

远程桌面 概要设计说明书			
配置项编号	DRD-0V-2025	版本号	V1.0
文档密级	内部	编制部门	架构部
编制人	李昌泽	日期	2025 年 11 月 26 日
审核人	待定	日期	待定
批准人	待定	日期	待定

修订记录

序号	版本号	修订内容描述	修订日期	修订人	审核人	批准人
1	V1.0	首版	2025-11-26	李昌泽	待定	待定

目录

1. 概述	4
1.1. 目的	4
1.2. 术语说明	4
1.3. 参考资料	4
2. 系统设计	4
2.1. 设计原则	4
2.2. 需求分析	4
2.3. 主要模块	6
2.3.1. 客户端连接管理	7
2.3.2. 屏幕采集管理	7
2.3.3. 编码与传输管理	7
2.3.4. 输入事件管理	8
2.3.5. 连接安全管理	8
2.3.6. 服务重定向管理	8
2.3.7. 远程会话管理	8
2.3.8. 远程会话权限控制	9
2.3.9. 配置与隐私管理	9
2.3.10. 进程管理	9
2.3.11. 虚拟屏幕管理	10
2.3.12. RDP Seat 管理	10
2.3.13. UI 模块管理 (Greeter、控制中心、任务栏)	11
2.4. 关键接口设计	11
2.5. 关键流程	14
2.5.1. 桌面共享流程	14
2.5.2. 远程 Greeter 登录	15
2.5.3. 远程单点登录	16
2.5.4. 登录已存在远程会话用户	16
2.5.4.1. 远程 greeter 登录已经存在远程会话的用户	16
2.5.4.2. 远程单点登录已经存在远程会话的用户	17
2.6. 关键数据结构设计	17
2.7. 数据库设计	18
2.7.1. 数据库设计概述	18
2.7.2. 数据库信息	18
2.7.3. 数据库 ER 图设计	18
2.7.4. 数据库表结构	18
2.7.5. 表结构约束	18
2.7.6. 数据库视图	19
2.7.7. 存储过程	19
2.7.8. 触发器	19
2.7.9. 函数	19
2.7.10. 数据安全	19
2.8. 主要人机交互设计	19
3. 非功能性设计	19
3.1. 安全性	19
3.2. 性能	19

3.3. 可靠性	19
3.4. 易用性	20
3.5. 可用性	20
3.6. 兼容性	20
3.7. 韧性	20
3.8. 隐私	20
3.9. 可维护性	20
4. 部署与实施	20
5. 专利	21
6. 附录	22

1. 概述

1.1. 目的

- deepin-remote-desktop (简称 drd) 提供 Linux 上的现代 RDP 服务端，实现远程登录、桌面共享。本文件面向熟悉 GLib/FreeRDP 的工程师，说明系统设计原则、模块边界、关键流程及 LightDM 扩展，为后续详细设计、测试与部署提供统一基线。

1.2. 术语说明

- RDP: Remote Desktop Protocol, 本项目基于 FreeRDP 3.x 服务器栈。
- Rdpgfx: Graphics Pipeline 虚拟通道, 用于 Progressive/RFX 帧推送。
- NLA/CredSSP: 网络级认证, 控制凭据交付方式。
- LightDM SeatRDP: LightDM 中的远程 seat 实现。
- Routing Token: Server Redirection 使用的 msts cookie, 用于系统守护识别重连上下文。
- SSO: Single Sign-On, 即单点登录, 客户端直接配置服务端账户用户名和密码完成登录, 无需在 greeter 界面进行登录操作。
- PAM: Pluggable Authentication Modules, deepin 认证栈使用的可插拔框架。
- PDU: Protocol Data Unit, 指 RDP 会话在虚拟通道或主通道上传输的结构化数据 (如 Server Redirection PDU、Rdpgfx Frame PDU)。
- 桌面共享: RDP 客户端连接后直接复用当前已存在的本地图形会话, 客户端和服务端的画面一致;
- 远程登录: RDP 客户端连接后登录一个未在本地登录的用户, 创建一个远程图形会话;

1.3. 参考资料

1. FreeRDP 3.x API 文档。
2. LightDM Seat 与 DisplayManager 接口。
3. deepin systemd/DBus 规范。

2. 系统设计

2.1. 设计原则

- 使用 C 语言并基于 GLib 进行开发。
- RDP 协议栈依托 FreeRDP 3.x 库实现。
- 遵循 deepin/UOS systemd/DBus 规范。
- 遵循 SOLID: 监听、会话、媒体、系统守护、LightDM Seat 均以单一职责交付, 抽象层稳定, 具体实现可替换。
- 安全默认开启: TLS + NLA, System/Handover 重定向链路加密。

2.2. 需求分析

本系统主要解决以下关键需求:

1. 远程桌面功能需求

- 支持 X11 环境下的桌面共享。
- 支持 X11 环境下的远程 Greeter 登录。
- 支持 X11 环境下的远程单点登录。

2. 技术目标

- 支持主流 RDP 客户端连接。
- 支持多客户端并发连接。
- 支持远程会话可重入。
- 控制 CPU/内存等资源占用在合理范围内。

3. 使用场景分析

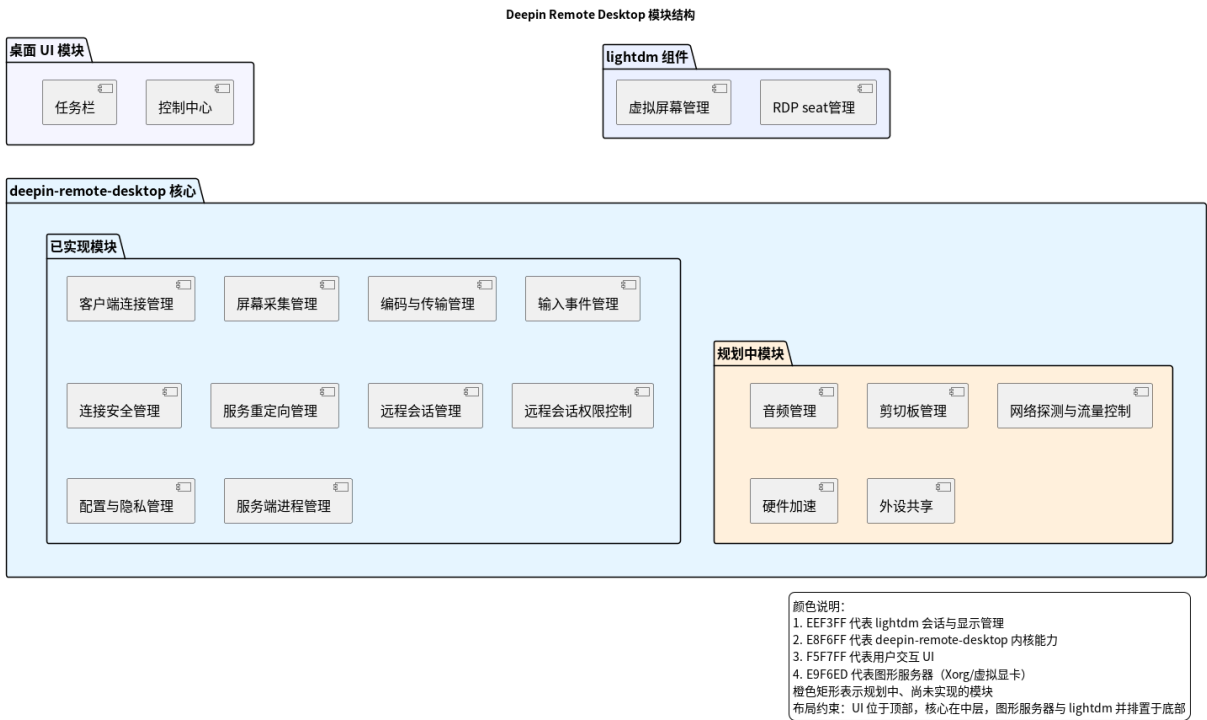
- TODO

4. 技术方案概述

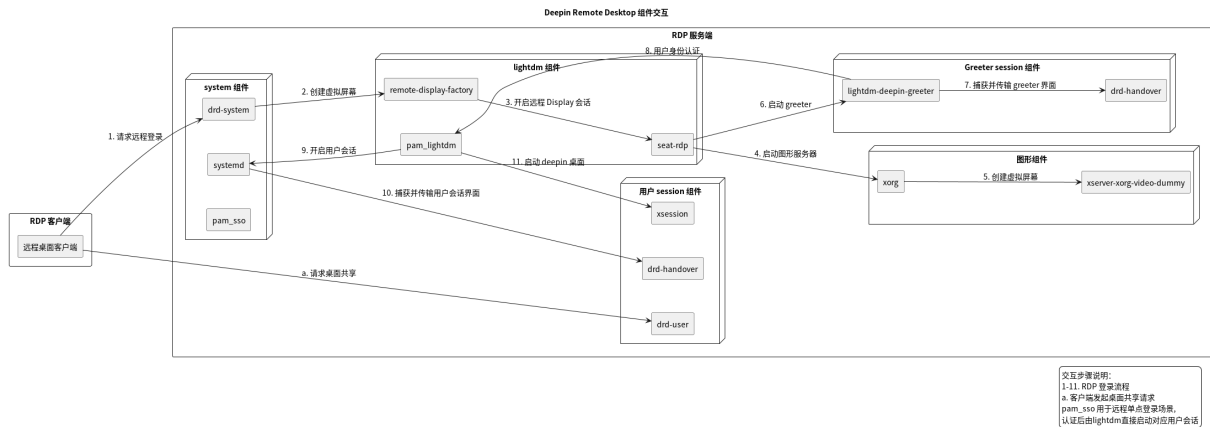
- 使用 XShm 与 XDamage 进行屏幕捕获。
- 采用 Rdpgfx Progressive 与 RemoteFX 两种方式进行编码与传输。
- 将画面拆分成 64×64 网格进行比对，仅更新变化区域以降低带宽占用。
- 通过 xserver-xorg-video-dummy 创建虚拟屏幕。
- 借助 LightDM 创建远程会话与单点登录会话。
- 使用 PAM 进行单点登录认证。
- 采用 RDP Server Redirection 策略，确保远程登录过程中客户端不断连并允许远程会话复用。
- 使用 XTest 模拟键鼠输入事件。

2.3. 主要模块

- 本系统主要包含两大模块：deepin-remote-desktop 约 10 个子模块，LightDM 约 2 个子模块。
- deepin-remote-desktop 的进程之间通过 DBus 通讯，deepin-remote-desktop 与 LightDM 及桌面 UI 组件之间同样通过 DBus 协作；LightDM 父子进程之间通过管道通信；deepin-remote-desktop 监听 3389 等网络端口与 RDP 客户端通讯。
- 本系统整体结构如下：



- 系统组件结构与组件交互如下：



- drd-system 为 deepin-remote-desktop --mode system 进程，作为 system 服务运行，使用 deepin-system-daemon 用户。
- drd-handover 为 deepin-remote-desktop --mode handover 进程，在登录会话和用户会话中按需启动。
- 远程登录依赖 drd-system 和 drd-handover 配合实现。
- drd-user 为 deepin-remote-desktop --mode user 进程，为桌面共享提供支撑。
- 远程登录和桌面共享在客户端连接、屏幕采集、编码传输、输入、安全以及配置与隐私管理等模块上完全复用，但除端口号外两者运行时不交叉。

2.3.1. 客户端连接管理

作用：

- 监听 RDP 客户端连接，并与客户端完成协商与握手。
- 对连接进行暂停、终止等管理。

机制：

- 通过 GSocketService 监听端口并注册 incoming 回调。
- 注册 freerdp_peer 回调完成 FreeRDP 配置、会话与上下文管理。
- 校验 routing token 以决定是否需要重定向本次连接。
- 未使用 freerdp_listener，因为该封装无法直接处理底层 fd。
- 在远程登录场景，可配置断开连接时自动注销当前用户。

2.3.2. 屏幕采集管理

作用：

- 持续采集桌面图像，向编码器提供统一帧源。

机制：

- 在独立线程订阅 XDamage 事件，并借助 XShm 共享内存复制像素，再以单帧阻塞队列交付最新画面。

2.3.3. 编码与传输管理

作用：

- 选择合适的编码格式并通过 Rdpgfx 或 SurfaceBits 推流。

机制：

- 启动 RDP Graphics Pipeline 虚拟通道。
- 虚拟通道就绪前使用 SurfaceBits + RemoteFX 进行编码与推流。

- 虚拟通道就绪后使用 RemoteFX Progressive (RLGR1) 进行编码与推流。
- TODO

2.3.4. 输入事件管理

作用：

- 将远程键鼠操作注入本地桌面，保持交互一致。

机制：

- 通过 FreeRDP 输入回调统一缩放坐标，再用 XTest 将事件注入 X11。

2.3.5. 连接安全管理

作用：

- 提供 TLS/NLA/PAM 的凭据认证方式，确保所有连接都在安全的信道中完成认证。

机制：

- 禁用纯 RDP Security。
- 桌面共享支持密码与免密两种方式：密码连接使用 NLA+TLS，凭据即 NLA 密码；免密连接采用 TLS+用户确认。
- 远程 Greeter 登录仅支持密码连接，使用 NLA+TLS，客户端展示服务端 Greeter 界面后由用户输入账户密码完成登录。
- 远程单点登录仅支持密码连接，采用 TLS+PAM 方案；客户端配置的凭据即服务端账户与密码，drd-system 进程获取后触发 pam_sso 认证，认证通过后向 LightDM 请求启动目标用户会话。
- 自动生成 TLS 证书和指纹，并支持周期性轮换。
- 支持自动生成 NLA 强密码。
- 将 NLA 密码存放于可信路径或设备。
- 服务重定向过程中使用随机密码和一次性 token。
 - handover 进程不会持有真实系统账户密码和 NLA 密码，可显著降低泄漏风险。
 - 每次重连都使用随机密码，不担心被截获。

2.3.6. 服务重定向管理

作用：

- 串联远程登录中从 Greeter 到用户会话的完整链路。
- 串联单点登录中 system 进程到 handover 进程的链路。
- 在整个链路中实现用户无感知的 RDP 服务端切换。

机制：

- drd-system 进程监听远程登录端口，并通过 DBus 信号协调整个服务重定向流程。
- 在 Greeter 会话和用户会话分别启动 drd-handover 进程承担屏幕捕获与传输，并基于 RDP Server Redirection 协议让客户端自动重连。
- 在重连阶段切换 RDP 服务端，例如从 Greeter handover 切换到用户会话 handover。
- drd-system 首次握手后，将 handover 密码与随机 routing_token 打包进 Server Redirection PDU 下发客户端；客户端按 token 第二次重连后，drd-system 将 accept 返回的 fd 通过 DBus 移交给 handover，由 handover 完成剩余握手并负责后续传输。
- 整个重定向过程中 routing_token 唯一且保持稳定。
- 单点登录遵循相同机制，将 system 进程收到的连接重定向到目标 handover 进程。

2.3.7. 远程会话管理

作用：

- 支持同时接收多个客户端连接请求。
- 确保客户端能够重复连接既有远程会话。

机制：

- 通过维护 routing token、client_id 以及 session_id 避免连接错配。
- routing token 与 client_id 一一对应。
- handover session: 由 drd-system DBus 服务创建的对象，可通过 client_id 与 session_id 快速定位。
- remote session: 由 LightDM DBus 服务创建的对象。
- handover session 与 remote session 一一对应，单个远程会话仅包含这一对对象。
- 根据 drd-handover 进程的 session_id 属性，可准确分配 handover session。
- 客户端重定向携带的 routing token 会被转换成 client_id，进而找到对应 handover session 并对其执行所需操作。

2.3.8. 远程会话权限控制

作用：

- 对系统中的远程会话的 polkit 鉴权进行统一管理。

机制：

- 使用 polkit 的 pkla 与 rules 机制覆盖远程会话的鉴权策略。

2.3.9. 配置与隐私管理

作用：

- 统一管理 drd-user/drd-system/drd-handover 三个进程的运行配置。
- 保护用户名、密码与登录密钥等敏感字段。

机制：

- 使用 INI 格式存储配置。
- 支持 drop-in 配置机制。
- 默认配置存放在 /usr/share/deepin-remote-desktop/conf.d 目录。
- 支持加载 /etc/deepin-remote-desktop/conf.d 下的配置覆写默认值。
- drd-user 进程需要加载 ~/.config/deepin-remote-desktop/conf.d 覆盖默认配置。
- drd-user 动态配置保存在 ~/.config/deepin-remote-desktop/conf.d 中。
- drd-system 动态配置保存在 /var/lib/deepin-remote-desktop/conf.d 中。
- 在日志中用** 掩码隐藏敏感内容。
- 提供 CLI 命令修改运行配置：
 - 端口号；
 - TLS 证书和 Key 文件路径；
 - 配置文件；
 - 编码器；
 - NLA 用户名和密码；
 - 运行模式；

2.3.10. 进程管理

作用：

- 对 drd-user/drd-system/drd-handover 三个进程的生命周期与权限进行统一管理。

机制：

- drd-user 进程由 systemd user 服务管理，默认不自启动；开启桌面共享后在 graphical-session-pre.target 中启动，断开连接不会立即退出。
- drd-system 进程由 systemd system 服务管理，默认不自启动；开启远程登录后在 graphical.target 中启动，运行用户为 deepin-system-daemon，其他服务满足 DBus 安全规范要求，并作为守护进程在后台持续监听端口。
- drd-handover 进程采用两套管理方案：
 - 登录会话：在 /etc/deepin/greeters.d 中放置 11-deepin-remote-desktop-handover 脚本，于 Greeter 启动前拉起。
 - 用户会话：由 systemd user 服务托管，默认不自启动；开启远程登录后在 graphical-session-pre.target 中启动，如无待接管连接会立即退出。
 - 通常在发出 Server Redirection PDU 后退出，断开连接时保持运行，远程会话结束由 session scope 统一回收。

2.3.11. 虚拟屏幕管理

作用：

- 管理虚拟屏幕的生命周期和 DISPLAY。

机制：

- LightDM 使用 xserver-xorg-video-dummy 创建虚拟屏幕，配置示例如下：

```
Section "Device"
    Identifier "dummy_device"
    Driver      "dummy"
    VideoRam    256000
EndSection

Section "Monitor"
    Identifier "dummy_monitor"
    HorizSync 30-80
    VertRefresh 60
    # 也可以添加 Modeline 指定自定义分辨率
    Modeline "1920x1080" 172.80 1920 2048 2248 2576 1080 1083 1088 1120 -hsync
    +vsync
EndSection

Section "Screen"
    Identifier "dummy_screen"
    Device      "dummy_device"
    Monitor     "dummy_monitor"
    DefaultDepth 24
    SubSection "Display"
        Depth 24
        Modes "1920x1080"
    EndSubSection
EndSection
```

- 对 DISPLAY 进行管理，默认从 100 开始。

2.3.12. RDP Seat 管理

作用：

- 对 RDP 远程会话的进程与环境变量进行统一管理。

机制：

- 创建图形服务器。
- 设置环境变量：
 - 设置 `deepin-remote-session=TRUE`，便于 Greeter 和用户会话进程识别远程会话信息（可从 `login1` 获取）。
 - 设置 `XDG_SEAT` 为空，open session 时即可清空 seat 属性（`polkitd` 会据此判断是否为远程会话）。
- 创建 Greeter 会话。
- 启动用户会话。

2.3.13. UI 模块管理（Greeter、控制中心、任务栏）**作用：**

- 为用户展示远程连接状态并提供操作入口。

机制：

- 根据环境变量对相关 UI 进行屏蔽或调整，保证本地与远程体验一致。

2.4. 关键接口设计

- CLI 支持以下运行参数：

<code>-p, --port=PORT</code>	Bind port (default 3390 unless config overrides)
<code>--cert=FILE</code>	TLS certificate PEM path
<code>--key=FILE</code>	TLS private key PEM path
<code>-c, --config=FILE</code>	Configuration file path (ini)
<code>--encoder=MODE</code>	Encoder mode (raw rfx)
<code>--nla-username=USER</code>	NLA username for static mode
<code>--nla-password=PASS</code>	NLA password for static mode
<code>--enable-nla</code>	Force enable NLA regardless of config
<code>--disable-nla</code>	Disable NLA and use TLS+PAM single sign-on
<code>--mode=MODE</code>	Runtime mode (user system handover)

- 配置文件支持如下参数：

```
## 深度远程桌面完整示例配置，覆盖所有支持的字段。
## 生产环境请根据实际网络、认证方式调整。

[server]
# 监听端口，默认为 3389
port=3389

[tls]
# 证书与私钥支持相对路径（相对当前配置文件目录）
certificate=/usr/share/deepin-remote-desktop/certs/server.crt
private_key=/usr/share/deepin-remote-desktop/certs/server.key

[encoding]
# 编码模式：raw 或 rfx（默认 rfx）
mode=rfx
# 是否启用帧间差分，默认为 true
enable_diff=true
```

```
[auth]
# NLA 凭据，仅在启用 NLA 时使用
username=uos
password=1
# 是否启用 NLA，设置为 false 可改用 RDP SSO
enable_nla=true
# 可选：覆盖默认 PAM Service 名称
pam_service=deepin-remote-ss0

[service]
# 运行模式：user、system 或 handover
runtime_mode=handover
# 当启用 rdp_sso 时将自动禁用 NLA
rdp_sso=false
```

- LightDM 新增 DBus 接口：org.deepin.DisplayManager
- TODO 需要支持多屏参数

```
<!DOCTYPE node PUBLIC "-//freedesktop//DTD D-BUS Object Introspection 1.0//EN"
"http://www.freedesktop.org/standards/dbus/1.0/introspect.dtd">
<node>
  <interface name="org.deepin.DisplayManager.RemoteDisplayFactory.Session">
    <property name="UserName" type="s" access="read"/>
    <property name="Address" type="s" access="read"/>
    <property name="ClientId" type="s" access="read"/>
    <property name="SessionId" type="s" access="read"/>
  </interface>
  <interface name="org.deepin.DisplayManager.RemoteDisplayFactory">
    <method name="CreateRemoteGreeterDisplay">
      <arg name="ClientId" direction="in" type="u"/>
      <arg name="Width" direction="in" type="u"/>
      <arg name="Height" direction="in" type="u"/>
      <arg name="Address" direction="in" type="s"/>
      <arg name="SessionPath" direction="out" type="o"/>
    </method>
    <method name="CreateSingleLogonSession">
      <annotation name="org.gtk.GDBus.C.UnixFD" value="true" />
      <arg name="ClientId" direction="in" type="y"/>
      <arg name="Width" direction="in" type="u"/>
      <arg name="Height" direction="in" type="u"/>
      <arg name="PipeFd" direction="in" type="h"/>
      <arg name="Address" direction="in" type="s"/>
      <arg name="SessionPath" direction="out" type="o"/>
    </method>
  </interface>
</node>
```

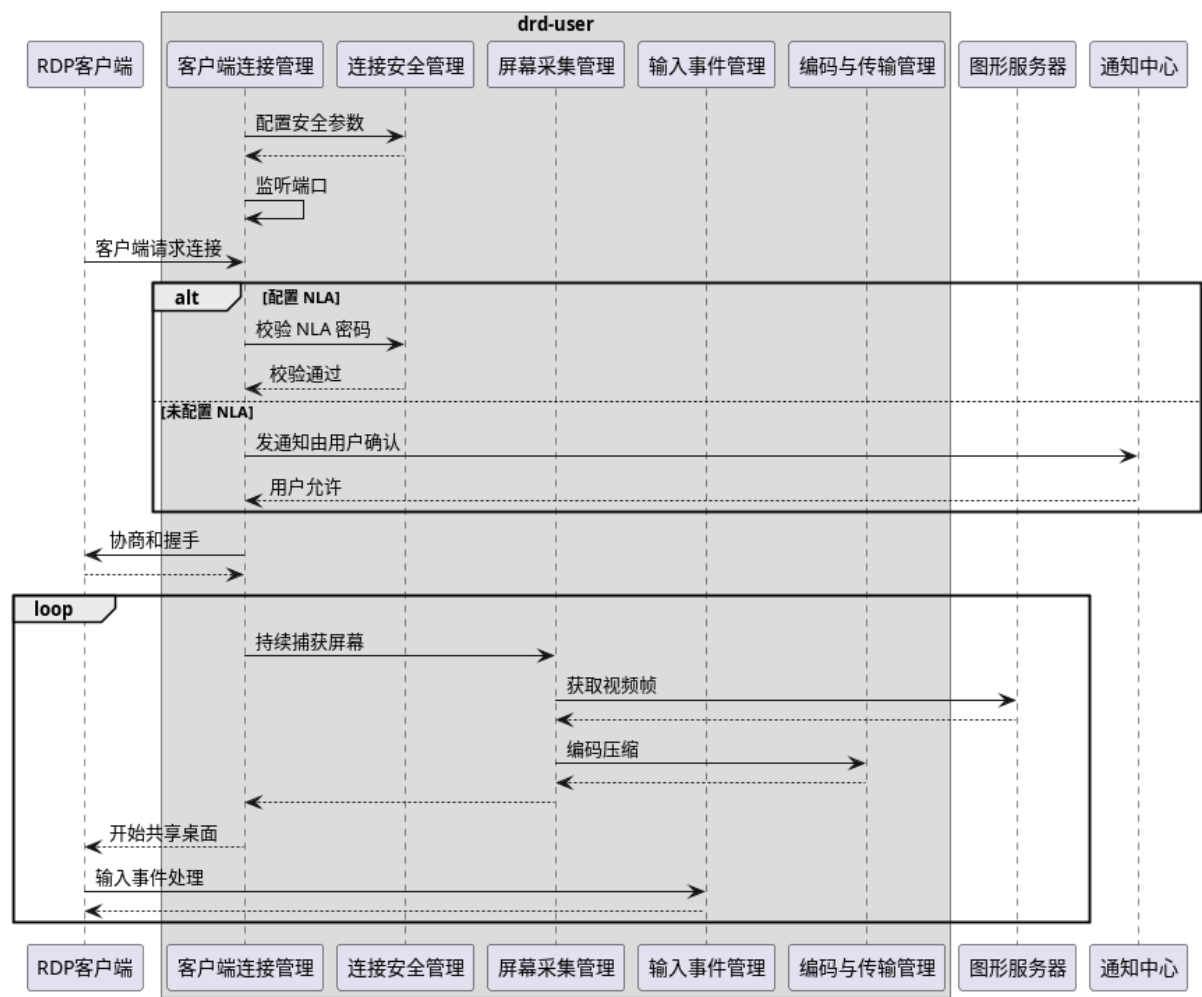
- deepin-remote-desktop 新增 DBus 接口：org.deepin.RemoteDesktop

```
<!DOCTYPE node PUBLIC
'-//freedesktop//DTD D-BUS Object Introspection 1.0//EN'
'http://www.freedesktop.org/standards/dbus/1.0/introspect.dtd'>
```

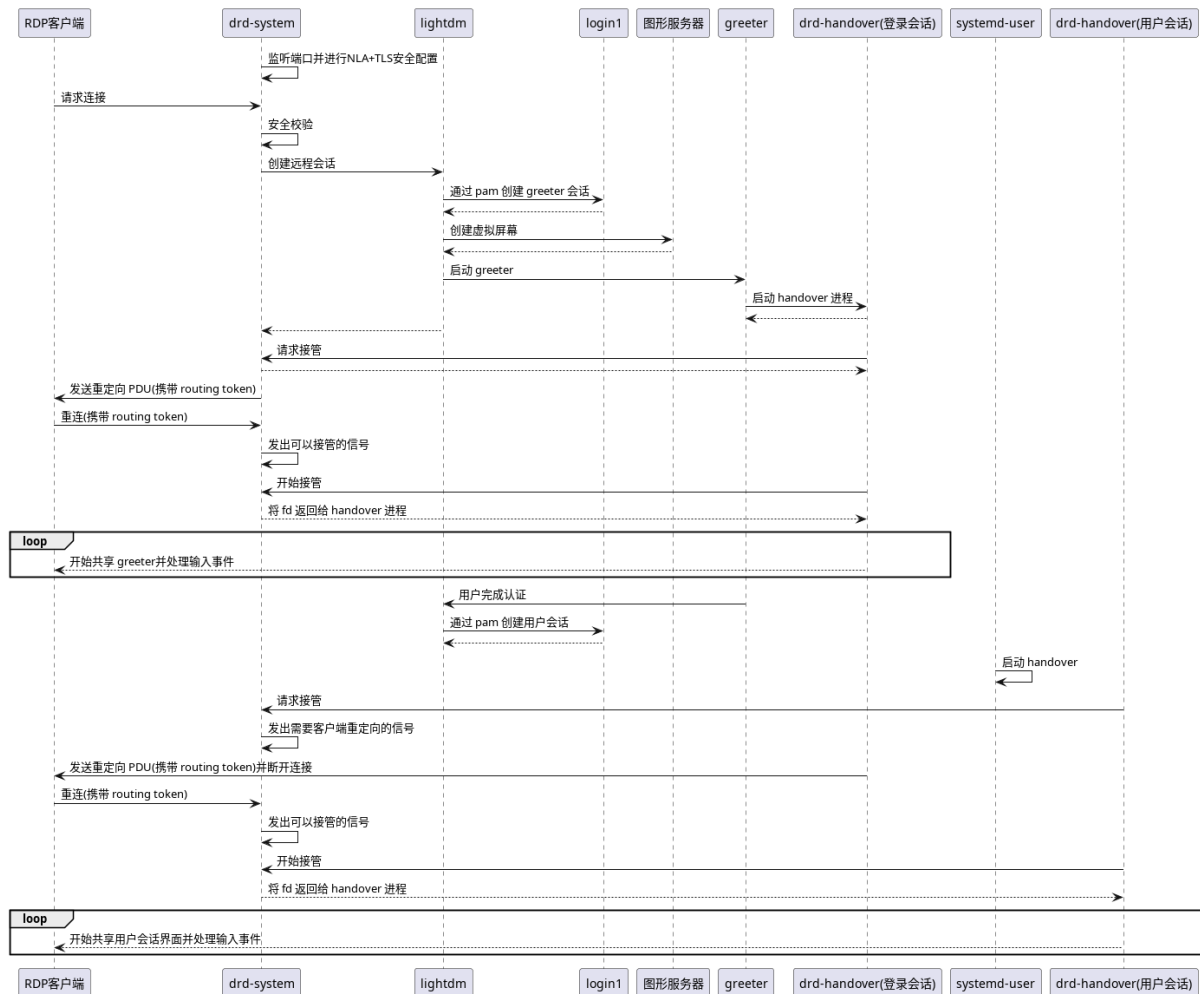
```
<node>
  <interface name="org.deepin.RemoteDesktop.Rdp.Server">
    <property name="Enabled" type="b" access="read" />
    <property name="Port" type="i" access="read" />
    <property name="TlsCert" type="s" access="read" />
    <property name="TlsFingerprint" type="s" access="read" />
    <property name="TlsKey" type="s" access="read" />
    <property name="ViewOnly" type="b" access="read" />
  </interface>
  <interface name="org.deepin.RemoteDesktop.Rdp.Dispatcher">
    <method name="RequestHandover">
      <arg name="HandoverPath" direction="out" type="o" />
    </method>
  </interface>
  <interface name="org.deepin.RemoteDesktop.Rdp.Handover">
    <method name="StartHandover">
      <arg name="Username" direction="in" type="s" />
      <arg name="Password" direction="in" type="s" />
      <arg name="Certificate" direction="out" type="s" />
      <arg name="Key" direction="out" type="s" />
    </method>
    <signal name="RedirectClient">
      <arg name="RoutingToken" type="s" />
      <arg name="Username" type="s" />
      <arg name="Password" type="s" />
    </signal>
    <signal name="TakeClientReady">
    </signal>
    <method name="TakeClient">
      <annotation name="org.gtk.GDBus.C.UnixFD" value="true" />
      <arg name="Fd" direction="out" type="h" />
    </method>
    <signal name="RestartHandover" />
  </interface>
</node>
```

2.5. 关键流程

2.5.1. 桌面共享流程



2.5.2. 远程 Greeter 登录



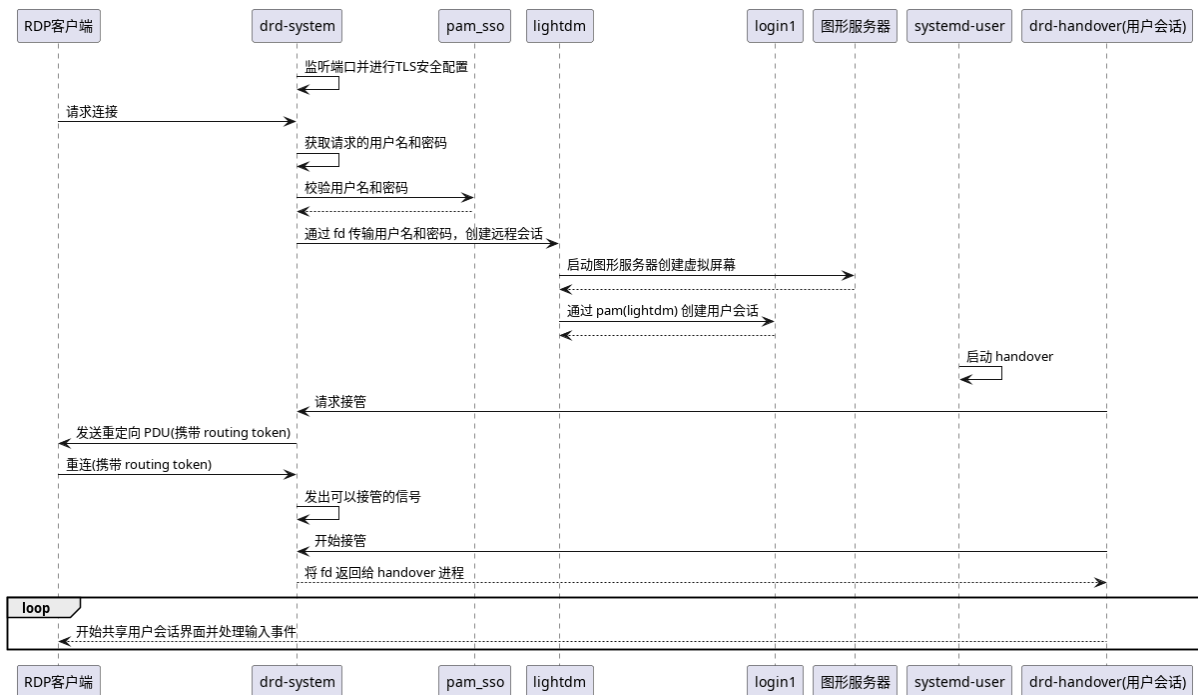
- 屏幕共享相关步骤与 2.5.1 一致，本节不再重复。

细节描述

- 客户端每次请求远程登录连接时, drd-system 进程都会生成一个 UUID 作为 client_id, 并据此生成 routing token。
- drd-system 进程会创建并导出 handover session DBus 对象, 该对象持有 client_id, 仅处理对应客户端的事件。
- drd-system 调用 LightDM 接口创建 remote session, 该对象包含 client_id 与 session_id; session_id 与 login1 的 session id (open_session 之后更新属性) 保持同步, handover session 在监听属性变更后也会更新为同一 session_id。
- Greeter handover 调用 drd-system 的 RequestHandover, 根据进程的 session_id 获得对应 handover session 的 DBus path 并开始监听。
- Greeter handover 调用 handover session 的 StartHandover, system 进程向客户端发送携带 routing token (即 client_id) 的 Redirect PDU。
- 客户端再次连接时 (携带 routing token), system 依据 routing token 还原 client_id, 找到对应 handover session, 发送 TakeClientReady 信号, greeter handover 随后调用 TakeClient 接管新连接的 fd 并完成握手。
- 在 Greeter 界面完成认证之后, LightDM 开启新的用户会话, 并将原有 remote session 的 session_id 更新为新用户会话的 ID。

- system 进程监听到 remote session 的 session_id 变化信号后，会同步更新对应 client_id 的 session_id。
- 用户会话的 handover 进程启动后，再次调用 system 的 RequestHandover，并根据自身的 session_id 获得对应 handover session 的 Dbus path（即刚刚更新 session_id 的 handover session）。
- handover 进程调用 handover session 的 StartHandover，system 进程发送 Redirect 信号，Greeter handover 收到后发送 Redirect PDU 并退出。
- 客户端再次连接时，system 根据新的 client_id 找到对应 handover session，发出 TakeClientReady 信号，随后用户 handover 调用 TakeClient 接管新连接的 fd 并完成握手。

2.5.3. 远程单点登录



- 屏幕共享相关处理与 2.5.1 中一致，在此不再重复。
- 通过 session_id 的比对，可以正确把 handover session 分配给对应的 drd-handover 进程。

2.5.4. 登录已存在远程会话用户

2.5.4.1. 远程 greeter 登录已经存在远程会话的用户

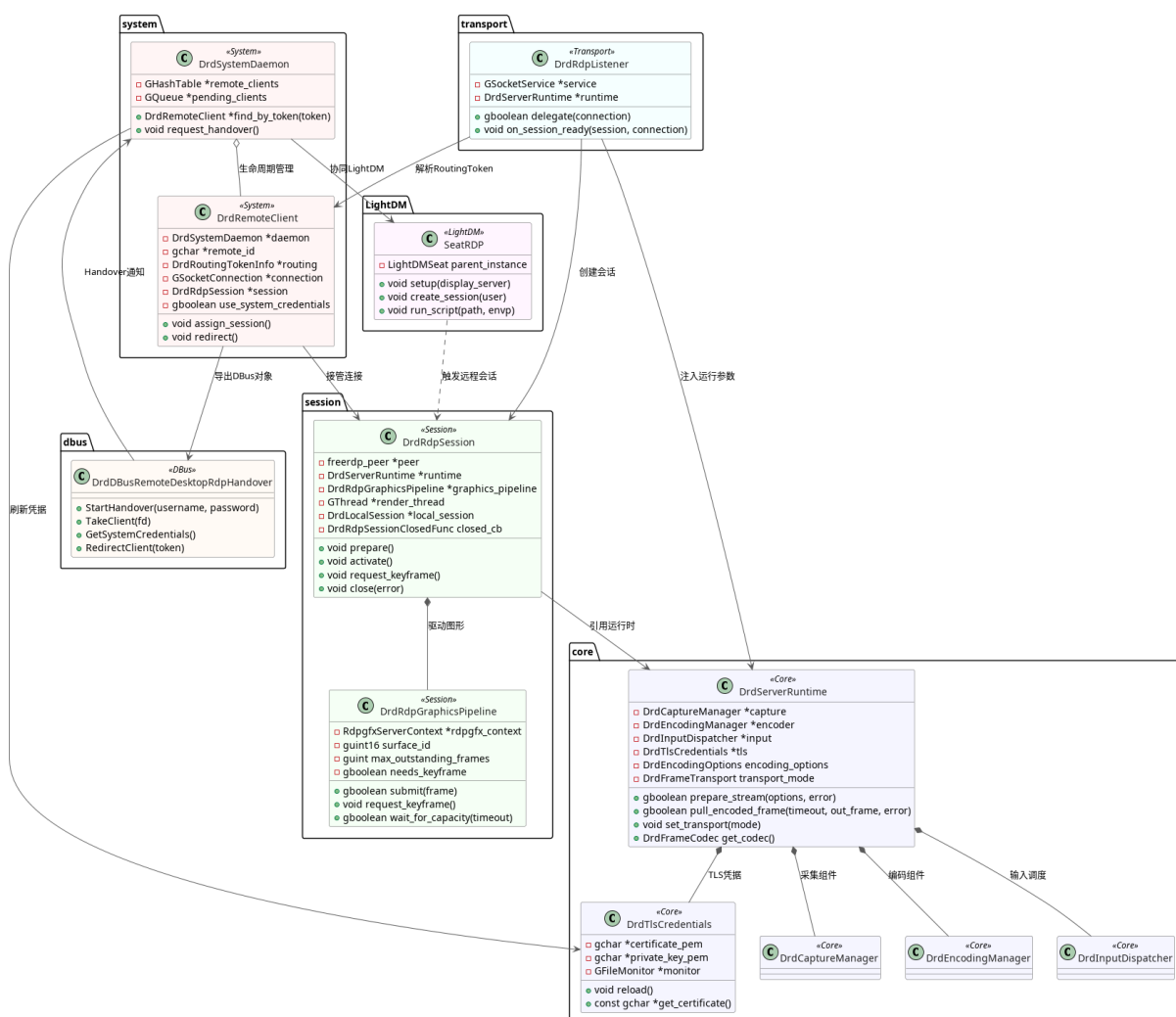
- Greeter 界面认证之前的流程与 2.5.2 相同，不再赘述。
- 新建的 Greeter remote session 持有新的 client_id 和 session_id (Greeter)，原 remote session 持有旧的 client_id 和正在运行的 session_id。
- 在 Greeter 界面完成认证后，若检测到用户已存在远程会话，LightDM 不再 open session 和 run session。
- 根据 session_id 找到原 remote session，并把新的 Greeter remote session 上报的 client_id 写回原 remote session。
- system 进程监听到 client_id 变化后，对应 handover session 同样更新 client_id，并发出 RestartHandover 信号，handover 进程重新调用 handover session 的 StartHandover。

- 根据该 client_id 找到 Greeter handover session, 发送 RedirectClient 信号, Greeter handover 接收到信号后发送 Redirect PDU 并退出。
- 再次连接时, drd-system 进程依据 client_id 找到 handover session, 发出 TakeClientReady 信号, 用户 handover 进程调用 TakeClient 接管新连接的 fd 并完成握手。

2.5.4.2. 远程单点登录已经存在远程会话的用户

- LightDM 在判断远程单点登录用户已存在远程会话时, 不再 open session 和 run session。
- 根据既有 session_id 找到 remote session, 并把新的 client_id 写入该对象。
- system 进程监听到 client_id 变化后, 对应 handover session 同样更新 client_id, 并发 RestartHandover 信号, drd-handover 进程重新调用 handover session 的 StartHandover。
- drd-system 进程向客户端发送 Redirect PDU。
- 客户端再次连接时, drd-system 依据 client_id 找到 handover session, 发出 TakeClientReady 信号, 然后用户 handover 进程调用 TakeClient 接管新连接的 fd 完成握手。

2.6. 关键数据结构设计



- **DrdServerRuntime**: 核心运行时对象, 集中持有 `DrdCaptureManager`、`DrdEncodingManager`、`DrdInputDispatcher` 与 `DrdTlsCredentials`, 并保留当前 `DrdEncodingOptions` 与 `DrdFrameTransport`。`drd_server_runtime_prepare_stream()` 会原子化启动编码/输入/采集链路, 失败时按相反顺序回滚;
`drd_server_runtime_pull_encoded_frame()` 负责串行化 “等待捕获帧 → 以当前 codec 编码 → 返回 `DrdEncodedFrame`”, 避免额外缓存造成延迟;
`drd_server_runtime_set_transport()` 会在切换传输模式时强制刷新关键帧。
- **DrdRdpSession**: 面向单个 RDP 客户端的 GLib 对象, 封装 `freerdp_peer`、事件线程、`renderer` 线程与 `DrdRdpGraphicsPipeline`。会话 `Activate` 时启动 `renderer` 线程读取 `runtime` 并驱动图形管线; `DrdLocalSession` 和 `passive_mode` 字段确保同一个进程下既能复用现有用户桌面, 也能在 `system` 模式下仅转发 TLS/NLA/PAM。
`drd_rdp_session_set_runtime()` 会立即尝试初始化图形通道, 确保连接时序稳定。
- **DrdRdpGraphicsPipeline**: 包装 `Rdpgfx Server Context`, 并通过 `GMutex + GCond` 管理 `outstanding_frames` 背压。`needs_keyframe` 标记在 `Rdpgfx Reset`/失败后禁止增量帧;
`max_outstanding_frames` 显式限制同时在路上的帧数,
`drd_rdp_graphics_pipeline_wait_for_capacity()` 坚持等到客户端发送 `ACK` 才允许新帧, 保证编码速率与客户端解码能力一致。
- **DrdRemoteClient (system 守护)**: 以 `remote_id` 和 `routing_token` 为索引, 跟踪 `GSocketConnection`、关联的 `DrdRdpSession`、`DBus handover skeleton` 以及 `use_system_credentials` 等运行状态。`assigned` 与 `handover_count` 帮助 `system` 守护判断是继续排队还是立即交给 `handover` 进程, 保证多段重定向不丢连接。
- **DrdTlsCredentials**: 负责加载、缓存并热更新 TLS PEM 证书/私钥, 向 `runtime`、`listener` 与 `handover` 暴露 `rdpCertificate`、`rdpPrivateKey` 以及原始 PEM 文本; 当路径发生变化时可以在不中断会话的前提下刷新内存快照, 满足 “安全默认开启” 的约束。
- **LightDM SeatRDP**: 在 `SeatRDPClass` 中覆写 `setup()`、`create_display_server()`、`create_session()` 与 `run_script()`, 用于创建 `remote seat`、设置 `XDG_SEAT` 等环境变量, 并通过 `RequestHandover` 与 `system/handover` 交互, 实现远程登录、`greeter` 登录与单点登录的统一入口。

2.7. 数据库设计

2.7.1. 数据库设计概述

系统不引入数据库, 配置来自 `INI + CLI`, 临时状态驻留内存, `LightDM` 通过 `DBus` 交流; 若未来需要持久化审计, 将单独设计 `KV/SQL`。

2.7.2. 数据库信息

当前无数据库实例; 部署中无需维护数据库连接池或权限。

2.7.3. 数据库 ER 图设计

不适用。

2.7.4. 数据库表结构

不适用。

2.7.5. 表结构约束

不适用。

2.7.6. 数据库视图

不适用。

2.7.7. 存储过程

不适用。

2.7.8. 触发器

不适用。

2.7.9. 函数

不适用。

2.7.10. 数据安全

未存储数据库数据；凭据通过 TLS/NLA/PAM 即时处理，SAM 文件使用后删除。

2.8. 主要人机交互设计

- deepin-remote-desktop 支持通过命令行工具进行配置与运行。
- 控制中心可配置桌面共享与远程登录。
- 远程登录时在 Greeter 界面进行用户选择和登录。
- 桌面共享时在任务栏提供状态提示。

3. 非功能性设计

3.1. 安全性

- 默认启用 TLS + NLA，禁止 RDP Security，SAM 文件按连接生成一次性凭据后即清理。
- 敏感信息在 DBus 中使用 fd 进行传输。
- 日志中不输出用户敏感信息。
- drd-system 进程的运行用户为 deepin-system-daemon（非 root），并遵循 DBus/systemd 安全规范配置相关参数。

3.2. 性能

- 事件驱动的帧采集：订阅 XDamage，只有检测到脏矩形才触发 XShm 拷贝，并使用只保留一帧的 `DrdFrameQueue` 把最新像素交给 runtime，避免积压造成额外内存占用。
- 单帧编码管线：在 renderer 线程内直接完成编码，确保 `freerdp_peer` 与 capture 线程之间只有一次 memcpy。
- 瓦片差分 + codec 自适应：编码器默认生成 64×64 tile 的差分区域，与 Rdpgfx Progressive/RFX 阶段性数据完全对齐，通过哈希 + 按需逐行比较的方式避开整帧 memcmp，减少冗余编码。
- 背压与批次控制：精准控制推流速率，只有客户端回传 FrameAcknowledge 才继续发送，杜绝“编码完成却因未 ACK 被丢弃”的浪费。SurfaceBits 回退策略会依据客户端的 `negotiated_max_payload` 拆包，遵循 mstsc/FreeRDP 的最优 payload。
- 输入/事件路径：输入注入不绑定在 renderer 线程上，采用独立线程与队列，实时更新缩放矩阵，保证网络延迟变化不影响本地事件刷新。

3.3. 可靠性

- 重定向与回收：system/handover 依赖 `routing_token` → `remote_id` 的互逆映射维护 `DrdRemoteClient`，即便 handover 崩溃也可借助 token 在 system 端重新登记；

`RestartHandover` 和 `RedirectClient` 信号使得同一客户端可以在 `greeter/user` 会话间无缝移动。

- 线程生命周期管理：`drd_rdp_session_stop_event_thread()`、`drd_rdp_session_disable_graphics_pipeline()` 在连接关闭时先停止事件线程、再释放 pipeline，确保不会引用失效的 `freerdp_peer`；`DrdServerRuntime` 在 `stop()` 中按 `capture → encoder → input` 的顺序清理资源，确保 X11 fd、共享内存与编码上下文都能正确落地。

3.4. 易用性

- Meson 一键构建/测试/安装；`meson setup build && meson compile -C build`。
- 配置文件遵循分段结构，适配 config management。
- CLI 带有 `--help` 描述，日志统一格式方便筛查。
- LightDM SeatRDP 直接复用现有 Greeter，人机交互一致。

3.5. 可用性

- systemd unit (`system/handover/user`) 提供自动重启与日志整合。
- DBus 接口支持 `RestartHandover`、`RedirectClient`，可由桌面环境主动恢复。

3.6. 兼容性

- FreeRDP 3.x/mstsc/Remmina 均可连接；SurfaceBits Raw 回退兼容旧客户端。
- 输入层支持扩展扫描码、AltGr、滚轮缩放；LightDM SeatRDP 兼容现有 `logind/polkit`。
- 配置文件遵循标准 INI，可被现有管理工具处理。

3.7. 韧性

- Rdpgfx 超时/ACK 丢失自动降级 Raw，确保画面可恢复。
- System/Handover crash 后可通过 systemd 重启 + routing token 缓存恢复连接。

3.8. 隐私

- NLA 凭据、SAM 文件与一次性密码使用完立即擦除；日志严禁输出明文密码。
- TLS 证书路径配置可指向受管密钥仓。
- 将 NLA 密码存放于可信路径或设备。
- 服务重定向过程中使用随机密码和一次性 token。
 - handover 进程不会持有真实系统账户密码和 NLA 密码，可降低泄漏风险。
 - 每次重连都使用随机密码，不担心被截获。
- LightDM SeatRDP 通过 `PAM_RHOST` 告警潜在来源，便于审计。

3.9. 可维护性

- 目录分层明确 (`core/capture/encoding/input/security/session/system/transport/utils/doc/upstream`)，architecture 与本文档同步更新。
- `.codex/plan` 管理任务，`doc/changelog.md` 记录所有文档/代码变更。
- 代码规范需遵循《deepin 开源风格指南》。
- 日志规范应参考《正确使用日志记录》。

4. 部署与实施

1. 依赖：

```
sudo mk-build-deps --install # 安装依赖
```

2. 构建:

```
dpkg-buildpackage -us -uc -nc
```

3. 安装:

```
sudo dpkg -i ./*.deb
```

4. 关键文件路径:

```
├─ etc
│   ├── dbus-1
│   │   └─ system.d
│   │       └─ org.deepin.RemoteDesktop.conf
│   └─ deepin
│       └─ greeters.d
│           └─ 11-deepin-remote-desktop-handover
├─ usr
│   ├── bin
│   │   └─ deepin-remote-desktop
│   ├── lib
│   │   └─ systemd
│   │       ├── system
│   │       │   └─ deepin-remote-desktop-system.service
│   │       └─ user
│   │           ├── deepin-remote-desktop-handover.service
│   │           └─ deepin-remote-desktop-user.service
│   └─ share
│       ├── deepin-remote-desktop
│       │   ├── certs
│       │   │   ├── server.crt
│       │   │   └─ server.key
│       │   └─ config.d
│       │       ├── default-handover.ini
│       │       ├── default-pam-system.ini
│       │       ├── default-system.ini
│       │       ├── default-user.ini
│       │       └─ full-example.ini
│       └─ doc
│           └─ deepin-remote-desktop
│               ├── architecture.md.gz
│               ├── changelog.Debian.gz
│               ├── changelog.md.gz
│               └─ README.md.gz
```

5. 专利

- 当前未计划申请专利，若未来涉及编码/安全算法专利将单独立项评估。
- 以目前掌握的现状，在已知 Linux 发行版中，本系统是唯一一个集成了 X11 桌面共享、X11 远程 Greeter 登录与 X11 远程单点登录的方案。

6. 附录

- 暂无