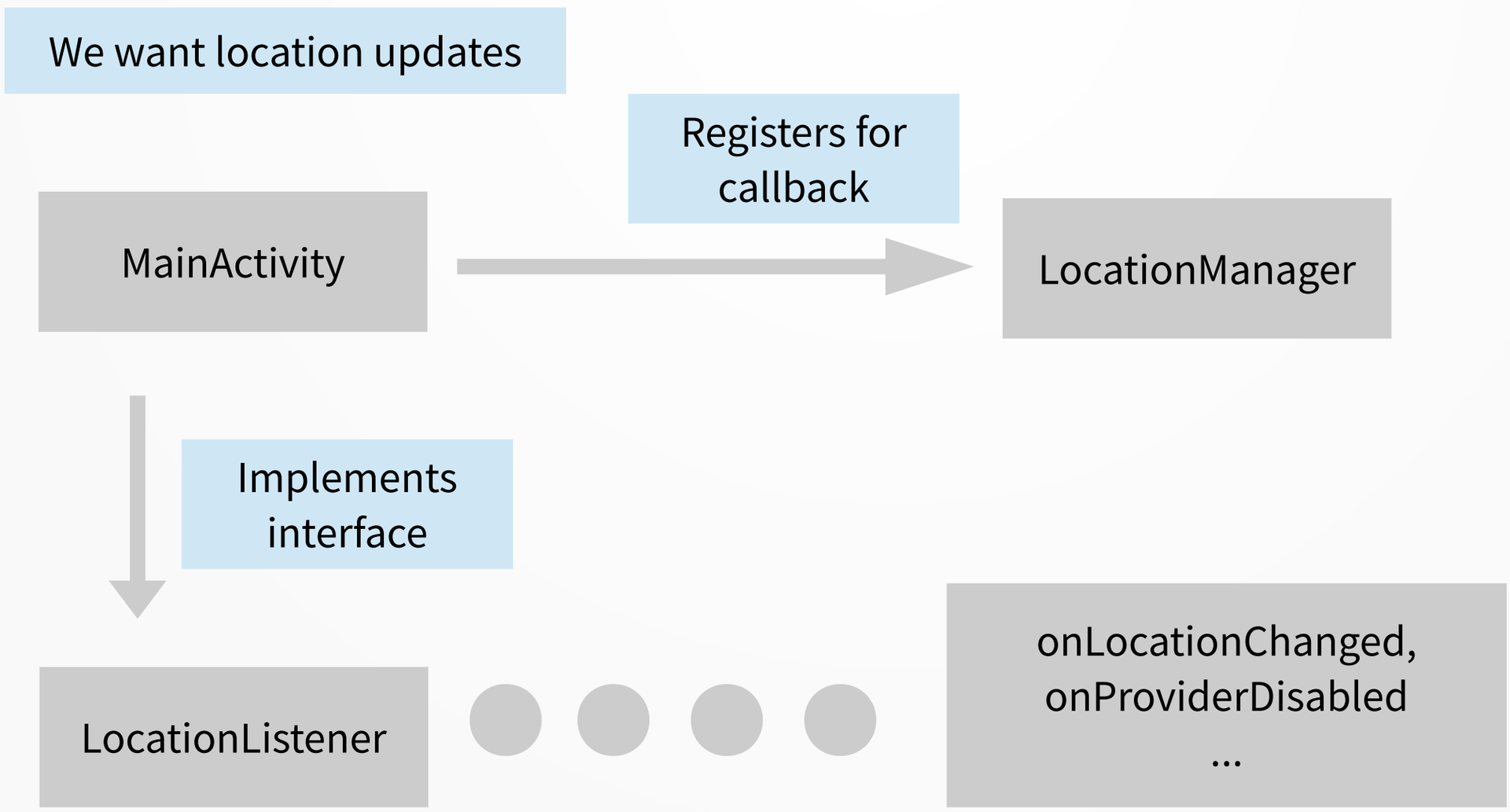
Android Sensor Programming

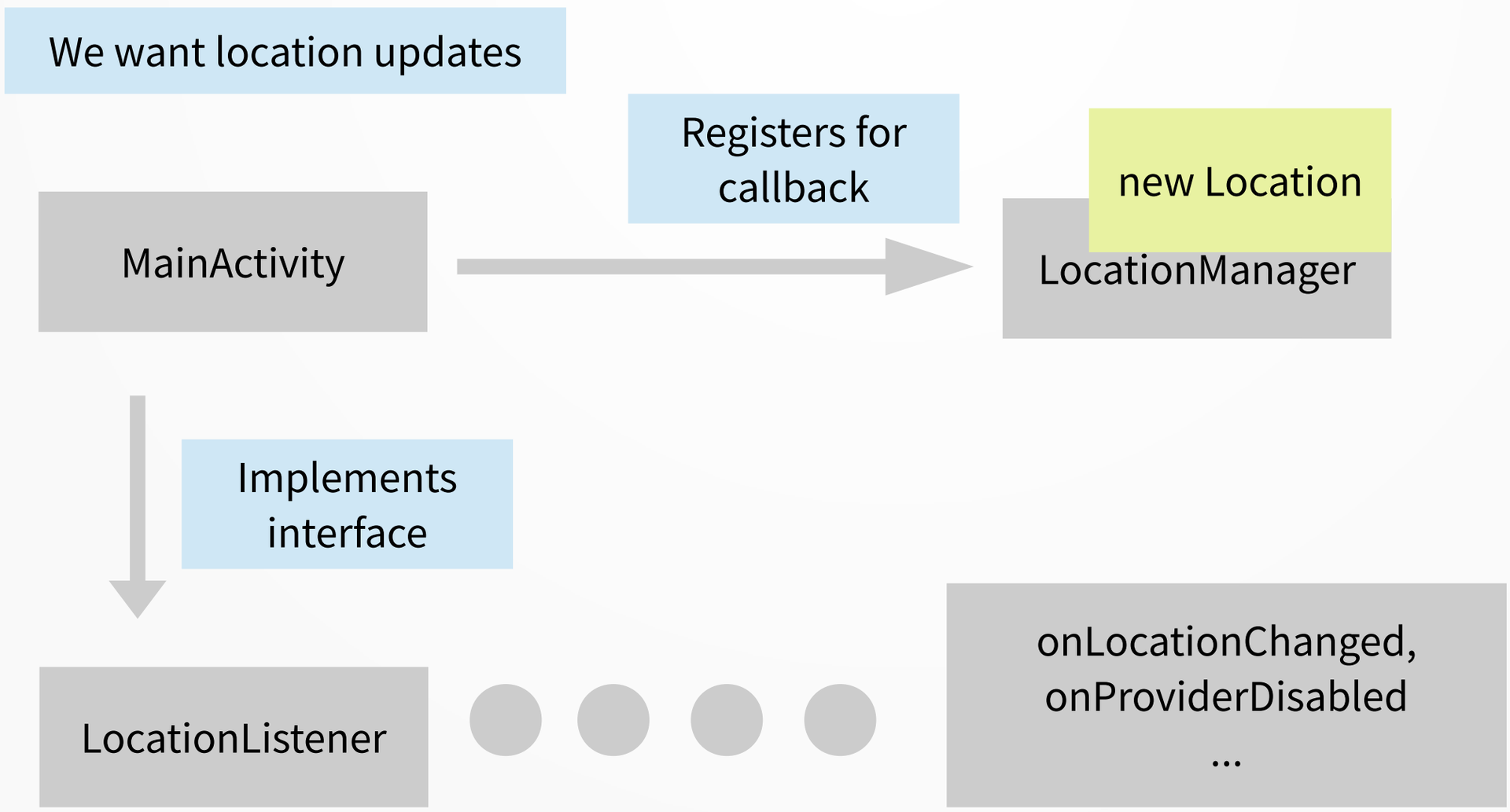
Sensors

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

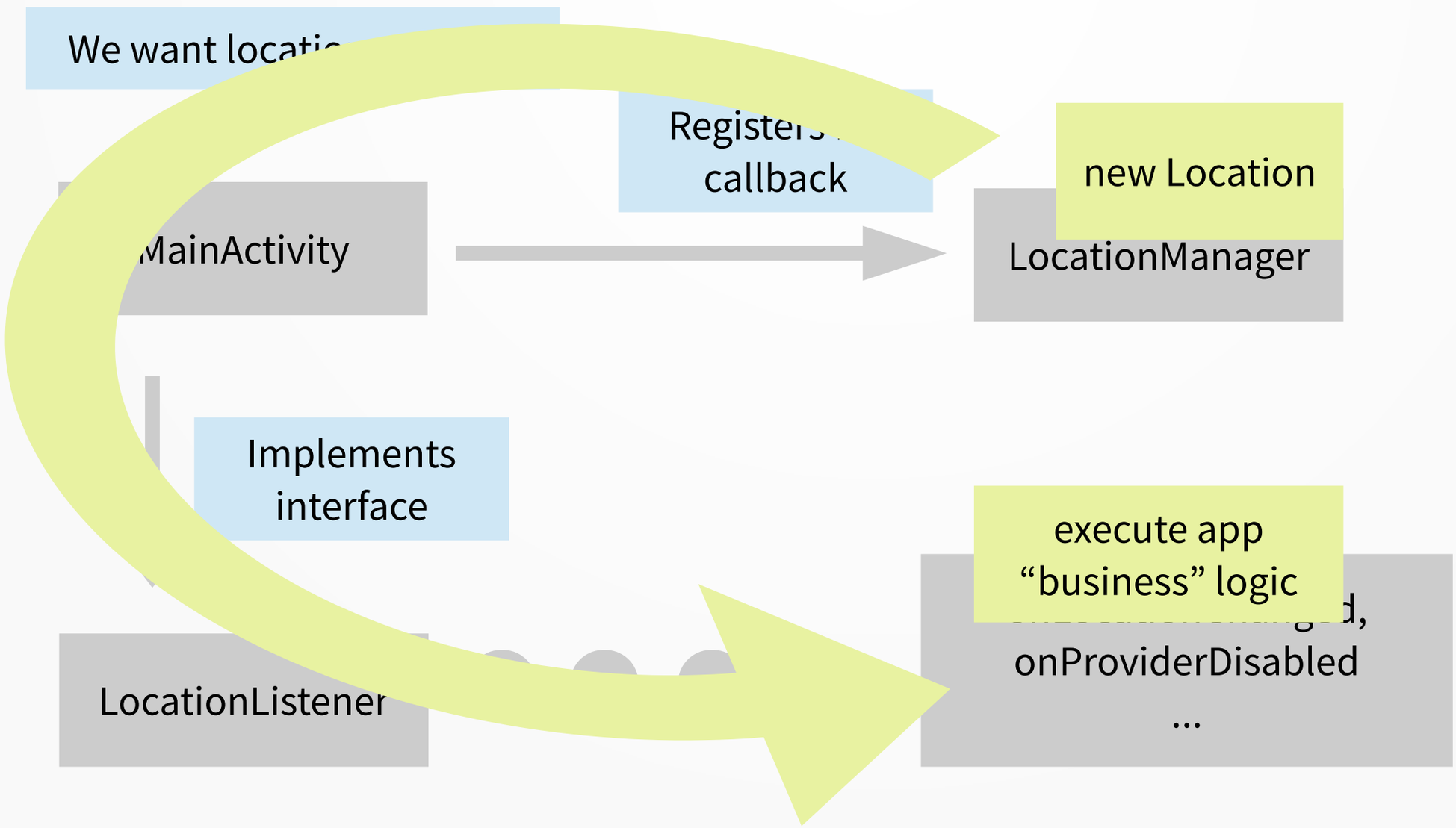
General Pattern



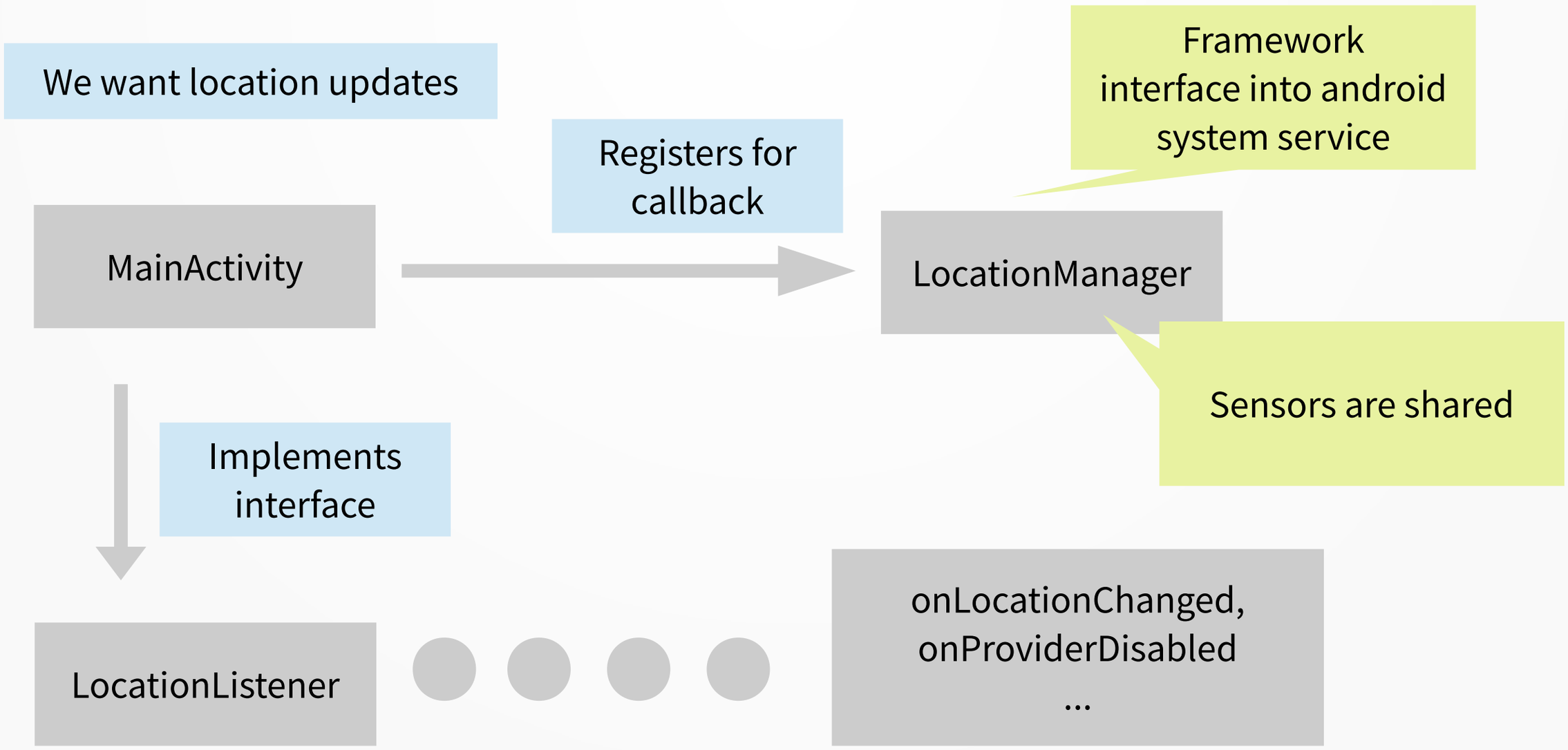
General Pattern



General Pattern



General Pattern



LocationManager

```
public class MyActivity implements LocationListener {

  private LocationManager locationManager_;

  public void onCreate(){
   ...
   locationManager_ =
     (LocationManager) getSystemService(LOCATION_SERVICE);
   locationManager_.requestLocationUpdates(
      LocationManager.GPS_PROVIDER, 10, 50, this);
  }

}
```

Shared system service

LocationManager

```
public class MyActivity implements LocationListener {

  private LocationManager locationManager_;

  public void onCreate(){
   ...
   locationManager_ =
     (LocationManager) getSystemService(LOCATION_SERVICE);
   locationManager_.requestLocationUpdates(
      LocationManager.GPS_PROVIDER, 10, 50, this);
  }

}
```

Type of location provider

Min time between updates (ms)

Min time distance between updates (meters)

Callback object that implements LocationListener

LocationManager

```
public class MyActivity implements LocationListener{

  @Override
  public void onLocationChanged(Location location) {
     // Called when your GPS location changes
  }

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

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

  @Override
  public void onStatusChanged(String provider, int status, Bundle extras) {
    // Signals a state change in the GPS (e.g. you head through a tunnel and
    // it loses its fix on your position)
  }
}
```

Main hook that we need

Other hooks allow us to respond to changes of sensor

LocationManager

```
public class MyActivity implements LocationListener{

  @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
  }

}
```

Up to application to use location updates in the app logic

Sensor data likely drives major application computation (e.g, update turn-by-turn directions from app)

LocationManager

```
public class MyActivity implements LocationListener{

  @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
  }

}
```

Up to application to use location updates in the app logic

Sensor data likely drives major application computation (e.g, update turn-by-turn directions from app)

What technique do you know to logically (and physically) split this work?

Android Multithreading

- General rule of thumb: Push any processing that can “hang” to a separate thread
- Familiar model
  - Java `Runnable` objects
  - A `Hander` class handles thread lifecycle

Android Multithreading

```
public class MyActivity implements LocationListener {
  Handler _handler = new Handler();
  Location _dest;

  private class TurnByTurn implements Runnable {
    Location _now;
    TurnByTurn(Location now) { _now = now; }
    public void run() {
      double d = _now.distanceTo(_dest);
      // do some heavy lifting with d
    }
  }

  @Override
  public void onLocationChanged(Location location) {
    TurnByTurn tbt = new TurnByTurn(location);
    handler.post(tbt);
  }

}
```

Bottle up your unit of work as a Runnable instance

Post to handler

Android Multithreading

```
public class MyActivity implements LocationListener {
  Handler _handler = new Handler();
  Location _dest;

  @Override
  public void onLocationChanged(Location location) {
    handler.post(new Runnable() {
      double d = location.distanceTo(_dest);
      // do some heavy lifting with d
    });
  }

}
```

Dont forget about inline class creation

If unit of work is small, favor this syntactic sugar approach

LocationManager

	Register for location updates using the named provider, and a pending intent.
void	requestLocationUpdates(long minTime, float minDistance, Criteria criteria, LocationListener listener, Looper looper) Register for location updates using a Criteria, and a callback on the specified looper thread.
void	requestLocationUpdates(String provider, long minTime, float minDistance, LocationListener listener, Looper looper) Register for location updates using the named provider, and a callback on the specified looper thread.
void	requestLocationUpdates(long minTime, float minDistance, Criteria criteria, PendingIntent intent) Register for location updates using a Criteria and pending intent.
void	requestLocationUpdates(String provider, long minTime, float minDistance, PendingIntent intent) Register for location updates using the named provider, and a pending intent.

LocationManager

	Register for location updates using the named provider, and a pending intent.
void	requestLocationUpdates(long minTime, float minDistance, Criteria criteria, LocationListener listener, Looper looper)
	read.
void	
	d
void	
	criteria, PendingIntent intent) Register for location updates using a Criteria and pending intent.
void	requestLocationUpdates(String provider, long minTime, float minDistance, PendingIntent intent) Register for location updates using the named provider, and a pending intent.

Why all the interfaces?

LocationManager

	Register for location updates using the named provider, and a pending intent.
void	requestLocationUpdates(long minTime, float minDistance, Criteria criteria, LocationListener listener, Looper looper) read.
void	d
void	criteria, PendingIntent intent) Register for location updates using a Criteria and pending intent.
void	requestLocationUpdates(String provider, long minTime, float minDistance, PendingIntent intent) Register for location updates using the named provider, and a pending intent.

Programming hardware is hard!

LocationManager

Choosing a sensible value for minTime is important to conserve battery life. Each location update requires power from GPS, WIFI, Cell and other radios. Select a minTime value as high as possible while still providing a reasonable user experience. If your application is not in the foreground and showing location to the user then your application should avoid using an active provider (such as `NETWORK_PROVIDER` or `GPS_PROVIDER`), but if you insist then select a minTime of 5 * 60 * 1000 (5 minutes) or greater. If your application is in the foreground and showing location to the user then it is appropriate to select a faster update interval.

Directly from google docs

LocationManager

Choosing a sensible value for minTime is important to conserve battery life. Each location update requires power from GPS, WIFI, Cell and other radios. Select a minTime value as high as possible while still providing a reasonable user experience. If your application is not in the foreground and showing location to the user then your application should avoid using an active provider (such as `NETWORK_PROVIDER` or `GPS_PROVIDER`), but if you insist then select a minTime of 5 * 60 * 1000 (5 minutes) or greater. If your application is in the foreground and showing location to the user then it is appropriate to select a faster update interval.

Directly from google docs

Ever used an app that destroys battery? This is why

LocationManager

Choosing a sensible value for minTime is important to conserve battery life. Each location update requires power from GPS, WIFI, Cell and other radios. Select a minTime value as high as possible while still providing a reasonable user experience. If your application is not in the foreground and showing location to the user then your application should avoid using an active provider (such as `NETWORK_PROVIDER` or `GPS_PROVIDER`), but if you insist then select a minTime of 5 * 60 * 1000 (5 minutes) or greater. If your application is in the foreground and showing location to the user then it is appropriate to select a faster update interval.

Directly from google docs

Ever used an app that destroys battery? This is why

Lots of research into this energy “misuse” (mine!)

LocationManager

android.location

Added in API level 1

Contains the framework API classes that define Android location-based and related services.

This API is not the recommended method for accessing Android location.
The Google Location Services API, part of Google Play services, is the preferred way to add location-awareness to your app. It offers a simpler API, higher accuracy, low-power geofencing, and more. If you are currently using the android.location API, you are strongly encouraged to switch to the Google Location Services API as soon as possible.

To learn more about the Google Location Services API, see the Location API overview.

Google recognized the issue of “too much” control of low level details for high level programmers

Location Services

Accuracy

You can specify location accuracy using the `setPriority()` method, passing one of the following values as the argument:

- `PRIORITY_HIGH_ACCURACY` provides the most accurate location possible, which is computed using as many inputs as necessary (it enables GPS, Wi-Fi, and cell, and uses a variety of Sensors), and may cause significant battery drain.
- `PRIORITY_BALANCED_POWER_ACCURACY` provides accurate location while optimizing for power. Very rarely uses GPS. Typically uses a combination of Wi-Fi and cell information to compute device location.
- `PRIORITY_LOW_POWER` largely relies on cell towers and avoids GPS and Wi-Fi inputs, providing coarse (city-level) accuracy with minimal battery drain.
- `PRIORITY_NO_POWER` receives locations passively from other apps for which location has already been computed.

Express "intent", not details

Frequency

You can specify location frequency using two methods:

- Use the `setinterval()` method to specify the interval at which *location is computed for your app*.
- Use the `setFastestInterval()` to specify the interval at which *location computed for other apps is delivered to your app*.

Location Services API

```
public class MyActivity {
  FusedLocationProviderClient _fused;
  protected void onCreate(…) {
    _fused = LocationServices.getFusedLocationProviderClient(this);
  }

  protected LocationRequest createLocationRequest() {
    LocationRequest locReq = LocationRequest.create();
    locReq.setInterval(10000);
    locReq.setFastestInterval(5000);
    locReq.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);
    return locReq;
  }

  private void startUpdates() {
    _fused.requestLocationUpdates(
       createLocationRequest(),
       new LocationCallback() {
         public void onLocationResult(LocationResult locationResult) {
           if (locationResult == null) {
             return;
           }
           for (Location location : locationResult.getLocations()) {
             // Update UI with location data
             // ...
           }
        };
       },
       null);
  }
}
```

Similar API, but expressing “intent” instead of “details”

Google will choose a provider that satisfies the request

App Battery Usage

Does application energy usage really matter?

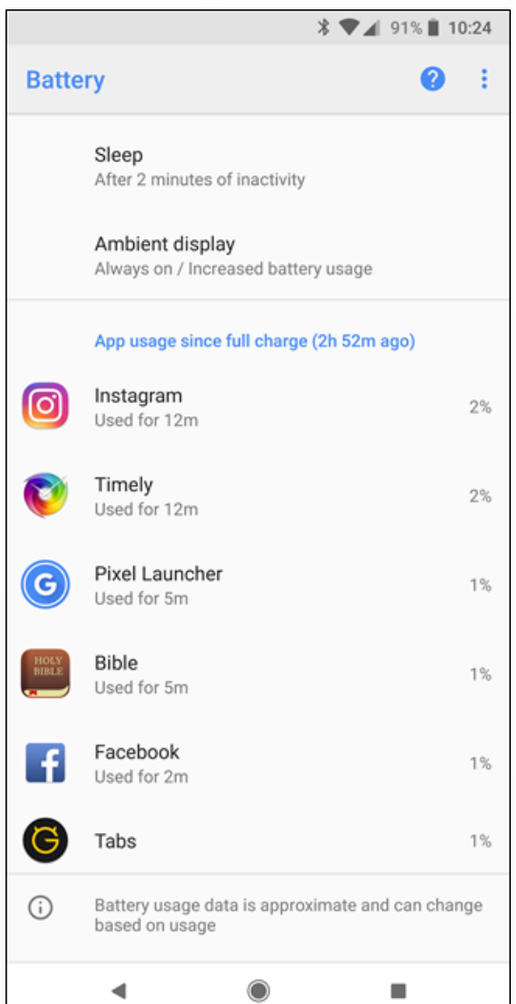
Can I just ignore it?

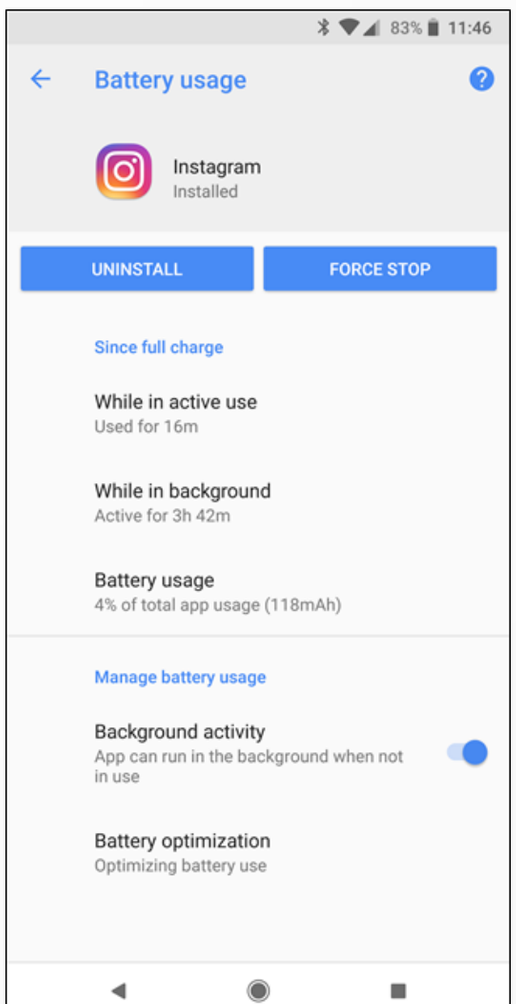
App Battery Usage

Does application energy usage really matter?

Can I just ignore it?

Not likely! Android is becoming more and more power aware





Android Oreo Limits

To lower the chance of these problems, Android 8.0 places limitations on what apps can do while users aren't directly interacting with them. Apps are restricted in two ways:

- Background Service Limitations: While an app is idle, there are limits to its use of background services. This does not apply to foreground services, which are more noticeable to the user.
- Broadcast Limitations: With limited exceptions, apps cannot use their manifest to register for implicit broadcasts. They can still register for these broadcasts at runtime, and they can use the manifest to register for explicit broadcasts targeted specifically at their app.

There used to be a question of whether programmers should consider power at all

In an effort to reduce power consumption, Android 8.0 (API level 26) limits how frequently background apps can retrieve the user's current location. Apps can receive location updates only a few times each hour.

> ★ **Note:** These limitations apply to all apps used on devices running Android 8.0 (API level 26) or higher, **regardless of an app's target SDK version**.

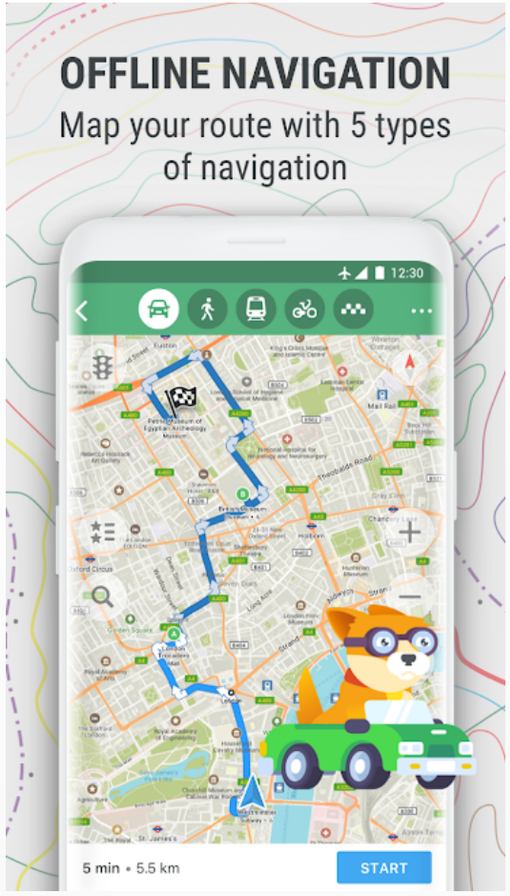
This location retrieval behavior is particularly important to keep in mind if your app relies on real-time alerts or motion detection while running in the background.

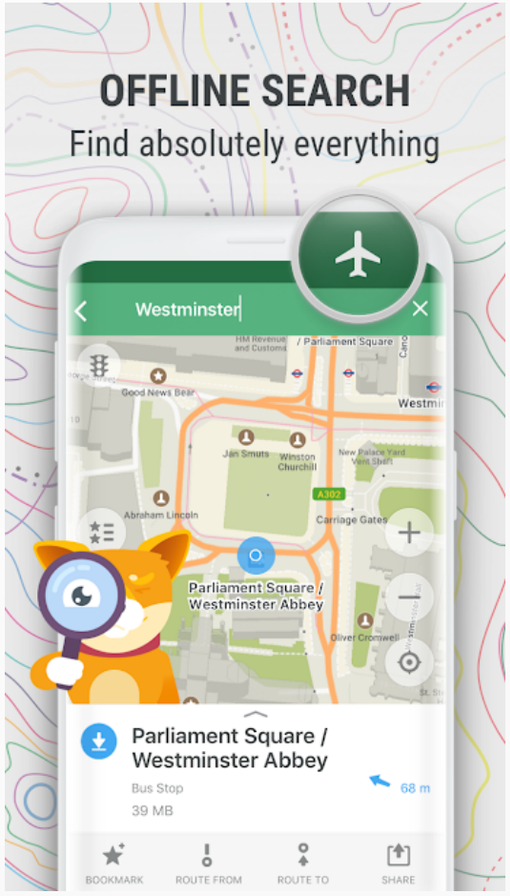
It's no longer a question

Maps.Me

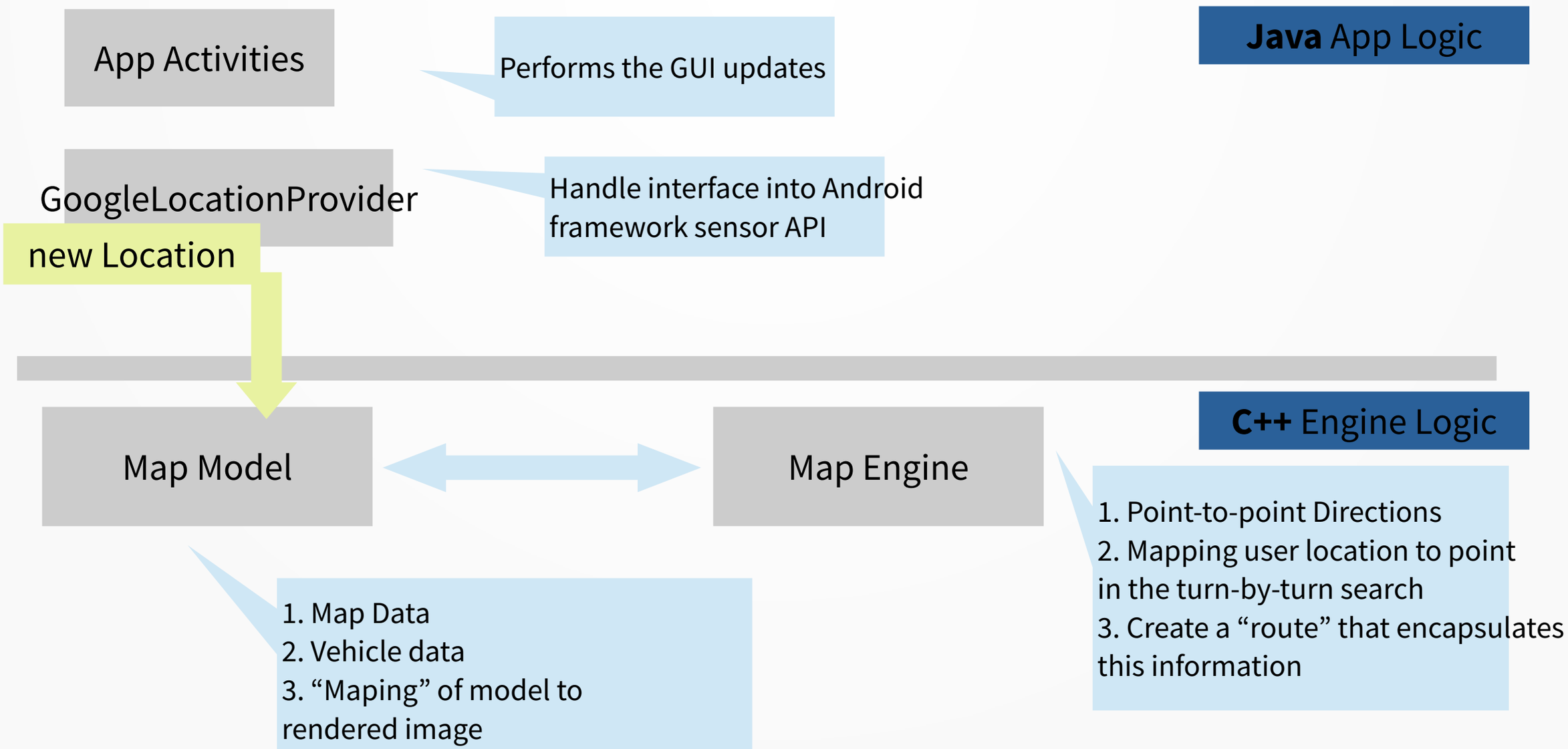
Open source GPS app for android / iOS

Very mature, fully functional





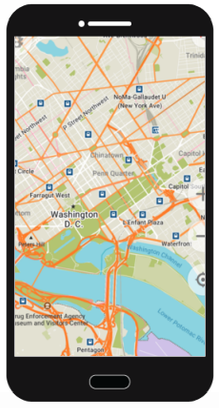
Maps.Me


Java App Logic
App Activities
Performs the GUI updates
GoogleLocationProvider
Handle interface into Android framework sensor API
new Location
C++ Engine Logic
Map Model
Map Engine
1. Point-to-point Directions
2. Mapping user location to point in the turn-by-turn search
3. Create a “route” that encapsulates this information
1. Map Data
2. Vehicle data
3. “Maping” of model to rendered image

Maps.Me

- Demo maps.me app
  - MwmActivity
  - LocationHelper
  - GoogleFusedLocationProvider
  - SensorHelper

Application-Level Energy Optimization



GPS Update Rate
{10s,5s,1s}

GPS Power
{LOW,HIGH}

10s

LOW

20% Battery
Drain Rate

1s

HIGH

30% Battery
Drain Rate

Application-Level Energy Optimization

GPS Update Rate
{10s,5s,1s}

GPS Power
{LOW,HIGH}

10s
LOW

20% Battery
Drain Rate

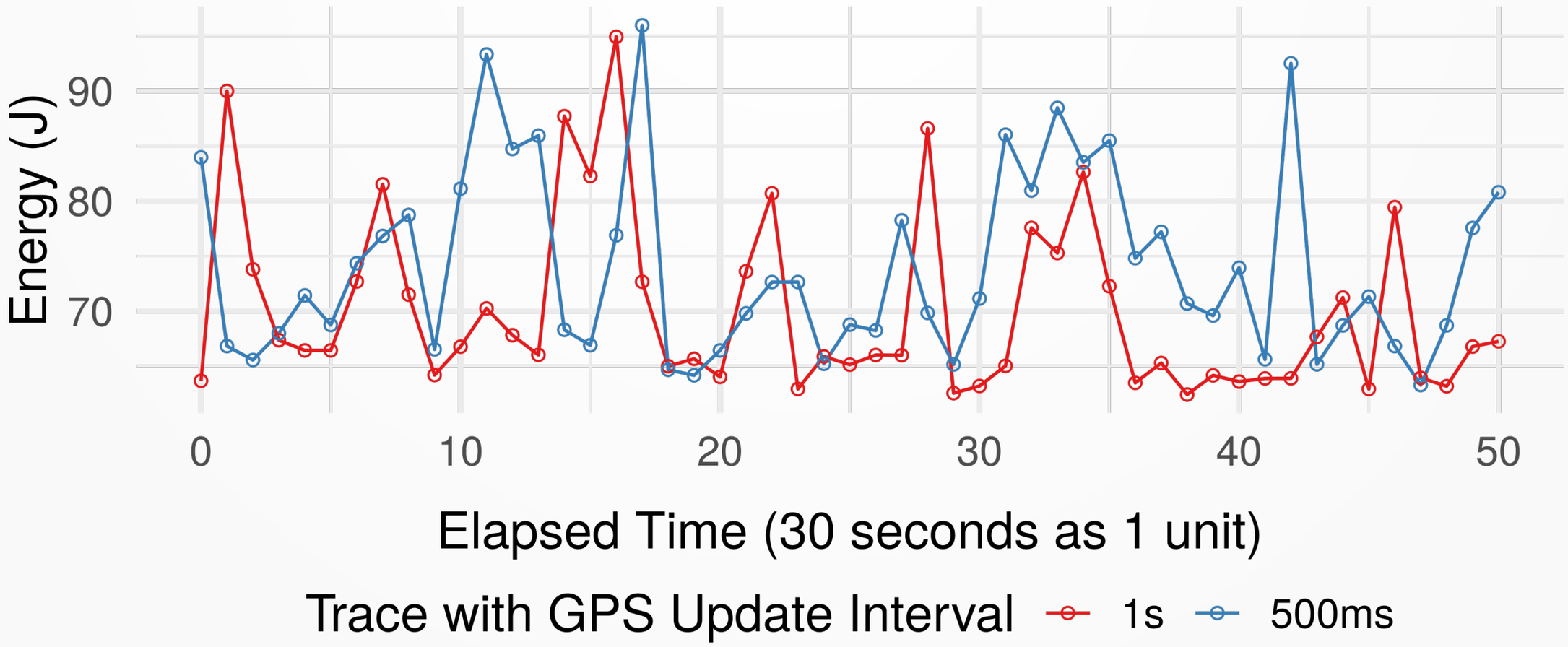
?
?

25% Battery
Drain Rate

1s
HIGH

30% Battery
Drain Rate

Energy Behavior of Mobile Devices



Aeneas API

```
class LocationProvider implements Interactor {
  AeneasMachine aeneas;
  Knob gpsUpdate = new DiscreteKnob(
    new Integer[]{10, 5, 1});

  LocationProvider() {
    aeneas = new AeneasMachine(
      new VBDE(),
      new Knob[]{gpsUpdate},
      new Reward(), this);
  }

  void onInteract() {
    LocationRequest req =
      new LocationRequest();
    req.setUpdate(gpsUpdate.read());
    requestLocationUpdates(req);
  }
}
```

```
class Reward {
   float valuate() {
       float e = this.perInteractionEnergy();
       return this.batteryRate(e);
   }
   float SLA() { return 30; }
}
```

Aeneas API

```
class LocationProvider implements Interactor {
  AeneasMachine aeneas;
  Knob gpsUpdate = new DiscreteKnob(
    new Integer[]{10, 5, 1});

  LocationProvider() {
    aeneas = new AeneasMachine(
      new VBDE(),
      new Knob[]{gpsUpdate},
      new Reward(), this);
  }

  void onInteract() {
    LocationRequest req =
      new LocationRequest();
    req.setUpdate(gpsUpdate.read());
    requestLocationUpdates(req);
  }
}
```

```
class Reward {
   float valuate() {
       float e = this.perInteractionEnergy();
       return this.batteryRate(e);
   }
   float SLA() { return 30; }
}
```

Encoding of an energy specification

Aeneas API

```
class LocationProvider implements Interactor {
  AeneasMachine aeneas;
  Knob gpsUpdate = new DiscreteKnob(
    new Integer[]{10, 5, 1});

  LocationProvider() {
    aeneas = new AeneasMachine(
      new VBDE(),
      new Knob[]{gpsUpdate},
      new Reward(), this);
  }

  void onInteract() {
    LocationRequest req =
      new LocationRequest();
    req.setUpdate(gpsUpdate.read());
    requestLocationUpdates(req);
  }
}
```

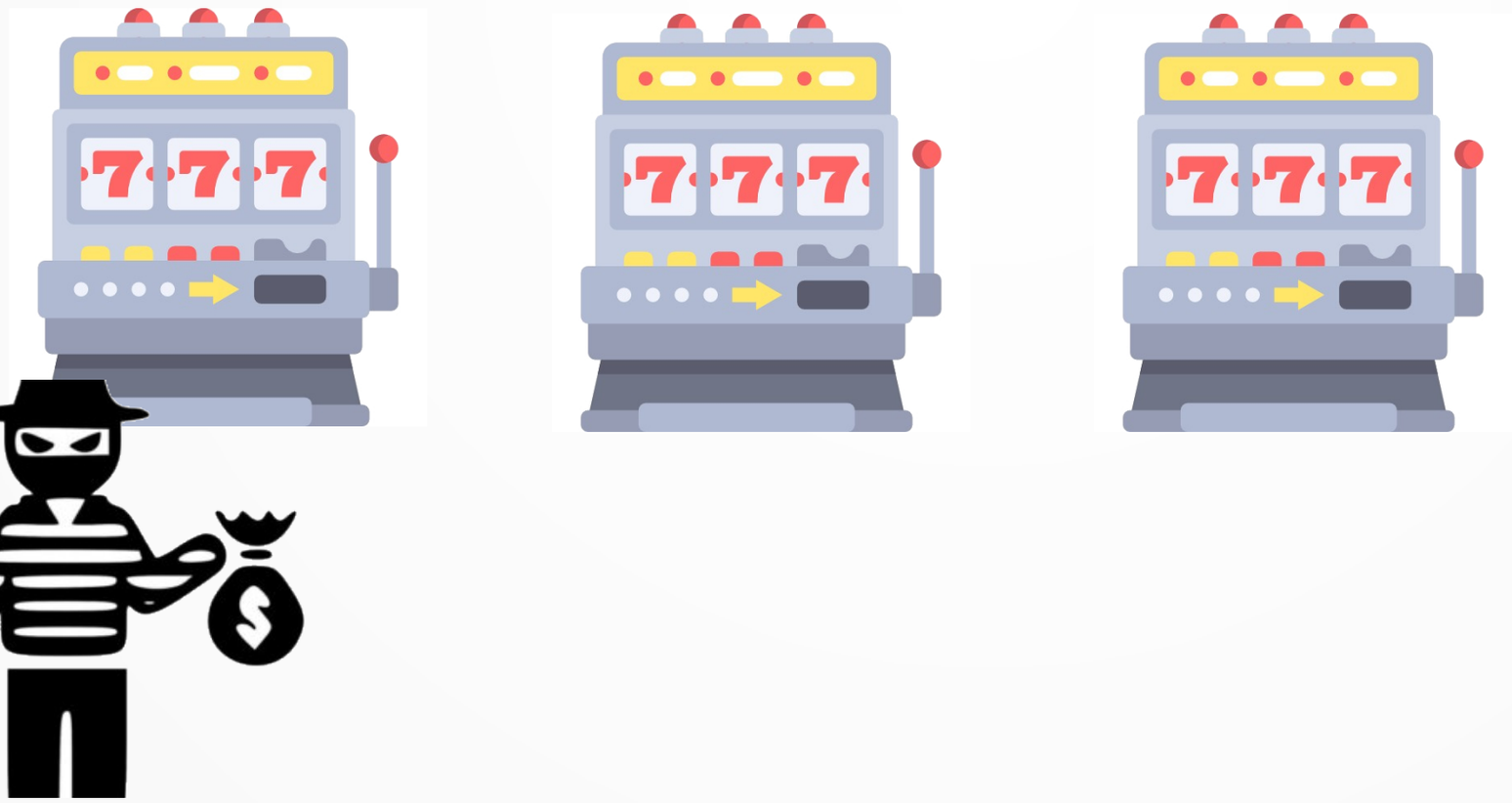
```
class Reward {
   float valuate() {
       float e = this.perInteractionEnergy();
       return this.batteryRate(e);
   }
   float SLA() { return 30; }
}
```

As far as a user is concerned, they are done! But I will briefly discuss details

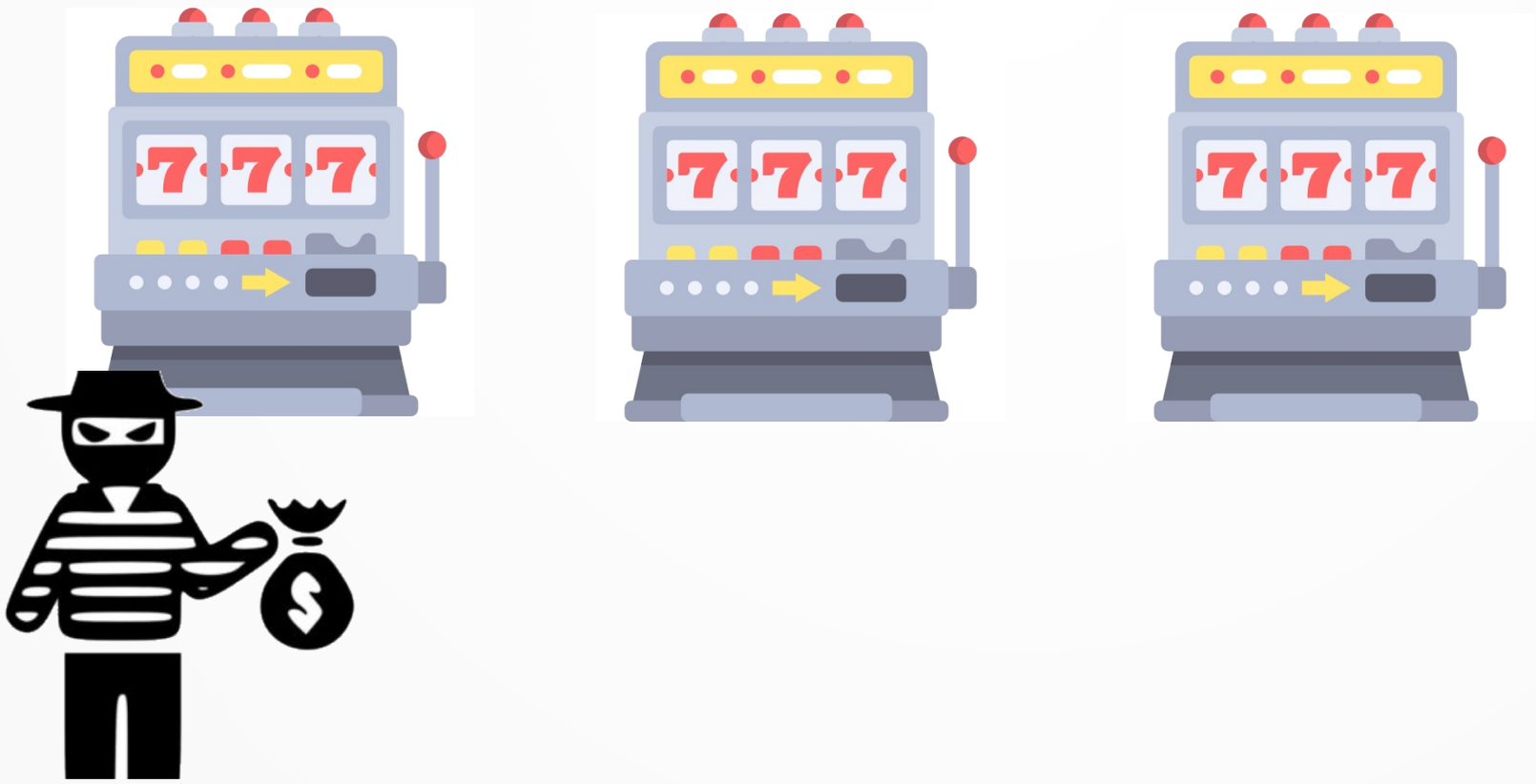
Multi-Armed Bandit



Multi-Armed Bandit



Multi-Armed Bandit



$5

Multi-Armed Bandit

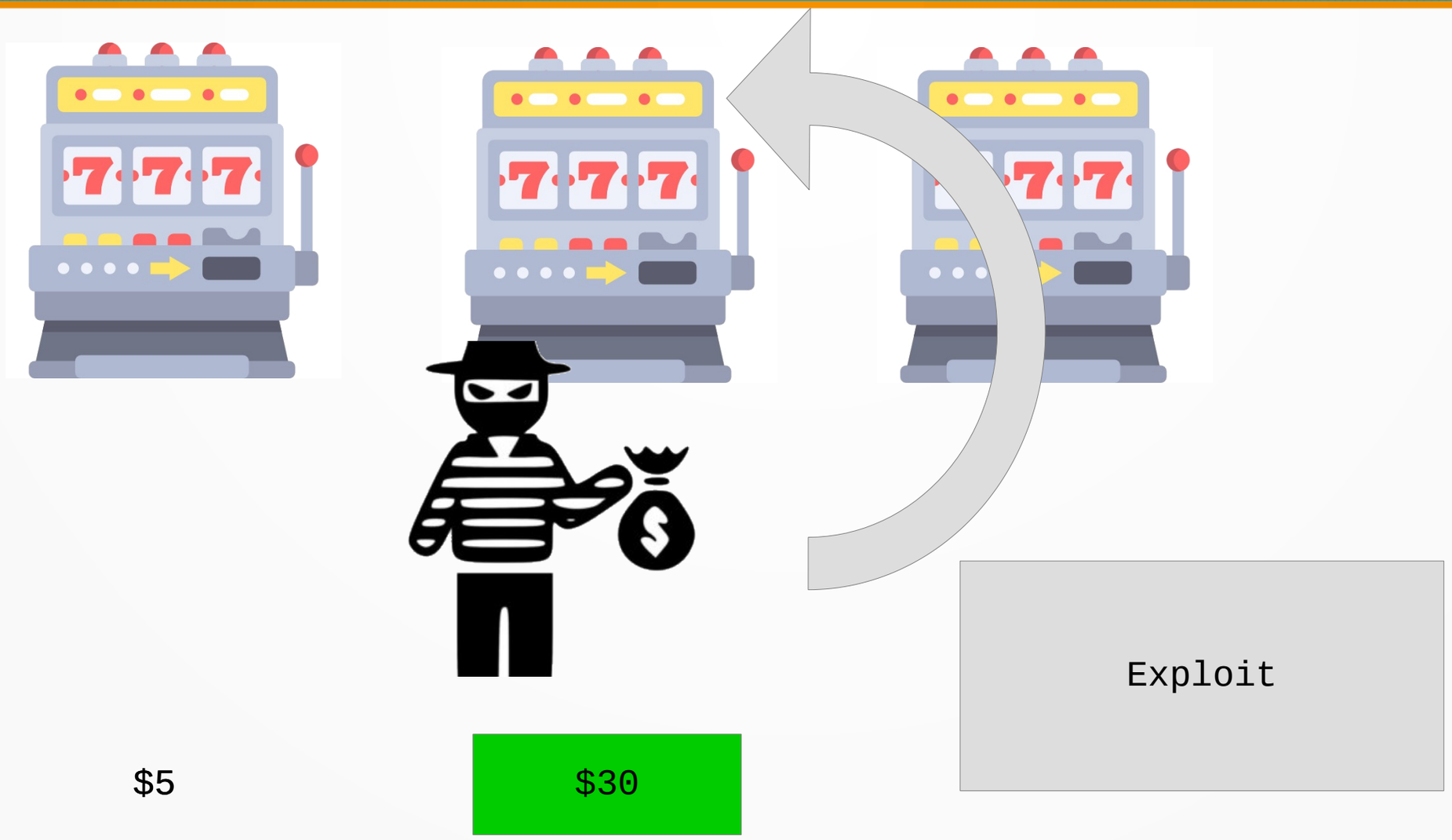
$5

Multi-Armed Bandit

$5

$30

Multi-Armed Bandit



Multi-Armed Bandit

$5

$30

Explore

Multi-Armed Bandit

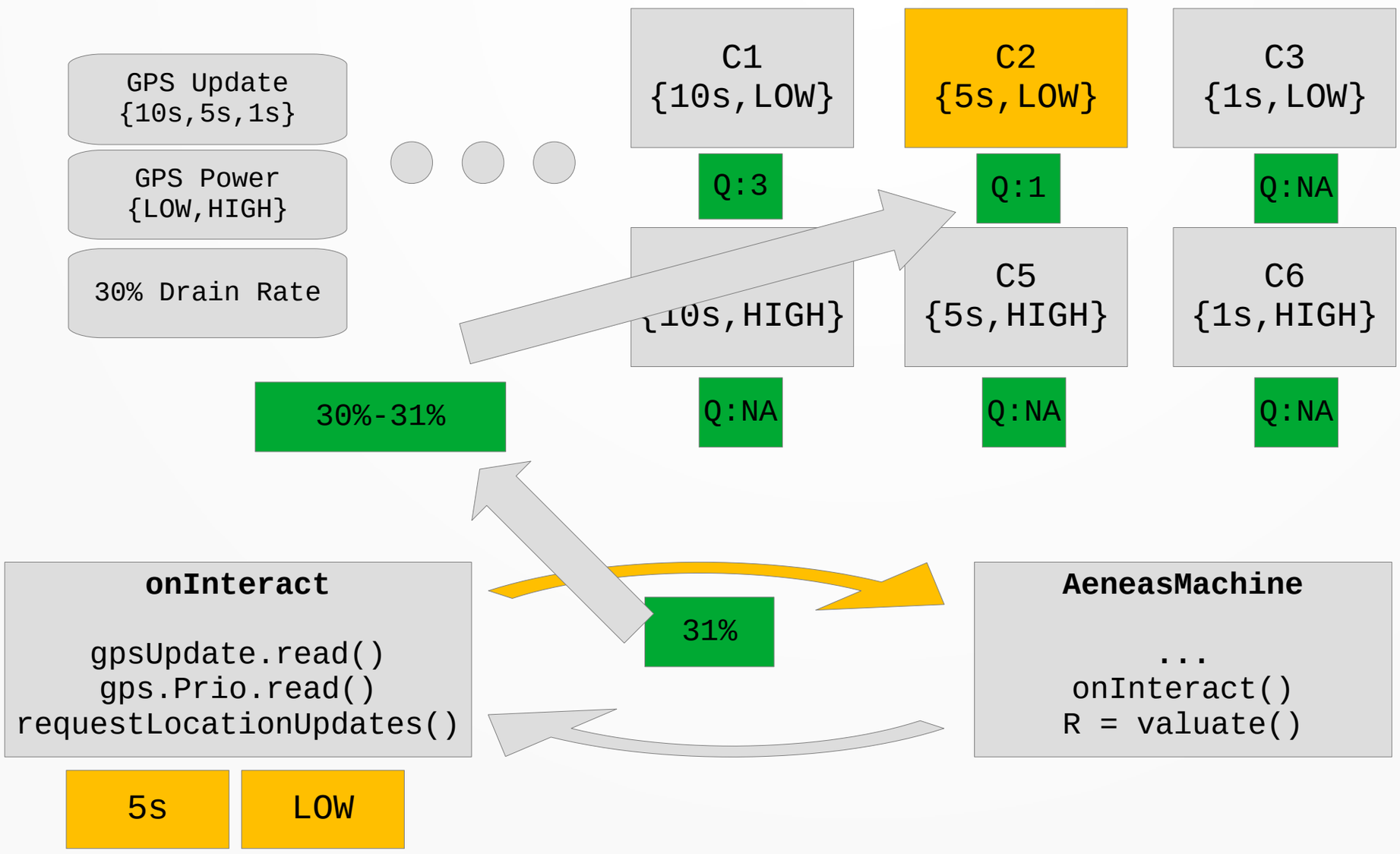
Key Point: Let the machine learn details about the environment

$5

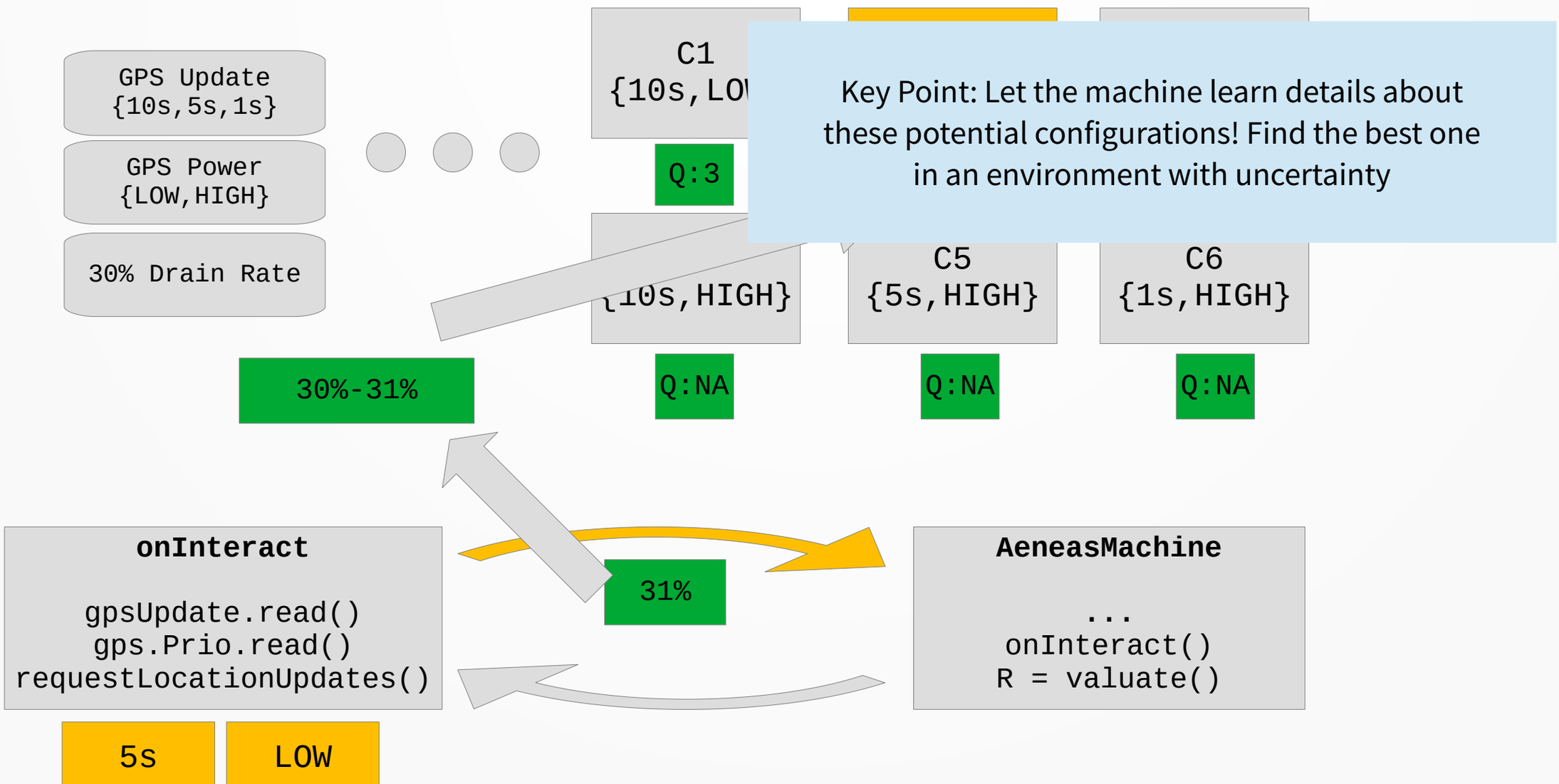
$30

Explore

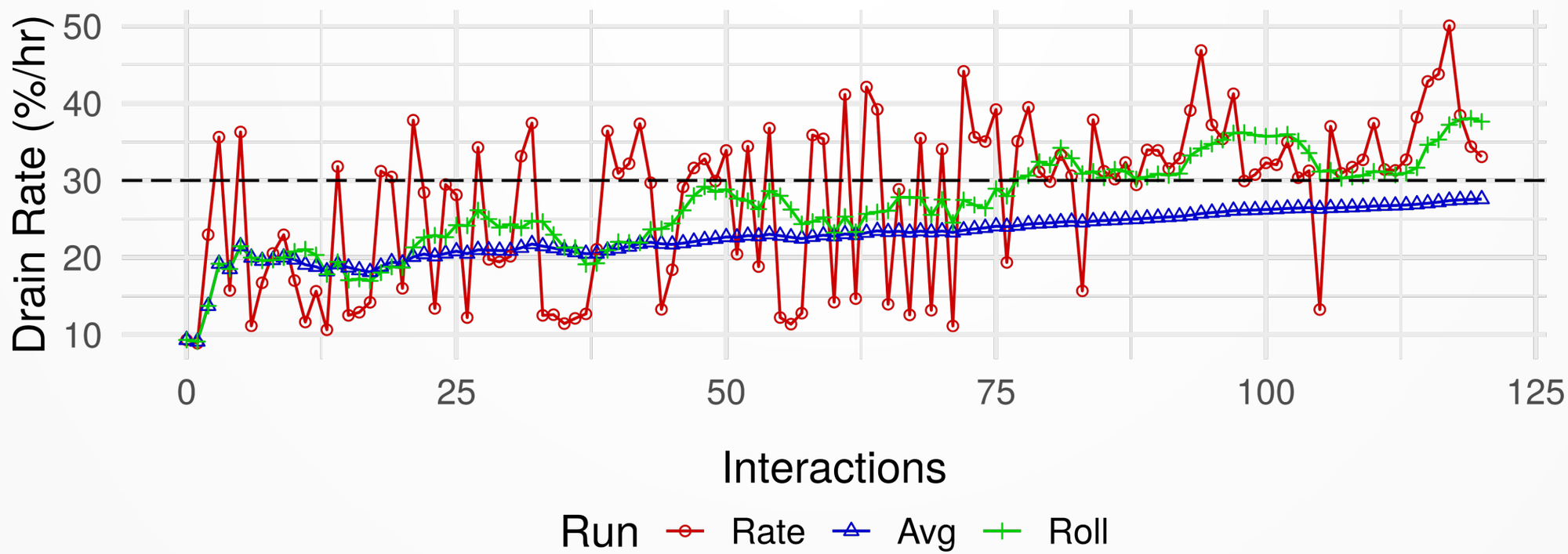
Aeneas Runtime



Aeneas Runtime



Aeneas In Action




```
3 Knobs:
GPS Update Rate {10s,7.5,5,3,1}
GPS Power {LOW,HIGH}
GPS Max Wait {5x,3x,1x}
```

Acknowledgments

- https://github.com/mapsme/omim
- https://developer.android.com/reference/android/location/package-summary
- Yu David Liu, CS476/576
- https://www.howtogeek.com/244748/how-to-see-which-apps-are-draining-your-battery-on-an-android-phone-or-tablet/