UNIVERSITY OF LIVERPOOL

An Open Resource-Oriented Architecture for Decentralized Cyber Physical Systems

Thesis submitted in accordance with the requirements of the University of Liverpool for the degree of Doctor in Philosophy by

Yuji Dong
Supervisor: Dr. Kaiyu Wan
Co-Supervisor: Prof. Yong Yue, Dr. Dominik Wojtczak

June 2018

Abstract

Cyber Physical Systems (CPS) are complex hybrid systems integrating physics-based and digital worlds. Compared to the Internet of Things (IoT), CPS applications often have many complex physical behaviours, which makes the solutions monolithic and complex; thus making the developed CPS applications difficult to reuse. Furthermore, the classical isolated model-based design can be complicated when developing decentralised CPS applications. To lower the entry-barrier for developing the CPS applications, this thesis proposes an Open Resource-Oriented Architecture (OROA) to build the infrastructures to develop further varied CPS applications. Stakeholders, like developers, can exploit the advantages from the infrastructures for some features like low entry barrier, scalability and interoperability. This work is inspired by the REST (REpresentational State Transfer) architectural style, which is the network-based software architecture designed for the web.

In recent years, the REST architectural style is also widely used in IoT applications since many of these have similar requirements as web applications. However, REST architectural style has many limitations when dealing with physical environments, because most current IoT applications are data-centric without complex physical behaviours. To meet these challenges, we discussed lacking design principles and features in the REST architectural style to develop the CPS applications such as abstracting ability, uncertainty handling and access control. We proposed a solution for each design principle, and eventually used the resource concept and Semantic Web technologies to accommodate all the requirements and implement the Rinfra framework to build the infrastructures for decentralised CPS applications. To cooperate with the designed RInfra (Resource Infrastructure), Constrained Application Protocol (CoAP) was extended to support context-adaptation.

Scenarios in the smart transport and smart building have been presented as case studies to explain how the proposed architecture works and how it can benefit the decentralised CPS applications in heterogeneity, extensibility, interoperation and adaptation.

Acknowledgements

Firstly, I would like to express my sincere gratitude to my primary supervisor Dr. Kaiyu Wan for her continuous support in my PhD research and guidance throughout my journey. I would also like to extend my thanks to my co-supervisors Prof. Yong Yue and Dr. Dominik Wojtczak for all their support.

I wish to thank my fellow labmates for their stimulating input, the sleepless nights when working together to meet deadlines and for the joyous times we have shared in the last four years.

Finally, I am eternally grateful to my dear parents; Mrs. Lafang Jiang and Mr. Jiagen Dong for their unconditional love, support and patience throughout my life and education.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Cyber Physical Systems (CPS) are designed to integrate physical components with computations constructed in the large distributed network. They are playing increasingly important roles in a broad range of industries such as energy, infrastructure, healthcare, manufacturing and military [135]. Differing from other similar concepts like Internet of Things (IoT) or Ubiquitous Computing, the concept of CPS is more fundamental about the interSection rather than union of the physical and the cyber worlds. This is because CPS need to combine engineering models and methods from different, older disciplines with the newly developed software and network based on computation. Generally speaking, the concept of CPS is similar yet more complicated compared to IoT. CPS have more fundamental concerns with integrating cyber and physical sides, therefore they usually contain more complex behaviours with control technologies compared to IoT applications. IoT applications are usually networks that can interconnect ordinary physical objects with identified addresses based on the traditional information carriers including internet and telecommunication networks. The interconnection and addresses are not necessarily required for CPS though [168]. CPS and IoT are similar, however, they have the different concerns.

Decentralised CPS are from the decentralised vision of CPS where networking plays a central role in the system. In Figure 1.1, the relationships between IoT, CPS and decentralised CPS are shown. We can consider the decentralised CPS as the complicated version of the IoT systems with control technologies involved. Smart Grid is one of the typical decentralised CPS using distributed sensing and control technologies to build future energy systems [85]. The decentralised CPS application is not only a type of CPS, it

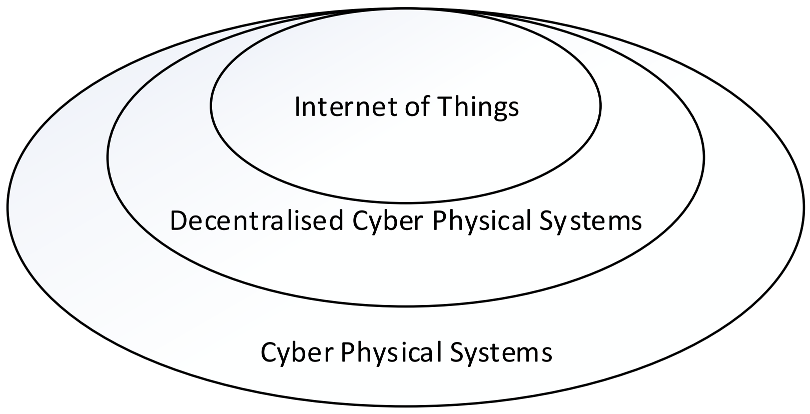


Figure 1.1: The Differences between CPS, IoT and Decentralised CPS

also stands for a different perspective of treating the CPS. Because the network can be easily reused by different applications, the decentralised CPS can easily reuse the tools and resources attached to the network infrastructures.

1.1 Motivation

According to the definition of CPS, computation, networking and physical processes are three essential parts. To combine them together, the general idea is to use one field as the core aspect and integrate the other two. For example, *cyberising the physical world* is to wrap software abstractions around physical subsystems. *Physicalising the cyber world* can endow software and network components with abstractions and interfaces that represent their dynamics in time. Following this kind of strategy, many different approaches are proposed for the design, development and deployment of the CPS from different perspectives. To deal with the complexity in developing CPS, model-based design [41] plays an essential role in modelling the behaviours for further development and analysis in recent years. However, the model-based CPS design is also a general approach which includes different modelling paradigms like the differential equations in the control theory or imperative programs in the computer science theory. The modelling process is very powerful and flexible with its paradigms and abstracting levels, while the relative solutions are not easy to be reused in other different applications. Especially in the decentralised CPS, where the networking plays an important role, modelling different behaviours is challenging and hardly reused because of the lack of a more unified approach. Because there are thousands of requirements involving many factors and dynamic participation of multiple stakeholders for complex decentralised CPS, existing approaches are not able to elicit, communicate, and maintain all of them.

The difficulty has already affected the development of the CPS markets. To lower the entry barrier for the CPS and thus to fully explore the demands, we propose to build a general infrastructure layer for the CPS from the networking perspective to reuse the different designs and developments of the CPS. Our approach is not only about the detailed technologies, but also about changing the perspective from either control-based CPS or software-based CPS to network-based CPS.

For the three different aspects - physical process, computation and networking in the CPS, existing approaches are mostly either from the control-based perspective for *physicalising the cyber world* or from the software-based perspective for *cyberising the physical world*. From the software-based perspective, an early illustration wraps the physical measurements of a sensor network in the database abstraction in [121] and with the development of computation ability and artificial intelligence, *cyberising the physical world* becomes a trend in many research fields especially for the heterogeneous environments. “Physicalising the cyber world”, is to endow software and network components with explicit Temporal Semantics based on control theory. To support Temporal Semantics, different programming languages like Equation-based Object Oriented Programming [31] and tools like PTIDES (Programming Temporally-Integrated Distributed Embedded Systems) [40] are proposed. Compared to the control-based and software-based perspectives, the network-based perspective can provide a more unified infrastructure foundation to the decentralised CPS regardless of their different functionalities. Thus different CPS applications can reuse the network and related tools. Since we intend to build the general infrastructure layer to support higher level CPS applications, the network-based perspective is a better choice. Furthermore, the design and development of CPS from a network-based perspective particularly fits the requirements of decentralised CPS.

Among all the existing network-based applications, architecture styles like data-flow style [7] and peer-to-peer style [161], the REST (REpresentational State Transfer) architecture style is one of the most successful architectures designed for the web. It satisfies the requirements including low entry barrier, extensibility, distributed hypermedia, anarchic scalability and independent deployment [58]. Ever since the publication of REST, this architecture has inspired a great deal of research and technological practices. It is also used to guide web technologies. For example, the RESTful web service is a strong competitor for SOAP/WS-* solution [143]. The REST architectural style also contributes to the field of IoT (Internet of Things), due to its advantages such as low entry barrier, decentralisation, scalability, robustness and easy deployment [65]. [66] has an experiment based on a programming exercise and feedback received from a group of sixty-nine computer science students who learned about RESTful and WS-* Web service, and implemented mobile phone applications that accessed sensor data from different sensor nodes using both approaches in teams. This result indicates that REST stands out as the favourite service technology in the context of the conducted study from the evaluation of the developers' preferences. In addition, the REST architectural style and its web implementations are widely used in many practices of IoT such as *Intelligent Buildings* [43], *Smart Homes* [97], *Smart Grids* [128] and *Smart Cities* [140]. Among the thirty-nine available IoT platforms that are surveyed in [129], only seven platforms do not have REST API.

Because of the similarity between the IoT and CPS, researchers also intend to use the REST architectural style in the CPS applications. However, the original REST architecture style is not sufficient for most of the complex CPS applications. Even though some RESTful approaches like the solutions in [117] or [150] are used in the CPS, they cannot be used to represent complex behaviours, because the advantages of REST architectural style conflict with some CPS fundamental requirements. In [52] the author explains the limitations of using REST-based Architecture to control a robot. For example, the unified interface and stateless interaction provide lower entry barrier and higher scalability. However, describing or implementing some complex behaviours especially related to the physical environment becomes very difficult, because only very limited simple operations like *GET* or *POST* are provided. The CPS usually integrate the continuous and discrete variables, and are compounded by the uncertainty in the physical environment. Furthermore, CPS should support possible autonomy and cooperation, therefore it is highly demanded that CPS should ensure safety, security, and reliability, which are not achieved in the REST style.

In this thesis, we extend the REST architecture style and propose the Open Resource-Oriented Architecture (OROA). Based on the OROA, we develop the RInfra (Resource Infrastructure) framework as a prototype to build the general resource infrastructure layer for the decentralised CPS from the network-based perspective to reuse the different designs and developments of the CPS applications.

1.2 Research Objectives

If the infrastructure layer is successfully built, the different CPS can be treated as different applications based on the infrastructure resources and configured with the appropriate mechanisms. Therefore, the infrastructures can be reused by different stakeholders to lower the entry barrier and inspire creativity. The general concept is illustrated in Figure 1.2. In particular, the resources are the meta-data to describe all the system components and therefore CPS can be developed, configured and maintained dynamically based on the resources.

Imagine a world where the general infrastructure resources are openly provided, just like the World Wide Web (WWW) today supporting the informational infrastructure resources. Here the general infrastructure resources can be any usable items such as heaters, self-driving cars, road lamps, delivery robots and rescue robots. Furthermore, the stakeholders can also reuse the resources to develop some further cross-domain decentralised CPS applications and provide more complex services. For example, the self-driving cars can be cooperated with other software services to provide the taxi service. The firemen can construct a temporal application with some rescue robots to rescue the people in the fire scenes. The decentralised CPS applications have a great potential, while we need the general infrastructural supports to explore people's creativity and high-level applications.

For the above scenes, the existing infrastructural supports from the web are not sufficient, especially due to the limitations of the REST architectural style. However, the REST architectural style still has its own advantages for providing the infrastructure resources. Furthermore, keeping compatible with the web technologies can be significantly helpful to popularise the software architecture and related technologies. This is because many softwares and services can be reused and many people are already well-trained in the REST architectural style.

The proposed Open Resource-Oriented Architecture is compatible with the REST architectural style as much as possible. To accommodate the different engineering requirements from the decentralised CPS applications, we first use some concepts in the REST architecture style. Then, based on the requirements in the decentralised CPS, especially for the continuous physical behaviour modelling, the uncertainty in the physical environment, and the access control issues, we design the OROA to build the resource infrastructure layer. The cross-domain CPS applications can be built based on the OROA with some specific services and configurations.

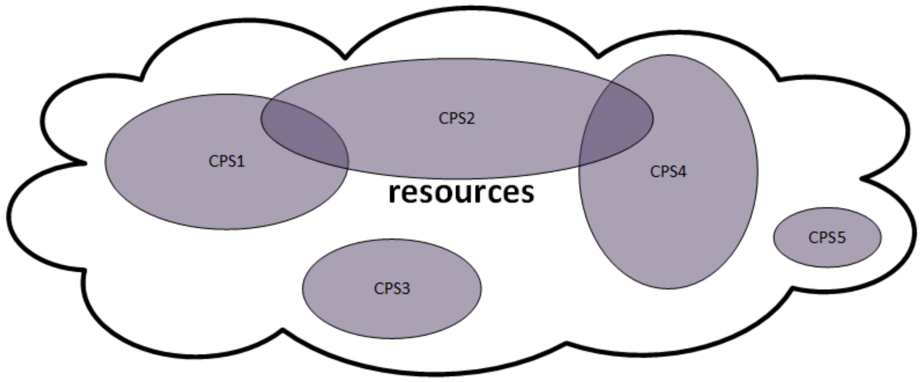


Figure 1.2: The Resource Layer to Support the CPS Applications

The whole OROA is also divided into several parts and each part is a separate approach for the IoT applications which are considered as the simplified decentralised CPS, therefore each part of the OROA can also be separately used in the IoT applications for different purposes such as system states estimation and fault detection. In the end, the integration of all features of OROA together can be used as a comprehensive solution for the decentralised CPS applications.

1.3 Contributions

In this Section, we outline the main contributions of the thesis towards developing the decentralised CPS applications:

1. Propose the Feedback-based Adaptive Service-Oriented Paradigm (FASOP) for the *Behaviour Abstraction* in Chapter 4.

   For the *Behaviour Abstraction*, we first express the issues of using the REST architectural style in the IoT applications, such as system states verification and physical behaviours implementation. Then we propose a Feedback-based Adaptive Service-Oriented Paradigm (FASOP) to solve these problems. The paradigm is intended to enhance the *Behaviour Abstraction* ability at a high abstract level, which can also be used in other environments.

2. Extend the Constrained Application Protocol (CoAP) to implement the FASOP for the *Behaviour Abstraction* in Chapter 5.

The FASOP is an abstract behaviour model which can be used in different ways. To fully apply the FASOP to the REST architectural style, we extend the CoAP to support real-time context adaptation for complex physical behaviour abstraction. The extended protocol includes four essential aspects: messaging model, message option, QoS and security.

3. Propose the semantic-based reputation framework for the *Uncertainty Handling* in Chapter 6.

   For *Uncertainty Handling*, the primary concern is to keep compatible with the original REST architectural style and web technologies. A significant difference between the web resources and the resources built in the Internet of Things, is that the entities mapped from the web resources can be guaranteed to be mostly correct while the resources built in the Internet of Things are not because the changing physical environments, unreliable devices and unreliable networks lead to much more uncertainty. To let the proposed OROA and the developed RInfra solve this problem, we propose to design the loosely-coupled reputation framework based on the semantic match, thus the different nodes in the large decentralised CPS applications can match together to construct new resource nodes to monitor each other. The constructed networks can use data fusion to provide more accurate results, and also self-adaptively detect any running fault in the applications.

4. Propose the Context-States-Aware Access Control for the *Usage Policies* in Chapter 7.

   The final required feature is the *Complex Usage Policies*, which is a much more critical aspect in the CPS than the web. It is not only because the CPS applications are usually more safety-critical, but also because the scenarios in the CPS applications usually require more complex usage policies. For the particular requirements of providing the complex usage policies in the decentralised CPS applications, we propose a hybrid access control mechanism - Context-States-Aware Access Control, which combines different access control mechanism to satisfy the requirements of the decentralised CPS applications.

5. Seamlessly integrate the four different solutions together based on the Semantic Web technologies and implement the RInfra (Resource Infrastructure) framework in Chapter 8.

The solution and implementation of the Open Resource-Oriented Architecture for the decentralised CPS needs to include all proposed solutions for the missing functionalities, and we use the Semantic Web technologies to combine all of them. The RInfra is developed as the single software node in the decentralised CPS, and it is based on the Jena framework. Any RInfra deployed in the decentralised CPS applications is a resource registry which stores all the related metadata. Based on the metadata provided by the RInfra and the provided mechanisms, the nodes can operate the appropriately registered and delegated resources in the particular RInfra. With many decentralised deployed RInfras, we can build the complex infrastructure resource layer, and more high-level CPS applications can be developed based on the registered resources.

The above contributions have been published or submitted for publication as academic papers:

Submitted for publication:

- Yuji Dong, Kaiyu Wan, Yong Yue, Xin Huang and Shiyao Zhang, "Open Resource-Oriented Architecture for the Decentralized Cyber Physical Systems

- Yuji Dong, Kaiyu Wan, Yong Yue and Xin Huang, "Support Context-Adaptation in the Constrained Application Protocol (CoAP) : Chapter 5

Accepted or published:

- Yuji Dong and Kaiyu Wan, "Semantic-based Reputation Framework for the Internet of Things, Journal of Universal Computer Science, 2018 [46] : Chapter 6

- Yuji Dong, Kaiyu Wan, Xin Huang, and Yong Yue, "Contexts-States-Aware Access Control for Internet of Things, International Conference on Computer Supported Cooperative Work in Design, May 9-11, 2018, Nanjing, China [47] : Chapter 7

- Yuji Dong and Kaiyu Wan, "Reputation-based Framework with Semantic Match for the Internet of Things", The International Conference on Recent Advancements in Computing, IoT and Computer Engineering Technology, 2017 [45] : Chapter 6

- Yuji Dong, Kaiyu Wan, and Yong Yue, "A Feedback-based Adaptive Service-Oriented Paradigm for the Internet of Things, International Conference on Service-Oriented

Computing (ICSOC), Workshop (ISyCC 2017), Malaga, Nov 13-17, 2017 [50] : Chapter 4

Some papers are also published as the early work in the field of resource-centric service-oriented architecture for CPS:

- Kaiyu Wan, Yuji Dong, Qian Chang, and Tengfei Qian. "Applying a dynamic resource supply model in a smart grid." Algorithms 7, no. 3 (2014): 471-491. [184]
- Yuji Dong, Kaiyu Wan, and Yong Yue. "A Dynamic Resource Supply Model towards Cyber Physical System (CPS)." In Computer, Consumer and Control (IS3C), 2014 International Symposium on, pp. 183-186. IEEE, 2014. [48]
- Kaiyu Wan, Vangalur Alagar, and Yuji Dong. "Specifying Resource-Centric Services in Cyber Physical Systems." Transactions on Engineering Technologies. Springer, Dordrecht, 2014. [183]
- Yuji Dong, Kaiyu Wan, and Yong Yue. "Unified Dynamic Resource Supply Model to Support Cyber Physical System." In Proceedings of the International MultiConference of Engineers and Computer Scientists, vol. 2. 2014. [49]

1.4 Organization

The rest of the thesis is organised as follows. Chapter 2 describes the background of CPS and related works in the literature. The overview of the proposed architecture is explained in Chapter 3. From Chapter 4 to Chapter 7, the four approaches for the issues from applying REST in the CPS are proposed, including behavioural abstract support from the paradigm, behavioural abstract support from protocol, uncertainty handling and access control. Chapter 8 then explains the overall implementation based on Semantic Web technology and gives the case studies with OROA. Chapter 9 concludes the thesis and discusses the future works.

Chapter 2

Related Works

2.1 Backgrounds

The term "Cyber Physical Systems" emerged around 2006 when it was coined by Helen Gill at the National Science Foundation in the United States. Unlike more traditional embedded systems, a full-fledged CPS is typically designed as a network of interacting elements with physical input and output instead of as stand-alone devices. It is generally treated as a concept to integrate computer science, software engineering, control theory and networking together to support scientific foundation for future complex systems in the diverse fields like aerospace, automotive, chemical processes, civil infrastructure, energy, health-care, manufacturing, transportation, entertainment, and consumer appliances.

CPS is quite related to the currently popular terms such as the Internet of Things (IoT), Industry 4.0, Machine-to-Machine (M2M), Web of Things (WoT), and Fog computing. All of these reflect a vision of a technology that profoundly connects the physical world with the information world although they are from different perspectives. The concept of IoT is about connecting different "Things" from the Internet. Compared to WoT, IoT is based on the Communication Layer, while the WoT is based on the Application Layer. Industry 4.0 is focused on the manufacturing systems though. M2M wants to provide the direct connections between different things rather than Internet or Web. Fog computing is highly related to the communication equipment.

Compared to the above terms, the CPS is more fundamental because it is trying to fuse software engineering, control theory and networking theory to create a new foundation. In other words, CPS is about the interSection of all these theories, *not* the union.

2.2 Models and Methods for CPS

Modelling plays a central role in all the scientific and engineering fields, and it can provide the appropriate abstraction to simplify problems. However, the CPS applications usually contain physical processes which require the models and paradigms from the control theory. For example, when a helicopter is flying, the control of its motion in space needs models like Newtonian Mechanics and closed-loop control. The modelling and abstracting methods from computer science and control theory are different, and sometimes even conflicting. The limitations and issues have been pointed out by Lee in [112] and he also claims the requirements of rebuilding computing and networking abstractions to realise the full potential of CPS. In order to seamlessly integrate computing, networking and physical processes, many different fundamental models and related methods are proposed from different perspectives.

In the CPS, because of the involved physical behaviours, the classical computation models are not able to correctly express the effect of timing. From the higher abstracting level, the theoretical formalisms include different models changing from classical Finite-State Machine [101], Petri Net [145], Process Algebra [19] to some time-sensitive models like Hybrid Automata [76], Timed Automata [6], Timed Petri Net [186], and Real-Time Process Algebra [9], thus the computation models can be more expressive for complex physical behaviours.

From the lower abstracting level, the design and development of CPS can use different models such as the Agent Model [137], Event Model [172], Actor Model [4], Component Model [59] and Service Model [178] depending on the specific requirements.

With the agent model, Sztipanovits presents a theory of composition for the heterogeneous systems [170]. The approach is a passivity-based design inspired by the control theory to decouple stability from timing uncertainties caused by networking and computation. The robustness advantage is exploited in the design of a networked multi-agent system. Since the agent model can provide the autonomous and intelligent decision-making ability, it is usually used for self-organisation, self-adaptation to solve heterogeneity or dependencies in CPS. In [118], Lin presents a multi-agent model for CPS, where the sensor network can provide information about the physical processes as the cyberinfrastructure to support semantic capabilities. The multi-agent model can provide meaningful semantics to support the static structure and dynamic behaviour of the CPS applications. To build efficient energy management system in the building structures, [205] uses a multi-agent decision-making control methodology for the energy optimisation in electrical, heating, and cooling energy zones. For manufacturing systems, [203] utilises a smart machine agent combined with the self-organising model and self-adaptive model to improve the reconfigurability and responsiveness of a Cyber Physical System designed for manufacturing shop floor.

The event model is another powerful model where different properties can be easily attached, thus it is also widely used in the CPS. In [172], the temporal and spatial properties of events are explored, and a layered Spatio-temporal Event Model is developed as a function of attribute-based, temporal, and spatial event condition for CPS. It helps to relieve the heterogeneity of CPS with the formal temporal and spatial analysis. In [173], the model is richer as a concept Lattice-based Event Model with more theoretical support for the CPS designs. In this model, a CPS event is uniformly described by the event type with its internal attributes and external attributes. Based on a CPS concept lattice, the CPS event can be composed of a set of event composition rules. This approach illustrates some advantages from the event model such as flexibility, QoS support, and complexity. The concept lattice-based event model uses the traditional first-order logic to specify rules compositions, however, using first-order logic may cause inconsistency in rules compositions. To solve these problems, an adaptive discrete event model is proposed in [194] to overcome possible inherent inconsistencies in composing rules by using discrete event calculus. Furthermore, the abnormal events are defined to provide the adaptation in the CPS to handle unexpected events.

The actor model is a useful abstraction for concurrent programming with message passing, especially in the distributed and heterogeneous systems. In the CPS applications, the actor model is also used for the heterogeneity and complexity of the developments. By focusing on the interactions between diverse models to reduce the difference between different models to achieve a well-defined composition, the actor semantics [114] is expressed as the abstract semantics to handle many heterogeneous models. By adopting a service-oriented computing approach with the actor model, the middleware services can improve the portability and enable the creation of heterogeneous CPS applications [127]. The actor model can improve programmability of the complex CPS applications. In [10], a coordinated actor-based approach is proposed to build a reusable and scalable model for self-adaptive CPS applications. The MAPE-K feedback loop is extended with the interactions between the actors to handle the unexpected changes with predicting behaviours of the model. SenseWeb [63] offers a platform for people to share their sensory readings using a Web service to transmit data onto a central server based on a centralised repository. In this approach, devices are considered as passive actors which are only able to push data.

The component model has been widely used in the traditional computer systems, and it can provide higher abstraction than the object model in the object-oriented programming languages. To compose the components using different paradigms and tools from a unified modelling framework, an integration language and its component-based design is presented in [108]. So components can be compatible with models from different tools, formalisms, and paradigms. For the dynamic adaptation in the CPS, μ-Kevoree, which is a component model based on "models@run.time" for micro-controllers, is designed to push dynamicity and elasticity concerns into resource-constrained devices for reconfiguration [59]. In [138], a component model called F6COM is designed explicitly for CPS applications operating in the highly dynamic, resource-constrained, and uncertain environment, and it is developed in the context of a cluster of fractionated spacecraft. For better comprehensive support to design and develop complex CPS, the OpenMETA toolchain is described in [171], and it can build the new integration layer to support the model integration, the tool integration and the design process integration.

The service model becomes increasingly popular now, especially in the web applications, because of its loosely-coupled intrinsic nature. To fit in the special requirements from CPS, a context-sensitive resource-explicit service model [81] is designed, and the corresponding composition formalisms are developed to help automate the composition process under real-time constraints as well as under various physical resource constraints. However, it is still challenging to compose the services provided from both the cyber and physical entities together to achieve specific goals, therefore a Physical-Entity (PE) service-oriented model [80] including the concepts of PE-ontology and PE-SOA specification, is designed to solve the problem. The composition process is expressed as a two-level compositional reasoning approach from both the abstract level and physical level. For better service discovery, selection and composition, a PT-SOA (Physical Things - Service Oriented Architecture) model is proposed in [207] with resource specifications based on the extended OWL-S where the advantages of ontology are explored more.

The above approaches are at a high abstract level which is more like engineering methodology abstraction. In this thesis, our approach is from a software architecture perspective which is a lower abstract level with more constraints and detailed paradigms.

2.3 Software Architecture for Decentralised CPS

Software architecture is an essential abstraction in the design and construction of any complex software systems as a guide to organise the system elements such as the components, connectors and data with specific constraints in their relationships to achieve the desired set of architectural properties. For CPS, especially decentralised CPS, many different software architectures are proposed for the special challenges.

The multi-agent technology has its advantages in the decentralised decision-making and control, therefore the Agent-Oriented Architecture is proposed in the decentralised CPS for the self-organisation and self-adaptation from smart decentralised controls. In [179], an architectural approach based on the technology of multi-agent systems is proposed to implement Cyber Physical Production System. The application demonstrator called myJoghurt is built and realised in cooperation with several different German research institutes. The Agent-Oriented Architecture is mostly used in the manufacturing CPS like in [25], [116], [187] and [18]. Most of the Agent-Oriented Architectures used in the manufacturing CPS are for self-organisation and self-adaptation based on the decentralised decision making because the multi-agent model can be easily integrated with artificial intelligence. Another CPS field which widely uses agent-oriented architecture, is the intelligent energy system, especially the smart grid. In [199] and [200], the agent-oriented architecture for smart automation is proposed for industrial practical automation architecture specific to the power-system automation. In [98], [205] and [84], the multi-agent approach is used in the smart building for building energy and comfort management where the agent model is mainly used for resource optimisation based on the decentralised decision making.

From the perspective of component-based software architectures, many different software architectures which are defined as a set of system components, connectors and constraints, are proposed for CPS. In [152], the Acme ADL [61], which has strong specification support for flexible software architectures, is extended by adding additional elements and rules for CPS. With the support from the new tool, i.e., AcmeStudio, the implementation of the CPS architectural style includes the behavioral annotations from either Finite State Processes (FSP) or Linear Hybrid Automata (LHA), while the Labeled Transition System Analyzer (LTSA) and Polyhedral Hybrid Automata Verifier (PHAVer) can be respectively used for behavior analysis. For the dynamic behaviours responding to and influencing the environment, the Distributed Emergent Ensembles of Components (DEECo) [23] is proposed to replace typical system configuration with dynamic component assemblies defined by the predicate-based membership. To explore the runtime dynamic from DEECo [23], a strengthening architecture of smart CPS is proposed with the SOFA NG component model [24] which is extended from SOFA 2 and DEECo. The architecture focuses on the dynamic by modelling the CPS as runtime product-lines and utilising the benefits of explicit architectures of hierarchical components to the design of smart CPS. However, the physical continuities are abstracted away in the SOFA NG component model, so it is difficult to verify the systems' physical behaviours. To fit the continuous physical behaviours, formal architectural abstractions of hybrid programs and formulas are built to analyse hybrid programs at the component level in [158]. Based on the formal architectural abstractions, some verifications are applied and analysed in the CPS. For example, the local safety in the system components can be verified individually and composed together to construct the whole system to satisfy a global contract in [133]. The rich architectural description is used in [157] to analyse the dependency loops and resolve such loops. For some other concerns in the CPS, in [151] a multi-view architecture framework is proposed to capture various aspects of the systems design to support heterogeneity. Different models are treated as views of the system structure, and the consistency among different views are guaranteed by structural and semantic mappings to enable system-level verification in a hierarchical and compositional manner.

With the development of cloud computing, many software architectures coping with cloud are designed for CPS. In [181], a multi-layer cloud-assisted context-aware architecture is proposed with two crucial service components, vehicular social networks and context-aware vehicular security. In [74], the approach for vehicular Cyber Physical Systems is more focused on data with cloud services. For manufacturing CPS, cloud computing plays an important role that even brings a concept of cloud manufacturing [191]. Therefore, there are plenty of different solutions [176, 188, 79].

Service Oriented Architecture (SOA) has been a widely used approach in the CPS because the service model can provide the appropriate abstraction to coordinate both the computational and physical parts of a system. Via QoS (Quality-of-Service), the limitation from the resource-constraint embedded devices can also be controlled. In [105], a three-tier SOA consisting of Environment Tier, Control Tier, and Service Tier, is proposed with cloud computing. The physical components and services are monitored in the Control Tier to support dynamic composition to ensure service adaptability. In general, most of the service-oriented CPS contain three layers of the access layer, service layer and application layer and many approaches have generic functional designs in SOA for CPS [92, 119, 91].

However, since the CPS is a concept consisting of a set of different fields and different types of CPS usually have different requirements, many different SOA approaches have their own domain-specific designs for the related CPS. For manufacturing CPS applications, an agile automation architecture is proposed in [208] based on SOA for CPS to support dynamical changes of the manufacturing equipment and the automation software with low effort. As a more mature solution, [175] proposes a framework, i.e., new IT-driven service-oriented smart manufacturing with different service-based technologies, to achieve the concept of "Manufacturing-as-a-Service". For smart transport, a set of different SOA solutions are designed for different functionalities such as accident management [115], vehicle-to-cloud for driving assistant [96] and integrating to mobile cloud [180, 120].

Resource Oriented Architecture (ROA) could be somehow categorised in the Service Oriented Architecture, whereas the difference is that ROA is usually based on the Web compared to SOA. To utilise the existing infrastructures and technologies at most, the architecture proposed in this thesis is a type of Resource-Oriented Architecture, and some related works are discussed in the next Section.

2.4 Resource Oriented Architecture for Internet of Things

IoT is generally less complicated than CPS, so lots of lightweight solutions with lower entry barrier are proposed. Dominique Guinard proposes a resource-oriented architecture for the web of things based on the RESTful principles [65]. It makes many current web technologies usable for the IoT applications and realises a lower entry barrier. Because of the advantages from the REST architecture style designed for web, the resource-oriented architecture becomes popular in many different solutions for the web of things. In [123, 122], the Resource-Oriented Architecture is proposed to easily mash-up constrained application protocol (CoAP) resources and virtualise the physical devices into their own digital virtualisation thus the system can provide better inter-operation. To build ubiquitous WoT applications that work in and across multiple environments, the author of [134] proposes a distributed resource management architecture and implement the architecture to a WoT platform named *uBox*. In [34], resource-oriented architecture is integrated with the multi-agent technology, and the approach enables to design some applications from high-level abstractions.

The success of using web technologies in the Internet of Things also inspires people of using the Semantic Web technologies. In [146], the author proposes an architecture of Semantic Web of Things that makes the deployment and use of semantic applications involving Internet-connected sensors via building, searching and reading web pages. Subsequently, more approaches with Semantic Web technologies are proposed in the Internet of Things from different perspectives. The work in [95] focuses on the inter-operability for Pervasive Computing and Internet of Things. In [110], the author presents a Linked Stream Middleware to integrate time-dependent stream data into the Semantic Web Things. In [70] a framework is built based on the Semantic Web technologies to explicitly describe the meaning of sensor measurements to interpret sensor data and to combine domain knowledges.

The Resource-Oriented Architecture is mostly used in the Internet of Things, or Web of Things, not Cyber Physical System, because some of its architectural constraints are conflicted with the CPS requirements. Figure 1.1 has already indicated the inclusive relationship between the IoT, CPS and decentralised CPS. In the next Chapter, we will discuss the limitations of the REST architectural style in the decentralised CPS and describe the overview of the proposed OROA for the decentralised CPS. After the overview, some crucial features of OROA are further explained in different Chapters respectively and in each Chapter, the related works about the specific feature of OROA are given respectively.

Chapter 3

Open Resource-Oriented Architecture Overview

The proposed Open Resource Oriented Architecture can be treated as an extension of the REST Architectural Style for the decentralised CPS applications. In this Chapter, we first introduce the REST Architectural Style with details and the related Resource Oriented Architecture that is usually used in the Web of Things. Next, the limitations of the classical Resource Oriented Architecture in the CPS are discussed, and then our approach is described.

3.1 REST Architectural Style

The REST Architectural Style was first proposed in the Dr. Roy Fielding's PhD thesis: Architectural Styles and the Design of Network-based Software Architectures [58]. The REST Architectural Style has been broadly used, as discussed in Chapter 1 and Chapter 2. In order to apply the REST architectural style, several constraints should be followed in the following ways;

Client-Server The Client-Server is the first constraint in the REST architectural style, which can separate the user interface concerns from the data storage concerns. Therefore the portability of the user interface is improved across multiple platforms and the server component can be simplified for scalability improvement.

Stateless Communication must be stateless, which means each request from the client to the server must contain all of the necessary information to understand the request, thus the requests cannot take advantage of any stored context on the server. This constraint is intended to achieve the features such as visibility, reliability, and scalability.

Cache The cache constraint is used for network efficiency. That is, the data within a response to a request is labelled as *cacheable* or *non-cacheable* implicitly or explicitly. If a response is cacheable, then a client cache is assigned the right to reuse that response data for the following equivalent requests. The advantage of adding cache constraints is that they have the potential to partially or completely eliminate some interactions, thus improve efficiency, scalability, and user-perceived performance by reducing the average latency of a series of interactions. The trade-off, however, is that a cache can decrease reliability if stale data within the cache differs significantly from the data that would have been obtained had the request been sent directly to the server.

Uniform Interface The central feature that distinguishes the REST architectural style from other network-based styles is its emphasis on a uniform interface between components. By applying the software engineering principle of generality to the component interface, the overall system architecture is simplified and the visibility of interactions is improved. Implementations are decoupled from the services they provide, which encourages independent evolvability. In order to obtain a uniform interface, multiple architectural constraints are needed to guide the behaviour of components. REST is defined by four interface constraints: identification of resources; manipulation of resources through representations; self-descriptive messages; and, hypermedia as the engine of application state.

Layered System The layered system constraint is added to further improve behaviour for Internet-scale requirements. The layered system style allows an architecture to be composed of hierarchical layers by constraining component behaviours. For example, each component cannot *see* beyond the immediate layer with which they are interacting. By restricting knowledge of the system to a single layer, a bound is placed on the overall system complexity, and substrate-independence is promoted. Layers can be used to encapsulate legacy services and to protect new services from legacy clients. The components can then be simplified by moving infrequently used functionality to a shared intermediary. Intermediaries can also be used to improve system scalability by enabling load balancing of services across multiple networks and processors.

Code-On-Demand REST allows client functionality to be extended by downloading and executing code in the form of applets or scripts. This simplifies clients by reducing the number of features required to be pre-implemented. Allowing features to be downloaded after deployment improves system extensibility. However, it also reduces visibility, meaning this is only an optional constraint within REST.

The REST architectural style has also been used in many other systems other than the web. In order to apply the REST in the Resource-Oriented Architecture for Web of Things, the constraints of the REST are summarised as follows:

C1 Resource Identification The Web relies on Uniform Resource Identifiers (URI) to identify resources, therefore, links to resources (C4) can be established using a well-known identification scheme.

C2 Uniform Interface Resources should be available through a uniform interface with well-defined interaction semantics, i.e., Hypertext Transfer Protocol (HTTP). HTTP has a minimal set of methods with different semantics (safe, idempotent, and others), which allows interactions to be effectively optimised. The vast majority of Web-facing applications offer RESTful interfaces, while the back ends are implemented using different interaction models (such as database systems), and the same approach can be employed for the Web of Things.

C3 Self-Describing Messages Agreed-upon resource representation formats make it much easier for a decentralised system of clients and servers to interact without the need for individual negotiations. On the Web, media type support in HTTP and the Hypertext Markup Language (HTML) allow peers to cooperate without individual agreements. For machine-oriented services, media types such as the Extensible Markup Language (XML) and JavaScript Object Notation (JSON) have gained widespread support across services and client platforms. JSON is a lightweight alternative to XML that is widely used in Web 2.0 applications and directly parsable to JavaScript objects.

C4 Hypermedia Driving Application State Clients of RESTful services are supposed to follow links they find in resources to interact with services. This allows clients to explore a service without the need for dedicated discovery formats. It also allows clients to use standardised identifiers (C1) and a well-defined media type discovery process (C3) for their exploration of services. This constraint must be backed by well-defined resource representation (C3) methods in which they expose links that can be followed.

C5 Stateless Interactions This requires requests from clients to be self-contained, in the sense that all information to serve the request must be part of the request. HTTP implements this constraint because it has no concept beyond the request/response interaction pattern, however, there is no concept of HTTP sessions or transactions. It is important to observe that there may be state involved in an interaction, either in the form of state information embedded in the request (HTTP cookies), or in the form of server-side state that is linked from within the request content (C3). Even though these two patterns introduce states into the service, the interaction itself is completely self-contained, that is, it does not depend on the context for interpretation and is therefore stateless.

In HTTP, the uniform interface constraint (C2) has four principal operations, GET, PUT, POST, and DELETE. This can be mapped rather naturally in the Web of Things: GET is used to retrieve the representation of a resource, e.g., the current consumption of an electricity sensor. PUT is used to update the state of an existing resource or to create a resource by providing its identifier. For example, it can be used to turn a light on or off. DELETE is used to remove a resource. For example, it can be used to delete a threshold on a sensor or to shut down a device. Finally, POST is used to create a new resource, e.g., to create a new feed used to trace the location of a tagged object.

3.2 Issues of Applying REST in the Decentralised CPS Applications

Modelling, designing and developing CPS requires multi-level abstraction, which has only limited support in the REST architectural style. The semantics of the URIs can explicitly express the hierarchical structures of resources. However, it is difficult to provide the complex composition and at the same time partially expose some interfaces of the resources to the users. When REST only expresses the resources' logical positions via hierarchical URIs, if the decentralised CPS applications need to reuse many system components or services in different situations, giving correct abstraction is not always possible. This is because the same component may play different roles in different applications. Furthermore, dynamic physical behaviours, like breaking a car or moving a robot, are rarely supported by REST architectural style.

There is some uncertainty from physical devices and environments in CPS. For example, the resources in the Web, based on REST, usually assume that the resources and the mapped entities such as physical devices and software components can match correctly. However, in the physical environment, the devices naturally have interferences. Therefore, the resources mapping physical devices cannot always guarantee to produce the correct values. Furthermore, the devices in the physical environment have high damage risk.

In addition, some physical devices cannot be shared by many different users simultaneously, which is different from the web applications. These always allow and require highly concurrent visits. The REST architectural style does not have multiple policies to use the resources, and it can cause security issues or even system bugs.

To summarise, the following shows three mainly required functionalities of CPS that REST architectural style does not provide:

1. Stronger abstract ability from both the structural abstract and behaviour abstract.
2. Built-in mechanisms to handle uncertainty.
3. Built-in access control mechanisms.

To deal with the above issues, we adopt some concepts in the REST architectural style with some extra features and supports to meet the CPS requirements. In Table 3.1, we list four essential requirements that CPS systems must meet, the corresponding approaches that REST architectural style have taken, and the approaches we provide with the details discussed in the following Chapters.

3.3 Design Principles of Extending REST for Decentralised CPS

Figure 3.1 illustrates the overview of the OROA. The **Resource Registry** stores all the resources in an acyclic directed graph to describe both the mapped virtual objects and mapped physical objects, and the format can be found at the **Resource Template Library**.

The **Reputation Evaluate Engine** is used to integrate the reputation-based framework in the resource registry to provide the uncertainty handling mechanisms in data fusion and fault detection. The **Authorization Management** assists the token-based access mechanism in providing accurate and secure policies to use the resources. The **Resource Discovery** component is to provide the necessary functions in this architecture to look up a specific resource, and the **Reasoning Engine** is to provide some self-management mechanisms.

In the rest of this Section, we describe the design principles of the proposed OROA. These design principles give the constraints in the architecture level to build the resource infrastructure layer for further development.

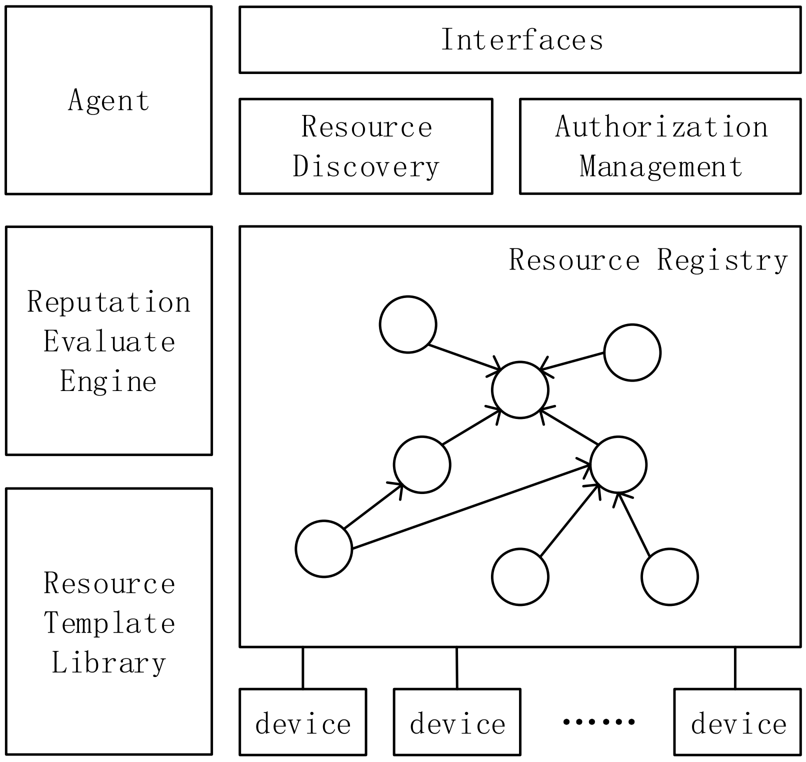


Figure 3.1: The Overview of OROA

3.3.1 Resource as the Core Concept

Resource Definition

The resource concept in our architecture is compatible with the abstraction in REST. A resource R is a temporally varying membership function $M_R(t)$, which for time t maps to a set of entities, or values, which are equivalent. The entities here can be virtual objects or physical devices like pictures, sensors or even software components, and the mapping resources can describe and address the mapped entities.

For any single resource, it has a unique identifier to address the resource. The solution like URI (Uniform Resource Identifiers) for naming, addressing and identifying resources has been developed for a long time [125]. A generic URI adopts the following form:

$$URI = scheme : [//host[: port]][/]path[?query][\#fragment]$$

However, typical URIs on the Web are usually only used to express the resources on the Web, but some physical devices which are not located on the web cannot use this representation. For those physical devices not located on the web (e.g. some devices connect via ZigBee network), we need to have a synchronisation mechanism to transfer the data. The users can have two ways of accessing the physical devices. If the physical devices are located on the web, we can use the URI to access the devices directly. Otherwise, the resources act as proxies to synchronise all information between the physical devices and the mapping resources. In this case, the system needs to support the required protocols (other than HTTP) to build the connection between the resource registry and the physical devices.

Resource Representation

In the traditional Web, resources are represented in Hypertext Markup Language (HTML). In the Internet of Things or decentralised Cyber Physical Systems, there are lots of different data models such as web mash-ups based on Web 2.0 [67], linked open data [109] or sensor data with domain knowledge [154].

In the OROA, the representation of resources are mixed with Web 2.0 technology such as HTML, structured format like XML or JSON, and linked data model like ontology or RDF[107] to accommodate different data models.

The use of XML is suggested to represent any simple resource document while using JSON to represent more lightweight data from the resources. The HTML can be used to build the visualised platform to perform the operations on the resources. The linked data type like RDF can express the structure and relationships of the resources for the resource compositions.

3.3.2 Structural Abstract - Resource Abstracting

Not all resources have to map to real devices in CPS, and one resource may map to a set of different entities, so the resources need an abstract mechanism to support meaningful semantics between different related resources.

In some RESTful architectures for the Internet of Things, the resources representing both physical objects and virtual objects are organised in a hierarchy structure [166], while in some Semantic Web of Things, the applications are designed and developed with semantic web technologies and the resources are organised in the acyclic directed graphs[190].

In our design, the resources are also organised in the acyclic directed graphs. However, different from the Semantic Web of Things approach, our approach does not require full semantic supports based on ontology. Most properties of the resources are not indicated in the graphs but in the resource representation, so the graphs can be condensed, and the barrier can be lower without full semantic supports. Fundamental constraints are claimed here, however, in the implementation part we use the existing RDF standards and technologies for convenience.

In the acyclic directed graph structure, any node is a resource and any directed edge stands for a composition relationship. If a resource node does not point to any other resources, it is mapped to a physical entity. That is, it is usually the connection between the physical and cyber world and can be treated as an endpoint for the physical environment.

For example, a self-driving car may contain a camera, ultrasonic, radar, laser, steering, braking, motors, GPS, etc. Any device in this car can be mapped to a resource to specify the functions or non-functional features of this device. However, a camera, ultrasonic, radar and laser may be grouped together to construct a sensor module which needs detailed specification and maps to a software component in the system. The Figure 3.2 is an example expressed with RDF. The usage of this kind of structural abstract is explained with more details in Chapter 8.

With different abstract levels, the granularity of software components can be more flexible. Any resource node can open its interface to reduce the traffic flow pressure from any particular component in this system. Some technologies like machine-to-machine (M2M) [78]

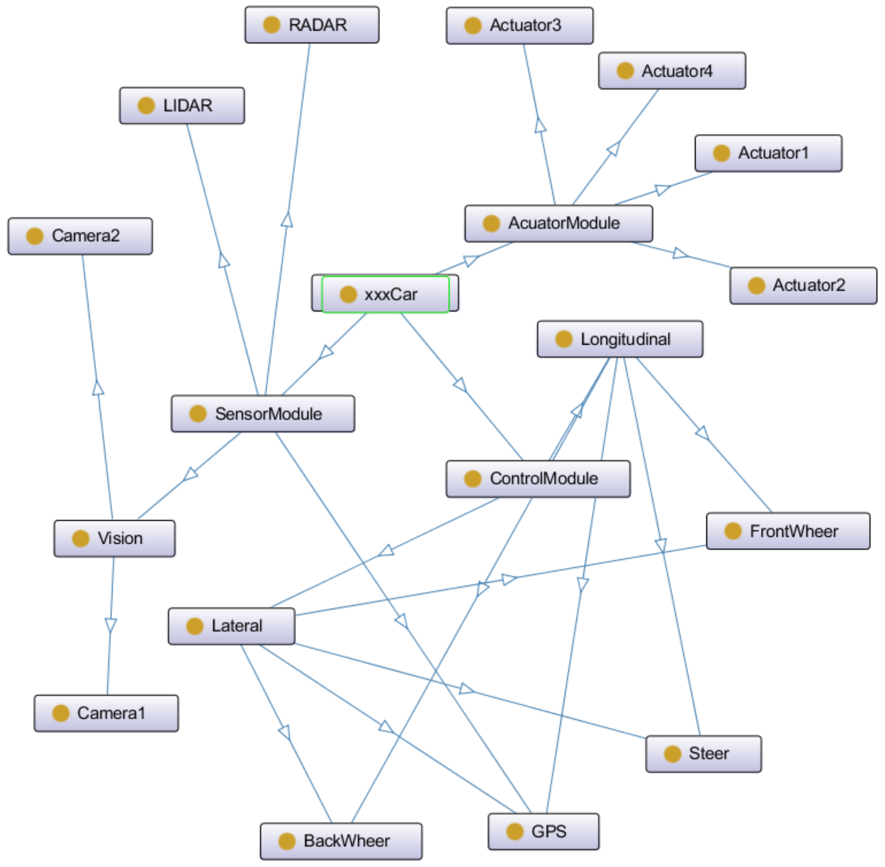


Figure 3.2: An Example of a xxxCar Abstracted from Many Lower Level Resources

can also cooperate well with this design, and it makes the whole system extremely flexible because it can always interact with other systems and environments at any abstract level.

3.3.3 Behavioural Abstract - Feedback-based Adaptive Service-Oriented Paradigm

The REST architecture already shows many advantages by using a unified interface in the software system to make the components' interactions scalable and simplify the software API development. However, compared to the hypermedia data, the physical environment contains much more complex physical behaviours like braking a car or cleaning the floor, which cannot be simply implemented by the original designed unified interface in the

REST. To enhance the physical behaviour modelling ability via the unified interface in the decentralised CPS applications, we define the *Feedback-based Adaptive Service-Oriented Paradigm* (FASOP) as the guide to design the related API, software systems and protocols. The detailed motivation examples are given in Chapter 4.

The FASOP is specially designed for a type of actuating objects which can make active operations that may disturb the environments. Different from most virtual objects or perceptive objects like sensors, the operations over the actuating objects usually need the environmental feedback to adapt for the specific purposes based on control theory.

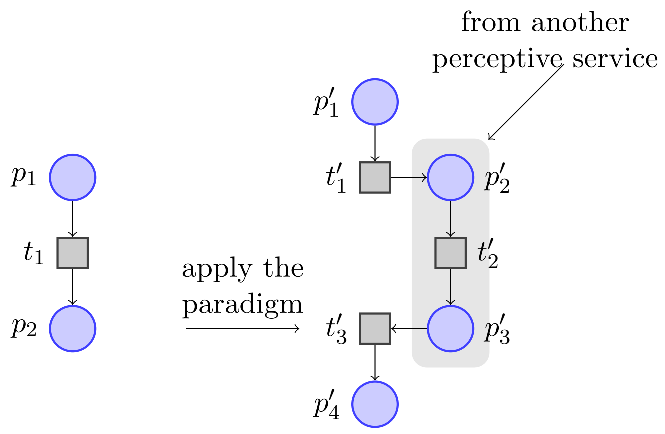


Figure 3.3: The Petri Net Behavioural Model for an Actuating Service

We use the Petri Net to describe the behavioural model for complex physical behaviours. As shown in the left side of Figure 3.3, in the traditional REST interface, t_1 is a transition provided by the *Actuating Objects* and p_1, p_2 are pre-condition and post-conditions of t_1 respectively. From the process point of view, if the operation in t_1 is a function call $< result : func(params..) >$, then p_1 is usually to match the function name $func$ and parameters $(params..)$, and p_2 is to check the return value $result$. However, if the $< result : func(params..) >$ has any action in physical environment, it is nearly impossible to guarantee all the post-conditions from programming language level, because the post-conditions of t_1 may contain some physical affects.

To solve these problems, we construct a feedback-loop by requesting another resource or external service to obtain the environment changing information. The importance of the feedback-loop in the self-adaptive systems has already been discussed in [21].

The behavioural model of the FASOP is on the right side of Figure 3.3. The service call at t_1 is changed from $< result : func(params..) >$ to $< result : func(params.., PS.funcPS, t) >$, which contains another function $funcPS$ from another service to detect the changing environment and the latency time t is the waiting time to get the feedback perceptions. In this way, the verification of post-conditions is more reasonable, since the post-conditions with physical properties can be verified through the perceptions of the physical environment. Then the *place* p'_2 can verify whether the operation $func$ is operated successfully and *place* p'_4 can eventually check if the operation $func$ has desired behaviours based on the perceptions. In Figure 3.4, a sequence diagram shows a process from the implementation perspective.

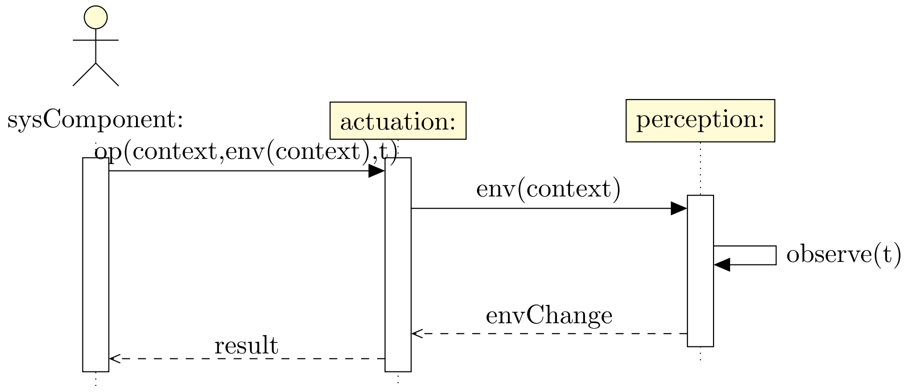


Figure 3.4: The Sequence Diagram for Calling a Actuating Service

3.3.4 Uncertainty Handling - Reputation Framework

The physical environment brings a great deal of uncertainty in CPS, therefore, dealing with the uncertainty and provide fault tolerance has been an issue in CPS. The most common uncertainty is from the physical devices like sensors' noise and other unpredictable faults due to different reasons like device damages. In order to limit fault-propagation due to unreliable components in the CPS architectures, Crenshaw et al. [36] propose a simplex reference model containing of external context, domain model, machine and safety requirements. This model has widely influenced all the fault tolerance models in CPS. However, this model is a co-design solution taking both hardware and software into consideration simultaneously, which is not suitable for our architecture. Furthermore, it cannot effectively use our multi-level abstraction. On the other hand, compared to model-based mechanisms for uncertainty, the data-driven approaches like reputation-based methods support a more common principle of looking for consistency among the data reading in the set without extra specific models to provide self-adaptation and scalability. Since [60] used this approach in the Wireless Sensor Network (WSN), the reputation-based mechanism has been used in many solutions.

However, the reputation-based framework is rarely used in the IoT or decentralised CPS. This is because, compared to all similar functionality sensors in the WSN, the heterogeneity in the decentralised CPS makes it difficult to check the consensus between different system components and to evaluate each reputation.

In our architecture, since the resources are at the infrastructure layer and should run in the long-term, the reputation-based framework is then able and suitable to be integrated into the architecture. We use the *Belief* property in the resource registry to claim the accurate possibility for each resource. In fact, the belief inaccuracy estimation was investigated in [5] to drive architectural adaptation aiming to increase the dependability of the running CPS systems. Moreover, our architecture can construct a directed graph to build the structures to check consistency in the resource registry. Based on all the historical data and appropriate algorithms, we can have the belief value in each physical resource to support further development as part of the contracts. In the following subSections, three necessary steps are briefly introduced to explain how to include the reputation-based framework in the architecture. This Section only includes the structural description. More details are covered in Chapter 6.

Semantic Match

The reputation-based uncertainty handling is one of the classical methods for data fusion and fault detection in wireless sensor network. However, in the highly heterogeneous systems, the system components have different data types, thus it is difficult to find the meaningful consistency checking between different data in the decentralised CPS.

Via the resource registry with the directed graph structure, we can construct the consistency checking structure based on the semantics from different system components. A directed graph sample with six individual components and four resources is illustrated in Figure 3.5. The system components which point to the same resource have the same semantic, therefore they can compare the values for further data fusion and fault detection.

To give a formal description, any arrow a_{ij} from c_i to r_j indicates a functionality of the

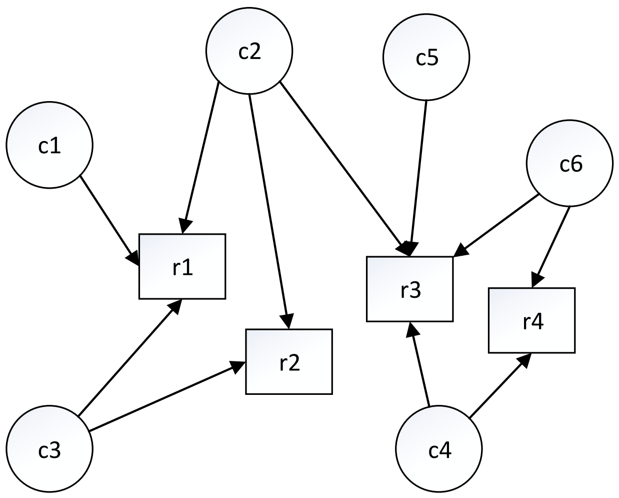


Figure 3.5: The Semantic Match to Construct the Consistence Check

component c_i as a sequence of data with variable time t

$$f_{a_{ij}}(t) = (\vec{d}_{c_i,r_j,t_1}, \vec{d}_{c_i,r_j,t_2}, ..., \vec{d}_{c_i,r_j,t_k}) \tag{3.1}$$

Where any $\vec{d}_{c_i,r_j,t_p} =< v_1, v_2, ..., v_m >$ is a vector produced from component c_i at time t_p. Any resource r_μ has a unique URI_{r_μ}. At time t_p, it receives α number of data from α number of system components and the received data in resource r_μ is expressed as a set of $\vec{D}_{r_\mu,t_p}$:

$$\vec{D}_{r_\mu,t_p} = \{\vec{d}_{c_1,r_\mu,t_p}, \vec{d}_{c_2,r_\mu,t_p}, ..., \vec{d}_{c_\alpha,r_\mu,t_p}\} \tag{3.2}$$

Generally, any system component c_i can have several different functionalities mapped to different conceptual resources. Each functionality can produce a vector of data $\vec{d}_{c_i,r,t_p}$ at time t_p and the data will be sent to the resource r. Further more, each system component c_i has a special property b_{c_i} as the *Belief* of the component c_i assigned from the matching resources. Because every system component has its mapping resources, the mapping resources can keep the *Belief* property updated. Essentially, the process of the *Semantic Match* is to add some extra resource nodes in the resource registry and construct the specified framework to check consistency between different components with the same functionality. To elaborate, for example, the hybrid localisation solution [197] that the GPS positioning module, WiFi positioning module and Cellular positioning module can all map to a concept of the entity location as three semantically equivalent components. This solution cannot only make the localisation more accurate and reliable, but also make it possible to detect any abnormal fault from these three modules as shown in the Figure 3.6. Because in the long-run, if any module suddenly starts to give unbelievable drift values compared to the other two modules, the module is considered to fail.

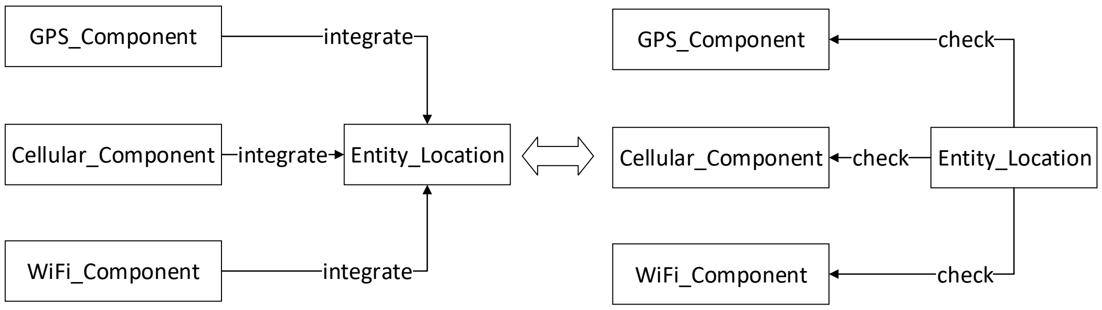


Figure 3.6: The Hybrid Localisation Solution as a Semantic Match Example

Data Fusion

After the *Semantic Match*, some resources that have different sources of data express the same concept, therefore it is possible to integrate different data to achieve more accurate and reliable data. There are many different approaches for the data fusion that can be integrated into our architecture since we have real-time updated *Belief* property for each source. This is a well-studied field, and some popular strategies including fuzzy set theory, Bayesian theory and Dempster-Shafer theory [94] can all be used depending on the systems' running contexts.

Belief Updating

Based on all the data produced from different components in the same *Semantic Match*, the resource can update its *Belief*. We assume the following:

- If more resources think one component is correct, the component has a higher reputation.

- If the data from different components are more consistent, these components are more reliable than others and will get the higher reputation.

For any component $c \in C_{r_i}$, where C_{r_i} is the set of all semantic equivalent components to resource r_i, we have the *Belief* of this component c as:

$$b_{c_{r_i}} = \hbar(c, C_{r_i}) \tag{3.3}$$

Where the $\hbar(c, C_{r_i})$ is a function to check the difference between the data from the component c and other components in the C_{r_i}. The function will get a higher value if the difference is less, and a lower value if the difference is more. The equation means that if the component's values are more close to other components' which are semantic equivalent to a resource, the component's values are assumed to be more accurate.

If the component c matches to multiple resources and receives *Belief* updating from a set of different resources R_c, the *Belief* of this component c is:

$$b_c = g(B_{R_c}), B_{R_c} = \{b_{c_{r_i}} | r_i \in R_c\} \tag{3.4}$$

Where the $g(B_{R_c})$ is a function used to integrate different resources' opinions and the range is between 0 to 1. In general, this equation can be concluded as: if more resources think one component is reliable, it is so.

Fault Detect

We have a threshold *Belief* value b_{thr}. If any component' belief value is lower than b_{thr}, we assume this component is failed.

3.3.5 Usage Policies - Context-State-Aware Access Control

Different from the resources on the web, some physical resources have more complex usage policies. Take controlling a robot as an example; if two users try to control it at the same time, the commands may conflict thus the behaviours are not predictable. Furthermore, the commands for the actuators also cause more potential security issues, since dangerous operations may be made. For example, if the access permission of the motors in a car is misinformed, it may cause serious traffic accidents. Therefore the resource usage policies management should be one of the core features of the proposed OROA.

The proposed access control mechanism needs to have the following features since the CPS applications are in the open environment:

1. The authorisation of accessing any resource.
2. The fine-grained access control options for some complex resource accessing policies.
3. The context-awareness in accessing resources to adapt to the dynamic changing open environments.
4. To track all the resources' states to avoid the conflicts between the non-shareable objects like actuators.

To satisfy all the above requirements, we design the hybrid mechanism, i.e., *Contexts-States-Aware Access Control* (CSAAC). The CSAAC is based on the Attribute-Based Access Control to provide the fine-grained and flexible access control and with the extra Context-States-Awareness to adapt to the dynamic changing open environments. Figure 3.7 illustrates the conceptual Context-States-Aware Access Control Model and the model details are given in Chapter 7.

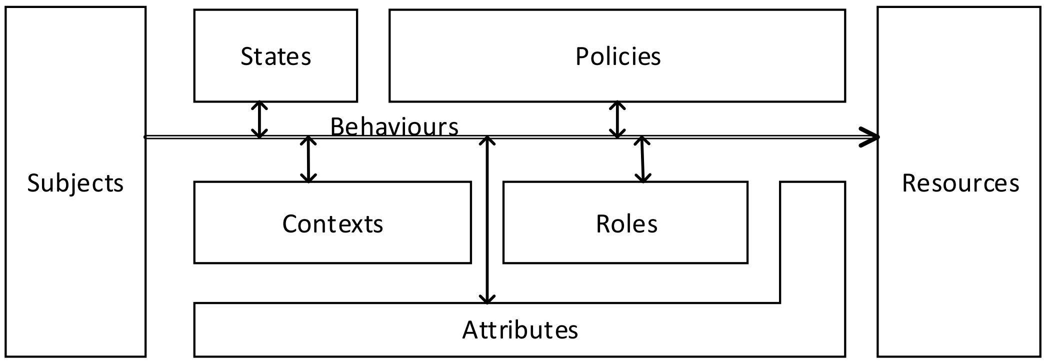


Figure 3.7: The Conceptual Context-States-Aware Access Control Model

3.4 Summary

In this Chapter, the overview of the proposed OROA is illustrated by briefly introducing important features. From Chapter 4 to Chapter 7, the three most important features in the OROA are explained respectively in detail. Each feature and related technologies can be separately used in the IoT applications which can be treated as the simplified decentralised CPS. On the other hand, because each individual feature in the OROA has its own limitations of dealing with issues in the decentralised CPS, the term 'CPS' is not used in the following Chapters. Instead, in each individual Chapter describing the separate feature of the OROA, the IoT applications are used as the discussion scenarios. That being said, the integration of all features of OROA together can be used as a comprehensive solution for the decentralised CPS applications. In Chapter 8, we discuss the implementation of OROA for the decentralised CPS applications and use case studies to illustrate that OROA can be used for the decentralised CPS applications.

Table 3.1: The Intended Changes from REST architectural style

The Essential Features CPS should support	The approaches that REST has taken	The approaches that in our design
Structural Abstract	Hierarchical Structure	Directed Graph Structure (RDF implementation)
Behavioural Abstract	Unified Interface	Feedback-based Context-Adaptive Paradigm (Chapter 4)
Behavioural Abstract	CoAP	Real-Time Context-Adaptive CoAP (Chapter 5)
Uncertainty Handling	None	Reputation-based Framework (Chapter 6)
Access Control	None	Contexts-States-Aware Access Control (Chapter 7)

Chapter 4

Behavioural Abstract

4.1 Introduction

IoT is envisioned to integrate the physical world into computer-based systems. Recently, with the advanced technology development on sensors, networking and data processing etc. IoT has illustrated a great potential in various fields [64]. However, even after decades of research on system aspects of the IoT, developing IoT based systems is still facing many challenges at high-level system requirements like scalability, inter-operability and fault tolerance [132]. Moreover, most current IoT applications are coping with data collecting and processing issues without involving many complex physical behaviours. This is due to current IoT solutions and usage scenarios still being very limited in modelling complex behaviours in continuously changing physical environment.

Context adaptation plays an important role in continuously changing physical environment. In recent years, because of the rapid development of mobile computing and big data, there are plenty of context-sensitive data in the IoT systems, therefore the context-awareness in IoT draws a lot of research attention. For example, there are many investigations on context-awareness in models [192], architectures [32] and middlewares [28]. On the other hand, adaptation is more challenging than context-awareness. It is usually solved by constructing the feedback loop [21] at different abstracting levels like architectures [27], behaviour models [26] and frameworks [163].

REST (Representational State Transfer) is a widely used architecture style and also popular in the IoT fields because of its low entry barrier and scalability merits. However, the REST architecture style was particularly designed for distributed hypermedia systems, and it sometimes does not fit the IoT requirements. In particular, it is difficult for REST to support complex operations and high-level abstraction, while in the IoT systems, the physical behaviours usually need complex behaviour models which REST cannot provide. Therefore two main issues arise, i.e., system states verification and physical behaviour implementation, which we will discuss in more details in Section 4.3.

To address the above issues, we propose the Feedback-based Adaptive Service-Oriented Paradigm (FASOP) which can be applied at the programming language level to support context adaptation in the IoT systems. Furthermore, the FASOP can be used to add more constraints to use the REST style in the IoT systems to overcome these two limitations, i.e., system states verification and physical behaviour implementation.

The rest of the Chapter is organised as follows. Section 4.2 compares FASOP with some existing works. Section 4.3 explains the motivation of the proposed approach. Then the definition and description of the FASOP is presented in Section 4.4. In Section 4.5, the FASOP is applied in the REST as an extra constraint. Section 4.6 illustrates a simple implementation of the FASOP. In addition, the two issues discussed earlier in Section 4.3 are solved with the FASOP in Section 4.7. The conclusion and future work are given in Section 4.8.

4.2 Related Works

There are many pieces of research on the development of the IoT systems to support context-awareness and adaptation. In [165], a platform is developed as ContextServ to simplify the development of context-aware Web services using high-level modelling language. In [22], a design for adaptation approach is proposed to support the development, deployment and execution of systems in dynamic environments by exploiting service refinement and re-configuration techniques. In [163], the MAPE-K feedback loop is used to support a synchronisation and adaptation mechanism for the real-world process as a process-based framework. It uses a different perspective from combining processes' virtual world with real-world effects to build self-adaptive IoT systems. This work can achieve a high level of autonomy and resilience against failures for physical world process. In [37], the authors provide the methodology of using a model-based service-oriented architecture with service composition to support self-adaptation. The work is solid and also provides fault tolerance mechanism. In [26], the service adaptation is achieved using service composition for automatic reconfiguration based on the rich interface specifications. Following this idea, they used the Discrete-Time Markov Chains in a language to describe the impact of adaptation tactics and the assumption about the environment.

Our approach is a paradigm that can be used in the current service-based technologies, especially for those technologies using REST style. However, because of the special constraints of the REST style, the REST style services are not suitable for context-adaptation in the IoT system. Compared to others' work, we use a different perspective of designing, i.e., FASOP, to support context-adaptation in service development, especially for the service development with REST style which rarely supported the context-adaptation. Furthermore, we prove that this paradigm can overcome the two issues in using REST style services in the IoT systems in Section 4.7.

4.3 Motivation

Using REST architecture style in the IoT systems may cause two problems in the system states verification and physical behaviours implementation. Below, two examples explain them respectively.

4.3.1 Issue of System States Verification

To address the issue of system state verification, below is a scenario in the *Smart Home* to explain how the REST style may cause a wrong system states verification.

The scenario is to turn on/off a lamp in a room. Assume there is a controller for a lamp in the room, and it has two operations *switchOn* and *switchOff*. The typical model and design with the RESTful interface for this scenario can be shown in Figure 4.1. Based on the HTTP standard, if the response status code is *"200 OK"*, the operation successes; and if the response status code is *"5xx"*, the operation fails with the error at the service side.

However, the problem is that even if the response of the status code *"200 OK"* is obtained, the whole operation cannot guarantee to be successful. The returned *"200 OK"* only means the controller has been successfully triggered, but the lamp may still be off for some unknown reasons, for example, due to the network problem, the returned status code cannot reflect the real situation.

These kind of problems can be fixed by other fault tolerance mechanisms in the middleware, however, it makes the solution more complex with extra requirements on techniques and tools. Especially because some methodologies may break the constraints

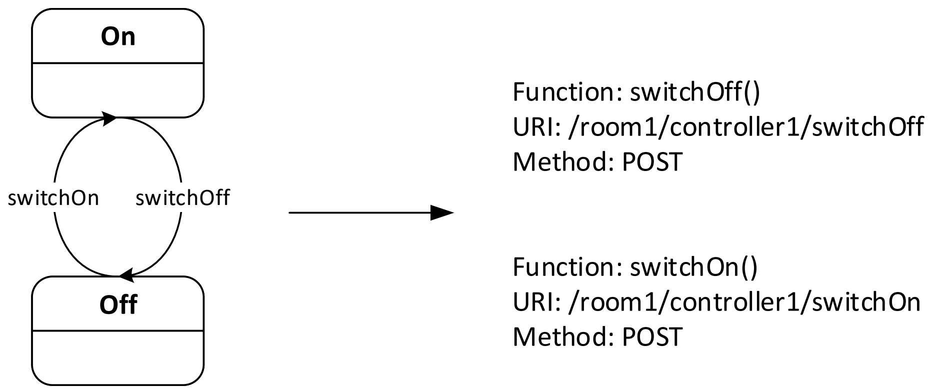


Figure 4.1: The Model and Design with RESTful Interface to Turn On/Off a Lamp

in the REST style, these solutions make it more difficult to model the system states and behaviours.

This is one of our motivations to provide a paradigm with the feedback mechanism for better system states verification in the IoT systems, so the services developed in the IoT systems, especially in REST style, can be more accurate and reliable.

4.3.2 Issue of Physical Behaviours Implementation

The second issue related to physical behaviours implementation is a big problem for developing REST style services in the IoT systems. More specifically, any implementation of continuous physical behaviours with REST style services can be difficult, because the REST style services have limited operations (GET, POST, PUT, DELETE) that cannot fully match the continuously changing physical behaviours. Below, we use a scenario of braking a car to explain the limitation.

Assume we need a braking service, which can brake a car based on current conditions and decisions. This scenario cannot be modelled by a simple state machine. The dynamic physical behaviours of the car can be expressed as follows:

$$\dot{s} = \frac{ds}{dt} = v, \dot{v} = \frac{dv}{dt} = a \tag{4.1}$$

Where s represents the passage within time t with the velocity v, and a is the acceleration.

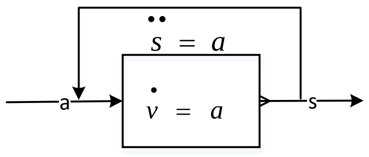


Figure 4.2: The Model for Braking Service

From the REST style services development point of view, we need the GET methods for three variables, s, v and a first, and a POST method to call the *brake service* with parameters a and v and the expected passing distance s.

However, it is impossible to ignore all disturbances and uncertainties in the physical environments, so calling a simple *brake service* with parameters a and v and the expected passing distance s may cause unpredictable effects, that is, the real passing distance s' is far from s. Furthermore, it is very difficult to map the physical braking device to the braking service because quantitatively describing the action is hard. To overcome the limitations of the open-loop controller, the control theory introduces feedback and a closed-loop controller that can use feedback to control states and outputs of a dynamic system. The Figure 4.2 indicates the physical behaviour. However, the traditional REST style services cannot perfectly implement this model.

The paradigm proposed in this Chapter can also be used to enhance the functionalities of the REST to support more complex operations in the physical world in a natural way, because it fits the mathematical form of calculus.

4.4 Feedback-based Adaptive Service-Oriented Paradigm

In the SOA based IoT systems, we distinguish three types of different services, i.e., *Virtual Service*, *Perceptive Service* and *Actuating Service*. The three types are evaluated by the interactive patterns from the service providers to the physical environment.

Virtual Service Most of the traditional software services are virtual services with no interaction with the physical environment. Even in IoT systems like smart home, for example, most of the services are still virtual services which can store temperature data or convert temperature from one unit to another.

Perceptive Service Perceptive services are usually provided by sensors and responsible for detecting the physical environment. The perception that is provided by the perceptive service can be temperature, pressure or vision etc.

Actuating Service Actuating Services are expressed as services executing real actions in the physical environment. For example, in the smart home, turning on/off a light or air condition are actuating services.

An interface I of a service SP, denoted by I_s is defined by a signature and a behavioural model. In the IoT, for any given *Actuating Service*, its interface I_{ac} can be specified by *context*, *signature* and *behavioral model*. *Context* defines information depending on service requesters and service environment. *Signature* corresponds to operation profiles provided by the actuating service. *Behavioral model* is represented by Petri nets to describe the adaptive pattern.

Definition 4.4.1 (Context)**.** We define the context as a typed relation [182]. Let DIM denote the set of all possible dimensions, and U denote the set of all possible tags. A context c is a finite relation $\{(d,x) \mid d \in DIM \ \wedge \ x \in U\}$. The *degree* of the context c is $\mid dom\ c \ \mid$. The empty relation corresponds to *Null* context. The degree of Null context is 0. We formalize context as *a relation*, set of ordered pairs of (d,x) where d is a dimension and x is a tag value.

Definition 4.4.2 (Signature)**.** A Signature is a set of operation profiles. An operation profile is the description of an operation containing the name of an operation, with its argument types and its return type. For the actuating service interface I_{ac}, its signature is defined by a tuple $< O_{as}, O_{ps}, \Gamma >$, where O_{as} is a set of operation profiles provided by the actuating service and O_{ps} is a set of dependent operation profiles provided by other perceptive services. Γ is a set of functions $\Gamma : O_{as} \to O_{ps}$. For any single operation profile $o_{as} \in O_{as}$, it has a set of callable operations from other perceptive services $O_{ps'} \subseteq O_{ps}$ and $O_{ps'} \neq \emptyset$. For the operation profile o_{as}, it has the function $\gamma : o_{as} \to O_{ps'}$, $O_{ps'} \neq \emptyset$, and it is obvious $\gamma \in \Gamma$.

Definition 4.4.3 (Behavioral model)**.** The behaviour in the service can be modelled as a Petri net $SN =< P, T, F, i, o >$, where P and T are disjoint sets of *places* and *transitions*. Places represent states that contain tokens with multiple attributes, and transitions represent activities that can be guarded; transitions are fired when all the tokens in the corresponding input places arrive. Places and transitions are connected through arcs.

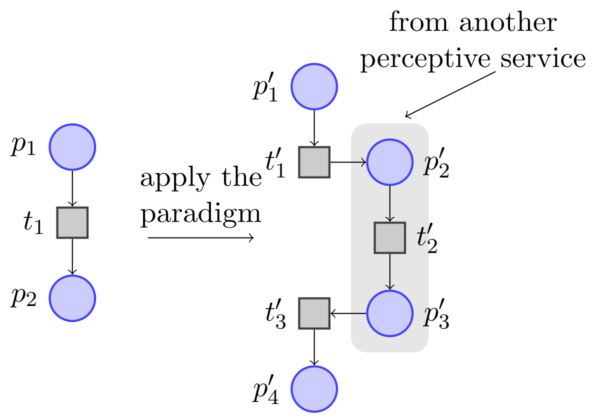


Figure 4.3: The Petri Net Behavioural Model for an Actuating Service

Definition 4.4.4 (Service Interface)**.** A service interface is a tuple $< CP, S, B >$, where: CP is a context profile, and S is a signature with its corresponding behaviour model B.

The behavioural model of FASOP is represented as a Petri net to indicate the atomic operations in *Actuating Services*. As shown in the left side of Figure 4.3, before applying the paradigm, t_1 is a transition provided by the *Actuating Service* and p_1, p_2 are pre-condition and post-conditions of t_1 respectively. From the process point of view, if the operation in t_1 is a function call $< result : func(params...) >$, then p_1 is to map the function name $func$ and parameters $(params...)$, and p_2 is to check the return value $result$. However, if the $< result : func(params...) >$ has any action in physical environment, it is nearly impossible to guarantee all post-conditions from the programming language level, because the post-conditions of t_1 may contain some physical effects that cannot be detected by the *Actuating Service* itself.

In our approach, the proposed paradigm is a mechanism with a feedback mechanism to solve this problem at the programming language level. Feedback control is a central element of control theory, and the importance in self-adaptive systems has already been discussed in [21].

Among the three types of the services in IoT, the feedback loop can be constructed by *Actuating Services* and *Perceptive Services*. A service signature explicitly exposed by a *Actuating Service* is a set of operations that need to declare reachable *Perceptive Services* with specific operations. For a single operation profile, it is expressed as shown in Table 4.1. Then, any service call to $funcAS$ has to pass all required parameters including context

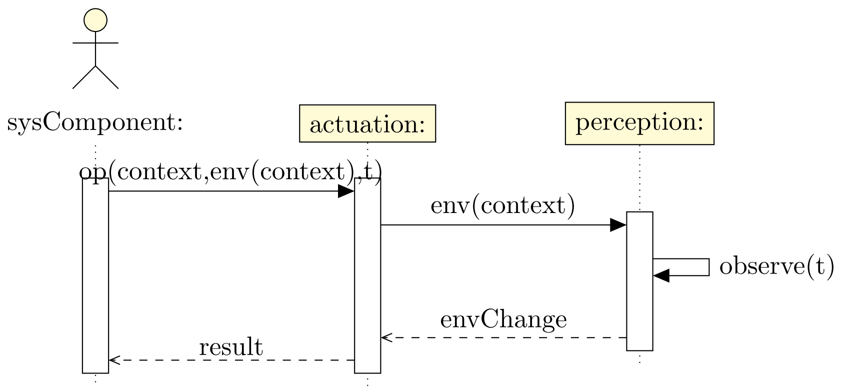


Figure 4.4: The Sequence Diagram for Calling an Actuating Service

Table 4.1: The Single Operation Format in an Actuating Service

Operation Name	$funcAS$
Parameters	$p_1, p_2, ..., p_n$
Available Operations	$PS_1.op1, PS_2.op_1, PS_2.op_2, ..., PS_i.op_j$

information and at least one extra service call as an available *Perceptive Service*. The behavioural model is on the right side of the Figure 4.3. The service call at t_1 is changed from $< result : func(params...) >$ to $< result : func(params..., PS.funcPS, t) >$, which contains another function $funcPS$ from another *Perspective Service PS* and the latency time t which is the waiting time to get the feedback perceptions. In this way, the verification for post-conditions is more reasonable, since the post-conditions with physical properties can be verified through the perceptions of the physical environment by the *Perceptive Services*. With this new paradigm, the *place* p'_2 can verify whether the operation $func$ is operated successfully and *place* p'_4 can eventually check if the operation $func$ has desired behaviours based on the perceptions. In Figure 4.4, a sequence diagram shows the detailed processes from the implementation perspective.

4.5 Extending REST for the IoT based on FASOP

In [52], an example of using REST-based architecture server to control a robot is presented. The author concludes that REST sometimes is inconvenient compared to other RPC style web services because it does not have the functionalities like callback to support complex modelling with the states. The key problem is that keeping all required information in a single request to model physical behaviour while keeping stateless interactions is difficult. In the physical environment, most of the continuously physical behaviours are modelled based on differential equations, so it needs at least two states to express a continuous physical behaviour. Therefore, at least two states in the response are needed to model the physical behaviours in any single request. With the FASOP, any single service call to an *Actuating Service* becomes a transaction, so all required information can be wrapped up to model physical behaviours within one request. This paradigm can be simply converted to an extra constraint for the REST, and the new constraint is expressed as follows:

- Any operation from the *Actuating Services* has to operate in a complete feedback loop containing the perception to the physical environment and the response need to have at least two states of the requested entity.

The high-level model we use for IoT systems is based on [99], which is defined as a tuple $RS = \langle R, I, B, \eta, C, D, \sim, OPS, RETS \rangle$, where R is a set of resources; I is a set of resource identifiers; $B \subseteq I$ is a finite set of root identifiers; $\eta : I \rightarrow R$ is a naming function, mapping identifiers to resources. C is a set of client identifiers, and D is a set of data values, with an equivalence relation $\sim \subseteq (D \times D)$; OPS is a finite set of methods; and $RETS$ is a finite set of return codes. The detailed model is similar to the model provided by [148], where the only difference is for modelling services for actuators.

Resource identifiers are modelled as URIs, and represented as the following scheme:

$$URI = scheme : [//host[: port]][/]path[?query][\#fragment]$$

The descriptions of a service can be obtained by sending a GET to a particular resource via *URI*. Most of the service calls are at protocol level via message delivery, and the main difference is on the *Actuating Service* calling. That is, the request to an *Actuating Service* needs to contain at least one available *Perspective Service* operation.

4.6 Implementation

In this Section, we will use Java web service to express a simple implementation of this paradigm.

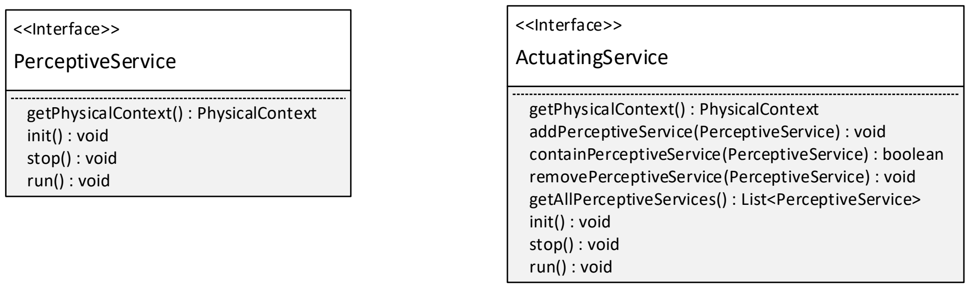

<<Interface>>
PerceptiveService
getPhysicalContext() : PhysicalContext
init() : void
stop() : void
run() : void
<<Interface>>
ActuatingService
getPhysicalContext() : PhysicalContext
addPerceptiveService(PerceptiveService) : void
containPerceptiveService(PerceptiveService) : boolean
removePerceptiveService(PerceptiveService) : void
getAllPerceptiveServices() : List<PerceptiveService>
init() : void
stop() : void
run() : void

Figure 4.5: The Basic Class Diagram of the two Interfaces with FASOP

Based on the former definition in Section 4.4, the services in the IoT systems are concluded as: *Virtual Service*, *Perceptive Service* and *Actuating Service*. Since *Virtual Service* is just normal web service, we only develop two extra interfaces: *PerceptiveService* and *ActuatingService*. For simplicity, we assume all the services can remotely call another service from a different device based on the RPC framework or the Actor model [3]. Figure 4.5 indicates a basic example of these two interfaces.

The main purpose of the interface design is to do type checking in the development. By using annotation in Java, we can restrict the developer to include at least one *PerceptiveService* as a parameter in any *Actuating Service* annotated by @WithFeedback. However, the type checking at this level needs to call a method remotely. That is, if a developer wants to use the JAX_RS (Java API for RESTful Web Services)[71] standard to develop REST style services, the parameters are all String type for the services mapped from the URI, thus the developer cannot do type checking to confirm the *PerceptiveService* as a parameter. In this case, the developer needs to check the *PerceptiveService* in the function of the service.

The implementation is only a lightweight version of the FASOP implementation, because we need to extend the HTTP or CoAP (Constrained Application Protocol) to fully support the FASOP, which is considered as a part of future work though.

4.7 Case Studies

In this Section, we use the two examples presented in Section 4.3 to illustrate the advantages of the FASOP.

```
public class LampService implement ActuatingService{
        ......
        @POST
        @Path("/room1/controller1/switchOn/{perceptiveservice}\{time}")
        public String switchOn(@PathParam("perceptiveservice") String p,@PathParam("time") String time){
                if(!PerceptiveServices.containskey(p)){
                        switchOn();
                        return "No PerceptiveService Found, Switched On";
                } else {        switchOn();
                        Thread.sleep(Integer.parseInt(time)*1000);
                        String state = PerceptiveServices.get(p).getResponse();
                        if(state.equals("On")){
                                return "Switched On, Successfully";
                        } else{   return "Failed"; } } }
        ......
}
```

Figure 4.6: The Implementation Sample of Using FASOP to Turn On a Lamp

4.7.1 Turn on/off a Lamp in the Smart Home

For the scenario in the Smart Home to turn on/off a lamp in a room, the issue is that the response status code cannot express the correct system status. To solve this problem, we use the FASOP to modify the original approach and the changes are as follows:
Function: switchOn() $\rightarrow$ switchOn(PerspectiveService,t)
URI: /room1/controller1/switchOn
$\rightarrow$ /room1/controller1/switchOn/?perceptiveservice=lightsensor&time=1s
Method: POST
The implementation details are expressed in Figure 4.6. In this implementation, the successful status code correctly reflects a guaranteed successful confirmation.

4.7.2 Brake an AutoDriving Car

Compared to the traditional REST style development, the FASOP can help to transfer the physical behaviour model to software development in a more natural way. Below we use the example introduced in Section 4.3 to explain the transfer process.

Based on the Equation 4.1, in a very short time $\Delta t = t' - t$, we have the following form of the equation:

$$\dot{s} = \frac{ds}{dt} = \frac{s' - s}{t' - t} = v, \dot{v} = \frac{dv}{dt} = \frac{v' - v}{t' - t} = a = \frac{s' - s}{(t' - t)^2}$$

With the traditional REST style service development, it is very difficult to evaluate and analyse acceleration quantitatively. However, the FASOP can fit the closed-feedback model, thus the exact acceleration value can be easily evaluated via the distance and time. Furthermore, we can continuously change the acceleration via braking physically, and all effect can be evaluated through the service in real-time. The function of this braking service can be as follows:
Function: braking(PerspectiveService,t)
URI: /car/brake/braking/?perceptiveservice=distancesensor&time=1ms
Method: POST
In any moment, with this braking service, we can also predict the future passing distance s_f during the time period t_f. The prediction can be based on: $s = v * t - \frac{a * t^2}{2}$ if the acceleration remains the same.

4.8 Conclusion

The main reason why there are some issues caused by using the REST style services in the IoT systems, as we described in Section 4.3 is that behaviour abstract ability is missing in the REST style. To solve the problem, FASOP is proposed in this Chapter to provide the context adaptation ability at a low level for service development, therefore the REST style services can implement complex behaviour processes based on the context adaptation.

The implementation in this Chapter is a rather simplified version to use the current web technologies. To fully implement the FASOP in the REST style, we develop a protocol by extending the CoAP (Constrained Application Protocol) based on REST model which will be introduced in Chapter 5.

Chapter 5

Behavioral Abstract Support from Protocol

5.1 Introduction

In recent years, IoT has illustrated a great potential in various fields [64]. When connecting IoT devices to the Internet, the IoT devices and services are expected to inter-operate at the application layer. Inspired by the success of the World Wide Web, Guinard [65] proposed the concept of Web of Things, which advocates using the REST (Representational State Transfer) architectural style to design IoT applications and using the ubiquity of web to interact with devices via HTTP. However, HTTP over TCP has problems in constrained environments, particularly, with the small frame sizes and the lossy links of low-power wireless communication, because it requires more resources to keep the connection. Instead of adding the compression techniques to solve the problems, the IETF designed a new web protocol from scratch: the Constrained Application Protocol (CoAP) [164]. CoAP follows the style of REST, but is tailored to the requirements of low cost devices and IoT application scenarios. It uses a compact binary format and runs over UDP or DTLS (Datagram Transport Layer Security) when security mechanisms are enabled. The protocol also enables multicast communication. At the top level, the request/response model enables RESTful interaction through the well-known methods GET, PUT, POST, and DELETE as well as response codes that are defined in accordance to the HTTP specification. CoAP resources are addressable by URIs, and Internet Media Types are used to represent resource states. RESTful caching and proxying enable network scalability.

However, the application layer protocol is responsible for the application level support, where the RESTful API is not sufficient for all functional requirements of the IoT systems. Thus the resource observe mechanism [73] is designed for many environment monitoring scenes from the sensors as the publish-subscribe model.

Even though the resource observe mechanism can naturally cooperate with the functionalities of the sensors, the CoAP has no support for the actuators from the context-adaptation perspective. However, the Real-time Context-Adaptive support on the unreliable network (like Internet-based network) is critical to implement complex physical behaviours in the IoT systems. In this chapter, we will give some general models for the physical behaviour modelling and related implementations from the Real-time Context-Adaptation perspective.

Different from other solutions, the support from the protocol level is more general than other approaches, especially when the CoAP becomes one of the IoT standard protocols. There is much work based on the CoAP from different perspectives such as the Security [153], cooperation with the Cloud Computing [206], and the applications [103]. However, the CoAP has no mechanism to support the Real-time Context-Adaptation, which limits its capabilities in many complex IoT scenarios.

The rest of the chapter is organised as follows. Section 5.2 express our approach to support the Real-time Context-Adaptation in the CoAP. Specifically, our approach is illustrated in details from six aspects such as the motivation, requirements, context-adaptation messaging model, context option, real-time support and security. Section 5.3 explains the basic implementation based on the existing library and Section 5.4 describes the future work.

5.2 Real-time Context-Adaptation in the CoAP

5.2.1 Motivation

Scalability, extensibility and interoperability among heterogeneous things and their environments, are key requirements and challenges in the IoT systems. Since these requirements are similar to the World Wide Web, which is one of the most successful distributed systems, the Web of Things (WoT) concept is proposed to integrate the smart things into the Web (the application layer) rather than the Internet (the network layer). To achieve the integration, some common patterns used for the Web such as REST Architectural Style are applied in the IoT applications [58]. HTTP (Hypertext Transfer Protocol) [57] is the classical implementation of REST architectural style. However, the devices and the networks in the IoT environments may be extremely constrained, thus the typical approaches based on the HTTP protocol can be too heavy to support the IoT systems. Following the REST architectural style with tailored features for IoT applications and Machine-to-Machine (M2M) scenarios with resource-constrained devices, the IETF Constrained RESTful Environments (CoRE) working group, therefore, designed the CoAP [20].

However, the REST architectural style was mainly designed for the World Wide Web, which satisfies many different requirements compared to the IoT applications. A typical scene is an environment monitoring case when a client is interested in having a current representing of a resource over a period. For example, a temperature sensor is used to detect the abnormal temperature in the industrial environment. Instead of the active roll polling requesting to get the information, it may be more preferable to let the sensor notify the system when the temperature value is out of the normal range. The IETF community, therefore, extends the CoAP with the *Observing Resources* mechanism based on the *Observe Design Pattern* with which the targeted resource can be used to observe the related environment and send the notification if the observations satisfy specific pre-defined policies.

REST architectural style is network-centric [58], which means the basic operations like GET, POST, PUT and DELETE are defined in the protocol level as the unified interface. While this design principle supports the web's scalability and interoperability, it also constrains the request-response model and operations for the different web resources. However, the request-response model with four basic operations is not sufficient for the IoT applications' requirements. Although the extended *Observing Resources* mechanism provides the basic messaging model for most sensors, the actuators' requirements from using the CoAP cannot be satisfied.

If we want to use the actuators to support the continuous physical behaviours via the CoAP, it would be nearly impossible because the CoAP does not support this kind of operations in its unified interface. The CoAP can only be used as a communicating protocol to support some simple operations for the actuators, because no messaging model can precisely control the behaviours on the networks. Primarily, the current CoAP cannot provide complex physical behaviours modelling or implementation, thus largely limits the usage scenarios of the IoT applications.

To enrich the modelling ability and operations in the CoAP, we provide a real-time context-adaptation mechanism, which can use some control theory to model the physical behaviours over the networks.

The Real-time Context-Adaptation support in the protocol has many advantages compared to other solutions, because the protocol is one of the necessary infrastructures for the IoT applications and it can provide the general support without specific middlewares. On the other side, because it provides the Real-time Context-Adaptation support at the general infrastructure level, it is difficult to provide very general behaviour modelling support. We will analyse the requirements in the following Section.

5.2.2 Requirements

The complex physical behaviours modellings and implementations over the unreliable networks place greater demands on providing the Real-time Context-Adaptation mechanism. Essentially the feedback loop in control theory needs to be adapted in the unreliable networks and distributed devices via the application layer protocol. Regardless of many detailed mechanisms, the most significant requirements for the Real-time Context-Adaptation in the CoAP can be summarised as four aspects shown below:

Context-Adaptation Messaging Model The basic messaging models to support context-adaptive behaviours;

Context-Adaptation Option The message format to define the Context-Adaptation Option;

Real-time Support The real-time support and the consistency model for the distributed devices to guarantee the time-sensitive behaviours over the networks;

Security Support The security support is naturally required, because the physical behaviours from the actuators usually affect the safety of the systems.

The critical finding in the [21] has already confirmed that in designing self-adaptive systems, the feedback loops that control self-adaptation must become first-class entities. In essence, we need to correctly apply the feedback mechanism in the CoAP from the control theory perspective to support the real-time context adaptation. The Figure 5.1 uses Petri Net to indicate the basic concept of applying the feedback loop in the basic request-response model. More details about how to manage the devices from the CoAP are expressed in the following Sections. Furthermore, the issues from the unreliable networks and possible solutions are discussed below.

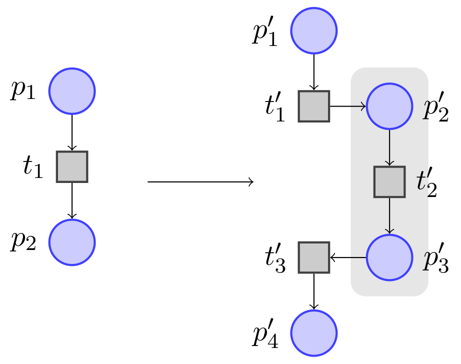


Figure 5.1: To Apply the Feedback Loop in the Basic Request-Response Model for Real-time Context-Adaptation

To keep the advantages of REST architectural style, and the consistency between our extended protocol and the original CoAP, we use all the existing basic messaging models provided in the CoAP and extra mechanisms to extend its functionalities without changing the original protocol and REST constraints. Since the whole Real-time Context Adaptation mechanism in the CoAP is hugely complex, only some basic designs and proposals are introduced with details in the following Sections.

5.2.3 Context-Adaptation Messaging Model

The main purpose of the Real-time Context Adaptation is to support the complex physical behaviours modelling and development on the unreliable networks. Because scalability is one of the key concerns in the IoT applications, based on the conclusion in the [38], the decentralised control solution is preferable. The Context Adaptation Messaging Model is extended from the original messaging models in the CoAP, therefore the feedback control loops would not affect the scalability of the CoAP too much.

As a fully decentralised approach, the management of the decentralised feedback loops is a complicated problem as the contexts and targets can always change. To enable the basic request-response model to support context-adaptive behaviours, the requested targets need to obtain all the required contexts information. Based on the existing messaging models and mechanisms in the CoAP, we propose the following two patterns to locate and retrieve the contexts information.

1. The requested message contains all the addresses of the required context information,

and the requested node will issue another request to the addresses of the context information. The final response will include the requested behaviour and all the changes of the related context information within the given time period.

2. The *Resource Observe* mechanism can make the pre-register between different IoT nodes. If the requested node has already registered itself to several other nodes and the requested message does not contain other addresses, the requested node will send back the requested behaviour and the changes of the pre-registered context information within the given time period.

Apart from the direct request-response process, the CoAP also provides the proxy mechanism to let the proxy forward some messages. Since the proxy allows more flexible and efficient ways to handle the messages [174], we also support the Context-Adaptation Messaging Model with the proxies involved.

Messaging Model without Proxy

Figure 5.2 illustrates the basic Context-Adaptation Messaging Model extended from the direct request-response model. To provide the reliable control, the context adaptive requests are required to transfer with the reliable transmission, which means the message has to be marked as Conformable (CON). When the *actuator* receives the requested context adaptive message, it will issue another request to the specified *sensor*. Since the message is still marked as the context adaptive request, the requested sensor will monitor the environments within the given time period and respond to the requested *actuator* with the context changing information. Then the actuator can combine the data together and send back to the original client.

In Figure 5.3, the context adaptive request does not specify another node's address and the requested *actuator* has already *Observe* another sensor with the observe relationships before this context adaptive request, so the *actuator* will wait for the given time period and eventually respond with the possible changed context information. This form of the Context-Adaptation Messaging Model can hide the details about the required *sensor*, if the IoT applications do not want the client to know the *sensor*. However, it requires the pre-register with *Resource Observe*, which may affect the scalability if the IoT applications contain too many this kind of dependencies.

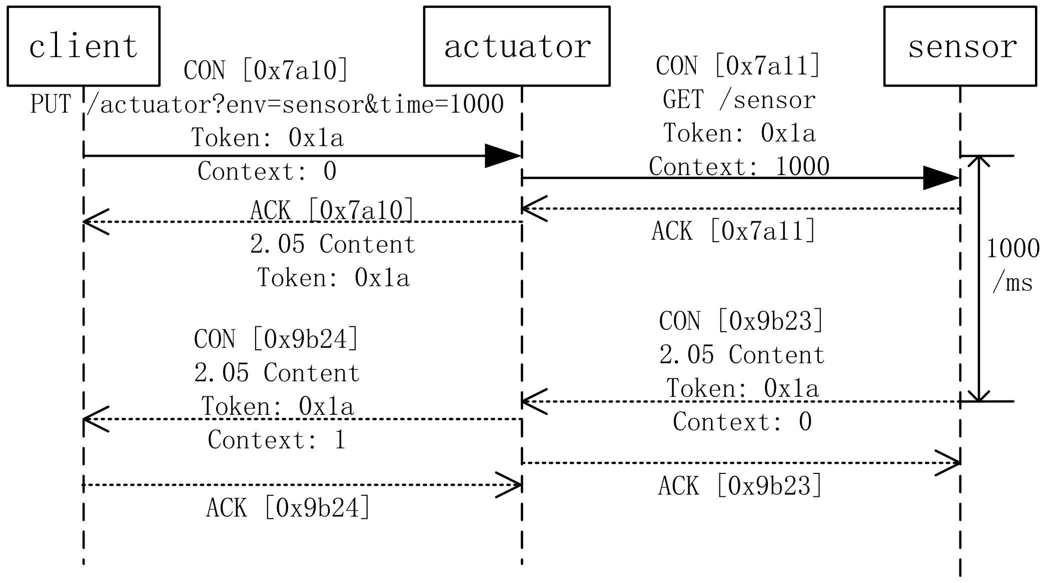


Figure 5.2: The Illustration of the Basic Context-Adaptation Messaging Model without Proxy

Messaging Model with Proxy

The proxy can play an important role in real IoT applications, because the powerful devices can be used to handle large amounts of computation and network pressure as the proxies. Figure 5.4 illustrates an example of requesting a context adaptive message from a *proxy*. Different from the other forwarding messages, if the *Proxy* detects a context adaptive message, it will split the message into two messages and send two different requests to the *actuator* and the *sensor* respectively. The proxy is responsible for managing the Context-Adaptation Messaging Model and carries all the pressures of handling all the coming requests from different nodes. In this case, it is just a normal original CoAP message for the requested *actuator*.

In Figure 5.5, the message sequences are similar as the case in Figure 5.3. The difference is that the *proxy* takes responsibility for managing the Context-Adaptation Messaging Model, so the context adaptive message for the requested actuator is a normal original request. The advantage of using a proxy to support the Context-Adaptation Messaging

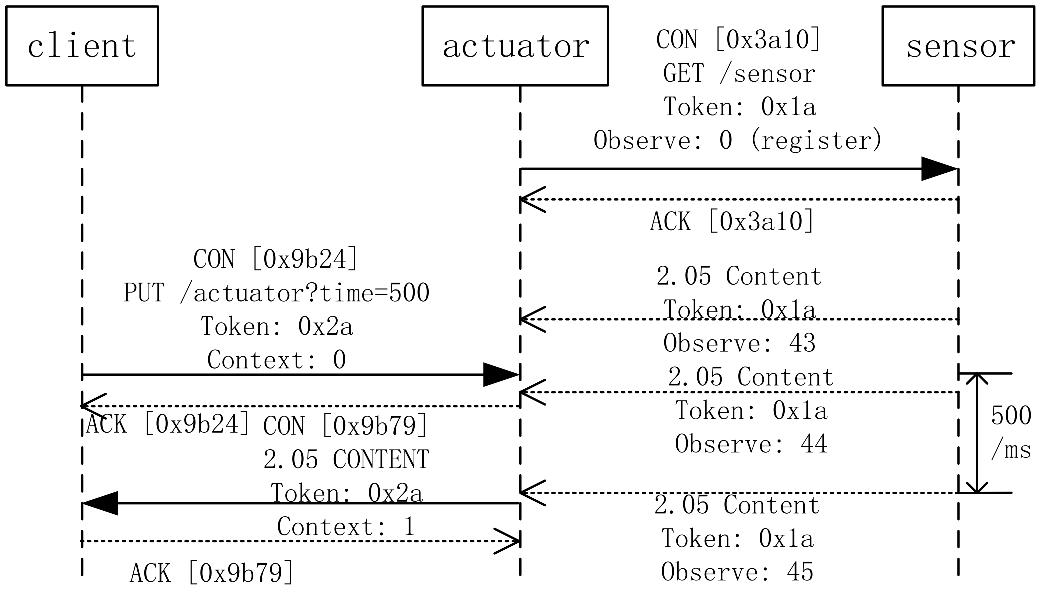


Figure 5.3: The Illustration of the Context-Adaptation Messaging Model in Observing Resources without Proxy

Model with Resource Observe mechanism is that the proxies are able to manage complex dependencies as a centralised node to reduce the pressures for other resource-constrained nodes.

5.2.4 The Adaptation Option

Similar as the Observe Option for Resource Observe mechanism, the Adaptation Option's format is defined in Table 5.1 where C stands for Critical, U stands for Unsafe, N stands for No-Cache-Key and R stands for Repeatable. If it is included in a PUT request or in a response message, the requested node knows it is a context adaptive message.

Table 5.1: The Adaptation Option

No.	C	U	N	R	Name	Format	Length	Default
0		x	-		Adaptation	uint	0-8 B	(none)

When the Adaptation Option is included in a PUT request with the value 0, it extends

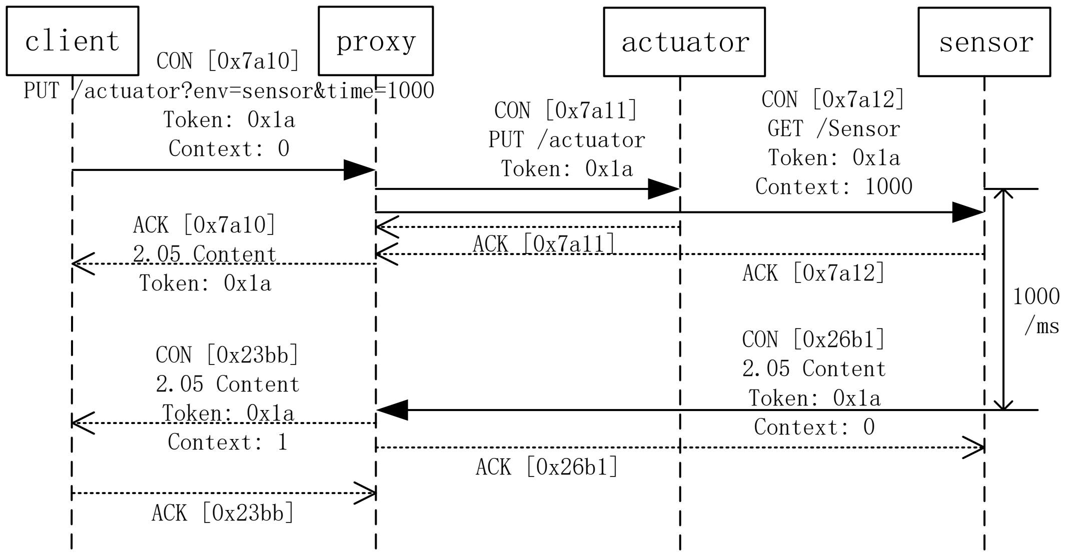


Figure 5.4: The Illustration of the Basic Context-Adaptation Messaging Model with Proxy

the PUT method so it will parse the requested URI to detect if there is any required node's address. Then the requested node will check if itself is observing any valid resource. The requested node handles the context adaptive message following the Context-Adaptation Messaging Models described in Section 5.2.3. When the Adaptation Option is included in a GET request with an option value which is not 0 or 1, it extends the GET method, so it not only retrieves a current representation of the target resource, but also records the representation of the target resource during the given time period based on the option value.

The option value 1 is the exceptional value which is used to confirm the end of a context adaptive request. If the response contains an Adaptation Option with the option value 1, the receiving node knows that the response is from a context adaptive request and it can do the further analysis and decision based on the response.

The Observe Option is not critical for processing the request. If the server is unwilling or unable to support context adaptation, then the request falls back to a normal PUT request, and the response does not include the Adaptation Option.

The Adaptation Option is not part of the Cache-Key: a cacheable response obtained

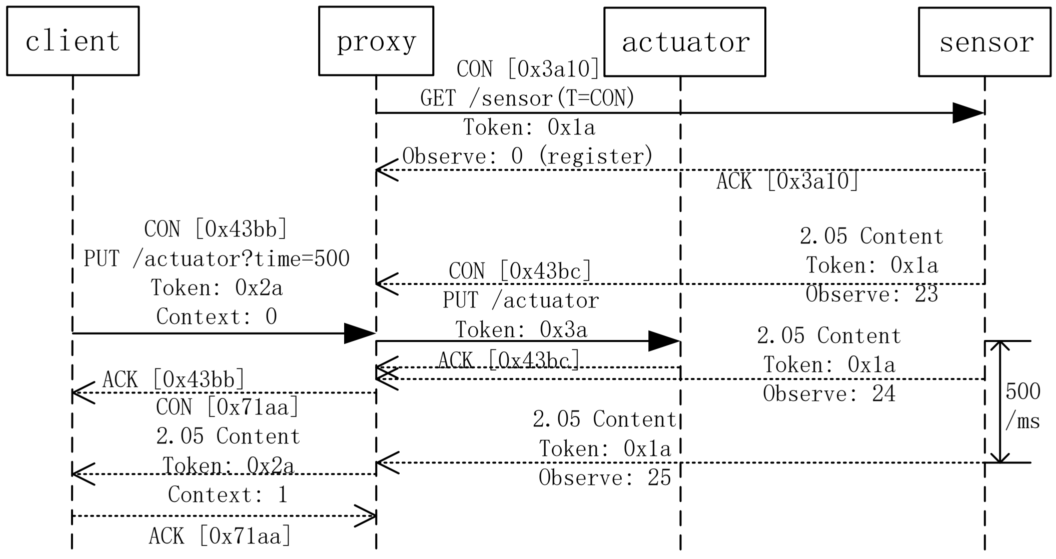


Figure 5.5: The Illustration of the Context-Adaptation Messaging Model in Observing Resources with Proxy

with an Adaptation Option in the request can be used to satisfy a request without an Adaptation Option, and vice versa. When a stored response with an Adaptation Option is used to satisfy a normal GET/PUT request, the option must be removed before the response is returned.

To track the entire process of one Context Adaptation Messaging, the *Token* is used to confirm if the transferred messages are contained in one context adaptive request.

5.2.5 Real-time Support

The former context-adaptive mechanism is already sufficient for some simple discrete behaviours such as to turn on/off a controller for a lamp. However, for some complex behaviours, we need the precise time to make accurate control via solving the related physical models like differential equations.

The distributed control has been researched for decades, and different solutions are proposed. Different from other distributed control systems, the devices and networks in the IoT systems are usually unreliable. As an application layer protocol, the protocol needs to overcome the uncertainty in the devices and networks to implement the physical behaviour models and estimate the states, because the protocol cannot control the reliability on the communication layer. To extend the protocol, we use the Quality-of-Service (QoS) at the application layer to set the contracts with other protocols and applications.

To support real-time feature with QoS, there are two main concerns in our approach: network latency and time consensus.

The network latency is an unavoidable effect if the IoT systems are running on the Internet-based communication protocols. For most of the current IoT systems, the network latency only affects the system performance, however, the network latency will affect the system correctness if the real-time physical behaviours require being implemented because the precise and reliable time for the physical models is needed [113]. To solve this problem, we can firstly consider a case with the basic Context-Adaptation Messaging Model shown in the Figure 5.6.

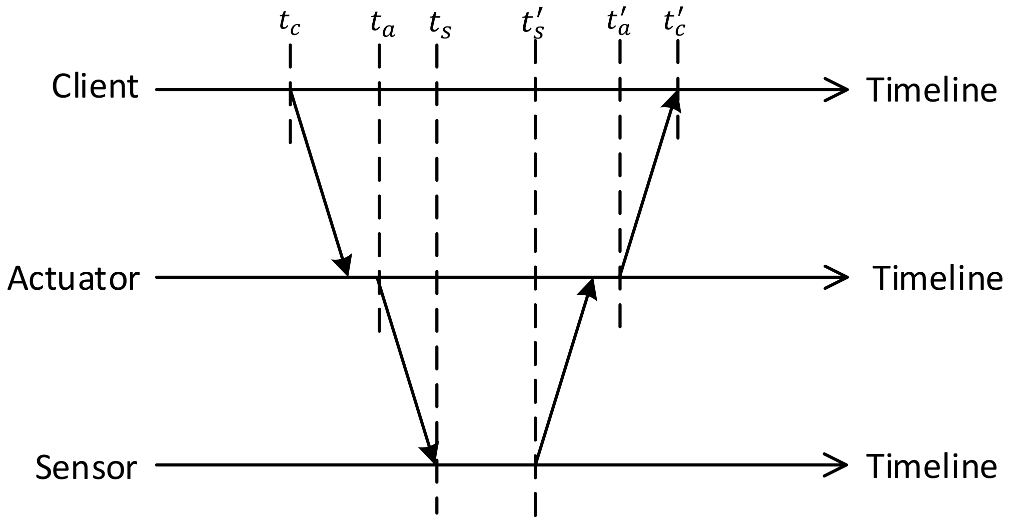


Figure 5.6: The Time-Aware Messaging Sequence of the Basic Context-Adaptation Messaging Model

In Figure 5.6, the t_c is the start time of the request from the *Client*. The t_a is the time when the *Actuator* starts the requested operation and the t_s is the time that the *Sensor* starts to detect the environment. In the *Client*'s own time-line, we can only know the precise time of t_c and t'_c, however, the *Actuator* and *Sensor* can record their time stamps in the response messages, therefore the requesting *Client* can know the precise time from the *Actuator*'s and *Sensor*'s time-lines.

However, this mechanism will bring a new problem, that is, the different nodes' timers are different. In Figure 5.6, the time t_a at the *Actuator* cannot be guaranteed as the same time recorded in the *Client* node. To solve this problem, the timers in all the requests nodes should be as same as possible, and this requirement needs another solution, i.e., the time synchronisation mechanism.

In the CoAP, we use the QoS to satisfy the Real-time feature from three QoS properties with network latency and time consensus. Table 5.2 indicates the three main QoS requirements for the Real-time support for the Context Adaptation in the CoAP. These QoS policies are at the application layer, and they still need the cooperations with other communication protocols and time synchronisation protocols which are not mentioned in this thesis.

5.2.6 Security Support

Currently, the CoAP uses the Datagram Transport Layer Security (DTLS) as the underlying security protocol for authenticated and confidential communication. However, the Real-time Context Adaptation support in the CoAP may cause new problems. The main security concerns in the approach are about the access control and Denial-of-Service (DoS) attack.

Because the adaptive resource can forward the requests to other resources, the clients or the adaptive resources may be able to access some resources' information that they cannot access directly. This possibility risks the entire systems' privacy and security, which need more security mechanisms, especially in the access control aspect.

Another concern is about the DoS attack, because all the nodes supporting Real-time Context Adaptation are time sensitive, and the Dos attack can significantly affect their system behaviour correctness rather than performance. For example, if a node under the DoS becomes slow, it may never reach the QoS requirements to provide the Real-time Context Adaptation feature. For this concern, we recommend to use the Context Adaptation Messaging Model with Proxy, therefore, the network pressure can transfer to the proxy side where more mature solutions in the gateways for the DoS attacks exist.

5.3 Implementation

The implementation is mainly for the Context Adaptation Messaging Model part only because this part of work is still at the early stage to support the Real-time Context Adaptation in the CoAP yet, and the QoS policies for the Real-time support require lots of the implementation on the communication layer protocols and other application layer protocols which are not existing yet.

The implementation is based on the open-source CoAP Java library from the Arm Mbed [126]. Similar as the *Resource Observe* mechanism, an abstract class *AbstractAdaptableResource* is extended from the *CoapResource* to specify some basic requirements of the CoAP resources that can support Context Adaptation Mode.

Compared to the *Resource Observe* mechanism, we have more different adaptation modes, meanwhile, the requests' states need to be maintained during the environment detecting time. The management can be extremely complex, however, we only provide a basic implementation where the Adaptive Resource uses two HashMaps to manage the external observable resources and process environment detecting resources (like sensors).

In the *LinkFormat* class, a static property of "LINK_ADAPTABLE" with value "ada" is added to specify the Adaptive Resources. For an Adaptable Resource, the general steps of processing a context-adaptive message are as follows:

1. Check if the request is a context-adaptation message.

2. If yes, check if the request contains the target resource to detect the contexts. If yes, the Adaptive Resource does the requested operations first and then forwards the rest of the message to the target resource.

3. Store the request and the target resource in a HashMaps to wait for the response from the target resource.

4. If the request does not contain the target resource to detect the contexts, check if the Adaptive Resource is observing any other CoAP resources. If no, it gives an exception message back to the client; if yes, it sets up a timer to wait for the certain time included in the request.

5. Eventually, the Adaptive Resource combines its own operation status and the detected environment together, and sends it back to the client.

During this procedure, the Adaptive Resource has one HashMap to check all the observing resources, and another hash map to manage the states of the context-adaptive request.

However, the implementation here does not consider the different permissions for all the requesting, and the strict time constraints issues. The proxy is also not supported in the implementation. We consider to fully implement them in the future.

5.4 Conclusion

This chapter has described how the Real-time Context Adaptation can be applied into the CoAP to functionally enhance the physical modelling abilities in the IoT systems from the protocol level. Compared to other context adaptation solutions, the Real-time Context Adaptation in the CoAP is a more general infrastructure level support, especially when the CoAP has already been one of the standards for the IoT systems.

However, to provide precise Real-time Context Adaptation in the CoAP is difficult because it is usually based on the unreliable networks (UDP) and devices. In this chapter, we only give the four types of Messaging Models to support Context Adaptation while the QoS for the real-time feature still needs the cooperation of other communication protocols and time synchronisation protocols. The security issues for this approach only have some general discussions without detailed solutions. The implementation is only for the basic Context Adaptation Messaging Model, and it is based on the open-source CoAP Java library from the Arm Mbed.

Future work will consist of completing the QoS support, adding security mechanisms, and implementing the full protocol. In the future, we plan to use formal methods to verify the proposed protocol extension. We also plan to use the extended protocol in some real applications and test the performance.

Table 5.2: The required QoS for Real-time

QoS policy	Entity	Description
Deadline(t)	DR	Max expected elapsed time between arriving data samples.
	DW	Max committed time to publish samples or instances.
Latency Budget(t)	T, DR, DR	Indication on how to handle data that requires low latency. Provides a maximum acceptable delay from the time the data is written to the time it is received by the subscribing applications.
Time Consensus (%)	TC	The time synchronization level for different nodes. Provide a rate to describe the minimum required time synchronization level for this node.

Chapter 6

Uncertainty Handling

6.1 Introduction

Internet of Things (IoT) is envisioned to integrate the physical world into computer-based systems. Recently, with the advanced technology development on sensors, networking, data processing etc., IoT has demonstrated a great potential in various fields [64]. However, even after decades of research on system aspects of IoT, developing IoT based systems is still facing many challenges like scalability, inter-operability and fault tolerance [132]. Because of the complex interaction with the physical environment, the IoT systems face more uncertainty than the traditional software systems and demand different fault tolerance mechanisms.

Uncertainty is a classical and critical problem being well studied in many complex systems from different perspectives. For the modern hybrid systems interacting with the physical environments, the hardware components can be easily affordable yet significantly affected by the physical environments, thus the systems are more unreliable due to the inaccurate detections, actions or possible damages. The uncertainty from the unreliable hardware components and unpredictable behaviours may confound the results or cause a severe loss if the components are critical.

The uncertainty in the hybrid systems causes various problems. Based on the systematic classification in [201], the uncertainty of the hybrid systems can be specified by three levels: *Application*, *Infrastructure* and *Integration*. Therefore a conceptual uncertainty model is defined as a general reference model for a different kind of uncertainty. If the uncertainty is classified by analysing the causes in a more detailed way, the uncertainty can come from different levels such as hardware fault [160], channel uncertainty in communication [106], software design [15], software development [87], and hostile attacks [204]. The solutions for a different kind of uncertainty usually only work in a specific situation using a particular model.

Compared to other specifically designed models for uncertainty, the reputation-based approach is a more general solution for different cases with intrinsic self-adaptation. Since a reputation-based framework for Wireless Sensor Network (WSN) is designed in [60], this kind of approach has been widely used in the WSN applications with different technologies. One rationale in the reputation-based approaches in WSN is that the sensors usually have the same functionality in the WSN applications. Therefore, they can use distance-based outlier detection [100] or density-based outlier detection [141] to evaluate other sensors' reputations. However, in the highly heterogeneous IoT systems, most of the devices have different functionalities making it difficult to evaluate each other's reputation from different devices and system components.

One possible solution to deal with the heterogeneity is to integrate social networking concept into IoT, which leads to the investigation on the Social Internet of Things (SIoT) [136]. However, defining all the relationships and behaviours among the different nodes in IoT systems from social networking perspective remains to be one of the major challenges in SIoT. To use social networking concept for uncertainty in IoT systems, extra networks with many redefined relationships and behaviours are required to meet the requirements, which is still a challenge.

Instead of using social networking concept in the IoT systems, semantics is used in our framework to extend the classical reputation-based framework for WSN applications. Via extracting functionalities' semantics, we can create some conceptual "resources" and construct a directed graph to make the system components providing the same functionality point to the same "resource" node because they are semantically equivalent. Since the system components that point to the same "resource" have the same type of functionality, they have the ability to evaluate each others' reputations. The whole reputation updating mechanism is based on three assumptions which will be introduced in this Chapter with details. The reputation-based approach proposed in this Chapter is a general and distributed framework that can be seamlessly integrated to any IoT systems. We can use some basic concepts and technologies that are currently widely used in the web to construct the framework without requiring too much prior knowledge, specific models or additional supporting technologies.

This Chapter is organised as follows; in Section 6.2, we present some related works on data fusion and fault detection with different approaches, especially reputation-based methods in IoT or WSN. Our reputation-based framework is described in Section 6.3 which is divided into four parts. Section 6.4 introduces how this framework is implemented via RESTful web service technology. The evaluation results are expressed and discussed in Section 6.5. Finally, Section 6.6 concludes the Chapter and discusses the future work.

6.2 Related Works

The issues of uncertainty in the IoT systems come from many different aspects such as physical randomness, noise, software faults or attack. The research fields and technologies in handling uncertainty in the hybrid systems mainly include data fusion, fault diagnosis, security, etc. In this Chapter, the proposed approach focuses on the data integrity and accuracy with data fusion, and adaptive fault diagnosis.

Ever since the Multi-sensor data fusion was developed, it benefits many different fields like military, manufacturing and robotics [72]. In general, the approaches for data fusion can be classified into two types, model-driven and data-driven. The model-driven approach needs prior knowledge of the expected data model and can be more domain specific, while the data-driven approach has a more general principle. Since a general reputation framework is proposed in [60] for high integrity sensor networks with peer-to-peer rating reputation based on sensor nodes' behaviours via the watchdog components, the reputation-based approach becomes popular for sensor networks. Many different algorithms like probability theory, fuzzy set theory, and DempsterShafer evidence theory are used in the multi-sensor data fusion [94].

However, it is easier to deploy reputation system in WSN because most nodes have the similar functionality thus they can easily provide enough redundancy sensors. The resources can choose the neighbour nodes to evaluate their reputation based on their transactional behaviours [55][39]. In some highly heterogeneous systems like IoT, the devices all have different functionalities. It is therefore difficult to require all the nodes to have enough interactions with their neighbour nodes. Furthermore, besides sensors, there are many other types of system components in IoT. This reveals difficulty to integrate those components into the old reputation-based framework provided for the WSN. Some researchers use the social networking concept in the IoT systems to build the more complex networks with the social relationship to construct the reputation framework [16][17][136]. It is a very cutting-edge research field and has many challenges such as scalability, lookup, communication protocols and social networking management [124]. Because of the different foundations of social networks and IoT systems, many concepts have to be redefined from a new perspective such as “Honesty” in the SIoT [86] to construct the reputation framework. In [177], the trust concept, definition, and model are rebuilt in the context of the SIoT to support the trust evaluation in the SIoT environment.

Different from the works mentioned above, the Semantic-based Reputation Framework (SRF) proposed in this Chapter uses many concepts from the web for general IoT systems. Semantics is becoming an important concept in dealing with many IoT challenges. For example, in the [42], the semantics is used in the IoT framework to support RESTful devices' API interoperability. In this Chapter, the framework can match different system components with high-level semantics to construct the reputation frameworks to reduce the runtime uncertainty self-adaptively and catch some general fault without classifying all different fault types. Therefore, the SRF cannot only deal with highly heterogeneous IoT systems, but also provides the self-adaptation feature for large-scale IoT applications. Furthermore, the development and implementation of this framework can reuse many existing web technologies thus it is easily compatible with most of the current technologies and deployed in the existing IoT systems.

6.3 Semantic-based Reputation Framework

The entire framework can be divided into four parts: *Semantic Match*, *Data Fusion*, *Belief Updating* and *Fault Detection*. In particular for device noise, the *Data Fusion* is used to support more reliable and accurate values produced from the devices, and for unpredictable fault, the *Fault Detection* can generally detect the unknown fault. The detailed uncertainty models are discussed later in the respective Sections. The *Semantic Match* is to construct the framework's structures for the heterogeneity in the IoT systems, and the *Belief Updating* provides the mechanism for evaluating the reputation values. Before the detailed framework structures and algorithms are presented, several important concepts are defined as follows:

Definition 6.3.1 (Uncertainty)**.** For the uncertainty in the IoT systems, the Semantic-based Reputation framework is mainly designed for two types of uncertainty - device noise and unpredictable fault. The typical devices used in the IoT, like sensors, always have the deviation of measurements. Based on the *central limit theorem* [155], even though the

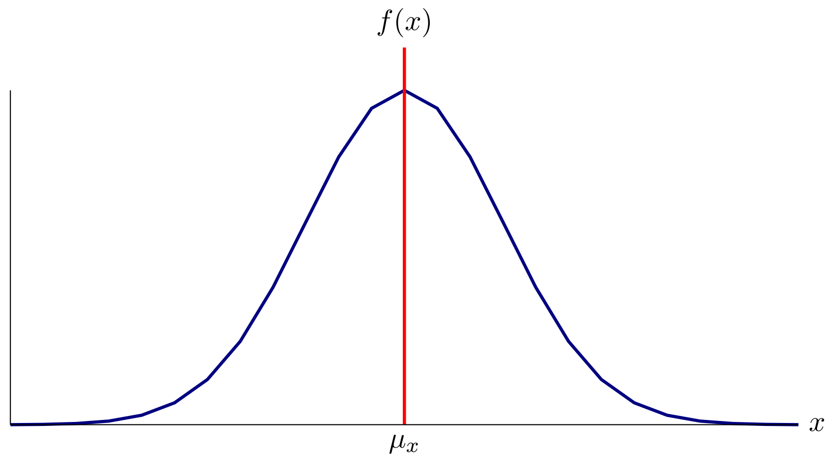


Figure 6.1: The Uncertainty of the Devices fitting Gaussian Distribution

individual constituent deviations may not be Gaussian distributed, the combined deviation is approximately so, therefore we assume the uncertainty from the device noise follows the Gaussian Distribution, which is expressed in the Figure 6.1. For the unpredictable fault, it is defined as the general fault caused from any possible reasons. For example, the device may be damaged by animals, and starts to give wrong values. The general faults in the IoT can be detected by the *Fault Detection* part in the Semantic-based Reputation framework.

Definition 6.3.2 (Component)**.** The component is defined as the system component, containing both the software and hardware parts. The most common example component in the IoT environment is the sensor with the device and related software component.

Definition 6.3.3 (Resource)**.** The concept *resource* is a conceptual mapping to a set of entities, which was originally proposed in the REST (Representational State Transfer) architectural style to build the modern web [58]. The definition of the *resource* is narrowed down in the reputation framework, where the *resource* can be treated as the functional description of the components from high-level semantics. If the system has one temperature sensor at location $l_{(x,y)}$ to detect temperature T and the function can be written as $T_{l_{(x,y)}}(t)$ where the t is the *time* variable, the function $T_{l_{(x,y)}}(t)$ can be a resource mapping to "the real-time temperature at location $l_{(x,y)}$" with a unique URI (Uniform Resource Identifier) to name and address this resource. Any resource can be an abstract concept or function mapping to a type of data that can be detected or evaluated by the components in the IoT systems.

Definition 6.3.4 (Belief)**.** Belief is the value to describe the reputation of the component. The value of the belief is given by the mapped *resource* to express how much the component can be trusted.

Figure 6.2 illustrates a big picture of the runtime framework in which many different system components are matched to different resource nodes with similar semantic functionalities. The blue nodes are *resource*s expressing the functional semantics matched from the components. The arrows from different devices to the resources indicate the semantic match relationships. The resource nodes will fuse the data from different components and give the evaluated belief properties to all the matching components. The system will detect the fault if any components' belief property is lower than the given thresholds. In the figure, the green nodes are in the normal running mode, while the red nodes are in the detected fault mode.

6.3.1 Semantic Match

Reputation-based approaches have been widely used in many WSN applications to deal with uncertainty. However, it is very challenging to apply reputation concept in the IoT systems because most system components are heterogeneous and multi-functional, therefore it is difficult to require any system component to evaluate other system components' behaviours when they have entirely different functionalities.

However, even in the most heterogeneous systems, some system components still have the same or similar functionalities. If we can construct a special structure to check the data and behaviours between these components, it is possible to have distributed reputation evaluating processes via consensus-checking between different system components which are semantically equivalent. To achieve this goal, in this framework, we use a commonly utilised abstract concept in the web - resource, to build the consistency-based structure for evaluating the reputations, and the process of building the structure is named as *Semantic Match*.

After a conceptual resource is created, several different system components can be bound to this resource with some specific functionalities matching to the resource. A classical example to explain this concept is the hybrid localisation solution [197] in which

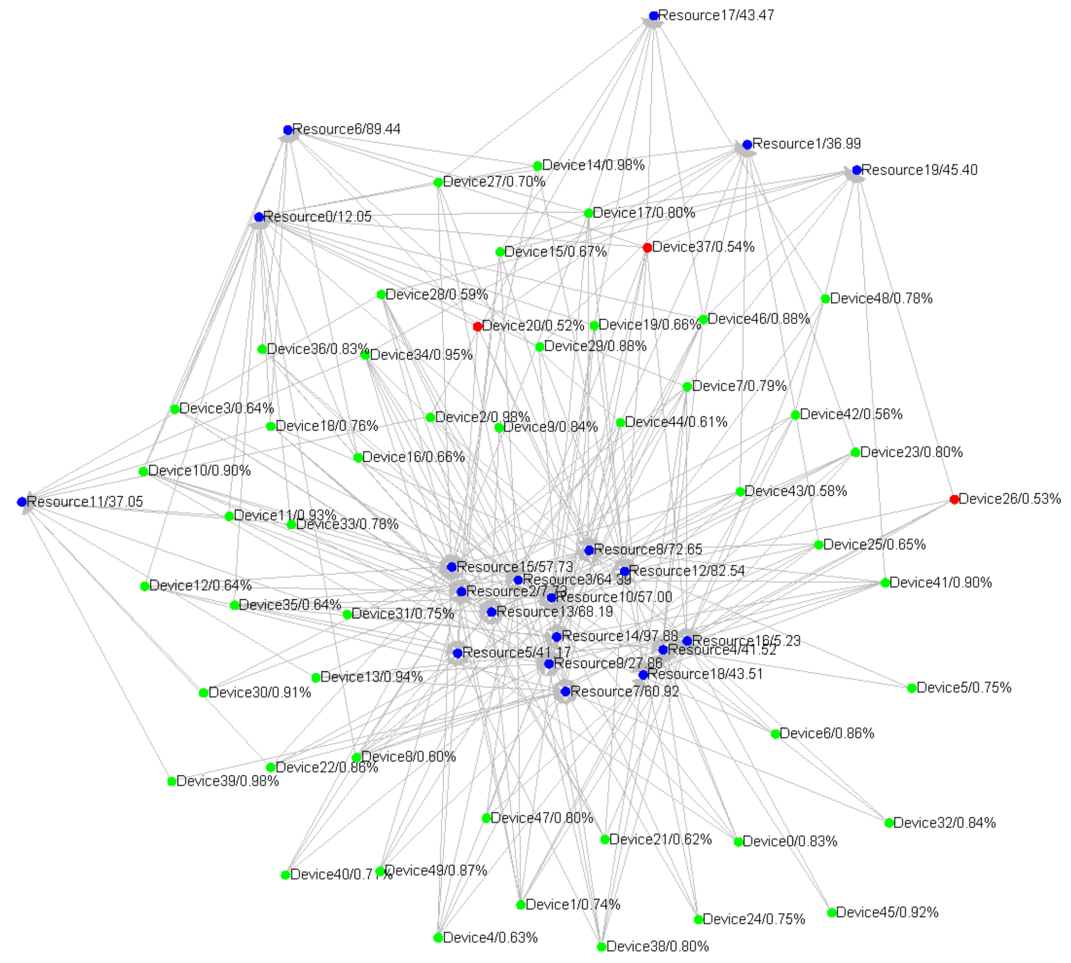


Figure 6.2: The Visualised Semantic-based Reputation Framework Concept

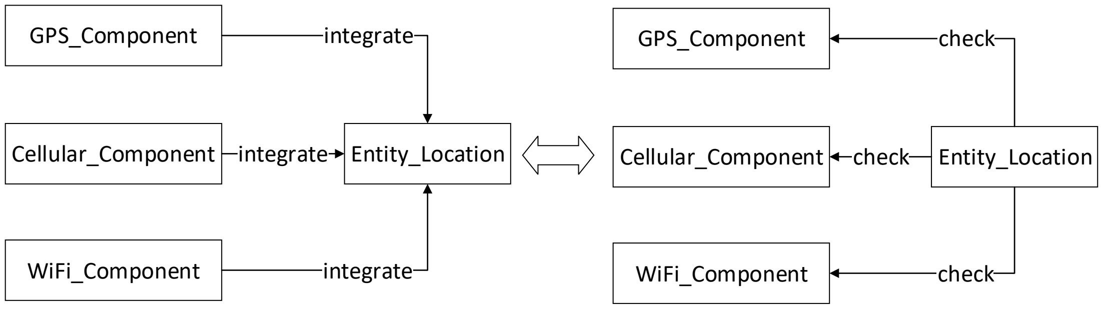


Figure 6.3: The Hybrid Localization Solution as a Semantic Match Example

the GPS positioning component, WiFi positioning component and Cellular positioning component all map to a concept of the entity location as three semantically equivalent entities. Thus we can create a resource $e_{(x,y)}$ to express the location of entity e and bind the three components to this resource. This solution cannot only be used to make the localisation more accurate and reliable but also make it possible to detect any abnormal fault from these three components, illustrated in Figure 6.3. Because in the long-run, if any component suddenly starts to give unbelievable drift values compared with the other two components, it is possible that the component may have deviated from the normal running status.

A complex IoT application may contain many different components in addition to a few localisation modules, thus the system can construct a directed graph containing the system component nodes, the conceptual resource nodes, and the arrows pointing from the system component nodes to the conceptual resource nodes. As an example, Figure 6.4 illustrates a directed graph with six individual components and four resources.

To give a formal description, any arrow a_{ij} from c_i to r_j indicates a functionality of the component c_i as a sequence of data with variable time t

$$f_{a_{ij}}(t) = (\vec{d}_{c_i,r_j,t_1}, \vec{d}_{c_i,r_j,t_2}, ..., \vec{d}_{c_i,r_j,t_k}) \tag{6.1}$$

Where any $\vec{d}_{c_i,r_j,t_p} =< v_1, v_2, ..., v_m >$ is a vector produced from component c_i at time t_p. Any resource r_μ has a unique URI_{r_μ}. At time t_p, it receives α number of data from α number of system components and the received data in resource r_μ is expressed as a set of

$\vec{D}_{r,t_p}$:

$$\vec{D}_{r_\mu,t_p} = \{\vec{d}_{c1,r_\mu,t_p}, \vec{d}_{c2,r_\mu,t_p}, ..., \vec{d}_{c\alpha,r_\mu,t_p}\} \tag{6.2}$$

Generally, any system component c_i can have several different functionalities to be mapped to different conceptual resources. Each functionality can produce a vector of data $\vec{d}_{c_i,r,t_p}$ at time t_p and the data will be sent to the resource r. Furthermore, each system component c_i has a special property b_{c_i} as the *Belief* of the component c_i assigned from the matching resources. Because every system component has its mapping resources, the mapping resources can keep the *Belief* property updated. Essentially, the process of the *Semantic Match* is to add some extra resource nodes in the resource registry and construct the specific structures with algorithms to check consistency between different components with the same functionality. After the *Semantic Match* structure is built, the resources can fuse data from different components to produce more accurate results and use the results to update all matched components' *Belief* properties to evaluate their reputations.

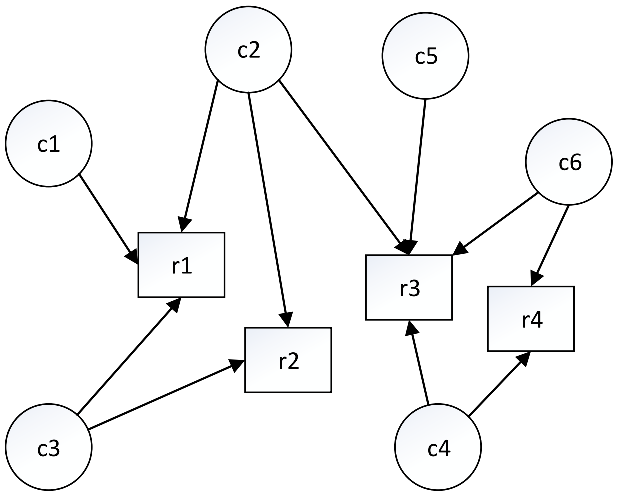


Figure 6.4: The Semantic Match to construct a Directed Graph

6.3.2 Data Fusion

Because one resource has different sources of data from different components, it is possible to integrate different data to achieve more accurate and reliable data. This process is called

Data Fusion. The targeted uncertainty is from the device noise, and it is assumed that the accuracy of the devices follows the Gaussian Distribution. Based on the redundant data from the different components, the effects of the device noise can be reduced by applying the algorithms. There are many different approaches for the data fusion, especially when there is real-time updated *Belief* property for each source. In this Chapter, for simplicity, we only use a linear model based on Fuzzy Set [196] to explain how the data fusion works in our framework.

The belief b_{c_i} of any component c_i is between 0 and 1, $0 < b_{c_i} < 1$, that $b_{c_i} = 1$ means the system believes the component c_i is 100% correct and $b_{c_i} = 0$ means the system believes the component is 100% wrong.

A component has its belief property b_{c_i} and the output from this component to a resource r at time t_p is a vector $\vec{d}_{c_i,r,t_p} =< v_i, v_2, ..., v_m >$. Due to the device noise uncertainty, we assume the real data is $\vec{d}_{c_i,r,t_p,real}$ and $b_{c_i} \cdot \vec{d}_{c_i,r,t_p} \leq \vec{d}_{c_i,r,t_p,real} \leq (2 - b_{c_i}) \cdot \vec{d}_{c_i,r,t_p}$, thus:

$$\vec{d}_{c_i,r,t_p,real} =< v_{1,real}, v_{2,real}, ..., v_{\varrho,real}, ..., v_{m,real} > \tag{6.3}$$

where $1 \leq \varrho \leq m$ and $\forall v_{\varrho,real} \in [b_{c_i} \cdot v_\varrho, (2 - b_{c_i}) \cdot v_\varrho]$. We use:

$$\vec{d}_{c_i,r,t_p,real} = \zeta(b_{c_i}, \vec{d}_{c_i,r,t_p}) \tag{6.4}$$

to express the real data at time t_p from the component c_i to the resource r.

Any resource r is mapped from a set of components C_r, where any component $c_i \in C_r$ produces the vector data $\vec{d}_{c_i,r,t_p}$ at time t_p. Let's assume the resource r successfully received correct time-stamped data from all the source components in C_r and the size of the components is $\kappa = |C_r|$. The received data in the resource r can be denoted as a set of vector data $\vec{D}_{r,t_p}$:

$$\vec{D}_{r,t_p} = \{\vec{d}_{c1,r,t_p}, \vec{d}_{c2,r,t_p}, ..., \vec{d}_{c_\kappa,r,t_p}\} \tag{6.5}$$

and the respective belief values from all the matched components can be expressed as a vector $b_{r,t_p} =< b_{c1,t_p}, b_{c2,t_p}, ..., b_{c_\kappa,t_p} >$. Then, based on the definition 6.4, the real value should be expressed as:

$$\vec{D}_{r_{real},t_p} = \{\zeta(b_{c1,t_p}, \vec{d}_{c1,r,t_p}), \zeta(b_{c2,t_p}, \vec{d}_{c2,r,t_p}), ..., \zeta(b_{c_\kappa,t_p}, \vec{d}_{c_\kappa,r,t_p})\} \tag{6.6}$$

Since we eventually need a single scalar quantity output for further services and to update the belief properties, the defuzzification is required to produce the expected output. There are many different methods to do the defuzzification such as max membership principle [156], centroid method [111], weighted average method [44], or mean max membership [169]. In this thesis, we assume the uncertainty of the devices fitting Gaussian Distribution, therefore, the output membership function of $\vec{d}_{c_i,r,t_p,real} = \zeta(b_{c_i}, \vec{d}_{c_i,r,t_p})$ is symmetrical. It satisfies the condition of the weighted average method, which is more computationally efficient, so we choose it to produce the results. The general algebraic expression of the weighted average method is defined in [156] as follows:

$$z^* = \frac{\sum \mu_{\tilde{c}}(\bar{z}) \cdot \bar{z}}{\sum \mu_{\tilde{c}}(\bar{z})} \tag{6.7}$$

Where $\sum$ denotes the algebraic sum and $\bar{z}$ is the centroid of each symmetric membership function. In our framework, the expected real output value can be expressed as follows:

$$d^*_{t_p} = \frac{\sum_{i=1}^{n} \vec{d}_{c_i,r,t_p} \cdot b_{c_i}}{\sum_{i=1}^{n} b_{c_i}} \tag{6.8}$$

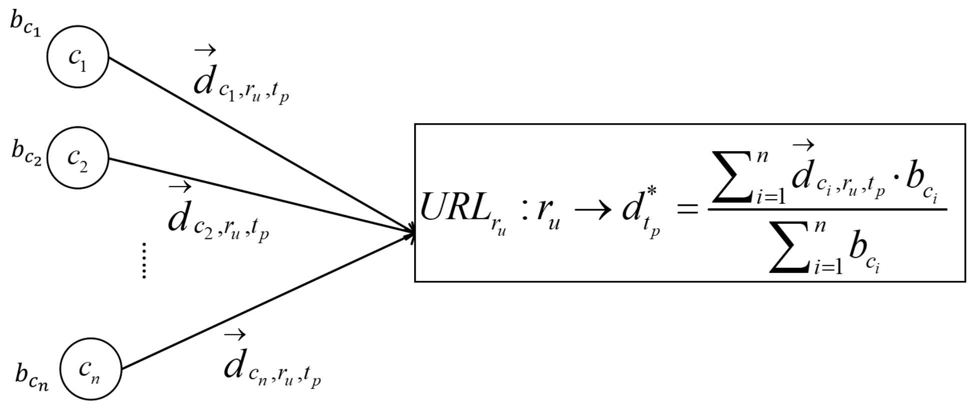

b_{c_1}
c_1
$\vec{d}_{c_1,r_u,t_p}$
b_{c_2}
c_2
$\vec{d}_{c_2,r_u,t_p}$
b_{c_n}
c_n
$\vec{d}_{c_n,r_u,t_p}$
$URL_{r_u} : r_u \rightarrow d^*_{t_p} = \frac{\sum_{i=1}^{n} \vec{d}_{c_i,r_u,t_p} \cdot b_{c_i}}{\sum_{i=1}^{n} b_{c_i}}$

Figure 6.5: The Data Fusion Process in the Resource r_u at the Moment t_p

The data fusion process is running at the resource r_μ with URI_{r_u}, which expresses the semantics of r_μ. The resource r_μ collects all the incoming data from the semantically matched components with the same time stamp. Combined with all the related belief values from different components, the resource r_μ can calculate the fusion data d^*_{tp} based on equation 6.8 as expressed in Figure 6.5. The calculated data $d^*_{t_p}$ is attached to the resource r_μ to express the fusion result, which could be treated as the semantic value. The fusion result $d^*_{t_p}$ can also be used in the next *Belief Updating* part to update all the matched components' belief values.

To take an example showing how the data fusion works, we assume the framework contains three positioning components; c_1, c_2 and c_3 with respective belief value of 0.9, 0.6 and 0.4. At the time t, if the three positioning components respectively send their detected location values $< 56, 43 >$, $< 63, 47 >$ and $< 40, 35 >$ to the resource r_p with $URL_{r_p} : systemX/entityA/location$. The evaluated result $d^*_{r_p,t_p}$ in the resource r_p is:

$$
\begin{aligned}
d^*_{r_p,t_p} &=< \frac{\sum_{i=1}^{n=3} d_{c_i,r_p,t_1} \cdot b_{c_i}}{\sum_{i=1}^{n=3} b_{c_i}}, \frac{\sum_{i=1}^{n=3} d_{c_i,r_p,t_2} \cdot b_{c_i}}{\sum_{i=1}^{n=3} b_{c_i}} > \\
&=< \frac{56 \cdot 0.9 + 63 \cdot 0.6 + 40 \cdot 0.4}{0.9 + 0.6 + 0.4}, \frac{43 \cdot 0.9 + 47 \cdot 0.6 + 35 \cdot 0.4}{0.9 + 0.6 + 0.4} > \\
&=< 54.8, 42.6 >
\end{aligned}
$$

The value of $d^*_{r_p,t_p}$ is the semantic value of r_p to express the location of *entityA*.

6.3.3 Belief Updating

To obtain the correct evaluated data based on the above data fusion process, one of the most important factors is to assign the correct *Belief* values to all the components.

There are many different methods to assign the components' *Belief* properties. The method can be static, based on the historical data and prior knowledge. It can also be dynamic, based on Bayesian probability or other theories. In this Section, we only give some requirements as the guide to use this framework, in which the algorithms updating the belief can be flexible.

Different from other approaches, the belief updating processes in our approach are not individual end-to-end processes from different nodes rating each other. Each resource can become an individual agent to decide all the matched system components' beliefs based on their behaviours, which is a centralised decision in the resource rather than peer-to-peer decision.

Based on all the data from different component sources in the same *Semantic Match*, the resource can update its mapped components' *Belief* properties. The updating rules are based on the following two basic assumptions:

- If more resources think one component is correct, the component has a higher reputation.

- If the outputs from different components are more consistent, these components are more believable than others and will get a higher reputation.

With the two assumptions, we can have the *Belief Updating* mechanism in the following:

For any component $c \in C_{r_i}$, where C_{r_i} is the set of all semantically equivalent components to the resource r_i, we have the *Belief* of this c as:

$$b_{c_{r_i}} = \hbar(c, C_{r_i}) \tag{6.9}$$

Where the $\hbar(c, C_{r_i})$ is a function to check the difference between the data from the component c and other components in the C_{r_i}. The function will get a higher value if the difference is less, and a lower value if the difference is more. The equation means that if the component's output values are more close to other components' that are semantically equivalent to this resource, the component's output values are assumed to be more accurate.

Many system components are multi-functional, which means they can be mapped to different resources based on the semantic equivalences between the functionalities and resources. If a component c is mapping to multiple resources R_c, every mapping resource can give the belief values to it separately, and eventually the *Belief* of this component c is:

$$b_c = g(B_{R_c}), B_{R_c} = \{b_{c_{r_i}} | r_i \in R_c\} \tag{6.10}$$

Where the $g(B_{R_c})$ is a function used to integrate the beliefs from different resources and the range is between 0 to 1. In general, this equation can be summarised in the following: if more children components are reliable, the parent component is believed to be more reliable.

The above two functions only indicate the underlying assumptions, however, the exact algorithms need to consider many different aspects such as the network latency and data loss. In the implementation Section, we will illustrate how to update the belief for each different component using our data structures.

6.3.4 Fault Detection

With the real-time updating *Belief* properties, any system component in the system can self-adaptively detect some runtime faults, especially when one system component causes some permanent faults.

We can set a threshold value Ψ, and if any component has the belief value lower than Ψ, the system will trigger the event of warning the related components. In some complicated situations, the different components can also have different threshold values, thus we can have more fine-grained operations on the fault detecting.

The fault detection is based on components' belief values, which are automatically updating at runtime, therefore the fault alarm is self-adaptive, which can let the deployed IoT systems to detect any unexpected fault in real-time without any prior knowledge. This is important because it is impossible to consider all possible states when developing the software systems, and this self-adaptive fault alarm mechanism can deal with most of the undefined behaviours and data.

6.4 Implementation

In our implementation, we use meaningful URIs (Uniform Resource Identifiers) with web technologies to build the directed graphs with the semantic match. Specifically, the Semantic-based Reputation Framework is developed based on the Resource-Oriented Architecture with REST (Representational State Transfer) architectural style principle, and the detailed implementation technology is based on Spring framework with Java programming language.

The framework is built from different RESTful web services which use the URIs to annotate the service's semantic descriptions. All the operations in the reputation framework are based on the HTTP request-response model. Following the REST principles, we use the *GET* to get the resources' descriptions, *POST* to generate new resources, and *PUT* to update the resources. Some RESTful web services in our implementation are expressed in the table 6.1. For any system component, as long as it can request web services, it can register itself to a specific resource node via *POST* operation, and use *PUT* operation to give real-time values to the related resource node. The resource node can receive the real-time data from different system components and produce the final data by evaluating those data produced with the same time stamp. Then, the resource node can give related belief values to all the requested system components.

REST Verb	URI	Description
GET	/iotSystem	Get the system description
GET	/iotSystem/resourceNodes	Get all the resource nodes
POST	/iotSystem/resourceNodes	Add a new resource node
POST	/iotSystem/{resourceNode}	Attach some system components to a specific resource node
PUT	/iotSystem/{resourceNode}	Update the data and get the belief

Table 6.1: The Example of RESTful Web Services in the Framework

To allow the resource node respond to the request from different matching system components, we designed the particular data structures to compare the different data from other semantically equivalent components. The structure is used to manage the data from different system components, which is shown in Figure 6.6. Since the networks are not always stable and all system components and web services cannot guarantee the time synchronisation, we designed the paralleled linked lists to support the data fusion and belief updating mechanism via tracking the *timeslot*s. The parameter T_{in} is to decide the size of the *timeslot*s, which depends on the system contexts.

Each resource node maintains several linked lists to handle the requests from different matching system components. Each linked list acts as the buffer to store the data from related components, and it only keeps the size of k depending on the time duration that the historical data is kept. After every T_{in}, each linked list will drop the last element and add a new element from the other end. This runtime updating structure can guarantee the resource node always gets the data from the components and compares them in the same timeslot with the similar time stamps even when there are network latencies or unbalanced source components updating frequencies.

Every semantically matching system component can send the request to the related resource nodes to fuse their data and get updated belief value. Assume a component c^{ra} sends a PUT request with the data d_t^c and a time stamp t^c is attached to the $/iotSystem/ra$, the resource ra will process the request with its own *process* function, and the algorithm is indicated in the Algorithm 1 shown below.

The belief updating strategy is based on both historical belief values and the current calculated value, therefore another buffer is used to store historical belief values. The historical belief values can be used to apply PI (proportional-integral) control mechanism [8] to stabilise the belief value. In this implementation, we can simply set a length of l, and

Algorithm 1: The process service for any matched component: *process()*

```
input  : request from a component with its name, time-stamp, and value
output : the updated belief value for the component

currentTime = timer.getTime();
if components.contains(requestComponent) then
    /* Decide the correct time slot and store the data */;
    $intervalCount = (componentTime - currentTime)/intervalTime$;
    oldElement = listC.get($currentPoint + intervalCount$);
    if oldElement.time < currentTime then
        listC.set($currentPoint + intervalCount$,request);
    /* Data Fusion to get the evaluated value with FuzzyWeightedAverage algorithm
     */;
    for each component list lc in the resource do
        if lc.get($currentPoint + intervalCount$) != null then
            fusingDataList.add(lc.get($currentPoint + intervalCount$));
            beliefWeightList.add(beliefC);
    $evaluatedData =$
     $FuzzyWeightedAverage.fuse(fusingDataList, beliefWeightList)$;
    /* Update the belief value of the request component */;
    $beliefC = beliefUpdate(evaluateData, comingData, updateStrategy)$;
```

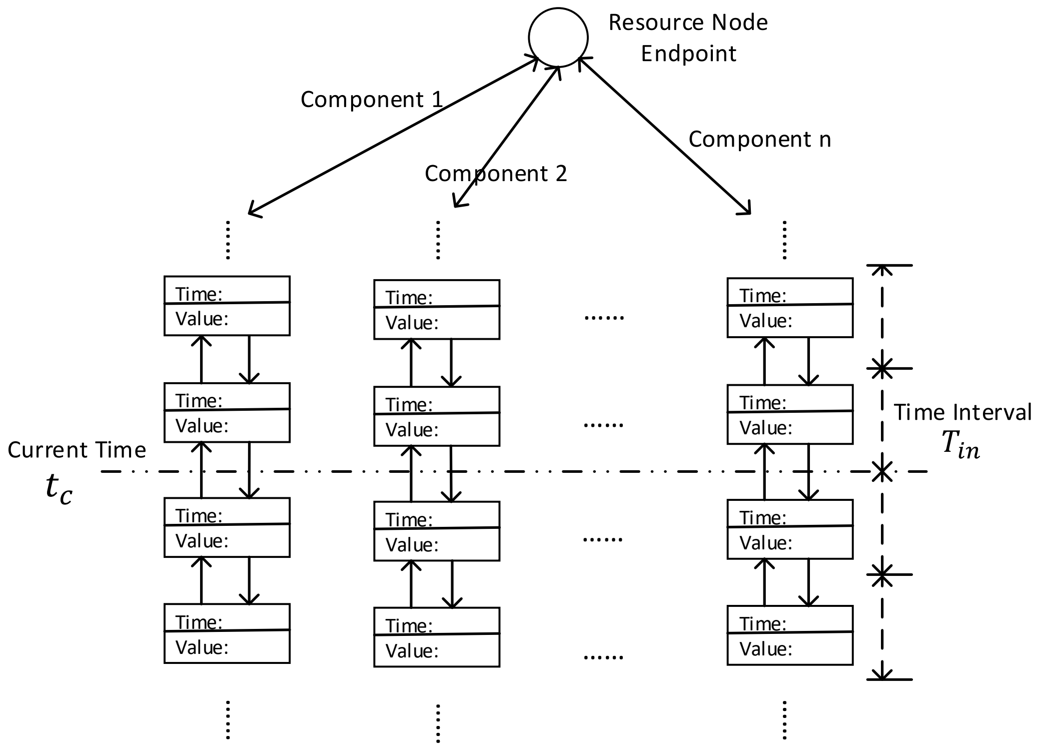


Figure 6.6: The Data Structure in the Resource

only calculate the average value of the latest l belief values. The length of l is very critical because it can significantly affect the framework's performance. Generally, if the length l is longer, the evaluated belief value will be more stable and fault-tolerant. However, it will also become very insensitive to all faults in the meantime, thus some faults may not be reported promptly. Therefore, the parameter needs to be chosen carefully to balance the different requirements from the specific systems.

In this implementation, we did not have consistency mechanism to allow different resource nodes to negotiate the final belief value for any system component. Therefore, the belief value fusion from different resource nodes' responses has to be done at the client side, which means the system component needs to decide its final belief value by itself.

6.5 Evaluation

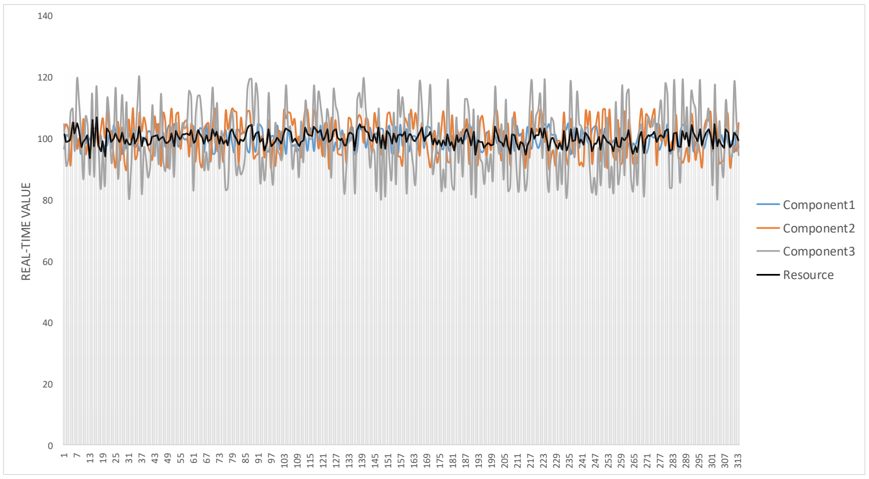
Because the framework is not deployed in the real IoT systems, randomly generated data is used to simulate the real environment to test the framework with specially designed data structures and algorithms. The evaluation data is generated from client-side component nodes from the component nodes at the client side with *RestTemplate* which is a technology provided by Spring Framework.

In this Section, we assume we have three different components matching with a specific resource, and they have the different type of accuracy level. Component1 is the most reliable one, yet component3 is the worst. The real value in the physical world is around 100, and each request their data to the resource node.

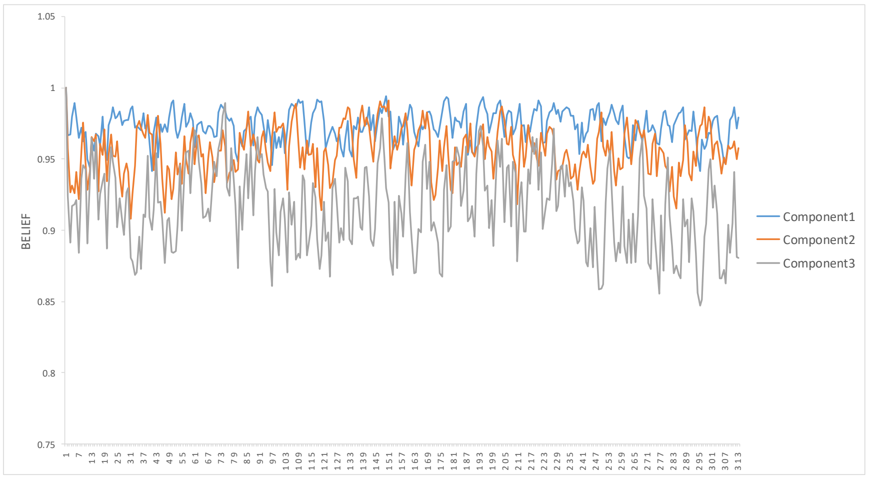
In the framework, the belief updating part only gives the basic assumptions without detailed algorithms. The method to evaluate the belief values can be difficult. However our purpose is to test the performance of the framework, therefore to make the simulation clearer, we use simple linear algorithms to evaluate the belief values. That is, if the evaluated data is d_e, and the requested data is d_r, the belief value for component i is $b_i = 1 - \frac{|d_e - d_r|}{d_e}$. We decide to use the PI control mechanism, which means we store and use the historical belief values, otherwise, the assigned belief values will be very unstable and may cause occasional problems. The Figure 6.7 indicates the results without PI control mechanism where we do not use historical values to stabilise the output.

If we want the assigned belief values more stable in the given environments, we can apply the PI control mechanism by calculating the average value from the latest historical belief values with the length l, where l is the size of the timeslots. The PI control mechanism can help to reduce the effect from the short term device deviations. In Figure 6.8, we set the length l as 7 and it can be observed from the figure that the belief values become stable after about 80 timeslots.

With the same configuration to test the performance of fault detection, we let the *Component2* generate 0 after a certain time. The results are expressed in the Figure 6.9. It is observed that the belief value of the *Component2* starts to drop down when it only produces 0. The evaluated value from the resource is also affected from the faults at the beginning. However, it eventually returns to the correct value when the belief value of the *Component2* becomes very low. Therefore we can conclude if we choose to give up some components in the reputation framework when their belief values are lower than the certain threshold, the experiment reputation is probably more robust.



(a) Data Fusion

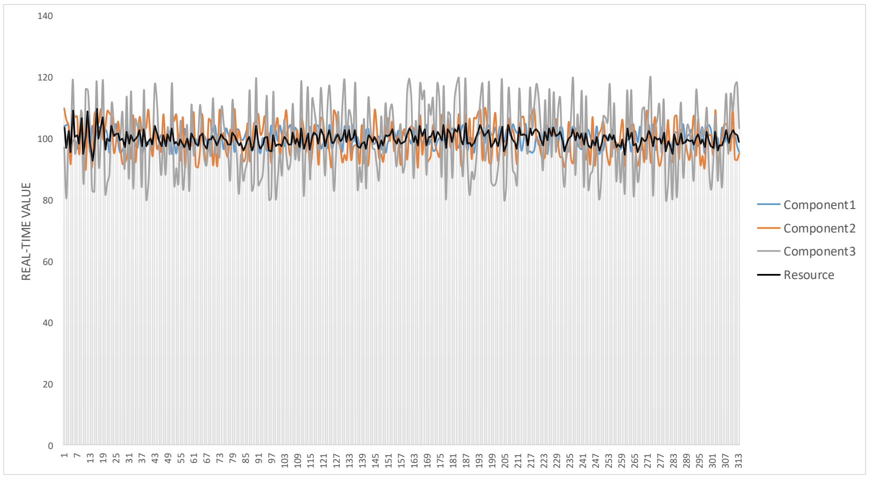


(b) Belief Updating

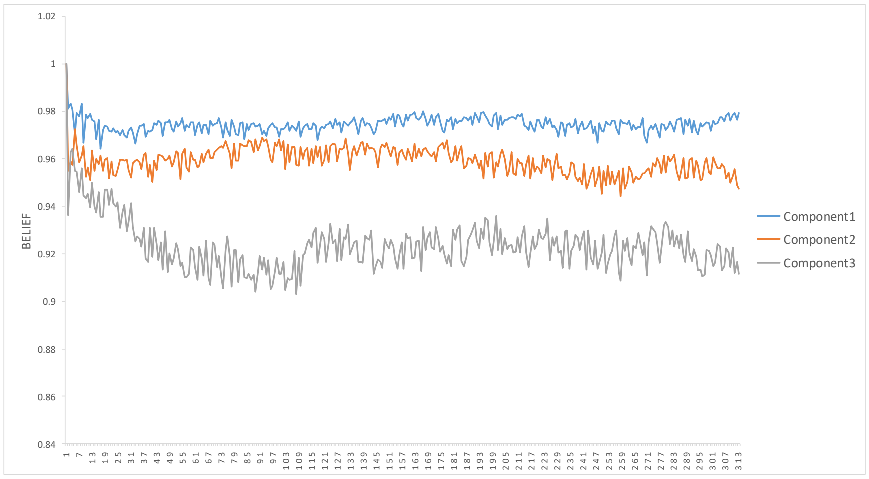
Figure 6.7: The Normal Running Mode without PI Control

To further our investigation, if we set the length I, which is the large scale of the historical data used in the PI control, the belief values will become very stable, which is illustrated in Figure 6.10. This effect is from the PI control mechanism.

However, if there is a permanent fault with the *Component2*, the evaluated value will be affected too much and other components' belief values will also be affected. Figure



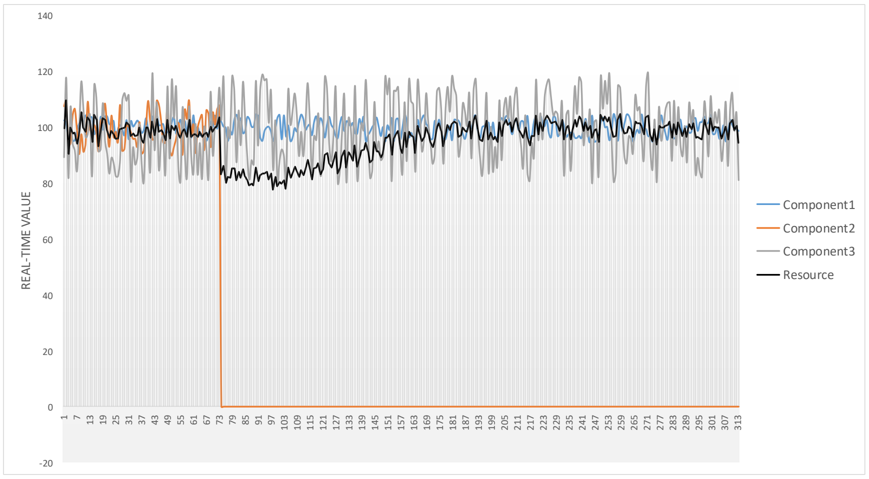
(a) Data Fusion



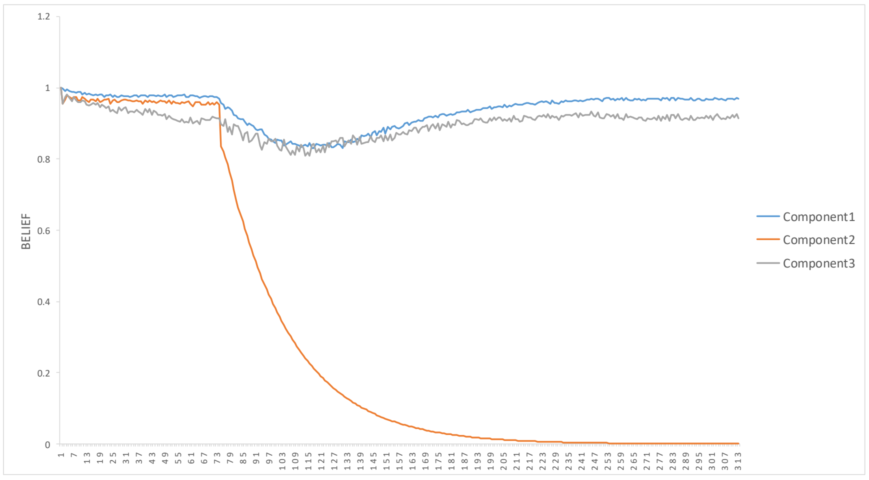
(b) Belief Updating

Figure 6.8: The Normal Running Mode with PI Control

6.11 illustrates the results when the *Component2* causes a permanent fault with high l. Fortunately, even in this case, it can be detected if something goes wrong because all the three components' belief values will decrease quickly. Therefore, this kind of situation can also be detected by applying specially designed policies.



(a) Data Fusion

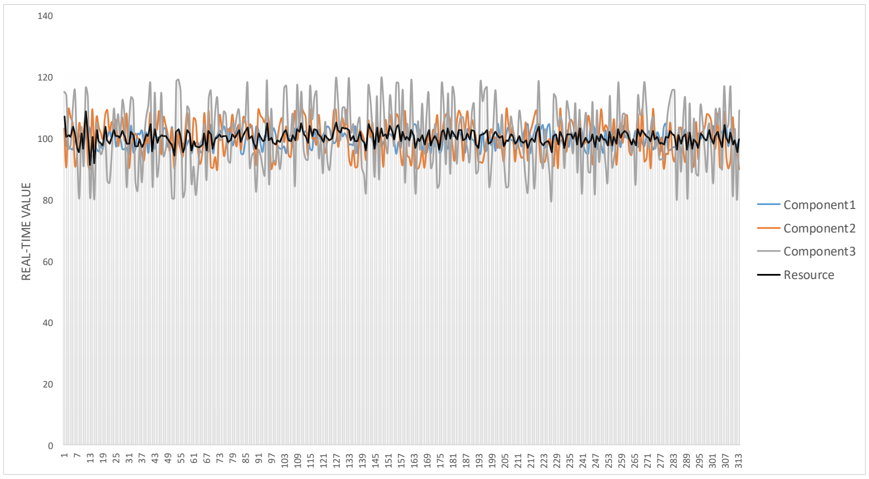


(b) Belief Updating

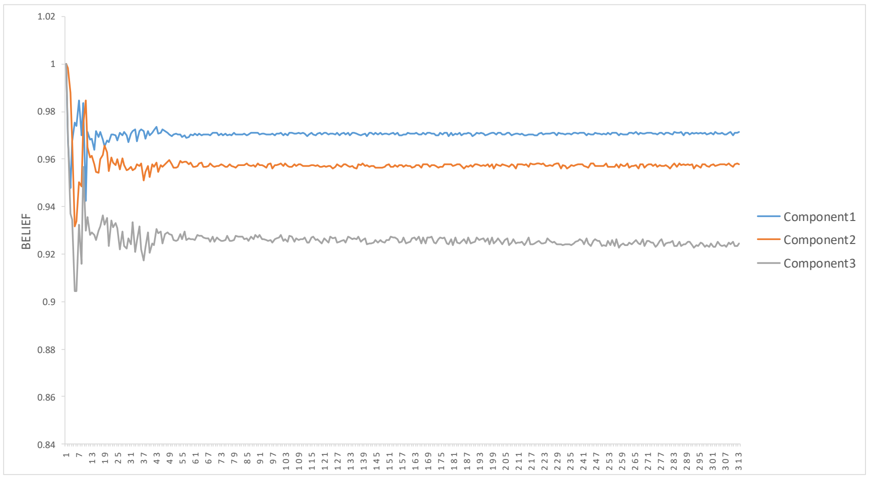
Figure 6.9: The Permanent Fault in the Component2 is Detected

6.6 Conclusion

In this Chapter, we propose a novel Semantic-based Reputation Framework for the Internet of Things to handle the uncertainty at runtime. The framework focuses more on data accuracy with data fusion, and fault detection. Compared with some existing reputation-



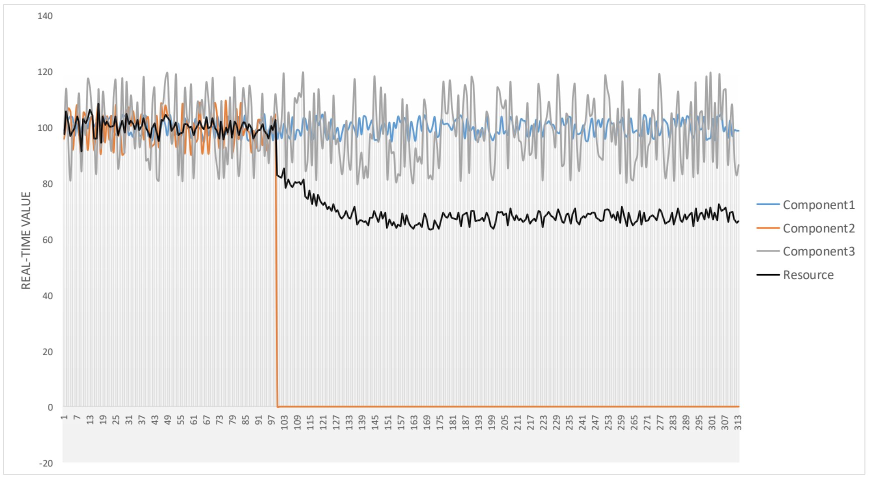
(a) Data Fusion



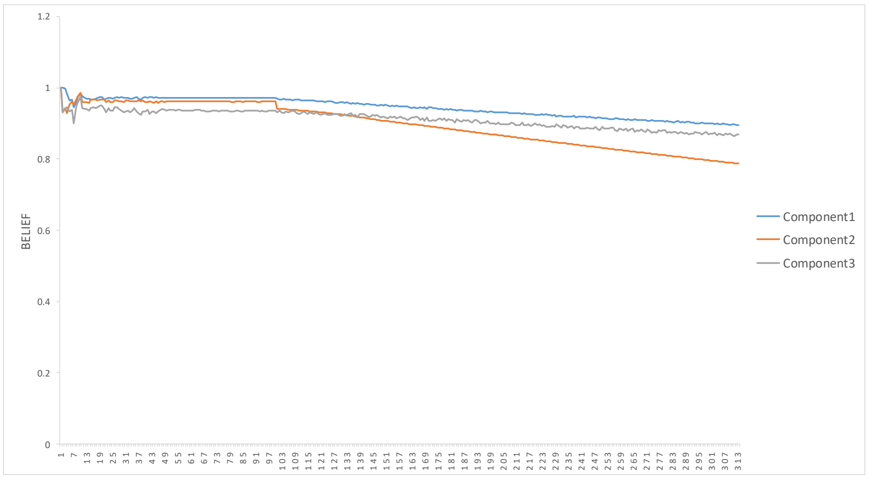
(b) Belief Updating

Figure 6.10: The Normal Running Mode with PI Control and the High l

based framework for WSN applications [60], this framework can handle more heterogeneous systems via constructing the directed graphs with the *Semantic Match* mechanism. We implemented the framework with the RESTful web service technology via Java Spring framework and we designed our own data structures and algorithms to handle the requests from different systems components. Some experiments in the simulation environment are



(a) Data Fusion



(b) Belief Updating

Figure 6.11: The Permanent Fault in the Component2 Causes Wrong Data Fusion

made to evaluate the proposed Semantic-based Reputation framework and the specially designed data structures and algorithms. In the experiments, the PI control mechanism is used to stabilise the randomly generated simulation data. The evaluation results indicate that the framework can work as expected.

Compared with other cutting edge approaches like Social Internet of Things, our approach can use all existing well proven web technologies such as HTTP and RESTful Web service which can handle large scale systems. The algorithms used in this Chapter are some simple algorithms, however, based on the provided framework and structures, more complicated algorithms can be used to adapt to different IoT systems.

The core structure of this framework is built by the *Semantic Match*, which uses the semantics as the bridge to connect different physical devices and software components. The implementation of the semantic match discussed in this Chapter is rather simple since only the basic syntax of “resource” and URI are used to express the semantics. However, it is possible to use rich semantics related technologies like RDF (Resource Description Framework) to do reasoning and inferring in our framework in order to express more complex semantics, thus any existing artificial intelligence technology in semantic reasoning with uncertainty like [29] and [30] can be used in the IoT systems using our proposed framework. And this is one of the advantages of our framework.

The future work is to enrich the expression of the *Semantic Match* with Semantic Web technologies to allow the framework to automatically detect the appropriate system components to construct the reputation networks via the *Semantic Match*. Furthermore, we also plan to deploy the framework in some more complicated applications to test the performance, more importantly, to implement different algorithms to test the generality and extensibility of our framework.

Chapter 7

Access Control

7.1 Introduction

Due to the cost reduction in hardware and network infrastructure, the rapidly growing number of Internet-connected devices bring us the prospect of the IoT applications in the future. This will result in a seamless integration of the physical world into the digital world, and tremendously benefit various domains, as shown in some works related to intelligent transport systems [88], smart homes [62], smart grid [195], smart buildings [189] and smart cities [198].

For the broad scale cross domain IoT applications, security and privacy become one of the essential challenges [130], because the requirements are different from the traditional computer systems or Internet applications. The IoT applications are usually deployed in the open environments involving various different users, who might damage the applications or steal sensitive information. Furthermore, because of the inherently spatio-temporal feature of the IoT applications, the security and privacy mechanisms for IoT are usually very context-sensitive, which is not the primary focus of the traditional security and privacy protection mechanisms. Therefore, the access control mechanisms in the IoT applications need to be fine-grained and flexible to offer features such as scalability, interoperability, efficiency and context-awareness.

Some traditional access control approaches like Role-Based Access Control (RBAC) [159] cannot be applied to IoT application directly because they cannot provide adequate functionality to adapt to dynamically changing contexts. Although the Attribute-Based Access Control (ABAC) [193] can provide fine-grained control, the process of authorisation is complex and inflexible, thus it is difficult to be used in the open IoT applications which may include millions of changing users [139]. To accommodate the different requirements, many current approaches deal with different aspects with hybrid mechanisms such as Attribute-Role-Based Hybrid Access Control (ARBHAC) [90], Capacity-Based Access Control (CapBAC) [69][77] and Context-Aware Role-based Access Control (CA-RBAC) [104].

The dynamic access control mechanism proposed in this Chapter is also a hybrid approach consisting of context, role, and attribute-based access control. We design the mechanism specifically for the open dynamic changing physical environment and the most different perspective is that the access control mechanism is for services rather than data. Sometimes the services are continuous processes, thus we also have the special life-cycle management which is not contained in most of the other approaches. The mechanism is implemented in the OWL (Web Ontology Language) to provide a flexible way to represent and reason the access policies.

The Chapter is organised as follows; In Section 7.2, some related works are briefly introduced. The motivation is explained in Section 7.3. Section 7.4 expresses the whole Context-States-Aware Access Control model, including the basic conceptual model in Section 7.4.1, its ontology model in Section 7.4.2 and the policy model in Section 7.4.3. The logical architecture of this access control mechanism is introduced in Section 7.5. Then Section 7.6 uses the example from Section 7.3 to illustrate how to implement the policy rules using SWRL (Semantic Web Rule Language). Finally, Section 7.7 concludes the Chapter and explains future work.

7.2 Related Works

Access control is a mechanism to determine whether a request to access a service or an information resource provided by a system should be permitted or denied. As one of the most classical access control mechanisms, the RBAC has been continuously developed for many years from RBAC [159], NIST RBAC (National Institute of Standards and Technology RBAC) [56], to GTRBAC (Generalized Temporal RBAC) [89] and risk-aware RBAC [33]. Even though the RBAC provides a powerful abstraction to overcome some safety issues and limitations in the Discretionary Access Control (DAC) or Mandatory Access Control (MAC), people start to realize that many other aspects like spatio-temporal information also need to be paid attention to satisfy some complex requirements, therefore the Generalized

Spatio-Temporal Role-Based Access Control (GSTRBAC) model [1] is proposed.

During the progress, some important aspects like contexts [180] or semantics [75] are defined and explored to integrate into the security mechanism, especially with the development of web applications, and IoT applications, which are usually deployed in the open dynamic environment. The CA-RBAC (context-aware RBAC) combines the context-awareness with RBAC in the pervasive computing systems [104]. In the [93], a context-aware access control model is proposed based on ontology with semantic web technologies. While many different focuses in access control are proposed like privacy preserved access control [82], criticality aware access control [68] and purpose-based access control [144], a fine-grained mechanism, ABAC, attracts a great deal of attention [83][185]. However, since the ABAC model is too generic and difficult to be deployed, people still prefer using a hybrid mechanism such as ARBHAC [90], role and attribute-based access control with Semantic Web technologies [35] or ABAC in Authorisation and Authentication Infrastructures with Ontologies [149].

ABAC model is very powerful and able to provide fine-grained access control policies, however, it cannot control the way that the users use the assigned services. Due to this limitation, a different access control model called Usage Control (UCON) is proposed [142]. The main novelties and advantages of the UCON are mutability of attributes and continuity of access decision evaluation. A context-aware usage control model (ConUCON) is proposed in the [11] for the Web of Things.

7.3 Motivation

Most current access control mechanisms still aim at data rather than services. The biggest difference is that offering data is always a transient behaviour while offering services can be a continuance process. Therefore accessing some actuating services in the open environment may lead to the conflicts or unachievable policies. The following scenario illustrates that most existing access control mechanisms cannot satisfy our requirements including the fine-grained access policies, context-awareness, and continuously controls of the delivered access permissions:

Scene #1: Student David has an English class with other 20 students and one teacher in the classroom *RoomA* in his university on a hot summer afternoon from 2pm to 3pm.

If we want to define following access policies:

1. The *RoomA* allows any person who belongs to this university and is in the room to access the equipment in the *RoomA*.

2. No one can turn on the heaters or turn the air conditioners when the outside temperature is over 25°.

3. When there is a class in the *RoomA*, only the people who are involved in the class and are in the *RoomA* can access the attached equipment.

4. When the class is over, and no one stays in the *RoomA*, all the former operations on the equipment in the room during the class should be revoked. Normally the lights and all HVAC (Heating, Ventilation and Air Conditioning) equipment should be turned off.

5. During the class, all the attending students have the permission to access air-conditioning and no one should own the air conditioning exclusively. Therefore, their permissions need to be managed with appropriate strategies.

The RBAC mechanisms can only support the first above mentioned policy. Most of ABAC mechanisms and Context-aware Access Control mechanisms can support the first, second and third policies but not the fourth policy, because the *State* is a missing piece of concepts for most of the existing access control models. When the user is authenticated to a specific object's functionality like turning on the air-conditioning, the user has the full permission for the functionality and the classical access control mechanism cannot control how the user uses the functionality. The Usage Control model can deal with the 4th policy because it has the element of *State*, however, the awareness of *State* in the Usage Control model is still simple. If several users are accessing the same air-conditioning, the existing models cannot manage the potential conflicts between different users. For example, the user A wants the target temperature of the air-conditioning to 20°C, while the user B wants the target temperature of the air-conditioning to 25°C. To avoid the conflicts, the existing models can only let any user to exclusively use the device, which is not fair sometimes. The *State* element should play a more important role with a more flexible definition to apply fine-grained access control policies based on the different usage situations.

In order to address the above issues, we need to have the fine-grained and flexible access control from ABAC, the efficiency from RBAC, and the method to control how the subjects use the resources even after the subjects are authorised to access the resources. Therefore a hybrid model combining the RBAC, ABAC with some extra state management supports is proposed in this Chapter.

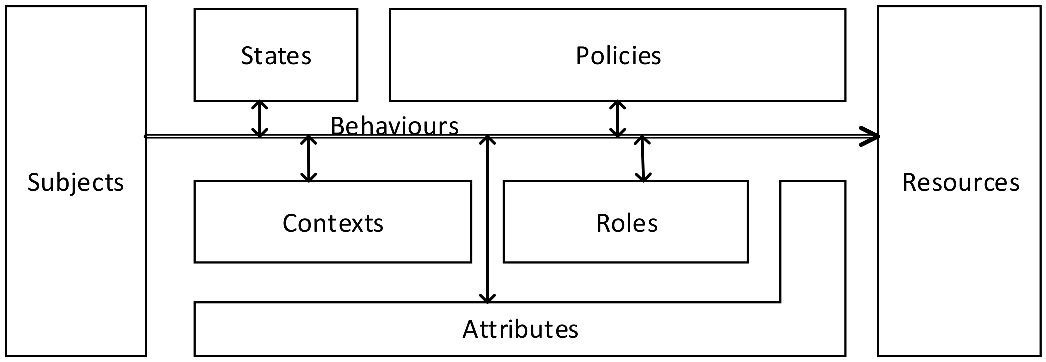


Figure 7.1: The Conceptual Context-States-Aware Access Control Model

7.4 Context-States-Aware Access Control

7.4.1 Context-States-Aware Access Control Model

To provide the fine-grained and powerful access control mechanisms for the different type of access policies in the open IoT systems, the ABAC is chosen as the foundation for full functionalities supports. To reduce the complexity and difficulty of ABAC, the *Roles* and *Contexts* are used as the "cache" to store the attributes that are used more frequently. When the *Roles* and *Contexts* information are sufficient to verify the access policies of the *Behaviours*, we only use the *Roles* with *Contexts* to confirm the permissions. The other attributes are only required when the information of *Roles* and *Contexts* are not sufficient. In order to manage the continuous *Behaviours* from the *Subjects* to the *Resources*, the states of the *Behaviours* need to be tracked as the feedback to do further control. The Figure 7.1 illustrates the conceptual Context-States-Aware Access Control (CSAAC) Model, which is based on the ABAC model. The detailed components of the CSAAC Model will be explained from the three perspectives: 1. *Attributes* 2. *Roles* and *Context* and 3. *States.*

1. Attributes Definitions

In the ABAC, for the access control purpose, three types of attributes are defined:

- *Subject Attributes.* A subject is an entity (e.g., a user, application, or process) that takes action on a resource. Each subject has associated attributes which define the identity and characteristics of the subject. Such attributes may include the subjects identifier, name, organisation, job title, and so on. A subjects role, naturally, can also be viewed as an attribute.

- *Resource Attributes.* A resource is an entity (e.g., a provided service, structured data, or system component) that is acted upon by a subject. With subjects, resources have attributes that can be leveraged to make access control decisions. A patient's medical record document, for example, may have attributes such as owner, privacy level, and date. Resource attributes can often be extracted from the Metadata of the resource. In particular, a variety of service metadata attributes may be relevant for access control purposes, such as ownership, service taxonomy, or Quality of Service (QoS) attributes.

- *Context Attributes.* These attributes describe the operational, technical, and even situational environment and context in which the information access occurs. For example, attributes such as current date and time, the current virus/hacker activities, and the networks security level (e.g., Internet vs. Intranet), are not associated with a particular subject or a resource, but may be relevant in applying an access control policy.

With these three types of attributes, we are able to set most of the access policies for transient behaviours, however, it is still impossible to apply special policies for some continuous processes with the "life cycle" management.

2. Roles Definitions

Any specific *Role* is assigned to a series of permissions, thus it is more simple and efficient to verify the access policies if the subject has an appropriate role. The relationships between the attributes expressions and roles can be one-to-one or many-to-one. For some frequently used attributes combinations, it is better to assign some reasonable roles to them.

If the attributes expressions and roles have been in a one-to-many relationship, we may need to infer the relationships to eliminate the redundancy and inconsistency. The problem has been solved based on policy languages in [90].

3. States Definitions

States are especially used to describe the continuous *Behaviours* from the *Subjects* to the

Resources. Assume the status of (s, b, r) is $\theta(s, b, r)$, where the s is a *Subject*, the b is a *Behaviour* and the r is a resource. Then a state ϕ is a set of $\theta(s, b, r)$: $\phi = \Theta_{S,B,R}$ where $\forall\theta(s, b, r) \in \Theta_{S,B,R}$.

In the mechanism designed from the UCON [202], the function $\theta(s, b, r)$ is mapped to six different types of states, thus $\theta(s, b, r) \in \{initial, requesting, denied, accessing, revoked, end\}$. However, this definition cannot express more detailed information of the states, for example, in case of the *Resource* used by multiple *Subjects*.

In this Chapter, we use two variables to describe the state: $\theta(s, b, r) =< access : p, revoke : r >$, where $p, r \in [0, 1]$. If the *Subject* s cannot access the *Resource* r with *Behaviour* b, then $\theta(s, b, r) =< access : 0, revoke : 0 >$. If the s exclusively accesses the r with b, then $\theta(s, b, r) =< access : 1, revoke : 0 >$. For some special resources like air conditioners, when the subject shares the accessing permission with others, the state is $< access : p, revoke : 0 >$ and $p \in (0, 1)$. The p stands for the percentage of permission for the subject. When the behaviour is finished, if the resource does not need to reset a device, for example a temperature sensor, the state can immediately go back to $< access : 0, revoke : 0 >$. If the resource needs to reset, for example turning off the air-condition and lights when the user leaves the room, the *revoke* variable is assigned with 1. After the revoking behaviour is finished, the state changes back to $< access : 0, revoke : 0 >$.

7.4.2 Context-States-Aware Model Ontology

Since our access control model is a hybrid model based on ABAC combined with RBAC and Context-States-Awareness, it is possible that access policies may be redundant or even conflicted. To make it more secure, safe, high-scale, and easier to be operated, we use the semantic web technologies for reasoning and inferring attributes and policies, which can simplify the specification and maintenance of the policies.

Based on the above requirement analysis and model definitions, we design a sample ontology of the CSAAC model for general IoT applications. As shown in the Figure 7.2, the ontology contains some most important classes in the CSAAC including *Subject*, *Behaviours*, *Context*, *Role* and *Resource*. Apart from these, we also give some common classes like *Temporal_Context* and *Spatial_Context*.

The ontology is designed based on the CSAAC model expressed in the Figure 7.1 to describe some important attributes. The basic structure is from $Subjects - Behaviours - Resources$, that is, any *Subject* can execute some *Behaviours* on the specific *Resource* with

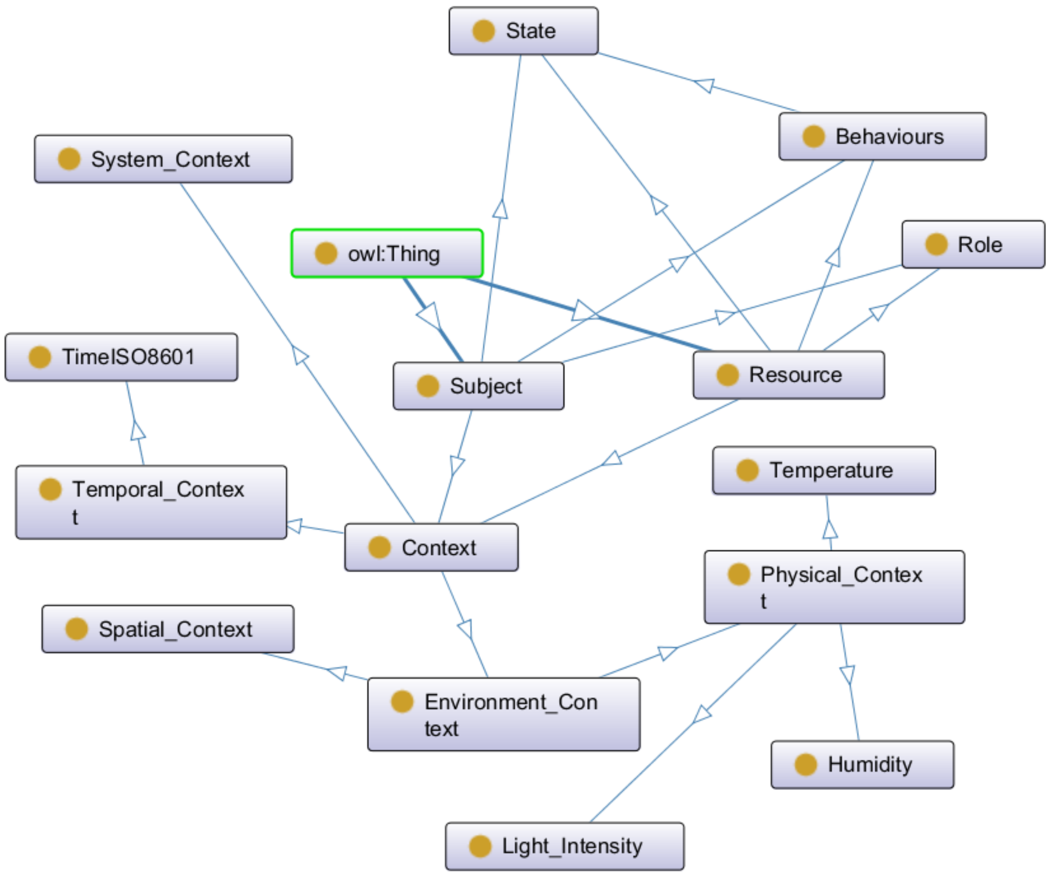


Figure 7.2: A Sample Ontology of the CSAAC Model

the related authorisation under some conditions. Any resource can have some functionalities and these functionalities are the *Behaviours* belonging to the resource.

The *Context* class is used to define some conditions thus the access control policies can be more dynamic to support some special cases. For example, we can define the access control policy for an air conditioner to let it only be turned on when the room temperature is higher than 30°.

The *State* can be used for all the *Subject*, *Resource* and *Behaviours* depending on if they need to have the continuous control over the behaviours. In some cases, if a device contains an actuator, it is very state-sensitive, therefore the state of the device needs to be tracked.

7.4.3 Context-States-Aware Policy Model

To reduce the difficulty and cost of the ABAC, we separate some concerns from all the attributes - *Role*, *Context* and *State*. The policy model can be expressed as follows:

1. S, R, and C are subjects, resources, and contexts, respectively, while Γ are roles and Φ are the states of the subjects accessing the resources. Based on above definition, a state is defined as $\Theta_{S,B,R} \subset \Phi$.

2. $S_k^A (1 \leq k \leq K)$, $R_m^A (1 \leq m \leq M)$, and $C_n^A (1 \leq n \leq N)$ are the pre-defined attributes for subjects, resources, and contexts, respectively, while $\Gamma_\lambda^A \subseteq S_k^A (1 \leq \lambda \leq k)$ are roles attributes.

3. $\eta(s)$, $\eta(r)$ and $\eta(c)$ are *attribute assignment relations* for the subject s, the resource r, and the context c, respectively:

   - $\eta(s) \subseteq \Gamma_\lambda^A \times (\prod^{k-\lambda} S_k^A \setminus \Gamma_\lambda^A)$
   - $\eta(r) \subseteq R_1^A \times R_2^A \times ... \times R_M^A$
   - $\eta(c) \subseteq C_1^A \times C_2^A \times ... \times C_M^A$

4. The *Policy Rule* in our approach is different from the classical ABAC, which uses a Boolean function to decide whether the access is granted or denied. The states Φ we mentioned in Section 7.4.1 will be used to decide whether the behaviours are the continuous processes. If the behaviour is a continuous process and it is assigned with the permission successfully, the state will be used to control how to use the resource. In this situation we may need to control the certain behaviours for a period of time with different mechanisms.
   Accept Policy Rule: $transientAccess(s, r, c) \longleftarrow$
   $continuousAccess(s, r, c) \cdot \Theta_{S,B,R} \leftarrow f(\eta(s), \eta(r), \eta(c))$

5. There is a policy rule base containing many different policy rules maintained by the administrators, covering many subjects and resources. The access control decision process is to evaluate the applicable policy rules.

7.5 System Architecture

The Figure 7.3 depicts a generic architecture for the Context-States-Aware Access Control which can be treated as a context-aware attributed-based access control with extra states tracking support. The architecture is extended from the reference architecture of XACML (extensible Access Control Markup Language) specification [131]. The architecture here is not intended to be fully implemented by Semantic Web technologies because the state evaluation and verification in the continuous processes are not easy to be implemented via Semantic Web technologies. The *States Engine* is developed as a set of state machines to manage the life cycles from all requests from *Access Requesters* to *Resources*.

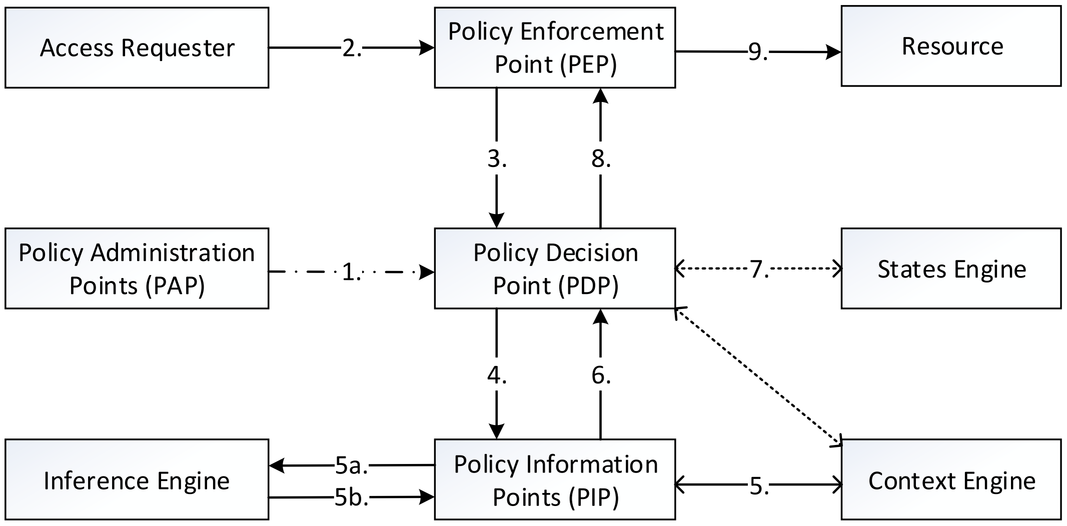

Access Requester
2.
Policy Enforcement Point (PEP)
9.
Resource
3.
8.
Policy Administration Points (PAP)
1.
Policy Decision Point (PDP)
7.
States Engine
4.
6.
Inference Engine
5a.
5b.
Policy Information Points (PIP)
5.
Context Engine

Figure 7.3: The CSAAC Architecture

An access control decision and enforcement may involve the following steps:

1. The PAP (Policy Administration Point) provides the access policies with Semantic Web Rule Language (SWRL) to the PDP (Policy Decision Point). The PDP needs to confirm that any newly added policy based on SWRL is not conflicted with all the existing policies.

2. The user (access requester) sends a request to the policy enforcement point to access a specific behaviour of the requested resource.

3. The PEP (Policy Enforcement Point) forwards this request to the PDP to check if the access requester can get the permission. The first option is to check if the user's *Roles* with *Contexts* are sufficient. If they are not, the PDP checks if the other attributes contained in the request are sufficient.

4. The PDP requests the required attributes from a policy information point.

5. If some attributes are still missing, we can try to deduce them from the attributes included in the request and other attributes provided by the PIP (Policy Information Point), using Semantic Web technologies. The *Inference Engine* is connected to a knowledge base which contains many other ontologies.

6. The PIP delivers the attributes back to the PDP to make further decisions.

7. Based on the decision and the requested *Behaviour* type for the *Resource*, the system can finish the state tracking of this request, or keep observing on the states of how the resource is used. In some cases, the behaviours from the subject to the resource need to be revoked after the behaviour is finished and the revoking process is based on the records in the State Engine.

8. The PDP gives the final decision to the PEP and the PEP shows the decision to the access requester.

9. The PEP meets possible obligations to let the access requester access the resource.

7.6 Implementation

Due to the inherent limitation of the Description Logic in the OWL-DL, some operations of our access control model cannot be fully implemented, especially for the Contexts-States-Awareness. However, we can use SWRL, which is based on the combination of the OWL-DL and Horn logic clause.

According to the requirements, we implement these access control policies for the five intended policies mentioned in Section 7.3, and some reasoning rules listed in the Table 7.1.

From some basic reasoning rules like $S1$:
$belongTo(?s1, ?s2) \leftarrow hasRole(?s1, ?ro) \wedge belongTo(?ro, ?s2)$, we can see how the semantic reasoning helps to integrate the RBAC into the ABAC, because it can infer the relationships

between different roles and attributes. Therefore, it is easier to create new roles or merge some roles. And the roles can also make verification of the policies easier in order to give the appropriate access permissions.

The $S2$ explains that if one resource $r1$ is owned by a subject, then any resource contained in the resource $r1$ is owned by the subject. Thus if a student is authorised for a classroom, all the pieces of equipment belonging to this classroom are authorised at the same time.

The $S3$ is an example used for the *Temporal_Context* to decide if the time constraint condition is satisfied, the related actions can be triggered.

A similar rule used in the context is the S_{i+1}, where the context condition is the temperature. The related air conditioner resource can only be set to the heating mode when the temperature is less than 25°C.

For the *State*, the rule S_n indicates that when the class is over and no one is in the classroom any more, the behaviours done to the pieces of equipment in the classroom during the class should be revoked, so all the resources in the classroom can be as same as before.

7.7 Conclusion

In this Chapter, we propose a CSAAC model which combines the RBAC and ABAC with extra contexts-states-awareness support. The *Role* and *Context* are used as the cache to store some important attributes for dynamic management. We give a different state definition that can flexibly describe some complex states. To implement the model, we use the semantic web technologies to create a sample ontology for the CSAAC model and we use the SWRL to provide some examples of access control policies. Furthermore, a logical architecture is also provided based on the reference architecture of XACML specification.

The CSAAC model is a hybrid model combining many different access control mechanisms to absorb the advantages from different models. The implementation of this model is very difficult and we only use the semantic web technologies to provide a simplified version. Using the SWRL to define the access control policies will bring some issues because it sometimes can be indecisive.

The detailed strategies to manage the states are not discussed in details, especially for the continuous behaviours. Although the complex situation can be described in our model, the strategies can be extremely complex when the different users are accessing the same resource with potential conflicts. To guarantee the fairness, some game theory models -

like negotiation models are considered to be included as future works. In the future, the proposed model CSAAC needs to be applied in the real applications to discuss the detailed control strategies and validate the model technicalities.

No.	Reasoning rules
S1	$belongTo(?s1,?s2) \leftarrow hasRole(?s1,?ro) \land belongTo(?ro,?s2)$
S2	$ownedBy(?r2,?s1) \leftarrow ownedBy(?r1,?s1) \land hasResource(?r1,?r2)$
S3	$satTemCon(?c0) \leftarrow hasTime(?c0,?t) \land op:time-greater-than(?t,?td)$
	$\land op:time-less-than(?t,?tu) \land temConRange(?td,?tu)$
S4	$AssignedResource(?r,?s1) \leftarrow belongTo(?s1,?s0) \land ownedBy(?r,?s0) \land isAvailable(?r)$
Si	......
Si+1	$AssignedBehaviour(heating,?s1) \leftarrow AssignedResource(airCondition,?s1) \land swrlb:lessThan(25)$
Sn-1	......
Sn	$revoke(?s,?b) \leftarrow Subject(?s) \land hasTime(?c0,?t) \land op:time-greater-than(?t,classtime)$
	$\land \neg hasPeople(RoomA)$

Table 7.1: Some Reasoning Rules Examples in SWRL

Chapter 8

Implementation and Case Study

In the above Chapters, the different approaches have been proposed and implemented to solve the specific problems in the decentralised CPS development, as summarized in the following:

- Implement the FASOP for the Behavioural Abstraction in Chapter 4.
- Extend the CoAP to support Real-time Context-Adaptation in Chapter 5.
- Implement the SRF for the Uncertainty Handling in Chapter 6.
- Implement the CSAAC for the Access Control in Chapter 7.

To satisfy all the requirements to develop the decentralised CPS from the resource-oriented perspective, we can use Semantic Web technologies to accommodate all the design principles for the architecture and keep the consistency between different design features. The *Semantics* can coordinate the different engineering requirements. The Resource Description Framework (RDF) can naturally provide the directed graph structure for the resources. Any resource can easily have a *Belief* property using the specific ontology. The resource ontology and semantic web technologies here are used as the specification with a set of tags to integrate the above implementations together as a framework named RInfra (Resource Infrastructure).

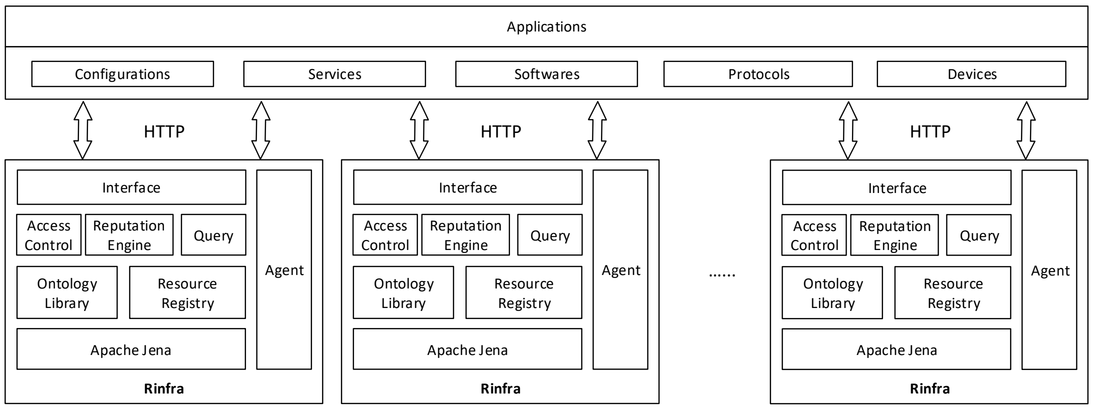


Figure 8.1: The Cross-domain Applications Based on All the Resource Infrastructures

8.1 Resource Ontology in Rinfra

In [12] the existing ontologies in the IoT-domain are summarised. However, based on all the requirements we discussed before, there is no perfect ontology to describe the resource concept in the OROA. According to the requirements of our architecture, the ontology should include sensors, actuators, context information including location and time, *belief* property, platforms, protocols, and services. Therefore we modify the resource ontology proposed in [147], and illustrate it in Figure 8.3. This ontology can contain all the concepts appearing in the OROA.

In this ontology, each *Resource* is modelled as a composition of *Component*s, and each component can provide functionalities specified in the *Capability*. The resource can also be a composition of different resources, all of which are the specifications and proxies for the mapped components. Components can operate their functionalities via the *invokeOperation* with parameters. The *Physical Resource*s can have *Belief*s and this belief property is used to build the reputation-based framework for data fusion and fault detection to handle the uncertainty within the resource infrastructure layer. For all resources, they can have *Token*s, which are used to define the permission for any resource to access the other resources within the resource registries. The resources can be exposed by *Service*s, and the *Interface*s are incremental ones whose types depend on the *Disturbance* to the environment. If the disturbance is high, the resource can choose to use the FASOP to design its API. The disturbance can be specified in the resource.

Compared to the original ontology, the most significant changes that we made are described as the following four aspects.

- Integrate the *Belief* property to build the reputation-based framework.
- Use the Service model instead of the original processes model.
- Define the incremental interface with FASOP to let the system support context adaptation.
- Use the *Token* to set up the authorisation mechanism for security.

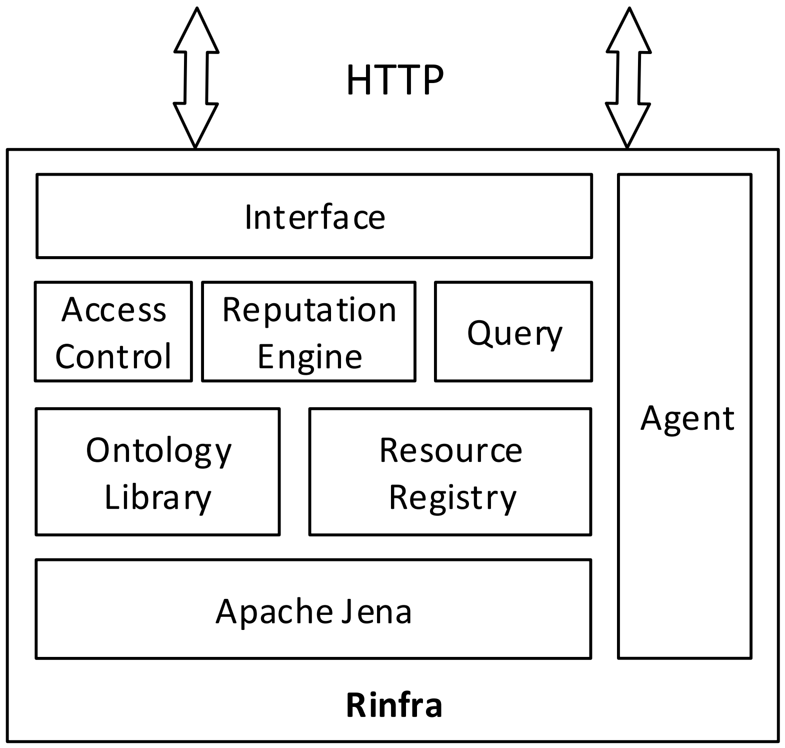


Figure 8.2: The Resource Model Implementation to Build the Infrastructure Layer

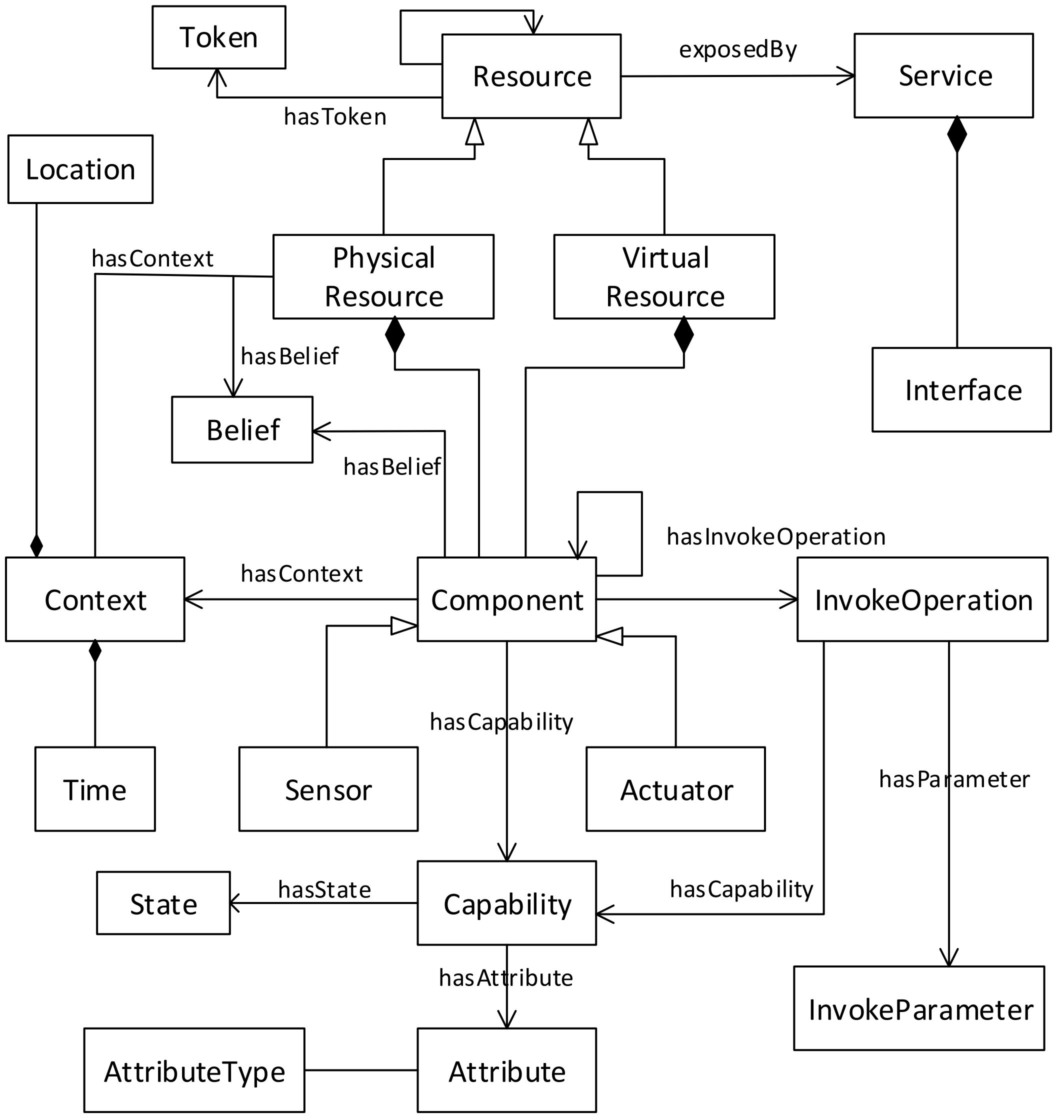


Figure 8.3: The Resource Ontology in the OROA

8.2 RInfra - Infrastructure Unit

We use Apache Jena to develop the *RInfra* framework for the resource infrastructures with the OROA. The basic unit is the *RInfra* server expressed in the Figure 8.2.

Any *RInfra* is a web application that can be treated as a server and a client (agent) at the same time. As a server, it can accept the requests from other *RInfra* or users, and it can also send requests to other *Resource Infrastructural Services* or web applications. The main operations between different *RInfra*s are *register*, *query* and *de-register*.

Since Jena already provides the Resource Description Framework (RDF), the resources registered in the *RInfra* can be organised as the directed graph provided by RDF. In our implementation, Jena is mainly used as the triple store to save all the resource data. The *resource registry* is just an encapsulation to let the Resource Infrastructural Service have one directed graph to store resources. The *Ontology Library* keeps all the required ontology as the resource templates to support the resource descriptions.

The *Access Control* is an integrated module to implement Contexts-States-Aware Access Control based on the resource descriptions. The *Reputation Engine* can be treated as a separate service to keep all physical resources updating their *Belief* values. The *Interface* exposes the interfaces for remote operations, thus in the other applications, users or developers can discover the resources and use them via the HTTP on the web.

The *Query* uses SPARQL Query and SPARQL Update languages to query and update the resource registry. Furthermore, the *resource register* can use the SPARQL update to insert the specific resources, and the SPARQL update also includes the “DELETE” method to *de-register* any resource.

8.3 Cross-domain Applications Development

Using the Service-Oriented Architecture, especially the recently popular Micro-Service Architecture [102], we can develop high-level applications based on the resource infrastructure layer. The Figure 8.1 indicates the overview of the development and deployment of the different CPS applications based on many *RInfra* units as infrastructures with metadata. It can provide more flexibility and interoperability for the CPS applications.

The developers, users and applications can access and request the different *RInfra*s via HTTP. The resources contain various metadata of all system components, allowing the high-level applications to be configured based on these specifications, and the services in the application can be dynamically composed by these resource infrastructures. The HTTP is used to access all the resources in the resource infrastructure layer, however, to access the components mapped by the related resources, we can use different protocols if the components and the applications support the protocols.

For any single resource infrastructural service, its registered resources are dynamically changing based on the high-level applications and environment. In the next Section, we will simulate some scenes in smart transport and smart building as examples to explain how the mechanisms in the OROA work and how they can help to develop the high-level decentralised CPS applications.

8.4 Case Studies

The proposed software architecture and implemented framework is a highly general solution which can be used in most of the decentralised CPS applications. To express the generality of the OROA and RInfra, we use two cases, i.e., smart transport and smart building, to explain how to develop decentralised CPS based on the RInfra and how the different features in the OROA can benefit the systems in different scenes.

8.4.1 Smart Transport Scenarios

This Section refers to how the proposed software architecture and framework can be exploited to develop the applications in the smart transport scenarios. We consider some decentralised CPS applications developed from the evolutionary processes based on the OROA software architecture and the RInfra framework.

One of the basic units in the smart transport is the vehicle, where we use a smart car developed based on the Raspberry Pi to simulate. With the deployed *RInfra* in the smart car, we can show how the future vehicles can work in the context of the smart transport as part of the decentralised CPS.

The Figure 8.4 is a general scene of using OROA in the smart transport where each vehicle is an individual CPS, and they can cooperate based on the infrastructure resources to build the decentralised CPS applications in the smart transport. In a big picture, the resource infrastructure layer is built from the different RInfra servers running on different devices for different services. In the Figure 8.4, it is obvious that each vehicle has its own RInfra server and the crossroad has its RInfra server to handle the requests from

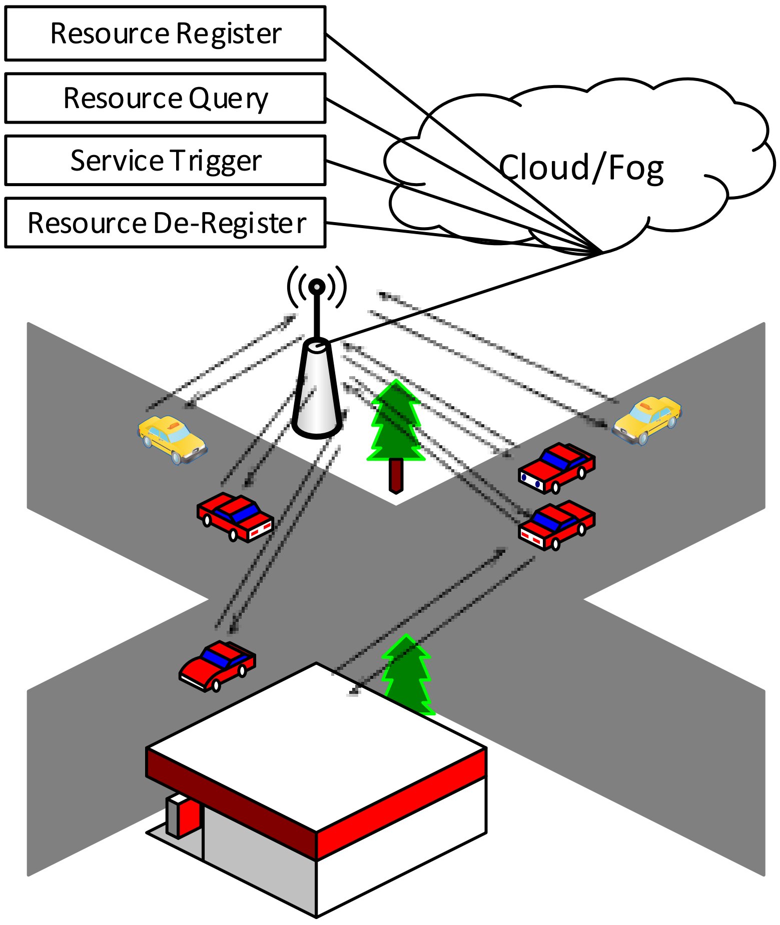


Figure 8.4: The Scenario of Using the OROA for Smart Transport

the vehicles. Moreover, the gas station also has its RInfra server to handle the related applications. For more devices not shown in the figure, the street lamps, street cameras and other sensors are considered and will be used in the following simulation services for different scenes.

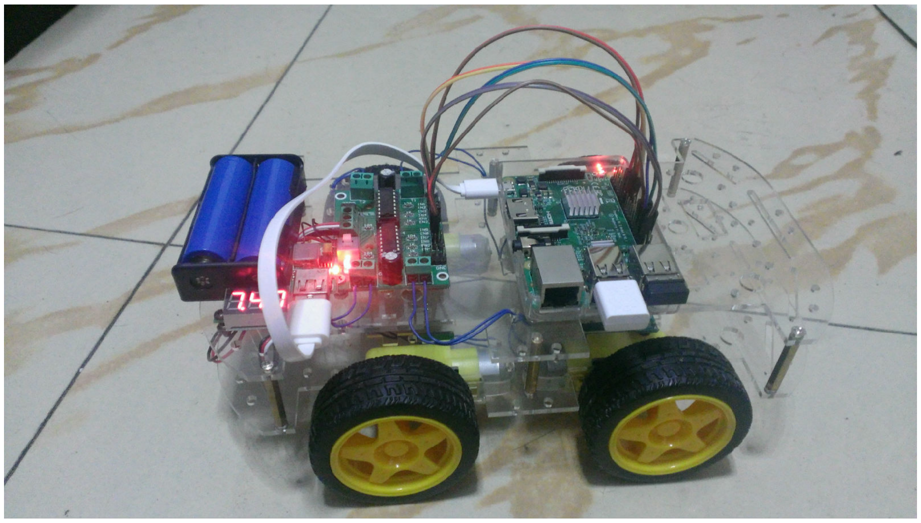


Figure 8.5: The Car Built from Raspberry Pi

Build the Resource Infrastructure Layer

The resource infrastructure layer is built based on the dynamic Register/De-Register processes, thus the different applications can share some owned resources. In this part, we introduce how the Register/De-Register processes work and how the different RInfra servers can share their resources.

The smart car is shown in the Figure 8.5 with one ultrasonic sensor, one GPS and four motors. This car model is used to simulate a real vehicle running on the roads. To bring this car into the smart transport platform, a *RInfra* is deployed in the car to specify all the available resources and expose the related API. The resource registry for this car is illustrated in the Figure 8.6, where all the components in the car are registered. Moreover, the ultrasonic sensor and GPS are composed into the "PerceptModule", and the four motors are composed into the "ActuateModule". The two modules can provide a higher level abstraction of the basic devices to support more complex functionalities. For example, the "PerceptModule" can open an API function to detect the location of the car from several different sensors without giving algorithm details. This directed graph structure can even produce further useful and powerful mechanisms in developing decentralised CPS, which will be explained in the following scenarios.

We assume the entire city is divided into many different blocks. For each block, a deployed RInfra server is responsible for all the resources located in the block. When the car

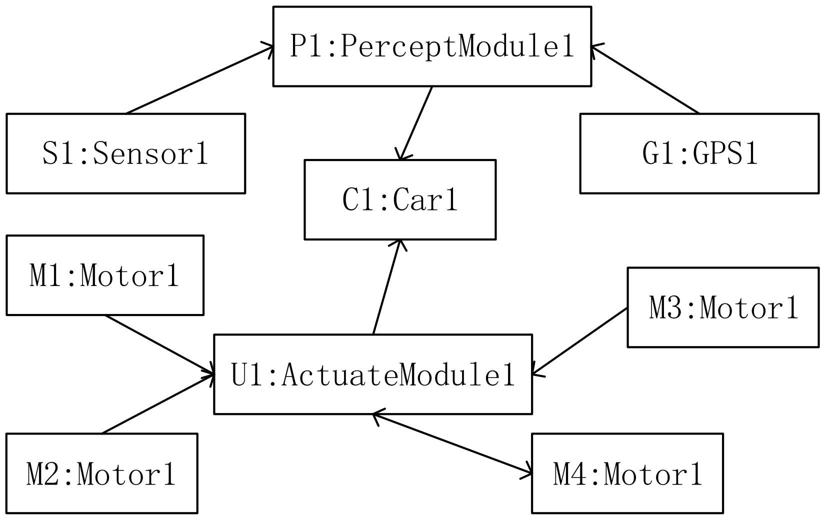


Figure 8.6: The Car's Resource Registry

arrives at a crossroad as shown in the Figure 8.4, if the car can detect the resource registry of this block, it can choose to register itself into the resource registry. In our simulation, we use a laptop as the server of this block to host the street resource registry, and it is also the Wi-Fi hotspot providing the wireless connection to the car. We assume the street resource registry is shown in the Figure 8.7 where the resources are simulated in the laptop. The Figure 8.8 explains the registration process, if the car $C1$ chooses to register itself into this street $S2$, the process will merge these two graphs. If the car has driven away from the street, it can de-register from the street resource registry and all the related resources of the car will be removed from the street resource registry. During this process, the car can also choose to partially register its modules depending on the policies.

The Register/De-Register mechanism is very important in the RInfa framework to build the resource infrastructure layer.

1. It allows the different RInfra servers to query and share their own resources.

2. It limits the size of the resource registries in different RInfra servers, because the unnecessary resources can be de-registered and removed from the directed graph of any particular RInfra.

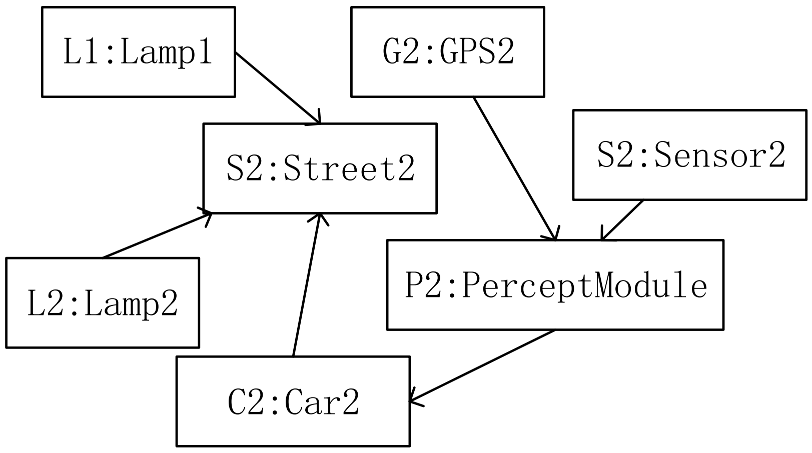


Figure 8.7: The Street's Resource Registry

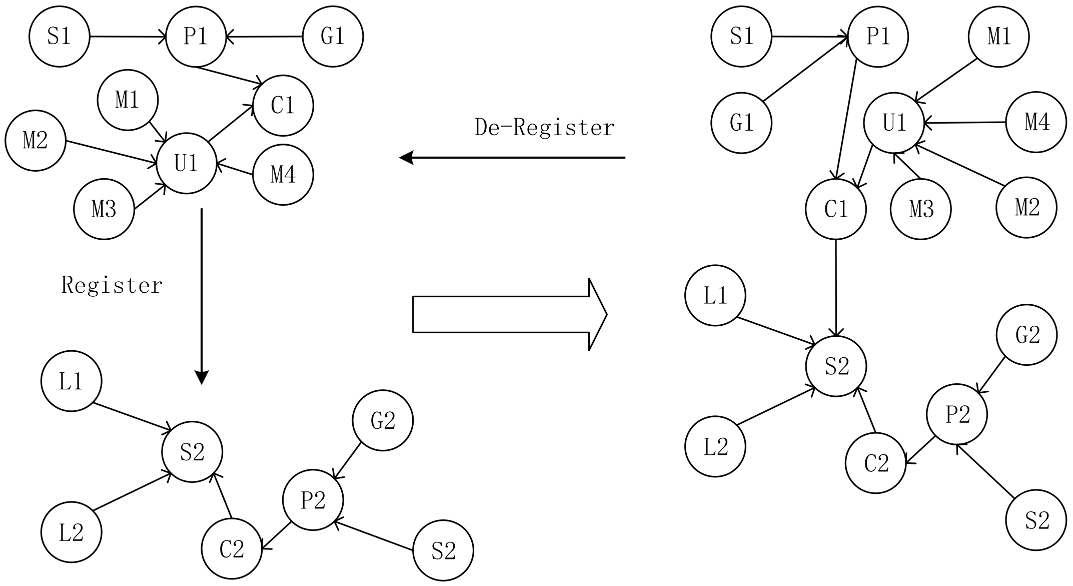


Figure 8.8: The Street Resources Change via Resource Register/De-Register from a Car

In the above scene, when the car registers into the block of a crossroad, which can be treated as part of the transport systems, it will be permitted to find some available information or trigger some existing developed services. The permission is given based on the attributes which are expressed as the resource description in the resource ontology. In our implementation, the car can share its perceptions and speeds on the server side, therefore other registered resources like other vehicles can extend their perception to drive safer and more efficiently.

Belief Updating via the Dynamic Reputation Networks

To illustrate the concept of uncertainty handling mechanisms in the OROA, an example of positioning the car is given in the Figure 8.9.

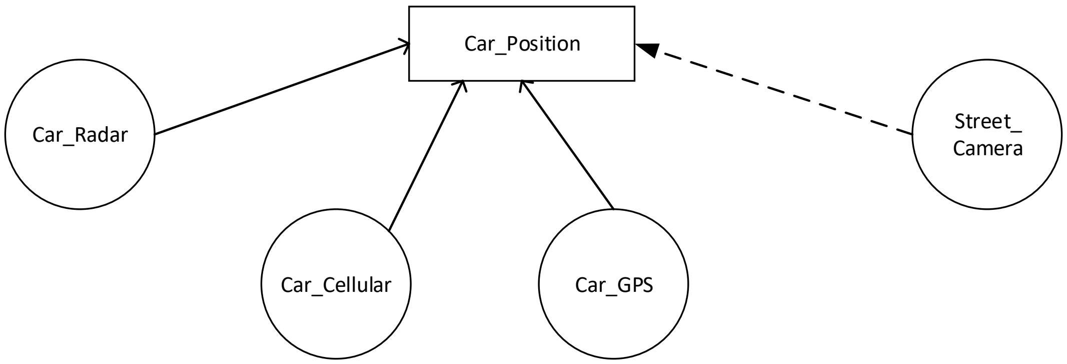


Figure 8.9: The Temporal Semantic-based Reputation Network

Assume the car itself already has three Components: “Car_Radar”, “Car_Cellular” and “Car_GPS” and they can all provide the service to the “Car_Position” resource with the position data of the car. In the existing reputation network in the car, all the components have their belief values, and they can be used for data fusion in positioning the car.

When the car is passing on the registered street, and the related street camera catches the car, the “Car_Position” resource can have a new external component with the semantic matching. The mechanisms and algorithms in the semantic-based reputation framework can be triggered for further data fusion and fault detect.

This feature of OROA can significantly help multi-dimensional data fusion in smart transports and support all the decentralised CPS supported by the infrastructure resources to automatically self-detect the unpredictable faults. Via semantic match, many devices in the different systems can provide the similar functionalities to support data fusion from different dimensions.

Smart Street Lamps Applications

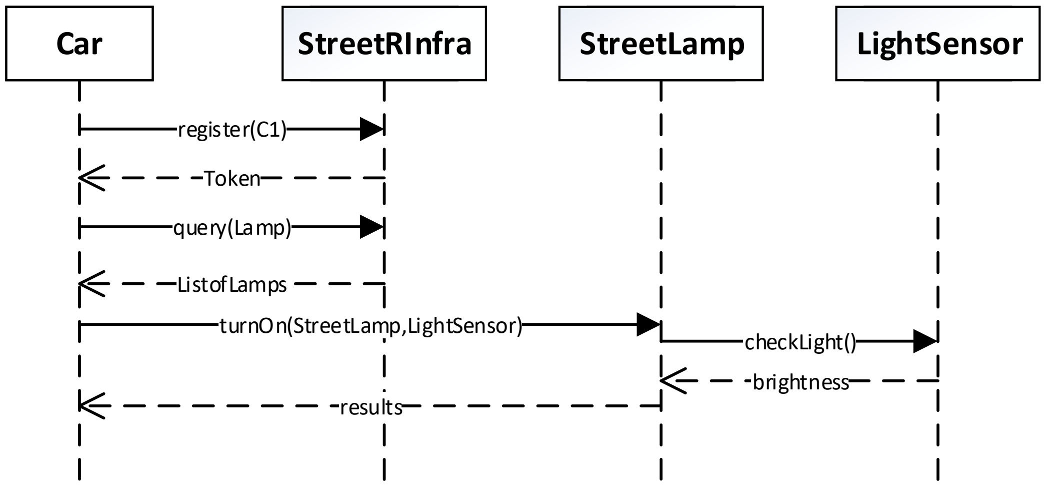


Figure 8.10: The Sequence Diagram to Turn On the Street Lamps

Apart from sharing the traffic information, the RInfra framework can also help to develop cross-domain applications in the smart transport scenes. For example, the smart transport system can contain a smart street lamp application to save energy. Most of the street lamps are off. However, any car can turn a lamp on by sending it a request to trigger the *turnOn* service and allow it to illuminate for a period of time t. When the car registers itself into the cross-road, it can *Query* the resource registry to find available road lamps around the area managed by the RInfra deployed in this block. As shown in the example, the car $C1$ can send a request, then turn on these lamps and the whole process can be described in the Figure 8.10. The behaviour here is following the Feedback-based Context-Adaptive Paradigm, so the car can have the entire required information. The services here can help the streets to save more electrical energy since the street lamps are only on when they are requested. After a specific time period t_μ, the street lamps will be turned off unless the “turn on” service is used again.

One of the advantages of the OROA is that we can always develop and deploy different services via the evolutionary processes. Consider a different required service from the Smart Street Lamps Applications, every car, bike and pedestrian has its own sphere of light provided by a set of smart street lamps. The sphere is determined by the number of lamps that increase the brightness of their LED lights. The size of the sphere is based on the vehicle’s or pedestrian’s speed and adapts to the speed at runtime. Furthermore, if the vehicle is travelling over the speed limit, the vehicle is given an alarm. This requirement is more complex than the former one, however, based on the existing deployed resource infrastructures, the development of the service is not difficult.

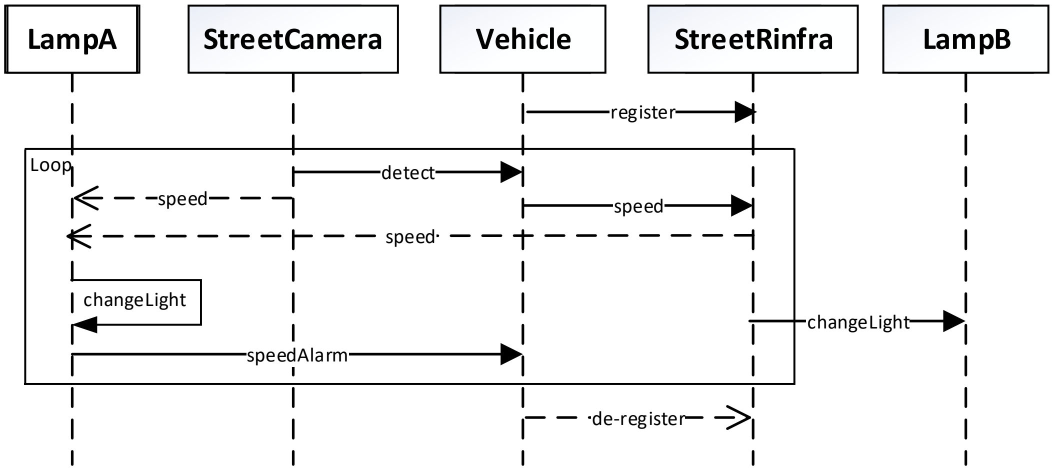


Figure 8.11: The Sequence Diagram for Smart Lighting Services

To complete the required service, there are four required functions: 1) detect the presence of an object (car, bike, or pedestrian); 2) compute an object’s speed; 3) increase and decrease the brightness of the related lights; 4) send and receive messages to and from neighbour lamps. If we need to develop this service from scratch, it is not an easy task. However, if all required devices are already deployed as the resource infrastructures like in the former scene, the service can be developed rapidly based on the existing resources.

Figure 8.11 indicates the sequence diagram of this service.

The vehicle initially registers into the RInfra server of this street, and the street camera can detect the vehicle at the same time. The camera can report to lamp A about the vehicle's speed and the vehicle itself can report its speed to the street RInfra. Based on the data fusion in the reputation framework, lamp A can accurately read the vehicle's speed and decide how much brightness it provides. The decision is not only based on the information of the vehicle, but also on all moving objects that this street can detect. At the same time, lamp A will also cooperate with neighbouring lamps, such as lamp B, to provide the appropriate brightness. If lamp A finds out the vehicle exceeds, the lamp will activate the alarm.

Vehicle Driving Assistant

When many vehicles all register themselves into the street resource registry, this resource registry can grow very large. It also contains lots of real-time traffic data which is the crowd sourcing from all registered resources. By querying the resource registry, the car can obtain the information of other vehicles running on this road. If all other vehicles expose their sensors and perception data by querying surrounding vehicles, all of these can share their readings. These extended perceptions can help vehicles evaluate the real situation and make appropriate decisions. Furthermore, all shared real-time traffic data can be used to manage the public transport systems and optimise traffic. With the support from built resource infrastructures based on the OROA, we can develop different cross-domain applications. More significantly, most of the developments and deployments can be reused. It can reduce the difficulty in developing and deploying the decentralised CPS, and also inspire the creativity.

Within the topic of self-driving vehicles, the RInfra framework can also help the process of remote controlling the vehicle if it can assign its actuators' permission to a remote user. Assuming the StreetRInfra contains a central controller to dispatch all the registered cars in this StreetRInfra, because the StreetRinfra can have more street information than any single vehicle, theoretically, the StreetRInfra is a better option to optimise the traffic. The sequence diagram 8.12 indicates the process of remotely controlling a car.

For a vehicle wanting to transfer the control permission to the central traffic controller for remote control, the vehicle can register itself into the RInfra server of the street. The vehicle can choose to assign its actuators' permission to the central traffic controller. The

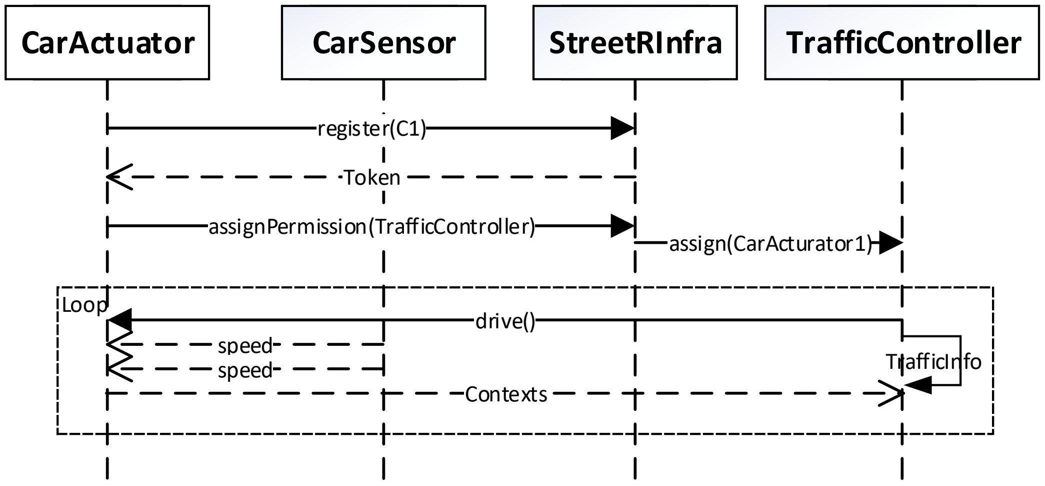


Figure 8.12: The Sequence Diagram to Remote Control the Car

advantage of the central traffic control is that it has much more traffic information than any individual vehicle. The central traffic controller not only has all the information reported from the registered vehicle, but the entire street's resource infrastructures such as street cameras. The large data collected from the real-time traffic environments help the central traffic controller to have comprehensive cognitions to make smart decisions. Furthermore, the central traffic controller can afford better computation resources than any individual vehicle.

In summary, based on the OROA and RInfra, many possible services and applications are available in the context of smart transport, and the proposed solutions can greatly lower the entry barrier for decentralised CPS in the smart transport area.

8.4.2 Smart Building Scenarios

Compared to the smart transport, the smart building is a more well-defined field, thus there are already many solutions and case studies in [2][167][51][14].

For the existing technologies used in the smart building, REST architecture is a popular choice, and many solutions are proposed [53][54] ever since a web service-based approach is presented to integrate resource-constrained sensor and actuator nodes into an IP-based network using REST architecture in [162]. Because the proposed OROA architecture is based on the REST, in this Section, we focus on selected cases highlighting the advantages of OROA compared to the REST.

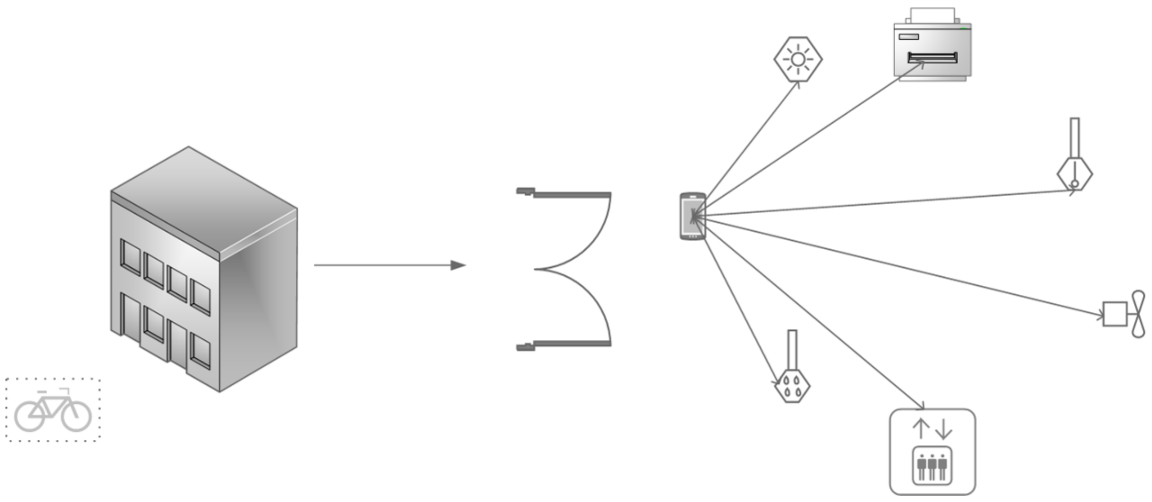


Figure 8.13: The Scenario of using the OROA for Smart Building

The Figure 8.13 is a big picture of the scenarios in the smart building. We consider the mobile phone as the client side terminal to interact with the equipment in the smart buildings. For any person who has the related mobile app, the appropriate permission will be given to use any available pieces of equipment in the smart building.

For most scenarios, the OROA can provide similar support as the REST architecture, however, special features in the OROA can significantly help the smart building in different aspects as shown below.

More Accurate System States Estimation

One of the biggest issues in managing commercial HVAC (Heating, Ventilation, and Air Conditioning) systems in the building is the unclear errors [13]. A very common scene is happening probably every day when you switch on a controller of a light, and the light is not turned on successfully. In this case, you will usually try a few times, and if it is still not working, you will report the malfunction to the building operators.

Now if the building is developed as the smart building, all the HVAC equipment can be controlled from digital messages. If the REST architecture style is to develop such systems, all equipment will be modelled as resources providing the unified interface of "GET", "POST", "PUT" and "DELETE" to operate the resources. The client can call the provided services via HTTP or CoAP.

Based on the REST architecture, the most common method to turn on a light is to send a "PUT" request to the light resource. If the client gets the response of "200 OK", the operation is assumed successful. The problem of this approach is analysed in more details in Chapter 4. The successfully triggered controller cannot guarantee the successfully running actuators.

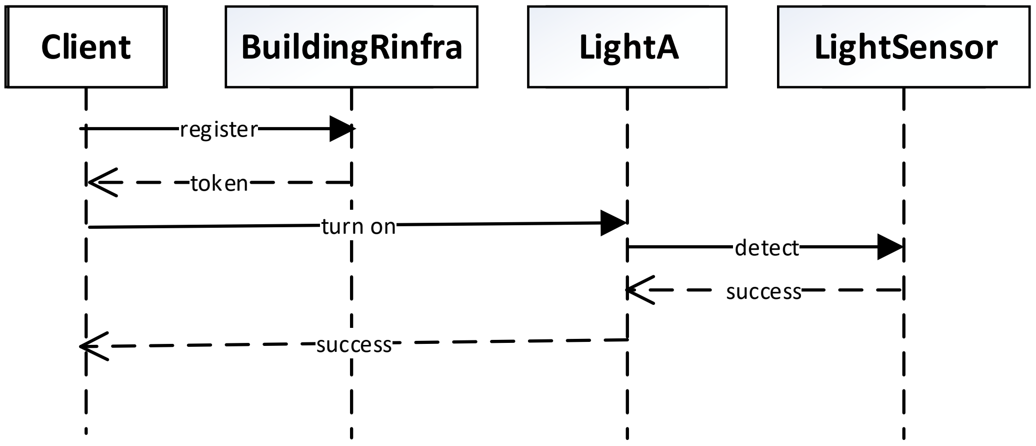


Figure 8.14: The Sequence Diagram to Turn On a Light in the Building

On the other hand, in the OROA, because the Feedback-based Adaptive Service Paradigm is used, the system can provide better system states estimation. Figure 8.14 illustrates the process of turning on a light with OROA and Figure 8.15 illustrates the process of turning on an air conditioner to a given temperature with OROA. Therefore, in our approach, most of the faults in the actuators themselves or in the networks between the controllers and actuators can be detected from each request to the actuators.

Therefore with the support of Feedback-based Adaptive Service Paradigm in the OROA, more accurate system state estimations can be obtained, which can eventually benefit the

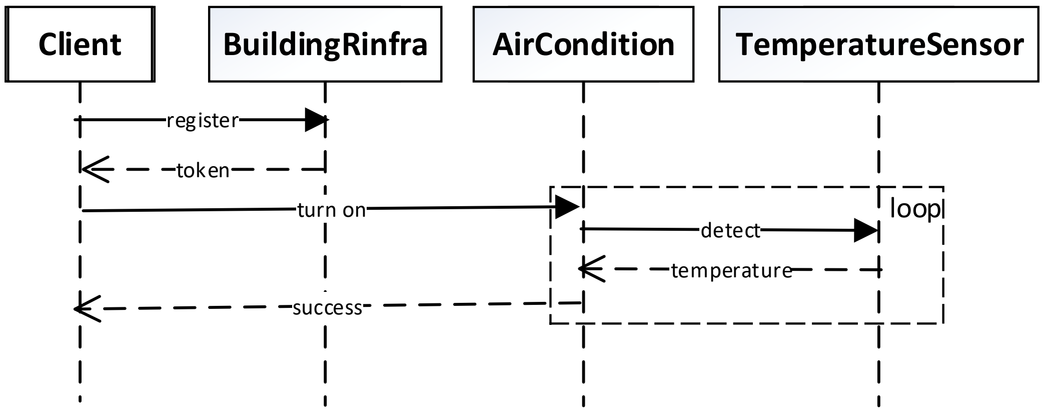


Figure 8.15: The Sequence Diagram to Turn On an Air-conditioner to a Given Temperature with OROA

developing and maintaining of the smart buildings.

Self Faults Detection

In the smart building scenarios, there are many sensors to monitor the environmental conditions such as the temperature and humidity. Collecting the correct data from the sensors is extremely important, because many high-level services and applications are running based on the collected data.

In the REST architecture, the data collected from the sensors are usually assumed to be correct. However, even if the device noises from the sensors are ignored, there are some damage risks in the sensors running in the physical environments.

For this issue, the feature of Semantic-based Reputation Framework in OROA is very useful by evaluating the resources' belief. The detailed mechanism is explained in details in Chapter 6 and here we use a sample to show how this mechanism can be used in the smart building.

The Figure 8.16 indicates part of the reputation framework in the smart building. Assume the room has three temperature sensors to detect the correct temperature, because these three sensors are all cheap yet inaccurate and unreliable, the output of the room

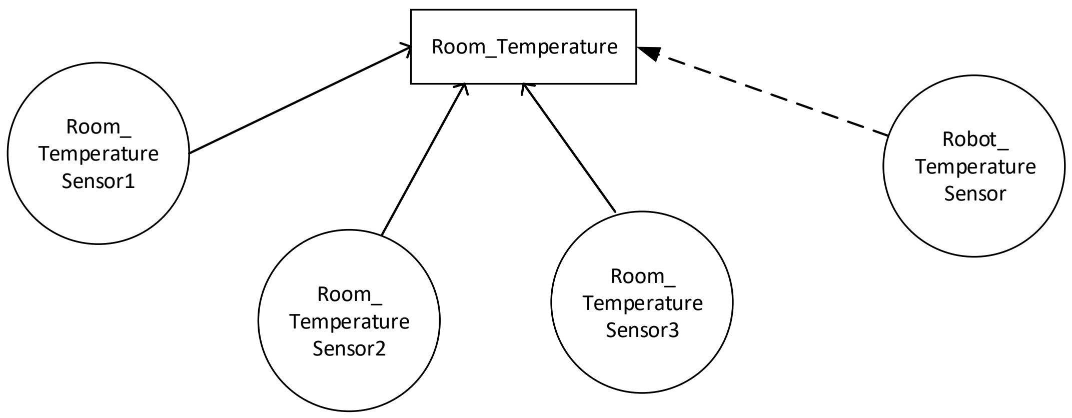

Room_Temperature
Room_ Temperature Sensor1
Room_ Temperature Sensor2
Room_ Temperature Sensor3
Robot_ Temperature Sensor

Figure 8.16: The Temporal Reputation Network When the Robot is Cleaning the Room

temperature needs data fusion from the different sensors. At the same time, a cleaning robot is responsible for cleaning the building, and it has a more accurate, reliable and expensive temperature sensor.

Every day, when the robot enters the room to clean, it will join the reputation network in there and the temporal network is expressed in Figure 8.16. The robot reports the data from its temperature sensor to the room temperature resource, and the room temperature resource will update the robot's temperature sensor's belief value as shown in Figure 8.17.

From the above example, we can conclude the following: the proposed OROA approach helps to build a more reliable decentralised CPS from cheaper devices thanks to the proposed dynamic semantic-based reputation framework. With the support from this framework, the cheaper devices can produce better output via data fusion. In addition, we can use more expensive devices to improve the cheaper devices' performance. Furthermore, in our dynamic semantic-based reputation framework, if any device in the reputation networks is broken, the belief value will be dropped down quickly, therefore the fault can be detected rapidly, and the faults can be dealt with accordingly.

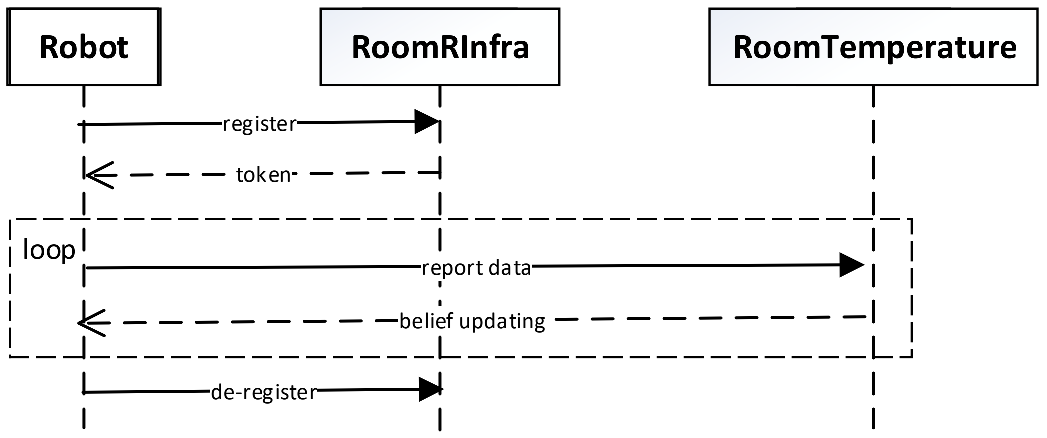


Figure 8.17: How to Construct the Temporal Reputation Framework in the Smart Building

Chapter 9

Conclusion

9.1 Summary

This thesis proposes the Open Resource Oriented Architecture that aims to overcome issues arising when using the resource-oriented architecture in the decentralised CPS. In the decentralised CPS, the networking plays an important role, therefore we have to consider the effects of the networking during the design and development, which makes the traditional controller-based perspective more difficult. The proposed architecture in this thesis is a network-centric approach extended from the REST architecture style.

REST architectural style has many advantages such as low entry barrier, extensibility, anarchic scalability and independent deployment. However, the REST architectural style has some inherent limitations for designing and developing CPS applications. In Chapter 3, the issues of applying REST architectural style into the CPS are discussed and the overview of the Open Resource Oriented Architecture is expressed. The four important additional required functionalities in developing the decentralized CPS when using Resource Oriented Architecture include *Structural Abstract*, *Behavioural Abstract*, *Uncertainty Handling* and *Access Control*.

For each of these required features, we designed the approach respectively. These approaches are eventually integrated together with the proposed Open Resource Oriented Architecture, which is the extended REST with the additional constraints to provide comprehensive support for designing and developing the decentralised CPS. Structural Abstract is briefly introduced in Chapter 3. Behavioural Abstract contains two solutions with the same model of Feedback-based Context-Adaptation. One is the paradigm proposed in Chapter 4 with weaker constraints. Another is the proposed extended protocol explained in Chapter 5 with stronger constraints. In Chapter 6, the proposed approach for uncertainty handling is a self-adaptive reputation framework. The proposed reputation framework can be constructed by Semantic Match which uses resource URIs to map different system components. Chapter 7 explains the access control mechanism for complex usage policies in the decentralised CPS. To accommodate the different requirements and different approaches, Semantic Web technologies are used to implement the proposed architecture. The implementation of the OROA is the RInfra framework and a resource ontology is proposed as a resource template to describe the registered resources.

The RInfra framework provides the decentralised resource infrastructures to develop, configure and deploy the high-level decentralised CPS applications. A smart car based on the Raspberry Pi is built to simulate the smart transport scenes and some scenes in the smart building are discussed and designed based on the OROA and RInfra. The different case studies illustrate how to use the OROA and RInfra in the decentralised cross-domain CPS applications.

Compared to the REST architecture style, the proposed Open Resource Oriented Architecture added more features in abstract ability, uncertainty handling and usage policies to fit the decentralised CPS. Meanwhile, compared to the other massive solutions for the CPS, the proposed architecture is more flexible to support heterogeneous, scalable and interoperable systems. Furthermore, most technologies used in the Open Resource Oriented Architecture and RInfra are common, and have been widely used, therefore it is easier to promote our comprehensive solution.

9.2 Future Work

While we provide the RInfra framework as comprehensive support for developing and designing the decentralised CPS, it is still at the early stage. Therefore, further works can be conducted in the near future.

Firstly, the extended CoAP only contains the basic context adaptation part. To support the real-time feature, we need the QoS support from the communication layer.

In the reputation framework, the current semantics is expressed from the resource URI, which consists of weak expression. Since semantic web technologies are already used in the RInfra, it is possible to develop the reasoning engine to let the reputation framework become self-organizable.

The current RInfra is still a simple prototype without fully exploring the functionalities of Semantic Web technologies. Currently, Jena is used as a graph database, however, the Semantic Web technologies are considered to be used to support inferring and reasoning.

Finally, the current prototype is simple and only implemented in the small car based on Raspberry Pi and other simple web services to simulate some scenes. In the future, the RInfra should be deployed in some real-world applications.

Bibliography

[1] Ramadan Abdunabi, Mustafa Al-Lail, Indrakshi Ray, and Robert B France. Specification, validation, and enforcement of a generalized spatio-temporal role-based access control model. *IEEE Systems Journal*, 7(3):501–515, 2013.

[2] Yuvraj Agarwal, Bharathan Balaji, Rajesh Gupta, Jacob Lyles, Michael Wei, and Thomas Weng. Occupancy-driven energy management for smart building automation. In *Proceedings of the 2nd ACM workshop on embedded sensing systems for energy-efficiency in building*, pages 1–6. ACM, 2010.

[3] Gul A Agha. Actors: A model of concurrent computation in distributed systems. Technical report, MASSACHUSETTS INST OF TECH CAMBRIDGE ARTIFICIAL INTELLIGENCE LAB, 1985.

[4] Gul A Agha, Ian A Mason, Scott F Smith, and Carolyn L Talcott. A foundation for actor computation. *Journal of Functional Programming*, 7(1):1–72, 1997.

[5] Rima Al Ali, Tomas Bures, Ilias Gerostathopoulos, Jaroslav Keznikl, and Frantisek Plasil. Architecture adaptation based on belief inaccuracy estimation. In *Proceedings of the 2014 IEEE/IFIP Conference on Software Architecture*, WICSA '14, pages 87–90, Washington, DC, USA, 2014. IEEE Computer Society.

[6] Rajeev Alur and David L Dill. A theory of timed automata. *Theoretical computer science*, 126(2):183–235, 1994.

[7] Gregory R Andrews. Paradigms for process interaction in distributed programs. *ACM Computing Surveys (CSUR)*, 23(1):49–90, 1991.

[8] Karl Johan Åström and Tore Hägglund. *Advanced PID control*. ISA-The Instrumentation, Systems and Automation Society, 2006.

[9] Jos C. M. Baeten and Jan A. Bergstra. Real time process algebra. *Formal Aspects of Computing*, 3(2):142–188, 1991.

[10] Maryam Bagheri, Ilge Akkaya, Ehsan Khamespanah, Narges Khakpour, Marjan Sirjani, Ali Movaghar, and Edward A Lee. Coordinated actors for reliable self-adaptive systems. In *International Workshop on Formal Aspects of Component Software*, pages 241–259. Springer, 2016.

[11] Guangdong Bai, Lin Yan, Liang Gu, Yao Guo, and Xiangqun Chen. Context-aware usage control for web of things. *Security and Communication Networks*, 7(12):2696–2712, 2014.

[12] Garvita Bajaj, Rachit Agarwal, Pushpendra Singh, Nikolaos Georgantas, and Valerie Issarny. A study of existing ontologies in the iot-domain. *arXiv preprint arXiv:1707.00112*, 2017.

[13] Bharathan Balaji, Nadir Weibel, and Yuvraj Agarwal. Managing commercial hvac systems: What do building operators really need? *arXiv preprint arXiv:1612.06025*, 2016.

[14] Bharathan Balaji, Jian Xu, Anthony Nwokafor, Rajesh Gupta, and Yuvraj Agarwal. Sentinel: occupancy based hvac actuation using existing wifi infrastructure within commercial buildings. In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*, page 17. ACM, 2013.

[15] Linden J Ball, Balder Onarheim, and Bo T Christensen. Design requirements, epistemic uncertainty and solution development strategies in software design. *Design Studies*, 31(6):567–589, 2010.

[16] Fenye Bao and Ing-Ray Chen. Dynamic trust management for internet of things applications. In *Proceedings of the 2012 international workshop on Self-aware internet of things*, pages 1–6. ACM, 2012.

[17] Fenye Bao, Ray Chen, and Jia Guo. Scalable, adaptive and survivable trust management for community of interest based internet of things systems. In *Autonomous Decentralized Systems (ISADS), 2013 IEEE Eleventh International Symposium on*, pages 1–7. IEEE, 2013.

[18] José Barbosa, Paulo Leitão, Emmanuel Adam, and Damien Trentesaux. Dynamic self-organization in holonic multi-agent manufacturing systems: The adacor evolution. *Computers in industry*, 66:99–111, 2015.

[19] Jan A Bergstra and Jan Willem Klop. Process algebra for synchronous communication. *Information and control*, 60(1-3):109–137, 1984.

[20] Carsten Bormann, Angelo P Castellani, and Zach Shelby. Coap: An application protocol for billions of tiny internet nodes. *IEEE Internet Computing*, 16(2):62–67, 2012.

[21] Yuriy Brun, Giovanna Di Marzo Serugendo, Cristina Gacek, Holger Giese, Holger Kienle, Marin Litoiu, Hausi Müller, Mauro Pezzè, and Mary Shaw. Engineering self-adaptive systems through feedback loops. In *Software engineering for self-adaptive systems*, pages 48–70. Springer, 2009.

[22] Antonio Bucchiarone, Martina De Sanctis, Annapaola Marconi, Marco Pistore, and Paolo Traverso. Design for adaptation of distributed service-based systems. In *International Conference on Service-Oriented Computing*, pages 383–393. Springer, 2015.

[23] Tomas Bures, Ilias Gerostathopoulos, Petr Hnetynka, Jaroslav Keznikl, Michal Kit, and Frantisek Plasil. Deeco: an ensemble-based component system. In *Proceedings of the 16th International ACM Sigsoft symposium on Component-based software engineering*, pages 81–90. ACM, 2013.

[24] Tomas Bures, Petr Hnetynka, and Frantisek Plasil. Strengthening architectures of smart cps by modeling them as runtime product-lines. In *Proceedings of the 17th international ACM Sigsoft symposium on Component-based software engineering*, pages 91–96. ACM, 2014.

[25] Stefan Bussmann. An agent-oriented architecture for holonic manufacturing control. In *Proceedings of first international workshop on IMS, Lausanne, Switzerland*, pages 1–12, 1998.

[26] Javier Camara, Carlos Canal, and Gwen Salaün. Behavioural self-adaptation of services in ubiquitous computing environments. *SEAMS*, 9:28–37, 2009.

[27] Javier Cámara, Antónia Lopes, David Garlan, and Bradley Schmerl. Adaptation impact and environment models for architecture-based self-adaptive systems. *Science of Computer Programming*, 127:50–75, 2016.

[28] Mauro Caporuscio, Pierre-Guillaume Raverdy, and Valerie Issarny. ubisoap: A service-oriented middleware for ubiquitous networking. *IEEE Transactions on Services Computing*, 5(1):86–98, 2012.

[29] Rommel Carvalho, Kathryn Laskey, Paulo Costa, Marcelo Ladeira, Laécio Santos, and Shou Matsumoto. Unbbayes: modeling uncertainty for plausible reasoning in the semantic web. In *Semantic Web*. InTech, 2010.

[30] Rommel N Carvalho, Kathryn B Laskey, and Paulo CG Da Costa. Uncertainty modeling process for semantic technology. *PeerJ Computer Science*, 2:e77, 2016.

[31] Francesco Casella, Filippo Donida, and Marco Lovera. Beyond simulation: Computer aided control system design using equation-based object oriented modelling for the next decade. In *Proceedings of the 2nd International Workshop on Equation-Based Object-Oriented Languages and Tools*, number 029, pages 35–45. Linköping University Electronic Press, 2008.

[32] Harry Chen, Tim Finin, Anupam Joshi, Lalana Kagal, Filip Perich, and Dipanjan Chakraborty. Intelligent agents meet the semantic web in smart spaces. *IEEE Internet Computing*, 8(6):69–79, 2004.

[33] Liang Chen and Jason Crampton. Risk-aware role-based access control. In *International Workshop on Security and Trust Management*, pages 140–156. Springer, 2011.

[34] Andrei Ciortea, Olivier Boissier, Antoine Zimmermann, and Adina Magda Florea. Give agents some rest: A resource-oriented abstraction layer for internet-scale agent environments. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, pages 1502–1504. International Foundation for Autonomous Agents and Multiagent Systems, 2017.

[35] Lorenzo Cirio, Isabel F Cruz, and Roberto Tamassia. A role and attribute based access control system using semantic web technologies. In *Proceedings of the 2007*

OTM Confederated international conference on On the move to meaningful internet systems-Volume Part II, pages 1256–1266. Springer-Verlag, 2007.

[36] T. L. Crenshaw, E. Gunter, C. L. Robinson, L. Sha, and P. R. Kumar. The simplex reference model: Limiting fault-propagation due to unreliable components in cyber-physical system architectures. In *28th IEEE International Real-Time Systems Symposium (RTSS 2007)*, pages 400–412, Dec 2007.

[37] Javier Cubo, Carlos Canal, and Ernesto Pimentel. Model-based dependable composition of self-adaptive systems. *Informatica*, 35:51–62, 2011.

[38] Rogério De Lemos, Holger Giese, Hausi A Müller, Mary Shaw, Jesper Andersson, Marin Litoiu, Bradley Schmerl, Gabriel Tamura, Norha M Villegas, Thomas Vogel, et al. Software engineering for self-adaptive systems: A second research roadmap. In *Software Engineering for Self-Adaptive Systems II*, pages 1–32. Springer, 2013.

[39] Hongmei Deng, Yi Yang, Guang Jin, Roger Xu, and Weisong Shi. Building a trust-aware dynamic routing solution for wireless sensor networks. In *GLOBECOM Workshops (GC Wkshps), 2010 IEEE*, pages 153–157. IEEE, 2010.

[40] Patricia Derler, Thomas H Feng, Edward A Lee, Slobodan Matic, Hiren D Patel, Yang Zheo, and Jia Zou. Ptides: A programming model for distributed real-time embedded systems. Technical report, CALIFORNIA UNIV BERKELEY DEPT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE, 2008.

[41] Patricia Derler, Edward A Lee, and Alberto Sangiovanni Vincentelli. Modeling cyber–physical systems. *Proceedings of the IEEE*, 100(1):13–28, 2012.

[42] Beniamino Di Martino, Antonio Esposito, Salvatore Augusto Maisto, and Stefania Nacchia. A semantic iot framework to support restful devices' api interoperability. In *Networking, Sensing and Control (ICNSC), 2017 IEEE 14th International Conference on*, pages 78–83. IEEE, 2017.

[43] Ngoc-Thanh Dinh and Younghan Kim. Restful architecture of wireless sensor network for building management system. *KSII Transactions on Internet and Information Systems (TIIS)*, 6(1):46–63, 2012.

[44] WM Dong and FS Wong. Fuzzy weighted averages and implementation of the extension principle. *Fuzzy sets and systems*, 21(2):183–199, 1987.

[45] Yuji Dong and Kaiyu Wan. Reputation-based framework with semantic match for the internet of things. *The International Conference on Recent Advancements in Computing, IoT and Computer Engineering Technology*, 2017.

[46] Yuji Dong and Kaiyu Wan. Semantic-based reputation framework for the internet of things. *Journal of Universal Computer Science*, 2018. accepted.

[47] Yuji Dong, Kaiyu Wan, Xin Huang, and Yong Yue. Contexts-states-aware access control for internet of things. In *Computer Supported Cooperative Work in Design (CSCWD), 2018 IEEE 22st International Conference on*. IEEE, 2018.

[48] Yuji Dong, Kaiyu Wan, and Yong Yue. A dynamic resource supply model towards cyber physical system (cps). In *Computer, Consumer and Control (IS3C), 2014 International Symposium on*, pages 183–186. IEEE, 2014.

[49] Yuji Dong, Kaiyu Wan, and Yong Yue. Unified dynamic resource supply model to support cyber physical system. In *Proceedings of the International MultiConference of Engineers and Computer Scientists*, volume 2, 2014.

[50] Yuji Dong, Kaiyu Wan, and Yong Yue. A feedback-based adaptive service-oriented paradigm for the internet of things. In *International Conference on Service-Oriented Computing*. Springer, 2017.

[51] Varick L Erickson, Miguel Á Carreira-Perpiñán, and Alberto E Cerpa. Observe: Occupancy-based system for efficient reduction of hvac energy. In *Information Processing in Sensor Networks (IPSN), 2011 10th International Conference on*, pages 258–269. IEEE, 2011.

[52] R. Esteller-Curto, E. Cervera, A. P. del Pobil, and R. Marin. Proposal of a rest-based architecture server to control a robot. In *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2012 Sixth International Conference on*, pages 708–710, July 2012.

[53] Orestis Evangelatos, Kasun Samarasinghe, and Jose Rolim. Evaluating design approaches for smart building systems. In *Mobile Adhoc and Sensor Systems (MASS), 2012 IEEE 9th International Conference on*, pages 1–7. IEEE, 2012.

[54] Orestis Evangelatos, Kasun Samarasinghe, and Jose Rolim. Syndesi: A framework for creating personalized smart environments using wireless sensor networks. In *Distributed Computing in Sensor Systems (DCOSS), 2013 IEEE International Conference on*, pages 325–330. IEEE, 2013.

[55] Renjian Feng, Xiaofeng Xu, Xiang Zhou, and Jiangwen Wan. A trust evaluation algorithm for wireless sensor networks based on node behaviors and ds evidence theory. *Sensors*, 11(2):1345–1360, 2011.

[56] David F Ferraiolo, Ravi Sandhu, Serban Gavrila, D Richard Kuhn, and Ramaswamy Chandramouli. Proposed nist standard for role-based access control. *ACM Transactions on Information and System Security (TISSEC)*, 4(3):224–274, 2001.

[57] Roy Fielding, Jim Gettys, Jeffrey Mogul, Henrik Frystyk, Larry Masinter, Paul Leach, and Tim Berners-Lee. Hypertext transfer protocol–http/1.1. Technical report, 1999.

[58] Roy Thomas Fielding. *Architectural styles and the design of network-based software architectures*. PhD thesis, University of California, Irvine, 2000.

[59] Francois Fouquet, Brice Morin, Franck Fleurey, Olivier Barais, Noel Plouzeau, and Jean-Marc Jezequel. A dynamic component model for cyber physical systems. In *Proceedings of the 15th ACM SIGSOFT symposium on Component Based Software Engineering*, pages 135–144. ACM, 2012.

[60] Saurabh Ganeriwal, Laura K Balzano, and Mani B Srivastava. Reputation-based framework for high integrity sensor networks. *ACM Transactions on Sensor Networks (TOSN)*, 4(3):15, 2008.

[61] David Garlan, Robert T Monroe, and David Wile. Acme: Architectural description of component-based systems. *Foundations of component-based systems*, 68:47–68, 2000.

[62] Khusvinder Gill, Shuang-Hua Yang, Fang Yao, and Xin Lu. A zigbee-based home automation system. *IEEE Transactions on consumer Electronics*, 55(2), 2009.

[63] William I Grosky, Aman Kansal, Suman Nath, Jie Liu, and Feng Zhao. Senseweb: An infrastructure for shared sensing. *IEEE multimedia*, 14(4), 2007.

[64] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami. Internet of things (iot): A vision, architectural elements, and future directions. *Future generation computer systems*, 29(7):1645–1660, 2013.

[65] D. Guinard, V. Trifa, and E. Wilde. A resource oriented architecture for the web of things. In *Internet of Things (IOT), 2010*, pages 1–8, Nov 2010.

[66] Dominique Guinard, Iulia Ion, and Simon Mayer. In search of an internet of things service architecture: Rest or ws-*? a developers perspective. In *International Conference on Mobile and Ubiquitous Systems: Computing, Networking, and Services*, pages 326–337. Springer, 2011.

[67] Dominique Guinard and Vlad Trifa. Towards the web of things: Web mashups for embedded devices. In *Workshop on Mashups, Enterprise Mashups and Lightweight Composition on the Web (MEM 2009), in proceedings of WWW (International World Wide Web Conferences), Madrid, Spain*, volume 15, 2009.

[68] Sandeep KS Gupta, Tridib Mukherjee, and Krishna Venkatasubramanian. Criticality aware access control model for pervasive applications. In *Pervasive Computing and Communications, 2006. PerCom 2006. Fourth Annual IEEE International Conference on*, pages 5–pp. IEEE, 2006.

[69] Sergio Gusmeroli, Salvatore Piccione, and Domenico Rotondi. A capability-based security approach to manage access control in the internet of things. *Mathematical and Computer Modelling*, 58(5):1189–1205, 2013.

[70] Amelie Gyrard, Soumya Kanti Datta, Christian Bonnet, and Karima Boudaoud. Cross-domain internet of things application development: M3 framework and evaluation. In *Future Internet of Things and Cloud (FiCloud), 2015 3rd International Conference on*, pages 9–16. IEEE, 2015.

[71] Marc Hadley and Paul Sandoz. Jax-rs: Java api for restful web services. *Java Specification Request (JSR)*, 311, 2009.

[72] David L Hall and James Llinas. An introduction to multisensor data fusion. *Proceedings of the IEEE*, 85(1):6–23, 1997.

[73] Klaus Hartke. Observing resources in the constrained application protocol (coap). 2015.

[74] Wu He, Gongjun Yan, and Li Da Xu. Developing vehicular data cloud services in the iot environment. *IEEE Transactions on Industrial Informatics*, 10(2):1587–1595, 2014.

[75] Zhengqiu He, Lifa Wu, Huabo Li, Haiguang Lai, and Zheng Hong. Semantics-based access control approach for web service. *Journal of Computers*, 6(6):1152–1161, 2011.

[76] Thomas A Henzinger. The theory of hybrid automata. In *Verification of Digital and Hybrid Systems*, pages 265–292. Springer, 2000.

[77] Jose L Hernandez-Ramos, Antonio J Jara, Leandro Marın, and Antonio F Skarmeta. Distributed capability-based access control for the internet of things. *Journal of Internet Services and Information Security (JISIS)*, 3(3/4):1–16, 2013.

[78] Jan Holler, Vlasios Tsiatsis, Catherine Mulligan, Stefan Avesand, Stamatis Karnouskos, and David Boyle. *From Machine-to-machine to the Internet of Things: Introduction to a New Age of Intelligence.* Academic Press, 2014.

[79] Biqing Huang, Chenghai Li, Chao Yin, and Xinpei Zhao. Cloud manufacturing service platform for small-and medium-sized enterprises. *The International Journal of Advanced Manufacturing Technology*, 65(9-12):1261–1272, 2013.

[80] Jian Huang, F Bastani, I-L Yen, Jing Dong, Wenke Zhang, F-J Wang, and H-J Hsu. Extending service model to build an effective service composition framework for cyber-physical systems. In *Service-Oriented Computing and Applications (SOCA), 2009 IEEE International Conference on*, pages 1–8. IEEE, 2009.

[81] Jian Huang, Farokh Bastani, I-Ling Yen, and Jun-Jang Jeng. Toward a smart cyber-physical space: A context-sensitive resource-explicit service model. In *Computer Software and Applications Conference, 2009. COMPSAC'09. 33rd Annual IEEE International*, volume 2, pages 122–127. IEEE, 2009.

[82] Xin Huang, Rong Fu, Bangdao Chen, Tingting Zhang, and AW Roscoe. User interactive internet of things privacy preserved access control. In *Internet Technology And Secured Transactions, 2012 International Conference for*, pages 597–602. IEEE, 2012.

[83] Junbeom Hur and Dong Kun Noh. Attribute-based access control with efficient revocation in data outsourcing systems. *IEEE Transactions on Parallel and Distributed Systems*, 22(7):1214–1221, 2011.

[84] LA Hurtado, PH Nguyen, WL Kling, and W Zeiler. Building energy management systemsoptimization of comfort and energy use. In *Power Engineering Conference (UPEC), 2013 48th International Universities'*, pages 1–6. IEEE, 2013.

[85] Marija D Ilic, Le Xie, Usman A Khan, and José MF Moura. Modeling of future cyber–physical energy systems for distributed sensing and control. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 40(4):825–838, 2010.

[86] Upul Jayasinghe, Hyun-Woo Lee, and Gyu Myoung Lee. A computational model to evaluate honesty in social internet of things. In *Proceedings of the Symposium on Applied Computing*, pages 1830–1835, 2017.

[87] Magne Jorgensen. Evidence-based guidelines for assessment of software development cost uncertainty. *IEEE Transactions on Software Engineering*, 31(11):942–954, 2005.

[88] A. D. Joseph, A. R. Beresford, J. Bacon, D. N. Cottingham, J. J. Davies, B. D. Jones, Haitao Guo, Wei Guan, Yong Lin, Houbing Song, L. Iftode, S. Fuchs, B. Lamprecht, K. Kyamakya, J. G. Fernandez, J. C. Yelmo Garcia, Y. S. Martin Garcia, J. de Gracia Santos, M. Nimesh, Gang Pan, Zhaohui Wu, Qing Wu, Zhenyu Shan, Jie Sun, Jian Lu, Guoqing Yang, M. K. Khan, and Jiashu Zhang. Intelligent transportation systems. *IEEE Pervasive Computing*, 5(4):63–67, Oct 2006.

[89] James BD Joshi, Elisa Bertino, Usman Latif, and Arif Ghafoor. A generalized temporal role-based access control model. *IEEE Transactions on Knowledge and Data Engineering*, 17(1):4–23, 2005.

[90] Sun Kaiwen and Yin Lihua. Attribute-role-based hybrid access control in the internet of things. In *Asia-Pacific Web Conference*, pages 333–343. Springer, 2014.

[91] Woochul Kang, Krasimira Kapitanova, and Sang Hyuk Son. Rdds: A real-time data distribution service for cyber-physical systems. *IEEE Transactions on Industrial Informatics*, 8(2):393–405, 2012.

[92] Woochul Kang and Sang H Son. The design of an open data service architecture for cyber-physical systems. *ACM SIGBED Review*, 5(1):3, 2008.

[93] ASM Kayes, Jun Han, and Alan Colman. Ontcaac: an ontology-based approach to context-aware access control for software services. *The Computer Journal*, 58(11):3000–3034, 2015.

[94] Bahador Khaleghi, Alaa Khamis, Fakhreddine O Karray, and Saiedeh N Razavi. Multisensor data fusion: A review of the state-of-the-art. *Information Fusion*, 14(1):28–44, 2013.

[95] Jussi Kiljander, Alfredo Delia, Francesco Morandi, Pasi Hyttinen, Janne Takalo-Mattila, Arto Ylisaukko-Oja, Juha-Pekka Soininen, and Tullio Salmon Cinotti. Semantic interoperability architecture for pervasive computing and internet of things. *IEEE access*, 2:856–873, 2014.

[96] Heejae Kim, Jiyong Han, Seong-Hwan Kim, Jisoo Choi, Dongsik Yoon, Minsu Jeon, Eunjoo Yang, Nhat Pham, Sungpil Woo, Jeongkyu Park, et al. Isv2c: an integrated road traffic-network-cloud simulator for v2c connected car services. In *Services Computing (SCC), 2017 IEEE International Conference on*, pages 434–441. IEEE, 2017.

[97] Sehoon Kim, Jin-Young Hong, Seil Kim, Sung-Hoon Kim, Jun-Hyung Kim, and Jake Chun. Restful design and implementation of smart appliances for smart home. In *Ubiquitous Intelligence and Computing, 2014 IEEE 11th Intl Conf on and IEEE 11th Intl Conf on and Autonomic and Trusted Computing, and IEEE 14th Intl Conf on Scalable Computing and Communications and Its Associated Workshops (UTC-ATC-ScalCom)*, pages 717–722. IEEE, 2014.

[98] Laura Klein, Jun-young Kwak, Geoffrey Kavulya, Farrokh Jazizadeh, Burcin Becerik-Gerber, Pradeep Varakantham, and Milind Tambe. Coordinating occupant behavior for building energy and comfort management using multi-agent systems. *Automation in construction*, 22:525–536, 2012.

[99] Uri Klein and Kedar S Namjoshi. Formalization and automated verification of restful behavior. In *International Conference on Computer Aided Verification*, pages 541–556. Springer, 2011.

[100] Edwin M Knox and Raymond T Ng. Algorithms for mining distancebased outliers in large datasets. In *Proceedings of the International Conference on Very Large Data Bases*, pages 392–403. Citeseer, 1998.

[101] Zvi Kohavi and Niraj K Jha. *Switching and finite automata theory.* Cambridge University Press, 2009.

[102] Alexandr Krylovskiy, Marco Jahn, and Edoardo Patti. Designing a smart city internet of things platform with microservice architecture. In *Future Internet of Things and Cloud (FiCloud), 2015 3rd International Conference on*, pages 25–30. IEEE, 2015.

[103] Koojana Kuladinithi, Olaf Bergmann, Thomas Pötsch, Markus Becker, and Carmelita Görg. Implementation of coap and its application in transport logistics. *Proc. IP+ SN, Chicago, IL, USA*, 2011.

[104] Devdatta Kulkarni and Anand Tripathi. Context-aware role-based access control in pervasive computing systems. In *Proceedings of the 13th ACM symposium on Access control models and technologies*, pages 113–122. ACM, 2008.

[105] H. J. La and S. D. Kim. A service-based approach to designing cyber physical systems. In *2010 IEEE/ACIS 9th International Conference on Computer and Information Science*, pages 895–900, Aug 2010.

[106] Amos Lapidoth and Prakash Narayan. Reliable communication under channel uncertainty. *IEEE Transactions on Information Theory*, 44(6):2148–2177, 1998.

[107] Ora Lassila and Ralph R Swick. Resource description framework (rdf) model and syntax specification. 1999.

[108] Zsolt Lattmann, Adam Nagel, Tihamer Levendovszky, Ted Bapty, Sandeep Neema, and Gabor Karsai. Component-based modeling of dynamic systems using heterogeneous composition. In *Proceedings of the 6th International Workshop on Multi-Paradigm Modeling*, pages 73–78. ACM, 2012.

[109] Danh Le-Phuoc and Manfred Hauswirth. Linked open data in sensor data mashups. In *Proceedings of the 2nd International Conference on Semantic Sensor Networks-Volume 522*, pages 1–16. CEUR-WS. org, 2009.

[110] Danh Le-Phuoc, Hoan Quoc Nguyen-Mau, Josiane Xavier Parreira, and Manfred Hauswirth. A middleware framework for scalable management of linked streams. *Web Semantics: Science, Services and Agents on the World Wide Web*, 16:42–51, 2012.

[111] Chuen-Chien Lee. Fuzzy logic in control systems: fuzzy logic controller. i. *IEEE Transactions on systems, man, and cybernetics*, 20(2):404–418, 1990.

[112] Edward A Lee. Cyber physical systems: Design challenges. In *Object oriented real-time distributed computing (isorc), 2008 11th ieee international symposium on*, pages 363–369. IEEE, 2008.

[113] Edward A Lee. Computing needs time. *Communications of the ACM*, 52(5):70–79, 2009.

[114] Edward A Lee. Heterogeneous actor modeling. In *Proceedings of the ninth ACM international conference on Embedded software*, pages 3–12. ACM, 2011.

[115] Jeong Kyu Lee, Young Sik Jeong, and Jong Hyuk Park. s-itsf: a service based intelligent transportation system framework for smart accident management. *Human-centric Computing and Information Sciences*, 5(1):34, 2015.

[116] Wilfried Lepuschitz, Alois Zoitl, Mathieu Vallée, and Munir Merdan. Toward self-reconfiguration of manufacturing systems using automation agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 41(1):52–69, 2011.

[117] Qiang Li, Weijun Qin, Bing Han, Ruicong Wang, and Limin Sun. A case study on rest-style architecture for cyber-physical systems: Restful smart gateway. *Comput. Sci. Inf. Syst.*, 8:1317–1329, 2011.

[118] Jing Lin, Sahra Sedigh, and Ann Miller. Modeling cyber-physical systems with semantic agents. In *Computer Software and Applications Conference Workshops (COMPSACW), 2010 IEEE 34th Annual*, pages 13–18. IEEE, 2010.

[119] Kwei-Jay Lin and Mark Panahi. A real-time service-oriented framework to support sustainable cyber-physical systems. In *Industrial Informatics (INDIN), 2010 8th IEEE International Conference on*, pages 15–21. IEEE, 2010.

[120] Jianqi Liu, Jiafu Wan, Bi Zeng, Qinruo Wang, Houbing Song, and Meikang Qiu. A scalable and quick-response software defined vehicular network assisted by mobile edge computing. *IEEE Communications Magazine*, 55(7):94–100, 2017.

[121] Samuel Madden, Michael J Franklin, Joseph M Hellerstein, and Wei Hong. Tag: A tiny aggregation service for ad-hoc sensor networks. *ACM SIGOPS Operating Systems Review*, 36(SI):131–146, 2002.

[122] Luca Mainetti, Vincenzo Mighali, and Luigi Patrono. A software architecture enabling the web of things. *IEEE Internet of Things Journal*, 2(6):445–454, 2015.

[123] Luca Mainetti, Vincenzo Mighali, Luigi Patrono, Piercosimo Rametta, and Silvio Lucio Oliva. A novel architecture enabling the visual implementation of web of things applications. In *Software, Telecommunications and Computer Networks (SoftCOM), 2013 21st International Conference on*, pages 1–7. IEEE, 2013.

[124] Ibrahim Mashal, Osama Alsaryrah, Tein-Yaw Chung, Cheng-Zen Yang, Wen-Hsing Kuo, and Dharma P Agrawal. Choices for interaction with things on internet and underlying issues. *Ad Hoc Networks*, 28:68–90, 2015.

[125] Larry Masinter, Tim Berners-Lee, and Roy T Fielding. Uniform resource identifier (uri): Generic syntax. 2005.

[126] Arm Mbed. java-coap. `https://github.com/ARMmbed/java-coap`, 2018.

[127] Kirill Mechitov and Gul Agha. Building portable middleware services for heterogeneous cyber-physical systems. In *Proceedings of the Third International Workshop on Software Engineering for Sensor Network Applications*, pages 31–36. IEEE Press, 2012.

[128] Alessio Meloni and Luigi Atzori. A cloud-based and restful internet of things platform to foster smart grid technologies integration and re-usability. In *Communications Workshops (ICC), 2016 IEEE International Conference on*, pages 387–392. IEEE, 2016.

[129] Julien Mineraud, Oleksiy Mazhelis, Xiang Su, and Sasu Tarkoma. A gap analysis of internet-of-things platforms. *Computer Communications*, 89:5–16, 2016.

[130] Daniele Miorandi, Sabrina Sicari, Francesco De Pellegrini, and Imrich Chlamtac. Internet of things: Vision, applications and research challenges. *Ad Hoc Networks*, 10(7):1497–1516, 2012.

[131] Tim Moses et al. Extensible access control markup language (xacml) version 2.0. *Oasis Standard*, 200502, 2005.

[132] Subhas Chandra Mukhopadhyay and NK Suryadevara. Internet of things: Challenges and opportunities. In *Internet of Things*, pages 1–17. Springer, 2014.

[133] Andreas Müller, Stefan Mitsch, Werner Retschitzegger, Wieland Schwinger, and André Platzer. A component-based approach to hybrid systems safety verification. In *International Conference on Integrated Formal Methods*, pages 441–456. Springer, 2016.

[134] Naoya Namatame, Yong Ding, Till Riedel, Hideyuki Tokuda, Takashi Miyaki, and Michael Beigl. A distributed resource management architecture for interconnecting web-of-things using ubox. In *Proceedings of the Second International Workshop on Web of Things*, page 4. ACM, 2011.

[135] National institute of standards and technology. *Foundations for Innovation in Cyber-Physical Systems*, Rosemont, Illinois, January 2013. energetics incorporated Columbia, Maryland.

[136] Michele Nitti, Roberto Girau, and Luigi Atzori. Trustworthiness management in the social internet of things. *IEEE Transactions on knowledge and data engineering*, 26(5):1253–1266, 2014.

[137] Reza Olfati-Saber, J Alex Fax, and Richard M Murray. Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, 95(1):215–233, 2007.

[138] William R Otte, Abhishek Dubey, Subhav Pradhan, Prithviraj Patil, Aniruddha Gokhale, Gabor Karsai, and Johnny Willemsen. F6com: A component model for resource-constrained and dynamic space-based computing environments. In *Object/Component/Service-Oriented Real-Time Distributed Computing (ISORC), 2013 IEEE 16th International Symposium on*, pages 1–8. IEEE, 2013.

[139] Aafaf Ouaddah, Hajar Mousannif, Anas Abou Elkalam, and Abdellah Ait Ouahman. Access control in the internet of things: Big challenges and new opportunities. *Computer Networks*, 112:237–262, 2017.

[140] Federica Paganelli, Stefano Turchi, and Dino Giuli. A web of things framework for restful applications and its experimentation in a smart city. *IEEE Systems Journal*, 10(4):1412–1423, 2016.

[141] Spiros Papadimitriou, Hiroyuki Kitagawa, Phillip B Gibbons, and Christos Faloutsos. Loci: Fast outlier detection using the local correlation integral. In *Data Engineering, 2003. Proceedings. 19th International Conference on*, pages 315–326. IEEE, 2003.

[142] Jaehong Park and Ravi Sandhu. Towards usage control models: beyond traditional access control. In *Proceedings of the seventh ACM symposium on Access control models and technologies*, pages 57–64. ACM, 2002.

[143] Cesare Pautasso, Olaf Zimmermann, and Frank Leymann. Restful web services vs. big'web services: making the right architectural decision. In *Proceedings of the 17th international conference on World Wide Web*, pages 805–814. ACM, 2008.

[144] Huanchun Peng, Jun Gu, and Xiaojun Ye. Dynamic purpose-based access control. In *Parallel and Distributed Processing with Applications, 2008. ISPA'08. International Symposium on*, pages 695–700. IEEE, 2008.

[145] James L Peterson. Petri net theory and the modeling of systems. 1981.

[146] Dennis Pfisterer, Kay Romer, Daniel Bimschas, Oliver Kleine, Richard Mietz, Cuong Truong, Henning Hasemann, Alexander Kröller, Max Pagel, Manfred Hauswirth, et al. Spitfire: toward a semantic web of things. *IEEE Communications Magazine*, 49(11):40–48, 2011.

[147] Henrique Brittes Pötter and Alexandre Sztajnberg. Adapting heterogeneous devices into an iot context-aware infrastructure. In *Software Engineering for Adaptive and Self-Managing Systems (SEAMS), 2016 IEEE/ACM 11th International Symposium on*, pages 64–74. IEEE, 2016.

[148] Christian Prehofer. Models at rest or modelling restful interfaces for the internet of things. In *Internet of Things (WF-IoT), 2015 IEEE 2nd World Forum on*, pages 251–255. IEEE, 2015.

[149] Torsten Priebe, Wolfgang Dobmeier, Christian Schläger, and Nora Kamprath. Supporting attribute-based access control in authentication and authorization infrastructures with ontologies. *Journal of software: JSW*, 2(1):27–38, 2007.

[150] Weijun Qin, Qiang Li, Limin Sun, Hongsong Zhu, and Yan Liu. Restthing: A restful web service infrastructure for mash-up physical and web resources. In *Embedded and Ubiquitous Computing (EUC), 2011 IFIP 9th International Conference on*, pages 197–204. IEEE, 2011.

[151] Akshay Rajhans, Ajinkya Bhave, Ivan Ruchkin, Bruce H Krogh, David Garlan, André Platzer, and Bradley Schmerl. Supporting heterogeneity in cyber-physical systems architectures. *IEEE Transactions on Automatic Control*, 59(12):3178–3193, 2014.

[152] Akshay Rajhans, Shang-Wen Cheng, Bradley Schmerl, David Garlan, Bruce H Krogh, Clarence Agbi, and Ajinkya Bhave. An architectural approach to the design and analysis of cyber-physical systems. *Electronic Communications of the EASST*, 21, 2009.

[153] Shahid Raza, Hossein Shafagh, Kasun Hewage, René Hummen, and Thiemo Voigt. Lithe: Lightweight secure coap for the internet of things. *IEEE Sensors Journal*, 13(10):3711–3720, 2013.

[154] Wonwoo Ro, Giyong Park, Sejin Chun, and Kyong-Ho Lee. Complex sensor mashups for linking sensors and formula-based knowledge bases. In *Information Reuse and Integration (IRI), 2015 IEEE International Conference on*, pages 126–133. IEEE, 2015.

[155] Murray Rosenblatt. A central limit theorem and a strong mixing condition. *Proceedings of the National Academy of Sciences*, 42(1):43–47, 1956.

[156] Timothy J Ross. *Fuzzy logic with engineering applications*. John Wiley & Sons, 2009.

[157] Ivan Ruchkin, Bradley Schmerl, and David Garlan. Analytic dependency loops in architectural models of cyber-physical systems. *Joint proceedings of ACES-MB 2015–Model-based Architecting of Cyber-physical and Embedded Systems*, page 3, 2015.

[158] Ivan Ruchkin, Bradley Schmerl, and David Garlan. Architectural abstractions for hybrid programs. In *Proceedings of the 18th International ACM SIGSOFT Symposium on Component-Based Software Engineering*, pages 65–74. ACM, 2015.

[159] Ravi S Sandhu, Edward J Coyne, Hal L Feinstein, and Charles E Youman. Role-based access control models. *Computer*, 29(2):38–47, 1996.

[160] Siva Kumar Sastry Hari, Man-Lap Li, Pradeep Ramachandran, Byn Choi, and Sarita V Adve. mswat: low-cost hardware fault detection and diagnosis for multicore systems. In *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*, pages 122–132. ACM, 2009.

[161] Rüdiger Schollmeier. A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications. In *Peer-to-Peer Computing, 2001. Proceedings. First International Conference on*, pages 101–102. IEEE, 2001.

[162] Lars Schor, Philipp Sommer, and Roger Wattenhofer. Towards a zero-configuration wireless sensor network architecture for smart buildings. In *Proceedings of the First ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings*, pages 31–36. ACM, 2009.

[163] Ronny Seiger, Steffen Huber, Peter Heisig, and Uwe Assmann. Enabling self-adaptive workflows for cyber-physical systems. In *International Workshop on Business Process Modeling, Development and Support*, pages 3–17. Springer, 2016.

[164] Zach Shelby, Klaus Hartke, and Carsten Bormann. The constrained application protocol (coap). Technical report, 2014.

[165] Quan Z. Sheng, Sam Pohlenz, Jian Yu, Hoi S. Wong, Anne H. H. Ngu, and Zakaria Maamar. Contextserv: A platform for rapid and flexible development of context-aware web services. In *Proceedings of the 31st International Conference on Software Engineering*, ICSE '09, pages 619–622, Washington, DC, USA, 2009. IEEE Computer Society.

[166] Z. Sheng, H. Wang, C. Yin, X. Hu, S. Yang, and V. C. M. Leung. Lightweight management of resource-constrained sensor devices in internet of things. *IEEE Internet of Things Journal*, 2(5):402–411, Oct 2015.

[167] Thanos G Stavropoulos, Dimitris Vrakas, Danai Vlachava, and Nick Bassiliades. Bonsai: a smart building ontology for ambient intelligence. In *Proceedings of the 2nd international conference on web intelligence, mining and semantics*, page 30. ACM, 2012.

[168] Ivan Stojmenovic. Machine-to-machine communications with in-network data aggregation, processing, and actuation for large-scale cyber-physical systems. *IEEE Internet of Things Journal*, 1(2):122–128, 2014.

[169] Michio Sugeno. An introductory survey of fuzzy control. *Information sciences*, 36(1-2):59–83, 1985.

[170] J. Sztipanovits, X. Koutsoukos, G. Karsai, N. Kottenstette, P. Antsaklis, V. Gupta, B. Goodwine, J. Baras, and S. Wang. Toward a science of cyber-physical system integration. *Proceedings of the IEEE*, 100(1):29–44, Jan 2012.

[171] Janos Sztipanovits, Ted Bapty, Sandeep Neema, Larry Howard, and Ethan Jackson. Openmeta: a model-and component-based design tool chain for cyber-physical systems. In *Joint European Conferences on Theory and Practice of Software*, pages 235–248. Springer, 2014.

[172] Ying Tan, Mehmet C Vuran, and Steve Goddard. Spatio-temporal event model for cyber-physical systems. In *Distributed Computing Systems Workshops, 2009. ICDCS Workshops' 09. 29th IEEE International Conference on*, pages 44–50. IEEE, 2009.

[173] Ying Tan, Mehmet C Vuran, Steve Goddard, Yue Yu, Miao Song, and Shangping Ren. A concept lattice-based event model for cyber-physical systems. In *Proceedings of the 1st ACM/IEEE International Conference on Cyber-physical Systems*, pages 50–60. ACM, 2010.

[174] Giacomo Tanganelli, Carlo Vallati, Enzo Mingozzi, and Matthias Kovatsch. Efficient proxying of coap observe with quality of service support. In *Internet of Things (WF-IoT), 2016 IEEE 3rd World Forum on*, pages 401–406. IEEE, 2016.

[175] Fei Tao and Qinglin Qi. New it driven service-oriented smart manufacturing: framework and characteristics. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2017.

[176] Fei Tao, Ying Zuo, Li Da Xu, and Lin Zhang. Iot-based intelligent perception and access of manufacturing resource toward cloud manufacturing. *IEEE Transactions on Industrial Informatics*, 10(2):1547–1557, 2014.

[177] Nguyen Binh Truong, Hyunwoo Lee, Bob Askwith, and Gyu Myoung Lee. Toward a trust evaluation mechanism in the social internet of things. *Sensors*, 17(6):1346, 2017.

[178] Mark Turner, David Budgen, and Pearl Brereton. Turning software into a service. *Computer*, 36(10):38–44, 2003.

[179] Birgit Vogel-Heuser, Christian Diedrich, Dorothea Pantförder, and Peter Göhner. Coupling heterogeneous production systems by a multi-agent based cyber-physical production system. In *Industrial Informatics (INDIN), 2014 12th IEEE International Conference on*, pages 713–719. IEEE, 2014.

[180] Jiafu Wan, Daqiang Zhang, Yantao Sun, Kai Lin, Caifeng Zou, and Hu Cai. Vcmia: a novel architecture for integrating vehicular cyber-physical systems and mobile cloud computing. *Mobile Networks and Applications*, 19(2):153–160, 2014.

[181] Jiafu Wan, Daqiang Zhang, Shengjie Zhao, Laurence Yang, and Jaime Lloret. Context-aware vehicular cyber-physical systems with cloud support: architecture, challenges, and solutions. *IEEE Communications Magazine*, 52(8):106–113, 2014.

[182] Kaiyu Wan. A brief history of context. In *International Journal of Computer Science Issues, Volume 6, Issue*. Citeseer, 2009.

[183] Kaiyu Wan, Vangalur Alagar, and Yuji Dong. Specifying resource-centric services in cyber physical systems. In *Transactions on Engineering Technologies*, pages 83–97. Springer, 2014.

[184] Kaiyu Wan, Yuji Dong, Qian Chang, and Tengfei Qian. Applying a dynamic resource supply model in a smart grid. *Algorithms*, 7(3):471–491, 2014.

[185] Zhiguo Wan, Jun'e Liu, and Robert H Deng. Hasbe: a hierarchical attribute-based solution for flexible and scalable access control in cloud computing. *IEEE transactions on information forensics and security*, 7(2):743–754, 2012.

[186] Jiacun Wang. Time petri nets. In *Timed Petri Nets*, pages 63–123. Springer, 1998.

[187] Shiyong Wang, Jiafu Wan, Daqiang Zhang, Di Li, and Chunhua Zhang. Towards smart factory for industry 4.0: a self-organized multi-agent system with big data based feedback and coordination. *Computer Networks*, 101:158–168, 2016.

[188] Xi Vincent Wang and Xun W Xu. An interoperable solution for cloud manufacturing. *Robotics and computer-integrated manufacturing*, 29(4):232–247, 2013.

[189] Chuyuan Wei and Yongzhen Li. Design of energy consumption monitoring and energy-saving management system of intelligent building based on the internet of things. In *Electronics, Communications and Control (ICECC), 2011 International Conference on*, pages 3650–3652. IEEE, 2011.

[190] Wang Wei and Payam Barnaghi. *Semantic Annotation and Reasoning for Sensor Data*, pages 66–76. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[191] Xun Xu. From cloud computing to cloud manufacturing. *Robotics and computer-integrated manufacturing*, 28(1):75–86, 2012.

[192] Song Yanwei, Zeng Guangzhou, and Pu Haitao. Research on the context model of intelligent interaction system in the internet of things. In *IT in Medicine and Education (ITME), 2011 International Symposium on*, volume 2, pages 379–382. IEEE, 2011.

[193] Eric Yuan and Jin Tong. Attributed based access control (abac) for web services. In *Web Services, 2005. ICWS 2005. Proceedings. 2005 IEEE International Conference on*. IEEE, 2005.

[194] Ke Yue, Li Wang, Shangping Ren, Xufei Mao, and Xiangyang Li. An adaptive discrete event model for cyber-physical system. In *Analytic Virtual Integration of Cyber-Physical Systems Workshop, USA*, pages 9–15, 2010.

[195] Miao Yun and Bu Yuxin. Research on the architecture and key technology of internet of things (iot) applied on smart grid. In *Advances in Energy Engineering (ICAEE), 2010 International Conference on*, pages 69–72. IEEE, 2010.

[196] Lotfi A Zadeh. Fuzzy sets. *Information and control*, 8(3):338–353, 1965.

[197] Paul A Zandbergen. Accuracy of iphone locations: A comparison of assisted gps, wifi and cellular positioning. *Transactions in GIS*, 13(s1):5–25, 2009.

[198] Andrea Zanella, Nicola Bui, Angelo Castellani, Lorenzo Vangelista, and Michele Zorzi. Internet of things for smart cities. *IEEE Internet of Things journal*, 1(1):22–32, 2014.

[199] Gulnara Zhabelova and Valeriy Vyatkin. Multiagent smart grid automation architecture based on iec 61850/61499 intelligent logical nodes. *IEEE Transactions on Industrial Electronics*, 59(5):2351–2362, 2012.

[200] Gulnara Zhabelova, Valeriy Vyatkin, and Victor N Dubinin. Toward industrially usable agent technology for smart grid automation. *IEEE Transactions on Industrial Electronics*, 62(4):2629–2641, 2015.

[201] Man Zhang, Bran Selic, Shaukat Ali, Tao Yue, Oscar Okariz, and Roland Norgren. Understanding uncertainty in cyber-physical systems: a conceptual model. In *European Conference on Modelling Foundations and Applications*, pages 247–264. Springer, 2016.

[202] Xinwen Zhang, Francesco Parisi-Presicce, Ravi Sandhu, and Jaehong Park. Formal model and policy specification of usage control. *ACM Transactions on Information and System Security (TISSEC)*, 8(4):351–387, 2005.

[203] Yingfeng Zhang, Cheng Qian, Jingxiang Lv, and Ying Liu. Agent and cyber-physical system based self-organizing and self-adaptive intelligent shopfloor. *IEEE Transactions on Industrial Informatics*, 13(2):737–747, 2017.

[204] Kai Zhao and Lina Ge. A survey on the internet of things security. In *Computational Intelligence and Security (CIS), 2013 9th International Conference on*, pages 663–667. IEEE, 2013.

[205] Peng Zhao, Siddharth Suryanarayanan, and Marcelo Godoy Simões. An energy management system for building structures using a multi-agent decision-making control methodology. *IEEE Transactions on Industry Applications*, 49(1):322–330, 2013.

[206] Jiehan Zhou, Teemu Leppanen, Erkki Harjula, Mika Ylianttila, Timo Ojala, Chen Yu, Hai Jin, and Laurence Tianruo Yang. Cloudthings: A common architecture for integrating the internet of things with cloud computing. In *Computer Supported Cooperative Work in Design (CSCWD), 2013 IEEE 17th International Conference on*, pages 651–657. IEEE, 2013.

[207] Wei Zhu, Guang Zhou, I-Ling Yen, and Farokh Bastani. A pt-soa model for cps/iot services. In *Web Services (ICWS), 2015 IEEE International Conference on*, pages 647–654. IEEE, 2015.

[208] Detlef Zühlke and Lisa Ollinger. Agile automation systems based on cyber-physical systems and service-oriented architectures. In *Advances in Automation and Robotics, Vol. 1*, pages 567–574. Springer, 2011.