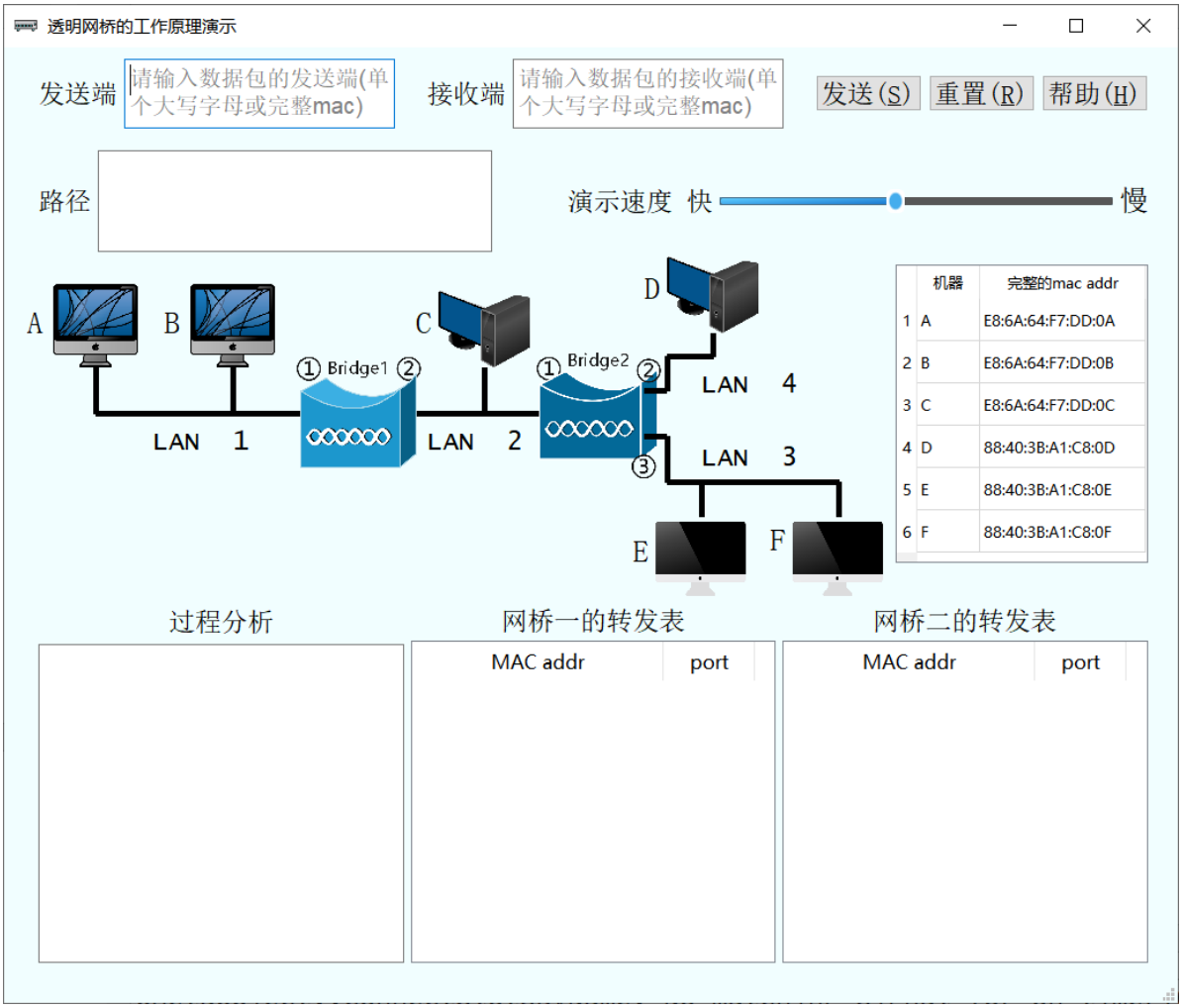
10.透明网桥工作原理-技术文档

1952640 刘羽茜 1951247 钟伊凡 1953597 刘云帆

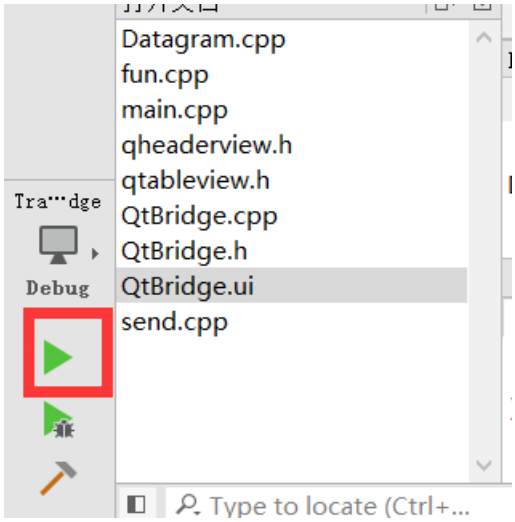
0.实现结果的展示



1.编译环境

Windows2019, Qt 5.14.2 MinGW 64-bit Debug模式或Release模式

点击.pro文件，使用QtCreate打开，左下角绿色按钮构建即可。



2.文件

头文件：

QtBridge.h声明了QtBridge类，fun.h声明了输入内容的处理函数。

源文件：

main.cpp 调用Bridge类，页面开始

QtBridge.cpp ui初始化内容的填充，成员的赋值（网桥的转发表，各个机器的信息及其所在位置等都是类的成员变量），重启函数，表格更新函数等

Datagram.cpp 数据包发送的动画的实现，包括出现、从某个节点复制多份、接收和不接收的动画等等

send.cpp 数据的转发、学习、泛洪等的实现，调用Datagram中的函数实现正确的动画

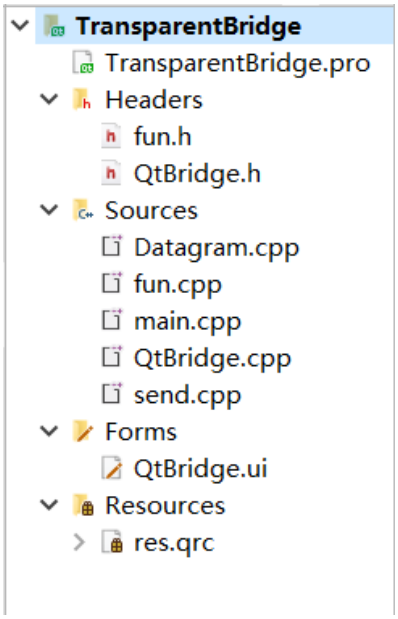
fun.cpp 数据读入的处理

ui文件：

QtBridge.ui设计页面布局

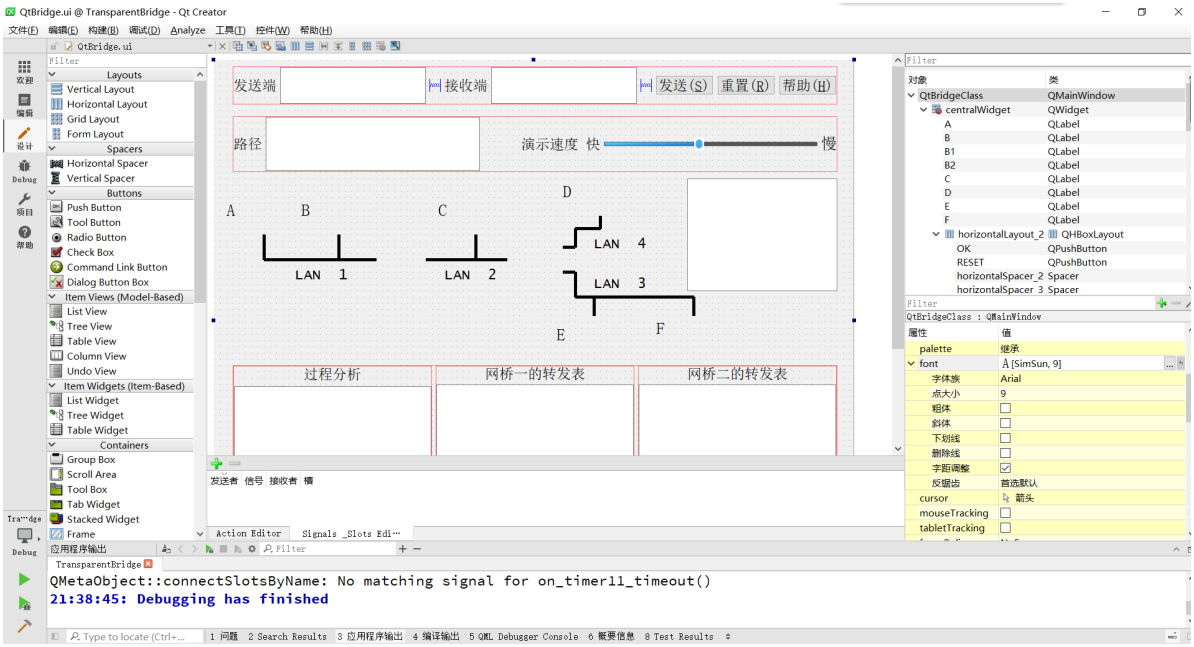
qrc文件：

导入使用的图片



3.初始化UI

使用QtCreate点击ui文件，可以放置需要的控件，设置格式等等



4.模块的划分与函数功能

以下是主要的部分

所在文件	函数	功能
QtBridge.cpp	QtBridge::QtBridge(QWidget *parent) : QMainWindow(parent)	ui初始化内容的填充，成员的赋值
QtBridge.cpp	void QtBridge::initialPos()	初始化动画中各个节点的位置
QtBridge.cpp	void QtBridge::update_table(int table,string mac_addr,int portno)	更新转发表
QtBridge.cpp	void QtBridge::restart()	重置
QtBridge.cpp	void QtBridge::on_horizontalSlider_valueChanged(int value)	响应对动画速度的控制
QtBridge.cpp	void QtBridge::on_pushButton_clicked()	“帮助”按键的实现
Datagram.cpp	int QtBridge::getDatagram(const struct pos &startPos)	出现一个数据包的图案
Datagram.cpp	void QtBridge::moveDatagram(int id, const struct pos &targetPos)	对该数据包进行移动
Datagram.cpp	void QtBridge::hideDatagram(int id)	数据包隐藏（用于流动画）
Datagram.cpp	int QtBridge::forkDatagram(int id)	复制一个数据包图案出来（用于泛洪）
Datagram.cpp	void QtBridge::closeDatagram(int id, bool reached)	让报文图案消失，并选择消失样式
send.cpp	void QtBridge::send()	对发送的信号进行相应，递归调用lan_to_bridge函数完成泛洪学习转发，把包送到目的地址
send.cpp	bool QtBridge::lan_to_bridge(string begin_addr, string end_addr, int lan_no,int data_no,int port_choose,int inundate)	将数据从某个lan发送到和lan相连的两个桥上，和桥的转发表做对比，更新或者发送等等，返回值代表往这条路能不能发送到目标地址处 能返回1
fun.cpp	bool check_addr(string port,string& port_smp,map<string,string>mac_tot)	输入内容的处理

5.重要功能的实现方法

1.透明网桥的泛洪、学习与转发

通过递归调用lan_to_bridge函数实现，每次从一个lan发送到lan对应的透明网桥中，朝这个lan上除了源机器以外的机器发数据，是目的机器就接收，不是就不接收。

如果目的地点是在发往的网桥的转发表上，就不泛洪了，安装转发表发送，如果不在就泛洪，继续调用lan_to_bridge

如果要将数据包发向多个地方，就调用 int QtBridge::forkDatagram(int id)函数复制数据包

用void QtBridge::moveDatagram(int id, const struct pos &targetPos)函数把数据包移动到对应节点上

2.动画的实现

我们的动画效果主要是模拟了包发送的完整过程，使用Qt提供的QPropertyAnimation库，实现了以下几个函数：

```
void initDatagram();
int getDatagram(struct pos const& startPos);
void moveDatagram(int id, struct pos const& targetPos);
void hideDatagram(int id);
int forkDatagram(int id);
void closeDatagram(int id, bool reached);
```

void executeTask(int id)的思想是，程序开始时，初始化指定个数个数据包图片，设为不可见。

如果动画演示需要数据包，就可以通过getDatagram函数在指定位置获得一个数据包，再通过moveDatagram来移动指定的数据包。

还要注意，Qt中的动画是独立执行的，也就是说，调用animation.start()之后，程序就会继续向下运行，而动画也会自顾自地运行。这就导致如果在调用动画的语句后面有需要动画执行完毕才执行的程序，就会立即被执行，从而出错。

为了规避这个错误，我们采用的方式是维护每个数据包的动画队列，每次需要这个数据包执行什么动画效果，就加入到它的队列中，但并不执行。只有调用executeTask(int id)的时候才从队首取一个任务来执行，并且设置一个定时器，在这个动画效果执行完毕之后产生时钟中断来再次调用executeTask(int id)以及相关处理。定时器的好处是它是在后台计时的，完全不妨碍主程序流的进行，只在时间到的时候产生中断；如果不是用定时器而使用睡眠来等待，则动画进行的时候主程序流也被阻塞住，不可取。

使用我们的实现方法，就又能达到正常的动画效果又不会影响到程序的正常运行。