IAL: An Information Abstraction Layer for IoT Middleware

Andrei Günter[1,4], Christopher Schwarzer[2,4], Matthias König[3,4]

Abstract: The internet of things is an ever-expanding world of connected devices and services. Through observation of analog and digital processes, plenty of information is continuously produced. More than often, the information thus obtained is tailored to serve one separate purpose. Existing architectures omit the fact that partial results can be enriched with meta information and can be shared among network participants while information is processed. We show that enriched and simplified information packages can be shared to facilitate cooperation between constrained and smart devices as well as to serve for system optimization. In this work, we propose an implementation of the so-called information abstraction layer which serves as collective resource for querying mechanisms to provide information. To emphasize and illustrate the need for a standardized information abstraction layer into existing middleware, we outline a realistic example of use and introduce concepts to build and evaluate shared information in networks.

Keywords: internet of things; iot; middleware; architecture; abstraction layer; sensor redundancy; sensor networks; multi modality; light detection and ranging; lidar

1 Introduction

Billions of devices become connected with the emergence of smart environments and the Internet of Things (IoT). Processing data collected by these devices enables software developers to provide useful services. A smart environment offers various possibilities to develop a service since a broad range of IoT devices can be used to gather helpful information.

For instance, a presence detector could be realized with the following two approaches: 1. by using a passive infrared (PIR) sensor or 2. by using a depth camera which recognizes motion. Both devices sense helpful information for this specific task, but each device provides a different *modality* of information, in this case a typical PIR sensor provides a voltage and a depth camera a point cloud. These modalities are too different to allow a reasonable comparison between them, but both can be interpreted uniquely to achieve the higher level information *presence detected*.

This example is illustrated in Fig. 1, where a service robot and a smart light are controlled by IoT devices in the environment to provide services. Therefore, two sensors gather

[1] andrei.guenter@fh-bielefeld.de
[2] christopher.schwarzer@fh-bielefeld.de
[3] matthias.koenig@fh-bielefeld.de
[4] Bielefeld University of Applied Sciences, Campus Minden, Artilleriestraße 9, 32427 Minden, Germany

information of different modalities: 1. a depth camera provides points clouds, which are processed by a smart device with high performance to support controlling the robot and 2. a PIR sensor is used to deduce presence information with a constrained device to control a light. The depth data also carries helpful information for controlling the light, since a people detection algorithm is able to provide a presence detection as well. However, the constrained device is not able to process depth data due to a lack of performance. For the smart device, it is of little effort to share presence information in a level of detail which is appropriate for the constrained device, since it is already processing the depth data.

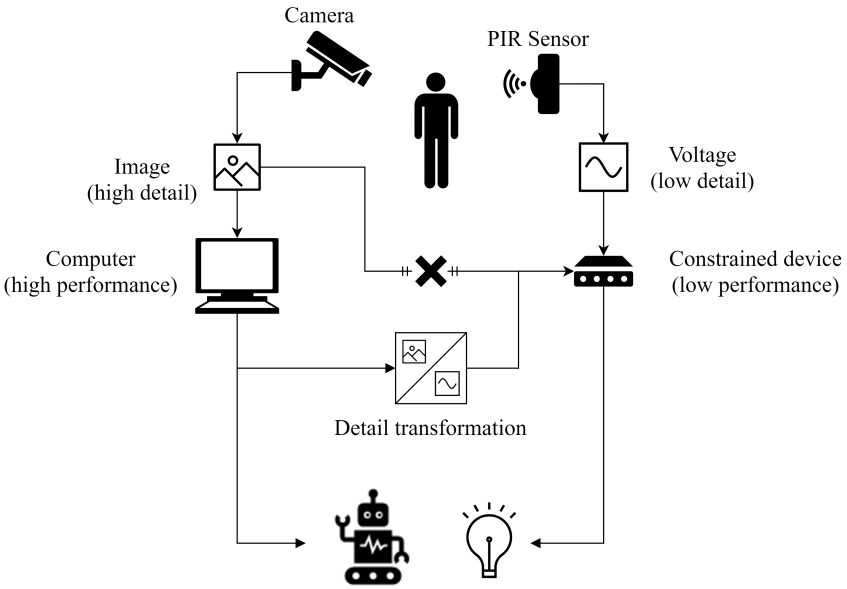

Fig. 1: A simplified example of providing information in different levels of detail in a smart environment.

A common approach in software development is to process a particular modality of information to provide a service. In turn, this restricts a software to rely on a certain type of device. In other words, the software is coupled to this certain type of device. The advantage of our proposed concept is a decentralized structure of devices and a loose coupling between hardware and software components.

Current architectures for closed systems and IoT networks provide several abstraction layers, e.g. hardware abstraction layers to support integration of different hardware, so that information can be exchanged seamlessly. This work focuses on enhancing the process of exchanging information in existing architectures by introducing an Information Abstraction Layer (IAL), which allows services to process information of different modalities to decouple software services from hardware. This task is part of a project named *"Dynamic runtime for organically (dis-)aggregating IoT-processes"* (DoRIoT). One goal of DoRIoT is to improve

the adaptability and resilience of IoT services by complementing unavailable devices with suitable alternative devices. Considering the example above, this means that the service for detecting presence remains functional or partially functional for as long as a PIR sensor, a depth camera or any other device capable of detecting presence provides correct data. This work contributes to this challenge by:

1. providing an architectural overview to handle multiple modalities of information,
2. implementing a use case, which is similar to the example given with Fig. 1,
3. evaluating the implementation with resulting improvements for further work.

The following sections are structured as follows: Section 2 summarizes related work, Section 3 presents a detailed formulation of the concept, Section 4 shows an actual implementation, Section 5 evaluates our implemented system, and finally Section 6 concludes this work.

2 Related work

The idea of IoT attracts researchers from multiple computer science domains since the diversity of devices involves many aspects: energy management, networking, safety, security, data persistence and many more. Over the past two decades, each of these research domains introduced comprehensive requirements for designing adaptable, intelligent, and resource friendly network architectures. This lead to a multitude of approaches for abstraction layers in IoT networks, i.e. IoT middleware.

Existing architectures for IoT middleware can be categorized into three layers: service-based, cloud-based and actor-based [Ng17]. The requirements of paradigms like big data and neural networks can be seen as cornerstones for service-based [Ca14; So15] and cloud-based [Ng17] architectures, since both architectures can be described as heavyweight in terms of computational power or data persisting capabilities.

One goal of this work is to develop an abstraction layer for IoT middleware allowing cooperation between constrained and smart devices. Service-based architectures will not be further discussed in this work, since service-based architectures are not designed for integrating constrained devices, and the same applies for cloud-based architectures, since these provide limited support for constrained devices [Ng17].

The development of communication protocols [BCS12; BG18] and operating systems [Ba13; Ba18; Le05] designed for constrained devices allowed researchers to setup actor-based architectures [PA15], which enable constrained devices to cooperate with smart devices.

It was shown that actor-based architectures improve existing software development approaches, since a heterogeneously designed application running on a smart device can be

executed on a network of constrained devices [Me17]. In other words, actor-based architectures decouple the execution of services from the performance of single connected devices, but did not focus on decoupling of software components from information modalities gathered by physical devices.

In this work, we focus on the development of a data-driven actor-based architecture to decouple services from modalities of information by abstracting information from raw data and translating it to a shared modality. This shared modality can be consumed to provide services independently from actual hardware.

3 Concept

The fundamental idea of our concept is that nearly every information can be used to serve multiple purposes and can be extracted into different levels of detail. Extracting information into a smaller level of detail allows constrained devices to process information which originally was achieved by devices acquiring information with a high level of detail, as shown with Fig. 2.

Smart devices produce or consume detailed information of models very close to the real world as opposed to constrained devices which produce or consume only simplified models. For instance, a very complex model might be a three-dimensional (3D) model of depth data in high resolution enhanced with colored data representing a full space and contained objects. While a comparable very abstract model would be a single distance from one point to another in that same space.

As the detail of an information decreases, it becomes feasible to process the information with a constrained device, since less performance is required to process smaller data sizes. In contrary, the requirement for reasoning the data and mapping it to meta information increases. For instance, a single distance is too abstract to make reasonable decisions in a process if there is no meta information on what exactly this distance is representing.

Information can be distinguished based on whether it is in a raw form, as it was acquired, or it is in a synthesized form enriched with meta information describing its context. In most cases, sharing information is of great interest, when it was synthesized before and represents a higher level context. The IAL is responsible to enrich information with context and enable information to be reused by multiple network participants. In other words, the integration of an IAL into existing IoT middleware allows services to use and share data extended with meta information for providing services decoupled from raw information modalities and consequently from its hardware.

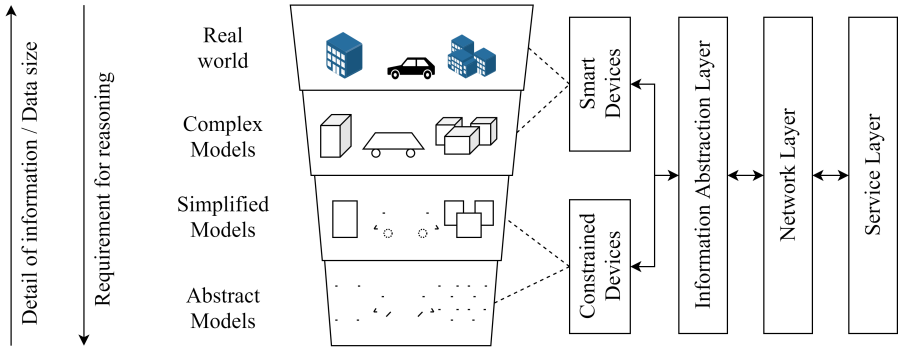

Fig. 2: Concept for different levels of detail in representations of information.

3.1 Problem formulation

A detailed formulation of our concept is given in the following. We assume that an environment contains agents which provide services by processing information. Agents are computing units optionally equipped with sensors or actors to interact with the environment. We denote a set of agents in an environment as:

$$\mathbb{A} = \{a_1, ..., a_{m-1}, a_m\} \tag{1}$$

Every agent uses information as input and optionally generates information as output. We define n inputs and outputs for an agent a_i as:

$$\mathbb{Z}_{a_i} = \{z_1, ..., z_{n-1}, z_n\} \tag{2}$$

A proper generation of outputs requires agents to presume particular data types for inputs. We define modalities for inputs and outputs to distinguish between different data types of information and to describe the context of information. With the following equation we define k modalities for an agent a_i, whereas k is less or equal to n since multiple pieces of information might be described with the same modality:

$$\mathbb{M}_{a_i} = \{m_1, ..., m_{k-1}, m_k\} \mid k \leq n \tag{3}$$

Information of high detail can be reduced or divided into multiple pieces of information, where each piece of information can be classified differently. For instance, a voice record can be classified based on information about number of different voices, duration of each occurring voice, gender of voices, or whether the record is a speech, a reading, or a song. Each classification might be described within a different context and might be represented with a different data type. In other words, information of high detail can be reduced, divided,

or transformed into pieces of information, where each piece might be described with a different modality. We define that a modality m_i can be transformed to another modality m_j if there is an existing set of process steps $\mathbb{P}_{i,j}$ which allow a deduction of m_j from m_i.

$$\mathbb{P}_{i,j} = \{p_1, ..., p_{l-1}, p_l\} \tag{4}$$

We define that a service s_i can be provided by an agent a_i if a necessary subset of information $\mathbb{Z}_{s_i} \subseteq \mathbb{Z}_{a_i}$ is available and up to date. Further, we denote that all modalities of $\mathbb{Z}_{s_i}$ can be described with $\mathbb{M}_{s_i} \subseteq \mathbb{M}_{a_i}$. In this work we show that services in IoT networks can be optimized by choosing proper modalities for $\mathbb{M}_{s_i}$ which enable to reuse information and allow a cooperation between agents equipped with different sensors leading to a loose coupling between hardware and software. In the remainder of this section, we propose a system architecture to provide an overview of necessary components to implement the presented concept. Fig. 3 depicts our proposed decentralized system architecture, which is composed of multiple participants in a network, namely: agents and brokers.

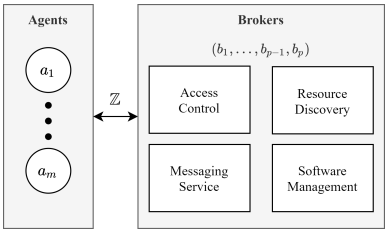

Fig. 3: Proposed system architecture.

Agents $\mathbb{A}$ generate information $\mathbb{Z}$ which is shared via networking by communicating with brokers. Brokers are responsible for managing and distributing information among network participants and can be described as IoT middleware, which consists of four components: a messaging service, an access control, a resource discovery, and a software management. A messaging service is required to send and receive messages through the network. A typical setup for a messaging service would be a publish and subscribe scenario where each participant is able to send information by publishing to a topic or to receive information by subscribing to a topic.

If a topic contains incomplete, outdated or corrupted data, then other related topics should be available for use. Therefore, a resource discovery allows consumers and producers to query for information. A resource discovery also supports aggregation of new devices to the system, since devices can be directed to existing topics. Newly aggregated devices from other networks or devices containing processing units also contribute to the system by running local software. Detecting and synchronizing updated software can be beneficial for network participants, especially in case of required security updates. In contrary, software updates also involve security issues. Therefore, an access control is used to allow only permitted participants to access messaging services, resource discovery, and software management.

4 Implementation

We did not implement every aspect of our presented concept, since the following components are handled as distinct research topics in the DoRIoT project: access control, resource discovery, and software management. We focused on implementing four agents and one broker with a messaging service. An overview of the implementation is depicted in Fig. 4.

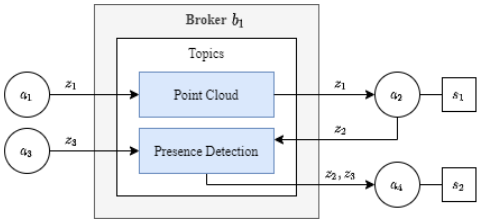

Fig. 4: An overview of the implemented components.

The system provides two services s_1 and s_2, where s_1 is provided by agent a_2 and s_2 is provided by agent a_4. The agent a_2 provides a multi-object tracking based on point cloud data with a graphical user interface, where each moving object is shown with a distinct color in a map relative to real world coordinates. The agent a_4 controls a smart light based on presence information. Therefore, the system processes five different types of information, whereas we omit two raw information types for simplicity in Fig. 4 since those are directly processed by agent a_1 and agent a_3. A raw point cloud is acquired via a solid state LiDAR sensor and initial noise is removed by agent a_1. The result z_1, a noise reduced point cloud represented by an array of 3D coordinates (x,y,z) and annotated with a sensor identifier as context, is send to a broker b_1 and can be described with modality m_1. The other raw information is a voltage, which is observed and processed with a PIR sensor by agent a_3. The result z_3, a binary value representing presence or absence of people in a room with an identifier, is also send to the broker b_1 and can be described with modality m_2.

The agent a_2 consumes information of modality m_1 and provides the service s_1 by applying a change detection to a sequence of point cloud frames. If a change is detected, then a clustering and a tracking is executed and visualized. The change detection is implemented via a software library [RC11] and the multi object tracking is realized via an open source algorithm [Pa19]. The described change detection is part of the algorithm for processing the point cloud and it directly relates to the function of the PIR sensor. In other words, the change detection also detects presence of objects and humans. Therefore, the agent a_2 is also used to publish presence information z_2 to the broker b_1. As already described with modality m_2, information z_2 can also be described with a binary modality representing presence or absence of humans in a room with the same identifier as described with m_2. In other words, z_2 and z_3 share the same modality m_2.

The service provided by a_4 makes use of agents with completely different hardware, since the light is controlled with presence information initially acquired with a PIR sensor or acquired with a solid state LiDAR sensor. Thus, service s_2 allows optimization of several

criteria, such as availability, redundancy, and network load. The hardware characteristics of all agents are listed in Tab. 1, which highlights the differences in terms of performance between all used devices.

Agent	Sensor	Processor	Memory	Connectivity	OS
a_1	Solid state LiDAR	1.5 GHz	4 GB DDR-4	Ethernet	Rasp. Buster Lite
a_2	none	1.9 GHz	16 GB DDR-4	Ethernet	Ubuntu 18.04
a_3	PIR	216 MHz	512 KB SRAM	Ethernet	RIOT-OS 2020.04
a_4	none	216 MHz	512 KB SRAM	Ethernet	RIOT-OS 2020.04

Tab. 1: Characteristics of employed agents.

5 Evaluation

Based on the previously presented implementation, we can further specify the dependencies of a service by analyzing the flow of information $\mathbb{Z}$ which is supplied to services by agents. Fig. 5 depicts the dependencies for service s_2 provided by a_4 allowing the statement, that the service s_2 is available for as long as one element of $\mathbb{Z}_{a_4} = \{z_2, z_3\}$ is delivered frequently. Further, it can be seen that there is a transformation $\mathbb{P}_{1,2}$ from modality m_1, which is described with point cloud data, to modality m_2, which is a binary value representing *presence detected* or *absence detected*.

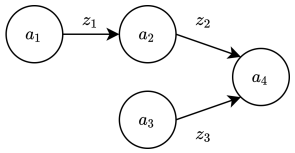

Fig. 5: A graph showing the dependencies for service s_2 provided by agent a_4.

We identify missing descriptions of occurring modalities in our implementation by analyzing each process step of our implementation. Fig. 6 shows each process step for the agents a_2 and a_3 and illustrates resulting outputs of information. By checking the communication with the broker b_1, it can be seen that clustered data is not shared and subsequently not described with a modality. Clustered depth data is interesting for services based on point clouds, since clustering is a major procedure in a multitude of algorithms used in depth data, e.g. a classification of objects.

Fig. 6 also serves as appropriate illustration to motivate the software management component presented in Section 3. Homogeneously designed applications could be distributed among network participants with lower computing capabilities by sharing the source code for each process step of a_2 to multiple other agents $a_f, ..., a_{g-1}, a_g$. An implementation could look similar to an already proposed solution in recent work [Me17].

In this paragraph we are going to outline the capability of implementing a resource discovery component on top of the existing implementation. As defined with the proposed concept, there are existing meta annotations describing modalities $\mathbb{M}_{a_i}$ of agents. These modalities

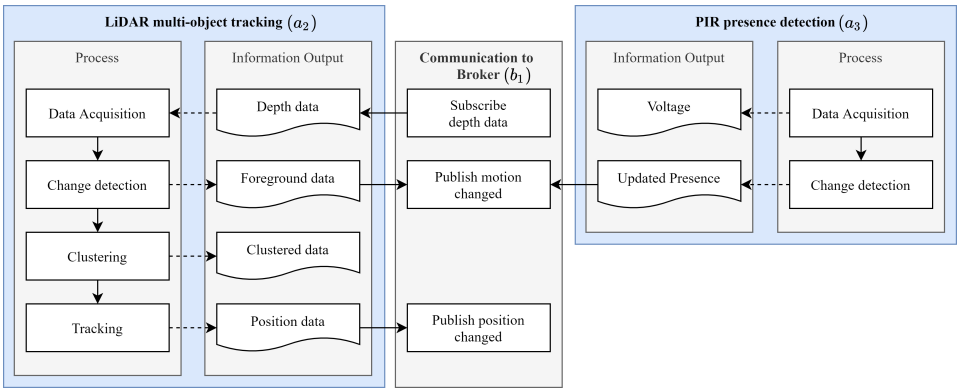

Fig. 6: Process steps and respective information outputs of agents a_2 and a_3.

can serve for querying certain types of information and brokers are able to provide a list of available topics which match the query. For now, modalities are represented by a simple identifier and a datatype and can be configured by developers. We recommend to use meaningful identifiers for modalities, since this enables to use semantic querying techniques while keeping the architecture lightweight.

6 Conclusion

Services in existing architectures often consume raw information and produce highly abstracted information for one specific application. Partial results are not shared which increases a system's redundancy because intermediate processing steps have to be carried out for each requirement on several occasions.

Our presented concept allows constrained devices and smart devices to cooperate even when running completely different sensors or actors. This was accomplished with an IAL component to transfer original raw information achieved from real world environments into a shared modality.

To highlight a use case for different services with different devices, we implemented a multi-object tracking with a LiDAR sensor and a presence detection using a PIR sensor. Then, we abstracted a modality describing raw information achieved by the LiDAR sensor into a binary shared modality *presence detected* or *absence detected*. The implementation was enabled by abstracting the presence information from depth data with a change detection step, which is a common procedure to apply in tracking services. Thus, and due to the proposed decentralized architecture, it is of little effort to share this abstracted presence information with other IoT devices.

Further work offers a multitude of use cases. Networks can be optimized by discovering redundant devices or by rating the services with quality metrics like availability, scalability, or resilience. With future work, we are going to focus on the implementation of remaining components, which were proposed with our architecture: access control, service discovery, and software management.

References

[Ba13] Baccelli, E.; Hahm, O.; Günes, M.; Wählisch, M.; Schmidt, T. C.: RIOT OS: Towards an OS for the Internet of Things. In: IEEE Conference on Computer Communications Workshops. Pp. 79–80, 2013.

[Ba18] Baccelli, E.; Gündoğan, C.; Hahm, O.; Kietzmann, P.; Lenders, M. S.; Petersen, H.; Schleiser, K.; Schmidt, T. C.; Wählisch, M.: RIOT: An Open Source Operating System for Low-End Embedded Devices in the IoT. IEEE Internet of Things Journal 5/6, pp. 4428–4440, 2018.

[BCS12] Bormann, C.; Castellani, A. P.; Shelby, Z.: CoAP: An Application Protocol for Billions of Tiny Internet Nodes. IEEE Internet Computing 16/2, pp. 62–67, 2012.

[BG18] Buschsieweke, M.; Güneş, M.: Access Control for Medical Devices: Tweaking LCap for Health Informatics. In: 2018 IEEE Globecom Workshops (GC Wkshps). Pp. 1–7, 2018.

[Ca14] Calbimonte, J. P.; Sarni, S.; Eberle, J.; Aberer, K.: XGSN: An Open-source Semantic Sensing Middleware for the Web of Things. In: International Workshop on Semantic Sensor Network. 2014.

[Le05] Levis, P.; Madden, S.; Polastre, J.; Szewczyk, R.; Whitehouse, K.; Woo, A.; Gay, D.; Hill, J.; Welsh, M.; Brewer, E.; Culler, D.: TinyOS: An Operating System for Sensor Networks. In: Ambient Intelligence. Springer, pp. 115–148, 2005.

[Me17] Mehta, A.; Baddour, R.; Svensson, F.; Gustafsson, H.; Elmroth, E.: Calvin Constrained — A Framework for IoT Applications in Heterogeneous Environments. In: 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS). Pp. 1063–1073, 2017.

[Ng17] Ngu, A. H.; Gutierrez, M.; Metsis, V.; Nepal, S.; Sheng, Q. Z.: IoT Middleware: A Survey on Issues and Enabling Technologies. IEEE Internet of Things Journal 4/1, pp. 1–20, 2017.

[PA15] Persson, P.; Angelsmark, O.: Calvin – Merging Cloud and IoT. Procedia Computer Science 52/, pp. 210–217, 2015.

[Pa19] Palanisamy, P.: Multiple Object Tracking from Point Clouds v1.0.2, 2019, URL: https://github.com/praveen-palanisamy/multiple-object-tracking-lidar.

[RC11] Rusu, R. B.; Cousins, S.: 3D is here: Point Cloud Library (PCL). In: IEEE International Conference on Robotics and Automation. Pp. 1–4, 2011.

[So15] Soldatos, J.; Kefalakis, N.; Hauswirth, M.; Serrano, M.; Calbimonte, J. P.; Riahi, M.; Aberer, K.; Jayaraman, P. P.; Zaslavsky, A.; Žarko, I. P.; Skorin-Kapov, L.; Herzog, R.: OpenIoT: Open Source Internet-of-Things in the Cloud. Springer, 2015.