universität wien

Harnessing Untapped Potentials in the Internet of Things: Retrofitting and Upgradeability

Public Lecture Series: Sustainability in Computer Science

Kaspar Lebloch

Intro

Kaspar Lebloch BSc MSc
University Assistant Prae Doc and PhD Student
Research Group Cooperative Systems
Faculty of Computer Science
University of Vienna
kaspar.lebloch@univie.ac.at

universität wien

Agenda

1. UN SDG 12: Responsible Production and Consumption
2. Product Obsolescence and the legal Framework in the EU
3. Domain Specific Challenges of IoT Retrofitting
4. SerIoT: An Interface for Upgradeability-by-Default
5. Towards a Right to Improve

universität wien

1. UN SDG 12: Responsible Production and Consumption
2. Product obsolescence and the legal Framework in the EU
3. Domain Specific Challenges of IoT Retrofitting
4. SerIoT: An Interface for Upgradeability-by-Default
5. Towards a Right to Improve

https://sdgs.un.org/goals

https://sdgs.un.org/goals

11 Targets

https://sdgs.un.org/goals

11 Targets

Target **12.5:** “By 2030, substantially reduce waste generation through **prevention**, **reduction**, recycling and **reuse**”*

*https://sdgs.un.org/goals/goal12#targets_and_indicators

E-Waste

"In 2019, the amount of e-waste generated was **7.3 kg** per capita, with only 1.7 kg per capita documented to be managed in an environmentally sustainable manner. […]*"

https://sdgs.un.org/goals/goal12#progress_and_info

E-Waste

"[…] E-waste generation is expected to grow by 0.16 kg per capita annually to reach **9 kg** per capita in 2030."*

*https://sdgs.un.org/goals/goal12#progress_and_info

universität wien

Prevention, Reduction, Recycling and Reuse

- Design for durability
- Favor repair over replacement
  - In Design
  - In Use
- Beyond repair: **Improvement**
  - Upgrade
  - Retrofit
- Anticipate and encourage **repurposing**

- How are these goals currently enforced?

universität wien

1. UN SDG 12: Responsible Production and Consumption
2. **Product Obsolescence and the legal Framework in the EU**
3. Domain Specific Challenges of IoT Retrofitting
4. SerIoT: An Interface for Upgradeability-by-Default
5. Towards a Right to Improve

universität wien

Product Obsolescence in Household Appliances

- Obsolescence: ”The process or fact of becoming obsolete or outdated, or of falling into disuse.”*
- Product obsolescence is **inevitable**
  - Durability of materials and parts
  - Technological advancement: Safety, efficiency, features, …
  - Fashion

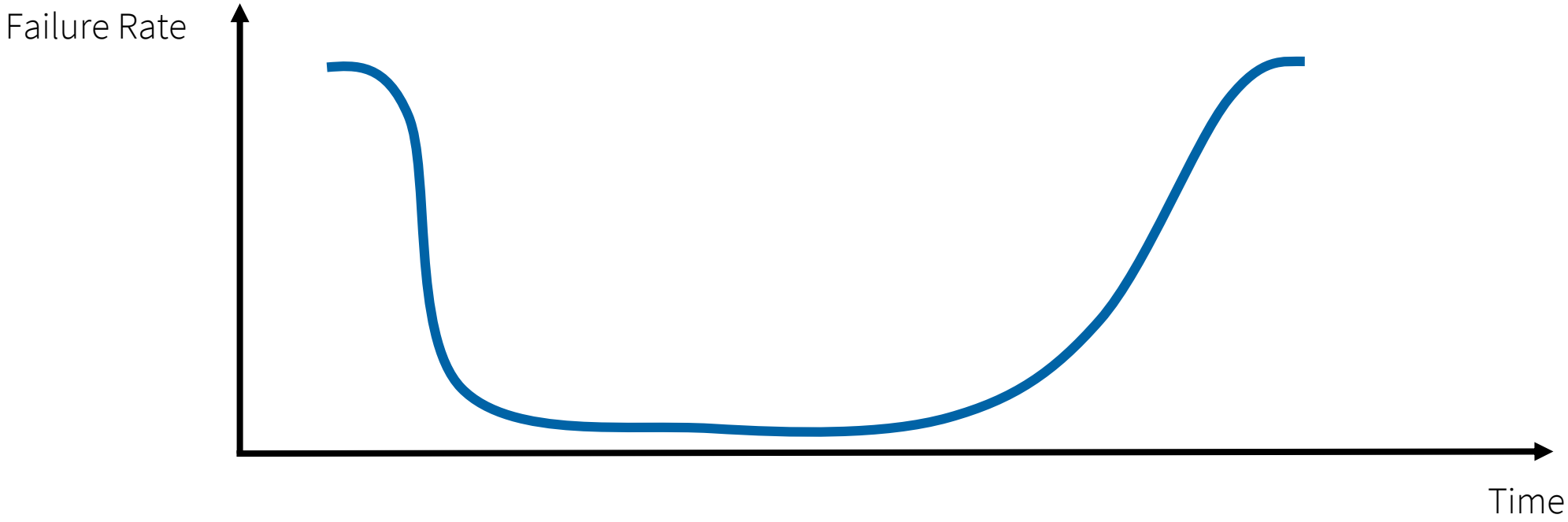
→ Design for obsolescence: efficient use of resources

- Planned obsolescence: premature failure due to deliberate design flaws
- New phenomenon: Software-based obsolescence
  - Lack of (security) updates

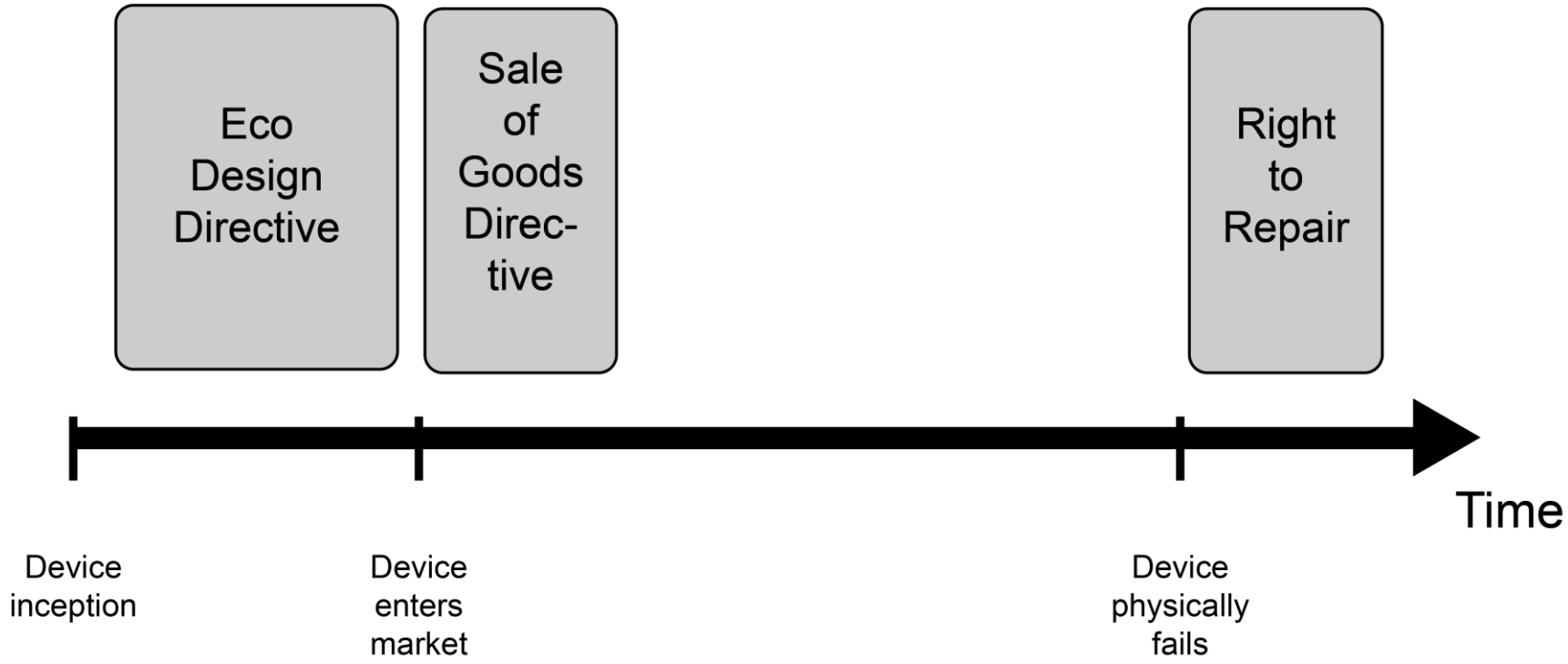
* Per the Oxford Dictionary

universität wien

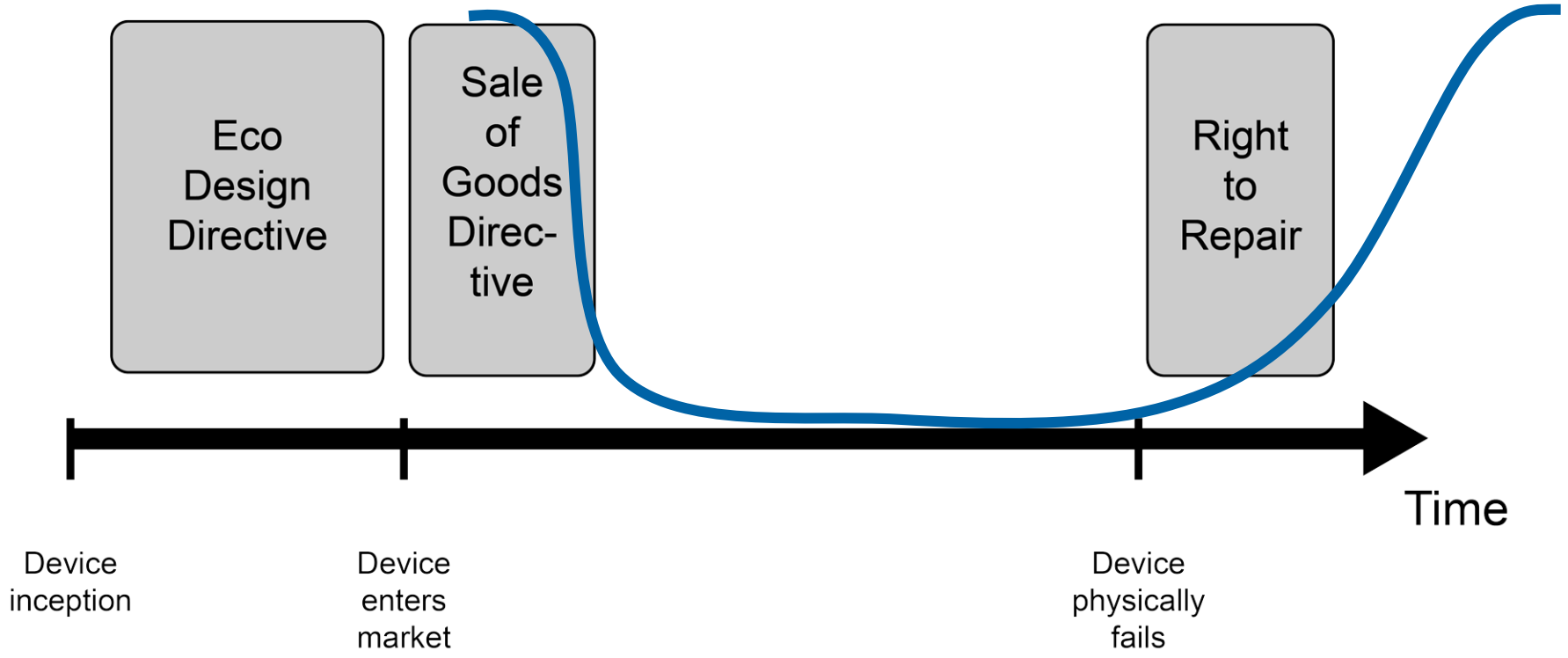
The Bathtub Curve of Device Failure Rate over Time



The Legal Framework in the EU



The Legal Framework in the EU



universität wien

Takeaways

- Obsolescence is inevitable and designed for
  - Failure rates / obsolescene incidence can be modeled with the „Baththub curve“
- The Ecodesign Directive
  - Mandates parameters for appliance design
  - Applies to specific product categories only
- The Sale of Goods Directive
  - Protects consumers for two years after a sale
  - Concerns vendors, not manufacturers
- The Right to Repair
  - Is yet to come and will only apply to Ecodesign Directive categories
  - Entitles consumers to receive a repair service

universität wien

The legal framework handles product obsolescence in the EU. How can Computer Science contribute beyond that?

universität wien

1. UN SDG 12: Responsible Production and Consumption
2. Product Obsolescence and the legal Framework in the EU
3. Domain Specific Challenges of IoT Retrofitting
4. SerIoT: An Interface for Upgradeability-by-Default
5. Towards a Right to Improve

universität wien

The (Residential) Internet of Things

- „Network of physical objects—“things”—that are embedded with sensors, software, and other technologies for the purpose of connecting and exchanging data with other devices and systems over the internet”*
- Residential IoT: Smart Home
  - „Smart“ Household appliances
  - Remote control and monitoring
  - Home automation
  - Competing standards
  - Competing manufacturers
  - Selective cross-integrations

*https://www.oracle.com/internet-of-things/what-is-iot/#:~:text=What%20is%20IoT%3F,and%20systems%20over%20the%20internet.

universität wien

IoT Retrofitting – what is it, and why?

- Outfitting existing, often unprepared appliances with IoT Technology
  - Network modules
  - Sensors
  - Actuators
- Observed mostly in Industrial context
  - Fabrication machinery
  - Industry 4.0
  - Appliances are significant investment
  - Delaying obsolescence motivated from economic view
- Consumer-grade IoT retrofitting
- Goal: **Prolong the use phase** of an appliance as opposed to replacing it with a new, smart one

universität wien

Domain Specific Challenges of IoT Retrofitting: Three Case Studies

- Domains:
  1. Physical
  2. Electronic
  3. Digital

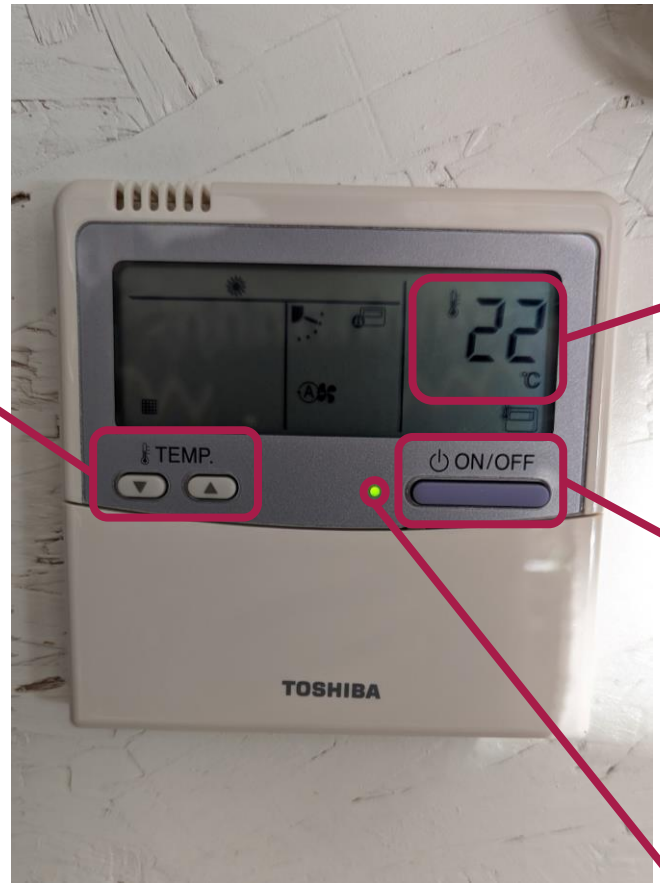
- Preconditions: Test subject is a motivated computer scientist or developer, **not a domain expert**.
- Method: **Autoethnography** of the process (self, supervised, and online-sourced)
- Limitations: the challenges are only the ones **we identified** for the target demographic (there may be many more)

Case Study I: Physical IoT Retrofitting

- Physical user interface components:
  - Buttons, levers, knobs and gears
- Target: Heating control panel
- Motivation: Conserve energy (fix inefficient programming)

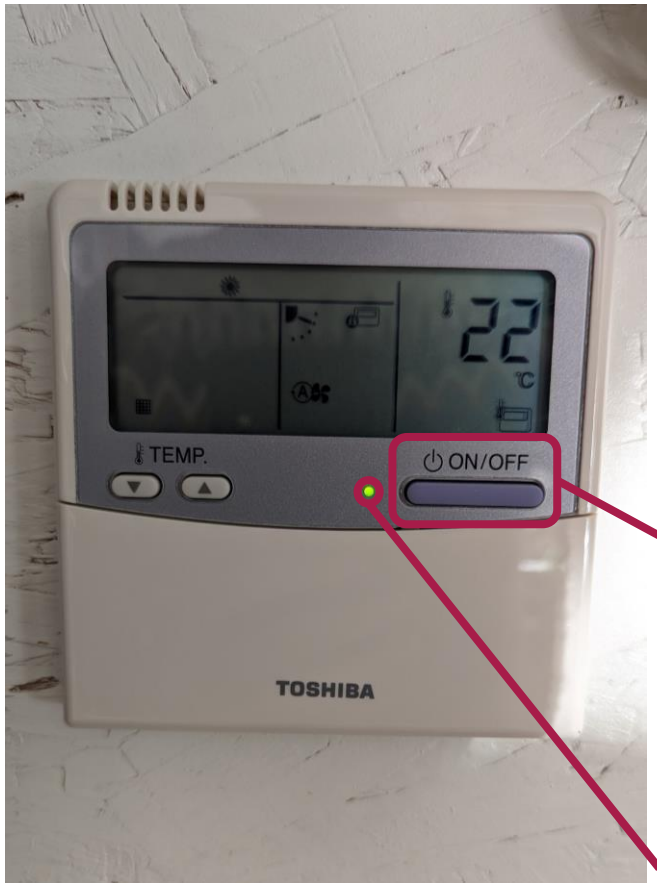
Case Study I: Physical IoT Retrofitting

- Steps:
  1. Analyze the interface ←
  2. Identify necessary elements
  3. Find compatible hardware
  4. Build custom solution (HW and SW)



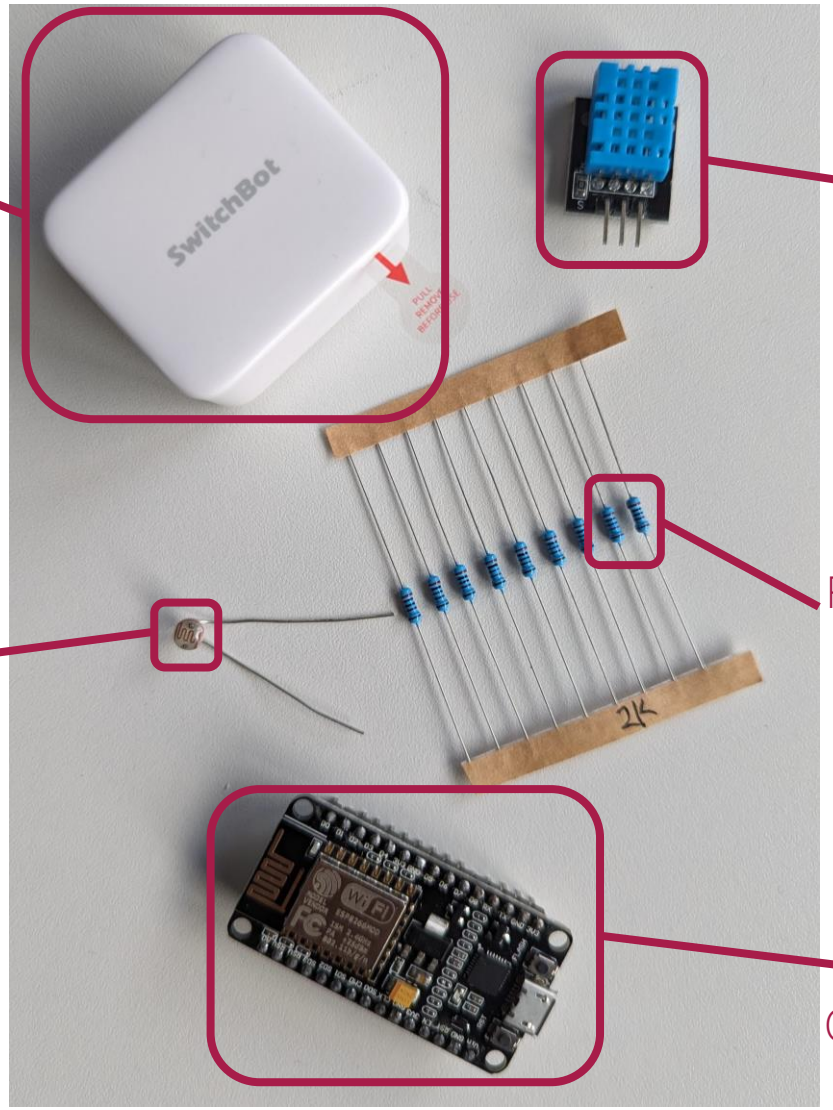
Case Study I: Physical IoT Retrofitting

- Steps:
  1. Analyze the interface
  2. Identify necessary elements ←
  3. Find compatible hardware
  4. Build custom solution (HW and SW)

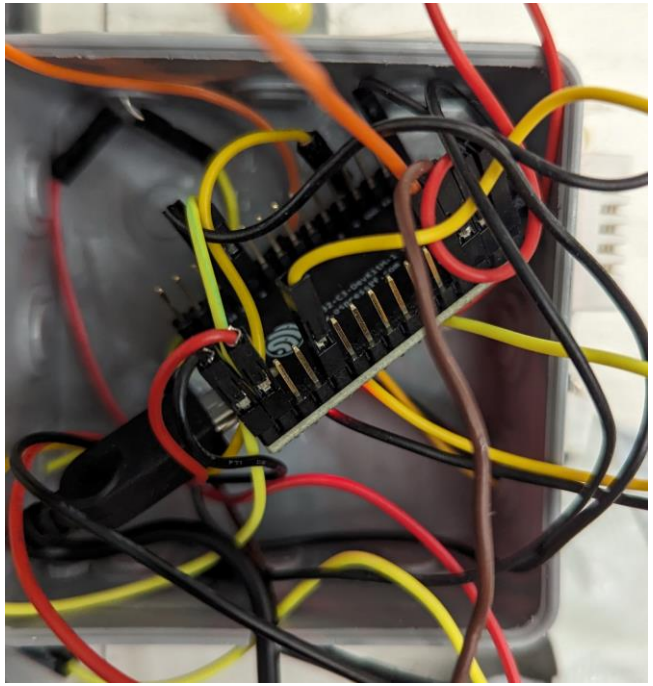


Case Study I: Physical IoT Retrofitting

- Steps:
  1. Analyze the interface
  2. Identify necessary elements
  3. Find compatible hardware ←
  4. Build custom solution (HW and SW)

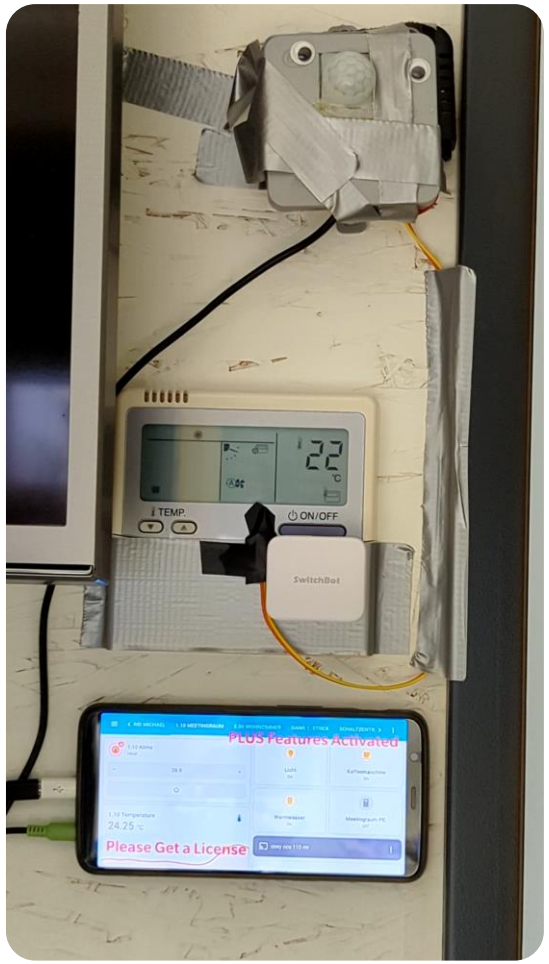


Case Study I: Physical IoT Retrofitting



- Steps:
  1. Analyze the interface
  2. Identify necessary elements
  3. Find compatible hardware
  4. Build custom solution (HW and SW) ←

Case Study I: Physical IoT Retrofitting



- Steps:
  1. Analyze the interface
  2. Identify necessary elements
  3. Find compatible hardware
  4. Build custom solution (HW and SW)
- Result:
  - A complex prototype consisting of multiple interlinked HW components

universität wien

Case Study I: Domain Specific Challenges

- Complexity
  - Requires creative cross-domain approach
  - Requires multiple different HW and SW components
- Functional impact
  - Solution may obstruct original UI
- Aesthetic impact
  - Solution impairs appearance of appliance
- Specific solution
  - Solution has to be custom fitted to appliance
  - Solution is not functionally generic

universität wien

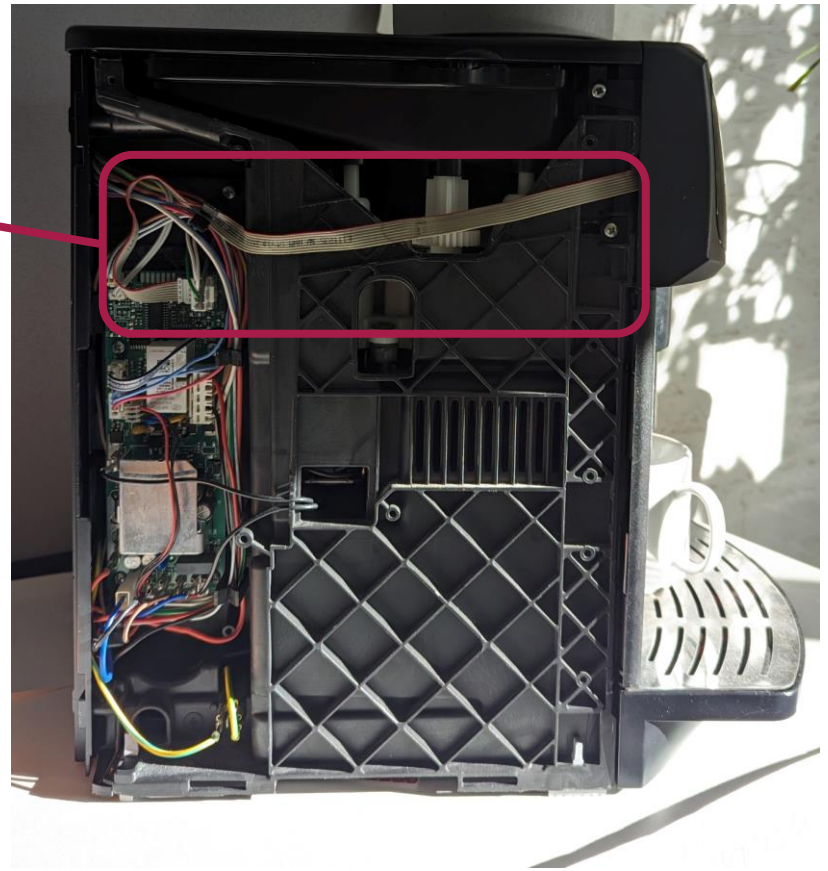
Case Study II: Electronic IoT Retrofitting

- Electronic interfaces:
  - Electrical contacts, interception of voltages
- Target: Espresso maker
- Motivation: Extend feature set, delay obsolescence

⚠⚠ Don‘t try this at home! ⚠⚠

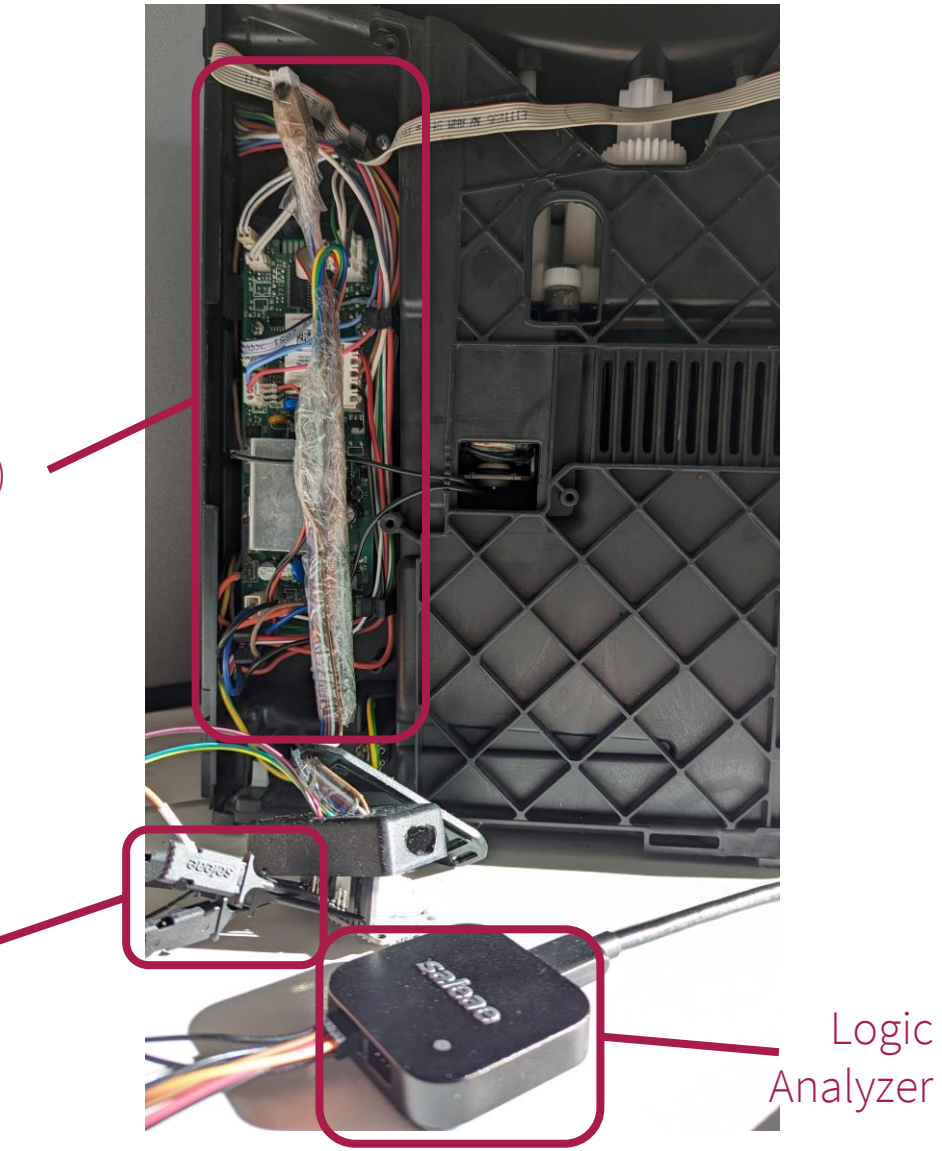
Case Study II: Electronic IoT Retrofitting

- Steps:
  1. Find suitable entry point ←
  2. Measure and document electronic signals
  3. Reverse engineer meaning of electronic signals
  4. Reproduce electronic signals
  5. Build custom SW solution



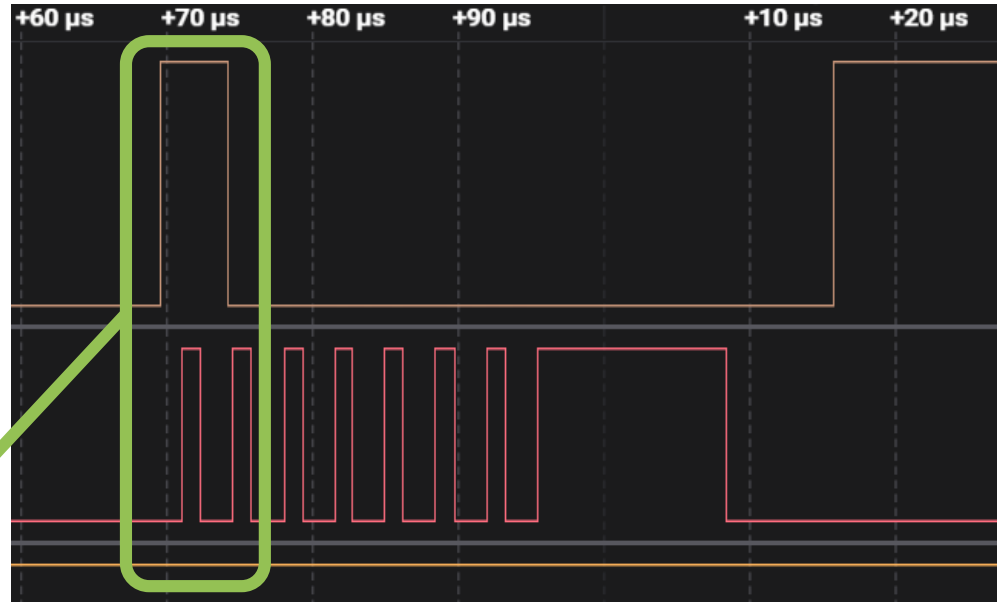
Case Study II: Electronic IoT Retrofitting

- Steps:
  1. Find suitable entry point
  2. Measure and document electronic signals ←
  3. Reverse engineer meaning of electronic signals
  4. Reproduce electronic signals
  5. Build custom SW solution



Case Study II: Electronic IoT Retrofitting

- Steps:
  1. Find suitable entry point
  2. Measure and document electronic signals
  3. Reverse engineer meaning of electronic signals ←
  4. Reproduce electronic signals
  5. Build custom SW solution



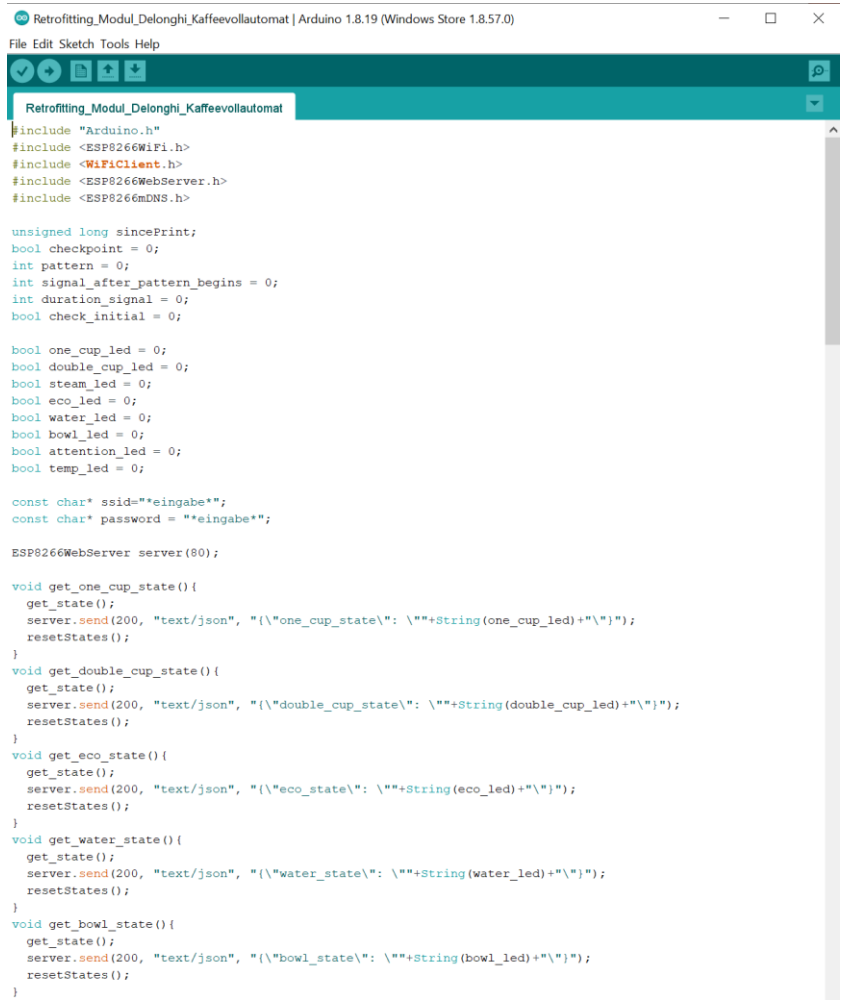
Single espresso button pressed

Case Study II: Electronic IoT Retrofitting

- Steps:
  1. Find suitable entry point
  2. Measure and document electronic signals
  3. Reverse engineer meaning of electronic signals
  4. Reproduce electronic signals ←
  5. Build custom SW solution

Case Study II: Electronic IoT Retrofitting

- Steps:
  1. Find suitable entry point
  2. Measure and document electronic signals
  3. Reverse engineer meaning of electronic signals
  4. Reproduce electronic signals
  5. Build custom SW solution ←



Case Study II: Electronic IoT Retrofitting

- Steps:
  1. Find suitable entry point
  2. Measure and document electronic signals
  3. Reverse engineer meaning of electronic signals
  4. Reproduce electronic signals
  5. Build custom SW solution
- Result:
  - A software prototype for ESP microcontrollers

universität wien

Case Study II: Domain Specific Challenges

- Measureability
  - Requires specialized hardware (oscilloscope/logic analyzer)
- Reverse engineering
  - Requires patience and precision
- Signal reproducability
  - Solution components have to be able to reproduce original signal (analog and digital)
- Specific solution
  - Software is specific to appliance (model)

universität wien

Case Study III: Digital IoT Retrofitting (Extending Work by Jirků*)

- Digital Interfaces: Serial ports, wireless ports, APIs
- Target: Soundbar speaker
- Motivation: Add Features, delay obsolescence

https://at.yamaha.com/de/products/audio_visual/sound_bar/yas-207/index.html

*see also https://wejn.org/tags/#yas_207

Case Study III: Digital IoT Retrofitting

- Steps:
  1. Identify interface ←
  2. Intercept communication
  3. Reverse engineer protocol
  4. Build custom SW solution

Bluetooth Classic ←→ BLE

universität wien

Case Study III: Digital IoT Retrofitting

- Steps:
  1. Identify interface
  2. Intercept communication ←
  3. Reverse engineer protocol
  4. Build custom SW solution

universität wien

Case Study III: Digital IoT Retrofitting

- Steps:
  1. Identify interface
  2. Intercept communication
  3. Reverse engineer protocol ←
  4. Build custom SW solution

universität wien

Case Study III: Digital IoT Retrofitting

- Steps:
  1. Identify interface
  2. Intercept communication
  3. Reverse engineer protocol
  4. Build custom SW solution ←

universität wien

Case Study III: Digital IoT Retrofitting

- Steps:
  1. Identify interface
  2. Intercept communication
  3. Reverse engineer protocol
  4. Build custom SW solution
- Result:
  - Software prototype for any platform depending on programming language

universität wien

Case Study III: Domain Specific Challenges

- Interception of communication
  - Requires product adherence to documented standards
  - Requires compatible hardware (easy to come by)
- Protocol reverse engineering
  - Requires specialized programming skills or example
  - Requires protocol to not be encrypted
- Genericity of Result
  - Solution not generic due to prioritization of own use case & lack of standardization (but appropriable)

universität wien

Takeaways:

- IoT retrofitting often **requires specialized hardware and skills**
- Different domains for retrofitting projects have different requirements
- **Physical**:
  - Cross-domain thinking
  - Different sensors/actuators
- **Electronic**:
  - Specialized hardware
  - Signal reproducability
- **Digital**:
  - Protocol readability
  - Standard adherence
- **Global challenge: Solutions lack genericity**

universität wien

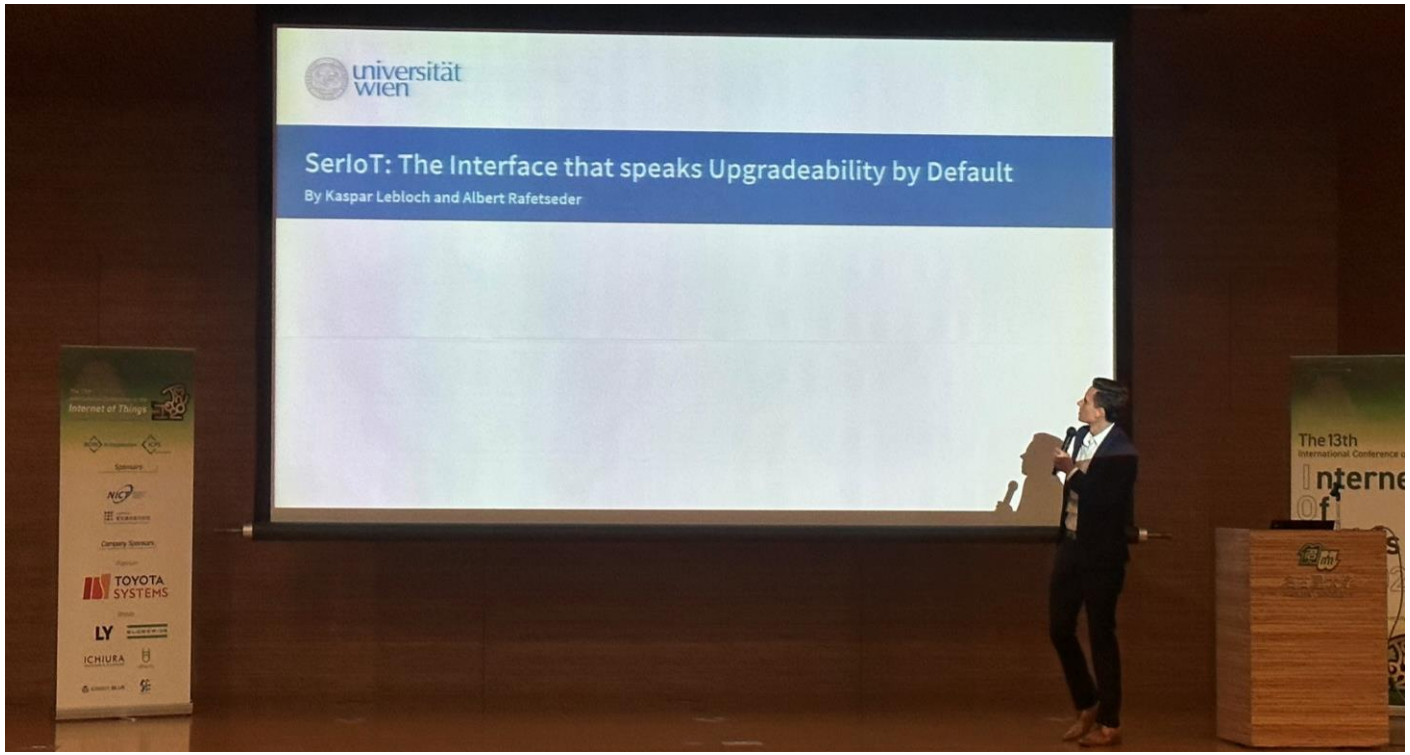
We established: IoT retrofitting of legacy hardware is challenging. But what about current IoT devices?

universität wien

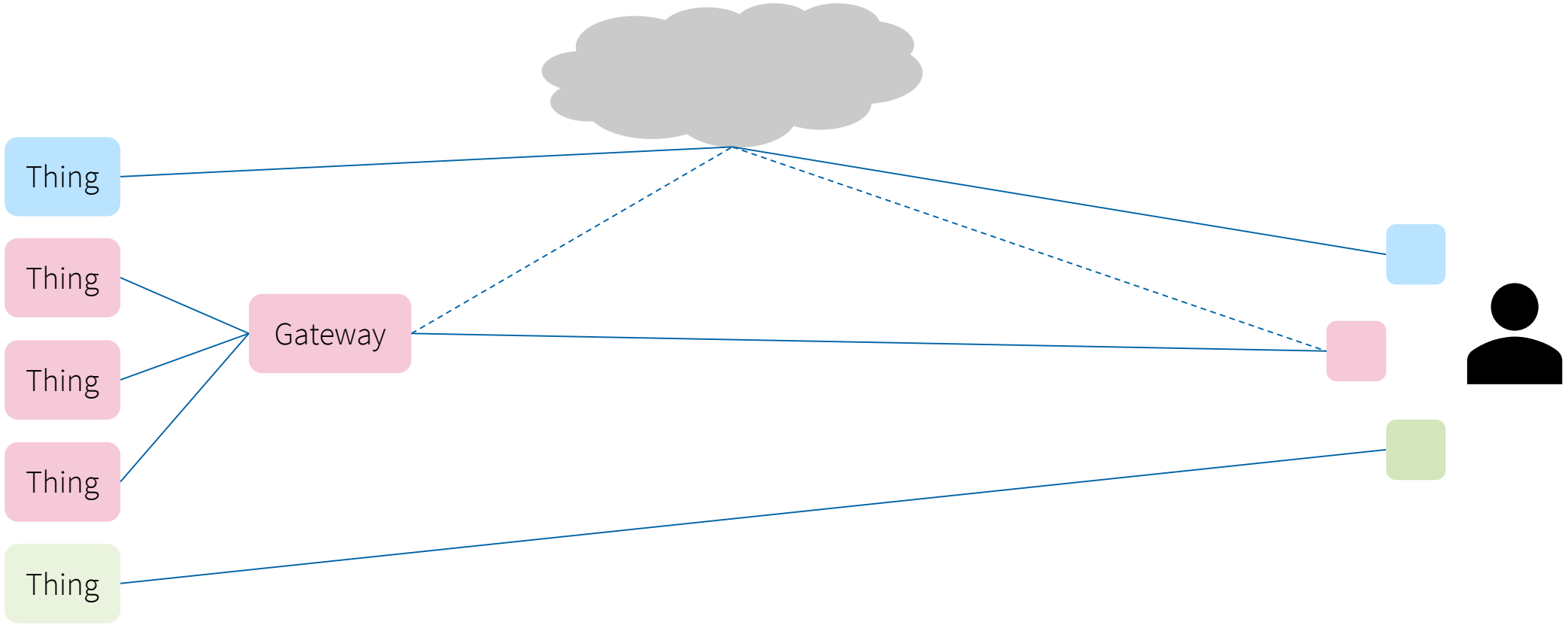
1. UN SDG 12: Responsible Production and Consumption
2. Product Obsolescence and the legal Framework in the EU
3. Domain Specific Challenges of IoT Retrofitting
4. SerIoT: An Interface for Upgradeability-by-Default
5. Towards a Right to Improve

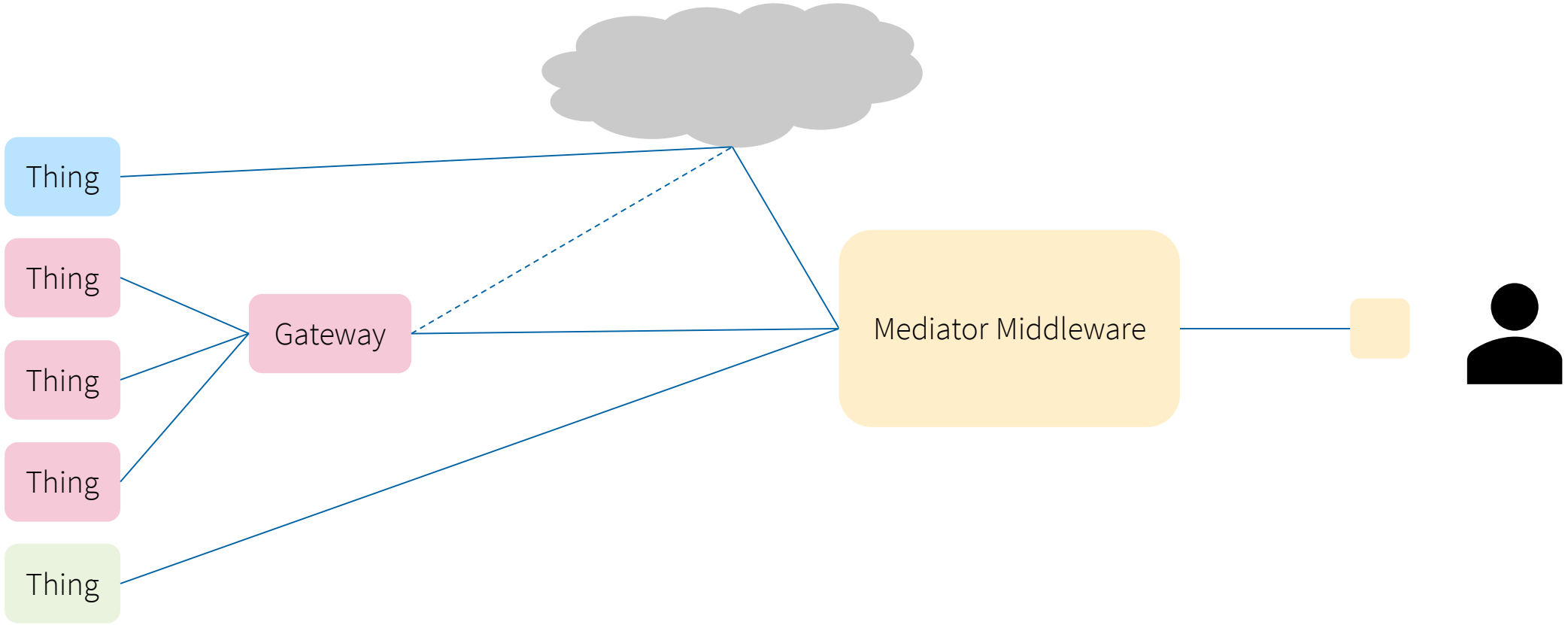
universität wien

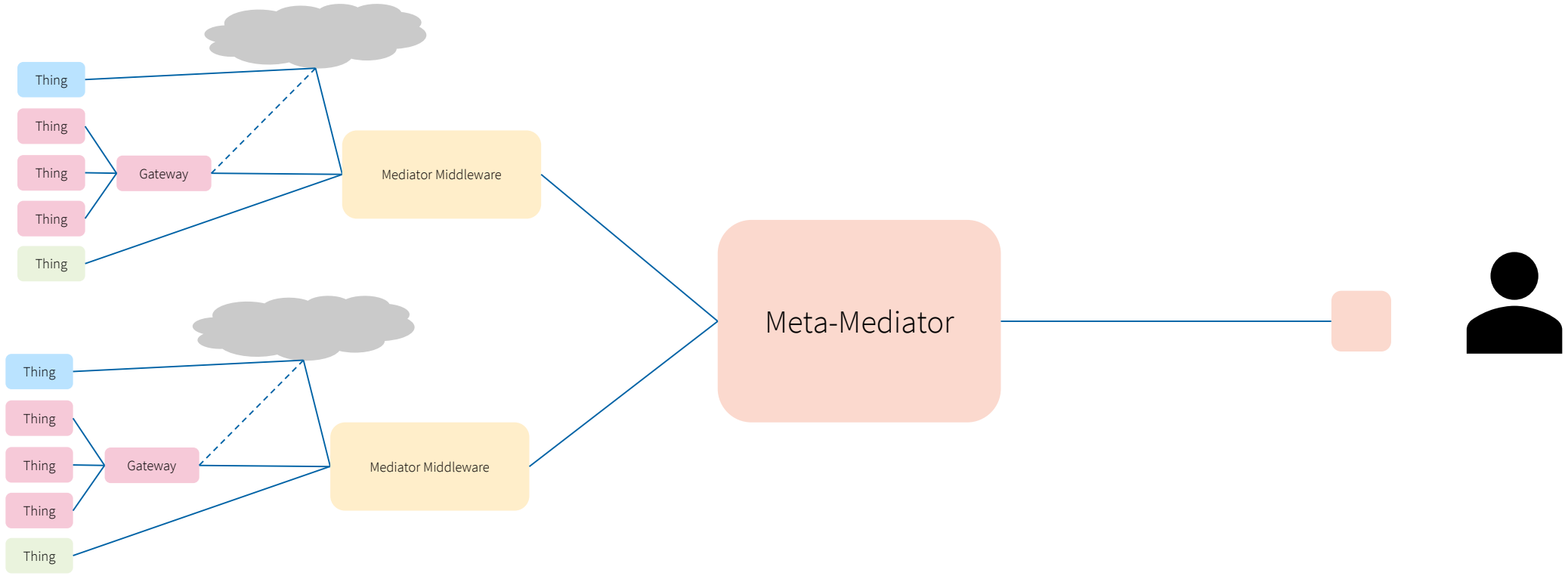
SerIoT: The Interface that speaks Ugradeability by Default

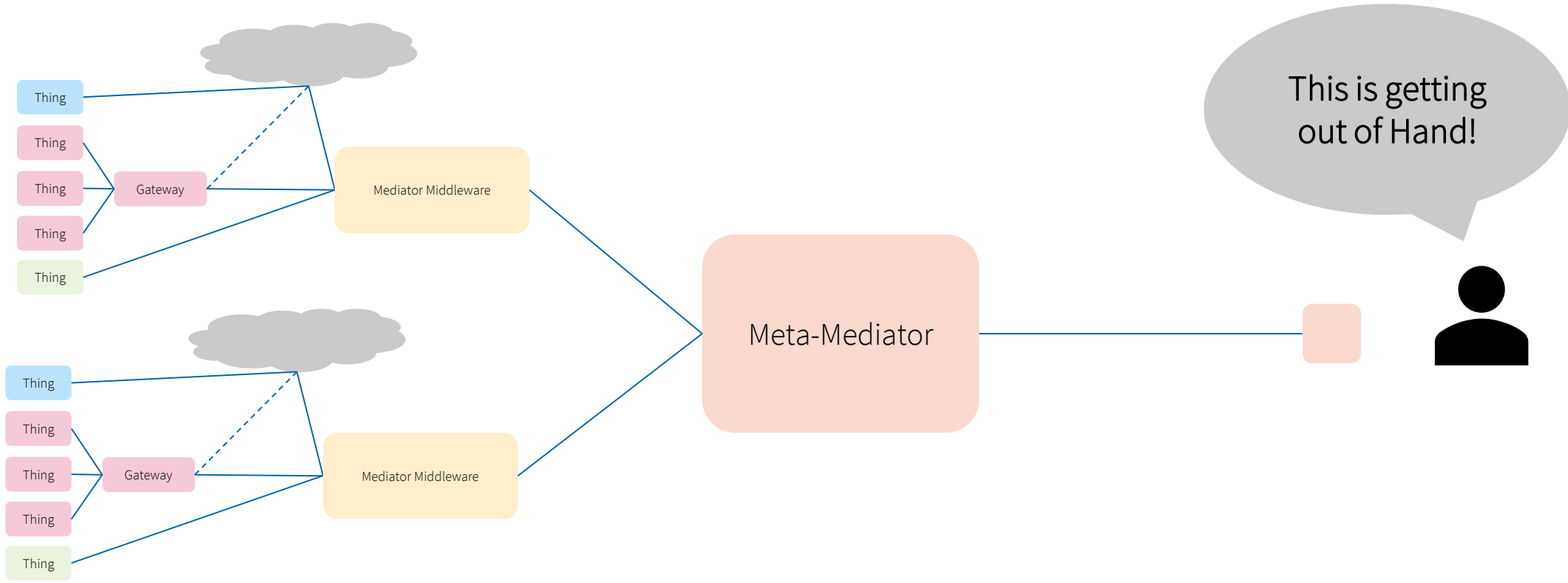


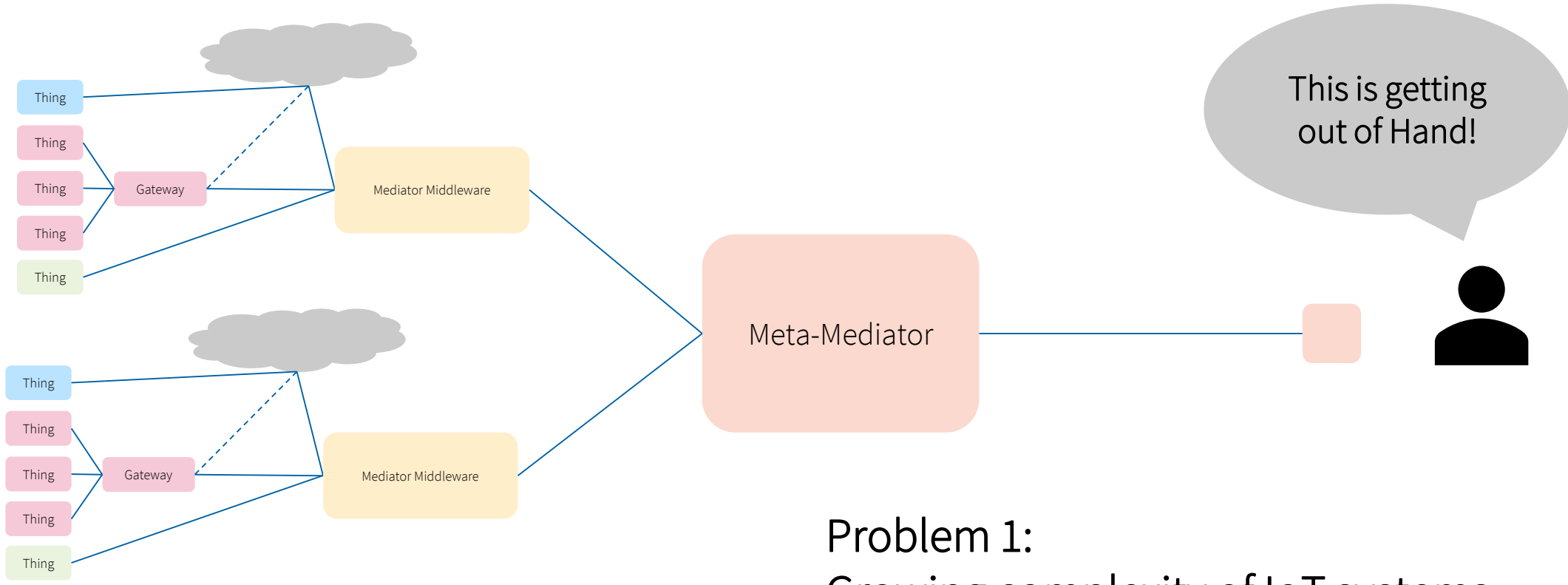
The corresponding paper will be available in the ACM-DL soon.


Thing
Thing
Thing
Thing
Thing
Gateway

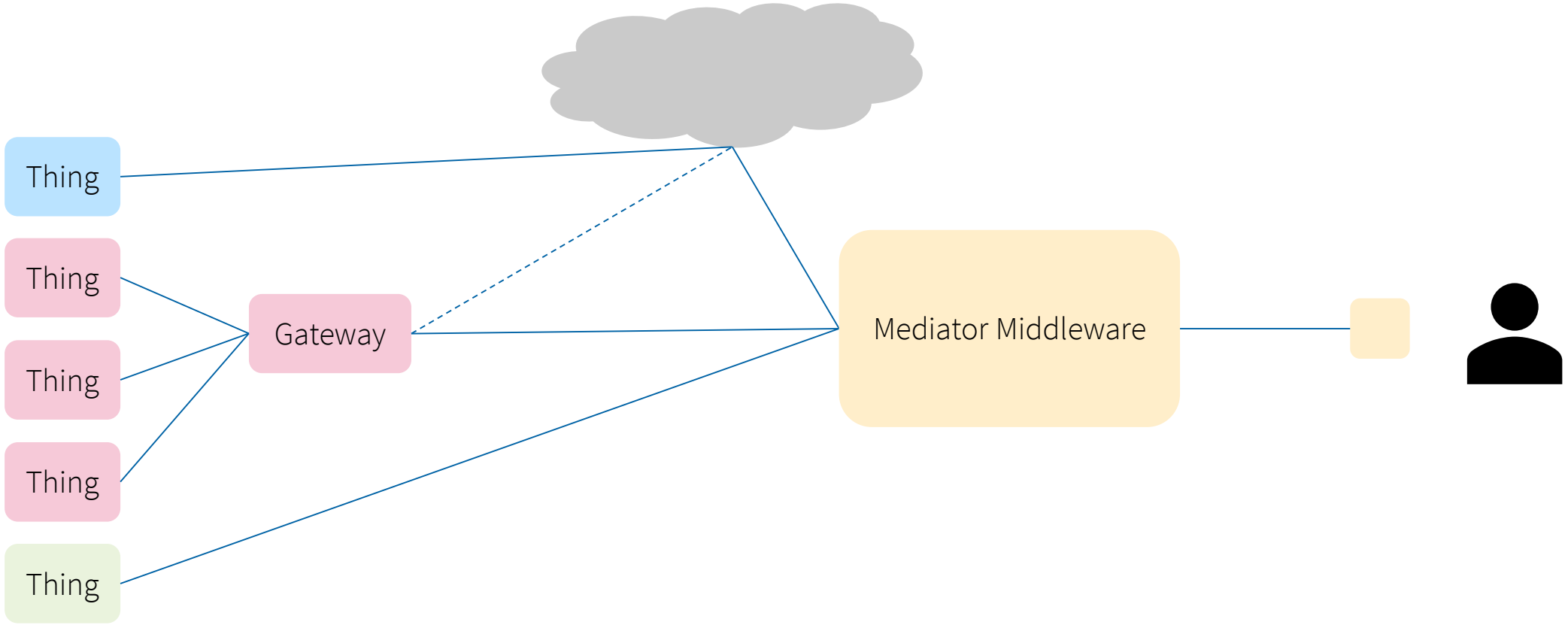

Thing
Thing
Thing
Thing
Thing
Gateway
Mediator Middleware

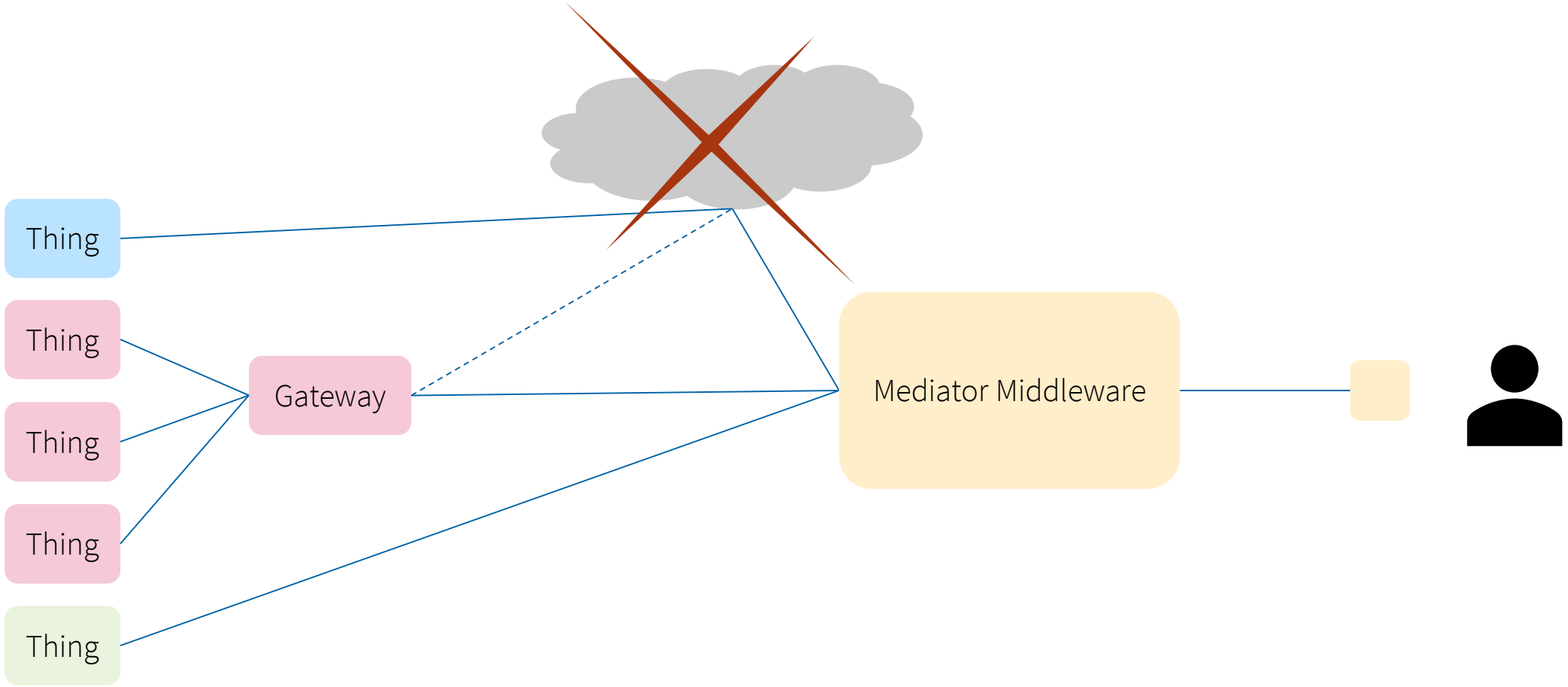

Thing
Thing
Thing
Thing
Thing
Gateway
Mediator Middleware
Thing
Thing
Thing
Thing
Thing
Gateway
Mediator Middleware
Meta-Mediator

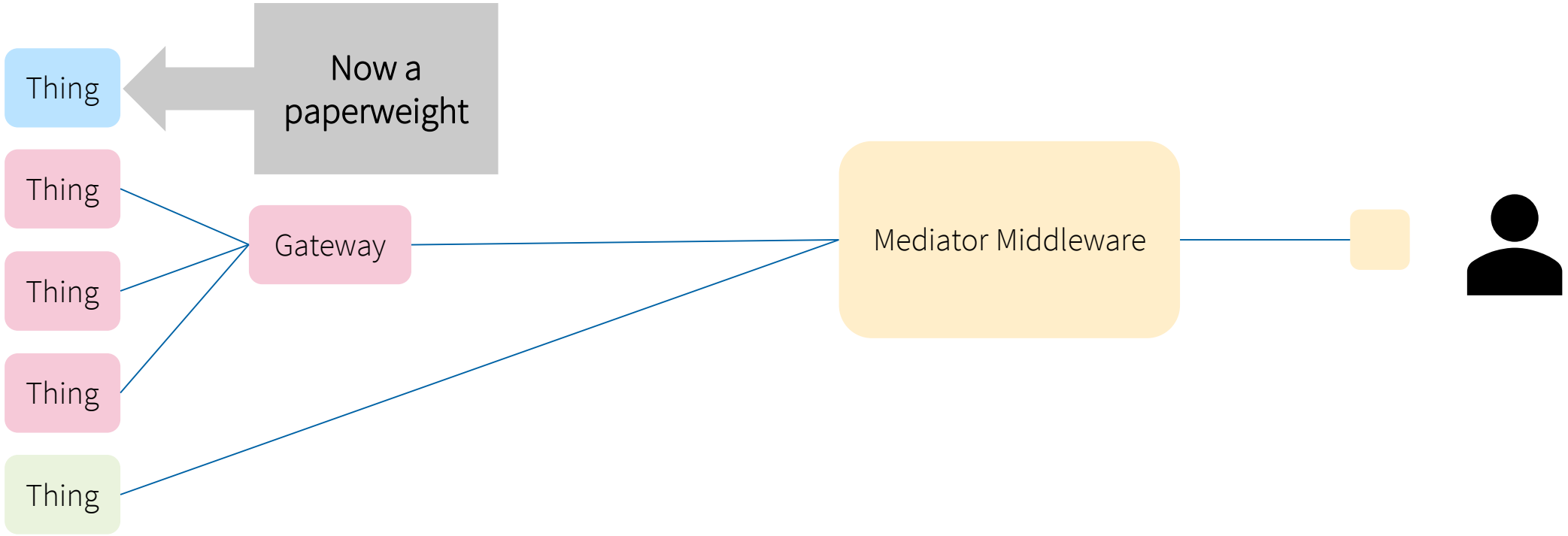

Thing
Thing
Thing
Thing
Thing
Gateway
Mediator Middleware
Thing
Thing
Thing
Thing
Thing
Gateway
Mediator Middleware
Meta-Mediator
This is getting out of Hand!

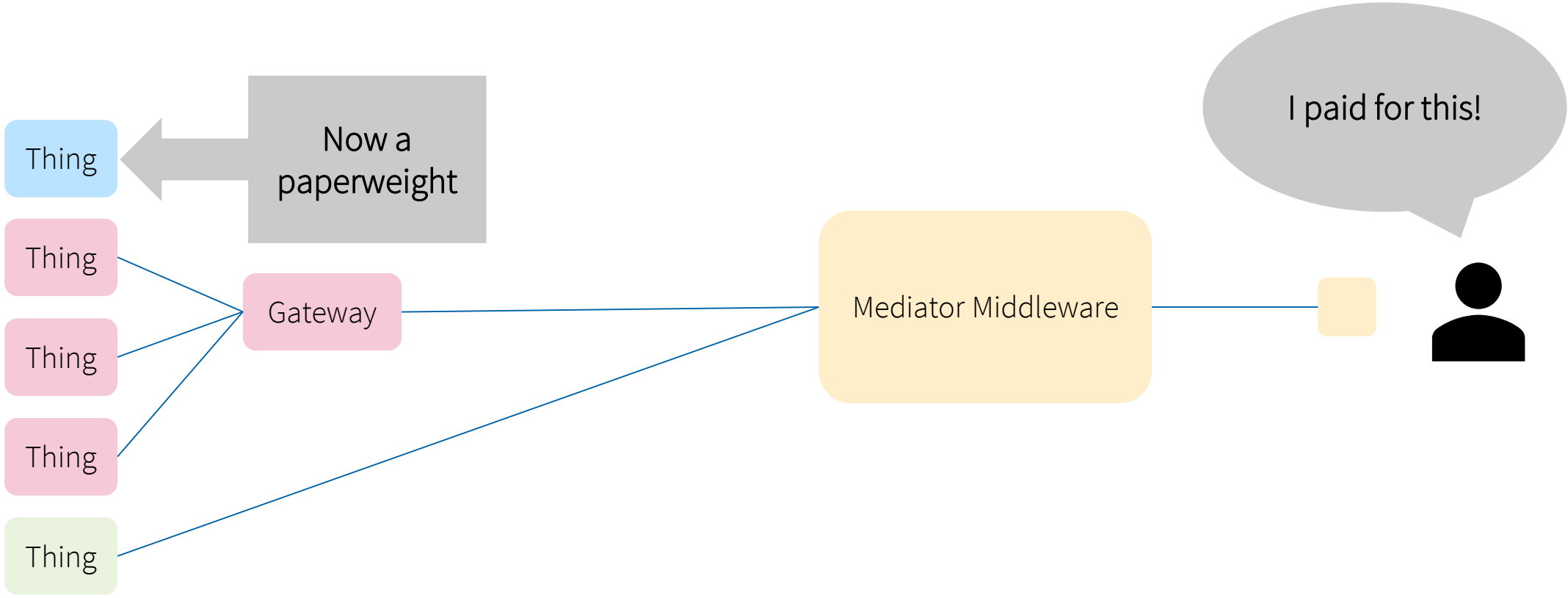


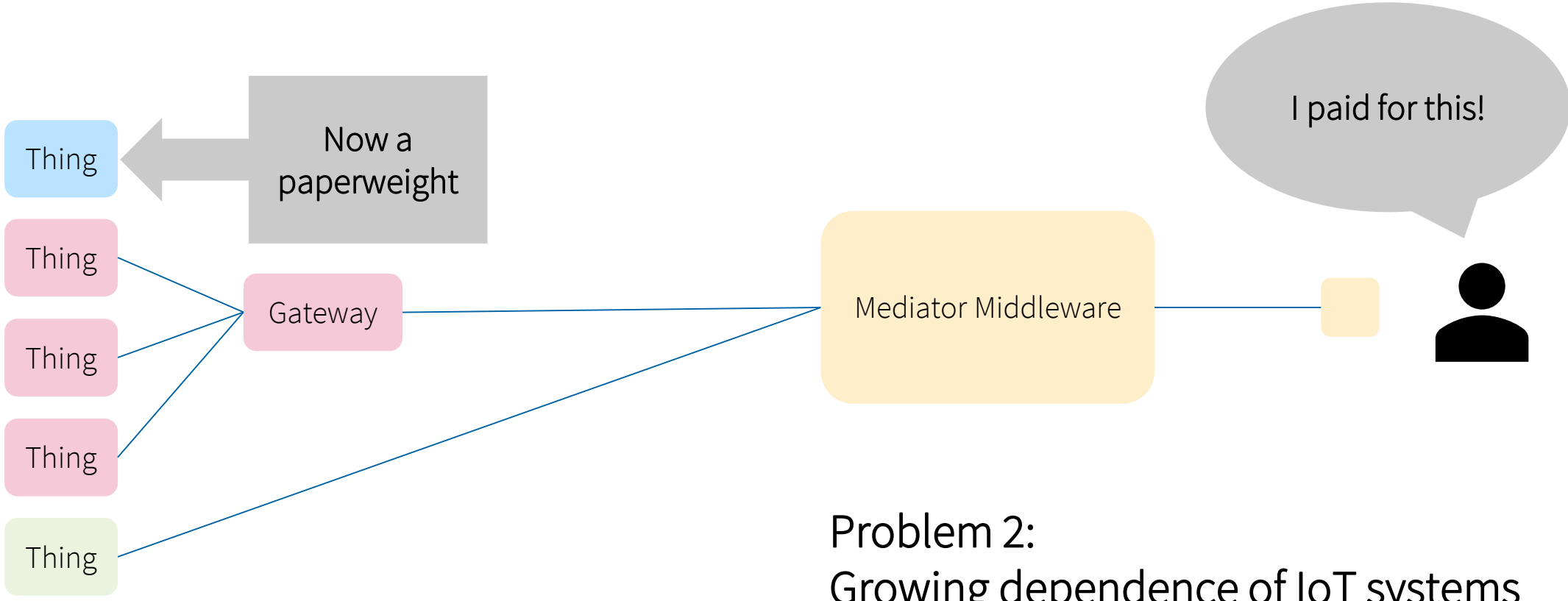
Problem 1:
Growing complexity of IoT systems


Thing
Thing
Thing
Thing
Thing
Gateway
Mediator Middleware

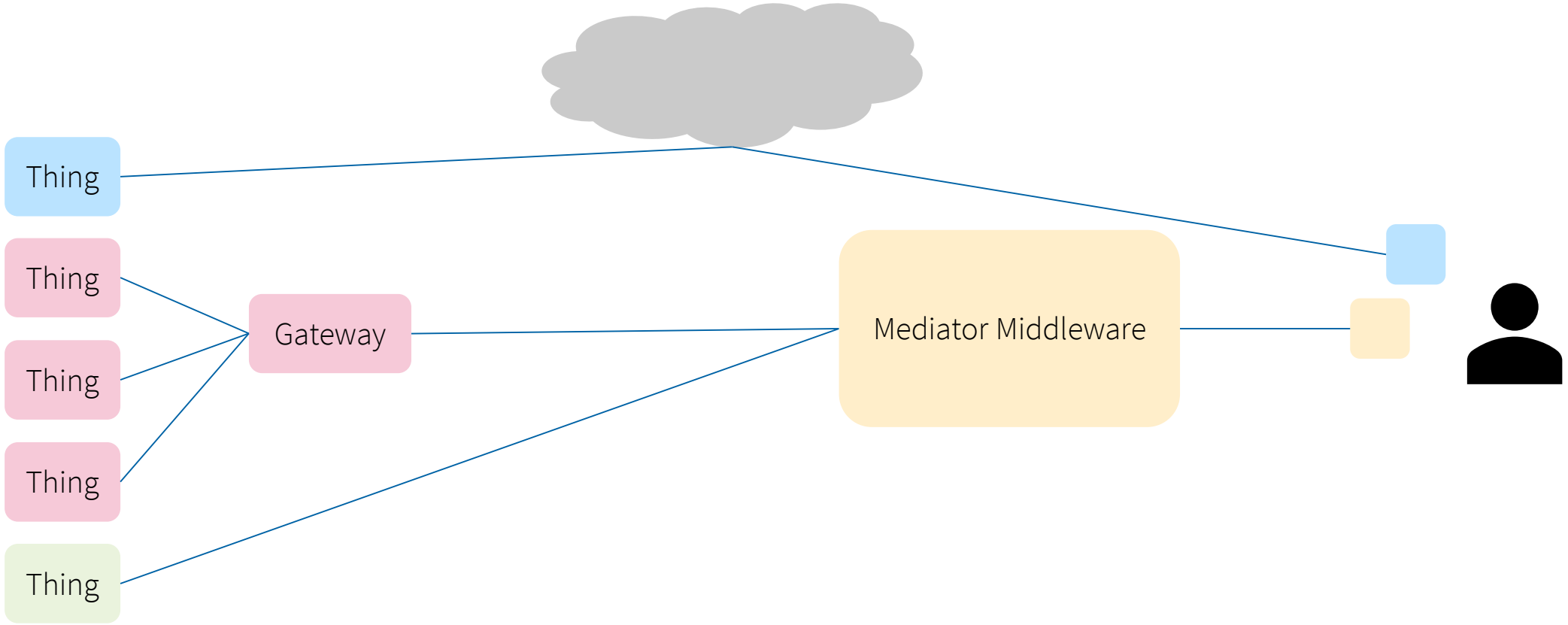

Thing
Thing
Thing
Thing
Thing
Gateway
Mediator Middleware

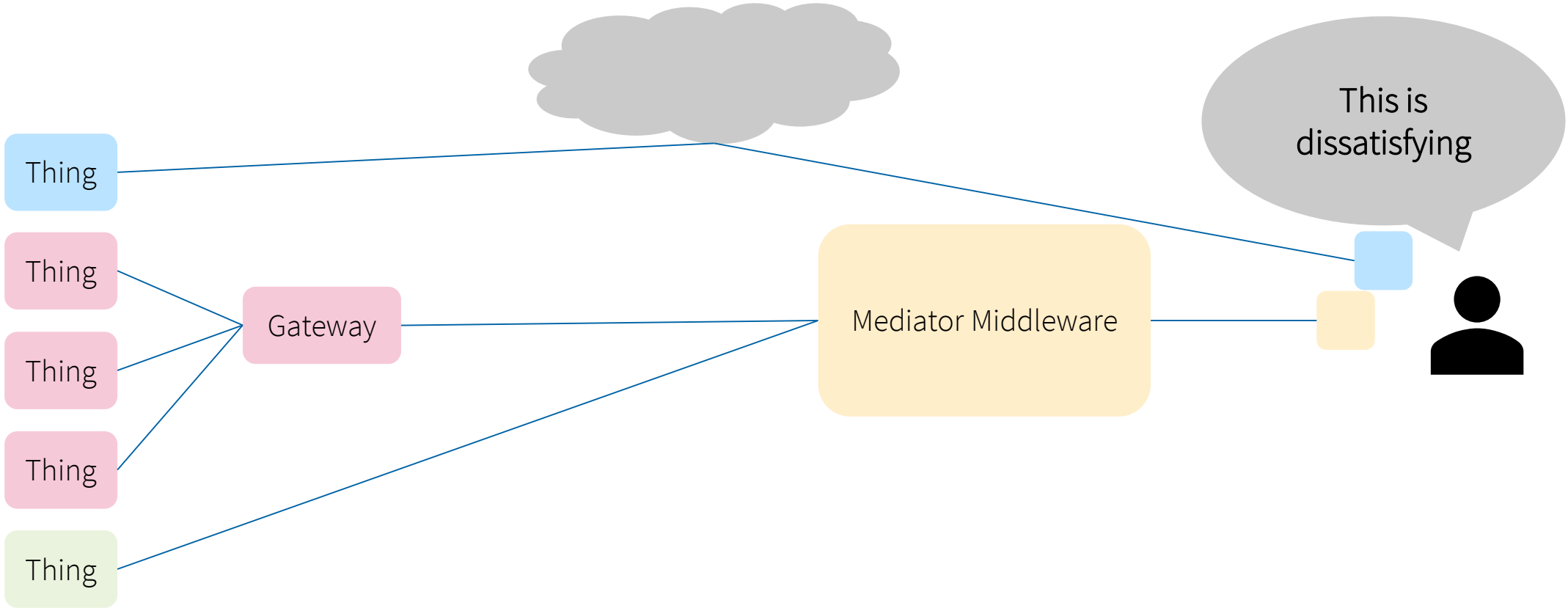

Thing
Now a paperweight
Thing
Thing
Thing
Gateway
Mediator Middleware

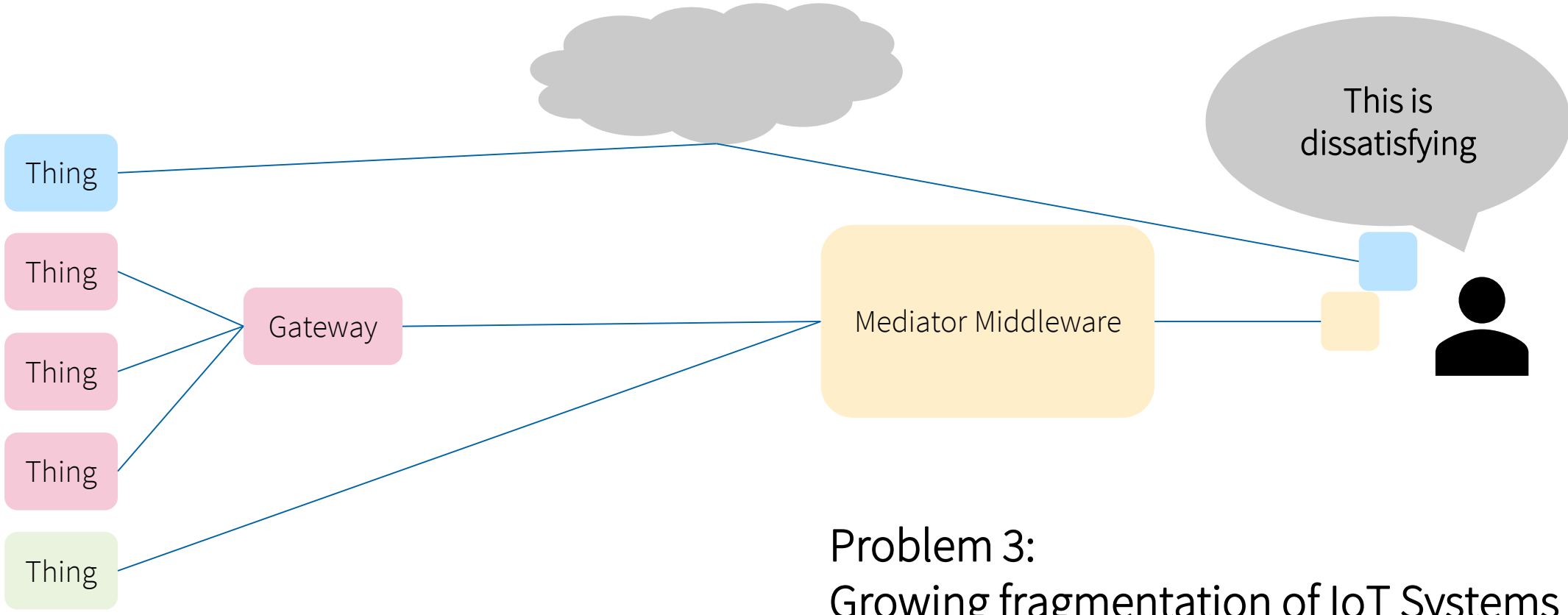

Thing
Now a paperweight
Thing
Thing
Thing
Gateway
Mediator Middleware
I paid for this!



Problem 2:
Growing dependence of IoT systems on manufacturer cloud services

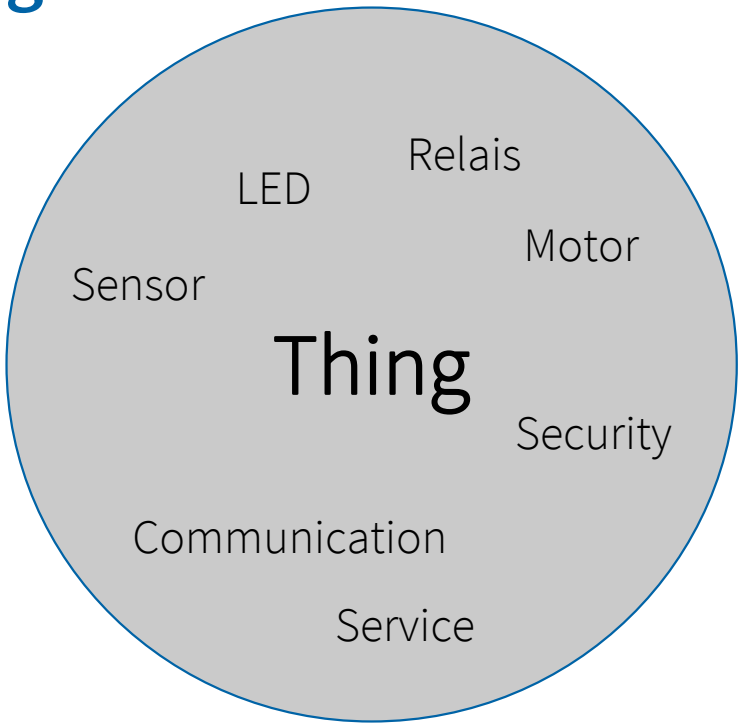

Thing
Thing
Thing
Thing
Thing
Gateway
Mediator Middleware


Thing
Thing
Thing
Thing
Thing
Gateway
Mediator Middleware
This is dissatisfying

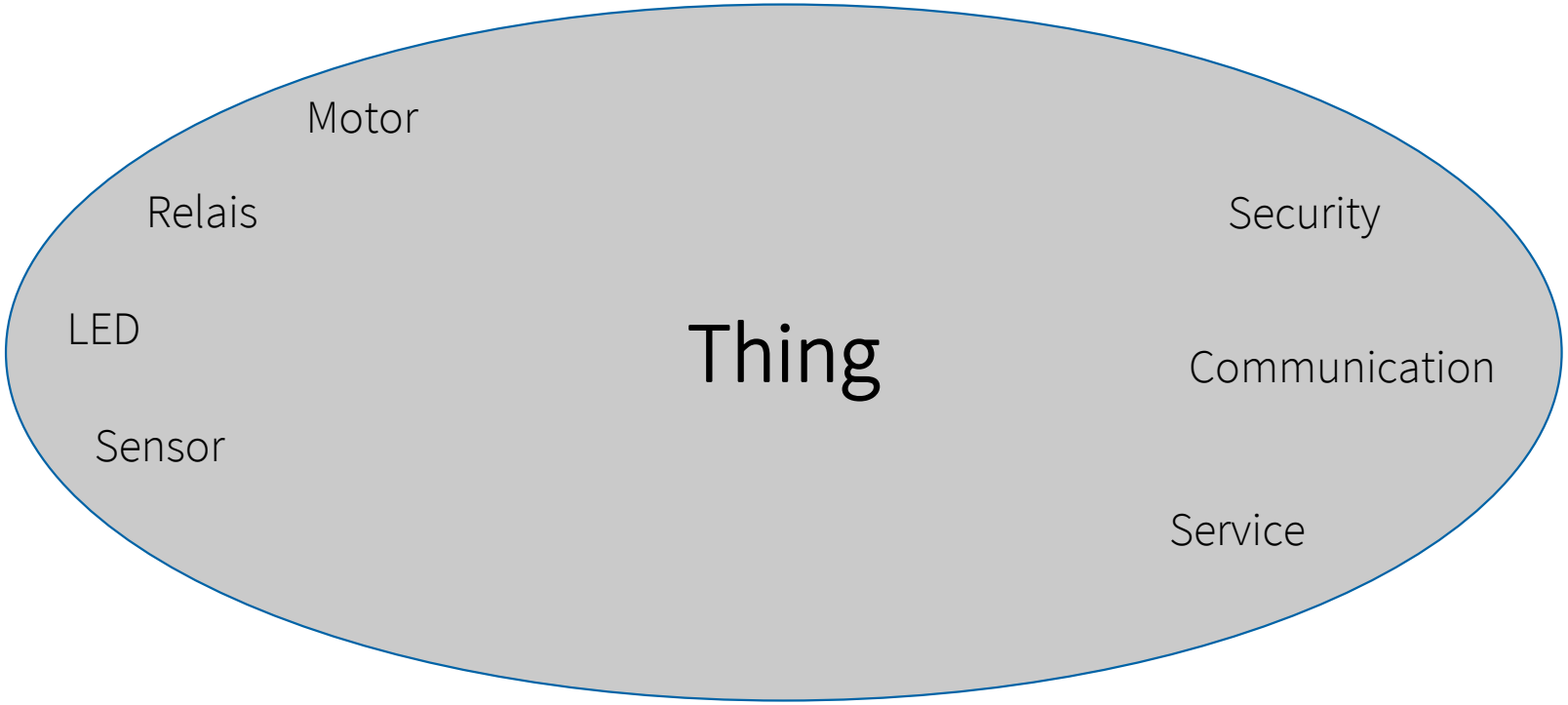


Problem 3:
Growing fragmentation of IoT Systems

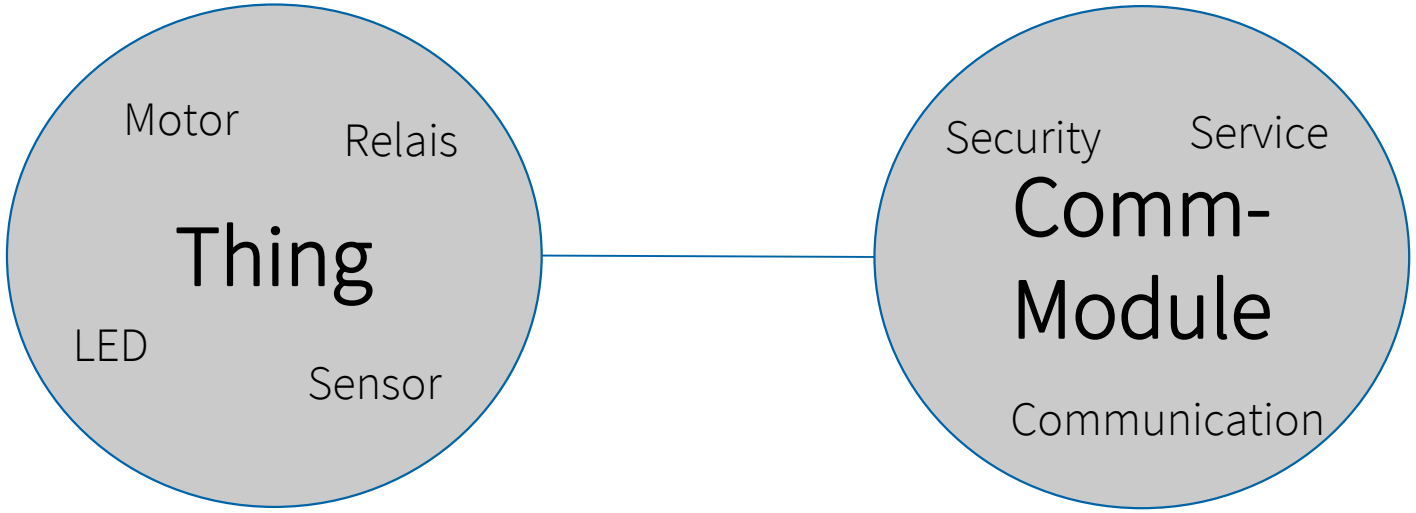
The Conventional „Thing“



tight integration

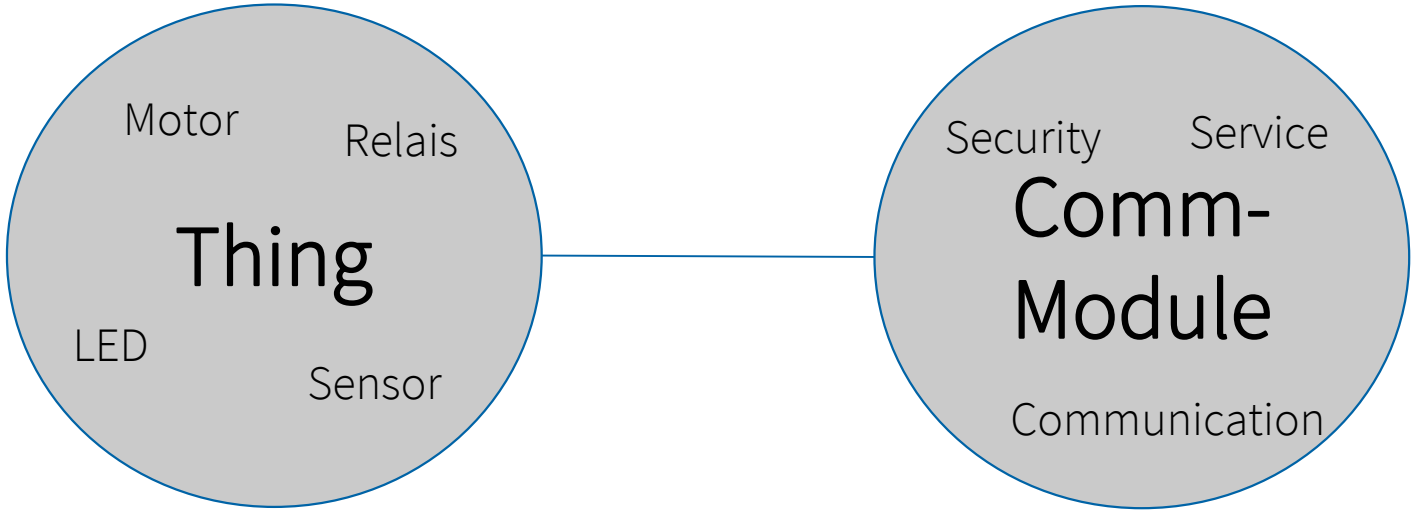

Motor
Relais
LED
Sensor
Thing
Security
Communication
Service

We Propose:
A Separation of Concerns



universität wien

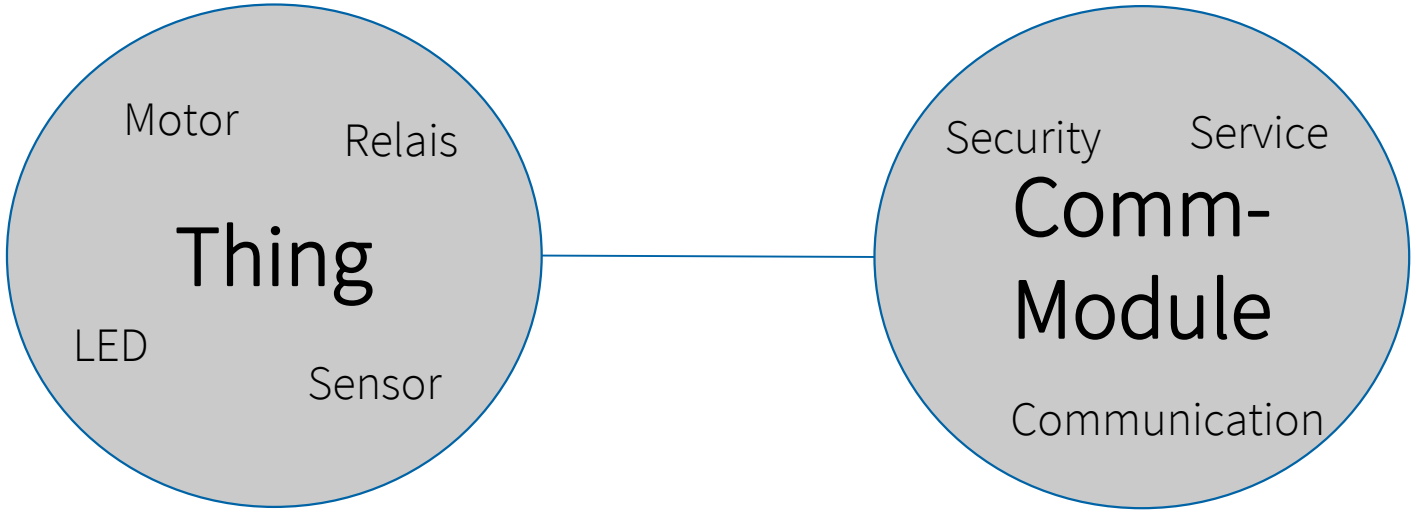
We Propose:
A Separation of Concerns



PHYSICAL and DIGITAL

Capabilities

We Propose:
A Separation of Concerns

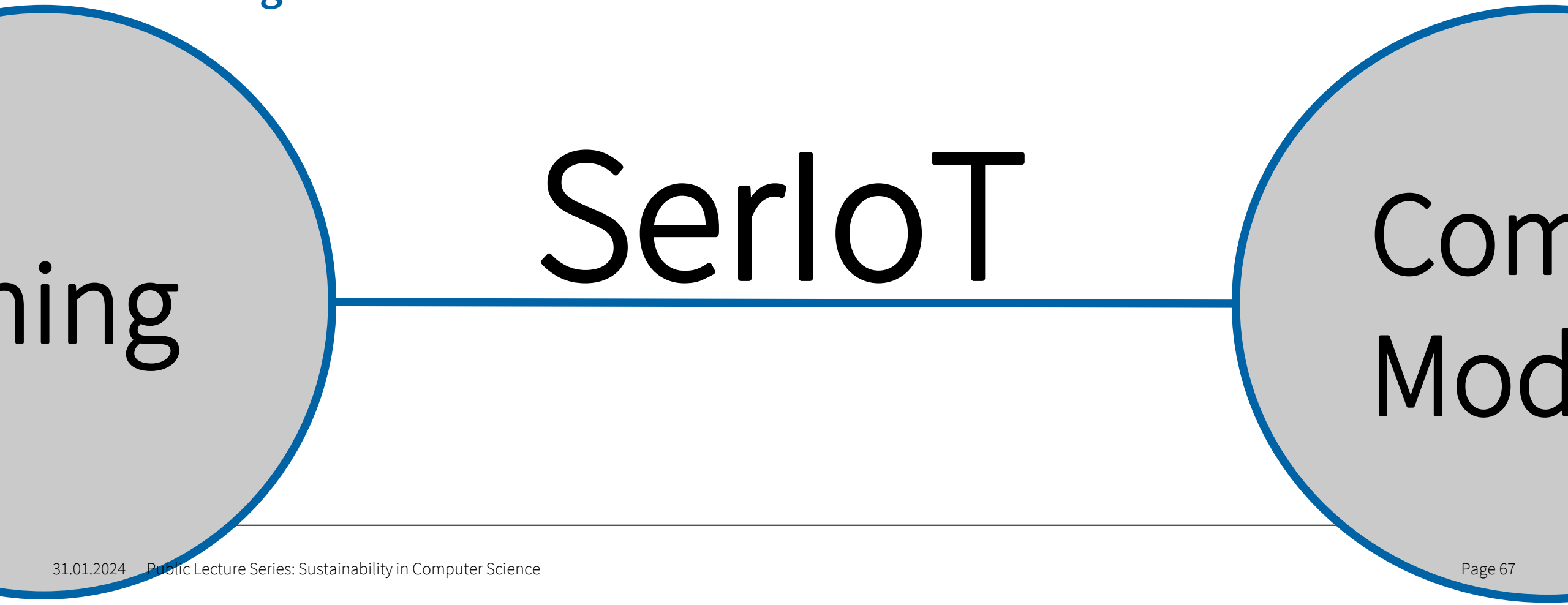


IMMUTABLE and DYNAMIC

Properties

universität wien

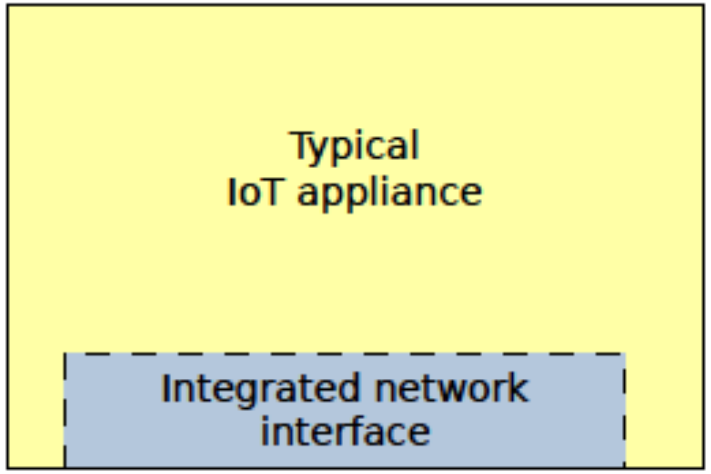
Introducing



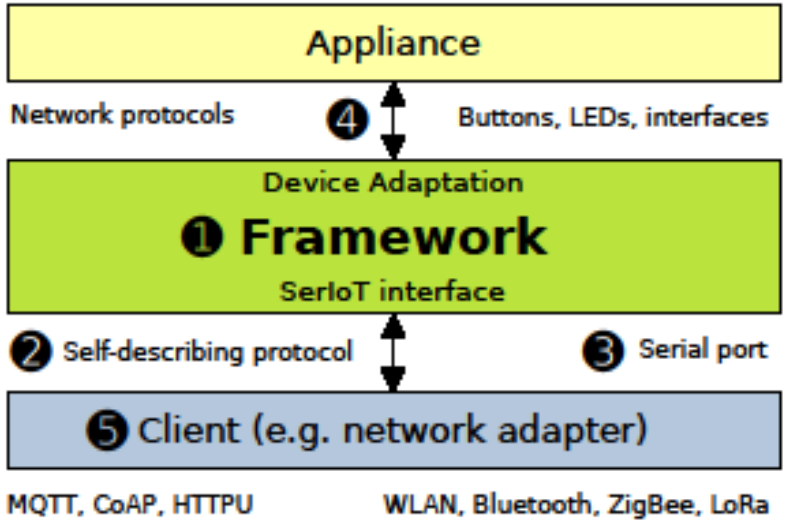
universität wien

Overview

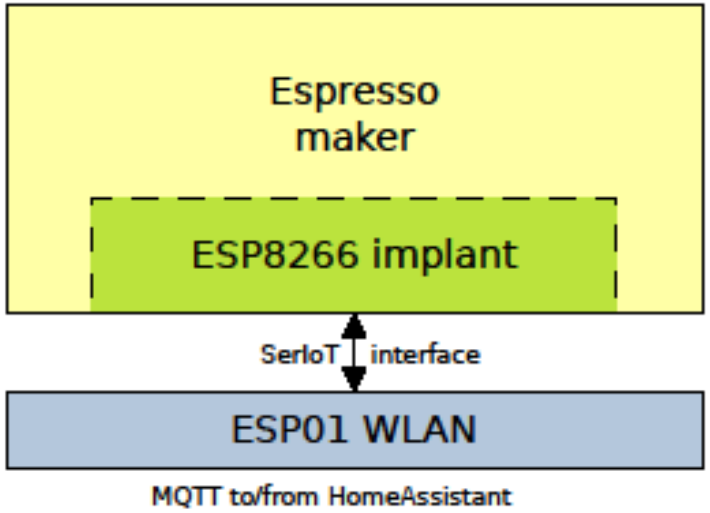
STATE OF THE ART



DESIGN



IMPLEMENTATION



universität wien

Physical Protocol

- Simple low-voltage serial port
- Based on Universal Asynchronous Receiver-Transmitter (UART)
- Supported by many microcontrollers and other hardware
- Can be interfaced from PC with simple adapter
- Connector would need to be standardized in the future

- → High compatibility and readily available hardware

universität wien

Logical Protocol

- ASCII-based
- Human readable
- JSON
- „start“-command to request device description
- Around 128 bytes per interaction affordance

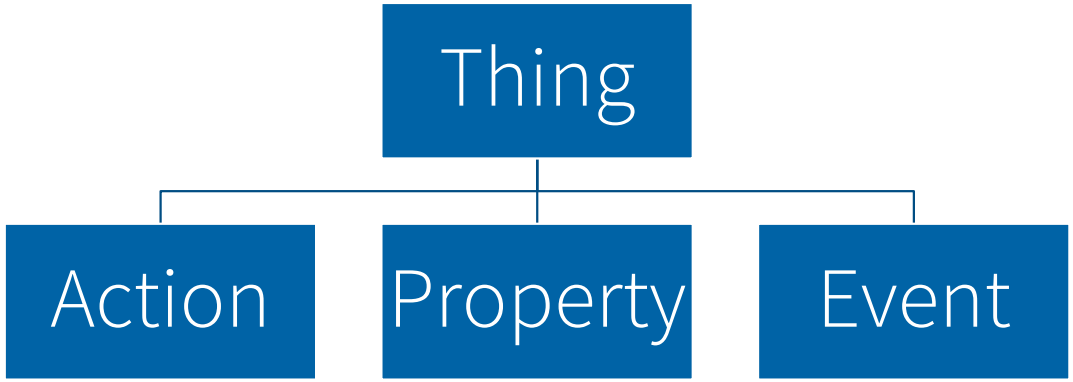
- → Anything needed to operate the interface is provided by the interface itself
- → Client devices can always be developed anew


```
{
  "id": 1,
  "name": "Espresso Maker",
  "model": "Caffe Cortina",
  "manufacturer": "DeLonghi",
  "description": "Automatic Espresso maker with steam wand",
  "properties": [
    {
      "title": "eco_state",
      "description": "The Machine is in Eco State",
      "type": "bool"
    },
    {
      "title": "attention_state",
      "description": "There is a problem, check the Machine",
      "type": "bool"
    }, ...
  ],
  "actions": [
    {
      "title": "turn_on",
      "description": "Turn on the Espresso Maker"
    },
    {
      "title": "single_espresso",
      "description": "Make a single Espresso"
    },  ...
  ]
}
```

universität wien

Data Model

- Based on the **Web of Things**‘ Things Description
  - Actions
  - Properties
  - Events



→Simple and comprehensible

→ Potential for Integration with WoT Technologies

universität wien

Device Adaptation Layer

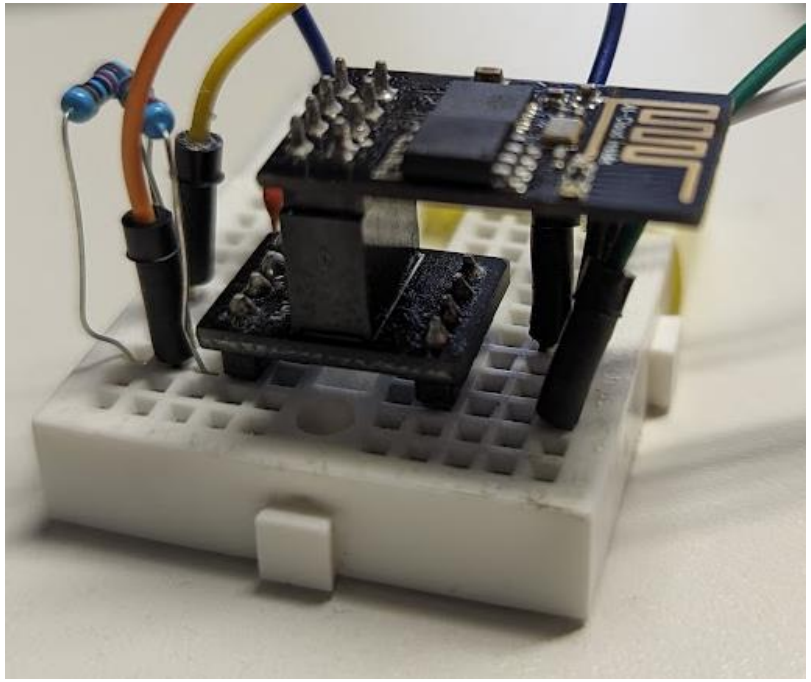
- SerIoT needs Device Adaptation Layer (DAL) code to access actual device functions
- Callable functions are required for the interaction affordances within the DAL
- Mapped to the framework and SerIoT with glue code

→ The DAL-Code is interchangeable and any Arduino-compatible retrofitting project can be integrated with SerIoT

universität wien

Client

- Connects to the SerIoT interface
- Can be anything
  - Alternative physical user interface (e.g. for Accessibility)
  - Use any communication standard (e.g. LoRa, ZigBee, Ethernet)
- Needs to map SerIoT device description to own communication-oriented data model



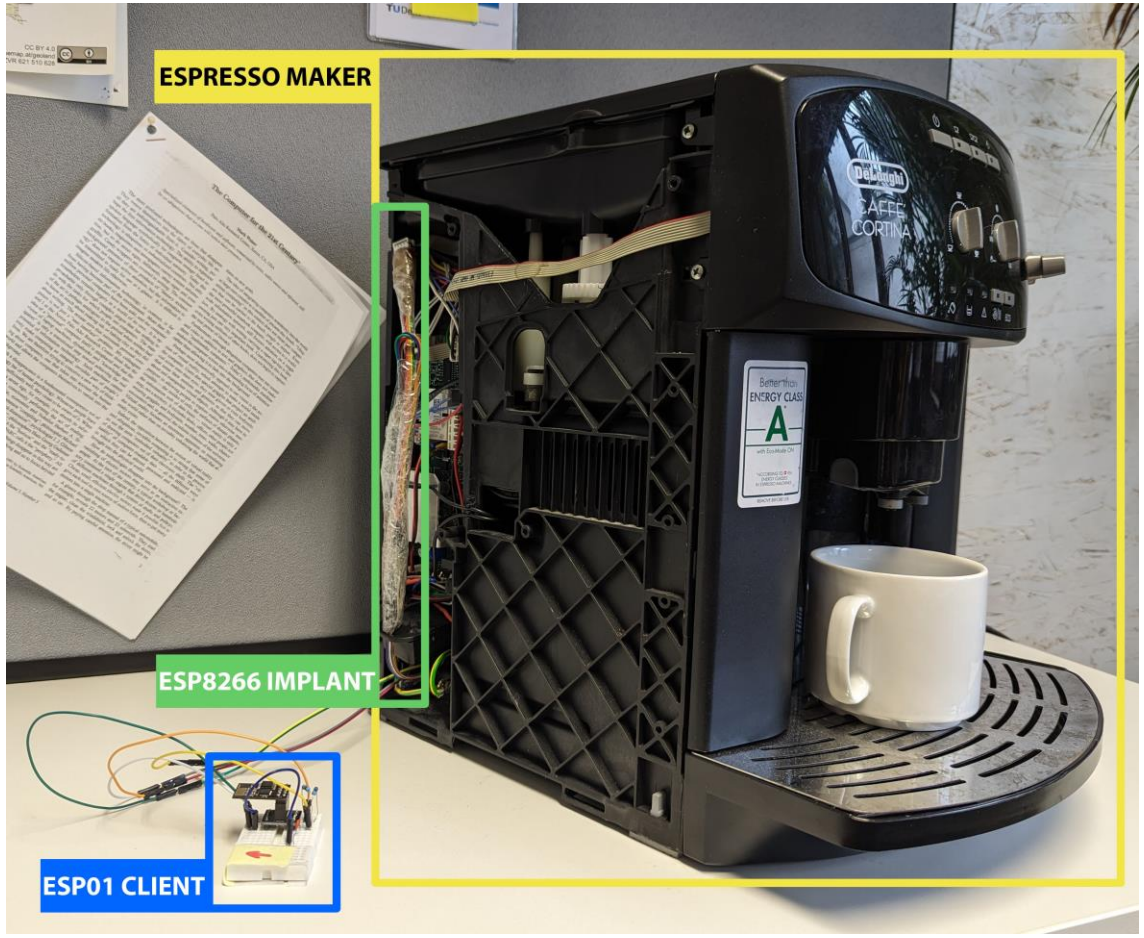
→ Consumers can select service providers and communication technologies

→ Manufacturer bankruptcy scenario sidestepped

universität wien

Implementation: The Big Picture

- Implant for espresso maker
  - Framework in Arduino C++
  - Glue code
  - Pre-existing device specific DAL code
- Client device with WiFi
  - Configurable via WiFi hotspot
  - MQTT as transport protocol
  - Supports Home Assistant MQTT autodiscovery
  - Generic: would work on any SerIoT interface



universität wien

Evaluation

- Proof of concept

→ Successful establishment of communication between Home Assistant and DAL

Part	SLOCs	RAM	Flash
Model only	222	-	-
Model + SerIoT	261	28.1KB	265KB
Glue code	15	-	-
DAL	408	28.0KB	260KB
Full implementation	**684**	**29.5KB**	**280KB**
Original retrofit	343	29.2KB	302KB

Table 1: Source code, memory, and storage consumption of our Framework, compared to original retrofit

- Evaluation of resource consumption on microcontroller
  - RAM
  - Flash

→ Minimum overhead compared to DAL only

→ Negative overhead compared to original DAL with webserver on Microcontroller

universität wien

Takeaways:

- SerIoT provides Upgradeability by Default
  - Human readability & self description
  - Basic protocols
  - Basic data model
  - Accessible interface
- SerIoT is retrofit-friendly
  - Framework is open source
  - Framework is designed to incorporate foreign code
- SerIoT enables competition for IoT services
  - Open local interface allows for developing new adapters and services

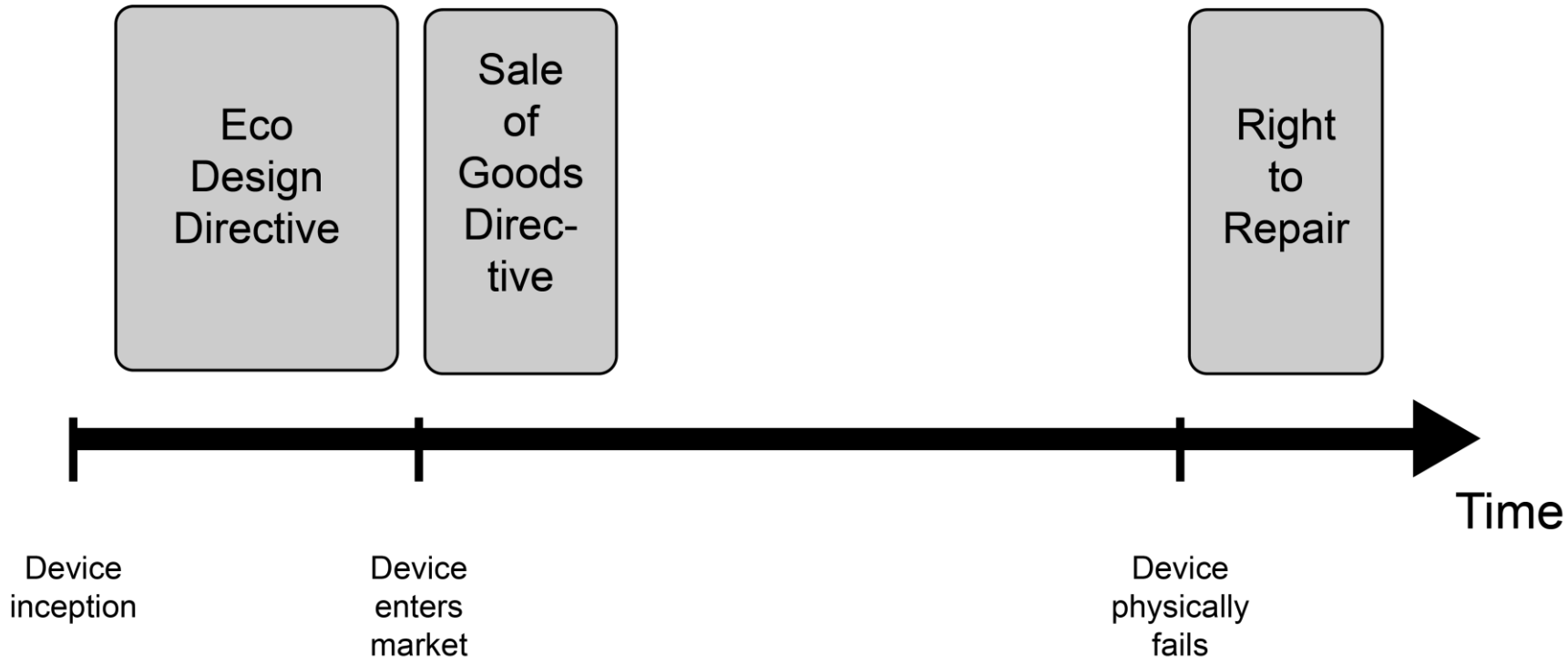
universität wien

Going Forward?

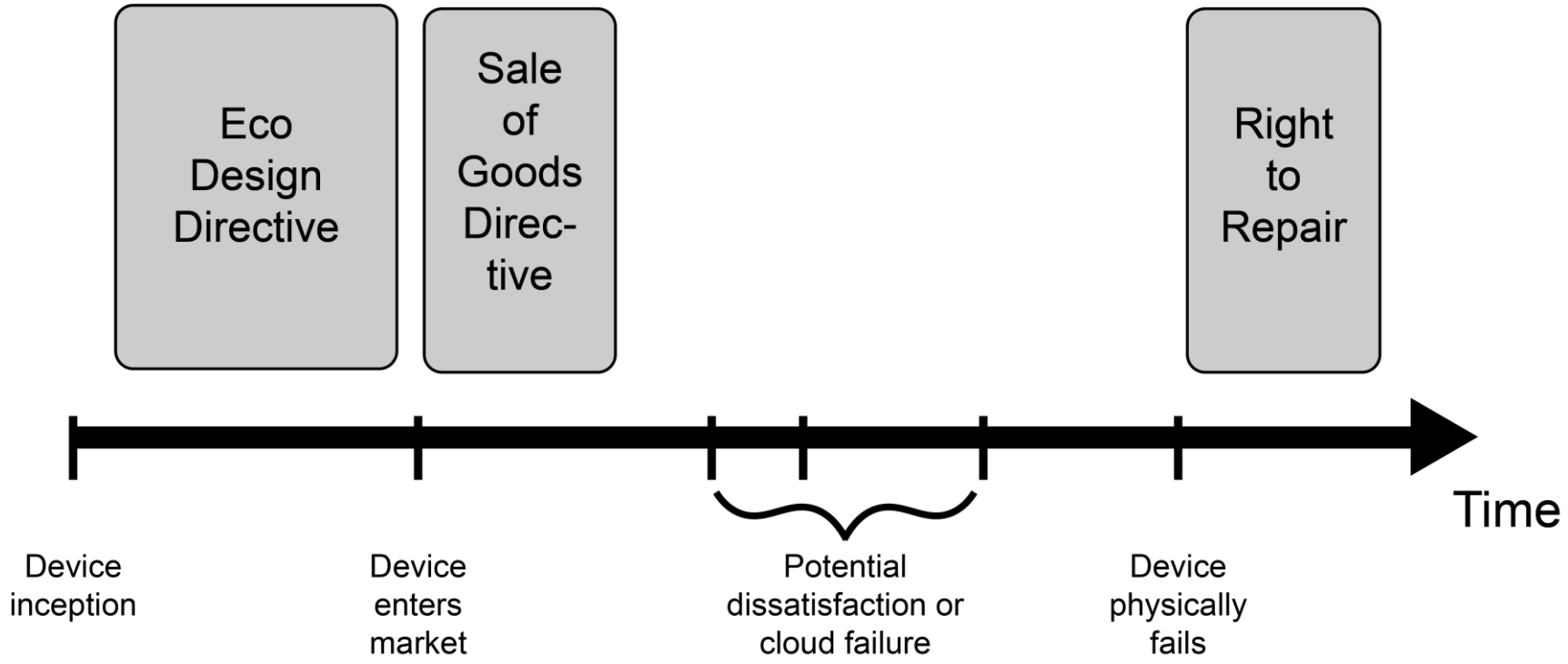
universität wien

1. UN SDG 12: Responsible Production and Consumption
2. Product Obsolescence and the legal Framework in the EU
3. Domain Specific Challenges of IoT Retrofitting
4. SerIoT: An Interface for Upgradeability-by-Default
5. Towards a Right to Improve

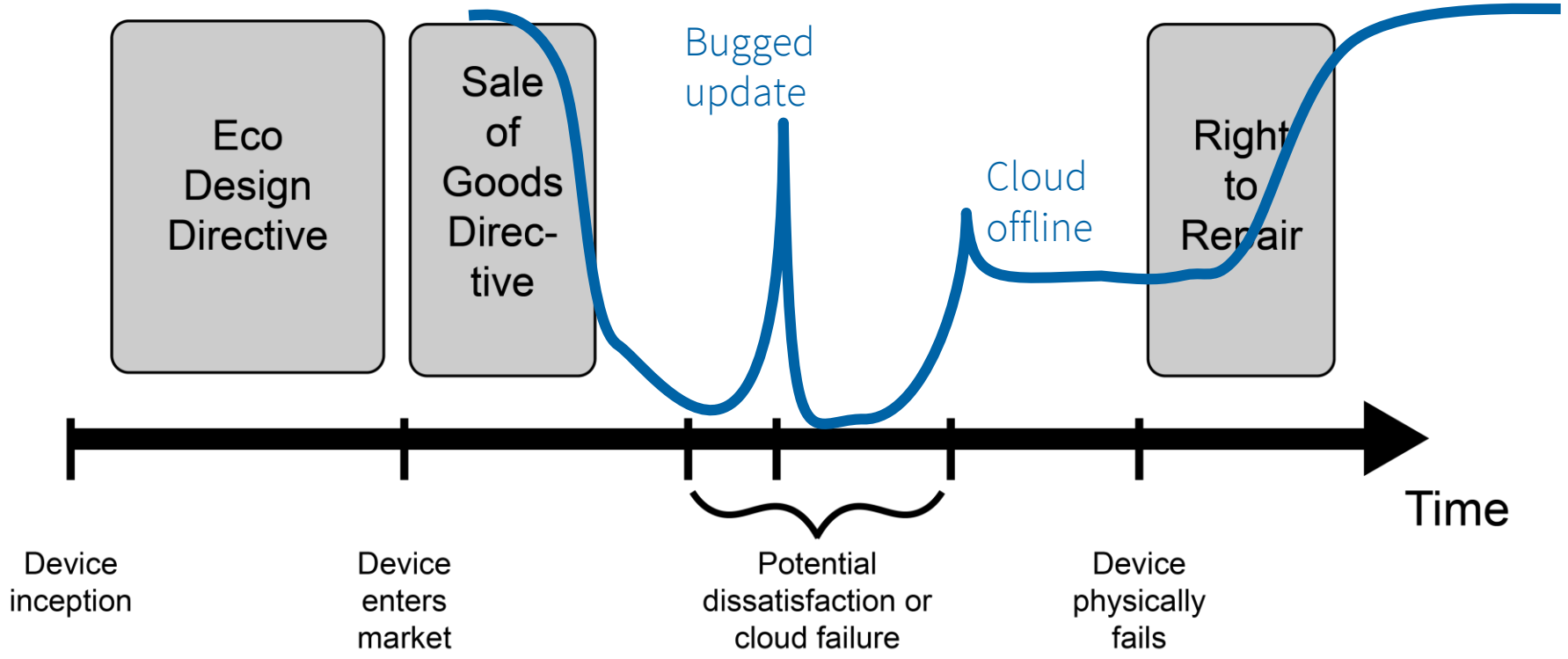
Remember The Legal Framework in the EU?



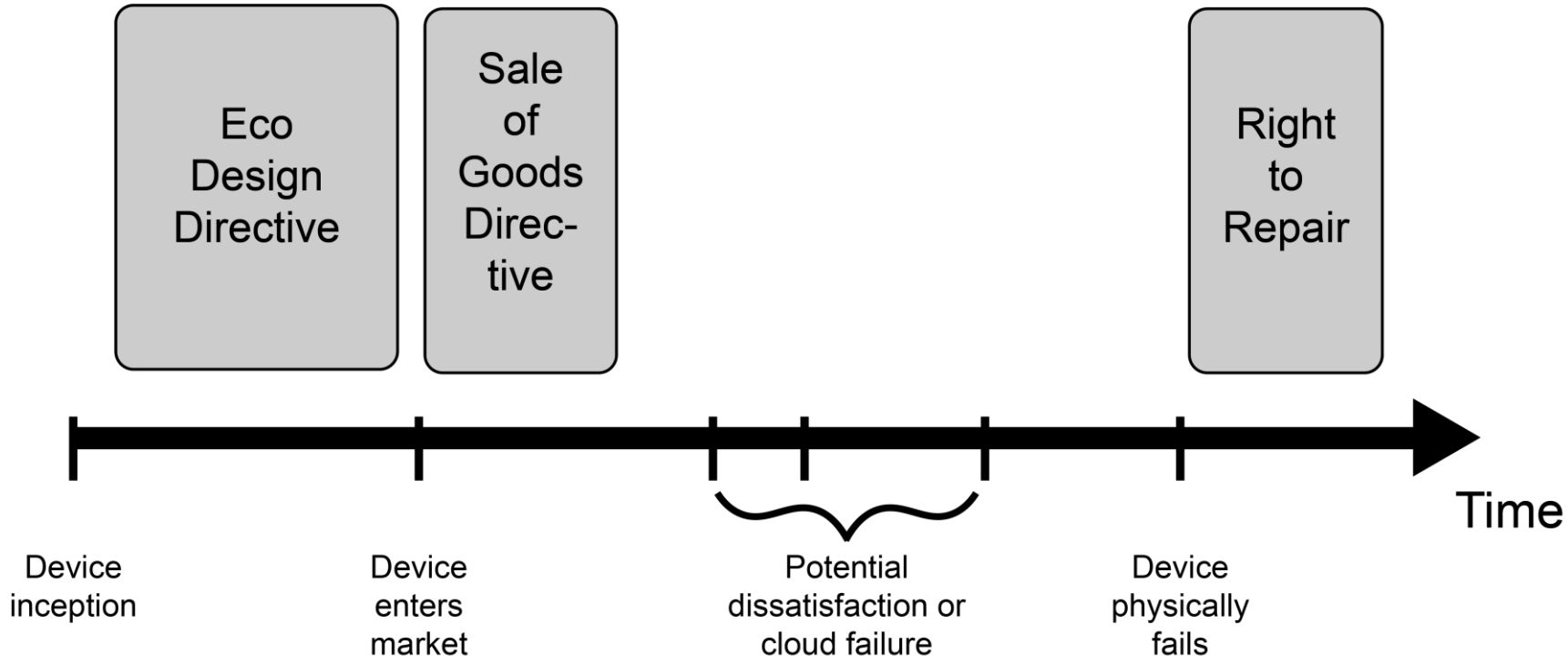
The Legal Framework in the EU misses the IoT



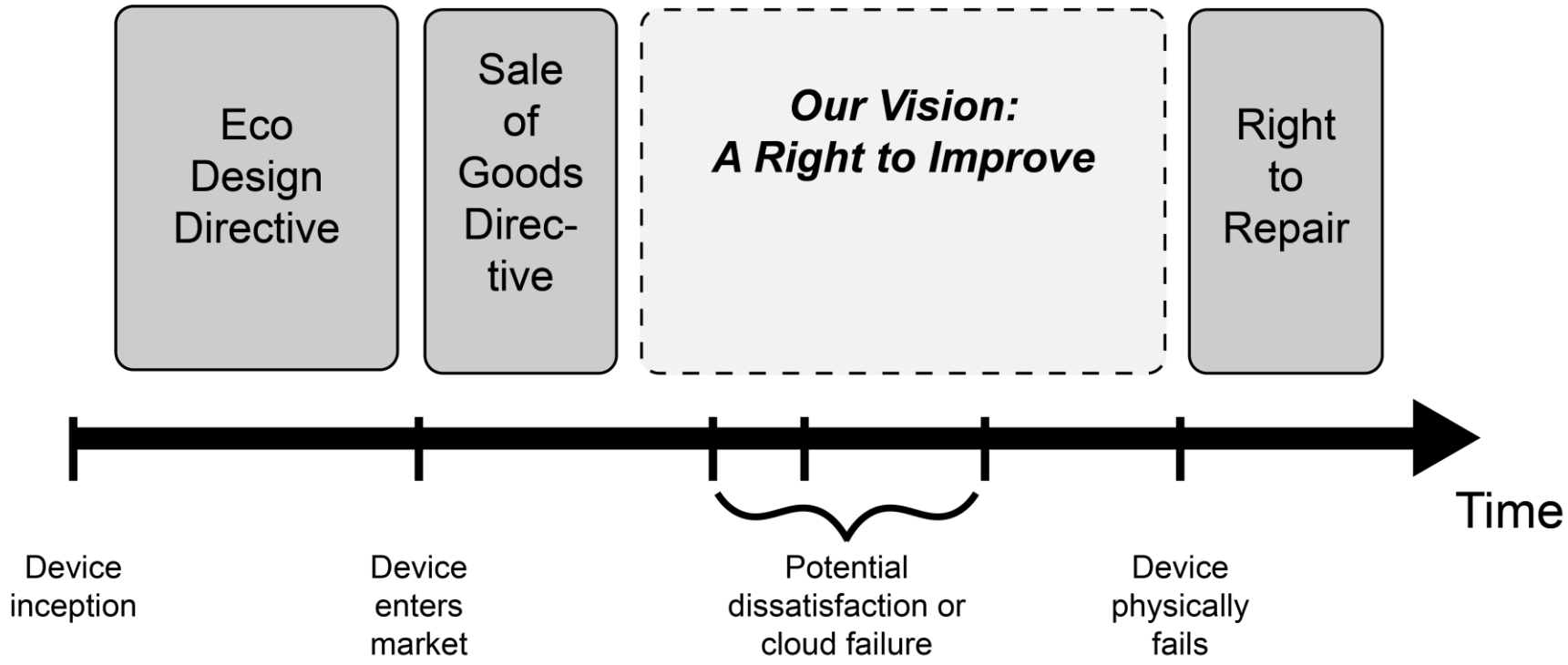
The Legal Framework in the EU misses the IoT



The Legal Framework in the EU misses the IoT



A Right to Improve



universität wien

Towards a Right to Improve

- Systematic review of academic literature
- Screened over 400 Sources in
  - ACM-DL
  - IEEEXplore
- Analyzed over 80 Sources in-depth
  - For arguments supporting a Right to Improve

universität wien

Evidence supporting a Right to Improve

- Techical feasibility
  - Case studies & SerIoT above
  - (Industrial) IoT retrofitting in literature

- User desires in literature
  - Desire of users to „compose“ smart home devices
  - Failure of current smart home systems to support specific use cases

universität wien

How could a Right to Improve look like?

Manufacturer Obligation

- Analogous to the Ecodesign Directive
- Mandate of Interface
- Interface specifications
- See also: USB-C charging interface for mobile phones

Consumer Right

- Analogous to the Right to Repair
- Consumer entitlement to accessible interface
- More freedom in design
- Abstraction leaves Scope open

universität wien

Conclusion! Today, you‘ve heard about:

- The SDG #12 and some aspects of how the EU hopes to further it
  - But we can act beyond that!
- Product obsolescence and the „Bathtub curve“
  - No „Thing“ lasts forever
- Retrofitting IoT capabilities to household appliances
  - Adding features is already possible, but challenging
- SerIoT, an interface for retrofitting and upgradeability for IoT devices
  - An alternative technological path
- Our vision for a Right to Improve
  - A Vision, founded in a systematic review of academic literature*

*and to appear in Frontiers in the Internet of Things (currently under review)

universität wien

Special Thanks to:

- Albert Rafetseder
- Raphael Ornetsmüller
- Harry Fesenmayr

universität wien

The SerIoT Framework is available as Open Source Software on Github →

https://github.com/harryf98/Device-Framework

Thank you for your attention!

Kaspar Lebloch, kaspar.lebloch@univie.ac.at

universität wien

Discussion