目录

APRS 客户端软件系统设计与开发

摘要

APRS 是自动位置报告系统（Automatic Position Reporting System）的简称，它是业余无线电操作者能迅速的将实时事件的相关数据发布出去，并在接收端的计算机上图形化的表示这些数据的软硬件系统。自从 1992 年诞生以来，APRS 得到了很大的发展，在很多领域都有了广泛的应用。而一个以包含 Java Applet 的网页形式进行发布的 APRS 客户端软件系统，一定会因为它的使用方便，以至于能让没有任何无线电知识和计算机知识的人都可以顺利使用而大受欢迎，这也将有助于 APRS 的继续发展和普及。

关键词

APRS，自动位置报告系统，客户端，Java，Applet，Swing，Socket，Servlet

Abstract

APRS is short for Automatic Position Reporting System.It is a software and hardware system that can be used by amateur radio operators to send the data about real-time events rapidly and show the data on the receiver’s computers graphically.Since it was born in 1992,APRS has gained a great progress and has been used in a lot of areas.An APRS client software system released in the form of web pages which contain Java Applet must be popular for its convenience.People without any knowledge about radio and computer also can use the system successfully,which is helpful to APRS’s continuing development and popularization.

Key Words

APRS, Automatic Position Reporting System, client, Java, Applet, Swing, Socket, Servlet

前言

APRS 客户端软件系统是 APRS 的重要组成部分，它是直接与用户进行交互的部件，它的好坏直接影响用户对整个 APRS 系统的评价。我们的目标是设计开发一个使用极其简便，只要会上网的人都可以使用的 APRS 客户端软件系统。本文分四个章节对本系统的设计开发进行的介绍，第一章介绍了 APRS 的基本知识，使大家可以认识 APRS；第二章对系统进行了总体设计，把系统划分为三个模块；第三章分模块对系统进行了详细的开发，介绍了每个模块中重要的类以及开发过程中遇到的技术难题的解决；第四章对整个毕业设计进行了总结。

第一章　APRS 基础

要设计和开发 APRS 客户端软件，就必须先要了解什么是 APRS。APRS 是自动位置报告系统（Automatic Position Reportig System）的简称，它是一项新兴

的业余无线电活动内容，业余无线电操作者能迅速的将实时事件的相关数据发布出去，并在接收端的计算机上图形化的表示这些数据的软硬件系统。

它将无线电数据通信、全球卫星定位系统（GPS）、计算机和因特网有机结合，引起了众多业余无线电爱好者的兴趣。它不仅提供了诸如位置跟踪、气象信息等有用的服务，而且在业余无线电应急通信(ARES)以及紧急遇险救援时发挥重要作用，同时 APRS 还为技术爱好者提供了一个广阔的平台，许多其它应用都等待着我们的进一步开发。

1．1　APRS 的历史与发展

1992 年，被称为“APRS 之父”的美国爱好者 Bob Bruninga，WB4APR 在 APRL 和 TAPR 数字通信会议第一次引入了 APRS 的名称。初期的 APRS 是完全建立在传统业余无线电分组通信的基础上的。1999 年美国爱好者 Steve Dimse，K4HG 引入了 APRS 第一个因特网的接口，APRS 的传输媒介出现了无线分组通信和因特网共存的局面。APRS 软件和硬件的迅速增加印证了 APRS 的飞速发展。从 1992 年到 1999 年，只出现了 6 个应用软件，而现在，应用软件的数量已然超过 20 个。新版的 TNC 一般都增加了对 GPS 和 APRS 的支持。另外，一些新的专门为 APRS 设计的硬件也日益增多，比如兼容 TAPR TNC2 的 UIDIGI，它烧入 ROM 后可将旧的 TAPR TNC2 改造成专门用于 APRS 的数字中继，又如 Tinytrak，它将 GPS 数据转换成无线分组通信的发射音调，专门用于设置 APRS 发射台。

1．2　APRS 基本系统组成与设备

1．2．1　APRS 基本系统组成

APRS 基本系统包含：全球定位系统（GPS）接收器、分组终端节点控制器（TNC）、业余无线电（电台、天线）、计算机和 APRS 软件。

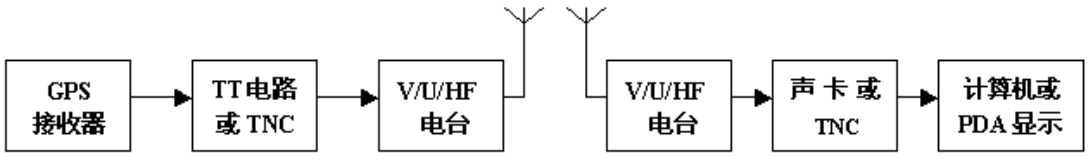

图 1.1　　APRS 基本系统组成

GPS 接收器只要含有 NMEA－0183 格式的数据输出都能用于 APRS。APRS 不需要任何特殊的电台设备。任何能通过无线电管理部门检测，频率稳定，话音失真比较小的电台设备都可使用。用于 APRS 的计算机的规格需求也因不同的 APRS 软件版本而异，市场上近几年的计算机一般都可应用于 APRS。在计算机上连接设置 TNC 或 GPS，目前还常常通过计算机的串口来完成。现在很多笔记本电脑上不配备串口，则只能通过 USB 转串口线来解决。USB 端口的 TNC 和 GPS 现也已出现。如果你希望将 APRS 电台同时作为 IGate，则性能稳定的因特网宽带接入是必须的。TNC 可使用硬件解决，也可通过计算机声卡和软件来虚拟 TNC。

APRS 的软件很多，可在 Mac、PC、笔记本和 PDA 上工作，运行于 LINUX、MAC、WINDOWS、DOS 等操作系统。以下是三个基于 Windows 系统的软件。

UI－View 32bit V2.03 是由已故英国爱好者 G4IDE 开发。支持 Windows 98/2000/XP，功能强大，风靡欧洲，正席卷全球。也是国内爱好者使用最多的软件。

WinAPRS 是美国最为流行的 APRS 软件，支持精确地图。

AGWPE 声卡 TNC 是 SV2AGW 的作品，界面完全图形化，非常美观，操作简便。但占用大量 CPU 资源，运行慢。

其它软件还有 MacAPRS、APRS/CE、XASTIR 等等。

1．2．2　APRS 的设备

APRS 的设备大抵分为 APRS 电台、无线数字中继、APRS 网关、APRS 服务器等。

在整个 APRS 中，各电台地位是均等的，既可以把自己的位置数据发给所有的电台，也可以接收来自所有电台的位置数据。我们把能接收和发送 APRS 数据包的电台称为 APRS 电台。APRS 电台可以细分成几类。第一类是仅作发送的电台，比如，一个装在汽车上的不断发送经纬度数据的电台，一个不断发送气象信息的电台等。这种电台的构成包括数据源（GPS 接收器、气象传感器等）、数据变换和处理器（即终端电台控制器 TNC）和数据发送器（无线电发射机），TNC 的作用是将数据源的数据变换成 APRS 的格式，调制后送给发射机发送。第二种是仅作接收的电台，一个连接因特网并运行 APRS 软件的计算机就是最简单的接

收电台，在没有因特网接入的环境下，无线电接收机、TNC 和单片机为核心的 LCD 显示器或者笔记本电脑也可构成接收电台。这种电台的构成包括数据接收器（因特网或无线电接收机）、数据变换和处理器（TNC 或者声卡加软件）、数据显示设备（运行软件的计算机或者单片机为核心的 LCD 显示器）。第三种就是接收和发送电台的合一，一个连接 GPS 接收机、连接因特网并运行 APRS 软件的计算机，一个终端加 TNC 加收发信机（即典型的分包通信设备）都可以成为收发合一的 APRS 电台。

由于电台的发射范围有限，所以人们在分组通信系统中采用了无线数字中继。无线数字中继实际上就是一个分组通信电台，只不过它只负责转发分组数据，不产生或者最终处理数据。

APRS 中，业余无线分组网络与英特网的互联由 IGate 完成。它承担无线分组网络的 AX.25 分组与英特网上的 TCP/IP 分组的转换与转发。

APRS 服务器提供 APRS 的英特网直接接入。也就是说，如果只是为了了解 APRS，无需购买昂贵的终端节点控制器和电台，也就不必拥有电台执照，可以通过因特网接入 APRS 服务器，就可以向全球的 APRS 宣告你的存在，并查看全球 APRS 的所有信息。比如 UI－View 软件，支持无线分组－无线分组、无线分组－因特网、因特网－因特网三种方式的 APRS 通信。其最主要的功能是将接收到的 APRS 数据中的经纬度数据转换成电子地图上的物体并显示出来。当然，你可以通过周期性的向 APRS 服务器发送自己的 APRS 数据包告诉世界你的位置。

1．3　APRS 工作原理与应用

APRS 系统使用无线分组通信（Packet Radio）将数据进行发布。在传统的无线分组通信操作中，通信是使用有连接分组基于一对一产生的，即，两个电台是虚拟的互相连接的。在 APRS 中，电台使用无连接分组基于一对多进行数据传播，类似于广播电台和听众的关系，所以，将“听众”的范围扩展到所有可以接收到该分组的电台。

APRS 使用传统无线分组通信的“信标”（Beacon）功能完成这种模式的通信。一个 APRS 分组以某种特定格式包含电台位置（经度和纬度）和电台类型（家中的、便携的、移动的、数字中继、气象站等）信息，使处于接收 APRS 电台端的计算机上运行的软件可以处理包含的信息，并在地图上显示相对应的图标，显

示发信标电台的位置。如果 APRS 电台是那种处于移动之中的便携或者移动电台，当收到位置更新的新位置分组后，APRS 在地图上改变图标的位置。

移动 APRS 电台包含一个普通电台、TNC 和 GPS（全球定位系统）接收器。GPS 接收器接收地球轨道卫星信号自动计算其位置并传送给 TNC，通过 TNC 的处理将位置数据转换成音频信号，此音频信号通过电台发送出去更新 APRS 接收端地图上移动 APRS 电台的位置。

在家庭固定 APRS 电台中，由于位置是固定的，一般不需要 GPS 接收器，只需要事先将家庭的经纬度数据设置到 APRS 应用软件中，当需要的时候，软件发送已事先设置的位置信息给 TNC，TNC 再转发给电台以便发射。

除了跟踪移动电台，APRS 也可以向系统输入对象位置的方式跟踪任何对象。比如，你可以输入一个飓风的经纬度，则飓风的位置就可以出现在任何该信道的 APRS 电台的地图上。在气象应用中，你可以将气象监测设备的接口连接到 APRS 电台以便向其它电台发布实时气象信息。APRS 可使用数字中继（Digipeater）进行转信，扩大 APRS 使用区域。

为了将 APRS 传播到整个世界，有的 APRS 电台作为 IGate（因特网网关）将接收到的 APRS 分组转发到因特网上的服务器。这些服务器将世界上实时的 APRS 数据搜集并转发，有的还提供 Web 页面的格式，以便让用户通过支持 Java 的网络浏览器查看 APRS 活动。

1．4　APRS 的协议

1．4．1　APRS－IS

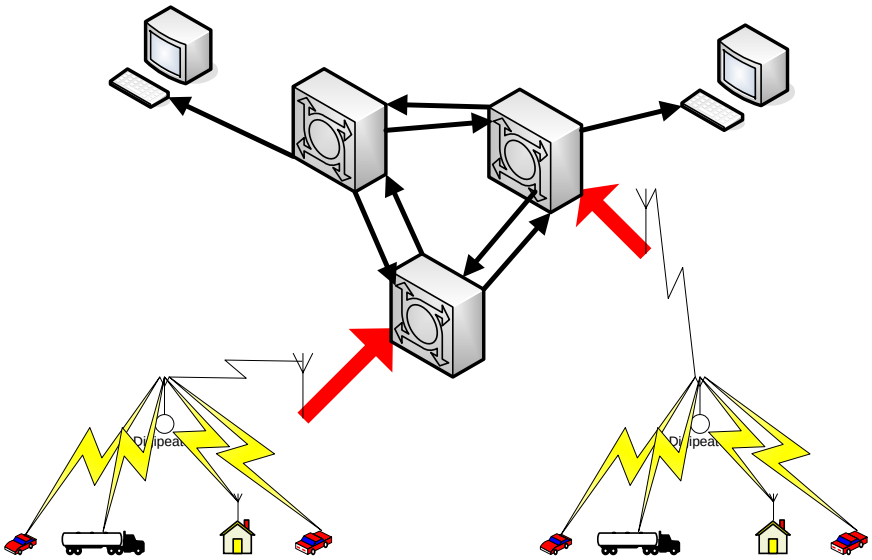

图 1－3　APRS－IS 核心体系结构

APRS－IS（Automatic Position Reporting System-Internet Service）是一种基于 Internet 的网络的通称，这种网络相互连接了遍及全世界的各种 APRS 无线电网络。其中的 APRS 网关负责射频（RF）网络的 AX.25 分组和 Internet 网络的 TCP/IP 分组的转换和转发。

1．4．2　APRS 数据格式

在链路层，APRS 使用 AX.25 协议，专有的利用未编号信息（UI－Unnumbered Information）帧。这意味着 APRS 以无连接模式运行，AX.25 帧被传送出去并不期望任何相应，另一端的接收也没有保证。

AX.25 UI 帧的格式如图 1－4 所示。

	AX.25 UI-FRAME FORMAT								
	Flag	Destination Address	Source Address	Digipeater Addresses (0-8)	Control Field (UI)	Protocol ID	INFORMATION FIELD	FCS	Flag
Bytes:	1	7	7	0–56	1	1	1–256	2	2

图 1－4　AX.25 UI 帧格式

- **Flag** —— 位于帧的两端，是个位串，值为 0x7e，用来分隔每一个帧。
- **Destination Address** —— 这个域可以包含一个 APRS 的目的地呼号或者

APRS 数据。APRS 数据被编码以保证这个域符合标准的 AX.25 呼号的格式（比如，6 个数字或文字字符加上一个 SSID）。如果 SSID 非零，它指定了一个普通的 APRS 数字中继路径。

- **Source Address** —— 这个域包含发送电台的呼号和 SSID。在某些情况下，如果 SSID 非零，它指定了一个 APRS 显示的符号码。
- **Digipeater Address** —— 包含 0 到 8 个数字中继的呼号。
- **Control Field** —— 这个域被设为 0x03（UI 帧）。
- **Protocal ID** —— 这个域被设为 0xf0。
- **Information Field** —— 这个域包含更多的 APRS 数据。第一个字符是 APRS 数据类型标识符。
- **Frame Check Sequence** —— 这个域用来校验收到的帧的完整性。

Source Address 的 SSID 如图 1－5 所示。

SSID-Specified Icons in the AX.25 Source Address Field

SSID	Icon	SSID	Icon
-0	*[no icon]*	-8	Ship (power boat)
-1	Ambulance	-9	Car
-2	Bus	-10	Motorcycle
-3	Fire Truck	-11	Balloon
-4	Bicycle	-12	Jeep
-5	Yacht	-13	Recreational Vehicle
-6	Helicopter	-14	Truck
-7	Small Aircraft	-15	Van

图 1－5

Destination Address 的 SSID 如图 1－6 所示。

APRS Digipeater Paths in Destination Address SSID

SSID	Path	SSID	Path
-0	Use VIA path	-8	North path
-1	WIDE1-1	-9	South path
-2	WIDE2-2	-10	East path
-3	WIDE3-3	-11	West path
-4	WIDE4-4	-12	North path + WIDE
-5	WIDE5-5	-13	South path + WIDE
-6	WIDE6-6	-14	East path + WIDE
-7	WIDE7-7	-15	West path + WIDE

图 1－6

Information Field 如图 1－7 所示。

	Generic APRS Information Field			
	Data Type ID	*APRS Data*	*APRS Data Extension*	*Comment*
Bytes:	1	n	7	n

图 1－7

1．5　本章小结

APRS 作为一种新兴的业余无线电活动，正被中国的业余无线电爱好者所关注。无论台湾、香港还是大陆的 HAM 们都已开始进行实践。在本章中，从 APRS 的历史与发展开始，我们初步了解了 APRS 的硬件，软件，设备，工作原理，协议等方面，对 APRS 既有了整体的概念，也有了细节的理解，为设计开发良好的 APRS 客户端软件系统打下了基础。

第二章 APRS 客户端软件系统总体设计

APRS 客户端软件是 APRS 基本系统的重要组成部分，它主要负责电子地图的绘制和 APRS 电台的显示，完成自动位置报告的功能。如果没有 APRS 客户端软件的配合，硬件和其它软件的功能再强大也是无济于事的，因为用户无从得知 APRS 电台的位置，也就不存在自动位置报告了。为了避免 APRS 成为具有丰富无线电知识和计算机知识的专业人士的专利，为了让 APRS 普及到一般用户，一个简单易用、直观明了的 APRS 客户端软件就成了关键。我们的目标就是设计开发一个“傻瓜”型的 APRS 客户端软件系统，用户不必具有任何的无线电知识和

丰富的计算机知识，只要会上网，就可以顺利地使用。下面我们就从编程语言和程序类型的选择、地图类型的选择、功能介绍、模块划分几个方面对系统作一个总体设计。

2．1　编程语言与程序类型选择

Java 语言自从 1996 年 2 月 SUN 公司正式发布以来，因其广泛的用途及在 Web 上的独特功能，越来越来受到计算机程序员的喜爱，正在成为全世界成千上万的程序员广泛使用的现代技术。正如比尔·盖茨所说：“Java 是有史以来最卓越的编程语言”。

Java 语言可以创建三类程序：Application Program（应用程序）、Applet（小应用程序）、Handler（处理程序）。通常的应用程序（Application Program）是运行在本地计算机上的，即通过敲入命令或用鼠标双击来运行。Java 的 Applet 不同于通常的应用程序，甚至不同于 Java 的应用程序。Applet 是专门设计成在 Web 页面中运行，它只能运行在浏览器环境内。

当一个 Java Applet 设计好后，必须嵌入到 Web 页面中。其方法是通过 HTML 命令描述 Applet，用 HTML 文件的<APPLET>标记来调用并且当 Web 浏览器载入 Web 页面时同时载入 Applet。当用户访问这个页面时，它就被下载到用户的计算机中开始运行。因为 Applet 每次运行都必须从 Web 上下载到用户的计算机中，所以比 Applet 比大多数应用程序小，这样可以减少 Applet 的下载时间，故有人称 Applet 为小应用程序。

Java Applet 的这一独特特点使得 Java 成为最适合于 Web 网络的编程语言。当然，测试小应用程序是可以在浏览器之外使用 Applet Viewer 工具运行，而不需要浏览器的辅助。这是为了有助于方便地开发程序，从而减少启动浏览器所需要的时间和内存。

安全性对于网络访问来说是至关重要的。病毒和黑客地频频出现使得上网的人们时常为安全而担忧。Java 被认为是一种安全的语言。安全性是 Java 语言，特别是 Java Applet 的重要特性，是 Java 环境的最根本的部分。Java 使我们能够创建没有病毒和黑客的系统。

Java 具有对内存的保护。Java 是比 C/C++简单的编程语言。与 C/C++相比，Java 完全抛弃了指针，不需要使用指针就可以通过引用来传递所有的数组和对

象。这样做的结果是不可能进入内存地址，使程序不可能在内存的某个地址处偶然或人为地重写数据。这使得 Java 具有更高的安全性。

Java 具有安全性检查。JVM 在执行 Applet 字节码之前先检查它们。如果代码不是合法的 Java 代码，那么它就不执行。在异常的代码出现问题之前先阻塞它。比如，不允许进行对内存的非法访问、非法类访问和非法数据访问。因此，潜在的病毒就没有办法访问数据结构、对象和内存地址。

Applet 访问资源有所限制。对于 Applet 来说，它的安全性还在于：所有的 Java Applet 程序都被当作是在受托的环境中执行未受托的代码。这就意味着从 Web 上下载的所有 Applet 程序都受限于它们到达用户机器中时他们可以做的事情。即使在通过字节码检查过程后，Applet 程序也不可能访问用户计算机上的文件。它们不能在它们所在的 Web 服务器之外进行网络连接，这就防止了 Applet 程序访问除了下载它们的站点以外的 Web 资源。作为附加的安全手段，Applet 程序被禁止执行任何代码（如外部代码库和应用程序）。

由于 Applet 只能在 Web 页面运行，导致 Applet 在组织结构上有某些独特的特点。Applet 不在运行的线程的控制之下，它仅仅对浏览者或观察者的要求作出反应。这就是说，Applet 具有响应事件的能力。对于特定事件（如：当包含 Applet 的 Web 页被用户观察或离开时），Applet 是通过调用一系列标准方法（methods）来处理的。

综上所述，虽然 Applet 的执行极受限制，通常被称为是“在沙盒里头做事情”，因为有个无形的家伙（Java 执行期安全系统）时刻在进行监督。但是 Applet 具有足够的优点来吸引我们的目光，还有就是 Applet 不再有安装的问题，它拥有真正与平台无关的能力，所以不需要为不同平台修改代码，也不会让你的用户因为安装而苦恼。Applet 的这些优点正是我们设计开发“傻瓜”型的 APRS 客户端软件系统所需要的，所以我们选择用 Java 语言开发一个 Applet 类型的 APRS 客户端软件系统，让用户以浏览网页的方式来使用。

2．2　地图类型选择

地图是 APRS 客户端软件系统中最为基本的组成部分，它是自动位置报告的基础。没有地图的绘制，APRS 电台的显示就没有任何意义。因为只有在地图上才能根据 APRS 电台的经纬度在对应的位置上画出电台的图标，完成自动位置报

告的功能。

APRS 地图有两种类型:向量地图和光栅地图。向量地图是一个包含计算机用来根据要求绘制地图的原始数据的文件。线条和其它对象都被定义成坐标对的形式。向量地图的特征能够被动态编辑。向量地图的线条定义如下所示。光栅地图

```
BEGIN LINE
32.744033,-96.894967
32.744033,-96.894833
32.743967,-96.894767
32.743933,-96.894767
32.743883,-96.894767
32.743817,-96.894767
32.743700,-96.894700
32.743567,-96.894617
32.743417,-96.894567
32.743250,-96.894517
32.743100,-96.894483
32.743000,-96.894417
32.742983,-96.894300
END
```

主要是一幅地图图像和一个包含相关地理信息的文件。地图特征已经确定，很难再进行修改。地图图像文件的格式有.GIF，.JPG，GeoTiff 等。光栅地图是 UI-View 使用的地图类型，Xastir，WinAPRS 等其它 APRS 客户端软件也能使用。

由于在中国 UI-View 是使用最为广泛的 APRS 客户端软件，它的地图已广为“Ham”们所熟悉，而且 UI-View 地图的屏幕坐标和经纬度之间的换算也比较容易，所以我们选择 UI-View 地图作为我们系统所使用的地图。UI-View 地图有一个特征：地图的顶部是正北方向，经度随着 X 轴线性变化，纬度随着 Y 轴线性变化。如果不符合这个特征，在进行屏幕坐标和经纬度之间的换算时就会发生错误。UI-View 地图和所有的光栅地图一样，包含一个地图图像文件和一个包含相关地理信息的文本文件。这个文本文件主要包含地图左上角和右下角的经纬度。例如图 2－1 所示的世界地图，它的对应文本文件内容为：

```
180.00.00W,90.00.00N
180.00.00E,90.00.00S
```

Base Map of The World

第一行表示左上角的经纬度，第二行表示右下角的经纬度，经度和纬度之间都用逗号隔开，W 表示西半球，E 表示东半球，N 表示北半球，S 表示南半球。经纬度采用 DD.MM.mm 的格式来表示，其中 DD 表示度，MM 表示分的整数部分，mm 表示分的小数部分。第三行是对地图的简单描述。

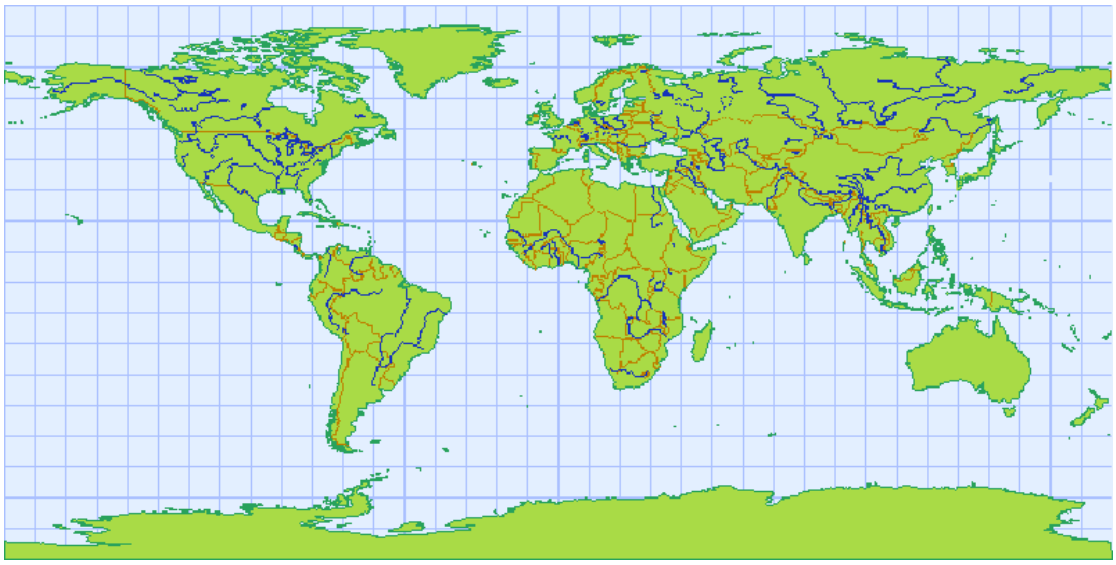

图 2—1

2．3　功能介绍

Internet 的迅速崛起和在全球范围内的飞速发展，使万维网（World Wide Web 简称 WWW 或 Web）成为高效的全球性的信息发布渠道。这一技术正在以很快的速度进入每家每户，它将地球变成一个小小的村落。这给了 APRS 普及的机会。要让缺乏无线电知识和计算机知识的人也能使用 APRS，除了 Internet 上众多 APRS 服务器的强力支持外，一个使用简单、功能完善的 APRS 客户端也是必不可少的。根据需求，APRS 客户端必须具备以下的基本功能：

- **地图显示**

在客户端上要能直观的定位被跟踪的物体的位置，实现自动位置报告的功能，那么地图的显示是必不可少的。没有地图，就没有经纬度的概念，任何的定位之说也就成了无稽之谈。

- **地图选择**

要能准确的定位所有 APRS 电台的位置，一张世界地图是不可能完成的。由于世界地图的图片大小有限，经纬度就非常的密集，很多的 APRS 电台图标

就会发生重叠，不能清晰的显示出来。所以客户端必须具备能让用户根据自己的需求选择不同地图的功能，通过一个下拉列表框就可以顺利解决。

- **地图移动**

客户端的窗口具有一定的大小，而地图通常会比较大，不可能全部显示出来。为了能让用户看到地图因客户端的窗口太小而被遮盖的部分，就必须具有地图移动的功能。通过鼠标的托拽，地图可以自由的移动，这是一个用户早已习惯的设计。

- **地图定位**

地图上的每一点都有其确定的位置坐标（经纬度）。用户在观察地图的时候经常会想知道某一些点的经纬度，尽管那些位置并没有 APRS 电台的出现。让鼠标在地图上移动时自动报告各点的经纬度，就可以很容易的实现地图定位地功能。用户想知道哪一点的经纬度，只需要把鼠标移到那一点就可以了。

- **APRS 电台显示**

APRS 电台的显示是客户端最为关键的部分，只有在地图上显示出 APRS 电台的图标才能让用户得知被跟踪物体的位置。我们要根据从服务器端接收的数据包中 APRS 电台的经纬度在地图上的相应位置进行 APRS 电台图标的绘制。

- **APRS 电台信息查询**

一个 APRS 电台除了经纬度之外还有很多其它的信息，比如源电台的呼号和 SSID、目标电台的呼号和 SSID、时间戳、评论等。通过鼠标双击地图上 APRS 电台图标，显示出相应 APRS 电台的所有信息是一个很重要也很有价值的功能。

2．4　模块划分

我们正在忙碌地建设我们生活的这个世界，而且我们正在电脑空间建立我们物理空间的复制品，网络上甚至有了可以存款的银行。互联网已经渗透到各行各业，信息高速公路上奔驰着越来越多的信息。随着 Internet 技术的不断发展和人们对 APRS 的需求，利用 Internet 在 Web 上发布 APRS 数据，为用户提供 APRS 的各种服务，已经成为众望所归的选择。

根据如图 2－2 所示的 APRS 系统结构图，可以把 APRS 客户端系统划分为

以下几个模块：

1. 包含 Applet 的网页编程模块

用户通过 HTTP 协议请求包含 Applet 的 HTML 页面，当浏览器遇到<applet>和</applet>这一对标记时，就下载相应的 Applet 代码并在本地计算机上执行。虽然这一模块非常简单，但是它是必不可少的。在这一模块中，可以对网页进行美工方面的设计与编程，使网页更加美观，更加吸引用户。

2. Applet 中的 Swing 编程模块

当浏览器下载相应的 Applet 并在本地计算机上开始运行的时候，Swing 编程模块便开始工作了。在 Applet 进行初始化的时候，Swing 编程模块先绘制默认情况下的地图，并且在一个下拉列表框中列出所有可以选择的地图名称，它还制造出一个新的线程用来进行 Socket 连接。接下来，Swing 编程模块就要进行各种相应，如鼠标滑动时要显示滑过各点的经纬度，鼠标左键双击时要显示 APRS 电台的详细信息，鼠标拖拽时要进行地图的移动，从服务器端接收到包含 APRS 电台信息的数据包时要在相应的位置绘制 APRS 电台图标，从下拉列表框中选择地图时要绘制被选中的地图等。

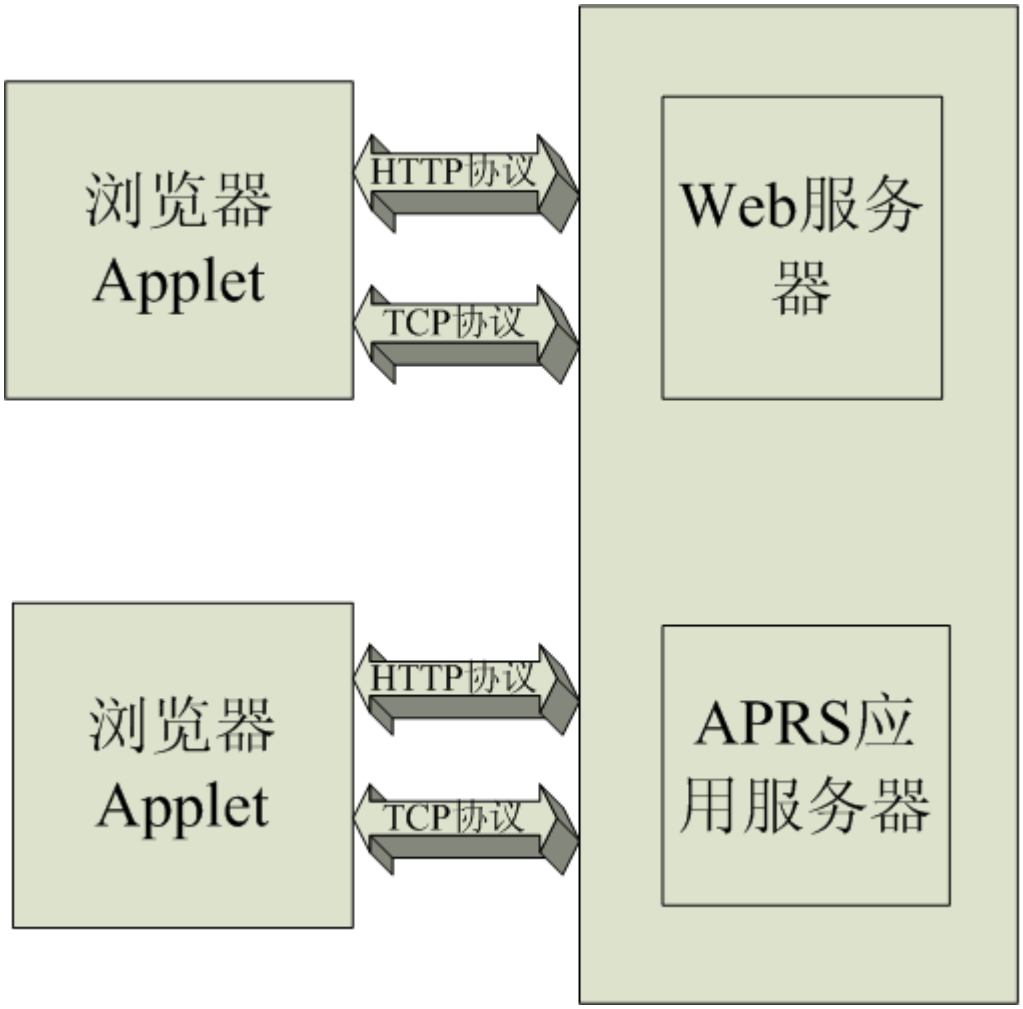

图 2－2　APRS 系统结构图

3.　Applet 中的 Socket 编程模块

Socket 是面向客户/服务器模型设计的，网络上的两个程序通过一个双向的通讯连接实现数据的交换，这个双向链路的一端称为一个 Socket。Socket 通常用来实现客户方和服务方的连接。客户程序可以向 Socket 写请求，服务器将处理此请求，然后通过 Socket 将结果返回给用户。Socket 通信机制提供了两种通信方式：有连接（TCP）和无连接（UDP）方式，分别面向不同的应用需求。使用有连接方式时，通信链路提供了可靠的，全双工的字节流服务。在该方式下，通信双方必须创建一个连接过程并建立一条通信链路，以后的网络通信操作完全在这一对进程之间进行，通信完毕关闭此连接过程。使用无连接方式时其系统开销比有连接方式小，但通信链路提供不可靠的数据报服务，不能保证信源所传输的数据一定能够到达信宿。在该方式下，通信双方不必创建一个连接过程和建立一条通信链路，网络通信操作在不同

的主机和进程之间转发进行。由于 APRS 对数据包正确性，完整性，可靠性的要求，采用 TCP 方式来进行通信。客户端和服务器端发送的数据包都由两部分组成：Packet head 和 Packet data。Packet head 指定数据包的类型,分为 REG（客户端发送到服务器端的注册包）、URR（客户端发送到服务器端的反注册包）、DTA（服务器端发送到客户端的数据包）、ACK（Acknowlegement）四种。具体的通信过程如下：

- 服务器在 1401 端口监听，客户端以 TCP 方式连接服务器。
- 客户端成功连接至服务器后，须发送一个 REG 注册包到服务器。
- 服务器认为注册包信息完整正确后会回复一个 ACK 给客户端，告诉客户端连接成功。
- 服务器会在之后的离散时间里随机地给客户端发送数据包，直到收到客户端的反注册包或客户端的 TCP 连接断开为止。
- 这些数据包是基本上参照 APRS 标准协议的，含有站点的经纬度，时间戳，自定义文本等信息，客户端需要将这些信息解析出来，并显示在地图的相应坐标上。
- 客户端最后发送反注册包 URR 并断开连接。

2．5　本章小结

在本章中，我们根据简单易用、直观明了、便于 APRS 普及的原则完成了系统的总体设计。我们选定 Java Applet 作为系统的程序类型，采用 UI-View 地图作为系统的地图。接下来，我们介绍了系统所要具备的功能，并对系统进行了模块划分。

第三章 APRS 客户端软件

系统详细开发

在开发软件系统的时候，我们通常会把系统划分为若干个功能相对分离的模块，然后逐个模块的进行详细开发。对于 Java 来说，一个模块就是相互关联，相互使用的一些类的集合。在进行总体设计的时候，我们已经把 APRS 客户端软件系统划分为三个模块：包含 Applet 的网页编程模块，Applet 中的 Swing 编程模块，Applet 中的 Socket 编程模块。除了网页编程模块之外，另外的两个模块我们都采用 Java 来开发，也就是我们要开发一些类来实现这两个模块要实现的功能。接下来，我们就逐个模块的进行详细开发，并对其中遇到的技术难点和解决之法进行详细说明。在这之前让我们来看一下系统的文件结构图，如图 3－1 所示。

3．1　网页编程模块详细开发

Applet 小应用程序必须嵌入到 HTML 页面中，才能得到解释执行；同时 Applet 可以从 Web 页面中获得参数，并和 Web 页面进行交互。含有 Applet 的网页的 HTML 文件代码中必须带有<applet>和</applet>这样一对标记，当支持 Java 的网络浏览器遇到这对标记时，就将下载相应的小应用程序代码并在本地计算机上执行该 Applet。这个 HTML 文件中关于 Applet 的信息至少应包含以下三点：

- 字节码文件名（编译后的 Java 文件，以.class 为后缀）
- 字节码文件的地址
- 在网页上显示 Applet 的方式

一个 HTML 文件增加 Applet 有关的内容只是使网页更富有生气，它并不会改变 HTML 文件中与 Applet 无关的元素。

在我们的系统中，本模块就是一个 APRSClient.html 文件，其中最为重要的代码就是<applet code="APRSClient.class" width="500" height="400"></applet>，它是整个 HTML 文件代码的关键所在。下面我们就对与 Applet 应用有关的参数作一个简单说明。

- CODE 标记

CODE 标记指定 Applet 的类名，WIDTH 和 HEIGHT 标记指定 Applet 窗口的象素尺寸。在 Applet 语句里还可使用其它一些标记。

- CODEBASE 标记

CODEBASE 标记指定 Applet 的 URL 地址。Applet 的通用资源定位地址 URL，它可以是绝对地址，也可以是相对于当前 HTML 所在目录的相对地址。如果 HTML 文件不指定 CODEBASE 标记，浏览器将使用和 HTML 文件相同的 URL。

- ATL 标记

虽然 Java 在 WWW 上很受欢迎，但并非所有浏览器都对其提供指出。如果某浏览器无法运行 Java Applet，那么它在遇到 Applet 语句时将显示 ATL 标记指定的文本信息。

- ALIGN 标记

ALIGN 标记可用来控制把 Applet 窗口显示在 HTML 文档窗口的什么位置。与 HTML<LMG>语句一样，ALIGN 标记指定的值可以是 TOP、MIDDLE 或 BOTTOM。

- VSPACE 与 HSPACE 标记

VSPACE 和 HSPACE 标记指定浏览器显示在 Applet 窗口周围的水平和竖直空白条的尺寸，单位为象素。

- NAME 标记

NAME 标记把指定的名字赋予 Applet 的当前实例。当浏览器同时运行两个或多个 Applet 时，各 Applet 可通过名字相互引用或交换信息。如果忽略 NAME 标记，Applet 的名字将对应于其类名。

- PARAM 标记

通用性是程序设计所追求的目标之一。使用户或者程序员能很方便地使用同一个 Applet 完成不同地任务是通用性地具体表现。从 HTML 文件获取信息是提高 Applet 通用性地一条有效途径。假设编制了一个把某公司的名字在屏幕上卷动的 Applet。为了使该 Applet 更加通用，则可以使该 Applet 从 HTML 文件获取需要卷动的文本信息。这样，若想显示另一个公司的名字，用不着修改 Java Applet 本身，只需要修改 HTML 文件即可。PARAM 标记可用来在 HTML 文件里指定参数，格式如下所示：

PARAM Name＝“name” Value=“Sky”

Java Applet 可调用 getParameter 方法获取 HTML 文件里设置的参数值。

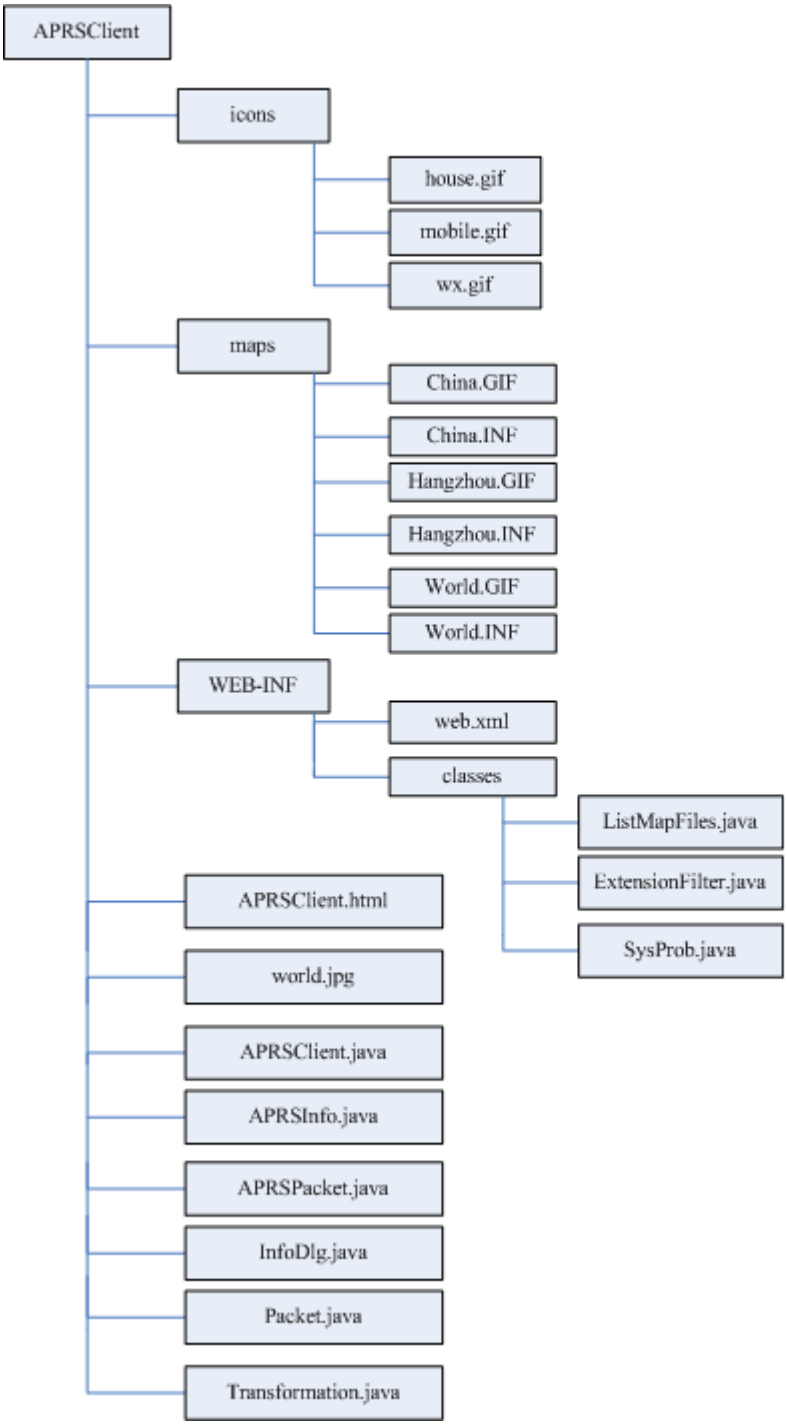

图 3－1 系统文件结构图（.class 文件省略）

3．2 Swing 编程模块详细开发

本模块是整个系统最为重要的模块，所有与用户界面和用户交互有关的功能全都由本模块来实现。在本模块众多类中，最为重要的是一个继承自 javax.swing.JApplet 类的一个子类 APRSClient，它也是整个系统的关键和精华所

在。接下来，我们就详细介绍这些类，以及在设计开发这些类时所遇到的问题及其解决之法。

3．2．1　类开发

APRSClient 类是 JApplet 类的一个子类，它从 JApplet 类中继承了 init()函数，而这个函数是由浏览器最先自动调用的，因此可以说 init()函数是整个程序的入口。在 APRSClient 类的 init()函数中，我们取得了 maps 目录下所有地图图像文件的名字，并把这些名字放在了一个 ArrayList 数据结构中。为了得到默认情况下的当前地图，我们把 ArrayList 数据结构中第零位的地图图像名字赋值给 APRSClient 类的一个字符串类型的数据成员 currentMap。根据 currentMap 的值，我们取得了当前地图对应的信息文件。通过解析此信息文件的内容，我们得到了地图左上角和右下角的经纬度，并把它们分别赋值给了 APRSClient 类的字符串类型的数据成员：leftTopLongitude，leftTopLatitude，rightBottomLongitude，rightBottomLatitude。关于如何取得 maps 目录下所有地图图像文件的名字以及如何取得当前地图对应的信息文件，这是开发本模块时所遇到的技术难题，将在下面进行详细讨论。接下来我们创建了两个内部类的对象，这两个内部类是继承自 JPanel 类的两个子类：MapPanel 类和 BottomPanel 类。MapPanel 类实现了地图和 APRS 电台图标绘制，鼠标移动时显示当前位置经纬度，鼠标拖拽时移动地图，鼠标双击 APRS 电台图标时弹出对话框显示此 APRS 电台的详细信息，接收到 Message 类型的数据包时弹出对话框显示此消息的内容。BottomPanel 类比较简单，它创建了两个文本框供鼠标移动时当前位置经纬度的显示，它还创建了一个下拉列表框供 ArrayList 数据结构中所有地图名字的显示，并让用户能够根据不同需求更换当前地图。在 MapPanel 类的构造函数中，我们先创建了一个 HashMap 类型的数据结构，通过 Socket 编程模块，可以把所有接收到的数据包存放在 HashMap 数据结构中。如果是 Message 类型的数据包，则以字符串“MESSAGE”作为 key；如果是 Station 类型的数据包，则以数据包中 APRS 电台的呼号作为 key。接下来我们添加了所有必须的鼠标事件的响应函数。地图和 APRS 电台图标的绘制是由 MapPanel 类中的 paint()函数来完成的。初始化程序或者移动地图或者选择地图或者接收到新的 Station 类型的数据包时，都会调用这个函数来重新绘制地图和 APRS 电台图标。在这个函数中，我们还用 APRS 电台图标所在的

矩形区域来构造 Rectangle2D 的对象，放在了一个 ArrayList 数据结构中；同时以这些 Rectangle2D 对象作为 key，以对应的 APRS 电台的信息数据包作为 value 放在了一个 HashMap 数据结构中。这两个数据结构会在鼠标双击事件的响应函数中被使用。由于 paint()函数和各个鼠标事件的响应函数都比较简单，就不多加解释了。在鼠标双击事件的响应函数，鼠标移动事件的响应函数以及 paint()函数中都要进行屏幕坐标和地图经纬度之间的换算，所以我们设计开发了一个 Transformation 类。在构造 Transformation 类时，我们需要传递地图左上角和右下角的经纬度，地图左上角相对于屏幕窗口原点的坐标以及地图图像的长度和宽度作为其构造函数的参数。在 Transformation 类中，我们可以把屏幕坐标（象素表示）转换成 DD.MM.mm 格式的经纬度（字符串），也可以把以秒表示的经纬度（int 型整数）转化成屏幕坐标（象素表示）。为了区分东西经和南北纬，我们假设东经和北纬是正数，西经和南纬是负数。由于我们选择了 UI－View 地图作为我们系统的地图，而 UI－View 地图最大的特点就是地图的顶部是正北方，经度随 X 轴线性变化，纬度随 Y 轴线性变化，所以换算比较简单。在具体的编码过程中，纬度随着 Y 轴的递增而递减，换算较经度比较方便。具体的函数代码如下所示：

```
public String toLatitude(int y)
{
    int top = toSeconds(leftTopLatitude);
    int bottom = toSeconds(rightBottomLatitude);
    int latitude = top-(int)(top-bottom)*(y-originalY)/imageHeight;
    //System.out.println(latitude);
    return toLat(latitude);
}
```

这是把 Y 坐标转换成纬度（字符串）的函数，而把纬度（int 型整数）转化成 Y 坐标的函数代码如下所示：

```
public int toY(int latitude)
{
    int top = toSeconds(leftTopLatitude);
    int bottom = toSeconds(rightBottomLatitude);
    int y = originalY+(int)imageHeight*(top-latitude)/(top-bottom);
```

```
        return y;
    }
```

在经度和 X 坐标互相换算的过程中，由于东西经 180 度是重合的，经度随着 X 轴的递增并不是规则的线性变化，存在着正负 180 的突变，所以必须分情况进行计算，具体代码如下所示：

```
    public String toLongitude(int x)
    {
        int left = toSeconds(leftTopLongitude);
        int right = toSeconds(rightBottomLongitude);
        if(left<right)
        {
            int longitude = left+(int)(right-left)*(x-originalX)/imageWidth;
            //System.out.println(longitude);
            return toLong(longitude);
        }
        else
        {
           int distance180=(int)imageWidth*(180*3600-left)/(360*3600-left+right);
            if(x-originalX <= distance180)
            {
                int longitude=left+(int)(360*3600+right-left)*
                        (x-originalX)/imageWidth;
                //System.out.println(longitude);
                return toLong(longitude);
            }
            else
            {
                int longitude=-180*3600+(int)(right+180*3600)*
                        (x-originalX-distance180)/(imageWidth-distance180);
                //System.out.println(longitude);
                return toLong(longitude);
            }
        }
```

```
    }
```

这是把 X 坐标转换成经度（字符串）的函数，而把经度（int 型整数）转化成 X 坐标的函数代码如下所示：

```
    public int toX(int longitude)
    {
        int left = toSeconds(leftTopLongitude);
        //System.out.println("left:"+left);
        int right = toSeconds(rightBottomLongitude);
        if(left<right)
        {
            int x = originalX+(int)imageWidth*(longitude-left)/(right-left);
            return x;
        }
        else
        {
             int distance180=(int)imageWidth*(180*3600-left)/
                    (360*3600-left+right);
            if(longitude>=left)
            {
                int x=originalX+(int)(longitude-left)*imageWidth/
                        (360*3600+right-left);
                return x;
            }
            else
            {
                int x=originalX+distance180+(int)(longitude+180*3600)*
                        (imageWidth-distance180)/(right+180*3600);
                return x;
            }
        }
    }
```

介绍完本模块中重要的类以后，就让我们接着介绍在开发这些类的过程中遇到的技术难点以及解决之法。

3．2．2　技术难点及解决之法

在开发 Swing 编程模块时，我们遇到了两个难题。一个是在 Applet 中读取文件，另一个是在 Applet 中 list 某个目录下的文件。

- **在 Applet 中读取文件**

在 Java Applet 中出于安全性的考虑，Applet 是不允许对文件进行操作的，不仅不允许写文件，而且不允许读文件。尽管我们在编制 Applet 时即使用了文件操作的语句 Java 也不会报错，在 Applet Viewer 中调试时也能够正常运行，但当我们在浏览器中运行这个 Applet 时浏览器就会报错。解决这个问题的诀窍就是我们不要将这些服务器上的文件作为普通文件来处理，而是将它们作为网络资源来获取它们的内容。在 Java 中可用于获取网络资源的类主要有两种，一是 URL 类，另一个是 URLConnection 类。两个类都提供了以字节流的方式读取资源信息的方法，而且可以对资源信息的类型作出判断，以便作相应的处理。不同之处是 URLConnection 类可提供的信息比 URL 类要多得多，它除了可以获取资源数据外，还可以提供资源长度、资源发送时间、资源最新更新时间、资源编码、资源的标题等许多信息。下面就是我们使用 URL 类和 URLConnection 类获取当前地图对应的信息文件内容并对 APRSClient 类的成员变量 leftTopLongitude，leftTopLatitude，rightBottomLongitude，rightBottomLatitude 进行赋值的函数代码。

```
public void setLongAndLat()
{
   String infoFileName = currentMap + ".INF";
   try
   {
        URL url = new URL(getCodeBase(),
                "/APRSClient/maps/"+infoFileName);
        //System.out.println(url);
        URLConnection con = url.openConnection();
        con.setUseCaches(false);
        InputStream in = con.getInputStream();
```

```
        BufferedReader d = newBufferedReader(
                new InputStreamReader(in));

        String line = d.readLine().trim();
        //System.out.println(line.length());
        int comma = line.indexOf(",");
        leftTopLongitude = line.substring(0,comma).trim();
        leftTopLatitude = line.substring(comma+1,line.length()).trim();
        //System.out.println(leftTopLongitude+":"+leftTopLatitude);

        line = d.readLine().trim();
        comma = line.indexOf(",");
        rightBottomLongitude = line.substring(0,comma).trim();
        rightBottomLatitude = line.substring(comma+1,line.length()).trim();
        //System.out.println(rightBottomLongitude+":"+rightBottomLatitude);
    }
    catch (Exception e)
    {
        System.out.println("set longitide and latitude fail!!");
        e.printStackTrace();
    }
}
```

- **在 Applet 中 list 某个目录下的文件**

同样出于安全性的限制，在 Applet 中是不允许使用 list()函数的。虽然使用了 list()函数的 Applet 在 Applet Viewer 中可以正常运行，但在浏览器中运行时就会发生错误。为了解决这个问题，我们采用了 Applet 和 Servlet 通信的机制。Servlet 完全运行在服务器端，没有安全性方面的限制，可以调用 list()函数而正常运行。它可以处理 web 请求，并返回数据或者 HTML。在我们的程序中，我们创建了一个 HttpServlet 类的子类 ListMapFiles。在 ListMapFiles 类中，我们调用了 list()函数，把得到的地图图像文件名放在了一个字符串数组里面，再通过如下代码把地图名字（去掉后缀的地图图像文件名）写到返回请求的数据流中。

```
for(int i=0;i<maps.length;i++)
{
    out.println(maps[i].substring(0,maps[i].length()-4));
}
```

由于使用了 Servlet，所以我们必须选择一个支持 Servlet 的 Web 服务器。Jakarta Tomcat 服务器是在 Sun 公司的 JSWDK（JavaServer Web DevelopmentKit，是 Sun 公司推出的小型 Servlet/JSP 调试工具）的基础上发展起来的一个优秀的 Servlet/JSP 容器，它是 Apache－Jakarta 软件组织的一个子项目。它不但支持运行 Servlet 和 JSP，而且还具备了作为商业 Java Web 应用容器的特征。所以我们就选用了 Tomcat 作为我们的 Web 服务器。为了让 Tomcat 启动时自动部署本网站，我们把整个 APRSClient 文件夹放在了 Tomcat 安装目录下的 webapps 目录下。我们把 ListMapFiles 类及其它用到的类放在了 WEB－INF 下的 classes 目录下，并在 web.xml 文件中添加了<servlet>和<servlet-mapping>元素，具体如下所示：

```
<servlet>
 <servlet-name>ListMapFiles</servlet-name>
 <servlet-class>ListMapFiles</servlet-class>
</servlet>

<servlet-mapping>
 <servlet-name>ListMapFiles</servlet-name>
 <url-pattern>/ListMapFiles</url-pattern>
</servlet-mapping>
```

这样我们就可以在 APRSClient 类中通过 URL 请求 ListMapFiles 这个 Servlet 并且得到 maps 目录下所有地图的名字。具体的代码和前面在 Applet 中读取文件很类似，如下所示：

```
try
{
    //System.out.println(getCodeBase());
    URL url = new URL(getCodeBase(),"/APRSClient/ListMapFiles");
    //System.out.println(url);
    URLConnection con = url.openConnection();
```

```
        con.setUseCaches(false);
        InputStream in = con.getInputStream();
        BufferedReader d = new BufferedReader(new InputStreamReader(in));
        String line = d.readLine();
        while(line != null)
        {
            maps.add(line);
            line = d.readLine();
        }
    }
    catch ( Exception e)
    {
        System.out.println("list map image files fail!!");
        e.printStackTrace();
    }
```

3．3　Socket 编程模块详细开发

本模块负责建立和 APRS 应用服务器的 TCP 连接，并解析从服务器端接收到的数据包，便于 Swing 编程模块的使用。同样地，我们将详细介绍本模块中的类以及在开发过程中所遇到的技术难点及其解决方法。

3．3．1　类开发

本模块的核心是 APRSInfo 类，它是继承自 Thread 类的一个子类，所以代表的就是一个完成特定任务的线程。在 APRSClient 类中，我们创建了 APRSInfo 类的一个对象并且调用了它的 start()函数，也就是说，我们在 APRSClient 类中制造了一个新的线程并且启动了这个线程。所有继承 Thread 类的子类都必须覆盖 Thread 类中的一个函数——run()函数。run()函数是由 start()函数自动调用的，它会完成线程所要完成的任务。所以，我们在编写代码的时候，要把有关线程所要完成任务的代码都放在 run()函数之中。当然，APRSInfo 类也不会例外。在 APRSInfo 类中，我们覆盖了 run()函数，在其中添加了和 APRS 应用服务器建立连接并解析收到的数据包的代码。由于 Applet 不能建立或接受外来的 Socket 连

接。所谓外来的是指这个连接超出了提供这个 Applet 类文件的主机（不是提供引用这个 Applet 的 HTML 所在的主机）。由于 APRSInfo 类是在 APRSClient 类中被使用的，而 APRSClient 类就是一个 Applet。所以，如果在 APRSInfo 类中建立或者接受了外来的 Socket 连接，APRSClient 在运行时就会发生错误。为了解决这个问题，我们把 Web 服务器和 APRS 应用服务器建在了同一台机子上。由于在 run()函数中和 APRS 应用服务器建立 Socket 连接时需要用到服务器的 IP 地址，所以我们在 APRSClient 类中通过取得 Web 服务器的 IP 地址并通过 APRSInfo 构造函数的参数来进行传递。在 APRSClient 类中取得 Web 服务器 IP 地址的代码如下所示：

```
try
{
    host = InetAddress.getByName(getCodeBase().getHost());
}
catch (IOException e)
{
    e.printStackTrace();
}
```

其中 host 是一个 InetAddress 类型的变量，表示了 Web 服务器的 IP 地址。如下代码表示了如何把 Web 服务器的 IP 地址传递给 APRSInfo 类。

```
getInfo = new APRSInfo(this,stations,host);
getInfo.start();
```

getInfo 就是一个 APRSInfo 类的对象，这两行代码表示制造了一个 APRSInfo 类所代表的线程并且启动了这个线程。这两行代码是写在 MapPanel 内部类的构造函数之中的，所以参数中的 this 表示 MapPanel 类的对象，而 stations 表示存放解析后的数据包的 HashMap 数据结构。在 APRSInfo 类中，使用这两个变量的代码如下所示：

```
APRSPacket packet = new APRSPacket(str);
if(packet.isMessage())
{
    synchronized(stations)
    {
        stations.put("MESSAGE",packet);
```

```
            }
        }
        else
        {
            String callSign = packet.getCallSign();
            synchronized(stations)
            {
                stations.put(callSign,packet);
            }
        }
        mapPanel.repaint();
```

APRSPacket 是对接收到的数据包进行解析并重新打包的类。我们把重新打包的数据以 key－value 的形式存放在 stations 这个 HashMap 数据结构中，并调用 mapPanel 的 repaint()函数让 mapPanel 重画地图和 APRS 电台的图标，而 mapPanel 在重画的时候也就可以很方便地使用 stations 数据结构中的数据包了。APRSPacket 类接收有特定格式的字符串类型的数据包，然后对数据包进行了解析和重新打包。由于 APRSPacket 类进行的都是对字符串的处理，所以比较简单，我们也就不多加解释了。

3．3．2　技术难点及解决之法

在和 APRS 应用服务器建立连接后，需要发送一个正确完整的 REG 注册包才能得到确认，之后才能接收到有关 APRS 电台的数据包。但是 APRS 应用服务器是用 C/C++开发的，而注册包是 C/C++中相对于 Java 独有的结构体格式，并且 Java 和 C/C++具有不同的字节序，所以按照结构体中的变量直接传送变量值是不能解决问题的。为此，我们开发了一个 Packet 类专门用来处理 Java 和 C/C++字节序不同的问题。注册包的结构体如下所示：

```
struct Packet
{
   int type;
   int len;
   CString sno;
};
```

在 Packet 类中，我们改变了 int 型整数和字符串的字节序，这样就可以和 C/C++中结构体相匹配了。具体代码如下所示：

```
public class Packet
{
    private byte[] buf = new byte[18];

    private static byte[] toLH(int n)
    {
        byte[] b = new byte[4];
        b[0] = (byte) (n & 0xff);
        b[1] = (byte) (n >> 8 & 0xff);
        b[2] = (byte) (n >> 16 & 0xff);
        b[3] = (byte) (n >> 24 & 0xff);
        return b;
    }

    public Packet(int type,int len,String sno)
    {
        byte[] temp = toLH(type);
        System.arraycopy(temp, 0, buf, 0, temp.length);
        temp = toLH(len);
        System.arraycopy(temp, 0, buf, 4, temp.length);
        temp = sno.getBytes();
        System.arraycopy(temp, 0, buf, 8, temp.length);
    }

    public byte[] getBuf()
    {
        return buf;
    }
}
```

在使用时我们只要创建一个 Packet 类的对象并调用它的 getBuf()函数，返回的就是和 C/C++结构体相匹配的注册包了。

3．4　本章小结

在本章中，我们根据第二章中对系统所划分的模块，逐个地进行了详细开发。无论是模块所使用的关键技术，还是模块中重要的类，或者是开发过程中所遇到的技术难题的解决，都进行了详细的说明，使整个系统的开发一幕了然，跃然纸上。

第四章　　总结

在设计开发这个系统之前，APRS 对于我来说，完全是一个新鲜的闻所未闻的概念。从了解 APRS 开始，到完成整个系统的开发，在此之间，我学会了很多无法从书本上得到的知识。首先，我学会了从网上浩如烟海的信息中寻找有用的资料。无论是 APRS 的介绍，APRS 的协议，APRS 的地图还是在开发过程中遇到的难题的解决，都可以在网上找到合适的资料。其次，我对 Java Applet 开发有了更加深入的了解，包括其安全性方面的限制。Java Applet 的“沙盒操作”是它在安全性方面的优点，但同时也成为了程序员开发 Applet 时的桎梏。所幸的是仍然存在很多的方法可以跨越 Applet 安全性方面的这个限制，比如 Applet 和 Servlet 的通信机制，这也是我在系统开发过程中所用到的。总之，通过毕业设计的这次系统开发，我在编程方面得到了很好的锻炼和成长，在这要忠心地感谢导师的细心指导。

参考文献

- The APRS Working Group，APRS PROTOCOL REFERENCE（Protocol Version 1.0），29 August 2000.
- Bruce Eckel， Java 编程思想（第二版），机械工程出版社，2002 年 9

月，P496－P649。

- 荣新华 BD6CR/4，自动位置报告系统 APRS，2004 年 12 月 18 日@无锡。
- Steve Dimse KO4HD，javAPRS Implementation of the APRS Protocols in Java.
- Cay S.Horstmann，Gray Cornell，Core Java（Seventh Edition），Prentice Hall PTR，August 17 2004.