# MK 远程桌面

## 需求概述

通过在bd产品上使用UI自动化测试工具，测试人员宣称发现了上百个手工测试无法发现的bugs，因此希望在mk产品也能支持UI自动化测试。考虑到自动化测试已有成熟的流程和部分PC软件的支持（录制、回放、随机回放、截屏比较等），在MK上提供自动化测试功能的需求转化为把MK屏幕（显示+输入）重定向到PC软件的功能；即相当于把MK作为服务器，把PC做客户机，在PC上远程操作MK。

## 显示部分

### 方案A：分别传递bs+wv的脏区域和像素（PAD\_BW: Pixels and Dirty）

MK上报脏区域的像素和几何信息，PC软件将这些信息贴图，完成显示。PC软件将鼠标、键盘消息反向发给MK，MK将这些消息翻译成Qt可识别的输入事件并投递到Qt的窗口系统。方案A本质上是在合成器的后端为本地Screen（对于BD，是xcb window；对于cms2，是windows window；对于MK，是lcd显存）并联1个远程Screen的代理（Remote Screen Proxy），本地Screen负责MK主机上的显示，远程代理负责把相同的信息（dirty region+pixels）打包发往PC。注意坐标始终是相对Screen的。此方法的好处是MK端和PC端的实现都非常简单。MK端的工作如前所述；在PC端，只需将Remote screen Stub上报的像素按同时上报的region信息贴图即可（MK Screen和Remote Screen Client做图像伸缩映射）。

当像素更新数据量较大时，可能会产生性能问题（如果压缩，cpu处理能力极其有限；如果不压缩，则网络带宽占用过高）。以600\*800@32bpp区域内滚动（假定FPS=10）为例，每秒纯像素的数据量为600\*800\*4\*10=18MB，已超过100M eth的理论带宽。



### 方案A’，直接传递fb中的传递脏区域和像素(PAD\_FB)



为方案A的变种，直接读取fb中脏区域的像素并传送，不分开处理波形和界面。

由于cpu性能的原因，压缩的可能性不大。[考虑简单的压缩方法]

数据量估计：

|  |  |  |
| --- | --- | --- |
|  | 带宽 |  |
| 波形 | 1.9MB/s | 20\*1000\*4@25Hz |
| 参数(按半屏算) | 3.9MB/s | 1920\*1080\*4\*0.5@1Hz |
| 菜单(按2/3屏算) | 5.3MB/次 | 1920\*1080\*4\*.667 |
| 滚动(按1/5屏,10FPS算) | 15.8MB/s | 1920\*1080\*4\*.2@10Hz |
| 光标(16x16 ARGB,25FPS) | 0.02MB/s | 16\*16\*4@25Hz |

### 方案B，传送绘制命令（UPE: Under Paint Engine）

PC软件将这些命令翻译成在PC上可以执行的绘图命令，完成显示。输入方法同方案A。

注意涉及到绘制命令的绘制目标有3类：backingstore（对应来自ui的绘制），waveimage（对应来自波形的绘制），以及cursor（对应光标）。



未精确计算数据量。

### 方案B’，传送绘制命令( IPE: In Paint Engine)

方案B的变种，在paint engine 接口层重定向，而非之下。

FontEngine

Font

未精确计算数据量。

### 显示方案对比

|  |  |  |  |
| --- | --- | --- | --- |
|  | PAD(BW/FB) | UPE | IPE |
| 同步的对象数量 | 无 | 较少 | 最多（image, rect, pixmap, path, …） |
| 传递的参数数量 | 最少 | 较少 | 最多 |
| 同步的全局配置 | 最少 | 少 | 少 |
| 同步的外部资源 | 无 | 无(span) | 字体(MK和PC必须使用相同的字体引擎和字体) |
| **字体渲染效率** | **低（像素）** | **较低（像素或Span）** | **高（Glyph）** |
| 图片渲染(动态、静态、Pattern) | 像素 | 像素(和字体不同,qt没有保留静态图片的文件信息) | 像素(和字体不同,qt没有保留静态图片的文件信息) |
| 几何渲染 | 像素 | Span/坐标+少量状态 | 坐标+较多状态 |
| 更新频率 | 最少 | 较多 | **最多** |
| qt改动量 | 最少 | 较多(drawhelper + rasterizer + private rasterize) | 最多(paint engine, object streaming, font engine, font resource) |
| 传递的波形数据 | 像素 | 像素 | 像素 |
| 综合性能预计 | 最差 | 较好 | 较好 |

## 输入部分

为复用qt中已有的输入部分的实现，对于来自远程的输入事件，仍沿用inpugmgr events的格式。

为简化链路拓扑，只维护Receiver与Frontend（Qt）之间的连接，Receiver与inputmgr无直接连接。

由于qt的input插件只接收InputMgr的事件及格式，因此来自PC的事件先经由Qt接收，翻译成InputMgr可识别的格式，并调用已有的（BD）vInputSendXXX发送给InputMgr进程，此时Qt充当虚拟输入设备（类似x11vnc的作用）；InputMgr进程收到事件后按原方法处理后，再次发送回Qt（此时由input插件接收并处理），后续流程和处理本地输入完全一样。

只需在Qt中增加1个接收PC输入事件并转发给InputMgr进程的子模块，其余部分都不必改动。InputMgr无需任何改动。



## 概要设计

### 方案B

Qt 支持层

1. 转发Primitive（for mk）
2. 同步全局配置（for mk）: device, format, …
3. 同步状态（for mk）: painter state, pen, brush, matrix, …
4. 接收输入（for mk）
5. Paint Engine On Widget Tunnel（for PC）
6. 转发 compositor command

Receiver(for PC)

1. PC窗口（the same as bs）
2. 捕获转发鼠标
3. 调用 command Tunnel 实现绘制(in bs)
4. 调用 command Tunnel 实现合成(in screen)

Protocol && Object streaming

1. 链路控制（连接维护、收发包）
2. 命令、参数、状态编解码
3. 对象序列化与反序列化

### 方案A’

Qt 支持层

1. 管理client会话【连接侦听】
2. 接收client输入，并转发给inputmgr
3. 监视fb的更新，并发送给client

Receiver

1. PC窗口布局
2. 捕获转发鼠标
3. 根据MK发来的脏区域更新PC窗口

Protocol && Object streaming

1. 链路控制
2. 协议编解码（基于protobuf）
3. 交叉编译（armLinux, windows）

## 物理设计

共3个进程：位于PC端的Receiver，以及位于MK主机上的Frontend（Qt）和InputMgr，相比改动之前，除PC显示用的程序Receiver外，没有增加新的进程。

控制流、数据流如前所述。

## 遗留问题及确认

~~波形的绘制状态~~

**多屏**

~~初步想法是运行2个PC程序，一个程序对应一个MK的屏幕。~~

**Iview**

~~MK为硬件开窗，初步想法是软件不处理。~~

**线程模型**

PC双线程（input + display + 收发）

MK 只增加线程（收发）

**限制**

~~贴图的性能~~

唯一TLW

~~滑动不能精确回放~~

~~PC后接上，显示也要保持完整。（主动 sync whole fb）~~

光标的单独处理更新：不支持（fb已混合光标）

滚动的启发式判断（copy rect）：不支持

窗口拖动的启发式判断（wireframe）：不支持

缩放：不支持

旋转：不支持

认证：不支持

增加流控（客户端查询）

## 评审结论

优选PAD(FB)方案，并进行简单的压缩（lzo）。