中国研究生操作系统开源创新大赛

项目功能说明书

学　　校：	电子科技大学
参 赛 队 伍：	MobiNetS
队 伍 成 员：	马浩天 李嘉诚 王振华
指 导 教 师：	赵志为
完 成 日 期：	2024年8月

目录

第 1 章 绪 论

1.1 目的与意义

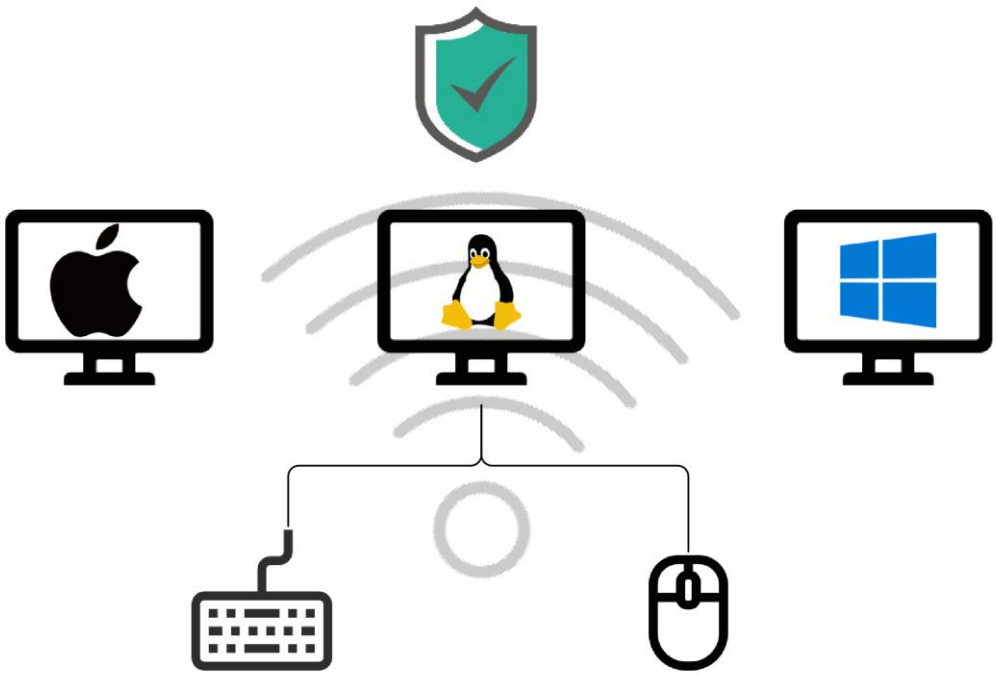


图 1 DeviceShare 项目示意图

在信息化时代，每个人可能会有不止一台计算机主机，而在操作计算机时鼠标、键盘等 Hid Input 设备又是必不可少的，如果为每一台主机都配备一套键鼠，会造成很大的资源浪费。如何使用一套键鼠方便地在这些设备实现跨平台协同，就成为了我们提高设备使用效率必须要面对的问题。为此，我们希望设计实现一套通用性强的多主机 Hid Input 设备共享方案，可以**不依赖任何硬件的前提下**仅通过软件实现一套键盘和鼠标在**多设备、多系统**上的无缝切换和共享。

该作品旨在解决现有设备共享方案中通用性，跨平台性，安全性，硬件依赖，使用难度等方面的问题。设计一套**跨平台、可扩展、高性能的异构主机间设备共享及协同智能解决方案，**仅需一套键鼠，用户就可以在多台主机之间流畅地切换，可以节省硬件成本的同时，极大地提高工作效率。此外该方案可支持 **Windows、Linux（支持 openKylin 及 Wayland 桌面环境）、Mac OS** 等主流操作系统，能提高不同操作系统间的协同能力。在桌面办公、软件开发管理、机房运维、会议演示、家庭娱乐等场景拥有广阔的应用前景。

表 1 作品功能完成情况说明

功能说明	完成情况
支持一套键鼠控制多台主机，同一时刻仅有一套设备响应。	✔
鼠标可跨设备移动、单击、双击、滚轮滑动、拖动。	✔
键盘支持基本跨设备输入，支持**组合快捷键**，支持**跨系统键位映射**。	✔
支持主机间剪切板**安全共享**。	✔
支持多主机间文件共享。	✔
支持以图形化的方式配置主机之间相对位置的功能，鼠标移动范围及方向受相对位置限制。	✔
高扩展性，理论支持主机连接数量无上限, 支持**多键鼠**控制	✔
具备良好的图形化界面及**桌面通知**机制。	✔
具备**自动主机发现**机制，支持**超时断连机制**。	✔
采用公私钥认证和加密机制，保证**安全性**。	✔
优秀的**跨平台**性能，支持 Linux（包括 openKylin），Windows，MacOS 操作系统。	✔
设备共享**延迟低**，可满足正常需求。	✔

项目开源地址：https://www.gitlink.org.cn/mahaotian/DeviceShare

项目演示 Demo 视频地址：https://img.qylh.xyz/mobinets_demo.mp4

1.2 项目背景

1.2.1 技术路线调研

通过前期调研发现目前已存在一些利用硬件和软件方法实现设备共享的解决方案，可按照技术路线分为远程桌面连接、硬件设备共享器、KVM 技术等：

1. 远程桌面

远程桌面是最常见的实现设备共享的方式。目前已有远程桌面协议（RDP），通过该协议，计算机可以利用网络通信实现远程桌面控制，使用户以可视化的方式浏览和控制远程计算机。(Microsoft, 2023)除此之外，还有一些可以实现远程控制的第三方软件，如 TeamViewer、AnyDesk、ToDesk 等，该类远程桌面方式往往适用于物理距离较远的多台主机，对通信带宽有很高的要求。

2. 硬件设备共享器

除了软件方式，也有一些硬件技术可以满足输入设备的共享，例如USB共享器，这一技术允许多台计算机共享一个 USB 设备（如键盘、鼠标、打印机等），通过按钮或自动检测切换控制，目前已有成熟的产品实现这一功能，如 Ugreen USB 3.0 Sharing Switch，其支持两台电脑共享一个 USB 设备，通过物理按钮快速切换 USB 设备的控制权。(Ugreen, 2023)但是该技术只允许两台计算机之间的设备共享，使得其应用受到限制，并且设备的切换只能通过硬件方式实现，无法完成主机的自由切换。

3. KVM 技术

KVM 切换器，一般简称 KVM，可以使用户通过一组键盘、显示器和鼠标控制多台电脑。此技术按照具体实现原理可以分为以下三种：

1) 硬件 KVM 切换器

图 2 硬件 KVM

作为一种硬件设备，硬件 KVM 切换器可以通过一组键盘、显示器和鼠标来控制多台计算机。用户可以通过物理按钮、键盘快捷键或远程管理来切换控制不同的计算机，相比 USB 共享器，其切换器可以支持更多的接口，包括 HDMI、DVA 等。目前 ATEN、Belkin、StarTech 等厂家提供了这种切换器设备，产品包括 ATEN CS1922（支持 2 台电脑，支持 HDMI、USB 接口）(ATEN, 2024)、Belkin SOHO F1DS104L（支持 4 台电脑，支持 DVI、USB 接口）等(Belkin, 2024)。

2) 软件 KVM

软件 KVM 技术利用网络来共享键盘和鼠标，允许用户通过网络连接来控制多台计算机，用户一般在需要共享设备的计算机上安装软件客户端，使用网络连接而不是硬件切换器实现设备共享。

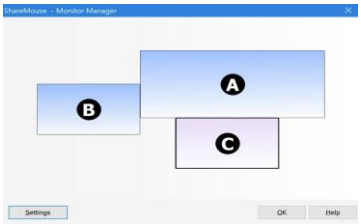
表 2 技术路线对比

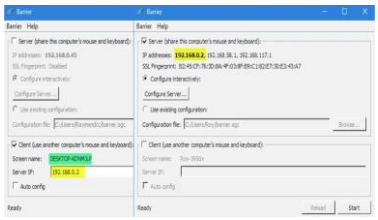
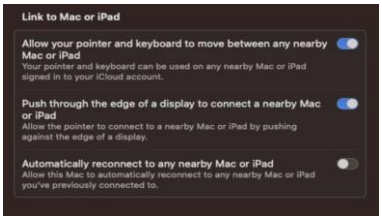
技术路线	优点	缺点
远程桌面	✧ 支持屏幕共享 ✧ 不受物理距离限制	■ 依赖网络带宽，延迟 ■ 对机器性能要求高
	✧ 性能优秀 ✧ 延迟低	■ 需额外硬件支持 ■ 受物理位置限制

硬件设备共享器		■ 切换需手动操作 ■ 无法共享剪切板 ■ 无法实现文件共享
硬件 KVM	✧ 性能优秀 ✧ 延迟低 ✧ 支持显示器共享	■ 需额外硬件支持 ■ 受物理位置限制 ■ 切换需手动操作
软件 KVM	✧ 无需硬件支持 ✧ 延迟较低 ✧ 不受物理位置限制 ✧ 切换便捷	■ 较为依赖网络稳定性

技术路线对比如表 2 所示，经过对比发现软件 KVM 方案相比硬件切换方法，无需额外硬件支持，不收物理位置限制，使用更加便捷，更符合设计需求。软件 KVM 现有 Synergy(Synergy, 2017)、ShareMouse(ShareMouse, 2023)等不同实现，具体如表 4 所示：

表 3 软件 KVM 现有解决方案调研

现有方案	功能描述	缺点
Mouse WithOut Borders		■ 只支持 Windows ■ 延迟高 ■ 稳定性差
ShareMouse		■ 商业软件，价格昂贵 ■ 仅支持一台显示器； ■ 不支持 Linux

Input Director		■ 仅支持 Windows ■ 配置繁琐
Synergy		■ 商业软件 ■ 不支持文件传输 ■ 不支持 Wayland 桌面环境
Barrier		■ 配置繁琐 ■ 仅支持 2 台设备 ■ 不支持 Wayland
Universal Control		■ 仅支持 MacOS 或 IOS 系统
Logitech Flow		■ 仅支持罗技设备

1.2.2 需求分析

项目主要面向的场景是用户需要控制多台主机，若为每台主机均配备一套输入设备可能需要频繁切换设备，造成效率低下，且过度浪费硬件资源。因此需要一种方案，能够实现多台主机共享一套输入设备，能实现设备在主机间的自由切换。基于赛题要求及现有解决方案的痛点难题，整理该作品的需求如表 4 所示：

表 4 项目需求

需求类型	需求项	需求描述
基础需求	鼠标共享	使用主机鼠标控制多台主机，实现光标在主机间的无缝切换，支持左右键单击、拖拽、滚轮滑动等鼠标操作。
	键盘支持	支持键盘输入共享，支持组合按键及快捷键，支持多系统键码映射。
	剪贴板共享	能够实现主机间剪切板内容共享
	文件共享	利用键鼠操作和网络通信实现不同主机间的快捷文件传输。
	多主机支持	支持至少 5 台主机间共享 Hid Input 设备
	单设备响应	同一时刻只有一台设备响应输入事件。
	相对位置配置	提高 GUI 界面以高效配置主机间相对位置。
额外需求	**自动化配置**	**能够快捷高效地进行自动化配置，降低使用难度**
	安全性需求	**提供主机准入机制及数据加密机制。**
	通用性需求	**在无额外硬件支撑的条件下即可运行**
	跨平台性需求	**支持在 Windows、MacOS、Linux(wayland、x11 等桌面环境)下运行，支持不同操作系统之间进行设备共享。**
	性能需求	**设备共享时输入到响应延迟低，满足正常使用需求。**
	可扩展性需求	**通过扩展可以支持更多数量的主机连接，支持用户自定义开关相关功能**
	可用性需求	**提供用户友好的软件界面。**

1.2.3 项目贡献和突破

总的来说，我们的项目实现了**一套跨平台、可扩展、高性能的异构主机间设备共享及协同智能方案**，相比现有方案其**优势**如下：

- **完美的输入适配：**支持几乎所有键鼠输入操作，包括拖动、快捷键等。
- **丰富的功能：**支持剪切板及文件共享。
- **更强的通用性和跨平台性：**支持在不同硬件不同操作系统（Windows、Linux、MacOS）的主机平台上运行，支持在异构主机间共享 Hid Input 设备。同时还针对 Wayland 桌面协议进行了适配。
- **更高的安全性：**基于公私钥认证和加密机制，可避免恶意设备加入连接，数据安全性得以保证。
- **用户友好：**开箱即用的配置方案，高效的 GUI 配置界面，智能的主机自动发现机制，无需额外学习成本。
- **优秀的性能：**基于多线程并行模型设计，运行性能优秀，可满足正常需求。
- **优异的可扩展性：**理论支持主机数量无上限，支持多个键鼠共同使用。

第 2 章 技 术 理 论

2.1 设计思想

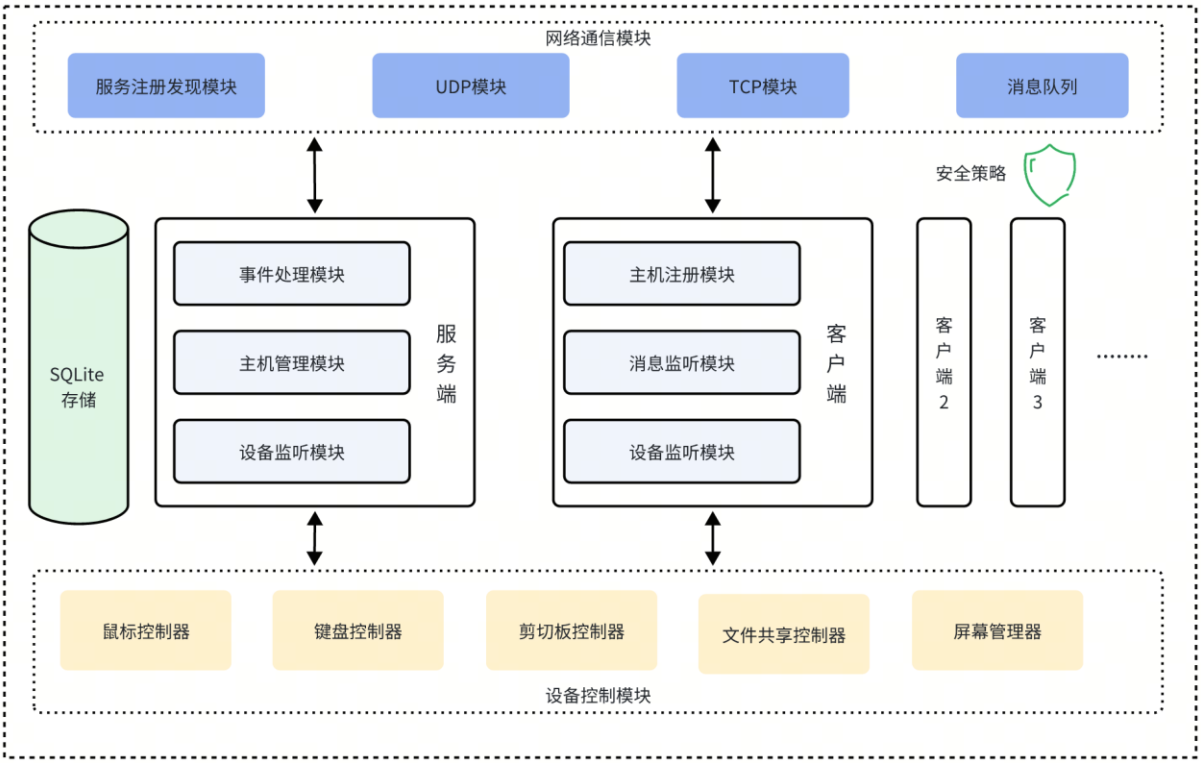


图 3 设计框架

软件整体以 C/S 架构运行，项目的整体设计框架如图 3 所示，整体由以下六个部分构成：

1. 服务端：服务端为 Hid Input 设备的拥有者，可向其他被客户端主机共享其拥有的输入设备。
2. 客户端：客户端为使用主机共享的输入设备的主机。
3. 网络通信模块：用于服务端和客户端的数据传输。
4. 设备控制模块：用于读取 Hid Input 设备信息及控制 Hid Input 设备。
5. 安全策略模块：提供设备准入机制、公私钥加密机制，保证数据安全。
6. 数据存储模块：基于 SQLite 存储本地数据。

软件的运行流程如图 4 所示，可以概括为以下几个步骤：

1. 具备 Hid 设备的主机作为服务端启动，进行服务注册。
2. 其他主机以客户端形式启动，向服务端发起连接请求。
3. 服务端处理客户端连接请求。
4. 当服务端主机的光标移出屏幕范围后，会自动判断被控的目标主机，并将本机输入设备产生的输入拦截，通过网络模块转发给客户端，客户端收到输入信息后响应相应的控制信号。
5. 当客户端的光标移出范围后向服务端主机发送事件标志，服务端主机停止控制信号的转发，并恢复输入事件的响应。

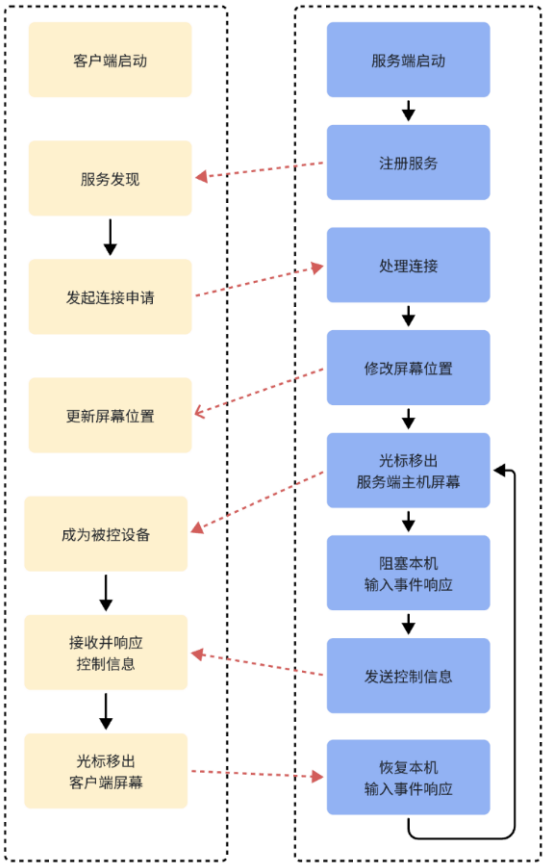


图 4 运行流程

2.1.1 服务端

服务端为具备 Hid Input 设备的主机，其状态机模型如图 5 所示，服务端由以下几个线程构成：

- 主线程：服务注册及启动其他线程
- TCP 监听线程：用于监听处理 TCP 连接，并为每一个连接创建子线程。
  - TCP 处理线程，用于处理与客户端的 TCP 连接
- 消息监听线程：监听客户端消息，主要为心跳信息和剪切板信息
- 设备监听线程：监听 Hid Input 设备的输入信息和剪切板信息。
- GUI 线程： GUI 界面的显示和处理。
- 协调线程：协调控制上述三个线程的运行。

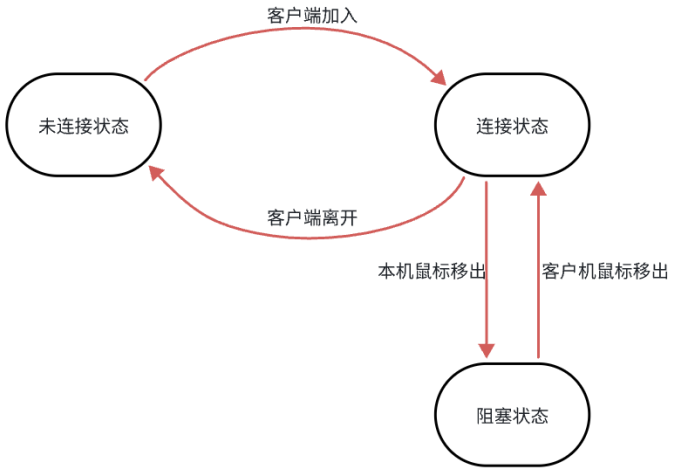


图 5 服务端状态机

2.1.2 客户端

客户端为需要使用服务主机的 Hid Input 设备的主机，其状态机模型如图 6 所示，客户端由以下几个线程构成：

- 主线程：处理服务发现，发起连接请求，启动其他线程。
- 心跳包线程：定期发送心跳包。
- 消息监听线程：用于接收并响应主机传递的控制信息。
- GUI 线程： GUI 界面的显示和处理。
- 设备监听线程：用于监听鼠标移出事件及剪切板更新事件。

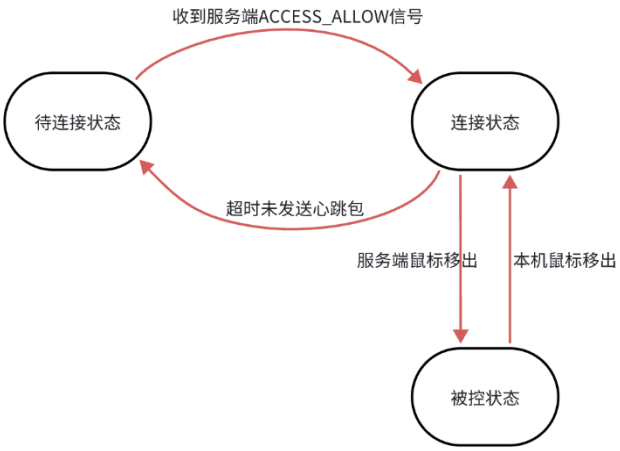


图 6 客户端状态机

2.2 技术路线

2.2.1 设备控制方案选择

不同的操作系统以及不同的桌面环境对 Hid Input 设备的控制逻辑不同，为此我们选用了基于 Python 语言的 pynput 来对 Hid Input 设备进行控制和监听(Palmér, 2024)，然而由于 Wayland 桌面环境去除了 X server，导致无法获取和处理输入事件，我们选用 evdev 获取输入事件(Gvalkov, 2024)，选用 uinput 以虚拟设备的方式进行事件输入。

2.2.2 网络通信协议

为了在不同设备之间实现键鼠事件的传输，我们选用高效的网络通信协议，采用了 TCP/IP 协议进行数据传输，保证数据的可靠传递。同时，针对键鼠事件的实时性要求，进行了协议优化，尽量减少数据包的大小和传输延迟。

2.2.3 主机发现与连接管理

在多设备键鼠共享系统中，自动化配置可以提高软件使用效率，降低学习成本。为此我们借助 mDNS 协议进行服务注册和服务发现(IETF, 2013)。这样设备可以自动发现网络中的其他主机。同时，为了提高连接的稳定性，设计超时断连机制，确保当设备离线或网络中断时，系统能够及时响应并重新建立连接。

2.2.4 GUI 界面设计

为了保证软件的跨平台性，我们使用了 Python 语言结合 QT 来开发应用界面和核心逻辑。我们基于 PyQT5 设计良好的图形化用户界面，使用户可以方便地配置和管理多个设备。(Pypi, 2024)用户可以通过拖放操作在界面上设置设备之间的相对位置，系统会根据这些设置限制鼠标的移动范围及方向。基于 PyQT5 还可实现桌面通知机制，当设备连接或断开时，用户会收到相应的通知，提升用户体验。

2.2.5 安全性

为保证跨设备通信的安全性，我们采用 **RSA 公私钥认证和数据加密机制**(Ron Rivest, 1983)。在设备连接时，双方需要进行公私钥的认证，确保通信双方的合法性。数据传输过程中，所有数据都经过加密处理，防止数据在传输过程中被截获或篡改。

2.2.6 剪切板与文件共享

在实现剪切板和文件共享功能时，我们需要考虑不同操作系统之间的差异。对于剪切板共享，我们采用跨平台的剪切板管理库 pyperclip，确保不同系统之间剪切板内容的兼容性(Sweigart, 2014)。然而由于 wayland 环境的安全性考量，pyperclip 无法正常工作，采用 wl-clipboard 方案进行适配。文件共享功能则通过网络文件传输协议实现，用户可以在不同设备之间方便地传输文件，提升工作效率。

2.2.7 延迟优化

为满足日常办公的基本需求，需对系统的延迟进行了优化。通过减少数据包大小、优化网络传输协议和高效的事件处理机制，我们将设备共享的延迟控制在可接受的范围内，确保用户在使用过程中不会感受到明显的延迟。

2.3 代码原创说明

本项目主要实现多设备键鼠共享功能，涉及图形用户界面、键鼠事件捕捉与发送、主机发现与连接管理等多个模块。表 5 为各个模块中第三方模块的详细引用情况说明：

表 5 第三方 Python 库引用情况

名称	版本	使用说明	引用方式	开源协议
qt-material	2.14	用于优化界面风格	Import	BSD License
pillow	9.5.0	图像处理库，用于 GUI 界面显示器示意图片渲染	Import	BSD License
PyQT5	5.15.11	用于搭建 GUI 界面	Import	GPL v3
pynput	1.7.7	用于鼠标控制及键盘控制模块	Import	LGPLv3
pyperclip	1.9.0	用于剪切板控制模块	Import	BSD License
pyevdev	1.6.1	用于鼠标及键盘控制模块，读取/dev/input 数据	Import	BSD License
zeroconf	0.132.2	实现局域网内设备的自动发现和连接	Import	LGPLv2
netifaces	0.11.0	提供网络接口信息	Import	MIT License
rsa	4.9	用于实现 RSA 公私钥加密和解密	Import	Apache Software License
screeninfo	0.8.1	提供屏幕分辨率和多显示器信息	Import	MIT License

第 3 章 软件介绍

3.1 软件功能介绍

软件基于 C/S 架构，整体架构如图 7 所示。借助设备控制模块进行输入事件的监听和模拟，借助网络通信模块进行主机间连接和事件通信。使用 SQLite 作为本地数据存储方案。

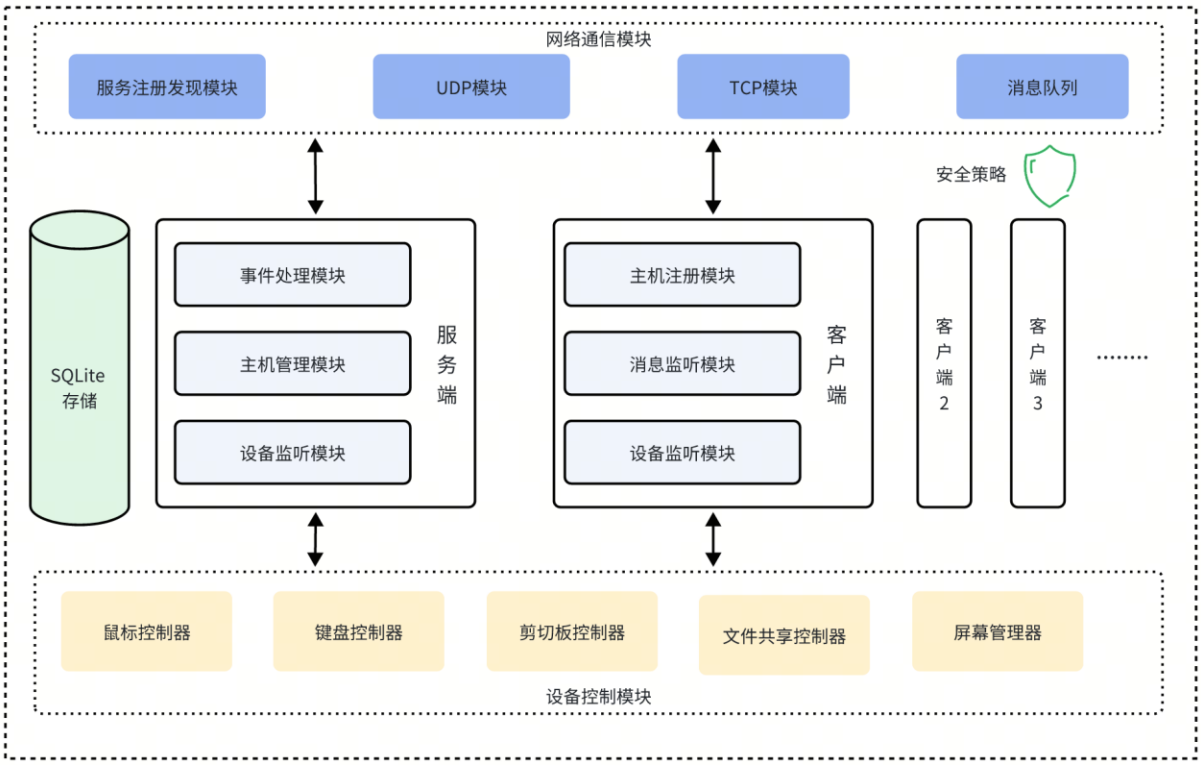


图 7 软件架构框图

3.1.1 设备控制模块功能介绍

该作品通过封装设备控制模块提供 Hid Input 设备控制和输入监听功能，其具体实现原理为监听主控机上的输入事件，通过网络模块传递到被控设备，模拟进行事件输入。主要基于 pynput 实现，同时基于 evdev 和 uinput 对 Wayland 桌面环境进行了额外适配。具体原理是使用多个线程通过读取/dev/input 设备文件监听潜在的输入设备，并借助 uinput 创建虚拟输入设备，模拟输入设备的输入，写入输入事件。

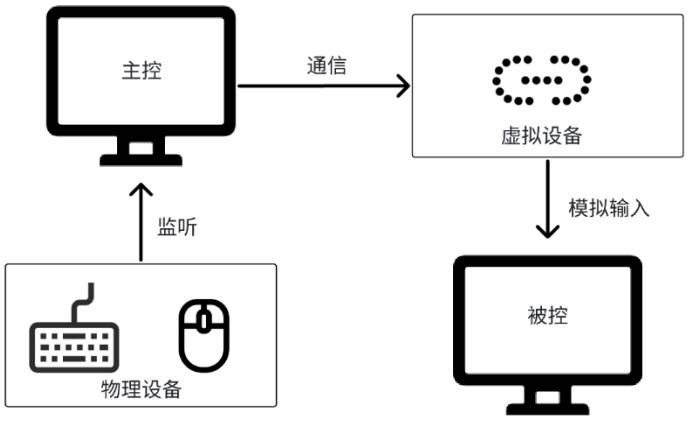


图 8 设备控制原理

该模块由鼠标控制模块、键盘控制模块、剪切板控制模块、文件共享控制模块四个子模块构成，各个模块功能如表 6 所示，同时由于操作系统的不同，在子模块内部实现针对不同的操作系统进行了适配，保证了上层模块调用时的透明性。

表 6 设备控制模块功能表

模块	函数	功能
鼠标控制模块	get_position	鼠标位置获取
	update_last_position	鼠标位置记录，用于获取位移
	move_to	鼠标位置设置
	move	鼠标位移控制
	click	鼠标点击控制
	scroll	鼠标滚轮控制
	mouse_listener	鼠标事件监听器

键盘控制模块	click	键盘点击控制
	press	键盘 press 控制
	release	键盘 release 控制
	keyboard_listener	键盘监听器
	encode	键盘编码
	decode	键盘解码
剪切板控制模块	paste	设置剪切板内容
	copy	获取剪切板内容
	clipboard_listener	剪切板监听器
文件共享控制模块	file_listener	文件监听器
	save_file	存储文件
	send_to_all	主控机器向被控设备分发文件
	send_to_master	被控设备向主控设备传输文件

3.1.2 网络通信模块功能介绍

项目的网络通信模块包含 UDP 模块和 TCP 模块，UDP 主要用于广播，心跳机制以及输入设备控制信息的转发。TCP 模块主要用于主机间事件的通信以保证通信的可靠性。同时为了平衡设备间数据处理能力的差距，还增加了消息的缓冲队列。所有传输的数据由消息类型和消息体构成，目前系统的消息类型及说明如表 7 所示：

表 7 消息类型说明

消息类型	说明	协议
CLIENT_HEARTBEAT	心跳包	UDP
SUCCESS_JOIN	服务端对主机成功加入的回应	UDP
MOUSE_BACK	光标从被控机移出回到主机	TCP
MOUSE_MOVE	光标移动事件，消息体为具体坐标	UDP
MOUSE_MOVE_TO	光标移动事件，消息体为移动位移	TCP
MOUSE_CLICK	鼠标点击事件，包括 Press 和 Release	UDP
MOUSE_SCROLL	鼠标滚轮事件	UDP
KEYBOARD_CLICK	键盘点击事件，包括 Press 和 Release	UDP
CLIPBOARD_UPDATE	剪切板更新事件	UDP 广播
POSITION_CHANGE	屏幕位置更新事件	TCP
SEND_PUBKEY	发送公钥	TCP
TCP_ECHO	TCP 的回应	TCP
KEY_CHECK	公私钥校验	TCP
KEY_CHECK_RESPONSE	公私钥校验结果	TCP
ACCESS_DENY	允许被控机连接	TCP
ACCESS_ALLOW	禁止被控机连接	TCP

为了简化设备连接的操作，基于 Zeroconf 实现了被控机主机自动查找服务地址功能，在局域网环境下无需人工干预即可发现服务地址(IETF, 1999)，基本原理是服务端将服务的地址等信息创建为一个服务信息对象并借助 mDNS 广播该服务信息。被控机监听 mDNS 广播，使用 ServiceBrowser 对象进行服务发现，当有符合条件的服务出现时便能及时获得服务地址。

3.1.3 加密认证功能介绍

设计实现基于公私钥机制的准入模块设计，可以确保系统的安全性和数据的完整性，保证 Hid input 设备共享时的安全性。仅有经过授权的主机才可加入连接，其流程图 9 所示，具体流程如下：

1. 客户端生成公私钥对
    a) 客户端首次启动时，使用 RSA 算法生成一对公私钥（K_{Cpub}、K_{Cpri}）
    b) K_{Cpri}保存在客户端本地，K_{Cpub}将用于身份验证
2. 客户端发起连接请求
    a) 客户端向服务器（Server）发起连接请求，同时携带其公钥K_{Cpub}
3. 服务器处理连接请求
    a) 服务器接收到连接请求后，提取客户端的公钥K_{Cpub}。
    b) 如果服务器拒绝该连接请求，则直接断开连接。
    c) 如果服务器接受该连接请求，生成一个随机序列 R 作为验证字符
    d) 服务器使用客户端的公钥K_{Cpub}对 R 进行加密生成密文$C = E(K_{Cpub}, R)$。
    e) 服务器将加密后的密文 C 返回给客户端。
4. 客户端处理验证消息
    a) 客户端接收到加密的随机序列 C 后，使用K_{Cpri}进行解密，$R' = D(K_{Cpri}, C)$得到序列R'。
    b) 客户端将解密后的结果 R' 发送回服务器。
5. 服务器验证解密结果

a) 服务器接收到客户端发送的解密结果R'

b) 服务器将收到的解密结果R'与 R 进行比较。

c) 如果两者一致，则验证成功，允许连接。如果不一致，则验证失败，拒绝连接。

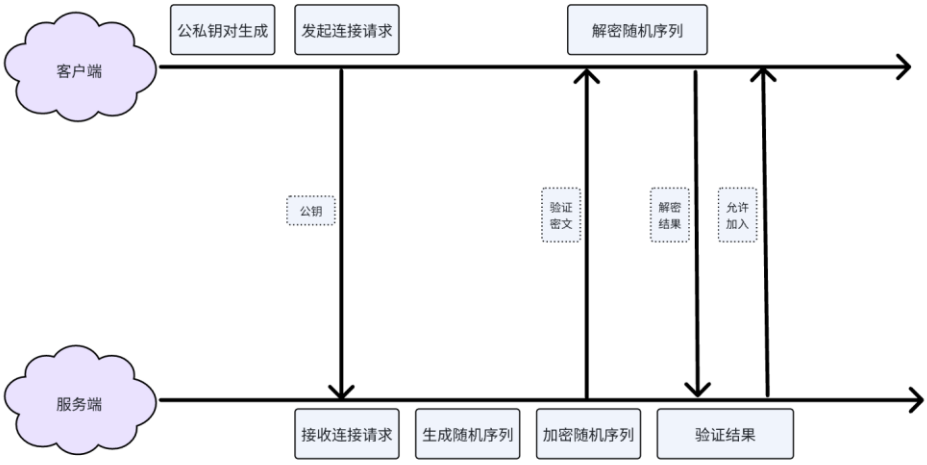


图 9 认证机制流程图 A

以上机制可以防止未经授权的客户端伪装为授权客户端加入连接，但无法处理恶意主机伪装为服务端，如需应对此安全风险，可增加安全级别，在客户端上存储服务端的公钥，在此情况下连接流程如图 10，具体流程如下：

1. 客户端提前存储服务端公钥

   a) 客户端在初始配置阶段，存储服务端的公钥K_{Spub}

2. 客户端发起连接请求

   a) 客户端生成一个随机序列R_a

   b) 客户端使用服务端的公钥K_{Spub}加密R_a，生成密文$C_a = E(K_{Spub}, R_a)$

   c) 客户端向服务器发起连接请求，携带自身的公钥K_{Cpub}和加密后的C_a，消息体为$\{K_{Cpub}, C_a\}$

3. 服务器处理连接请求

a) 服务器接收到连接请求后，提取并验证客户端的公钥K_{Cpub}。

b) 如果服务器拒绝该连接请求，则直接断开连接。

c) 如果服务器接受该连接请求，使用私钥K_{Spri}解密C_a，得到序列$R_a{}'$

d) 服务器生成一个随机序列R_b，使用客户端的公钥K_{Cpub}加密R_b，生成密文$C_b = E(K_{Spub}, R_b)$

e) 服务器返回消息体$\{R_a{}', C_b\}$

4. 客户端验证服务器响应

a) 客户端接收到服务器的响应后，验证解密后的$R_a{}'$是否与最初生成的一致。若不一致则放弃连接请求

d) 一致则客户端使用私钥K_{Cpri}进行解密，得到序列$R_b{}' = D(K_{Cpri}, C_b)$。

b) 客户端将解密后的$R_b{}'$返回给服务器。

5. 服务器验证客户端响应

a) 服务器接收到客户端发送的$R_b{}'$后，验证其是否与最初生成的R_b一致，如果一致，则允许连接。如果不一致，则拒绝连接。

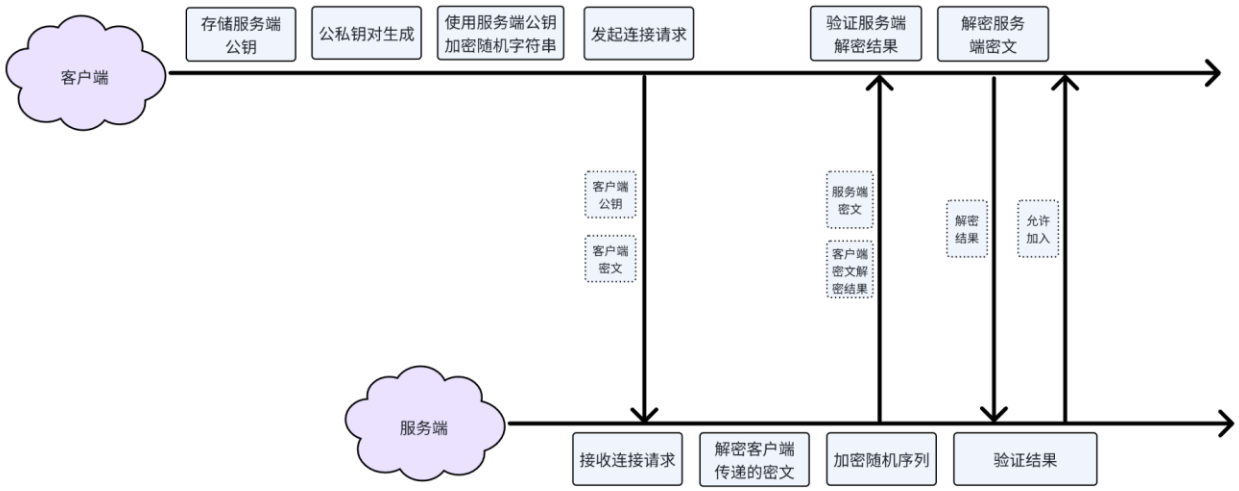


图 10 认证机制流程图 B

经过以上处理可以保证连接的安全性，此外还需保证数据的安全性，确保数据不被泄露不被伪造，方案流程如图 11，具体流程如下：

1. 服务端生成控制信息并签名

服务端使用自己的私钥K_{Spri}对控制信息$M_{control}$进行签名，生成签名S，将签名附加到控制信息上，形成消息$\{M_{control}, S\}$

$$S = \text{Sign}(K_{Spri}, M_{control})$$

服务端使用目标客户端的公钥K_{Cpub}对消息进行加密，生成密文C，将加密后的密文 C 发送给客户端

$$C = E(K_{Cpub}, \{M_{control}, S\})$$

2. 客户端解密和验证签名

客户端接收到密文 C 后，使用自己的私钥K_{Cpri}进行解密，得到消息$\{M_{control}, S\}$，客户端使用服务端的公钥K_{Spub}验证签名S，如果验证通过，客户端接受并处理控制信息$M_{control}$。

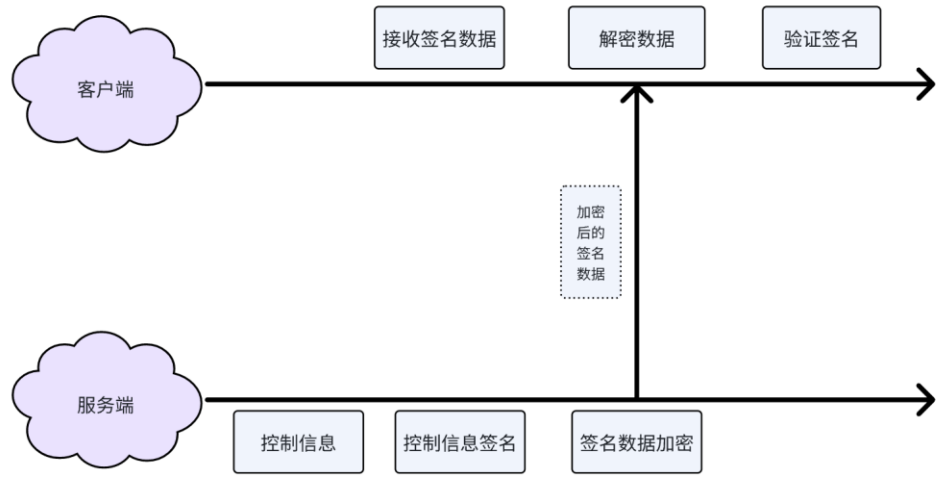


图 11 数据签名加密流程图

此外由于剪切板内容涉及到多台主机之间的共享，为保证整个过程的安全性，还需对剪切板内容进行处理，处理流程如图 12，具体流程如下：

1. 客户端加密剪切板信息

客户端使用服务端的公钥K_{Spub}对剪切板信息$M_{clipboard}$进行加密，生成密文$C = E(K_Spub, M_{clipboard})$。将加密后的密文$C$发送给服务端。

2. 服务端解密剪切板信息

服务端接收到密文C后，使用自己的私钥K_{Spri}进行解密，得到剪切板信息$M_{clipboard}$。

3. 服务端分发剪切板信息

服务端分别使用各个目标客户端的公钥K_{Cpub_i}对剪切板信息$M_{clipboard}$ 进行加密，生成对应的密文$C_i = E(K_{Cpub_i}, M_{clipboard})$。将加密后的剪切板信息$C_i$发送给各个客户端。

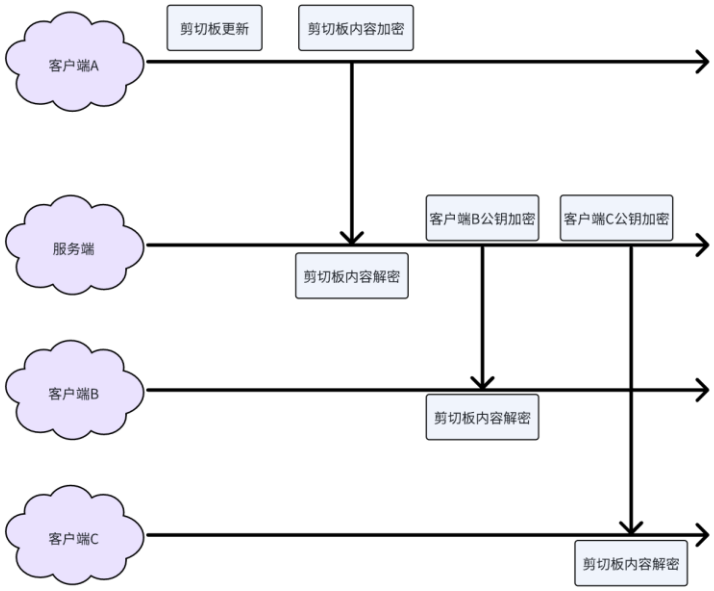


图 12 剪切板内容加密机制

3.1.4 文件共享功能介绍

为了实现文件共享功能，为每个主机增加一个共享文件夹，当次文件夹下的文件发生变更时，触发同步机制，实现多主机文件共享。具体流程如下：

- 所有主机服务启动后，建立共享文件夹，并启动监听线程监听文件夹下文件信息。

- 当主机监听到文件信息改变时，将改变的文件信息传输给主控机器。
- 主控机机器对收到的文件进行校验后，分发给其他机器。

3.2 软件界面

项目基于 PyQt 框架构建了桌面软件进行人机交互，从而便于用户查看主机信息并配置屏幕相对位置，更好地实现设备共享。

首先，软件将主控机和被控机两部分功能模块集成起来，并在桌面软件上提供了选择接口，作为软件的入口界面，设计如图 13 所示：

图 13 入口 GUI 界面

在入口界面中，提供了主控机和被控机两种角色供用户进行选择，当用户点击下方“进入”按钮时，根据其选择的角色转入相应模块执行。在选择受控机（被控机）角色的情况下，该主机会加入共享设备集群，并向服务器发送信息宣布其已上线或请求加入。在选择主控机（服务器）角色的情况下，用户能够实时地查看集群内的被控机信息并配置其相对位置，主控机界面如图 14 所示：

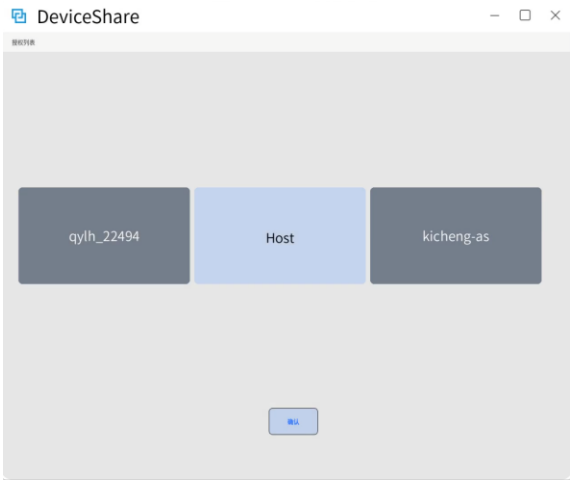


图 14 被控机相对屏幕位置配置界面

主控机角色下的 GUI 界面主要提供了两方面功能，分别是查看被控机状态以及配置相对位置。

1. 软件设计了位于软件坐上角的授权列表，展示被控机的在线、离线状态，此列表悬停在软件左上角，通过菜单栏的“授权列表”控制其显示和隐藏。
2. GUI 的主体是配置屏幕相对位置功能，在这一部分中，考虑到被控机相对服务器屏幕有上下左右四种相对位置，将界面设计成十字形，中心表示主控机屏幕，四个方向分别用 4 个矩形框表示相应的屏幕，用相应位置是否为空表示该方向是否配置了被控机，并对离线状态的被控机进行一定的虚化处理以直观地表现其状态。当用户拖动图像时，可以将其移动至不同方向的矩形框，以修改对应被控机的相对位置，为了提供用户友好的图形操作界面，软件还进行了一定的优化处理，如移动到某个矩形框附近时，对应矩形框高亮显示以提示用户，在此状态下释放鼠标，图形自动和中心图像，即服务器屏幕对齐，从而提供更好的用户体验。下方的确认按钮用于对修改的相对位置信息进行保存。

为了方便处于主控机角色下的用户实时地获取被控机状态，软件还设计了消息弹窗，在被控机上线或下线时立即通知主控机。此外，软件还提供了托盘图标，通过右键即可进行简单的登录或退出，进一步简化了操作。当作为被控机的主机首次上线时，可以发出加入集群的申请，服务器对此进行处理，得到同意的被控机会被主控机加入其授权列表。此后，被控机的上线或下线消息均会发送至主控机，并在 GUI 界面上更新其状态信息。其效果如图 15 所示：

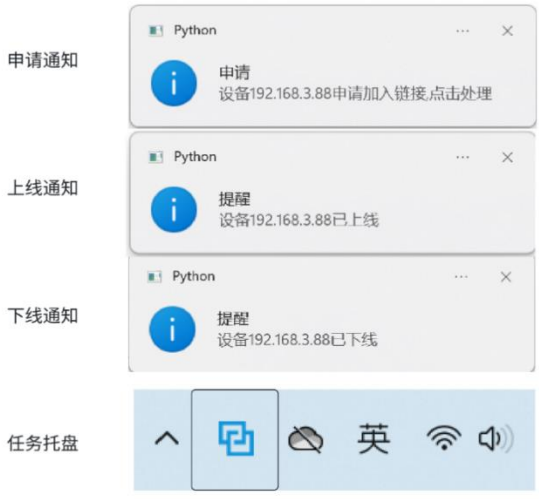


图 15 通知弹窗及系统托盘

此外为了给用户提供更多自由度，用户可以在如图 16 所示的 GUI 界面上对相关功能进行开关控制，还可以根据鼠标性能配置不同的传输粒度以保证共享鼠标的性能。

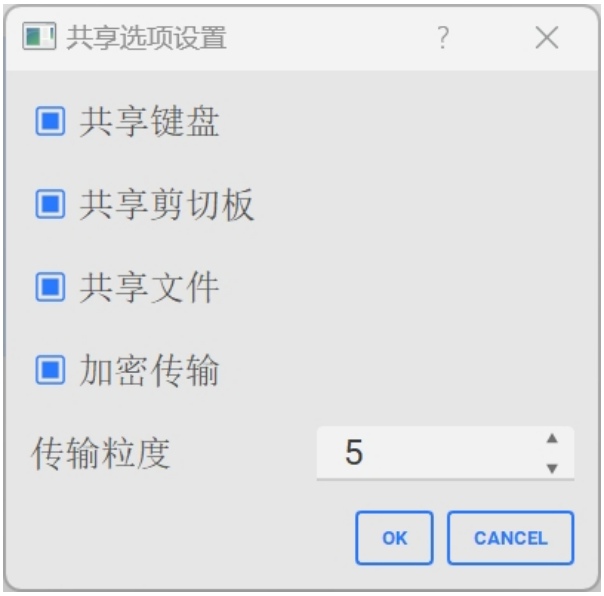


图 16 共享功能设置

第 4 章 软件测试

4.1 软件使用说明

针对 x86 架构的 Windows、openKylin 操作系统，我们打包构建了可执行程序，可在 Release 界面下载合适的版本。

若在 Linux 下运行，采用脚本 run.sh 启动程序，将 run.sh 复制到 dist 目录下，执行 sudo chmod 777 run.sh 赋予执行权限，执行 bash run.sh 启动程序,windows 下无需执行此步骤,直接运行 exe 文件即可。

若构建的版本无法支持目标机器，可选择源码运行或自行打包。该方案需具备 Python 3 环境，具体步骤如下：

1. 获取项目源代码

2. 使用 pip install -r requirements.txt 命令安装依赖

3. 执行 python deviceShare.py 启动程序

4. 安装 pyinstaller: pip install pyinstaller

5. 使用 pyinstaller 打包目标程序: pyinstaller deviceShare.spec

6. 运行 dist 目录下生成的可执行文件

7. 将 resources 目录复制到 dist 目录下

使用时，将主控机和被控机连接到同一局域网环境下，保证两台机器能够互相 ping 通，分别按照要求启动软件，选择各自的角色。

1. 被控机上线后，主控机将弹出通知，点击通知允许被控机连接。
2. 右键单击主控机右下角的桌面托盘，点击设置，可以看到被控机屏幕位置示意图，可拖动修改位置，修改完成后点击确定。
3. 在主控机上将光标移动到靠近被控机方向的边缘即可进入被控机。
4. 在被控机上将光标移动到靠近主控机方向的边缘即可回到主控机。
5. 在主控机和被控机上分别右键单机托盘图标选择退出即可关闭软件，注意在 openKylin 环境下由于收到消息提醒，此时托盘图标可能变成小灯泡形式。

注意 Kylin 操作系统在安装 python 的 evdev 依赖时可能出现错误，请选择安装预编译版本 evdev-binary，并使用包管理器安装 python3-dev 和 libevdev-dev 包，参考 https://python-evdev.readthedocs.io/en/latest/install.html

此外该系统也支持运行在图 17 所示的树莓派，Jetson 等小型主机上，以提高便携性，并在无法连接外设的场景下使用。

图 17 小型主机示意图

4.2 软件测试说明

项目在 Windows10、Debian12、Ubuntu20.4、MacOS、openKylin 等多个平台进行运行测试，服务端主机连接 usb 键鼠，所有主机以无线局域网的形式连接，测试环境如图 18 所示。

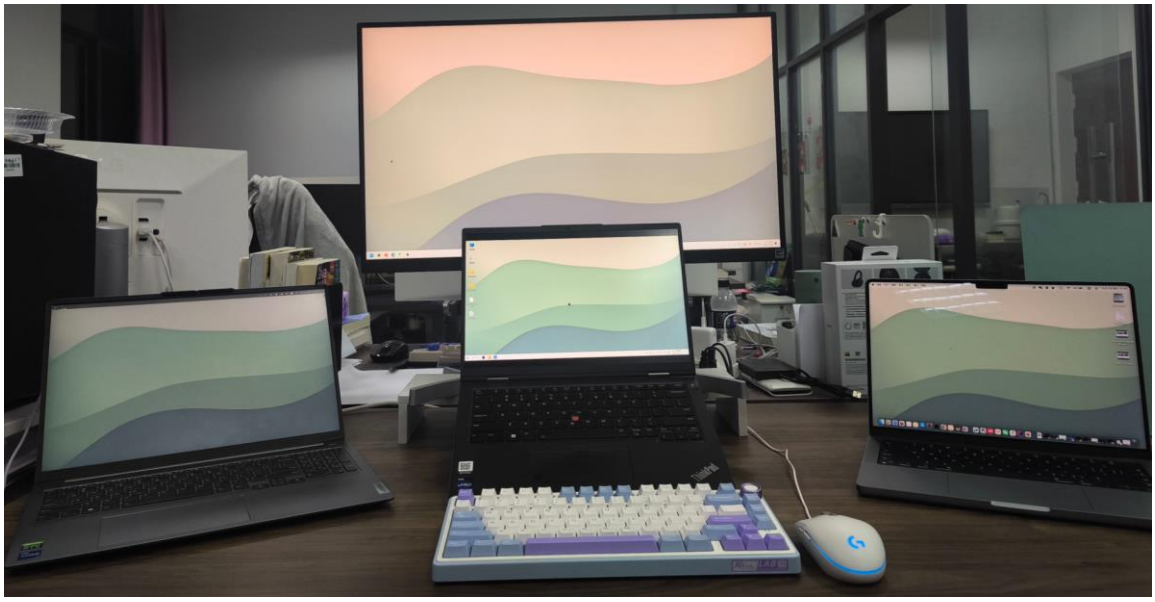


图 18 测试环境

4.3 软件测试结果

为了确保项目运行的正确性，在测试环节使用不同的测试环境对程序的运行进行了测试，检验其运行流程，从功能、性能、兼容性、稳定性等方面进行测试。

4.3.1 功能测试

功能测试主要针对多屏幕管理、设备监听、鼠标控制、键盘控制、剪切板控制等功能模块进行。

1. GUI 功能测试

此模块用于实现多台主机进行设备共享时，通过可视化方式操纵屏幕间的相对位置，实现服务器主机对被控机的不同控制方式。

表 8 GUI 功能测试

测试功能	测试步骤	结果
身份选择	启动软件，分别选择成为主控机和被控机	软件正常启动，分别进入主控机和被控机程序。
连接请求	被控机启动，发送连接申请 主控机分别选择同意和拒绝	点击同意后设备加入，显示上线通知。点击拒绝后设备无法加入。
连接记忆	同意被控机连接后，被控机重新上线	无需再次确认，直接显示设备上线通知。
设备加入监听	在主机上启动 client 程序，通知服务器	服务器监听到主机加入并在可视化界面添加对应屏幕，并显示桌面通知。
相对位置控制	拖动被控机对应图像	通过鼠标拖动操作实现了被控机屏幕相对主机的位置改变

位置保存	点击确定	被控机的信息及相对位置，鼠标进行相应移动发现位置修改成功。

2. 设备监听功能测试

被控机启动时，可以通过广播方式通知服务器其加入了设备共享，此模块用于实现该功能并实现对多被控机的管理。

表 9 设备监听测试

测试功能	测试步骤	结果
设备加入	在被控机启动 client 程序	服务器正常接收被控机的广播。消息并将其加入客户列表
设备退出	中止 client 程序	主控机检测到被控机通信中止，将其从客户列表删除

3. 鼠标控制模块

此模块用于实现多主机的鼠标共享，在主控机，当控制鼠标移动到特定位置后（具体取决于多屏幕管理中的相对位置设置），可以通过网络通信控制对应的被控机，并需要实现鼠标在不同主机间的自由切换。

表 10 鼠标控制测试

测试功能	测试步骤	结果
主控机鼠标控制	在主控机移动鼠标，并进行点击等操作	屏幕光标可随着鼠标控制移动，且点击、滚轮等操作均正常执行

切换至被控机	在主控机移动鼠标至屏幕右侧	主控机的光标移动到固定位置且不可移动，被控机端光标位置可随着鼠标移动移动
鼠标移动	在切换至被控机后进行鼠标移动操作	被控机端光标可以随着主控机的鼠标进行移动。
鼠标点击	在切换至被控机后进行左键右键点击操作	在被控机器上鼠标点击事件可以正确接收且处理，同时主控机阻塞响应。
鼠标滚轮	在切换至被控机后进行鼠标滚轮操作	在被控机器上滚轮事件可以正确接收并处理，同时主控机阻塞响应。
鼠标拖动	在切换至被控机后进行鼠标拖动操作	在被控机器上鼠标拖动可以正确接收并处理，同时主控机阻塞响应。
回到主控机	在对应模式下将光标移动至屏幕左侧	被控机端光标无反应，鼠标可以正常控制主控机

4. 键盘控制模块

此模块用于实现多主机的键盘共享。

表 11 键盘控制测试

测试功能	测试步骤	结果
主控机鼠标控制	在主控机进行键盘点击、press、release 等操作	键盘操作均生效

切换至被控机	在主控机移动鼠标至屏幕右侧	主控机的光标移动到固定位置且不可移动，被控机端光标位置可随着鼠标移动移动
被控机键盘点击	在切换至被控机后，在记事本等程序上测试键盘点击	主控机的键盘按键可被接收并能正常在被控机输入
被控机组合键	在被控机进行组合键测试 ctrl+c	事件可被正常处理
回到主控机	在对应模式下将光标移动至屏幕左侧，并进行键盘测试	键盘操作仍可在主控机生效

5. 剪切板控制模块测试

表 12 剪切板控制测试

测试功能	测试步骤	结果
剪切板内容获取	在主控机控制被控机，通过 copy 等操作获取	剪切板内容可被获取
剪切板内容设置	在主控机通过 paste 等操作设置剪切板内容	剪切板内容可以正常传输

6. 文件共享模块测试

表 13 文件共享模块测试

测试功能	测试步骤	结果
主控机文件更新	在主控机上更新文件	所有被控机均可获取到文件

被控机文件更新	在一台被控机更新文件	主控机和所有被控机均可获取到文件

7.

4.3.2 性能测试

在性能测试中，我们首先测试了在局域网环境下鼠标和键盘在共享时的操作传输延迟，该延迟定义为主控设备发生操作到被控设备产生响应的时间间隔，由于两台设备间存在始终偏差，使用往返时间和除二的方式计算除控制信号传输时延约为 3ms。

另一方面鼠标的回报率是影响用户体验的一大指标，根据调研，市面上的大部分鼠标回报率在 125Hz-1000Hz 间，如图 19 我们测试了在使用共享鼠标时鼠标移动的回报率，测试发现回报率可以达到 250Hz，可满足日常需求。

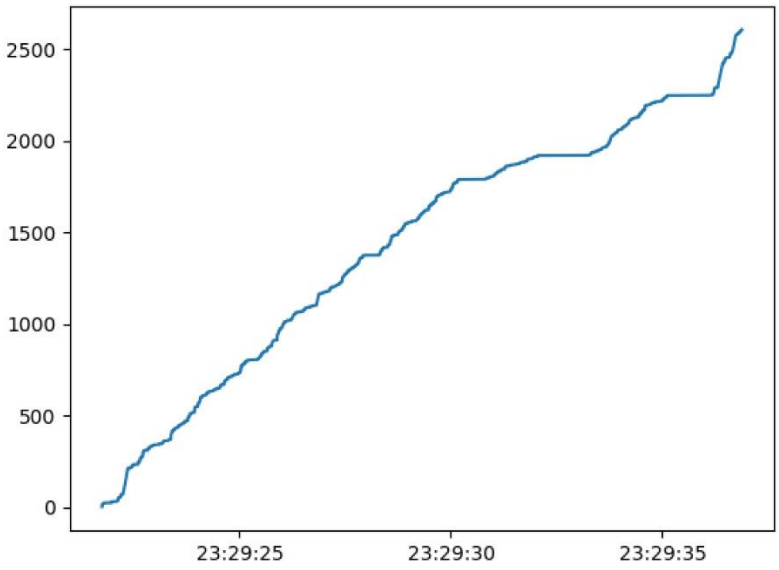


图 19 鼠标回报率测试

4.3.3 兼容性测试

考虑到项目在多平台上的兼容性，在测试时使用了不同的操作系统，试运行并调试了项目代码。

测试环节使用 openKylin1.0.2、Windows11、Debian12、Ubuntu20.04、MacOS 13 四个平台均运行了程序，并尝试在不同平台上分别运行被控机和服务器，结果显示，项目可以实现在以上平台上的兼容。

第 5 章 实现难点说明

DeviceShare 实现多设备键鼠共享软件涉及多个复杂的技术挑战，包括跨设备兼容性、跨系统兼容性、通用性和安全性，本章主要对实现过程中的重难点问题进行梳理介绍。

5.1 兼容性和通用性问题

在系统 API 和驱动差异性上，不同操作系统（如 Windows、macOS、Linux）以及 Linux 不同的桌面协议(Wayland、X11、Xorg 等)下有不同的驱动机制(Linux community, 2024)，这会影响键鼠的输入和输出，处理这些差异要编写针对不同操作系统的适配性程序；在权限和安全的差异性上，不同操作系统对硬件访问有不同的权限限制和安全机制，比如在 Wayland 环境下会限制应用程序对输入设备的访问。此外在输入设备对操作系统的兼容性上，键盘本身可能会具有不同的布局和功能键，例如相同的键盘键位对于 Windows 操作系统和 MacOS 操作系统中可能会有不同的意义。

为了解决上述问题，保证软件的兼容性在实现时采取了以下解决措施：

- 使用 Python 语言开发，一套代码可在多个平台上运行和打包
- 采用跨平台的开发包，如 PyQT5、pynput 等
- 针对 Wayland 环境进行单独适配，依靠 evdev、uinput 等更底层的方案进行输入设备的控制和监听。
- 设计键码转换中间层，进行不同平台的键盘编码转换。

5.2 软件配置难度高

由于软件使用的网络环境可能比较复杂，手动配置通信地址会给非专业用户带来困扰，产生额外的学习成本。为了减少软件的配置难度，降低用户使用门槛，在实现过程中一方面采用基于 mDNS 广播的 ZeroConf 协议，系统启动时自动检测当前网络环境和设备配置，进行服务注册，被控机上线时自动发现潜在的服务，用户无需进行手动配置。另一方面设计了用户友好的 GUI 交互界面及通知机制，使用方式一目了然，用户可高效地配置软件。

5.3 安全性

在一些保密场景下，数据安全是十分重要的，因此在不同设备之间传输键盘和鼠标数据时，需要确保数据的保密性和完整性，防止敏感信息泄露。在设备的访问控制上，要考

虑防范病毒软件和脚本程序伪装成主控机或被控机进行恶意攻击和破坏性操作，确保只有通过授权的用户和设备才能加入。具体来说可能存在以下三种风险：

- 恶意机器伪装成被控机加入连接
- 恶意机器伪装成主控机控制其他主机
- 恶意窃取软件运行过程中的网络数据（如键盘输入信息，剪切板内容）

为此 DeviceShare 设计实现了公私钥加密认证机制来保证安全性，通过主控机和被控机的身份验证及加密数据的交换，防止未经授权的访问，并保护键鼠和设备交互信息的机密性和完整性。具体细节可参考 3.1.3 节介绍。

5.4 高性能低延迟

在多设备间实现键鼠共享，要求系统能够实时响应用户的操作，低延迟是关键指标，不同设备的硬件性能和网络环境也会影响通信的实时性。在开发过程中发现鼠标共享时存在较大的延迟，会给使用带来明显的卡顿。为此我们从两个维度进行了优化：

- 网络传输调优
  - 合理的网络协议选型：根据不同的传输需求选用不同的网络协议。
  - 消息队列模型：用高性能消息队列优化数据传输速度和稳定性。
  - 多线程异步网络 IO。
  - 高效序列化机制：设计实现了基于 Json 的高效的序列化机制来编码和解码数据。
- 设备 I/O 处理优化
  - 异步事件监听处理：异步监听和处理输入设备的 I/O 事件，提高系统效率
  - 批量事件处理: 由于 uinput 涉及到用户态和系统态的切换，会产生较大的性能开销，可以累积一定数量的事件之后统一发送。

5.5 鼠标回报率优化

由于不同的鼠标型号具有不同的回报率（鼠标向主机发送信号报告鼠标指针位置和移动信息的频率从几百赫兹到上千赫兹不等），对于高回报率鼠标时如果每次位置更新都将位移数据传输给其他主机会给网络带来很大负载，造成移动延迟。最直接的解决方案是将多次采样合并到一次进行传输，但这种方案在使用低回报率鼠标时会带来严重的卡顿感。为此我们在 GUI 界面上提供了鼠标移动传输粒度的调整选项，用户可根据所用的鼠标情况自己调整累计多少次采样触发一次传输，同时也支持根据网络情况或位移量动态调整。

5.6 主机连接数量优化

为了提高系统的可扩展性，应对机房运维等复杂使用场景，需要让系统支持尽可能多的主机连接数量，为此系统采用如下优化策略来提高支持的主机连接数量：

- 点对点连接，在设备共享时主要处理主控机和被控机的连接，尽量减少其他主机与主控机的频繁 IO，提高系统上限。
- 多线程模型及线程池机制，采用多线程模型使系统能够应对多主机连接情况，并次啊用线程池机制避免线程频繁创建和销毁的开销。
- 多路 IO 复用提高系统的并发性和效率。

第 6 章 总 结

在整个项目过程中，我们依托 Gitlink 平台按照严谨的流程进行开发。整个项目经历了需求分析、架构设计、技术选型、设计开发、测试优化等多个阶段，在 6 个分支上进行了 300 余次 commit，最终实现了一个**跨平台、可扩展的多主机设备共享及协同智能解决方案**，该方案具有**跨平台支持，多维度性能调优，安全的自主化配置协议，无上限的主机连接数量等特性**。

在整个项目的开发过程中锻炼了我们的创新意识、协作和解决问题的能力。此外我们对开源精神有了进一步的理解和认识，也将继续激励我们未来投身开源项目中，凝聚智慧，推动技术进步、知识共享和社会创新。感谢我们的指导教师赵志为教授的倾心指导，以及电子科技大学计算机科学与工程学院（网络空间安全学院）的各位老师给予的支持和宝贵建议。此外还要感谢 openKylin 社区在国产操作系统上提供的支持，感谢 Gitlink 提供的代码协作平台，感谢 pynput、pyevdev 等第三方开源库的开发者们。

目前 DeviceShare 已经完成了赛题的基本要求，并在赛题的基础上增加了主机发现、认证加密等额外功能，同时具备非常好的的跨平台兼容性，能够在搭载不同操作系统的主机间共享输入设备。性能可满足日常办公需求。未来团队将在以下方面继续进行完善该项目：

- 细化配置粒度，探索非局域网主机设备高效配置方案
- 优化代码逻辑，优化并发方案，提高软件性能
- 多显示器支持
- 优化适配更多操作系统

参 考

[1]. openKylin. openKylin document. https://docs.openkylin.top/en/home

[2]. Microsoft. 了解远程桌面协议 (RDP). https://learn.microsoft.com/zh-cn/troubleshoot/windows-server/remote/understanding-remote-desktop-protocol [2023-12-26]

[3]. Ugreen. UGREEN 4 Port USB 3.0 5Gbps High-Speed Switch Selector. https://uk.ugreen.com/products/ugreen-4-port-usb-3-0-5gbps-high-speed-switch-selector [2024]

[4]. ATEN. 2-Port USB 3.0 4K DisplayPort KVMP™ Switch (Cables included) - CS1922. https://www.aten.com/global/en/products/kvm/desktop-kvm-switches/cs1922 [2024]

[5]. Belkin. F1DS102L and F1DS104L User Guide. https://www.belkin.com/cn/support-article/?articleNum=3804 [2024]

[6]. Ubuntu. SynergyHowto. https://help.ubuntu.com/community/SynergyHowto. [2017-09-16]

[7]. ShareMouse. Share one Mouse and Keyboard with Multiple Computers. https://www.sharemouse.com [2024]

[8]. Palmér. Pynput. https://pynput.readthedocs.io/en/latest/index.html [2024-03-10]

[9]. Gvalkov. Pyevdev. https://python-evdev.readthedocs.io/en/latest/. [2024-03-08]

[10]. Cheshire. Multicast DNS. https://www.rfc-editor.org/rfc/rfc6762 [2013-2]

[11]. Pypi. PyQt5. https://pypi.org/project/PyQt5/. [2024-7-19]

[12]. Ron Rivest et. 1983. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems[J]. Communications of the ACM. 26: 96-99

[13]. Wayland. https://wayland.freedesktop.org/.

[14]. Sweigart. Pyperclip's documentation. https://pyperclip.readthedocs.io/en/latest [2014]

[15]. IETF Zeroconf. Zero Configuration Networking (Zeroconf). http://www.zeroconf.org [2024]

[16]. Python Software Foundation. tkinter.dnd — Drag and drop support. https://docs.python.org/3/library/tkinter.dnd.html [2024-07-31]

[17]. Linux community. The Linux Kernel. https://www.kernel.org/doc/html/latest/usb/usbip_protocol.html.