CS378 - Mobile Computing

Location

(Location, Location, Location, Location)
Cheap GPS

http://xkcd.com/407/
Android and Location

• inputs to location for Android device include:
  • GPS
  • cell-ID (cell tower)
  • Wi-Fi networks
    – Network Location Provider combines cell-ID and Wi-Fi data
Location, Location, Location

- Dead reckoning
- Radar fix
- Visual fix
- Loran
- Omega
- Navsat
- GPS
- Active Sonar
- Inertial Navigation System
Location is Important

"On 8 January 2005 at 02:43 GMT, the *USS San Francisco* collided with an undersea mountain about 675 kilometers (364 nautical miles, 420 statute miles) southeast of *Guam* while operating at flank (maximum) speed at a depth of 525 feet."

- Wikipedia article on the USS San Francisco, SSN - 711

- Dead reckoning
- radar fix
- visual fix
- Loran
- Omega
- Navsat
- GPS
- Active Sonar
- Inertial Navigation System
USS San Francisco -
http://tinyurl.com/l6vuucm
Global Positioning System

• GPS
• US System that provides position, navigation, and timing
• Space Segment, Control Segment, User Segment
• US Air Force develops, maintains, and operates the space segment and control segment
GPS Space Segment

- 24 core satellites
- medium earth orbit, 20k km above the earth
- 6 orbital planes with 4 satellites each
- generally 4 satellites in line of sight at any spot on the earth
- recently upgraded to 27 sats
GPS Space Segment

- satellites circle the earth twice a day
- upgraded over time with different generations of satellites
- Current generation of satellites being developed by Lockheed - Martin (FOCS)
GPS Control Segment

- Ground facilities that monitor transmissions, perform analysis, and send commands and data to satellites
GPS User Segment

- Onboard clocks with accuracy of 1 nanosecond (1 billionth of a second)
- Satellites transmit one-way
- Receiver calculates position and course by comparing time signals from multiple satellites with the known position of those satellites
GPS User Segment

• accuracy normally within 5 - 10 meters
• precision requires accuracy of clocks and timing signal on the order of 20 nanoseconds
• the Special and General theories of Relativity must be taken into account to achieve the desired accuracy
• Special relativity predicts clocks on satellites go slower, on the order of 10 microseconds per day
• General relativity predicts the mass of the earth will have an effect
GPS Accuracy

Horizontal Position Error Histogram: 1 January - 31 March 2011

<= 95% Horizontal Error
(2.199 m)
GPS Accuracy

- Selective Availability: intentional degradation of signals for civilian use — ended in 2000

**SA Transition -- 2 May 2000**

![Graph showing GPS accuracy with horizontal and vertical error metrics](image-url)
GPS Accuracy

• civilian GPS: aka SPS
• military GPS: aka PPS
• military broadcasts on two frequencies, civilian only one
• "This means military users can perform ionospheric correction, a technique that reduces radio degradation caused by the Earth's atmosphere. With less degradation, PPS provides better accuracy than the basic SPS."
Android and Location

• Currently 3 methods of obtaining location
  • GPS
  • NETWORK
    – combines cell tower triangulation and wireless networks
  • PASSIVE
    – not a real provider, just piggy back off other applications
    – similar to software sensors
GPS

• most accurate but,
• only works OUTDOORS
• quickly consumes battery power
• delay in acquiring satellites or re-acquiring if lost
Network

• Combines cell tower triangulation and in range wireless networks

• If no cellular capability or plan on device (tablets?) then just wireless networks
Wireless Network and Location

• Use to use StreetView cars

• Now, use the devices themselves to map locations to wi-fi spots

• Apple and Microsoft do the same thing

• default on dev phones was checked
Google Location Services

Use wireless networks
See location in applications (such as Maps) using wireless networks

Location consent
Allow Google's location service to collect anonymous location data. Collection will occur even when no applications are running!

Agree  Disagree
Finding Location

- Add appropriate permission to `AndroidManifest.xml`
- Get instance of `LocationManager` using `getSystemService` method using `LOCATION_SERVICE`
- Choose location provider (from all providers or using `getBestProvider` method)
- Implement a `LocationListener` class
- Call `requestLocationUpdates` method with chosen provider so `LocationListener` start receiving location information
LocationManager
Quickly Finding Location

- If you want a simple fix (location) get the LocationManager and ask for the last, best known position

```java
mgr = (LocationManager) getSystemService(LOCATION_SERVICE);

log("\nLocations (starting with last known):");
Location location
    = mgr.getLastKnownLocation(LocationManager.NETWORK_PROVIDER);
    // Location location
    // = mgr.getLastKnownLocation(LocationManager.PASSIVE_PROVIDER);
    // Location location
    // = mgr.getLastKnownLocation(LocationManager.GPS_PROVIDER);
```
AndroidManifest.xml

• User Permission in manifest

```xml
<manifest ...
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
...
</manifest>
```

• Options: ACCESS_FINE_LOCATION or ACCESS_COARSE_LOCATION

• ACCESS_COARSE_LOCATION for use of NETWORK_PROVIDER using cell-ID and Wi-Fi

• ACCESS_FINE_LOCATION: GPS or NETWORK_PROVIDER
Uses Features

• In addition to request permissions the AndroidManifest.xml file can list features the app uses.

• Google Play uses these tags to filter applications for users

• examples of features: bluetooth, camera, location, network, microphone, nfc (near field communication), sensors, and more!

<uses-feature android:name="android.hardware.location.gps"/>
<uses-feature android:name="android.hardware.location.network"/>
Location Manager

• Obtain Location Manager

```java
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    mgr = (LocationManager) getSystemService(LOCATION_SERVICE);
```
Simple Location Program

• Just to demonstrate capabilities
• After setting up listener show all providers
• mgr is LocationManager

/** Write information from all location providers */
private void dumpProviders() {
    List<String> providers = mgr.getAllProviders();
    for (String provider : providers) {
        dumpProvider(provider);
    }
}
Properties of Location Providers

- name
- enabled
- accuracy
- power requirements
- monetary cost
- requires cell
- requires network
- requires satellite
- supports altitude
- supports bearing
- supports speed
Bearing

• direction
• 360 degrees
• degrees east of north
• 0 = north
• 90 = east
• 180 = south
• 270 = west
Program Output

• network (wifi and cell tower id)
• gps
• passive
  – use location updates requested by other applications or services
## dev Phones (no cell service)

<table>
<thead>
<tr>
<th>Name</th>
<th>Network</th>
<th>Passive</th>
<th>GPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>enabled</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>accuracy</td>
<td>coarse</td>
<td>invalid</td>
<td>fine</td>
</tr>
<tr>
<td>power req.</td>
<td>low</td>
<td>invalid</td>
<td>high</td>
</tr>
<tr>
<td>monetary cost</td>
<td>true??</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>request cell</td>
<td>true</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>requires network</td>
<td>true</td>
<td>false</td>
<td>true?</td>
</tr>
<tr>
<td>requires satellite</td>
<td>false</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>supports altitude</td>
<td>false</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>supports bearing</td>
<td>false</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>supports speed</td>
<td>false</td>
<td>false</td>
<td>true</td>
</tr>
</tbody>
</table>
LocationListener

- Implement class that implements `LocationListener` interface

<table>
<thead>
<tr>
<th>Public Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>abstract void</code> onLocationChanged (Location location)</td>
<td>Called when the location has changed.</td>
</tr>
<tr>
<td><code>abstract void</code> onProviderDisabled (String provider)</td>
<td>Called when the provider is disabled by the user.</td>
</tr>
<tr>
<td><code>abstract void</code> onProviderEnabled (String provider)</td>
<td>Called when the provider is enabled by the user.</td>
</tr>
<tr>
<td><code>abstract void</code> onStatusChanged (String provider, int status, Bundle extras)</td>
<td>Called when the provider status changes.</td>
</tr>
</tbody>
</table>
Obtaining Locations

• Register the LocationListener to receive location updates
• locationManager.requestLocationUpdates(LocationManager.NETWORK_PROVIDER, 15000, 10, locationListener);
  – provider: name of provider to register with
  – minTime: the minimum time interval for notifications, in milliseconds. only a hint to conserve power, and actual time between location updates may be greater or lesser than this value.
  – minDistance: min distance interval for notifications in meters
  – the listener itself
requestLocationUpdates

• More on arguments
• 0 for minTime AND minDistance indicate obtain updates as frequently as possible
• for *background services* recommended minTime \(>= 300,000\) ms to avoid consuming too much power with the GPS or Wi-Fi receivers
• 300,000 ms = 5 minutes
• clearly less for apps in the foreground
Location Listener

// Acquire a reference to the system Location Manager
LocationManager locationManager = (LocationManager) this.getSystemService(Context.LOCATION_SERVICE);

// Define a listener that responds to location updates
LocationListener locationListener = new LocationListener() {
    public void onLocationChanged(Location location) {
        // Called when a new location is found by the network location provider.
        makeUseOfNewLocation(location);
    }

    public void onStatusChanged(String provider, int status, Bundle extras) {
    }

    public void onProviderEnabled(String provider) {
    }

    public void onProviderDisabled(String provider) {
    }
};

// Register the listener with the Location Manager to receive location updates
locationManager.requestLocationUpdates(LocationManager.NETWORK_PROVIDER, 0, 0, locationListener);
Location Data

- `onLocationChange` method in the `LocationListener` receives `Location` objects
- `toString` shown
- latitude, longitude, timestamp, possibly altitude, speed, and bearing

```
Locations (starting with last known):
Location[mProvider=network,
  mTime=1333326955116,mLatitude=30.5158771,
  mLongitude=-97.7139099,mHasAltitude=false,
  mAltitude=0.0,mHasSpeed=false,
  mSpeed=0.0,mHasBearing=false,
  mBearing=0.0,mHasAccuracy=true,
  mAccuracy=47.0,mExtras=Bundle[mParcelledData.
  dataSize=148]]
Location[mProvider=network,
  mTime=1333327587135,mLatitude=30.5159603,
  mLongitude=-97.7133946,mHasAltitude=false,
  mAltitude=0.0,mHasSpeed=false,
  mSpeed=0.0,mHasBearing=false,
  mBearing=0.0,mHasAccuracy=true,
  mAccuracy=35.0,mExtras=Bundle[mParcelledData.
  dataSize=148]]
```
Sample GPS Locations

- units for altitude and accuracy: meters
- units for speed: meters / sec
  - 1 mps $\approx$ 2.2 mph
- units for time: UTC (coordinated universal time) time of this fix, in milliseconds since January 1, 1970.
- Java Date class has constructor that accepts UTC
Sample GPS Locations

Location[mProvider=gps,
mTime=1333414860000,mLatitude=30.51601886364164,mLongitude=-97.71347013064596,mHasAltitude=true,mAltitude=210.0,mHasSpeed=true,
mSpeed=0.29998752,mHasBearing=true,
mBearing=315.55438,mHasAccuracy=true,
mAccuracy=20.0,mExtras=Bundle[mParcelledData.
dataSize=4]]
new toString for Location

onLocationChanged CALLED:

Location[gps
30.286450,-97.736539 acc=50
et=+1h8m52s912ms
alt=175.70001220703125
vel=2.4394498 bear=110.0
{Bundle[mParcelledData.dataSize
=44]}]
Location Strategies

• Location aware applications
  – compelling? better information to user?

• GPS -> slow, only works outdoors, consumes lots of power, very accurate

• Network -> fast, works indoor and outdoor, uses less power, less accurate

• Issues: multiple sources (cell id with call plan, wifi, gps), user movement, accuracy of locations
Getting a Fix

• Some applications (driving directions, sport tracking) require constant location data
  – using battery is expected
Periodic Location Updates

• Many location aware applications do not need a constant stream of location updates

• Obtaining location pattern:
  1. Start application.
  2. Sometime later, start listening for updates from desired location providers.
  3. Maintain a "current best estimate" of location by filtering out new, but less accurate fixes.
  4. Stop listening for location updates.
  5. Take advantage of the last best location estimate.

Getting Location

• Timeline for getting location based on pattern described:

![Timeline Diagram]

Figure 1. A timeline representing the window in which an application listens for location updates.
Last Known Location

• Recall, application is part of a larger system
• other applications may have asked for location and we can use those locations via the LocationManager

```
LocationProvider locationProvider = LocationManager.NETWORK_PROVIDER;
// Or use LocationManager.GPS_PROVIDER

Location lastKnownLocation = locationManager.getLastKnownLocation(locationProvider);
```
Current Best Estimate

• The most recent location, may not be the most accurate

• Evaluating a location
  – how long has it been since the current best estimate?
  – is the accuracy of the new location update better than the best estimate?
  – what is the source of the location? which do you trust more?
LocationManager - Useful Methods

• addProximityAlert(double latitude, double longitude, float radius, long expiration, PendingIntent intent)
  – Sets a proximity alert for the location given by the position (latitude, longitude) and the given radius.

• List<String> getAllProviders()
  – Returns a list of the names of all known location providers.

• Location getLastKnownLocation(String provider)
  – Returns a Location indicating the data from the last known location fix obtained from the given provider.

• Location class: float distanceTo(Location dest)
  – Returns the approximate distance in meters between this location and the given location.
GEOCODING
GeoCoding

- LocationManager and GooglePlayServices provide a Location object
- Contains latitude and longitude
- Latitude, degrees north of south of equator North Pole = 90N or +90, equator = 0, South Pole = 90S or -90
- longitude, degrees east or west of prime meridian, 0 to -180 to west, 0 to +180 to west
GeoCoding

• Various databases exist to give address(es) at a given lat and long
• Web Service?
• GeoCoder class in Android provides:
  – geocoding: convert physical address to lat / long
  – reverse geocoding: convert lat / long to physical address
Geocoding

- Accessing Google database of addresses
- Making network calls may block the Activity
- Key to not "BLOCK THE UI THREAD"
- Meaning, don't do expensive computations or possibly slow operations in the code for an Activity
Geocoding

• When address button clicked use a Geocode to try and get address

```java
// address Button clicked, show address
public void showAddress(View v) {
    tryCount = 0;
    getAddress();
}

private void getAddress() {
    AsyncTask<Geocoder, Void, List<Address>>
    addressFetcher = new AddFetch();
    Geocoder gc = new Geocoder(this, Locale.US);
    addressFetcher.execute(gc);
}
Reverse Geocoding

• Get lat and long from last known location of LocationTest app

```java
private void tryReverseGeocoding(Geocoder gc) {
    if (lastKnownLocation != null) {
        double lat = lastKnownLocation.getLatitude();
        double lng = lastKnownLocation.getLongitude();
        Log.d(TAG, "REVERSE GEO CODE TEST lat: " + lat);
        Log.d(TAG, "REVERSE GEO CODE TEST long: " + lng);
        addresses = null;
        try {
            addresses = gc.getFromLocation(lat, lng, 10); // maxR
        } catch (IOException e) {
        }
    } else {
        output.append("No locations yet. Please try later.\n\n");
    }
}
```
If Successful

protected void onPostExecute(List<Address> result) {
    if (result == null) {
        tryAgain();
        Log.d(TAG, "No addresses from Geocoder. Trying again. " +
              "Try count: " + tryCount);
    } else {
        output.append("Number of addresses " +
                      "at current location :" + addresses.size());
        output.append("BEST ADDRESS FOR CURRENT LOCATION:
                      \n" + addresses.get(0).toString());
        Log.d(TAG, "reverse geocoding, " +
              "addresses from lat and long: " +
              addresses + " " + addresses.size());
    }
}

• Geocoder returns a list of possible addresses
Demo

onLocationChanged
CALLED:

Location[gps
30.286679,-97.736755
acc=15
let=+8d18h59m26s963ms
salt=153.4000244140625
del=0.56295 bear=345.0
{Bundle[mParcelledData.dataSize=432]}

BEST ADDRESS FOR CURRENT
LOCATION:Address[addressLines=[0:"2317
Speedway",1:"Austin, TX
78712",2:"USA"],feature=2
317,admin=Texas,sub-
admin=null,locality=Austi
n,thoroughfare=Speedway,
postalCode=78712,countryCode=US,countryName=
=United
LOCATION SERVICES VIA GOOGLE PLAY SERVICES
Alternative?

• Google Location Services API
  – "part of Google Play Services, provides a more powerful, high-level framework that automatically handles location providers, user movement, and location accuracy. It also handles location update scheduling based on power consumption parameters you provide. In most cases, you'll get better battery performance, as well as more appropriate accuracy, by using the Location Services API."
Google Play Services
Google Play Services

- Extra APIs provided by Google
- Allows your app to access various Google services such as:
  - maps
  - Google+
  - game services
  - location APIs
  - ads
  - Oauth
Using Google Play Services

- Download Google Play Services for Android SDK via SDK Manager
Setting up Google Play Services

• In target app must add dependency in the Gradle build file for the app / module - not the project app

```groovy
apply plugin: 'com.android.application'

...  

dependencies {
    compile 'com.google.android.gms:play-services:8.4.0'
}
```
Getting Location

• Must still request permission for location access in manifest file
• Create GoogleApiClient in onCreate method of Activity that wants location

// Create an instance of GoogleApiClient.
if (mGoogleApiClient == null) {
    mGoogleApiClient = new GoogleApiClient.Builder(this)
        .addConnectionCallbacks(this)
        .addOnConnectionFailedListener(this)
        .addApi(LocationServices.API)
        .build();
}
Getting Location

• Connect and disconnect client in `onStart` and `onStop` methods of Activity

```java
protected void onStart() {
    mGoogleApiClient.connect();
    super.onStart();
}

protected void onStop() {
    mGoogleApiClient.disconnect();
    super.onStop();
}
```
Getting Location

• Request last known location from Google Play Services

• Accuracy based on choice in manifest of COARSE or FINE location permission

```java
public class MainActivity extends ActionBarActivity implements
    ConnectionCallbacks, OnConnectionFailedListener {
    ...
    @Override
    public void connected(Bundle connectionHint) {
        mLastLocation = LocationServices.FusedLocationApi getLastLocation(
            mGoogleApiClient);
        if (mLastLocation != null) {
            mLatitudeText.setText(String.valueOf(mLastLocation.getLatitude()));
            mLongitudeText.setText(String.valueOf(mLastLocation.getLongitude()));
        }
    }
}
Geofencing

• Location Services also supports creating and monitoring geofences

• App can create a geo fence
  – latitude and longitude
  – radius

• Once a geofence is created app can ask Location Services for notifications
  – when device enters geofence
  – when device leaves geofence
  – when device has been present (dwells) in geofence for a specified amount of time
Geofencing