OpenFog Reference Architecture for Fog Computing

Produced by the OpenFog Consortium Architecture Working Group
www.OpenFogConsortium.org

February 2017

Use of this Document

Published February 2017.

Acknowledgements

The OpenFog Reference Architecture is the product of the OpenFog Architecture Workgroup, co-chaired by Charles Byers (Cisco) and Robert Swanson (Intel). It represents the collaborative work of the global membership of the OpenFog Consortium. We wish to thank these organizations for contributing to this work and to the advancement of fog computing technology, research and innovation: Aalto University; ABBALab; Arizona State University; ARM; AT&T; Caltech; Cisco; Dell; FogHorn Systems; Fujitsu; GE Digital; Hitachi; Foxconn; Indian Institute of Technology; Industrial Technology Research Institute; Institute for Information Industry; Institute of Network Coding; The Chinese University of Hong Kong; Intel; Internet Initiative Japan Inc.; ITOCHU techno-Solutions Corporation; Kii; LGS Innovations; MARSEC; Microsoft; Mitsubishi Electric Corporation; National Chiao Tung University; National Taiwan University; Nebbiolo Technologies; NEC Corporation; NGD Systems; NTT Communications; OSIsoft; Princeton University; PrismTech; Real-Time Innovations; relayr; SAKURA Internet; Schneider Electric; Shanghai Institute of Microsystem and Information Technology; ShanghaiTech University; Singapore University of Technology and Design; SRC Inc.; Stichting imec Nederland; The Chinese University of Hong Kong; Toshiba; Technische Universität Dresden; TTTech; University of Colorado Boulder; University of Georgia; University of Pisa; University of Southern California; Vanderbilt University; Wayne State University.

OpenFog Overview

Digital innovation from the Internet of Things (IoT), Artificial Intelligence, Virtual Reality, Tactile Internet and 5G applications is transforming the way we work, commute, shop and play. Data from newly-connected factories, homes, communities, cars, hospitals and more is expected to grow from 1.1 zettabytes (or 89 exabytes) per year in 2016 to 2.3 zettabytes (or 194 exabytes) per year by 2020.[1] Current "cloud-only" architectures cannot keep up with the volume and velocity of this data across the network, thereby reducing the value that can be created and captured from these investments.

Fog computing provides the missing link in the cloud-to-thing continuum. Fog architectures selectively move compute, storage, communication, control, and decision making closer to the network edge where data is being generated in order solve the limitations in current infrastructure to enable mission-critical, data-dense use cases.

Fog computing is a:

> ***A horizontal, system-level architecture that distributes computing, storage, control and networking functions closer to the users along a cloud-to-thing continuum.***

Fog computing is an extension of the traditional cloud-based computing model where implementations of the architecture can reside in multiple layers of a network's topology. However, all the benefits of cloud should be preserved with these extensions to fog, including containerization, virtualization, orchestration, manageability, and efficiency. In many cases, fog computing works with cloud. Pillars, which are common themes of the OpenFog reference architecture include security, scalability, openness, autonomy, RAS (reliability, availability and serviceability), agility, hierarchy, and programmability. In addition to the pillars, we describe the roles of each stakeholder in the fog value chain from silicon creator to the Operating System and application developer through a composite architectural description

Fog computing also is often erroneously called edge computing, but there are key differences. Fog works with the cloud, whereas edge is defined by the exclusion of cloud. Fog is hierarchical, where edge tends to be limited to a small number of layers. In additional to computation, fog also addresses networking, storage, control and acceleration.

The OpenFog Consortium was formed on the principle that an open fog computing architecture is necessary in today's increasingly connected world. Through an independently run open membership ecosystem of industry, end users and universities, we can apply a broad coalition of knowledge to these technical and market challenges. We believe that proprietary or single vendor solutions can limit supplier diversity and ecosystems, resulting in a detrimental impact on market adoption, system cost, quality and innovation. It is our intent to ensure the OpenFog reference architecture results in fully interoperable and secure systems, supported by a vibrant supplier ecosystem.

Contents

Figures

1 About Fog Computing and the Consortium

1.1 OpenFog Reference Architecture Overview

The OpenFog Consortium was founded by ARM, Cisco, Dell, Intel, Microsoft and Princeton University in November 2015. Through its global membership of leading technology & networking players, fog computing entrepreneurs and university researchers, the Consortium is helping to enable game-changing innovation enabled by fog computing through an open architectural framework.

We are guided by the OpenFog Board of Directors and the OpenFog Technical Committee. The technical committee is the technical governing body of all of the working groups of the Consortium. The chair of this group is elected by a vote of the OpenFog Board of Directors and reports directly to the board.

The OpenFog Reference Architecture (OpenFog RA) is intended to help business leaders, software developers, silicon architects, and system designers create and maintain the hardware, software and system elements necessary for fog computing.

Many different technical workgroups in the Consortium are responsible for the different aspects of this reference architecture, including: Communications, Software-Infrastructure and Security. The Architecture Framework workgroup is tasked with the creation and maintenance of this document and other technical publications of the Consortium. All new technical topics requiring investigation are assigned to this group. The charter of each group is managed and approved by the technical committee and Board of Directors.

For further information on these groups or how to participate, please reference www.openfogconsortium.org.

2 Areas of Opportunity

2.1 OpenFog Reference Architecture Content

The OpenFog RA is part of a suite of documents under construction by the OpenFog Consortium and our technical liaisons partners. It is a medium- to high-level view of system architectures for fog nodes and networks. Future documents will provide lower-level details, including formal, enumerated requirements that will form the basis of quantitative testbeds and the specified interoperability of fog elements. Future documents will also refine the use cases described in Chapter 3.

The OpenFog RA is divided into the following chapters:

- **Chapter 2** describes the OpenFog Consortium's mission and plans to accelerate fog computing. It also provides an overview of the OpenFog RA.
- **Chapter 3** presents some use cases where we see fog computing emerging. This list will grow and evolve as the OpenFog RA is refined.
- **Chapter 4** describes the pillars of the OpenFog architecture. These are the guiding principles for the OpenFog RA.
- **Chapter 5** provides an in-depth look at the full OpenFog RA. This section shows the abstract Architectural Description (AD) for the OpenFog RA.
- **Chapter 6** starts the conversation on adherence to the OpenFog architecture. It is our intention to drive standardization across the various interfaces.
- **Chapter 7** shows the abstract OpenFog RA applied to various use cases. This will further clarify each aspect of the OpenFog RA and what needs to be done for a successful implementation.
- **Chapter 8** contains some of the open areas of fog computing and new opportunities for research. OpenFog member companies and academic organizations are continually enhancing and refining the OpenFog RA. This section will help advance the overall architecture over time.

2.2 The Internet of Things, Cloud and the OpenFog RA

The Internet of Things (IoT) is driving business transformation by connecting everyday objects and devices to one another and to cloud-hosted services.

Current deployment models emphasize mandatory cloud connectivity; however, this is not feasible in many real-world situations. These are two of the primary issues with connecting edge devices to the cloud for all services:

- Connected devices are creating data at an exponentially growing rate, which will drive performance and network congestion challenges at the edge of infrastructure.
- There are performance, security, bandwidth, reliability, and many other concerns that make cloud-only solutions impractical for many use cases.

Unfettered cloud-only architectural approaches cannot sustain the projected data velocity and volume requirements of the IoT. To sustain IoT momentum, the OpenFog Consortium is defining an architecture to address infrastructure and connectivity challenges by emphasizing information processing and intelligence at the logical edge. This approach is called fog computing.

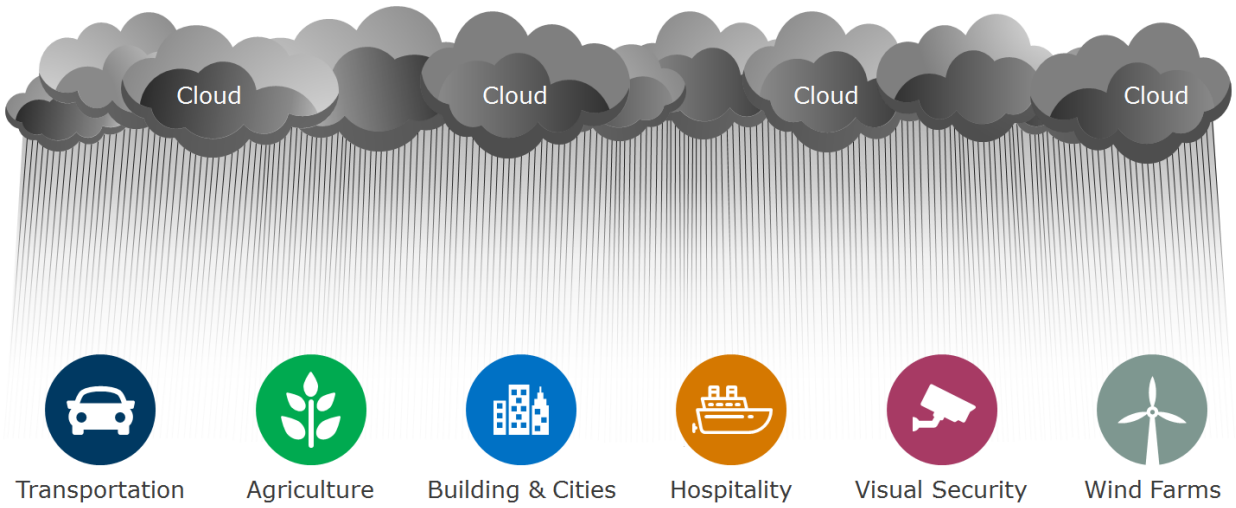


Figure 1 Unfettered Cloud Computing

While the cloud itself may play a vital role in many deployments, fog computing represents a shift from traditional closed systems and a reliance on cloud-only focused models. Fog computing is complementary to, and an extension of, traditional cloud-based models.

The fog computing model moves computation from the cloud closer the edge, and potentially right up to the IoT sensors and actuators. The computational, networking, storage and acceleration elements of this new model are known as fog nodes. These are not completely fixed to the physical edge, but should be seen as fluid system of connectivity.

2.3 OpenFog and Other Consortia

The OpenFog Consortium invites open participation from across industry, academia and non-profit organizations that have an interest in the emerging IoT landscape. The mission of the OpenFog Consortium is complementary to other IoT and technology industry alliance groups including the Industrial Internet Consortium (IIC), ETSI-MEC (Mobile Edge Computing), OPC-UA, Open Connectivity Foundation (OCF), OpenNFV, and many others. To avoid duplication of effort and market confusion, the OpenFog Consortium will de-emphasize efforts to optimize for some of these application spaces, and instead focus on optimally serving vertical markets not addressed by other initiatives. In the longer term, through liaisons with these and other bodies, we will drive more convergence across the IoT industry under a common view of edge and fog architectures.

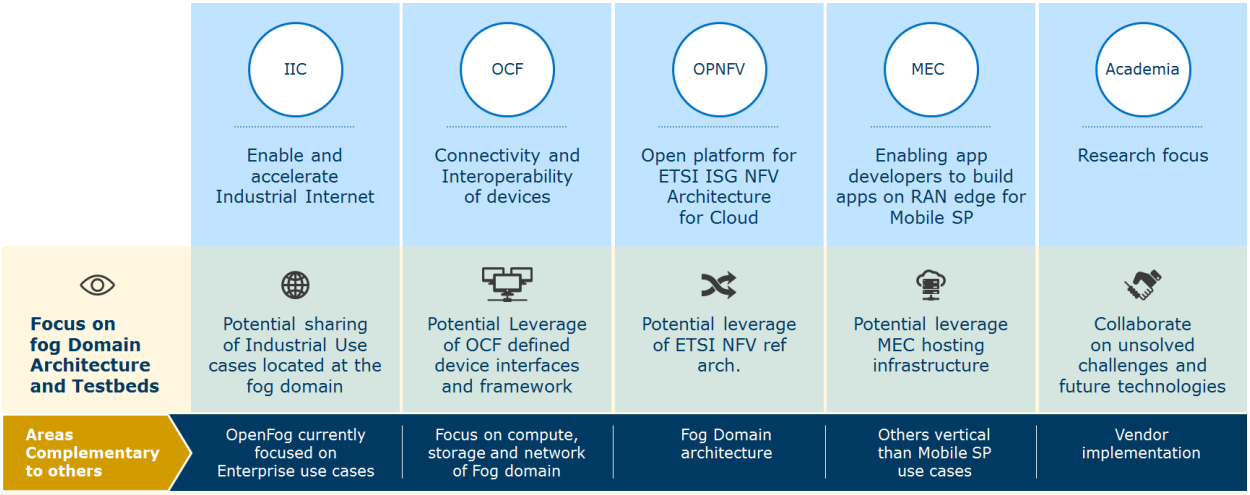


Figure 2 OpenFog Consortium and Other Consortia

3 Use Cases for Fog

Fog computing targets cross-cutting concerns like the control of performance, latency and efficiency are also key to the success of fog networks. Cloud and fog computing are on a mutually beneficial, inter-dependent continuum.

Certain functions are naturally more advantageous to carry out in fog nodes, while others are better suited to cloud. The traditional backend cloud will continue to remain an important part of computing systems as fog computing emerges. The segmentation of what tasks go to fog and what goes to the backend cloud are application specific. This segmentation could be planned, but also change dynamically if the network state changes in areas like processor loads, link bandwidths, storage capacities, fault events, security threats, cost targets, etc.

The OpenFog RA enables fog-cloud and fog-fog interfaces. OpenFog architectures offer several unique advantages over other approaches, which we term SCALE:

- **S**ecurity: Additional security to ensure safe, trusted transactions
- **C**ognition: awareness of client-centric objectives to enable autonomy
- **A**gility: rapid innovation and affordable scaling under a common infrastructure
- **L**atency: real-time processing and cyber-physical system control
- **E**fficiency: dynamic pooling of local unused resources from participating end-user devices

A quick use case example to illustrate the value of fog: Consider an oil pipeline with pressure and flow sensors and control valves. One could transport all those sensor readings to the cloud (perhaps using expensive satellite links) analyze the readings in cloud servers to detect abnormal conditions, and send commands back to adjust the positon of the valves.

There are several problems with this scenario: the bandwidth to transport the sensor and actuator data to and from the cloud could cost many thousands of dollars per month; those connections could be susceptible to hackers; it may take several hundred milliseconds to react to an abnormal sensor reading (during which time a major leak could spill significant oil); and if the connection to the cloud is down, or the cloud is overloaded, control is lost.

Now, consider placing a hierarchy of local fog nodes near the pipeline. They can connect to sensors and actuators with inexpensive local networking facilities. Fog nodes can be highly secure, lessening the hacker threat. Fog nodes can react to abnormal conditions in milliseconds, quickly closing valves to greatly reduce the severity of spills. Local control in the fog nodes produces a more robust control system. Moving most of the decision-making functions of this control system to the fog, and only contacting the cloud occasionally to report status or receive commands, creates a superior control system.

This document describes a set of high-level attributes of fog computing that we call the pillars (including some of the fog advantages described in the pipeline control scenario). These include security, scalability, openness, autonomy, reliability, agility, hierarchical organization and programmability. We will discuss all of these pillars in detail later in this document.

Platform as a service (PaaS) is a category of cloud computing services that provides a platform allowing customers to develop, run, and manage web applications without the complexity of building and maintaining the infrastructure typically associated with developing and launching an application. OpenFog RA defines the required infrastructure to enable building Fog as a Service (FaaS) to address certain classes of business challenges. FaaS includes Infrastructure as a Service (IaaS), Platform as a Service (PaaS), Software as a Service (SaaS), and many service constructs specific to fog. The infrastructure and architecture building blocks below show how FaaS may be enabled and will be expanded upon in the reference architecture.

The OpenFog RA describes a generic fog platform that is designed to be applicable to any vertical market or application. This architecture is applicable across many different markets including, but not limited to, transportation, agriculture, smart-cities, smart–buildings, healthcare, hospitality, financial services, and more, providing business value for IoT applications that require real-time decision making, low latency, improved security, and are network-constrained. In this section, we look at a few specific use cases.

3.1 Transportation Scenario: Smart Cars and Traffic Control

In 2016, the average person creates around 650MB of data every day and by 2020, some project that to more than double. However smart autonomous cars will generate multiple terabytes of data every day from the combinations of light detection and ranging (LIDAR), global positioning

systems (GPS), cameras, etc. When the smart car is coupled with intelligent infrastructure, it is clear that a cloud-only model will not work for autonomous transportation, and that a fog computing approach is required. Many of the architecture requirements we describe in smart cars and traffic control also apply to other transportation areas, such as ships/boats, trains, trucks, busses and drones. In this section we will highlight the opportunity for fog computing for smart cars and traffic control, and explain how the requirements are addressed by the OpenFog RA. The figure below is an overview of an intelligent highway application of the OpenFog RA.

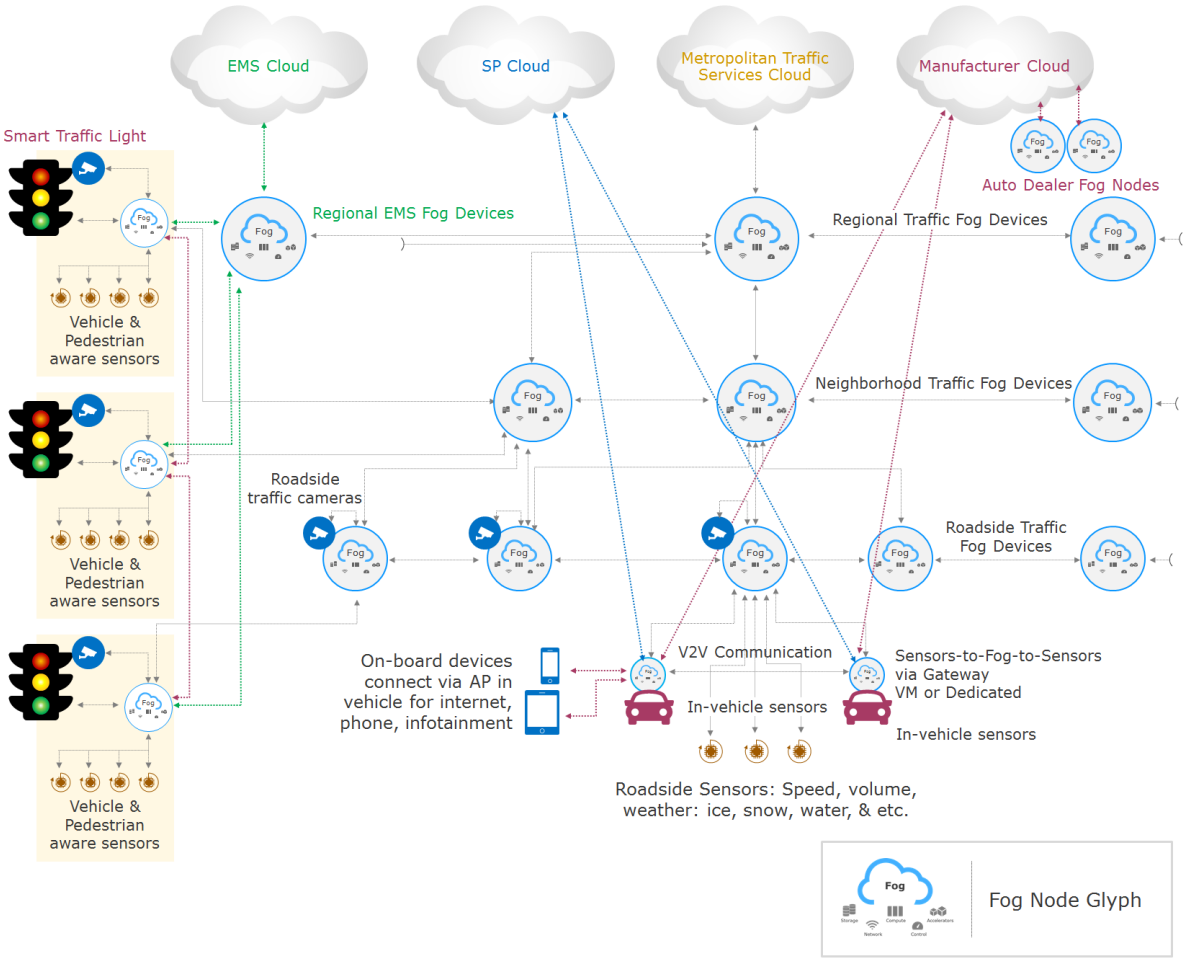


Figure 3 OpenFog Transportation: Smart Car and Traffic Control System

The smart car and traffic control use case presents an opportunity to examine a fog environment containing a rich set of interactions among multiple fog domains as well as multiple cloud domains. Among other things, this use case demonstrates:

- A rich set of interactions among multiple fog domains, as well as multiple cloud domains, including Element Management Systems (EMS), service provider (SP), metro traffic services, and system manufacturer clouds.
- Mobile fog nodes supporting vehicle-to-vehicle (V2V), vehicle-to-infrastructure (V2I), and vehicle-to-x (V2X) interactions.
- Multiple fog networks owned and operated by different authorities providing similar (and different) functionality
- Multi-tenancy across fog nodes will also be a burgeoning opportunity to consolidate multiple fog networks, improving efficiency.
- Both private and public fog and cloud networks used by a single end point device.

This use case shows also shows the hierarchical and distributed advantages of a fog architecture. As shown in the figure above, the system includes several types of sensors (and actuators) that we refer to as "things."

Things include roadside sensors (infrastructure) and on-vehicle sensors. These sensors provide data so that the various systems (lights, cars, etc.) can carry out their given functions (e.g. vehicle driving autonomously). Smart transportation systems also manage the actuators that control parts of the infrastructure, such as traffic signals, gates, and digital signs.

The vehicles connect to the cloud and a hierarchy of fog nodes that service the autonomous vehicle or traffic control systems.

Fog computing nodes in the vehicle

In this use case example, the vehicle is a mobile fog node that communicates with other fog nodes as they become available, an example of V2I interactions. The mobile fog node must also be capable of performing all required in-vehicle operations autonomously in circumstances where it cannot connect to other fog nodes or the cloud.

In-vehicle fog nodes provide services including infotainment, advanced driver assistance systems (ADAS), autonomous driving, collision avoidance, navigation, etc. Several different networking technologies, including Dedicated Short Range Communications (DSRC), cellular (e.g. 3G, LTE, 5G, etc.) and other networking technologies securely connect the vehicles to each other and the infrastructure.

The Transportation Fog Network

The Transportation Fog Network is comprised of a three-level hierarchy of fog nodes.

The first level of the hierarchy is the infrastructure fog nodes, or roadside fog nodes. At this level, the roadside fog sensors collect data from other devices such as roadside cameras. The fog nodes perform some local analysis for local action, such as alerting the vehicle about poor road conditions, triggering autonomous response to slow down, and perform some autonomous functions, even if connections to higher layers are unavailable. Data from the first level of interactions is aggregated and sent up the fog hierarchy to the second and third levels of the hierarchy—neighborhood and regional fog nodes—for further analysis and distribution. Some of the data may also be distributed east-west to other infrastructure nodes for their use. Typically, each fog layer in the hierarchy will provide additional processing, storage, and network capabilities in service of the vertical application at their level of the hierarchy. For example, higher level layers provide additional processing to provide data analytics or large storage capacities.

Traffic control systems

Traffic control fog nodes may receive input from other sources, such as smart traffic light systems, municipal managers, and cloud-based systems. Data flows between the traffic control system, infrastructure fog nodes and vehicles in all directions, insuring all levels of the hierarchy have the data and control capabilities they need.

The goal of the OpenFog RA for smart cars and traffic control is to ensure an open, secure, distributed, and scalable architecture that optimizes real time capabilities within a multi-supplier ecosystem. The transportation example shows a complex system of autonomous things and infrastructure generating massive amounts of data. We believe that this use case highlights the need for fog computing to enable safe and effective operations in IoT, 5G, AI and other advanced scenarios.

3.2 Visual Security and Surveillance Scenario

Surveillance and security cameras are being deployed worldwide. These cameras are used to ensure security of materials, people, and places. In addition, these cameras have the ability to generate a massive amount of data, which can exceed terabytes per day for a single camera. Traditional cloud models that were deployed for low-resolution cameras aren’t scalable with the 1080p and 4K cameras because of the sheer availability and/or cost of network transport. Additionally, decisions on

security need to be made at the camera or installation location and cannot be made solely in the cloud. Machine vision is also a prime candidate for accelerators and dynamic updating of various algorithms in both hardware and software. These cameras are capturing images of people, places, or things and are tightly coupled to decision–making, which requires a heightened level of security of the camera's software and hardware assets.

Smart cities, smart homes, retail stores, public transportation, manufacturing and enterprises increasingly rely on camera sensors to secure people, identify unauthorized access, and increase safety, reliability and efficiency. The sheer bandwidth of visual (and other sensor) data being collected over a large-scale network makes it impractical to transport all the data to the cloud to obtain real-time insights. A particularly demanding application space is surveillance of high value installations with many people and objects moving through them. Controlling a large network of cameras in an airport is a good example of such an application. This use case will be studied in greater detail later in this document to illustrate applying the architecture to a concrete application.

City-scale deployments that include placing cameras on traffic lights and other camera deployments in remote areas don't have high-bandwidth connectivity to the cloud to upload the collected video, even if the video could fit over the network infrastructure. Real-time monitoring and detection of anomalies (intruders into a building, the fall of an elderly citizen, the misfiring of a piece of manufacturing equipment) pose strict low latency requirements on surveillance systems; timeliness is important from the standpoint of both detection and response.

Additionally, privacy concerns must be addressed when using a camera as a sensor that collects image data so that the images do not reveal a person's identity or reveal confidential contextual information (e.g. intellectual property in a manufacturing plant) to any unauthorized parties. OpenFog RA deployments provide the opportunity to build real-time, latency-sensitive distributed surveillance systems that maintain privacy. It leverages fog nodes to intelligently partition video processing between fog nodes co-located with cameras and the cloud so as to enable real-time tracking, anomaly detection, and insights from data collected over long time intervals. Video analytics algorithms can be located on fog nodes close to the cameras, and take advantage of the heterogeneous processor capability of fog, running parts of the video analytics algorithm on conventional processors or accelerators.

The visual security for airports use case requires all of these fog capabilities to meet its performance, reliability, security and efficiency goals. This includes vehicle detection, people detection, smart retail, and other areas where machine vision via video analytics is important for fog computing. Please see the detailed analysis in chapter 7 for many more details including an application of the OpenFog RA.

3.3 Smart Cities Scenario

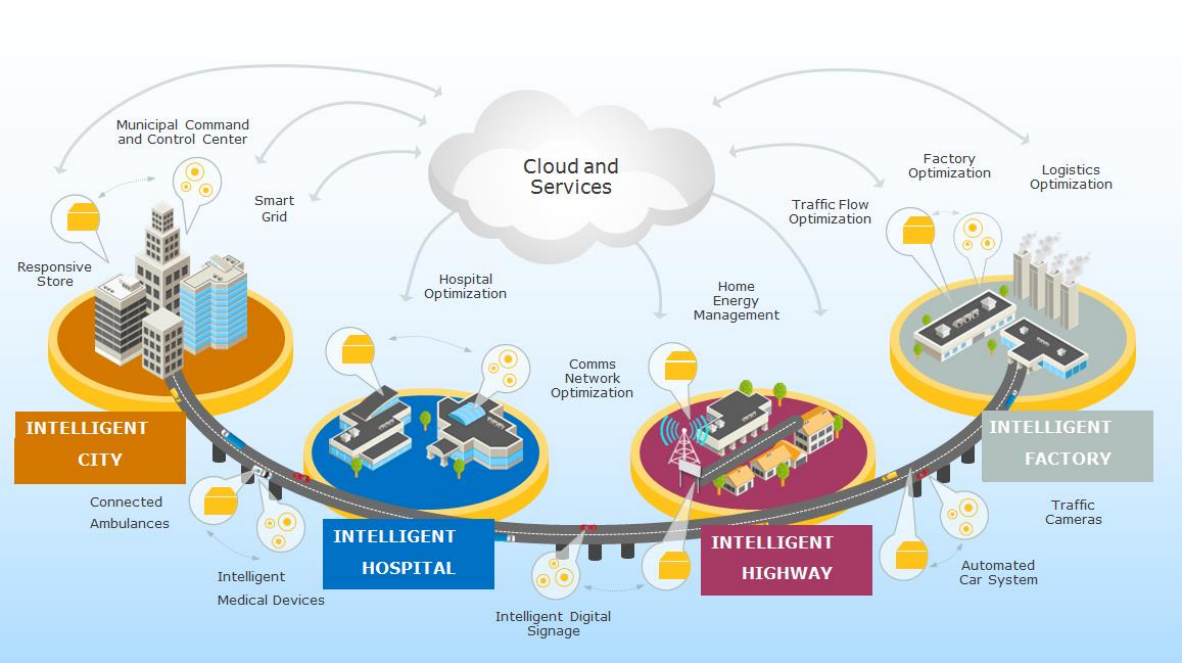


Figure 4 Opportunities for Smart Cities

Smart cities are using technology to deal with many challenges, including traffic congestion, public safety, energy consumption, sanitation, and public internet connectivity.

The OpenFog RA enables greater efficiency and economic realities of smart city operations. The figure above illustrates the various aspects where fog computing can impact smart cities including but not limited to:

- Intelligent city with smart parking, shopping, and infrastructure.
- Intelligent hospitals linking all aspects for greater patient care and healthcare delivery.
- Intelligent highway systems to optimize utilization of infrastructure.
- Intelligent factories and software defined industrial systems.

Connectivity

While most modern cities have one or more cellular networks providing city-wide coverage, these networks often have capacity and peak bandwidth limits that just barely meet the needs of their existing subscribers. This leaves little bandwidth for the advanced municipal services envisioned in a smart city. OpenFog RA deployments coupled with 5G technologies provide an opportunity to address this concern. Fog nodes can provide local processing and storage, and optimize network usage.

Safety and Security

Smart city planning also includes critical public safety and security requirements. For example:

- Municipal networks carry sensitive traffic and citizen data (e.g., police dispatches), and operate life-critical systems (e.g., first responder communications)
- Video security and surveillance systems capture suspicious or unsafe conditions (e.g., utility network problems, unauthorized use of public spaces, etc.)

Note that smart cars and traffic congestion, which is covered as a separate use case, are also top priorities for smart cities.

By providing secure data and distributed analytics, fog computing will play a key role in addressing public safety and security issues for smart cities.

3.3.1 Smart Buildings

Smart buildings may contain thousands of sensors to measure various building operating parameters, including temperature, humidity, occupancy, door open/close, keycard readers, parking space occupancy, security, elevators, and air quality. These sensors capture telemetry data at various intervals and transmit this information to a local storage server. Once this information is processed (analyzed), controller-driven actuators will adjust building conditions as necessary.

Some of this processing and response is extremely time-sensitive. For example, turning on fire suppression systems in response to a fire event or locking down an area if an unauthorized person tries to gain entry. Time-sensitive means real-time response, which requires processing in close proximity to the infrastructure devices.

The OpenFog RA model can be extended into the building's control hierarchy to create a number of smart, connected spaces within each building. Using the hierarchical design of the OpenFog RA, each floor, wing, or even individual room could contain its own fog node.

A fog node could be responsible for:

- Performing emergency monitoring and response functions.
- Performing building security functions.
- Controlling climate and lighting.
- Providing a more robust compute and storage infrastructure for building residents to support smartphones, tablets and desktop computers.

Locally stored operational history can be aggregated and sent to the cloud for large-scale analytics. These analytics can be applied to machine learning to create optimized models, which are then downloaded to the local fog infrastructure for execution.

3.4 Additional Use Cases

Fog computing is relevant to many more use cases in addition to those highlighted here. The OpenFog Board of Directors and Technical Committee will set the most important areas for the Consortium and its partners to focus on in the near and long term. We will continue to expand upon all use cases in further discussions, publications, and testbeds.

4 Pillars of OpenFog RA

The OpenFog RA is driven by a set of core principles called pillars. These pillars form the belief, approach and intent that guided the definition of the reference architecture. They represent the key attributes that a system needs to embody the OpenFog definition of a horizontal, system-level architecture that provides the distribution of computing, storage, control, and networking functions closer to the data source (users, things, et al) along the cloud-to-thing continuum.

The following sections describe each pillar of the architecture.

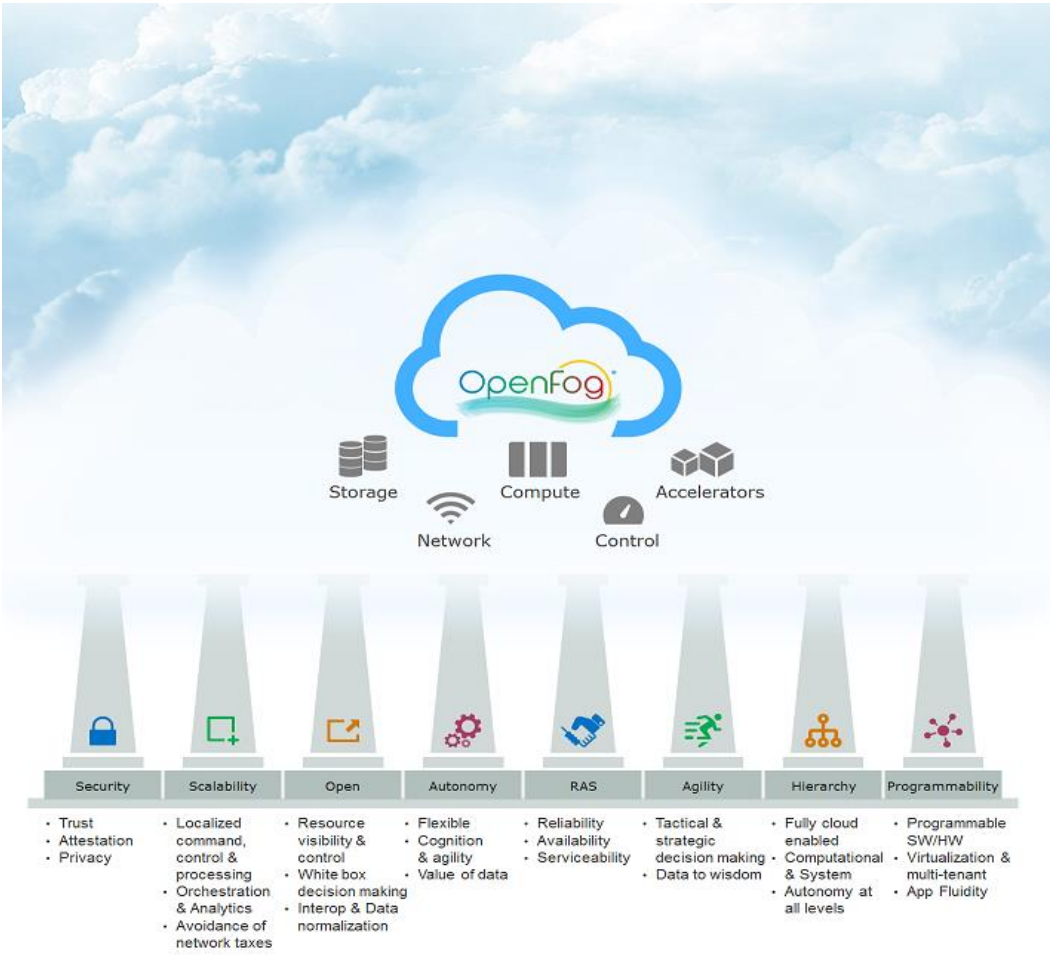


Figure 5 Pillars of OpenFog

4.1 Security Pillar

Many IoT applications supported by the OpenFog RA have privacy critical, mission critical, and even life critical aspects. As such, any security compromise in the fog network can have severe consequences. The OpenFog RA, as it abstracts these technologies, will enable the flexible creation of computing environments that address a broad spectrum of security concerns spanning from IoT devices to cloud and the fog networks in between.

Security in the OpenFog RA is not a one-size-fits-all architecture. Rather, it describes all of the mechanisms that can be applied to make a fog node secure from silicon to software application. Business case, target market, and vertical use case, as well as the location of the node itself, will all create a set of requirements for that node. However, there are certain foundational parts of the architecture, which must be in place in order to build a secure execution environment.

Security implementations have many different descriptions and attributes such as privacy, anonymity, integrity, trust, attestation, verification, and measurement. These are key attributes for the OpenFog RA. Achieving the foundational elements for security requires an approach to discover, attest, and verify all smart and connected "things" before trust can be established.

Conformance to the OpenFog RA requirements will ensure that an OpenFog deployment will be built on a secure end-to-end compute environment. This includes the OpenFog node security, OpenFog network security, and OpenFog management and orchestration security. This will allow architects and designers to focus on the high-value security and privacy problems specific to the types of devices used in their application.

In many applications, particularly for brownfield deployments, or for tiny devices and sensors with little to no security capability, an OpenFog node may act as a device's first point of secure entry into an OpenFog compute hierarchy and the cloud.

The security pillar of the OpenFog RA starts with a clear definition of base building blocks. All fog nodes must employ a hardware-based immutable root of trust. The Hardware Root of Trust is a trusted hardware component which receives control at power-on. It then extends the chain of trust to other hardware, firmware, and software components. The root of trust should then be attestable by software agents running within and throughout the infrastructure. Because of the proximity to the edge, nodes in fog networks often act as the first node of access control and encryption. This

means they must provide contextual integrity and isolation, and control aggregation of privacy-sensitive data before it leaves the edge.

As more complex topologies are created in FaaS implementations, the attestation continues as a chain of trust from the fog node, to other fog nodes, and to the cloud. Since fog nodes may also be dynamically instantiated or torn down, hardware and software resources should be attestable. Components that are not attestable should not be fully allowed to participate in the fog node or may not be deemed to have fully trustworthy data.

4.2 Scalability Pillar

The scalability pillar addresses the dynamic technical and business needs behind fog deployments. Elastic scalability cuts across all fog computing applications and verticals. The hierarchical properties of fog and its location at logical edges of networks add additional scaling opportunities.

- Individual fog nodes can scale internally, through the addition of hardware or software.
- Fog networks can scale up and out through the addition of fog nodes to assist heavily loaded nodes, either on the same level of the fog hierarchy or in adjacent levels.
- A network of fog nodes can be scaled up or down in a demand-driven elastic environment.
- Storage, network connectivity, and analytics services can scale with the fog infrastructure.

Because of the variability in the use cases for fog computing, the OpenFog RA enables elastic scaling of modest deployments through large mission critical deployments based on demand. This scalability is essential for fog computing implementations to adapt to workload, system cost, performance, and other changing business needs.

Scalability may involve many dimensions in fog networks:

- **Scalable performance** enables growth of fog capabilities in response to application performance demands (e.g., reducing latency between sensor reading and resulting actuator responses).
- **Scalable capacity** allows fog networks to change in size as more applications, endpoints, “things,” users, or objects are added or removed from the network. You can add capacity to individual fog nodes by adding hardware like processors, storage devices, or network

interfaces. You can also add capacity through software and various pay-as-you-grow licensing. The converse is also true.

- **Scalable reliability** permits the inclusion of optional redundant fog capabilities to manage faults or overloads. Redundant fog nodes also ensure a large deployment's integrity and reliability at scale, which is part of the reliability, availability, and serviceability (RAS) pillar. There are hardware and software aspects to scalable reliability. The scalability mechanisms supporting the reliability of fog networks must themselves be highly reliable. Availability (which is a scaling measure closely related to reliability) scales through similar methods.
- **Scalable security** is often achieved through the addition of modules (hardware and software) to a basic fog node as its security needs become more stringent. Capabilities like scalable distribution, rights access, crypto processing capacity, and autonomous security features contribute to scalable security.
- **Scalable hardware** involves the ability to modify the configuration of the internal elements of fog nodes, as well as the numbers of and relationships between fog nodes in networks.
    - Processors scale from modest single core CPUs to specialized accelerator chips with thousands of cores or millions of gates.
    - Networking scales from a single wireless or wire line interface to large arrays of wireless, wire line, and fiber interfaces with aggregate capacities of many Gb/s.
    - Storage can scale from a simple flash chip to large arrays of flash / rotating disks and network attached file systems.

These resources can be configured in initial deployments and retrofit into existing modular fog nodes on an as-needed basis. It is also possible to scale at the network level, by adding arrays of fog nodes at locations in the network where single nodes formally managed the entire load. Hardware scaling can also be in the downward direction, where modules or entire fog nodes that are no longer are needed at a specific location are powered down and/or removed (and perhaps reused elsewhere in a fog network where there is a higher need.

- **Scalable software** is also important and includes applications, infrastructure, and management.
    - The management infrastructure of fog must scale to enable the efficient deployment and ongoing operation of tens of millions of fog nodes in support of billions of smart and connected things.
    - Orchestration must be scalable to manage the partitioning, balance, and allocation of resources across the fog network.
    - Analytics as a capability of fog networks has particularly aggressive scalability targets. This is because analytics

algorithms undergo several orders of magnitude scaling due to increased capacity demands and several additional orders of magnitude due to ever-increasing sophistication of the analytics algorithms.
    - Composability and modularity are important aspects of scalability, where individual hardware and software components (perhaps numbering in thousands of types and millions of instantiations) are assembled into a fog network optimized to run the specific applications required.

Scalability enables fog nodes to provide basic support to address the business requirements and enable a pay-as-you-grow model for the FaaS, which is essential to the economics of its initial deployment and long-term success.

4.3 Openness Pillar

Openness is essential for the success of a ubiquitous fog computing ecosystem for IoT platforms and applications. Proprietary or single vendor solutions can result in limited supplier diversity, which can have a negative impact on system cost, quality and innovation. The openness pillar importance is highlighted in our desire for fully interoperable systems, supported by a vibrant supplier ecosystem.

Openness as a foundational principle enables fog nodes to exist anywhere in a network and span networks. This openness enables pooling by discovery, which means that new software-defined fog nodes can be dynamically created to solve a business mission. The security pillar shares a common theme and requirements in openness characteristics

- **Composability** supports portability and fluidity of apps and services at instantiation. Additional emphasis of composability is visible in the programmability pillar.
- **Interoperability** leads to secure discovery of compute, network, and storage and enables fluidity and portability during execution. The marketplace has clearly articulated its desire for a vibrant supplier ecosystem, with reasonable expectations that elements from one supplier can be freely substituted for elements from another supplier. This will be addressed through testbeds, FogFests (plug fest), standardization, and open implementations.
- **Open communication** enables features like pooling of resources near the edge of the network to collect idle processing power, storage capacity, sensing ability, and wireless connectivity. For example, a

compute-intensive application developed in fog architecture can leverage hundreds of gigabytes sitting idle on nearby laptops, systems, and set-top boxes in a household every evening, or among the passengers of a public transit system. The open discovery of these nearby compute resources is critical. Doing the functional work nearest the edge avoids additional network taxes when moving up the stack towards the cloud. We define network taxes as the cost of transmission.

- **Location transparency** of any node instance to ensure that nodes can be deployed anywhere in the hierarchy. Location transparency provides an alternative to network operator control. This means that any IoT device, such as a smart watch, does need its own carrier-owned data plan. Each thing or software entity can observe its local conditions and make decisions on which network to join. Each endpoint in a fog network can optimize its path to the computational, networking and storage resources it needs (no matter if those resources are in the hierarchical layers of the fog, or in the cloud).

4.4 Autonomy Pillar

The autonomy pillar enables fog nodes to continue to deliver the designed functionality in the face of the external service failures. In this architecture, autonomy is supported throughout the hierarchy. Decision making will be made at all levels of a deployment's hierarchy including near the device or higher order layers. Centralized decision-making in the cloud is no longer the only option. Autonomy at the network edge means intelligence from local devices and peer data can be used to fulfill the business' mission at the point where it makes the most business sense.

The OpenFog RA supports autonomy for a wide range of functions. It does not rely upon centralized entity for operation (e.g., a backend cloud). Some of the typical areas for autonomy at the edge include:

- **Autonomy of discovery** to enable resource discovery and registration. For example, an IoT device coming online in the field would typically "phone home" first to let the backend cloud know it is alive and its associated functions are available. But when an uplink network to the cloud is unavailable, it can stop the device from going live. An autonomous fog node can potentially act as a proxy for the device registration, which then allows the device to come online without the backend cloud.
- **Autonomy of orchestration and management** (O&M) automates the process of bringing services online and managing them through

the operational lifecycle and decommissioning. Autonomy of O&M entails a number of actions including: instantiation of services; provisioning the environment around the services, such as routing of data flows; and keeping track of the health and status of the resources. All these actions should be as automated as possible through programmability and policies. The architecture includes an autonomous and scalable O&M function that is set up to handle any surge of demand for resources, without real-time reliance on the cloud or the need for significant human labor.

- **Autonomy of security** enables devices and services to come online, authenticate themselves against a minimal set of fog security services, and perform their designed functions. In addition, these security services can store records for future audits. With autonomy, these actions can be performed where they are needed, when they are needed, and regardless of connectivity to the cloud. Fog nodes can autonomously react to evolving security threats, such as updating virus screening algorithms, determination of denial-of-service (DoS) attacks, etc. without administrator involvement.
- **Autonomy of operation** supports localized decision making by IoT systems. Sensors provide data, which is the basis for autonomous actions at the edge. If the cloud or a single place in the system's hierarchy is the only location where decisions can be made, this violates the ability to ensure reliability and as such, the architecture ensure operational autonomy.
- **Cost savings** is a key motivator for autonomy. Connectivity today costs money. The more data that is sent through the network, the higher the costs are for businesses due to network taxes. This drives the need for more processing at the edge of the network, with just-in-need and just-in-time data sent to the cloud as required for additional business insights. For example, when an oil rig generates 30,000 data points a second, not all of the data must be sent through an expensive satellite link. Local and fog domain analytics and pre-processing can autonomously filter out the unimportant data points and extract the more mission critical ones to be delivered to the next layer in the hierarchy.

A key aspect of fog computing is turning data into actionable wisdom. We term this DIKW, which stands for "**D**ata gathered becomes **I**nformation when stored and retrievable becomes **K**nowledge. Knowledge enables **W**isdom for autonomous IoT." This principle is the basis for localized analytics to enable autonomous decision making nearest the edge.

4.5 Programmability Pillar

The programmability pillar enables highly adaptive deployments including support for programming at the software and hardware layers. This means that re-tasking a fog node or cluster of fog nodes for accommodating operational dynamics, can be completely automated. The re-tasking can be done with the help of the fog nodes inherent programmability interfaces which we describe using general purpose compute or accelerator interfaces. Programmability of a fog node includes the following benefits:

- **Adaptive infrastructure** for diverse IoT deployment scenarios and support changing business needs.
- **Resource efficient deployments** maximizing the resources by using a multitude of features including containerization. This increases the portability of components and is a key design goal enabled by programmability.
- **Multi-tenancy** to accommodate multiple tenants in a logically isolated runtime environment.
- **Economical operations** that results adaptive infrastructure to changing requirements.
- **Enhanced security** to automatically apply patches and respond more quickly to evolving threats.

4.6 Reliability, Availability, and Serviceability (RAS) Pillar

Reliability, availability, and serviceability (RAS) is resident throughout successful system architectures and, as such, takes on great importance in the OpenFog RA. Hardware, software, and operations are the three main areas of the RAS pillar.

A reliable deployment will continue to deliver designed functionality under normal and adverse operating conditions. The reliability of the RAS pillar includes but is not limited to the following properties:

- Ensuring reliable operation of the underlying hardware upon which the software is operating, enabling reliable and resilient software and a reliable fog network, which is generally measured in uptime.
- Safeguarding the availability and integrity of data and compute on edge gateways using enhanced hardware, software, and network designs.
- Autonomous predictive and adaptive self-managing capabilities when required by the health of the system to initiate self-healing routines for

hardware and software and upgrade new firmware/application and security patches.
- Increasing customer satisfaction by simplifying support and device self-optimization and healing.
- Initiating requests for preventative maintenance, including new hardware and software patches, network re-routing, etc.
- Testing and validation of system components, including device drivers and diagnostic tools under a variety of environmental conditions.
- Providing alarms, reports, logs, etc.
- Validation of system platforms and architectures through interoperability certification test suites.

Availability ensures continuous management and orchestration, which is usually measured in uptime. The availability of the RAS pillar includes but not limited by the following properties:

- Secure access at all levels of a fog hierarchy for orchestration, manageability, and control, which includes upgradeability, diagnostics and secure firmware modification.
- Fault isolation, fault syndrome detection, and machine learning to help improve Mean Time To Repair (MTTR) of a failed system to achieve higher availability.
- Concept of cloud based back-end support with availability of interfaces throughout the system.
    - Secure remote access from a plurality of devices (not just a single console).
    - Redundant/duplicate device (peer-to-peer) IoT platform.
    - Mesh access capabilities of end-point sensor/peering.
    - Remote boot capabilities of the platform.
        - Modification and control from the lowest level firmware (BIOS) through to the highest software in the hierarchy (cloud).
    - Support for redundant configurations for persistent productivity.

Servicing a fog deployment ensures correct operation. Serviceability of the RAS pillar includes but is not limited by the following properties:

- Highly automated installation, upgrade, and repair to efficiently deploy fog computing at scale.
- Hardware or software can either autonomously heal or be serviced by the various manufacturers.
- Ease of use to accommodate maintenance.
- Serviceability of the system:
    - Hardware, software, applications, networking, and data

    - Ease of access/swap-out of the hardware (component interoperability).
    - Ease of secure upgradeability of software, BIOS, and applications locally or remotely and in real time.
    - Replication of system configuration over cloud on replaced/swap-out systems.
- Support for redundant configurations for persistent productivity.

RAS is especially important for OpenFog RA deployments in harsh environmental conditions and remote locations. This is why aspects from RAS are found throughout the architecture.

4.7 Agility Pillar

The agility pillar addresses business operational decisions for an OpenFog RA deployment. It is not possible for humans alone to analyze the data generated at the scale predicted by IoT as the basis for rapid, sound business and operational decisions. The agility pillar focuses on transforming this volume of data into actionable insights. Agility also deals with the highly dynamic nature of fog deployments and the need to respond quickly to change.

Data generation by sensors and systems in an OpenFog RA deployment are turbulent, bursty, and are often created in huge volumes. Most importantly, data may not have context, which is created only when the data is collated, aggregated, and analyzed. The analysis of data can be executed at the cloud level, but this subjects the data to increasing levels of latency. The ideal approach is to make operational decisions as soon as data can be turned into a meaningful context. The architecture enables the creation of context close to the data generation where it makes the most sense for a given scenario. More strategic, system-wide decisions and policy management can be made further up the layers in the fog hierarchy. This avoids network dependencies we termed as “network taxes” as described in other OpenFog RA pillars.

OpenFog architectural approaches allow IoT system developers to optimize the placement of their applications as decision making components.

4.8 Hierarchy Pillar

Computational and system hierarchy is not required for all OpenFog architectures but it is still expressed in most deployments. The OpenFog architecture is complementary to traditional cloud architectures due in part to the OpenFog hierarchy pillar.

OpenFog RA computing resources can be seen as a logical hierarchy based on the functional requirements of an end-to-end IoT system. Depending on the scale and nature of the scenario being addressed, the hierarchy may be a network of smart and connected partitioned systems arranged in physical or logical layers, or it may collapse into a single physical system (scalability pillar).

Using building automation from smart cities as an example, a company that manages a single office complex may have the entire fog deployment located locally. A large commercial property management company may have distributed fog deployments at local and regional levels feeding information to centralized systems and services. Each fog node is autonomous (autonomy pillar) to ensure uninterrupted operations of the facility it manages.

The figure below shows a logical view of the IoT system from a computational perspective. Each layer in the hierarchy addresses a specific concern of the IoT system.

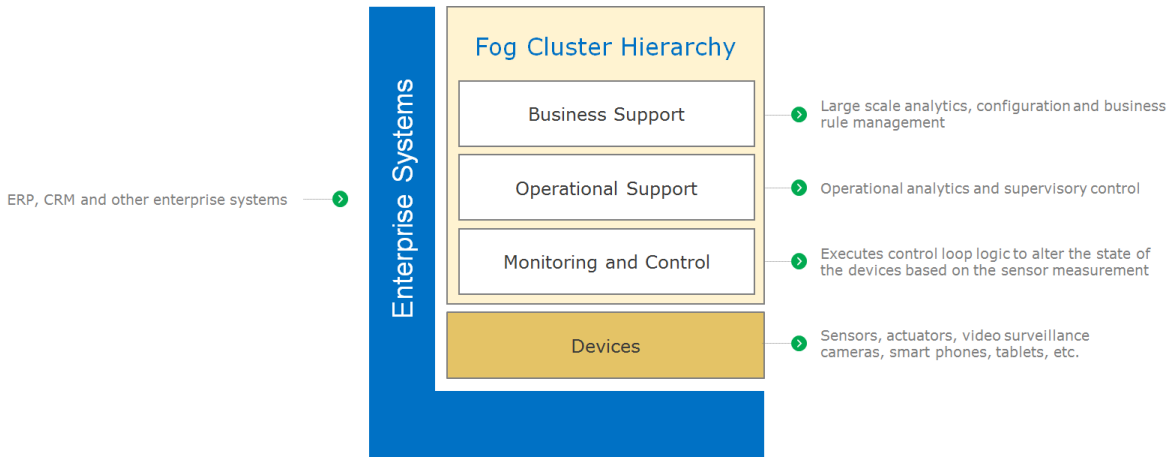


Figure 6 Layered Architecture View of an IoT System

Devices in the hierarchy: Sensors and actuator devices are the physical things and produce telemetry data to be consumed by the monitoring and control layer. This layer analyzes the telemetry and generates actuation commands if the process being monitored deviates from the desired state. Note that the term "process" is used in an abstract sense that it is represented by a set of measured parameters that depend on a set of actuator settings. Depending on the domain problem, some systems may not have any actuators. Similarly, for scenarios like mobile network acceleration, the core function is to accelerate content delivery and not

monitoring and control. On the other hand, systems like building operations may have actuators to change the HVAC and lighting based on occupancy.

Monitoring and control in the hierarchy: Sensors and actuators are connected to microcontrollers that are programmed to monitor and control the state of the processes. A process state is represented by a set of parameters measured by the sensors and modified by the actuators. The main responsibility of this layer is to execute control logic through stateful inspection of the sensor telemetry. This involves computing alarms and generating events, which may trigger workflows through machine-to-machine or human intervention.

Operational support in the hierarchy: The operational support layer is responsible for analyzing streaming telemetry and storing operationally oriented analytics. The analytics may be presented through interfaces like control room dashboards and mobile applications. The scope of the analytics at this layer is narrow; it focuses on the operational aspects of the physical environment for which the system is responsible. This layer combines drill down historical analytics with streaming analytics for a composite picture of real-time operations with some short-term history. The agility pillar is seen in the hierarchy as the implementation of complex event processing on the streaming telemetry data.

Surrogacy in the hierarchy: The computation of a complex operation in the fog nodes may be delegated to the hierarchical nodes to leverage adjacent resources. Consider virtual reality tasks associated with a wearable such as smart glasses. Some of the information retrieval and computation tasks may be carried out on the glasses, while an associated element in the hierarchy (e.g., a smartphone) may handle its storage and connectivity requirements. This hierarchical architecture may leverage all of these devices at the same time, with an intelligent division of labor across them.

Business support in the hierarchy: The primary responsibility of this layer is to store and analyze the entire history of the IoT operations that span multiple systems. This is the system of record for IoT operations as governed by the compliance and record retention policies. Petabyte scale analytics will help in mining insights, business planning, comparing the operational efficiency of processes, operational optimization through training machine learning models, etc. Additionally, metadata and reference data management, business rule management, and the operational health of lower layers are the other aspects of this layer. These are also viewed in the agility pillar.

4.8.1 Hierarchical Fog Deployment Models

The figures below show a subset of the combination of fog and cloud deployed to address various domain scenarios as framed by the layered view of IoT systems. Each fog element may represent a hierarchy of fog clusters fulfilling the same functional responsibilities. Depending on the scenario, multiple fog and cloud elements may collapse into a single physical deployment. Each fog element may also represent a mesh of peer fog nodes in use cases like connected cars, electrical vehicle charging, and closed loop traffic systems. In these use cases, fog nodes may securely discover and communicate with each other for exchanging context-specific intelligence.

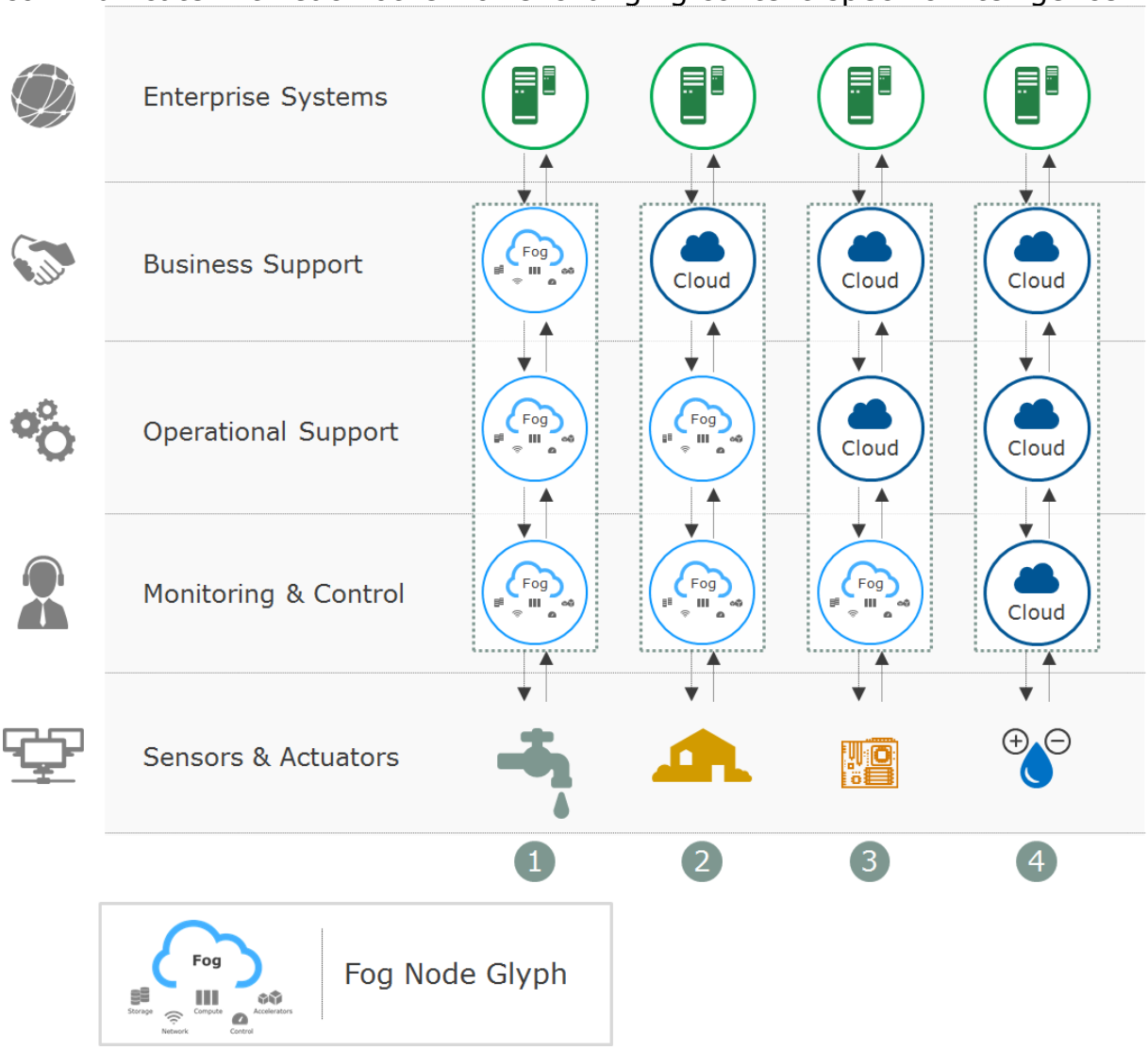


Figure 7 IoT System Deployment Models

Figure 1 shows a fog deployment hierarchy that is independent of the cloud. This model may be applicable for use cases where cloud can't be used for

reasons such as low event to action time window, regulatory compliance, military grade security and privacy, and unavailability of a central cloud in a particular geography. Examples include armed forces combat systems, drone operations, some healthcare systems, hospitals, and ATM banking systems.

Figure 2 shows the cloud used for information processing related to decision making that may have event-to-action time window ranging from hours to days to months. Operation-centric information processing is done by fog deployments located close to the infrastructure/process being managed. Use cases include commercial building management, commercial solar panel monitoring, and retail.

Figure 3 shows the local fog infrastructure used for time-sensitive computation, while the cloud is used for the balance of operational and business-related information processing. Use cases include commercial UPS device monitoring, mobile network acceleration, and content delivery networks (CDNs) for Internet acceleration.

Figure 4 supports use cases like agriculture, connected cars, and remote weather stations. These use cases leverage the cloud for the entire stack due to the constrained environments in which the deployment of fog infrastructure may not be feasible or economical. Fog nodes at the device layer may get some of the monitoring and control function for safety related control. The enterprise systems integrate with cloud for business operations.

Note that the functional boundaries shown in Figures 1-3 are fluid and can be physically deployed in multitude of combinations based on the architecture of the domain-specific solutions. In real world deployments as we discussed earlier, there may be many combinations of physical deployments that involve multi-tenants, fog, and cloud deployments owned by multiple entities. The following diagram illustrates some those models:

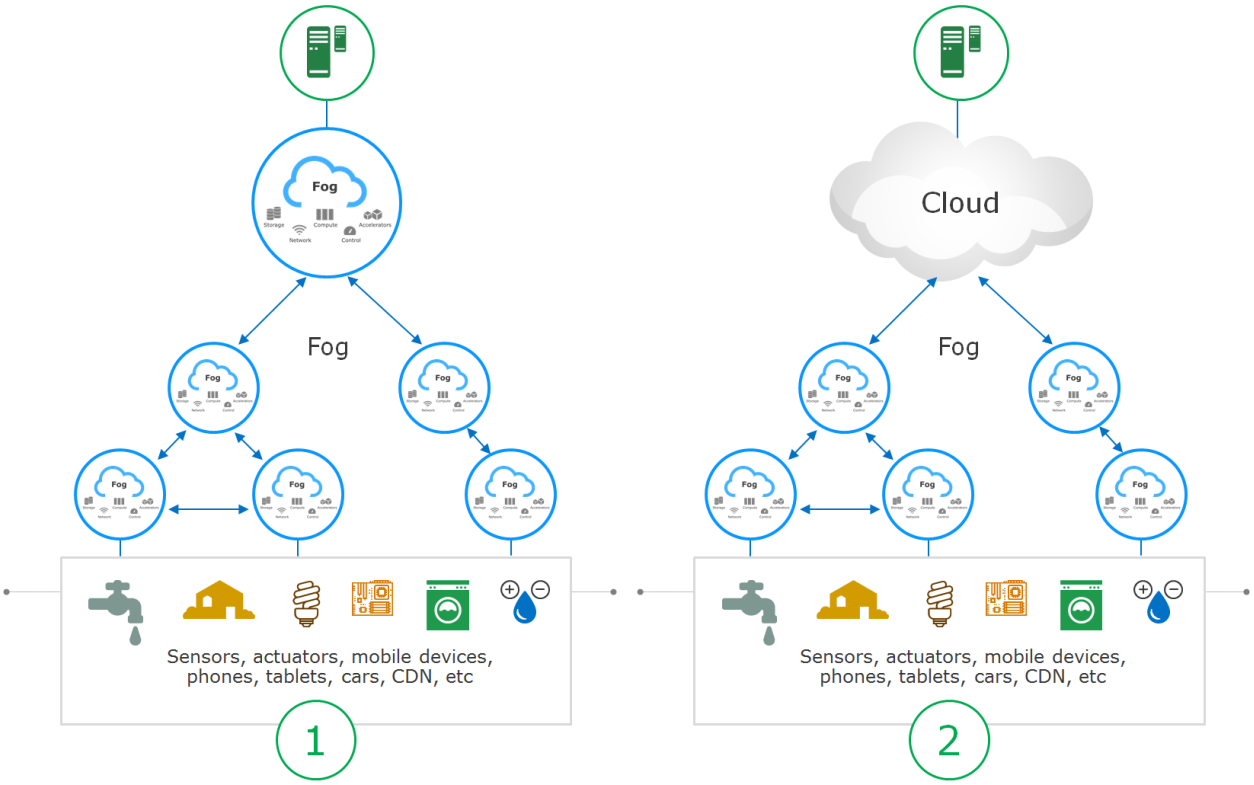


Figure 8 Fog Hierarchy Example

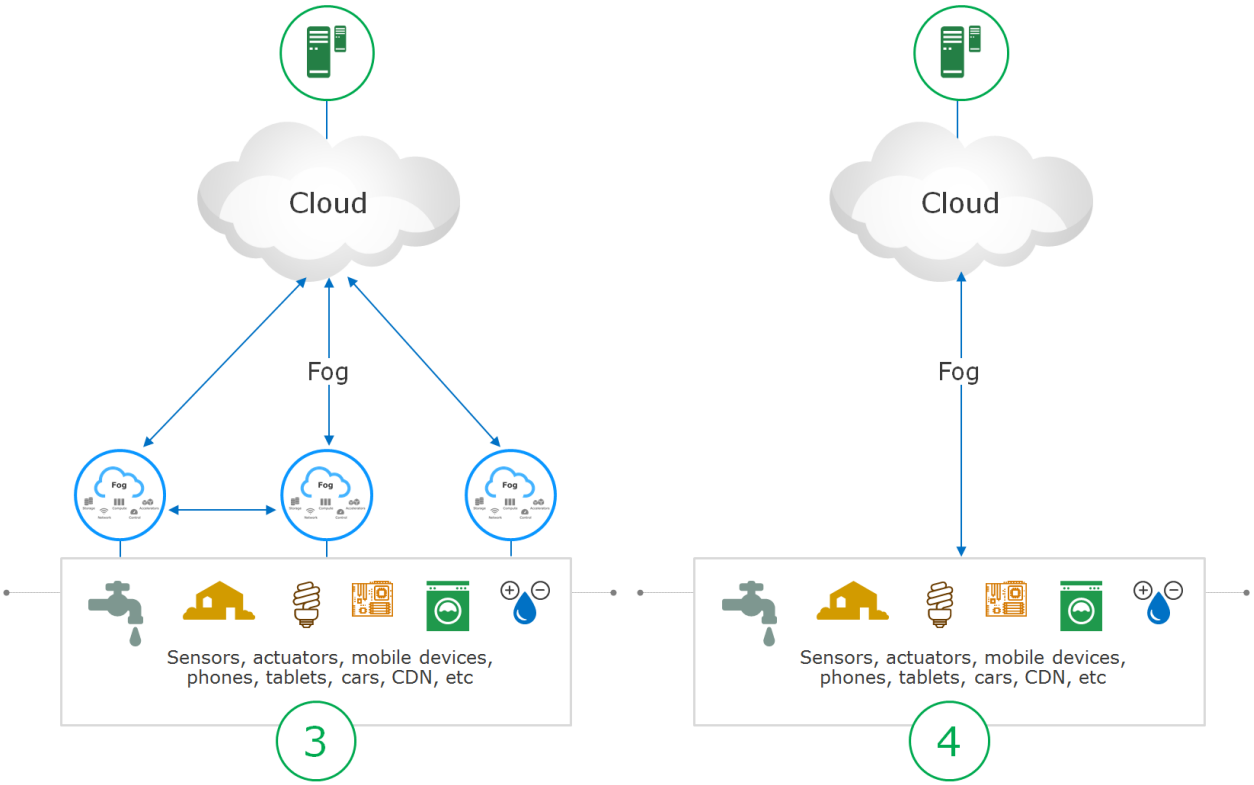


Figure 9 Fog Hierarchical Deployment Model

Many of the usages will occur as represented in Figures 2 and 3. The three-layer fog hierarchies shown here are for illustrative purposes only. Real-world fog deployments may have more or fewer levels. Different vertical application use cases may use a fog hierarchy differently. For example, in a smart city, there may be fog nodes in a region, neighborhood, street corner, and building level. In a smart factory, the hierarchy may be divided by assembly lines, manufacturing cells, and machines.

5 Reference Architecture Overview

The OpenFog reference architecture is based on the eight pillars described in the previous chapter. We use the ISO/IEC/IEEE 42010:2011 international standard as the guideline for describing architecture to stakeholders. The standard enables a common vocabulary across IoT to help support cross-organizational technical collaboration. These are some of the shared vocabulary terms used in the OpenFog RA:

- **OpenFog Architecture Description:** is an abstract representation of an instance of a fog node. It is a composite of multiple views we used to address stakeholders in the fog computing value chain.
- **Viewpoint:** A viewpoint is a way of looking at a system. These included but are not limited to Functional and Deployment viewpoints.
- **View:** A view is a representation of one or more structural aspects of the architecture. In the current revision of the OpenFog RA, the structural aspects are the Software view, System view, and Node view.
- **Perspective:** A perspective is a cross-cutting concern of the architecture.

5.1 Functional Viewpoint

The functional viewpoint of the architecture shows how we apply the OpenFog architectural elements and views to address the various concerns of the stakeholders to satisfy a given scenario. Each scenario chosen will focus on a different aspects and market opportunities for fog computing. We fully expect that the architectural description, views, and perspective may change over time. This change and refinement should be driven from testbeds and application of the OpenFog RA to multiple different markets we see as important to fog computing.

The first end-to-end scenario we will address is the visual security scenario. Our intent will be to define how that scenario works, and then work through various testbeds associated with those aspects to validate or modify the architecture.

5.2 Deployment Viewpoint

5.2.1 OpenFog Deployment Types

How fog software and fog systems are deployed to address a given scenario is also very important. There are many deployment types ranging from embedded to large clustered systems that are fully interconnected. The deployment type(s) chosen are scenario specific, but key aspects of the architecture remain visible regardless of deployment type. However, some may grow or shrink in importance.

5.2.2 N-Tier Fog Deployment

In most fog deployments, there are usually several tiers (N-tiers) of nodes. For this example, we will use a simple food processing plant to help solidify the logical tiers. There is a conveyor belt on which food is processed before moving on to the next level of packaging and shipment.

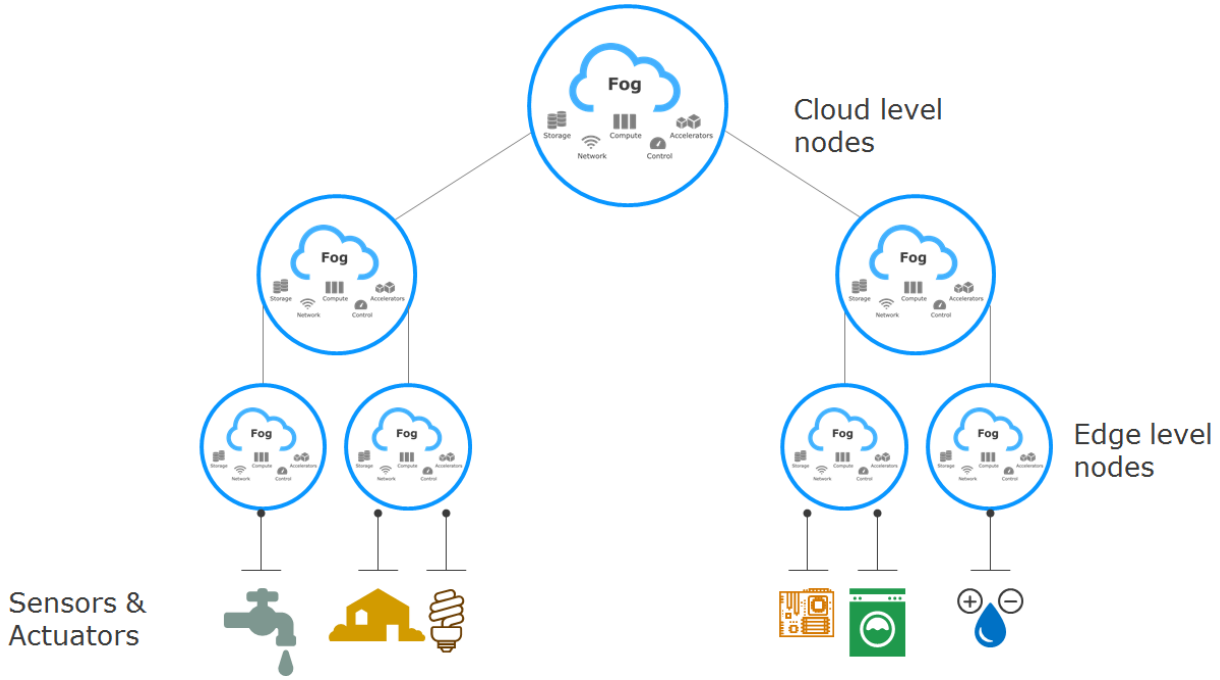


Figure 10 Multi-Tier Deployment

- Nodes at the edge are typically focused on sensor data acquisition/collection, data normalization, and command/control of sensors and actuators.
    - Using our example, the fog nodes nearest the physical technology edge (at the conveyor belt) will need to operate at

millisecond and sub-millisecond granularity from sensing to actuation to avoid product contamination and to ensure safety of operation.

- Nodes in the next higher tier are focused on data filtering, compression, and transformation. They may also provide some edge analytics required for critical real time or near real time processing. As we move away from the true network edge, we see higher level machine and system learning (analytics) capabilities.
    - Using our previous example, we can easily see that this tier needs to operate on a slightly higher level. It should be ensuring that the conveyor belt and others around it are operating more efficiently. This may still be in the latency of milliseconds. The key aspect of OpenFog RA to enable migration of applications, and functionality between the lowest tiers and the middle tiers as required by the scenario.
- Nodes at the higher tiers or nearest the backend cloud are typically focused on aggregating data and turning the data into knowledge. It's important to note that the farther from the true edge, the greater the insights that can be realized.

Note: In some deployment models, some degree of analytics may be located in nodes at the edge of the network (e.g. video analytics on surveillance cameras). This is because the network pipes may not be large enough to cost effectively carry the raw sensor data to higher layer fog nodes for processing. In reality as computational capabilities grow, the analytics functions at the lower tiers will grow. This will enable the overall growth of intelligence of fog deployments over time.

Machine Learning is in the forefront of research today that requires computation for both training models, and inference or scoring those models for close to real time response at the edge. We could use machine learning to optimize operations at a train station in a smart city. In the train station, we could monitor and sense occupancy, movement, and overall system usage and over time adapt our infrastructure to determine how to most efficiently use it.

Using the smart city as a continued example, by including more buildings and having them talk to each other and using another tier above them to gain additional insights, city blocks can operate more efficiently. Learnings from blocks of buildings provide insights on how to make the overall city more efficient and so on and so forth. The key message is that as you move farther away from the true network edge, you can gain operational insights and increase the overall system intelligence. Additionally, migration of data

between these fog nodes in both the horizontal and vertical increases the system performance and operational capabilities.

5.2.2.1 How Use Cases Determine the Number of Tiers

Fog deployments will come large-scale and small, based upon the given scenario being addressed. The number of tiers in a fog deployment will be dictated by the scenario requirements, including:

- Amount and type of work required by each tier
- Number of sensors
- Capabilities of the nodes at each tier
- Latency between nodes and latency between sensors and actuation
- Reliability/availability of nodes

In general, each level of the N-tier environment would be sifting and extracting meaningful data to create more intelligence at each level. Tiers are created in order to deal efficiently with the amount of data that needs to be processed and provide better operational and system intelligence. At the highest level this can be represented by the figure below.

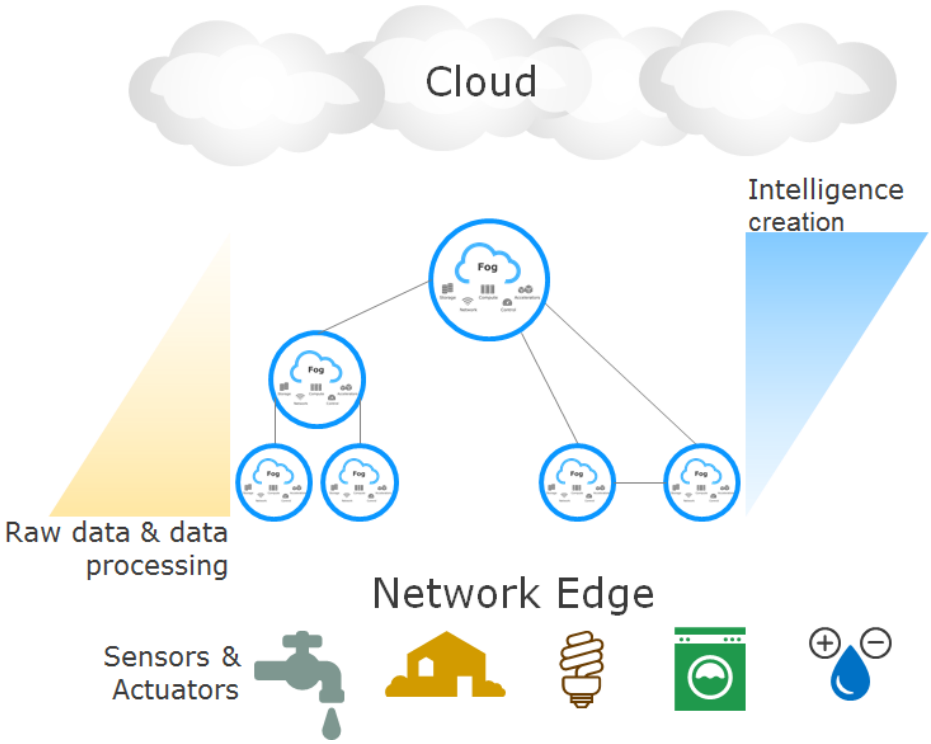


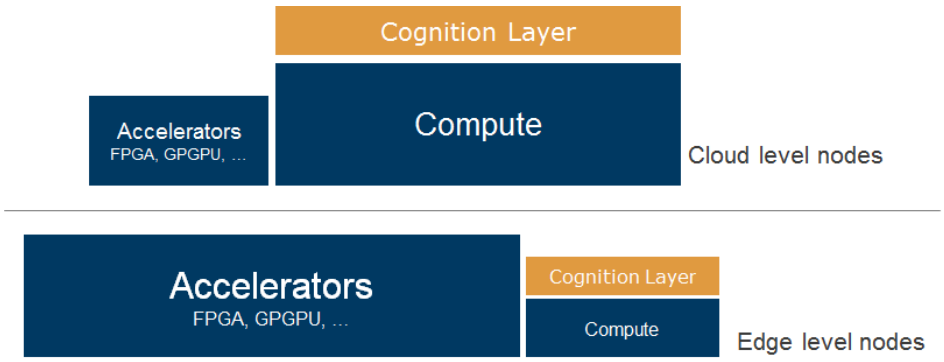
Figure 11 Intelligence from data

Important aspects we will address overtime is that the software running at every level and each node should be enabled to migrate across nodes, span

physical instantiations of hardware nodes, and change over time to address the needs of a given scenario. For this to be securely and safely achieved we need to address not only the software, but also the hardware on which that software is executed.

5.2.2.2 Fog Node Uniformity

The architectural elements of a node will vary based on its role and position within an N-tier fog deployment. As described previously, nodes at the edge may be architected with less processing, communications, and storage than nodes at higher levels. However, I/O accelerators required to facilitate sensor data intake at the edge may be much larger in aggregate than I/O accelerators designed for higher level nodes.



Fog nodes may be linked to form a mesh to provide load balancing, resilience, fault tolerance, data sharing, and minimization of cloud communication. Architecturally, this requires that fog nodes have the ability to communicate laterally (peer to peer or east to west) as well as up and down (north to south) within the fog hierarchy. The node must also be able to discover, trust, and utilize the services of another node in order to sustain RAS. Using our building and city example as before this is represented by the following figure.

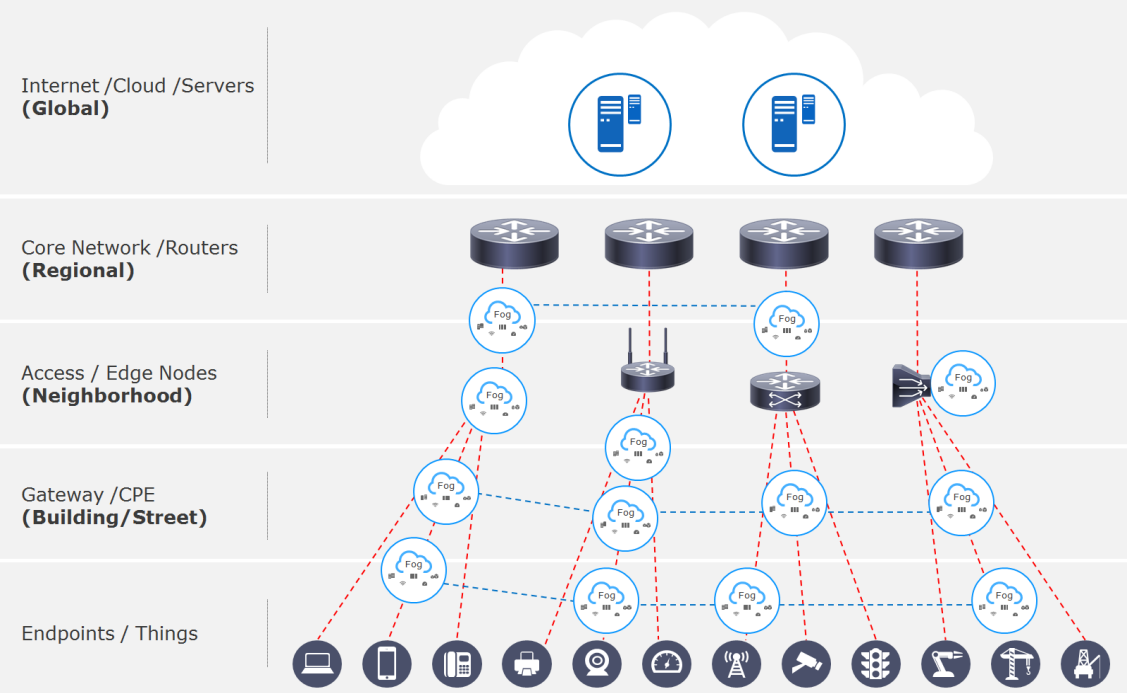


Figure 12 Fog Node East/West Communication

Each building is connected, neighborhoods, and regions are connected to provide an infrastructure that may be optimized for service delivery.

5.3 OpenFog Architecture Description

As we previously described, fog computing is critical because it enables low latency, reliable operation, and removes the requirement for persistent cloud connectivity to address many of today's emerging scenarios. We also described how fog nodes can be connected partially or fully to enhance the overall system intelligence and operation, and how system wide intelligence grows the farther away from raw data processing.

The next step is to describe the requirements for each stakeholder in the fog computing continuum. This includes the silicon manufacturer, system manufacturer, system integrator, software manufacturer, and application developer. We also believe that this architecture will help align the various disparate edge based computing but potentially divergent work under a singular vernacular so that we can have a common baseline and work towards fulfilling our desire of a multi-vendor interoperable fog computing ecosystem. The OpenFog RA description is a composite representation of these various stakeholder concerns which we call views. We have primarily identified these stakeholders and their associated views because they are required to facilitate any successful fog based deployment. Before going into

the lower level details of the view we believe it is important to first look at the composite architecture description.

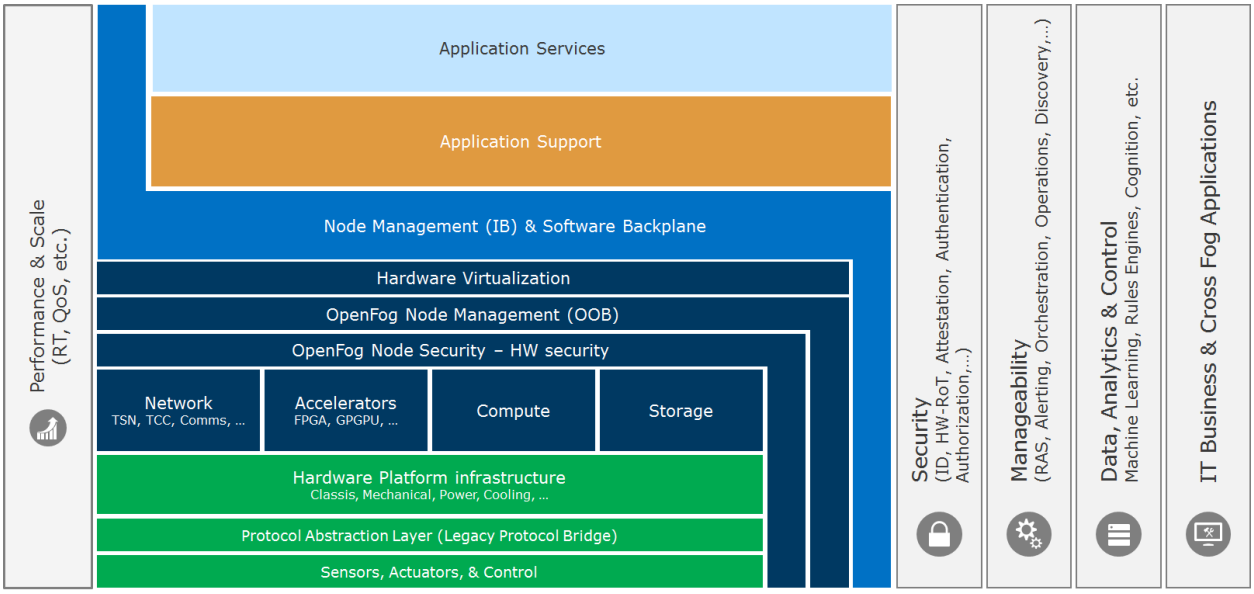


Figure 13 Architecture Description with Perspectives

The abstract architecture includes perspectives, shown in grey vertical bars on the sides of the architectural description. The perspectives include:

- **Performance:** Low latency is one of the driving reasons to adopt fog architectures. There are multiple requirements and design considerations across multiple stakeholders to ensure this is satisfied. This includes time critical computing, time sensitive networking, network time protocols, etc. It is a cross cutting concern because it has system and deployment scenario impacts.
- **Security:** End-to-end security is critical to the success of all fog computing deployment scenarios. If the underlying silicon is secure, but the upper layer software has security issues (and vice versa) the solution is not secure. Data integrity is a special aspect of security for devices that currently lack adequate security. This includes intentional and unintentional corruption.
- **Manageability:** Managing all aspects of fog deployments, which include RAS, DevOps, etc., is a critical aspect across all layers of a fog computing hierarchy.
- **Data Analytics and Control:** The ability for fog nodes to be autonomous requires localized data analytics coupled with control. The actuation/control needs to occur at the correct tier or location in the hierarchy as dictated by the given scenario. It is not always at the physical edge, but may be at a higher tier.

- **IT Business and Cross Fog Applications:** In a multi-vendor ecosystem applications need the ability to migrate and properly operate at any level of a fog deployment's hierarchy. Applications should also have the ability to span all levels of a deployment to maximize their value.

As previously discussed, the OpenFog RA description is a composite of perspectives and multiple stakeholder views used to satisfy a given fog computing deployment or scenario. The three views that we have identified include Software, System, and Node.

- **Software view:** is represented in the top three layers shown in the architecture description, and include Application Services, Application Support, and Node Management (IB) and Software Backplane.

- **System view:** is represented in the middle layers shown in the architecture description, which include Hardware Virtualization down through the Hardware Platform Infrastructure.

- **Node view:** is represented in the bottom two layers shown in Figure 19, which includes the Protocol Abstraction Layer and Sensors, Actuators, and Control.

Note: The fog platform coupled with the fog software creates the complete fog node. One or more fog nodes comprises a solution in a given market segment or scenario.

However, high-level architectures, including the OpenFog RA, are intended to help engineers, architects, and business leaders understand their specific requirements and how fog nodes can be applied to a given scenario. The goal of the OpenFog Consortium is to increase the number of market segments (use cases) for fog computing, and its business value. OpenFog will create test-beds to adapt the high-level architecture to these market segments. These testbeds will also provide opportunities for FogFests (plug fests) to help drive component level interoperability and accelerate time to market.

The following sections go into more detail about the structural aspects (views) and perspectives of the RA.

5.4 Perspectives (Cross Cutting Concerns)

Cross-cutting perspectives are employed throughout fog implementations. "Cross cutting" refers to capabilities that cut across architectural layers. The figure below shows the five cross-cutting perspectives of the fog computing architecture.

Figure 14 OpenFog Architecture Perspectives

5.4.1 Performance and Scale Perspective

When fog computing brings some of the intelligence of cloud-based applications and analytics to the edge of the network (or as close as possible to the data source), the performance of the overall system (however that system is defined) will improve. Fog computing will also enable the system to better adapt to changing traffic patterns. This means that performance improvements happen faster and are also more relevant and specific to business case requirements.

Another requirement for performance is that improvements in one area must not interrupt or slow other processes requiring a guaranteed quality of service or performance. When measuring a fog node's performance, we usually look at throughput and latency. Both depend on the ability to prioritize traffic types or classes throughout the whole system.

In the RA, virtualization and containerization technologies are both used in fog computing to help with scalability and isolation. These newer technologies further support the ability to assign higher priority or more

resources to specific applications or services dynamically. For example, high priority network traffic can be marked and classified by the network interface, the computational elements, and the appropriate higher-layer applications.

Network bandwidth and local storage can also be given higher priority on a dynamic basis. For example, if the fog node is used for traffic inspection, CPU/memory is assigned higher priority than storage and preservation of history.

5.4.2 Security Perspective

In a fog computing infrastructure, end-to-end security must cover everything between the cloud and the things on the edge of the network. In the architecture, security starts with the individual fog node hardware. If the node is not designed with the appropriate security to ensure that it is a trusted element, it isn't possible to build a trustworthy end-to-end fog computing infrastructure. Once trusted fog nodes have been deployed, a secure fog network can be layered on top of the node infrastructure, providing the basis for secure node-to-node, node-to-thing, and node-to-cloud communication.

5.4.2.1 Trusted and Trustworthiness

Trustworthy fog systems depend on using trusted elements that are responsible for maintaining the security policies specified for a given device. If one or more trusted components (hardware, firmware, or software) in a node are compromised, then the node—and, by extension, the system—is no longer trustworthy. The RA also determines trustworthy attributes including behavior by inspection of historical behavior at various levels of the hierarchy to determine if the system and its components are acting in a trustworthy manner.

Policies specify who or what can access which resource under what circumstances. Security mechanisms then implement the policy. Some policies may be embedded in the node's hardware and software. Other policies may be pushed from the fog management system to the node; they can be added, changed, or deleted by an authorized management or orchestrator administrator. Each layer in a fog deployment requires security as represented in the figure below.

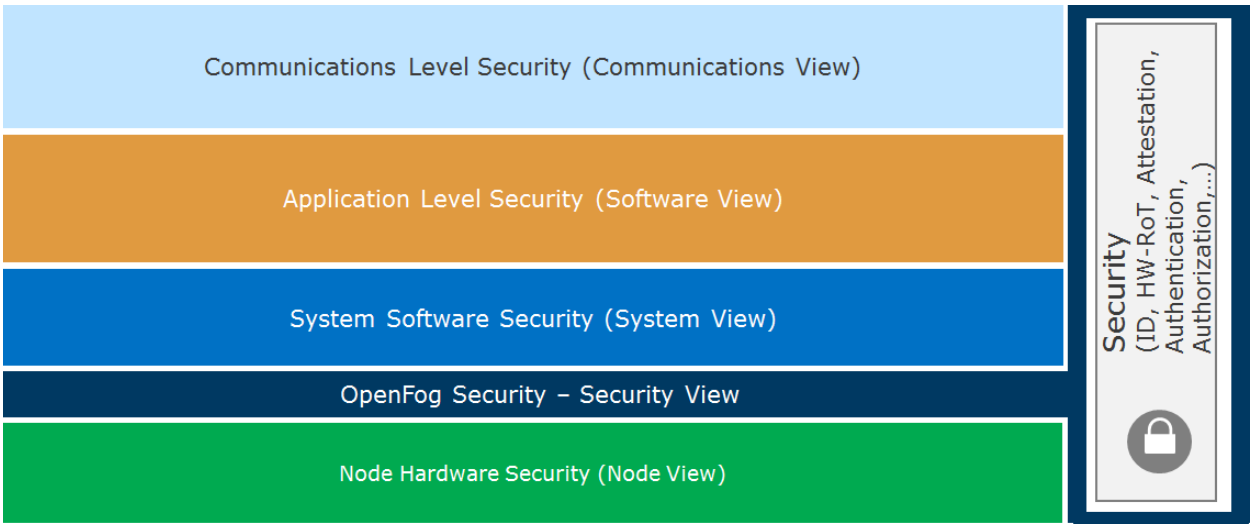


Figure 15 OpenFog Security Layers

5.4.2.2 Threat Models and Threat-Based Design

Fog deployments require security protection mechanisms implemented in a particular design dependent on the threat model and the value of the asset being protected for that fog node. In the architecture, we assume that attackers are actively looking to compromise the assets, looking for the most vulnerable entry point. The objective is to provide sufficient security for the threat model and upgrade the security, as needed, over time. **Error! R eference source not found.** lists some (non-exhaustive) examples of threats and attacks towards fog nodes. When designing for threats in a fog environment you need to understand the various views of each and the overall deployment model that is being addressed. In many fog deployments, you cannot assume physical possession is out of scope and that further adds requirements onto fog platforms.

Different use case models, even within a single use case vertical, may require different types and levels of security. Many different types of assets need to be protected against different levels of threats specific to their intended use and location. Assets may include:

- Information Technology infrastructure
- Critical infrastructure
- Intellectual property
- Financial data
- Service availability
- Productivity
- Sensitive information
- Personal information
- Reputation

Threat: An action (attack) that can damage an asset or cause a security breach.

Threat model: Specifies the types of threats that the system defends against, as well as threats that are not considered. A threat model should clearly specify what assumptions are being made about the system, its users, and potential attackers. A threat model need not describe the details of the attacks that it protects against. It should specify whether attacks on the operational system in the field are the only ones considered, or considers attacks during the development by an insider. Insider attacks are typically much harder to protect against, because the designer can build in a back door that can be exploited later.

Threat Categories / Intents / Attack Venues	Confidentiality Violation	Integrity Violation	Authentication Violation	Availability Violation	Privacy Violation
	Leaking information through overt/covert channels	Modifying data/code without proper authorization	Masquerading one entity as another entity	Rendering resources unreachable /unavailable	Leaking sensitive information of an entity (incl. identity)
Insider Attacks	Data Leaks	Data Alteration	Identity/Password / Key Leaks	Equipment Sabotage	Data/Identity Leaks
Hardware Attacks	Hardware Trojans, Side Channel Attacks	Hardware Trojans	Hardware Trojans	Radio Jamming, Bandwidth Exhaustion	Hardware Trojans, Side Channel Attacks
Software Attacks	Malware	Malware	Malware	DoS/DDoS, Resource Depletion	Malware, Social Network Analyses
Network Based Attacks	Eavesdropping	Message / Transaction Replay	Spoofing, Man-in-Middle Attacks	DoS/DDoS, Subnet Flooding	Traffic Pattern Analyses

Figure 16 Example Threats and Attacks

5.4.2.3 Confidentiality, Integrity, and Availability

There are three cornerstones of the fog security perspective:

Confidentiality: The prevention of the disclosure of secret or sensitive information to unauthorized entities.

Integrity: The prevention of unauthorized modification of protected data or code without detection.

Availability: The ability of a system to continue to provide service to authorized entities at the agreed upon service level as needed. Availability, in the security sense, has to consider external attacks, such as Denial of Service, rather than just hardware and software failures and faults.

5.4.2.4 Access Control

Restricting access to resources (objects) to only those allowed to access that information is key to building a secure system. Access Control encompasses Authentication, Authorization, and Accounting (AAA).

- **Authentication** answers the question "Who are you?" Authentication is used between humans and machines and between machines and machines.
- **Authorization** answers the question "What are you allowed to do?"
- **Accounting** refers to the record keeping and tracking mechanisms implemented in the system. This includes tracking and logging access to system resources.
- **Physical access** security to insure only authorized people are allowed to touch fog hardware is another aspect of this security mechanism.

5.4.2.5 Privacy

Privacy is the right to decide how one's information is used. (Confidentiality is the obligation to protect secret or sensitive information.) Privacy is a property of data. Fog systems must allow users to specify the privacy attributes of the data that they own on the system. In a multi-tenant system, this may involve specifying both privacy and sharing rights among tenants. If fog systems capture data for analysis at the edge, the privacy of that data must also be accounted for in the deployment.

5.4.2.6 Identity and Identity Protection

Public-key ciphers can be used for establishing a longer-term cyber identity, e.g., for authentication. In public-key cryptography, keys come in matched pairs (public key and private key) for each user, entity, computer, or subject. The private key must be accessible only to the subject and represents the subject's digital identity in cyberspace.

Hashes can be used to verify the integrity of code modules by taking the hash of the good known code module and using that to identify the module

(like a unique global name). The same code module infected with malware would have a different identity or hash value. Two identical code modules or data, but with different filenames, would have the same hash value, which means the same identity.

The private key of someone's key pair is like their digital identity. For example, an operation executed with Alice's private key can authenticate the person as Alice. Private keys must be kept confidential in order to protect someone's digital identity.

5.4.3 Manageability Perspective

Many fog computing deployments involve machine vision, and associated human-like functions. As such, they have the ability to see, respond, remember, move, and make autonomous decisions in order to participate in other fog services. This range of actions requires a higher level of manageability than a traditional static model. In addition, fog nodes may be deployed in a wide range of locations: remote, fixed and non-fixed, and environmentally harsh conditions.

Fog computing is driving changes to manageability service compared with in traditional IT and OT management systems.

5.4.3.1 Manageability Interfaces

Manageability interfaces deployed for fog computing should support In Band (IB) or Out of Band (OOB) management interfaces or both. There are pros and cons for using IB or OOB manageability, but the best choice is generally dependent on the given deployment scenario. However, both may be used as we move to autonomous levels of manageability.

- **IB manageability:** This refers to the manageability interfaces that are visible to the software and firmware running on a given system. IB manageability interfaces may communicate with a system service processor (SSP) or a baseboard management controller (BMC), if they are present on a given hardware node. However, this is not required. In some scenarios, IB manageability may be run on a separate OS thread or periodic service. Many systems use "heartbeats" to manage the health of a given system. If the IB management thread does not send the heartbeat, a higher-level management entity may restart or alert service systems to address the issue.
- **OOB manageability:** This refers to manageability interfaces to a manageability subsystem that is not running on the host operating system. These are generally discrete manageability systems that can survive and manage systems in all power states. Specifically, when a fog platform is powered off, and software is not executing on the host

platform, an OOB manageability interface can still be used communicate with the platform and perform things like inventory control, system health, power it one, etc. Examples include BMCs as defined by the IPMI specification. OOB management has potential security advantages, especially for business-critical IoT applications.

5.4.3.2 Management Lifecycle

Even the smallest fog node has a management lifecycle. The figure below shows the main components of a management lifecycle from a manageability perspective.

In all systems, there are one or more management agents. These may be implemented as discrete systems or software services. The purpose of the management agent is to ensure that each element of a fog node successfully goes through the management lifecycle. Automation is important during all phases of the lifecycle, because human intervention is impractical for large fog networks.

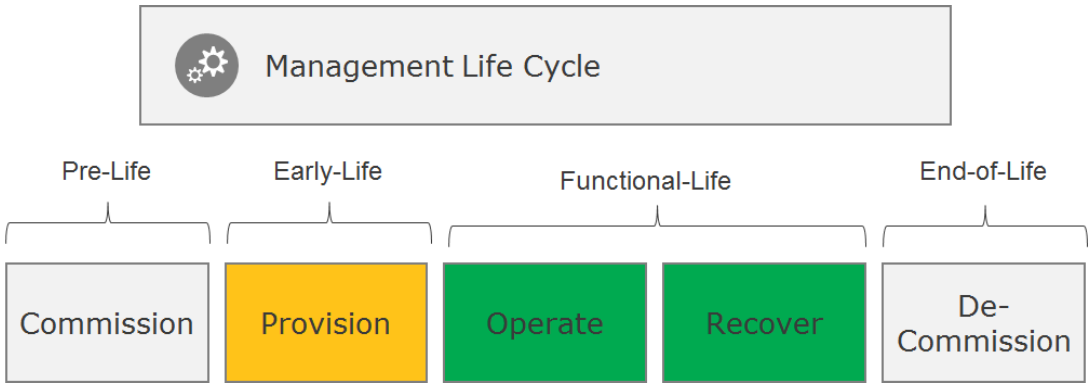


Figure 17 Management Lifecycle

Commission: This is the earliest phase of a fog platforms lifecycle. When a managed entity is commissioned, certain actions are required prior to provisioning. These include identification, certificates, calibration of time, etc. In addition, at this state the managed entity must:

- Include security that can be attested to and trusted in future phases of the lifecycle.
- Include RAS (reliability, availability, and serviceability).
- Be agile in data collection and monitoring.
- Open in order to allow control and provide visibility into its resources.

Provision: When a managed entity begins its early life in a fog node it must be enrolled. This includes discovery, identifications, advertisement of features and capabilities, trust, and deployment of features. The managed

entity must also be scalable upon provisioning. It must have the ability to support a multitude of hierarchies.
Operate: When a fog node is in normal operation. Manageability requirements cover all aspects of reliability, availability, and serviceability.
Recovery: When a fog node is operating out of expected norms, it must be autonomous in its ability to recover. It should attempt to self-heal and perform recovery operations. Other fog nodes may also assist with the recovery action, which is why the architecture defines both OOB and IB manageability interfaces.
De-Commission: Since many aspects of fog nodes may have Personally Identifiable Information (PII), the architecture specifies an ability for cleansing all aspects of hardware. This includes the ability to decommission fog node instances and re-use them for another deployment. It includes ways to securely wipe out all of the Non-Volatile (NV) storage so that future applications may not access the previous tenant's data.

5.4.3.3 Management Layer

As shown in management lifecycle, the fog management layer has many responsibilities, including automated discovery, registration, and provisioning of endpoint devices. Discovery services provide an efficient method for finding, identifying, onboarding, and managing components in the fog infrastructure. Both IB and OOB discovery methods are used. IB services are usually discovered using operating system or software agents. OOB discovery is usually performed through wireless, SMBus, or I2C interfaces which are easier to maintain during low power system states. The purpose of discovery is to gain a full understanding of the endpoint device's resources, establish a health baseline, and ensure correct operational state until the element is decommissioned.

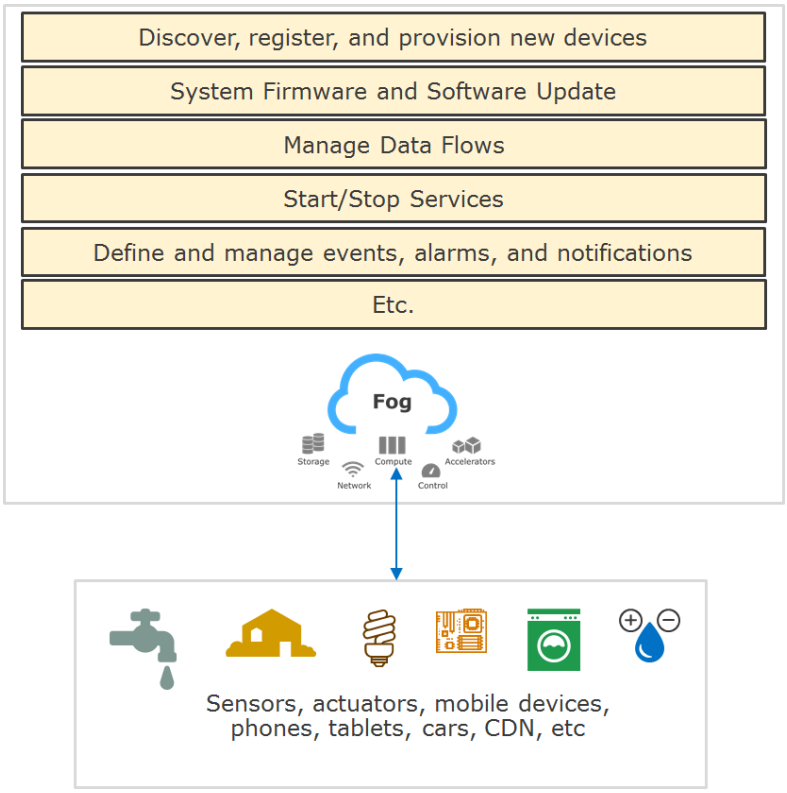


Figure 18 Management Layer

The most commonly used manageability aspects of a fog node are system software and firmware update and remote alerts of abnormal system operation. As fog-based systems often operate in harsh or remote environmental conditions, it is usually a requirement to provide "over the air" (OTA) firmware and software updates. The manageability layer is responsible for these updates.

5.4.4 Data, Analytics, and Control

The traditional way of delivering analytics is no longer efficient or, in some cases, even possible using traditional sensor to cloud models. This is due to the large volume of data that must be captured, stored, and transported to the data center or cloud for additional processing and analysis by large-scale business applications. As we look deeper into business and technical processes, more granular data elements will be needed to create actionable business knowledge from information. This data journey is an evolving vision for most companies and institutions. Some are interested in taking the full journey. Some just want to understand the current state of their operations and want Descriptive Analytics (the analysis of what's happening or what happened). Others are interested in Diagnostic Analytics, which entails root-cause analysis. Predictive Analytics uses all the knowledge from the previous

analysis and combines it with other knowledge about processes and tools to understand what will happen. Eventually companies may be interested in Prescriptive Analytics, which enables processes to optimize themselves.

The more that companies want to understand about their operations, the more data, compute, and data resources they will need. As we have shown multiple times, with fog computing, we have the tools to capture, store, analyze, and transport only relevant data. This is done by having the intelligence of upper layer data-center or cloud applications embedded as close as possible to the data-source. This means that with integration between the data-source and the Business Intelligence analytics applications, the "network" or the "edge" will capture the data from the source, process it for local purpose-specific analytics, deliver an action back to the process while at the same time sending same or other datasets to the Data Center or Cloud for further "business or operations-specific" processing. The hierarchical nature of fog helps this, allowing different components of the analytics algorithms to operate at different fog layers. We see this as cross cutting concern because it has to happen at the right layer as dictated by the scenario being addressed.

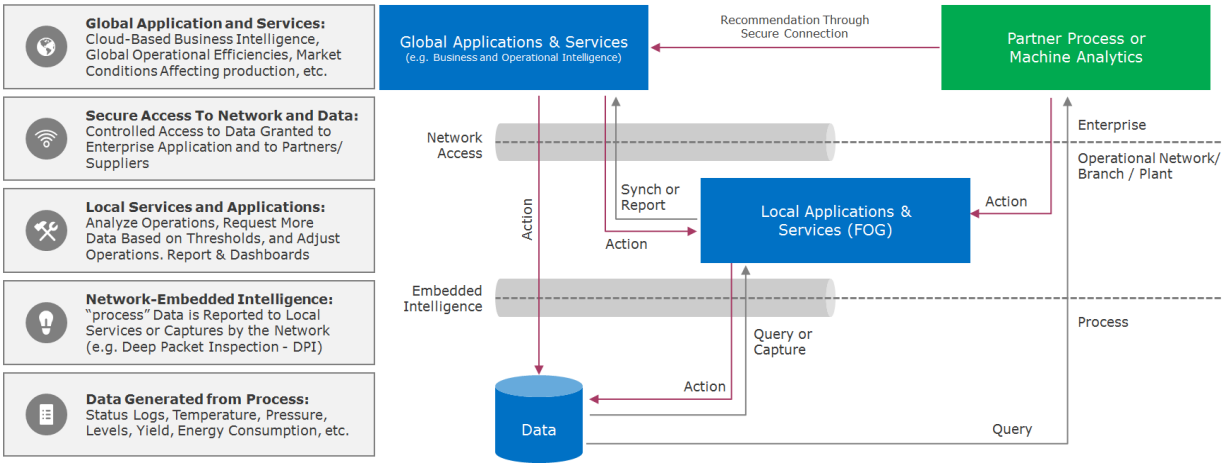


Figure 19 Business Intelligence

The figure above shows the integration of and open data exchange across all elements of the business process. This integration and exchange is necessary for the success and accuracy of business intelligence analysis. The OpenFog Consortium is working on developing (and evolving) various solutions around security and identity to facilitate this type of communication within an enterprise and between the enterprise and its partners and suppliers. Business intelligence will depend on well-defined flows, secure boundaries, facilitating data capture and exchange among the various data processing elements, and the data science capable of making sense of it all.

5.4.5 IT Business and Cross-fog Applications

Fog applications and services should have the flexibility to span and interoperate with various levels in a fog hierarchy. This is a foundational aspect of fog computing that enables a multi-vendor ecosystem. In addition, the data that is collected or generated by one fog node should be shareable with other nodes in the hierarchy. The figure below shows a cross-fog application spanning east to west nodes (it would also be able to span north to south).

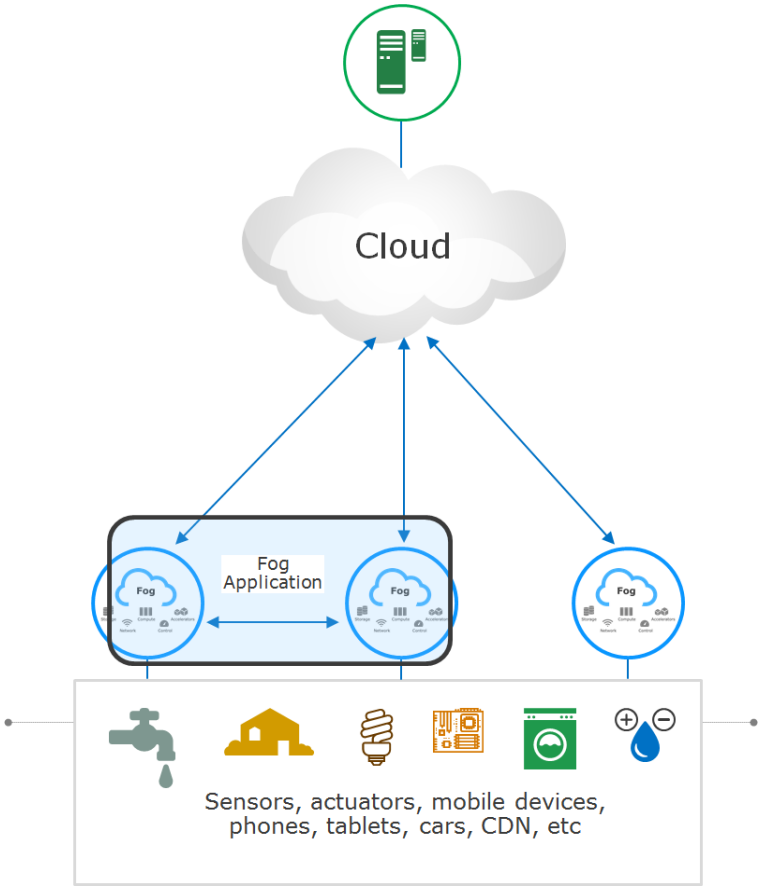


Figure 20 Cross Fog Application

Cross-fog applications require that we have an understanding and adoption of smart objects and associated data models. These are critical to those applications for interoperability and additional value creation.

5.5 Node View

As previously described, the OpenFog RA description is a composite of multiple stakeholder views. The node view is the lowest level view we currently utilize in the architectural description. The stakeholders involved in formulating the viewpoint (and subsequently this view) are the system on a

chip designers, silicon manufacturers, firmware architects, and system architects.

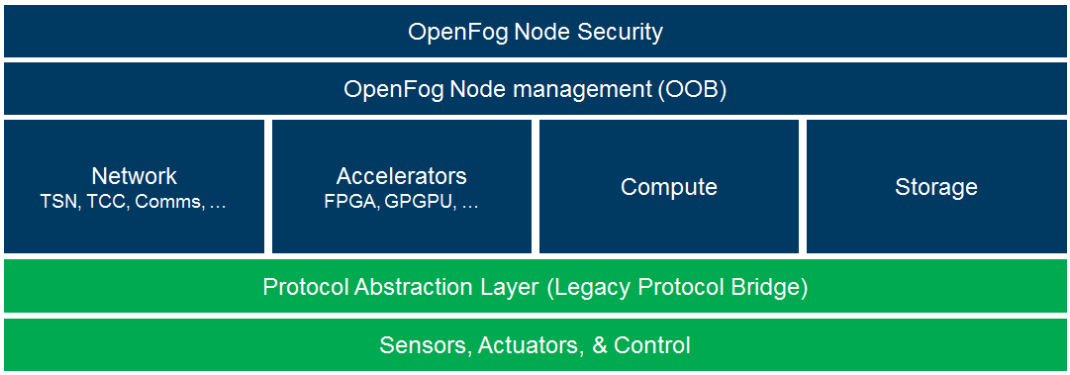


Figure 21 Node View

Before bringing a node into a fog computing network, the following aspects should be addressed:

Node Security: As described previously, node security is essential to the overall security of the system. This includes protection for interfaces, compute etc. In many cases a node will act as a gateway for legacy sensors and actuators to higher-level fog functions and therefore can act as a security gateway. It is important to note that Node security is shown as both a vertical perspective as well as a horizontal requirement for this view. This is an important concept as security must be considered at all levels from silicon to software.

Node management: A node should support the management interfaces, provided by the node being managed. Management interfaces enable higher-level system management agents to see and control the lowest level node silicon. The same management protocol can be used across many different physical interfaces.

Network: Every fog node must be able to communicate through the network. Since many fog applications are time sensitive and time aware, some fog computing networks may need to support Time Sensitive Networking (TSN).

Accelerators: Many fog applications utilize accelerators to satisfy both latency and power constraints as it relates to a given scenario.

Compute: A node should have general purpose compute capabilities. It is also important that standard software (e.g., Commercial off the Shelf or open source) be able to run on this node. This enables a higher level of interoperability between fog nodes.

Storage: An autonomous node must be able to learn. Before any learning is possible, it must have the ability to store data. Storage devices attached or embedded to this node need to meet the required performance, reliability, and data integrity requirements of the system and scenario. In addition,

storage devices should also provide information and early warnings about the health of the media, support self-healing properties, and support ID-based performance allocation. Some kind of local, standalone storage will be required for local context data, logging, code images, and to service applications that run on the node. There will often be more than one kind of storage required – e.g., local hard disk, SSD, and secure storage for keys and other secret material.

Sensors, Actuators, and Control: These hardware or software-based devices are considered the lowest level elements in IoT. There could be several hundred or more of these associated with a single fog node. Some of these are dumb devices without any significant processing capability, while others may have some basic fog functions. These elements generally have some amount of connectivity, and include wired or wireless protocols, such as I2C, GPIO, SPI, BTLE, ZigBee, USB, and Ethernet, etc.
Protocol Abstraction Layer: Many of the sensors and actuators on the market today are not capable of interfacing directly with a fog node. The protocol abstraction layer makes it logically possible to bring these elements under the supervision of a fog node so that their data can be utilized for analytics and higher level system and software functions.

Abstraction is also key to multi-vendor interoperability for both IoT things and fog nodes. Well-known inter-element interfaces supply a layer of abstraction. They enable vendors to share meta-data of fog architectural elements they support, which in turn fosters multi-vendor data interoperability and service composability. When meta-data is exposed, it also may be used for cross-layer optimizations, for example to optimally route data between fog nodes using information-centric networks (ICN) or to create dynamic fog topologies as with software-defined networks (SDN).

Future versions of the OpenFog RA will describe “Minimum Viable Interfaces”, which will include many more details about protocols and abstraction layers. Following subsections contain more details of the aspects mentioned above, except those of the sensors, actuators, controls and their internet protocol abstractions. Please refer to the security appendix for further discussions on IoT device connectivity and security issues.

5.5.1 Network

Fog nodes generally are of most value in scenarios where data needs to be collected at the edge and where the data from thousands or even millions of devices is analyzed and acted upon in micro and milliseconds. In these scenarios, various networks facilitate communication within the fog nodes to

sensors and up to higher levels in the hierarchy up to and including the cloud.

The network should provide the scalability, availability, and flexibility required by the communication pattern or process. The network should also provide whatever QoS is required to prioritize critical or latency-sensitive data and even guarantee delivery. QoS should be addressed at the lowest level, hence Node view, to provide guarantees of QoS. This enables higher level views of system providers (System view) and applications (Software view) to build upon a stable foundation.

In the following sections, we will explore various network elements from the point-of-view of a fog node's connectivity and communications requirements.

Note: Depending upon the deployment scenario, fog nodes will most likely exist within a network element, such as an access point, gateway, or router. In the architecture, we assume that the network requirements will be the same regardless of the placement of the fog node. This clearly will change based upon deployment and we will refine this through Testbed and other open deployments.

5.5.1.1 Wired Connectivity

The network connectivity model for a fog node will depend on the node's purpose and location. For example:

- A fog node in a factory used to gather and analyze manufacturing process data will most likely be connected to upper and lower layers using a wired network.
- A fog node used for gathering and analyzing personnel locations will most likely be connected to the sensor using a wireless network.
- Internal connections within fog nodes will most likely be connected using RDMA and other low latency interconnect technologies.

There are many standards, types, and interfaces for physical connectivity that can be utilized to connect a fog node. Physical connectivity is typically one or more Ethernet links supporting speeds ranging from 10 Mbps to 100 Gbps, supported on copper or fiber links to serve various reach requirements. We typically see copper connectivity used for links up to 100 meters in length and supporting speeds ranging from 10 Mbps up to 1 Gbps.

For connections requiring higher speeds and longer reach, optical fiber cables may be used. Optical fiber support different wavelengths and transmission modes to support the required distances and capacities. For

example, a single mode optical fiber cable (utilizing a single ray of light) supports longer distances than a multimode fiber (utilizing multiple rays and multiple wavelengths).

For connecting a fog node to IoT devices or sensors, there are a variety of standards and interfaces that are vertical or solution dependent (non-Ethernet protocols). For example, in an industrial environment, the fog node may be required to support a Controller Area Network (CAN) bus or other fieldbus standards for communicating with lower layer applications and processes.

For industrial automation uses, guaranteed data delivery is critical. This type of networking (usually using Ethernet) is called Time Sensitive Networking (TSN) also known as Deterministic Ethernet. TSN uses standards-based time synchronization technology (e.g., IEEE 1588) and bandwidth reservation (class-based QoS) to prioritize control traffic in a standard Ethernet environment.

If a fog node is required to interface with devices in industrial automation, automotive, or robotic environments over Ethernet, support for TSN may be needed.

5.5.1.2 Wireless Connectivity

Wireless connectivity is an essential part of the Internet of Things in particular, and the digital transformation in general. Wireless connectivity provides flexibility and enhances efficiency and productivity. Wireless interfaces come in a variety of protocols, standards, and mechanisms. The quality of connectivity is dependent on many conditions including but not limited to the level of flexibility, mobility, reach, availability, power constraints, and environmental or geographical conditions. For the various IoT applications envisioned for fog computing, wireless connectivity seems to be especially beneficial for southbound communication (sensor-to-fog node), but will also be used for fog node-to-fog node and fog-to-cloud interconnects.

Wireless support at the fog node will be dependent on a variety of parameters, including:

- Function and/or position in the hierarchy.
- Mobility is an essential property to support in-vehicle fog nodes and many classes of geographically-distributed IoT endpoints. Wireless interfaces are the only practical way to reach them.
- Coverage required to satisfy the deployment requirements.

- Data volume (throughput) and velocity (data transfer rates).
- The form factor required to support various types of antennas, modules, or transceivers.
- If the fog node is expected to receive, process, and relay information upstream to other layers in the hierarchy at a high rate, then the energy source (including efficiency, delivery, and dissipation considerations) becomes a significant design factor. Similarly, if transmission and processing rates are lower, then battery, harvested energy, and rechargeable sources may be used to fulfill design parameters.
- Wireless support is highly dependent on the environment, i.e., interference. For example, deploying wireless technologies in noisy environments or around highly reflective metallic surfaces may affect the performance required of the fog node. This is also true for maritime and other areas where certification of physical infrastructure requires that structural integrity of a fog node is not compromised by antenna attachments/cable attachments. In these scenarios, secure and reliable wireless communication is a requirement.
- Using a licensed spectrum normally requires a fee for accessing the frequency ranges, while, in most cases, using an unlicensed spectrum may be low or no cost.

Wireless connectivity can be grouped into three major areas: Wireless WAN (WWAN), Wireless LAN (WLAN), and Wireless Personal Area Networks (WPAN).

Note: It is not uncommon to refer to some of the types of communication technologies as Wireless Metropolitan Area Networks (WMAN). For the purpose of this discussion, and due to the interchangeable nature of WWAN and WMAN use-cases, we use WWAN as a superset of both. **Wireless WAN (WWAN):** WWAN technologies are used when large geographic area coverage is required. A variety of protocols and standards exist and we list the following standards a fog node may be required to support to communicate with the other devices or nodes within the same network:

- Cellular technologies, including 3G, 4G LTE, and 5G, feature high data transfer rates (>1Gbps). They also cost more (as a licensed spectrum) and are less power efficient. The majority of cellular communication is standardized by the Third Generation Partnership Project (3GPP).
    - Note: 5G promises higher speeds, higher capacity, and much lower latency than the current cellular systems. With 5G it may be possible to deliver double-digit Gbps speeds (10+ Gbps). 5G promises to facilitate higher adoption of IoT solutions and

devices. Fog nodes aggregating data from cars, mobile devices, and sensors will most likely need to support 5G.

- Fog nodes may be required to support cellular technologies for southbound or northbound traffic or both. Fog nodes may use cellular technology to back-haul sensor traffic. Most types of mobile fog nodes will use cellular northbound interfaces. Many software defined radio technologies that physically connect into the fog node address some of these scenario requirements.
- Narrow Band IoT (NB-IoT) is a 3GPP standard that addresses a variety of IoT applications and requirements and delivers on the promise of long-range and lower power requirements. NB-IoT is not widely available yet.
- Low-Power Wide Area Networks (LPWAN) have low data transfer rates, higher power efficiency, and are low cost. Proprietary LPWAN implementations are being tested by various organizations, such as the LoRa Alliance and Sigfox. LPWANs are currently being investigated for agriculture applications because of their ability to cover large areas of farming and rural lands.

Wireless LAN (WLAN): WLANs utilize a variety of topologies and protocols, but WLAN has become synonymous with WiFi. WLANs are a good communication choice for smaller geographical areas, often within a building or campus. Depending on the number of access points and the density requirements, WLANs may also be used in stadiums, manufacturing plants, and oil and gas refineries and fields. The following are some examples of WLANs that may be supported by the fog node:

- WiFi (WLAN) is defined by a group of IEEE 802.11 standards. These standards address multiple requirements and challenges for the environment in which they are deployed. They support data transfer rates from few Mbps up to multiple Gbps. The most common standards are IEEE 802.11a, b, g, n, and ac. IEEE802.11ac is the latest in a series of evolutions of the IEEE 802.11 standards; it supports higher densities and transfer rates.
- The IEEE 802.11 workgroups are currently working on new solutions to address the need for higher capacity, density, and speed, especially for IoT use cases. For example, the IEEE 802.11ax is expected to bring higher speeds and capacities beyond the capabilities of 802.11ac. The 802.11ah is tailored for IoT use cases requiring low power consumption and longer ranges. IEEE 802.11p will define standards for Vehicle-to-Vehicle as well as Vehicle-to-Roadside infrastructure communication.
- Future developments in free space optical communications such as Li-Fi hold promise as options for fog wireless networking.

Wireless Personal Area Networks (WPAN): WPAN is characterized by a short communication range, low power consumption, and low cost. WPANs may be used with wearable devices and home management systems. WPAN includes the following technologies:

- Bluetooth: Characterized by short-range communication. Specifications and standards managed by Bluetooth Special Interest Group (SIG).
- Infrared (IR): Characterized by line of sight wireless communication over IR light waves. Specification provided by IrDA (Infrared Data Association).
- ZigBee: Characterized by low power consumption, short range (up to 100 meters, given the right environmental conditions), and low data transfer rates.
- Z-Wave: Characterized by RF signaling and control mostly used in home automation.
- IEEE 802.15.4 (Low Rate WPAN) also applies to WLAN use cases. It is also the standard that defines Layers 1&2 of the OSI model.

Near Field Communication (NFC): NFC is a technology that may be used when fog nodes support devices that need to communicate in very close proximity. NFC technologies have been used in logistics and supply chain solutions for quite some time; now they're being used in vertical markets, including frictionless solutions for retail, agriculture, and healthcare. Solutions like Passive RFID, which also utilizes NFC, are used for asset tracking and physical access.

Wireless connectivity to the fog node allows the ability of various sensors and data to flow into the node where it will be processed. As node capabilities grow, it enables higher level secure communication functions to be utilized in the architecture.

5.5.1.3 Network Management

As the number of sensors and data sources increase, the need to manage all assets, nodes, and resources increases in importance. The ability of fog nodes to be supported by out-of-band (OOB) network management will help manage resources, security, health of the fog node, and the ability to adapt to changing conditions in the environment. The protocols and mechanisms available to manage sensors, nodes, and network devices vary based on the communication protocol used and the availability of connectivity options and CPU/memory resources. In some cases these are separate managed networks; in other cases the management communication is sent on the host network. There are multiple ways information is provided to the node,

and to ensure continued secure, reliable, and safe operation, it is important that these things and associated networks are capable of being managed with both sophistication and simplicity.

5.5.1.4 Network Based Security Threats and Mitigation

The fog node needs to be protected from various network-based security threats, which may include:

- Denial of Service attacks
- Intrusion
- DNS spoofing
- ARP spoofing or poisoning
- Buffer overflows

Fog nodes may not always be capable of protecting against these types of attacks. They will most likely depend on the network or adjacent devices to help protect them. The following are some of the common examples for network devices that help protect the fog node:

- Firewalls for blocking unauthorized access.
- Intrusion Prevention Systems (IPS).
- Secure Remote Access using Virtual Private Networks.
- Behavior-based anomaly detection appliances or software.

Several instances of massive DoS attacks that used network attached IoT devices could have been detected much faster and potentially mitigated by utilizing fog driven network security.

Please refer to the Section 10.1.3.1 for detailed discussions on the network and data security aspects of OpenFog architecture.

5.5.1.5 Network Design Considerations for Fog Nodes

Whether a fog node is being installed into an existing brownfield or new greenfield networked environment, the following design considerations should be kept in mind:

- Physical or virtual implementation of fog node capabilities depends on capacity and policy needs. For example, the ownership of the physical infrastructure could be the factor that determines access to the virtual elements supported within the infrastructure.
- Fog node-to-node communication considerations:
    - Direct or indirect

    - Distance between any two nodes, which influences energy use, bandwidth, cable complexity, and cost
    - Requirement for state preservation between two nodes (in a backup or high availability scenario)
    - Type of communication interface and protocol
- Capacity planning:
    - Start with the end state in mind when architecting for the scenario
    - Understand the impact of new traffic patterns on the fog node and the network as a whole
- Streaming data readiness:
    - Volume of data expected
- The frequency of maintenance and upgrades
- Convergence with IT: Choosing Ethernet and IP for node-to-things communication may allow for easier integration with the IT environment and contribute to the efficiency of the system.

5.5.2 Accelerators

In addition to traditional CPUs, some fog nodes, especially those engaged in enhanced analytics, require CPU throughput in excess of what can be economically (power or processing efficient) provided by standard current server and enterprise CPU chips. In these cases, accelerator modules will be configured next to the processor modules (or tightly integrated) to provide supplementary computational throughput. Here are some examples:

- Graphics Processing Unit (GPUs) often contain thousands of simple cores. For applications that can efficiently exploit their massive parallelism, they can be an order of magnitude faster and provide significant power and space savings. Multiple GPUs can be equipped on each standard CPU. However, to achieve this capability power delivery and physical and electrical connection to the node would need to be increased which can increase the overall node power consumption.
- Field Programmable Gate Arrays (FPGAs) are large collections of gate-level programmable hardware resources. They can be configured with custom logic designs to solve very specific problems very efficiently. However, depending upon deployment the additional power reduction compared with other accelerators may require additional lower level knowledge (e.g. VHDL). In many cases FPGA provide better power efficiency compared with discrete GPUs.
- DSPs are specialized processors optimized to operate on signals. Some DSPs are general purpose, while others are optimized for special functions, like video compression and manipulation.

When choosing an accelerator for a given node, you must balance the following:

- Interfaces to the node (electrical throughput and power delivery).
- General applicability to multiple scenarios. This means the dynamic ability to change (programmability pillar) the accelerator to address a new use case.
- Dynamically changing requirements.
- Environmental constraints.
- Higher level software programming interfaces and API support, especially the orchestration needed to discover an accelerator's existence and capabilities

The programmability pillar implies the node's ability to be flexible in how it is defined to address a given problem. This late binding is especially important in many of the areas of interest to the OpenFog consortium.

5.5.3 Compute

As more and more data is processed at the edge, this will increase the computational requirements at the true edge of the network. Having general purpose computation at the edge will continue to be important. Additionally, larger amounts of system DRAM paired with this compute will also grow in importance. To ensure higher reliability and accuracy of computation we will start to see ECC memory start to become more prominent in the compute. Another key importance of fog computing is that deployment lifecycles may be longer so it is important that long life and compute performance is calculated to be sufficient throughout a deployments lifecycle. Compute is likely to be implemented as one or more multi-core CPU. While other architectures exist, it is important to balance the programmability aspects when considering different architectures.

When considering a compute element, it is important to understand the environmental conditions it must correctly operate under. In many fog deployments, the compute must continue to operate well beyond 70 degrees C. In fact, the harshest environments of up to 100 degrees C is not uncommon in many parts of the world.

The compute function embodies many requirements for fog nodes. Following are some examples of computational requirements:

- Some multi-tenant installations will dedicate whole cores to each of their most critical applications. This capability may be required to ensure QoS.

- Memory management units may be required in fog to manage large virtual memory space, to isolate platform from application space, and to isolate applications from each other in multi-tenant environments.
- High performance I/O subsystems are required in fog in order to connect each CPU with its associated accelerators, storage, and network peripherals.
- In some fog node designs, the hardware root of trust is located within the CPU complex itself and verification of code only happens after the CPU verifies the signature.

5.5.4 Storage

Many types of storage will be required in fog nodes. As fog computing continues to emerge we will see storage tiers typically only seen in datacenters emerge on nodes as they collect and process this data across the hierarchy. This includes:

- **RAM Arrays:** As data is created from sensors, the node will need to operate on that data in close to real time operation. RAM arrays satisfy this requirement versus additional latency when accessing non-volatile storage. Many fog nodes will also have on-package memory to satisfy the latency aspects required for certain scenarios.
- **Solid State Drives:** Flash-based storage may be used for the majority of fog applications because of its reliability, IOPs, low power requirements, and environmental robustness. These include PCIe and SATA attached SSDs. Additionally, new classes of solid state media are beginning to emerge with new programming models. These include 3DXpoint and NVDIMM-P.
- **Fixed Spinning Disks:** For large, cost sensitive storage applications, fog nodes may contain rotating disks, sometimes arranged as Redundant Array of Inexpensive Disks (RAID) arrays.

The actual storage medium chosen depends upon the use case; within a given fog node, there will typically be a hierarchy of storage options. Ultimately, storage devices need to meet the cost, performance (IOPS, bandwidth, and latency), reliability, and data integrity requirements of the system. In addition, storage devices in fog computing need to support the OpenFog pillars, particularly the security and RAS pillars. However, the biggest advancements will come as flash based storage technologies continue the march to cost/byte and access latency times of DRAM.

Storage devices should provide support for encryption and key management and authentication by supporting standards such as AES-256 and TCG Opal

etc. The storage device should also provide real-time information and early warnings about the health of the media and support self-healing properties. Finally, in virtualized fog computing environments the storage device should support ID-based performance allocation by providing adjustable storage resources (IOPS or bandwidth) to specific applications or virtual machines. Supporting data encryption at rest is also important in most fog deployments as they will be deployed in areas where physical protection mechanism seen in the data center are no longer true.

5.5.5 OpenFog Node Management

OpenFog node management refers to manageability systems that are not running on the host operating system. These are generally discrete manageability systems that can survive and manage fog nodes in all power states. They are also sometimes called Hardware Platform Management devices (HPM).

Most fog nodes will include a HPM that is responsible for controlling and monitoring the other components inside the node (e.g. storage, accelerators, et al). The HPM is typically a small auxiliary processor on the main CPU or motherboard. It has various sensors and monitoring points to track variables like temperature, voltage, current, and various errors. These readings may periodically be reported to external RAS system. If serious errors are detected, alarm notifications can be escalated by the HPM subsystem.

The HPM system is also responsible for controlling the internal configuration of fog nodes. It may set communication parameters like IP addresses and line speeds. It can configure new hardware modules. If modules fail, it can isolate them and attempt to recover their functions. The HPM subsystem also cooperates with a trusted component in the chain of of trust to securely download software updates for the entire node. Sensors associated with the physical operation of fog node hardware, including ambient temperature, airflow, fan speed (if used), supply voltage, supply current, moisture, cabinet door tamper, etc., also send data through the HPM subsystem.

In many deployments because the HPM can operate out of band with the main computational elements, it too will have its own TPM and go through a secure boot process. The HPM just like other entities should support a HW root of trust.

5.5.6 OpenFog Node Security

As described in the Security Perspective, it is important to perform a security analysis and threat assessment of the fog implementation in order to properly identify the needs the fog node. Once this task is accomplished, you should have the information needed to determine the appropriate physical security measures, the optimal method for establishing and maintaining trust, and what type of policies to put in place in order for the fog node to securely manage and respond to its environment.

5.5.6.1 Physical Security and Anti-tamper Mechanisms

The level of physical security supported by a fog node should be aligned with the security policy for that device and threat level. This will depend on how difficult it is to access system components and what the consequences are if the system is breached. The location of the fog node and the degree of physical access available in that location will play a role in the evaluation. Fog nodes located in open public spaces such as shopping malls, street corners, utility poles, and even in a personal vehicle will provide greater opportunity for a physical attack. Note: There may be industry specific standards and requirements for providing physical security for these devices. These are not addressed here.

The goal of anti-tamper mechanisms is to prevent any attempt by an attacker to perform an unauthorized physical or electronic attack against the device. Anti-tamper mechanisms can be divided into four groups:

- Resistance
- Evidence
- Detection
- Response

It is important that legitimate maintenance measures should not damage the node due to anti-tamper mechanisms. To help prevent this, the node might have a special maintenance mode that can be configured by an authorized entity so that the tamper response is disabled while maintenance is in progress and then re-enabled.

5.5.6.2 Tamper Resistance

Tamper resistance uses specialized physical construction materials to make tampering with a fog node difficult. This can include such features as hardened steel enclosures, locks, encapsulation, or security screws. Implementing tight airflow channels (that is, tightly packing the components

and circuit boards within the enclosure) will increase the difficulty of using fiber optics to probe inside the node without opening the enclosure. When we look at the node view, this includes all interfaces enabled by the SoC manufacturer. Many manufacturers enable special modes of operation called manufacturing or test modes. These modes of operation must be protected from tampering from physical attack after the node is deployed.

5.5.6.3 Tamper Evidence

The goal of tamper evidence is to ensure that, when tampering occurs, visible evidence is left behind. Tamper evidence mechanisms are a major deterrent for minimal risk takers (e.g., non-determined attackers). Many kinds of tamper evidence materials and devices are available, such as special seals and tapes that make it obvious when there has been physical tampering. In the previous example, tampering could notify the HPM to ensure higher level management entities can determine tampering without physically being present.

5.5.6.4 Tamper Detection

Tamper detection means that the system is made aware of unwanted physical access. The mechanisms used to detect the intrusion typically fall into one of three groups:

- **Switches,** such as micro switches, magnetic switches, mercury switches, and pressure contacts, detect the opening of a device, the breach of a physical security boundary, or the movement of a particular component.
- **Sensors,** such as temperature and radiation sensors, detect environmental changes. Voltage and power sensors may detect glitch attacks. Ion beams may be used for advanced attacks to focus on specific electrical gates within an integrated circuit.
- **Circuitry,** such as flexible circuitry, nichrome wire, and fiber optics wrapped around critical circuitry or specific components on the board, is used to detect a puncture, break, or attempted modification of the wrapper. For example, if the resistance of the nichrome wire changes or the light power traveling through the optical cable decreases, the system assumes there has been physical tampering.
- **Mesh enclosures**, such as Gore's Tamper Responsive Surface Enclosure, are designed to protect the physical security boundary of a fog node and combine a number of tamper evidence and detection features.

All of these tamper detection mechanisms typically provide a hardware security violation signal to a security monitor when they are tripped.

5.5.6.5 Tamper Response

Tamper response mechanisms are the countermeasures taken when tampering is detected. The response to such an event should be configurable.

- **Soft Fail**: Sensitive data is cleared and a second interrupt signal is sent to the security monitor to confirm this has been done so that it can restart the processor and continue execution.
- **Hard Fail:** The actions for a Soft Fail are performed, plus the caches and memory are zeroed and the system is reset. Both lower and higher consequences may be available. The lowest consequence would be to do nothing, or the event can be logged for later analysis. An example of a higher consequence might be "bricking" the device. This means that after zeroing all sensitive data, caches, and memory, the node cannot be booted again and must be replaced.

The response to tampering needs to be understood and planned for by higher levels in the fog architecture. This dependency is often overlooked in system deployments and therefor an attack surface commonly exploited. Thus, response of Nodes must be understood by the System, and the Software running on them.

5.5.6.6 Establishing and Maintaining Trust

5.5.6.6.1 Trusted Computing Base

The Trusted Computing Base (TCB) refers to the platform hardware, software, and networking components that, if violated, would compromise the ability of the system to enforce its security policies. The more components and code that are in a TCB, the harder it is to guarantee that it is free of bugs and security vulnerabilities. It is desirable to ensure that the TCB is as small as possible to minimize its attack surface. Sometimes however, this is not achievable due to the complexity of the system required to satisfy a particular use case. Creating multiple isolated and protected regions from the rest of the system is one way of creating smaller TCBs to reduce the attack surface within a complex system environment.

5.5.6.7 Hardware Root-of-Trust

At the heart of the TCB and the security of the fog node is the root of trust. There must be no opportunity for malicious actors to highjack the early initialization or boot processes. Security needs to be anchored in the

hardware so that it cannot be circumvented. The Hardware Root of Trust (HW-RoT) is the key to a fog node's TCB.

5.5.6.7.1 Secure or Verified Boot (HW-RoT for Verification)

Minimally, fog nodes should support a HW-RoT for verification of the boot process. Secure or verified boot is an architecture for loading and verifying signed firmware images, boot loaders, kernels, and modules. It is important to note that this does not necessarily make use of a TPM even if one is present.

There are many implementations of verified boot. The process begins execution with code loaded from an immutable Read Only Memory (ROM); the compute entity in the fog node may only operate upon cryptographically signed images. Secure boot implementations may be proprietary in nature. It is recommended that system architects verify the capabilities and validate the security strengths of a verified boot implementation. There should be no mechanism to circumvent the signature method. Additionally, it is important that the HW-RoT not execute non-verified code. This includes option ROMs from PCIe devices and other elements that comprise the node view.

5.5.6.7.2 Trusted or Measured Boot (HW-RoT for Measurement)

Trusted booting is different from secure booting because higher level software can attest (programmatically verify) that firmware running is secure. One example approach, as described by TCG, defines methods to perform this function using a TPM. As code executes, it creates a cryptographic digest of the code modules that are stored in the TPM. The TPM term used for the storage space for each of these chains is Platform Configuration Register (PCR). A PCR can be thought of as a single trust chain used for some specific purpose. This is only one example of many possible implementations. Other implementations exist while still others may be described by future innovations. As for verified boot, it is recommended that system architects verify the capabilities and validate the security strengths of a measured boot implementation.

5.5.6.7.3 Securing the Boot Process

It is critical that the fog node has a method to securely establish a root of trust, and that trust is authenticated and extended through the remainder of the boot process in order to ensure that the node can be trusted.

A fog node must provide a method for ensuring that the firmware and system software have not been tampered with before components are executed. There are various methods in which this can be achieved. The selected and implemented solution to this requirement should be in

agreement with your organization's findings during the security analysis and threat assessment.

5.5.6.7.4 Identification

A fog node must be able to identify itself to other entities within the network, and entities that request services from the fog node must be able to identify themselves to the fog node. The best method for this identification is to have an immutable identifier with attestation. Attestation is the ability of a system to provide some non-forgeable evidence to a remote third-party verifier. One such approach that can enable fog systems to verify or attest to credentials of a given system while preserving privacy is the use of a Direct Anonymous Attestation (DAA) implementation.

5.5.6.7.5 Attestation

The security of systems that employ trusted processors depends on attestation (often referred to as remote attestation or software attestation). In a fog computing hierarchy, a remote attestation agent can attest to the authenticity and secure state of a fog system. Depending on the method used to create the chain of trust (i.e., either measured or verified), and therefore the trusted environment, the remote agent will attest to different properties and data. For a system which uses measurement to build the chain, the remote attestation agent would be able to remotely verify that the HW-RoT of measurement for a given fog node is correct. In either case, the objective is to be able to attest to the fact that the firmware and software running on it are known or trusted. If the firmware running on a given system fails attestation, it should not be used and remediation should occur. While there are a number of implementation options available for remote attestation, the OpenFog Consortium will work with the TCG and other standards-based approaches for remote attestation across multiple interfaces.

5.6 System Architecture View

The system view of the OpenFog RA is composed of one or more node views coupled with other components to create a platform. The stakeholders which typically create these systems to facilitate a fog deployment comprise the concerns of this viewpoint. Subsequently this view is intended to address the concerns of the system architects, hardware OEMs, and platform manufacturers. They also need to understand the Node view such that the systems they create can be deployed to address a given scenario. The following diagram shows our visual representation of a system view. We only show a singular composite image of a node embedded within it, but we also support the notion of multiple nodes being brought to create a system for

deployments that require redundancy or to satisfy other deployment requirements.

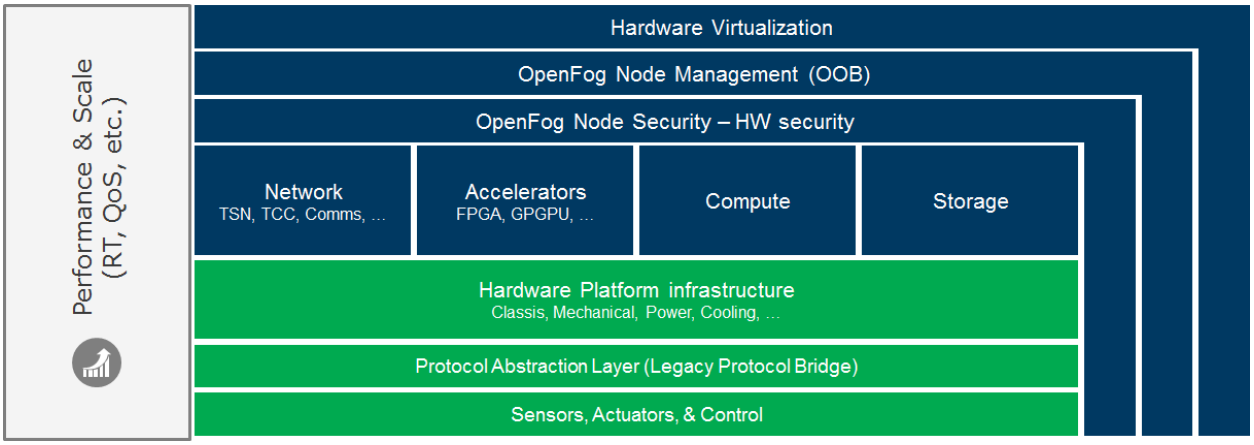


Figure 22 System Architecture View

5.6.1 Hardware Platform Infrastructure

Fog platforms must provide robust mechanical support and protection for their internal components. This first starts with components selected in the Node view and extends into the system view. In many deployments, fog platforms must survive in harsh environmental conditions, as described in the next section. Some requirements for fog platform enclosures (called the hardware platform infrastructure in the system architecture) include:

- Compliance with local regulations and standard practices.
- Protection from environmental factors (industrial or commercial temperature-rated components).
- Resistance to physical attack, vandalism, or theft.
- Acceptable size, power consumption, and weight properties.
- Functional safety requirements to protect people and things from harm.
- Mechanical support of internal components.
- Management of cooling for internal components.
- Support for node-level modularity and the ability to build and modify many configurations. This includes the ability to extend the function to address different deployments.
- Serviceability aspect of a platform.
- Acceptable aesthetics and other factors as fog platforms are deployed in the world around.

5.6.1.1 Environmental Conditions

Many fog platforms will be deployed in harsh environmental conditions. This means they should have specifications that comply with various international safety and environmental responsibility standards, such as UL, CSA, ROHS, and WEEE. Examples of safety and environmental requirements include:

- Temperature ranges required by vertical industries, such as industrial, automotive, and military. Many systems also cannot operate in temperatures higher than 60 degrees C. The following temperatures are generally accepted as norms and represented in Celsius.
    - Commercial temperatures: 0 – 70
    - Industrial: -40 – 85
    - Military: -55 – 125
- Environmental hazards, including humidity, shock, vibration, contamination, earthquake, and extreme solar load.
- International protection marking (IEC standard 60529) up to the IP 68 level (for example, a waterside node that may be subject to flooding).

5.6.1.2 Thermal

Depending upon deployment, fog platforms for harsh location deployment may be environmentally sealed. They should not require any fans or other active elements to maintain safe internal temperatures. They should not require air filters. However, due to the high-power dissipation and high packaging density of some higher performance fog nodes, especially those with large accelerator arrays, active cooling options are allowed. If forced-air cooling is used, air filters are required to reduce particulate contamination. Fans on critical fog nodes should be redundant, so that if a single fan fails, the fog node can continue at full capacity on the remaining fan(s). The thermal deployment scenario will dictate the enclosure used.
There is a strong correlation to power delivery, performance, and heat dissipation. This needs to be taken in consideration when designing the overall solution.

5.6.1.3 Modularity

Most fog nodes will be modular. On smaller designs, modularity could consist of a motherboard containing the common fixed components, and a few modular sockets into which configurable components may be installed. Most modular systems also have to trade-off capabilities for that modularity. For example, a modular adapter may require the use of a different enclosure to enable the in-field upgrade, but this modularity can also provide more levels of serviceability.

Examples of configurable components include:

- Faster CPUs
- Different RAM components
- Different storage configurations
- Configurable I/O to support both southbound edge interfaces and northbound networking interfaces.
    - Network interfaces, including varying numbers of wire line and optical interfaces with configurable physical layer options.
    - Southbound interfaces like RS232, Modbus, etc.
- Accelerators including FPGAs, etc.

Moderate sized fog platforms may substitute a backplane for the motherboard, and support modularity through the installation of boards into that backplane. This is typically seen in near edge or on-premises fog platforms. The largest fog platforms will resemble high capacity blade servers, supporting many modules, including high-end multi-socket CPU farms, large GPU arrays, petabyte class storage, and potentially thousands of I/O links.

A good example of these sizes in a deployment scenario are the fog platforms that required to support machine vision. A training system near the edge would utilize the larger fog platforms to train a neural network. The moderate sized fog platforms would take the trained model and use that for inference or recognizing images dynamically across many different video streams. A smaller fog platform could be embedded in a camera and utilize an embedded accelerator to recognize images coming off of a singular camera feed.

5.6.1.4 Module-Module Interconnect

Modular fog platforms may require interconnection between internal modules. These interconnections may be connected between a motherboard and daughterboard, or board-board over a backplane. Hundreds of GB/s may be required for module-module interconnection. The transport could be wire, optical, or other means. The connection between modules is sometimes referred to as a fabric.

In the largest fog platform, one or two fabric modules can be used as a central hub. The CPU, accelerator, storage and networking modules are used as spokes in a star topology. Ideally these interconnect facilities should conform to open standards like PCI Express or Ethernet to facilitate a widely interoperable hardware ecosystem.

5.6.2 Hardware Virtualization and Containers

Hardware-based virtualization mechanisms are available in almost all processor hardware that would be used to implement fog platforms. It may also play an important role in system security. Hardware virtualization for I/O and compute enables multiple entities to share the same physical system. Virtualization is also very useful in ensuring that virtual machines (VMs) may not utilize instructions or system components that they are not by design supposed to utilize.

Containers are a relatively new technology. Containers may offer a lower weight isolation mechanism within a fog computing environment. The isolation guarantees are only made by the OS and not fully based in silicon. This shifts the isolation requirements from the silicon to the software running on the silicon. The decision to use containers or VMs for isolation are usually based on security considerations for a given use case. We will discuss containers in greater depth in the software view.

5.7 Software Architecture View

The software view of the OpenFog RA is composed of software running on a platform that is comprised of one or more node views coupled with other components to create a system addressing a given scenario. The stakeholders of the software view include the system integrators, software architects, solution designers, and application developers of a fog computing environment. The software running on fog platforms is used to satisfy a given deployment scenario. A robust fog deployment requires that the relationship between a fog node, fog platform, and fog software are seamless.

5.7.1 Software View Layers

As shown in the figure below, the software of the fog node can be separated into three layers that sit on top of the platform hardware layer.

Application Services: Services that are dependent on infrastructure provided by the other two layers, fulfill specific end use case requirements, and solve domain specific needs.

Application Support: Infrastructure software that does not fulfill any use case on its own, but helps to support and facilitate multiple application services.

Node Management and Software Backplane: General operation and management of the node and its communications with other nodes and systems. IB in the diagram refers to In Band management. This is generally how software interacts with the management subsystem.

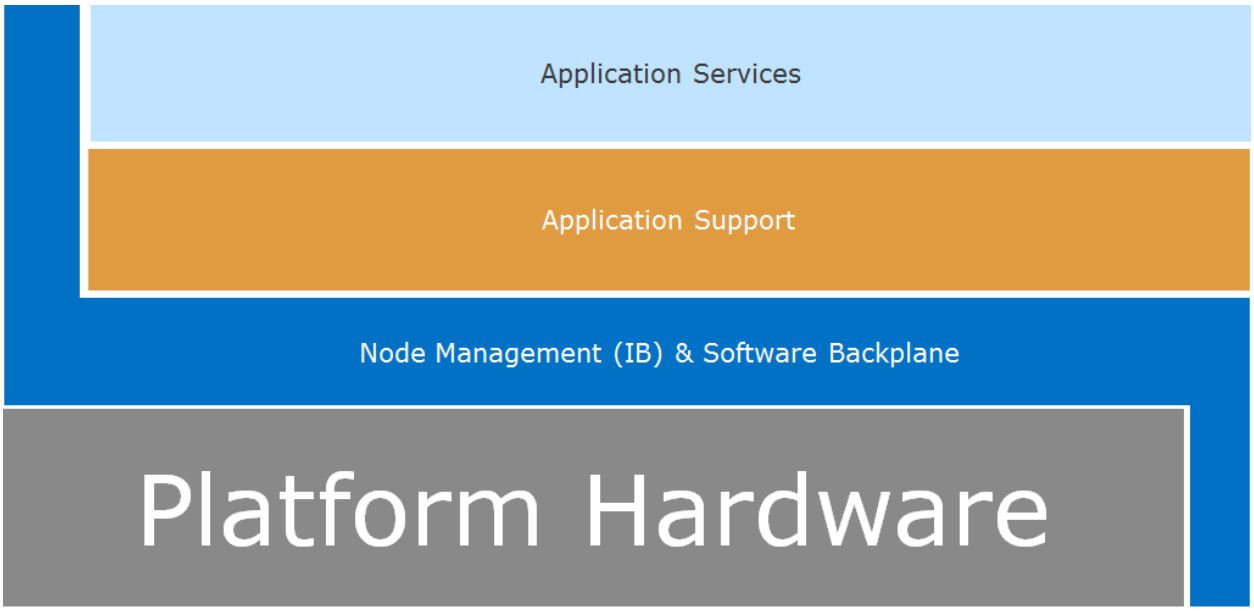


Figure 23 Software Architecture View

5.7.1.1 Software Backplane and Node Management

5.7.1.1.1 Software Backplane

The software backplane is required to run any software on the node and facilitate node-to-node communications (east-west as well as north-south). This includes:

- **OS:** May include unikernels that operate on top of a virtualization layer and extend all the way to application micro services.
- **Software drivers and firmware:** Interface with and enable hardware.
- **Communication services**: Enable communications and might help to define software-defined networks and protocol stacks
- **File system software**
- **Software virtualization**: To provide hardware-based virtualization support for running software and application micro services.
- **Containerization**: To provide OS based isolation support for running software and application micro services.

Software containers provide a good mechanism for fine-grained separation of applications and micro services running on the software backplane. A software container, unlike a VM, often does not require or contain a separate OS. A container uses resource isolation of the CPU, memory, block I/O, network, etc., and separate namespaces to isolate the application's view of the operating system.

Within a container, applications can be configured, resources isolated, and services restricted. Multiple containers share the same kernel, but each container can be constrained to only use a defined amount of resources, such as CPU, memory, or I/O.

Containers facilitate highly distributed systems by allowing multiple applications to run on a single physical compute node, across multiple VMs, and across multiple physical compute nodes. This is a critical capability for the elastic compute environment needed for fog computing.

Security of the software backplane is the component by which trust is established in the software layers above the backplane. The backplane should provide a means of verification of the application support and services layers by use of the chain of trust established by the node and platform. This verification may extend to remote attestation to an external system.

In order for containers, their application services, and micro services to be securely initialized and provide their intended services, the software backplane should enable the root of trust to be extended. Because the software backplane manages the creation and retirement of trusted execution environments and/or containers at the upper layers of the software stack, it is critical that the upper layers are able to verify attesting entities. The software backplane should define a policy by which it enforces responses based off data it receives from other devices. These policies may be as specific as only allowing communication with devices that use specific firewall rules, application mix, and installed patches. Or it may be more open policies, such as requiring a specific OS release.

The Software Backplane layer orchestrates thing-to-fog, fog-to-fog and fog-to-cloud communications. This layer must and should provide data confidentiality and integrity services to protect communications in all directions and must and should enforce data-origin and peer-to-peer authentication on connectionless and connection-oriented communications respectively. Nonrepudiation of data origin and destination may also be provided in this layer to north, fog-to-fog in order to support remote attestation required to support trusted computing. These services should derive their security and trustworthiness from the security credentials issued

by the Security Management maintained in the Hardware Root-of-Trust. All communications (wired or wireless) across the Software Backplane must use a base list of standard cryptographic functions to provide confidentiality, integrity, authentication and non-repudiation services[Section 10.1.1]. For performance reasons, these cryptographic functions are often performed by the Crypto Accelerators installed in the fog node. The following describes some additional facets of the Software Backplane:

- **Service discovery:** Is essential when multiple fog deployments need to come together they are able to create multiple trust boundaries in an ad-hoc manner for the purpose of cooperative information exchange and computation. Establishment of trust between fog deployments for transient collaboration requires well-founded trust framework and trust provider service graph.
- **Node discovery:** Applies to intra-fog discovery in a clustered deployment. When a new fog system is added to the cluster, it will broadcast its presence and joins the cluster. From then on, this node is available for sharing the workload.
- **State management:** Support for both stateful and stateless computational models. Stateful computational model can externalize the state or store the state within the fog cluster through a resilient replica model. Consensus algorithms may be utilized to ensure that multiple replicas of the same entity are eventually synchronized for preventing data loss. Externalized state requires the help of session/state micro-services that run on top of well-known database and storage technologies.
- **Publication and subscription management:** Application layers running on top of fabric runtime will need infrastructure support for publication of events, notification of state changes, and broadcast of messages. The publication and subscription management mechanism will support temporal as well as standing subscriptions and pluggable notification endpoints. The runtime stays abstract; it is the applications layer that composes payload pushes to the destination endpoints through the runtime layer.

5.7.1.1.2 Node Management (In Band)

When describing management, it is often acceptable to overload the term "node" with system. The fog In Band management layer is responsible for keeping the hardware and software of the fog node or system configured to the desired state, as well as keeping it running at specified levels for availability, resilience, and performance. The composition of the management layer will vary, in order to support fog nodes designed with different capabilities. For embedded and standalone deployment models, node management may be handled from a remote location. The hardware

platform management subsystem of the fog nodes cooperates with software on the main processor to perform this node management. The following are the list of capabilities needed by each fog node:

- **Configuration management:** Operating system and application support configuration is managed through a software agent that maintains the desired state of the OS and the application runtime. Agents are an optional aspect of node management as the deployment cost.
- **Operational management:** Operational telemetry of the fog nodes will be captured, stored, and presented for systems management personnel and automated systems responsible for monitoring the infrastructure. The information includes network operational events and alarms generated by the network, OS, and applications. The monitoring systems will manage the operational workflows for acting on critical alarms. The remediation of those alarms can be automated or manual depending upon the alert.
- **Security management:** Security management includes key management, crypto suite management, identity management, and security policy management.
- **Capacity management:** Monitor the capacity and page in additional compute, networking, and storage resources as demanded by the workload.
- **Availability management:** Critical infrastructure requires automatic healing in the event of the malfunction or crash of the software or hardware. The workload will be relocated to a different hardware node if hardware fails. In the event of a software failure, the VM or container may be recycled. Enough reserve system capacity should be kept in a ready state to meet any SLAs for the given scenario.

5.7.1.2 Application Support

Application support includes a broad spectrum of software used by and often shared by multiple applications (micro services). Application support is neither domain nor application specific, but may be dependent on the underlying layers (including virtualization, hardware, etc.). As shown in the figure below, depending on the deployment type or application, support software may be provided in multiple forms (e.g., the use of multiple application storage databases may be required in some deployments on some nodes).

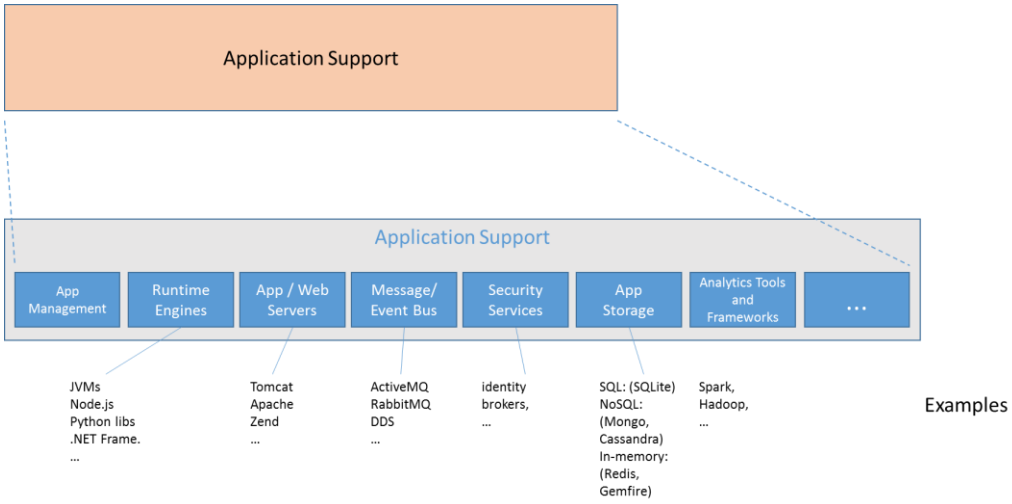


Figure 24 Application Support

While not required, elements within the application support layer may often be containerized deployed in some form of virtualization as provided by the software backplane. As examples, a message broker or NoSQL database which are used by several applications (within the applications services layer – see the figure below) within the Fog node, may itself be independently containerized and offer its supporting capability via the bounds of that containerization.

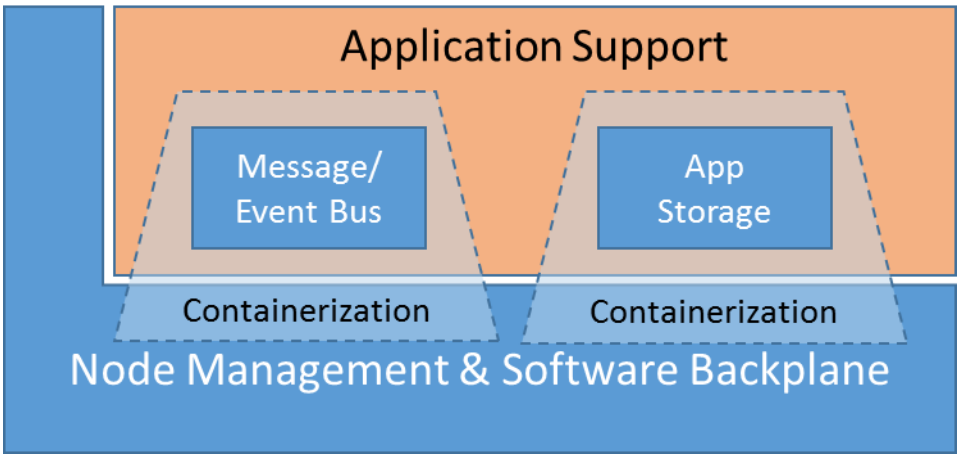


Figure 25 Containerization for Application Support

The containerization or virtualization of any support layer capability provides looser coupling, additional security, and can even allow the backplane to scale the supporting layer more quickly/easily.

Application support includes the following:

- **Application management**: The provisioning, verification, updating, and general management of the application support software, as well as application micro services. The same mechanisms also manage configuration images for accelerators like FPGAs and GPUs. Application management covers:

    - **Application provisioning:** The network runtime will host provisioning agents for receiving and acting on application provisioning requests from the management system. The management software will then use the application manifest to pick the right image from a versioned image store. The provisioning agents will help with the rollback requests.
    - **Image (application bits) management:** The fabric runtime layer will need the support of an image management framework hosted as a part of the fabric control infrastructure. The image management interface may include image verification for trust, malware, versioning, and dependency management.
    - **Image verification:** Fabric runtime, by default, only runs an image that is verified to be safe. This requires the fabric controller to support code authentication schemes like PKI.
    - **Version management:** The image management framework will allow the deployment of a particular version of the image. The create, read, update, and delete (CRUD) functions for storing the image are essential for populating the image store and provisioning a particular version in case of rollbacks.
    - **Transparent updates:** Depending on the stateful or stateless behavior of the hosted application endpoints, the fabric runtime will spread multiple instances of the same application into update zones. This enables the runtime to visit and update one update zone at a time, ensuring that the entire application is not taken down.

- **Runtime engines**: VMs, containers, platform runtimes, program language libraries, and executables provide the execution environments for applications and micro services. Examples include: Java Virtual Machines, Node.js (JavaScript runtime), NET Framework, Python Standard Library and runtime executables.
- **Application servers**: Application or web server hosting micro services or other node supporting infrastructure or applications. Examples include: Wildfly/JBoss, Tomcat, and Zend Server.
- **Messages and events (buses or brokers)**: Support for message and event-based applications and micro service communications (often categorized under message oriented middleware, message

broker, message bus, etc.). Examples include: message DDS, ActiveMQ, and ZeroMQ.

- **Security services**: Support for application security including encryption services, identity brokers, etc. Security services within the application support layer may also include deep packet inspection, intrusion detection and prevention systems, as well as system and network event monitoring, content filtering, and parental control. Details of these are included in the security appendices.
- **Application data management/storage/persistence**: Application data transformation capability and storage to include durable persistent as well as in-memory caches. Persistent storage may include both SQL and NoSQL databases, but other forms of durable storage should be considered, such as in-memory databases and caches (to address latency and performance concerns). Examples include: SQLite (SQL), Cassandra & Mongo (NoSQL), and Redis or Gemfire (in-memory databases). Considerations within this layer include:

    - Encoding/decoding/transcoding:

        - **Decoding:** From binary to JSON for application layer processing. For example, protocol translation from OPC UA/DDS/LONWORKS binary to JSON.
        - **Encoding:** From application layer payload to binary for transmission. For example, from JSON into OPC UA binary
        - **Transcoding:** Translation of a data structure from one format to another format within the same layer, possibly using a gateway
        - **Encryption/decryption:** Data in motion as well as data at rest

    - Information persistence/cache

        - Enable storage of structured and unstructured data
        - Durable/non-durable (e.g., in-memory, local disk, external storage service)
        - Pluggable into the encoding/decoding/transcoding pipe and filter process
        - Support streaming and batching models
        - Support multi-tenancy (e.g., information isolation between scopes/profiles)
        - Store and forward capability

        - Special support for digital media content (e.g., digital rights management, add insertion)

- **Analytics tools and frameworks**: Examples include Spark, Hadoop, and other MapReduce types of technologies.

5.7.1.3 Application Services Layer

Fog node applications will vary greatly based on deployment, use case, and resource availability. Fog computing applications are composed of a loosely coupled collection of micro services. As shown in Figure 31, these services can be separated into layers based on their roles.

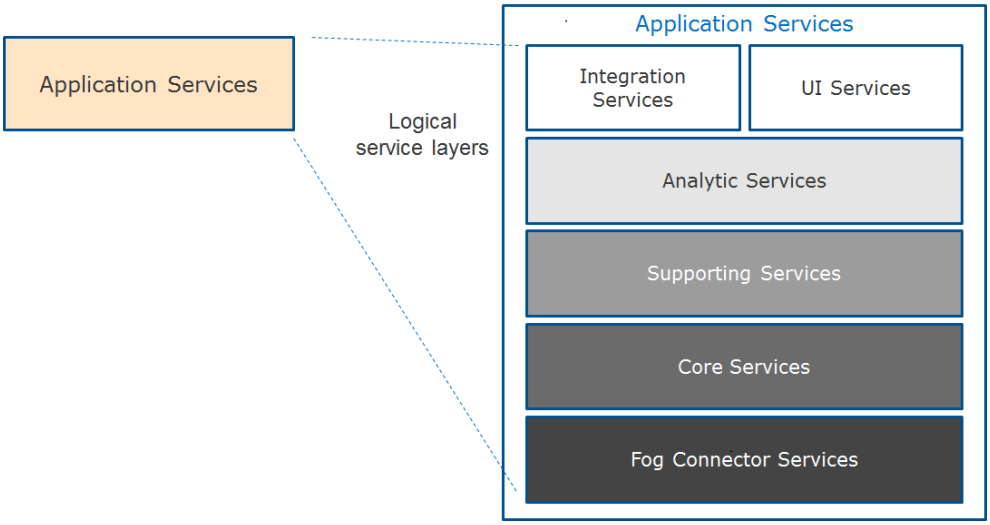


Figure 26 Application Services Layer

Applications, as with application supporting services, may run inside of containerized/virtualized environments as offered by the software backplane. These applications may take advantage of support layer services (like a database or message broker) that are containerized or may run on or be deployed to containerized supporting services (like a runtime engine).

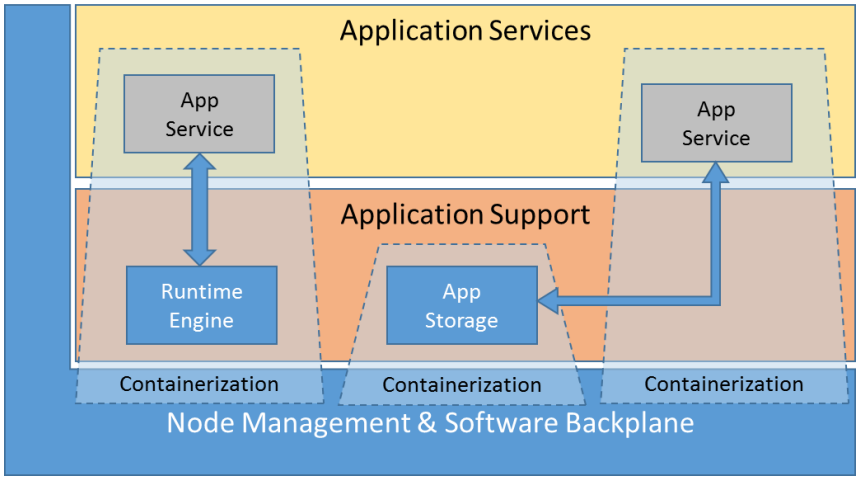


Figure 27 Containerization for Application Support

Fog connector services are used at the "south side" of the architectural application layer – that is the interfaces in the direction of the IoT things. This includes legacy transports like Modbus et al. These micro services contain a set of APIs for enabling higher-layer fog services to communicate with devices, sensors, actuators, and other platforms using the edge protocol of choice. Fog connectors operate on top of the protocol abstraction layer to translate the data produced and communicated by the physical things into a common data structures/formats and then send that into the core services.

Core services separate the edge from the enterprise. Core services collect data from the edge and make it available to other services and systems above, such as the cloud. Core services often route enterprise and other system requests to the appropriate edge resource, sometimes translating the requests for edge devices. This translation is a function of the fog connectors, which contain a set of APIs for receiving and translating commands from higher-level fog computing services (or the cloud) to edge devices for actuation.

Supporting services encompass a wide range of micro services that provide normal software application duties, such as logging, scheduling, service registration, and data clean up.

Analytics services may include both reactive as well as predictive capability. Closer to the edge, fog nodes will likely have services that are more reactive in nature. Fog nodes with more processing power and capability (usually farther from the edge) will have more predictive capability based on machine learning and other cognitive services.

Reactive analytics look at the raw incoming data and monitors for change = outside of expected norms. This includes but is not limited to:

- Critical event processing
- Simple anomaly detection
- Data out-of-bounds alert triggering
- Sensor fusion with stream processing
- Supervisory / distributed control
- State machine and state machine engine
- Expression language (versus SDK) for moving around the state machine
- SDK and/or API set to provide/update rules to the event processing engines.

Predictive analytics are defined as forecasting analytics. This includes but is not limited to:

- Fog node machine learning can support fog-only or hybrid models where some aspects (say training) are performed in the Cloud, while more processing intensive and high scale aspects (say an inference engine) execute in the fog. Models may be generated in the cloud and communicated down to the fog node agents for use.
- Connectivity to machine learning or other predictive-styled analytics engines that may be running off-node.
- The development of actionable intelligence gain from a collection of sensor/devices that typically could not be derived from a single sensor or device (referred to as sensor fusion). Data can be fused from similar sensors measuring in parallel (like an array of security cameras), or different sensors monitoring different (but interrelated) parameters. Sensor fusion may also include information securely collected from outside the node, such as information from the Internet.
- SDKs and tools focused on how to connect predictive or machine learning algorithms to the stream of data, model creation, etc.

Integration services allow outside fog nodes to register for data of interest collected or generated by the fog note and dictate where, when, how, and in what format the data should be delivered. For example, a client may request that all temperature-related data be sent via REST to a prescribed address, every hour, encrypted, and in JSON format. Integration services then provide the means to deliver the data using a pipe/filter mechanism as specified at the time of client registration.

User interface services are micro services dedicated to the display of:

- Data collected and managed at the fog node
- Status and operation of services operating at the fog node
- Results of analytics processing at the fog node
- System management and fog node operations

6 Adherence to OpenFog Reference Architecture

The OpenFog Consortium intends to partner with standards development organizations and provide detailed requirements to facilitate a deeper levels of interoperability. This will take time, as establishing new standards are a long and ongoing process. Prior to finalization of these detailed standards, the Consortium is laying the groundwork for component level interoperability and eventually certification. The OpenFog Testbed working group, in conjunction with the Technical Committee, will provide the details for OpenFog adherence to the architectural principles and various views that will be shown through our testbed initiatives. They will be used for the various technologies that can support the OpenFog RA and overall solutions to a given scenario. A technology that is used to facilitate part of a fog solution is termed OpenFog Technology Ready.

Figure 28 OpenFog Technology Ready

An OpenFog architectural E2E solution to a scenario is what we call OpenFog Ready.

Figure 29 OpenFog Ready

Prior to standardization the Consortium will leverage its technical working groups to score various implementations that claim to OpenFog or fog computing implementations. This scoring and the subsequent appeals

process will be visible to members and will be published to highlight and recognize progressive fog computing implementations and those that do not follow the OpenFog architectural principles and OpenFog reference architecture.

7 An End-to-End Deployment Use Case

Earlier, we described how the OpenFog RA would be used by developers, designers, and architects to create solutions for vertical market use cases. In the chapters that followed, the OpenFog RA provides the details of a generic fog platform. In this section, we address an end-to-end use case.

7.1 Airport Visual Security

Visual security (surveillance) for airports provides an excellent end-to-end scenario for fog computing. It illustrates the complex, data-intensive demands required for real-time information collection, sharing, analysis, and action.

First, let's look at the passenger's journey:

- Leaves from home and drives to the airport
- Parks in the long-term parking garage
- Takes bags to airport security checkpoint
- Bags are scanned and checked in
- Checks in through security and proceeds to boarding gate
- Upon arrival, retrieves bags
- Proceeds to rental car agency; leaves airport

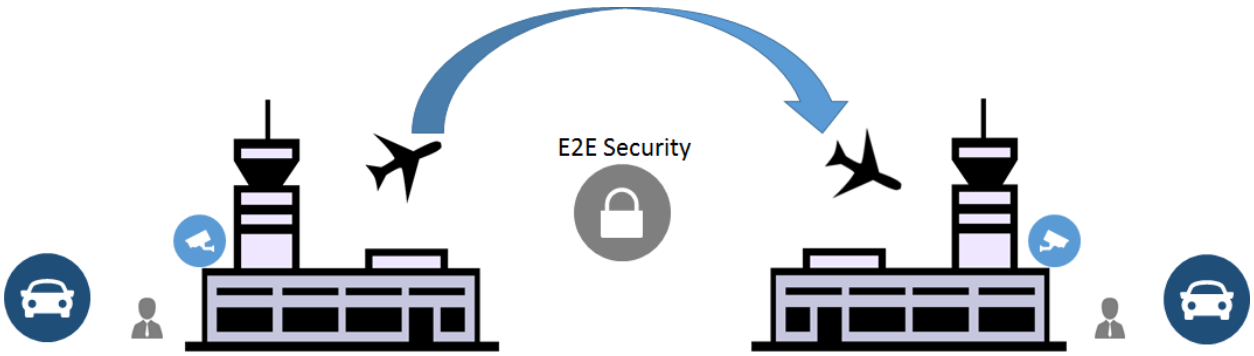


Figure 30 Airport Scenario

This travel scenario is without incident. But when we introduce any type or number of threats into this scenario, the visual security requirements become infinitely more complicated. For example:

- The vehicle entering the airport is stolen
- The passenger's name is on a no-fly list
- The passenger leaves his luggage someplace in the airport

- The passenger's luggage doesn't arrive with the flight
- The luggage is scanned and loaded on the plane, but it is not picked up by the correct passenger.
- An imposter steals (or switches) a boarding pass with another passenger and gets on someone else's flight.
- The passenger takes someone else's luggage at the arrival terminal

Catching these possible threats requires an extensive network of surveillance cameras across the both airports (several thousand cameras at each airport). An IP H.264 or H.265 camera produces 12Mbps at 30fps (frames per second) or approximately 1TB/day per camera that must be transmitted to security personnel. Or, more likely, the video streams will be forwarded to local machines for scanning and analysis.

In addition, law enforcement will need data originating from multiple systems about the passenger's trip, from the point of origination to arrival. Finally, all of this video and data must be integrated with a real-time threat assessment and remediation system.

7.1.1 Cloud and Edge Approaches

In an edge-to-cloud design, every camera (edge device) in the airport transmits directly to the cloud for processing, as well as the other relevant data collected from the passenger's travel records.

	Advantages	Disadvantages
Edge-to-Cloud Approach	• Store shared data in a common location • Historical analytics for threat prevention planning	• Latency (inability to process images and alert authorities with millisecond turnaround) • High cost of data transfer • Reliance on always available cloud
Edge-only Approach	• Low latency	• Limitations in sharing data and information across systems within the airport. • Limitations with sharing data between airports in near real time

In both of these approaches there are advantages and disadvantages. However, in both cases we believe that the disadvantages in Edge-to-Cloud and Edge-Only drive the requirement for Fog computing.

7.1.2 Fog Computing Approach

The power of fog computing is that we can insert computation where it is needed to address the given problem. Before going into the solution let's look at the OpenFog pillars and how they relate to our specific use case.

Figure 31 Key pillars of the OpenFog Architecture

The OpenFog pillars covered in great detail in chapter 4 are present throughout the airport visual security E2E architecture. There are some that require special attention because of this specific deployment scenario.

- **Security**: The airport visual security scenario is a physically distributed Fog deployment. Thus, physical possession is in scope for our security analysis. Transportation and storage of data must also be secure as much of the data which may contain personally identifiable information.

- **Scalability**: is essential for OpenFog implementations to adapt with the business needs as it relates to system cost and performance. When you add a new airport terminal, gate, or additional sensors and equipment the solution must scale and not require a completely new deployment.
- **Open**: Openness is essential for the success of a ubiquitous fog computing ecosystem for IoT platforms and applications. Proprietary or single vendor solutions can result in limited supplier diversity, which can have a negative impact on system cost, quality and innovation.
- **RAS**: The various aspects of the solution must be reliable, available, and serviceable which includes orchestration of existing or new resources. As new object recognition models are trained for visual analytics, these inference engine models should be updated on near edge devices without impacting availability of the solution.
- **Programmability**: Visual analytics is utilized to facilitate this scenario and hence programming at the hardware is utilized. In our implementation, we may utilized accelerators such as FPGAs to perform inference on images as they are seen in the scenario.

The OpenFog reference architecture includes several layers, perspectives or cross cutting concerns, and views to enable an OpenFog implementation for the airport visual security scenario.

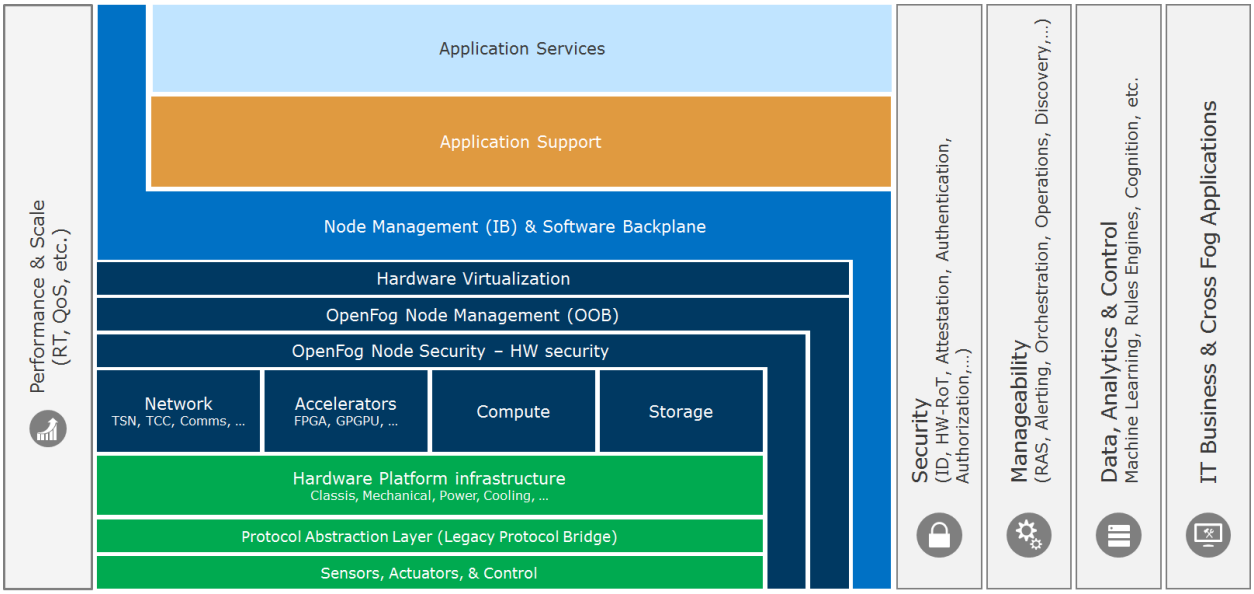


Figure 32 OpenFog Architecture Description

In this visual security scenario, we focus on the platforms, data analytics, performance, and higher-level software infrastructure.

- **Performance**: indicates that we need to include time to adequately process the information and provide useful latency sensitive information to enact an action.
- **Platforms**: indicates we need to have the correct hardware platforms with the appropriate accelerators and communications infrastructure needed to have each Fog node communicate with each other to address the scenario.
- **Data Analytics**: means that as we process this information at the individual Fog nodes, those nodes that are nearest the edge enable higher level intelligence at each level in the hierarchy.
- **Software Infrastructure**: indicates we can transition the data, intelligence, and environments across various Fog node deployments and enable higher-level computation.

The core aspects of a fog node can also be viewed as compute, storage, network, accelerators and control.

7.1.2.1 Functions of a Fog Node for Visual Security

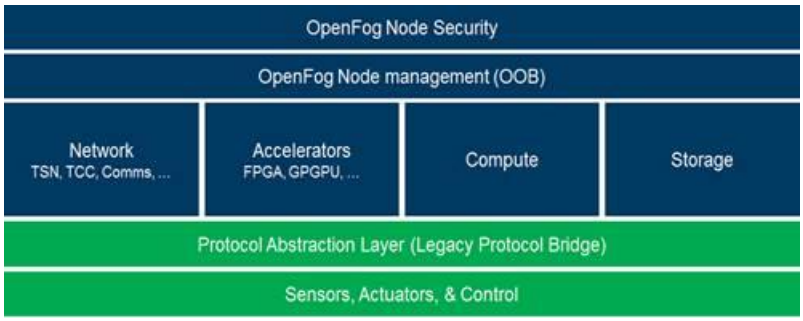


Figure 33 Node view for Visual Security

Sensors, Actuators and Control

In the end-to-end fog computing implementation, we start with the sensors at the edge of the network. Previously, we discussed the number of cameras in an airport surveillance installation, and the volume of data generated by these many thousands of cameras. In addition to cameras, there is a wealth of other edge devices collecting data that can be useful in preventing a threat. These include:

Physical security sensors	• Gates, doors, motion detectors,

	etc.
Safety sensors	• Fire, smoke, heat, and bomb sensors. Note: these independent systems can either be on the Ethernet or stationed behind a fog node (see next section on legacy protocol bridges).
Audio sensors and basic audio analytics	• Audio sensors can capture voices and sounds that may be important in detecting and assessing threats; this audio information can be forwarded for basic audio analytics and alerts sent to higher levels in the hierarchy for processing.
RFID sensors	• RFID sensors can be used to gather information from passengers, such as passport information

These sensors are connected to the fog node via a multitude of interfaces. These should be standard open interfaces like PCIe, USB, Ethernet, etc. They should also support open APIs but their implementation may be proprietary. This is important to ensure openness. After they are connected, they can be used by higher level software. For example, the RFID reader would be hooked up to the Fog node by Ethernet or USB. The fog node can them utilize the data and provide that to higher level entities.

Protocol Abstraction Layer (Legacy Protocol Bridge): A fog computing solution doesn't require a "greenfield" deployment; it assumes that there will be mixture of analog cameras combined with digital cameras. One method to convert the analog feed is to utilize accelerators (low cost FPGAs provide a good implementation option). The fog node needs to be able to take a multitude of sensors and perform sensor fusion, which, in turns, requires that the fog node have a variety of physical interfaces for converting legacy analog to digital. These interfaces include coax, USB, RS232, audio, PCIe, etc., and system interfaces like SPI. If a system can connect directly to an IP network, the RA utilizes the software backplane to provide sensor fusion for higher-level software to utilize. The reality is that in many implementations of open interfaces and protocols, they have slight variations and hence require a protocol abstraction layer to effectively utilize them.

Network: Each IP-connected camera (and many other sensors) is connected to a fog node using Ethernet. In our scenario cameras represent the largest population of sensors. We believe that a singular fog node can aggregate four to eight camera streams with a 1Gb (camera to node) and 10/1Gb (fog-to-fog) interconnect. Fog nodes require a minimum of 16 Ethernet ports to support camera-to-node, node-to-node, and node-to-cloud communications. First-level fog nodes may also support multiple non-IP protocols for aggregating both IP and non-IP traffic (e.g., BLE, Z-Wave, and badge reader information). Time-sensitive networking (TNS)/ deterministic communication is supported by the network and the fog node to prioritize alerts across a busy or noisy network.

Accelerators: The visual security scenario offers many places to include accelerators. As described above, FPGAs may be used to convert the analog input into a digital format. In this application, the FPGA may be connected to the node using a traditional PCIe interface. Accelerators also play a role when performing visual recognition (face, object, etc.). AlexNet, GoogleNet, TensorFlow, and other neural networks can be used to accelerate a matching an image with a threat. This is usually called inference scoring based upon a given trained model. The recommended interface to the node in this example is the same for the analog-to-digital format (most likely PCIe). This enables future upgrade of FPGAs as cost and technology changes over time.

Compute: As a fog node is installed, higher-level software needs to understand the capabilities by which it processes the data generated by all connected sensors and cameras. Compression can also be added to the cameras to reduce the general-purpose computation requirements at the camera. The next level in the hierarchy would require higher processing to perform video analytics using general purpose compute resources or accelerators. In our scenario, it is recommended that most of the compute be performed in the fog node and not necessarily on the camera.

Storage: For a given camera feed, a fog computing design should capture 24 hours of rolling data. This requires ~1TB of localized storage for each camera feed. The interface to these devices is either SATA or PCIe; the medium is either flash-based or spinning disk. In one topology, you can see how a single fog node can act as the storage for approximately eight camera feeds. This is a traditional Network Video Recorder (NVR) function.

In some implementations, storage will be resident on the camera. However, we believe it is more optimal for the airport visual security scenario to configure the storage on the hierarchy of fog nodes. This will mean reduced camera cost and combining NVR functions with the video analytics

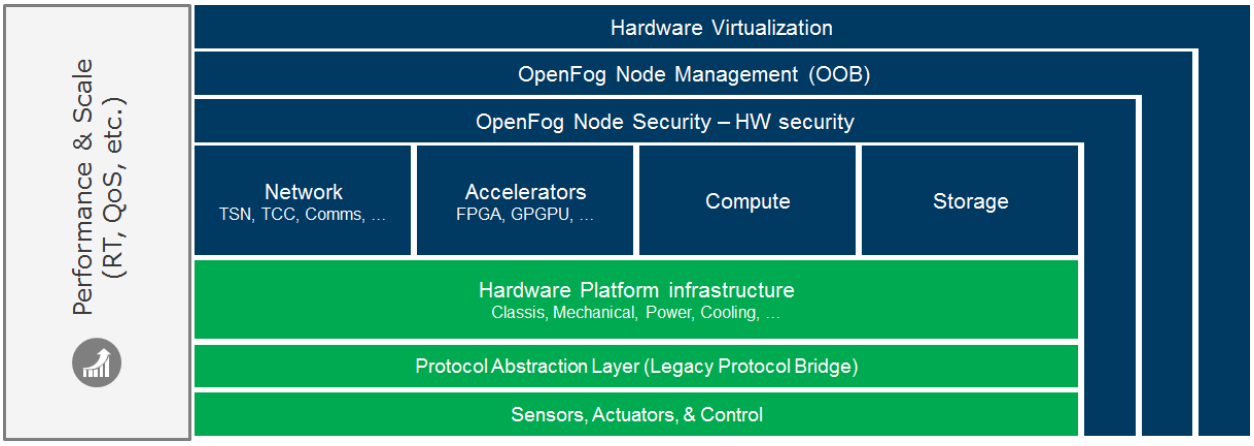
capabilities in the same fog node. However, this will increase the compute capability requirements at the next level in the hierarchy.

Management (OOB): If the fog node has updateable firmware and software, the design can specify remote updates. This is part of the serviceability requirements of the RAS pillar. The RA also requires that the node implement a mechanism by which a management entity can ascertain the health of the overall node (minimally, a healthy or failed state). Traditionally, these notifications are accessible via I2C or SMBus interfaces and routed to a management controller. This function can also be made available to an in-band interface to the operating system. This allows higher-level software to interface with the management subsystem. The RA recommends that any component that can be field repaired or removed should have a health indication which can be accessed via this interface.

Security: Nodes must have an immutable hardware-based root of trust for verification. This is to ensure that agents capable of communicating in the airport are operating from good firmware. The RA also recommends a hardware-based root of trust for measurement to provide attestation for the software and firmware on the node. This can be supported by a TPM or firmware TPM providing it does not impact the overall hardware-based root of trust.

7.1.2.2 System View for Visual Security

This section describes how the components of the fog system view interact to support this end-to-end use case.



System View Hardware Virtualization: The hardware of the hierarchy of fog nodes used in the visual security example should support virtualization techniques. Applications should also avoid being dependent upon which level

of the hierarchy their various processes are hosted on. If one fog node in the network hierarchy is overloaded or down, the system level orchestration will move its responsibilities to adjacent nodes in the same level or adjacent levels. Processing, accelerators, storage and networking functions should all be virtualized to maximize the efficiency and flexibility of the fog system. Virtualization may also incorporate aspects of containerization, depending upon the software layers that run on the hardware.

System View Management: The sophistication of the management system must be balanced with the simplicity of usage. The RA addresses a network-level, end-to-end view of installation, configuration, operations, monitoring, troubleshooting, repair, growth and decommissioning of all elements of the system. System-level node management must be as autonomous as possible to minimize complexity of nodes that a given fog infrastructure can handle. By doing this we can increase the overall scalability of the solution.

Additional management includes a higher-level hardware manager that is responsible for all system management. The management system monitors the health of all cameras, storage and other hardware assets. This management interfaces with the software backplane to satisfy the overall systems management.

System View Security: At the system level, all the fog nodes in the hierarchy must cooperate to ensure the network remains secure. This includes:

- Nodes in higher levels of the hierarchy should monitor the functions of the lower level nodes to ensure no ongoing or emergent security threats exist.
- Peer-level nodes on the same hierarchy level should monitor their neighbors to detect security threats.
- All node-to-thing and node-to-node communications links should be encrypted and monitored for suspicious traffic.
- Physical security, including tampering, must also be monitored.
- If a system is tampered with, the management subsystem must be notified so that appropriate security measures can be executed.
- Communication pathways between nodes must be encrypted.
- Visual security scenarios also include capture of personally identifiable information (PII), so any data captures must be encrypted while at rest. This includes end-to-end network security.

System View Network: The network links that interconnect the nodes in the visual security architecture transport multiple types of traffic.

- At the lowest level is the interconnection of adjacent near-edge fog nodes. These are typically direct point-to-point Ethernet links, operating at 1 or 10Gbs. They link together the composite picture of the fused sensor readings across all nodes. For example, stitching together all camera images on a given fog node with a subset of the images from selected cameras on adjacent fog nodes.
- Another type of interconnect is between each near-edge fog node and the second level fog node that serves it. These links usually carry higher level messages and analytics data that has been distilled by the near-edge fog nodes.
- Another interconnect type is between the second and third level fog nodes. The third level fog nodes are master controllers for the entire visual security implementation. Redundant links are provided for load balancing and fault tolerance. These links can carry significant traffic bandwidth, and run for kilometer lengths (usually over fiber). They are typically 10 or 100GE IP connections.
- Finally, there a set of links between the third-level fog nodes and the cloud backbone.

System View Accelerators: The airport visual security scenario studied here may include hundreds and thousands of cameras. Each of which will produce images that must be carefully analyzed in real-time to detect a large number of conditions. The accuracy of the analytics for visual security is vital, to avoid swamping the first responders with false alarms, or worse, missing some threat scenario that should generate an alarm. Computation accelerators make these intense computations fast and energy efficient. Some combination of accelerators (such as FPGAs) at the near-edge fog nodes, and even larger farms of accelerators at the second and third level of the fog hierarchy should be used in the design. Accelerators at the third level may be used in image training (facial recognition), while accelerators in first level fog nodes would be used for inference on a trained model.

System View Compute: General-purpose computation is vital at all levels of the system view. These compute resources will typically be multi-core processors, sometimes configured as multi-socket servers at higher levels in the hierarchy. Significant memory (from tens to hundreds of GB) are required on each fog node to avoid performance bottlenecks in video-intensive applications. System-level compute resources are required for things like:

- Executing tasks (such as control algorithms and user interfaces) that can’t be more efficiently run on accelerators or require high single thread performance
- Managing the networking capabilities of the fog system

- Incorporating license plate recognition scenarios.
- The following are usually offloaded to accelerators in our example, but are kept here for reference:
    - Facial recognition.
    - People counting
    - Threat detection

System View Storage: Storage will be required at all levels of the fog hierarchy. During the normal operation of the system, some visual data will be transmitted from the lower levels of the hierarchy to upper levels and eventually to the cloud.

The storage on the near edge fog nodes should be sized to keep approximately 24 hours of full-resolution video online locally. For a typical fog node with about eight 4K resolution cameras, assuming a compressed data rate of 10Mb/s per camera, this requires just under one terabyte of storage. Note: data from the non-camera sensors on the fog nodes does not contribute significantly to the storage requirement. This storage may be implemented in many different ways (there are NVR solutions that address this capture requirement). However, storage is finite and the RA balances the need to retain older data while addressing near term concerns. In these cases, we believe it is acceptable to have a mechanism to reclaim older storage for new purposes providing the user explicitly opts in for this behavior.

Before old data is overwritten on each near edge fog node, the recommendation is to convert it to a lower resolution (e.g., 720P or 480P) and forwarding the data to the second level fog nodes. The second level fog nodes accept these down-sampled streams from all the near-edge fog nodes they support, and store it for 30 days (in this scenario). This may require approximately 20TB of storage (assuming down-sampling of all video streams to 1Mb/s), requiring the second level fog nodes to have significantly larger storage arrays, perhaps using rotating disks and RAID array techniques. The third-level fog nodes are basically big servers, and will probably require big data style storage strategies. The benefits of keeping this older data in the fog is that we can reduce cloud transmission costs. If we do decide to transmit the down-sampled data, the cost of transport is less than if we had to send up the 4K or 1080P image. As described earlier, "data at rest" on the storage device should be encrypted to protect against physical theft and compromise of PII (Personally Identifiable Information).

System View Hardware Platform Infrastructure: The hardware platform infrastructure has a common set of infrastructure components—including chassis, backplane, power, cooling operating system and management—into

which hardware and software modules are plugged to customize the function of specific fog nodes. The same infrastructure elements can then be used to produce lightly populated nodes without significant computational, networking or storage requirements. The same elements could also be used in fully-equipped fog nodes (perhaps higher in the hierarchy supporting greater application demands).

System View Protocol Abstraction Layer: Protocol abstraction permits the fog network to operate independently of the specific protocols used in sensors and actuators or in the cloud. Adaptation logic (both hardware and software) converts native protocols used outside the fog network to internal standard protocols, storage formats and data abstraction models within the fog hierarchy. This abstraction is one of the reasons that fog computing can easily support the diversity required for this end-to-end airport visual security scenario.

System View Sensors, Actuators and Control: As described in the section on sensors, actuators and control above, there is a very rich set of sensors and actuators involved in the airport visual security scenario. Using sensor fusion techniques, the fog hierarchy will assemble the inputs from many sensors into a cohesive view of a threat situation at the airport. Equally important, fog computing provides the low latency required for swift response. Latency is also described under Performance.

System View Performance: Low latency is a key attribute of fog computing, as delayed detection and analysis of a threat is unacceptable. To achieve this low latency, the analytics algorithms have significant performance requirements, related to factors like accuracy, object database size, CPU utilization, energy efficiency, etc.

System View Scale: The airport depicted in our example is of moderate size, with 24 gates. A fog network will scale to support changing airport requirements. For example:

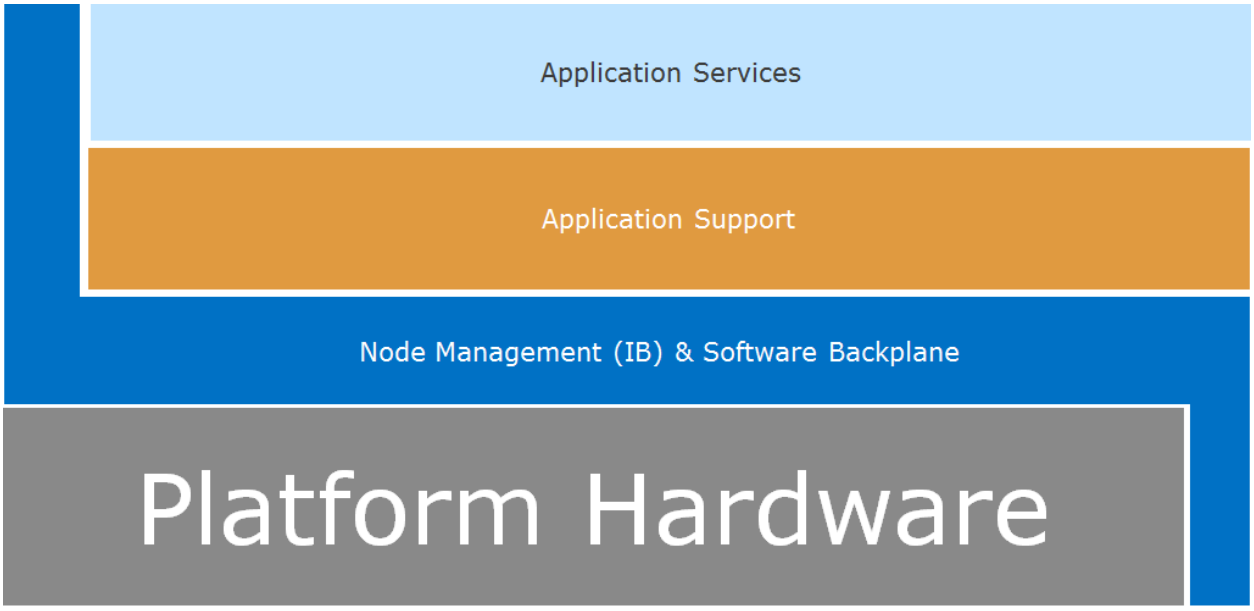
- The same fog network design can be scaled for small airports to hub airports.
- New algorithms will be introduced continually, requiring additional compute, accelerator, network and storage capabilities; the fog architecture is designed so that most modules can be upgraded without requiring the complete replacement of the infrastructure.
- Using the Fog as a Service (FaaS) model, the fog hierarchy will scale to support many different tenants with widely different needs, all on a single hierarchical network owned by a single landlord (e.g., airport authority).

7.1.2.3 Software Infrastructure View for Visual Security

As shown in figure below, the software of a fog node can be separated into three layers. The functions that facilitate the general operation and management of the node and its communications with other nodes/systems are found in the backplane and node management layers.

Infrastructure software that does not fulfill any use case on its own, but helps to support and facilitate multiple application services is found in the application support layer.

Services that are dependent on infrastructure provided by the other two layers fulfill specific use case requirements—such as this visual security scenario—and solve domain specific needs are provided in the application services layer.



7.1.3 Application to Airport Visual Security

Now that we have analyzed different aspects of the architecture that should be applied to our solution we will further investigate the requirements and assumptions.

The following picture provides a simple view of an airport terminal for our scenario. There are entrances, parking structures, security stations, etc. We

will refer to the diagram for various usages like License Plate Recognition as vehicles enter the airport property.

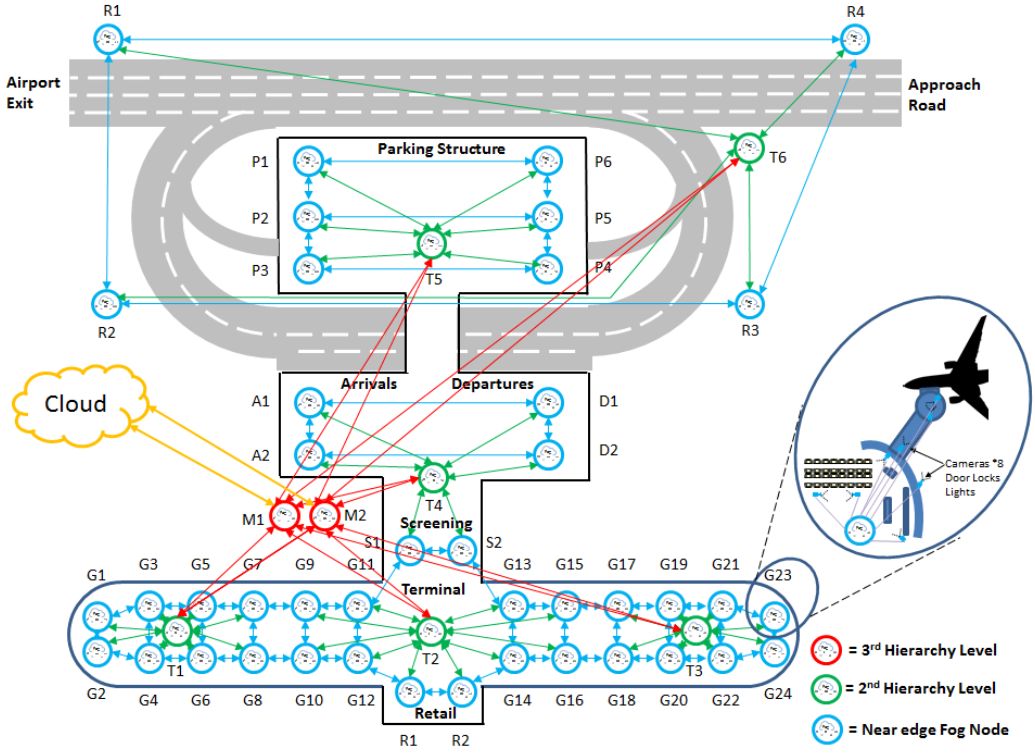


Figure 34 OpenFog Approach to Visual Security Scenario

We will also deploy multiple fog nodes in each airport and at different levels in the hierarchy. There may also be a fog node that is responsible for the entire airport and ensuring that interoperability across systems to achieve the visual security mandate is in upheld. This is also important so that airports can share normalized information. Additionally, each fog node may be connected to another level in the hierarchy. These fog nodes work in concert to satisfy the requirements of the scenario.

We have several areas to address which we include but are not limited to:

1. License Plate Recognition as vehicles enter the airport property.
2. Passenger Arrival/Departure
    a. Parking structures where passengers may exit vehicles and walk into the airport facility.
    b. Arrivals is also a location where passengers may walk into the airport facility.
3. Passenger Security screening where the passengers are required to provide identification and boarding passes.
4. Terminals where screened passengers may walk to their gate, shop, and eventually leave the airport.

When the passengers leave the airport, their information should be made available to the airport where the plane is landing. This is represented in the following figure with the fog node at a higher level in the hierarchy.

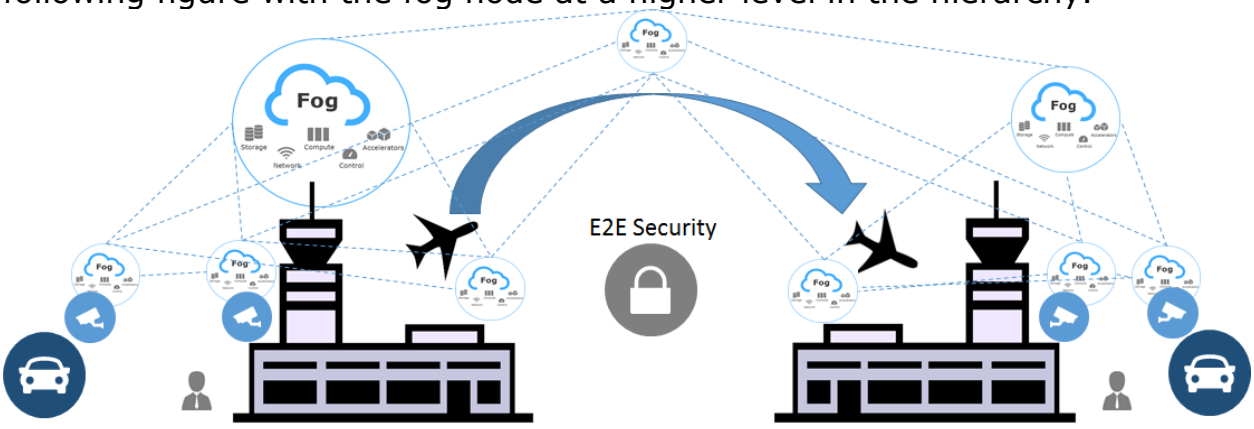


Figure 35 OpenFog realized for Visual Security

The following is a description of the various physical fog nodes that would be deployed in our scenario.

- Fog nodes for License Plate Recognition (LPR):
  - Fog nodes that are surrounding the airport property. In our example, these are cameras and security devices. These nodes will be thin in nature and report in adjacent fog nodes. We estimate that we can service 4 video streams with a singular fog node.
- Fog nodes that are stationed around the parking structure and arrival station:
  - These nodes will be comprised of video cameras, and as in the LPR case and those cameras are connected to fog nodes for visual analytics.
- Fog nodes that are in the arrivals and departure area (just prior to security screening). These are the same as the parking structure.
- Fog nodes that support the screening process
  - These fog nodes are connected to both passive RFID readers and other sensors as well as cameras.
- Fog nodes that are in the terminal
  - These nodes are connected to cameras and additional sensors.
- Fog nodes that watch ingress and egress of passengers to planes.
  - These nodes are connected to cameras and additional sensors.
- Hierarchical fog nodes that support and monitor a grouping of fog nodes.

- These fog nodes work on pre-processed data and perform higher level functions that support the overall mission of safety and security of the airport.

A complete representation of the realized solution will include the interconnectedness, security, and software that runs on all the nodes.

7.1.3.1 Machine Vision for Visual Security

Fog computing dictates that we process the image at the appropriate level in the hierarchy versus sending it to the cloud for analysis. A good mechanism to address machine vision requirements for LPR, passenger tracking, people counting and other usages in this specific scenario is the use of a Convolutional Neural Network (CNN).

When dealing with CNNs it is common to discuss both training systems and classification systems. Training systems are used to build a CNN network topology and compute weights against which the image classifications are validated. This iteration process of adjusting weights or fine tuning weights are continued till we achieve satisfactory level of accuracy in classifying. Once we get to this satisfactory level of accuracy (Say a minimum of 98% accuracy), we push both the both the topology and the corresponding weights to Target or classification system (aka inference or scoring). AlexNet which is a well-known CNN for Object recognition and uses a 1.2 million training image set to build a 1000 different classifications. This gives an estimation of images required for given number of classifications. Higher number of images is always better for training.

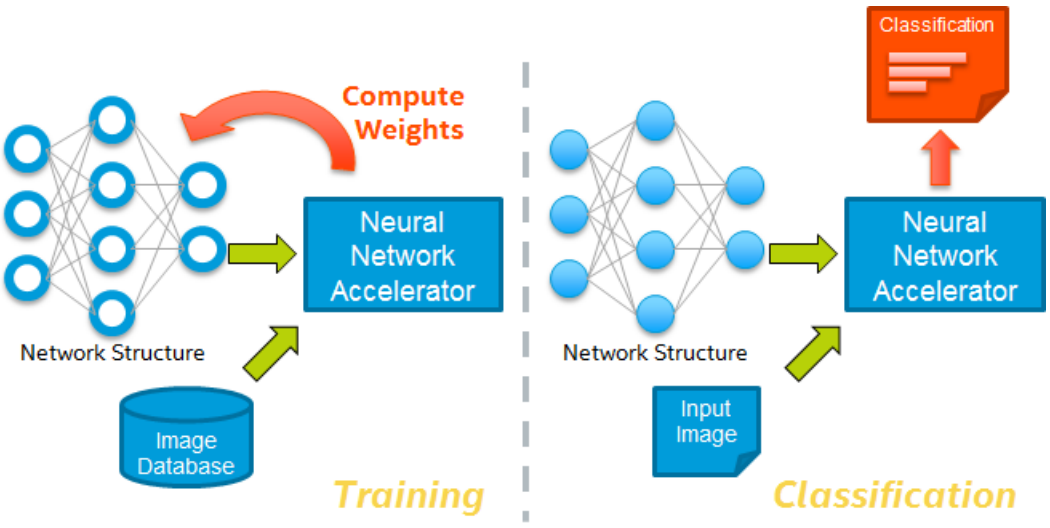


Figure 36 Training and Classification system for Machine Vision

In the LPR case as vehicles are going through the various lanes there will be a LPR camera and illumination so that we can capture the license plate image and send that information to the fog node. There are multiple options for this part. The camera can just capture and send the picture of the vehicle and license plate doing more work at the edge of the network or the entire video stream can be compressed and sent to the fog node for additional analytics.

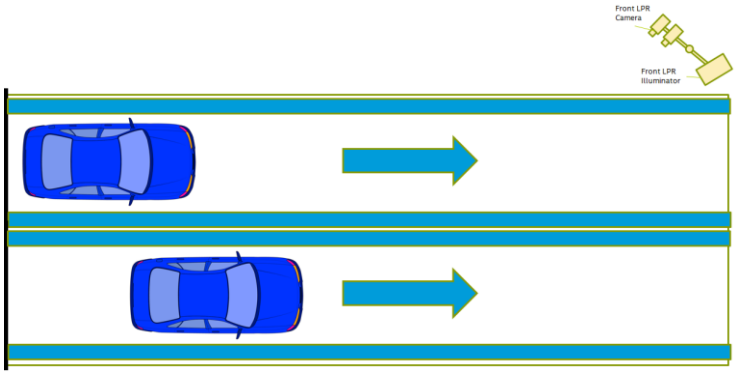


Figure 37 Airport License Plate Capture

The LPR fog nodes will be trained on license plate images and once the license plate is capture the fog node will perform 1) Localization 2) Character segmentation and 3) Optical Character Recognition (OCR) to determine the license plate state.

Passenger recognition follows a similar flow of using a training system to train a model based upon an image database. The models for passenger and vehicles should also be updated frequently based upon new unclassified images so that the system will learn over time and increase overall accuracy.

The following diagram show several assumptions in how we will get various information shared between different agencies and private entities. While this is a simplification of a real scenario it provides the base requirement of an optimal system where we can securely share information.

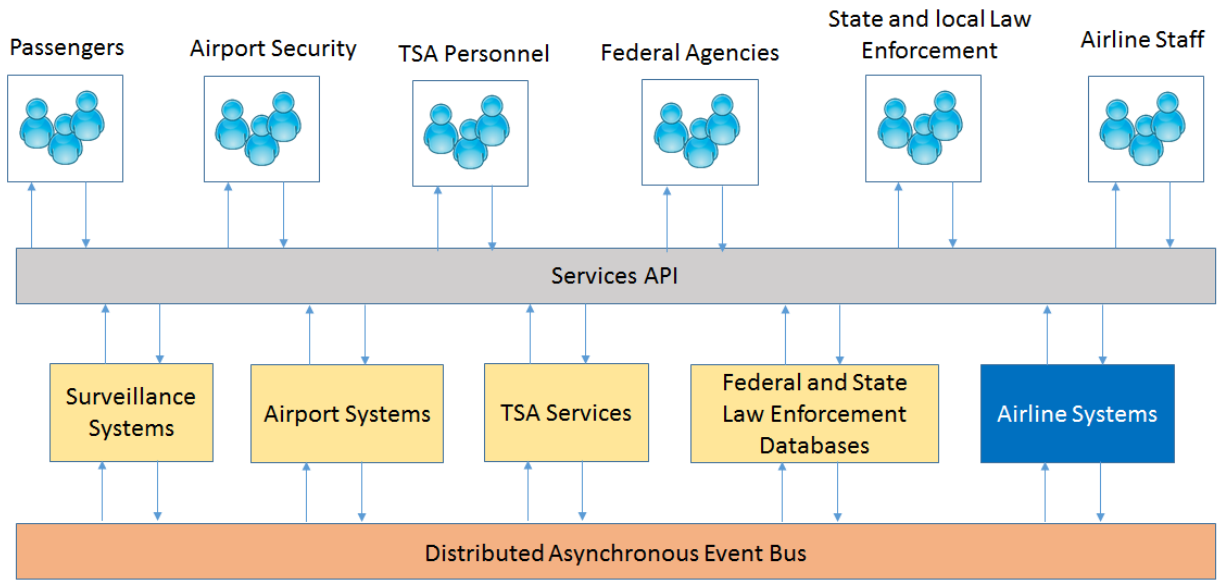


Figure 38 Integrated Air Travel Infrastructure - Assumptions

7.1.3.2 Airport Passenger

To help simplify the flow we will start with a standard case of a passenger utilizing the airport for transportation we showed earlier.

The passenger leaves from home and drives to the airport.

There is a fog node associated with each group of cameras that monitors vehicles as they come onto the airport. This node is responsible for capturing the license plate image, performing video analytics, and capturing the face of the driver as they enter the airport property. Fog nodes should also have the ability interface directly with RFID readers and other data acquisition devices and sensors to provide local, high performance identification of people and objects in the vehicle.

- Data security and privacy concerns are addressed throughout the network. Camera firmware will be protected by a hardware-based root of trust for verification and, optionally, measurement. This ensures that processed images are coming from hardware that is operating in a known configuration and has not been tampered with.
- Privacy concerns for visual images stored on the camera requires that all saved data is protected from inspection even with physical possession. This protection involves strong cryptography on all data links and storage repositories used to hold images of people. It should be encrypted when at rest and in transport.

- If a stolen or suspicious car is detected as it enters airport property authorities should be notified to address the situation.
- Images that cannot be accurately detected should be saved and used for retraining.

Passenger parks in the long-term parking garage

At the garage gate entrance where the passenger collects his parking garage ticket, the cameras catch the first images and persistent data objects in this end-to-end scenario. The following figure illustrates the software applications and edge devices that are collecting useful information for this threat scenario.

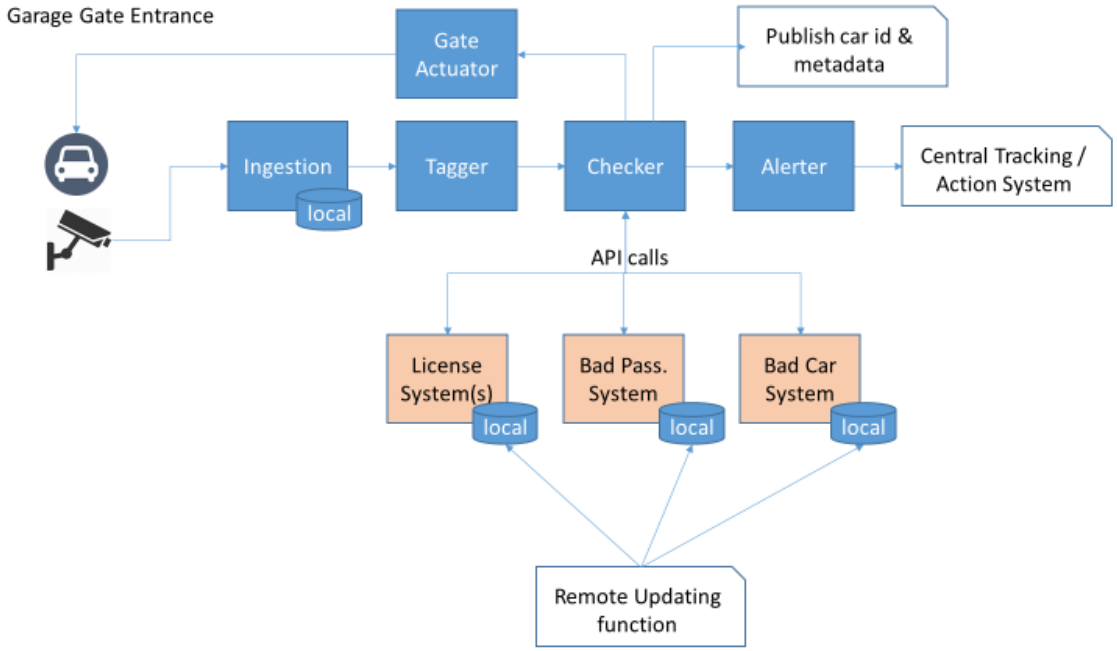


Figure 39 Garage Gate Entrance

Fog node hardware and software performs video analytics on the camera feeds. This involves an image processing pipeline that can be split across multiple layers of fog nodes in the hierarchy, or multiple peer nodes to balance load. The fog network processes the images, and recognizes certain people or objects. For example, if a license plate is detected that falls on a "bad vehicle" list, the system can discover this with less than a second latency, and use that information to control traffic gates (without significant delay). Similarly, the fog network can perform facial recognition, and detect people on the no-fly list. Fog is valuable in this case, because the latency is low (permitting automatic actuation of turnstiles, etc.), and the local processing and storage of the fog nodes can protect passenger privacy.

Objects (like abandoned luggage, for example) can be detected and reacted to quickly and securely by the fog network.

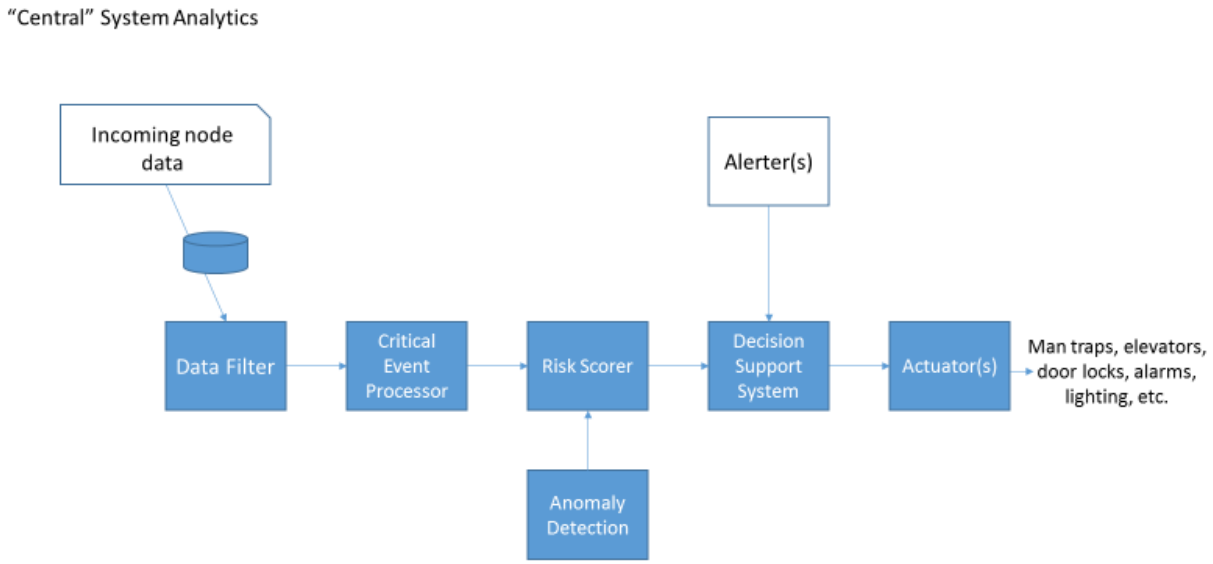


Figure 40 Central System Analytics

The image processing pipeline is an example of fog's distributed analytics capabilities, processing the raw images from the cameras in several steps, shown in the figure above, and described in detail below.

- Data Filter: Cleans and filters incoming data from a variety of fog nodes capturing raw sensor data.
- Anomaly Detection (machine learning): Detects different types of anomaly and asynchronously generates models to provide to the risk scoring system, for example the detection of a person on the no-fly list, or abandoned luggage
- Critical Event Processor: Rules engine that monitors incoming data flags events of importance (based upon airport policies stored in the fog network) and passes them to the risk scoring system.
- Risk Scoring System: Generates a risk score for vehicles, passengers, baggage or other entities known to the system. Passes high-risk targets to decision support systems.
- Decision Support System: Receives high-risk targets from the risk scoring system; takes actions automatically or raises alerts.
- Operating actuators (for example, running parking lot gates, turnstiles, alarms, etc.).

- Training System for edge algorithms as described in the earlier section on machine vision.

Entering Arrivals on the way to the Screening Area

At the airport entrance the security system will have to have multiple cameras to cover all of the vehicles pulling up and people getting out of cars, grabbing bags and entering the airport. This can also be after the passenger parks his car and proceeds to the screening area. A key attribute of this phase of the process is the tracking of the passenger, and everything he brought to the airport from his vehicle, through the airport departure area and to baggage check. Notice how the local fog nodes perform sensor fusion and data checking/correlation to efficiently implement this step.

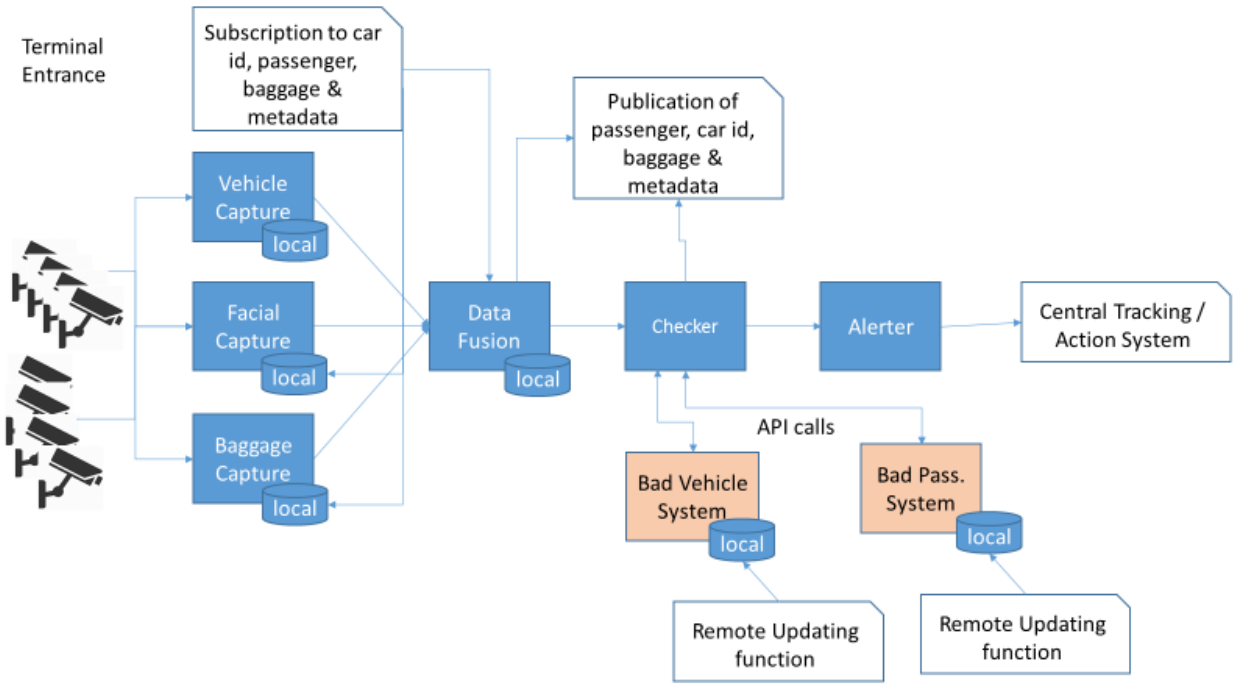


Figure 41 Terminal Entrance

The software components for the entrance have many overlaps with the parking garage software. However, there are many more instances of the capture components as there are many more cameras. Consequently, fog node processing is much higher here.

Also, since we are seeing new vehicles we will re-use the Bad Vehicle System. As the passenger proceeds from the garage to the entrance, the facial capture and baggage capture components will have received updates of his face and baggage data objects through their subscription through the software backplane and related data sharing software services. Multiple instances of each of these software components will be supported by the

software backplane. The agility of fog systems make this sort of re-use/multiuse possible.

Sophisticated analytics systems process all the information gathered by the fog hierarchy, normalize it, and send it to the next higher level of the hierarchy for further processing. These analytics algorithms may employ machine learning techniques to continuously adapt to ever-changing conditions and threat models. Each level of the hierarchy will combine the inputs of cameras and other sensors into a more holistic view of the airport, further digest the view into a more "concentrated" form, which is then sent to higher levels, where ultimately airport security policies can be applied, and any necessary enforcement actions (like denying passage of a vehicle or person through a barrier) will be carried out. This is how processed data becomes wisdom

After the visual analytics is performed at the node, it can cross check the information learned about the driver, car status, and flight status if applicable and package this information for more detailed processing at the next level. From there, the fog network can determine if the issue requires security personnel to be alerted.

Passenger takes bags to airport security checkpoint

Throughout the time when the passenger entered the airport property, a database should be created with all of the various information obtained. This includes LPR information, car, and images of the passenger and associated people. The fog network appends the previously processed images and database entry with new surveillance images of the luggage area, facial recognition, and other data related to this passenger, such as bag scans, updated ticket information, etc. Using this additional information, the fog network analytics can predict where the passenger should be going, associate any time he is recognized where his bag is not visible, and/or associate other risks with his behavior, appearance, etc. This correlation of multiple security camera images with the output of various other sensors and the database of information about the passenger is an example of fog's advanced sensor fusion capability.

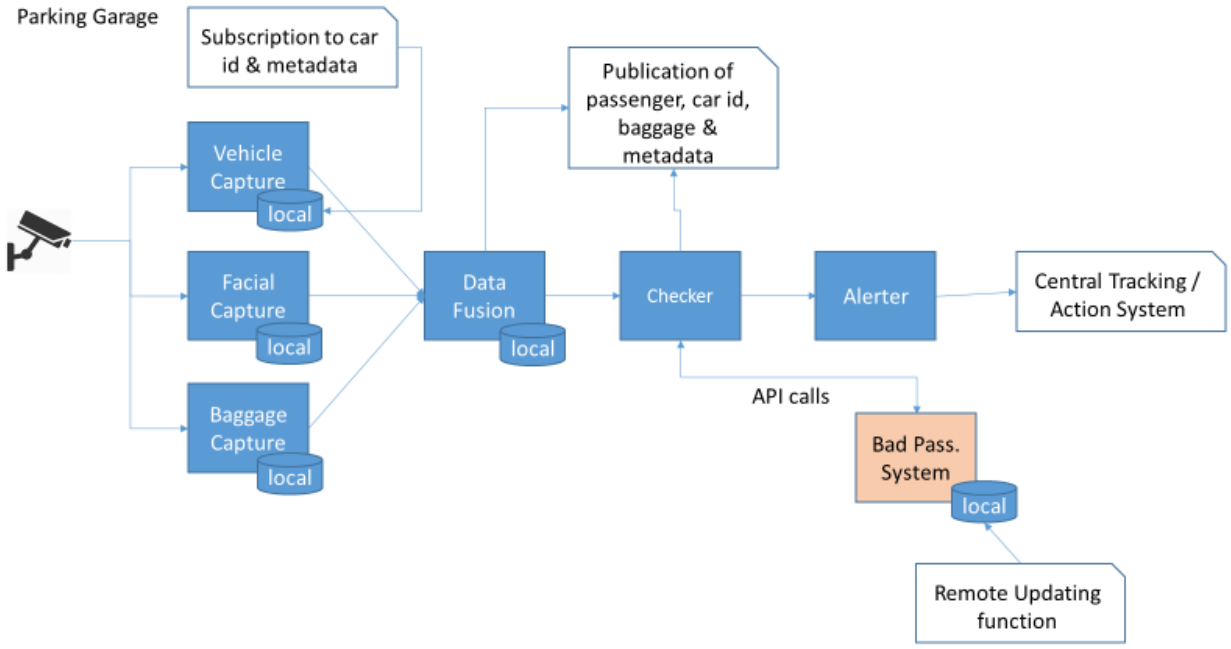


Figure 42 Baggage flow

- Using techniques similar to the operations described for the parking garage, the above figure shows some of the software components of the fog application that manages the baggage check-in process.
- Vehicle Capture: converts camera images to license info, make/model info, parking spot & pair this information with unique identifier for the vehicle that came through the garage gate. Provides API for other systems to request raw image based on id.
- Facial capture: converts camera images into unique person identifier. Provides API for other systems to request raw image based on the id.
- Baggage capture: converts camera images into unique baggage identifier. Provides API for other systems to request raw image based on the id.
- Data Fusion: Associates vehicle identifier to parking spot. Associates persons to vehicle identifier. Associates baggage identifier to person identifier.
- Checker: attempts to match facial capture data against bad passenger images
- Alerter: responsible for signaling possible issue detected to centralized tracking system
- Bad Passenger System: registry of persons under surveillance by authorities and/or people on no-fly lists, on arrest warrants, etc.

Data for this system is held locally and updated on a regular basis.

Bags are scanned and checked in

Cameras will pick up the passenger's new location. The baggage information will be updated and added to his database record entry. This data will be published to the various software components in other parts of the system for risk analysis.

- The figure shows the baggage screening process. Multiple local fog nodes support each security line, their number is scalable depending upon compute needs. A mix of sensors including cameras, millimeter wave machines, and bomb sensors (sniffers) outputs are combined with all the context from previous stages of the process to provide a very effective screening process.

Here are some of the Components used in this step.

- Bomb Sensor: Gathers data from bomb sniffer and passes any events to the Data Fusion system for correlation and eventually action
- Facial Capture: Converts camera images into unique person identifier. Provides API for other systems to request raw image based on the id.
- Mm Wave Screen: Gathers data from mm Wave Machines, scans images for problems, alerts airport screeners for additional screening, and passes any events to the Data Fusion system for correlation and eventually action.
- Behavior Monitor: Uses various camera feeds to monitor for bad behavior among people waiting in line. Any events are passed to the Data Fusion system for correlation with facial and passenger data.
- Baggage Capture: Converts camera images into unique baggage identifier. Provides the APIs for other systems to request raw images based on the identification.
- Data Fusion: Associates passengers in line with baggage, behavior alerts, MM Wave screen alerts, and proximity to bomb sniffer alerts.
- Checker: Attempts to match facial capture data against bad passenger images, and forwards alerts from the sensor systems.
- Alerter: Responsible for signaling possible issues to centralized tracking system.
- Bad Passenger System: Registry of persons under surveillance by authorities and/or people on no-fly lists, on arrest warrants,

etc. Data for this system is held locally and updated on a regular basis.

Check in through security and proceed to boarding gate

At this stage the data analytics should have enough information produce cognition from the various fog nodes – that is the raw data from the cameras and sensors has been converted by the fog network through the steps of becoming information, and is now knowledge of the fact that the passenger has maintained control of his luggage, and it has been analyzed by the security screening policies. The fog network will now be able to autonomously determine if there is a credible threat involving the passenger. This entire process will take milliseconds and no longer than a few seconds. Namely the gate operator must be notified to make a determination to allow the passenger to enter the plane, validate that their bags were in his possession at all times, and that they are allowed to fly.

The system needs to alert the pilot, or in future scenarios the autonomous plan to not take off. This will be solely based upon the entire suite of analysis performed in the fog. If the passenger is determined to pose a minimal threat, all barriers in his path will open with sub-second latency, and he can board his plane without delay. If he is determined to pose a threat, several levels of escalation are possible. The system could alert airport authorities (either in central security control, or by locating and informing the officer physically nearest to Bob). Barriers he may pass through like turnstiles or mantraps could hold him. Many other fully automated or semi-automatic responses are possible, depending upon the treat level detected by the fog, and airport policy.

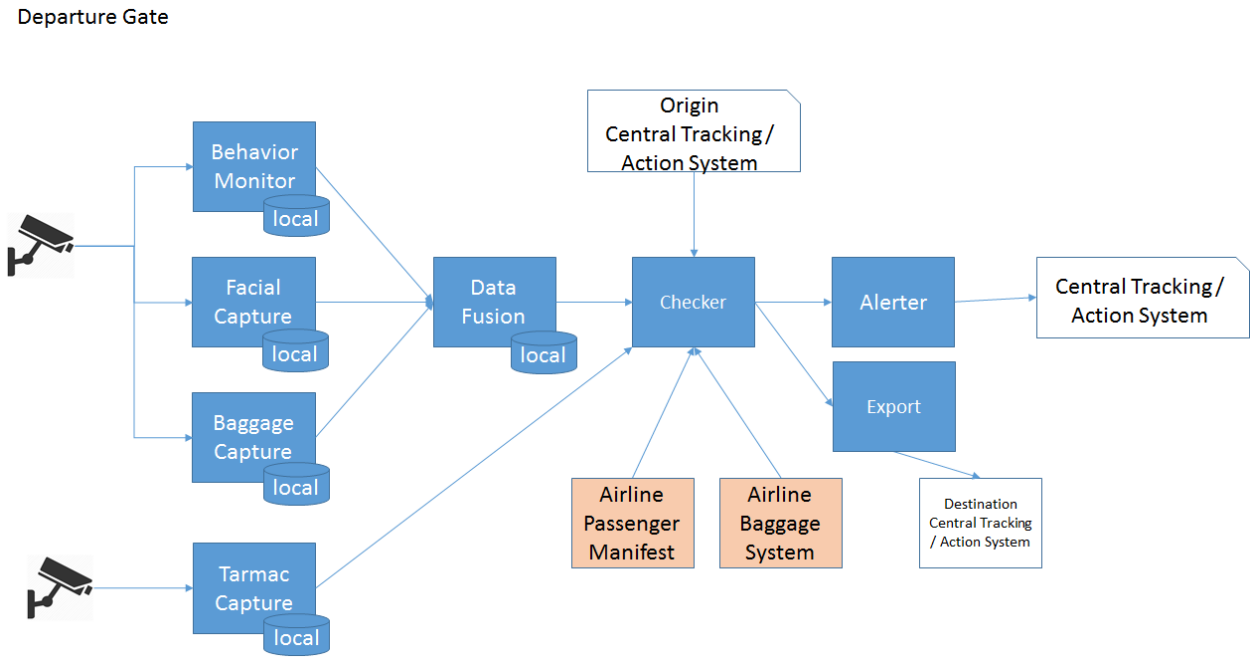


Figure 43 Departure Gate

- The above figure describes some of the steps the fog network will take as Bob makes his journey to the plane.

System Components

- Facial Capture: Converts camera images into a unique person identifier. Provides APIs for other systems to request raw image based on the identification.
- Behavior Monitor: Uses various camera feeds to monitor for bad behavior. Any actions that raise flags are passed to the Data Fusion for correlation with facial and passenger data.
- Baggage Capture: Converts camera images into unique baggage identifier. Provides APIs for other systems to request raw image based on the identification.
- Tarmac Capture: Uses various camera feeds to monitor the aircraft and tarmac for unusual behavior, security breaches, and potential aircraft damage.
- Data Fusion: Associates passengers in gate area (by facial recognition) with baggage and behavior alerts.
- Airline Passenger Manifest System: Provides data about passengers that are checked onboard the plane.
- Airline Passenger Baggage System: Provides data about passenger checked baggage.
- Checker: Final assembly and correlation of data from all available system sources.

- Passenger appears to match credentials provided and face consistently matches with images captured since entering the system.
- Passenger baggage is accounted for and consistently matches baggage captured since entering the system.
- Aircraft does not appear to have been tampered with.
- No warnings or issues have been detected by other systems since the passenger entered the airport space.

- Export: Responsible for sending all relevant passenger data to the destination Central Tracking / Action System
- Alerter: Responsible for signaling possible issue detected to centralized tracking system

Upon arrival, retrieve bags

Security cameras at the arrival airport have data about the passenger. As early as the arrival gate, the fog computing network can determine if the passenger arrived and has retrieved his baggage.

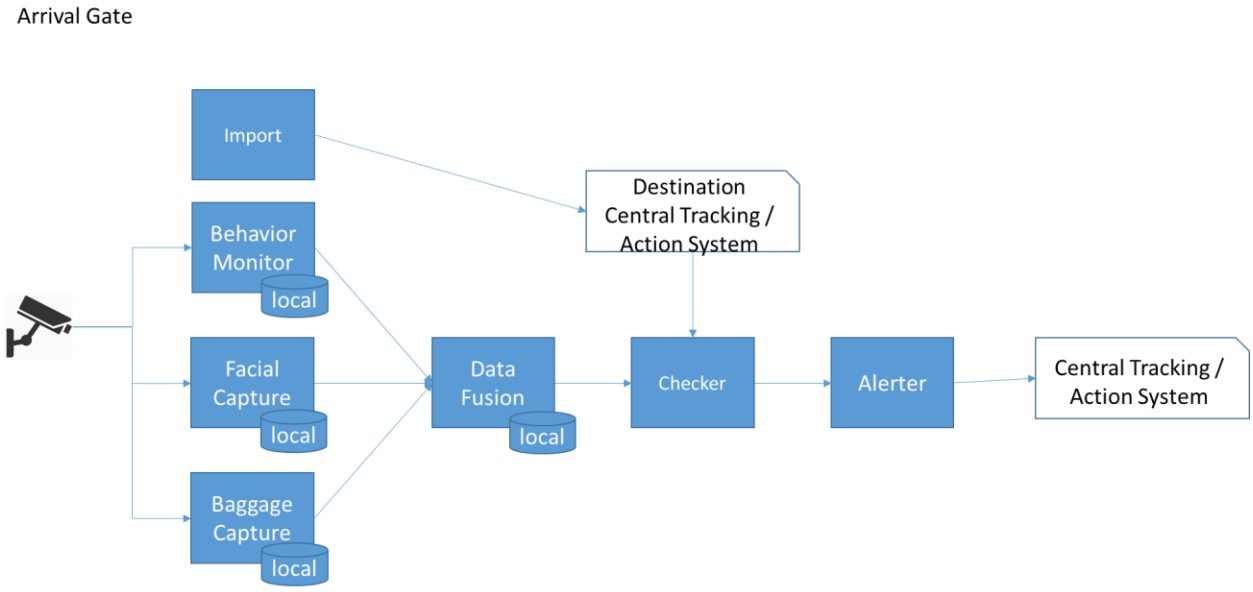


Figure 44 Arrival Gate

- The reverse of the arrival process is followed when the passenger reaches his destination, and many of the same fog-based processes will insure safety during this part of the journey, as shown in the above figure.

The process components include:

- Facial Capture: Converts camera images into unique person identifier. Provides API for other systems to request raw image based on the identification.
- Behavior Monitor: Uses various camera feeds to monitor for anomalous behavior. Any events are passed to the Data Fusion system for correlation with facial and passenger data.
- Baggage Capture: Converts camera images into unique baggage identifiers. Provides APIs for other systems to request raw image based on the identification.
- Data Fusion: Associates passengers in gate area (by facial recognition) with baggage and behavior alerts.
- Checker: Correlates data from facial capture and baggage capture final with incoming (imported) passenger data received from the origin airport. Ensures all passengers originally on the aircraft exit the aircraft and all expected baggage also exits the aircraft.
- Import: Responsible for receiving all relevant passenger data from the origin central tracking / action system into the destination central tracking / action system
- Alerter: Responsible for signaling possible issue detected to centralized tracking system

Note that throughout this process, many fog nodes within an airport, and fog networks at two different airports must maintain high performance, highly secure fog node-fog node communications. This is possible because of the highly secure fog infrastructure on all nodes, and the strong cryptography applied on all node-node traffic.

Proceed to rental car agency; leaves airport (if authorized)

The data collected required interoperability such that each node can operate upon and gain higher-level insights to protect. In many cases this data obtained throughout passenger's journey can be shared with local governmental agencies to also track if they can only be in the country for a certain amount of time, if he is on parole, etc. Assuming he is not a threat in his destination, they are cleared to rent a car, and the rental car company can have much higher levels of confidence that the passenger is who he says he is, and poses minimal danger, because this information is selectively shared between the arriving airport's fog systems and its car rental agencies.

- The above scenario about a passenger's journey serves to illustrate some of the key attributes of fog. Fog's distributed processing capabilities and hierarchy support the sophisticated analytics and sensor fusion algorithms that analyze their appearance and actions. Fog's low latency permits nearly instantaneous reaction to his actions (for example, opening a barrier in milliseconds, where cloud-based processing of the same sophistication could take seconds). The highly secure nature of the OpenFog implementation insures that the passenger's privacy is maintained and visibility up to higher levels in the system hierarchy are constrained. Fog's reliability insures the system will continue operating even if a fog node, inter-node link or the connection to the cloud goes down. Fog's bandwidth efficiency insures high bandwidth traffic like video traverses only the most capable links.

This detailed use case as applied to airport visual security scenario is intended to illustrate the key benefits of the OpenFog Reference Architecture. It is intended to be used as a reference for those exploring the application of fog computing to similar concepts and techniques to solve similar challenging problems.

8 Additional Opportunities

The OpenFog Consortium collaborates tightly between its academia/research and traditional industry members. This allows the Consortium to leverage the research and publishing focus of academics with the business requirements of industry. Among the additional areas of fog computing research are:

- Interactions between fog and cloud computing including dynamic and secure shifting and sharing of resources.
- Security refinements not covered by existing efforts of industry associations.
- Enhancements required for deepened management and orchestration of fog computing.
- Fog based training to support deep learning and machine learning without requiring cloud.
- Fully development the Fog as a Service (FaaS) model.
- Performance modeling and measurement to ensure that designers and architects are achieving the proper QoS for a given implementation scenario.
- Generating rigorous, enumerated requirements to help facilitate higher levels of interoperability.
- Government and societal impacts of a software-defined autonomous world of fog computing.
- Environmental impacts of a more optimized computing environment.
- Education, research and development for new engineers and scientists who will be instrumental in shaping the implementation of the fog architecture. This will include software development for fog computing.

9 Summary and Next Steps

The OpenFog Reference Architecture (OpenFog RA) is the baseline document in developing an open, interoperable architecture for fog computing. It is the first step in creating new industry standards to enable interoperability in IoT, 5G, Artificial Intelligence, Tactile Internet, Virtual Reality and other complex data and network intensive applications.

The OpenFog RA represents an industry commitment toward cooperative, open and inter-operative fog systems to accelerate advanced deployments in smart cities, smart energy, smart transportation, smart healthcare and smart manufacturing. Its eight pillars describe requirements to every part of the fog supply chain: component manufacturers, system vendors, software providers, application developers. The OpenFog Consortium believes that without this open architecture, there will be limited interoperability, reliability and security, resulting in slower adoption and limited functionality.

The OpenFog Reference Architecture is the first step in creating industry standards for fog computing. The OpenFog Consortium will establish detailed guidance, interface with standards organizations such as IEEE on recommended standards and specify APIs for key interfaces in the reference architecture over the next year. Our technical community is working on a suite of follow-on specifications, testbeds which prove the architecture, and new use cases to enable component-level interoperability. Eventually, this work will lead to certification of industry elements and systems, based on compliance to the OpenFog Reference Architecture.

For more information on the work of the OpenFog Consortium, please visit www.OpenFogConsortium.org.

10 Appendix – Deeper Security Analysis

This appendix currently contains an initial discussion of several security aspects in an OpenFog computing environment. This important discussion was placed here for a couple of reasons. First, security is perhaps the largest technical concern among critical IoT systems; hence, we wanted to discuss it in a special section of the document. Second, the discussion contains highly specialized technical details, which, if included in the main body of the document, may impact its readability. Thus, we decided to collect the materials that are most interested to the security professionals in one place.

Future versions of the Reference Architecture will also include appendices that describe other high-priority cross-cutting perspectives from the architectural description including performance, manageability, data analytics and control with similar levels of detail.

10.1 Security Aspects

Security is a critical concern for fog computing. We strongly believe there must be a common security baseline to ensure basic interoperability and protection. We are also aware that there exists a combination of regional and governmental requirements that fog computing must satisfy. The following sections describe our preliminary attempt to accommodate diversity in approaches while trying to establish a unified practice in the security realm of OpenFog Architecture.

10.1.1 Cryptographic Functions

Cryptography provides mechanisms to implement security services such as confidentiality, integrity, authentication and non-repudiation. Cryptographic functions can be implemented in a Platform Security Processor (PSP) to protect cryptographic keys and security policies, which then protect other objects[1]. Cryptographic functions can also be used to provide a secure execution environment for trusted software, and protect its memory,

[1] The cryptographic functions used by software executing on OpenFog platform may not necessarily be implemented in the Platform Security Processor (PSP); they can be implemented independently in software or hardware outside of the PSP.

storage, and communications.

The current version of the document describes the initial base list of required standard cryptographic algorithms that MUST be available on all OpenFog nodes. Requiring this minimum set of algorithms is intended to guarantee interoperability among OpenFog nodes. We understand that this initial list is limited in its ability to enable global interoperability. Going forward OpenFog is making it a priority to develop a more complete list that includes the set of standard algorithms of regional standards bodies from, for example, Europe, China, Japan, and the US.

There are three basic types of cryptographic functions:

- Symmetric (or Secret-Key) Ciphers for confidentiality protection;
- Cryptographic Hash Functions for integrity protection and authentication of communicating parties[2]
- Asymmetric (or Public-Key) Ciphers for generating secret keys, establishing long-term security credentials and providing non-repudiation services.

The NIST FIPS 140-2 specification [ref-a] defines the security requirements for cryptographic modules. This specification covers a list of approved cryptographic functions as well as a formal process for validating the implementation of these functions in conformance to the specification. The OpenFog Reference Architecture adopts a subset of FIPS 140-2 approved cryptographic functions as described below in order to guarantees a base level of interoperability among its components. The formal validation of the cryptographic module (i.e. FIPS 140-2 certification) is left as an option for each vendor. Due to the growing importance of FIPS 140-2, vendors are encouraged to subject their products to FIPS 140-2 certification.

The OpenFog cryptographic module MUST support the following FIPS approved cryptographic functions at a minimum:

- Symmetric Key Ciphers[3]
  - AES (with at least 128-bit keys)
  - Triple-DES
- Asymmetric Key Ciphers

[2] Message Authentication Codes (MAC) can be used for authentication, but not for non-repudiation.
[3] The Escrowed Encryption Standard (ESS) was withdrawn on December 31, 2015.

    - $\mathbb{Z}_p, \mathbb{Z}_n^*$ Based: DH, RSA, DSA
    - Elliptic Curve Based: ECDH, ECDSA, ECQV[4]
- Cryptographic Hash Functions[5]
    - SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, SHA-512/256
- Random Number Generators
    - See Annex C:
      Approved Random Number Generators for FIPS PUB 140-2, Security Requirements for Cryptographic Modules
- Message Authentication Codes
    - CCM, GCM, GMAC[6], CMAC, HMAC[7]

As defined in Annex A of FIPS 140-2 [ref-b], due to the successful breaking of cryptographic algorithms in use or the availability of more powerful computing techniques, NIST provides guidance on updating the list of approved cryptographic functions. Please refer to NIST "Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths" [ref-c] for the latest guidance. Subsequent versions of this document will consider a transition approach that works for all regional cryptographic algorithms.

Note: Compliance is not security; some of the FIPS approved cryptographic functions may be considered weaker in strength and leave their implementation open to potential compromise. In the design of various OpenFog components, the cryptographic functions selected for their implementation SHOULD be appropriate for their use and in agreement with the findings from their stakeholder's threat assessment.

10.1.1.1 Crypto Accelerators

Cryptographic functions can be implemented either in software or in a hardware accelerator. While such a hardware accelerator provides an important security function for the system, the device itself must also be secure. If it is implemented in a virtual environment, it must be implemented as a hardware virtualized device so that it can be securely accessed by multiple independent processes and/or VMs while maintaining a

[4] ECQV is used for Implicit Certification.
[5] SHA-1 has been deprecated since 2010.
[6] See NIST 800-38D.
[7] See RFC 2104 and FIPS 198.

context specific to each virtual instance and providing access protection for each address space that "owns" the virtual interface.

Many platforms implement a hardware security processor of some kind (e.g., TPM, PSP, a Secure Trusted Execution Mode on each core). The PSP hardware is inherently trusted by the platform. Among other things, this security processor typically responds to requests to perform certain operations (e.g., store a record of the measured boot sequence for later enquiry, provide secure storage for asymmetric or symmetric keying material, provide key access protection, provide encryption and decryption of small amounts of secure material, provide an internal True Random Number Generator (TRNG), etc.). However, these security processors are generally not virtualized so that access from an OS running in a virtual machine is not possible – the security processor typically assumes a one-to-one relationship with the OS (e.g., the TPM provides only a single owner, a single Storage Root Key and password, a single Endorsement Key, and a single set of PCRs), so that the security processor functions are only available to the hypervisor in a virtual environment. That means that even though an OS executing in a virtual machine is capable of utilizing the secure storage and cryptographic functions of the security processor, as it would running in a bare-metal environment, it is not able to do so.

The solution to this lies in the implementation of a virtual security processor (e.g., vTPM, vPSP) that allow a large number (preferably limited only by available resources rather than design) of virtual machines to maintain a one-to-one relationship with the virtual Platform Security Processor (vPSP) that is allocated to it. A virtual implementation of a vPSP allows both the VMs and the PSP to be unaware of the virtualization and they can both look and act as they do in a non-virtualized environment. It is necessary to implement to full interface capabilities of the PSP in a vPSP as well as protected management functions to create and destroy vPSPs. The software to implement the vPSP must be integrated into both the hypervisor and the guest. In the guest a proxy driver is needed to field the API calls and pass them to the hypervisor component where they can be authenticated – i.e., can this VM access that object? The proxy in the guest must also return the result to the requester. The behavior of the vPSP in the guest must be the same as the behavior of the physical PSP when it is presented with a request. The requests can be redirected to the physical PSP or emulated, as required. The vPSP can protect platform configuration, measurement data and provide attestation with regard to the state of the platform configuration, data protection for the OS and its applications, and can help facilitate remote attestation. The implementation should ensure that implementations that work with the current PSP continue to work with the vPSP.

PSP operations should never occur in the performance path of any software implementation. Therefore, the additional latency imposed by a virtual implementation should be minimized.

10.1.1.2 True Random Number Generator (TRNG)

A TRNG extracts randomness (entropy) from a physical source of some type and then uses it to generate random numbers. The physical source is also referred to as an entropy source.

Almost all cryptographic protocols require the generation and use of secret values that must be unknown to attackers. For example, random number generators are required to generate public/private key pairs for asymmetric (public key) algorithms including RSA, DSA, and Diffie-Hellman. Keys for symmetric and hybrid cryptosystems are also generated randomly. RNGs are used to create challenges, NONCE (salts) values.

Because security protocols rely on the unpredictability of the keys they use, random number generators for cryptographic applications must meet stringent requirements. The most important property is that attackers, including those who know the RNG design, must not be able to make any useful predictions about the RNG outputs.

The major use for hardware random number generators is data encryption, for example to create random cryptographic keys to encrypt data. They are a more secure alternative to pseudorandom number generators (PRNGs) - software programs commonly used in computers to generate "random" numbers. PRNGs use a deterministic algorithm to produce numerical sequences. Although these pseudorandom sequences pass statistical pattern tests for randomness, by knowing the algorithm and the conditions used to initialize it, called the "seed", the output can be predicted. Because the sequence of numbers produced by a PRNG is predictable, data encrypted with pseudorandom numbers is potentially vulnerable to cryptanalysis. Hardware true random number generators (TRNGs) produce sequences of numbers that are not predictable, and therefore provide the greatest security when used to encrypt data.

Fog systems should implement a TRNG as opposed to a PRNG solution. The functionality may be implemented as an ISA extension or via a separate accelerator device for example. If it is a device, it should be hardware virtualized, to allow for secure access from multiple VMs and/or containers in order to preserve secure access.

10.1.1.3 Secure Key Generation, Encryption and Storage

The PSP may act as a secure vault for certificates, keys and passwords, negating the need for costly tokens.

10.1.2 Node Security Aspect

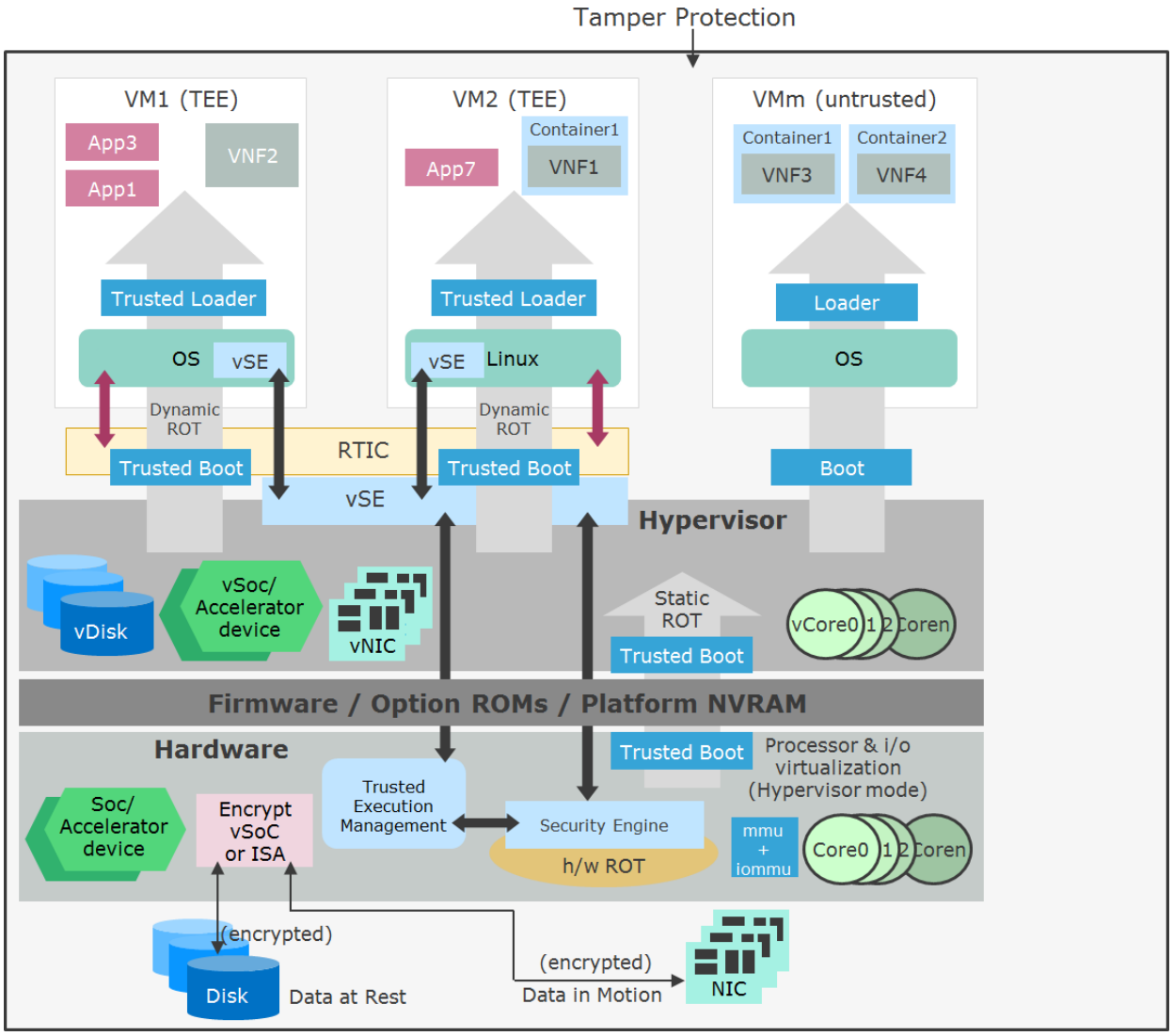


Figure 45 OpenFog Node Security Architecture

The previous figure is divided into four horizontal "zones": First, at the bottom, is the hardware component layer (including external devices). A number of optional (depending on use case requirements) hardware accelerators may be present here. Shown is an encryption device on the SoC (it may also be an external device or present as special instructions in the processors ISA). Other generic accelerators are also shown here. The system

mmu and iommu (which may be a single implementation or split) are also located at this level along with the physical cores. The Hardware Root-of-Trust (HW-RoT) is also a part of the hardware infrastructure and may be embedded on-chip or in an external device which provides this function.

At the next level up is the system Firmware, Option ROMs, and Platform NVRAM. The exact nature and existence of these components is platform dependent. In order to support the HW-RoT and the extension of the Chain of Trust, there must be an immutable firmware implementation resident on trusted system ROM that is the first code to execute on the platform after power on.

Above that is the Hypervisor layer. It instantiates and manages the virtual device instances, e.g., the vSoC devices shown, and assigns them to the virtual machines as directed by the OAM (Operations, Administration, and Management) system. It also instantiates other virtual devices representing physical external devices (such as the vNICs shown). These virtual devices may be entirely supported by hardware that bypasses the hypervisor for data (such as a sr-iov compliant device) or as software emulated virtual instances (such as a hard disk that is shared). The virtual cores, which may or may not be hardware threads if SMT (Simultaneous Multi-Threading) is supported by the physical core, allow for the presentation of additional virtual cores.

The final layer is the layer where VMs are instantiated. The physical resources are mapped here as virtual resources by the hypervisor. The OS in the VM manages the application address spaces which may be instantiated as separate application address spaces or as [Linux] containers.

There are a number of functions which connect the layers and provide system services that help create a secure Chain of Trust comprised of trusted components. These are represented by the vertical arrows between the layers.

One example is the “Security Engine” that instantiates a Trusted Execution Environment and provides services to the hypervisor. The hypervisor, in turn, virtualizes that engine, the vSE – virtual Security Engine shown in the hypervisor layer with an agent resident in each trusted VM. The other is the Trusted Boot firmware and software that verify/measure each subsequent load of firmware or software to establish a Chain of Trust that includes the VM.

The Trusted Boot/Trusted Loader mechanism is meant to ensure that each successive code load, be it firmware or software, is trusted allowing for the extension of the Chain of Trust.

Optionally, untrusted software may be instantiated in a VM to create an untrusted environment. This configuration may be useful for testing untrusted material in an otherwise trusted environment and is secured using the isolation and other security mechanisms described previously.

The RTIC mechanism is described further in 10.1.2.1. It exists in the context of the hypervisor and monitors the state of the areas of memory that should not be modified during execution.

Around the perimeter of the abstract fog node system diagram pictured in is a red line used to describe Physical Security and anti-tamper boundaries implemented by the physical security and anti-tamper mechanisms. We will start with that discussion and then proceed to the Root-of-Trust discussion and work outward from there.

10.1.2.1 Run-time Integrity Checking (RTIC) and Introspection

Secure or measured boot do not ensure that the software that has been securely instantiated is either free of bugs, infection or remains uncompromised during execution. The intent of Runtime Integrity Checking (sometimes called [Hypervisor] Introspection) is to monitor and detect changes to code and static data in the image during execution. This is done by "understanding" the image construction, i.e., where code and static data pages are, by running a set of RTIC specific tools over them before execution. The hypervisor hosts the RTIC mechanism. The underlying assumption is that the hypervisor itself is trusted. RTIC is only used to check VMs. The mechanisms used are mostly passive as the page tables are modified to detect writes to pages that should not be written to. The action on detecting an unauthorized modification is driven by policy. Typically, the VM is terminated.

There are no existing product implementations of this approach but at least one implementation is under development for both the KVM and Xen hypervisors.

The other approach that can address this problem in part is memory encryption as discussed previously. This protects the code and data in an encrypted "container" (not to be confused with Linux containers) from outside attacks. It still may be possible for bugs to be exploited or for previously infected images to be compromised. These "containers" are also

vulnerable to compromised services when they have to go outside the container for services or data. While the code and data (static and dynamic) can be protected in this way, none of the code or data or data outside of the container(s) are protected.

When a more mature solution to this problem is available, fog nodes should implement an RTIC approach to protect the node from being compromised. This does not necessarily prove more useful for nodes in public places than those in protected areas as the attacks do not necessarily require physical access.

10.1.2.2 Debug, Performance Monitoring and Profiling Control

All forms of debug (both hardware and software), performance monitoring, and profiling control should be turned off after system deployment. These mechanisms provide a way for third parties with either physical access or remote access (depending on the mechanism) to provide a technique to either defeat security mechanisms in place or to gain insight into the behavior of the system that allows for future side-channel attacks.

If debugging or other monitoring or profiling information is required in the field, then there must be mechanisms in place to ensure secure provisioning of the authorization for the specific access by legitimate personnel.

10.1.3 Network Security Aspect

As a pervasive computing infrastructure deployed between the OT frontend devices and the cloud computing data centers, a secure OpenFog platform is not merely capable of offering highly-available real-time trusted computing services but also well-positioned to implement dynamic multi-tier defense-in-depth strategies to protect the cyber-physical systems that are critical to our daily living. In order to fulfill these dual missions, the OpenFog platform must augment the enforcement of node security with the provisioning of Network Security and the support of continuous Security Monitoring and Management.

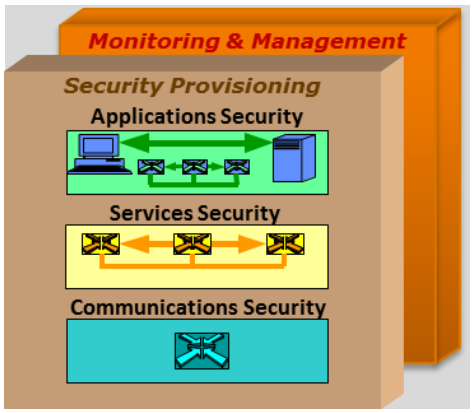


Figure 46 OpenFog Security Functional Layers and Operational Planes[8]

The figure above illustrates such an architecture for providing end-to-end security with its two operational planes: Security Provisioning and Security Monitoring and Management, and its three functional layers: Communications Security, Services Security and Applications Security. This architecture complies with ITU-X.805 Recommendation [X.805] and also conforms to the Software Defined Networking (SDN) Architecture [ONF/SDN] recommended by Open Networking Foundation (ONF). Following subsections discuss the functional layers in greater detail.

10.1.3.1 Communications Security Layer

This layer implements the following communication security services recommended in [X.800] in all the physical/virtual communication channels among all the entities in the Device-Fog-Cloud Computing Hierarchy.

- Confidentiality
  - Connection and Connectionless Data Confidentiality
  - Traffic Flow Confidentiality
- Integrity
  - Connection Integrity with Recovery
  - Connectionless Integrity with Detection
  - Anti-replay Protection
- Authentication
  - Data Origin Authentication for Connectionless Communications
  - Peer Entity Authentication for Connection-based Communications
  - Authenticated Channel Access Control
- Nonrepudiation (optional)
  - Nonrepudiation of Origins
  - Nonrepudiation of Destination

[8] The three security layers conform to the reference architecture of both Open Network Foundation (ONF) and ITU-X.805 recommendation.

The communications occurring in the Device-Fog-Cloud Computing continuum can be categorized into three kinds of Secure Communication Pathways:

- Node-to-Cloud Secure Communication Pathways
- Node-to-Node Secure Communication Pathways
- Node-to-Device Secure Communication Pathways

Since the fog nodes often function as the proxies of cloud servers towards their associated frontend devices while aggregating and representing these frontend devices to the cloud servers, these pathways shall cooperate to preserve the interoperability among the frontend devices and the cloud servers. The following paragraphs highlight the expected functions and the recommended practice of each kind of pathway.

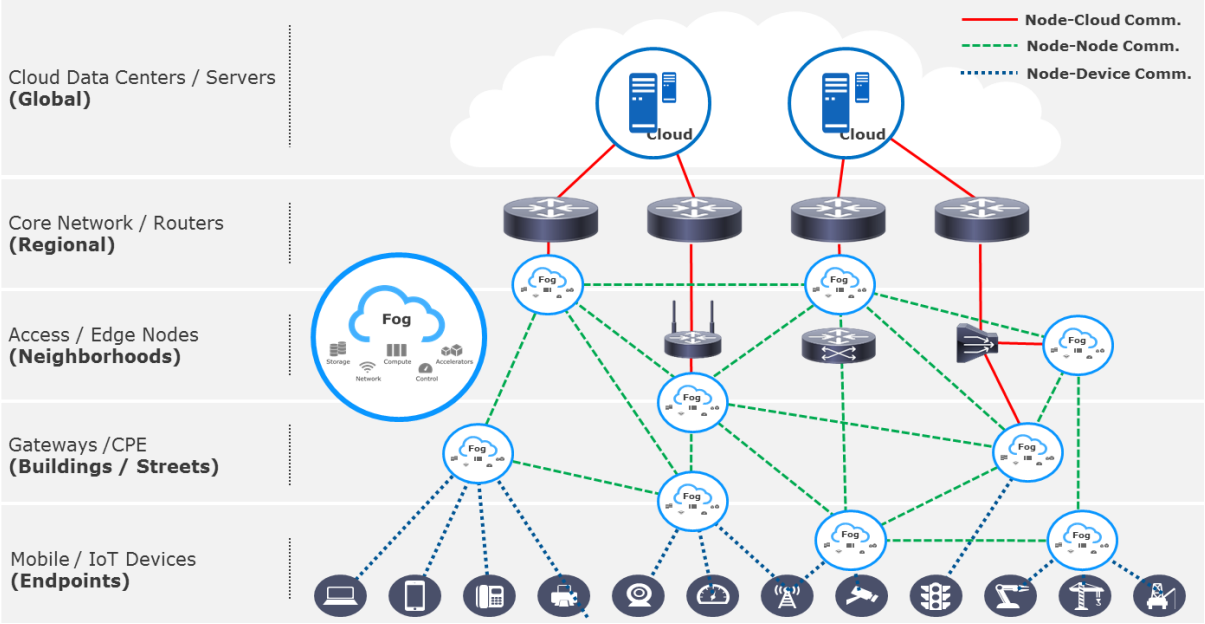


Figure 47 OpenFog Secure Communication Pathways

10.1.3.2 Node-to-Cloud Secure Communication Pathways

In order to secure these Communication Pathways, the fog nodes are expected to implement all the X.800 communication security services (including non-repudiation) for themselves and on behalf of the frontend devices they represent. Strong authentication and non-repudiation services shall be implemented using security credentials derived from the hardware root-of-trust installed in the fog node. Channel Access Control shall be enforced according to the Communication Security Policies established between the cloud service providers and the fog node managers as a part of their service level agreements. All cryptographic operations shall be

performed by the crypto accelerators embedded in the fog nodes while the cryptographic keys shall be managed as a part of the security monitoring and management operation.

These pathways are also expected to preserve the Internet communication protocols and APIs employed by cloud servers to communicate with the frontend devices including IoT devices, personal mobile devices, point-of-sales (POS) terminals, stand-alone computers and servers. Almost all these communications are currently conducted as web service transactions via the following two protocol suites.

Applications	Transaction Protocols	Security Protocols
Enterprise Apps	SOAP over HTTP	WSS
Mobile/Personal Apps	RESTful HTTP/COAP	TLS/DTLS

Figure 48 Protocol Suites for Secure Node-to-Node Communications

10.1.3.3 Node-to-Node Secure Communication Pathways

A distributed fog computing platform may consist of a hierarchy of fog nodes spanning across multiple Internet subnets or administrative domains, and yet these fog nodes are expected to coordinate with one another to accomplish specific objectives. Inter-node information interchanges based on the transaction based client-server computing model and the event based publish-subscribe messaging patterns shall both be implemented in order to enable direct and timely interactions. The following protocol suites are commonly used to implement these paradigms.

Paradigms	Transaction Protocols	Security Protocols
Client-Server	SOAP, RESTful HTTP/COAP	WSS, TLS/DTLS
Publish-Subscribe	MQTT, AMQP, RTPS	TLS/DTLS

Figure 49 Protocol Suites for Secure Node-to-Node Communications

Like the node-to-cloud pathways, the node-to-node pathways expect the fog nodes as the communication endpoints to implement all the X.800 communication security services including non-repudiation. Strong authentication and non-repudiation services shall be implemented using security credentials derived from the hardware root-of-trust installed in the fog nodes. Channel Access Control shall be enforced according to the communication security policies established among the fog node managers as part of their service level agreements. All cryptographic operations shall be performed by the crypto accelerators embedded in the fog nodes while the cryptographic keys shall be managed by the security monitoring and management operation.

10.1.3.4 Node-to-Device Secure Communication Pathways

Often function as the proxies of the cloud servers, the fog nodes communicating are expected to preserve the communication protocols and APIs used by the frontend devices. Unfortunately, the choices of device communication protocols are diversified among different applications and communication media. Efforts on protocol convergence among wireless, powerline communication and industrial automation have been made through adaptation of the Internet (TCP/UDP/IP) protocol suite.

Most of the X.800 communication security services (perhaps, excluding non-repudiation) can be implemented over the wired/wireless Ethernets and on the Internet network and transport layers by well-known security protocols. Among the frontend devices adapted to Internet protocols, strong authentication can be implemented using security credentials issued to the frontend devices. Channel Access Control can be enforced according to the communication security policies specified by the fog service providers. All cryptographic operations can be performed by the crypto-enabled embedded processors in the frontend devices while the cryptographic keys can be managed as a part of the security monitoring and management operation.

However, among many frontend devices that are not Internet savvy and often resource-constrained, only limited cryptographic capability such as symmetric ciphers using manually installed keys is available. These devices must be installed in physically protected environments and connected via hardware connections to one or more fog nodes that can provide most of the X.800 communication security services.

As we further investigate fog computing we will continue to expand the coverage of node-to-device communications.

Layers	Protocols
PHY & MAC Layer	• WLAN: 802.11 • WPAN: 802.15 • PLC: PRIME • Automation: CIP
Wireless Protocol Stacks	• WiFi • Bluetooth • ZigBee
Adaptation Layer	• WLAN/WPAN: 6LowPAN • PLC: PRIME IPv6 SSCS • Automation: EtherNet/IP

Transport/Network Layers	• UDP over IPv6 • TCP over IPv6 • uIPv6 Stack
Application Layer (Publish-Subscribe Messaging)	• CoAP • MQTT • AMQP • RTPS
Routing	• RPL • PCEP • LISP (Cisco)
Security	• 802.1AR – Secure Device Identity • 802.1AE - Media Access Control (MAC) Security • 802.1X – Port-Based (Authenticated) Media Access Control • IPsec AH & ESP, Tunnel/Transport Modes • (D)TLS – (Datagram) Transport Layer Security

Figure 50 Protocol Suites for Secure Node-to-Device Communications

10.1.3.5 Services Security Layer

This layer offers information security services that are provided traditionally by network security appliances such as the following:

- Deep Packet Inspection (DPI)
- Application Layer Proxy
- Lawful Message Intercept
- Intrusion Detection and Protection Systems (IPS/IDS)
- System/Network Event and State Monitoring
- Content Filtering and Parental Control

It may also offer networking services often bundled with security services such as:

- vRouters
- WAN Accelerators
- Network Address Translators (NAT)
- Content Delivery Servers

With the increasing use of Software Defined Networking (SDN) implementations to replace dedicated devices, these "appliances" are increasingly being

implemented as software solutions in virtual machines and Linux containers. Along with other appliances listed above, this category of security appliances is generally referred to as Network Function Virtualization (NFV) or individually as Virtual Network Functions (VNFs). These VNFs, along with other individually packaged services will likely be chained together on a Service Function Chain (SFC) and will use Network Service Headers to route the packets in the selected Service Function Path (SFP).

In many cases, it is believed that these Service Functions will be implemented in OpenFog systems. The NFV and SFC environment present their own set of security issues and include many approaches already discussed but also introduce some new challenges discussed below.

Trusted VNF-to-VNF communication that provides and preserves data integrity and confidentiality requires a number of features from the platform hardware, firmware, and software. In addition to a chain of trust developed from a hardware root of trust, the following features will be required:

- Secure key provisioning for VNF + CA basis for establishing identity
    - Authentication of the Virtual Network Function (VNFC)
    - Asymmetric crypto
- Bulk data encryption
    - Symmetric crypto
- Secure persistent key store
    - For private keys
- Trusted VNF-to-OAM/MANO communication (Integrity, Confidentiality)
    - Secure software update
    - (Same provisioning as above)
- Attestation
    - Both parties are in a secure state

Additional security considerations are introduced in the areas of:

- Service Overlay: Transport forwarding for SFFs
    - Use packet encryption between SFs/VNFs
    - SFF must Authenticate SF/VNF endpoints
- Boundaries of SFC-enabled Domain
    - Authenticate Trusted Parties at Boundary: prevent Spoofing, DDoS, etc.
- Classification

    - Authentication and Authorization of Classification Policy from OAM
- SFC Encapsulation
    - Metadata needs to be authenticated as to origin
    - Selective sharing of sensitive metadata: encrypted or transformed

In addition, the Network Service Header (NSH) provides functionality that creates dynamic relationships that may not be authenticated ahead of time and may fall to the Service Function Forwarder (SFF) to implement.

- Any Service Function (SF) or SFF can update the Service Function Path (SFP) on the fly.
- The SFP can list an SF more than once in the SFP.

In addition:

- The NSH can contain arbitrary metadata fields (fixed or variable length), added by the original classifier or by the SFs or SFFs as the SFP is traversed. These are used to communicate context information that might be useful to other SFs in the chain.

This introduces another data Confidentiality and Privacy condition. Since the metadata can contain any data that one of the components in the SFP deems needed, it is also not clear how to selectively hide (or encrypt) some fields from one SF (when the packet may cross a service provider, customer, or department boundary where certain information is considered proprietary, secret, or sensitive) to others in the chain. The architecture has not yet addressed these issues. It is a complex problem involving dynamic, unknown parties.

10.1.4 Data Security Aspect

There are three general categories in which data resides in a system:

- In memory during processing
- On some kind of non-volatile memory
- In messages sent and received on network interfaces

10.1.4.1 Data in Use

Data is resident in the memory system hierarchy (e.g., SRAM, DRAM, caches, swap space, etc.) during processing. Some of this data, such as

keying material, personal data, company proprietary data and, in some cases, even proprietary algorithms, are considered secret and need to be protected from being read or altered by unauthorized parties.

As already discussed, memory management units (e.g., the mmu, iommu, smmu) can be used to protect memory from unauthorized access from other address spaces (such as VMs) and from devices (either physical or logical/virtual). The read/write/no-execute page attribute bits also provide restricted access within an address space (this assumes that the OS or hypervisor managing the hardware resources is also trusted). The hypervisor can add some additional protection by abstracting and virtualizing hardware that could directly affect the execution context of another virtual machine.

Memory that is resident on swap space should also be protected. The objective is to prevent unauthorized parties from reading the data in the pages on disk by, for example, removing it and reading it on another system. This can be done using encryption. Whole disk encryption is one approach to providing this functionality at a relatively low overhead.

Access to memory using external hardware debuggers, such as JTAG, and software debuggers should not be enablable in production systems in the field. JTAG should always be turned off when leaving a lab or controlled environment. When debugging is required, and if permitted, in the field, controls must be in place to ensure only an authorized user can use the debug interface and that it is disabled for all other access.

Encrypted Memory

Memory encryption is used to provide Confidentiality of code and data during execution. It is used to protect secrets in memory even when parts of the rest of the system have been compromised. Memory encryption is used based on the fact that only the CPU package is considered trusted – the memory is not. As a side effect, it also prevents an attacker from injecting code into a running image as decryption would result in corrupt code and program failure.

Memory encryption schemes typically use symmetric key cryptography because of its speed. This functionality requires both additional hardware support, including an encryption device resident in the memory management subsystem, some operating system support to manage the encryption hardware, and a method for managing the keys associated with the encrypted memory.

Memory encryption is not without a cost. The dynamic decryption/encryption of memory as it is fetched into cache and written back to memory from cache affects memory response times.

A complete discussion of memory encryption technology is beyond the scope of this paper. However, the technology has clear security advantages for at least some classes of data and some applications in a fog computing environment. It may be implemented where it is justified by the threat analysis.

10.1.4.2 Data at Rest

Data at Rest refers to data resident on some non-volatile storage, such as hard disk or, SSD, USB thumb drives, CDs, DVDs, etc. Encryption is the front-line defense for data at rest. Among other classes of data, it protects personally identifiable information (Privacy) and other sensitive data (Confidentiality). It limits access to those with the correct keys, preventing anyone who doesn't have the keys from accessing the data. It provides protection against unauthorized access to data should the storage media become physically compromised in some way.

It also meets many compliance requirements, removes any concern regarding retirement of the storage media and obviates the threat of physical compromise of the data against unauthorized access - even if someone with physical access walks away with the drive from the fog node, they will not have access.

Encryption by itself is not sufficient - keys, policies and certificates must be actively managed in a secure store, making sure that they are not compromised and do not fall into the wrong hands.

A process for monitoring who, what, where, when and how data is accessed from within databases, applications, and the OS/file system should be put in place. Both access to sensitive information and unauthorized access attempts should be monitored. All security events should be logged for subsequent forensic analysis and use by the Operations, Administration, and Maintenance (OAM) system – e.g., it may use the access data to determine if an attack is underway. The policies are specified by the OAM system.

Secure data at rest mechanisms must be built on a secure chain of trust, from power-on through the boot phases, through the instantiation of the hypervisor (if used), and through instantiation of the operating system and application in the VM (if a virtual environment is used).

There are generally three methods of securing and encrypting data at rest:

Full Disk Encryption

Full-disk encryption is typically implemented using a hardware-based encryption mechanism in the disk firmware although software disk encryption implementations also exist. It works by automatically encrypting all data written to the disk and automatically decrypting all data read from the disk. Both operations depend on having the correct authentication key. Without the proper authentication key, even if the hard drive is removed it cannot be read by another machine running the same or different software. The advantage of full-disk encryption is that it requires no special attention on the part of software or the OAM system. If software encryption is used, because everything on the hard drive is encrypted, including the operating system, the encrypt/decrypt process can increase data access times. Full-disk encryption will be most useful for fog devices located in publically accessible locations (e.g., malls, lamp posts, street corners, roadside, in vehicles, etc.). Because one key is used to encrypt the entire hard drive, the OAM system should provide an encryption key backup mechanism in case the system becomes non-functional for some reason and data retrieval is required. Carefully managed secure backups may also be used.

File System (and Database) Encryption

File system-level encryption provides a means to use a separate key-based access and authentication mechanism to protect specific files on a file or directory/folder basis. It is used when individual files stored on disk (or other media) need to be protected even from other applications (or users) that have access to a fully encrypted disk. In use files are encrypted using symmetric File Encryption Key (FEK). The FEK in turn is encrypted using owner's public key. The encrypted FEK is stored with the encrypted file. To decrypt the file, the file system first decrypts the embedded FEK using the private key that matches the owner's public key. Then the file is decrypted using the FEK. Whole databases, or individual records or fields in records may also be encrypted. File system encryption may be used by applications, running in the same VM, that consider their data proprietary or for files that contain otherwise sensitive or private data.

File System Access Control Mechanisms

File system access control mechanisms may be used to restrict access to specific files or groups of files by userid or groupid. All modern file systems implement file permissions in some form. For discussion purposes, the Linux

file system is used here. Most of this type of access control will be similar in other systems.

The Basic File Permissions are applied to Permission Groups using Permission Types. This is not intended to be a complete or extensive discussion of file system permissions, but is intended for discussion purposes in the document.

Permission Groups - Each file and directory has three user based permission groups:

- owner - The Owner permissions apply only the owner of the file or directory. They do not impact the actions of other users.
- group - The Group permissions apply only to the group that has been assigned to the file or directory. They do not affect the actions of other users.
- all users - The All Users permissions apply to all other users on the system. This is the permission group that is usually most important.

Permission Types - Each file or directory has three basic permission types:

- read - The Read permission refers to a user's capability to read the contents of the file.
- write - The Write permissions refer to a user's capability to write or modify a file or directory.
- execute - The Execute permission affects a user's capability to execute a file or view the contents of a directory.

These mechanisms are important controls within the context of the OS that defines userids and groupids. Usually an administrator. Operating from the OAM, will specify access permissions when a userid and/or groupid is set up for a specific OS file system environment. This may or may not be used in a given fog system. It may be important if different applications are running in the same OS (VM) context, each with the need to different access to the data in a shared file system (e.g., read-only for some, and read-write for the data producing application.

10.1.4.3 Data in Motion

Data in motion, sometimes known as *data in transit*, is used here to describe packets sent and received on a network interface (including virtual network interfaces) from or to a fog node – i.e., information that is moving through a network. Encryption should be implemented for all sensitive or private data

in motion: using VPNs, SSL and other technologies which can protect data from being compromised or seen in plaintext form while in transit.

There are two ways to use encryption when trying to protect data in motion: using an encrypted connection or using an encrypted file.

An encrypted connection is one in which anything that is sent over a network connection is automatically encrypted, regardless of the encryption status of the information to be sent. For example, if sending an already encrypted file, it will get encrypted again (with a different key) while being sent.

Another method of ensuing data in motion is secure during transit is to use an already encrypted file. Since an encrypted file exists in encrypted form, it will always be encrypted, and therefore, protected.

11 Glossary

Term	Definition	Source
Access Control	Means to ensure that access to assets is authorized and restricted based on business and security requirements. *Note:* Access control requires both authentication and authorization.	ISO/IEC 27000:2014
Actuators	"An actuator is a mechanical device for moving or controlling a mechanism or system. It takes energy, usually transported by air, electric current, or liquid, and converts that into some kind of motion."	[Sclater2007]
Address	An address is used for locating and accessing – "talking to" – a Device, a Resource, or a Service. In some cases, the ID and the Address can be the same, but conceptually they are different.	IOT-A
Analytics	Synthesis of knowledge from information.	NIST Interagency Publication 8401-1
Appliance	A computer appliance is generally a separate and discrete hardware device with integrated software, specifically designed to provide a specific computing resource.	Wikipedia
Application Software	"Software that provides an application service to the user. It is specific to an application in the multimedia and/or hypermedia domain and is composed of programs and data".	[ETSI- ETR173]

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".	[IEEE-1471-2000]
Architecture Description	Work product used to express architecture.	[ISO/IEC 42010:2011]
Architecture Framework	Conventions, principles and practices for the description of architectures established within a specific domain of application and/or community of stakeholders	ISO/IEC 42010:2011
Architecture Vision	"A high-level, aspirational view of the target architecture."	[TOGAF9]
Aspiration	"Stakeholder Aspirations are statements that express the expectations and desires of the various stakeholders for the services that the final [system] implementation will provide."	[E-FRAME]
Authentication	Authentication is the process of verifying a user's true identity. This may involve the use of one or more means of proof of identification, also known as factors, such as PIN codes and smart cards.	Nexus IoT Glossary
Authorization	Granting of rights, which includes the granting of access based on access rights.	[ISO 7498-2:1989]
Autonomy	The ability of an intelligent system to independently compose and select among different courses of action to accomplish goals based on its knowledge and understanding of the world, itself, and the situation.	IHMC
Availability	Property of being accessible and usable upon demand by an authorized entity.	ISO/IEC 27000:2014
Business Logic	Goal or behavior of a system involving Things serving a particular	IOT-A

	business purpose. Business Logic can define the behavior of a single Thing, a group of Things, or a complete business process.	
Choreography	Type of composition whose elements interact in a non-directed fashion with each autonomy part knowing and following an observable predefined pattern of behavior for the entire (global) composition.	ISO/IEC DIS 18834-1
Collaboration	Type of composition whose elements interact in a non-directed fashion, each according to their own plans and purposes without a predefined pattern of behaviour	ISO/IEC DIS 18834-1
Confidentiality	Property that information is not made available or disclosed to unauthorized individuals, entity, or processes	ISO/IEC 27000:2014
Cloud	Or, "The Cloud," is generally used as shorthand for Cloud Computing. The name "Cloud" comes from the fluffy cloud typically used in Visio-style network diagrams to represent a connection to the Internet.	IoT Guide
Cloud Computing	A general term for the delivery of various hosted services over the Internet. The "as-a-Service" moniker is used for cloud services such as Software-as-a-Service, Platform-as-a-Service and Infrastructure-as-a-Service. The back-end for many IoT devices may be delivered via the Cloud.	IoT Guide
Communication Model	The communication model aims at defining the main communication paradigms for connecting elements. This model provides a set of communication rules to build interoperable stacks, together with insights about the main interactions among the elements of the domain model.	IOT-A

Composition	Result of assembling a collection of elements for a particular purpose	ISO/IEC DIS 18834-1
Constrained Network	A constrained network is a network of devices with restricted capabilities regarding storage, computing power, and / or transfer rate.	IOT-A
Controller	Anything that has the capability to affect a Physical Entity, like changing its state or moving it.	IOT-A
Credentials	A credential is a record that contains the authentication information (credentials) required to connect to a resource. Most credentials contain a user name and password.	IOT-A
Cryptography	Discipline that embodies principles, means, and mechanisms for the transformation of data in order to hide its information content, prevent its undetected modification and/or prevent its unauthorized use	ISO/IEC 18014-2:2009
Data-centricity	Scalable, real-time, dependable, high-performance and interoperable data exchanges between publishers and subscribers.	Object Management Group
Device	Physical entity embedded inside, or attached to, another physical entity in its vicinity, with capabilities to convey digital information from or to that physical entity	IIC
Device Endpoint	Endpoint that enables access to a device and thus to the related physical entity.	IIC
Digital Entity	Any computational or data element of an IT-based system.	IOT-A
DIKW	**D**ata gathered becomes **I**nformation when stored and retrievable becomes **K**nowledge. Knowledge enables **W**isdom for autonomous IoT.	
Discovery	Discovery is a service to find	IOT-A

	unknown resources/entities/services based on a rough specification of the desired result. It may be utilized by a human or another service. Credentials for authorization are considered when executing the discovery.	
Edge Gateway	Endpoint that provides an entry point into enterprise or service provider core networks	IIC
Element	Unit that is indivisible at a given level of abstraction and has a clearly defined boundary Note: An element can be any type of entity	ISO/IEC DIS 18834-1
Endpoint	One of two components that either implements and exposes an interface to other components or uses the interface of another component.	ISO/IEC 24791-1:2010
Enterprise	Segment of computing mostly focused at traditional IT and Industrial IT.	OpenFog
Edge Computing	Also referred to as Mesh Computing, this concept places applications, data and processing at the logical extremes of a network rather than centralizing them. Placing data and data-intensive applications at the Edge reduces the volume and distance that data must be moved.	IoT Guide
Fog Computing	Fog computing is a system-level horizontal architecture that distributes resources and services of computing, storage, control and networking anywhere along the continuum from Cloud to Things, thereby accelerating the velocity of decision making. Fog-centric architecture serves a specific subset of business problems that cannot be successfully implemented using only	OpenFog Consortium

	traditional cloud based architectures or solely intelligent edge devices.	
Fog Node	The physical and logical network element that implements fog computing services. It is somewhat analogous to a server in cloud computing.	OpenFog Consortium
Gateway	A Gateway is a forwarding element, enabling various local networks to be connected.	IOT-A
Global Storage	Storage that contains global information about many entities of interest. Access to the global storage is available over the internet.	IOT-A
Identity	Properties of an entity that makes it definable and recognizable.	IOT-A
Industry 4.0	Refers to the fourth industrial revolution, following the first (mechanization of production through water and steam power), second (use of electricity for mass production), and third (use of electronics and IT for automation). Experts believe that the fourth revolutionary leap will entail full computerization of traditional industries. A key element of Industry 4.0 is the Smart Factory marked by adaptability, resource efficiency and ergonomics as well as intelligent processes and communication. Technological basis are cyber-physical systems and the Internet of Things.	Nexus
Industrial Internet	An Internet of things, machines, computers and people, enabling intelligent industrial operations using advanced data analytics for transformational business outcomes.	IIC

Information Model	"An information model is a representation of concepts, relationships, constraints, rules, and operations to specify data semantics for a chosen domain of discourse. The advantage of using an information model is that it can provide sharable, stable, and organized structure of information requirements for the domain context. The information model is an abstract representation of entities, which can be real objects such as devices in a network, or logical, such as the entities used in a billing system. Typically, the information model provides formalism to the description of a specific domain without constraining how that description is mapped to an actual implementation. Thus, different mappings can be derived from the same information model. Such mappings are called data models."	[AutoI]
Infrastructure Services	Specific services that are essential for any IoT implementation to work properly. Such services provide support for essential features of the IoT.	IOT-A
Internet	"The Internet is a global system of interconnected computer networks that use the standard Internet protocol suite (TCP/IP) to serve billions of users worldwide. It is a network of networks that consists of millions of private, public, academic, business, and government networks of local to global scope that are linked by a broad array of electronic and optical networking technologies. The Internet carries a vast array of	[Wikipedia IN]

	information resources and services, most notably the inter-linked hypertext documents of the World Wide Web (WWW) and the infrastructure to support electronic mail. Most traditional communications media, such as telephone and television services, are reshaped or redefined using the technologies of the Internet, giving rise to services such as Voice over Internet Protocol (VoIP) and IPTV. Newspaper publishing has been reshaped into Web sites, blogging, and web feeds. The Internet has enabled or accelerated the creation of new forms of human interactions through instant messaging, Internet forums, and social networking sites. The Internet has no centralized governance in either technological implementation or policies for access and usage; each constituent network sets its own standards. Only the overreaching definitions of the two principal name spaces in the Internet, the Internet-protocol address space and the domain-name system, are directed by a maintainer organization, the Internet Corporation for Assigned Names and Numbers (ICANN). The technical underpinning and standardization of the core protocols (IPv4 and IPv6) is an activity of the Internet Engineering Task Force (IETF), a non-profit organization of loosely affiliated international participants that anyone may associate with by contributing technical expertise.”	

Internet of Things (IoT)	The digital network is soon going to connect physical objects ("things"), persons, machines, devices and processes. It is expected that 50 Billion devices will be connected to the Internet by 2020. Contrary to the Internet as we know it, where only persons have digital identities, the Internet of Things equips physical objects with digital identities. The objects are embedded with software, electronics and sensors that allow them to communicate with other objects or persons in the digital or physical world. IoT will transform all industries – it is expected that the new connectivity will set off automation in almost all fields of business. Establishing secure infrastructures and trustworthy identities is vital for the successful deployment of this new kind of network.	Nexus
Interoperability	The ability to share information and services. The ability of two or more systems or components to exchange and use information. The ability of systems to provide and receive services from other systems and to use the services so interchanged to enable them to operate effectively together.	[TOGAF 9]
IoT Service	Software component enabling interaction with resources through a well-defined interface. Can be orchestrated together with non-IoT services (e.g., enterprise services). Interaction with the service is done via the network.	IOT-A
Local Storage	Special type of resource that contains information about one or only a few entities in the vicinity of	IOT-A

	a device.	
LTE	Long Term Evolution commonly used in 4G.	
Microservices	Microservices can be considered a specialization or extension of service-oriented architectures (SOA) used to build distributed software systems. As with SOA, services in a microservice architecture are processes that communicate with each other over a network in order to fulfill a goal. Also, like SOA, these services use technology-agnostic protocols. The microservices' architectural style is a first realization of SOA that followed the introduction of DevOps and is becoming more popular for building continuously deployed systems. SOA is more focused on reusability and segregation whereas microservices focus on replacing a large application(s), with a system that can incrementally evolve and is easier to manage.	Wikipedia
Middleware	Middleware is computer software that provides services to software applications beyond those available from the operating system. It can be described as "software glue". Middleware makes it easier for software developers to implement communication and input/output, so they can focus on the specific purpose of their application.	Wikipedia
Mobile Edge	A standard mostly concerned with	MEC

Computing (MEC)	equipping computational resources at or near base stations in mobile / cellular networks	
Modularity	A property of network elements where individual capabilities can be added or removed without substantial impact of other components.	OpenFog Consortium
Multi-tenancy	Software Multitenancy refers to a software architecture in which a single instance of a software application runs on a server and serves multiple tenants. A tenant is a group of users who share a common access with specific privileges to the software instance. With a multitenant architecture, a software application is designed to provide every tenant a dedicated share of the instance including its data, configuration, user management, tenant individual functionality and non-functional properties.	Wikipedia
Network resource	Resource hosted somewhere in the network, e.g., in the cloud.	IOT-A
On-device Resource	Resource hosted inside a Device and enabling access to the Device and thus to the related Physical Entity.	IOT-A
On-Premises Software	On-premises software (sometimes abbreviated as "on-prem") is installed and runs on computers on the premises (in the building) of the person or organization using the software, rather than at a remote facility such as a server farm or cloud.	
Operational Technology	Operational Technology (OT) is the use of computers (or other processing devices) to monitor or	Wikipedia

	alter the physical state of a system, such as the control system for a power station or the control network for a rail system. The term has become established to demonstrate the technological and functional differences between traditional IT systems and Industrial Control Systems environment, the so-called "IT in the non-carpeted areas".	
Orchestration	Type of composition where one particular element is used by the composition to oversee and direct the other elements. *Note:* the element that directs an orchestration is not part of the orchestration.	ISO/IEC DIS 18834-1
Private Cloud	Private cloud is cloud infrastructure operated solely for a single organization, whether managed internally or by a third-party, and hosted either internally or externally.	Wikipedia
Reference Architecture	A Reference Architecture (RA) is an architectural design pattern that indicates how an abstract set of mechanisms and relationships realizes a predetermined set of requirements. It captures the essence of the architecture of a collection of systems. The main purpose of a Reference Architecture is to provide guidance for the development of architectures. One or more reference architectures may be derived from a common reference model, to address different purposes/usages to which the Reference Model may be targeted.	IOT-A

Reference Model	A reference model is an abstract framework for understanding significant relationships among the entities of some environment. It enables the development of specific reference or concrete architectures using consistent standards or specifications supporting that environment. A reference model consists of a minimal set of unifying concepts, axioms and relationships within a particular problem domain, and is independent of specific standards, technologies, implementations, or other concrete details. A reference model may be used as a basis for education and explaining standards to non-specialists.	[OASIS-RM]
Reliability	Ability of a system or component to perform its required functions under stated conditions for a specified period of time.	ISO/IEC 27040:2015
Resilience	The condition of the system being able to avoid, absorb and/or manage dynamic adversarial conditions while completing assigned mission(s), and to reconstitute operational capabilities after casualties.	IIC
Resource	Computational element that gives access to information about or actuation capabilities on a Physical Entity.	IOT-A
Requirement	A quantitative statement of business need that must be met by a particular architecture or work package.	[TOGAF9]
Scalability	A property of networks where their capabilities can grow or shrink without undue expense of loss of efficiency	OpenFog Consortium
Sensor	A sensor is a special Device that	IOT-A

	perceives certain characteristics of the real world and transfers them into a digital representation.	
Security	The correct term is 'information security' and typically information security comprises three component parts: ▪ Confidentiality. Assurance that information is shared only among authorized persons or organizations. Breaches of confidentiality can occur when data is not handled in a manner appropriate to safeguard the confidentiality of the information concerned. Such disclosure can take place by word of mouth, by printing, copying, e-mailing or creating documents and other data etc.; ▪ Integrity. Assurance that the information is authentic and complete. Ensuring that information can be relied upon to be sufficiently accurate for its purpose. The term 'integrity' is used frequently when considering information security as it represents one of the primary indicators of information security (or lack of it). The integrity of data is not only whether the data is 'correct', but whether it can be trusted and relied upon; ▪ Availability. Assurance that the systems responsible for delivering, storing and processing information are accessible when needed, by those who need them.	[ISO27001]
Service	Services are the mechanism by which needs and capabilities are brought together	[OASIS-RM]

Smart Gateway	A Gateway is a forwarding element, enabling various local networks to be connected. A Smart (or Intelligent) Gateway additionally provides more resources for local (edge) computing. These resources can include middleware, microservices and applications. As such, a Smart (or Intelligent) Gateway begins to resemble a fog Node, as a network element that provides some fog computing services. Smart Gateways and fog Nodes are thus also Appliances.	OpenFog Consortium
Storage	Special type of Resource that stores information coming from resources and provides information about Entities. They may also include services to process the information stored by the resource. As Storages are Resources, they can be deployed either on-device or in the network.	IOT-A
System	A collection of components organized to accomplish a specific function or set of functions.	[IEEE-1471-2000]
Thing	Generally speaking, any physical object. In the term 'Internet of Things' however, it denotes the same concept as a Physical Entity.	IOT-A
Unconstrained Network	An unconstrained network is a network of devices with no restriction on capabilities such as storage, computing power, and / or transfer rate.	IOT-A
View	The representation of a related set of concerns. A view is what is seen from a viewpoint. An architecture view may be represented by a model to demonstrate to stakeholders their areas of interest in the architecture. A view does not have to be visual or graphical in	[TOGAF 9]

	nature.	
Viewpoint	A definition of the perspective from which a view is taken. It is a specification of the conventions for constructing and using a view (often by means of an appropriate schema or template). A view is what you see; a viewpoint is where you are looking from - the vantage point or perspective that determines what you see.	[TOGAF 9]
Virtual Entity	Computational or data element representing a Physical Entity. Virtual Entities can be either Active or Passive Digital Entities.	IOT-A
Wireless communication technologies	Wireless communication is the transfer of information over a distance without the use of enhanced electrical conductors or "wires". The distances involved may be short (a few meters as in television remote control) or long (thousands or millions of kilometers for radio communications). When the context is clear, the term is often shortened to "wireless". Wireless communication is generally considered to be a branch of telecommunications.	[Wikipedia WI]
Wire line communication technologies	A term associated with a network or terminal that uses metallic wire conductors (and/or optical fibers) for telecommunications.	[setzer-messtechnik2010]
Wireless Sensors and Actuators Network	Wireless sensor and actuator networks (WSANs) are networks of nodes that sense and, potentially, control their environment. They communicate the information through wireless links enabling interaction between people or computers and the surrounding environment.	[OECD2009]

References

[IOT-A] EU IOT-A Terminology.

Online at: http://www.iot-a.eu/public/terminology/copy_of_term

[AIMglobal] Association for Automatic Identification and Mobility.

Online at: http://www.aimglobal.org/

[AutoI] Information Model, Deliverable D3.1, Autonomic Internet (AutoI) Project.

Online at: http://ist-autoi.eu/autoi/d/AutoI_Deliverable_D3.1_-_Information_Model.pdf

[CCSDS 312.0-G-0] Information architecture reference model.

Online at: http://cwe.ccsds.org/sea/docs/SEA-IA/Draft%20Documents/IA%20Reference%20Model/ccsds_rasim_20060308.pdf

[COMPDICT-M2M] Computer Dictionary Definition

Online at: http://www.yourdictionary.com/computer/m2-m

[E-FRAME] E-FRAME project, available.

Online at: http://www.frame-online.net/top-menu/the-architecture-2/faqs/stakeholder-aspiration.html

[EPCglobal] EPC Global glossary (GS1).

Online at:

http://www.epcglobalinc.org/home/GS1_EPCglobal_Glossary_V35_KS_June_09_2009.pdf

[ETSI-ETR173] ETSI Technical report ETR 173, Terminal Equipment (TE); Functional model for multimedia applications.

Online at:
http://www.etsi.org/deliver/etsi_etr/100_199/173/01_60/etr_173e01p.pdf

[ETSI TR 102 477] ETSI Corporate telecommunication Networks (CN); Mobility for enterprise communication.

Online at:
http://www.etsi.org/deliver/etsi_tr/102400_102499/102477/01.01.01_60/tr_102477v010101p.pdf

[IEEE-1471-2000] IEEE 1471-2000, "IEEE Recommended Practice for Architectural Description of Software-Intensive Systems"

[ITU-IOT] the Internet of Things summary at ITU.

Online at:
http://www.itu.int/osg/spu/publications/internetofthings/InternetofThings_summary.pdf

[ISO/IEC 2382-1] Information technology -- Vocabulary -- Part 1: Fundamental terms

Online at:
http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=7229

[ISO 27001] ISO 27001: An Introduction to Information, Network and Internet Security

[OGS] Open GeoSpatial portal, the OpenGIS abstract specification Topic 12: the OpenGIS Service architecture.

Online at: http://portal.opengeospatial.org/files/?artifact_id=1221

[OASIS-RM] Reference Model for Service Oriented Architecture 1.0

Online at: http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.pdf

[OECD2009]: "Smart Sensor Networks: Technologies and Applications for Green Growth", December 2009.

Online at: http://www.oecd.org/dataoecd/39/62/44379113.pdf

[Sclater2007] Sclater, N., Mechanisms and Mechanical Devices Sourcebook, 4th Edition (2007), 25, McGraw-Hill

[setzer-messtechnik] setzer-messtechnik glossary, July 2010.

Online at: http://www.setzer-messtechnik.at/grundlagen/rf-glossary.php?lang=en

[TOGAF9] Open Group, TOGAF 9, 2009

[Wikipedia IN] Internet page on Wikipedia, online at: http://en.wikipedia.org/wiki/Internet

[ROZANSKI2005] Software Architecture with Viewpoints and Perspectives.

Online at: http://www.viewpoints-and-perspectives.info/doc/spa191-viewpoints-and-perspectives.pdf

[Wikipedia WI] Wireless page on Wikipedia.

Online at: http://en.wikipedia.org/wiki/Wireless

[IoT Guide] Internet of Things Guide.

Online at: http://internetofthingsguide.com/d/cloud.htm

[Nexsus] Nexus IoT Glossary.

Online at: https://www.nexusgroup.com/en/glossary/?letter=C

[UF IoT] Universal Framework IoT Glossary.

Online at: https://universalframeworks.com/industrial-internet-of-things-i-iot-glossary/

[AutoI] Information Model, Deliverable D3.1, Autonomic Internet (AutoI) Project.

Online at: http://ist-autoi.eu/autoi/d/AutoI_Deliverable_D3.1_-_Information_Model.pdf

[TOGAF9] Open Group, TOGAF 9, 2009

[Ref-a] - http://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf

[Ref-b] - http://csrc.nist.gov/publications/fips/fips140-2/fips1402annexa.pdf

[Ref-c] - http://csrc.nist.gov/publications/nistpubs/800-131A/sp800-131A.pdf

Mobile Edge Computing: http://www.etsi.org/technologies-clusters/technologies/mobile-edge-computing

Industrial Internet Consortium: http://www.iiconsortium.org/

Open Connectivity Foundation: http://openconnectivity.org

