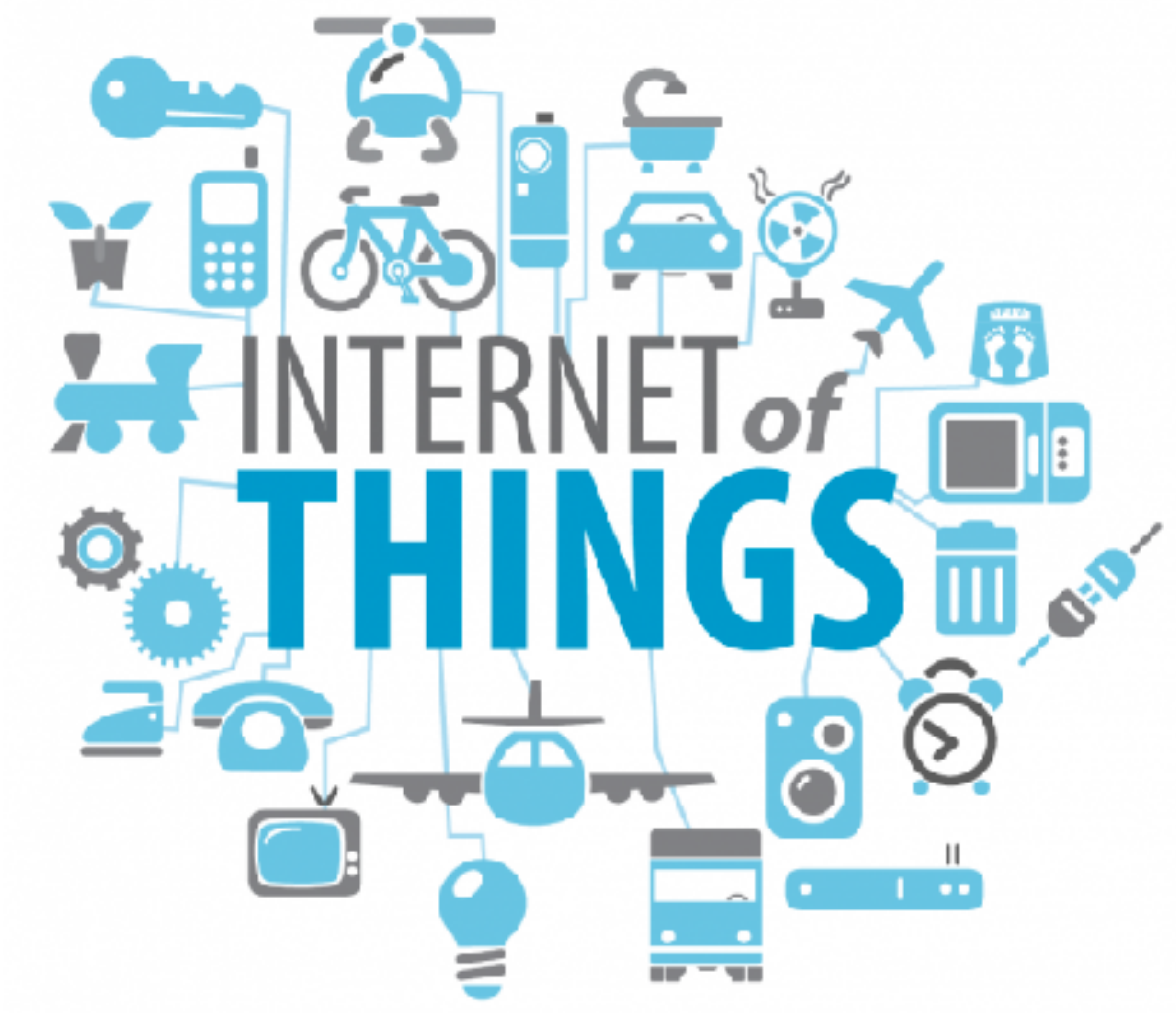
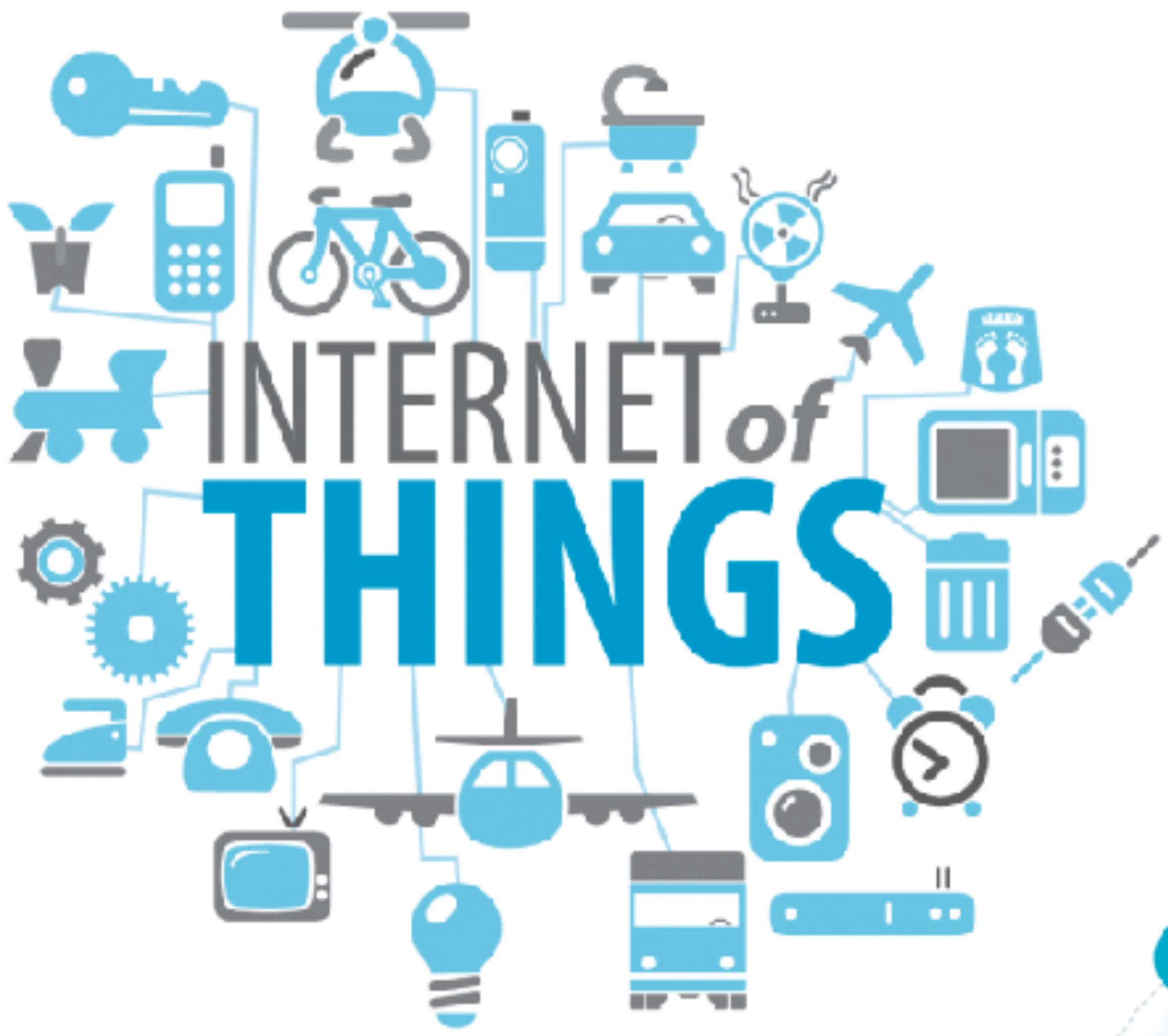
Applying the Opacified Computation Model to Enforce Information Flow Policies in IoT Applications

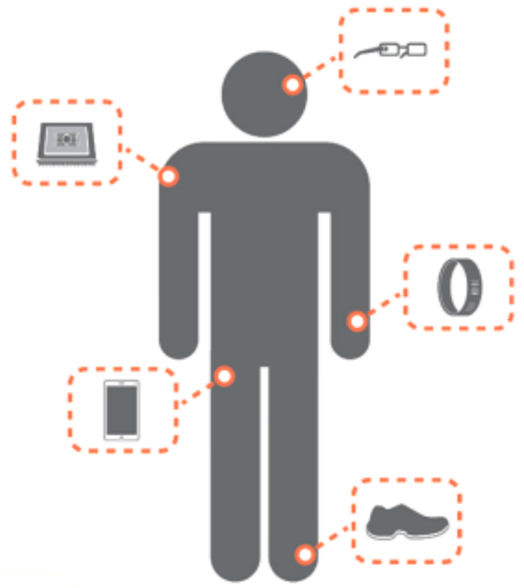
Amir Rahmati, Earlence Fernandes, Atul Prakash

Applying the Opacified Computation Model to Enforce Information Flow Policies in IoT Applications

Amir Rahmati, Earlence Fernandes, Atul Prakash


INTERNET of
THINGS

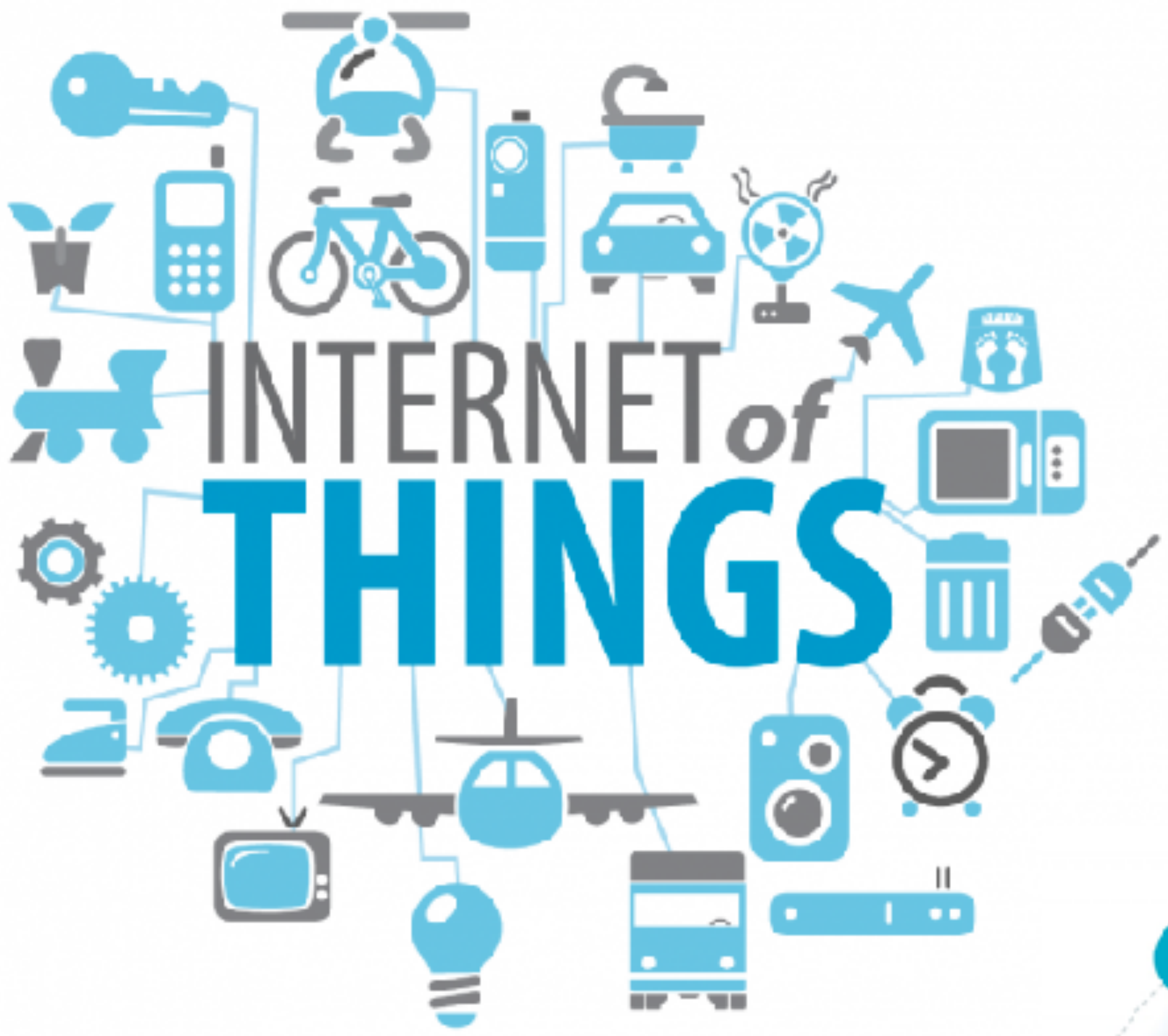


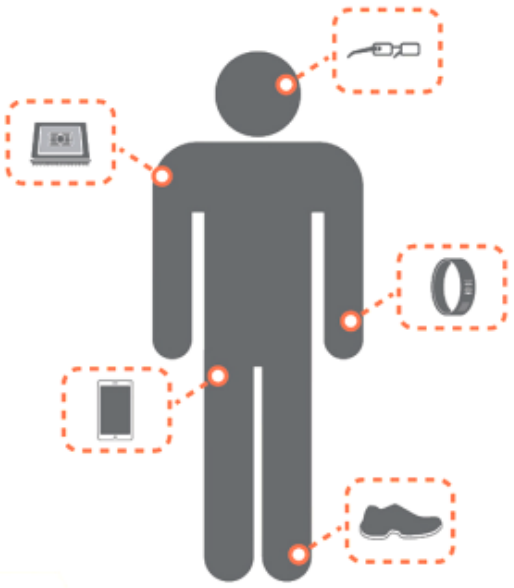


Wearables

Smart Home

Connected Health





Wearables

Smart Home

Connected Health

Frameworks

android wear

WEAVE On WEAVE

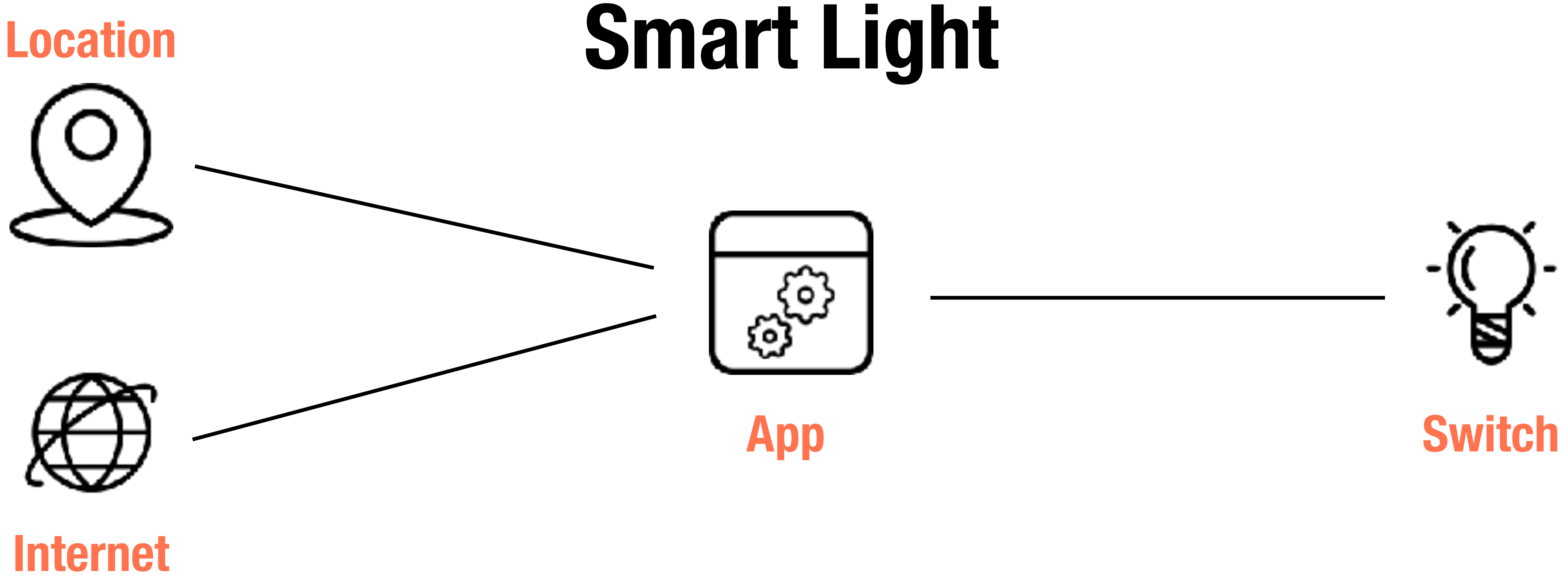
Google Brillo inside

SAMSUNG SmartThings

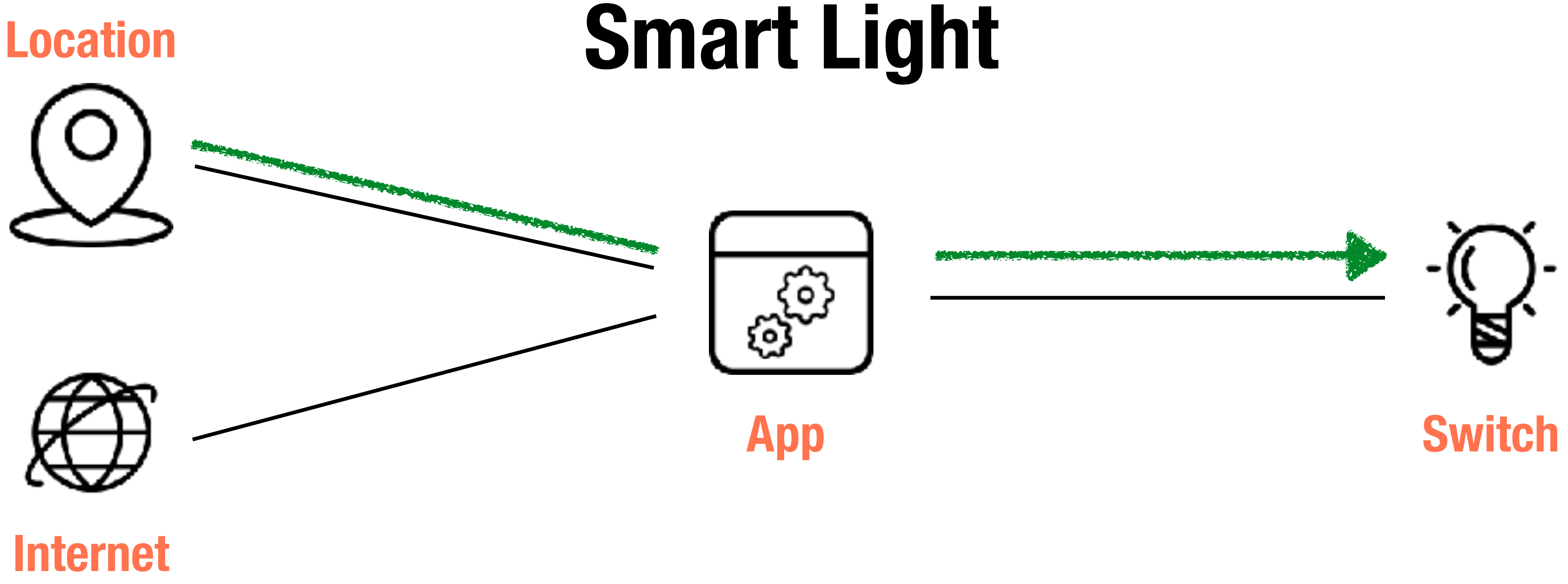
HomeKit

IoTivity

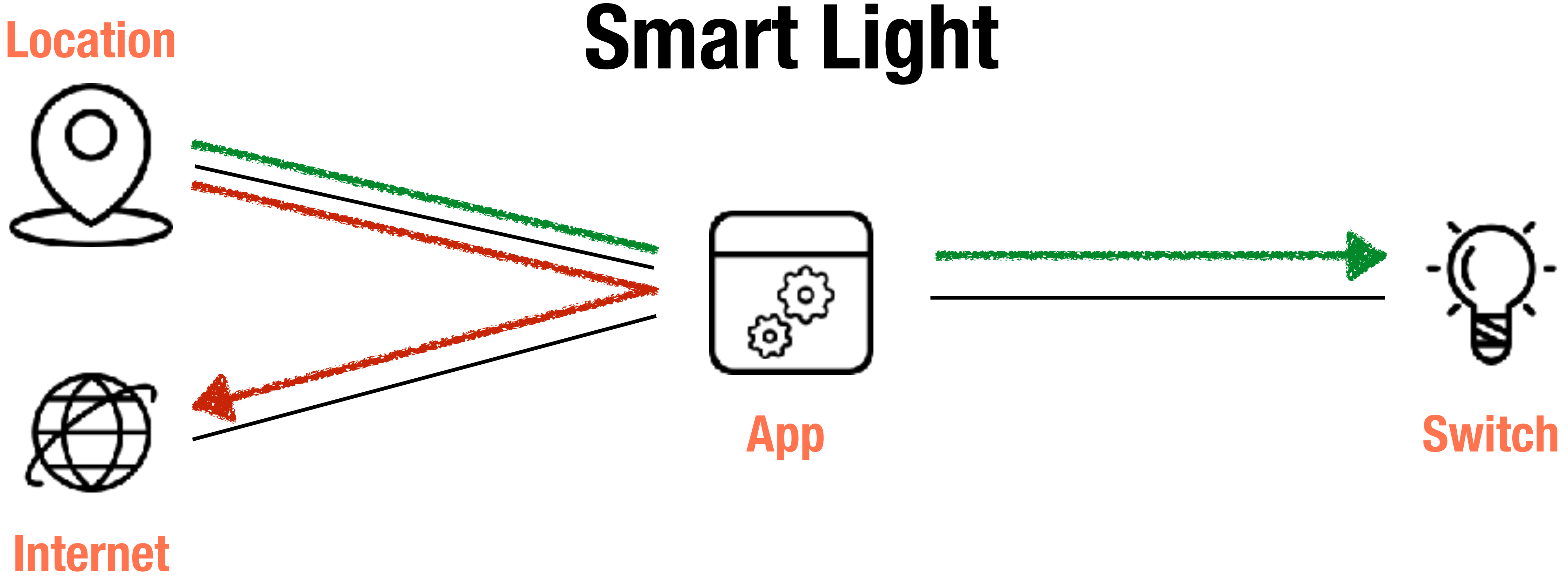
Smart Light



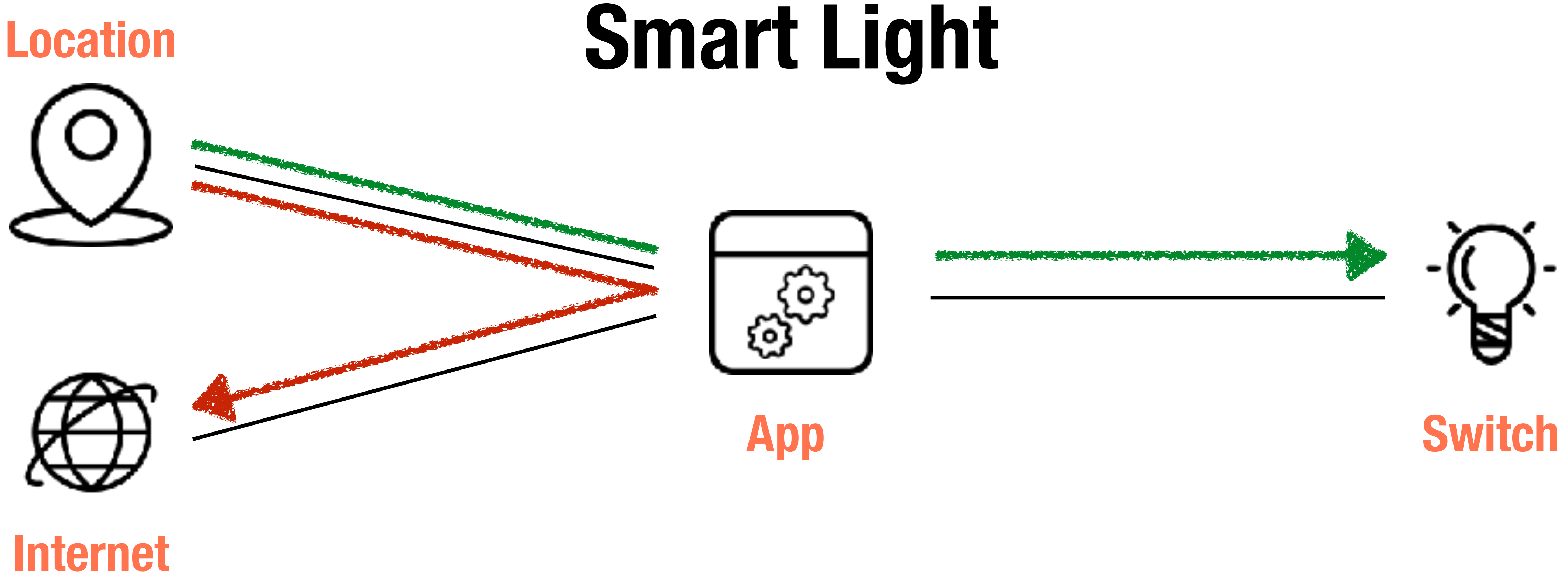
Smart Light



Smart Light



Smart Light



Permissions

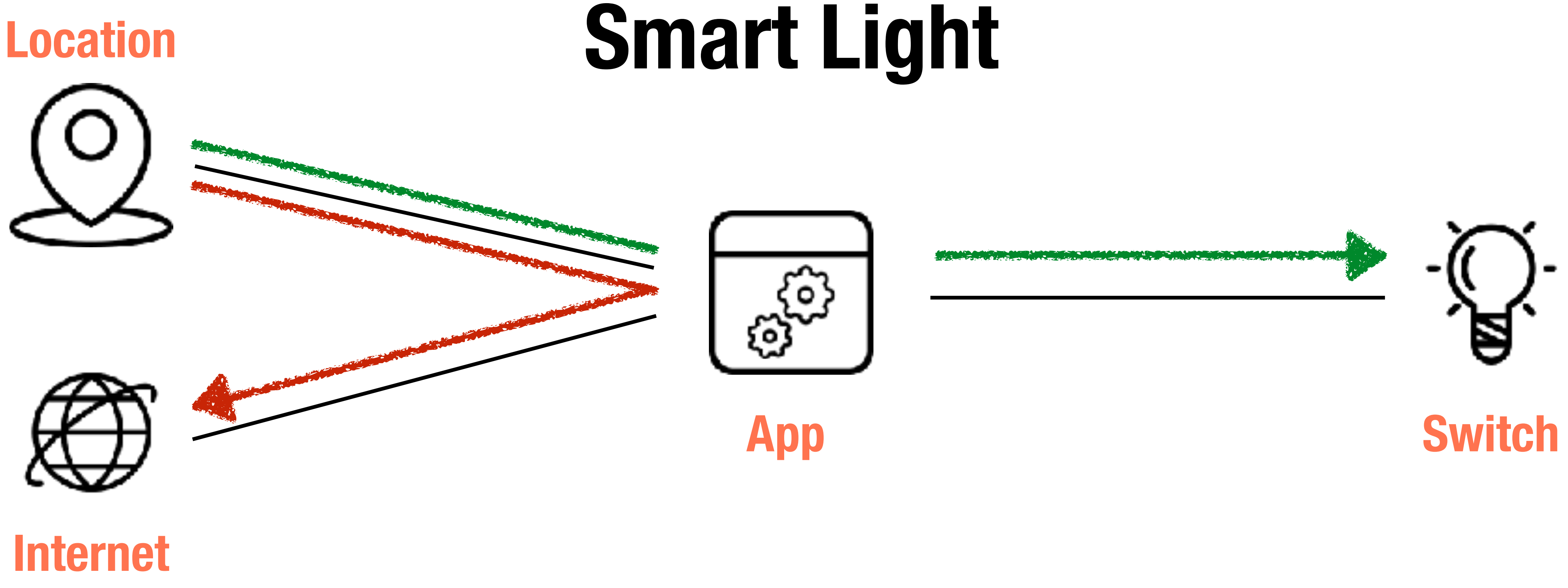
Access Control

Location

Internet

Switch

Smart Light



Permissions

Access Control	Flow Control
Location	Location → Switch
Internet	Location → Internet
Switch	

Enable apps to compute on sensitive data while mitigating data abuse

Label-based Flow Control

- Component level information tracking
- Enforce flows through label policies

Label-based Flow Control

- Component level information tracking
- Enforce flows through label policies

+

Language-based Flow Control

- Restructure apps to obey flow rules
- Developer declares flows

Label-based Flow Control

- Component level information tracking
- Enforce flows through label policies

+

Language-based Flow Control

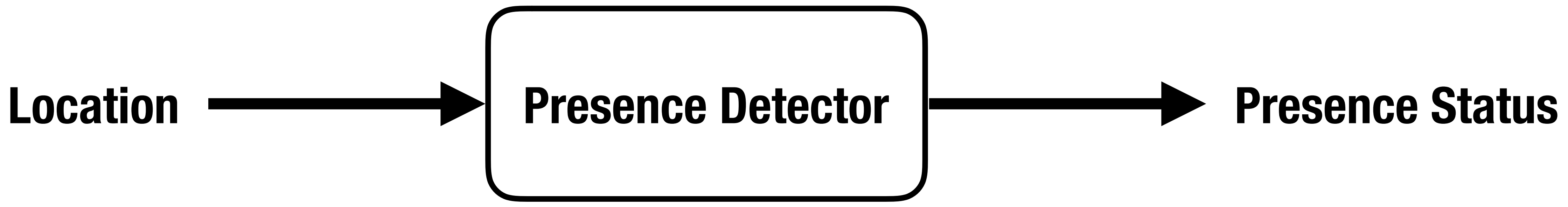
- Restructure apps to obey flow rules
- Developer declares flows

FlowFence

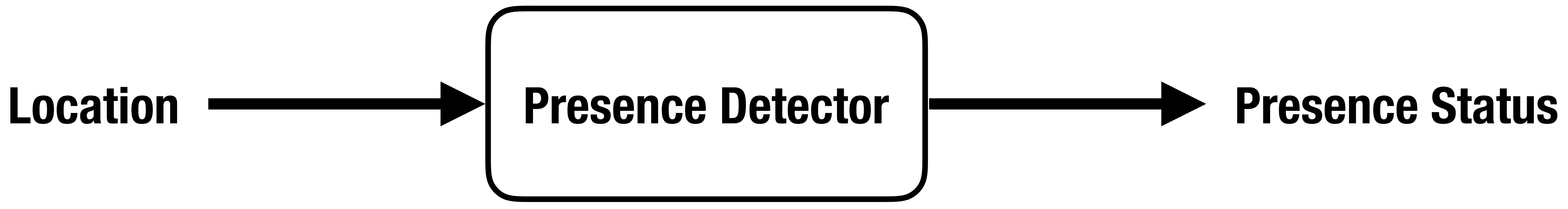
- Supports source- and user-approved data flows
- Allows use of existing languages, tools, and OSes

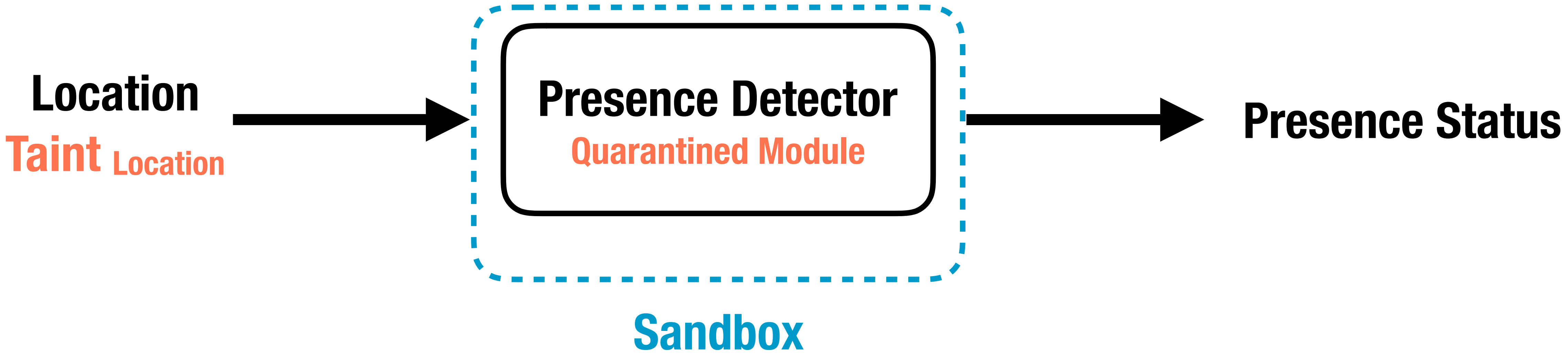
Primitives

Primitives

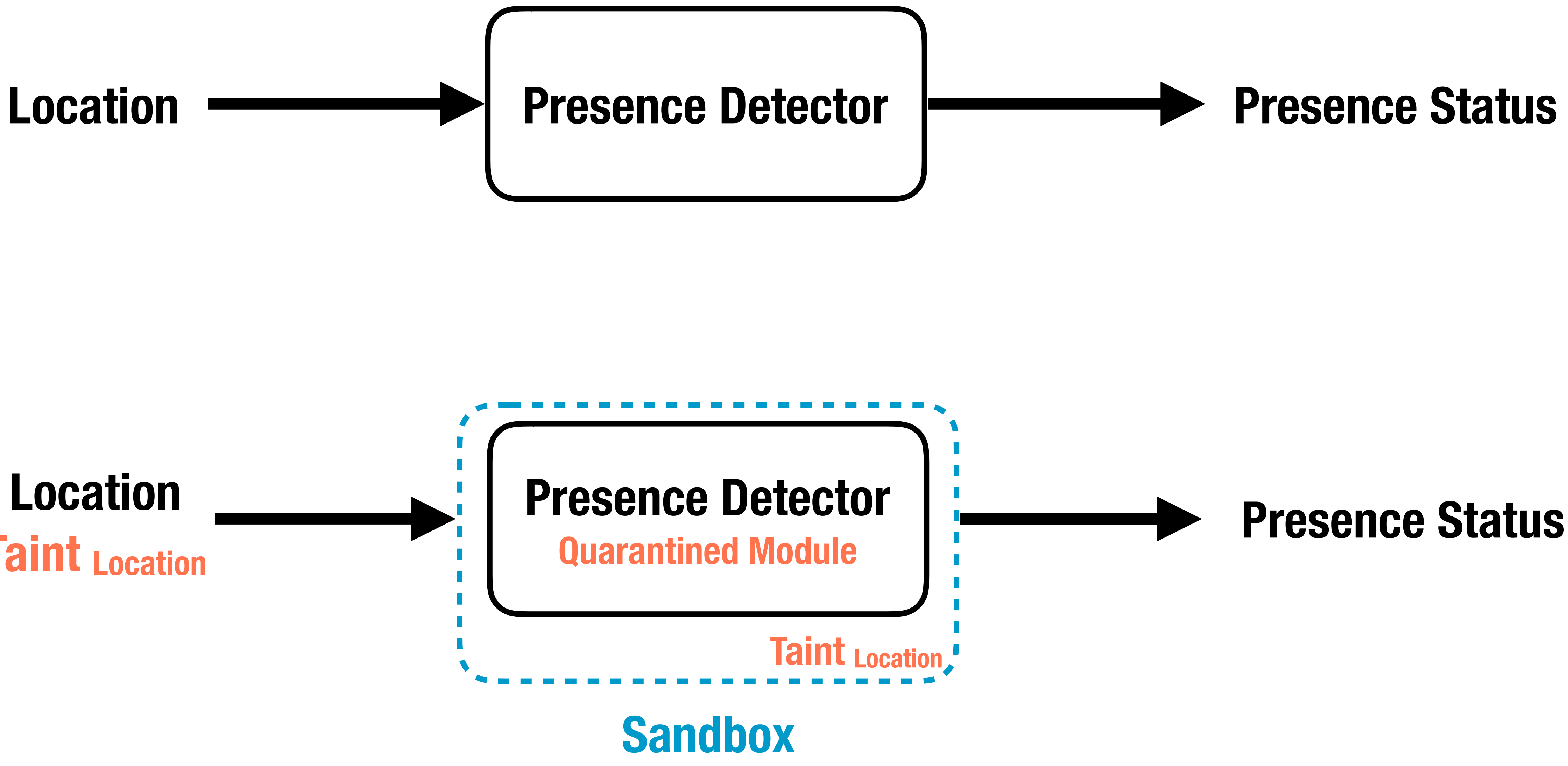


Primitives

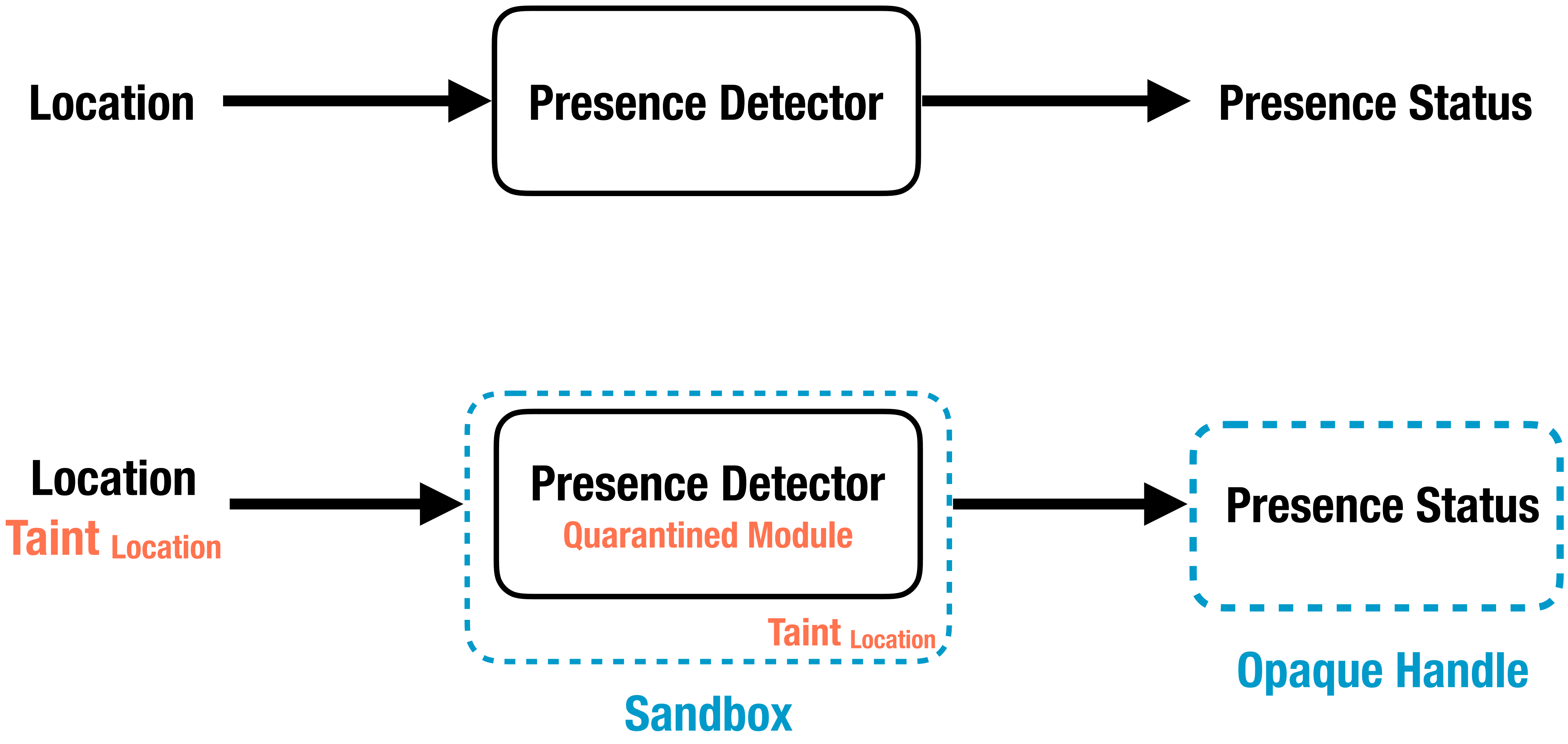

Location
Presence Detector
Presence Status


Location
Taint Location
Presence Detector
Quarantined Module
Presence Status
Sandbox

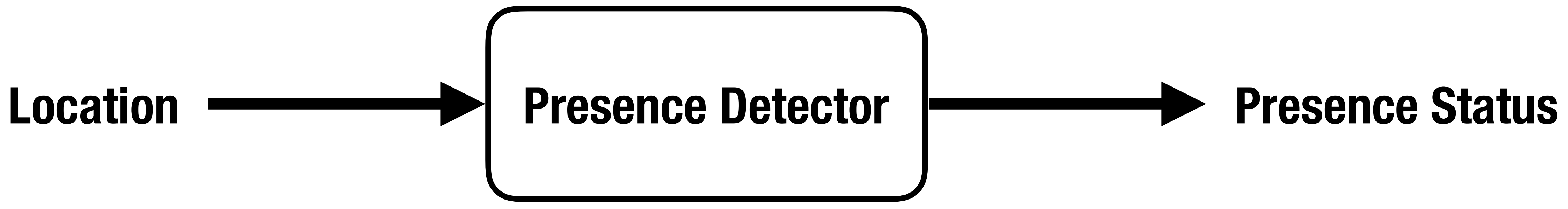
Primitives

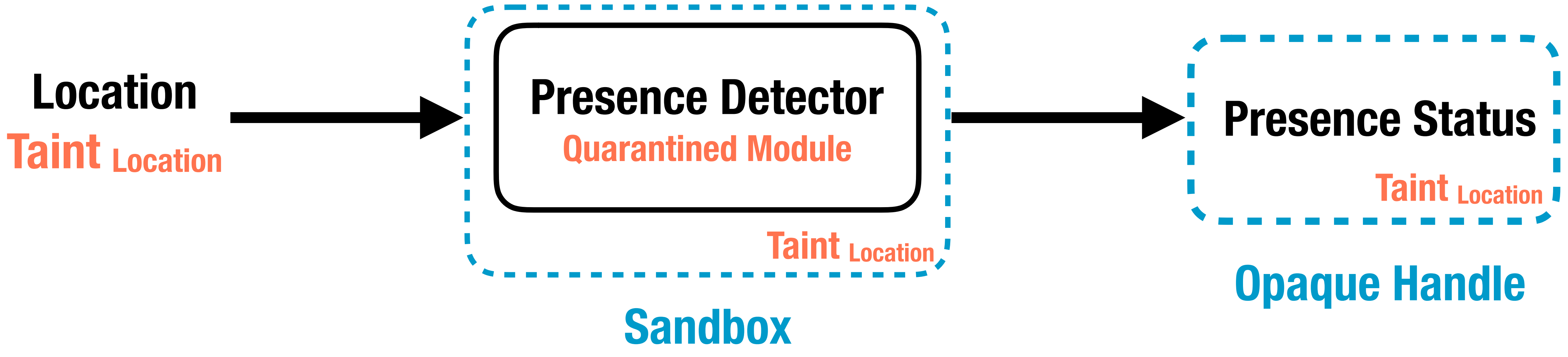

Location
Presence Detector
Presence Status
Location
Taint Location
Presence Detector
Quarantined Module
Taint Location
Presence Status
Sandbox

Primitives

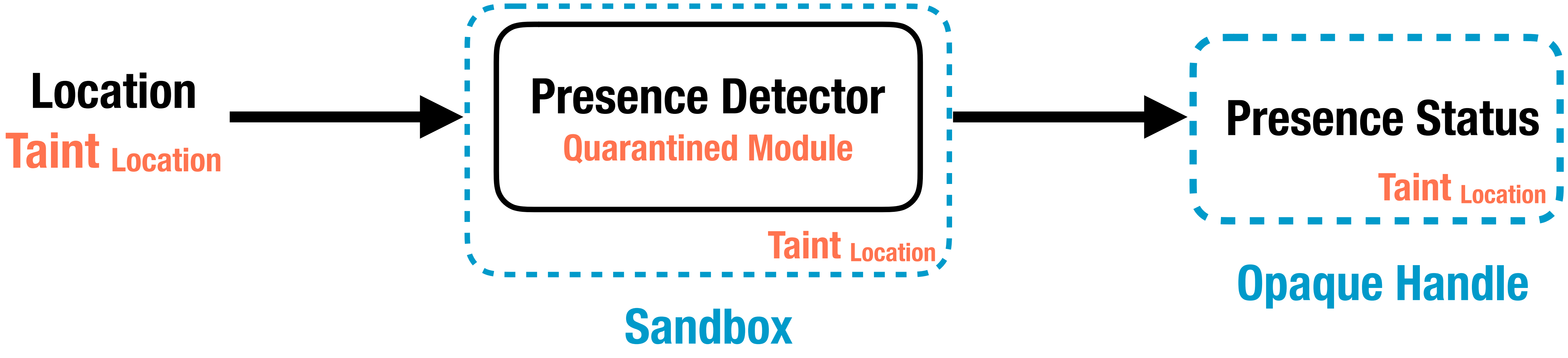

Location
Presence Detector
Presence Status
Location
Taint Location
Presence Detector
Quarantined Module
Taint Location
Sandbox
Presence Status
Opaque Handle

Primitives


Location
Presence Detector
Presence Status


Location
Taint Location
Presence Detector
Quarantined Module
Taint Location
Sandbox
Presence Status
Taint Location
Opaque Handle

Primitives

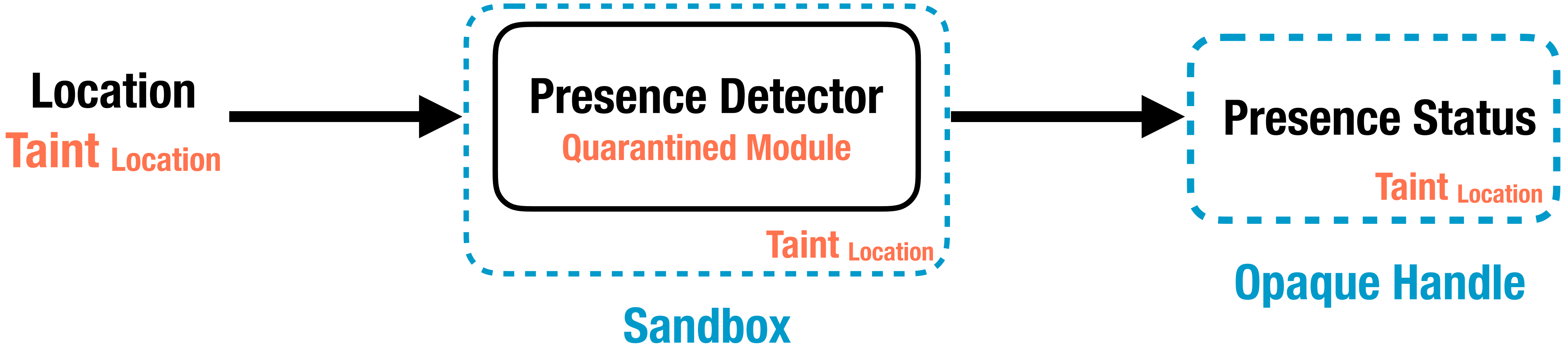


Sandboxes

Provide two methods for data sharing:

- Key-value store
- Event channels

Primitives



Sandboxes

Provide two methods for data sharing:

- Key-value store
- Event channels

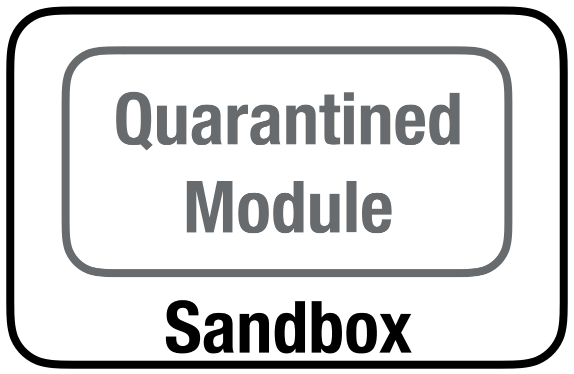
Opaque Handles

do NOT reveal

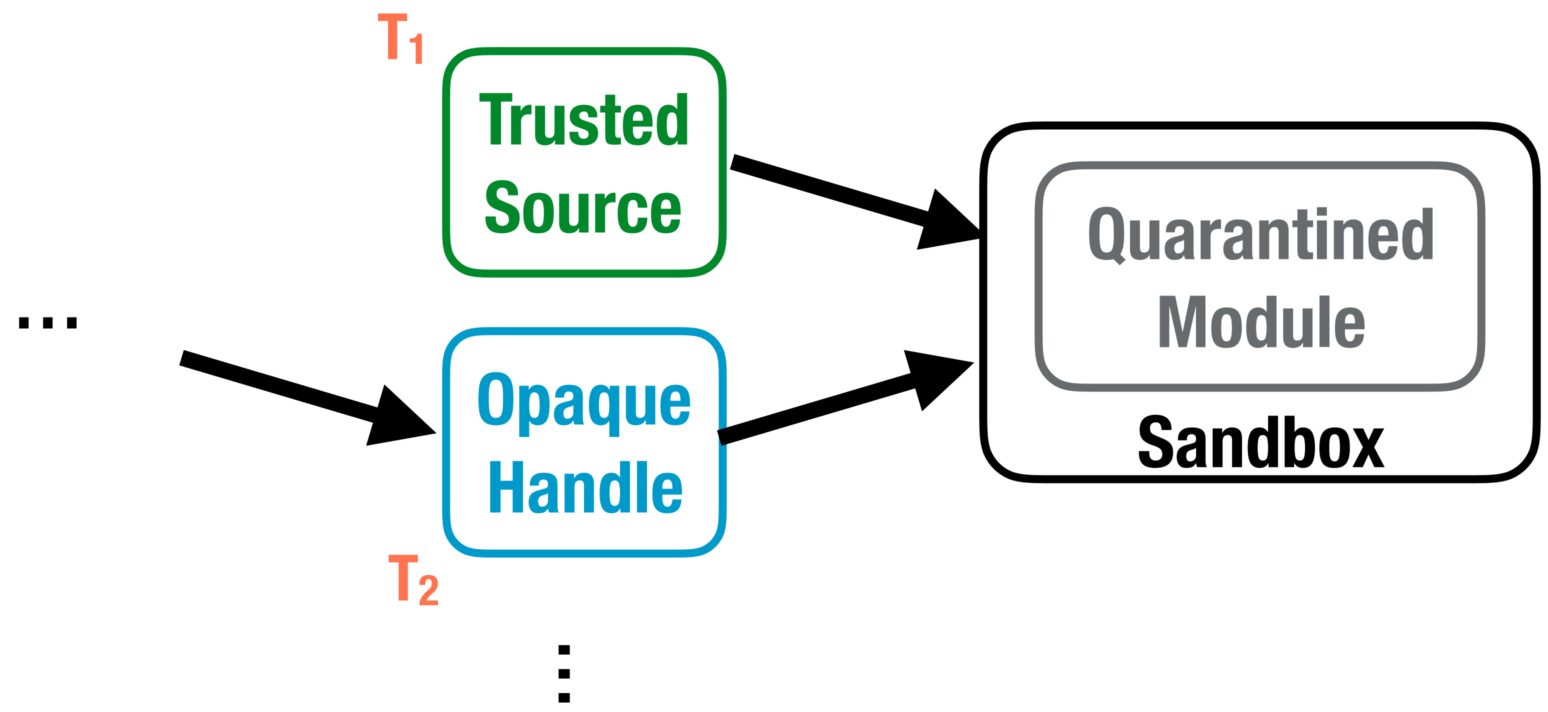
- Raw data
- Data type
- Taint label
- Data size
- Exceptions

to code not running in a QM

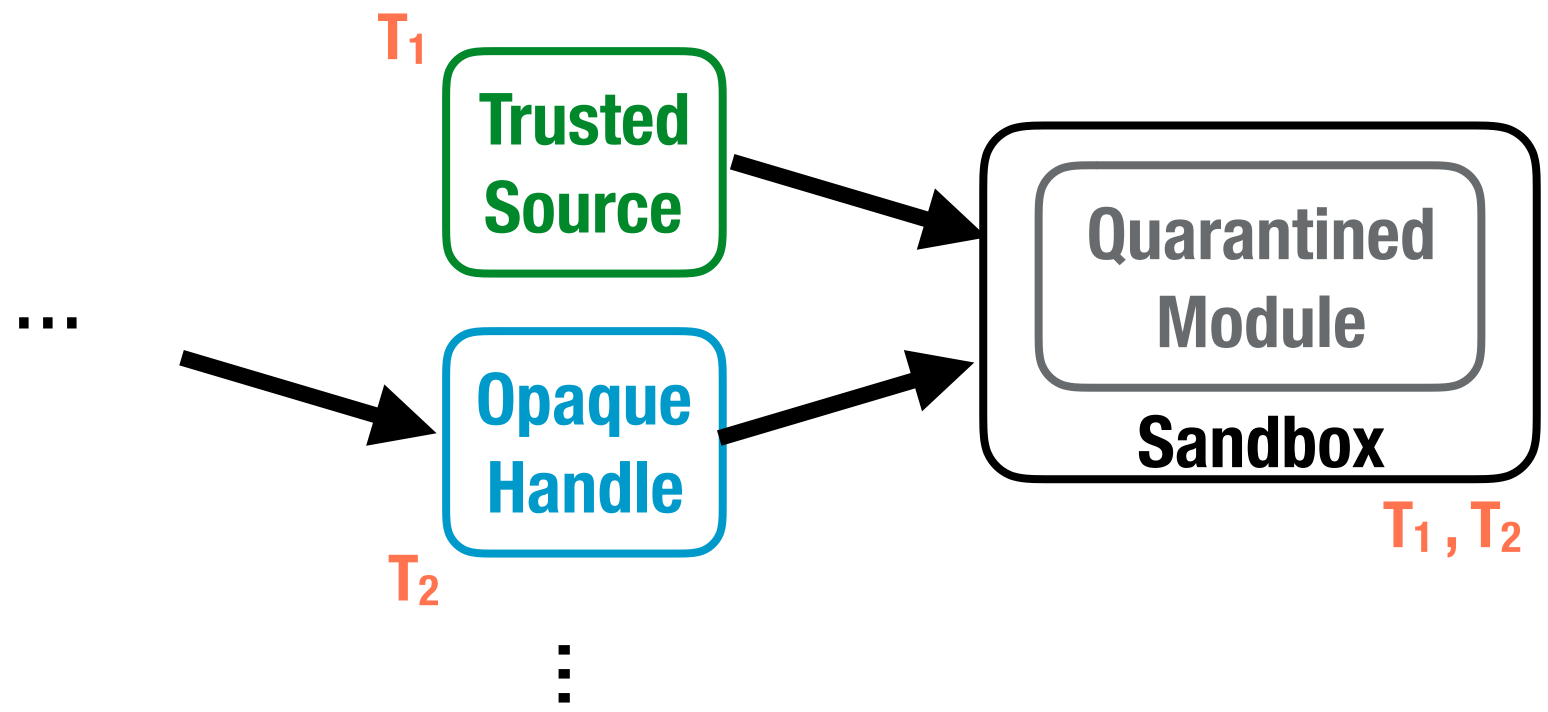
Policy Decisions



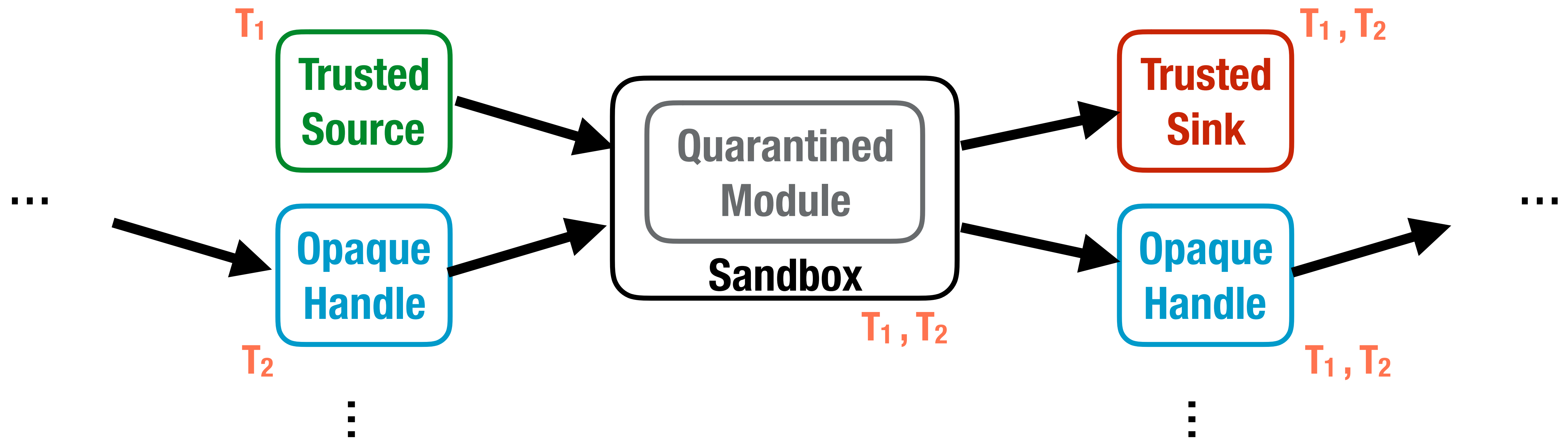
Policy Decisions



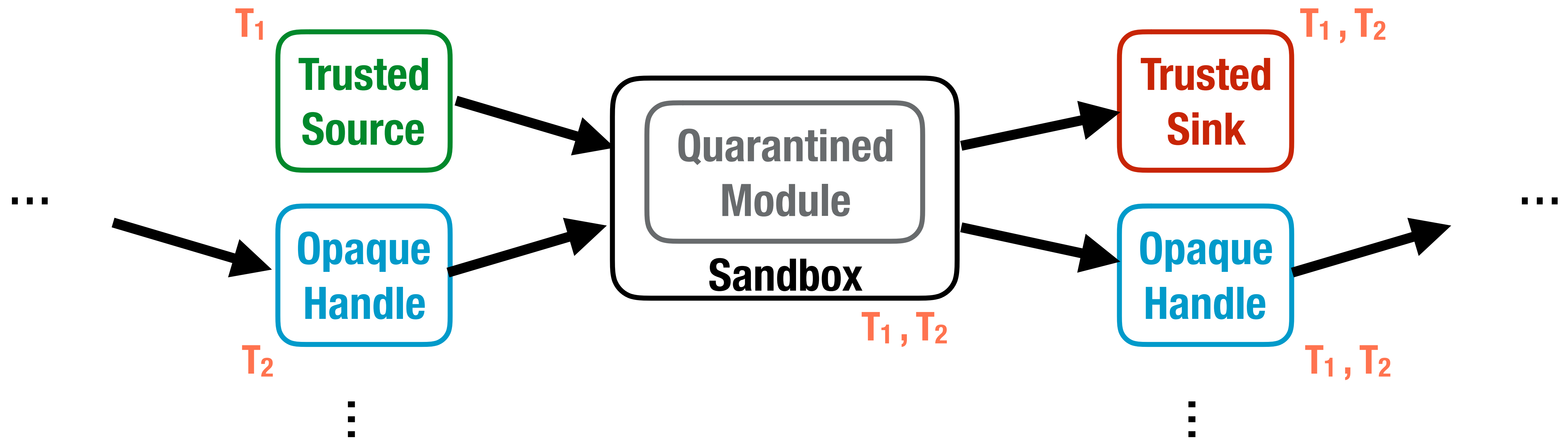
Policy Decisions



Policy Decisions

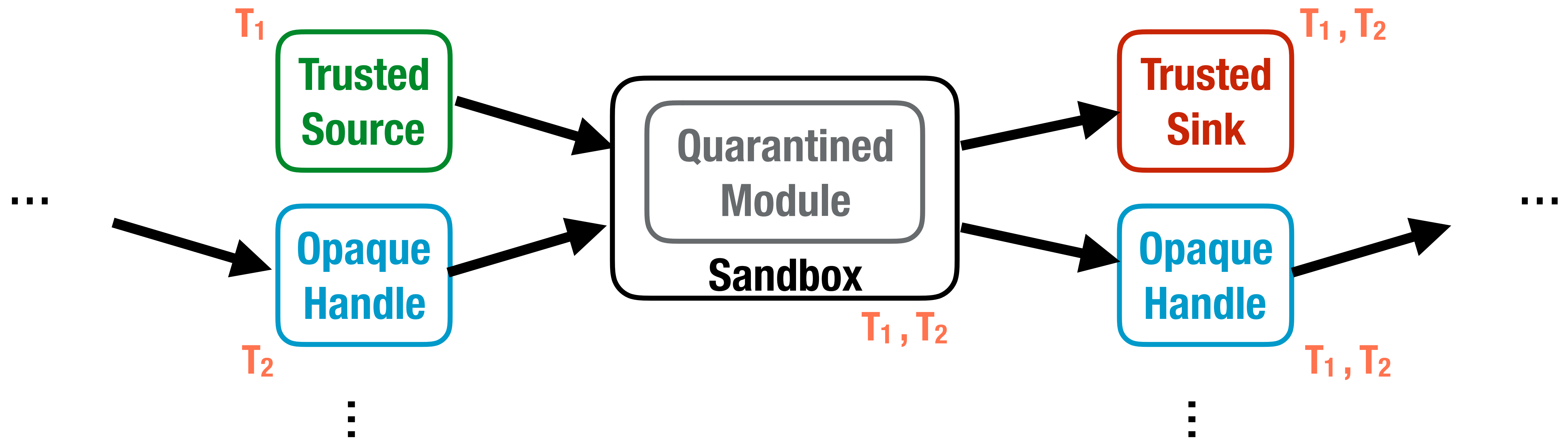


Policy Decisions



$$T_1, T_2 \longrightarrow \text{Sink}$$

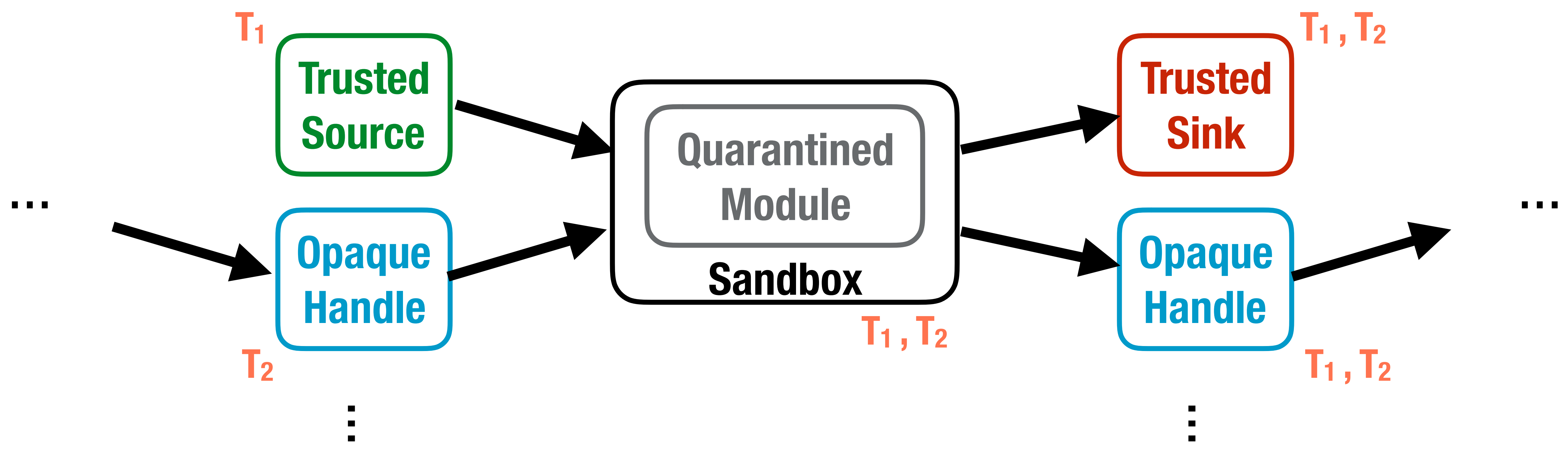
Policy Decisions



$T_1 , T_2 \longrightarrow$ Sink

Source-Approved Policy

Policy Decisions



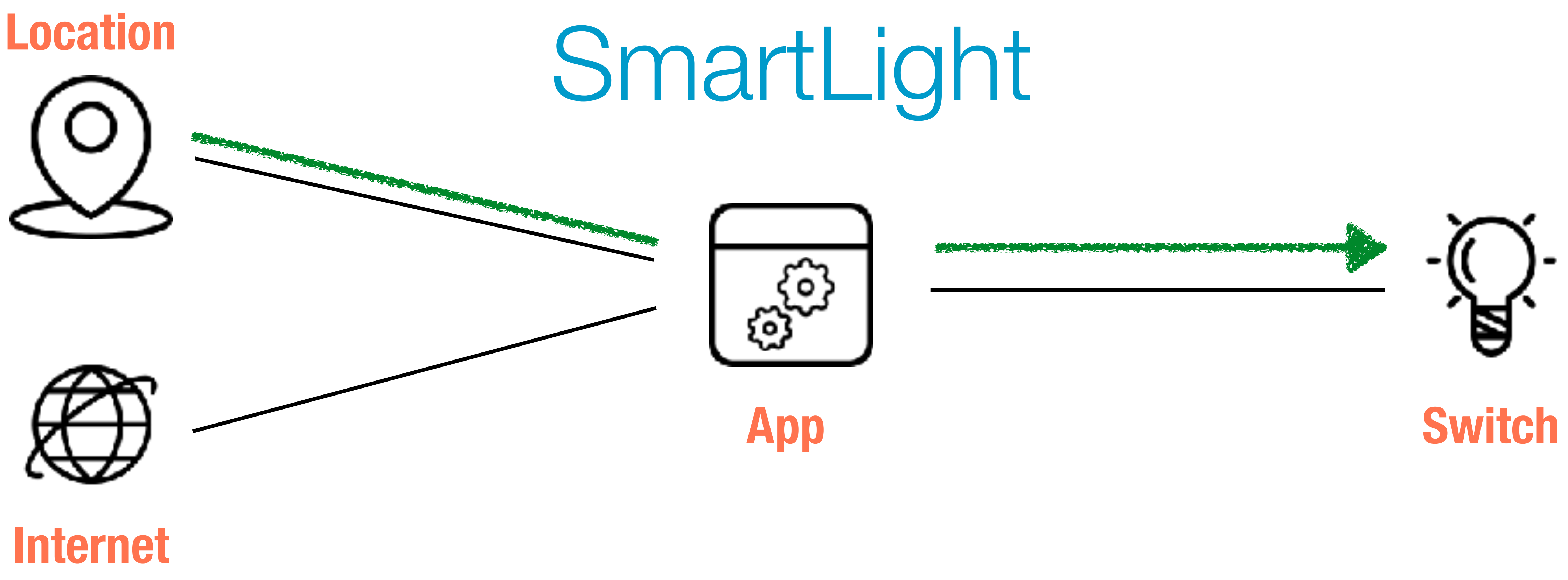
$T_1, T_2 \longrightarrow$ Sink

Source-Approved Policy

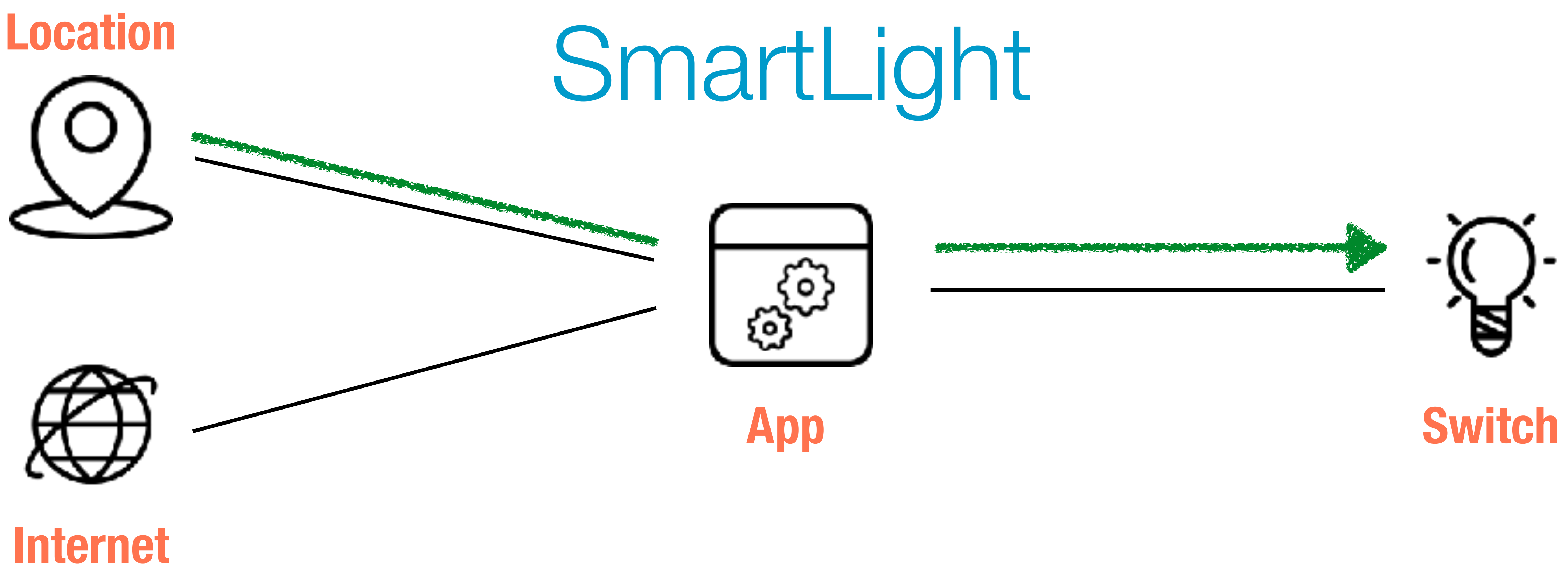
User-Approved Policy

SmartLight

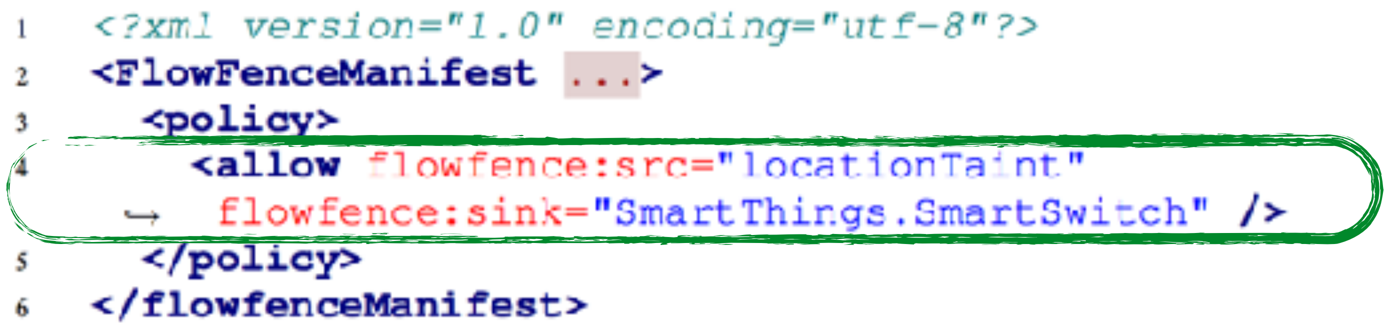
SmartLight



SmartLight

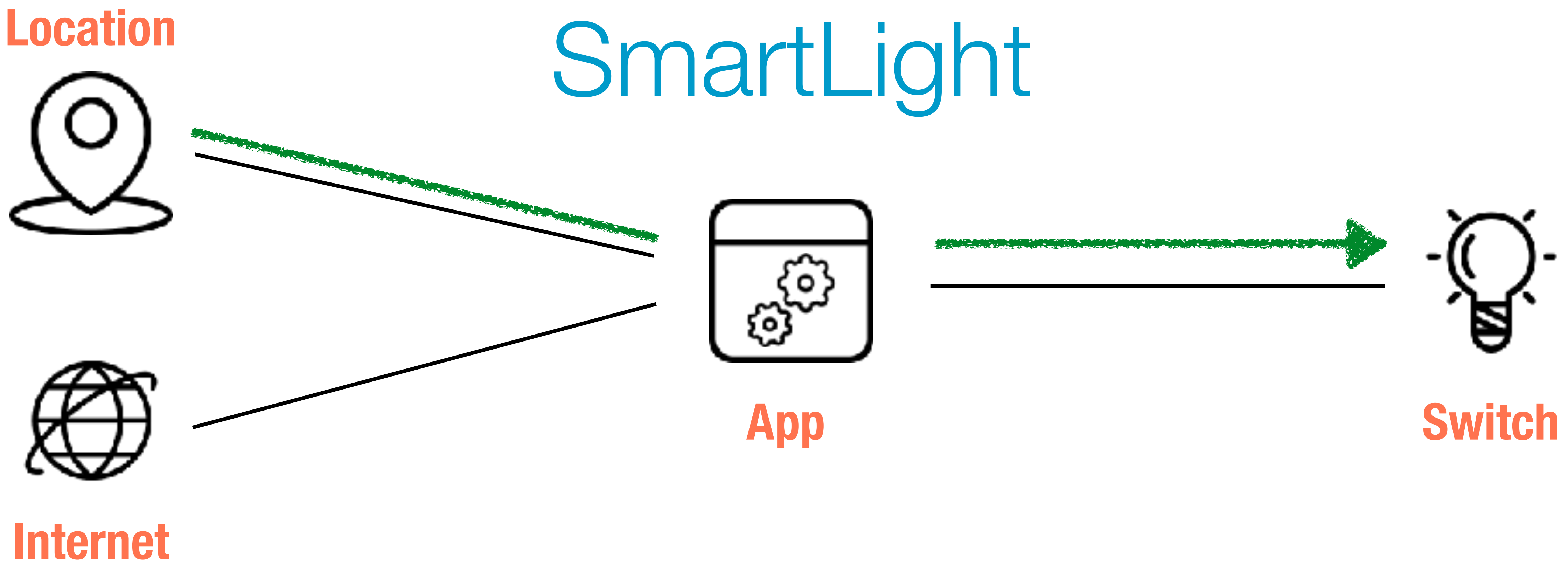


SmartLight Policy

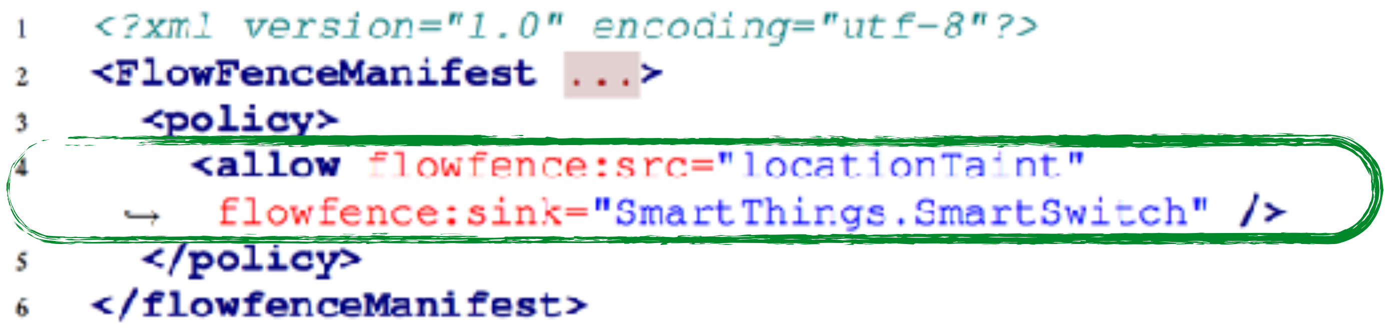


```
<?xml version="1.0" encoding="utf-8"?>
<FlowFenceManifest ...>
  <policy>
    <allow flowfence:src="locationTaint"
   ↪  flowfence:sink="SmartThings.SmartSwitch" />
  </policy>
</flowfenceManifest>
```

SmartLight



SmartLight Policy

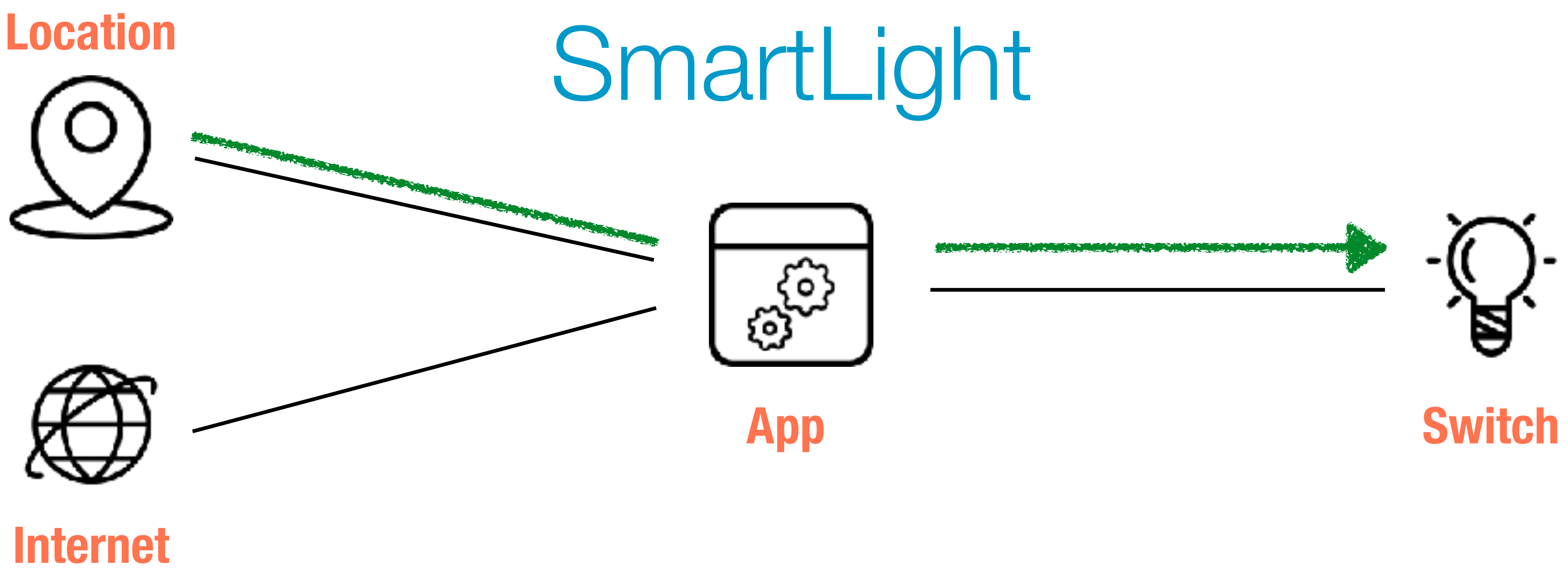


```
<?xml version="1.0" encoding="utf-8"?>
<FlowFenceManifest ...>
  <policy>
    <allow flowfence:src="locationTaint"
  ↪  flowfence:sink="SmartThings.SmartSwitch" />
  </policy>
</flowfenceManifest>
```

Source-Approved Policy

User-Approved Policy

SmartLight



SmartLight Policy

```
<?xml version="1.0" encoding="utf-8"?>
<FlowFenceManifest ...>
  <policy>
    <allow flowfence:src="locationTaint"
  ↪  flowfence:sink="SmartThings.SmartSwitch" />
  </policy>
</flowfenceManifest>
```

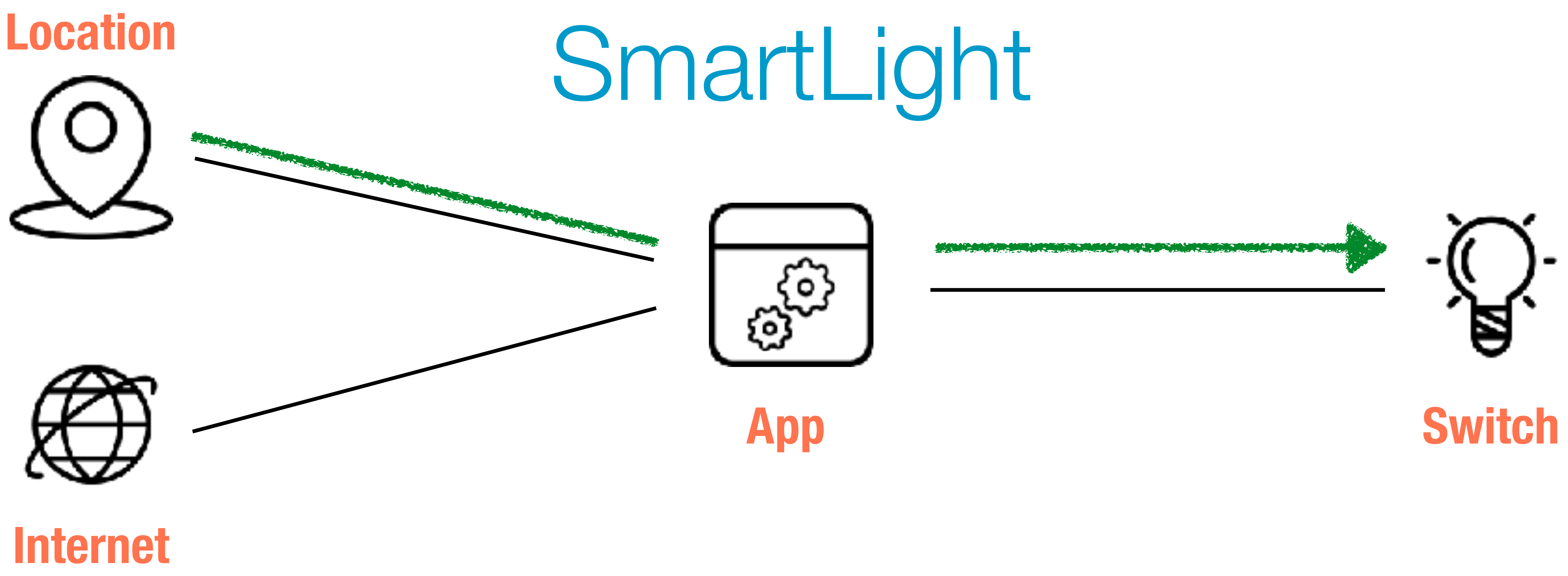
Source-Approved Policy

Location Service Policy

```
<?xml version="1.0" encoding="utf-8"?>
<FlowFenceManifest ...>
  <event-channel flowfence:name =
  ↪  "presenceUpdateChannel" flowfence:exported =
  ↪  "both" />
</flowfenceManifest>
```

User-Approved Policy

SmartLight



SmartLight Policy

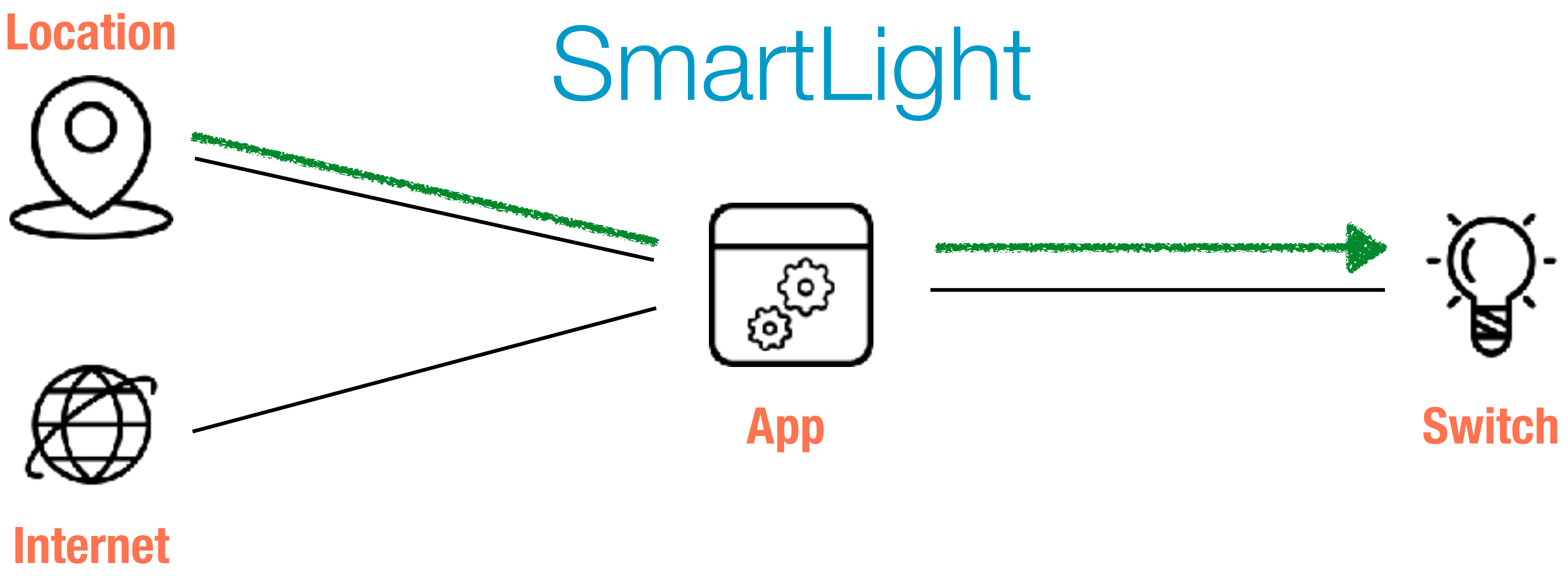
```
<?xml version="1.0" encoding="utf-8"?>
<FlowFenceManifest ...>
  <policy>
    <allow flowfence:src="locationTaint"
  ↪  flowfence:sink="SmartThings.SmartSwitch" />
  </policy>
</flowfenceManifest>
```

~~Source-Approved~~ Policy

Location Service Policy

```
<?xml version="1.0" encoding="utf-8"?>
<FlowFenceManifest ...>
  <event-channel flowfence:name =
  ↪  "presenceUpdateChannel" flowfence:exported =
  ↪  "both" />
</flowfenceManifest>
```

User-Approved Policy



SmartLight Policy

```
<?xml version="1.0" encoding="utf-8"?>
<FlowFenceManifest ...>
  <policy>
    <allow flowfence:src="locationTaint"
    ↪ flowfence:sink="SmartThings.SmartSwitch" />
  </policy>
</flowfenceManifest>
```

~~Source-Approved~~ Policy

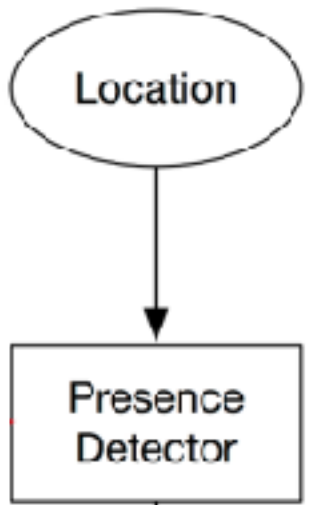
Location Service Policy

```
<?xml version="1.0" encoding="utf-8"?>
<FlowFenceManifest ...>
  <event-channel flowfence:name =
  ↪ "presenceUpdateChannel" flowfence:exported =
  ↪ "both" />
</flowfenceManifest>
```

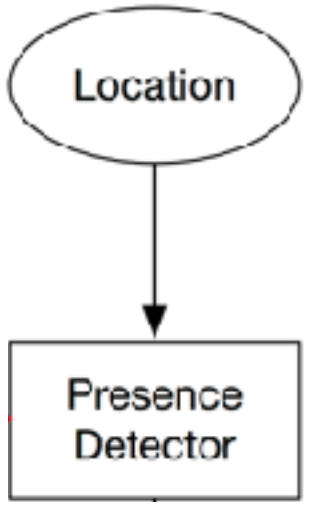
User-Approved Policy

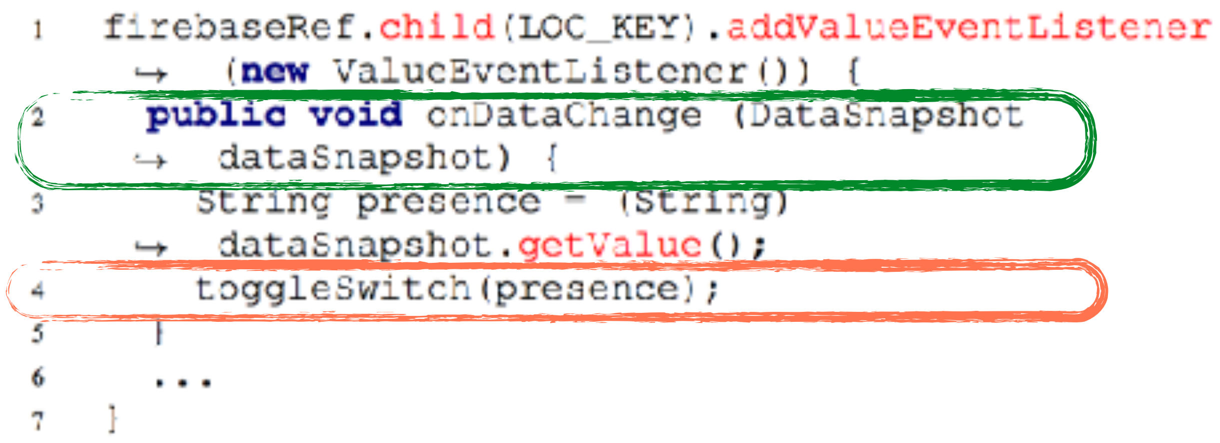
Normal Structure

Normal Structure



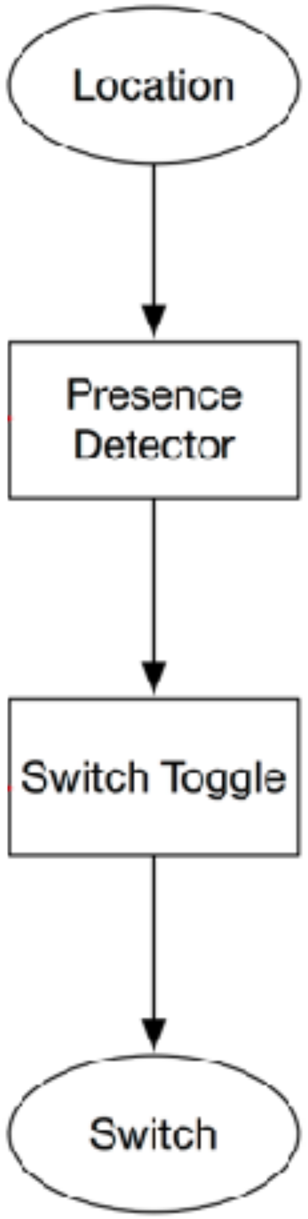
Normal Structure



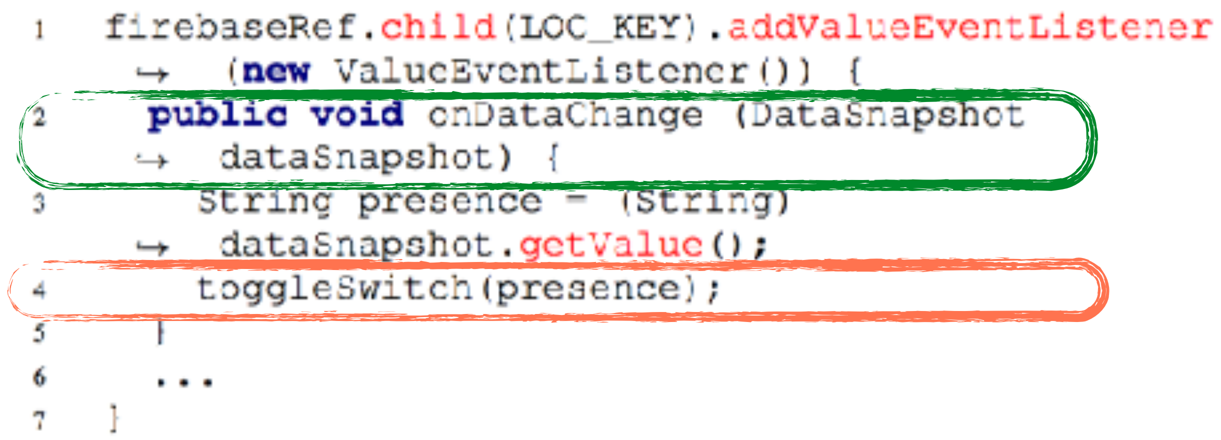


```
firebaseRef.child(LOC_KEY).addValueEventListener
 ↪   (new ValueEventListener()) {
  public void onDataChange (DataSnapshot
 ↪   dataSnapshot) {
    String presence = (String)
 ↪   dataSnapshot.getValue();
    toggleSwitch(presence);
  }
  ...
}
```

Normal Structure

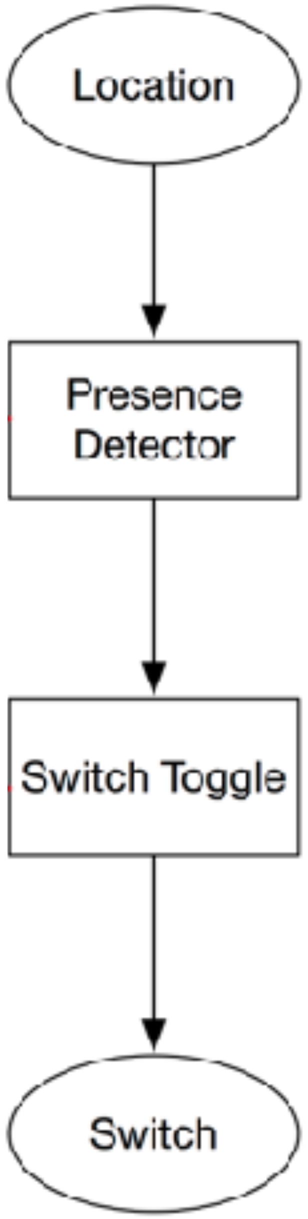


(a) Default

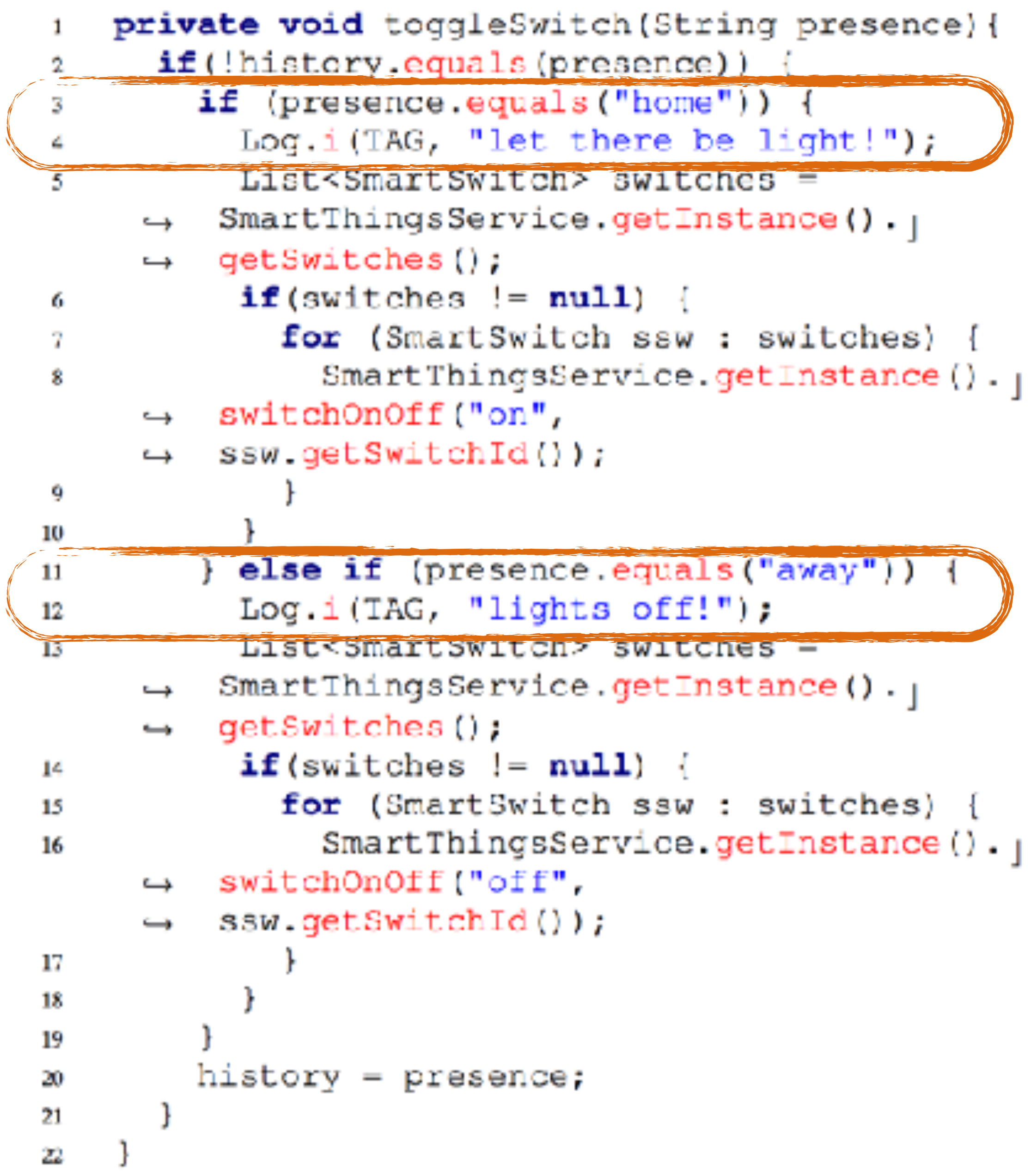


```
firebaseRef.child(LOC_KEY).addValueEventListener
  ↪  (new ValueEventListener()) {
   public void onDataChange (DataSnapshot
  ↪  dataSnapshot) {
      String presence = (String)
  ↪  dataSnapshot.getValue();
      toggleSwitch(presence);
    }
    ...
}
```

Normal Structure

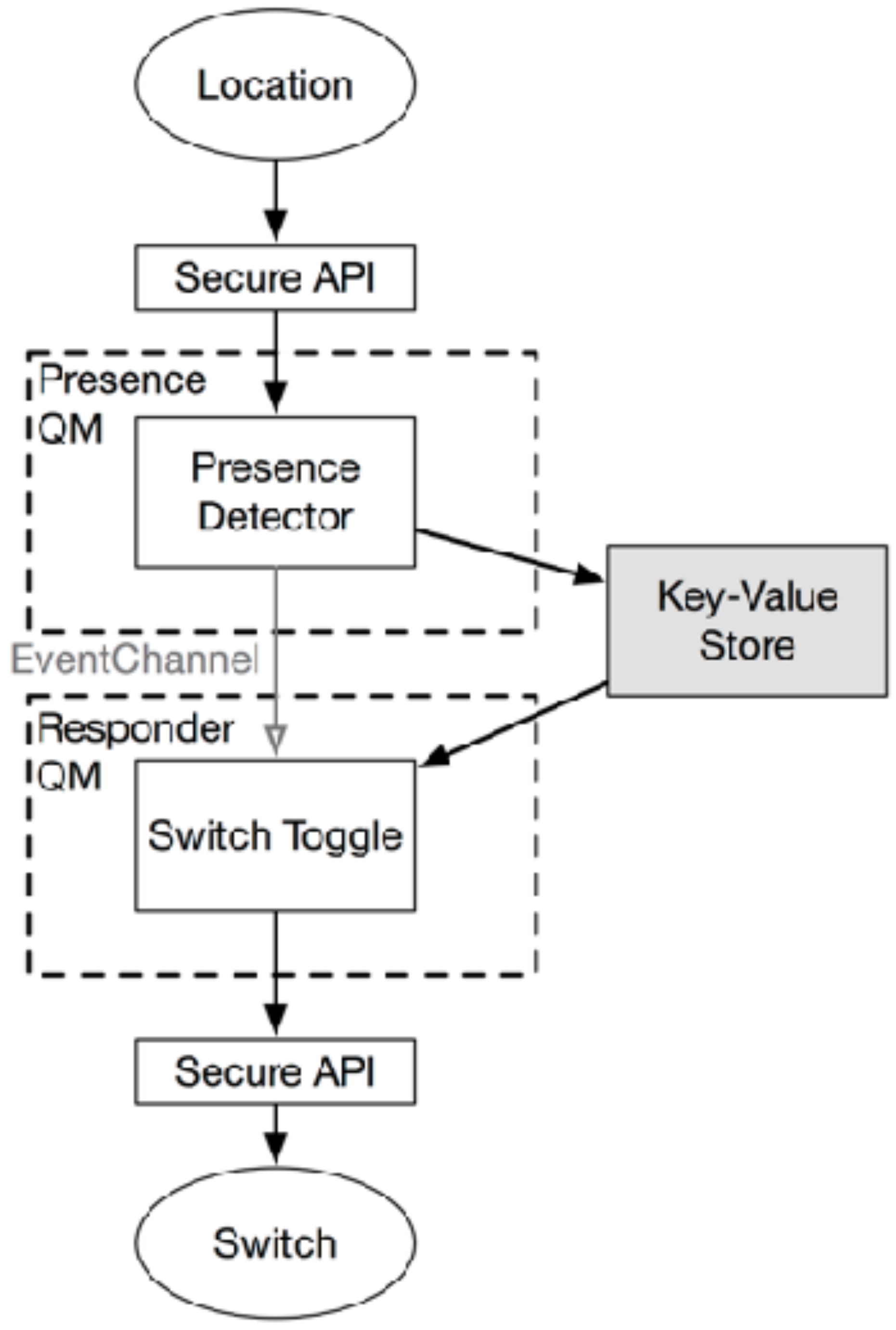


(a) Default



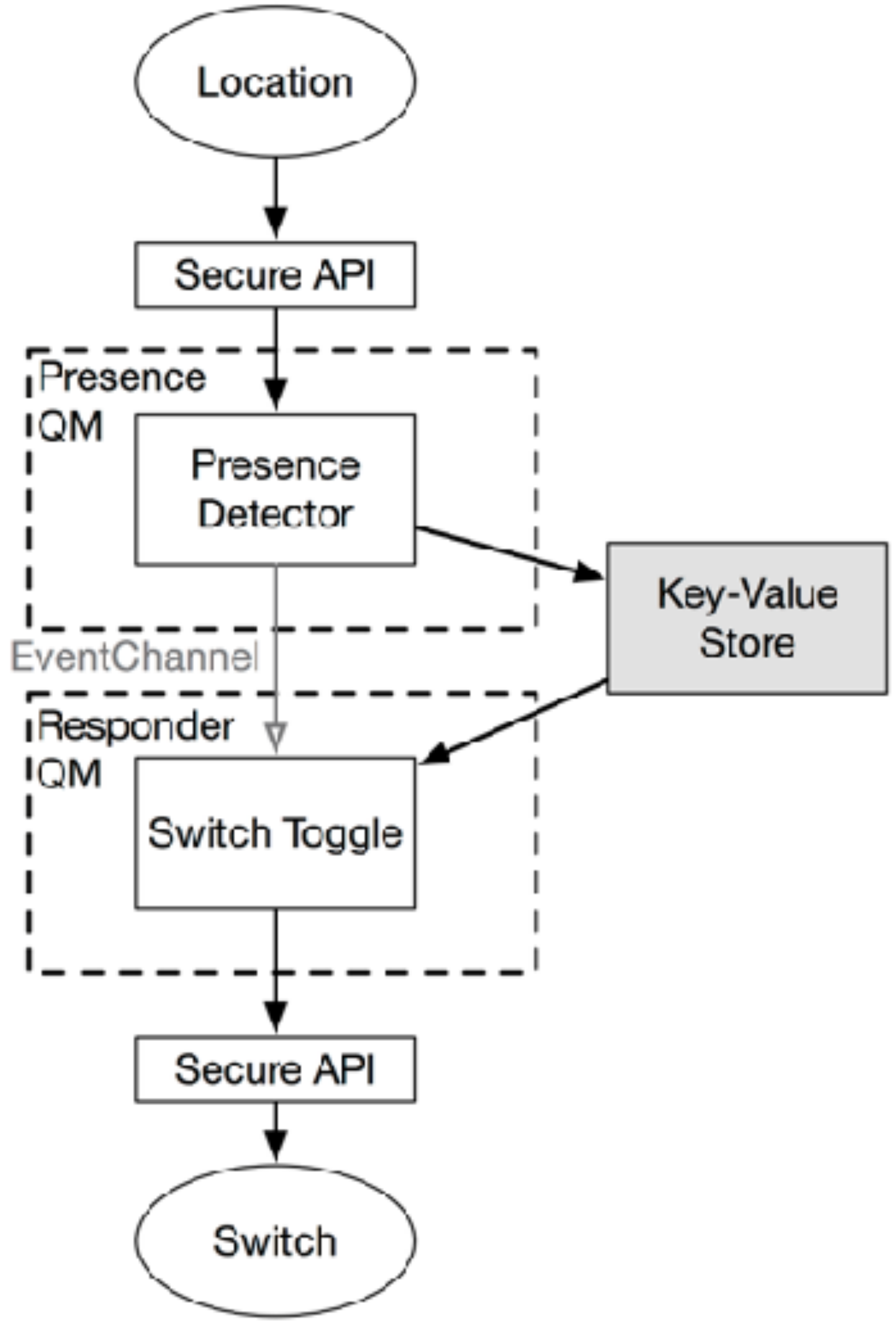
```
private void toggleSwitch(String presence){
  if(!history.equals(presence)) {
    if (presence.equals("home")) {
      Log.i(TAG, "let there be light!");
      List<SmartSwitch> switches = SmartThingsService.getInstance().getSwitches();
      if(switches != null) {
        for (SmartSwitch ssw : switches) {
          SmartThingsService.getInstance().switchOnOff("on", ssw.getSwitchId());
        }
      }
    } else if (presence.equals("away")) {
      Log.i(TAG, "lights off!");
      List<SmartSwitch> switches = SmartThingsService.getInstance().getSwitches();
      if(switches != null) {
        for (SmartSwitch ssw : switches) {
          SmartThingsService.getInstance().switchOnOff("off", ssw.getSwitchId());
        }
      }
    }
    history = presence;
  }
}
```

Presence Detector

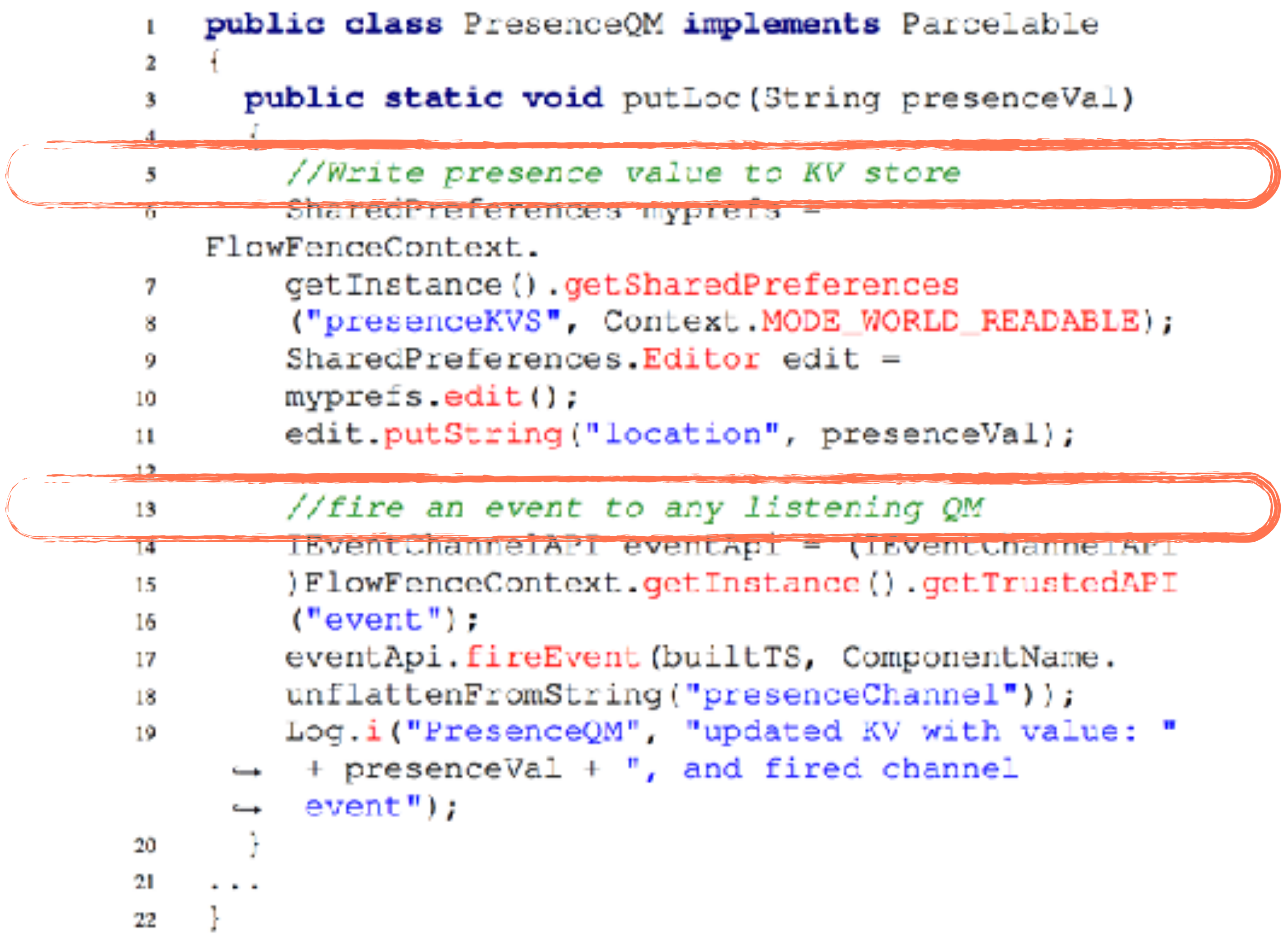


(b) FlowFence

Presence Detector



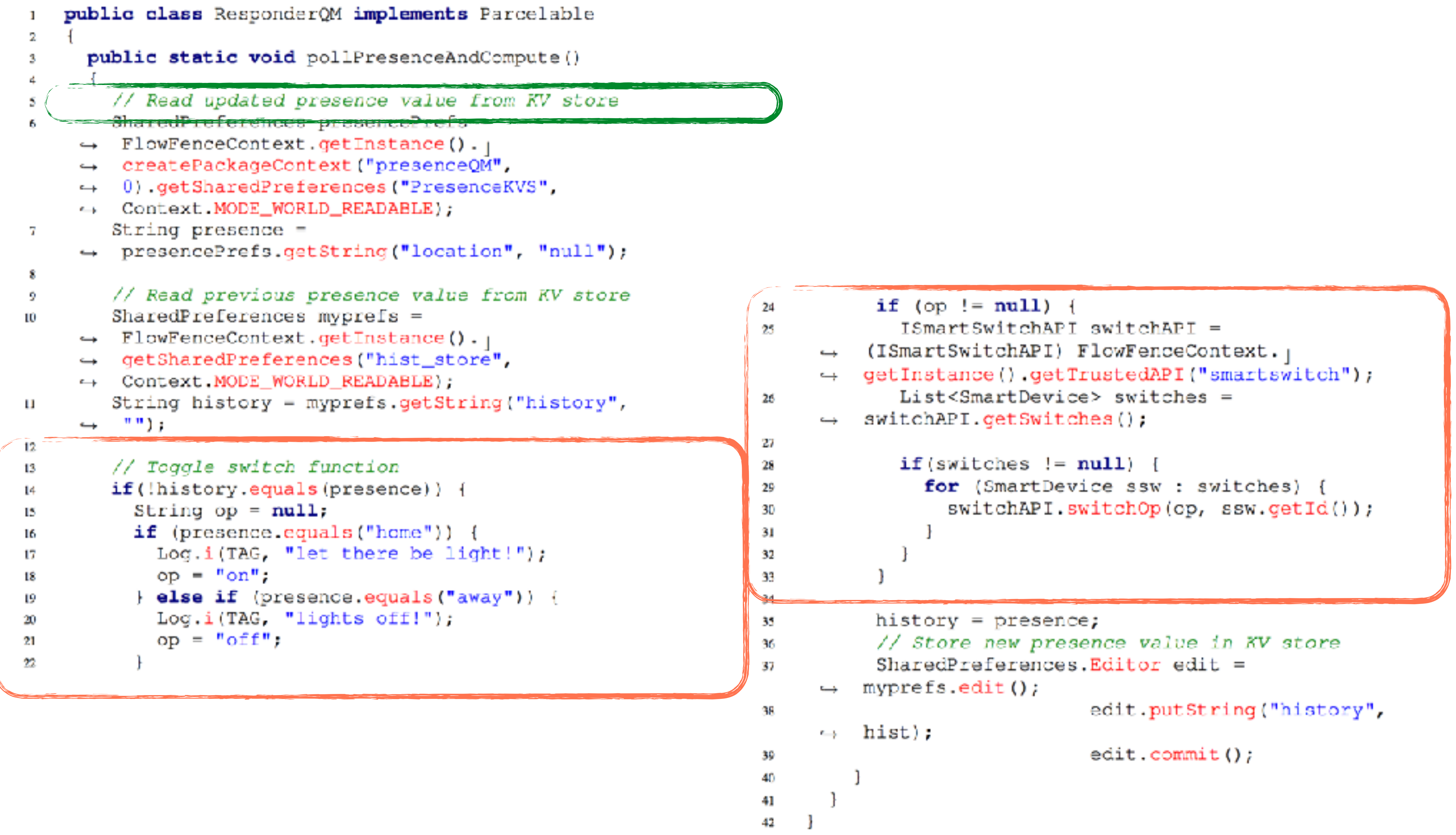
(b) FlowFence



```
public class PresenceQM implements Parcelable
{
  public static void putLoc(String presenceVal)
  {
    //Write presence value to KV store
    SharedPreferences myprefs =
FlowFenceContext.
    getInstance().getSharedPreferences
    ("presenceKVS", Context.MODE_WORLD_READABLE);
    SharedPreferences.Editor edit =
    myprefs.edit();
    edit.putString("location", presenceVal);

    //fire an event to any listening QM
    IEventChannelAPI eventApi = (IEventChannelAPI
    )FlowFenceContext.getInstance().getTrustedAPI
    ("event");
    eventApi.fireEvent(builtTS, ComponentName.
    unflattenFromString("presenceChannel"));
    Log.i("PresenceQM", "updated KV with value: "
  ↪  + presenceVal + ", and fired channel
  ↪  event");
  }
  ...
}
```

App Structure



```
public class ResponderQM implements Parcelable
{
  public static void pollPresenceAndCompute()
  {
    // Read updated presence value from KV store
    SharedPreferences presencePrefs =
     FlowFenceContext.getInstance().
     createPackageContext("presenceQM",
     0).getSharedPreferences("PresenceKVS",
     Context.MODE_WORLD_READABLE);
    String presence =
     presencePrefs.getString("location", "null");

    // Read previous presence value from KV store
    SharedPreferences myprefs =
     FlowFenceContext.getInstance().
     getSharedPreferences("hist_store",
     Context.MODE_WORLD_READABLE);
    String history = myprefs.getString("history",
     "");

    // Toggle switch function
    if(!history.equals(presence)) {
      String op = null;
      if (presence.equals("home")) {
        Log.i(TAG, "let there be light!");
        op = "on";
      } else if (presence.equals("away")) {
        Log.i(TAG, "lights off!");
        op = "off";
      }
      if (op != null) {
        ISmartSwitchAPI switchAPI =
       (ISmartSwitchAPI) FlowFenceContext.
       getInstance().getTrustedAPI("smartswitch");
        List<SmartDevice> switches =
       switchAPI.getSwitches();

        if(switches != null) {
          for (SmartDevice ssw : switches) {
            switchAPI.switchOp(op, ssw.getId());
          }
        }
      }

      history = presence;
      // Store new presence value in KV store
      SharedPreferences.Editor edit =
     myprefs.edit();
                    edit.putString("history",
     hist);
                    edit.commit();
    }
  }
}
```

Future Work

Future Work

- Information flow tracking across multiple environments

Future Work

- Information flow tracking across multiple environments
- Mitigating side channel

Future Work

- Information flow tracking across multiple environments
- Mitigating side channel
- Policy management

Future Work

- Information flow tracking across multiple environments
- Mitigating side channel
- Policy management

FlowFence code will be released on December 1st
https://iotsecurity.eecs.umich.edu

Opacified Computation

- Enables practical data flow control for IoT applications.
- Uses Quarantined Modules and Opaque Handles to Explicitly embed control and data flows within app structure.
- Supports publisher and consumer flow policies.
- **FlowFence code will be released on December 1st.**

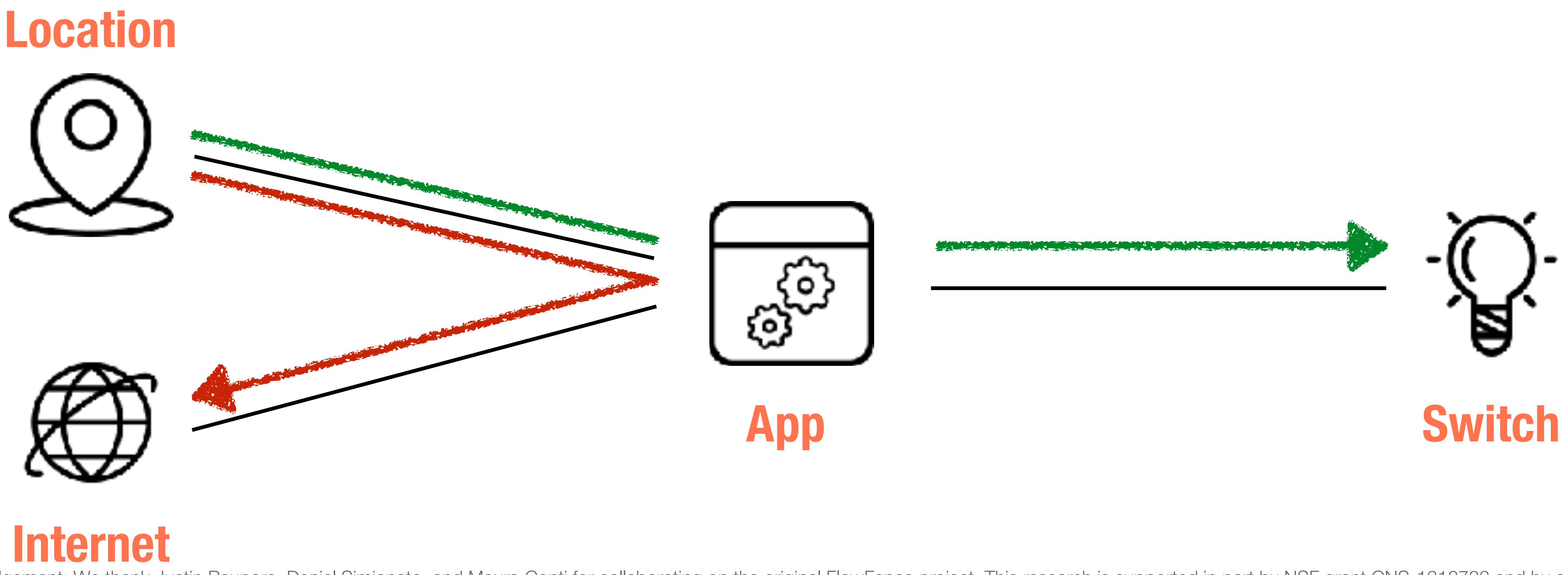
https://iotsecurity.eecs.umich.edu

Opacified Computation

- Enables practical data flow control for IoT applications.
- Uses Quarantined Modules and Opaque Handles to Explicitly embed control and data flows within app structure.
- Supports publisher and consumer flow policies.
- **FlowFence code will be released on December 1st.**

Amir Rahmati
amir.rahmati.com

https://iotsecurity.eecs.umich.edu



Acknowledgement: We thank Justin Paupore, Daniel Simionato, and Mauro Conti for collaborating on the original FlowFence project. This research is supported in part by NSF grant CNS-1318722 and by a generous gift from General Motors. Any opinions, findings, conclusions, and recommendations expressed in this work are those of the authors and do not necessarily reflect the views of the sponsors. Icons copyright © 2016 Icons Mind.