Building Mobile Cyber-Physical Systems with Smartphones and Cloud Computing

Jules White

Cyber-Physical Systems & Quality of Service

- A Cyber-Physical System (CPS) involves close coordination between the system's computational & physical elements
- CPS applications are essential in mission- & safety-critical domains, such as aerospace, automotive, chemical processes, civil infrastructure, energy, entertainment, healthcare, manufacturing, & transportation
- CPS applications must adhere to complex sets of Quality of Service (QoS) constraints, such as real-time scheduling, to ensure safety & efficiency
Physical World Interaction Requires Strict QoS

- Since a CPS interacts with the physical world, it's not enough for it compute the right answer, it must also compute:
  - Fast enough to keep up with real-world processes
  - Efficiently enough to run for long periods of time without wasting resources
  - Safely to prevent damage to the physical world

- The complexity of interacting with the physical world with the correct Quality of Service (QoS), makes CPS design & optimization hard

Software developer uses slower algorithm for higher accuracy & improved safety but needs faster processor.

Design changed to use smaller power supply for weight reduction.
Physical World Interaction Requires Strict QoS

- Since a CPS interacts with the physical world, it’s not enough for it to compute the right answer, it must also compute:
  - Fast enough to keep up with real-world processes
  - Efficiently enough to run for long periods of time without wasting resources
  - Safely to prevent damage to the physical world
- The complexity of interacting with the physical world with the correct Quality of Service (QoS), makes CPS design & optimization hard

CPS designers need models, methods, & algorithms to help them design systems that meet the QoS demands of interacting with the physical world
Specialized, Proprietary Hardware

• Traditionally, customized hardware/software solutions have been used to deal with these complex QoS demands

• However, these specialized approaches:
  • Make hardware/software more brittle and difficult to maintain
  • Increase the overall cost substantially
  • Are hard to develop

Your Smartphone (e.g. iPhone or Android)

- Internet Connection
- Sensors: GPS, Camera, Accelerometer, etc.
- Your Social Network
Can we build mobile CPS applications on top of this type of commodity software/hardware?

Mobile Cyber-physical Systems

I want your phones to:
1. Know what you are doing
2. Where/when you are doing it

Then, have systems that:
1. Aggregate the information
2. Make decisions based on the information
3. Execute actions that affect the physical world

Apple, Google and several other developers of mobile operating systems were sent letters by a House committee that oversee privacy issues, seeking more information about whether they are tracking users’ locations. Yukari Kane reports.
I want everyone’s phones to:
1. Know what you are doing
2. Where/when you are doing it

Then, have systems that:
1. Aggregate the information
2. Make decisions based on the information
3. Execute actions that affect the physical world

Eventual Goal is Mind Control & Arrest for Future Crimes

Apple, Google and several other developers of mobile operating systems were sent letters by a House committee that oversees privacy issues, seeking more information about whether they are tracking users’ locations. Yukari Kane reports.
Traffic Accidents

Car accidents are one of the leading causes of death in the US, causing over 100 fatalities daily.

1. In 2007, there were 10.6 million traffic accidents
2. Reducing accident response time by one minute saves 6% more lives
3. Automated crash notification systems eliminate time between accident and first responder dispatch

Example: WreckWatch 

Detecting Traffic Accidents with Smartphones

WreckWatch uses your phone to detect when you have been in a traffic accident.

1. Automatically contacts first responders
2. Maps accidents
3. Allows bystanders to provide imagery to first responders
4. Automatically notifies emergency contacts
During a disaster, there may be little information about certain affected areas.

1. Often, “citizen scientists” can provide valuable reporting (e.g., iNews, tornado spotters, etc.)
2. During the Nashville flood in 2010, best imagery came from Facebook/Twitter
3. Distributed relief effort with individuals using personal boats, etc. to help out

Example: CLEAR
Tracking Environmental Disasters with Smartphones

Cloud Environmental Analysis and Relief (CLEAR)
1. Allows scientists and first responders to push data collection tasks to citizen scientists
2. Automatically reports data back from citizen scientists and aggregates in the cloud
3. Allows researchers to track environmental impact
Data Collection with Phones

1. Have “citizen scientists” or researchers collect data with their phones
   - Annotated Images
   - GPS
   - Forms
   - Etc.

2. Automatically upload and aggregate data from hundreds or thousands of phones

3. Share data with others

Additional Image:
http://www.complexification.net/gallery/machines/tree
Data Collection with Phones

3. Allow researchers to analyze the data

Other Examples

1. “Street Bump” uses drivers’ smartphones to detect potholes and send GPS-tagged reports to Boston govt.
2. “Visibility” app uses pictures from cell phones to estimate air quality
3. Distributed cell network mapping using smartphones
1. Maintenance / charging by owners
2. Centralized / standardized software distribution mechanisms
3. Low cost
4. Persistent network connections
5. Variety of sensors
6. Naturally mobile and able to measure human phenomena

Why Build CPS Applications on Smartphones

There are Far More Mobile Phone Users than PC Users

![Graph showing Internet and Mobile Phone Users, Worldwide](Image)
Those Mobile Phones will Soon be Smartphones (or tablets)

Smartphone Sales To Beat PC Sales By 2011

Source: RBC Capital Markets estimates

Smartphones are the Future of Gaming

Major Game Systems vs. Mobiles
Challenges…

Although smartphones have a variety of sensors, detecting high-level contexts is hard:

1. Sensor precision / jitter is not ideal
2. No high-level sensor for “walking” or “eating”
3. Sensors may be obscured or affected (e.g. camera) b/c of how device is carried

Challenge 1: Context Identification with Sensors
Challenge 1: Context Identification with Sensors

Example: Compass Noise

Compass data reported while device is stationary

Challenge 2: Power

Although Moore’s law applies to processors, it does not apply to batteries

1. Batteries technology is developing at a much slower rate
2. Software engineers aren’t used to equating software design decisions to power
Battery Energy Density vs. Other Computing Advances

Challenge 2: Power
Example: Power Consumption on Android

Source: Values measured using an industrial power meter at 1Hz sampling rate, and taking average power with lowest standard deviation.
Software engineer’s choice of data format (or maybe adherence to a standard) impacts battery life:

Software engineer’s choice of whether or not to compress data has a big impact depending on how data is sent/recvd:
Challenge 2: Power
Example: Power vs. Sensor Accuracy

- Accelerometer/magnetic sensors
  - Normal: 10mA (used for orientation detection)
  - UI: 15mA (about 1 per second)
  - Game: 80mA
  - Fastest: 90mA

Challenge 3: Data Aggregation & Processing

If you are receiving and processing data from hundreds of thousands of devices on a server, how do you:

1. Scale effectively while maintaining QoS
2. Run power efficiently when load is related to physical phenomena (e.g. users go to sleep)
3. Store massive amounts of data
Questions?

Platform Selection...
There are 2 key platforms that in smartphone dev:

1. Android
2. iOS (iPhone, iPod, iPad)

There are certainly other smartphone platforms, but we do not work with them extensively in my research group (e.g. Blackberry, Windows Phone 7, etc.)

iOS is the basis of all of Apple’s mobile products

1. Applications are written in Objective-C
2. Excellent developer tools in Xcode
3. Requires a $99 yearly developer license to test on an actual device
4. Mature APIs
5. Easy to make beautiful looking applications
Android

Android is an open-source mobile platform created by Google

1. Applications are written in Java
2. Good (not great) developer tools in Eclipse
3. No developer license required
4. Mature APIs
5. Easy to hack for research purposes

Android vs. iOS
Programming Language

A major difference between the platforms is the programming language

1. Java is much more widely known – may be easier for education purposes
2. Java uses more familiar C-style syntax
3. Objective-C has more dynamic/reflective features – can send a message to nil!
4. Objective-C allows you to memory manage (feature or challenge..?)
5. Java is often used on the server-side as well…possibly making dev more cohesive
6. Java "may" have more open-source libs
iOS uses Xcode for development while Android uses Eclipse

1. Eclipse + Java provide incremental compilation and immediate flagging of many errors
2. Xcode’s GUI builder is orders of magnitude better than Eclipse’s
3. Xcode / iOS mechanism for connecting GUI elements to code is MUCH better
4. Code signing / setup is much easier in Eclipse
5. Xcode appears more stable / faster than Eclipse

Android vs. iOS
Development Tools

Android uses a VM to emulate Android and iOS uses libraries compiled against OS X

1. Eclipse + Android Emulator = SLOW
2. iOS emulator is faster but can make apps appear too fast…developer may not realize something isn’t going to work on a real device
3. Android Emulator has third-party plugins to allow for more control over sensor inputs (e.g. accelerometer)
4. iOS emulator is faster
5. Android emulator is opensource
Android vs. iOS

**APIs**

Android and Xcode have roughly the same set of features:

1. Android APIs appear to be developing more rapidly than iOS
2. Xcode GUI APIs are much better than Android’s
3. Android Intents, Activities, ContentProviders, and Services make code reuse easier than on Xcode

**As a Research Platform**

As a research platform, Android definitely has the edge in multiple areas:

1. The Android security model is “tell the user about it and it is OK” – makes doing unusual things easier
2. Intents/Services make developing sensing / background apps easier
3. Android does not require a developer license
4. Android is opensource and based on Linux
5. Easy to find grad students with Java skills
Questions?

A Look at Android...
Why Android?

- Android has 28% of the smartphone market (#2)
- iPhone has 21% of the smartphone market (#3)
- Blackberry is still #1 (not sure if this is still true)...but the platform is currently not nearly as interesting to develop for as Android/iPhone
- The Blackberry platform is not nearly as well developed as Android/iPhone

<table>
<thead>
<tr>
<th>Top Smartphone Platforms</th>
<th>Share (%) of Smartphone Subscribers</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Feb-13</td>
</tr>
<tr>
<td>Total Smartphone Subscribers</td>
<td>100.0%</td>
</tr>
<tr>
<td>RIM</td>
<td>42.1%</td>
</tr>
<tr>
<td>Apple*</td>
<td>25.4%</td>
</tr>
<tr>
<td>Microsoft</td>
<td>15.1%</td>
</tr>
<tr>
<td>Google</td>
<td>9.0%</td>
</tr>
<tr>
<td>Palm</td>
<td>5.4%</td>
</tr>
</tbody>
</table>

Why Android?

- Android is:
  - one of the fastest growing smartphone platforms
  - opensource and works on multiple platforms
    - no need to own a Mac
    - no need to join a developer program
  - easy to learn for Java (and C++) programmers
  - much easier to transition to than Objective-C
  - very hackable from a research perspective
Developing Android Apps

- Android uses the Eclipse Integrated Development Environment
- The Android Eclipse Plugins provide:
  - Wizards for creating new apps
  - A visual editor for creating GUIs
  - Special editors for manipulating the Android XML descriptors needed for your app
  - An emulator for testing your apps on your PC
  - A debugger for finding errors in the emulator or on a device

Setting Up an Android Dev Environment

- You need to download and install “Eclipse Classic” from: http://www.eclipse.org
- You will also need to download and install the Android SDK from: http://developer.android.com/sdk/index.html
- Once Eclipse and the Android SDK are installed, follow the “Installing the ADT Plugin for Eclipse” instructions here: http://developer.android.com/sdk/installing.html
Figuring Out Android

- Android is well documented
- The Android javadoc references will be critical reference material for your projects:
- The developer guide is another important resource:
- Stack Overflow is a great Android resource

What is Android?

- Android is a complete software stack for mobile phones (and more)
- Android includes:
  - Operation System
    - Linux Variant
  - Specialized Java Virtual Machine
    - Dalvik, optimized for power consumption
  - Middleware Stack for:
    - Telephony
    - GUIs
    - Apps
    - App Distribution
Your First App

• Create a new Android project from the file menu:
  • File->New->Android Project

Your App’s Settings

• The new project wizard is where you setup critical information for your app:
  • You need to set the version of the Android platform that you will be targeting
  • You will also specify the name of your application
  • You will provide a Java package that will hold all of your app’s code
  • You will specify a default Activity for your project
Android Project Anatomy

- The core of an Android project is the code that you provide for your app.
- This code will live inside the “src” folder in the package that you specified in the new project wizard.

Android Project Anatomy

- Android generates a number of files for you that make it easier to build GUIs and fetch resources.
- The “gen” folder contains the generated code produced by the Android plugin. You should never put any code in this folder or modify any of the generated code.
- The “R.java” file is used to fetch GUI resources and widgets (we will discuss it later).
Android Project Anatomy

- The “res” folder contains non-code resources, such as images, that are used by your app.
- You include images for your app in the various “drawable-*” folders.
- You can define your GUI using XML or the Android layout editor in the various XML files beneath the layout folder.
- The values/strings.xml file contains...

Android Project Anatomy

- The AndroidManifest.xml file contains critical information needed by the Android Runtime to execute your app.
Testing the App in an Emulator

• The Android SDK includes an emulator for testing and debugging apps
• To run the emulator, you first need to create an Android Virtual Device (AVD)
• An AVD specifies the configuration of a test phone configuration:
  • Screen size
  • Android platform version
  • SD card information
• The AVD manager lists the available AVDs

Creating an AVD

• To create your first AVD, click the “new” button on the AVD manager
• The AVD creator allows you to specify the properties of the virtual device:
  • You must provide a name for the AVD
  • You must provide the target version of Android that you want on the emulator
  • You should provide a size for the SD card on the emulator
  • You can also specify other custom capabilities or limitations in the Hardware editor (ignore for now)
Running the Hokie Football App

• The Eclipse “Run Configurations” menu allows you to launch Java applications and Android apps
• Select “Android Application” and then click the “new launch configuration” button in the upper left

Running the HokieFootball App

• Name your configuration, select your app’s project, and set the target AVD
• Then, click “run”
Running the Hokie Football App

- When the emulator finishes booting, your app should immediately launch
- Booting the emulator takes forever….
- You do not need to reboot the emulator to test changes to your app
  - Right-click on your app’s project
  - Select run as->android application
  - This will update the app on the emulator without rebooting it

Android Component Model

- Android uses a component model based on Activities, Services, and ContentProviders
- Each application runs in a separate process and is composed of one or more of these components
- These components can be reused outside of the application that provides them
- This reuse approach allows developers to quickly build apps from pieces of other apps
Android Activities as Components

- An Activity is a component that can be used to accomplish a specific task
  - Sending an email
  - Sending an SMS
  - Viewing a list of email

- Your application can rely on both your own Activities and Activities exposed by other applications
- Activities are reusable ways of accomplishing a task
  - Incorporate an Activity for sending Email into your app

Screens are Managed by Activities

- Dialer
- Contacts
- View Contact
- New Contact
An Activity has several important methods that are called by the Android runtime to control its life-cycle:

- **onCreate()** – this method is called when the Activity is first created. You will almost always override this method and provide setup code in this method.
- **onStop()** – this method is called when the user leaves your Activity for another Activity (your Activity is not visible).
- **onPause()** – the user leaves your Activity but it is still visible in the background (e.g. transparent or partial foreground coverage).

- **onResume()** – this method is called when the user returns to your Activity from another Activity.
- **onStart()** – this method is called after your Activity is created or stopped.
- **onDestroy()** – the Activity is being released and needs to clean up all resources.
When the user clicks on your app’s icon, your main Activity is launched
onCreate() is called

When the main Activity for your app shows on screen the
onStart() method is called
Clicking on the picture gallery button will open a new Activity to display the gallery.

When the picture gallery Activity is launched, its onCreate and onStart methods are called.

The prior Activity’s onPause() and onStop() methods are called.
When the user completely exits the app, the original default Activity’s `onDestroy()` method is called.

Activities are Event-driven

- Most Activities are event-driven
- Your primary work is to create one or more Activities
- Each Activity sets up a GUI in its `onCreate()` method
- Event listeners are added to the GUI elements
- Sensor listeners are registered in `onCreate()`
- Subsequent code in your Activity is triggered by your event listeners
- Lifecycle methods register/unregister event listeners
Simple Example

```java
public class AnotherActivity extends Activity implements OnClickListener{
    public void onCreate(){
        super.onCreate();

        //Setting up the GUI
        setContentView(R.layout.main);

        Button b = (Button)findViewById(R.id.someButton);
        b.setOnClickListener(this);
    }

    public void onClick(ClickEvent evt){
        //called later
    }
}
```

Activity Stack

- Apps are composed of multiple Activities
- As Activities are launched, Android tracks them as a stack
- Launching a new Activity pushes it onto the top of the stack
- Pressing the back button pops the current Activity off the top of the stack and brings the new top Activity to the foreground
- Android may kill off your Activity while it is not at the top of the stack
Being Prepared to Save State

Android Intents

• An application can leverage existing Activities by sending Intents
• An Intent is a message object that is sent to the Android platform to tell it that you want to complete a specific action
• For example, you can send an Intent to Android to tell it that you want to send an email
• When you send Android an Intent, it figures out which Activity needs to be run in order to complete the action described by the Intent
Implicit vs. Explicit Intents

- Android supports two different kinds of Intents:

  1. **Explicit Intents** specifically state which Activity you want to invoke

  2. **Implicit Intents** are broadcasts that define a type of action that needs to be completed but does not specify which Activity should respond to the Intent

- Intents can include data that the chosen Activity should use to perform the action
- After an Activity sends an Intent, it can receive a result after the invoked Activity is finished

### Implicit Intent Actions:

- ACTION_MAIN
- ACTION_VIEW
- ACTION_ATTACH_DATA
- ACTION_EDIT
- ACTION_PICK
- ACTION_CROOSER
- ACTION_GET_CONTENT
- ACTION_DIAL
- ACTION_CALL
- ACTION_SEND
- ACTION_SENDTO
- ACTION_ANSWER
- ACTION_INSERT
- ACTION_DELETE
- ACTION_RUN
- ACTION_SYNC
- ACTION_PICK_ACTIVITY
- ACTION_SEARCH
- ACTION_WEB_SEARCH
- ACTION_FACTORY_TEST

Sending an Explicit Intent

- An Explicit Intent specifies the Java class of the Activity you would like to start
- Explicit Intents are normally used to transition between screens in Android
- E.g. You select a contact from a list in Android and it fires an Intent to open a new Activity to display the contact details

```java
public class MyActivity extends Activity{
    public void openAnotherActivity(){
        Intent intent = new Intent(this, AnotherActivity.class);
        startActivity(intent);
    }
}

public class AnotherActivity extends Activity {
    public void onCreate(){
        super.onCreate();
        ...
    }
}
```
Sending Data with an Explicit Intent

- Because you do not explicitly construct the new Activity, there is no way to call any methods on it to pass data.
- In order to pass data to the new Activity, you must attach it to the Intent with the putExtra() method.
- The receiving Activity can get the calling Intent by calling getIntent().
- The receiving Activity can then read the passed data from the Intent.

```java
class MyActivity extends Activity {
    public void openAnotherActivity() {
        Intent intent = new Intent(this, AnotherActivity.class);
        intent.putExtra("some data", "Hello World");
        startActivity(intent);
    }
}
class AnotherActivity extends Activity {
    public void onCreate() {
        super.onCreate();
        Intent callingIntent = getIntent();
        String data = callingIntent.getStringExtra("some data");
    }
}
```

Sending an Implicit Intent

- An Implicit Intent specifies and action that should be completed but not which Activity should be responsible for completing the action.
- The intent can also include a URI that specifies the location of the data that you would like the action taken on.

```java
class MyActivity extends Activity {
    public void openAnotherActivity() {
        Intent intent = new Intent(Intent.ACTION_VIEW);
        Uri uri = Uri.parse("sms:5555555555");
        intent.setData(uri);
        intent.putExtra("sms_body", "Woohooo");
        intent.setType("vnd.android-dir/mms-sms");
        startActivity(intent);
    }
}
```
Receiving Implicit Intent Broadcasts

- Android sends out a number of Implicit Intent broadcasts that your Activities can register to receive
- An Activity can be registered to receive broadcasts by defining an Intent Filter in the manifest file that lists the types of intents it would like to receive
- An Activity can also programmatically register broadcast receivers

Activities can have filters that differentiate based on:
- Type of Data (e.g. photo, video, contact, etc.)
- Type of Action being requested on the data
- Category of component required to handle the intent
Intents as Notifications of State Change

Many system state changes are broadcast as Intents

<table>
<thead>
<tr>
<th>Intent Name</th>
<th>Broadcast Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACTION_BATTERY_LOW</td>
<td>broadcast receiver A warning that the battery is low.</td>
</tr>
<tr>
<td>ACTION_HEADSET_PLUG</td>
<td>broadcast receiver A headset has been plugged into the device, or unplugged from it.</td>
</tr>
<tr>
<td>ACTION_SCREEN_ON</td>
<td>broadcast receiver The screen has been turned on.</td>
</tr>
<tr>
<td>ACTION_TIMEZONE_CHANGED</td>
<td>broadcast receiver The setting for the time zone has changed.</td>
</tr>
</tbody>
</table>

Receiving the Result of an Activity

- An Activity can be notified when an Intent it broadcasts is finished being serviced
- The Intent that is passed into onActivityResult() will have the data that was returned by the called Intent
- The protocol for accessing that data is dependent on the type of action

```java
public class MyActivity extends Activity {
    ...

    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        if (resultCode == REQUEST_TYPE_1) {
        }
        else if (resultCode == REQUEST_TYPE_2) {
        }
    }
```
Process Model

- Android runs each app in separate process
- Core system functionality, such as GPS, runs as a service in a system process
- Cannot directly access certain features from app process

Inter-process Communication

- Communication between processes is a key concept in Android
- Android has a custom IPC mechanism, called the Binder
- The Binder is implemented as a kernel driver, C++ lib, and Java lib
Object-oriented IPC

- The Binder provides an Object-based interface to IPC
- Objects can be sent to a foreign process as proxies
- Method calls on proxies are translated into binder calls to native process of an object
- Binder automatically manages threading

Binder and Event Listeners

- Nearly all outside events for your application are delivered through the Binder
- Sensor events are delivered through the Binder
- Touchscreen events are delivered through the Binder
- Intents are delivered through the Binder
Android Security

• Android has a security model that is designed to protect users from malevolent apps
• All Android apps are required to ask permission to use any platform facility that could adversely impact the user or platform
• Examples:
  • Internet
  • Camera
  • Contacts
• However, if you are given permission to use a resource, you can use it however you want!

Binder in Security

• Many parts of Android security rely on services provided by the Binder
• All remote calls include the process ID and user ID of the process sending the IPC call
• Binder proxy objects can be used as tokens to prove identity
• Throughout Android, Binder tokens, process IDs, and user IDs are utilized for identity checking and permission determination
• Is the Binder a potential angle of attack against Android?
Android GUI Layouts

- Each Activity is responsible for a separate GUI screen
- Android allows GUIs to be built either programmatically or through Layout XML files
- Individual GUI elements in Android are instances of the View class
- Layout XML files define a hierarchy of Views that can be parsed by the system and loaded for your Activity at runtime
- Multiple versions of Layout XML files can be provided to adapt the GUI for different versions of Android, screen sizes, etc.

Connecting Java to Layout XML

- GUI elements defined in XML are automatically built by the platform at runtime
- In order for your application to programmatically manipulate them (e.g. add event listeners), you must obtain handles to these elements
- Activities can lookup specific GUI element instances in their View hierarchy using the findViewById() method
Extracting Views from a Hierarchy

- Every Activity has a findViewById(…) method
- The method takes an id as an argument
- You can refer to the named ids that you provided in the layout editor through the R class
- Notice how we extract the Button using the R.id.GetFootballScheduleButton variable

```java
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    ScrollView sp = (ScrollView)findViewById(R.idScrollView01);
    Button b = (Button)findViewById(R.id.GetFootballScheduleButton);
}
```

Android Fragmentation

- Android does not provide a single cohesive target like iOS
- Devices can vary widely in screen size, capabilities, and Android implementation
- Typically want to use standard automatic layouts to manage GUI rather than absolute pixel perfect layouts
- Android provides tools like “display independent pixels” to help
- Possible differences in Intents based on platform implementation
  - Intent to take a picture does different things on different phones
Android Sensors Overview

- Android Sensors:
  - MIC
  - Camera
  - Temperature
  - Location (GPS or Network)
  - Orientation
  - Accelerometer
  - Proximity
  - Pressure
  - Light

Async Callbacks

- Android’s sensors are controlled by external services and only send events when they choose to.
- An app must register a callback to be notified of a sensor event.
- Each sensor has a related XXXListener interface that your callback must implement.
  - E.g. LocationListener
Getting the Relevant System Service

- The non-media (e.g. not camera) sensors are managed by a variety of XXXXManager classes:
  - LocationManager (GPS)
  - SensorManager (accelerometer, gyro, proximity, light, temp)
- The first step in registering is to obtain a reference to the relevant manager
- Every Activity has a getSystemService() method that can be used to obtain a reference to the needed manager

```java
public class MyActivity ...
    private SensorManager sensorManager;
    public void onCreate()
        ...
        sensorManager_ = (SensorManager) getSystemService(SENSOR_SERVICE);
    }
```

Registering for Location Updates

- The LocationManager handles registrations for GPS and network location updates
- In order for an object to receive updates from GPS, it must implement the LocationListener interface
- Once the LocationManager is obtained, an object registers for updates by calling requestLocationUpdates (there are multiple versions you can use)
- The arguments passed into the requestLocationUpdates method determine the granularity of location changes that will generate an event
  - send updates that are at least X meters apart
  - send updates at least this far apart in time
  - send updates that have this minimum accuracy

```java
public class MyActivity ... implements LocationListener{
    private LocationManager locationManager;
    public void onCreate()
        ...
        locationManager_ = (LocationManager) getSystemService(LOCATION_SERVICE);
        locationManager_.requestLocationUpdates(LocationManager.GPS_PROVIDER, 10, Criteria.ACCURACY_FINE, this);
    }
```
Location Providers

- The phone's location can be determined from multiple providers
  - GPS
  - Network
- GPS location updates consume significantly more power than network location updates but are more accurate
  - GPS: 25 seconds * 140mA = 1mAh
  - Network: 2 seconds * 180mA = 0.1mAh
- The provider argument determines which method will be used to get a location for you
- You can also register for the PASSIVE_PROVIDER which only updates you if another app is actively using GPS / Network location

```java
public class MyActivity ... implements LocationListener{
    private LocationManager locationManager_;
    public void onCreate(){
        locationManager_ = (LocationManager) getSystemService(LOCATION_SERVICE);
        locationManager_.requestLocationUpdates(LocationManager.PASSIVE_PROVIDER, 10,
                Criteria.ACCURACY_FINE, this);
    }
}
```

The LocationListener Interface

```java
public class MyActivity ... implements LocationListener{
    ...  
    // Called when your GPS location changes
    @Override
    public void onLocationChanged(Location location) { ...
    }

    // Called when a provider gets turned off by the user in the settings
    @Override
    public void onProviderDisabled(String provider) { ...

    // Called when a provider is turned on by the user in the settings
    @Override
    public void onProviderEnabled(String provider) { ...

    // Signals a state change in the GPS (e.g. you head through a tunnel and it loses its fix on your position)
    @Override
    public void onStatusChanged(String provider, int status, Bundle extras) { ...
    }
}
```
public class MyActivity ... implements LocationListener{
    ...
    // Called when your GPS location changes
    @Override
    public void onLocationChanged(Location location) {
        double altitude = location.getAltitude();
        double longitude = location.getLongitude();
        double latitude = location.getLatitude();
        float speed = location.getSpeed();
        float bearing = location.getBearing();
        float accuracy = location.getAccuracy(); // in meters
        long time = location.getTime(); // when the fix was obtained
        // Other useful Location functions:
        // location.distanceTo(dest)
        // location.bearingTo(dest)
    }
}

Being a Good Citizen...

• It is very important that you unregister your App when you no longer need updates
• For example, you should always unregister your listener when your Activity is paused!
• If you unregister when you pause, you must also reregister when you resume
  • This is true for all sensors!

public class MyActivity ... {
    private LocationManager locationManager_;
    public void onCreate(Bundle savedInstanceState) {
        locationManager_ = (LocationManager)getSystemService(LOCATION_SERVICE);
    }
    protected void onPause() {
        locationManager_.removeUpdates(this);
    }
    protected void onResume() {
        locationManager_.requestLocationUpdates(LocationManager.GPS_PROVIDER, 10,
            Criteria.ACCURACY_FINE, this);
    }
}
Registering for Sensor Updates

- The SensorManager handles registrations for
  - Accelerometer, Temp, Light, Gyro
- In order for an object to receive updates from GPS, it must implement the SensorEventListener interface
- Once the SensorManager is obtained, you must obtain a reference to the specific sensor you are interested in updates from
- The arguments passed into the registerListener method determine the sensor that you are connected to and the rate at which it will send you updates

```java
public class MyActivity ... implements SensorListener{
    private Sensor accelerometer;
    private SensorManager sensorManager;
    public void connectToAccelerometer() {
        sensorManager_ = (SensorManager)getSystemService(SENSOR_MANAGER);
        accelerometer_ = sensorManager_.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
        sensorManager_.registerListener(this, accelerometer_, SensorManager.SENSOR_DELAY_NORMAL);
    }
}
```

The SensorEventListener Interface

- Because there is one interface for multiple types of sensors, listening to multiple sensors requires switching on the type of event (or creating separate listener objects)
- Also forces registration at the same rate per listener
- Simple approach:

```java
public class MyActivity ... implements SensorListener{
    // Called when a registered sensor changes value
    @Override
    public void onSensorChanged(SensorEvent sensorEvent) {
        if (sensorEvent.sensor.getType() == Sensor.TYPE_ACCELEROMETER) {
            float xaccel = sensorEvent.values[0];
            float yaccel = sensorEvent.values[1];
            float zaccel = sensorEvent.values[2];
        }
    }

    // Called when a registered sensor's accuracy changes
    @Override
    public void onAccuracyChanged(Sensor arg0, int arg1) {
        // TODO Auto-generated method stub
    }
}
```
Android System Services

- Each App runs in its own process
- Each Android system service, such as the LocationManager, also runs in its own thread

This has important implications:
1. Communication with the system services is through IPC
2. The thread that delivers an event will be a special thread that is dedicated to processing incoming IPC calls
3. If you directly update the GUI from any thread other than the display thread, bad things happen

How to Update the GUI with Sensor Data

- Android has a built-in mechanism for queuing work that needs to be run on the display thread
- The Handler class allows you to create a queue inside of your Activity that can store work for the display thread
- You create a handler once and then post work to it

```java
public class MyActivity ... implements SensorListener {
    private class AccelWork implements Runnable {
        private Location data_;
        public AccelWork(Location d) {data_ = d;}
        public void run() {
            // do something with the data to the GUI
        }
    }
    private Handler myHandler_ = new Handler();
    // Called when a registered sensor changes value
    @Override
    public void onSensorChanged(SensorEvent sensorEvent) {
        AccelWork work = new AccelWork(sensorEvent);
        myHandler_.post(work);
    }
}
```
Android App Building Demo

Questions?
Connecting Apps to the Cloud

Processing / Storing Data Off the Phone

1. Shared Perspective of Participants
2. Smartphone App for Exhibition Used to Capture/Upload Images
3. Thousands of Participants Contribute Images Through the Smartphone App
Cloud Computing Motivation

• In the real world, you may need tens, hundreds or thousands of servers to handle peak request load
  • Who has the most servers:
    • Intel has 100,000 servers
    • Google is estimated to have 450,000 servers
    • Microsoft has around 300,000 servers
  • The number of servers that you need to handle the requests fluctuates
    • e.g. Amazon needs a lot more servers at Christmas
  • Servers cost money to buy, maintain, and supply power to
  • Ideally, you only want as many servers running as you actually need for the current request load

Cloud Computing Motivation

• Cloud computing uses virtualization to allow computing resources to be dynamically provisioned to applications
• Applications can request virtual machines with specific configurations and the cloud platform automatically allocates and boots them
  • Memory
  • Processor capabilities
  • Disk space
  • OS type
• VMs insulate applications running on the same physical host from each other
Cloud Computing Costs

- Cloud computing platforms typically charge for resources per hour.
- Prices vary based on a variety of factors:
  - Type of resource
  - Location of resource
    - North America
    - Europe
  - Load on resource
  - Up-front contracts
- As a consumer of cloud computing services, your goal is to minimize your overall costs.

Amazon EC2 Costs

<table>
<thead>
<tr>
<th>Region</th>
<th>Linux/UNIX Usage</th>
<th>Windows Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>US East (N. Virginia)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Standard On-Demand Instances</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Small (default)</td>
<td>$0.085 per hour</td>
<td>$0.12 per hour</td>
</tr>
<tr>
<td>Large</td>
<td>$0.24 per hour</td>
<td>$0.18 per hour</td>
</tr>
<tr>
<td>Extra Large</td>
<td>$0.68 per hour</td>
<td>$0.56 per hour</td>
</tr>
<tr>
<td>Micro On-Demand Instances</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Micro</td>
<td>$0.02 per hour</td>
<td>$0.03 per hour</td>
</tr>
<tr>
<td>Hi-Memory On-Demand Instances</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Extra Large</td>
<td>$0.39 per hour</td>
<td>$0.53 per hour</td>
</tr>
<tr>
<td>Double Extra Large</td>
<td>$1.00 per hour</td>
<td>$1.24 per hour</td>
</tr>
<tr>
<td>Quadruple Extra Large</td>
<td>$2.00 per hour</td>
<td>$2.48 per hour</td>
</tr>
<tr>
<td>Hi-CPU On-Demand Instances</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Medium</td>
<td>$0.17 per hour</td>
<td>$0.29 per hour</td>
</tr>
<tr>
<td>Extra Large</td>
<td>$0.68 per hour</td>
<td>$1.10 per hour</td>
</tr>
<tr>
<td>Cluster Compute Instances</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Quadruple Extra Large</td>
<td>$1.00 per hour</td>
<td>N/A **</td>
</tr>
<tr>
<td>Cluster GPU Instances</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Quadruple Extra Large</td>
<td>$2.00 per hour</td>
<td>N/A **</td>
</tr>
</tbody>
</table>

Cloud Auto-scaling

- Ideally, at any point in time your application will only use exactly the computing resources it needs to handle its current request load.
- Auto-scaling is the process of automatically monitoring application load and adjusting resource allocation to match.
- Lots of different ways to decide when to autoscale:
  - CPU load
  - Request response time
- Key issue is determining when to scale and by how much:
  - Out of control applications.
Apps Commonly Use Cloud Backends

- Many apps are built by one or two person teams with no startup funding
- Clouds don’t require up front infrastructure costs in order to have good quality server infrastructure
- Auto-scaling and automation can significantly reduce admin time and allow for more time to be focused on development
- Developers may not have the right skills for managing server infrastructure
- If servers break, new VMs can be allocated

Downside…

- You don’t control the infrastructure
- Things do break…
  - Amazon EC2 has had major outages
  - Google App Engine has had major outages
- Maintenance schedules are out of your hands
  - Google App Engine decides when it puts key services on hold for maintenance and you have no say
- Design patterns are critical so that your application is not tightly-coupled to provider-specific services
  - Amazon Simple DB
  - Google App Engine datastore
Two Cloud Approaches

There are two primary approaches to cloud computing platforms

- **Infrastructure as a Service** – provision and configure VMs exactly as needed, full access to OS, middleware, and other configuration options
  - Amazon EC2

- **Platform as a Service** – just run my server-side application
  - Google App Engine
  - Amazon Elastic Beanstalk

Amazon EC2

Amazon EC2 is probably the most mature and stable cloud computing platform

- Primarily an infrastructure as a service platform
- Wide variety of services
  - Message queues
  - No-SQL datastore
  - Relational datastore
  - Load balancing
  - Network configuration
  - Storage

- Near complete control over your infrastructure but higher setup cost
- Free service for new users
Amazon EC2 Services

Google App Engine

- Google’s App Engine is a cloud computing “platform as a service”
- App Engine is based on Java Servlets or Python
- App Engine automatically starts/stops new instances of your servlets on multiple servers to match your application’s load
- App Engine is 100% free up to a point
- Once you exceed your free App Engine quota, you pay for the CPU, database, and other resources that you use
Research Platform

My group typically uses App Engine for research projects:

• 100% free
• Simple to create apps and automatically launch them with Eclipse-based tools
• Good local testing environment based on Eclipse
  • Simplifies debugging
• Web-based admin portal for controlling user access
• Very little work beyond creating your Servlet-based application
• But... Amazon Elastic Beanstalk probably does all of this too…

Typical App + App Engine Architecture

Phone-initiated Communication

HTTP

The Cloud
Servlet

Server-initiated Communication

Push Notifications
Communicating with App Engine in Android

Most of the time, you need to fetch or send data to the server from the phone:
- Android can send HTTP requests to the cloud using its HTTP Client library.
- The HTTP Client allows you to programmatically construct, send, and get the result of HTTP requests.
- Provides facilities for storing cookies, performing multipart uploads, etc.

If the server needs to push data to the phone, it can use Google’s Cloud 2 Device Messaging (C2DM) system.

How Does App Engine Handle Your HTTP Requests?

App Engine is based on Java Servlets:
- A Java servlet is a class that is responsible for handling HTTP requests.
- Typically, a servlet dynamically generates content (probably from a database).
- Each servlet handles requests for one or more specific sets of resources.
- A web application container starts, manages, and stops servlets.
- The web application container decides which servlet an incoming request should be routed to.
- App Engine automatically manages the web application container.
What is a Java Servlet?

- POST /photouploadurl

Servlet Internals

- All Servlets must implement the javax.servlet.Servlet interface
- Most servlets extend the built-in base class javax.servlet.http.HttpServlet
- Servlets plug into a web application server
  - There is no main method
  - A Servlet must implement a series of life-cycle methods so that the application server can create, initialize, destroy the servlet
  - On startup, the init(...) method gets called
    - This is where you grab references to resources that the Servlet needs
  - On shutdown, the destroy() method gets called
    - You release resources here
  - Every incoming request calls the service(...) method
    - In general, you don’t override this method but instead override the more fine-grained doXXX methods that we will discuss in a minute
A Simple Servlet

```java
package org.vt.app1;
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class SomeServlet extends HttpServlet {
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        resp.setContentType("text/plain");
        resp.getWriter().write("Go Hokies!");
    }
}
```

Handling Different Request Methods

- Each servlet has one or more doXXXX methods
- Each request type has a corresponding doXXXX method
  - Examples:
    - doPost
    - doGet
    - doPut
- In order to handle a request method type, your servlet must implement the corresponding doXXXX method

```java
protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
    // your code
}

protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
    // your code
}
```
Accessing Data from the Request

- The doXXX methods are passed a HttpServletRequest object that has methods for accessing data from the request
- You can obtain parameters that were sent with the request
- You can figure out who made the request and what resource they were requesting (in case you are handling multiple resources in the same servlet)

```java
protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
    String firstName = req.getParameter("FirstName");
    String lastName = req.getParameter("LastName");
    String addr = req.getRemoteAddr();
    String host = req.getRemoteHost();
    String url = req.getRequestURL().toString();
}
```

Generating Output for the Response

- The doXXX methods are passed an HttpServletResponse object that can be used to generate the response sent back to the client
- First, you need to tell the client what type of content will be returned to it by calling setContentType (this is going to generate a content type header)
- Next, you ask for a writer to generate content for the message body of the response
- You can also send binary content (e.g. image data) back by getting an OutputStream from the response

```java
protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
    resp.setContentType("text/plain");
    resp.getWriter().write("Go Hokies!");
    // or, you can write binary data with an OutputStream
    OutputStream out = resp.getOutputStream();
    out.write(…your binary data…);)
```
App Engine Web Apps

- App Engine uses the standard Java WAR format
- Google’s Eclipse-based tools can automatically create a compliant project and build setup
- WAR format can be ported with no code changes to another web application container (e.g. Jetty on EC2) as long as no App Engine-specific APIs are used
  - Image serving
  - URL Fetch
  - Blob store
  - Etc.

Anatomy of a Web App

- Servlets
- Web App Configuration and URL Mapping
- Static Resources
- App Engine Configuration
- Web App Configuration and URL Mapping
The Mapping from Resources to Servlets

- Servlets are added to your web application by first adding a new `<servlet>` node and associated children.
- The servlet is mapped to one or more resources by specifying a pattern to match against incoming URLs of requests.

```xml
<?xml version="1.0" encoding="utf-8"?>
<web-app>
  <servlet>
    <servlet-name>WebApp1</servlet-name>
    <servlet-class>org.vt.app1.WebApp1Servlet</servlet-class>
  </servlet>
  <servlet>
    <servlet-name>WebApp2</servlet-name>
    <servlet-class>org.vt.app1.SomeServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>WebApp1</servlet-name>
    <url-pattern>/webapp1</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
    <servlet-name>WebApp2</servlet-name>
    <url-pattern>/foo</url-pattern>
  </servlet-mapping>
</web-app>
```
Data Persistence

- App Engine provides data persistence using its datastore
- The datastore has custom query languages for searching for data
- The datastore has some very specific limitations / design decisions that you must be aware of to build scalable apps
- Store data in entities
- App Engine provides a web-based admin panel for viewing/editing/deleting data

Key-oriented Lookups

- App Engine datastore is designed for very fast key/value lookup operations – operate closer to a distributed hashtable than a database
- On AppEngine, a direct key lookup is 4-5X faster than a query
- Design your application around key/value pairs
- Wherever possible, use functions to calculate keys for lookups rather than queries to find keys

```java
PersistenceManager pm = PMF.get().getPersistenceManager();
Key k = KeyFactory.stringToKey(key);
MyImage e = pm.getObjectById(MyImage, k);
```
Fine-grained Entities

• When you fetch data by key, you are grabbing all the data associated with the key
• If possible, use fine-grained entities to only store exactly the data that you need for a particular operation to avoid unnecessary data transfer
• Example: A file system in a data store
  • Option 1: Store both file metadata and content in the same entity
  • Option 2: Store file metadata and content in separate entities
  • Producing a listing of files will be MUCH faster with Option 2 because reading through the list of files does not require loading the contents of every file!

Pre-computed Views

• Because reads using keys are so much faster, it is better to pre-compute and store “views”
• Rather than querying and computing values on demand, create a separate database entity to store the precomputed value of the query
• Each time a write takes place, update any affected precomputed entities

• Example: A user can create articles on Reddit. Articles can be voted up or down by other users. A user’s karma is calculated as the average rating of their articles.
  • Option 1: Create a complex query to select and process all of a users’ articles on demand to generate their karma
  • Option 2: Create a separate “karma” entity that stores the current average karma and article count. Each time an article is voted up/down automatically update the karma value
Write Contention & Sharding

- Writes require locking a specific datastore entity
- While the lock is held, all other writes to that same entity block

- Example: You track the number of votes that each particular web page on your site has received. Every time a new vote comes in, you update the vote count for the corresponding page.

- Problem: If you have a popular page with multiple votes per second, the voting will be serialized and a major performance bottleneck.

- Solution: Shard your counters into multiple copies. Each time a vote comes in, randomly choose one of the multiple shards to increment the count for. To get the overall count, sum the values of the shards.

Object-relational Mapping

- AppEngine and other cloud-based platforms as a service use object-relational mappings (ORM) on top of the datastore
- ORM allows developers to interact with a datastore as if it was a collection of persistent OO objects
- Rather than reading/writing rows and columns, you write code that reads/writes objects and their member variables
- The goal is to make database programming closer to object-oriented programming

- In ORM, you create objects, set member variable values, and persist the objects
Storing Data

• The App Engine uses Java Data Objects (JDO) to store/retrieve data in the database
• JDO is a standard for Object-relational Mapping (ORM)
• ORM uses objects as the main abstraction for data in a database rather than rows/columns
• Each object is stored in a row in the database
• Queries return objects rather than raw data
• An object stored in the database is called an “entity”

```java
PersistenceManager pm = PMF.get().getPersistenceManager();
Query query = pm.newQuery("select from MyImage where name==imageName");
query.declareParameters("String imageName");
List<MyImage> results = (List<MyImage>) query.execute(name);
```

Storing Data

• The JDO platform is used to automatically create tables and decide their schema
• Developers markup their classes with special comments called annotations, that dictate how instances of the class should be persisted in the database

```java
@PersistenceCapable
class MyImage {
    @PrimaryKey
    @Persistent(valueStrategy = IdGeneratorStrategy.IDENTITY)
    private Key key;
    @Persistent
    private String name;
    @Persistent
    private String userName;
    @Persistent
    private String comment;
}
```

This annotation means that the class can be stored in the database.

This annotation will make this member variable the primary key and auto-generated.

This annotation means that this member variable should be stored in the database.
Querying Data

- Queries are specified using an SQL-like syntax that refers to the member variables of the stored entities
- Rather than returning results as sets of rows, results are automatically converted into objects

```java
PersistenceManager pm = PMF.get().getPersistenceManager();
Query query = pm.newQuery("select from MyImage where name==imageName");
query.declareParameters("String imageName");
List<MyImage> results = (List<MyImage>) query.execute(name);
```

Storing/Querying Geographic Data

- Data is stored using lat/lon
- Because reads using keys are so much faster, you don't want to issue queries like `l < lat1 && l > lat2 && lon < lon1 && lon < lon2`
- Instead, you use lat/lon to precompute a key that combines the lat and lon

Hash: 0
The key computation works by successively slicing the world in half.
Each time you slice the area, you decide which half of the slice the location falls into.
Each new slice divides the remaining area in half.
Mobile Augmented Reality (AR)

- Blends virtual information with real-world imagery
- Goal is to allow virtual information to be associated with a real-world visual context

AR Approaches

- Smartphone AR uses the on-screen camera preview to serve as a “looking glass” that overlays virtual information on top of real-world imagery
- There are currently ~4 approaches to smartphone AR
  - GPS/Compass AR
  - Feature Tracking AR
  - Fiducial Marker AR
  - Fused Cyber-physical Model AR
GPS / Compass AR

- GPS / Compass AR works by geotagging cyber-information so that it has latitude and longitude
- On the phone, the GPS, accelerometer, and compass are used to estimate the user’s location and field of view
- Once the field of view is determined, the cyber-information that should be in view is overlaid on top of the camera preview

Pros:
- Fast / scalable
- Works well from a distance

Cons:
- Inaccurate due to sensor noise
- Visual jitter
- Performs poorly up close

GPS / Compass AR Issues

- GPS and compass are not 100% accurate
- GPS may be off by several meters
- Compass readings may vary by several degrees
- Readings may change value even when phone is stationary
- Inaccuracy / Jitter causes problems in field of view calculations and may cause information that should be in view not to show up or vice-versa
Improving Accuracy with Filtering

• Filters can be applied to the compass and GPS data to reduce jitter and better determine actual sensor state
  • e.g. Finite Input Response (FIR) filter
  • Filtering, however, reduces the “responsiveness” of the augmented display
  • As the user begins moving the phone, the initial changes in position may be filtered out as noise causing a lag in updating the virtual information in view

GPS / Compass AR Toolkits

• http://www.iphonear.org/
• https://github.com/haseman/Android-AR-Kit/
Feature Tracking AR

- Feature tracking works by identifying key reference points in images
- Cyber-information is associated with these key points and overlaid on the real-world imagery

Pros:
- Can be very accurate
- Works well up close

Cons:
- Extremely resource intensive
- Does not scale on phones
  - e.g. track a few features
- May not work well at distance

Feature Detection

- There is no single definition of what constitutes an image feature
- Generally, a feature is a point of “interest” in an image
- For example, an image may be manipulated to produce multiple versions with different brightnesses
- The points in the image that indicate changes in
  - Color, brightness, etc. may be identified as features

- A database of image features is stored
- As imagery comes in, the new features are matched
- Once a match is made, associated information is overlayed
Fiducial Marker AR

- Fiducial markers are fixed images that can be attached to physical objects
- Image processing is used on each frame of incoming real-world imagery to determine if a fiducial marker is present
- If a marker is present, because its size/shape is known,
- Pros:
  - Very accurate
  - Works well up close
- Cons:
  - Resource intensive
  - Does not scale past ~5 tags on a phone
  - May not work well at distance

Fiducial Marker AR Toolkits

- http://studierstube.icg.tugraz.ac.at/handheld_ar/stbtracker.php
- https://ar.qualcomm.com/ghostnet/event/developer_guide
Hybrid 4-Dimensional Augmented Reality

Paul Miranda, Dr. Mani Golparvar-Fard, and Dr. Jules White

Cyber-physical Information Association

Construction sites use a cyber model, called a Building Information Model (BIM)

BIM is a 3D CAD model annotated with additional cyber-information:
- Cost
- Materials
- Codes
- Carbon Footprint
- Schedule

(c) M. A. Mortenson Company
Motivating Scenario

Field engineer on site is staring at this foundation wall.
The field engineer suspects that the wall is behind schedule does not meet code or is deviating from the spec.
How can the field engineer query the schedule, codes, and specs for wall?
Current Approach: walk back to the trailer, take out hardcopy plans or laptop, find wall.
No way to visualize spec on top of wall.

Key Challenge: How to Associate Cyber Information with Physical Objects

Construction sites use a cyber model, called a Building Information Model (BIM).
BIM is a 3D CAD model annotated with additional cyber-information:
- Cost
- Materials
- Codes
- Carbon Footprint
- Schedule

Wall ID #123 is behind schedule.
Challenge 1: How to Determine Cyber-Identity of Physical Objects

GPS + Compass can be used to determine location and field of view
- However, GPS + Compass has a lot of error and jitter, can only localize to within meters and calculate field of view...very inaccurately
- Computer vision, try to identify objects in view
  - Very hard to do in real-time
  - Difficult (if not impossible) to identify arbitrary objects quickly
- Possible to attach tags (e.g. barcodes, RFID tags, etc.)
  - Expensive
  - Physical effort tagging large numbers of elements
  - Other challenges (next slide)

Challenge 2: Environmental Change

Construction sites by nature are in transition
- Expand, grow, change over time
- Attaching tags, etc. to construction elements is challenging
  - Huge number of elements
  - Visual tags require getting close
  - Elements are constantly changing
    - Drywall
    - Paint
    - Etc.
  - Visual approaches have to deal with lots of occlusions (e.g. blocked lines of sights) due to moving vehicles, etc.
Fiducial Markers

Bar codes, QR codes, or specific visual markers that can be tracked by CV software

Puts extra burden on users to actively modify area

Poses potential risk of getting in the way of normal user operation since physical entity is required

Limited usage

Limited to tagging physical objects

Can't tag everything

http://reactivision.sourceforge.net/

Challenge 3: Visualizing Cyber-Information in Physical Context

For the information to be useful to the user, it must be precisely overlaid on the user image in accordance to the physical object it represents

Direct visual correlation provides simple association

Information about Object A is not useful if it appears to be associated with Object B or C

Construction requires very high precision in the overlay – a foot of error is WAY too much

It is therefore vital for the location and orientation of the user image to be precisely determine for overlaying information
Visualization of construction progress monitoring

Detection of progress deviation
- Superimpose as-planned model and extract relevant image parts
- Analyze progress

e.g., Color presentation of progress deviations

12/02/2006; 1:13:00 PM (As-built)
12/02/2006; 1:13:00 PM (As-planned)
01/03/2007, 12:35:13 AM

Components that are behind schedule
Components that are ahead of schedule
On Schedule
Ahead of Schedule
Behind Schedule

Project: College of Business Instructional Facility, Photograph Courtesy of Facilities & Services, UIUC.
Overview of the D₄AR modeling pipeline

1. Reconstruct image-based 3D point clouds
2. Develop 4D (3D + time) point clouds
3. Superimpose 4D BIM + 4D point clouds
4. Analyze physical progress deviations

More results on dense reconstruction

<table>
<thead>
<tr>
<th>Project</th>
<th>Images</th>
<th>SfM Points</th>
<th>D₄AR Points</th>
<th>D₄AR Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Student Dining Hall (Basement)</td>
<td>286</td>
<td>61,638</td>
<td>835,644</td>
<td>792,972</td>
</tr>
<tr>
<td>Student Dining Hall (Steel)</td>
<td>168</td>
<td>31,661</td>
<td>302,708</td>
<td>792,972</td>
</tr>
<tr>
<td>Student Dining Hall (Circuitry)</td>
<td>127</td>
<td>4,463</td>
<td>62,786</td>
<td>792,972</td>
</tr>
</tbody>
</table>
Steel Structure Reconstruction

24 images; only 3.0 Mpixel, 2,568,461 points; 19min

Bridge 3D Reconstruction
This is Possible... but it Happens Offline Over Several Hours

Previously, an offline approach
No mobile interface
Hybrid 4-Dimensional Augmented Reality (HD^4AR)

Preliminary research has developed a mobile interface
Can query existing information from a cyper-physical model in seconds using photos from a phone
Much more research to develop more storage, dynamic information association, and other components.

Goal
Overview of Bootstrapping a Cyber-physical Model

1. Photographs

Initial photographs of construction site taken

3D model extracted from photographs with computer vision

2a. 3D Model Extracted from Photographs Using Computer Vision Techniques

Physical 3D Model Extracted with Computer Vision
Overview of Bootstrapping a Cyber-physical Model

1. Photographs
   - Physical 3D Model Extracted with Computer Vision

2a. 3D Model Extracted from Photographs Using Computer Vision Techniques
   - Fused Cyber-physical Model

3. Manual Alignment Creates Fused Cyber-physical Model
   - Manually Created Cyber 3D CAD Model

4. Information Associated with CAD Model Elements
   - Building Code
   - Cost
   - Schedule
   - Material
   - Load Limits

Using the Cyber-physical Model

Query: What are the building codes for this wall?

Feedback: Insufficient photos of foundation wall to derive progress, take a photo over here.

Research Challenges: How do you use the fused 3D cyber-physical model to query, write, and receive feedback from the cyber-information model using a mobile device in close to real-time?
Figure 6: Overview of HD4AR

Photograph taken of wall

1. Field Engineer Takes Photograph
2. Field-of-View and Location Calc.
3. Feature Point Detection
4. Feature Points, Location/View, Error Probability Sent to Server
5. Probabilistic Photo Filtering
6. Feature Matching
7. Structure from Motion Calculations
8. HD4AR Server
9. Augmented Photo Known to Field Engineer
10. Cyber-information and measurement Returned to Device
11. In-view confidence decreases towards edges

Cyber-physical model is used to localize camera position and info
HD⁴AR Overview

1. Field Engineer Takes Photograph

2. Field-of-View and LocationCalc.

3. Feature Point Detection

4. Feature Points, Location/View, Error Probability Sent to Server

5. HD⁴AR Server

6a. Feature Matching

6b. Structure from Motion Calculations

7. Cyber-Identity Determination From 2D to 3D Projections

8. Augmented Photo Returned to Engineer

9. Cyber-information and locations in 2D photo are returned to phone and displayed

Cyber-information in view is determined based on camera position.
No Tagging

Use what is already present in the object (appearance or other quality) as a means of identification, which can be matched to other images of the same object or scene.

GPS alone is too imprecise for good localization and too limited in functionality (cannot operate indoors, does not adjust for user orientation, etc.)

Based on Extracting Image Features and their 3D Position

Extract features from images using detection algorithm (SIFT/SURF/ASIFT).

Use features to build 3D point cloud using Structure-from-Motion.

Low setup cost to build model.

Requires no external modification to site.
Structure From Motion

Uses 2D images to create 3D point cloud
3D position of feature points determined by triangulation using multiple images
Initial model creation takes 20 minutes to hours
Subsequent single images can be localized in 3D model in a few seconds

Research Prototype

Android Prototype

Information retrieval from phone in ~10s
Accurate to within a few centimeters
No need for expensive, specialized hardware
Still much to do:
  Create new information using phone
  3D visualization on phone
  Advanced interactivity
  Query optimization on sever
  Scaling on the cloud
Questions?

Analyzing Mobile Application Power Consumption using SPOT

• Hamilton Turner,
• Chris Thompson, Jules White, Douglas Schmidt
Power as a First-class Design Constraint

Battery capacity is one of the slowest trends in mobile computing
Power as a First-class Design Constraint

Power Consumption should be a first-class design constraint for mobile computing.

Power efficiency should be considered early in the software lifecycle.

What are the goals and challenges for developers in early power consideration?

Ideal Goal: Comparative Analysis of Software Architecture Power Use
Ideal Goal: Comparative Analysis of Software Architecture Power Use

Understand how design decisions impact power consumption

Compare heterogeneous software architectures for power consumption
Challenge: Layers of Abstraction

Hardware
Middleware
Operating System
App 1
App 2
App 3
App 4
App 5

Power consumed
Challenge: Layers of Abstraction

How do we predict the effects of high-level software decisions on low-level power consumption?
Battery life is a key limitation of mobile devices.

Due to abstraction of operating systems, middleware, networking stacks, etc, there is a major disconnect between software design and power consumers.
Current Solution

- Implement and test
  - Code and fix!
- Changes late in development cycle are costly
  - The extra risk directly impacts changes of project success
    - Accurate power instrumentation is tough and error prone

Current Issues in Designing for Power Efficiency

Significant Complexity between power consumers and software decisions

Inability to quantify tradeoffs

1sec vs 100second GPS updates?
Model driven power analysis

System Power Optimization Tool (SPOT) combines modeling with automatic code generation to provide design-time power consumption analysis.

Solution: System Power Optimization Tool (SPOT)

Model Creation → Code Generation → Execution → Analysis
DSML: System Power Optimization Model Language (SPOML)

- Allows visual declaration and configuration of hardware consumers on mobile devices
  - CPU Consumers
  - Memory Consumers
  - Sensor Consumers
  - GPS, Accelerometer, etc
  - Network Consumers
  - Screen Drawing Agents
  - Custom Code Modules

- Generic Components Allow Code Generation for Multiple Target Smartphone Platforms
Modeling Application Architectures with SPOML: WreckWatch

• Components to Model
  • GPS Consumer
  • Network Consumer
  • Accelerometer Consumer
Walk-thru SPOT configuration

Model Creation → Code Generation → Execution → Analysis

Walk-thru SPOT configuration

- NetworkConsumer
  - UseCellNetworkOnly – Boolean
  - TransmissionScheme – Integer
  - Data – String
  - UseWiFiOnly – Boolean
Generating Power-Instrumented Code

• Visual modeler outputs XML describing resource consumers

• Generic resource consumers have been implemented for each testing platform

• Generic consumers are configured using XML properties

Platform-specific executable code generated from generic components

Power instrumentation code generated to interface with the power API

Types of instrumentation
  • Event-focused
  • Periodic

Executable containing architecture and power instrumentation

Model Creation → Code Generation → Execution → Analysis

Generated Code

<table>
<thead>
<tr>
<th>Software Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU Consumer</td>
</tr>
<tr>
<td>GPS Consumer</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Power Instrumentation Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collector</td>
</tr>
<tr>
<td>Recorder</td>
</tr>
</tbody>
</table>
Generated code is pushed to real device
Configured hardware consumption occurs
Power-instrumentation occurs
Data recording

Using SPOT to Evaluate Software Architecture Decisions

- Initial Power consumption is uniform
- Differences in Sensing Rates Only Substantially Affect Power over long periods of sampling
- Sampling <2 minutes implies that greater sampling rates can be used with minimal negative effect on battery life
Evaluating Generated Code Power Consumption Accuracy

Method: Compare Generated Code Power Consumption with Real Code Power Consumption

- Modeled two existing smartphone applications
- Compared power consumption of SPOT equivalents with original applications
- SPOT predicted both applications to within 4% power consumption
Future Work

• Power consumption for certain system-level operations, such as garbage collection, are not included in SPOT
• Some sensors draw substantially more power
• Hardware consumers are not typically used to describe software architectures
• Context-dependent power consumption

Conclusion

• We presented SPOT, a useful tool for design time power analysis
• A case study showed the SPOT methodology is effective for assisting early power tradeoffs
• SPOT’s accuracy was roughly verified via manual comparison
• Future work and tools are needed for design-time power consumption
• We plan to examine context-dependent power consumption
Questions?
What is electronic pairing?

The act of connecting two electronic devices for communication

- Bluetooth pairing
- Wireless printers
- Laptop and WLAN

Importance of secure pairing

- Sensitive data transmissions
- Man-in-the-Middle attacks, packet sniffing, backdoor attacks[2]
Existing methodologies

Visual (IR), aural (biometric), short-range wireless (Near Field Communication), physical contact (Bump) [1]

PBC (HW/SW buttons), PIN (shared number), USB (mediator) [3]

Example – Disaster Relief

First responder and victim in a disaster area
Need to share information: personal information, medical records, emergency contacts
Sensitive data needs quick security w/o 3rd party
Problem

How can smartphones translate “physical identity” into “virtual identity”?

Need to verify the end-receiver is the intended phone.

Challenge 1: No Third Party

Reliance upon a 3rd party, like Cloud, increases security risk

How to achieve desired security with just peer-to-peer methods
Challenge 2: Ease of Use

Some pairing techniques are not intuitive or make it difficult to pair. How to establish identity without peripherals or walkthroughs.

Challenge 3: Security uniqueness

Given what is only on the smartphone, how can it’s identity be unique and secure to another user? What sensors or hardware can provide this?
Solution

Gesture-based pairing using smartphone accelerometer.

Augmented security for Bluetooth requires users to move their phones in similar manners in order to communicate.

Accelerometer data is shared between two phones, then compared for similarity.

Similar gestures = authentification.
Demo

Design challenge: sensor noise

Accelerometers are noisy (mechanical noise and electrical thermal noise)\(^4\)

What to do to make data more useful?

Filtering? Bound-checking?
Design decision: ease of pairing vs. key space size

Two approaches for handling and comparing accelerometer data:

Average data over time and compare acceleration averages of 25 samples with an empirically-determined threshold of 1 m/s² difference in average acceleration.

Compare gestures sample-by-sample with the same threshold between 25 concurrent data points.

Second approach made it impossible to verify gestures. Too much uniqueness due to noise.

Results

Table 1. Average acceleration comparison 0.5 m/s² threshold

<table>
<thead>
<tr>
<th>Gesture</th>
<th>Success Rate (10 trials), 0.5 error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Up-Down Pivot (Perpendicular)</td>
<td>3 out of 10</td>
</tr>
<tr>
<td>Up-Down Pivot (Parallel)</td>
<td>6 out of 10</td>
</tr>
<tr>
<td>Up-Down (Parallel)</td>
<td>3 out of 10</td>
</tr>
<tr>
<td>Right-Left Pivot (Parallel)</td>
<td>1 out of 10</td>
</tr>
<tr>
<td>Right-Left (Parallel)</td>
<td>6 out of 10</td>
</tr>
<tr>
<td>Front-Back (Parallel)</td>
<td>4 out of 10</td>
</tr>
</tbody>
</table>

Table 2. Average acceleration comparison 1.0 m/s² threshold

<table>
<thead>
<tr>
<th>Gesture</th>
<th>Success Rate (10 trials), 1.0 error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Up-Down Pivot (Perpendicular)</td>
<td>6 out of 10</td>
</tr>
<tr>
<td>Up-Down Pivot (Parallel)</td>
<td>10 out of 10</td>
</tr>
<tr>
<td>Up-Down (Parallel)</td>
<td>8 out of 10</td>
</tr>
<tr>
<td>Right-Left Pivot (Parallel)</td>
<td>9 out of 10</td>
</tr>
<tr>
<td>Right-Left (Parallel)</td>
<td>7 out of 10</td>
</tr>
<tr>
<td>Front-Back (Parallel)</td>
<td>7 out of 10</td>
</tr>
</tbody>
</table>
Results

Table 3. Most consistent gesture data

<table>
<thead>
<tr>
<th></th>
<th>X1:</th>
<th>Y1:</th>
<th>Z1:</th>
<th>X2:</th>
<th>Y2:</th>
<th>Z2:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average</td>
<td>-0.0255</td>
<td>2.007155</td>
<td>8.39965</td>
<td>0.303718</td>
<td>2.08648</td>
<td>8.47674</td>
</tr>
<tr>
<td>Standard Deviation:</td>
<td>0.184264</td>
<td>0.798864</td>
<td>0.459517</td>
<td>0.278235</td>
<td>0.874768</td>
<td>0.344801</td>
</tr>
<tr>
<td>StDev Sum:</td>
<td>1.442645014</td>
<td>1.497803551</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Average Diff:</td>
<td>0.3292175</td>
<td>0.079325</td>
<td>0.07709</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X2 - X1</td>
<td>Y2 - Y1</td>
<td>Z2 - Z1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Conclusion

People can understand the physical world, and smartphones can use physical events to secure key exchange.

Automatic gesture-based pairing provides the ease of use desired as well as the uniqueness in security.
Questions?

References

Motivating Example: Mapping Cellular Signal Strength

Consumers currently use cell coverage maps generated by the company that runs the network:

- The map is rough e.g. may only have 3-4 different 'signal strength' indications
- The information used to create this map is hidden, and there is no guarantee the information is recent
- The map is 'general' e.g. not very useful for a specific model of phone
- The maps are often based on signal propagation mathematics, not real world data
Incorporating Mobile Cyber-Physical Systems into Cellular Signal Mapping

- Smartphones can be used to . . .
  - Sample the cellular signal strength at their current GPS location
  - Submit samples to a server, along with type and location of smartphone
- The server can then combine these measurements
  - The ‘general’ signal strength map can be more trusted, as it’s based on recent, real-world data
  - Signal maps specific to one model of phone or one brand of cellular chipset can be generated

Is this idea feasible?

- Smartphones can be used to . . .
  - Sample the cellular signal strength at their current GPS location
  - Submit samples to a server, along with type and location of smartphone
- The server can then combine these measurements
  - The ‘general’ signal strength map can be more trusted, as it’s based on recent, real-world data
  - Signal maps specific to one model of phone or one brand of cellular chipset can be generated
But does it work?

- What guarantees do we have regarding the accuracy of the generated signal strength maps?
- How efficient is the heatmap generation e.g. are we using 20 readings when we could be performing the same task with 5?
- How resilient is the heatmap generation e.g. if there is a sudden change in the network (new tower online, lightning strike, etc), will we see that change reflected in a reasonable amount of time?
- How resilient is the network to defective sensors or malicious information?
- Measuring this in the field would require a lot of time and money, so we turn to simulation.

Simulating Smartphone Agents

- We chose to use agent-focused simulation, as smartphones have multiple individual properties:
  - Movement pattern
  - Sensors enabled / disabled over time
  - Sensor properties over time e.g. accuracy of the location lock, error of the signal strength measurement
- Additionally, agent simulation allows us to enter purposefully defective or malicious entities and monitor how well the system reacts.
Empower: Smartphone-centric Simulation Environment

- Allows multiple simulation parameters
  - Number and location of smartphones over time
  - Smartphone properties such as sensor accuracy, battery life, etc
  - Sampled environment (incl. changes over time)
  - Policy Decisions
- Generates high-level visualization
- Generates data metrics for simulation analysis
Analyzing cellular map generation efficiency using Empower

- Returning to our prior question: ‘How efficient is the heatmap generation e.g. are we using 20 readings when we could be performing the same task with 5’
- We created two data collection policies to test this
  - ‘static’ data collection implies that a phone will always collect ad report signal strength data
  - ‘context-aware’ data collection implies that a phone will report more frequently if it’s location changes significantly, and less frequently if it’s location remains relatively constant
- To gather results, we ran two simulations with all parameters the same except one had all ‘static’ data collection, and one had all ‘context-aware’

Problem: Wasted Data Readings

- Chart showing comparison between static policy and context-aware data collection over time.
Result Analysis and Next Steps

- We were able to show that even simple context-awareness (e.g. monitoring location) has potential to increase system efficiency.
- This is not an unexpected result, but the work to date provides a solid foundation for the next stages of cutting edge research in smartphone-powered opportunistic or participatory sensing systems.
- Next steps:
  - Testing other methods of context awareness.
  - Examining other research questions using Empower.
  - Preparing novel components of Empower for inclusion in mainstream simulation environments.
  - Using field data or running field trials to verify Empower results.