**基于WASM和跨平台技术的桌面网络代理系统设计与实现**

**一、选题依据与工程应用价值**

**1.1 选题依据**

网络代理系统是通过拦截计算机的网络请求，将捕获到的请求根据计算机不同层的网络协议解析成不同的报文信息，再进一步对报文信息进行分析和统计并反馈给使用者。在这个网络高度普及的世界，我们身边的每一台终端，无论是电脑、平板还是手机，甚至一些IOT设备都与网络紧密相连，共同构建起宏大的信息社会。而作为程序员，目前电脑还是我们的第一生产力工具，在日常开发中，我们经常需要在本地电脑开发连接一些不同的线上资源环境进行测试、开发、调试等等。而频繁的切换开发环境无疑是一件重复的、低效率的操作，所以通过网络代理工具可以高效率的切换环境，让我们更加专注于内容的开发研究，而不会过多承担环境等因素带来的心智负担。同时在开发过程我们需要经常的调试接口、管理API，而网络代理工具可以自动化的帮助管理开发应用过程中的接口迭代，保障应用的生命周期管理。随着云时代的来临，线上环境更加斑驳繁杂、应用系统的生命周期管理等都会面临更大的挑战，通过网络代理工具可以更加智能化应对这些挑战。

网络代理的目前实现方式主要分为两种：一种是通过直接监听计算机上操作系统的网卡，拦截所有经过网卡的数据帧，再根据网络协议树解析为对应的协议数据包，从而获取到网络数据包在产生和运输过程中的行为；第二种方式是通过启动一个代理服务，再将计算机上的HTTP/HTTPS请求都通过代理服务代理，从而实现代理服务对请求数据包的获取。

关于网络代理的解决方案，大型互联网公司基本都是在原有的一些底层工具上做二次开发，以满足特定的差异化需求。因为常见的网络代理工具有一些缺点，比如功能不够丰富、不支持跨平台、无页面、可操作性差、复杂度高等等。因此通过毕业设计想要实现一个基于NodeJS代理HTTP/HTTPS请求的跨平台、高性能的网络代理系统，通过对代理的请求管理应用开发生命周期中接口的迭代同时通过网络代理提高对不同环境场景下的适应能力，致力于让开发人员能够精确的掌握自己的开发环境，通过代理使用规则转发、修改每一个请求和响应的内容；同时通过对包信息解析反馈给开发人员，提升开发人员的开发效率和编程体验。

**1.2 工程应用价值**

作为软件开发人员，要知道每一次新的软件功能在上线之前，会有本地环境、测试环境、预发布环境、正式环境、分地区环境等等甚至更多。因为有如此多的环境，每个环境又有其差异性和特殊性，连接不同的环境可能要事先在程序中写好硬编码代理或者还要针对不同的环境做一些接口差异化处理。虽然这样的方式保证产品在上线之前的健壮性，提高线上服务的稳定性，但同时也给开发过程带来一定的挑战和困难。同时，在开发一个软件的过程是漫长的，软件接口版本可能在这个过程迭代了很多次，那如何管理软件生命周期过程中接口迭代也是面临的挑战。

对于程序开发人员，网络代理工具能够更加准确的掌控当前的开发环境。开发环境是影响开发效能的最大因素之一，不可用的环境、不稳定的上游环境、线上本地环境的差异等等，都会严重影响开发体验与效率。通过网络代理工具管理应用的API生命周期, 可以帮助我们分析当前程序中的接口依赖程度、接口的稳定性、接口的迭代等等，让我们拥有更好的开发体验。对于运维人员，网络代理工具可以帮助排查线上问题，解决线上问题难查、难以复现的问题。对于普通用户，网络代理工具可以帮助你可视化你的浏览历史，屏蔽非法请求。

网络代理工具能够给开发人员带来什么样的开发体验？

(1) 稳定性

         好的开发环境首先要应该是要稳定可用，不应该在开发测试的过程中因为环境的问题，频繁的切换上下文而分散开发重心。通常依赖于接口或者底层服务开发，最大的心智负担就是底层服务、接口变更导致开发终止。而网络代理工具可以快速切换依赖的环境，从而减少依赖环境带来的不稳定因素，让开发人员更加专注于核心业务的开发。

1. 快速验证

因为本地开发环境和目标环境的不一致，我们经常需要判断当前开发的功能是否在目标环境也可以正常运行，修改代码能够在尽可能短的时间内得到验证也是一个基本诉求，如果每次修改代码，都要经过一遍公司内部复杂的流程才能在线上得以验证，那对开发人员体验会大大打击。同时当依赖的接口突然不可用，我们也可以通过代理工具Mock数据，降低影响，不会带来开发的中断感。当想要测试一个线上页面改动可能会带来的后果，代理工具可以不经过复杂的发布过程快速在本地验证功能的正确性。

1. 环境一致性

随着云时代的到来，项目的线上环境是极为复杂的再加上大型应用的后端服务依赖也是多样的，网络代理工具通过代理获取到线上环境的真实数据对我们本地程序进行验证。

(4) 确定性

代理工具通过对数据包信息的解析，可以提供给开发人员更多的信息，帮助开发人员定位实际的问题。因为很多程序复杂度高，又有一些本地缓存、DNS缓存、服务器缓存甚至是云缓存等等，会给定位问题带来巨大的挑战。所以需要明确到底有什么生效、是否是缓存导致等等不确定因素。

**二、国内外研究现状与发展动态**

本小节将从桌面跨平台技术、WebAssembly技术、网络代理技术三个方面分别介绍其相关的发展动态。

**2.1 桌面应用跨平台的研究**

桌面应用开发是一类在软件开发最早开始的一种开发类别。桌面应用开发需要注意的一些地方，用户本地数据存储、应用本地配置、系统文件权限、系统托盘、剪贴板操作、系统软（环境变量）硬件（处理器内存显卡显示器分辨率等）环境、注册表、可执行文件、安装包制作、卸载清理处理，桌面应用道德规范、版本迭代，强制更新考虑、平滑升级、基础建设（撤销/重做，滚动条等）、 快速响应/无网络运行 等等。

桌面应用开发因为要依赖于不同的操作系统，所以不同的操作系统上都有自己相对应的开发语言和开发框架。因为操作系统市场份额占比原因，windows操作系统几乎是占比一大半，其次是MacOS操作系统，再就是类Linux的操作系统。但是Linux操作系统大多用在服务器端，所以对桌面应用的需求也不高。所以目前桌面应用开发以Windows和MacOS为主，很多开发规范和开发框架都是在这两个基础上迭代更新，如图一所示：

|  |  |  |
| --- | --- | --- |
| 框架/类库 | 语言/环境 | 适用平台 |
| QT | C/C++ | 跨平台 |
| WPF | .NET/C#/VB | Windows |
| WinForm | .NET/C# | Windows |
| MFC | C/C++ | Windows |
| Swift | Objective-C | Mac |
| Swing/JavaFx/SWT | Java | 跨平台 |
| NW.JS | JavaScript | 跨平台 |
| Electron | JavaScript | 跨平台 |

图一 桌面应用开发规范

通过上图，将桌面应用开发简单分为三个方面：

1. 直接将语言编译成可执行文件，直接调用系统API，完成UI绘制等。从不跨平台的角度，在windows平台上主要是C#，从MFC方案到如今的WPF和WinForm，如果没有跨平台的诉求，那么WPF无疑是最好的解决方案。MacOS上则是以Objective-c或Swift cocoa为主，可以方便调用底层的 API，缺点是不跨平台，文档不友好，UI 库并不丰富。

这类开发技术，有着较高的运行效率，但一般来说，开发速度较慢。

1. 基于Qt、javaFx

从跨平台的角度，则分为Qt、java类、web类。首先Qt支持跨平台，非常强大，但是学习成本较大，迭代效率不高，涉及到协议、qml等，开发周期会非常长。这是很多客户端跨平台的首选，因为开源、UI 库和各种功能的类库非常丰富。C++ duilib，这是 windows 下开源的 directUI（微软提出的分离 UI 和逻辑的思想）库，它是迎合互联网桌面软件小而美的趋势发展起来。

Java swing/javaFx，这是一类比较大的阵营，优势是跨平台和流行开发语言 Java 的天然结合。但是生态较差，相关资料不够。并且javafx中需要jvm内存占用问题，系统原生功能支持不够。其次java类的技术感觉是比较落后、目前生态较差。

第三、基于Web的跨平台应用系统研究，从NW.JS到Electron, Electron是2014年正式开源垮桌面应用技术

而我们重点说下web解决方案，当下的桌面软件项目，必须具备完整的开发、部署、运行、测试等，尤其是一些大型软件有大量的增删查改表单等，这部分如果使用Web作为解决，是非常容易且轻松的，而且美观性和易用性都非常高。

为了提供原生系统的GUI支持，Electron内置了原生应用程序接口，对调用一些系统功能，如调用系统通知、打开系统文件夹提供支持。

在开发模式上，Electron在调用系统API和绘制界面上是分离开发的。Electron是一个基于Web构建桌面应用程序的底层工具框架。 它允许使用 Node.js 和Chromium 完成桌面 GUI 应用程序的开发。通过嵌入Chromium 和 Node.js 到二进制的 Electron 可以构建跨平台桌面程序(Mac、Windows、Linux)。而Chromium是Google公司基于V8引擎内核库。

Electron框架的优势，可以提升用户体验效果，并且更新速度快，压缩迭代周期。

* 将 Chromium 与 Node.js 集成到同一个运行时环境中
* 不必考虑兼容性问题，只需关注 Chromium 版本即可
* 使用 Node 强大的API，可以使web页面操作文件，调用系统API，操作数据库…
* 也不必考虑跨域问题

同样，相比原生C++等开发的桌面应用，基于Electron开发的应用系统性能上肯定是比原生应用要低，同时打包后的体积会更大。相比较Qt、javaFx来说，Web跨平台带来的是更丰富的组件、更快的开发周期、更加灵活的技术架构。

当然这两个缺点也是跨平台的开发体验所带来的，所以我们要通过C++/Rust来编译WebAssembly这种高性能编码来提高性能、减少体积，从而优化这两个缺点，也是毕业设计需要重点研究和学习的地方。

**2.2 高性能编码WebAssembly的研究**

WebAssembly(简称wasm) 是一个可移植、体积小、加载快并且兼容 Web 的全新格式，是一种新的编码方式，可以在现代的网络浏览器中运行。它是一种低级的类汇编语言，具有紧凑的二进制格式，可以接近原生的性能运行，并为诸如C / C ++ / Rust 等语言提供一个编译目标，以便它们可以在js runtime上运行。它也被设计为可以与JavaScript共存，允许两者一起工作。

体积小：由于浏览器运行时只加载编译成的字节码，一样的逻辑使用更高级解释型语言文件体积小很多。

加载快：由于文件体积小，再加上无需解释执行，WebAssembly能更快的加载并实例化，并减少运行前的等待时间。

兼容问题少：WebAssembly是非常底层的字节码规范，制定好以后很少变动，就算发生变化，只需要从高级语言编译成字节码过程做兼容。

一些其他抽象概念：

1. (1) 模块：表示一个已经被编译为可执行机器码的WebAssembly二进制代码。一个模块是无状态的，并且像一个二进制大对象（Blob）一样能够被缓存到IndexedDB中或者在windows和workers之间进行共享。
2. (2) 内存：ArrayBuffer，大小可变。本质上是连续的字节数组，WebAssembly的低级内存存取指令可以对它进行读写操作。
3. (3) 表格：带类型数组，大小可变。表格中的项存储了不能作为原始字节存储在内存里的对象的引用（为了安全和可移植性的原因）

**2.3 对于网络数据包捕获的研究**

第一种直接捕获网卡的数据包

数据包捕获的基本原理

以太网是一种总线型的网络, 总线的特点是:当一台计算机发送数据时, 总线上所有计算机都能检测到这个数据。从逻辑上来看是由一条总线和多个连接在总线上的节点所组成, 各个节点采用CSMA/CD协议进行信道的征用和共享, 每个节点通过网卡来实现这种功能。由于采用以太网广播信道争用的方式，使得监听系统与正常通信的网络能够并联连接，并可以捕获任何一个在同一冲突域上传输的数据包。IEEE802.3标准的以太网采用的是持续CSMA的方式，正是由于以太网采用这种广播信道争用的方式，使得各个站点可以获得其他站点发送的数据。

网卡具有如下的几种工作模式:

(1) 广播模式:该模式下的网卡能够接收到网络中的广播信息。

(2) 组播模式:该模式下的网卡能够接收组播数据。

(3) 直接模式:工作在直接模式下的网卡只能接收目的地址是自己Mac地址的帧。

(4) 混杂模式:工作在混杂模式下的网卡接收所有的流过网卡的帧, 数据包捕获程序就是在这种模式下运行的。

通常, 网卡的缺省工作模式包含广播模式和直接模式。因此, 为了捕获到所有经过该网卡的数据包就必须先将网卡设置成混杂模式。这样, 就可以捕获到流经该网卡的所有数据包, 实现对网络进行监测的目的。

网卡主要的工作是完成对于总线当前状态的探测，确定是否进行数据的传送，判断每个物理数据帧目的地是否为本站地址，如果不匹配，则说明不是发送到本站的而将它丢弃。如果是的话，接收该数据帧，进行物理数据帧的CRC校验，然后将数据帧提交给LLC子层。

第二种，就是基于代理的网络数据捕获

         因此，本论文将基于以上三个方面的前沿技术，结合作为开发者的需求体验，基于Electron、WebAssembly、Whistle以及其他相关技术，实现适当可行的网络代理系统，使其可以在多平台操作系统上进行正常使用

**nodejs因为是异步模型会有较高的吞度量，适合做IO服务器。**

1、采用事件驱动、异步编程，为网络服务而设计。其实Javascript的匿名函数和闭包特性非常适合事件驱动、异步编程。而且JavaScript也简单易学，很多前端设计人员可以很快上手做后端设计。

2、Node.js非阻塞模式的IO处理给Node.js带来在相对低系统资源耗用下的高性能与出众的负载能力，非常适合用作依赖其它IO资源的中间层服务。3、Node.js轻量高效，可以认为是数据密集型分布式部署环境下的实时应用系统的完美解决方案。Node非常适合如下情况：在响应客户端之前，您预计可能有很高的流量，但所需的服务器端逻辑和处理不一定很多。

而网络代理恰恰符合这些要求，因为计算机上网络连接是很多的，同一时刻有很多端口都在和网络交互。

         网络代理相关的工具、应用有很多，比如WireShark、Whistle、Charles、Postman还有国内互联网公司基于一些基础工具做的二次开发，为了更好的满足公司内部的需求。

 其实whistle是基于 Node 实现的跨平台调试代理工具，主要用于查看、修改 HTTP、HTTPS、Websocket 的请求、响应，也可以作为 HTTP 代理服务器使用。whistle采用的是类似配置系统hosts的方式，一切操作都可以通过配置实现，支持域名、路径、正则表达式、通配符、通配路径等多种匹配方式.Whistle 不是入侵侦测系统（Instrusion Detection System)。Whistle不会产生警报或提示网络上的异常流量行为。但是，仔细分析Whistle捕获的数据包可以帮助用户更清楚地了解网络的行为。在开发移动端网页的时候，我们不仅需要抓包，更想要真机能够直接访问到本地 server。

**三、研究内容以及研究目标**

**3.1 研究目标**

本论文将基于WebAssembly、Node proxy以及Electron架构为技术基石，对网络代理应用系统做设计与实现。主要有三大目标：

第一目标，功能目标，实现对网络数据包的捕获、对HTTP/HTTPS协议请求的代理、数据包解析、应用接口的迭代管理等主要功能模块。帮助开发人员快速切换依赖开发环境、接口数据Mock以及应用系统的生命周期管理。

第二目标，跨平台目标，针对不同的操作系统，可以正常使用系统功能。同时针对不同的操作系统底层接口和功能做差异化的编码方案，统一不同操作系统的使用体验。

第三目标，优化目标，针对CPU密集型计算，尤其是可视化部分、网络请求捕获分析模块的实现可以通过WebAssembly高性能编码来优化性能。

**3.2 研究内容**

主要的研究内容分为以下几个方面，

1. 首先是网络代理系统的需求分析，通过需求分析明确网络代理系统的基本功能和使用场景，明确开发过程中需要用到的技术。
2. 网络代理系统的设计，根据需求分析进行设计。主要是根据所掌握的技术，针对不同的技术场景合理规划。尤其是性能部分，使用相应的优化手段进行设计。
3. 网络代理系统的实现，根据设计的架构图、功能需求图等设计，进一步具体实现系统。
4. 网络代理系统的测试，对实现的系统进行功能性测试和非功能性测试，同时对测试结果进行评估。

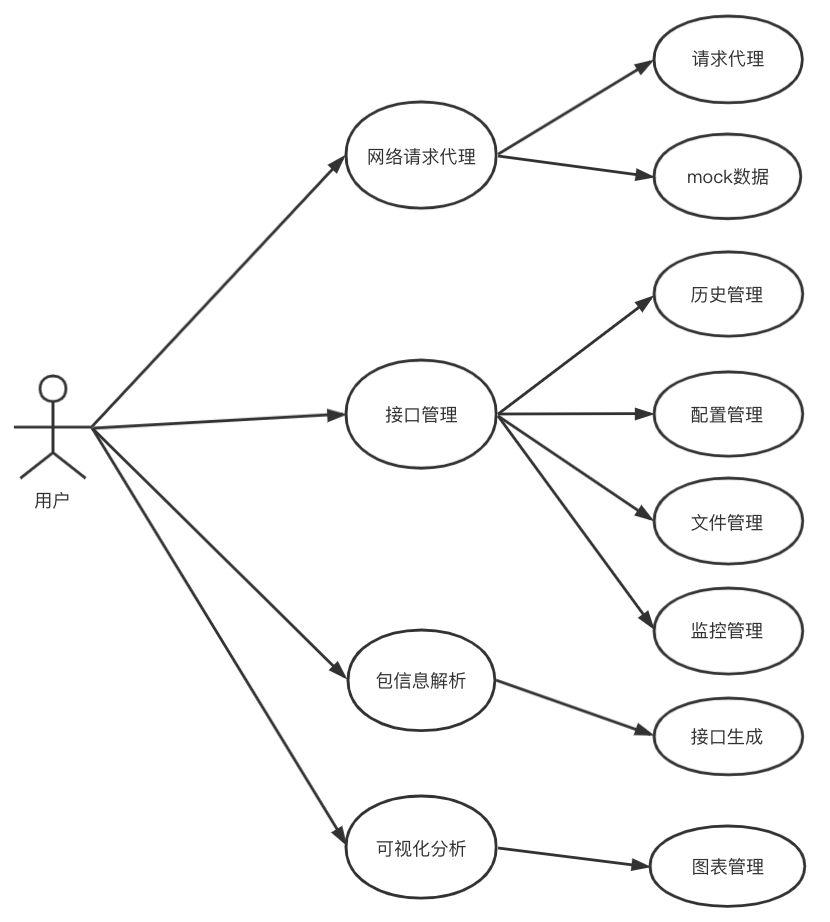
*3.2.1* 需求分析

         首先，明确系统主要使用目标人群以及使用场景，主要是针对程序开发人员或者有一定网络代理需求的用户开发的一个工具类桌面系统应用，这个系统要满足跨平台、高性能、网络信息流捕获等等要求，基于此目标，我们将需求分为功能性需求和非功能性需求。

(1) 功能需求

分为四个主要方面网络请求代理、接口管理、包信息解析、应用生命周期管理

用例图



以上是整个系统的用例图，可以拆分为四个模块

(1) 用例名称：网络请求代理用例

参与者：用户

说明：用户通过规则设置对特定请求进行请求头、mock数据代理

前置条件：用户计算机系统成功设置代理服务

后置条件：请求按照我们的规则生效

(2) 用例名称：接口管理用例

参与者：用户

说明：用户通过设置请求的相关信息，成功在系统中请求，并对请求进行管理

前置条件：用户计算机系统成功设置代理服务

后置条件：请求正常工作并分类管理

(3) 用例名称：包信息解析用例

参与者：用户

说明：用户通过对捕获的网络请求包进行分析，解析并生成请求代码

前置条件：用户计算机系统成功设置代理服务并且对用户请求成功捕获

后置条件：请求按照我们的规则生效

(4) 用例名称：应用生命周期管理用例

参与者：用户

说明：用户再一段时间内所设置的特定接口进行数据管理

前置条件：用户计算机系统成功设置代理服务并且对用户请求成功捕获

后置条件：数据以时间为主线，多维度展开

(2) 非功能需求

1、跨平台

指可以在常见的不同种类的操作系统上，通过一套代码编译后的程序，都可以满足基本的使用。我们主要以适配Windows操作系统和MacOS操作系统为主，因为这两类操作系统占有率已经高达90%以上。

|  |  |
| --- | --- |
| 操作系统 | 占有率 |
| Windows | 74.27% |
| MacOS | 16.05% |
| Linux | 2.09% |

2、高性能

高性能主要是对比C++/Rust编译为WebAssembly相比传统的JavaScript的性能提升。以及Chromium V8引擎对运行时的优化、WebGL带来的渲染性能。

3、安全性

遵循桌面应用开发道德底线，不破坏侵害目标操作系统，同时对自身安全、内存保证隔离安全。

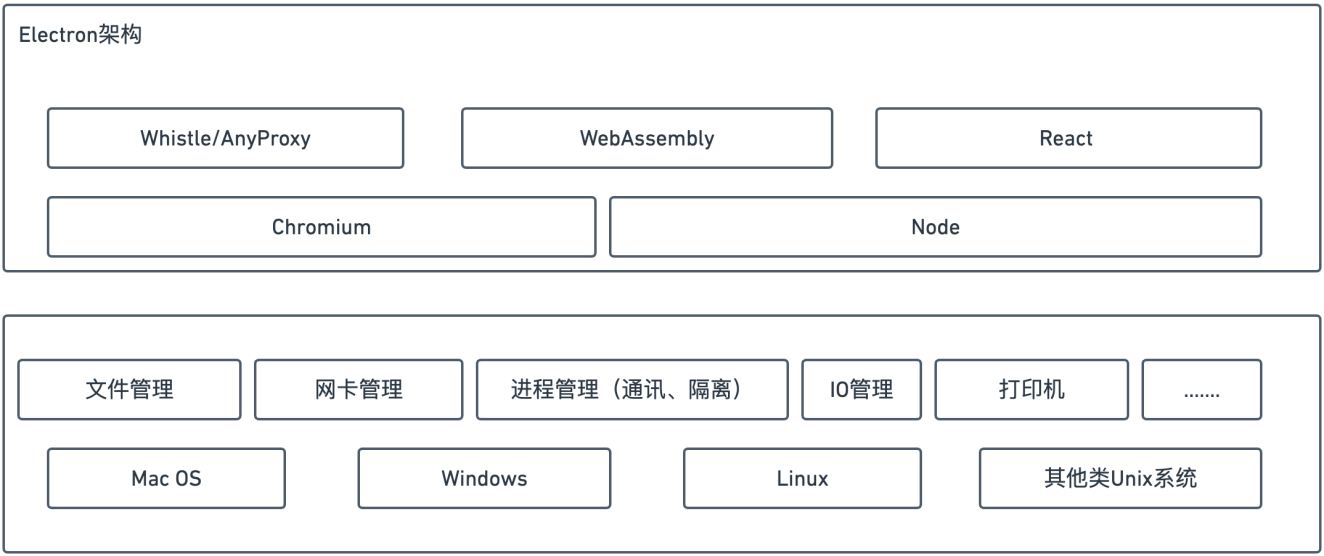
4、易恢复

作为桌面应用，要具备易恢复、重启动、可强制杀死进程等等

*3.2.2* 系统设计

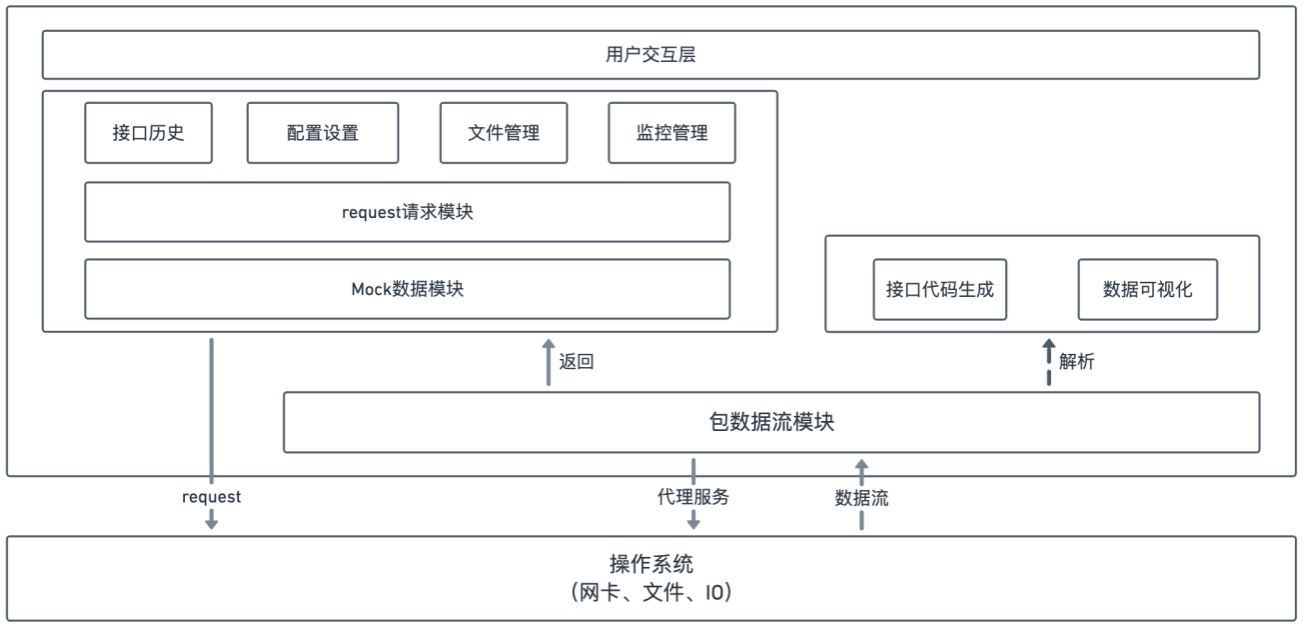
系统设计的主要工作以需求为引导、根据实际情况对系统整体进行架构分层设计，定义层与层之间的交互，各个模块之间的协同性以及我们所要采用的技术选型和架构选型。系统设计的目的是帮我们从整体了解系统的结构，保障得到一个可以使用、满足需求、技术可行、稳定可靠的系统框架。为下一步系统实现打下坚实的理论依据。

(1) 系统架构设计



(2) 功能模块设计

以需求分析部分的功能需求为主，形成对系统整体的功能模块图



(3) 性能需求设计

*3.2.3* 系统实现

完成系统的设计后，我们利用JavaScript代替原来的GUI开发桌面应用。对于CPU密集型计算性能消耗，比如大量的数据包捕获和解析、渲染图像，通过使用C++或Rust编程语言编译成二进制码和WebGL渲染引擎来优化来进行开发优化。最后根据系统架构设计和功能模块设计，将业务需求和功能模块串联起来，实现网络代理系统的功能。

*3.2.4* 系统测试

在完成系统开发后，我们进行系统的功能性测试和非功能性测试。

对于功能测试，需要在不同场景下验证系统的功能，主要使用灰盒测试、自动化测试对系统中的模块进行联合测试和单独测试。我们采用一个复杂的线上开发应用的api作为测试目标，查看是否可以对API进行捕获、环境切换、数据包Mock等。

对于非功能测试，系统需要满足跨平台必须满足在*Window*、*Mac*、*Linux*实现可下载安装、可正常使用大部分功能。高性能需要对比同等条件下，与不使用*WASM*编码方式做横向比较。

**四、实施方案与可行性分析**

**4.1 研究基础**

桌面跨平台技术是追求开发一致性以及抹平底层差异，解放程序员生产力的重要技术。WASM编码技术让缩小了JavaScript语言本身带来的性能问题，WebGL渲染引擎提供了高效的图像渲染能力。网络代理工具更是保障程序员在多线上环境、本地开发的情况下，提升开发体验，提高效率，保障应用的生命周期。本论文要实现的桌面应用系统的功能需求和非功能需求，功能需求包括：网络请求代理、接口管理、包信息解析、应用生命周期管理；非功能需求主要包括：实现跨平台和高性能编码和WebGL提高性能满足大量网络请求捕获和可视化图像渲染。

**4.2 主要技术方案**

*4.2.1 Electron框架*

*1.1 Electron架构*

Electron结合了Chromium、Node.js和用于调用操作系统本地功能的API，能够使用纯JavaScript来创建桌面应用。

(1)Node.Js。

Node.js是一个JavaScript运行环境，是对Google V8引擎进行了封装，用于方便地搭建响应速度快、易于扩展的网络应用。Node.js使用事件驱动、非阻塞I/O模型而得以轻量和高效，非常适合在分布式环境中运行数据密集时实时应用。

(2）系统API。

为了提供原生系统的GUI支持，Electron内置了原生应用程序接口，对调用一些系统功能，如调用系统通知、打开系统文件夹、访问操作系统剪贴板等提供支持。

(3）进程类别。

Electron区分了两种进程：主进程和渲染进程。

一个Electron应用总是有且只有一个主进程，主进程职责：

1）创建渲染进程（可多个）；

2）控制应用生命周期（启动、退出APP以及对APP做一些事件监听）；

3）调用系统底层功能，调用原生资源。

渲染进程相当于一个浏览器窗口，其主要职责：

1）用HTML和CSS渲染界面；

2）用JavaScript做一些界面交互。

1.2 Electron技术优势

桌面应用利用了Web技术的丰富性并通过其他V8引擎等保障运行时的效率和性能。可以即时启动，不需要等待资源从网络下载下来。可以访问计算机的操作系统和硬件资源，包括可以读写用户计算机中的文件系统。可以更好地控制软件的用户体验，不需要担心兼容性问题。可以用Web前端技术开发跨平台的桌面应用：使用纯JavaScript语言开发，只需要写一份代码，打包出来的应用可以同时在Windows、Linux、Mac操作系统上运行。可以从Node.js的生态获得极大的助力：Node.js这个大生态下很多成熟模块可以直接引入使用，避免重复造轮子，提高开发效率。进程隔离：基于Chromium多进程模式的应用模块集成，天然提供了应用模块之间的隔离性，其中某一应用模块的死机故障不影响其他应用模块及整个应用软件。

4.2.2 WASM

WebAssembly是可移植、体积小、加载快并且兼容Web的一种新的字节码格式，主流浏览器都已经支持WebAssembly。WebAssembly字节码和底层机器码很相似，可快速装载运行，因此性能相对于JS解释执行大大提升。WebAssembly作为一份字节码标准，需要用高级编程语言编译出字节码放到WebAssembly虚拟机中才能运行，浏览器厂商根据WebAssembly规范实现虚拟机。

对于网络平台而言，WebAssembly具有巨大的意义——它提供了一条途径，使得以各种语言编写的代码都可以以接近原生的速度在Web中运行。在这种情况下，以前无法以此方式运行的客户端软件都将可以在Web中运行。而且，这是W3C WebAssembly Community Group开发的一项网络标准。

以下主要阐述基于WebAssembly的边缘计算架构的技术特性。

* 跨平台性

WebAssembly字节码是一种抹平了不同CPU(central processing unit，中央处理器）架构的机器码，WebAssembly字节码不能直接在任何一种CPU架构上运行，但由于非常接近机器码，可以非常快地被翻译为对应架构的机器码，因此WebAssembly运行速度和机器码接近。

WebAssembly有如下优点。

体积小：由于浏览器运行时只加载编译成的字节码，同样的逻辑比用字符串描述的JS文件体积要小很多。

加载快：由于文件体积小，再加上无需解释执行，WebAssembly能更快地加载并实例化，减少运行前的等待时间。

兼容性问题少：WebAssembly是非常底层的字节码规范，制订好后很少变动，就算以后发生变化，也只需在从高级语言编译成字节码的过程中做兼容。可能出现兼容性问题的地方在于JS和WebAssembly桥接的JS接口。

每个高级语言源码到不同平台的机器码的转换工作是重复的，高级语言只需要生成底层虚拟机（low level virtual machine,LLVM）认识的中间语言（LLVM intermediate representation,LLVM IR),LLVM能实现LLVM IR到不同CPU架构机器码的生成、机器码编译时性能和大小的优化。除此之外，LLVM还能实现LLVM IR到Web Assembly字节码的编译功能，也就是说只要高级语言能转换成LLVM IR，就能被编译成Web Assembly字节码。

通常负责把高级语言翻译成LLVM IR的部分叫作编译器前端，把LLVM IR编译成各架构CPU对应机器码的部分叫作编译器后端；现在越来越多的高级编程语言选择LLVM作为后端，高级语言只需专注于提供开发效率更高的语法，同时保持翻译到LLVM IR程序的执行性能。

* 性能

WebAssembly适合用于需要大量计算的场景，例如以下场景。

在浏览器中处理音视频，flv.js用Web Assembly重写后性能会有很大提升。

React的dom diff中涉及大量计算，用WebAssembly重写React核心模块能提升性能。Safari浏览器使用的JS引擎JavaScriptCore也已经支持WebAssembly,RN应用性能也能提升。

4.2.3 基于node的http代理

基于Node实现的跨平台web调试代理工具，类似的工具有Windows平台上的Fiddler，主要用于查看、修改HTTP、HTTPS、Websocket的请求、响应，也可以作为HTTP代理服务器使用，不同于Fiddler通过断点修改请求响应的方式，whistle采用的是类似配置系统hosts的方式，一切操作都可以通过配置实现，支持域名、路径、正则表达式、通配符、通配路径等多种匹配方式，且可以通过Node模块扩展功能。

**4.3 可行性分析**

自身掌握开发中所需要的的编程语言JavaScript和C++，对跨平台设计有一定了解，前期通过阅读文献和在公司实际开发实习积累相关的经验，熟悉了在开发过程中使用的React、Whistle、WebAssembly、Electron等等一些基本框架和技术，初步分析表明本方案具备可行性。校内外导师对相关领域具备专业能力，对论文撰写具有指导学生的经验。

**五、参考文献**

 [1] 罗青林,徐克付,臧文羽,刘金刚.Wireshark环境下的网络协议解析与验证方法[J].计算机工程与设计,2011,32(03):770-773.DOI:10.16208/j.issn1000-7024.2011.03.068.

[2] Andreas Haas,Andreas Rossberg,Derek L. Schuff,Ben L. Titzer,Michael Holman,Dan Gohman,Luke Wagner,Alon Zakai,JF Bastien. Bringing the web up to speed with WebAssembly[P]. Programming Language Design and Implementation,2017.

[3] Toman Zinah Hussein,Toman Sarah Hussein,Hazar Manar Joundy. An In-Depth Comparison Of Software Frameworks For Developing Desktop Applications Using Web Technologies[J]. Journal of Southwest Jiaotong University,2019,54(4):

[4] 薛超. 基于WebAssembly的JavaScript性能优化方案研究与实现[D].西北大学,2019.