Citrix Systems.Citrix HDX Technology[EB/OL], http://www.citrix.com/products/xendesk- top / tech-info .html#hdx, 2009-10-02/2012-03-24 [↑](#endnote-ref-9)
9. [] VMware Inc.VMware View Overview[EB/OL], http://www.vmware.com/products/view/ overview.html, 2007-03-18/2012-05-12 [↑](#endnote-ref-10)
10. [] Microsoft.Microsoft Enterprise Desktop Virtualization [EB/OL], http://www.microsoft. com/en-us/windows/enterprise/products-and-technologies/mdop/med-v.aspx, 2008-07-21/ 2012-04-16 [↑](#endnote-ref-11)
11. [] 王淑红，刘晓辉．Microsoft虚拟化应用指南宝典[M].北京:中国铁道出版社, 2009:730-768 [↑](#endnote-ref-12)
12. [] 绿色IT—LVCC联想虚拟云终端系统方案[EB/OL], http://wenku.baidu.com/view/ 0c6259b569 dc5022aaea0098.html, 2012-03-14 /2012-08-10 [↑](#endnote-ref-13)
13. [] 深信服.aDesk桌面云方案[EB/OL],http://sangfor.com.cn/product/adesk/outline.html, 2013-02-15 [↑](#endnote-ref-14)
14. [] 洪亮.浅谈Vebula-Fit协议与其他协议的比较[J],移动通信, 2013.10(10):20 [↑](#endnote-ref-15)
15. [] 岑志松.RDP协议的研究及其应用设计[M].广东:华南理工出版社, 2004:35-52 [↑](#endnote-ref-16)
16. [] Citrix, XenDesktop Resource and Support [EB/OL], http://www.citrix.com/products/xendesktop/resources-and-support.html, 2013-08-13 [↑](#endnote-ref-17)
17. [] 敖青云,蒋文蓉.基于KVM和QEMU的虚拟桌面系统的实现与应用[J].计算机应用与软件, 2012.11(11): 225 [↑](#endnote-ref-18)
18. [] 刘坚,余综. VNC多媒体数据实时传输的研究与实现[J].计算机工程与设计, 2012.07 (07): 198 [↑](#endnote-ref-19)
19. [] Winter D. De, Simoens P., Deboosere L. A hybrid thin-client protocol for multimedia streaming and interactive gaming applications[C], NOSSDAV '06: Proceedings of the 2006 international workshop on Network and operating systems support for digital audio and video,2006-01-01 [↑](#endnote-ref-20)
20. [] Frédéric Fok A.C., Benoit Lécroart, Émilie Chan, et al. An adaptive approach to optimize thin client protocols[C], 2010 Future Network and Mobile Summit, vol. ISSU, 2010 [↑](#endnote-ref-21)
21. [] Tolia N, Andersen D, Satyanarayanan M. Quantifying interactive user experience on thin clients[J]. Computer, 2006, 39(3):46–52 [↑](#endnote-ref-22)
22. [] T. Prichardson, The RFB protocol version 3.8[EB/OL], http://www.realvnc.com/docs/rfb- p roto.pdf , 2010-11-26/2013-03-19 [↑](#endnote-ref-23)
23. []Zinca D. Design of a modified RFB protocol and its implementation in an ultra-thin client[C]. Electronics and Telecommunications (ISETC), 2010 9th International Symposium on, 2010:157-160. [↑](#endnote-ref-24)
24. [] Liao X, Jin H, Hu L,et al. Towards virtualized desktop environment[M]. Concurrency Computation Practice and Experience, 2010, 22(4):419-440 [↑](#endnote-ref-25)
25. [] Redhat . Spice protocol[EB/OL], http: // spice-space.org / docs / spice protocol.pdf, 2013-06-28 [↑](#endnote-ref-26)
26. [] RedHat．VD Interfaces Draft[EB/OL], http://spice-space.org/docs/vd\_interfaces.pdf, 2009/2013-07-15 [↑](#endnote-ref-27)
27. [] SPICE. SPICE user manual[EB/OL], http://www.spice-space.org/docs/spice\_user\_m- anual.pdf, 2012-10-05 [↑](#endnote-ref-28)
28. [] Redhat . Spice for newbies[EB/OL],http: // spice-space.org/docs/spice for newbies.pdf, 2009/2013-01-24 [↑](#endnote-ref-29)
29. [] Starosolski R. Simple Fast and Adaptive Lossless Image Compression Algorithm[J], Software: Practice and Experience, 2007, 37(1):65-91 [↑](#endnote-ref-30)
30. [] Storer J A, Szymanski T G. Data compression via textual substitution[J]. Journal of the ACM (JACM), 1982, 29(4):928-951 [↑](#endnote-ref-31)
31. [] Ziv J, Lempel A. A universal algorithm for sequential data compression[J]. Information Theory, IEEE Transactions on, 1977, 23(3): 337-343 [↑](#endnote-ref-32)
32. []ISO/IEC 15444-1:2002, JPEG 2000 Image encoding system-part I:Core coding system[S]. Information technology. 2002 [↑](#endnote-ref-33)
33. [] ISO/IEC 15444-3:2002, JPEG 2000 Image encoding system-part 3:Motion JPEG2000[S], Information technology, 2002 [↑](#endnote-ref-34)
34. [] 刘岩,张静,张春田.运动图象压缩标准Motion JPEG2000及其应用[J],电视技术,2003, No.2 l6-18 [↑](#endnote-ref-35)
35. [] Deshpande S, Zeng W. HTTP streaming of JPEG2000 images[C]//Information Technology: Coding and Computing, 2001. Proceedings. International Conference on. IEEE, 2001:15-19 [↑](#endnote-ref-36)
36. [] Gormish M J, Lee D, Marcellin M W. JPEG 2000: overview, architecture, and applica- tions[C]//Image Processing, 2000. Proceedings. 2000 International Conference on. IEEE, 2000, 2: 29-32. [↑](#endnote-ref-37)
37. [] 刘方敏,吴永辉,俞建新. JPEG2000图像压缩过程及原理概述[J], 计算机辅助设计与图形学学报, 2002,14(10) [↑](#endnote-ref-38)
38. [] Fukuhara T, Katoh K, Kimura S, et al. Motion-JPEG2000 standardization and target market[C]//Image Processing, 2000. Proceedings. 2000 International Conference on. IEEE, 2000, 2: 57-60. [↑](#endnote-ref-39)
39. [] Chen L, Shashidhar N, Liu Q. Scalable Secure MJPEG Video Streaming[C]//Advanced Information Networking and Applications Workshops (WAINA), 2012 26th International Conference on. IEEE, 2012: 111-115. [↑](#endnote-ref-40)
40. [] Yu W, Qiu R, Fritts J E. Advantages of motion-JPEG2000 in video processing[C] //Electronic Imaging 2002. International Society for Optics and Photonics, 2002: 635-645. [↑](#endnote-ref-41)
41. [] Schulzrinne H. Real time streaming protocol (RTSP)[J]. 1998. [↑](#endnote-ref-42)
42. [] 王小燕.一种高效点播流媒体服务器的设计与实现[J].计算机工程与科学. 2010. 02(02):124-162 [↑](#endnote-ref-43)
43. [] GStreamer[EB/OL], http://gstreamer.freedesktop.org/, 2013-09-24 [↑](#endnote-ref-44)
44. [] 茅炎菲,黄忠东.基于RTSP协议网络监控系统的研究与实现[J].计算机工程与设计. 2011,Vol.32, No.7 [↑](#endnote-ref-45)
45. [] 方群,王敏,吉逸.基于RTSP/RTP的媒体点播服务器的设计与实现[J]．计算机工程与设计,2006,27(1):4-7 [↑](#endnote-ref-46)
46. [] IEEE Standard G-2.1.6. Draft Standard for the measurement of Visual Impairment in Digital Video Using JND Scale[S], P1486. [↑](#endnote-ref-47)
47. [] 李健.视频质量的主客观评估方法研究[D].西安:西安电子科技大学, 2009 [↑](#endnote-ref-48)
48. [] Nieh J, Yang S J, Novik N. Measuring thin-client performance using slow-motion benchmarking[J]. ACM Transactions on Computer Systems (TOCS), 2003, 21(1): 87-115. [↑](#endnote-ref-49)
49. [] Yang S J, Nieh J, Selsky M, et al. The Performance of Remote Display Mechanisms for Thin-Client Computing[C]//USENIX Annual Technical Conference, General Track. 2002: 131-146. [↑](#endnote-ref-50)