Analyzing Android

Shuying Liang, Matt Might*, Cambell Christensen, Hao Hou, Petey Aldous, Celeste Hollenbeck, Will Byrd
A terrible idea
WHAT COULD POSSIBLY
GO WRONG?
DARPA: APAC
App auditor
Good app

App auditor
App auditor
App auditor
App auditor

Good app
You shall not pass

Good app
Step 1
Solve the Halting problem
Step 2
Pass the Turing test
App auditor
Good app
Engagements
In practice?
Vision
50%
90%
93%
How does it work?
MAGIC!?
Small-step analysis
AAM
class MyActivity {
  public MyActivity() {
    activateMic();
  }
}

.class MyActivity
    .method public MyActivity
        invokedynamic activateMic
    .end method
.end class
Look, a malware!
\( pc, \hat{\rho}, \hat{\sigma}, \hat{\kappa} \)
.class MyActivity
    .method public MyActivity
        invokedynamic activateMic
    .end method
.end class
\( \rho \), \( \hat{\sigma} \), \( \hat{\kappa} \)
Method: AAM
\[
\begin{align*}
\langle \text{nop} :: \text{stmt}, fp, \sigma, \kappa \rangle & \mapsto \langle \text{stm}, fp, \sigma, \kappa \rangle \\
\langle \text{move-object}(r_d, r_s) :: \text{stmt}, fp, \sigma, \kappa \rangle & \mapsto \langle \text{stmt}, fp, \sigma[(r_d, fp) \mapsto \sigma(r_s, fp)], \kappa \rangle \\
\langle \text{return-void} :: \text{stmt'}, fp', \sigma, \text{fink}(\text{stmt}, fp, \kappa) \rangle & \mapsto \langle \text{stmt}, fp, \sigma \rangle \\
\langle \text{return-object}(r) :: \text{stmt}, fp', \sigma, \text{fink}(\text{stmt}, fp, \kappa) \rangle & \mapsto \langle \text{stmt}, fp, \sigma[(r, fp) \mapsto \sigma(n, fp')], \kappa \rangle \\
\langle \text{const}(r, c) :: \text{stmt}, fp, \sigma, \kappa \rangle & \mapsto \langle \text{stmt}, fp, \sigma[(r, fp) \mapsto c], \kappa \rangle \\
\langle \text{throw}^\ell(r) :: \text{stmt}, fp, \sigma, \kappa \rangle & \mapsto \langle \text{stmt}, fp', \sigma[(\text{exn}, fp') \mapsto \sigma(r, fp)], \kappa' \rangle \\
& \text{where } (\ell', fp', \kappa') = \mathcal{H}(\ell, fp, \kappa) \\
\langle \text{goto}(\ell) :: \text{stmt}, fp, \sigma, \kappa \rangle & \mapsto \langle \text{stmt}, fp, \sigma \rangle \\
\langle \text{new-instance}(r, \tau) :: \text{stmt}, fp, \sigma, \kappa \rangle & \mapsto \langle \text{stmt}, fp, \sigma[(r, fp) \mapsto o], \kappa \rangle \\
& \text{where } o = \text{new}(\tau) \\
\langle \text{if-eq}(r, r', \ell) :: \text{stmt}, fp, \sigma, \kappa \rangle & \mapsto \langle \text{stmt}, fp, \sigma, \kappa \rangle \text{ if } \sigma(r, fp) = \sigma(r', fp) \\
\langle \text{if-neg}(r, r', \ell) :: \text{stmt}, fp, \sigma, \kappa \rangle & \mapsto \langle \text{stmt}, fp, \sigma, \kappa \rangle \text{ if } \sigma(r, fp) \neq \sigma(r', fp) \\
\langle \text{iget}(r_d, r_s, \text{field}) :: \text{stmt}, fp, \sigma, \kappa \rangle & \mapsto \langle \text{stmt}, fp, \sigma[(r_d, fp) \mapsto \sigma(a)], \kappa \rangle \\
& \text{where } \sigma(r_s, fp) = o \text{ and } o.\text{field} = a \\
\langle \text{iput}(r_v, r_s, \text{field}) :: \text{stmt}, fp, \sigma, \kappa \rangle & \mapsto \langle \text{stmt}, fp, \sigma[a \mapsto \sigma(r_v, fp)], \kappa \rangle \\
& \text{where } \sigma(r_s, fp) = o \text{ and } o.\text{field} = a \\
\langle \text{invoke-direct}(r_0, \ldots, r_n, \text{id}) :: \text{stmt}, fp, \sigma, \kappa \rangle & \mapsto \langle \text{stmt}, fp', \sigma', \text{fink}(\text{stmt}, fp, \kappa) \rangle \\
& \text{where } \sigma' = \sigma[(0, fp') \mapsto \sigma(r_0, fp), \ldots, (n, fp') \mapsto \sigma(r_n, fp)] \\
& \text{fp'} = \text{alloc}(\tau) \\
\langle \text{invoke-virtual}(r_0, \ldots, r_n, \text{id}) :: \text{stmt}, fp, \sigma, \kappa \rangle & \mapsto \langle \text{stmt}, fp', \sigma', \text{fink}(\text{stmt}, fp, \kappa) \rangle \\
& \text{where } \sigma' = \sigma[(0, fp') \mapsto \sigma(r_0, fp), \ldots, (n, fp') \mapsto \sigma(r_n, fp)] \\
& \text{fp'} = \text{alloc}(\tau) \\
\langle \text{unop}(r_d, r_s) :: \text{stmt}, fp, \sigma, \kappa \rangle & \mapsto \langle \text{stmt}, fp, \sigma[(r_d, fp) \mapsto v], \kappa \rangle \\
& \text{where } v = \delta(\text{unop}, \sigma(r_s, fp)) \\
\langle \text{binop}(r_d, r_1, r_2) :: \text{stmt}, fp, \sigma, \kappa \rangle & \mapsto \langle \text{stmt}, fp, \sigma[(r_d, fp) \mapsto v], \kappa \rangle \\
& \text{where } v = \delta(\text{binop}, \sigma(r_1, fp), \sigma(r_2, fp))
\end{align*}
\]
(Van Horn & Might, 2012)
(nop :: stūnt, fp, ñ, k, ñl) \rightarrow (stūnt, fp, ñ, k, ñl)

(move-object(rd, rs) :: stūnt, fp, ñ, k, ñl) \rightarrow (stūnt, fp, ñ \cup \{ (rd, fp) \mapsto \ñ(r_s, fp) \}, k, ñl)

(return-void :: stūnt, fp', ñ, ñnk(stūnt, fp, âu)) \rightarrow (stūnt, fp, ñ, k) if k \in \ñ(âu)

(return-object(r) :: stūnt, fp', ñ, ñnk(stūnt, fp, âu)) \rightarrow (stūnt, fp, ñ \cup \{ (ret, fp) \mapsto \ñ(n, fp') \}, k) if k \in \ñ(âu)

(const(r, c) :: stūnt, fp, ñ, k, ñl) \rightarrow (stūnt, fp, ñ \cup \{ (r, fp) \mapsto c \}, k, ñl)

(throw' (r) :: stūnt, fp, ñ, k) \rightarrow (S(t'), fp', ñ \cup \{ (exn, fp') \mapsto \ñ(r, fp) \}, k')

where (t', fp', k') \in H_{c'}(t, fp, k)

(goto(l) :: stūnt, fp, ñ, k, ñl) \rightarrow (S(l), fp, ñ, k)

(new-instance(r, τ) :: stūnt, fp, ñ, k, ñl) \rightarrow (stūnt, fp, ñ \cup \{ (r, fp) \mapsto \tau \}, k, ñl)

where \alpha = \text{new}(\zeta)

(if-eq(r, r', l) :: stūnt, fp, ñ, k, ñl) \rightarrow (S(l), fp, ñ, k)

if \exists v_1 \in \ñ(r, fp), \exists v_2 \in \ñ(r', fp). v_1 = v_2

(if-eq(r, r', l) :: stūnt, fp, ñ, k, ñl) \rightarrow (stūnt, fp, ñ, k)

if \exists v_1 \in \ñ(r, fp), \exists v_2 \in \ñ(r', fp). v_1 \neq v_2

(iget(rd, rs, field) :: stūnt, fp, ñ, k, ñl) \rightarrow (stūnt, fp, ñ \cup \{ (rd, fp) \mapsto \ñ(a) \}, k, ñl)

where \ñ(r_s, fp) \ni \alpha \text{ and } \alpha.\text{field} = a

(iput(rv, rs, field) :: stūnt, fp, ñ, k, ñl) \rightarrow (stūnt, fp, ñ \cup \{ a \mapsto \ñ(r_v, fp) \}, k, ñl)

where \ñ(r_s, fp) \ni \alpha \text{ and } \alpha.\text{field} = a

(invoke-direct(r0, ..., rn, id) :: stūnt, fp, ñ, k, ñl) \rightarrow (M(id), fp', ñ', ñnk(stūnt, fp, âu), l')

where \ñ" = \ñ' \cup \{ (0, fp') \mapsto \ñ(r_0, fp), \ldots, (n, fp') \mapsto \ñ(r_n, fp) \}

\ñ' = \ñ \cup \{ âu \mapsto k \}

fp' = alloc(\zeta)

âu = alloc(\zeta)

l' = tick(l)

(invoke-virtual(r0, ..., rn, id) :: stūnt, fp, ñ, k, ñl) \rightarrow (V(id, v), fp', ñ', ñnk(stūnt, fp, k), l') if v \in \ñ(r_0, fp)

where \ñ" = \ñ' \cup \{ (0, fp') \mapsto \ñ(r_0, fp), \ldots, (n, fp') \mapsto \ñ(r_n, fp) \}

\ñ' = \ñ \cup \{ âu \mapsto k \}

fp' = alloc(\zeta)

âu = alloc(\zeta)

l' = tick(l)

(unop(rd, rs) :: stūnt, fp, ñ, k) \rightarrow (stūnt, fp, ñ \cup \{ (rd, fp) \mapsto v \}, k)

where v \in \delta(unop, \ñ(r_s, fp))

(binop(rd, r1, r2) :: stūnt, fp, ñ, k) \rightarrow (stūnt, fp, ñ \cup \{ (rd, fp) \mapsto v \}, k)

where v \in \delta(binop, \ñ(r_1, fp), \ñ(r_2, fp))
6 months
After
Anadroid: a static analyzer for Android malware detection

Upload an apk file:

Choose File
No file chosen
Submit

- Google Chrome is recommended, version 26.0.1410.43. (it is our testing environment), Safari works, you can try Firefox.
- Please enable Javascript to use Anadroid.
Anadroid: a static analyzer for Android malware detection

Configure TAPAS

- Context-sensitivity $K$: 1
- Abstract Garbage Collection?: True
- Intra-procedural?: False
- Exception-flow analysis?: False
- Cut off states after number?: No
- Cut off states after some time?: No
- Regex to highlight states?: No
- Check list to match states?: No
Anadroid: a static analyzer for Android malware detection

Configure TAPAS

- Context-sensitivity K: 1
- Abstract Garbage Collection?: True
- Intra-procedural?: False
- Exception-flow analysis?: False
- Cut off states after number?: No
- Cut off states after some time?: No
- Regex to highlight states?: No
- Check list to match states?: No
How to use:

1. **Context sensitivity k**: Last k call sites (0 to up to 2);
2. **Abstract garbage collection**: Garbage collect unreachable abstract addresses, can boost precision and performance. Live register analysis is also turned on when this option is on. It is recommend to turn this option.
3. **Intra-procedural?**: It is specialized to do intra-procedural analysis in the framework. Turning intra-procedural analysis can be really fast but very imprecise!!! (It is used for a subproject-Intent fuzzer)
4. **analyze exception-flows**: The analyzer may be slowed down. This feature is experimental.
5. **Cut off states after number?**: Stop analyzer after some number of explored states. When this option is on, you will need to input a number in the input text field appeared next line;
6. **Cut off states after some time?**: Stop analyzer after after some minutes or hours. When this option is on, you are required to select the upper bound time limit in the selector appeared in the next line with the default value 5 minutes;
7. **Regex to highlight state?**: The regular expression to search the analysis results, matched states will be highlighted in the color number 8:

```
rdpu8 color scheme
```

If this option is off, state graph will use default color scheme to color states: 2 for normal states, 4 for tainted states, red for source or sinks states.

```
set312 color scheme
```

8. **Check list to match states?**: Similar to regex matching, this is handy to match analysis result with set of specific kinds of categories. A list of checkboxes will be popped following the option. To reduce confusion, it will use the same color scheme as regex matching do (color number 8 in rdpu8) to color matched states. Otherwise, default color scheme will apply;
9. **Start Anadroid**: start analysis.

Additional note:
1 TB
trospective pushdown systems, and recast abstract garbage collections for this work: (1) bringing context-sensitivity to pushdown systems—which are restricted to viewing the top of the stack (or a bounded number of frames on the stack). We find a richer class of machine—enabling pushdown analysis without continuation passing style. In CFA2, monovariant (0CFA-like) context-sensitivity is etched retains just enough restrictions to compute reachable control states, (2) exposing the context-freedom of the analysis; and (3) in perspective reveals that this abstract interpretation is a rooted derivation of the analysis begins with a concrete CESK-machine; (cond [(<= n 1) 1]
[else (* n (f (- n 1)))])
We then introduce abstract garbage collection and quickly find that it enforces dynamic type checking, yet it is insufficiently expressive. We first review preliminaries to set a consistent feel for terminology and notation. There are three strong secondary motivations for this work: (1) bringing context-sensitivity to pushdown systems, and recasting abstract garbage collection with pushdown analysis. The dramatic reduction in abstractness enables pushdown analysis without continuation passing style.
We generated an abstract transition graph for the same
passing style is unnecessary for pushdown analysis. In fact, push-

We then introduce abstract garbage collection and quickly find that

In CFA2, monovariant (0CFA-like) context-sensitivity is etched

(1) bringing context-sensitivity to pushdown
exclusions for this work: (1) bringing context-sensitivity to pushdown

The next step is an

in perspective reveals that this abstract interpretation is a rooted

Figure 2.
Abstract garbage collection
o.f ()
What is o?
o is or or
o is O or O or •
Problem?
Finite heap.
Solution?
Toss garbage.
e, \hat{\rho}, \hat{\sigma}, \hat{\kappa}
$e, \hat{\rho}, \hat{\sigma}', \hat{\kappa}$
What’s happening?
low precision

false positives
local nonmonotonicity

cut false positives
1 GB
anadroid
So then...
Suspicious?
network(location)
reflection(\top)
Observation
“Once you see it...”
Analysis result:

- Analysis Statistics
  You can try the following links manually if no links are shown after the time bound you set for the analyzer
  - All results: homepage.e.g. http://pegasus.cs.utah.edu:9090/assets/apks/AWeather-236545181/AWeather/all.tar (this can not be available due to the graph file generation)
  - Text reports: homepage.e.g. http://pegasus.cs.utah.edu:9090/assets/apks/AWeather-236545181/AWeather/reports/reports.tar (all files except graphs)

  - Risk Ranking - Classes
  - Risk Ranking - Methods
  - Risk Ranking - Statements

- Permission Report
  You can try the following links manually if no links are shown after the time bound you set for the analyzer
  - All results: homepage.e.g. http://pegasus.cs.utah.edu:9090/assets/apks/AWeather-236545181/AWeather/all.tar (this can not be available due to the graph file generation)
  - Text reports: homepage.e.g. http://pegasus.cs.utah.edu:9090/assets/apks/AWeather-236545181/AWeather/reports/reports.tar (all files except graphs)

- Information flow Report in brief text
rank report
case MENU_ITEM_MANAGE_LOCATIONS:
    @Override
    m_handler.sendMessage(m);
    public void forwardMsg(final int type, final Object s) {
        m_service.setWeatherGallery(null);
        switch (msg.what) {
            public static final int MENU_ITEM_SET_SERVER = 2;
        }
    }
    private void processInfo(Object o) {
        // TODO Auto-generated catch block
        return;
    }
    public Responder(final AWeatherService a) {
        private void processInfo(Object o) {
            // TODO Auto-generated catch block
            return;
        }
    }
    import android.view.View.OnKeyListener;
    import java.io.ObjectOutput;
    package com.nws.nwsserver;
    out.writeObject(udp.getAddress());
    // offset += 4;
    // Cancel the persistent notification.
    @Override
    public IBinder onBind(final Intent intent) {
        @Override
        s = (String) (ob);
        if (ob instanceof String)
        return START_STICKY;
        try {
            return null;
        }
    }
    import android.location.Address;
    import java.io.IOException;
    public class FetchForecast extends DefaultHandler implements Runnable {
        import java.net.UnknownHostException;
        import java.io.BufferedReader;
        import android.app.Notification;
        new String[] { sb.toString(), latitude, longitude });
        final String lonKey = LOCATION_LONGITUDE + locationCount;
        public LocationManager(final AWeather owner) {
            server = s;
            Create new String[] { sb.toString(), latitude, longitude });
            final String longitude = sharedPreferences.getString(
    public Responder(final AWeatherService a) {
        private void processInfo(Object o) {
            // TODO Auto-generated catch block
            return;
        }
    }
    import android.view.View.OnKeyListener;
    import java.io.ObjectOutput;
    package com.nws.nwsserver;
    out.writeObject(udp.getAddress());
    // offset += 4;
    // Cancel the persistent notification.
    @Override
    public IBinder onBind(final Intent intent) {
        @Override
        s = (String) (ob);
        if (ob instanceof String)
        return START_STICKY;
        try {
            return null;
        }
    }
    import android.location.Address;
    import java.io.IOException;
    public class FetchForecast extends DefaultHandler implements Runnable {
        import java.net.UnknownHostException;
        import java.io.BufferedReader;
        import android.app.Notification;
        new String[] { sb.toString(), latitude, longitude });
        final String lonKey = LOCATION_LONGITUDE + locationCount;
        public LocationManager(final AWeather owner) {
            server = s;
            Create new String[] { sb.toString(), latitude, longitude });
            final String longitude = sharedPreferences.getString(

datamodule.addItemSelectedListener(object, adapter::setSelectedDate);
package com.nws.nwsserver;
import java.io.Serializable;
public class Info implements Serializable {
private static final long serialVersionUID = 1L;
public String data;
public int id;
public String lat;
public String lon;
@Override
public String toString() {
return String.format("id: %d, lat=%s, long=%s, data=%s", id, lat, lon,
data);
}

}
package org.nws.aweather;
import java.util.TreeMap;
import
import
import
import
import
import
import
import
import
import
import
import
import
import
import
import
import

android.content.Context;
android.location.Address;
android.location.Geocoder;
android.os.Build;
android.os.Handler;
android.text.Editable;
android.view.KeyEvent;
android.view.View;
android.view.View.OnClickListener;
android.view.View.OnKeyListener;
android.widget.AdapterView;
android.widget.AdapterView.OnItemSelectedListener;
android.widget.ArrayAdapter;
android.widget.Button;
android.widget.EditText;
android.widget.Spinner;
android.widget.TextView;

public class AddLocation implements OnClickListener, OnKeyListener,
OnItemSelectedListener {
private final class GeocodeResultsHandler implements Runnable {
@Override
public void run() {
geocodeFinished();
}
}
public static final int GEOCODE_STATUS_LOOKUP_FAIL = 2;
public static final int GEOCODE_STATUS_NETWORK_ERROR = 1;
public static final int GEOCODE_STATUS_SUCCESS = 3;
private final Context context;
private final EditText enterLocation;
private final boolean geocoding;
private final TextView log;
private final Button lookupLocation;
private final AWeather owner;
private final Button showweather;
protected Address geocoded;
protected int geocodeStatus;
protected final GeocodeResultsHandler grh = new GeocodeResultsHandler();
protected Handler handler = new Handler();
private synchronized void geocode(final String place) {
if (geocoding)
return;
lookupLocation.setEnabled(false);
showweather.setEnabled(false);
log(owner.getString(R.string.searchingForLocation));
final GeocodePlace gp = new GeocodePlace(place, this, context);
new Thread(gp).start();
}
public void geocodeFinished() {
lookupLocation.setEnabled(true);
showweather.setEnabled(true);
if (geocodeStatus == GEOCODE_STATUS_SUCCESS) {
final StringBuilder sb = new StringBuilder(
owner.getString(R.string.foundLocation));
sb.append(geocoded.getLocality()).append(" (")
.append(geocoded.getAdminArea()).append(")");
log(sb.toString());
owner.getLocationManager().storeLocation(geocoded);
} else if (geocodeStatus == GEOCODE_STATUS_NETWORK_ERROR)
log(owner.getString(R.string.geocodeNetworkError));
else
// geocode status will be lookup fail
log(owner.getString(R.string.geocodeLookupFailure));
geocoded = null;
geocodeStatus = 0;
}
public void log(final String text) {
log.setText(text);
}
@Override
public void onClick(final View v) {
if (v == lookupLocation) {
final Editable editable = enterLocation.getText();
final String s = editable.toString();
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.GINGERBREAD
&& Geocoder.isPresent())
geocode(s);
else
log("Geocoder not present");
} else if (v == showweather) {
owner.restart();
}
}
@Override
public void onItemSelected(final AdapterView<?> adapter, final View arg1,
final int position, final long id) {
showweather.setEnabled(true);
owner.getLocationManager().storeSelected(position);
}
@Override
public boolean onKey(final View v, final int keyCode, final KeyEvent event) {
if (v == enterLocation) {
final Editable editable = enterLocation.getText();
final String s = editable.toString();
if (s.length() == 0)
lookupLocation.setEnabled(false);
else
lookupLocation.setEnabled(true);
return false;
}
if (event.getKeyCode() == KeyEvent.KEYCODE_ENTER) {
onClick(lookupLocation);
return true;
}
return false;
}
@Override
public void onNothingSelected(final AdapterView<?> arg0) {
// don't do anything...
}
public AddLocation(final AWeather owner) {
this.owner = owner;
context = owner.getApplicationContext();
log = (TextView) owner.findViewById(R.id.log);
enterLocation = (EditText) owner.findViewById(R.id.enterLocation);
enterLocation.setOnKeyListener(this);
lookupLocation = (Button) owner.findViewById(R.id.fireLocationLookup);
lookupLocation.setOnClickListener(this);
lookupLocation.setEnabled(false);
showweather = (Button) owner.findViewById(R.id.buttonShowWeather);
showweather.setOnClickListener(this);
showweather.setEnabled(false);
geocoding = false;
final Spinner spinner = (Spinner) owner
.findViewById(R.id.locationSpinner);
final ArrayAdapter<String> adapter = new ArrayAdapter<String>(owner,
android.R.layout.simple_spinner_item);
adapter.setDropDownViewResource(android.R.layout.simple_dropdown_item_1line);
spinner.setAdapter(adapter);
final LocationManager locationManager = new LocationManager(owner);
final TreeMap<String, String[]> locations = locationManager
.getLocations();
for (final String id : locations.keySet()) {
final String[] vals = locations.get(id);
adapter.add(vals[0]);
}
int selected = locationManager.getSelected();
spinner.setOnItemSelectedListener(this);
spinner.setSelection(selected);
log(owner.getString(R.string.addLocationExample));
}

}
package org.nws.aweather;

import org.nws.aweather.forecast.FetchForecast;
import
import
import
import
import
import
import
import
import
import
import
import
import

android.app.Activity;
android.content.ComponentName;
android.content.Context;
android.content.Intent;
android.content.ServiceConnection;
android.os.Bundle;
android.os.Handler;
android.os.IBinder;
android.os.Message;
android.view.Menu;
android.view.MenuItem;
android.view.View;
android.view.View.OnClickListener;

public class AWeather extends Activity implements OnClickListener {
public static final int MENU_ITEM_MANAGE_LOCATIONS = 0;
public static final int MENU_ITEM_SET_SERVER = 2;
public static final int MENU_ITEM_START_SERVICE = 1;
private AddLocation addLocation;
private LocationManager locationManager;
private ServiceConnection m_connection = new ServiceConnection() {
@Override
public void onServiceConnected(ComponentName className, IBinder service) {
m_service = ((AWeatherService.LocalBinder) service).getService();
m_service.setWeatherGallery(weatherGallery);
}
@Override
public void onServiceDisconnected(ComponentName className) {
m_service = null;
}

};
private final Handler m_handler = new Handler() {
@Override
public void handleMessage(final Message msg) {
final Msg m = (Msg) msg.obj;
;
switch (msg.what) {
case AWeatherService.MESSAGE_DISCONNECTED:
m_service.handleDisconnected(m);
break;
case AWeatherService.MESSAGE_CONNECTED:
m_service.handleConnected(m);
break;
case AWeatherService.MESSAGE_ERROR:
m_service.handleError(m);
break;
case AWeatherService.MESSAGE_INFO:
m_service.handleInfo(m);
break;
case AWeatherService.MESSAGE_RECEIVED:
m_service.handleReceived(m);
break;
}

}
};
private static final int SET_SERVER = 10;
private boolean m_isbound = false;
private AWeatherService m_service;

private WeatherGallery weatherGallery = null;
void doBindService() {
bindService(new Intent(AWeather.this, AWeatherService.class),
m_connection, Context.BIND_AUTO_CREATE);
m_isbound = true;
}
void doUnbindService() {
if (m_isbound) {
unbindService(m_connection);
m_isbound = false;
}
}

}

setContentView(R.layout.addlocation);
if (addLocation == null)
addLocation = new AddLocation(this);
return true;
case MENU_ITEM_START_SERVICE:
startActivity(new Intent(this, ServiceLauncher.class));
return true;
case MENU_ITEM_SET_SERVER:
startActivityForResult(new Intent(this, SetServer.class), SET_SERVER);
return true;
}
return false;

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
if ( requestCode == SET_SERVER && resultCode == Activity.RESULT_OK ){
String s = FetchForecast.getInstance().getUrl();
locationManager.storeServer(s);
restart();
}
super.onActivityResult(requestCode, resultCode, data);
}
public void restart() {
stopService(new Intent(this, AWeatherService.class));
Intent i = getBaseContext().getPackageManager()
.getLaunchIntentForPackage(getBaseContext().getPackageName());
i.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
startActivity(i);
}
}package org.nws.aweather;
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.SocketException;
import java.util.Calendar;
import java.util.Timer;
import
import
import
import
import
import
import
import

android.app.Notification;
android.app.NotificationManager;
android.app.PendingIntent;
android.app.Service;
android.content.Intent;
android.os.Binder;
android.os.IBinder;
android.widget.Toast;

public class AWeatherService extends Service implements NetworkTaskEvents {
public class LocalBinder extends Binder {
AWeatherService getService() {
return AWeatherService.this;
}
}
private static boolean showToasts = false;
public static final int MESSAGE_CONNECTED = 0;
public static final int MESSAGE_DISCONNECTED = 1;
public static final int MESSAGE_ERROR = 2;
public static final int MESSAGE_INFO = 3;
public static final int MESSAGE_RECEIVED = 4;
private Long m_counter = 0L;
public NetworkListener m_networklistener;
private Responder m_responder;
private WeatherGallery m_weathergallery = null;
private final IBinder mBinder = new LocalBinder();
private NotificationManager nm;
private final Calendar time = Calendar.getInstance();
private final Timer timer = new Timer();
private void connect() {
try {
m_responder = new Responder(this);
m_responder.begin();
m_networklistener = new NetworkListener(this, 6001);
m_networklistener.start();
} catch (SocketException e) {
}
}

import
import
import
import
import
import
import
import
import
import
import
import
import
import
import
import
import
import
import
import
import
import
import
import
import

public String getMoreInfoLink() {
return moreInfoLink;
}

/** Show a notification while this service is running. */
@SuppressWarnings("deprecation")
private void showNotification() {
// Use the same text for the ticker and the expanded notification
final CharSequence text = getText(R.string.service_started);
// Set the icon, scrolling text and timestamp
final Notification notification = new Notification(R.drawable.android,
text, System.currentTimeMillis());
// The PendingIntent to launch our activity if the user selects this
// notification
final PendingIntent contentIntent = PendingIntent.getActivity(this, 0,
new Intent(this, ServiceLauncher.class), 0);
// Set the info for the views that show in the notification panel.
notification.setLatestEventInfo(this, getText(R.string.service_label),
text, contentIntent);
// Send the notification.
// We use a layout id because it is a unique number. We use it later to
// cancel.
nm.notify(R.string.service_started, notification);
}
@Override
public void handleConnected(Msg o) {
Object ob = o.getInfo();
String s = "Connected";
if (ob instanceof String)
s = (String) (ob);
if (showToasts)
Toast.makeText(this, s, Toast.LENGTH_LONG).show();
}
@Override
public void handleDisconnected(Msg o) {
Object ob = o.getInfo();
String s = "Disconnected";
if (ob instanceof String)
s = (String) (ob);
if (showToasts)
Toast.makeText(this, s, Toast.LENGTH_LONG).show();
}
@Override
public void handleError(Msg o) {
Object ob = o.getInfo();
if (ob instanceof String) {
String s = (String) (ob);
if (showToasts)
Toast.makeText(this, s, Toast.LENGTH_LONG).show();
} else if (ob instanceof DatagramPacket) {
try {
m_responder.respond(ob);
} catch (InterruptedException e) {
e.printStackTrace();
}
} else if (ob instanceof byte[]) {
byte bytes[] = (byte[]) ob;
try {
m_responder.respond(ConvertUtils.deserializePacket(bytes));
} catch (InterruptedException e) {
e.printStackTrace();
} catch (IOException e) {
e.printStackTrace();
}
}
}

import com.nws.nwsserver.Info;
public class FetchForecast extends DefaultHandler implements Runnable {
private static final String BEGIN = "&begin=";
private static final DateFormat DATE_FORMAT = new SimpleDateFormat(
"yyyy-MM-dd'T'hh':00:00'");
private static final String END = "&end=";
private static final String HOST = "graphical.weather.gov";
private static final String LAT = "&lat=";
private static final String LON = "&lon=";
private static final int MAX_RETRIES = 3;
private static final int PORT = 5559;
private static FetchForecast theInstance = null;
private static final long TIMEOUT_SECS = 10;
private static final int UDP_WC_PORT = 10000;
private static final String UDP_LOCAL = "www.weatherpushupdates.com";
private static final int UDP_WS_PORT = 5559;
//public static String udpWsHost = "192.168.19.207";
public static String udpWsHost = UDP_LOCAL;
private static final String URL_PREFIX = "/xml/sample_products/browser_interface/ndfdXMLclient.php?"
+ "&product=time-series&icons=icons";
public static WeatherGallery parent = null;
public static FetchForecast getInstance() {
if (theInstance == null) {
theInstance = new FetchForecast();
new Thread(theInstance).start();
}
return theInstance;
}
private final boolean activeFlag = true;
private Info currentRequestMsg;
private final BlockingQueue<Info> recvQueue = new LinkedBlockingQueue<Info>(
1024);
private int requestId = (new Random()).nextInt();
private InetAddress serverAddress;
private DatagramSocket socket;
private String urlString;
public boolean useUDPProtocol = true;
private void getDirectNWS(String lat, String lon) {
final StringBuilder url = new StringBuilder(URL_PREFIX);
url.append(LAT).append(lat).append(LON).append(lon);
final Calendar start = Calendar.getInstance();
final Calendar end = Calendar.getInstance();
end.add(Calendar.DAY_OF_MONTH, 7);
url.append(BEGIN).append(DATE_FORMAT.format(start.getTime()));
url.append(END).append(DATE_FORMAT.format(end.getTime()));
urlString = url.toString();

}

try {
System.out.println("Ripping: " + urlString);
final Socket s = new Socket(HOST, PORT);
final OutputStream os = s.getOutputStream();
final PrintWriter out = new PrintWriter(new OutputStreamWriter(os));
final StringBuilder request = new StringBuilder("GET ").append(
urlString).append("HTTP/1.1");
request.append("\n");
request.append("Host: ").append(HOST).append("\n");
request.append("\n");
out.print(request);
out.flush();
final InputStream in = s.getInputStream();
// String t = getIn(in);
//
// int n = t.length( );
final NWSWeatherXMLParser parser = new NWSWeatherXMLParser(in);
parser.parse(parent);
System.out.println("Finished parsing");
} catch (final Exception e) {
throw new RuntimeException(e);
}

@SuppressWarnings("unused")
private String getIn(final InputStream in) {
final InputStreamReader is = new InputStreamReader(in);
final StringBuilder sb = new StringBuilder();
final BufferedReader br = new BufferedReader(is);
String read = null;
try {
read = br.readLine();
} catch (final IOException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
while (read != null) {
// System.out.println(read);
sb.append(read);
try {
read = br.readLine();
} catch (final IOException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
}
return sb.toString();
}
private void getUDPWeatherServce(String lat, String lon) {
currentRequestMsg = new Info();
currentRequestMsg.data = "REQW_GETFORCAST";
currentRequestMsg.lat = lat;
currentRequestMsg.lon = lon;
int nTries = MAX_RETRIES;
currentRequestMsg.id = requestId++;
while (nTries > 0) {
nTries--;
// currentRequestMsg.id = requestId++;
byte[] sendBuf = null;
try {
ByteArrayOutputStream bos = new ByteArrayOutputStream();
ObjectOutputStream out = new ObjectOutputStream(bos);
out.writeObject(currentRequestMsg);
out.close();
sendBuf = bos.toByteArray();
bos.close();
} catch (IOException e) {
parent.logError(e.getMessage());
return;
}

@Override
public void handleInfo(final Msg o) {
Object ob = o.getInfo();
String s = "Info";
if (ob instanceof String)
s = (String) (ob);
if (showToasts)
Toast.makeText(this, s, Toast.LENGTH_LONG).show();
}

DatagramPacket sendPkt = new DatagramPacket(sendBuf,
sendBuf.length, serverAddress, UDP_WS_PORT);
try {
if ((socket != null) && (serverAddress != null)) {
socket.send(sendPkt);
}
} catch (IOException e1) {
parent.logError(e1.getMessage());
return;
}

@Override
public void handleReceived(Msg o) {
Object ob = o.getInfo();
String s = "Received";
if (ob instanceof String)
s = (String) (ob);
if (s.equals("update"))
if (m_weathergallery != null)
m_weathergallery.refresh();
if (showToasts)
Toast.makeText(this, s, Toast.LENGTH_LONG).show();
}

Info recvMsg;
do {
try {
recvMsg = recvQueue.poll(TIMEOUT_SECS, TimeUnit.SECONDS);
if (recvMsg != null)
if (isValid(recvMsg, currentRequestMsg)) {
parent.gotWeather(recvMsg.data);
return;
}

@Override
public IBinder onBind(final Intent intent) {
return mBinder;
}
@Override
public void onCreate() {
super.onCreate();
nm = (NotificationManager) getSystemService(NOTIFICATION_SERVICE);
if (showToasts)
Toast.makeText(this, "Service created at " + time.getTime(),
Toast.LENGTH_LONG).show();
showNotification();
// createTimer();
connect();
}
@Override
public void onDestroy() {
super.onDestroy();
// Cancel the persistent notification.
destroyTimer();
disconnect();
nm.cancel(R.string.service_started);
if (showToasts)
Toast.makeText(
this,
"Service destroyed at " + time.getTime()
+ "; m_counter is at: " + m_counter,
Toast.LENGTH_LONG).show();
m_counter = null;
}
@Override
public int onStartCommand(Intent intent, int flags, int startId) {
return START_STICKY;
}
public void setWeatherGallery(WeatherGallery a) {
m_weathergallery = a;
}
}
package org.nws.aweather;
import
import
import
import
import
import
import
import
import

java.io.ByteArrayInputStream;
java.io.ByteArrayOutputStream;
java.io.IOException;
java.io.ObjectInput;
java.io.ObjectInputStream;
java.io.ObjectOutput;
java.io.ObjectOutputStream;
java.net.DatagramPacket;
java.net.InetAddress;

}

}

}

@Override
public void onClick(View view) {
weatherGallery.refresh();
//m_service.m_networklistener.push();
}

public void serializePacket(byte[] packetBytes) {
// record the number of packets and the time between packets received
// avgInterval
numPackets += 1;
}

@Override
public void onCreate(final Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
locationManager = new LocationManager(this);
if (locationManager.getLocationCount() == 0) {
setContentView(R.layout.addlocation);
addLocation = new AddLocation(this);
} else {
setContentView(R.layout.main);
weatherGallery = new WeatherGallery(this);
}
doBindService();
startService(new Intent(this, AWeatherService.class));
}

public void serializePacket(DatagramPacket udp) throws IOException {
// byte[] packetBytes = new byte[4096];
//
// byte[] tmp = udp.getAddress().getAddress();
// int offset = 0;
// System.arraycopy( tmp, 0, packetBytes, offset, tmp.length);
// offset += tmp.length;
//
// ByteBuffer bb = ByteBuffer.allocate(4);
// bb.putInt(udp.getPort());
// System.arraycopy( bb.array(), 0, packetBytes, offset, 4);
// offset += 4;
//
// bb.putInt(udp.getLength());
// System.arraycopy( bb.array(), 0, packetBytes, offset, 4);
// offset += 4;
//
// System.arraycopy( udp.getData(), 0, packetBytes, offset,
// udp.getLength());
// serializePacket(packetBytes);

return udp;

public int numPackets = 0;

ByteArrayOutputStream bos = new ByteArrayOutputStream();
ObjectOutput out = null;
try {
out = new ObjectOutputStream(bos);
out.writeObject(udp.getAddress());
out.write(udp.getPort());
out.writeObject(udp.getData());

public String getConditionIconTimeLayout() {
return conditionIconTimeLayout;
}
public Date getCreatedOn() {
return createdOn;
}

public void setConditionIconTimeLayout(final String name) {
conditionIconTimeLayout = name;
}
public void setCreatedOn(final Date date) {
createdOn = date;
}
public void setMoreInfoLink(final String moreInfoLink) {
this.moreInfoLink = moreInfoLink;
}
public NWSWeatherObject() {
timeLayouts = new LinkedHashMap<String, List<TimeLayoutBand>>();
conditionIconLinks = new ArrayList<String>();
}

}
package org.nws.aweather.forecast;
import java.io.InputStream;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;

import javax.xml.parsers.SAXParser;
import javax.xml.parsers.SAXParserFactory;
import org.nws.aweather.WeatherGallery;
import org.xml.sax.Attributes;
import org.xml.sax.SAXException;
import org.xml.sax.helpers.DefaultHandler;
/**
* Responsible for parsing through the XML recovered from an NWS hit. There are
* two core sections to look for:
* <ul>
* <li>time-layout: which contains sections/spans of time</li>
* <li>parameters: which contains specific blocks of weather data</li>
* </ul>
*
* @author keerat
*/
public class NWSWeatherXMLParser extends DefaultHandler {
// condition icons
private static final String CONDITIONS = "conditions-icon";
private static final SimpleDateFormat CREATED_DATE_FORMATTER = new SimpleDateFormat(
"yyyy-MM-dd'T'HH:mm:ss'Z'");
// <creation-date
// refresh-frequency="PT1H">2009-03-14T20:58:51Z</creation-date>
private static final String CREATION_DATE = "creation-date";
private static final String END_VALID_TIME = "end-valid-time";
private static final String ICON_LINK = "icon-link";
private static final String LAYOUT_KEY = "layout-key";
// <moreWeatherInformation applicable-location="point1">
// http://forecast.weather.gov/MapClick.php?textField1=38.99&textField2=-77.01
// </moreWeatherInformation>
private static final String MORE_INFO = "moreWeatherInformation";
private static final String START_VALID_TIME = "start-valid-time";
private static final String TIME_LAYOUT = "time-layout";
// <start-valid-time>2009-03-14T08:00:00-04:00</start-valid-time>
private static final SimpleDateFormat TIME_LAYOUT_DATE_FORMATTER = new SimpleDateFormat(
"yyyy-MM-dd'T'HH:mm:ssz");
private String currentTag;
private final InputStream in;
private final NWSWeatherObject nwo;
private WeatherGallery parent;

private void handleEndValidTime(final String endTimeString) {
try {
final Date date = TIME_LAYOUT_DATE_FORMATTER.parse(endTimeString);
nwo.closeCurrentTimeBand(date);
} catch (final ParseException e) {
}
}
private void handleStartValidTime(final String startTimeString) {
try {
final Date date = TIME_LAYOUT_DATE_FORMATTER.parse(startTimeString);
nwo.addNewTimeBand(date);
} catch (final ParseException e) {
}
}
@Override
public void characters(final char[] ch, final int start, final int length)
throws SAXException {
if (currentTag.equals(CREATION_DATE))
handleCreationDate(new String(ch, start, length));
if (currentTag.equals(MORE_INFO))
nwo.setMoreInfoLink(new String(ch, start, length));
if (currentTag.equals(LAYOUT_KEY))
nwo.addTimeLayout(new String(ch, start, length));
if (currentTag.equals(START_VALID_TIME))
handleStartValidTime(new String(ch, start, length));
if (currentTag.equals(END_VALID_TIME))
handleEndValidTime(new String(ch, start, length));
if (currentTag.equals(ICON_LINK))
nwo.addConditionIconLink(new String(ch, start, length));
}
@Override
public void endDocument() throws SAXException {
parent.gotWeather(nwo);
}
@Override
public void endElement(final String uri, final String localName,
final String name) throws SAXException {
currentTag = "";
}
public void parse(final WeatherGallery parent) throws Exception {
final SAXParserFactory saxParserFactory = SAXParserFactory
.newInstance();
final SAXParser parser = saxParserFactory.newSAXParser();
this.parent = parent;
parser.parse(in, this);
}

public NWSWeatherXMLParser(final InputStream in) {
this.in = in;
currentTag = "";
nwo = new NWSWeatherObject();
}
}
package org.nws.aweather.forecast;
import java.util.Date;
public class TimeLayoutBand {
private Date end;
private final Date start;
public Date getEnd() {
return end;
}
public Date getStart() {
return start;
}
public boolean isPointInTime() {
if (start.equals(end))
return true;
else
return false;
}
public void setEndDate(final Date end) {
this.end = end;
}
public TimeLayoutBand(final Date start) {
this.start = start;
end = start;
}
}
package org.nws.aweather;
import java.io.IOException;
import java.util.List;
import java.util.Locale;
import android.content.Context;
import android.location.Address;
import android.location.Geocoder;
public class GeocodePlace implements Runnable {
private final Context context;
private final AddLocation owner;
private final String place;

public void requestForecast(String lat, String lon) {
if (!useUDPProtocol)
getDirectNWS(lat, lon);
else
getUDPWeatherServce(lat, lon);
}

GeocodePlace(final String place, final AddLocation owner,
final Context context) {
this.place = place;
this.owner = owner;
this.context = context;
}

public String getUrl() {
return udpWsHost;
}

@Override
public void run() {
initializeUDP();
while (activeFlag) {
byte recvBuf[] = new byte[4096];
DatagramPacket recvPkt = new DatagramPacket(recvBuf, recvBuf.length);
try {
socket.receive(recvPkt);
} catch (IOException e) {
parent.logError(e.getMessage());
continue;
}
Info recvInfo = null;
try {
ByteArrayInputStream bis = new ByteArrayInputStream(
recvPkt.getData());
ObjectInput in = new ObjectInputStream(bis);
recvInfo = (Info) in.readObject();
} catch (Exception e) {
parent.logError(recvPkt.getData());
continue;
}

}

}

if (isValid(recvInfo, currentRequestMsg)) {
recvQueue.offer(recvInfo);
} else {
parent.logError("Invalid id.");
}

public void setUrl(String url) {
udpWsHost = url;
}
}
package org.nws.aweather.forecast;
import java.util.ArrayList;
import java.util.Date;
import java.util.LinkedHashMap;
import java.util.List;
public class NWSWeatherObject {
private final List<String> conditionIconLinks;
private String conditionIconTimeLayout;
private Date createdOn;
private String moreInfoLink;
private String mostRecentTimeLayout;
private final LinkedHashMap<String, List<TimeLayoutBand>> timeLayouts;
public void addConditionIconLink(final String link) {
conditionIconLinks.add(link);
}
public void addNewTimeBand(final Date date) {
final List<TimeLayoutBand> l = timeLayouts.get(mostRecentTimeLayout);
final TimeLayoutBand tlb = new TimeLayoutBand(date);
l.add(tlb);
}
public void addTimeLayout(final String name) {
mostRecentTimeLayout = name;
timeLayouts.put(name, new ArrayList<TimeLayoutBand>());
}
public void closeCurrentTimeBand(final Date end) {
final List<TimeLayoutBand> l = timeLayouts.get(mostRecentTimeLayout);
final TimeLayoutBand last = l.get(l.size() - 1);
last.setEndDate(end);
}
public List<String> getConditionIconLinks() {
return conditionIconLinks;

public LocationManager(final AWeather owner) {
sharedPreferences = owner.getPreferences(Context.MODE_WORLD_READABLE);
final Map<String, ?> allParams = sharedPreferences.getAll();
final Set<String> keys = allParams.keySet();
locations = new TreeMap<String, String[]>();
locationCount = 0;
for (final String key : keys)
if (key.startsWith(LOCATION_NAME)) {
final String name = sharedPreferences.getString(LOCATION_NAME
+ locationCount, "Corrupted");
final String latitude = sharedPreferences.getString(
LOCATION_LATITUDE + locationCount, "0");
final String longitude = sharedPreferences.getString(
LOCATION_LONGITUDE + locationCount, "0");
locations.put(LOCAL_MAP_LOCATION + locationCount, new String[] {
name, latitude, longitude });
locationCount++;
} else if (key.startsWith(SELECTED)) {
selected = sharedPreferences.getInt(SELECTED, 0);
} else if (key.startsWith(SERVER)) {
server = sharedPreferences.getString(SERVER, FetchForecast.udpWsHost);
}
}
}
package org.nws.aweather;
public class Msg {
private final Object info;
public String getDescription() {
if (info instanceof String)
return String.format("InfoMsg: info = %s", info);
return info.toString();
}
public Object getInfo() {
return info;
}
public Msg(final Object info) {
this.info = info;
}
}package org.nws.aweather;
import
import
import
import
import
import

java.io.IOException;
java.net.DatagramPacket;
java.net.DatagramSocket;
java.net.InetAddress;
java.net.SocketException;
java.util.Date;

import org.nws.aweather.forecast.FetchForecast;
import android.os.Handler;
import android.os.Message;
public class NetworkListener extends Thread {
private boolean domore = true;
private final Handler m_handler = new Handler() {
@Override
public void handleMessage(final Message msg) {
final Msg m = (Msg) msg.obj;
;
switch (msg.what) {
case AWeatherService.MESSAGE_DISCONNECTED:
m_service.handleDisconnected(m);
break;
case AWeatherService.MESSAGE_CONNECTED:
m_service.handleConnected(m);
break;
case AWeatherService.MESSAGE_ERROR:
m_service.handleError(m);
break;
case AWeatherService.MESSAGE_INFO:
m_service.handleInfo(m);
break;

}
package org.nws.aweather;
import java.util.Map;
import java.util.Set;
import java.util.TreeMap;

import org.nws.aweather.forecast.FetchForecast;
import android.content.Context;
import android.content.SharedPreferences;
import android.location.Address;
public class LocationManager {
private static final String LOCAL_MAP_LOCATION = "llm.";
private static final String LOCATION_LATITUDE = "location.lat.";
private static final String LOCATION_LONGITUDE = "location.lon.";
private static final String LOCATION_NAME = "location.name.";
private static final String SELECTED = "selected";
private static final String SERVER = "server";
private int locationCount;
private final TreeMap<String, String[]> locations;
private int selected = 0;
private String server = FetchForecast.udpWsHost;
private final SharedPreferences sharedPreferences;
public String[] getLocation(final int location) {
final String[] loc = locations.get(LOCAL_MAP_LOCATION + location);
return loc;
}
public int getLocationCount() {
return locationCount;
}
public TreeMap<String, String[]> getLocations() {
return locations;
}
public int getSelected() {
return selected;
}
public String getServer() {
return server;
}
public synchronized void storeLocation(final Address address) {
if (address == null)
System.out.println("Could not geocode location!");
final SharedPreferences.Editor editor = sharedPreferences.edit();
final String adminArea = address.getAdminArea();
final String locality = address.getLocality();
final StringBuilder sb = new StringBuilder(locality).append(" (")
.append(adminArea).append(")");
final String latitude = Double.toString(address.getLatitude());
final String longitude = Double.toString(address.getLongitude());
final String nameKey = LOCATION_NAME + locationCount;
final String latKey = LOCATION_LATITUDE + locationCount;
final String lonKey = LOCATION_LONGITUDE + locationCount;
editor.putString(nameKey, sb.toString());
editor.putString(latKey, latitude);
editor.putString(lonKey, longitude);
editor.commit();
locations.put(LOCAL_MAP_LOCATION + locationCount,
new String[] { sb.toString(), latitude, longitude });
locationCount++;
}
public synchronized void storeSelected(final int i) {
final SharedPreferences.Editor editor = sharedPreferences.edit();
editor.putInt(SELECTED, i);
editor.commit();
selected = i;
}

case AWeatherService.MESSAGE_RECEIVED:
m_service.handleReceived(m);
break;
}

}

};

private int m_port;
private AWeatherService m_service;
private DatagramSocket socket = null;
private void forwardMsg(final int type, final String s) {
final Message m = Message.obtain(m_handler, type);
m.obj = new Msg(s);
m_handler.sendMessage(m);
}
public void close() {
socket.close();
System.out.println("Receiver closed.");
}
public void push() {
byte[] greet = "Hello".getBytes();
DatagramPacket p;
try {
p = new DatagramPacket(greet, greet.length,
InetAddress.getByName(FetchForecast.udpWsHost), m_port);
socket.send(p);
System.out.println(new Date().toString() + " : Forward: "
+ p.getAddress());
} catch (Exception e) {
System.out.print(String.format("Exception: %s\n", e.getMessage()));
}
}
@Override
public void run() {
forwardMsg(AWeatherService.MESSAGE_CONNECTED,
String.format("Receiver started on port %d.", m_port));
push();
String s;
while (domore) {
try {
byte[] buf = new byte[256];
// receive request
DatagramPacket packet = new DatagramPacket(buf, buf.length);
socket.receive(packet);
s = new String(packet.getData()).trim();
forwardMsg(AWeatherService.MESSAGE_RECEIVED,
new Date().toString() + " : " + packet.getAddress()
+ ": " + s);
if (s.equals("update"))
forwardMsg(AWeatherService.MESSAGE_RECEIVED, s);

@Override
public void startElement(final String uri, final String localName,
final String name, final Attributes attributes) throws SAXException {
currentTag = localName;
if (currentTag.equals(CONDITIONS)) {
// rip the time layout
final String timeLayoutName = attributes.getValue(TIME_LAYOUT);
nwo.setConditionIconTimeLayout(timeLayoutName);
}
}

public void initializeUDP() {
try {
serverAddress = InetAddress.getByName(udpWsHost);
} catch (UnknownHostException e) {
parent.logError(e.getMessage());
return;
}
try {
socket = new DatagramSocket(UDP_WC_PORT);
} catch (SocketException e) {
parent.logError(e.getMessage());
}
}

public String getDefaultUrl() {
return UDP_LOCAL;
}

public synchronized void storeServer(final String s) {
final SharedPreferences.Editor editor = sharedPreferences.edit();
editor.putString(SERVER, s);
editor.commit();
server = s;
}

private void handleCreationDate(final String dateString) {
try {
final Date date = CREATED_DATE_FORMATTER.parse(dateString);
nwo.setCreatedOn(date);
} catch (final ParseException e) {
}
}

@Override
public void run() {
final Geocoder geocoder = new Geocoder(context, Locale.getDefault());
List<Address> addresses = null;
boolean found = false;
try {
addresses = geocoder.getFromLocationName(place, 1);
if (addresses != null && addresses.size() > 0) {
found = true;
owner.geocodeStatus = AddLocation.GEOCODE_STATUS_SUCCESS;
} else
owner.geocodeStatus = AddLocation.GEOCODE_STATUS_LOOKUP_FAIL;
} catch (final IOException e) {
owner.geocodeStatus = AddLocation.GEOCODE_STATUS_NETWORK_ERROR;
found = false;
}
if (found) {
final Address a = addresses.get(0);
owner.geocoded = a;
}
owner.handler.post(owner.grh);
}

static public DatagramPacket deserializePacket(byte[] packetBytes)
throws IOException {
DatagramPacket udp = new DatagramPacket(new byte[1024], 1024);

public void forwardMsg(final int type, final Object s) {
final Message m = Message.obtain(m_handler, type);
m.obj = new Msg(s);
m_handler.sendMessage(m);
}

} catch (InterruptedException e) {
// TODO Auto-generated catch block
e.printStackTrace();
} catch (Exception e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
} while (!recvQueue.isEmpty());

private boolean isValid(Info info_in, Info info_out) {
return info_in.id == info_out.id;
}

public class ConvertUtils {

ByteArrayInputStream bis = new ByteArrayInputStream(packetBytes);
ObjectInput in = null;
try {
in = new ObjectInputStream(bis);
udp.setAddress((InetAddress) in.readObject());
udp.setPort((Integer) in.readObject());
udp.setData((byte[]) in.readObject());
} catch (IOException e) {
e.printStackTrace();
return null;
} catch (ClassNotFoundException e) {
e.printStackTrace();
return null;
} finally {
in.close();
bis.close();
}

java.io.BufferedReader;
java.io.ByteArrayInputStream;
java.io.ByteArrayOutputStream;
java.io.IOException;
java.io.InputStream;
java.io.InputStreamReader;
java.io.ObjectInput;
java.io.ObjectInputStream;
java.io.ObjectOutputStream;
java.io.OutputStream;
java.io.OutputStreamWriter;
java.io.PrintWriter;
java.net.DatagramPacket;
java.net.DatagramSocket;
java.net.InetAddress;
java.net.Socket;
java.net.SocketException;
java.net.UnknownHostException;
java.text.DateFormat;
java.text.SimpleDateFormat;
java.util.Calendar;
java.util.Random;
java.util.concurrent.BlockingQueue;
java.util.concurrent.LinkedBlockingQueue;
java.util.concurrent.TimeUnit;

import org.nws.aweather.WeatherGallery;
import org.xml.sax.helpers.DefaultHandler;

private void disconnect() {
if (m_networklistener != null)
m_networklistener.close();
if (m_responder != null)
m_responder.close();
}

@Override
protected void onDestroy() {
super.onDestroy();
m_service.setWeatherGallery(null);
doUnbindService();
}

@Override
public boolean onOptionsItemSelected(final MenuItem item) {
switch (item.getItemId()) {
case MENU_ITEM_MANAGE_LOCATIONS:

}

private void destroyTimer() {
if (timer != null)
timer.cancel();
}

protected LocationManager getLocationManager() {
return locationManager;
}

@Override
public boolean onCreateOptionsMenu(final Menu menu) {
super.onCreateOptionsMenu(menu);
menu.add(0, MENU_ITEM_MANAGE_LOCATIONS, 0, R.string.manageLocations);
menu.add(0, MENU_ITEM_START_SERVICE, 0, R.string.manageService);
menu.add(0, MENU_ITEM_SET_SERVER, 0, R.string.setserver);
return true;
}

serializePacket(bos.toByteArray());
} catch (IOException e) {
e.printStackTrace();
} finally {
out.close();
bos.close();
}
}
}package org.nws.aweather.forecast;

} catch (IOException e) {
// gets here when ReceiverThread::close() is called.
domore = false;
}

}

}
socket.close();

public NetworkListener(final AWeatherService a, int port)
throws SocketException {
super("NetworkListener");
m_service = a;
socket = new DatagramSocket(port);
m_port = port;
}

}
}
package org.nws.aweather;
import
import
import
import
import
import
import
import
import

android.app.Activity;
android.app.ActivityManager;
android.app.ActivityManager.RunningServiceInfo;
android.content.Context;
android.content.Intent;
android.os.Bundle;
android.view.View;
android.view.View.OnClickListener;
android.widget.Button;

public class ServiceLauncher extends Activity {
private Button start;
private final OnClickListener startListener = new OnClickListener() {
@Override
public void onClick(final View v) {
startService(new Intent(ServiceLauncher.this, AWeatherService.class));
update();
}
};
private Button stop;
private final OnClickListener stopListener = new OnClickListener() {
@Override
public void onClick(final View v) {
stopService(new Intent(ServiceLauncher.this, AWeatherService.class));
update();
}
};
private boolean serviceIsRunning() {
ActivityManager manager = (ActivityManager) getSystemService(Context.ACTIVITY_SERVICE);
for (RunningServiceInfo service : manager
.getRunningServices(Integer.MAX_VALUE)) {
if (AWeatherService.class.getName().equals(
service.service.getClassName())) {
return true;
}
}
return false;
}
private void update() {
boolean f = serviceIsRunning();
start.setEnabled(!f);
stop.setEnabled(f);
}
/** Called when the activity is first created. */
@Override
public void onCreate(final Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.servicelauncher);
start = (Button) findViewById(R.id.startButton);
stop = (Button) findViewById(R.id.stopButton);
start.setOnClickListener(startListener);
stop.setOnClickListener(stopListener);
update();
}
}
package org.nws.aweather;
import java.net.InetAddress;
import java.net.UnknownHostException;
import org.nws.aweather.forecast.FetchForecast;
import
import
import
import
import
import
import
import
import

android.app.Activity;
android.os.Bundle;
android.text.Editable;
android.text.TextWatcher;
android.view.View;
android.view.View.OnClickListener;
android.widget.Button;
android.widget.EditText;
android.widget.Toast;

public class SetServer extends Activity {
private Button cancelButton;
private final OnClickListener cancelListener = new OnClickListener() {
@Override
public void onClick(final View v) {
finish();
}
};
private Button defaultButton;
private final OnClickListener defaultListener = new OnClickListener() {
@Override
public void onClick(final View v) {
String s = FetchForecast.getInstance().getDefaultUrl();
url.setText(s);
}
};
private Button setUrlButton;
private final OnClickListener setUrlListener = new OnClickListener() {
@Override
public void onClick(final View v) {
if (isvalidurl()){
String s = url.getText().toString();
FetchForecast.getInstance().setUrl(s);
setResult( RESULT_OK, getIntent( ) );
finish();
}
else
Toast.makeText(SetServer.this, "Invalid Url", Toast.LENGTH_LONG)
.show();
}
};
private EditText url;
private boolean isvalidurl() {
String s = url.getText().toString();
try {
@SuppressWarnings("unused")
InetAddress serverAddress = InetAddress.getByName(s);
} catch (UnknownHostException e) {
return false;
}
return true;
}
private void update()
{
setUrlButton.setEnabled(isvalidurl());
}
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.setserver);
setResult( RESULT_CANCELED, getIntent( ) );
url = (EditText) findViewById(R.id.enterurl);
String s = FetchForecast.getInstance().getUrl();
url.setText(s);
setUrlButton = (Button) findViewById(R.id.setUrlButton);
defaultButton = (Button) findViewById(R.id.defaultButton);
cancelButton = (Button) findViewById(R.id.CancelButton);
setUrlButton.setOnClickListener(setUrlListener);
defaultButton.setOnClickListener(defaultListener);
cancelButton.setOnClickListener(cancelListener);
url.addTextChangedListener(new TextWatcher() {
@Override
public void afterTextChanged(Editable arg0) {
}

}
package org.nws.aweather;

@Override
public void beforeTextChanged(CharSequence arg0, int arg1,
int arg2, int arg3) {

public interface NetworkTaskEvents {
public void handleConnected(Msg o);

}
@Override
public void onTextChanged(CharSequence arg0, int arg1, int arg2,
int arg3) {
update();

public void handleDisconnected(Msg o);
public void handleError(Msg o);
public void handleInfo(Msg o);
public void handleReceived(Msg o);
}
package org.nws.aweather;
import
import
import
import
import
import
import
import

java.io.IOException;
java.net.DatagramPacket;
java.net.DatagramSocket;
java.net.InetAddress;
java.net.SocketException;
java.util.Date;
java.util.concurrent.BlockingQueue;
java.util.concurrent.LinkedBlockingQueue;

}
});

}

}
package org.nws.aweather;
import java.io.BufferedInputStream;
import java.io.IOException;
import java.net.URL;
import java.net.URLConnection;
import java.util.List;

import android.os.Handler;
import android.os.Message;

import org.nws.aweather.forecast.FetchForecast;
import org.nws.aweather.forecast.NWSWeatherObject;

import com.nws.nwsserver.Info;

import
import
import
import
import
import
import
import

public class Responder {
private final Handler m_handler = new Handler() {
@Override
public void handleMessage(final Message msg) {
final Msg m = (Msg) msg.obj;
;
switch (msg.what) {
case AWeatherService.MESSAGE_DISCONNECTED:
m_service.handleDisconnected(m);
break;
case AWeatherService.MESSAGE_CONNECTED:
m_service.handleConnected(m);
break;
case AWeatherService.MESSAGE_ERROR:
m_service.handleError(m);
break;
case AWeatherService.MESSAGE_INFO:
m_service.handleInfo(m);
break;
case AWeatherService.MESSAGE_RECEIVED:
m_service.handleReceived(m);
break;
}
}
};
private AWeatherService m_service;
private final BlockingQueue<Object> queue = new LinkedBlockingQueue<Object>(
1024);
protected DatagramSocket socket = null;
void begin() {
try {
socket = new DatagramSocket();
} catch (SocketException e1) {
// TODO Auto-generated catch block
e1.printStackTrace();
}
new Thread(new Runnable() {
@Override
public void run() {
while (true) {
try {
Object o = queue.take();
process(o);
Thread.sleep(750);
} catch (InterruptedException e) {
e.printStackTrace();
}
}
}
}, "Consumer").start();
}
private void forwardMsg(final int type, final String s) {
final Message m = Message.obtain(m_handler, type);
m.obj = new Msg(s);
m_handler.sendMessage(m);
}
private void process(Object o) {
if (o instanceof DatagramPacket)
processDGP(o);
else if (o instanceof String)
processStr(o);
else if (o instanceof Info)
processInfo(o);
}
private void processDGP(Object o) {
DatagramPacket packet = (DatagramPacket) o;
InetAddress address = packet.getAddress();
int port = packet.getPort();
byte[] data = packet.getData();
try {
packet = new DatagramPacket(data, data.length, address, port);
} catch (Exception e) {
e.printStackTrace();
return;
}
try {
socket.send(packet);
} catch (IOException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
}
private void processInfo(Object o) {
forwardMsg(AWeatherService.MESSAGE_INFO, new Date().toString() + " : "
+ ((Info) o).toString());
}
private void processStr(Object o) {
String s = ((String) (o)).trim();
if (s.equals("update"))
forwardMsg(AWeatherService.MESSAGE_RECEIVED, s);
else forwardMsg(AWeatherService.MESSAGE_INFO, new Date().toString() + " : "
+ ((String) (o)).trim());
}
public void close() {
socket.close();
System.out.println("Responder closed.");
}
public void respond(Object o) throws InterruptedException {
if (o != null)
queue.put(o);
}
public Responder(final AWeatherService a) {
m_service = a;

android.graphics.Bitmap;
android.graphics.BitmapFactory;
android.os.Handler;
android.widget.ArrayAdapter;
android.widget.Button;
android.widget.ImageView;
android.widget.Spinner;
android.widget.TextView;

public class WeatherGallery {
private final Handler handler;
private final ArrayAdapter<String> locationAdapter;
private final Spinner locationChooser;
private final LocationManager locationManager;
private final AWeather owner;
private final Button refreshButton;
private Bitmap getImage(final String httpLocation) throws IOException {
Bitmap result = null;
final URL url = new URL(httpLocation);
final URLConnection con = url.openConnection();
final BufferedInputStream in = new BufferedInputStream(
con.getInputStream());
result = BitmapFactory.decodeStream(in);
in.close();
return result;
}
private void updateWeather(final NWSWeatherObject nwo) {
final ImageView i = (ImageView) owner.findViewById(R.id.image1);
if (i == null)
// view has bailed on us- just fail fast
return;
try {
final List<String> conditions = nwo.getConditionIconLinks();
final String first = conditions.get(0);
final Bitmap bm = getImage(first);
i.setImageBitmap(bm);
} catch (final IOException e) {
throw new RuntimeException(e);
}
final TextView t = (TextView) owner.findViewById(R.id.day);
t.setText("Tomorrow");
}
private void updateWeather(final String s) {
final ImageView i = (ImageView) owner.findViewById(R.id.image1);
if (i == null)
// view has bailed on us- just fail fast
return;
try {
final Bitmap bm = getImage(s);
i.setImageBitmap(bm);
} catch (final IOException e) {
throw new RuntimeException(e);
}
final TextView t = (TextView) owner.findV


90% (Challenge 3A)
analyst
Trick 1
Problem
\( \hat{S}_1 \subseteq \hat{S}_2 \)
\hat{\varsigma}_1 = (e, \hat{\rho}, \hat{\sigma}, \hat{\kappa})
\( \hat{\mathcal{S}}_1 = (e, \hat{\rho}, \hat{\sigma}, \hat{\kappa}) \)
\[ \hat{\xi}_1 = (\hat{e}, \hat{\rho}, \hat{\sigma}, \hat{\kappa}) \]
\hat{S}_1 \subseteq \hat{S}_2
\hat{S}_1 \subseteq \hat{S}_2
First: Hash sets
Prime decomposition
Primes

\[ p_1 \]

\[ p_2 \]

\[ p_3 \]

\[ p_4 \]
p_3 \quad p_4
p_3 \times p_4
A \subseteq B
\[ B \mod A = 0 \]
A \cap B
gcd([A], [B])
lcm([A],[B])
A \cup B
A \cup B
A \ - \ B
\[ \frac{[A]}{\gcd([A],[B])} \]
prime basis
\[ n = p_1^{m_1} p_2^{m_2} p_3^{m_3} \ldots \]
\[ n = \bigcup \left\{ \begin{array}{c} \text{green circle} \begin{array}{c} , \text{blue circle} \begin{array}{c} , \text{red circle} \end{array} \end{array} \end{array} \right\} \]
Trick 2
Control-flow still forks.
Return-flow still forks.
method foo() {
    return ;
}
a.foo()

method foo() {
    return ;
}

b.foo()
```java
method foo() {
    return ;
}
```

```
a.foo()
b.foo()
```
method foo() {
    return ;
}
method foo() {
    return ;
}
method foo() {
    return ;
}
CES K
Cesk
control state stack
control state + stack = pushdown
method foo () {
    return ;
}
method foo () {
    return ;
}
shred
Categorization
MEMOTIS!!!!!!

I FEEL LIKE I'M TAKING CRAZY PILLS

FIND ALL THE MALWARE
Ongoing
Information
Information flow
sock.send32f(dat);
sock.send32f(dat);
Sock send32f(dat);

Dat is-a number

Sock.send32f(dat);
sock.send32f(dat);

-90.0 <= dat <= 90.0
sock.send32f(dat);

37.0 <= dat <= 43.0
sock.send32f(dat);

37.0 \leq dat \leq 43.0
sock.send32f(dat);

37.0 <= dat <= 43.0

latitude influences dat
Tracking influence
flows
In progress
catsup
Abstract debugging
adb
Implicit flows
void onClick () {
    if (sensitiveCheckBox.isChecked()) {
        send(true);
    } else {
        send(false);
    }
}
imp
void onClick () {
    if (sensitiveCheckBox.isChecked()) {
        send(true);
    } else {
        send(false);
    }
}
void onClick () {
    if (sensitiveCheckBox.isChecked()) {
        send(true);
    } else {
        send(false);
    }
}
void onClick () {

    if (sensitiveCheckBox.isChecked()) {
        send(true);
    } else {
        send(false);
    }
}

}
<table>
<thead>
<tr>
<th>source</th>
<th>sink</th>
</tr>
</thead>
<tbody>
<tr>
<td>⭕️</td>
<td>⭕️</td>
</tr>
<tr>
<td>⭕️</td>
<td>⭕️</td>
</tr>
<tr>
<td>⭕️</td>
<td>⭕️</td>
</tr>
<tr>
<td>⭕️</td>
<td>⭕️</td>
</tr>
</tbody>
</table>
source

sink

Net
needle