**毕业设计（论文）题目： 虚拟网络流量可视化技术研究**

**学 院： 信息与电子学院**

**专 业： 电子科学与技术**

**班 级： 05411202**

**姓 名： 王本亮**

**指导教师： 苏京霞**

目 录

[第1章 绪 论 1](#_Toc169323422)

[1.1 研究背景和意义 1](#_Toc169323423)

[1.2 研究历史和现状 2](#_Toc169323424)

[1.2.1 发展历史 2](#_Toc169323425)

[1.2.2 研究现状 1](#_Toc169323426)

[1.2.3 总结与分析 1](#_Toc169323427)

[1.3 研究内容和结构安排 1](#_Toc169323428)

[第2章 涉及的理论与技术基础 1](#_Toc169323429)

[2.1 引言 1](#_Toc169323430)

[2.2 虚拟网络组成与构建 5](#_Toc169323434)

[2.2.1 虚拟网络的组成 5](#_Toc169323435)

[2.2.2 虚拟网络的路由 6](#_Toc169323436)

[2.3 流量监听技术与实践方法 5](#_Toc169323434)

[2.3.1 流量监听 5](#_Toc169323435)

[2.3.2 实践方法 6](#_Toc169323436)

[2.4 数据存储方法 5](#_Toc169323434)

[2.5 服务器与客户端全双工通信方法 5](#_Toc169323434)

[2.5.1 Websocket技术简介 5](#_Toc169323435)

[2.5.2 Websocket应用场 6](#_Toc169323436)

[2.6 可视化技术理论基础 5](#_Toc169323434)

[2.6.1 动态页面设计方法 5](#_Toc169323435)

[2.6.2 Echart服务框架简介 6](#_Toc169323436)

[2.7 逻辑-视图-模版（MVT）框架 5](#_Toc169323434)

[2.7.1 MVC框架介绍 5](#_Toc169323435)

[2.7.2 MVT框架介绍 6](#_Toc169323436)

[2.7.3 MVT 与MVC的差异与共性 5](#_Toc169323435)

[2.7.4 MVT设计方法应用 6](#_Toc169323436)

[2.8 消息队列功能 5](#_Toc169323434)

[第3章 系统设计 9](#_Toc169323437)

[3.1引言 9](#_Toc169323438)

[3.2 系统总体设计 9](#_Toc169323438)

[3.1.1 技术路线与设计原则 9](#_Toc169323439)

[3.1.2 目标与功能需求 10](#_Toc169323440)

[3.1.3 系统总体结构 12](#_Toc169323441)

[3.3 流量监听模块 13](#_Toc169323442)

[3.2.1流量监听模块功能与性能要求 14](#_Toc169323443)

[3.2.2 技术框架选取 26](#_Toc169323446)

[3.4 通信服务器模块 28](#_Toc169323448)

[3.4.1 通信服务器功能 28](#_Toc169323449)

[3.4.2 通信服务器技术框架 28](#_Toc169323450)

[3.4.3 数据库功能与需求分析 29](#_Toc169323451)

[3.5 数据可视化展示模块 28](#_Toc169323448)

[3.5.1 数据展示要求 28](#_Toc169323449)

[3.5.2 数据展示技术架构 28](#_Toc169323450)

[第4章 系统实现 38](#_Toc169323456)

[4.1主要功能需求说明 38](#_Toc169323457)

[4.1.1 系统工作流程 28](#_Toc169323449)

[3.4.2 流量监听的硬件与软件环境 28](#_Toc169323450)

[3.4.3 流量监听性能指标 28](#_Toc169323449)

[3.4.4 通信服务器功能需求分析 28](#_Toc169323450)

[3.4.5 数据展示模块而功能需求分析 29](#_Toc169323451)

[4.2关键功能模块实现 38](#_Toc169323458)

[4.2.1流量监听模块 38](#_Toc169323459)

[4.2.2数据存储与读取功能模块 38](#_Toc169323460)

[4.2.3 通信服务器模块技术实现 38](#_Toc169323460)

[4.2.4 数据展示模块功能需求分析 38](#_Toc169323460)

[4.2.5 数据展示模块的逻辑实现 38](#_Toc169323460)

[4.3 系统功能与稳定性测试 43](#_Toc169323462)

[4.3.1 系统功能测试 38](#_Toc169323460)

[4.3.2 系统稳定性测试 38](#_Toc169323460)

[4.3.3 测试环境和条件 38](#_Toc169323460)

[4.3.4 评价方法 38](#_Toc169323460)

[4.3.5 结果与分析 38](#_Toc169323460)

[4.4 小结 43](#_Toc169323462)

[第5章 结束语 48](#_Toc169323463)

[5.1 全文总结 43](#_Toc169323462)

[5.2 工作展望 43](#_Toc169323462)

[致 谢 49](#_Toc169323464)

[参考文献 50](#_Toc169323465)

[附 录 51](#_Toc169323466)

**第1章** **引** **言**

## 1.1 研究背景和意义

随着信息产业技术的发展，各类计算机和移动终端的使用越来越普遍，而网络应用是其中最主要的应用。所以，当今社会人们对于网络信息的交流依赖度越来越高，带动了信息数量的增加和传播速度的上升。但与此同时，我们也不得不为此感到担忧，因为伴随着网络的扩张和信息量的增多，必将使网络基础设施负载更高的压力，影响网络性能的稳定性，这些都将会给人们在网络管理、分析和安全方面带来不小的挑战。

因此，网络提供者与相关的管理用户迫切希望能够在不影响网络性能和可靠性的前提下，通过对流量的收集和分析，更好的监测和控制网络。然而，网络流量的捕捉和分析却是个非常复杂的流程，当中涉及到流量的提取、分类和优先等级的分配，然后还要分发至网络管理、分析和安全等有关的工具。长久以来，传统的网络流量分析方法一直是通过大规模添加工具与系统，变更以太网[交换机](http://www.vsharing.com/module/27720.html)的用途，借助镜像端口复制流量，以及通过网络TAP（分路器）分拆流量。对各种用户来说，一个创新的流量可视化方案是他们亟需渴望的。因为这将为企业用户的IT部门和多种监测分析工具带来智能化的可视性，集中监控、简化运作，优[化工](http://www.vsharing.com/industry/1617.html)具性能，节约成本等实质性的好处。

## 1.2 研究历史和现状

### 1.2.1 发展历史

虚拟化技术在 20 世纪 60 年代首次开发，当时是为了对大型机硬件进行分区以提高硬件利用率。30 多年前，IBM 率先实施虚拟化，作为对大型机进行逻辑分区以形成若干独立虚拟机的一种方式。这些分区允许大型机进行“多任务处理”：同时运行多个应用程序和进程。由于当时大型机是十分昂贵的资源，因此设计了虚拟化技术来进行分区，作为一种充分利用投资的方式。在 20 世纪 80 年代和 90 年代，由于客户端-服务器应用程序以及价格低廉的x86 服务器和台式机成就了分布式计算技术，虚拟化实际上已被人们弃用。20 世纪 90 年代 Windows 的广泛使用以及 Linux 作为服务器操作系统的出现奠定了 x86 服务器的行业标准地位。x86 服务器和桌面部署的增长带来了新的 IT 基础架构和运作难题：基础架构利用率低；物理基础架构成本日益攀升；IT 管理成本不断攀升；故障切换和灾难保护不足；最终用户桌面的维护成本高昂。

1999 年，VMware 推出了针对 x86 系统的虚拟化技术，旨在解决上述很多难题，并将 x86 系统转变成通用的共享硬件基础架构，以便使应用程序环境在完全隔离、移动性和操作系统方面有选择的空间。x86 计算机与大型机不同，它在设计上不支持全面虚拟化，因此必须克服难以解决的难题才能在 x86 计算机上开发出虚拟机。在大型机和 PC 中，大多数 CPU 的基本功能都是执行一系列存储的指令（即软件程序）。x86 处理器中有 17 条特定指令在虚拟化时会产生问题，从而导致操作系统显示警告、终止应用程序或直接完全崩溃。因此，这 17 条指令是在 x86 计算机上首次实现虚拟化时的严重障碍。为应对 x86 体系结构中会产生问题的这些指令，VMware开发了一种自适应虚拟化技术。在生成这些指令时此技术会将它们“困住”，然后将它们转换成可以虚拟化的安全指令，同时允许所有其他指令不受干扰地执行。这样就产生了一种与主机硬件匹配并保持软件完全兼容性的高性能[虚拟机](http://www.vmware.com/cn/technology/virtual-machine.html)。

在VMware逐渐在企业级市场中被广泛的接受后，Xen也逐渐在互联网领域崭露头角。在成熟的服务器操作系统当中，Novell SUSE Linux Enterprise 10是第一个采用Xen技术的。当时的Xen还很不成熟，但是在2006年，RHEL 5.0发布的时候，红帽公司决定也将Xen加入到自己的默认特性当中，一时之间，在Linux服务器领域，Xen似乎成为了VMware之外的最佳虚拟化选择。但是，作为一项Linux平台上的虚拟化技术，Xen在很长一段时间内一直没有被接受到Linux内核的代码当中，这对于Xen的维护者而言，不仅意味着要多做很多工作，还意味着用户在废了半天劲装好Xen之后可能遇到意想不到的问题。

2008年9月，红帽收购了一家名叫Qumranet的以色列小公司，由此入手了一个叫做KVM的虚拟化技术（KVM，全称Kernel-based Virtual Machine，意为基于内核的虚拟机）。

2010年，IBM系统软件部市场与销售副总裁，同时也是开放虚拟化联盟委员会成员之一的Inna Kuznetsova公开表示：KVM已经对数据中心做好了准备。

为了进一步降低桌面虚拟化的投入成本，Citrix在2011年5月发布了IntelliCache技术。在过去，VDI项目都是使用共享存储，服务器的本地存储设备被闲置，是一个不小的浪费，IntelliCache技术可以把虚拟机的主镜像文件传送到服务器的硬盘上，使虚拟机对存储的读写由原来的共享存储，转向对服务器本地存储设备的读写，既节省了共享存储的投入成本，又有效利用了现有的投资，是一个双赢的结果。

在虚拟化技术大规模应用于生产之后，随着科学技术的不断发展,计算机网络技术的应用也日益普及.在目前的计算机网络安全工作中,虚拟网络技术是比较重要的一类,并且在很多方面都发挥了较大的积极作用。从成本、占地空间、硬件资源管理等各个方面来看，虚拟网络技术的应用是必然的路径。考虑到今后的计算机网络拥有较大的空间,在应用虚拟网络技术的过程中,需考虑不同的环境和应用标准,以此来实现计算机网络安全的巩固。

然而，在使用方便的同时，软件资源管理的问题也日益突出。

### 1.2.2 研究现状

目前数据可视化的研究主要偏向于统计数据的展示呈现，往往通过新的理念模式或者算法对海量的信息数据进行处理与计算得到统计学结果或者计算出二维乃至多维的数据相关性，再通过图表的形式展现出来。这种可视化技术，注重的是对大数据的分析，注重的是结果，而且可视化技术在该研究中往往属于非主要部分，即展现统计结果。

### 1.2.3 总结分析

目前的可视化技术主要存在三个问题：展示界面不够友好，由于常规的可视化技术更注重根据数据和算法所获得的计算结果的准确性，所以计算结果通常在命令行中用块儿状或者折线状视图展示，展示工作不够美观；实时性不够高，将可视化技术应用在流量分析工作中，对展示结果往往有着较高的实时性要求，传统的计算方法要花费较长的时间进行离线计算；占用过多的系统资源，传统的可视化技术在计算过程中，一方面由于要读取大量的数据，所以有较多的磁盘I/O，占用时间，占用内存，妨碍其他程序在服务器上高校运行，另一方面，由于算法大多为计算密集型程序，所以同样会占用较多的CPU资源，耗电，耗费计算节点的计算资源。

本文研究的虚拟网络流量可视化技术，在由于采用了新的通信协议，降低了服务器资源的损耗；采用消息队列的机制，保障了系统的实时性；使用开源的前端美化框架，使监控结果看起来更加清楚，更加美观。

## 1.3 研究内容和结构安排

### 1.3.1 研究内容

本问介绍了一套高性能、高可靠、高可扩展的虚拟网络流量监控系统。系统分为三个模块，分别为：虚拟网络流量监听及流量处理模块；系统通信服务器模块儿；流量数据可视化展示模块。

其中，虚拟网络流量监听及流量处理模块和系统通信服务器模块的设计架构为C/S架构；流量数据可视化展示模块和系统通信服务器模块的设计架构为B/S架构。两端的S部分为集成的系统通信服务器程序，便于程序的监控管理和后期的功能扩展以及个性化功能的定制。论文的主要研究内容包括以下几个方面：

1. 介绍虚拟网络流量监听常用技术与具体实现的方法，并根据监听到的流量数据提取对应参数，为后期研究提供数据储备和为线上部署做好准备；
2. 介绍Web应用中消息队列的原理与实现方法。使用Redis数据库作为数据存储仓库和消息队列实现的软件基础；从消费者和生产者模型入手，详细介Redis的消息的发布/订阅（Pub/Sub）模式，讲述二者的区别，以及使用Pub/Sub的原因；
3. 介绍Web程序的B/S架构设计模式中客户端与服务端的全双工通信方式即采用WebSocket协议，WebSocket是区别于HTTP的应用层协议；本文详细介绍了基于Tornado框架的WebSocket协议的代码实现过程，并且将消息队列应用于双工通信中；
4. 介绍MVC设计模式的原理，总结MVC设计模式的优点，研究使用Tornado框架来提高系统的稳定性的方法，并研究其在对应开发框架的变化和应用；
5. 简述html5的基本使用，介绍Echart开源服务框架的使用方法，以及个性化定制技术；
6. 基于以上技术实现的网络流量可视化系统的开发过程和其他相关技术的介绍；
7. 将网络流量可视化系统布置在虚拟网络环境下的自动化程序。

### 1.3.2 结构安排

论文共分为5章。第1章是引言部分，主要对虚拟化网络流量可视化技术的研究背景、研究意义做简单介绍；回顾虚拟化技术的发展历史，介绍虚拟化网络中流量可视化技术的研究现状，同时对国内外研究的现状作简要分析。第2章论述理论基础和相关的技术，包括虚拟化网络组建、虚拟化网络流量的监听技术、使用Redis数据库实现的消息队列、WebSocket协议的原理与使用方法、Tornado服务框架的原理与使用、Echart服务框架的使用与个性化定制等。第3章阐述了虚拟化网络流量可视化系统的总体设计的技术路线和设计原则，分析了目标与功能需求，详细描述了流量监听模块、通信服务器模块和数据可视化展示模块的技术架构选取原则与方法。第4章具体展示了虚拟网络流量可视化系统的各个模块的实现细节，并介绍了系统的测试方法，提供了一部分测试所得数据及结果分析。第5章为全文做了总结，并介绍了下一步的工作展望。

**第2章** **涉及的理论与技术基础**

## 2.1 引言

本系统分为三个模块，应用了一组C/S架构，一组B/S架构。涉及了虚拟网络组建、流量监听、NoSQL数据库、WEBSOCKET协议、可视化技术、MVC设计模式、消息队列等理论和技术。

## 2.2 虚拟网络组成与组建

### 2.2.1 虚拟网络的组成

虚拟网络是指在少量性能强大的服务器上才用虚拟化技术，构建起多台虚拟主机，但是其网络组建是各自独立的，虚拟主机在网络中的地位与物理主机并无差异；同构建实体网络一样，在设置好网络ID之后，为各个终端，即虚拟主机分配主机ID。为各个终端设置好IP地址、网关与子关掩码后，只需将虚拟主机所在的宿主机接入我们配置的网络，并在虚拟机管理程序中进行少量配置就可以把虚拟主机接入网络。虚拟主机的使用与物理主机使用并无差异，但是有方便管理、容灾性好、可以实时备份或者恢复等有点，同时，在批量配置虚拟主机时还可以使用镜像直接操作，方便快捷。

### 2.2.2 虚拟网络的路由

关于虚拟网络的路由，从虚拟主机的角度来看，其特性与物理主机并没有差异，其虚拟特性对自身来讲是透明的；但是虚拟主机的所有设备文件，都是基于虚拟化技术虚拟产生的，包括CPU、硬盘、内存等，这里当然包括网卡，但是主机想要进行网络I/O又必须需要物理网卡，所以由虚拟主机的宿主机来完成这一任务。

虚拟主机的网络配置基本分为三种，分别为桥接、NAT、Host-Only，目前市场上主流的虚拟软件，包括VMware Station，Virtual Box,，Virtual PC 以及企业级应用vSphere都支持这三种虚拟网络配置模式。

1. 桥接：桥接网络是指本地物理网卡和虚拟网卡通过VMnet0虚拟交换机进行桥接，物理网卡和虚拟网卡在拓扑图上处于同等地位，那么物理网卡和虚拟网卡就相当于处于同一个网段，虚拟交换机就相当于一台现实网络中的交换机,所以两个网卡的IP地址也要设置为同一网段。所以当我们要在局域网使用虚拟机，对局域网其他pc提供服务时，例如提供ftp、ssh、http等服务，那么就要选择桥接模式。
2. NAT模式：该模式就是让虚拟机借助NAT(网络地址转换)功能，通过宿主机器所在的网络来访问公网。NAT模式中，虚拟机的网卡和物理网卡的网络，不在同一个网络，虚拟机的网卡，是在vmware提供的一个虚拟网络。
3. Host-Only：在Host-Only模式下，虚拟网络是一个全封闭的网络，它唯一能够访问的就是主机。其实Host-Only网络和NAT网络很相似，不同的地方就是Host-Only网络没有NAT服务，所以虚拟网络不能连接到Internet。

本系统组建的虚拟网络的路由模式是才用桥接和NAT来年两种网络模式，拓扑图如图2-1所示：



图2-1 虚拟网络拓扑图

## 2.3 流量监听技术与实践方法

基于以太网的网卡，有两种工作模式。一种模式称为一般模式，即网卡正常工作，在接收数据帧时，网卡只关心与自己有关的数据包，这时候，网卡仅接受数据帧中目的地址与自身网络接口地址相同的数据帧，然后交给操作系统或者相关应用程序进行处理；另外一种为混杂模式，即网卡监听自身所在网络内的所有数据帧，接收所有数据帧，而不管这些数据帧的目的地址是哪里，全部交给自身上层的程序进行处理。

考虑到程序的稳定行，我们将在Linux下部署监听程序，因此系统的开发环境也是Linux。在Linux系统下进行网络流量的监听，首先需要将网卡设为混杂模式，以便于网卡监听自身所在网络内的所有数据帧；以太网上的数据帧监听，主要关心TCP/IP协议簇中的IP协议，ARP协议，以及TCP协议和UDP协议。

传统的数据包捕捉平台主要是基于网络中但节点的操作系统。最早被云公用的是Raw Socket原始套接字，但这种技术只能捕捉IP包，对ARP包无效；BSD Packet Filter（BPF）机制相对来说也是基于FreeBSD下使用较多的捕捉平台。它对数据包的处理简单易行，但是对于高流量数据包的处理表现并不好。一方面，考虑到本系统对实时性的要求较高；另一方面，由于过度依赖UNIX内部的一些机制，频繁地使用系统调用，不便于移植，因此并不合适。美国洛仑兹伯克利国家实验室所编写的专用语数据包截获功能的API函数库Libpcap的设计目标，就是要使得数据包监听程序在不同的操作系统平台上更加容易地进行移植，它有强大的性能和良好的移植性使我们实现流量监听程序的首选，另一方面，很多编程语言都提供了封装了pcap的库，使我们在编程实现程序时有更多的选择[[[1]](#endnote-1)]。

## 2.4 数据存储方法

在获取流量监听节点监听及处理后的数据之后，系统将把数据进行存储和展示，我们的系统使用Redis数据库和MongoDB这两种NoSQL数据库。

首先，我们的系统的目标是准确、实时展示各个计算节点的流量情况，而不关心数据帧的源地址和目的地址的关系，所以我们使用NoSQL数据库MongoDB进行存储数据帧的详细信息，包括数据帧的参数信息、捕获时间、存储时间，由于该数据库有灵活的set和list数据结构，在后期我们可以进行统计学计算，包括指定节点的流量情况，特定的数据帧类型等。这里我们只关心每一条记录，而并不关心各个记录之间的关系，MongoDB有存取速度快、查找方便、并发性能好、灾难恢复容易等诸多有点。

另一方面，我们在进行流量可视化的展示的时候，并不可以把监听节点的数据直接交给前台模块展示，因为考虑到网络I/O，硬盘I/O等问题，展示模块收到的每一条记录很可能不是流量监听节点捕获的流量信息的正确顺序。系统需要使用一个速度快，性能好的中间数据库作为缓冲。这里使用Redis数据库，按照时间顺序接受并存储流量监听节点处理后的数据内容，并按照其存储的先后顺序，将数据发送给通信服务器程序，最终交由前台展示页面将数据信息通过可视化技术展现出来。Redis有队列以及双端队列等数据结构，其底层细节用C语言的字符串数据结构来实现，效率很高，同时又提供了多种查询方法，适合对实时性准确性要求较高的应用。

## 2.5 服务器与客户端全双工通信方法

### 2.5.1 常规通信方法

通常情况下，Web应用都使用Http协议来进行工作，但是http协议是无状态的协议，即服务器每次返回给客户端对应内容之后，就关闭连接。这样以来，客户端就没有办法接受新的数据并展示了。常规的解决办法是在客户端使用ajax技术，即在客户端采取无限的轮询策略向服务端发送ajax请求，异步刷新客户端页面的展示情况。如果轮询间隔较长，那么系统的实时性无法保证；如果轮询时间很短，一方面，客户端浏览器要不断刷新，考虑到网络情况，很可能出现客户端无法展示内容的情况，另一方面，无论服务端有没有新的数据内容需要展示，客户端都会不断向服务器发送请求，服务器也必须处理这些请求，会占用大量的服务器资源，造成极大浪费。

### 2.5.2 WEBSOCKET协议简介

WebSocket是基于HTML5实现的一种新的应用层协议，是目前实现全双工通信的主要方法，webnsocket协议和http协议完全不同。该协议采用的是单socket全双工通信来进行推送或者接收信息，这不仅可以避免轮询的问题，也可以避免在客户端实现的javascript代码在不同浏览器平台上的可移植性问题。在客户端与服务端之间，websocket协议在第一次连接是通过握手实现的，同时它也是基于TCP/IP的底层协议进行实现的。

## 2.6 可视化技术理论基础

### 2.6.1 动态页面设计方法

### 2.6.2 ECHART服务框架简介

## 2.7 逻辑-视图-模版（MVC）设计模式

### 2.7.1 MVC框架介绍

### 2.7.2 MVT框架介绍

### 2.7.3 MVT 与MVC的差异与共性

### 2.7.4 MVT设计方法应用

**第3章** 系统设计

## 2.1 引言

### 2.1.1 发展历史

**第4章** **涉及的理论与技术基础**

## 2.1 引言

### 2.1.1 发展历史

**第5章** **涉及的理论与技术基础**

## 2.1 引言

### 2.1.1 发展历史

1. [] Lnux下的高流量数据包监听技术 [↑](#endnote-ref-1)