Programming the Swarm

Using Accessors

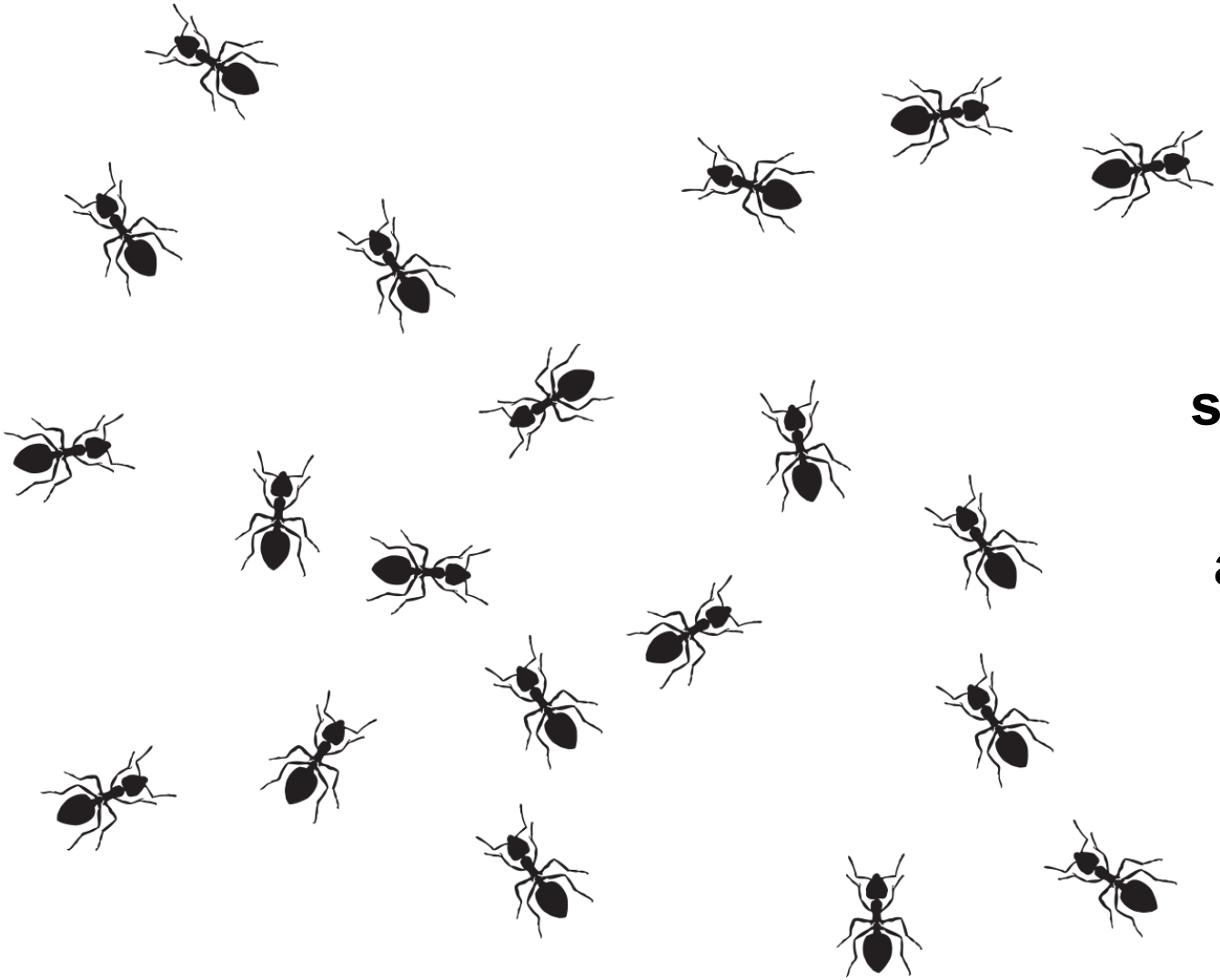
Marten Lohstroh
Graduate Student
UC Berkeley

Advised by Prof. Edward A. Lee

Invited Talk
At the Swarm Lab Retreat 2015

How do ‘things’ become ‘smart’?

Herbert A. Simon

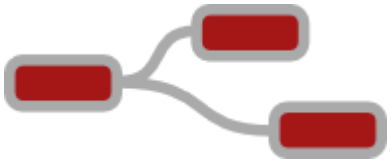


The sciences of the artificial (1996)

IoT: Heterogeneity

Standardization?

IEEE
Internet of Things



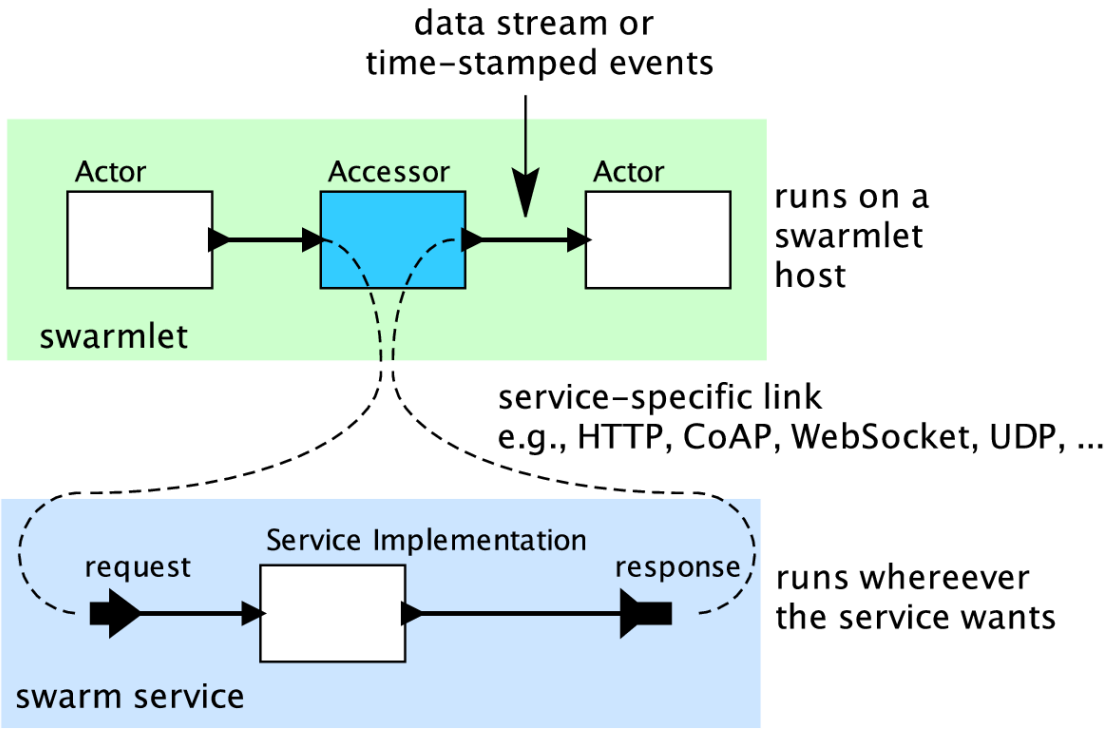
UPnP™

An Actor-based Design Pattern

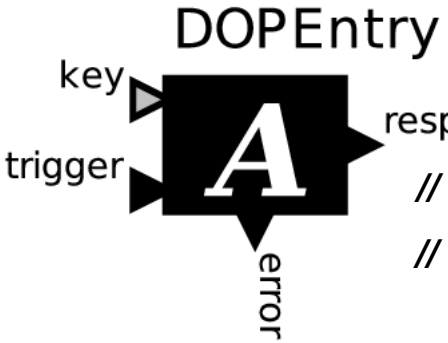
Accessors provide **access** to ***any*** **resource** that is reachable through an ***arbitrary*** **protocol** and exposes ***some*** **interface**

Accessors

- Wrap an existing service
- Export an actor interface
- Are composable with other actors
- Are executable on a accessor host

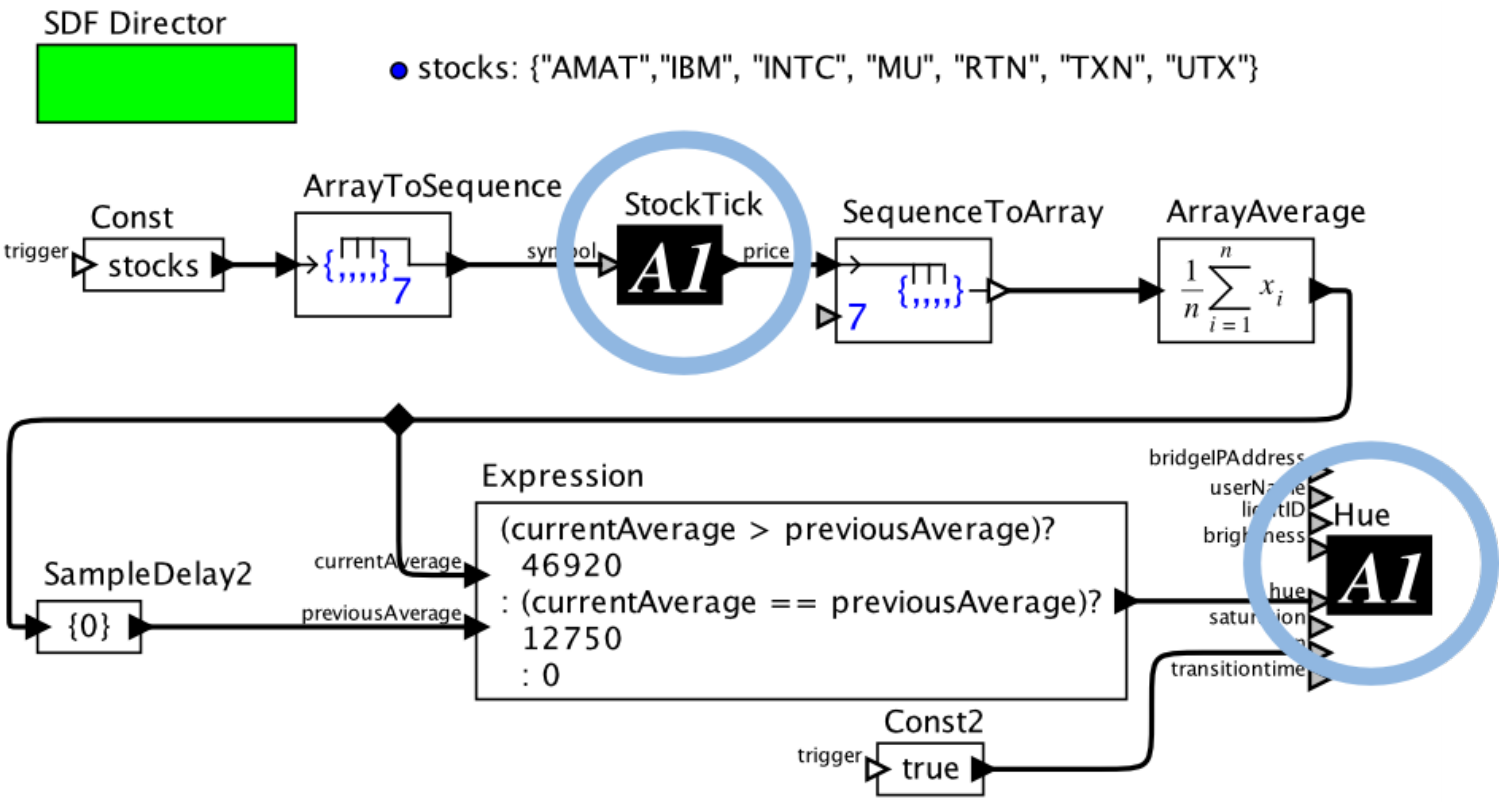


What an accessor looks like




```
// Accessor that opens the
// door of the DOP center
exports.setup = function() {
…
  accessor.input('trigger');
  accessor.output('response');
  accessor.input('key', {
     'type':'string',
     'value':'xxxxxxxx',
     'description':'Shared secret.'
   });
};
var http = require('httpClient');
var options = {
  'host':'xxx.xxx.xxx.xxx', 'port':8000,
  'protocol':'https',  'method':'GET',
  'path':'/unlock', 'headers':{},
  'keepAlive':false, 'query':'delay=1&key=',
  'trustAll':true
};
exports.initialize = function() {
  options.query += get('key');
}
exports.fire = function () {
    http.get(options, function (resp) {
       send(JSON.stringify(resp),
  'response');
    });
}
```

A “Hello world!”-example



See **www.terraswarm.org/accessors** for more examples.

The Accessor Host

Features

- **A scripting environment**

  This is an implementation of the base class of the Accessor.

  Our current prototypes only use JavaScript.

- **Models of Computation**

  E.g., Discrete Events, Dataflow, Rendez-vous.

  So far only supported in Ptolemy II.

- **An actor library**

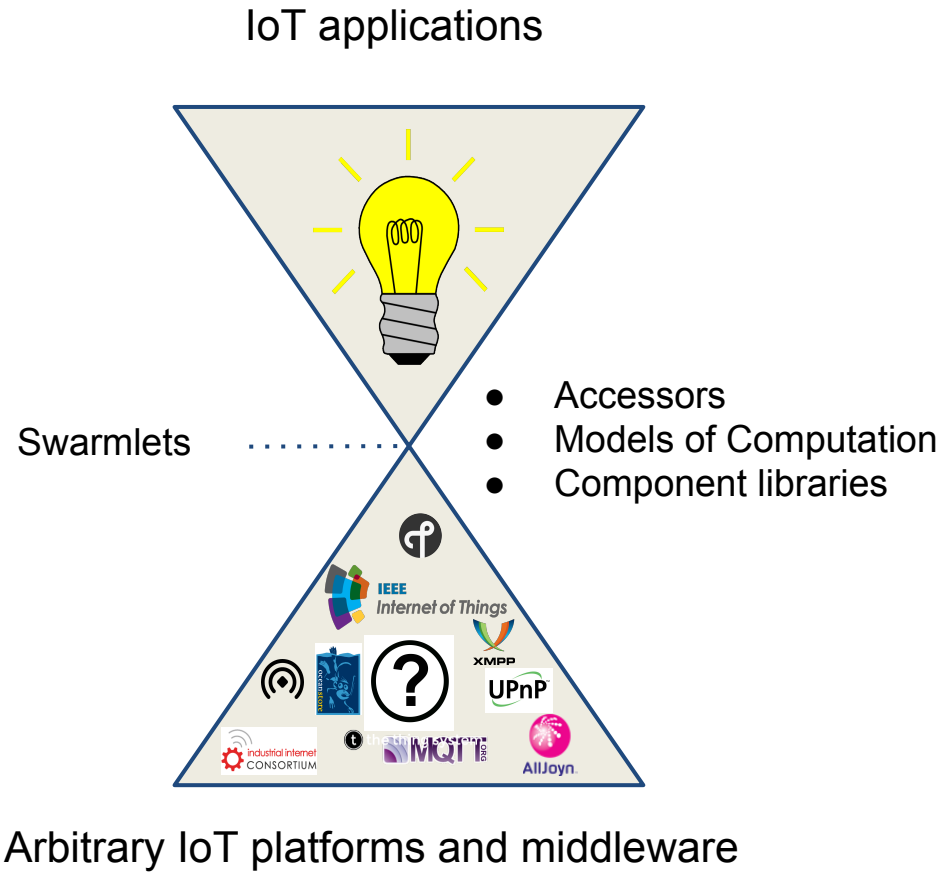
  Pre-installed trusted components.

The availability of these features determine the extend to which a Swarmlet hosts supports the requirements of an Accessor.

Prototypes

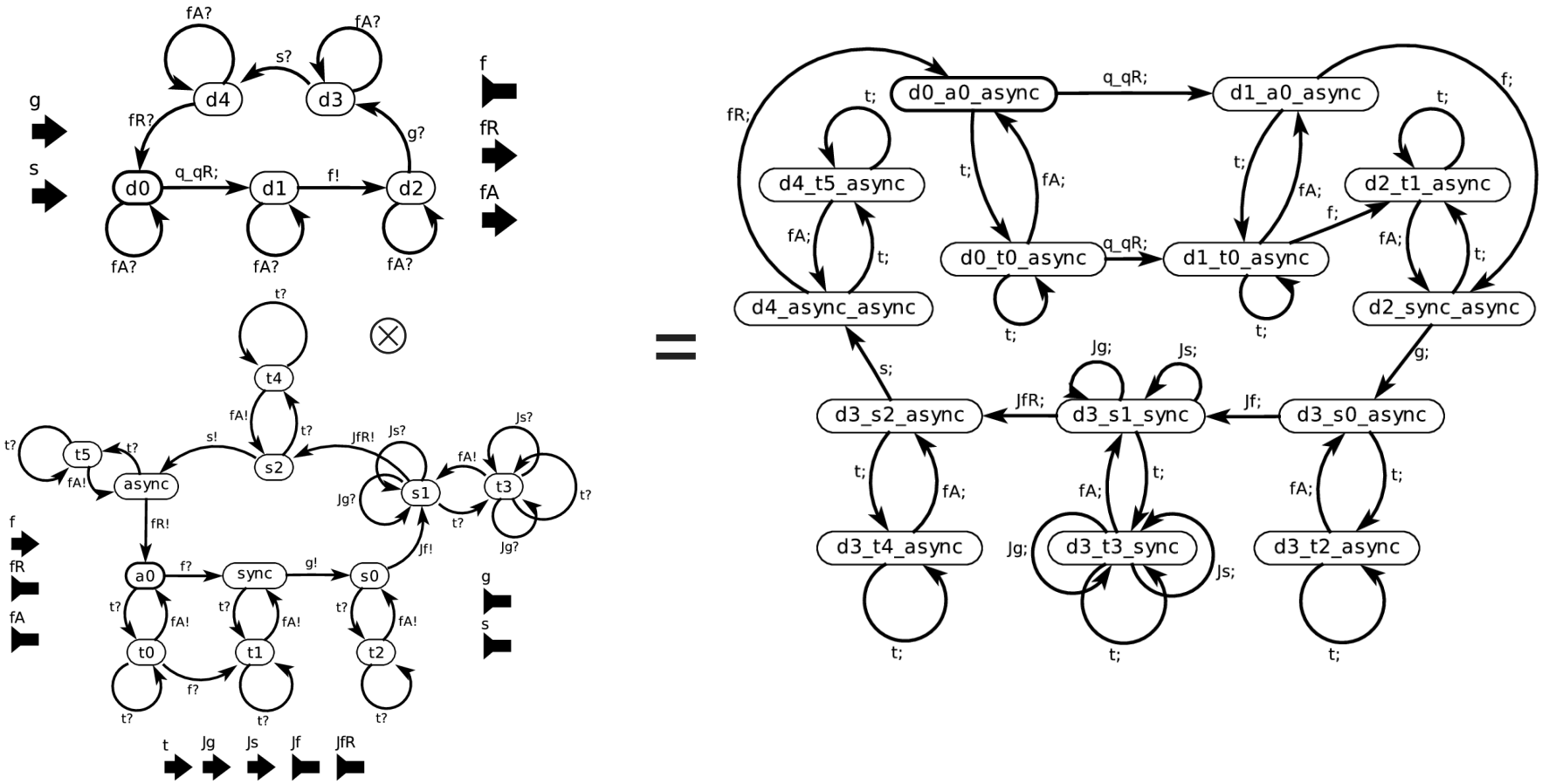
Platform-based Design

- The **Accessor host** is a universal platform for component- based design
- No one platform that rules them all, but one **platform** that incorporates them all
- **Accessors** can be provided by manufacturers as well as third parties
- Enable interoperability of **independently designed** components



Horizontal Contracts

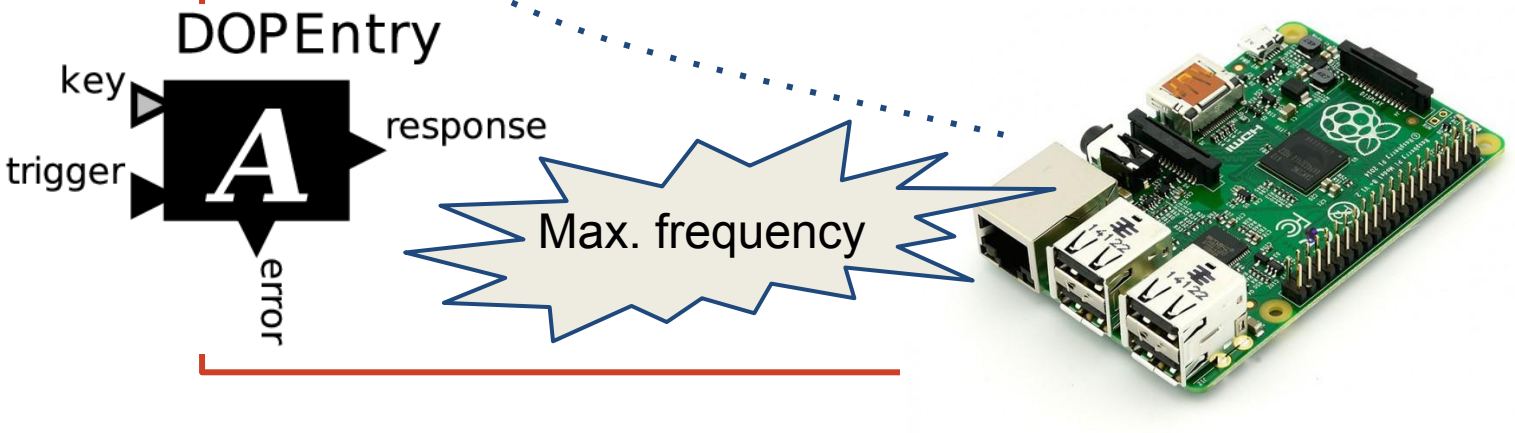
$$G(\neg fire \Rightarrow \neg output)$$



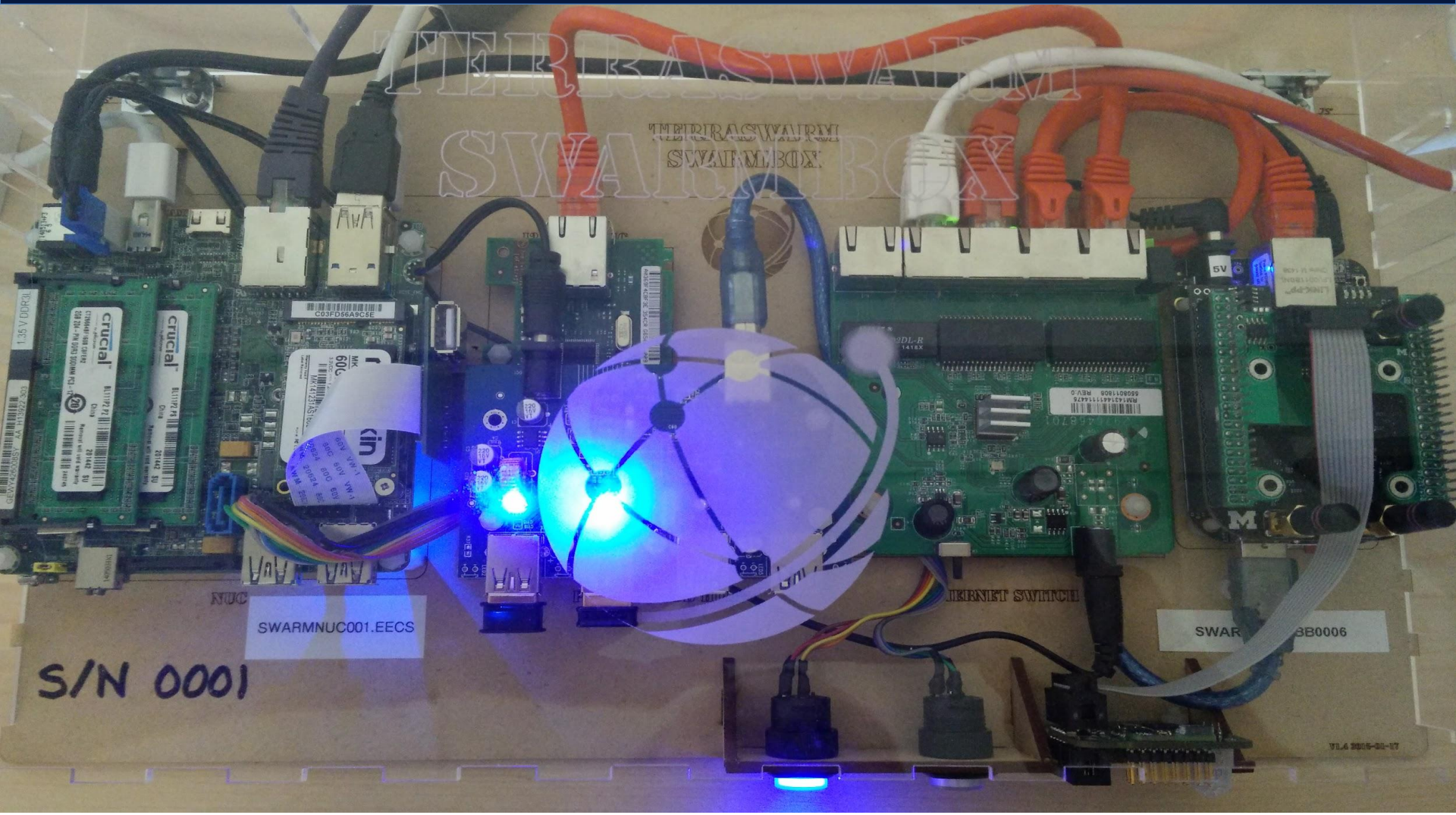
Vertical Contracts

- Modules provided by the host
- Protocol support
- Isolation properties
- Timing constraints
- …

Practical example: our door accessor



Accessing the Swarm



Future Work

- Interface
  - Subtyping?
  - Ontologies?
  - Contracts?
  - Discovery?
- Component
  - Languages?
  - Libraries?
  - Isolation?
  - Authentication?
- Composition
  - *Which MoC(s)?*
  - Time predictability?
  - Error handling?
- Host
  - Modules?
  - Deployment?

Questions