The Open Group Guide

Microservices Architecture for the Internet of Things (MSA-IoT)

THE Open GROUP

The Open Group Guide

Microservices Architecture for the Internet of Things (MSA-IoT)

ISBN:	1-947754-13-3
Document Number:	G187

Published by The Open Group, April 2018.

Comments relating to the material contained in this document may be submitted to:

The Open Group, Apex Plaza, Forbury Road, Reading, Berkshire, RG1 1AX, United Kingdom

or by electronic mail to:

ogspecs@opengroup.org

Contents

Preface

The Open Group

The Open Group is a global consortium that enables the achievement of business objectives through technology standards. Our diverse membership of more than 600 organizations includes customers, systems and solutions suppliers, tools vendors, integrators, academics, and consultants across multiple industries.

The Open Group aims to:

- Capture, understand, and address current and emerging requirements, establish policies, and share best practices
- Facilitate interoperability, develop consensus, and evolve and integrate specifications and open source technologies
- Operate the industry's premier certification service

Further information on The Open Group is available at www.opengroup.org.

The Open Group publishes a wide range of technical documentation, most of which is focused on development of Open Group Standards and Guides, but which also includes white papers, technical studies, certification and testing documentation, and business titles. Full details and a catalog are available at www.opengroup.org/library.

This Document

This document is The Open Group Guide to Microservices Architecture for the Internet of Things (MSA-IoT). It has been developed and approved by The Open Group.

Trademarks

Acknowledgements

The Open Group gratefully acknowledges the contribution of the authors in the development of this Guide, and would also like to thank Gillian Hughes (Capgemini SA) for her keen eye in review.

The Open Group also gratefully acknowledges the contribution of the following people in the development of this Guide:

- Members of The Open Group SOA Work Group, in collaboration with The Open Group IoT Work Group and The Open Group Open Platform 3.0™ Forum of which it is a part
- Members of The Open Group Architecture Forum
- Members of the MSA Project Team:

  — Ovace A. Mamnoon, DXC Technology (MSA Project Co-Chair)

  — Peter Maloney, Raytheon Company

  — Somasundram Balakrushnan, Salesforce.com (MSA Project Co-Chair)

  — John Bell, Ajontech LLC

  — Anurag Choudhry, Tata Consultancy Services Ltd.

  — Chris Harding, Lacibus Ltd.

  — Leszek Jaskierny, DXC Technology

  — John Knoepfle, Oracle Corporation

  — Satyajit Malavde, DXC Technology

  — Kumar Avishek Singh, Tata Consultancy Services Ltd.

  — Michelle Supper, The Open Group

About the Authors

Ovace A. Mamnoon, DXC Technology (MSA Project Co-Chair)

Ovace Mamnoon is a Principal Advisor at DXC Technology. His near 20 years of Architecture and Engineering experience encompass areas that have a strong bearing on Microservices Architecture (MSA) including developing embedded systems, SmartGrid design and implementation, SOA, Internet of Things (IoT), Cloud Native Solutions, Digital Enterprise, and others. All with a strong focus on Enterprise Architecture. Ovace is an active participant in The Open Group and is a Co-Chair of The Open Group MSA Project.

Peter Maloney, Raytheon Company

Peter Maloney is a Senior Engineering Fellow at Raytheon Company. He became interested in Enterprise Architectures and particularly SOA as a result of the ever-expanding need for providing access to increasingly complex data products to a diverse group of end users, with the resulting needs for collaboration, throughput management, and security. He is a Raytheon Certified Architect, a certification accredited by The Open Group, and a three-time winner of the Raytheon Excellence in Technology Award. He holds one patent and has authored more than a dozen papers.

Somasundram Balakrushnan, Salesforce.com (MSA Project Co-Chair)

Somasundram Balakrushnan is a Senior Program Architect at Salesforce.com and a TOGAF® 9 Certified Enterprise Architecture practitioner. He is an experienced Enterprise Architect and leader in SOA-based architecture development. He has led multiple teams in developing proof-of-concept architectures using microservices. His association with The Open Group includes: the SOA Work Group, the MSA Project, evolution of the TOGAF standard, and Open Platform 3.0™. Som is leading the MSA Project in the capacity of Co-Chair.

John T. Bell, Ajontech LLC

John Bell has worked in the Technology industry for 40 years and in the Hospitality industry for 15. He is an active member of the Hospitality Technology Next Generation (HTNG), The Open Group SOA Work Group, and the IEEE Standards Association. He is the founder and Principle Consultant for Ajontech LLC providing services to hospitality-related companies in the areas of IT Security, Privacy, and Enterprise Architectures. He is the author of several books on software development and was an Associate Professor for the Center of Applied IT at Towson University for 14 years.

Anurag Choudhry, Tata Consultancy Services Ltd.

Anurag Choudhry is a TOGAF® 8 Certified Solution Architect. He has over 17 years of experience in IT and currently he is part of the Architecture Focus Group of the Banking and Financial Services (BFS) business unit at Tata Consultancy Services (TCS). His focus area includes Digital Architecture Consulting, Cloud Native Architecture, Microservices Architecture, API Management, and Open Banking. He has authored numerous papers in reputed international journals.

Chris Harding, Lacibus Ltd.

Chris Harding has been working in the IT and telecommunications industry for over 40 years. Until recently he was Director for Interoperability at The Open Group, where his responsibilities included supporting the SOA Work Group and its MSA Project, and being Forum Director of the Open Platform 3.0™ Forum, whose mission is to help enterprises gain business advantages from recently-emerged technologies including cloud, mobile and social computing, big data, and the Internet of Things. He is now founder and chief executive of Lacibus Ltd., which provides services related to virtual data lakes and data-centered architecture.

Leszek Jaskierny, DXC Technology

Leszek Jaskierny is a Master IT Architect with extensive experience in all stages of software development and delivery. Working for Compaq/HP/HPE/DXC Technology since 2002, he designed complex software solutions and delivered projects for major Financial Services Industry customers. He gained programming, solution development, and project leading experience, building IVR systems, delivering data management projects and front-end applications. His current focus is on Microservices Architecture, IoT, and enterprise-scale distributed transnational systems.

John Knoepfle, Oracle Corporation

John Knoepfle is an Enterprise Cloud Architect at Oracle Corporation. With his experience in software development, application and Enterprise Architecture, and cloud computing, he provides thought leadership to clients migrating applications to the cloud. He also has significant experience in establishing SOA and successful transformation of mission-critical systems to that architecture at a former employer. John’s current focus areas include architecture consulting, cloud migration, microservices, and SOA.

Satyajit Malavde, DXC Technology

Satyajit Malavde is a Senior Technical Advisor at DXC Technology (formerly HP Enterprise). He has 18+ years of experience in Digital Transformation, Enterprise Architecture, application development, and application integration solutions. He has successfully transformed large business-critical applications to the cloud. His current focus areas include Digital Transformation consulting, cloud migration, microservices, IoT, and SOA.

Kumar Avishek Singh, Tata Consultancy Services Ltd.

Avishek Singh is an Enterprise Architect practitioner with over 19 years of experience in IT. He is an Open Group Certified IT Architect (Open CA) and is working as an Enterprise Architect with the Digital and Enterprise Transformation practice of HiTech business unit at Tata Consultancy Services Ltd. He also holds TOGAF® 8, MCP, MCTS, PMP, and CSM certifications and he has published numerous papers and articles on architecture and technology. His core focus areas include Enterprise Architecture consulting, Digital Enterprise Transformation, SOA, Integration Architecture, Microservices, API, and CloudApps.

Michelle Supper, The Open Group

Michelle Supper is the new Director of the Open Platform 3.0™ Forum at The Open Group. An experienced management consultant, and Business, Systems, and Enterprise Architect, Michelle has helped many clients in the defense, government, and public sectors to develop new solutions and transform their operations. In addition to holding a doctorate in X-ray Astrophysics, Michelle is TOGAF® 9 Certified, ArchiMate® 2 Certified, and Open FAIR™ Certified.

Referenced Documents

(Please note that the links below are good at the time of writing but cannot be guaranteed for the future.)

- Big Data Analysis for Smart Farming, C. Kempenaar, C. Lokhorst, E.J.B. Bleumer, R.F. Veerkamp, Th. Been, F.K. van Evert, M.J. Boogaardt, L. Ge, J. Wolfert, C.N. Verdouw, M. van Bekkum, L. Feldbrugge, J.P.C. Verhoosel, B.D. Waaij, M. van Persie; H. Noorbergen, Wageningen University & Research; refer to: www.wur.nl/en/Publication-details.htm?publicationId=publication-way-353037373634
- Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions, Gregor Hohpe, Bobby Woolf, Addison-Wesley Professional, 2003
- Internet of Food and Farm 2020, Harald Sundmaeker, Cor Verdouw, Sjaak Wolfert, Luis Pérez-Freire (2016), pp.129-151
- ISO/IEC CD 30141: Information Technology – Internet of Things Reference Architecture (IoT RA) (under development)
- Microservices Architecture, White Paper (W169), published by The Open Group, July 2016; refer to: www.opengroup.org/library/w169
- Open Messaging Interface (O-MI), an Open Group Internet of Things (IoT) Standard (C14B), published by The Open Group, September 2017; refer to: www.opengroup.org/library/c14b
- Osmotic Computing: A New Paradigm for Edge/Cloud Integration, M. Villari, M. Fazio, S. Dustdar, O. Rana, R. Ranjan, IEEE Cloud Computing, Volume 3, Issue 6, November/December 2016; refer to: http://ieeexplore.ieee.org/xpl/tocresult.jsp?isnumber=7802493
- Reference Architectures and Open Group Standards for the Internet of Things, White Paper (W16D), published by The Open Group, December 2016; refer to: www.opengroup.org/library/w16d
- Service-Oriented Architecture (SOA), White Paper (W074), published by The Open Group, July 2007; refer to: www.opengroup.org/library/w074
- SOA Reference Architecture, an Open Group Standard (C119), published by The Open Group, December 2011; refer to: www.opengroup.org/library/c119
- The OpenFog™ Reference Architecture for Fog Computing, The OpenFog Consortium; refer to: www.openfogconsortium.org/wp-content/uploads/OpenFog_Reference_Architecture_2_09_17-FINAL.pdf

1 Introduction

A Microservices Architecture (MSA) consists of a massively parallelized, distributed set of atomic function applications which together constitute a resilient, scalable, and flexible solution. These characteristics can also be found within Internet of Things (IoT) solutions, which typically consist of a large number of single function devices or sensors that are widely distributed.

In this Guide, we will explore the synergies between these two evolving solutions, and identify where MSA can be an optimal fit, and an enabler to, IoT solutions. Patterns and critical decision factors relating to security and architecture will be considered, and the benefits of the MSA approach will be further highlighted in a series of case studies.

This work builds on the foundational "Microservices Architecture" White Paper published by The Open Group (see Referenced Documents), by identifying practical applicability of MSA in IoT solutions. While it is certainly not a comprehensive coverage of the subject, it is hoped that this Guide will serve as a disruptor and innovation catalyst for developing optimal solutions by showing how MSA and IoT can be combined to produce a modern, digital enterprise-ready solution that is flexible, scalable, highly resilient, and able to meet constantly evolving needs.

MSA in conjunction with IoT makes the boundary between the IoT devices and sensors permeable, following a seamless flow between physical world and IT. In this way, the MSA style of architecture enables The Open Group vision of Boundaryless Information Flow™.

2 Microservices and MSA Overview

Microservices Architecture (MSA) is a style of architecture that defines and creates systems through the use of small independent and self-contained services that align closely with business activities. These "microservices" are the primary architectural building blocks of an MSA.

An MSA has the following three key characteristics:

1. **Service-independence**: a microservice is independent of other microservices or other services. Each service is developed, deployed, and evolved independently.
2. **Single responsibility**: each microservice is mapped to, and responsible for, one atomic business activity.
3. **Self-containment**: a microservice is a self-contained, independent deployable unit. A microservice encompasses all external IT resources (e.g., data sources, business rules) necessary to support the unique business activity.

For more detail on characteristics, features, and governing principles of microservices and MSA, please refer to The Open Group Microservices Architecture White Paper (see Referenced Documents).

Microservices are:

- **Technology-independent**: each microservice can be built on its own technology and the overall solution can include multiple technology platforms; this enables a flexibility to adopt any technology at any time
- **Reusable**: microservices can be readily applied to different situations and used with many kinds of devices
- **Independently deployable**: a microservice has its own lifecycle, and can be built, changed, tested, and deployed on its own; this allows the microservices-based solution to be highly flexible and adaptable
- **Horizontal elastic scalability**: scaling of an MSA is easily achieved by independently instantiating (or shutting down) additional microservice instances
- **Decoupling**: coupling is the degree of interdependence between services

MSA requires that services are decoupled, as a consequence of independence. This is an inherent characteristic of microservices. Clients remain unaware of the implementation details, and two microservices in an MSA will remain unaware of each other's function – decoupling is highly important in the microservices environment.

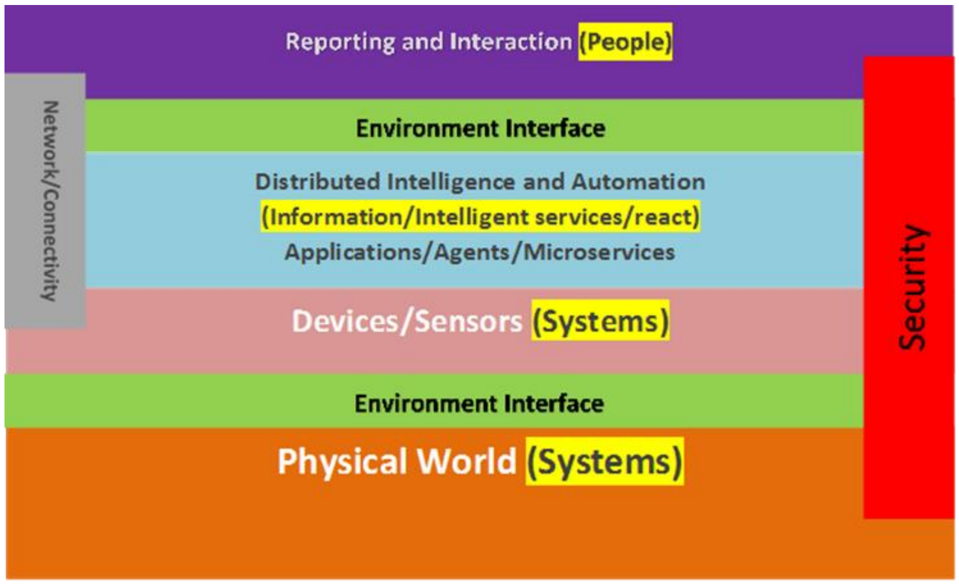
2.1 Internet of Things Overview

Internet of Things (IoT), like many other industry acronyms, has a variety of interpretations. Let us start with establishing a baseline definition that can be consistently leveraged throughout this

Guide. We will use the definition of IoT from The Open Group White Paper: Reference Architectures and Open Group Standards for the Internet of Things (see Referenced Documents). This definition of the basic concept aligns with that of the International Organization for Standardization (ISO):

"IoT is defined as an infrastructure of interconnected physical entities, systems, and information resources together with the intelligent services which can process and react to information of both the physical world and the virtual world and can influence activities in the physical world."

Draft ISO/IEC IoT Reference Architecture (as of 2016) definition of Internet of Things (IoT)



Reporting and Interaction (People)
Network/Connectivity
Environment Interface
Distributed Intelligence and Automation
(Information/Intelligent services/react)
Applications/Agents/Microservices
Devices/Sensors (Systems)
Environment Interface
Physical World (Systems)
Security

Figure 1: An IoT Stack, Based upon the ISO/IEC Definition

The essence of an IoT solution is the interaction via information services and the ability to react quickly. This is where microservices excel; they are focused on performing a single (atomic) function and can react to events. Microservices also have a small resource footprint which makes them particularly well suited to be deployed on sensor devices, and they are highly distributed (parallel) instances, which maps well to highly distributed mesh sensor networks. In the next section, and remainder of the Guide, we will explore these synergies.

2.2 Areas of Synergy and Benefits of MSA to IoT

It is the nature of an IoT network that the many IoT devices collect huge amounts of data through sensors and send this data either to the cloud or to a custom data lake/store. The reason that we want to collect all this data may be to extract knowledge, take decisions, provide real-time visualization and data feeds, or to perform historical and predictive analytics that will drive business decisions at velocity and provide real-time notification and status.

There are a number of areas of obvious synergy between MSA and IoT. In many ways, the two are an obvious fit for one another, and microservices will be an enabler for many IoT applications. Microservices can provide many benefits to an IoT application because of this synergy. Some of these areas and their associated benefits are:

- **Heterogeneous networks**: IoT networks are inherently distributed, comprised of varying components, sensors, and connecting services

  These networks are growing and constantly changing as a result of added or modified devices, and will therefore tend to leverage available infrastructure rather than depend upon top-down, purpose-built networks. These networks will comprise a diversified set of components and tend towards being *ad hoc*. Microservices provide an ideal way to operate over these networks, since their independence and decoupling allows them to be deployed and upgraded as necessary without impacting the remainder of the network.

- **Decentralized governance and data management**: heterogeneous networks composed of myriad distributed devices and services will almost by definition require decentralized governance and data management

  Component and service upgrades, additions, and replacement will take place across the network on an as-necessary schedule, and deployment of these upgrades will quite possibly be performed by independent parties. Centralized governance of such an architecture would be extremely difficult and would demand an intensive, ongoing effort at coordination. Conversely, the fully independent nature of microservices makes decentralized governance possible at a very granular level, since the only real requirement is that the microservice is runtime-compatible with the environment in which it is operating. Microservices within the same architecture do not even have to be written in the same programming language.

- **Resiliency (design for failure)**: a network composed of hundreds or even thousands of separate components cannot afford to be dependent on the health and status of a single or small number of these

  The major characteristic of an MSA is independence both in deployment and runtime of the services. It is this independence that provides for an architecture which is robust in the event of failure. Individual failed services will simply have their role taken over by new instantiations of those services. Failed devices may be replaced by other devices or, dependent on the application, may be emulated or extrapolated based on other devices in the network. In either case, the overall function required at the application level will proceed unimpeded by the individual failures. Additionally, the inclusion of MSA can improve the scalability of an IoT network. Since MSA is inherently scalable, placing an MSA layer above an IoT "grid" allows the network to handle more consumers than it might otherwise, through application of the Sensor Cache pattern (see Applicable Patterns on page 8). Novel approaches to the problems of failure may emerge as these highly resilient architectures continue to be developed.

- **Independence**: service-independence is the key characteristic of an MSA and the primary factor distinguishing an MSA from a more general Service-Oriented Architecture (SOA)

  The typical characteristics of an IoT application, such as heterogeneous networks, decentralized governance, and resiliency, will be much better served by an architecture that not only allows but requires service independence. This independence will be a prime requirement to allow these large, often *ad hoc* networks to grow and evolve over time in an efficient manner, without requiring massive oversight to coordinate and manage deployment of new and updated service components across the network. MSA provides a logical architectural choice to develop these new large-scale networks.

- **Evolution at the atomic level**: IoT architectures composed of many devices, components, and services will face constant evolution in the face of changing needs and rapid technological advancement

  The IoT network provides a real-world interface which collects copious amounts of data through its sensors, and through the use of intelligent services, can turn this collection of data into useful information, which the actors (people or systems) on the other side of the network can process, react to, and use as the basis for decisions or actions. To allow for the most flexibility in adapting to meet business needs, these IoT architectures will need to be designed to enable this change to happen at the most atomic level possible. Again, the independence provided by an MSA inherently allows for this flexibility, and allows for independent deployment, replacement, and addition without requiring a massive coordinated effort spanning the entire application space.

- **Smart endpoints and dumb pipes**: one of the classic descriptions of an MSA, this emphasis takes the need for intelligence and orchestration away from the network infrastructure and pushes it into the endpoints of the network

  A smart network includes sophisticated facilities for message routing, choreography, transformation, and applying business rules, which in turn requires much tighter coupling between elements, and will be much more expensive and complicated when deploying new functionality or components. Putting that intelligence and domain logic into the many endpoints of an IoT network removes the properties of the network from the critical path, and makes the network highly decoupled and cohesive; this enables the distributed governance and data management that will greatly simplify deployment and maintenance of the network.

- **Security**: numerous recent headlines have highlighted the need for security to be incorporated as an integral element in the design of networks, particularly networks as vast as those envisaged for IoT applications

  The ubiquitous, always-on nature of IoT networks will provide a vast breeding ground for Distributed Denial of Service (DDoS) attacks using these devices to form botnets, and other forms of malware. This problem is exacerbated by the fact that many web-enabled devices suffer from fairly primitive built-in security, and that security is rarely seen as a pressing concern by the owners of devices such as Digital Video Recorders (DVR) and web-enabled cameras. An MSA can produce a change to the security model of the network. Microservices will need to be capable of performing self-validation of security tokens (authentication and authorization) in order to maintain the concept of independence. This type of distributed security will be mandatory for a secure IoT network. Small, rapidly deployed microservices can be pushed across the network to the device level, responding rapidly to new threats and relieving the need to depend on embedded code stored in device firmware. This action can also take place without the need for conscious action on the part of the device owner, who is otherwise an obvious attack vector for malicious software. It is also possible to develop architectures that can provide a secure architecture through a defense-in-depth approach by aggregating the security provided at the individual service level. (It should be noted, however, that architecture alone cannot solve all security issues; use of "primitive", always-on devices will likely present attack vectors for malicious actors that security can mitigate, but not completely prevent.) An MSA will be an enabler for the kind of robust security that will be required for IoT networks to truly fulfil their potential.

These key aspects of MSAs, and the characteristics upon which they rely, can provide a number of benefits to IoT implementations. Some examples of these benefits follow:

- **Device provisioning and management**: an IoT device, upon initialization, will establish a relationship with its controlling environment (which may be the cloud or some other data center), usually through its unique identifier, such as a serial number, so that the business is notified that the device is active

  Further, a device registry for provisioning and associating with an end object (for example, a patient in a hospital) uses several other parameters such as object number (e.g., patient ID), manufacturer product line, model, and version, etc. This requires many administrative Application Programming Interfaces (APIs) to provide create, update, and delete operations as well as get operational data from the devices.

  Using microservice-based APIs (built-in to devices), businesses can send commands to the device for the purposes of provisioning, providing software updates, or updating local data caches, etc. The independence of services and decoupling provided by microservices is ideally suited for what will likely be a heterogeneous network of devices that will grow and change over time. Also, microservice-based implementations will provide data interchange standardization, vendor-neutral interfaces, and a high degree of agility. The decentralized governance of the MSA is another benefit (indeed, a necessity) for managing the large, distributed, and time-varying networks found in IoT implementations.

- **Telemetry ingestion**: devices may be sending multiple messages a second, and there may be millions of messages a day

  This requires an automated communications process by which data is collected at remote devices and transmitted to a receiving platform.

  A commonly used mechanism offered by cloud vendors is based on highly scalable publish-subscribe event-based consumers to process and analyze the massive amounts of data produced by connected devices and applications. Microservices offer the independence at deployment that will allow for a scalable, resilient architectural solution. It will also reduce the dependency with typical hub-based implementation, which is often becomes a single point of failure.

  Microservice-based APIs can be used to construct the messages (for example, a JavaScript™ Object Notation (JSON) message consisting of the identity of the device, the identity of the participant that is using the device, the longitude and latitude of where the device is located, a timestamp of when the sensor readings were taken, and a list of sensor readings, etc.) that will be sent to the receiving platform in a loosely-coupled manner and using a standard data exchange format/protocol.

  Further, it provides the flexibility to pre-process or otherwise modify data before it is sent to the centralized data platform (using an edge computing implementation). Some examples of this include:

  — The ability to store/pre-process several readings in cache and send at regular intervals (predefined) to reduce the network traffic

  — The combination of multiple devices and a local gateway to connect to a destination data platform (cloud or on-premise)

  — Sending the same data to multiple channels/data platforms

- **Device status and notifications**: the IoT solution requires the ability to visualize the status of the message pool in real time through tabular or graphical user interface tools

  In addition, some messages may contain information about alert, status, etc., so the IoT solution should provide a mechanism to provide real-time notifications.

  When messages arrive at the receiving platform from a source device, a microservice can read the messages related to the device status, the alarms and warnings (based on certain pre-defined parameters) from common repositories, and run custom business logic on a filtered subset of the incoming messages to create a cascading set of notification repositories. These custom notifications can further provide push notifications to mobile/other devices. At the same time, the microservice can log the alarms to Structured Query Language (SQL)-based storage for reporting and visualization purposes. The independently deployable nature of microservices makes them ideal for such applications, and security considerations will be paramount in any service responsible for conveying warnings or alerts.

- **Data transformation**: employing MSA on top of IoT allows for seamless transformation of raw data into useable information for later analysis

  MSA provides reliable and available storage while removing single points of failure. As the incoming message stream arrives, it is important to perform some analytics to make data useful for the consumer. Microservices can be leveraged to connect and consume events in the repository based on the pre-configured properties and temporal properties, such as arrival time. We can have dedicated microservices to select and process specific messages, and then direct these to one or more storage locations such as NoSQL and SQL-based databases, or send them to another repository for further processing. Since the raw data is a little cryptic, it makes sense to wrap the data with an API that provides context and, if necessary, business logic so that the data is provided in a meaningful way to the downstream analytics and data visualization applications.

  There could be sets of independent microservices to manage different and unique sets of data that may be further used to create data visualizations in responsive web applications. For example, a service which aggregates sensor location data, that can be consumed by a visualization tool to create a dashboard that displays the device locations on maps of Bangalore, Mumbai, and New York.

The implementation of a massive IoT network sending data to a centralized hub, even a cloud-based one, will strain the bandwidth constraints and networking resources of any kind of central server. There is an emerging consensus that even cloud architectures may eventually find themselves struggling to deal with the enormous volumes of data that will be generated by IoT networks. The OpenFog™ Consortium and the concept of Osmotic Computing (see Referenced Documents) have both been developed in response to this problem. Both rely on the concept of pushing out information processing to the edge of the network, with some kind of micro data center located at the edge of the network, to reduce the volume of data flowing back to the centralized cloud location.

Independently deployable microservices are ideal candidates to be implemented in this type of flexible architecture whose instantiation will change as a function of time and network/system performance. This implementation also benefits from the “smart endpoints, dumb pipes” characteristic, and of course the need for security will loom even larger with data processing taking place over such a widely distributed area.

3 Applicable Patterns

3.1 Pattern 1: Interpolation

A function can be used to determine approximate values between a set of known value points.

3.1.1 Problem Description

In a grid of network-enabled sensors individual sensors may fail when being read, leading to a loss of information.

3.1.2 Solution

It may be possible to replace the missing information with values determined by interpolating the data provided by nearby sensors. This pattern requires that there be a relationship between the position of the sensors and the value being measured. This is common for many sensor types.

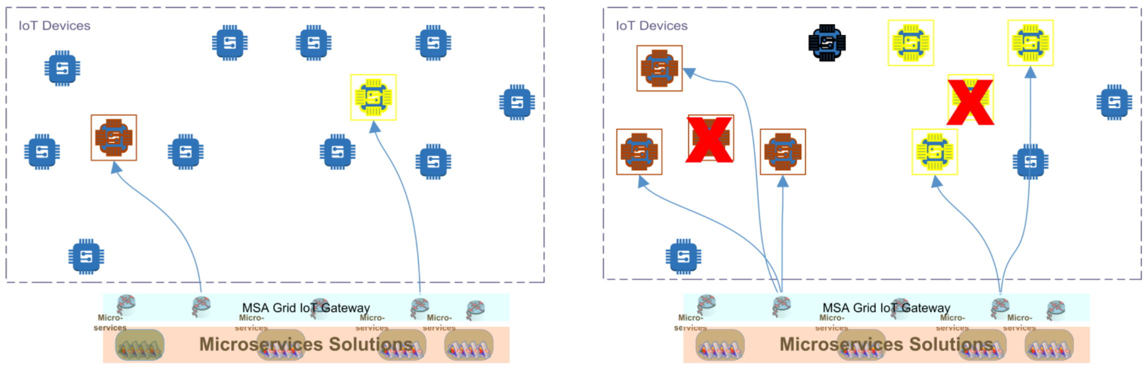


Figure 2: Interpolation Pattern

3.1.3 Use-Case

If temperature sensors are distributed throughout a facility, it is reasonable to expect that the value between two or more sensors will be related to the values of the surrounding sensors.

3.1.4 Implication

In an MSA of IoT devices, missing nodes can be replaced temporarily through interpolation of the data from surrounding nodes providing some of the same resiliency expected in a traditional MSA environment.

3.1.5 Relevance

This pattern addresses MSA resiliency in a situation where it may take some time to replace a failing node.

3.2 Pattern 2: Sensor Façade

A new interface can be placed in front of an existing interface to make the existing interface work like the desired interface.

3.2.1 Problem Description

A device acting as an IoT sensor does not implement its services in a way that is compatible with the desired consumers. For example, an analog to digital converter reading temperature from an analog device may return a value between 0 and 4095 corresponding to temperature range of –50°C to 125°C. The service needs to provide temperature readings.

3.2.2 Solution

Provide a façade service in front of the IoT device that converts between the raw 0 to 4095 values and the desired temperature values.

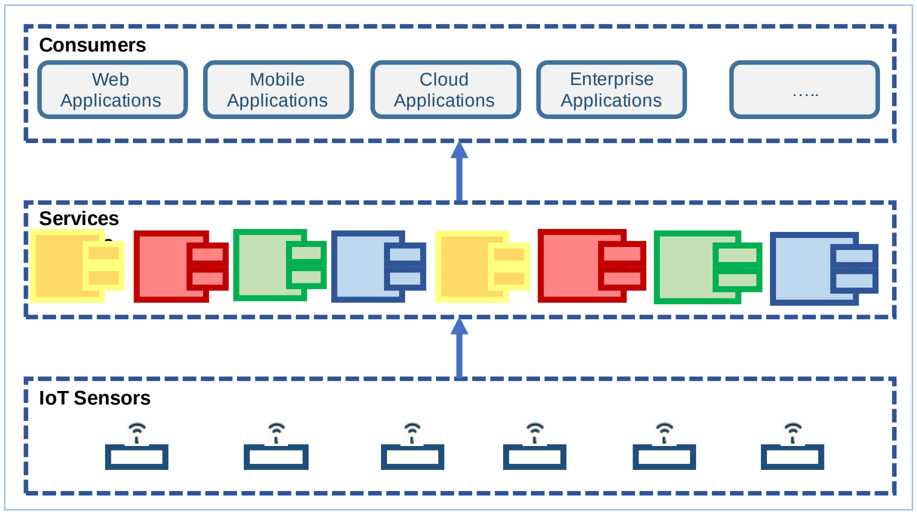


Figure 3: Sensor Façade Pattern

3.2.3 Use-Case

Many generic sensors provide readings in raw binary values that are ranged to meet the application needs. These raw values are not typically useful unless converted into meaningful units of measurement.

3.2.4 Implication

The service may also allow response with different units for the values returned; for example, Celsius and Fahrenheit.

3.2.5 Relevance

This is a common IoT pattern, not specific to MSA but relevant to MSA-IoT environments.

3.3 Pattern 3: Cache

In an MSA, caches are grid caches distributed across multiple service nodes. A cache stores a result so the effort to regenerate or read the result is not repeated. Cache entries typically expire after a time and need to be refreshed.

3.3.1 Problem Description

Sensor data may be desired on a regular basis by more consumers than a sensor can handle.

3.3.2 Solution

Assuming the sensor is typically accessed in a synchronous model, the sensor reads occur through a cache that returns the most current sensor value until the cache entry expires. This causes a single direct sensor read to refresh the cache entry. In an asynchronous model, each sensor periodically updates an entry directly into the cache allowing synchronous reads of the most recent update. If asynchronous notification is required then the Multicast pattern should be used instead.

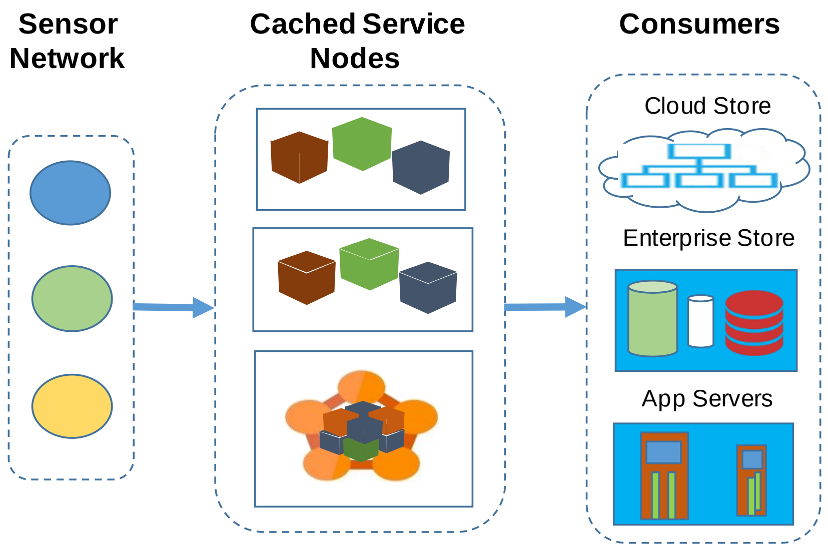


Figure 4: Cache Pattern

3.3.3 Use-Case

Each sensor in a network of sensors requires 500 milliseconds to generate a reading, but the reading is being requested 10 times per second by multiple service node instances within an MSA.

3.3.4 Implication

Reading from the service nodes is independent of the capabilities of the sensor due to caching. Caching also enables accessing an asynchronous sensor as if it were a synchronous sensor.

3.3.5 Relevance

Caching improves both the scalability and the resilience of an IoT MSA.

3.4 Pattern 4: Gateway

A gateway sits in front of services providing common interface and service support for those services that sit behind the gateway. These gateway services may include, for example, security enforcement, monitoring, and transformation. In the IoT world the gateway sits above the network of IoT devices.

For a Gateway pattern to be a part of an MSA the gateway must itself be implemented as an MSA with no dependency between any individual nodes of the gateway to any individual node of the MSA behind it.

3.4.1 Problem Description

A grid of IoT devices exists, but these devices must be accessed via a gateway due to limitations of the IoT devices. These limitations might include security enforcement, protocol transformation, or service enhancement through the Interpolation or Façade patterns, for example.

3.4.2 Solution

The solution is to create a microservice-based gateway where each microservice node can interconnect and communicate with each other and the IoT nodes. An MSA Gateway pattern should be used because a standard Gateway pattern does not provide the resilience required to ensure consistent access to the underlying grid of IoT devices. The gateway nodes provide a grid at a layer above the IoT grid. This hides the underlying IoT grid, but brings the MSA resilience to the Gateway pattern. Each gateway node may handle subsets of the IoT total information set and work together to return the desired service results.

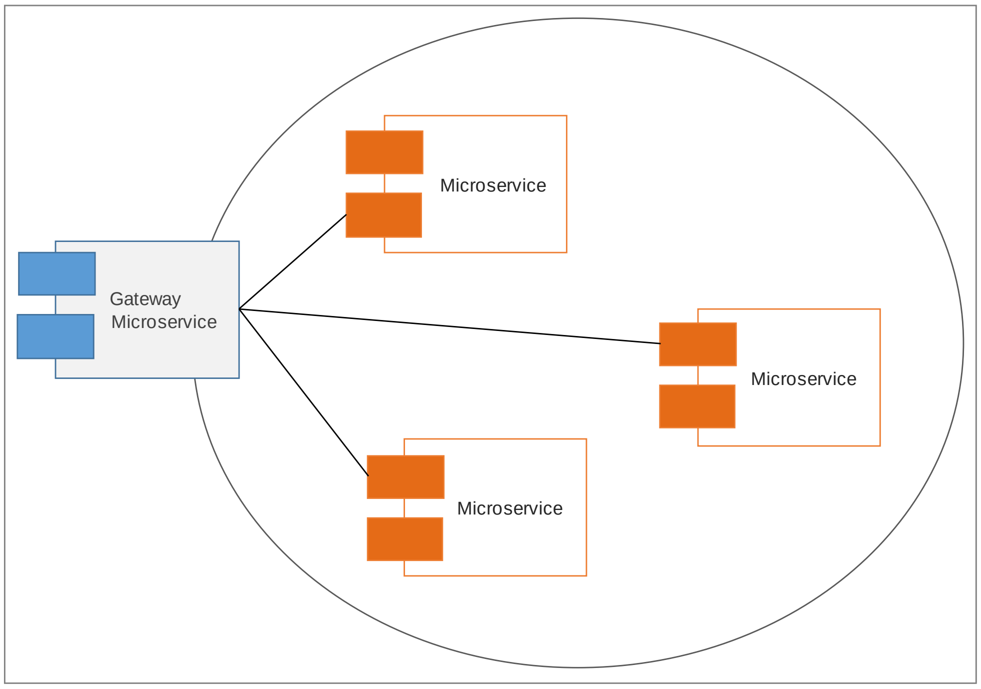


Figure 5: Gateway Pattern

3.4.3 Use-Case

A global energy use monitoring application is required to monitor the near real-time energy costs of operating facilities around the world. Each facility has sensors responsible for monitoring different aspects of energy consumption within the facility. The application needs to be widely accessible and highly available.

3.4.4 Implication

If the IoT Gateway pattern can be implemented as a microservice, the gateway can potentially also be used to implement the Interpolation and Façade patterns.

3.4.5 Relevance

The IoT Gateway pattern is a common IoT pattern. Implementation using an MSA facilitates use of many other IoT patterns within an MSA environment.

3.5 Pattern 5: Sensor Aggregator

An IoT device typically represents a single point of information or a datum. The Aggregator pattern recognizes that often the information value is not in the data of a single sensor, but in the data from the combined reading of many sensors.

3.5.1 Problem Description

Data from a large number of sensors is collected and needs to be analyzed with the analysis results exposed as operations within a microservice.

3.5.2 Solution

In this scenario a large number of sensors take readings and analysis is performed based on the sensor values. Results of the analysis are exposed as a microservice operation.

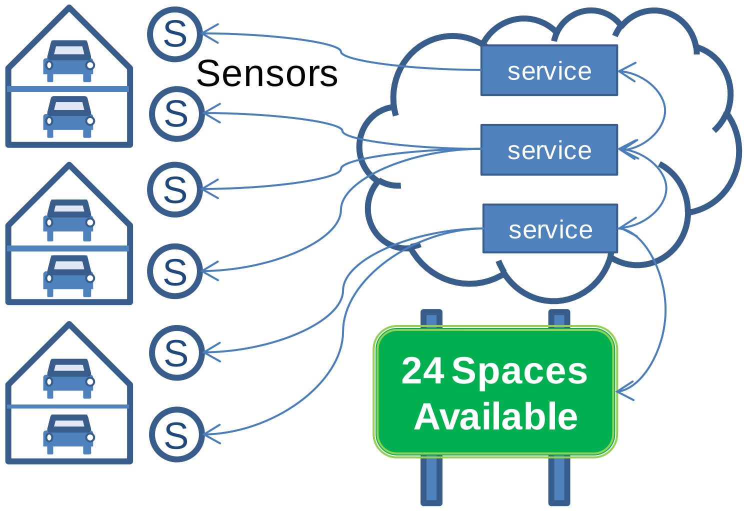


Figure 6: Sensor Aggregator Pattern

3.5.3 Use-Case

Sensors associated within a large parking garage facility are used to determine if each parking space is empty or full. The state information is maintained in an MSA and is analyzed to determine how many parking spaces are available in the facility overall and on each floor of the facility. This information is displayed on digital signs to drivers as they enter the facility to help them determine the most likely floor or parking lot to find an available parking space.

Other use-cases might include:

- Analysis of multiple roadside speed detectors to detect traffic slowdowns
- Analysis of multiple GPS systems in phones to detect traffic patterns

3.5.4 Implication

Loss of individual sensors typically has little impact on the outcome.

3.5.5 Relevance

This pattern applies to a large set of MSA-IoT problems.

3.6 Pattern 6: Control Aggregator

The Control Aggregator pattern uses analysis performed on the collected information of multiple sensors to create actions that may occur on multiple control points to achieve a desired outcome.

3.6.1 Problem Description

Data from a large number of sensors needs to be collected and analyzed, possibly leading to control actions taken across multiple devices.

3.6.2 Solution

In this scenario a large number of sensors take readings and analysis is performed based on the sensor values. Results of the analysis are exposed as a microservice operation. The results of the analysis are then used to generate operation commands to a collection of control points. These commands are a reaction to the analysis.

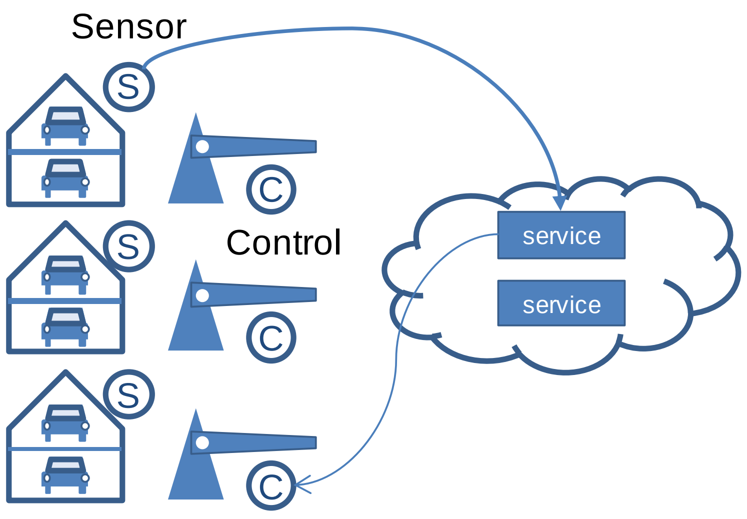


Figure 7: Control Aggregator Pattern

3.6.3 Use-Case

Sensors associated with a large parking garage facility are used to determine if each parking space is empty or full. The state information is maintained in an MSA and is analyzed to determine how many parking spaces are available in the facility overall and on each floor of the facility. If parking lots are full, entry gates will remain down in those lots and signage will direct drivers to empty lots or floors with available space. As cars depart a lot and a certain threshold of spaces become available, the entry gates will allow new cars to enter and the signage is changed to direct drivers to the newly available lot or floor.

3.6.4 Implication

This pattern allows a loose coupling between the measurement or sensor side of the problem and the control or action side of the problem.

3.6.5 Relevance

This pattern applies to a large set of MSA-IoT problems.

3.7 Pattern 7: Multicast

The Multicast pattern receives an incoming message and sends it to multiple endpoints. It is commonly used to distribute event notifications to several independent listeners. Typically, each listener registers their interest in receiving the notifications via a subscription.

There are typically two ways of receiving information from an IoT device: synchronous and asynchronous. A synchronous device waits for a request and responds appropriately. An asynchronous device sends a notification when an event occurs. In an MSA-IoT environment the notification message is sent to a microservice node instance that then distributes the message to the subscriber list.

3.7.1 Problem Description

Consumers desire to be notified when an event occurs on an IoT device.

3.7.2 Solution

Use the Multicast pattern when a consumer that would like to receive notifications registers their interest as a subscriber or listener. The device notification is sent to a broker service that broadcasts the notification to all subscribers.

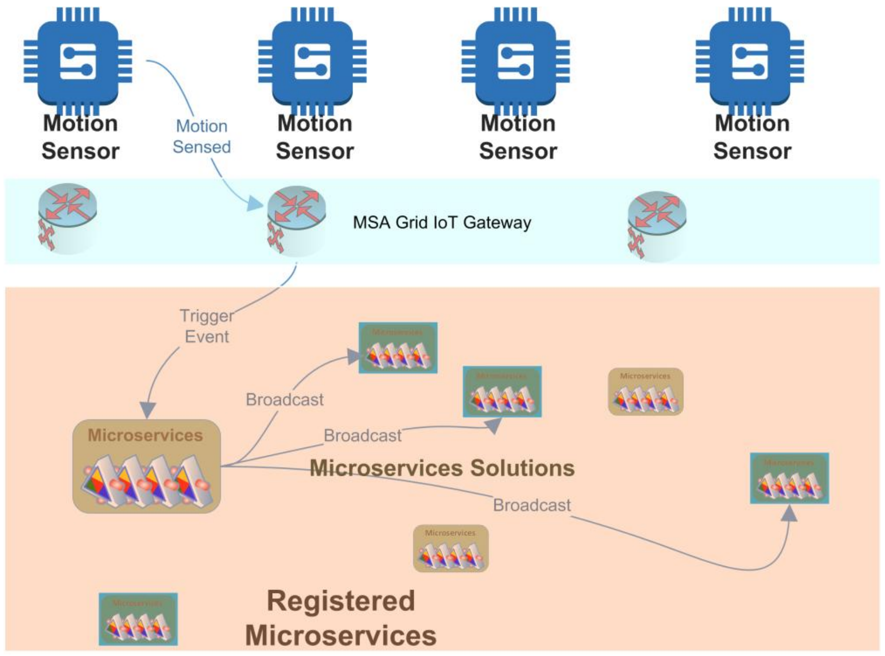


Figure 8: Multicast Pattern

3.7.3 Use-Case

An occupancy monitoring service wants to be notified when motion detector devices detect motion in a location.

3.7.4 Implication

This service enables asynchronous notifications to be shared with multiple listeners and typically requires subscribe, unsubscribe, and subscription-status operations.

3.7.5 Relevance

This pattern is important for event-based IoT devices.

4 Case Studies

4.1 Case Study 1: Large-Scale Agriculture – Enabling Smart Farming

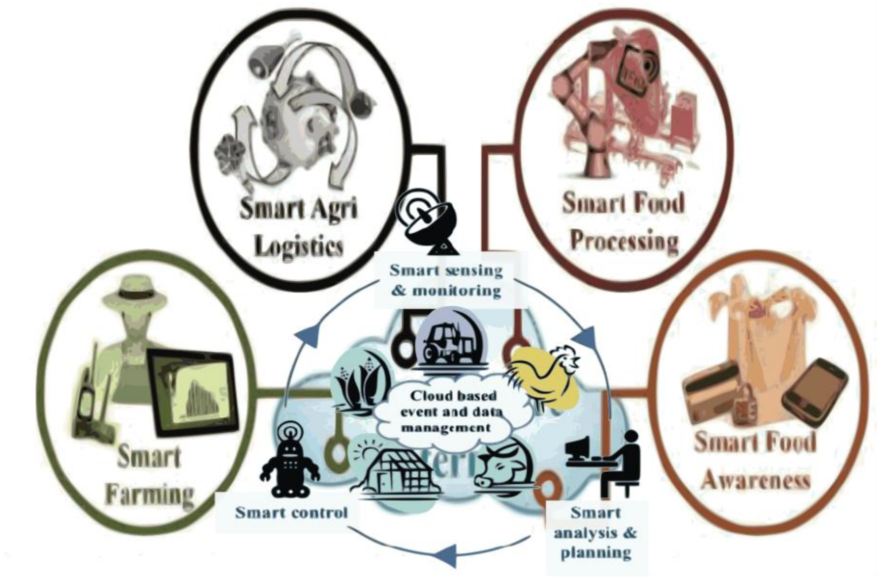
(Hypothetical Study to Demonstrate the Value of MSA for IoT)

Since IoT is beginning to get a strong foothold in this space already, this is an excellent time to explore the synergies and benefits of MSA for this IoT solution.

4.1.1 Farming the Future Today

Producers continue to integrate technology, mechanization, and improved processes to their operations. These allow farms to create efficiency, achieve scale, and maximize profitability. Farms are using data-driven technologies, such as GPS and GIS soil mapping, to bring increasing precision and accuracy to seeding, harvesting, and input use. Technologies in the greenhouse and livestock sectors are advancing the world of farm automation at the push of a button.

Statistics Canada, www.statcan.gc.ca[1]


Smart Agri Logistics
Smart Food Processing
Smart sensing & monitoring
Smart Farming
Cloud based event and data management
Smart Food Awareness
Smart control
Smart analysis & planning

[1] Table 004-0243: Census of Agriculture, farms reporting technologies used on the operation in the year prior to the census.

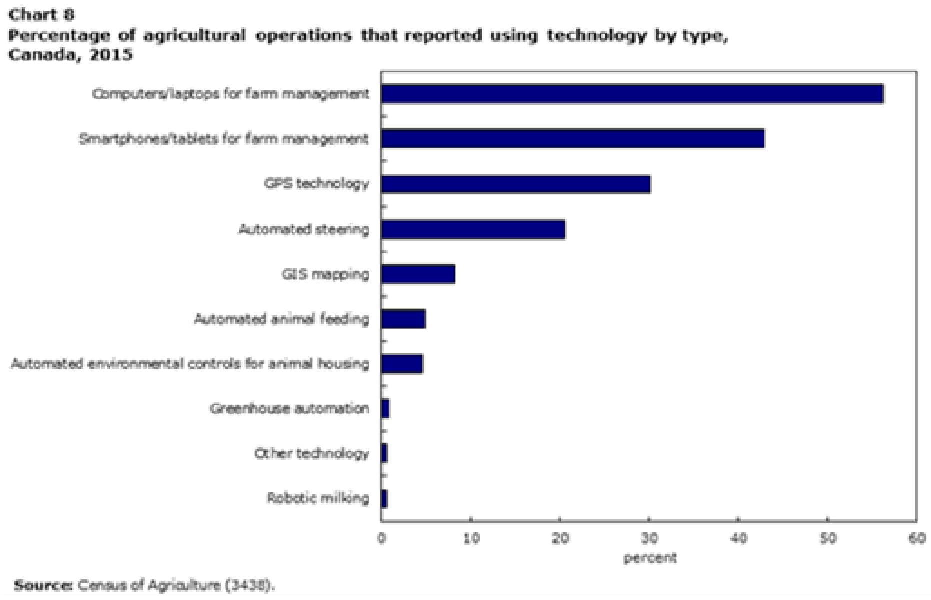


Figure 9: State of Future Smart Agriculture Using IoT

4.1.2 Problem Statement

As depicted in Figure 9, a large-scale agriculture operation is composed of multiple streams, each very different from the other, and all highly automated. This case study highlights how the inclusion of microservices will optimize and enhance smart farming IoT solutions. It will not define an automation solution using IoT; many of these already exist and are quite pervasive. Some of the aspects of smart farming addressed by IoT are:

- The need for constant monitoring, correction, and feedback
- The impact at each stage by many, very dynamic changes in climatic conditions (weather, soil, moisture, heat, length of day, and insects, to name a few)
- The long duration of the crop lifecycle, over several months
- The need for both predictive and proactive actions – in most cases it is not possible to make quick changes
- The requirement for system security
- The need for the system to be capable of vast physical and geographic distribution, including many remote sites

4.1.3 Solution

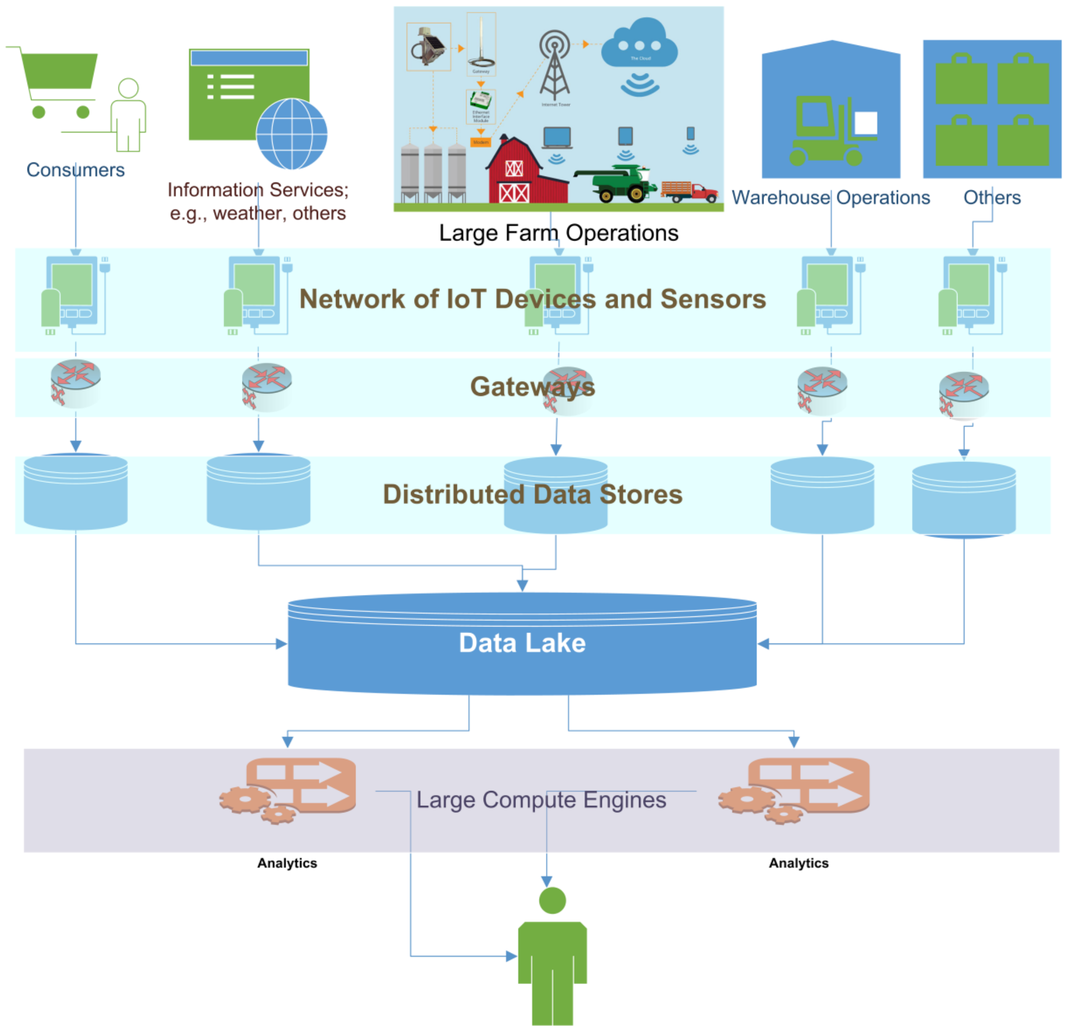


Figure 10: Typical Architecture of an IoT Solution in Farming

Today’s large and small-scale agricultural operations are already enriched with a significant number of IoT solutions that collect a deluge of data. This is useful, and with the current data analytics capabilities, equips the farmers with information that empowers their decisions.

The next challenge is to make this system smart, so it can proactively and automatically execute and provide recommendations where possible. MSA is a good fit to achieve this.

4.1.4 The Role of MSAs

The introduction of microservices into the IoT solution enables this to become a proactive and responsive smart farming solution. This includes many of the benefits discussed in Areas of Synergy and Benefits of MSA to IoT (on page 3):

- Heterogeneous networks
- Telemetry ingestion

- Device provisioning and management
- Status and notifications
- Resiliency (design for Failure)
- Evolution at the atomic level
- Decentralized governance and data management

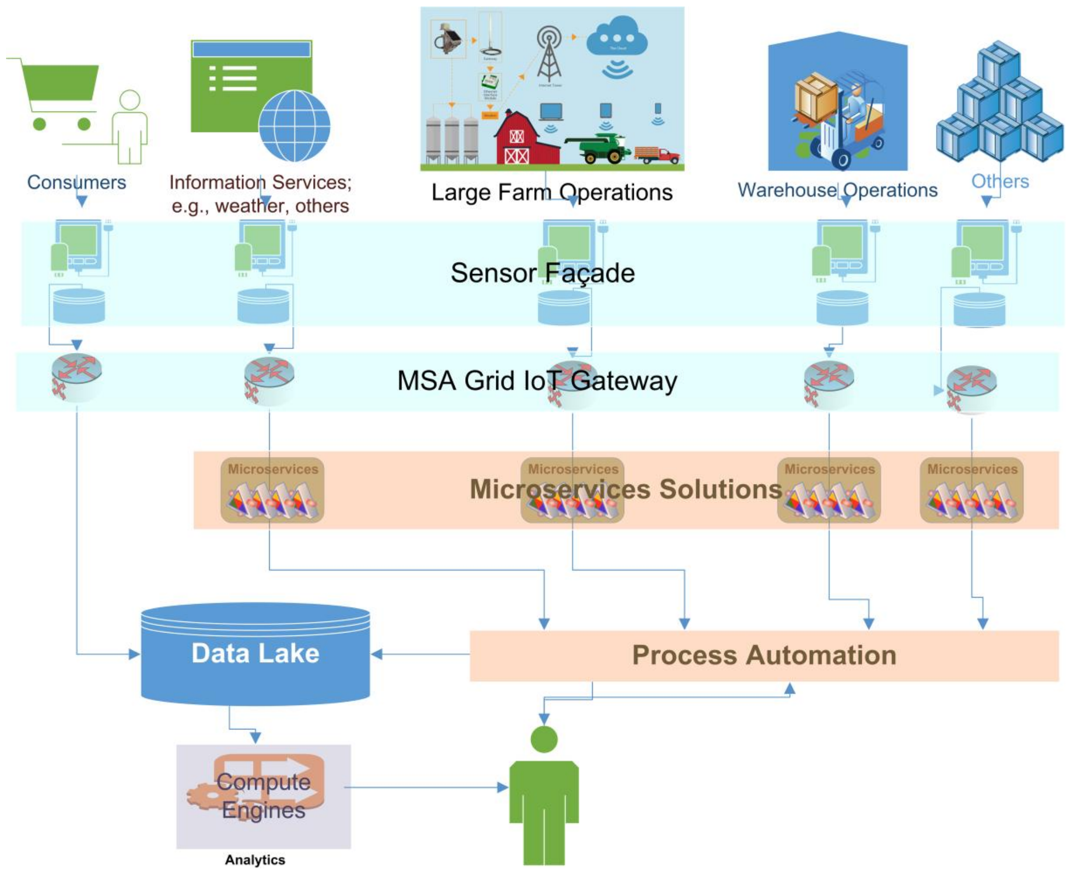


Figure 11: The Introduction of MSA to Create a Smart Farming Solution

Microservices are a set of applications that are highly parallelized and distributed, each performing an atomic function. This is synonymous with an IoT façade, which consists of a very large number of sensors and devices, each typically performing a single function. Hence, marrying the two results in an overall solution that is highly distributed can handle a much larger number of inputs and small data packets, and is highly scalable in both function and performance. It enables the processing of the atomic pieces of information in close proximity to the IoT devices in near real-time.

4.1.5 Recommendations and Suggestions

Although there is great synergy between IoT and microservices, we recommend a healthy dose of scepticism and caution in determining the right fit. If there is a need for processing data and events from each individual IoT device, then a microservice is a great way to optimally achieve this. If the devices are just collecting data, and no immediate action or reaction is needed, then using a microservice to simply collect and propagate this data may not be the optimal choice, as shown in Figure 11.

4.2 Case Study 2: Regional Power Management

4.2.1 Problem Statement

An operator of a large number of facilities distributed across the country is billed for electricity in each region. The amount billed is based on both the consumption in kilowatt hours and the peak value of use at each facility any time during the calendar month. The operator has the ability to reduce the power consumption of a facility for short periods of time by reducing the lighting, adjusting the heating system by a few degrees, reducing air-handling volumes, or disabling other optional systems. Each facility has a number of sensors monitoring power consumption for various systems including temperature and humidity throughout the facility, and other factors that impact the power use. Several regional power companies have offered reduced rates if the peak reduction across facilities within the region can be coordinated with power company consumption peaks.

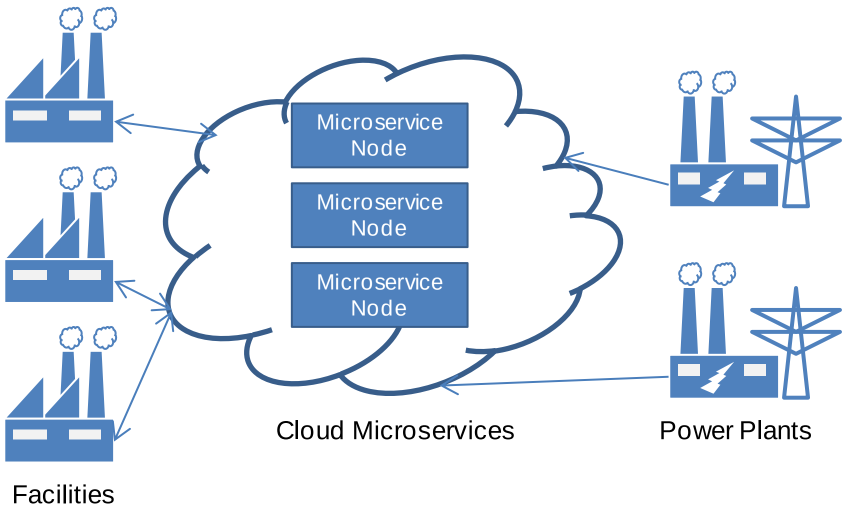


Figure 12: The Use of MSA to Monitor Power Demand and Supply for Regional Power Management

4.2.2 Task

The assignment was to create services and operations to monitor both the peak consumption of power globally within a region as provided by the power companies, and power consumption across facilities within the region, and then start to apply power load shedding strategies to facilities as regional peaks are recognized.

4.2.3 Action

Due to the cost-sensitive nature and need for robustness, it was decided to implement the solution using an MSA. Reading of individual sensors was aggregated from the facilities, and readings of facilities within a region were aggregated (Aggregation pattern) so the loss of data from any sensor or facility had little impact on the resulting calculations or microservices. The power companies sent notifications about their regional power use to a microservice endpoint node which distributes the notification to all of the microservice nodes (Multicast pattern) providing a reliable means of tracking when regional power reduction was needed. When the regional power companies experience a spike in consumption, the facilities may start load shedding (Control Aggregator pattern) even though any facility in the region may not be nearing a peak itself.

4.2.4 Result

The ability to work cooperatively with the power companies, helping them to manage their consumption peaks by reducing demand at all of the facilities in the region, was a big win for both the power companies and the facilities operator, saving over a million dollars in energy expenses annually. Using an MSA combined with the existing IoT monitoring and control framework already in place allowed the implementation to be both robust and cost-effective.

4.3 Case Study 3: Using the IoT Network to Get an Accurate and Deep Understanding of Water Quality

(Hypothetical Study to Demonstrate the Value of MSA for IoT)

There are many implementations of sensor-based IoT solutions designed to monitor various characteristics that determine the quality of water. These include waste water treatment plants, drinking water treatment plants, industrial water treatment plants, and systems used by government agencies to maintain the water quality in lakes and rivers. There is an ongoing effort to optimize the overall cost of maintaining the quality of water, by utilizing the IoT sensor data in more effective and efficient ways.

4.3.1 Problem Statement

In this situation:

- How can we make the sensor data available to multiple channels and platforms?
- How can we perform specific complex analytics on the properties of the water (for example, PH level, harmful chemicals, solid particulates, mineral percentage, and so on) and to deliver validated data to take corrective action?

The solution should feed the filtered data to specialized data analytics platforms, which ultimately should help in optimizing the overall cost of maintaining water quality and ensuring compliance.

4.3.2 The Role of MSAs

Microservices can be leveraged to consume events, or directly retrieve the filtered data from a repository, based on the pre-configured rules for the data consumer such as prediction analysis platform, pattern recognition tools, time series analysis platform, decision and visualization platform, etc. The microservices dedicated for each analytics platform can fetch the context-based data from the data lake, process the unstructured data to the required format, and then direct it to one or more analytics platforms. Each dedicated platform performs the required complex analytics and generates various visuals, reports, and recommendations, and feeds these to downstream systems to be actioned. An MSA provides a scalable solution that delivers more accurate, pre-processed data. This ultimately minimizes the overall cost to maintain the quality of water by informing the use of resources such as chemicals, power, machineries, and work force.

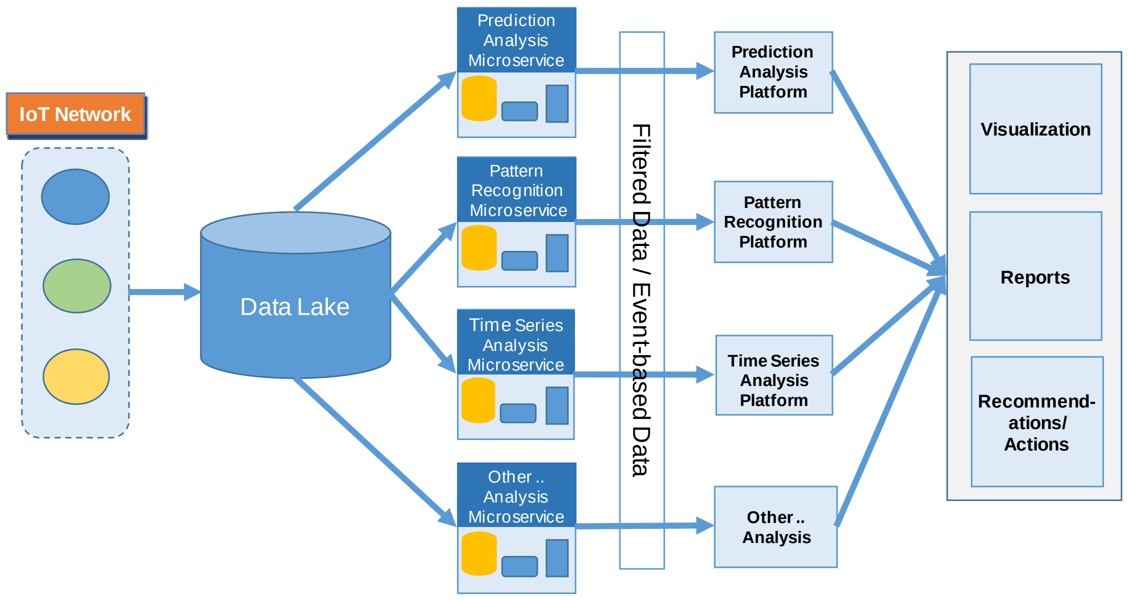

IoT Network
Data Lake
Prediction Analysis Microservice
Pattern Recognition Microservice
Time Series Analysis Microservice
Other .. Analysis Microservice
Filtered Data / Event-based Data
Prediction Analysis Platform
Pattern Recognition Platform
Time Series Analysis Platform
Other .. Analysis
Visualization
Reports
Recommend-ations/ Actions

Figure 13: The Use of MSAs to Monitor and Manage Water Quality

4.4 Case Study 4: Leveraging MSA to Create Multi-Purpose IoT Devices

Traditional IoT-like devices are designed to support a pre-defined set of capabilities, by leveraging highly optimized hardware and software components, that are designed to support specific functions, such as video processing or processing sensor data. This kind of device is still the most popular, because of its high performance and energy efficiency. By contrast, modern IoT devices are equipped with high-speed processors and sufficient memory to allow most of the computations to be performed without dedicated hardware accelerators. The flexibility provided by modern IoT devices makes it possible for the practical function of the device to be decided, or changed, at a late stage, possibly even when the device is in use.

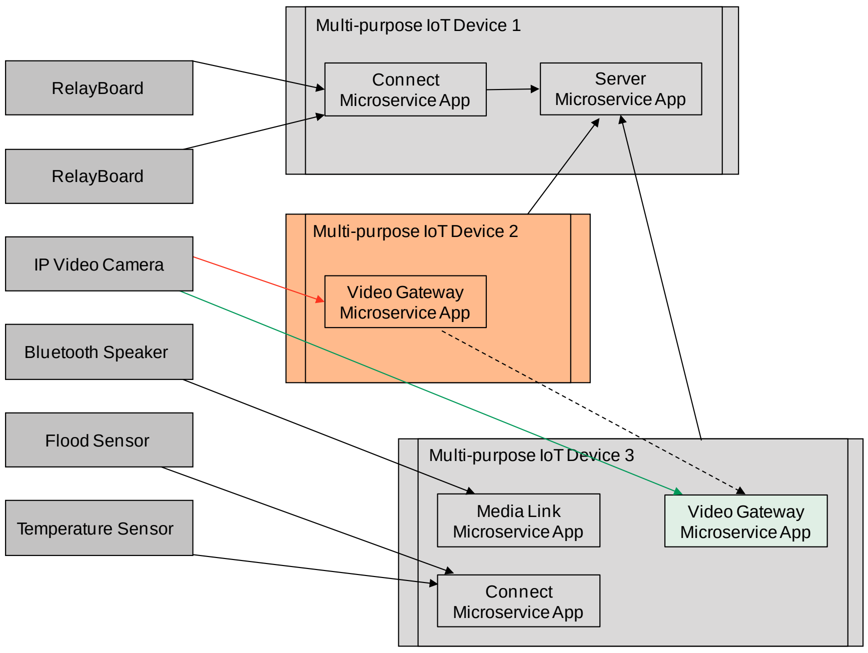
4.4.1 Problem Statement

The distributed computing power of the IoT devices connected on a local network should be capable of providing load-balancing and fail-over capability in case of the failure of a single device.

4.4.2 The Role of MSAs

In Figure 14, modern IoT devices are running platform-management software to handle both the communication between the devices, and the applications they are running. Platform-management software is also responsible for the replication of the data between the devices, based on the multi-master replication model. Replicated data contains mainly configuration information, but also selected data collected by the devices.

Here, the microservice application called "Video Gateway" is migrated from the broken Device 2 to the functioning and still available Device 3. The decision of where to create a new instance of "Video Gateway" is made by the platform management software, based on statistics collected from the devices; the device with lowest utilization will be selected as a target for the new instance.



Multi-purpose IoT Device 1
RelayBoard
Connect Microservice App
Server Microservice App
RelayBoard
Multi-purpose IoT Device 2
IP Video Camera
Video Gateway Microservice App
Bluetooth Speaker
Flood Sensor
Multi-purpose IoT Device 3
Media Link Microservice App
Video Gateway Microservice App
Temperature Sensor
Connect Microservice App

Figure 14: Using MSA to Manage Routing Choreography to Multi-Purpose Devices

4.4.3 Conclusion

The MSA creates a framework for optimal usage of the IoT devices on the local network, supporting both load balancing and fail-over of the microservice applications between the devices.

4.5 Case Study 5: Simple Greenhouse Monitoring Solution

The temperature in a greenhouse varies widely between day and night, and also between summer and winter. Growing conditions are improved when extreme hot and cold temperatures are avoided. This can be achieved by a Subterranean Heating and Cooling System (SHCS) in which the air in the greenhouse is pumped through underground tubes, cooling it when it is very hot, and warming it when it is cold. The cooling and heating is mainly achieved by the condensation and evaporation of water vapor in the air. The system performance is assessed by monitoring the airflow through the underground pipes, and the temperature and relative humidity of the air entering and leaving the pipes. This requires a simple IoT data collection system.

4.5.1 Problem Statement

SHCS greenhouses are not yet so common that cheap monitoring systems are available as consumer products. Creation of such a system for a home greenhouse requires use of cheap and freely available components and should not include extensive software development. The system should nevertheless be reasonably robust, and ideally be easily extensible to perform other greenhouse monitoring and control functions.

4.5.2 The Role of MSAs

An MSA provides effective software organization principles that enable robustness and flexibility without requiring a big investment in software infrastructure.

Figure 15 shows an implementation of the solution that includes most of the required components. Still to be implemented is a Humidity Reader microservice, which will be similar in principle to the Temperature Reader microservice. The components are shown positioned within The Open Group SOA Reference Architecture.

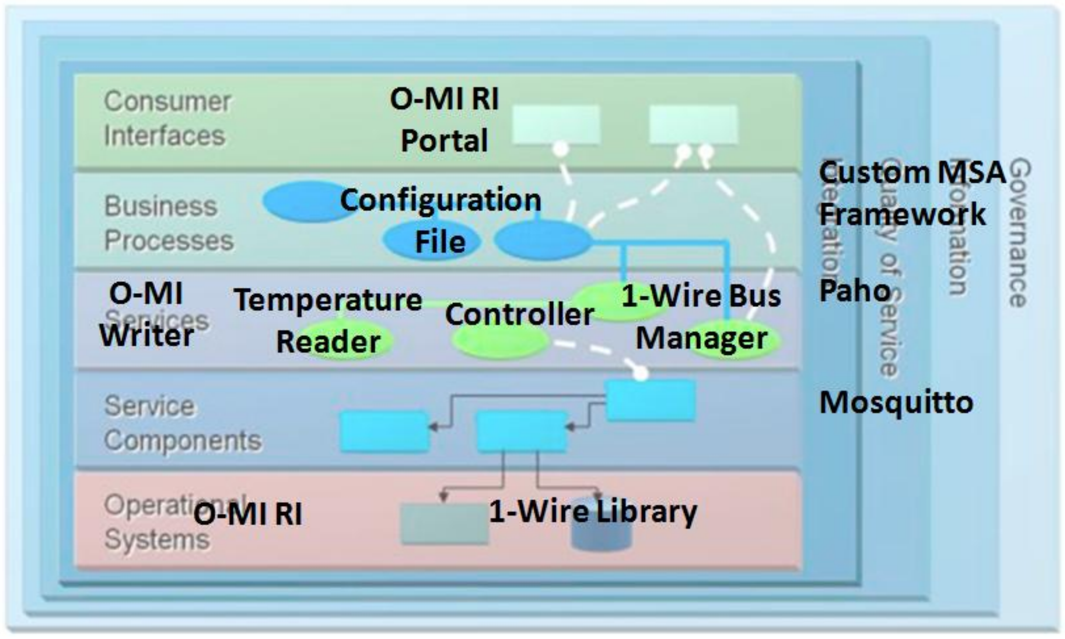


Figure 15: MSA-IoT Solution Architecture for Greenhouse Monitoring

The temperature and humidity values are read by sensors using the 1-Wire interface. 1-Wire is a widely-used proprietary interface by which readings are made available as digital values over a serial bus. (Other interfaces are available; for example, I2C, an open protocol with similar functionality, could be used in place of 1-Wire.) This serial bus is connected, via an adaptor, to a USB interface in a Raspberry Pi™ computer, which runs the software components shown in Figure 15.

The 1-Wire Library is a free software library. It is used by the 1-Wire Bus Manager microservice to drive the 1-Wire bus and take readings from the devices connected to it. The controller microservice requests the 1-Wire Bus Manager to initialize the bus. The Temperature Reader microservice regularly requests the 1-Wire Bus Manager to obtain temperature readings. A humidity reader microservice that regularly requests the 1-Wire Bus Manager to obtain relative humidity readings is to be added.

The 1-Wire Bus Manager sends the readings to the O-MI Writer microservice. This writes the readings to the O-MI Reference Implementation (O-MI RI), a freely available implementation of

The Open Group Open Messaging Interface (O-MI) standard. The O-MI RI stores the readings and makes them available for retrieval over the web using the O-MI protocol.

Communications between the microservices use the MQ Telemetry Transport (MQTT) protocol, supported by the open source Mosquitto server. The microservices use the Eclipse® Paho MQTT interface implementation.

The microservices are supported by a simple custom-built framework that deploys and configures them according to a text configuration file. The framework monitors the microservices and will restart any that stop working.

4.5.3 Conclusion

A microservices approach enables the development of simple, maintainable, and extensible solutions for IoT applications.

5 Recommendations and Conclusions

This Guide has provided an overview of how Microservices Architecture (MSA) can be combined with an Internet of Things (IoT) network to best exploit their potential strengths and provide maximum value to the users of the network. We have described the many areas of synergy between MSA and IoT networks, and attempted to enumerate the benefits of applying MSA to an IoT network. Some of these areas are fairly obvious. IoT networks will tend to be:

- Heterogeneous
- Composed of many low-level sensors
- Widely distributed
- Evolving continuously at the atomic level

These characteristics are well-served by application of an MSA since the independence, distributed governance, and resiliency inherent in an MSA will mesh well with the requirements of an IoT network.

It is important to note that, like any other architectural solution, an MSA is not a one-size-fits-all answer to all IoT applications. A key factor in determining the suitability of an MSA is the size of the network.

In applications where the loss of one (or a small number) of hardware sensors will result in a significant or complete loss of network functionality, the use of an MSA is not necessarily going to provide significant advantages over some other architectural approach, such as using simple services rather than a microservice implementation. In fact, the dependencies introduced in this case probably violate the key independence characteristic of an MSA.

However, in applications where the loss of individual sensors does not affect the overall viability of the network, as illustrated in the case studies, the inherent characteristics provided by an MSA, made possible through use of the patterns described in this Guide, will be an excellent fit for the application, providing a highly resilient, easily upgraded, and modified architecture, which will best take advantage of the unique characteristics provided by the expanding capabilities of IoT networks. This will allow for the development of novel applications which will provide new capabilities for the new era of ubiquitous, distributed computing and sensors.

Glossary

API

An Application Programming Interface (API) is a set of functions, methods, and protocols that define how one computer program can request the services of another.

Architecture

The fundamental organization of a system embodied in its components, their relationships to each other and to the environment, and the principles guiding its design and evolution.

Internet of Things

An infrastructure of interconnected objects, people, systems, and information resources together with the intelligent services to allow them to process information of the physical and the virtual world and react.

Microservice

An individual microservice is a service that is implemented with a single purpose that is closely aligned to a specific business capability, self-contained, and independent of other instances and services. The microservice is the primary architectural building block of the Microservices Architecture.

Microservices Architecture

An architectural style that structures an application or system as a set of loosely coupled, independent, and self-contained services, which align closely with a business capability.

Monolithic Application

A self-contained software application composed of functionally distinct components that tend to be tightly coupled and interdependent.

Resiliency

The ability of an application or system to react to problems in one of its components and continue to operate and provide its defined capability.

Scalability

The characteristic of a system, network, or process to handle an increasing amount of work.

Service-Oriented Architecture

Service-Oriented Architecture (SOA) is an architectural style in software design in which application components provide services to other components via a communications protocol, typically over a network. The principles of service-orientation are independent of any vendor, product, or technology.

Web-Oriented Architecture

Web-Oriented Architecture (WOA) is a software architecture style that extends SOA to web-based applications.

Index