基于Qt的业务监控系统界面设计与实现

朱吉佳,蔡家麟
(华东师范大学 信息学院 电子科学与技术系,上海 200062)

摘 要:随着3TNet网络的迅猛发展,专用于实时监控3TNet网络上各项业务流的业务监控系统越来越受关注,因而对于业务监控系统的研究亦愈显其重要性。主要针对3TNet业务监控系统界面设计和实现过程中所遇到的问题,提出了一种在Linux环境下基于Qt的实现方案。分析Qt在处理图形界面设计、实现信号传递及事件响应等方面的优秀性能。结合实际开发过程,详细描述了系统界面的整体结构设计,网页浏览器嵌入至Qt界面的方法,以及Qt信号/槽机制和事件的调用。

关键词:Qt;Konqueror/Embedded;信号/槽机制;事件
中图分类号:TP319 **文献标识码**:A **文章编号**:1673-629X(2008)03-0236-04

Development and Realization of Operation Monitoring System's GUI Based on Qt

ZHU Ji-jia, CAI Jia-lin
(Dept. of Electronic Eng., Coll. of Information, East China Normal Univ., Shanghai 200062, China)

Abstract: The special operation monitoring system used to scout 3TNet's data streams live gains much attention, along with the rapid development of 3TNet. Puts forward a scheme based on Qt in Linux, which aims at realizing 3TNet operation monitoring system's GUI. It introduces the excellent characteristics of Qt in building GUI, signals transferring, and event processing. Combined with the programming experience, it particularly describes this system's whole frame, the way of letting a web browser embedded in GUI, and how to make use of signals and events.

Key words: Qt; Konqueror/Embedded; signals and slots; events

1 系统概述

3TNet业务监控平台是架设于3TNet网络上,用于动态监控网络运行状态,实时监视网络服务质量,特别是IPTV数据服务质量的综合网管平台。当视频流质量发生问题时,业务监控平台能及时发出告警通知网络运维人员。本业务监控系统是3TNet业务监控平台的子系统,用于监控平台的用户终端实时监控系统和网络监控手持终端之中,为3TNet业务监控平台提供一个网络用户终端监控平台,也使得网络施工人员能实时检测网络质量和确认网络的正确安装。本系统主要功能为:

(1)模拟机顶盒的操作全过程,以WEB处理的方式交互各项业务流程,作为机顶盒应用处理错误仿真的实用平台。

(2)跟踪接入的各项消息流,全面地分析网络消息交互流程,以便快速地解决3TNet网络中的交互错误。

(3)对3TNet网络上传输的IPTV、VOD业务数据流进行网络统计、计算、评估等,从而分析当前业务视频流的质量,量化业务数据质量,找到优化网络的量化数据。

2 开发技术

本系统界面的开发是以安装有Linux(Fedora Core 5)的PC机作为调试环境,主要应用到的技术是Qt和Konqueror Embedded。

2.1 Qt

作为用户终端监控平台,对本系统界面有跨平台的要求;作为网络监控手持终端,要求本系统界面能在嵌入式系统上运行。因此选用Qt编写本业务监控系统图形界面。Qt是一个多平台的C++图形用户界面应用程序框架。它完全面向对象,有良好的封装机

收稿日期:2007-06-09
作者简介:朱吉佳(1983-),女,上海人,硕士研究生,研究方向为通信与信息系统;蔡家麟,副教授,硕士生导师,研究方向为移动通信、多媒体通信、通信网仿真及性能评估。

制,模块化程度高,可重用性好,容易扩展,允许真正的组件编程,提供给应用程序开发者建立艺术级图形用户界面所需的功能,提供了信号/槽机制替代回调函数,使组建间信号传递更安全、简单。它拥有强大类库,而且具有一个包括Qt设计者、语言学家和详细联机开发文档的跨平台开发环境[1]。Qt的开发商特别研制了Qt/Embedded来实现Qt在Embedded系统上的应用。这点也为将本业务监控系统移植到手持终端PDA上提供了后续支持。本系统界面中Qt的版本为Qt2.2.2,其源码下载地址为:http://www.qtopia.org.cn/ftp/mirror/ftp.trolltech.com/qt/source/qt-x11.2.2.2.tar.gz

2.2 Konqueror Embedded

Konqueror Embedded[2]是由KDE下的浏览器Konqueror派生而来,主要针对嵌入式Linux的。它将Konqueror中关于KHTML、SSL、Javascript等内容继承下来,同时简化了Konqueror中很多类的定义,剔除了依赖于KDElib部分,以适应在不同的嵌入式平台上移植和运行。本系统中Konqueror Embedded的版本为0.1,其源码下载地址为:http://developer.kde.org/-hausmann/Attic/konqueror-embedded-0.1.tar.gz

3 设计与实现

在本系统界面具体的代码设计与实现过程大致分为三步:

(1)用Qt建立GUI界面的主框架。

(2)将网页浏览器嵌入界面。

(3)调用Qt信号/槽机制与事件功能以完善界面的详细功能。

3.1 用Qt设计界面主体框架

3.1.1 Qt类库

Qt拥有十分强大的类库,拥有400多个面向对象的类,它们带有大多数构建跨平台服务器与客户端程序的底层基础构造函数。Qt的类库可以分成三部分:组件(Components)、框架(Framework)、效用工具(Utilities)。其中,组件包括环境、主窗口和相关类、标准对话框、基本窗口部件、高级窗口部件、组织者及帮助系统;框架包括对象、模型、抽象窗口部件、图形和打印、拖放、窗口部件外观和布局管理;效用工具包括通用工具类、图像处理、日期与时间、I/O处理和杂项。在程序设计中最常用的类包括QObject、QApplication、QWidget、QMainWindow、QLayout、QEvent等[3]。

(1)QObject类具有所有Qt对象的基类。它是Qt对象模型中心。信号/槽机制和属性都离不开QObject类。在启用信号/槽时,须加入Q_Object宏。QObject可以通过event()接收事件并过滤其它对象的事件。QObject把自己组织在对象树中,它可以自动添加、删除子对象和查找对象。QObject提供了Qt中最基本的定时器。

(2)QApplication类,它继承了QObject,管理图形用户界面应用程序的控制流和主要设置。它包括主事件循环,在其中来自窗口系统和其它资源的所有事件被处理和调度;处理应用程序的初始化和结束,并且提供对话管理;处理绝大多数系统范围和应用程序范围的设置。

(3)QWidget类,继承自QObject类和QPaintDevice类,是所有用户界面对象的基类。它从窗口系统接收鼠标、键盘和其它事件,并且在屏幕上绘制自己的表现。它被QButton、QComboBox、QLineEdit、QMainWindow、QTabWidget等类继承,这些子类提供了实际的功能。

(4)QMainWindow类,继承自QWidget,提供了一个有菜单条、锚接窗口和一个状态条的主要应用程序窗口。它使得封装中央部件、菜单和工具条以及窗口状态变得容易。

(5)QLayout类,是处理布局的基类,被QBoxLayout和QGridLayout所继承,以多种方式控制组件的排版布局。

(6)QEvent类,是所有事件类的基类。Qt的主事件回路从事件队列里取得本地窗口系统事件,转换为QEvent,并且把这些发给QObject。

3.1.2 GUI结构及各部分类属关系

本系统中的各个窗口都继承自QMainWindow类。将主窗口命名为Controller,负责Qt主程序的运行,用Tab页方式作子窗口间的切换显示。基于本系统的主要功能,定义三个子窗口类为StbWindow、InfWindow、NetWindow依次对应三个主要功能。其中,StbWindow类,包括网页浏览器、QButtonGroup、QToolBar、QAction、QPushButton、QComboBox等;InfWindow类,包括网页浏览器、QToolBar、QAction等;NetWindow类,包括QPainter、QGroupBox、QToolBar等。StbChild、InfChild、NetChild则对应上述的三个子窗口类,它们负责实现各个子窗口中所有的弹出设置窗口。本设计的类属结构示意图如图1所示。

3.2 网页浏览器Konqueror/Embedded嵌入Qt界面

浏览器越来越多地被嵌入至软件界面中,成为其必不可少的一部分,为用户带来"即时"查看网络信息的便利。例如,常见的BitComet软件中的"浏览功能"。在本系统界面中,网页浏览器承担网络交互和参数设置任务。

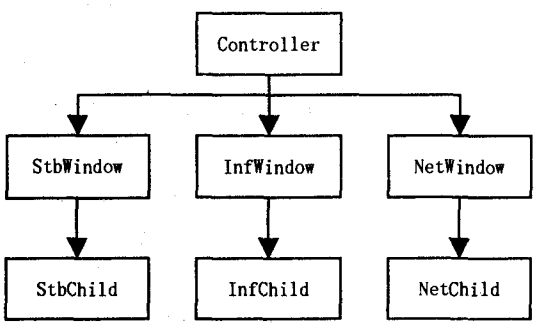


图 1 系统界面设计类属结构示意图

Konqueror/Embedded 主要由 KIO、Kparts、DCOP 等部分组成。KIO 是一个网络使用的文件管理类,包含了所有可能用到的文件管理方法;Kparts 负责管理整个应用和窗口;DCOP 实现 KDE 应用程序的进程间通讯的功能。

在开始嵌入工作之前,需修改浏览器的界面,以符合实际系统对界面的需求。例如,保留与网页浏览相关的部分,删除工具栏、菜单、地址栏等功能,可通过修改 Konqueror/konq-embed/src 目录下的 mainwindow.h 和 mainwindow.cc 中相关的类实现。笔者将修改后的类名分别改为 StbWindow、InfWindow,相应的.cc 文件改名为 stbwindow.cc 和 infwindow.cc。

为了使浏览器能在界面设计中被灵活应用,需要一个合适的 Makefile[4]来管理源代码。"嵌入"工作的主要思想是修改 Makefile 文件,使浏览器能被界面调用修改。由于 Konqueror/Embeded 本身是基于 Qt 库的,所以选择 Konqueror/Embedded 源代码的 Makefile(目录 konqueror-embedded/konq-embed/src/)作为蓝本修改。"嵌入"的具体实现步骤如下:

(1) 创建一个工作目录,取名为 monitor。

(2) 在 monitor 下创建一个源代码目录,取名为 src,用于存放源代码,如:main.cc、controller.cc、controller.h 等文件。

(3) 在 monitor 下创建一个库目录,取名为 lib,用于存放已经编译通过的 konqueror 库文件,即 konq-embed 目录下的 dropin 和 kdesrc 文件夹和 konqueror-embedded 目录下的 LIBTOOL。

(4) 修改 Makefile 文件中的 INCLUDES 变量。该变量指向一些库所在的目录。先定义变量 LIBDIR =../lib(指向 monitor/lib 目录),将 INCLUDES 修改为:

```
INCLUDES= -I$(LIBDIR)/dropin
    -I$(LIBDIR)/kdesrc/kdecore
  -I$(LIBDIR)/kdesrc/kio/http/kcookiejar
    -I$(LIBDIR)/kdesrc/khtml
  -I$(LIBDIR)/kdesrc $(QT_INCLUDES)
```

(5) 修改 Makefile 文件中 konq_LDADD 变量。该变量指向库,和(4)相似,就不一一举例了,以其中一个库为例。将 konq_LDADD =../kdesrc/kio/http/kcookiejar/libcookiejar.la 改成 konq_LDADD = $(LIBDIR)/kdesrc/kio/http/kcookiejar/libkcookiejar.la

(6) 修改 konq_nofinal_OBJECTS 变量。该变量定义了需要编译的源文件,因此需在其中添加 Qt 源程序的各个文件名。例如:konq_nofinal_OBJECTS = main.$(OBJEXT) controller.$(OBJEXT) netwindow.$(OBJEXT) toolwindow.$(OBJEXT) stbwindow.$(OBJEXT) infwindow.$(OBJEXT) stbchild.$(OBJEXT) infchild.$(OBJEXT) textview.$(OBJEXT) netchild.$(OBJEXT) toolchild.$(OBJEXT)

(7) 修改 Makefile 中有关生成 moc 文件的规则。例如:

```
stbwindow.moc: $(srcdir)/stbwindow.h
    $(MOC) $(srcdir)/stbwindow.h -o stbwindow.moc
infwindow.moc: $(srcdir)/infwindow.h
    $(MOC) $(srcdir)/infwindow.h -o infwindow.moc
```

(8) 修改 Makefile 中.moc 文件的依赖规则。例如:$(srcdir)/stbwindow.cc: stbwindow.moc

$(srcdir)/infwindow.cc: infwindow.moc

(9) 在.cc 源文件的末尾添加 moc 文件的引用。例如:在 stbwindow.cc 中添加 #include "stbwindow.moc",在 infwindow.cc 中添加 #include "infwindow.moc"。否则,在编译时会出现错误提示"未定义的引用"。

(10) 修改 main.cc,使其包含网页浏览器运行时所需的文件,并定义相关的对象。如:#include <kcookieserver.h>,并在 main 函数中加入"KCookieServer cookieEater;"语句。

重新 make,就可见在界面中已经可以调起网页浏览器了。

3.3 信号/槽机制和事件机制的调用

业务监控系统界面外观完成后,随后的任务是实现其界面功能。例如:以按钮来控制子窗口弹出、工具栏在不同模式下相互切换、业务统计结果以图形方式显示等等。在实现这些功能的过程中,应用到 Qt 两个特征功能:信号/槽机制、事件。

3.3.1 信号/槽机制

Qt 中的信号/槽机制主要用于实现对象间的通讯。当一个特定事件发生的时候,一个信号被发射,并

由一个被称作槽的函数来处理这一信号。信号/槽机制可以保证把一个信号和一个槽连接起来。

Qt 窗口部件有很多预定义的信号和槽。调用此种预定义的信号和槽只需一条 connect 语句即可。如:connect(print,SIGNAL(clicked()),this, SLOT (update ()));此语句将按钮的按动信号 clicked()与画板页面的刷新 update()槽相连。print,指向按钮的对象的指针,由 print = new QPushButton(this)创建。this,指向当前类的指针,connect 语句中的 this 指针指向调用画板的类。只要按动 print 一下,则画板刷新一次。注意,须将 Q_Object 宏写在需要调用信号/槽机制的类中。

也可以通过继承来加入自己定义的信号和槽。以本系统为例,期望在网络性能统计页面中,当检测到 ir 值低于极限值时,系统自动弹出预警窗口作为提示。以自定义信号和槽来实现此功能,步骤如下:

(1) 在.h 文件中加入:

```
public slots:
    void recIrArrived(); //定义一个槽
public:
    void checkIr(); //定义用于检查 ir 值的函数
signals:
    void irArrived(); //定义一个信号
```

(2) 在.cc 文件中加入:

```
connect( this,SIGNAL( irArrived() ), this,
    SLOT( recIrArrived())); //将信号与槽相连
//检查 ir 值是否达到极限值
void NetWindow::checkIr()
{
if ( ir <= IR_LIMITED )
    emit irArrived();
}
//槽的具体实现
void NetWindow::recIrArrived()
{
    alertChild->show(); //弹出预警子窗口
}
```

如此一来,只要在需要检测 IR 值时,调用函数 checkIr(),则一旦 ir 低于极限值就弹出预警窗口。

3.3.2　事件

在本系统界面设计中,期望显示对数据流的网络分析、统计、计算、评估的结果,并以直观且形象化的图形展示用户面前。因而,需要调用 Qt"事件"功能中的画板(QPaintEvent)[5]。

"事件"功能,简单来说就是,当一个事件产生后,相关控件作出回应。它的调用顺序是从 Qt 主事件回路 QApplication::exec()进入事件循环并且等待。主事件循环从窗口系统中接受事件,把它们转换成 QEvent 并且分派给 QObject。QObject 通过调用它们的 QObject::event()函数来接收这些转换过的事件,作出回应。

Qt 已经将如上这些繁琐的调用步骤封装,减轻了编程人员的负担。以笔者在负责网络分析的界面中,用饼图显示网络总体性能的分析结果为例,具体可以如下代码实现:

(1) 在.h 文件中加入:

```
//定义画板事件
protected:
    void paintEvent( QPaintEvent * );
//定义画饼图的函数
public:
    void drawColorWheel( QPainter * );
```

(2) 在.cc 文件中加入:

```
//具体实现画板事件
void NetWindow::paintEvent( QPaintEvent * )
{
    QPainter paint( this ); //定义画笔
    if ( ……) //触发条件
    drawColorWheel( &paint );
}
//具体实现画饼图的函数
void NetWindow::drawColorWheel( QPainter * p )
{
    …… //具体画图的方法不赘述
}
```

(3) 在 NetWindow 类的构造函数中加入:

```
//连接图形按钮的按动信号与画板刷新的槽
connect( totalAction, SIGNAL( activated() ),
```

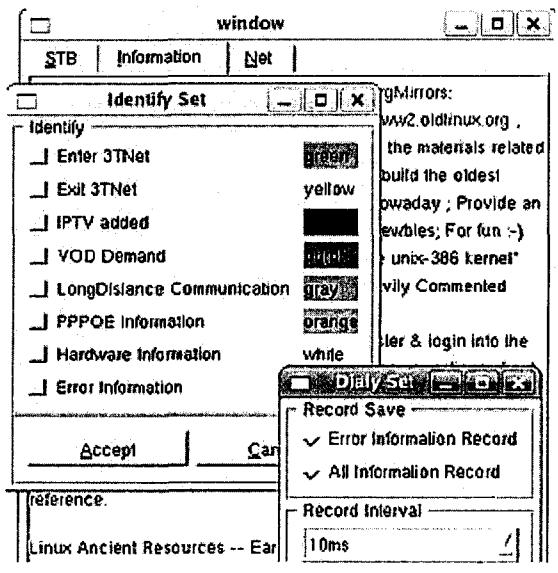


图 2　业务监控系统界面

(下转第 242 页)

看到,黑色圆点处为污染源源头,从污染源往河流下游方向去颜色由深逐渐变浅,说明越往下游污染物浓度越小,达到了较好的模拟效果。

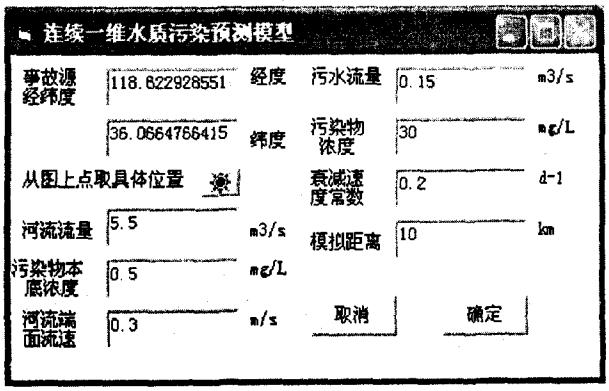
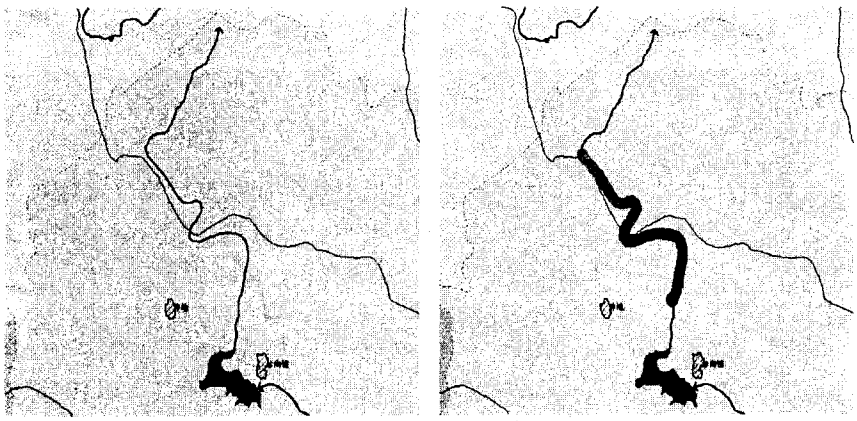


图 1 参数界面



(a) 模拟前河流 (b) 模拟效果图

图 2 模拟前后效果对比

4 结束语

在 Visual Basic 可视化编程开发环境下,结合 MapX 组件技术,利用 MapX 的符号对象(symbol)和专题图(theme)功能,只用很少的代码便可以轻松实现模型的可视化模拟,使模型数据与地理空间位置信息相结合,模型变得更加形象直观。与单独利用某一开发软件相比免去了复杂的设计,节省大量编程时间,大大提高了编程效率。为类似模型可视化实现提供了一种快速、简易的解决途径和解决思路。

参考文献:

[1] 谭汉松,陈红玲,田党清,等.基于 GIS 的长株潭生态评价系统与实现[J].计算机技术与发展,2007,17(1):145-147.

[2] 徐少平,吴信才,曾 文.基于 GIS 的供水管网水力计算模型[J].计算机技术与发展,2006,16(3):176-178.

[3] 王德文,赵文清.基于 MapX 地理信息系统设计与实现[J].微机发展,2003,13(5):59-61.

[4] 齐 锐,屈韶琳,阳琳赟.用 MapX 开发地理信息系统[M].北京:清华大学出版社,2003:1-11.

[5] 刘 光.地理信息系统——组件开发篇[M].北京:中国电力出版社,2003.

[6] 李连营,李清泉,李汉武,等.基于 MapX 的 GIS 应用开发[M].武汉:武汉大学出版社,2003.

[7] 丁 瑞,王翠珍,秦树林,等.基于 MapinfoMapX 环境信息系统的开发研究[J].浙江化学,2007,38(2):27-30.

[8] 翟卓韬,孙 洁,赵劲松,等.基于 MapX 的泄漏事故救援决策系统[J].计算机与计算机应用化学,2007,24(4):488-502.

[9] 谷志锋,郭跟成.对 Mapinfo 二次开发的三种方法的对比和研究[J].电脑知识与技术,2007,6:1691-1692.

[10] 刘培桐.环境学导轮[M].第 2 版.北京:高等教育出版社,2002:78-79.

(上接第 239 页)

```
this, SLOT( update() ) );
```

至此,界面的功能丰富了,本业务监控系统界面设计也完成了。实现后的本监控系统的用户界面。如图 2 所示。

4 结束语

提出了 3Tnet 业务监控系统界面实现的一个设计方案。这个方案的重点部分在于网页浏览器嵌入 Qt 界面,使两者有机结合。同时,Qt 的优良性能也保证了本系统界面的可靠性、高效性和跨平台性。实践表明,本系统界面设计方案是行之有效的。

参考文献:

[1] Trolltech. Qt - Cross - Platform C++ Development - Trolltech[EB/OL]. 2007. http://www.trolltech.com/products/qt/features/index.

[2] Konqueror.org Webmaster. Konqueror Embedded[EB/OL]. 2006. http://www.konqueror.org/embedded/.

[3] Trademarks. Qt 参考文档[EB/OL]. 2002. http://www.qiliang.net/qt/index.html.

[4] Gunguymadman. 跟我一起写 Makefile[EB/OL]. 2007. http://www.chinaunix.net/jh/23/408225.html.

[5] Blanchette J, Summerfield M. C++ GUI Programming wieth Qt 4[M]. USA: Prentice Hall, 2006.