MIUN.SE
DISCOVER YOUR OPPORTUNITIES

Tutorial & Workshop
The SensibleThings Platform

Sundsvall, September, 2013

Stefan Forsström
Department of Information Technology and Media
Mid Sweden University, Sundsvall, Sweden

Investing in your future
EUROPEAN UNION
European Regional Development Fund

Mittuniversitetet
MID SWEDEN UNIVERSITY

MIUN.SE
DISCOVER YOUR OPPORTUNITIES

THE IDEA

- In the future we will have a lot of connected things
  - Commonly referred to as the Internet-of-Things
- Current available solutions store data in databases
  - This will not scale to a global Internet-of-Things
  - Where all things will want to share sensor/actuator information
- We foresee the use of peer-to-peer technology
  - To enable more scalable system without any central points of failure

DEMANDED FEATURES

- From our research goals
  - *Scalable* – logarithmic or better
  - *No central point of* failure – fully distributed
  - *Bidirectional* – capable of communicating with both sensors and actuators
  - *Fast* – capable of signaling in real-time between end points
  - *Current* – all data retrieved should be the most current values
  - *Lightweight* – able to run on mobile devices with limited resources
  - *Seamless* – capable of handling NAT and different and devices
  - *Stable* – handle transient nodes with high churn rates
  - *Extensible* – capable of adding new features without redistribution

MIUN.SE
DISCOVER YOUR OPPORTUNITIES

APPROACH

- To solve this, our approach was to use:

- A fully distributed system
  - To avoid any central point of failure
- Distributed Hash Tables (DHT's)
  - To lookup identities in the system
  - Basically like a distributed DNS system
- Peer-to-Peer data exchange
  - Between connected entities
  - Similar to how file sharing work today
- Reliable UDP (RUDP)
  - To keep the response time minimal
  - Always data in the first packet, no initial handshaking

SENSIBLETHINGS PLATFORM

- Website
  - www.sensiblethings.se
- Current status of the open source project
  - DHT with a fingertable for a logarithmic lookup time
  - Fully distributed, but with a bootstrap node
  - RUDP, with the payload data in the first packet
  - Open Source LGPLv3, a non propagating license
  - Able to run on Android mobile phones and tablets
  - Capable of penetrating multiple NAT, using a proxy solution
  - Add-in and modularized architecture, capable of adding new features
    - Currently: Encryption, authentication, publish/subscribe, etc.

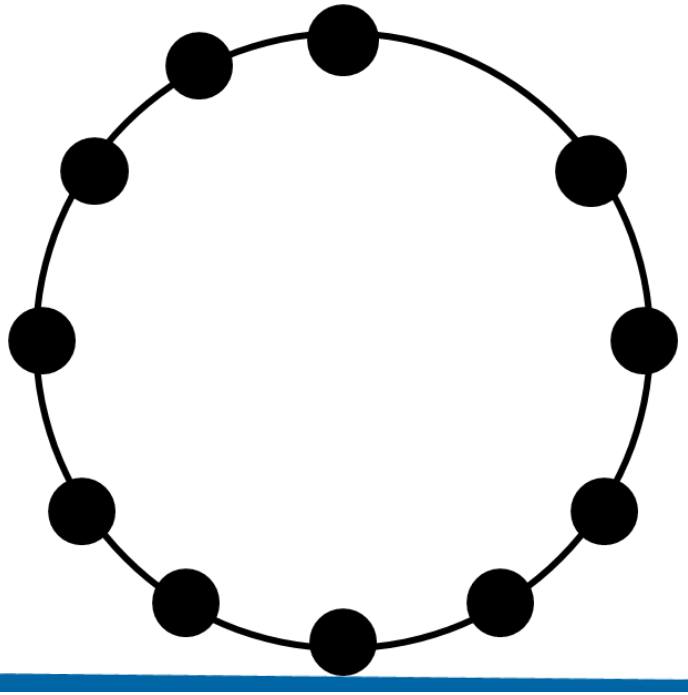
MIUN.SE
DISCOVER YOUR OPPORTUNITIES

How does it work? (Simplified)

Mittuniversitetet
MID SWEDEN UNIVERSITY

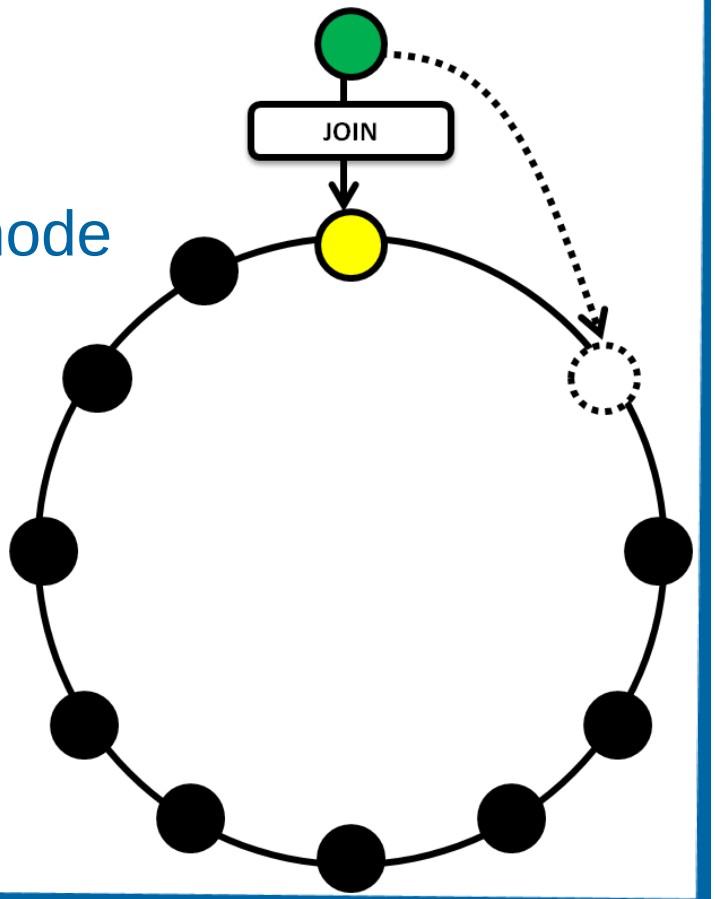
HOW DOES IT WORK?

- Entities organized in a ring like structure
  - In similarity to the Chord DHT
- A node is very ambiguous
  - It can basically be anything
    - A computer
    - A mobile phone
    - Set-top-box
    - Etc.
  - As long as it can run the code



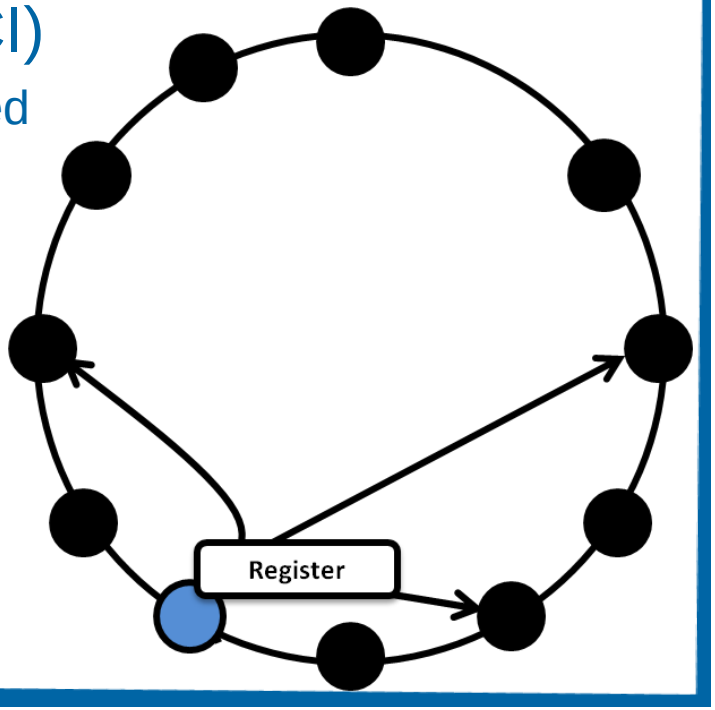
HOW DOES IT WORK?

- A new node can join the ring
  - By talking to any other node in the ring
- But we have a stable bootstrap node
  - To make it easy to join



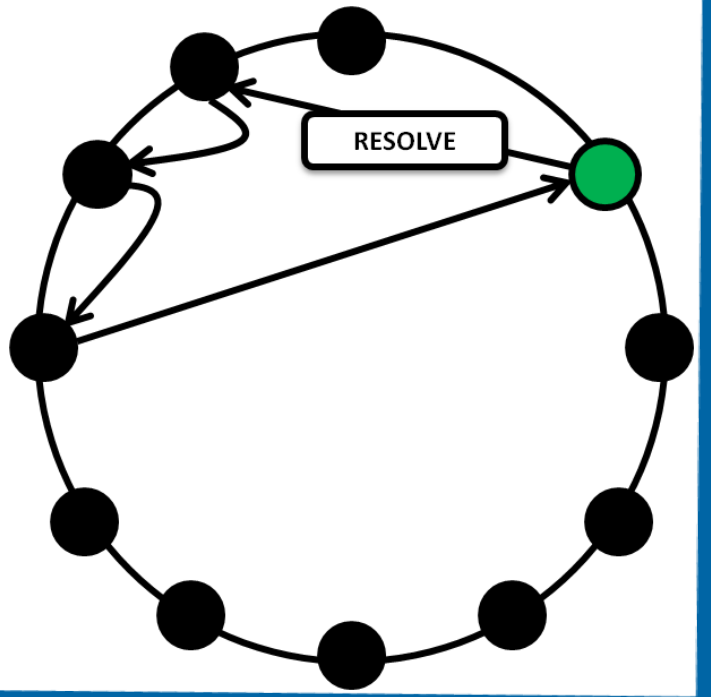
HOW DOES IT WORK?

- We can register things to ourselves
  - For example a sensor or actuator
- Universal Context Identifiers (UCI)
  - Similar to URL and email address combined
  - Ex. stefan@miun.se/temperature
- After registering a UCI
  - The whole DHT knows which UCI belongs to which node



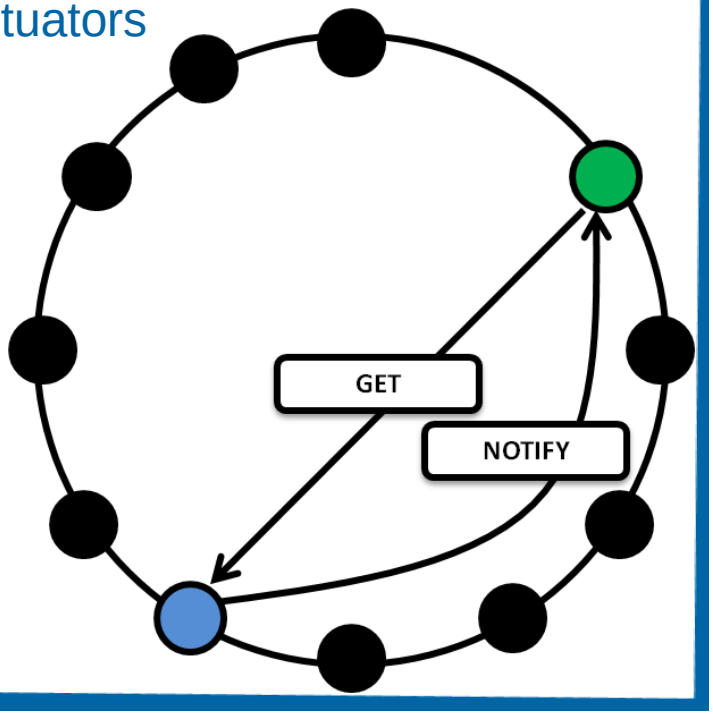
HOW DOES IT WORK?

- After registering, any other node can resolve the UCI
  - In order to find the node who owns a particular sensor or actuator
- The DHT will then translate
  - From UCI to Node address
- Quite similar to normal DNS
  - But without any centralized servers



HOW DOES IT WORK?

- When the node knows which other node owns a UCI
  - He can then set up a peer-to-peer session
  - For example to get sensor values or set actuators
- The primitives are
  - GET and SET
- But the platform also has
  - Subscriptions, Streaming, etc.



MIUN.SE
DISCOVER YOUR OPPORTUNITIES

The Code

Mittuniversitetet
MID SWEDEN UNIVERSITY

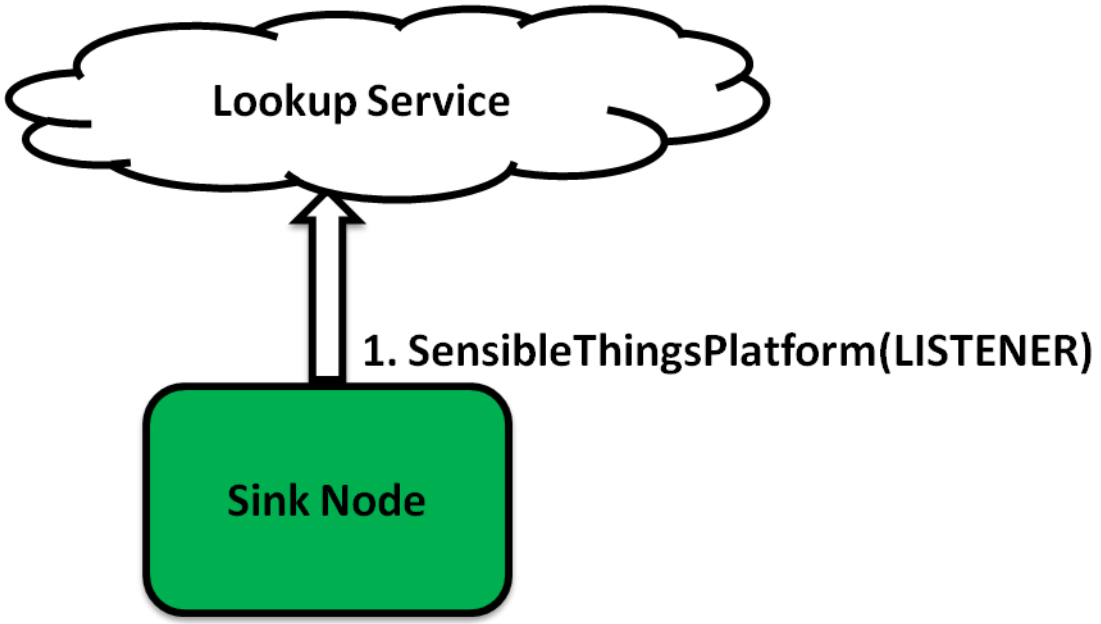
THE CODE

- The code is organized in layers (see paper for details)
  - Interface Layer
  - AddIn Layer
  - Dissemination Layer
  - Communication Layer
  - Sensor Layer

- But to create simple applications using the code
  - The only thing need to use is the interface layer
  - Called “SensibleThingsPlatform” in the code
  - For extended applications, you might want to use some add ins as well

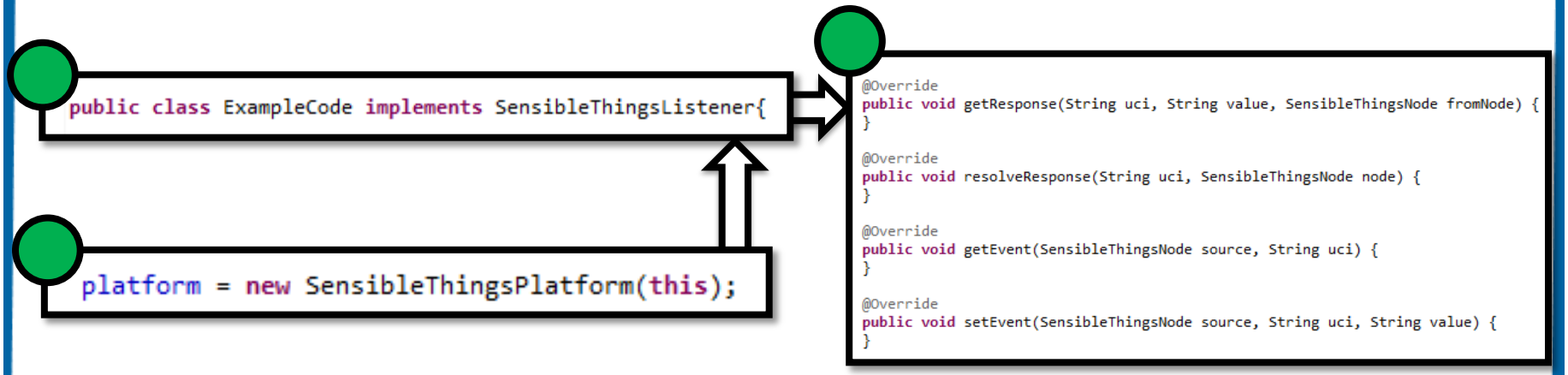
CODE TUTORIAL

- Illustrative figure for JOIN
  - 1. SensibleThingsPlatform(LISTENER)



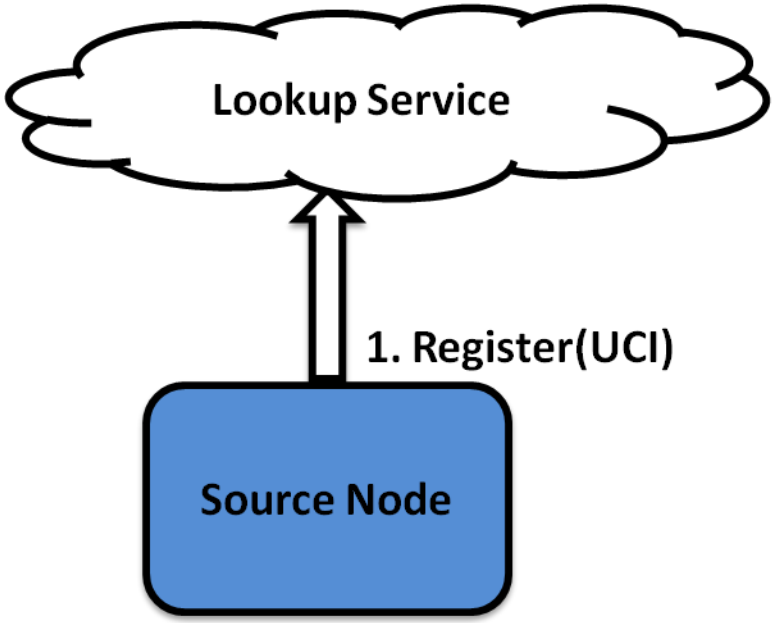
CODE TUTORIAL

- Implement SensibleThingsListener
  - To get callback functions
- Then start the platform using the constructor
  - With your SensibleThingsListener callback as the argument
  - This will use RUDP/DHT (or PROXY/DHT if behind NAT)



CODE TUTORIAL

- Illustrative figure for REGISTER
  - 1. Register(UCI)



CODE TUTORIAL

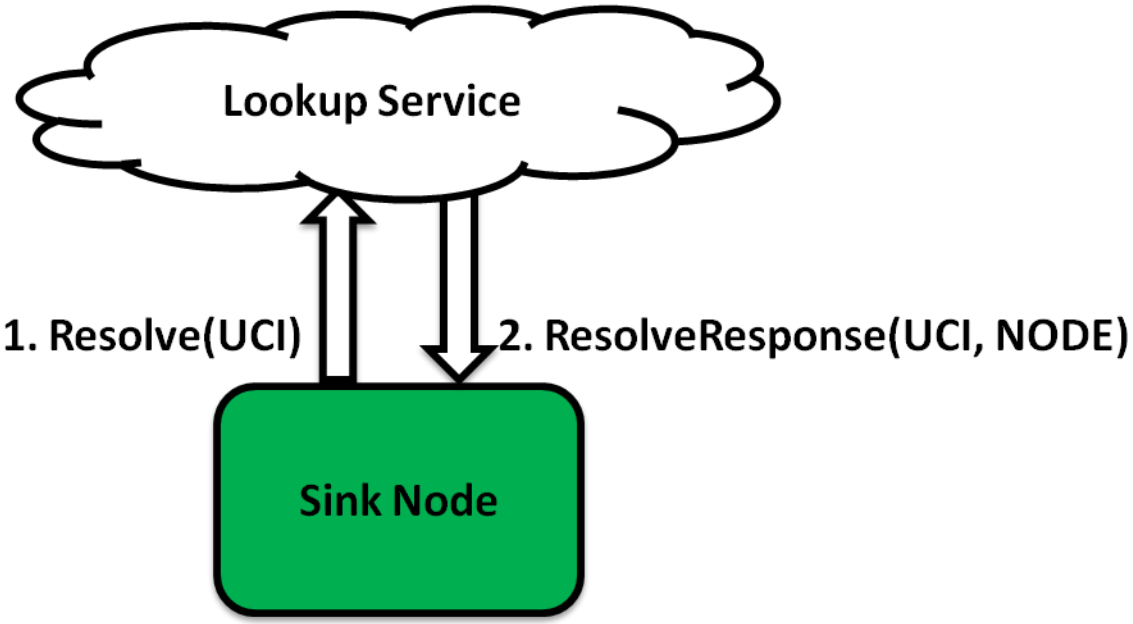
- Register a sensor using register

```
platform.register("example@miun.se/sensor");
```

- And it will now be available for resolving
  - By all nodes in the system

CODE TUTORIAL

- Illustrative figure for RESOLVE
  - 1. Resolve → 2. ResolveResponse



CODE TUTORIAL

- Resolve a sensor using resolve
  - Which will perform the lookup in the DHT

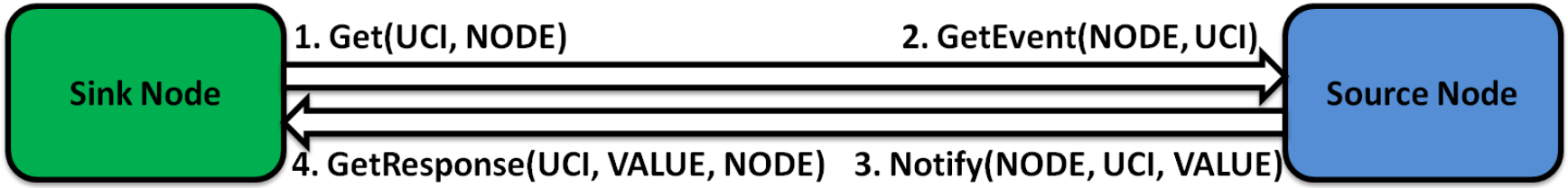
```
platform.resolve("example@miun.se/sensor");
```

- After the UCI have been resolved
  - It triggers the resolve response listener, with the results

```
@Override
public void resolveResponse(String uci, SensibleThingsNode node) {
}
```

CODE TUTORIAL

- Illustrative figure for GET
  - 1. Get → 2. GetEvent → 3. Notify → 4. GetResponse



CODE TUTORIAL

- After the resolve, we get the sensor value using get

```
platform.get(uci, node);
```

- This will trigger the GetEvent on the other side

```
@Override
public void getEvent(SensibleThingsNode source, String uci) {
```

- Send the sensor value back using Notify

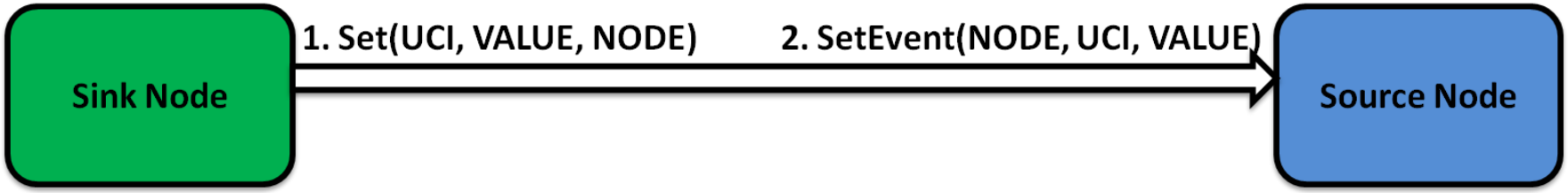
```
platform.notify(source, uci, value);
```

- Which triggers the GetResponse on the originating node

```
@Override
public void getResponse(String uci, String value, SensibleThingsNode fromNode) {
```

CODE TUTORIAL

- Illustrative figure for SET
  - 1. Set → 2. SetEvent



CODE TUTORIAL

- SET works in a similar manner

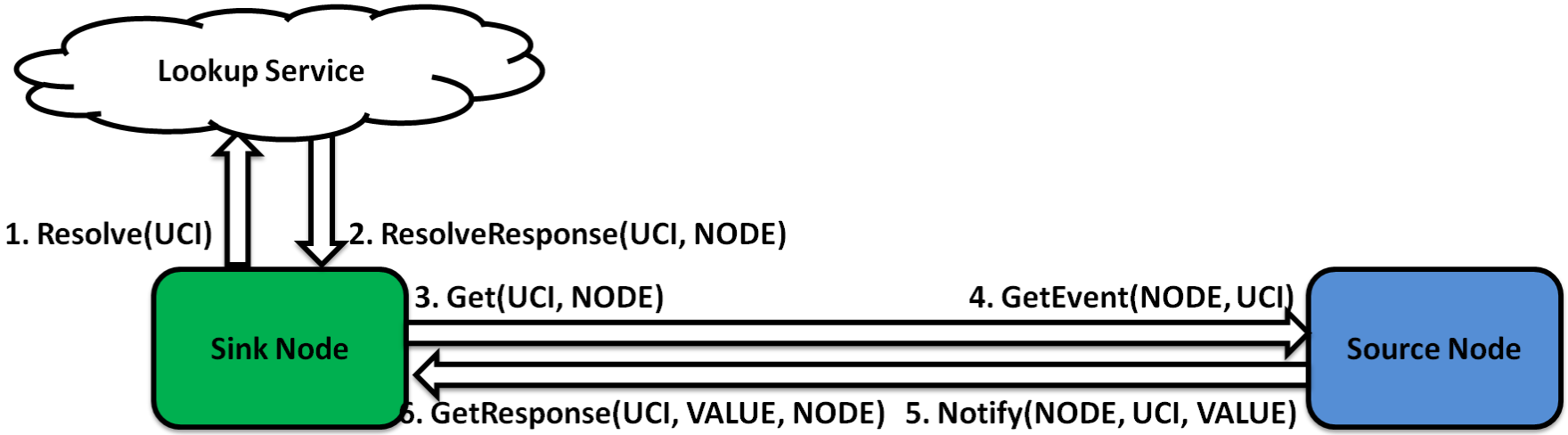
```
platform.set(uci, value, node);
```

- But triggers SetEvent on the other side instead

```
@Override
public void setEvent(SensibleThingsNode source, String uci, String value) {
}
```

CODE TUTORIAL

- Illustrative figure for a whole sequence
  - 1. Resolve → 2. ResolveResponse
  - 3. Get → 4. GetEvent → 5. Notify → 6. GetResponse



CODE TUTORIAL

- First we resolve

```
platform.resolve("example@miun.se/sensor");
```

- Which answers with a resolveResponse

```
public void resolveResponse(String uci, MediaSenseNode node) {
```

- Then we can call get

```
platform.get(uci, node);
```

- Which triggers a GetEvent on the other side

```
public void getEvent(SensibleThingsNode source, String uci) {
```

- The other side sends back a value with notify

```
platform.notify(source, uci, value);
```

- Which triggers a getResponse

```
public void getResponse(String uci, String value, SensibleThingsNode fromNode) {
```

MIUN.SE
DISCOVER YOUR OPPORTUNITIES

FAQ and Questions

Mittuniversitetet
MID SWEDEN UNIVERSITY

FAQ

- Problems with NAT, firewalls, etc.
  - All P2P systems generally have problems with NAT and firewalls
  - But in the latest versions of the platform
    - It will detect that you are behind NAT
    - And will tunnel you trough the NAT via a proxy
- This will effect performance
  - So for the pure experience, use public IP addresses
  - But this might be difficult, as most devices are behind NAT…
- The tunnel actually works quite good now
  - But it has problems with new messages types which it does know of
  - And ungraceful shutdowns takes ~30s to recover from

FAQ

- Add-ins are components that extend the platform
  - To add additional new functionality above the primitives
  - For example to enable caching, buffering, publish/subscribe, etc.
- All the add-ins use the add in manager
  - Simply create the add-in and new message types you want to use
  - And load it into the platform using the add-in manager
- If you want to create your own add-in
  - Simply create a class that implement either Extension or Optimization
  - And then create and load it as any other add-in
  - (Look at the source for the publish/subscribe add-in for a good example)

FAQ

- Encryption
  - The base version is unencrypted
    - All messages is sent is clear text
  - But it is possible to change to pure encrypted traffic
    - Simply use SSL or PROXY_SSL communication in the constructor to change
    - But the platform will be heavier to run and have a little worse response times

- Authentication
  - In the base version there is no real control over access control
    - Any other node can send you get/set-events
    - All you can do is observe the IP & port of the other node
  - But it is possible to enable simple authenticated access
    - Which uses a pre-shared-key scheme
    - See AuthenicationExtension for more details

MIUN.SE
DISCOVER YOUR OPPORTUNITIES

FAQ

- Java version
  - Current version is coded for Java 1.5
  - Which works on Android and other newer devices
- But it is possible to compile it to run on 1.4
  - Which is interesting for OSGI, and similar low end devices
  - To do this, you need to change some settings in Eclipse

MIUN.SE
DISCOVER YOUR OPPORTUNITIES

Thank You!

Contact Information
Lic. Eng. Stefan Forsström
E-Mail: stefan.forsstrom@miun.se

Mittuniversitetet
MID SWEDEN UNIVERSITY