Fog Use Case Scenarios

Use Case: Patient Monitoring
Vertical: Smart Healthcare

An OpenFog Consortium Architectural Use Case

1 Snapshot: Fog-Enabled Patient Monitoring

WHY FOG

Why is fog the best architecture for this use case?

With its real-time communications and analytics requirements for data from thousands of low-level sensors, hospital patient care requires the scalability and agility of fog. Doctors at anywhere or anytime may need access to patient information, even if they're in another hospital. Healthcare networks are large in terms of complexity and the number of metrics being tracked per patient, and fog is ideally architected to accommodate.

WHICH FOG PILLAR

Which fog pillar best describes this use case?

The OpenFog pillar that is most amplified by the Smart Healthcare use case is the Security pillar. Fog computing and networking provides localized, highly secure fog intelligence that can mitigate the hacker threat while satisfying the privacy requirements associated with HIPAA and other regulations.

VALUE

What are the business advantages of building this use case with fog?

The primary value of fog-enabled healthcare is better patient outcomes. There are also business advantages associated with improved processing efficiencies – with fog, doctors have access to real-time patient monitoring data from anywhere in the fog ecosystem.

CLOUD & EDGE

How does this use case augment or supersede cloud and edge architectures?

Fog enables real-time management of low-level sensor data that can't be done from the cloud. Fog's distributed architecture and hierarchical structure are critical to moment-to-moment healthcare operations. Additionally, fog provides for the security and privacy dimensions required in hospital and patient care settings.

2 Table of Contents

3 Introduction

Note: The preamble section of this document (pages 3 through 11) is common across all OpenFog use cases. It provides descriptions and reference points for fog architectural attributes and properties. The Patient Monitoring use case begins on page 11.

The OpenFog Consortium is defining applications and architectures for fog computing. The Consortium defines fog computing as: ***A horizontal, system-level architecture that distributes computing, storage, control and networking functions closer to the users along a cloud-to-thing continuum***.

The first step in this architectural process is understanding the spectrum of vertical markets and applications that we expect fog computing technologies may serve. This document focuses on a representative use case that we believe spans many aspects of fog computing and therefore serves to define the functions we hope fog architecture, fog implementations, and fog deployments will provide.

It is important to understand how this use case fits into the overall process the Consortium uses to define interoperable and certifiable architectures. As shown in Figure 1, the use case described in detail in this document is a starting point for the suite of OpenFog technical documentation. When taken together, OpenFog use cases cover the basic fog functions of approximately 80% of the comprehensive set of IoT network applications we have identified for fog.

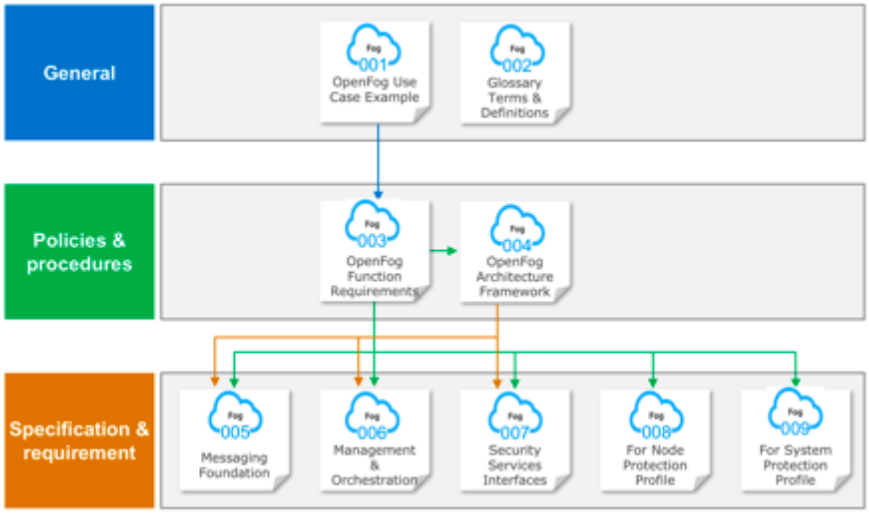

Figure 1. Hierarchy of OpenFog Consortium specification documentation

The composite of all use cases outlines a problem statement for OpenFog, describing the essential functions for all fog elements and networks. The Consortium extracts requirements from these use cases, and distills and correlates them to produce a detailed *Fog Requirements Document*. These requirements serve three important purposes:

1. To drive the OpenFog Reference Architecture;
2. To guide the development of OpenFog testbeds for testing and validation purposes; and
3. To provide guidance to implementers of fog nodes and networks.

The *Architecture Framework Document* is a compendium document that describes the key functional components of OpenFog as well as the interfaces between these components.

The Consortium also publishes additional documents, which describe details in areas such as security, management and orchestration, and messaging. Implementers may use the compendium as a guide for the conceptual planning and architecture design for their fog-based systems, and as implementation best practices for OpenFog elements and networks that will interoperate and can be certified as OpenFog compliant.

OpenFog Consortium workgroups reviewed and discussed hundreds of potential fog use cases spanning more than a dozen vertical markets related to IoT. The Consortium carefully selected a set of use cases that we believe spans a representative set of potential fog applications.

These use cases will highlight one or more representative attributes of fog such as latency, network bandwidth, reliability, security, programmability, scalability. The derived requirements from the use cases we include will cover an illustrative sample.

As mentioned, OpenFog technical requirements comprise a platform that covers approximately 80% of common fog functions. The remaining 20% of requirements needed to support specific use cases which are application dependent and won't be defined by the Consortium.

Readers should pay detailed attention to the subset of use cases that most closely match their areas of interest. We encourage you to browse additional use cases, as they may highlight less obvious aspects of fog that could prove valuable, and give insight into the rationale of the OpenFog requirements.

Readers are also encouraged to collect additional use cases and submit them to OpenFog for requirements extraction and potential inclusion in future use case documents.

4 Fog Computing Overview

Fog computing provides the missing link in the cloud-to-thing continuum. It is a critical architecture for today's connected world as it enables low latency, reliable operation, and removes the requirement for persistent cloud connectivity to address emerging use cases in Internet of Things (IoT), 5G, Artificial Intelligence (AI), Virtual Reality and Tactile Internet applications.

Fog architectures selectively move compute, storage, communication, control, and decision making closer to the network edge where data is being generated and used. This solves the limitations in current infrastructure to enable mission-critical, data-dense use cases.

Fog computing is an extension of the traditional cloud-based computing model where implementations of the architecture reside in multiple layers of a network's hierarchy. These extensions to the fog architecture may retain all the benefits of cloud computing, such as containerization, virtualization, orchestration, manageability, and efficiency.

The fog computing model provides the ability to move computation and storage from the cloud closer the edge, based on the needs of the data and the service requirements. These functions can potentially reside right next to the IoT sensors and actuators. The computational, networking, storage and acceleration elements of this new model are known as fog nodes. These nodes may also reside in the cloud, as they comprise a fluid system of connectivity and don't have to be fixed to the physical edge.

5 The OpenFog Reference Architecture

The OpenFog Consortium was founded on the principle that an open and interoperable 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 fog solutions are of limited value, as they can limit supplier diversity and ecosystems, resulting in a detrimental impact on market adoption, system efficiency, quality and innovation.

The OpenFog Reference Architecture (RA) is a medium- to high-level view of system architectures for fog nodes and networks. It is the result of a broad collaborative effort of the OpenFog ecosystem of industry, technology and university/research leaders. It was created to help business leaders, software developers, silicon architects and system designers create and maintain the hardware, software and system elements necessary for fog computing, as well as design, architect and develop solutions that enable fog-cloud, fog-thing and fog-fog interfaces.

6 Benefits of Fog

Fog computing targets cross-cutting concerns such as the control of performance, latency and efficiency, which are also key to the success of fog networks. Cloud and fog computing are on path to 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 and single purpose functions go to fog and what goes to the backend cloud, are application and implementation/use case specific.

This segmentation can be planned and static, but can also change dynamically if the network state changes in areas such as processor loads, link bandwidths, storage capacities, fault events, security threats, energy availability, cost targets, and so on.

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

To illustrate this concept, let's look at a quick use case example: 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) to 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 delayed or, in the worst case, completely 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. These fog nodes immediately establish a community which provides the ability to collaborate. They can be highly secure, lessening the hacker threat. Fog nodes can also be given the authority to 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.

Fog computing includes a set of high-level attributes of fog computing that we call the pillars; these include some of the fog advantages described in the pipeline control scenario. There are 8 pillars in total: security, scalability, openness, autonomy, reliability, agility, hierarchical organization and programmability. We will discuss all of these pillars in detail later in this document.

The 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 illustrate how FaaS may be enabled; this will be expanded upon in the reference architecture document.

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, 5G and AI applications that require real-time decision making, low latency, improved security, privacy protection and are network-constrained.

7 Use Case Scenario: Patient Monitoring

Use Case: Patient Monitoring

Vertical: Smart Healthcare

Executive Summary

Thirty years ago, health care technologists realized a simple truth: monitoring patients improves outcomes. Today, clinicians have a multitude of monitoring systems for bedside and ambulatory care. They can monitor multiple vital signs (like bedside monitors) or have a special purpose (like Holter monitors). Monitors are used in operating and emergency rooms and in many types of units, such as intensive care, cardiac care, and others.

Over the years, the healthcare industry has continued to see advances in monitoring and a wider range of applications (such as monitoring infusion pumps for IV drug delivery).

While monitoring technology has continued to improve, these systems are still relatively siloed. Hospital error is the third-leading cause of death each year in the U.S. Many deaths are due to the lack of integration between medical devices monitoring a patient. Hospitals recognize the tremendous benefits associated with sharing information between systems.

By networking monitoring systems to more medical devices and healthcare subsystems, these siloed systems can share data and enable more rapid and safer response to changes in a patient's condition. For example:

- Anyone who has worked in a hospital knows that alarms are constantly sounding from multiple devices (or multiple vital sign sensors). Smart alarms only sound when multiple devices or

changes in a certain combination and threshold of vital signs indicate errant physiological parameters.

- By connecting monitoring systems to treatment plans, smart drug delivery systems can react to patient conditions in a predictive manner, helping to prevent problems like overdoses with Patient-controlled Analgesia (PCAs) devices.
- By tracking patients around the hospital and connecting them to cloud/backend resources, efficiency of care can be dramatically improved.

 **Challenges**	• Healthcare systems consist of many silo'd medical devices and applications which increases cost, inhibits patient outcomes and even impacts patient safety • Integrating healthcare systems is challenging and costly at best • An example use case for patient controlled analgesia devices illustrates the device integration challenge and patient safety issue • An example use case for large, integrated patient monitoring systems illustrates the cost and patient outcome challenges • Security, reliability, interoperability, scalability.
 **Solution**	• Fog computing uses a distributed computing approach to create smart, connected healthcare systems • Fog's support for a continuum of compute from edge to cloud supports the cost-effective integration of the example patient monitoring system • Fog-based deployment of edge analytics and performant control applications support the creation of smart patient controlled analgesia systems that reduce patient deaths, increase patient outcomes and reduce healthcare staff workload.
 **Technology**	• A virtual compute environment on a series of fog nodes supports the flexible deployment of applications and streamlines the integration of healthcare systems • An open, fog architecture for healthcare, OpenICE, provides a flexible fog compute ecosystem based on the DDS connectivity standard and IEEE 11073 data model • A layered, fog computing architecture with the OpenICE standard streamlines the integration of large healthcare systems

- Data-stream and data-item specific security model in OpenICE, supported by the DDS standard ensures patient data privacy and cyber-security.

Introduction

The following is an example of an actual implementation of fog computing for integrated medical systems. The purpose of presenting this use case is to promote more architectural conversations about fog computing use cases for the healthcare industry.

Interoperability

Researchers and device developers are making quick progress on medical device connectivity. The Integrated Clinical Environment (ICE, also known as ASTM F2761) standard is one key effort to build such connected systems. There are other efforts, based upon other standards and communications infrastructure. This is an example of one standard that is currently gaining popularity. ICE combines standards as follows:

- It takes data definitions and nomenclature from the IEEE 11073 (x73) standard for health informatics
- It specifies communication via the Data Distribution Service (DDS) standard
- It then defines control, data logging, and supervisory functionality to create a connected, intelligent substrate for smart clinical connected systems.

To provide interoperability, ICE specifies standards at multiple layers. For syntactic interoperability (data type and protocol level) ICE specifies the DDS standard. In addition, ICE uses the data flow parameter functionality of DDS to shape the information communication behavior. This is one aspect of semantic interoperability. For semantic interoperability, ICE uses the IEEE 11073 data model standard.

While the PCA scenario is relatively simple, networking medical devices in a clinical environment is quite challenging due to the various information communication behaviors required. The information flows need to include slow data updates with fast waveforms.

Delivery timing control is critical. Integration with data from the emergency medical room (EMR), for example, must provide patient parameters such as allergies and diagnoses. Appropriate monitor readings and treatment history must also be written to the EMR. Large hospitals must match data streams to patients, even as physical location and network transports change during transfers between rooms. Devices from many different manufacturers must be coordinated and their proprietary interfaces and data models bridged to the ICE data connectivity.

ICE leverages DDS to address data connectivity. DDS models the complex array of variables as a simple "global data space," easing device integration. Within the data space, the "data centric" model elevates programs to exchange the data itself rather than primitive messages. DDS is used to sift through 1,000 beds and 100k devices to find the right patient, despite moves. DDS also addresses the need for fast and real-time data communications. These data communications include heart waveforms, image data, and time-critical emergency alerts, so the system uses DDS's quality-of-service parameters to tune the different data flows.

In addition, ICE needs to provide a common data model – the other important function for realizing semantic interoperability. For this, ICE specifies the data definitions and nomenclature from the IEEE 11073 (x73) standard for health informatics. The language ICE uses to capture the data model is OMG IDL, which is one data modeling language used by the DDS standard for data types.

For example, for the PCA scenario, an important message in the system is the command to halt the drug infusion pump when the pump control application detects patient distress from the integrated sensor values.

The data model is available on the MDPnP Github site. The following table shows the data type for the infusion pump command:

```
/**
        * Speculative topic used for the PCA demonstration. The supervisory
        safety app
        * publishes a sample with stopInfusion=1 to indicate the infusion pump
        may not
        * infuse. Currently a third topic, indicating that the pump has
        acknowledged
        * the safety interlock, has not yet been included. We should also explore
        the
        * possibility of a setup whereby the pump receives periodic 'ok to
        infuse'
        * information and stops when that information is not received.
        *
        * Stability: 1 – Experimental
        */
        struct InfusionObjective {
        UniqueDeviceIdentifier unique_device_identifier; //@key
        LongString requestor;
        boolean stopInfusion;
        }; //@top-level true //@Extensibility MUTABLE_EXTENSIBILITY
        #pragma keylist InfusionObjective unique_device_identifier
        const string InfusionObjectiveTopic = "InfusionObjective";
```

This table shows a summary of the ICE use case elements:

Devices
– PCA infusion pump, pulse oximeter, respiratory rate monitor (CO2)
Clinicians Involved
– Physician, Nurse, Nursing Assistant
Justification
– The use of two independent physiological measurements of respiratory function (oxygen saturation and respiratory rate) enables a smart monitor to optimize sensitivity to detecting respiratory compromise while reducing false alarms.
Comments
– Connectivity to multi-parameter monitor (to get SpO_2, CO_2, respiration rate) is an option for system configuration.

– If a respiratory CO_2 monitor (capnograph) is used to monitor respiratory rate, exhaled CO_2 levels could be incorporated in the respiratory depression algorithm. Sensor data is published at 1 0Hz.
– If a pulse oximeter is used to monitor pulse and blood oxygen concentration, both readings can be incorporated into the respiratory depression algorithm. Sensor data is generally published at 2 Hz.
– Use case value: *Improved Patient Safety*

Business Case

There are many reasons why hospitals need to do a better job of making systems smarter and integrating them to work together more effectively. Just one example included in this use case is patient-controlled analgesia (PCAs). These are programmable drug-delivery systems that dispense a dose of a narcotic pain reliever intravenously (the dose is pre-set by the clinician) when a patient pushes a button.

PCA overdoses kill 1-3 patients every day in the US (Source: "Challenges to Implement a Standards-Based ICE Platform, Tracy Rausch, Medical Device Interoperability Conference, 2013.) This seemingly simple system suffers from visitor interference and unexpected patient conditions.

But false alarm fatigue is a very real human factor in these overdoses. Integrating PCA monitoring with other monitoring devices—such as blood oxygen level monitors and respiratory rate monitors—can automatically halt the infusion pump to ensure the patient doesn't overdose.

Integrated Patient Monitoring Use Case

For the integrated patient monitoring application, we can start from the smart PCA application (covered below) and extend through the hospital and to some of the backend applications running elsewhere in the fog computing hierarchy.

Figure 2 shows one possible logical connection of the various applications that will run on various fog nodes at different layers from the edge to the backend. Interoperability through a data connectivity framework is key. In Figure 2, the clinical decision support (CDS) algorithms and applications may be scattered across multiple nodes as needed. The CDS system encompasses many of the analytics and intelligence capabilities for the PCA system and integrates that key information and data back to the higher-level applications/subsystems.

In addition, the "Other Devices" and "Other Infusions" in Figure 2 can encompass up to 20 additional patient monitoring and therapy devices attached to a patient in intensive care, for example. These devices all run considerable software for controlling the device and transforming raw, physiological data into parameters a healthcare professional can use in real-time.

Currently, these devices are all independent and do not communicate significantly with each other. Any integration of readings or status from the various devices is most likely done by a person watching the user interfaces.

In Figure 2, which shows a fog computing environment with interoperable communications between devices, device data is integrated and fused by various CDS applications scattered across the system. This information in turn is fed to various healthcare professionals where and when they need it. Alarms and alerts, if serious, can drive automatic responses in the system, like stopping the PCA infusion until a healthcare professional can check to ensure the patient is not in serious distress.

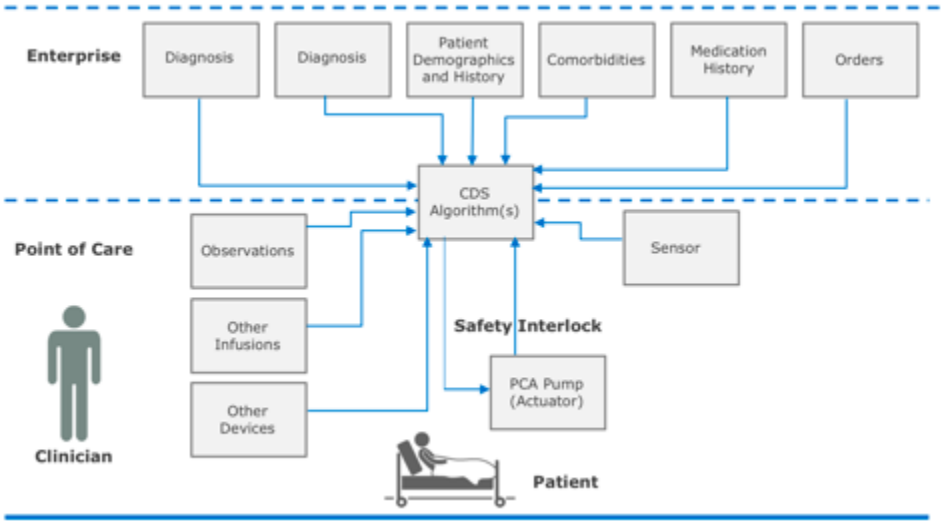

Figure 2. Schematic of an Integrated Clinical Environment (ICE). With an ICE-based fog deployment, a smart PCA can be integrated with backend systems to improve patient outcomes.

Modern hospitals use hundreds of types of devices for patient care and monitoring. These systems must work in a large hospital environment. As shown in Figure 3, integrating whole hospitals with thousands of devices across hundreds of patient rooms, presents scalability, performance, and data discovery challenges.

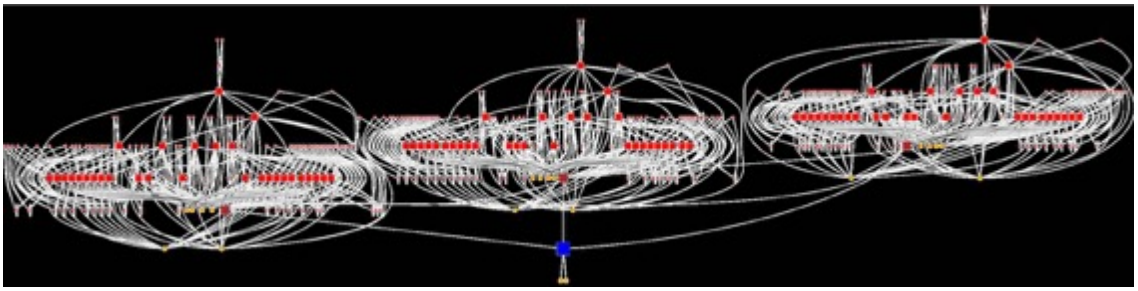

Figure 3. Medical devices must operate in a complex hospital environment. The system must be able to find data sources, track them as patients move, and scale to handle the load. This figure provides a graphical connection matrix that shows the scale and complexity of a multi-hospital, integrated patient monitoring system.

Since it is important to communicate real-time waveforms and video from devices and applications scattered across the system to various viewing locations, the potential network-wide dataflow is large. And since 62% of hospital patients move every day, the system needs to

identify data, devices and applications by patient – not by room, network location or compute node.

As part of the larger fog-based healthcare system, there will be many backend, real-time subsystems working with the information flowing from patients, devices and applications. For example, for each patient, summary information needs to be recorded to the patient's electronic healthcare record, while detailed monitoring and status data may be stored to a data of record storage system for later analytics or other uses.

Longer-term patient health analytic applications may retrieve data from the patient's data of record store and provide diagnosis assistance to the patient's doctor in a clinic across town. The data retrieved will include telemetry data from the patient's home health monitoring system devices.

The diagnosis will be passed along to a patient care planning application that the doctor uses to develop a course of treatment or care plan for the patient in the hospital. That care plan is then passed along to the healthcare team in the hospital and translated into schedules and protocols for the various medical devices (alarm levels, infusion levels, etc.).

The overall system needs to be highly performant. If there is an emergency situation for a particular patient, alarms, real-time monitoring data, healthcare team communications, and so on, all need to be readily available. The healthcare system also needs to be highly available in order to respond to this same emergency scenario.

The system has to be flexible in order to gather data from patients moving between home and hospital and from room to room. That patient data needs to be sent to clinical applications at various compute locations, and then provided to various viewing clients and data storage back ends.

Further, the applications, devices and data must be secure from cyber-threat and support privacy requirements of HIPAA. (Data is owned by the patient and must be managed as such).

Hackers that manage to break into these systems and reprogram devices like infusion pumps could easily injury or kill patients.

Communications must support standard IT networks found in hospitals and clinics, as well as home health networks, and work across WAN connections. Obviously, the system needs to be very scalable.

When fog is deployed, integrated clinical systems that span multiple hospitals, clinics and even patient home health systems, will streamline patient care. Fog will coordinate devices in each room, connect rooms into integrated whole hospitals, integrate a doctor's clinic, facilitate long-term healthcare planning and management, improve patient outcomes, decrease errors and reduce costs.

Information will flow easily and securely to various applications scattered across fog compute nodes throughout this system of systems. The healthcare system of the future will become intelligent, distributed and fundamentally enabled by fog computing.

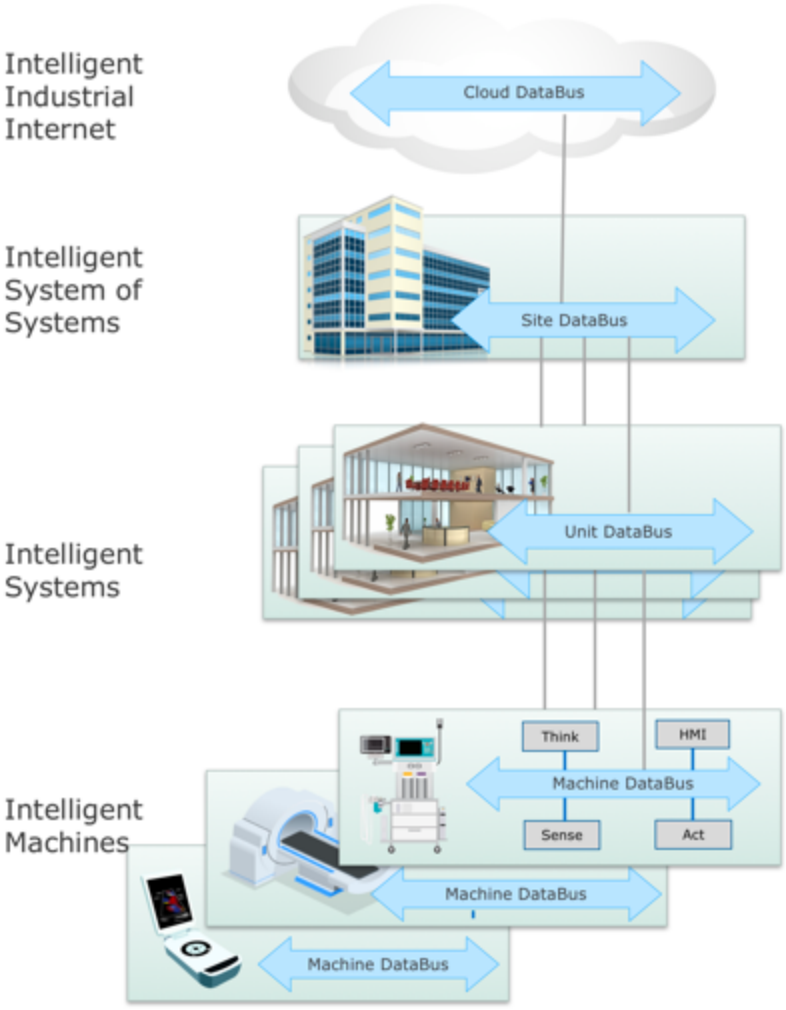

Figure 4. A layered fog computing approach to integrating healthcare system from the edge to the cloud or backend. Each layer interacts east-west as well and north-south with subsystems on different layers.

The key to the fog-based integrated patient monitoring solution is that it provides:

- A flexible architecture that enables easy management of the infrastructure
- A virtualized fog computing environment supporting various software applications
- Seamless interoperability between devices and applications, and
- The deployment of various applications.

In addition, the ability to scale across all the devices in a hospital room, across the hospital floor or department, throughout a hospital, and finally from a hospital group to the backend computing system, will drive significant patient outcome result improvements and cost savings for hospitals.

Smart PCA

After a procedure, many patients in the recovery unit are put on PCAs. The PCA system allows the patient to self-administer doses of painkiller medication by pressing a button. The idea is that a patient with sufficient pain medication will not press the button, and therefore be safe from overdose. Monitoring is not typically used due to high false/nuisance alarm rate.

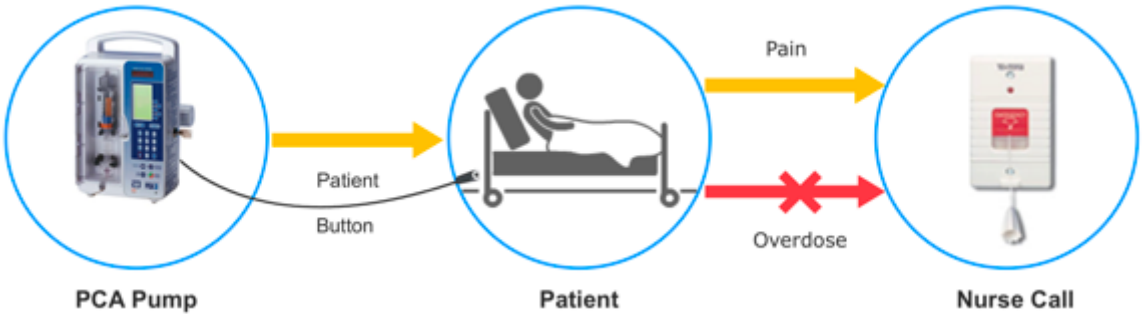

Figure 5. Patient Controlled Analgesia overview. The patient presses a button to receive intravenous pain medication.

Low oximeter readings cause many alarms. However, they are only likely real problems if accompanied by a low respiratory rate. A smart alarm that checks both oxygen (SPO_2) and carbon dioxide (CO_2) levels would eliminate many distracting false alarms.

To prevent overdoses, a fog computing-based solution automatically controls a PCA infusion pump based on correlated data that is coming from multiple medical devices. Features of the solution include the ability to stop and start the infusion pump automatically.

This solution depends on:

- Infrastructure-like data integration across the subsystems and applications involved
- A management system for monitoring system health and to provide lifecycle management of the devices and applications, and
- A virtualized fog environment for both security and ease of management.

Assumptions.

- To interoperate, medical devices produce and publish data to a data bus
- Supervisory applications subscribe to this data bus
- There is a common data model that is known to all publishing and subscribing applications.

Usage scenario.

1. While on the PCA infusion pump, a patient is monitored with a respiration rate monitor and a pulse oximeter
2. The physiological parameters move outside the pre-determined range
3. The infusion is stopped
4. Alarms are sent to notify the clinical staff
5. Clinical staff attends the patient
6. The patient becomes stable
7. The physiological parameters move inside the pre-determined range
8. The infusion is restarted.

Figure 6 illustrates how the fog computing enables a smart infusion pump (PCA) system. This logical architecture shows the various software applications, running on various fog nodes and the data they share over a data bus to coordinate with each other.

The fog nodes use virtualization to provide increased security, to simplify application lifecycle management, and to provide for scaling and flexibility as new applications are deployed or system improvements are needed. Security is required also to ensure HIPAA data privacy rules are met and audit trails are provided.

Data from the pulse oximeter (SPO_2) and the respiratory rate monitor (CO_2) are published to the data bus and subscribed to by the clinical decision support (CDS) application, which will include many algorithms and analytic elements.

These applications in Figure 6 all run on different fog computing nodes. Data rate requirements are flexible. Supervisory services may be located on fog computing nodes attached directly to the data bus, or they may execute on remote cloud/core compute nodes connected to the data bus via a bridge application.

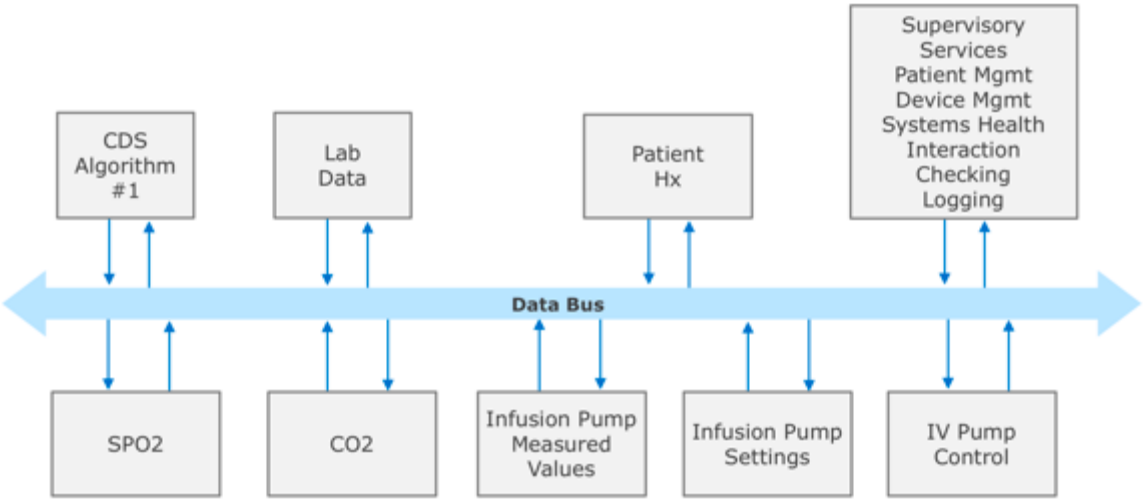

Figure 6. Logical architecture for a smart infusion pump application to support a safer, more effective PCA system.

This ICE-based fog computing architecture supports flexible application deployment to various potential deployment configurations. For example, the CDS application may be running on a server installed on the same floor as the patient's room while the sensor applications are running on small fog nodes next to the patient and the physical sensors. They also bridge the proprietary device data to the ICE data bus.

Or the CDS application could be running on a sensor device bridge fog nodes in the patient's room, depending on the particular hospital and their system architecture. Or there may be a CDS application running in both locations – with both subscribed to the same sensor data – but the server based application on the floor is running as a hot-swap backup to the primary application in the patient's room.

To support the flexibility required for the variety of deployment options, both application lifecycle and system management services are required, as is a data bus that decouples applications and their locations. In addition, real-time latency requirements for data communications between the applications scattered on different compute nodes in the patient room, floor server and backend applications must be provided for and managed.

Here are some key technical benefits of an ICE-enabled infusion pump (PCA) connected to the larger hospital IIoT system via fog.

- Devices are actively and automatically identified
- Devices and applications interoperate based on the data they need to share
- Data is collected, correlated and visualized
- Appropriate algorithms are selected and instantiated based on available devices and information being requested by healthcare team members and the patient care plan/protocol
- Supervisory algorithms and applications are selected and instantiated, based on available devices and the patient care plan/protocol
- Infusion is stopped based on advanced clinical criteria
- Alarms are activated based on advanced clinical criteria
- Event logs are created and data of record is stored
- Patients and care teams are discovered and connected; access control and privacy are enforced
- Data timing and reliability are ensured
- Software systems are repaired and updated in a compliant and secure way
- Devices and applications in the system are safe from cyber threats.

The Integrated Intensive Care Unit

Consider the picture in Figure 7. The devices in the picture are infusion pumps. Each administers a drug to a single patient. This picture is in an Intensive Care Unit (ICU); an operating room (OR) needs a similar array. In a heart surgery, for instance, the drugs sedate the patient, stop the heart, start the heart, and more. Of course, there are many more devices, including monitors, ventilators. During surgery, a trained anesthesiologist orchestrates delivery and monitors status. The team has their hands full.

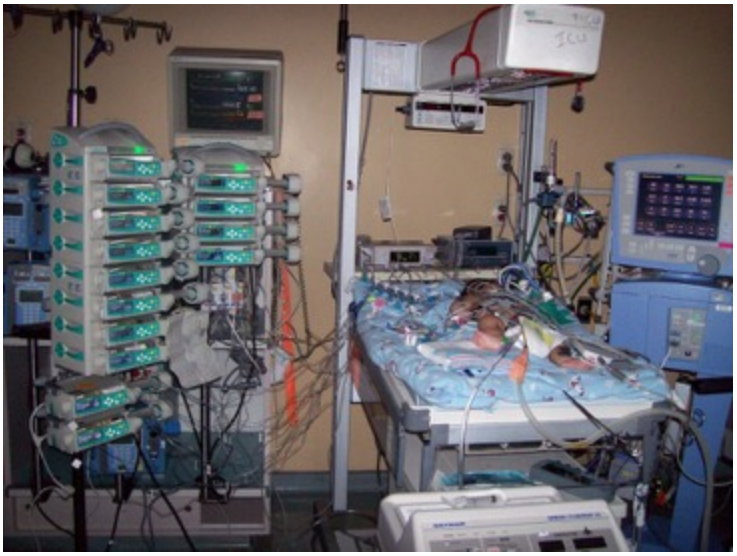

Figure 7. Medical Device Complexity Problem. These fourteen infusion pumps each administer a different drug to a single patient. They are currently completely independent from each other and the other devices and monitors.

After surgery, the patient must transfer to the Intensive Care Unit (ICU). This is a key risk moment. The drug delivery and monitor constellation must be copied from the operating room to the ICU. Today, the OR nurse calls the ICU on the phone and reads the prescription from a piece of paper. The ICU staff must then scramble to find and configure the correct equipment. The opportunity for small slips in transcription, coupled with the time criticality of the change, is fertile ground for deadly error.

Consider if instead these systems could work together in real time. In a fog computing scenario, the OR devices, working with a smart algorithm processor, could communicate the exact drug combinations to the Electronic Medical Record (EMR). The ICU system would check this data against its configuration. Paper and manual configuration produce far too many errors; the connected system eliminates dozens of opportunities for mistakes.

This is just one of scores of fog-based integration use cases for clinical settings.

Mapping to the 8 Pillars of OpenFog

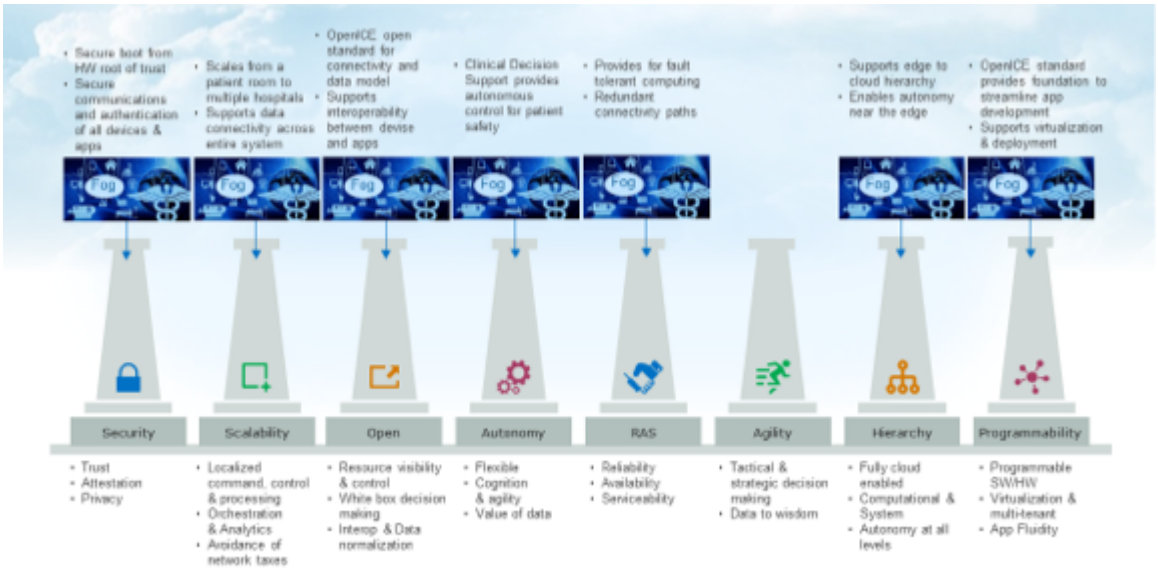

Figure 8. Mapping the use cases and system capabilities to the OpenFog 8 pillars.

Security. The OpenICE standard depends on hardware root of trust for ensuring authentication of the entire compute stack, including secure boot, device authentication and identity, compute stack and application authentication. Communication security is ensured through a fine-grained connectivity security model and implementation with DDS.

Scalability. Using a loosely coupled publish-subscribe connectivity solution ensures both interoperability between software apps and ease of integration of those apps. With this architecture, apps can be located and relocated much more easily, and can scale by connecting across all layers of the fog and cloud.

Open. The architecture solution used in these use cases must be based on open standards and an open architecture. In the OpenICE case, the overall architecture is an open standard, ASTM F2761, the data model is the IEEE standard 11073, and the connectivity framework is the open standard DDS. With this open architecture, device manufacturers, app developers and system integrators can easily add to the ecosystem, streamline system integration and ensure interoperability.

Autonomy. A critical element of these use cases is autonomous patient safety responses. If a patient is in imminent danger, the system should respond quickly by reverting to a pre-programmed "safe mode" for the patient while alerting clinical personnel to the emergency. Autonomy also is important for device and apps lifecycle management. Integrating a new device and the apps that support it in the fog compute layer above it should be as automatic as possible, thus depending on system management elements.

RAS. Fog-based healthcare systems are reliable and available to ensure patient safety and the continuous flow of monitoring data to all the needed locations across the edge-to-cloud system. As such, redundancy in apps and connectivity are required, and continuous health monitoring of the devices, compute nodes and apps is required. Excellent serviceability means any failures are found and repaired quickly

Hierarchy. For a simple use case like an integrated patient-controlled analgesia infusion pump, there are two layers of compute: the sensors and the pump are the edge devices, and the fog compute nodes for the monitoring apps and communications-to-backend systems are the next layer up. For the larger patient monitoring system, there is

significant hierarchy needed to integrate across hospital floors, entire hospitals, and to the backend data centers to support an entire hospital and clinic chain.

Programmability. There will eventually be hundreds of applications running on these large, integrated healthcare fog systems. As such, we need to provide for interoperability and ease of integration for these apps. An open architecture with open standards for data communications (including protocol, connectivity framework and data model) are required – and supplied in these use cases by the OpenICE architecture standard.

Mapping to the OpenFog Communications Architecture

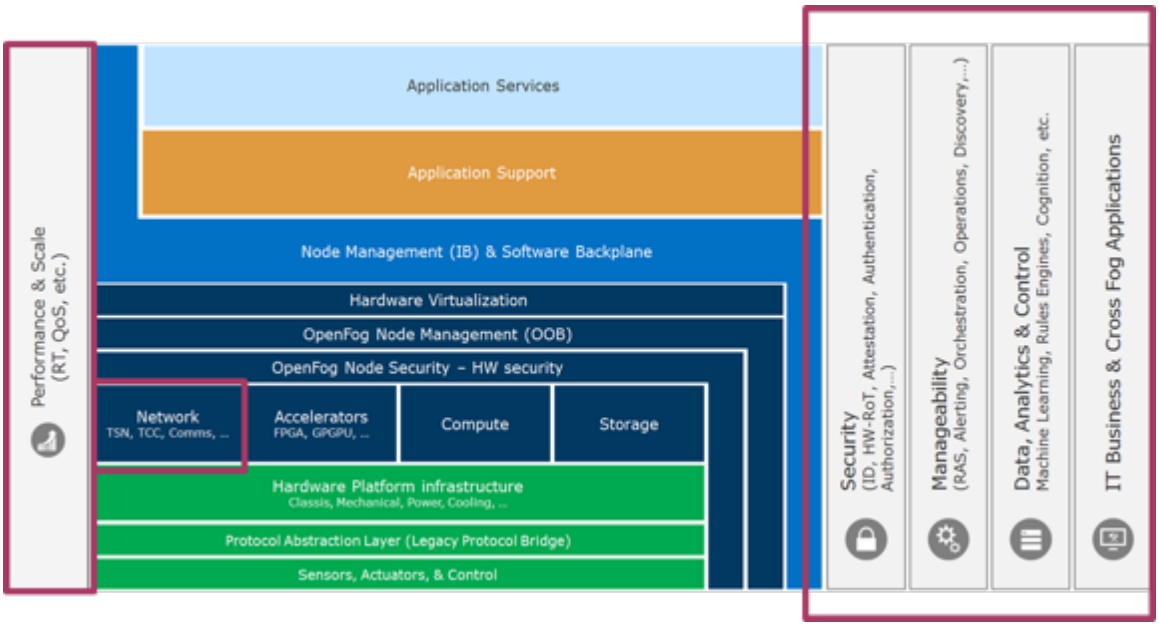

Figure 9. OpenFog communication for healthcare-based networking applications highlighted at the node level and across cross-cutting concerns.

Network

The physical and network layers of the fog-based healthcare system include a variety of protocols. Various standard Wi-Fi links are becoming increasingly prevalent in patient rooms. These will likely connect all the edge sensors and actuators together with the in-room compute nodes. Standard network switches and Ethernet links will

connect the various patient rooms back to each station in a clinic or on a hospital floor. Ethernet will again be the main protocol to link across the entire hospital and to on-prem data centers, if available.

Backhaul to cloud-based backend set of patient monitoring services, system management applications, asset tracking and certain "enterprise" systems will require various protocols and WAN links.

Performance & Scale

Low-latency performance is critical in patient rooms. Fog's connectivity framework assists in setting various QoS for the data communications needed between various elements in the patient room subsystem. QoS includes latency requirements (heartbeat intervals for data liveliness, latency limits for sensor values, app level acknowledgement for alarms, etc.).

In addition, the connectivity framework will scale across a hospital chain. It should be possible for a specialist in any hospital to pull up a patient monitoring dashboard and see live data for a patient anywhere across the hospital chain.

Security

Securing devices, compute stacks, apps and connectivity is necessary to protect the system and its various elements against cyber-attack. In addition, in healthcare there are patient data privacy requirements as set forth in HIPAA. This means the connectivity framework should manage the accessibility of the data streams from patients. Only certain authenticated healthcare practitioners should be allowed to view patient data based on need to know permissions. This applies to historical data as well.

Manageability

System management for these use cases is critical to ensure security, system health and ease of overall management (e.g., device provisioning, app lifecycle). There will need to be a trusted

connectivity framework level in place just for this "management plane," beyond the "application plane" supporting the actual patient monitoring.

Data, Analytics & Control, IT Business and Cross Fog Applications

Data analytics & control, IT business and cross-fog applications are key elements of the "application plane" for the fog computing system. In these use cases, these are represented by the patient monitoring functions.

To function effectively in a fog environment, these functions need to be linked together by a Connectivity Framework or Messaging Foundation into the overall application. All eight of the OpenFog pillars come into play in this cross-cutting function. The functionality for the overall system, on the application plane, is almost completely instantiated as multiple apps running on different compute nodes. It's a fundamentally distributed series of apps working together closely from the edge to the core/backend compute nodes.

As shown in Figure 10, secure, interoperable network connections below the IP Network Layer and data communications facilitated at the connectivity Framework layer are fundamental to the secure, interoperable connectivity required to enable distributed, fog computing for these healthcare systems.

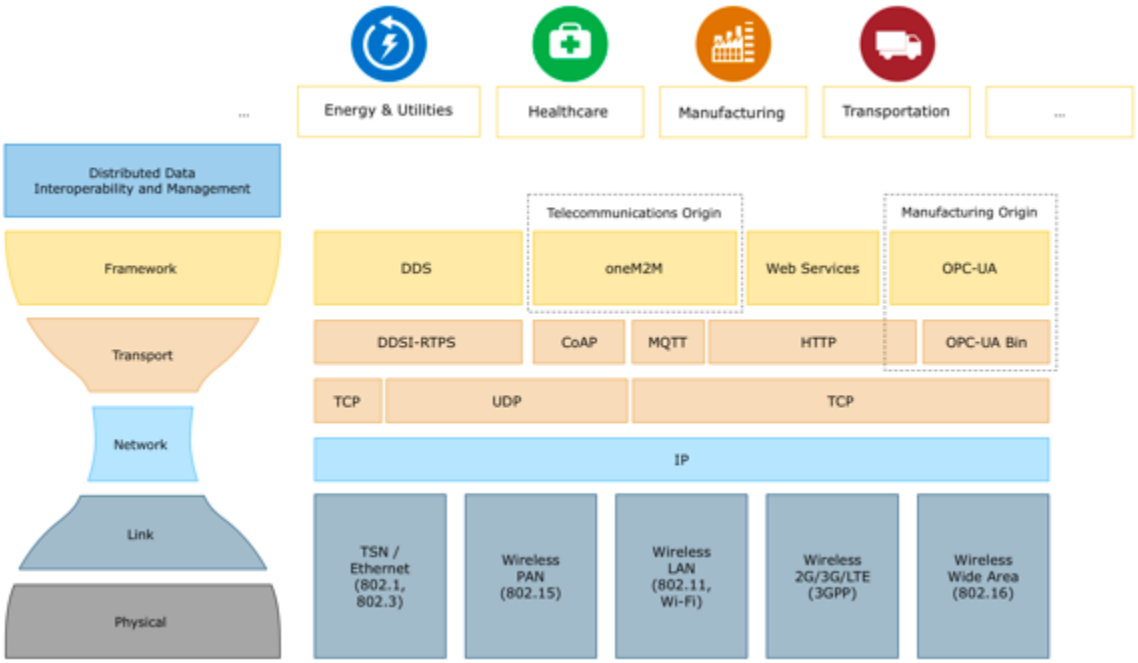

Figure 10. The Industrial Internet Consortium's IIoT network stack model with protocols and frameworks mapped.

Testbed Considerations

The hierarchy of OpenFog testbeds will be structured as follows:

1. Many small, research-oriented locations that OpenFog Members are able to access will focus on proving the high-level OpenFog architectural requirements and satisfying the minimum interoperability requirements via their Proof-Of-Technology(POT) Testbeds. The outcome of these Proof-Of-Technology testbeds could be open source code or a research publication available to OpenFog members.
2. Medium-sized, Interoperability Operation Model (IOM) testbeds will focus on overall solutions and end-to-end applications, with at least three OpenFog Sponsors participating to promote usage of diverse OpenFog Ready Solutions. They will demonstrate adherence to the OpenFog Reference Architecture and component-level interoperability and compatibility.
3. Large, regional testbeds will test pre-productization devices for application to the co-located OpenFog Certification Lab. After the

OpenFog Certification Lab validates a product, members will be able to release it as an OpenFog Certified product. We expect many verticals, use cases, and individual applications will have specific requirements for interoperability and preferences for certain types of testbeds, and the Consortium intends to adapt to their needs.

8 Adherence to the OpenFog Reference Architecture

The OpenFog Consortium intends to partner with standards development organizations and provide detailed requirements to facilitate a deeper level of interoperability. This will take time, as establishing new standards is a lengthy process. Prior to finalization of these detailed standards, the Consortium is laying the groundwork for component level interoperability and certification. Testbeds will prove the validity of the OpenFog Reference Architecture (RA) through adherence to the architectural principles.

9 Next Steps

The OpenFog Reference Architecture (RA) is the first step in creating industry standards for fog computing. It represents an industry commitment toward cooperative, open and interoperable fog systems to accelerate advanced deployments in smart cities, smart energy, smart transportation, smart healthcare, smart manufacturing and more. Its eight pillars imply requirements to every part of the fog supply chain: component manufacturers, system vendors, software providers, application developers.

Looking forward, the OpenFog Consortium will publish additional details and guidance on this architecture, specify APIs for key interfaces, and work with standards organizations such as IEEE on recommended standards. The OpenFog technical community is working on a suite of follow-on specifications, testbeds which prove the architecture, lists of requirements, and new use cases to enable component-level interoperability. Eventually, this work will lead to certification of interoperable elements and systems, based on compliance to the OpenFog RA.

For more information, please contact info@openfogconsortium.org.

10 About the OpenFog Consortium

The OpenFog Consortium was founded to accelerate the adoption of fog computing and address bandwidth, latency and communications challenges associated with IoT, 5G and AI applications. Committed to creating open technologies, its mission is to create and validate a framework for secure and efficient information processing between clouds, endpoints, and services. OpenFog was founded in November 2015 and today represents the leading researchers and innovators in fog computing.

For more information, visit http://www.openfogconsortium.org/; Twitter @openfog; and LinkedIn /company/openfog-consortium.

11 Authors and Contributors List

Authors	Contributors
Brett Murphy, RTI	Chuck Byers, Cisco Systems
Hamed Soroush, RTI	Evan Birkhead, OpenFog Consortium
David Niewolny, RTI	Judith Kelley, OpenFog Consortium
Paul Pazandak, RTI	

Note: All publicly available use cases are reviewed and approved by the OpenFog Technical Committee.

12 Copyright / Disclaimer

This reference document is designed to provide a foundation for extracting requirements when developing fog-based architectures. It is a compendium document to the OpenFog Reference Architecture. https://www.openfogconsortium.org/ra/