Rocket Weather App Design Document

Wayne Johnson <joh09024@umn.edu>, Jacob Long <longx329@umn.edu>,
William Beksi <beksi001@umn.edu>, Andrew Schmid <schm1498@umn.edu>

Table of Contents

Introduction

Executive Overview

This design document describes a project that will create a new phone app and its associated server software that will allow a user to be able to access various weather datum from the field. Its original purpose was to serve as a tool for high powered rocketry launches but will also be useful to just about any user interested in aeronautical weather information.

This project is being done to fulfill the requirements for project 1 of the U of M Foundations of Advanced Networking class (CSCI5221).

The members of this project team are Wayne Johnson <joh09024@umn.edu>, Jacob Long <longx329@umn.edu>, William Beksi <beksi001@umn.edu>, Andrew Schmid <schm1498@umn.edu>

Proposal and Background

Did you ever fly a model rocket when you were a kid? Nowadays model rockets can be much larger and go much higher. Some of these rockets are 20 feet tall and can go as high as 4 miles (or more). When you fly such large rockets you have to be very careful and know what you are doing. We have a need for accurate and timely weather information at our club launches. More than just the temperature and wind speed. We need weather information much like what pilots use. This information is available on the Internet, but the file formats leave much to be desired and are not easily readable from a mobile phone.

This project is designed to retrieve information from the National Weather Service (NWS) and National Oceanic and Atmospheric Administration (NOAA) web sites, reformat it, and make it available on a mobile phone.

The information will be downloaded from the NWS and NOAA FTP servers as simple ASCII text files. It will be decoded (not a trivial feat) to extract the needed data. The data will be cached on a local file server and then made available to any smartphone via a mobile app. Since the data will only be downloaded from the FTP servers when it changes (roughly every hour), it will minimize the load on the NWS and NOAA FTP servers. Caching on the local server will provide some continuity of service should there be problems at NWS and NOAA's servers. The data will be transmitted to the mobile phone without any presentation information (color, graphics, resolution, size, or orientation) to save on data transmission costs as well as allowing the individual phone app designer (and end users) the ability of customizing the display as they wish. The intention is to create only a Android app for this project, however there may be a possibility of mirroring the app design onto other mobile platforms if time permits.

Features

The following features are currently planned:

- Display the following data:
    - Current conditions (METARs)
    - 3 day aviation forecast (GFS MOS Guidance)
- Provide the user with the ability on the mobile app to switch from the default (“KROS”) weather station and remember this setting. This default should be provided as an external parameter in the case that the app is rebranded (i.e. used by a different organization).
- Provide an external mobile app parameter defining the IP name of the server in the case that the app is rebranded (i.e. used by a different organization).
- Be easily extended to provide additional features (see Future Development).
- Weather data is cached for each weather station on the server to allow for failures and reducing the load on NWS and NOAA resources.
- Server provides data only to the phones, allowing the phone to provide presentation specific details (i.e. graphics, formatting, etc.)

Design Decisions

- The decision was made to provide an intermediate server between NWS/NOAA and the mobile app to satisfy the project requirements for the class. It also provides caching of the data between NWS/NOAA and the mobile app in addition to failover capabilities in case the NWS/NOAA service is down. The class project requirements are for a non-trivial component on both server and mobile app sides as well as some functionality to solve network problems.

Other server options we considered were:

- Using one of the CSE lab Linux machines. Since we wanted to use LAMP, and do not have root access to the lab machines, this did not meet our requirements.
- An EC2 (Amazon) cloud instance. Again, root access would be an issue.
- Microsoft Azure. Since the student accounts were only temporary, and we wanted to maintain services even after class ended, this was rejected.
- Even a Raspberry Pi. A somewhat unconventional solution and may have sustainability issues.

History

January 31, 2013	Project was originally proposed in an email to Pengkui Luo <pluo@cs.umn.edu>, Zhi-Li Zhang <zhzhang@cs.umn.edu>, and Yanhua Li <yanhua@cs.umn.edu>
February 5, 2013	Original proposal was rejected by Pengkui Luo <pluo@cs.umn.edu> with the request that it be changed to provide a server side component and to be more network related.
February 6th, 2013	A modified proposal was accepted in principle by Pengkui Luo during class.
February 8th, 2013	Proposal was published on the project web site.
February 9th, 2013	We have a team and commencing design.
February 11th, 2013	Design Document started.
March 3rd, 2013	Preliminary Design Document submitted.
April 3rd, 2013	Final Design Document, source code, and progress report submitted.

Design

Project Resources

Source Code Repository

Project source code is hosted publically on Github: https://github.com/wjbeksi/aerowx

Server

The application server will be a VM created under one of the team member's personal ESXi VM hosts at his home. The VM is configured with 2 CPUs, 1Gb Memory, 40Gb disk, and one Ethernet interface connected to an external IP address. The VM will be loaded with CentOS 6.3 (equivalent to RedHat Enterprise Server 6.3) Linux distribution, Apache HTTP server, MySQL and Postgres databases, and Perl/PHP/Python. This setup is commonly known as LAMP.

Android Mobile Platform

The Android SDK is available, free of charge, from developer.android.com. The SDK contains all the development software and provides a device emulator. Several members of the team have Android phones (including Android 2.0), and one member has an ASUS Transformer pad that runs Android 3.0.

External Interfaces/Protocols/Messages

Metar data is available for a specific airport from a National Weather Service Aviation Digital Data Service (ADDS) server site (http://aviationweather.gov/adds). The METARs data follows a format that has been in use for many years. Back in the 1970s, the meteorological data was collected by NWS meteorologists and sent over teletypes using the BAUDOT data format. Nowadays the data is collected by automated weather stations and sent to the NWS over the Internet.

METAR Messages

METARS data contains current weather conditions including specific information for aircrafts. An example of a METARS string is:

```
KROS 122015Z AUTO 21007KT 10SM CLR 00/M07 A2981 RMK AO2 T10051072
```

This translates to:

Conditions at: KROS (RUSH CITY , MN, US) observed 2015 UTC 12 February 2013

Temperature: -0.5°C (31°F)
Dewpoint: -7.2°C (19°F) [RH = 60%]
Pressure (altimeter): 29.81 inches Hg (1009.6 mb)
Winds: from the SSW (210 degrees) at 8 MPH (7 knots; 3.6 m/s)
Visibility: 10 or more miles (16+ km)
Ceiling: at least 12,000 feet AGL
Clouds: sky clear below 12,000 feet AGL
Weather: automated observation with no human augmentation;
there may or may not be significant weather present at this time

Format for the message is located at
http://weather.unisys.com/wxp/Appendices/Formats/METAR.html
as well as:
http://en.wikipedia.org/wiki/METAR

Data items available from the METAR message:

- Weather station ID (4 letters)
- Time/Date of observation (in GMT)
- Temperature
- Dewpoint
- Air pressure
- Type of observation (i.e. automatic)
- Surface wind direction, speed, and optional wind gust speed
- Visibility and possible reason for obscurity (fog, smoke, etc)
- At various altitudes:
    - Cloud conditions (clear, scattered, broken, overcast)
- Ceiling altitude

GFS MOS MAV Messages

GFS MOS is an interpretation of the Global Forecast System for Model Output Statistics and comes in short range (MAV) and extended range (MEX) forecasts. We will focus on MAV forecasts for this project. Forecasts are provided as a small ASCII file containing data in a tabular format.

Example of this format is:

```
KROS   GFS MOS GUIDANCE    2/12/2013  1200 UTC
DT /FEB  12/FEB  13                /FEB  14                /FEB  15
HR   18 21 00 03 06 09 12 15 18 21 00 03 06 09 12 15 18 21 00 06 12
N/X                    17          37          24          27     1
TMP  28 32 29 25 22 21 20 26 33 36 34 32 30 29 26 26 26 25 20  9  3
DPT  15 16 17 18 18 17 17 19 21 22 23 24 24 23 21 19 15 11  8  0 -4
CLD  FW CL SC BK OV OV OV OV OV OV OV OV OV OV OV OV OV OV OV OV OV
```

```
WDR  21 22 23 24 25 26 27 27 25 21 18 17 26 33 33 34 33 33 32 33 31
WSP  08 07 05 04 04 03 03 04 05 06 04 04 04 07 08 11 12 12 09 07 06
P06         0     5     0     3    14    47    24     9     6  0  1
P12                     5          20          49          12     3
Q06         0     0     0     0     0     1     0     0     0  0  0
Q12                     0           0           1           0     0
T06      0/ 0  0/ 0  0/ 1  0/ 0  1/ 3  1/ 0  0/ 0  0/ 1  0/ 0  0/ 1
T12            0/ 2        0/ 1        1/ 3        0/ 1     0/ 5
POZ   0  0  0  0  1  3  3  2  1  1  0  0  0  0  0  0  0  0  0  0  0
POS 100100100100 99 96 96 98 98 97100 97 99 95 99 98100100100100100
TYP   S  S  S  S  S  S  S  S  S  S  S  S  S  S  S  S  S  S  S  S  S
SNW                     0                       1                 0
CIG   8  8  8  8  8  8  7  7  7  7  6  4  4  3  4  5  5  7  6  7  7
VIS   7  7  7  7  7  7  7  7  7  7  7  5  4  5  7  7  7  7  7  7  7
OBV   N  N  N  N  N  N  N  N  N  N  N  N BR  N  N  N  N  N  N  N  N
```

Description of the MAV message can be found at
http://www.nws.noaa.gov/mdl/synop/mavcard.php

Data items available from the GFS MOS MAV message:

- Weather station ID (4 letters)
- Time/Date of forecast (in GMT)
- High and low temperature
- For each 4 hour period:
  - Temperature
  - Dewpoint
  - Cloud cover
  - Surface wind direction, speed and optional wind gust speed
  - Probability of precipitation for previous 6 hours
  - Probability of precipitation for previous 12 hours
  - Quantitative precipitation forecast (accumulation) for previous 6 hours
  - Quantitative precipitation forecast (accumulation) for previous 12 hours
  - Probability of thunderstorms given the previous 6 hours
  - Probability of thunderstorms given the previous 12 hours
  - Probability of freezing precipitation
  - Probability of snow
  - Precipitation type
  - Snowfall accumulation
  - Visibility and possible reason for obscurity (fog, haze, smoke, etc)
  - Ceiling altitude

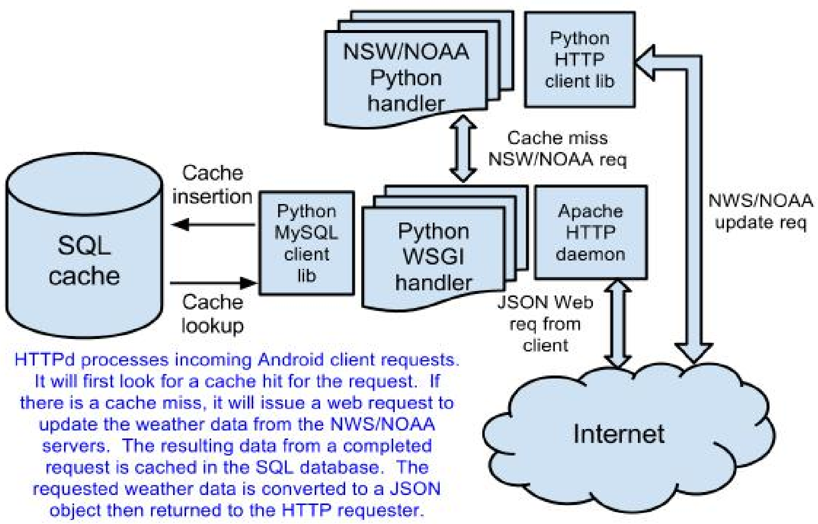
GFS MOV MAV files are located at:
http://www.nws.noaa.gov/mdl/synop/products.php

System Architecture

Server Components

The software stack used for the back end server will consist of the following components:

- Apache HTTP server handling HTTP API requests from Android clients.
- Python WSGI for processing API requests. Additional Python libraries will be used for JSON (JavaScript Object Notation) parsing, SQLAlchemy ORM database interface, and Urllib2 HTTP client communication to NWS/NOAA services.
- SQLite database to store and provide cached NWS/NOAA data.



[Figure 1 - Backend server architecture]

Server-side scripting will be written in Python. A brief overview of each module follows:

- **API Request Handler** - Interfaces with an Android client request. This module will process the incoming request, issue a cache lookup, request a NSW/NOAA update if needed, format the response in JSON and return the data to the client.
- **Cache Handler** - Process Lookup, Retrieve, and Store cache operations to the SQLite database. Purging of stale cache entries will also be done through this module.
- **NSW/NOAA Request Handler** - Issues Web requests to retrieve most recent weather updates. Data returned from this module is in "raw" format.
- **Data Parser** - Functional block to parse NSW/NOAA responses into a defined JSON format. Only JSON formatted data will be inserted into the cache and returned to the

client.

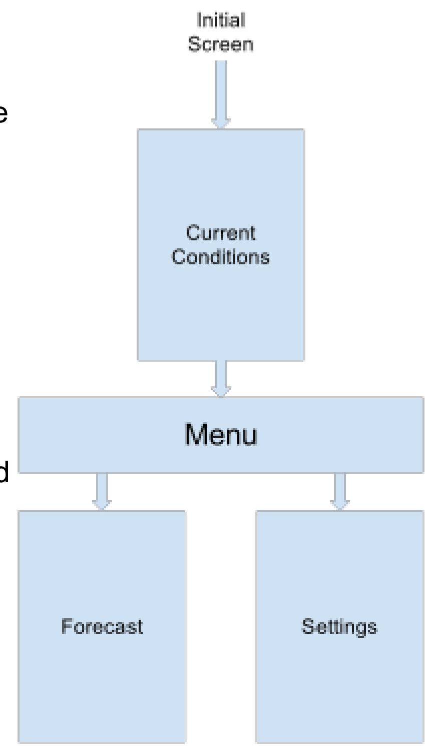
Mobile App Components

The mobile app will consist of several screens. The initial screen will bring up current weather data and selectors to switch to other screens. The other screens include forecast data and settings.

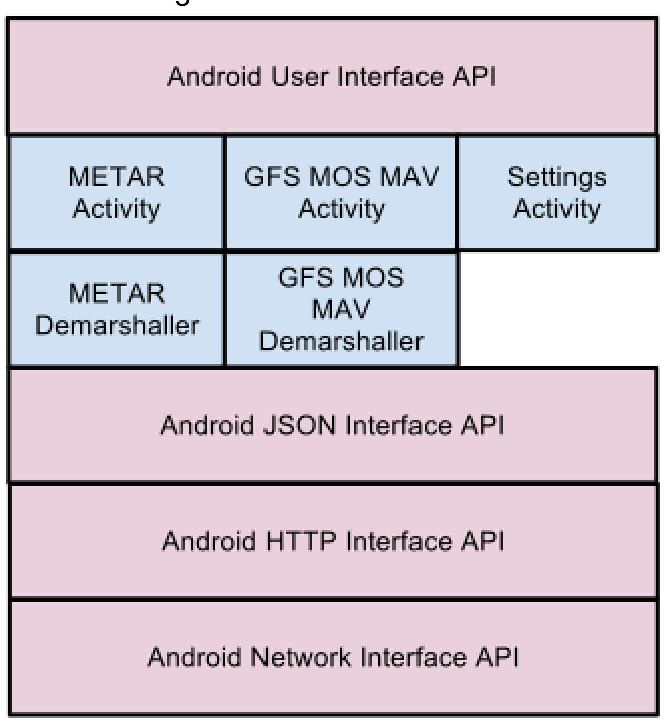
The mobile app will be written to support a minimum Android API Level 8 (Android 2.2/Froyo) and up to the current Android API Level 17 (Android 4.2/Jelly Bean). The following real devices are available to test on:

- HTC Inspire (480x800 display)
- ASUS Transformer (800x1280 display)

The mobile app will contain 3 activities (display pages). The initial activity displays the current conditions. This activity will have a menu that will allow the user to select either the forecast activity or the settings activity.
The mobile app exists within a layer between the Android UI API and the networking API.



Activity Hierarchy



Mobil App Architecture

Protocol API

The server and mobile app will communicate via JSON. This provides for a compact but also human readable format. Two JSON messages are defined, one for Metar data and one for GFS MOS MAV data. All fields are alphabetic strings. Field keys are listed followed by the description. Lists are denoted by [] in the field key. Subfields are denoted by indented list items

Client Request Message Format

A mobile client requesting a weather update will send a JSON request to the server with the following fields

- location - Weather station ID (location)
- time - Get weather data for this given time (for current or past data)
- source - Which weather source to query (Metar/GFS MOS MAV)

Server Response Message Format (Metar)

The following fields are in the Metar Message.

- station - Weather Station ID (4 letters)
- time - Time/Date of observation (in GMT)
- temperature - Temperature
- dew point - Dew point
- pressure - Air pressure/altimeter setting
- type - Type of observation (i.e. automatic)
- wind - Surface wind direction, speed, and optional gust speed
- visibility - Visibility distance and reason for obscurity (fog, smoke, etc)
- weather - weather description
- sky - sky conditions (cloud coverage, altitude, and type)
- remarks - various optional remarks

Server Response Message Format (GFS MOS MAV)

- wxid - Weather station ID (4 letters)
- time - Time/Date of forecast (in GMT)
- high - High temperature
- low - low temperature
- period[] - each 4 hour period
    - date - Date of period
    - hour - Beginning hour of period
    - temp - Temperature
    - dewpoint - Dew point
    - cover - Cloud cover
    - wind Surface Wind Direction, speed, and gust speed
    - pop6 - Probability of precipitation for previous 6 hours

    - pop12 - Probability of precipitation for previous 12 hours
    - qpf6 - Quantitative precipitation forecast (accumulation) for previous 6 hours
    - qpf12 - Quantitative precipitation forecast (accumulation) for previous 12 hours
    - thund6 - Probability of thunderstorms for previous 6 hours
    - thund12 - Probability of thunderstorms for previous 12 hours
    - popz - Probability of freezing precipitation
    - pops - Probability of snow
    - type - Precipitation type
    - snow - Snowfall accumulation
    - visibility - Visibility
    - obscurity - Possible reason for obscurity (fog, smoke, etc)
    - ceiling - Ceiling altitude

Database Schema

Since the database is used for caching requests, the database schema will simply cache JSON data as a string. Each cache entry is keyed by the Weather Station ID, Request Source, and a Timestamp.

Future Development

New features and enhancements can be done to the product in future development.

- Add winds aloft data
- Find the nearest weather station based on geolocation (GPS and/or network locator).

Credits

Several sources were involved in the creation of this system.

Sources for icon/logo mashup

http://commons.wikimedia.org/wiki/File:Umbrella_parapluie.PNG (licensed under Creative Commons Attribution 3.0 Unported).
http://source.android.com/ (licensed under Creative Commons Attribution 2.5.)

Android application

http://www.vogella.com/articles/AndroidJSON/article.html (no copyright listed)
Various examples from http://developer.android.com/training were used as the basis for some of the Activities used in this program. The field names have been changed to protect the innocent.

Open source software used by the back end server

CentOS: http://www.centos.org/
Python: http://python.org
Python-metar: https://pypi.python.org/pypi/metar/
Apache: http://www.apache.org/
mod_wsgi: https://code.google.com/p/modwsgi/
SQLAlchemy: http://www.sqlalchemy.org/
SQLite: http://www.sqlite.org/