A Reference Guide to the Internet of Things

A Collaboration by

Contributors:

Greg Dunko
Joydeep Misra
Josh Robertson
Tom Snyder

2017

A Reference Guide to the Internet of Things

Published by Bridgera LLC, 500 West Peace Street, Raleigh, NC 27603.

Additional books may be available. Digital editions are also available. For more information, contact: *(919) 230-9951* or *info@bridgera.com*

Content Editors:
Kayla Little and Ron Pascuzzi
Cover & Interior Designer:
Arpita Kirtaniya

Contributors:
Greg Dunko, Joydeep Misra, Josh Robertson, Tom Snyder
Compositor:
Makana Dumlao

March 2017: First Edition

Revision History for the First Edition:
2017-03-01 First Release
See *https://bridgera.com/ebook* for release details.

Table of Contents

Preface

We set out to create a reference guide describing the technology needed for a complete IoT system. The idea was to gain an insight into why there might be doubt about the 4th industrial revolution. In a CES 2017 review, an author resorted to sarcasm when it came to IoT, "… after all, who needs their washing machine connected to the internet?"

Revolutionary technological advances do not happen without overcoming challenges. The printing press for example took two generations to impact society. Two basic challenges contributed to this delay. First, people did not know how to read. Second, there were not enough writers. For two generations, the focus was on teaching people to read and write.

Likewise, IoT has two significant challenges:

1) There are not enough IoT ideas with a clear return on investment.
2) Developing an IoT system requires a significant investment.

For IoT to realize its full potential, we must educate CIO's and CTO's on critical make or buy decisions to reduce the barrier to entry. Today, these decisions need a cross-disciplined team to navigate.

You do not have to go beyond the first page of Google to read about IoT ideas. Home automation, smart cities, and industrial applications are the early adopters. Lately the most press attention is arguably the most ambitious, autonomous vehicles. If we can reduce the IoT investment cost, we can bring more of these ideas to life.

This eBook is not about ideas but about the IoT technology stack. What makes the IoT complex, where are design trade-offs made, and where are make or buy decisions made.

The complexity of the IoT is real. As a result, we assembled a cross-disciplined team to write this book. I hope you find it insightful and useful. As fast as technology is changing, it should be safe to assume we will produce updates, so keep an eye

out for them. On behalf of Bridgera LLC, RIoT, and the authors of this book, we look forward to the IoT revolution in front of us.

Ron Pascuzzi
SVP Sales and Marketing
Bridgera LLC

Author Biographies

Tom Snyder is Executive Director of RIoT, operating under the Wireless Research Center of North Carolina, a 501(c)(3) non-profit. He is co-instructor in Product Innovation Lab, a Forbes award winning multi-disciplinary course in Innovation and Entrepreneurship at NC State University. Prior to joining RIoT, Tom held an executive leadership role at the ASSIST Center, a National Science Foundation sponsored effort to create wearable electronics for healthcare monitoring. Previously, he spent two decades in engineering product development and technology incubation.

Josh Robertson is an early product development expert based in the RTP region of North Carolina. He has been a part of the development team for IoT products in the connected home and wearable markets. He has also consulted as a subject matter expert on system development projects for the Defense Advanced Research Project Agencies (DARPA), Special Operations Command (SOCOM) and the Department of the Navy (DON). When he is not working, he enjoys spending time with his wife and daughters.

Greg Dunko is a seasoned telecommunications professional with 25+ years' experience in satellite and mobile communications. In addition to working in highly technical roles at NASA and Ericsson, Greg has held senior leadership positions at Sony Ericsson, HTC, and BlackBerry. He has taught STEM courses at the university, community college, and high school level, and led the ECE Senior Design program at North Carolina State University. In addition, he is an accomplished inventor with over 80 patents filed. In his spare time, Greg enjoys mentoring a high school robotics team and spending time with his wife and two sons.

Joydeep Misra currently leads the IoT initiatives at Bridgera LLC in Raleigh, NC. Joydeep has 15 years of experience with full stack software architecture and development of customer software applications for enterprises. He has held positions in the banking, insurance and logistics industries and is now responsible for managing the evolution of Bridgera's IoT platform. Joydeep's vision for IoT is to create a software platform robust enough to support large scale enterprise grade IoT solutions and at the same time, make it easy enough for the IoT hobbyist to

quickly connect their devices to the cloud. He is passionate about creating business value through the innovative use of emerging technologies. Apart from work, he loves reading, listening to music, and spending time with his wife and son.

Introduction

The Internet of Things (IoT) may be one of the least understood technology buzzwords of the last decade. Marketing terms such the "Internet of Everything" have added to the confusion.

IoT is the intersection of the digital with the physical. Physical devices are now enhanced with sensing, computing, and communication capabilities. These enhanced devices are communicating with the digital world, producing data and insights that were never before possible.

In this book, we will attempt to give you a clear understanding of what IoT is from a system point of view. Our goal is to help you prepare yourself and your company to take advantage of it.

Some areas of IoT are beginning to standardize, but in most cases, there are still many choices. We want to help you make good choices whether you're specifying a project, evaluating a platform, or selecting a solution provider.

We organized the book to help you understand the IoT stack. The IoT is a system of systems with each system having unique technology demands.

IoT Stack

The System of Systems

Section 1: Physical Things. The IoT capabilities of physical things include sensors, actuators, processing, control, and power. We will discuss design trade-offs considering cost, size, performance, and environmental constraints.

Section 2: Communication Channels. Many combinations of internet protocols and connectivity solutions enable Thing-to-Thing, Thing-to-Server, or Server-to-Server data transfer. These will vary depending on the design use case.

Section 3: Software. Software provides the ability to ingest, process, store, and analyze data that originates from a Thing. Software also provides application level capabilities for humans to visualize data and interact with the IoT system.

Section 4: Operations. The accessibility of Cloud infrastructure may have trivialized operations, but providing a DevOps function for a complex IoT system is far from trivial. Purposely-made operational trade-offs are critical to deploying and maintaining an IoT system. Cloud infrastructure combined with resources dedicated to development, operations, and security across the entire IoT stack, are a few of the areas that need to align with the IoT strategy.

Section 5: Data. Data is a byproduct of the IoT system. Without data, the IoT would serve little purpose. The book will conclude with a section on the most valuable asset produced by the IoT: Data.

Systems of Systems Create Massive Value

Consumers are most familiar with the IoT through the products that most affect their day-to-day life. Wearables such as Fitbit and connected home devices like Nest are most commonly cited examples of IoT. These are however quite basic early IoT systems.

IoT is much broader than these narrow examples. Like the initial advent of the internet that laid the foundation, IoT has the potential to affect nearly every aspect of our lives.

A key principle to understand is that IoT systems are systems of systems. Data from one system combined with data from other systems create value that you would not have with disconnected systems.

Soon, an apple you buy at the grocery store will likely have been grown on a smart orchard. This orchard will contain an IoT system that tracks and controls soil moisture and nutrients to increase yields. Sensors will communicate with servers that are communicating with weather sensing networks. The combination of the

two data streams allows the system to determine how, and predict when, to water and fertilize.

Orchard monitoring will integrate with a smart shipping supply chain system that will bring just enough trucks at exactly the right time to harvest. The supply chain system will connect to grocery store inventory systems, optimizing deliveries and shelf space based on geography, availability, and regional demand in real-time. Sensors and controls in the delivery trucks will optimize temperature and humidity ensuring the freshest produce possible. All this means fresher apples at a cheaper price.

Not only can IoT save money and time, it can also save lives. IoT solutions are becoming popular at the city and county level to make emergency response services faster and more effective. One smart home fire detector on the market now can issue adverse weather alerts to warn people in cases such as fast moving tornadoes or forest fires.

The basic wearable devices become far more useful when heart rate, activity data, and other signals from the wearables are connected to first responder systems as EMT's rush to an emergency. Connect these systems to the greater hospital network and patients are sent to the ideal facility and doctor for their condition. Tie in the transportation grid and every traffic light can turn green to speed an ambulance to the ER.

Looking at these examples it becomes easier to understand why IoT has been referred to as the fourth industrial revolution. Like any revolution, the change won't come easy. The economic opportunity is predicted to be larger than the impact of the internet itself; the winners will be the ones that engage now.

Section 1: Physical Things

The Internet of Things is rarely discussed without the conversation steering to data and the new Data Economy. The intelligence and value from IoT systems is based on what can be learned by studying data. The fundamental source of IoT data is sensors.

Driven by new innovations in materials and nanotechnology, sensor technology is developing at a never before seen pace, with a result of increased accuracy, decreased size and cost, and the ability to measure or detect things that weren't previously possible. In fact, sensing technology is developing so rapidly and becoming so advanced that we will see a trillion new sensors deployed annually within a few years.

Sensors and Actuators

A better term for a sensor is a transducer. A transducer is any physical device that converts one form of energy into another. So, in the case of a sensor, the transducer converts some physical phenomenon into an electrical impulse that can then be interpreted to determine a reading. A microphone is a sensor that takes vibrational energy (sound waves), and converts it to electrical energy in a useful way for other components in the system to correlate back to the original sound.

Another type of transducer that you will encounter in many IoT systems is an actuator. In simple terms, an actuator operates in the reverse direction of a sensor. It takes an electrical input and turns it into physical action. For instance, an electric motor, a hydraulic system, and a pneumatic system are all different types of actuators.

In typical IoT systems, a sensor may collect information and route to a control center where a decision is made and a corresponding command is sent back to an actuator in response to that sensed input. Later, we will discuss where the control center resides in the greater IoT system.

Sensor to **Actuator** Flow

There are many different types of sensors. Flow sensors, temperature sensors, voltage sensors, humidity sensors, and the list goes on. In addition, there are multiple ways to measure the same thing. For instance, airflow might be measured by using a small propeller like the one you would see on a weather station. Alternatively, as in a vehicle measuring the air through the engine, airflow is measured by heating a small element and measuring the rate at which the element is cooling.

Different applications call for different ways of measuring the same thing.

The Importance of Accurate Sensors

Imagine that you are a bar owner and you want to measure the amount of beer coming out of one of your taps. One way you might do this is to install a sensor in line with the line that runs from the keg of beer to the tap. This sensor would most likely have a small impeller inside of it. When the beer ran through the sensor, it would cause the impeller to spin, just like the propeller on a weather station.

When the impeller spins, it will send a stream of electrical impulses to a computer. The computer will interpret the impulses to determine how much beer is flowing through. Sounds simple, right?

This is where sensors get interesting. If you look back at our description, you'll see that we never directly measured the amount of beer flowing through the sensor; we interpreted it from a stream of electrical impulses. That means that we must first figure out how to interpret it. Calibration.

To calibrate the sensor, we'd have to take a container with a known carrying capacity, say, a pint glass. Then we'd have to fill that container under a variety of conditions to determine what the electrical pulse signal looked like.

For instance, the first pour off a new keg might tend to have more foam, which would read differently than a pour from the middle of the keg that was all beer. It's only through repeated trials and a lot of data that we gain confidence that we can interpret the data and determine how much beer was poured.

Once the correlation is well known, a protocol can be developed to always assure the sensor is reading correctly. This is called calibration. Reputable manufacturers will deliver fully calibrated devices and provide instruction on how to recalibrate to verify sensor accuracy.

The accuracy of sensed data is paramount, since you will make mission-critical decisions based on later analysis of the data, which will hold little value if the data is wrong.

The Control System

Your IoT device, may be smaller than a coin or larger than a refrigerator. It may perform a simple sensing function and send raw data back to a control center. It may combine data from many sensors, perform local data analysis, and then take action. Additionally, your device could be remote and standalone or be co-located within a larger system.

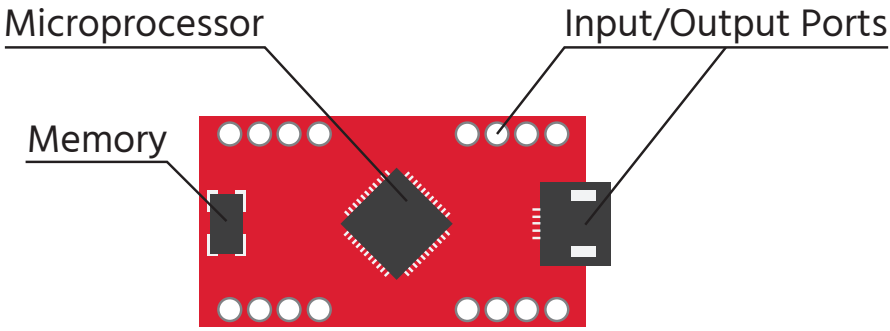
Regardless of the function, environment, or location, your IoT device requires two components, a brain and connectivity. The "brain" provides local control (or decision-making). Your device's function will determine the size and capabilities of the brain component. Connectivity is needed to communicate with external control. The environment and location of your device will determine how it connects.

The "Brain" of the IoT Device

Your IoT device will most likely use a microcontroller as its brain. Think of a microcontroller as a small computer with a microprocessor core, memory, and input/output (I/O) ports. The microprocessor core of your microcontroller is a central processing unit. It handles all the number crunching and local data manipulation and decision-making. The memory includes Read Only Memory (ROM) and Random Access Memory (RAM). ROM stores the microcontroller's software program. RAM stores and receives data while also supporting number crunching. The final microcontroller component, the I/O ports, may be either digital or analog. The

input ports collect data from sensors. While the outputs support any necessary actuation or local control in the IoT device.

Usually, microcontrollers control various devices or subsystems within embedded applications. By integrating the microprocessor, memory, and input/outputs, microcontrollers reduce cost and make development easier. This makes it more affordable and less complicated to control many IoT devices.



Simple Microcontroller

To determine which microcontroller to use for an IoT application, you first need to understand a few basic specifications:

- *Microprocessor Type ("Brain")*
  How fast your brain needs to think (clock speed) and how much information it can handle (data I/O bus size) determines the microprocessor type. Depending on your application, you may have a very simple microprocessor or a much faster, bigger, power hungry one. While you don't need to be an expert on microprocessors, you should have an idea of what you want it to be able to do.
- *Amount of Memory*

  ROM "stores" your application program in the microcontroller. The size of the program is a function of the complexity. RAM performs two functions: It reads and writes data for file storage, and it holds data waiting for processing by the microprocessor. An important distinction between ROM and RAM is volatility. ROM is non-volatile, meaning that the data remains stored with or without power. RAM is volatile, meaning it may lose data if it is without power. This is an important factor for IoT devices which may be prone to an interruption of the power supply.

- *Operating Voltage and Current*

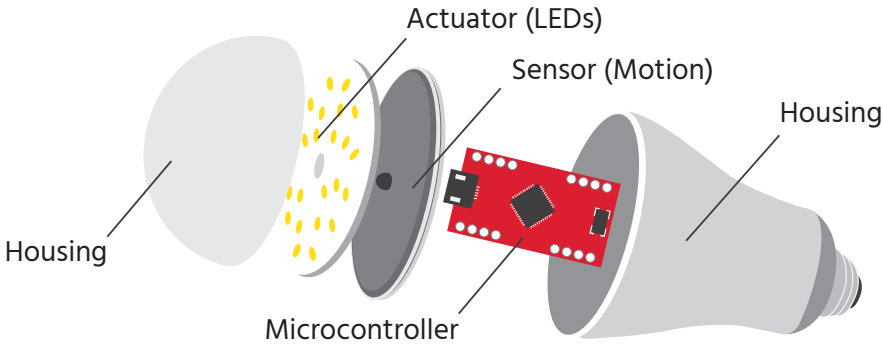
  The device needs adequate voltage to support its sensors and power sources. If a device is part of a larger system, it may use a local output voltage. Generally, sensor and battery technologies operate within a compatible voltage range. You will need higher output currents if you need control or activation (such as operating a motor).

- *Number and Type of Input/output (I/O) Ports*

  The I/O ports are the connection to your device sensors and actuators. There may be few, or many, depending on the application. The ports may act as inputs from sensor to device, as outputs from device to sensor (or actuator), or both. As mentioned above, I/O ports may be digital or analog. Digital ports are for simple logic, or yes/no type input. Examples include, closing a switch, tripping a wire, presenting/not presenting something. When acting as an output, a digital port may turn something on or off. Analog ports are for continuous input/output like a temperature or speed. Keep in mind; microcontrollers are digital devices so signals moving in and out must also be in digital format. Inputs from analog sensors use an analog-to-digital converter (ADC). The purpose of the ADC is to convert data into a format that is usable by the microcontroller. Once you define your IoT device application, you can determine the type and number of I/O ports you need.

- *Control Interface*

  A control interface is a protocol allowing peripheral devices and the microcontroller to communicate with one another. Depending on the application, you may need a specific control interface. There are different ways the interface may facilitate communication. It may use an inter-integrated circuit (I2C), a serial peripheral interface (SPI), or a controller area network (CAN) communications protocol. While you do not need to know technical details about these, you do need to know which protocols your system requires. Those will determine if you need a control interface and how that interface should communicate.



Anatomy of a **Thing**

Powering the IoT System

The size of an IoT device can range from tiny to very large. It might perform a single, simple function, or have complex on-board intelligence. It may transfer byte-sized data over short range radio-frequency identification (RFID). Or, it may transfer high bandwidth streams of data over long distance cellular. The IoT device may be in a fixed or varying location, and it may or may not be easy to access.

Despite the many differences, there is one common requirement for all device systems. They must have power (energy or electricity) to operate.

The Sources of Power

Three common options to power IoT Devices:

- *Mains Electricity*
  For many applications, you may only need to plug in the connected device. Home automation applications, such as connected light bulbs, can draw energy from existing wiring in the home. Industrial applications are often fixed-location devices, hardwired to a power grid.
- *Batteries*
  Found in everything from wearables to tools, batteries are a common solution for portable IoT devices. Your application will determine if it makes sense to use rechargeable or one-time use (primary) batteries.

  [Note: A capacitor is another energy storage and delivery device. It has different technical characteristics but, for now, we'll consider it the same as a battery.]

- *Harvesting*
  Several approaches for converting energy from ambient forms into electricity are becoming popular. Solar cells, converting light to electricity, are most common. It is also possible to convert air and fluid flow, heat, motion, RF and chemical energy to electricity.

Types of **Power** Sources

Note that across the entire IoT system, there may be more than one power solution. For example, wearable health monitors for factory workers may be battery powered, while mains power the central monitoring system.

Strategies for Extending Device Battery Life

A key design strategy for energy consumption is the "duty cycle". This is a technical term for how long devices are on and functioning, versus "sleeping" to conserve energy.

The 2016 Olympic Games taught us that nobody takes more than four steps per second. Usain Bolt's 100m gold medal was 41 steps in 9.81s. This indicates the accelerometer in an activity tracker (which can turn on and off quickly) doesn't need to take 1000 measurements per second to capture every step. Instead, 50 or fewer measurements per second are enough to capture a step at any point it might happen. The rest of the time the accelerometer can be in an off state (sleeping) to save power.

When working with your design team, ask them to develop a power budget. This is a table that shows the relative energy cost of each operation in the system. Everything from the microcontroller to sensors to wireless radios use energy. Each of these can also be designed to operate with a power optimized duty cycle to conserve energy.

Most often, radio transmission is the most expensive line item in the energy budget. Depending on the energy cost of transmission, it may make sense to add memory to your design. Doing so, you can capture and store sensor data locally to transmit in batches. Memory storage and retrieval functions are usually much lower power operations. Transmitting in batches, as opposed to real-time, allows the radio to be off most of the time, thus conserving energy.

Trading Features to Conserve Power

At a fundamental level, it is important to consider what features are necessary for your product. Some of the first activity trackers did not deploy full color screens, simply because they drained the battery. Putting each feature as a line item in your power budget allows you to cut out non-critical features until the product's energy life is sufficient.

Consider that batteries come at a cost. They will eventually be replaced or recharged. The activity tracker Fitbit knew users could not achieve 10,000 steps tethered to a wall outlet. However, it was reasonable to assume users would be willing to plug their device in to recharge the battery, as needed. Another option is to remove everything but the absolute, must-have features. Misfit took this approach to ensure an extremely long, primary battery life. (Primary batteries are non-chargeable, one-time use.)

Another cost to using batteries is size. Often, the battery is the largest, heaviest component in the device. It would not be worthwhile to make a Fitbit larger to extend battery life. This would negatively influence wearability, a significant product feature.

If you want or need to use mains electricity that is not readily available, you will need to find another energy source. This was the case for the creators of the world's largest wearable: The elephant collar. The Army Research Office and students at NC State University created this product for the safety of both elephants and humans. Tracking the animal helps to keep them away from human settlements where they could face inevitable death.

In this situation, replacing the battery was not an option because it conflicted with the goal to avoid human interaction with the animal. Mains electricity was not an option, considering there is no power grid in the Savannah. Therefore, solar energy harvesting was the best option to keep the collar charged. This technique keeps the device running, while also keeping the elephants away from areas occupied by humans.

In the medical space, wearables are forging an early path to implantable devices. Today, most pacemaker batteries require periodic surgery to be replaced. New options are being explored to make these devices sustainable for life without follow-up surgery. To keep the pacemaker charged, this approach would use piezoelectric materials that harvest energy from the motion of each heart contraction, as well as RF energy harvesting.

To summarize, developing a strategy for extending device battery life requires the following three steps. First, narrow down your list of features to only practical, must-haves. Second, closely assess your power budget. Third, weigh your must-have needs against your power budget. These steps will get you well on your way to fully energized, connected devices.

Section 2: Communication Channels

IoT systems enable Things to communicate with servers as well as other Things. As suggested by the name, the "Internet of Things", internet technology provides the foundation for this communication. An IoT system communicates by combining IoT specific internet protocols and a method for connectivity.

Internet Protocols

Existing internet protocols, such as Transmission Control Protocol/Internet Protocol (TCP/IP), are often too inefficient and power hungry to apply to emerging IoT applications. In this section, we will discuss some alternative internet protocols developed for use by IoT systems.

Message Queuing Telemetry Transport (MQTT) and Constrained Application Protocol (CoAP) are two alternative internet protocols. Both seek to improve IoT device operation while maintaining interoperation with the internet. The main difference between MQTT and CoAP is in their implementation. (We'll go more in-depth on MQTT and CoAP later)

Before going further, it is helpful to review existing internet layers and protocols. The internet is based on TCP/IP communications and operates using the 7 layer OSI model shown in the following chart.

Each layer performs a specific function. Together, the seven layers enable communication across different connectivity solutions, enabling a wide variety of applications.

- *Layer 1: Application Layer*
  In the application layer, the user inputs data and data is output to the user. This layer provides services for applications, such as email.
- *Layer 2: Presentation Layer*
  This layer converts incoming and outgoing data from one format to another for presentation.
- *Layer 3: Session Layer*
  This layer sets up and authenticates, coordinates, maintains, and terminates conversations. IoT sessions are very different from traditional internet sessions. Because of this, a lot of effort has gone into, and will continue

to go into, optimizing this layer for IoT applications. The session layer is where CoAP and MQTT usually operate.

- *Layer 4: Transport Layer*
  This layer provides communication session management support. This entails the packetization of data, delivery of the packets, and error checking once the data arrives. On the Internet, Transmission Control Protocol (TCP) and Universal Datagram Protocol (UDP) provide these services for most applications. TCP is used for most applications with human interactions with the Web (such as e-mail, Web browsing, gaming, etc.). It also supports acknowledgment, retransmission, and flow control. (Note: These capabilities increase overhead.) For some applications, TCP can be overkill, making UDP the preferred option. UDP is also better suited for real-time data applications such as voice and video. In these cases, it is unlikely you will need to retransmit a voice or video packet. Additional trade offs are discussed below.
- *Layer 5: Network Layer*
  This layer handles the addressing and routing of the data. This is where the Internet Protocol (IP) operates, and is where the IP address originates. The traditional internet operates on IPversion4 (IPv4), which uses 32-bit addresses. However, it is starting to evolve toward IPversion6 (IPv6), which uses 128-bit addresses to accommodate substantially more internet addresses.
- *Layer 6: Data Link Layer*
  The data link layer provides a reliable link between two directly connected nodes. The data link layer is also responsible for detecting and fixing packet errors that may form on the physical layer.
- *Layer 7: Physical Layer*
  This layer conveys the bit stream through the network at the electrical, optical, or radio level.

	7 LAYER OSI MODEL	4 LAYER INTERNET MODEL
1	**Application Layer** Message format, Human-Machine Interface	**Application Layer**
2	**Presentation Layer** Coding, ensryption, compression.	
3	**Session Layer** Authentication, permissions, session control	
4	**Transport Layer** End-to-end error control	**Transport Layer**
5	**Network layer** Addressing, routing, switching	**Network Layer**
6	**Data Link Layer** Error detection, flow control plysical link, access	**Link Layer**
7	**Physical layer** Bit stream, communication format, topology	

At times, you may see groupings of some layers, presented as a 4-layer model. The physical and data link layers combine to form a link layer. Session, presentation, and application combine to form an application layer. Neither approach is more correct that the other. However, we will reference the 4-layer model going forward.

Why Isn't the Existing Internet Protocol "Good Enough?"

There are a few reasons the traditional internet design is not sufficient. For starters, it cannot accommodate the major power constraints of some IoT devices. The connection may not always be stable or may have limited memory. Finally, not all IoT applications require guaranteed data delivery, data protection, or message acknowledgment. These capabilities may be beneficial, but the data overhead and power cost may not be worth it.

For example, take the overhead ratio of a standard TCP/IP datagram. (Overhead Ratio% = Overhead/Total Bytes). If considering an IP Datagram (over IPv6) of 1280 bytes. TCP overhead is 20 bytes. The overhead ratio would be 1.6% (20/1280). For CoAP, overhead is at least 4 bytes, resulting in an overhead ratio of 0.3% (4/1280). Not much of an improvement!

Now consider an IoT application. The data could include device temperature (4 bytes), GPS position (16 bytes), and time (10 bytes). Byte consumption amount totals 30 bytes, plus overhead. In this case, the TCP overhead ratio would be 40% (20/50) and the CoAP overhead ratio would be 11.8% (4/34). Use of this data format is almost 4 times more efficient, resulting in power savings. Of course, there are benefits of TCP such as sequencing and acknowledgment. However, the application may not need these capabilities or these challenges could be addressed higher up in the stack.

IoT Friendly Internet Protocols

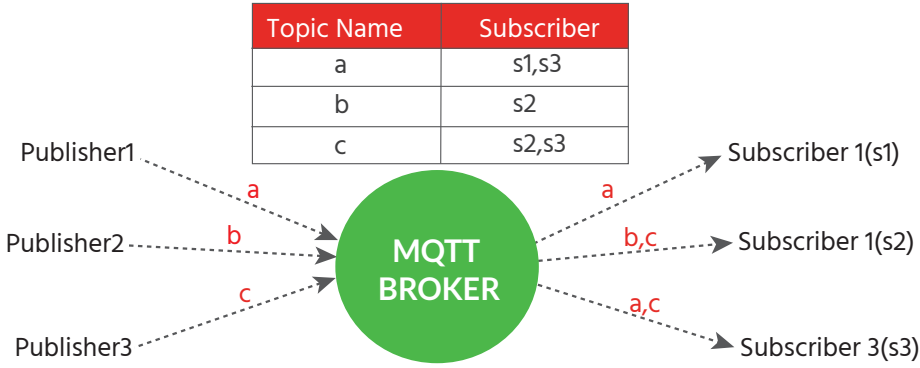
MQTT and CoAP. While there are other IoT friendly protocols in development and in use, these two are the most common. MQTT is an older standard. It was originally used in satellite applications but has evolved to handle today's range of IoT applications. CoAP, on the other hand, is somewhat new and has been gaining traction. Both are lightweight protocols that meet the requirements of IoT devices and applications: low cost, limited memory, low power operation, scalable with end node addressability in unique applications, and security support.

MQTT Overview

IBM invented MQTT for use in satellite communications with oil field equipment. At its core, MQTT is reliable and requires little power. Thus, it makes sense to use

this protocol in IoT networks. The MQTT standard is an open standard that has evolved to address a broader set of IoT applications.

MQTT uses a "publish/subscribe" message transport model. This model is lightweight and ideal for connecting small devices to constrained networks. A many-to-many protocol requires a central MQTT broker to receive, manage, and route messages among its nodes. The broker is also responsible for the authentication and authorization of clients. The MQTT "publish/subscribe" model scales well and can be power efficient. Brokers and nodes publish information, and others subscribe per the message content, type, or subject. Generally, the broker subscribes to all messages and then manages information flow to its nodes.



Publish/Subscribe Model

MQTT's publish/subscribe model offers several specific benefits outlined below:

- *Independent Operation*
  While a node and the broker need to have each other's IP address, nodes function independently of one another. Without any knowledge of each other, nodes can publish information and subscribe to other nodes' published information. Reason being: Everything goes through the broker.
- *Time and Synchronization Independence*
  A node can publish its information independent of a target node being active or not. The target node then can receive the published information from the broker once active. This allows nodes to self-manage low power sleep states, while still supporting interoperation.
- *Security Flexibility*
  MQTT uses Transmission Control Protocol (TCP) for its transport layer, enabling certain TCP security capabilities. The protocol can leverage

TLS/SSL security for encrypting traffic, but this can be resource-intensive. Another option is to encrypt only the packet payload (data in transit). This is a better approach for networks that consider power management a higher priority than security.

- *Varied levels of Guaranteed Delivery*
  MQTT offers three quality of service (QoS) levels. QoS is an important feature of MQTT because it guarantees message delivery, regardless of how unreliable the network is. The minimal level is QoS 0, often referred to as "fire and forget." This level makes a single effort, or transmit burst, with no guarantee of message arrival. QoS 1 attempts to guarantee the message is received, at least once. Finally, the slowest and safest level is QoS 2. This level guarantees the message is received, only once, and is also decoded.
- *Last Will and Testament (LWT)*
  MQTT also sends a "last will and testament (LWT)" message. The broker then stores the LWT message in the event a node unexpectedly disconnects from the network. If the node disappears, the broker notifies all subscribers of the node's LWT. And if the node returns, the broker notifies it of its prior state. This a beneficial feature when on unreliable networks.

The MQTT protocol offers great benefits for IoT applications, but there are some negative trade offs to consider. These include:

- *Need for a Central Broker*
  At the core of any publish/subscribe protocol is a central broker. This can be a drawback for some IoT applications. Being the central hub for every message, it is important for the broker to be scalable, open to integration, and failure-resistant. This can require overhead in the form of additional resources, software, and complexity that are not core to the end-node function. However, if the appropriate measures are not taken, the broker can make an entire network vulnerable.
- *TCP Protocol Implications*
  As previously mentioned, MQTT uses TCP for its transport layer. This provides for a reliable, ordered, and error-checked solution. However, these benefits come with additional overhead which may or may not be a worthwhile.

Recently, a new version of MQTT was designed to address some of these challenges. MQTT for Sensor Networks (MQTT-SN) operates using the user datagram protocol (UDP) in place of TCP. This provides broker support for indexing topics

(topic ID versus topic name). Because this protocol is still under definition, it is not well-supported.

CoAP Overview

With the growing importance of the IoT, the Internet Engineering Task Force (IETF) defined CoAP as an internet friendly protocol for "use with constrained nodes and constrained networks[1]." Like MQTT, CoAP is commercially supported and growing rapidly among IoT providers.

CoAP is a client/server protocol and provides a one-to-one "request/report" interaction model. The IETF has specified CoAP from the outset to support IoT with lightweight messaging. CoAP is a RESTful protocol designed to be interoperable with HTTP and the RESTful Web through simple proxies, making it natively compatible to the Internet.

The CoAP approach offers several specific benefits outlined below:

- *Use of the User Datagram Protocol (UDP) for Transport*
  CoAP runs over UDP, which uses a simpler approach to TCP with less overhead. It is inherently and intentionally less reliable than TCP. Instead of maintaining persistent connections, it depends upon repetitive messaging for reliability. UDP's connectionless datagrams enable faster wake-up and transmit cycles, as well as smaller packets with less overhead. This allows IoT devices to remain in a sleep state for longer periods, thus conserving power.
- *Possibility of Multicast Support*
  CoAP is inherently a one-to-one protocol. However, built on IPv6 addressing, it also supports one-to-many, or many-to-many multi-cast requirements.
- *Security*
  CoAP uses Datagram Transport Layer Security (DTLS) on top of its UDP transport protocol. Like TCP, UDP is unencrypted, but can be augmented with DTLS where needed. CoAP leverages DTLS, but since DTLS works over datagrams, data can be lost, duplicated, or received in the wrong order. Therefore, some end-to-end aspects of securing the data are not available.

1 http://coap.technology/

- *Resource/Service Discovery*
  CoAP uses a Uniform Resource Identifier (URI) to share capability and interaction expectations with network nodes. This allows a degree of autonomy in the message packets, since the target node's capabilities are partly understood by its URI details. A mapping between CoAP and HTTP is also defined. Enabling the build of proxies provides access to CoAP resources in a uniform way through HTTP.
- *Asynchronous Communications*
  The CoAP protocol uses the request/report model to exchange messages asynchronously. CoAP has a simplified "observe" mechanism (like MQTT's publisher/subscribe), that enables nodes to observe others without actively engaging them. [Note: CoAP does not have a broker requirement, therefore it cannot hold or queue messages for observers.]
- *Web Friendly Protocol*
  CoAP has many similarities to HTTP. For users with a web development background, this may make developing with CoAP relatively easy.

Like MQTT, the benefits CoAP offers IoT applications also come with negative trade offs:

- *CoAP Standard Maturity*
  Although CoAP is gaining market momentum, the protocol is less mature and less stable than MQTT. For now, designers should acknowledge that the standard is evolving, which could present interoperability challenges down the road.
- *Message Reliability*
  CoAP uses a simple approach for "confirmable" messages and "non-confirmable" messages. Like MQTT, the non-confirmable messages use a "fire and forget" approach. The confirmable message receives an acknowledgment message (ACK) from the intended recipient. This confirms the message got to where it needed to go, but it does not confirm the correct decoding of its contents.
- *Dependence upon Application Layer*
  Unlike MQTT which uses TCP for its transport, CoAP uses UDP. This requires the protocol's application layer to take on tasks usually handled by TCP. These include packet delivery, data ordering, caching, and retransmission. This increases your dependence on the application layer.

Protocol Summary

MQTT and CoAP are becoming the leading, lightweight messaging protocols for the IoT market. Each protocol offers unique benefits with some fundamental differences. Both protocols are used for mesh-networking applications, in which lightweight end nodes are a necessary aspect of almost every network. Both also act as a gateway for bridging logic to allow inter-standard communication.

Comparison of MQTT and CoAP protocals	
MQTT	CoAP
Many-to Many communication protocol	One-to-One communication protocol
Synchronous or Asynchronous*	Asynchronous only*
Medium Level Security**	Medium Level Security**
More reliable message delivery than CoAP	Less reliable message delivery than MQTT
Scalability is easier due to pub/ sub model	Scalability is more Complex
Simpler protocol spec may allow for simpler implementation	Easy translation to HTPP for simple web integration
No specified layout for data representation in a message	Provides inherent support for content negotiation and dynamic discovery
Does not support multicast operation	Supports multicast operation
Many software language content libraries	Many software language content libraries

* MQTT is most commonly run in Asynchronous only

** For higher-level security, other protocols such as Data Distribution Service (DDS) or Extensible Messageing and Presence Protocol (XMPP) should be considered.

Link Layer Improvements

MQTT and CoAP make improvements primarily in the Application layer. However, efforts are being made to offer improvements for IoT at the link layer.

IPv6 over Low-Power Wireless Personal Area Networks" – 6LoWPAN is a networking technology or adaptation layer that allows IPv6 packets to move within small link layer frames, such as those defined by IEEE 802.15.4. We will discuss more about 802.15.4 in the Short Range IoT Radio Solutions section below.

6LoWPAN is a link layer protocol that defines encapsulation and header compression mechanisms, as well as fragmentation and reassembly mechanisms that provide for adaptation between standard IP data packets (1280 bytes) and the IoT optimized packets of 802.15.4 (127 bytes). This allows packets sent from the Internet to be seamlessly transmitted across an 802.15.4 link and vice versa. The standard has the freedom of frequency band and physical layer, and is usable across multiple communication platforms.

Note: 6LoWPAN was originally conceived to support IEEE 802.15.4 low-power wireless networks in the 2.4-GHz band. However, it is adapting for use over a variety of frequencies and radio protocol solutions.

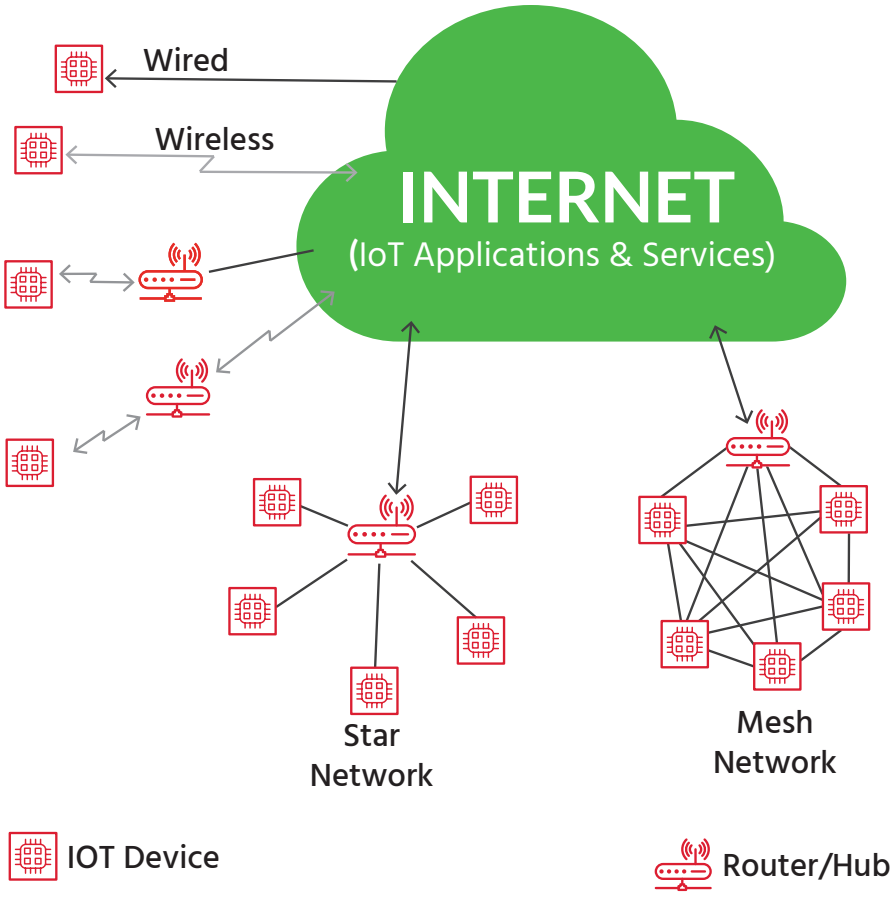
Connectivity Solutions

An internet protocol relies on connectivity to transport data between nodes. Connectivity can be thing-to-thing, thing-to-server, or server-to-server. There are several node architecture options. They can range from simple, point-to-point solutions, to complex, sophisticated solutions.

Connectivity options for IoT systems include:

- *Wired Direct*
  The IoT device is physically wired into the internet (e.g. Ethernet)
- *Wireless Direct*
  The IoT device wirelessly communicates with the internet (e.g. a Wi-Fi or cellular connection)

- *Wireless Local Connection, Wired Internet Connection*
  The IoT device communicates wirelessly with a local system using a short range, local connectivity solution. The local system, itself, is physically wired into the internet (e.g. via ethernet).
- *Wireless Local Connection, Wireless Internet Connection*
  The IoT device communicates wirelessly with a local system using a short range, local connectivity solution. The local system, itself, connects wirelessly to the internet.
- *Star Network*
  A star network has a central node managing connections with many other nodes. The central node is the hub, or access point (AP). The AP connects to the internet, providing the connection for all other nodes. Nodes do not talk to each other unless the AP forwards the message.
- *Mesh Network*
  A mesh network allows nodes within a specified range to talk to one another. Using multiple paths from the source node to the destination node, creates a robust network. However, the routing can get complicated depending on the requirements of the system. Typically, the internet connects to only one node in this network type.



Types of **Node Architectures**

Wireless Radio Technology

What is the best wireless technology for IoT applications? The answer is, there isn't one. There are numerous candidates for wireless technology. Which candidate is best is dependent on the design constraints of the IoT system. Below is a helpful chart to organize your constraints as you read through the IoT wireless solutions:

Constraint	Description
Environment	The operating location of deployed devices. A primary input to design. Input to power, communication range, and serviceability constraints.
Size	Device enclosure size. If constrained, may introduce constraints on antenna size or power supply.
Cost	Each sub-system in the IoT system should have a cost target. A primary input often initially ignored but it will introduce constraints everywhere.
Data	The amount and frequency of data to be captured and sent (e.g. bps) as well as the life span of that data (how long will it persist in each storage location). A primary input but often constrained by environment, size, or cost. Establishes storage requirements on-board a device and remote in the Cloud as well as bandwidth requirements between nodes.
Serviceability	Each system will have a finite life or a service requirement. Typically constrained by environment and cost, may introduce constraints on power. Also, drives standard or proprietary technology preferences.
Power	If main power is unavailable, power becomes a significant design consideration with dependencies on size, environment, cost, data, serviceability, and compute.
On-board Processing	Requirements for on-board vs. remote processing power, logic and storage capabilities. Constrained by size, cost, and power.

Simplex/ Duplex	The mode of operation is dictated by Data requirements, simplex (one way transmission, send data) or duplex (two-way transmission, send and receive).
Security	Like cost, a primary constraint that is too often ignored. Since these devices are connected into larger internet systems, they can become a backdoor for security hacking, etc.

Examples:

Your device transmits mass amounts of data frequently: You will need a high bandwidth solution.

Your device is size constrained: Your device's small size will force you to use a solution that can accommodate a smaller antenna and battery.

Your device must transmit data over a long distance: Your radio solution will need to operate at a lower data rate, use a lower frequency, implement a larger antenna, or increase your power capacity.

Your device must operate for days, weeks, or months without a re-charge of power: You will need to limit the range, reduce the data amount and frequency, and or invest in a more expensive power technology such as high density batteries or energy harvesting.

Frequency Bands

The wireless technology in your IoT device will likely operate at a radio frequency between 100 MHz and 5.8 GHz. In general, higher frequencies offer more channels and more bandwidth, requiring a smaller antenna. Lower frequencies propagate over greater distances and can achieve better building penetration. You can categorize frequency bands into different groupings, such as:

- *Unlicensed Bands*
  Radio spectrum is allocated by the government. In the US, there are frequency bands allocated for Industrial, Scientific, and Medical (ISM) applications, which are "open" for new application development. Popular ISM bands in recent years are 433 MHz, 868 MHz, 915 MHz, and 2.4 GHz. Some common uses of these bands have been for remote controls, cordless phones and Wi-Fi. Note: With the exception of the 2.4 GHZ band, many of these bands vary from country to country. Because the 2.4 GHz band is for unlicensed use in most regions, it has become popular for many commercial solutions (e.g. Bluetooth and Wi-Fi).
- *Licensed Bands*
  Licensed frequency bands require a license from a local regulator to operate a radio transmitter in a designated frequency channel. With these licenses, there is tighter control over who may transmit at these frequencies, reducing the chance of interference. TV and cellular communications are two familiar examples that use licensed frequency bands.
- *Forbidden Bands*
  These frequency bands are available for use by government agencies or public service organizations.

In terms of IoT applications, unlicensed ISM bands from 2.4 GHz up to 5 GHz have the best potential. There are also adjacent licensed frequency bands currently dominated by traditional cellular carriers. However, with the continuous rise of the IoT market, these will likely transition to IoT use in the future.

IoT Wireless Radio Solutions

Your application will dictate the connectivity method needed for your device. Connectivity range options include, short range solutions (e.g. RFID or Bluetooth), medium range solutions (e.g. ZigBee, Thread, or Wi-Fi), and long range Wide Area Networks (WAN) solutions (e.g. cellular or satellite).

Long Range IoT Radio Solutions

Many early IoT applications leveraged existing cellular networks for long range connectivity. The advantages to this approach include an established network with readily available development hardware and well-defined protocols and interfaces. Unfortunately, the following list of disadvantages associated with using cellular networks exceeds the advantages.

- Using cellular networks can be expensive.
- These networks are designed for voice and high data throughput/low latency communications, which are not typical requirements of IoT applications.
- The protocols and interfaces are difficult to customize for unique applications.
- The cellular device certification processes are time consuming and expensive.
- Cellular solutions are not equipped to handle the most difficult radio environments.
- Cellular solutions are not designed for very low power operation.
- With the push to increase data rates and high data spectrum efficiency, older 2G and 3G infrastructure will likely begin to phase out.

Instead of focusing on existing cellular technologies, we will introduce alternative long-range solutions.

Low Power Wide Area (LPWA) Networks

New LPWA networks are entering the IoT space to address some of the unique needs of IoT devices. For example, LPWA networks extend coverage into difficult radio environments. They also support very low power operation, thus enabling long battery life. Two of these emerging solutions are Licensed Spectrum LPWAs and Unlicensed Spectrum LPWAs.

Licensed Spectrum LPWA Solutions

These mobile, operator-managed LPWA IoT networks are based on 3rd Generation Partnership Project (3GPP) standards. They leverage existing mobile networks, as well as their licensed spectrum. Utilizing existing networks helps to retain some of the key advantages of cellular systems including strong security, excellent coverage and roaming capabilities, and assured quality of service (QoS) by operating in a licensed spectrum.

Recent changes in 3GPP standards are beneficial for IoT applications. Mass improvements have been made to increase power savings and extend battery life. Connectivity capabilities have also improved greatly. Connectivity is now capable of extending into subterranean locations and penetrating walls and floors.

The two most common licensed spectrum solutions are LTE-M (also referred to as LTE Cat-M1) and NB-IoT (Narrow-Band IoT). These networks efficiently connect devices to established mobile networks. They can handle small amounts of infrequent, 2-way data. They also consume minimal amounts of power and offer excellent radio coverage.

Like traditional cellular service, LTE-M and NB-IoT are Subscriber Identity Module (SIM)-based technologies. Thus, they deliver carrier grade, best-in-class security measures. These measures cover user identity confidentiality, entity authentication, data integrity, and mobile equipment identification.

Similar to cellular, LTE-M and NB-IoT devices are both highly mobile. LTE-M and NB-IoT devices can move seamlessly throughout a wireless network without interruption. They also incur fewer signal disruptions from technologies using the same frequencies.

LTE-M and NB-IoT are similar evolutions of 3GPP, with very similar pros and cons. Given the similarity and that details for LTE-M are well defined and readily available, we will focus here on NB-IoT only.

NB-IoT is a 3GPP standards-based technology developed to enable a wide range of new IoT devices and services on existing mobile networks. The 3GPP changes for NB-IoT improve power consumption, increase system capacity and spectrum efficiency, and improve coverage. In many cases, devices can achieve battery life of more than 10 years.

New signals and channels added to the physical layer help to meet demanding coverage and device complexity needs. Initial cost of the NB-IoT modules is comparable to GSM/GPRS. However, the underlying technology is much simpler

than today's GSM/GPRS devices. Because of this, we should see its cost decrease as demand increases.

All major mobile equipment, chip sets, and module manufacturers support NB-IoT. This allows the technology to co-exist with 2G, 3G, and 4G mobile networks. The first NB-IoT network trials are underway. We should see the first commercial launches take off early in 2017.

Advantages of NB-IoT:

- *Leverage Existing Networks*
  NB-IoT uses existing network infrastructure. This allows current systems to upgrade quickly, minimizing the need to deploy extra physical hardware. In addition, industries utilizing NB-IoT can rely on service providers to manage services so they do not have to do it themselves.
- *Open Standards*
  The open standard characteristic of NB-IoT reduces the risk of technology becoming redundant. It also helps to ensure that users of the technology are not "locked-in" to a specific vendor or operator.
- *Industry Support*
  NB-IoT is a product of existing 3GPP technologies. Thus, it will have long term support from a wide range of service providers, equipment providers, chipset and module makers.

Disadvantages of NB-IoT:

- *Royalties*
  One challenge with 3GPP standardization is the extra cost incurred by cellular solutions. This comes from royalties for intellectual property. At high volume, these royalties can add up.
- *System Synchronization Effects on Battery Life*
  Cellular networks operate synchronously. Therefore, individual devices must occasionally wake up, "check" for messages, and resynchronize. The synchronization process allows for low system latency. Unfortunately, this process also consumes a lot of energy, making it a major contributor to battery lifetime reduction.

Unlicensed Spectrum LPWA Solutions

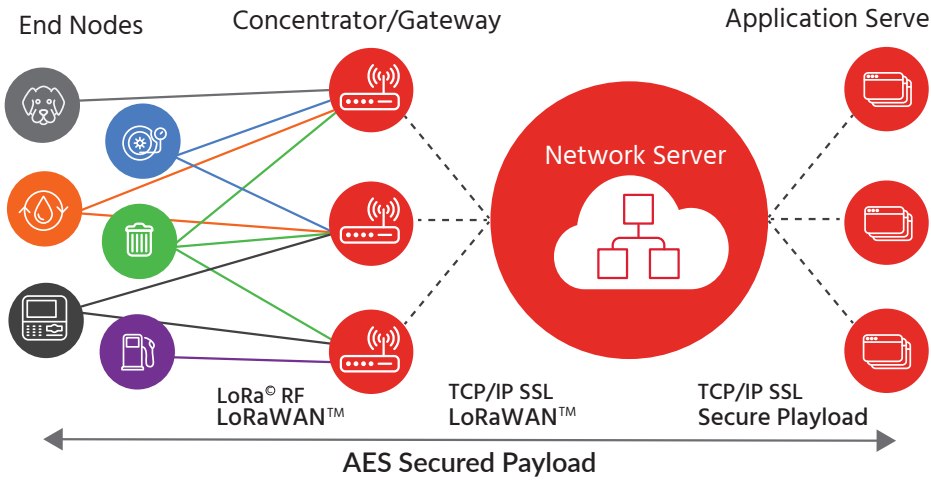
There are several LPWA solutions that are designed to operate on unlicensed frequency bands. Examples of unlicensed LPWA include LoRa, Sigfox, and Ingenu.

Some of these solutions are being standardized, whereas others are fully proprietary. Coverage in many cases is regional or limited.

LoRa

LoRa (short for Long Range) continues to gain attention in the marketplace. It offers a compelling mix of long range, low power consumption, deep indoor coverage, and secure data transmission. LoRa operates in the unlicensed <1GHz frequency range. It uses spread spectrum technology so that adjacent transmitters are less likely to interfere with each other. This increases the capacity of each gateway. Spread spectrum communications also provide a "coding gain" over narrow band communications. This results in a stronger communications link, which can support longer range communications. LoRa data rates range from 0.3 kbps to 50 kbps and can support a range of up to 15km.

The LoRa Alliance has developed an open protocol based on LoRa technology called LoRaWAN. This protocol helps in the delivery of secure communications. It also ensures all devices, servers, and software components work efficiently with one another. The result is simple interoperability between IoT devices. The network architecture has a star-of-stars topology (shown below). The gateways (the outer star network connecting to IoT sensor nodes) act as a bridge, relaying messages between edge-devices and the central network server.



Gateways connect to the network server via standard IP connections. End nodes/devices use single-hop wireless communication to one or many gateways. End-point communication is generally bi-directional. It also supports multicast operation for software upgrades or for other mass distribution messaging.

The network server handles all the intelligence and complexity associated with managing the network. For example, the server filters redundant packets, performs security checks, and schedules acknowledgments. A limitation of the network server is the seamless handover for mobile nodes. The network server does, however, manage device transitions between gateways.

Communication between IoT end-devices and gateways uses different frequency channels and data rates. When choosing the data rate, you face a trade off between communication range and message duration. The LoRaWAN network server uses an adaptive data rate (ADR) approach. This allows the server to manage the data rate and RF output for each end-device, individually. In doing so, battery life is preserved for both the end-devices and the overall network.

To enable secure communication, LoRaWAN uses several layers of encryption at the device, network, and application levels. To address application-specific needs, LoRaWAN offers options for end-point devices, balancing power consumption with return (down-link) communications timing/synchronization. LoRaWAN is also an asynchronous protocol, which is beneficial for improved battery life.

Despite an emerging presence in Europe, LoRaWAN has not gained much traction in the United States. The slow progression can be attributed to competition from new cellular options and Sigfox.

LoRa operates in unlicensed frequency bands. This helps reduce the cost of transmitter nodes (devices) because royalties are not involved. Unlike transmitter nodes, gateway receivers are very expensive. They support the connections to many transmitters. Because of this cost, LoRa may not be ideal for applications where price is key, like home automation. However, it could be a good solution for applications that are not price sensitive, like rural meter reading. Efforts are underway to reduce the cost of gateway receivers. Unfortunately, efforts to reduce price would also reduce functionality (likely through reduced capacity).

While the range of the LoRa system is attractive, there are some trade offs. To achieve the longest range, you must use very low data rates. For example, a 15km range uses a data rate of around 100-300 bps and the range drops quickly as data rates increase. The lower the data rate, the longer it takes to transmit data which, in turn, drains battery power. A solid LoRa design balances data volume and transmission speed with power consumption and range requirements. This system is best for applications that send only a few bytes of data, a few times per day. It is not ideal for systems that send large amounts of data, require guaranteed quality of service (QoS), or require low latency or tight synchronization.

LoRa Key Features:

- Operates in <1GHz band, with unique frequencies in different regions
- Uses spread spectrum for increased range and noise immunity
- Long range – up to 15 km
- Supports data rates up to 50 kbps
- High capacity of up to 1 million nodes
- Long battery life - over 10 years
- Reduced synchronization overhead and no hops in mesh network
- Secured and efficient network

Sigfox

Sigfox offers a proprietary, "complete LPWA solution" as a standalone operated telecommunication network. It uses a technology called Ultra Narrow Band (UNB) with binary phase-shift keying (BPSK) transmitted in the unlicensed bands below 1GHz.

Around since 2009, Sigfox is one of the older players in the IoT LPWA arena. Sigfox controls the back end, communications infrastructure and cloud management platform. Therefore, any customer who wants to use Sigfox must subscribe to both its communications infrastructure and its cloud platform. Radios and modules for endpoints are widely available from manufacturers like Texas Instruments, Atmel, and Telit, and it is likely that hardware costs will be reduced as this solution scales.

Sigfox offers a robust, power-efficient, and scalable network able to communicate with millions of battery-operated devices that are several kilometers apart. It is well-suited for low-bandwidth (<100bps) and low-frequency (< 140 messages/day) applications.

Designed for low throughput transmission, the UNB, wireless technology benefits from a high level of sensitivity. This enables a communication range of up to 40km in open field and communication with buried, underground equipment becomes possible.

Sigfox uses low data rate transmission and sophisticated signal processing to effectively avoid network interference, and ensures the integrity of the transmitted data. The solution allows bidirectional communication, but always initiated by the device. As such, Sigfox is effective for communications from endpoints to base stations (uploads). However, it is not effective for communications from base stations to endpoints (downloads). Sigfox consumes a fraction (1%) of the power

used through cellular communication. This network solution would be ideal for one-way systems including basic alarm systems, simple metering, and agricultural and environmental sensors.

In the Sigfox network, no negotiations take place between the device and a receiving station. Instead, the device emits in the available frequency band for the closest base station to detect. The base station will then decode the message and forward it to the control center. The network, itself, handles protocol operations and forwards messages to the customer application. A hash mechanism authenticates each message. The message then receives a private key, specific to the device, which is accessible using Sigfox's API.

An advantage of the sensitivity of UNB devices is network resource savings. By limiting the number of base stations, UNB allows Sigfox to deploy a low throughput communications network with approximately 1000 times fewer base stations when compared to the same level of coverage with a GSM cellular network. This reduces network deployment and operations cost. To further improve cost, Sigfox will setup base station antennas on existing cellular towers through partnerships with local mobile network operators.

Mobile operators have spent millions of dollars trying to develop a global network. To date, none have accomplished the feat. This is a reasonable concern for Sigfox. The inability to grow a global network would cause spotty regional coverage. This would present an issue for mobile or broadly deployed IoT devices. Another concern is relinquishing control. Some users may prefer more control over their data and do not want everything running through the Sigfox back end before making its way to them. Nevertheless, there are several undeniable benefits offered by Sigfox.

Sigfox Key Features:

- Operates in <1GHz band with unique frequencies in different regions
- Transmitters use Ultra Narrow Band (UNB) with binary phase-shift keying (BPSK)
- Long range – up to 40 km
- Very low data rates (on order of 100bps) and limited daily messages (max 140)
- Long battery life – up to 20 years
- Low cost network deployment
- Sigfox defined application APIs

Ingenu

Ingenu is another proprietary LPWA solution. It originally focused on smart meter and oil and gas applications. It has since expanded into other IoT applications including urban and agricultural environments.

The solution is proprietary in the sense that Ingenu is the sole developer and manufacturer of the hardware. In the past, Ingenu mainly sold hardware components to enterprises that built and controlled their own networks. However, the firm recently tweaked this business model and developed several public networks. It now also sells radio modules and recurring data subscriptions for these networks.

The Ingenu solution uses Random Phase Multiple Access (RPMA) technology. RPMA enables data rates in the hundreds of thousands of bits per second (50x other LPWA solutions). The system utilizes the 2.4 GHz unlicensed and universal frequency band. This offers larger bandwidth, greater flexibility, and reduces the chance of interference. Ingenu also uses channel coding (Viterbi algorithm) to guarantee data delivery and provide high quality of service (QoS). It is tightly synchronized to support low latency applications and uses 256-bit encryption and two-way authentication to provide enterprise level security.

These great features offered by Ingenu enable higher data throughput rates than Sigfox and LoRa. However, when combined, these benefits consume a significant amount of power. Operating in the 2.4 GHz band, Ingenu encounters more propagation loss from obstructions, like water or packed earth. Fortunately, the RPMA technology can counter this issue, providing excellent link margin. RPMA is one of the more complex radio technologies. It utilizes antenna diversity and requires much more processing power than other solutions. While this does enable applications not possible with LoRa or Sigfox, it also increases system complexity, power consumption, and hardware expenses.

Using RPMA, Ingenu claims to require fewer access points than cellular, LoRa, and Sigfox, while providing the same coverage area. The protocol also enables precise location tracking, which Sigfox and LoRa do not. Like LoRa, Ingenu is capable of effective bidirectional transmission. Overall, in weighing the pros and cons, Ingenu may be one of the best radio solution options.

Ingenu Key Features:

- Operates in 2.4 GHz band with global coverage
- Uses RPMA technology, requiring fewer access points than NB-IoT, LoRa, or Sigfox

- Long range with excellent coverage (one application covers 2,000 square miles with 17 towers)
- High data rates (>100k bps)
- Provide precise location tracking
- Highly scalable with support for up to 500k devices per tower
- Enterprise level security through 256-bit data encryption and two-way authentication
- Provides high QoS through channel coding
- Synchronized system provides support for low latency applications

Satellite

All LPWA networks require some terrestrial infrastructure for IoT device communication. In some cases, such as maritime applications or extremely remote areas, there may be no available gateways. Under these circumstances, satellite communications is an ideal alternative. Unlike the communication solutions we have discussed up to this point, satellite solutions can be very expensive. The higher cost is due to a low volume market and the expenses involved in satellite deployment. Satellite enabled devices require higher power and may also require a sizeable antenna.

Satellite communication technology has been added for completeness purposes. It may be one of the few viable options for some applications due to geography. Although it is an expensive solution, the cost may not be an issue for applications with infrequent data communication. Long-term, environmental monitoring studies is an application that may leverage satellite communications.

Medium Range IoT Radio Solutions

We define medium range as a radio solution with signal range no greater than 100 meters. Some of these technologies may use a star or mesh node architecture to acquire greater range. Nevertheless, no single link will span more than 100m. These technologies are sometimes referred to as "Local RF." The most common medium range technologies include ZigBee, Wi-Fi, z-Wave, and Thread.

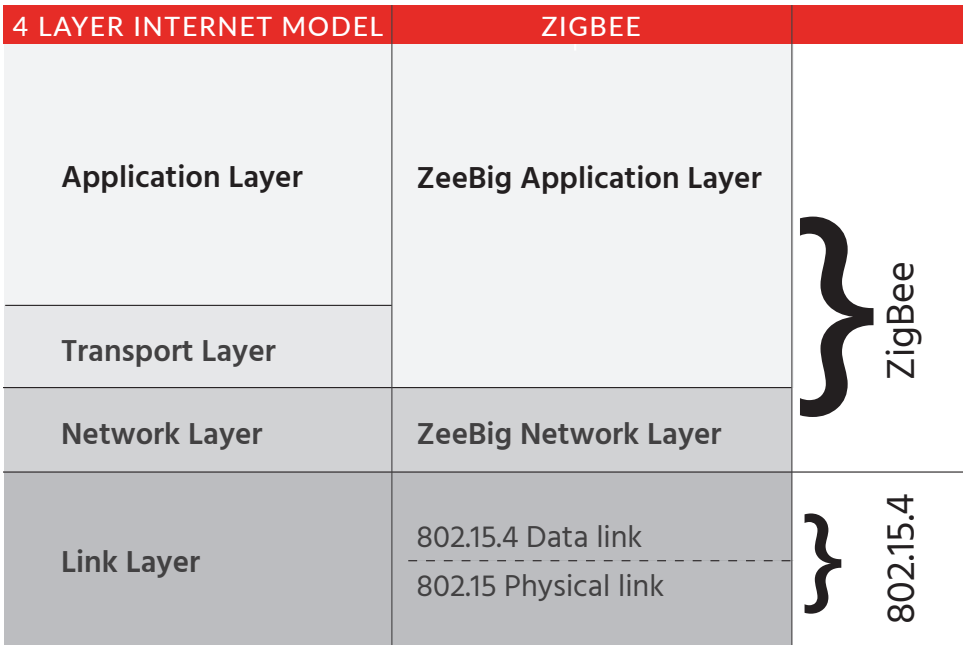
802.15.4 Standard

The Institute of Electrical and Electronics Engineers (IEEE) developed the 802.15.4 specification to address the need for a cost-effective solution that supports low data rates, low power operation, security, and reliability. The standard uses several physical radio types and a low-level, over-the-air protocol. It also offers a lot of flexibility. This enables ZigBee, Thread, and other standards to be built on top of the protocol.

ZigBee

Designed on top of the IEEE 802.15.4 standard, ZigBee is a self-healing, secure, robust, and mesh-capable protocol. It can scale to thousands of nodes across large areas. ZigBee has been around for ~10 years and has ~1 billion devices deployed globally. The specification continues to evolve.

As shown in the chart below, the ZigBee specification builds upon the 802.15.4 link layer and supports the remaining layers. ZigBee supports multiple network topologies, including point-to-point, point-to-multipoint, and mesh networks. The specification offers features like ZigBee Device Objects and ZigBee Cluster Libraries. The ZigBee Cluster Libraries define what kind of product the node is (light switch, power meter, remote control, etc.), what capabilities it has, security keys in use, routing tables, and more. The standard is completely defined and reinforced through a certification program, resulting in good interoperability between ZigBee devices made by different vendors. A company must be a member of the ZigBee Alliance before it can bring a ZigBee product to market or use the ZigBee logo.



Within a ZigBee system architecture, there are 3 device types. The ZigBee Coordinator (ZC), ZigBee Router (ZR), and ZigBee End Device (ZED). There is only one Coordinator in the network. The Coordinator selects the network topology, establishes the network, and administers configuration information. It acts as the gateway in and out of the network, so it must have the power to be on and running at all times. ZigBee routers relay information and move data through the network. They may also function as a sensor node. Since the routers represent the backbone of the network, they must always be on. The End Device is at the edge of the network and is the source or user of the network data. It is usually battery powered and can be placed in low power, sleep modes for long periods of time. The End Device is normally the least expensive device in the network.

Routing is the process for selecting the path used to relay the message from the End Device to its destination device. The ZigBee Coordinator and ZigBee Router are responsible for discovering and maintaining routes throughout the network.

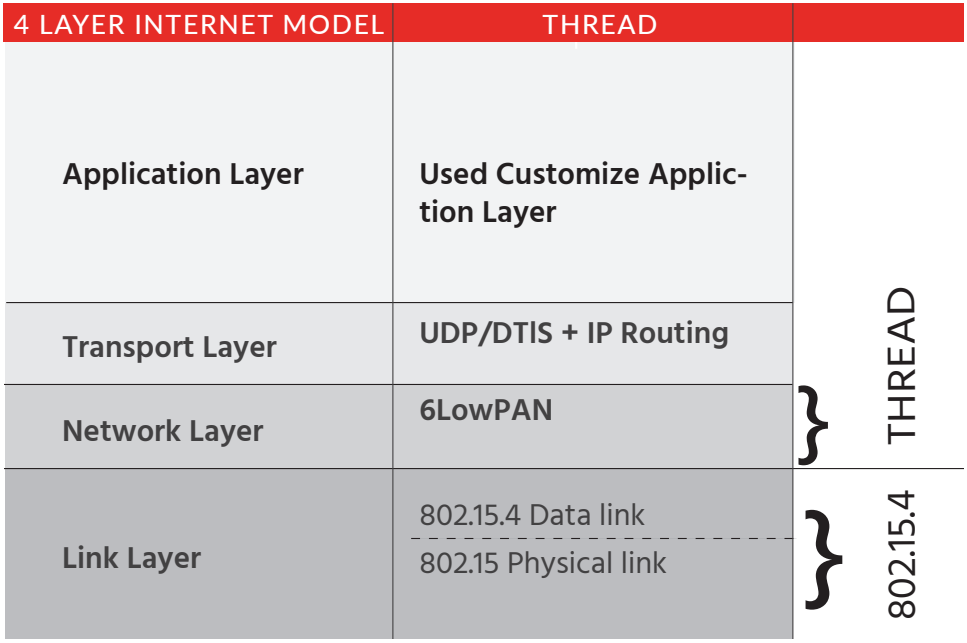
ZigBee Key Features:

- Operates at 2.4GHz (for global use), but the standard defines radios at 868MHz and 915MHz, as well
- Uses Carrier Sense Multiple Access with Collision Avoidance (CSMA-CA), allowing multiple devices to share the same frequency channel

- Supports data rates as high as 250kbps, though the data rates are typically much lower
- Uses spread spectrum communications to improve performance in multipath, noisy, and low signal strength radio environments
- Allows a range between 10 to 100 meters for ZigBee applications
- Supports up to 65,000 nodes
- Provides collision avoidance, retries and acknowledgments
- Uses association and disassociation to allow devices to join or leave a network. This process allows for self-forming and self-healing capabilities
- Utilizes binding to create logical links between related applications
- Provides security using 128-bit AES encryption for secure data connections

Thread

Thread is primarily used to connect and control products in the home. For ease of integration, it provides a simplified bridge between a Thread mesh network and the Internet. As shown below, it is an open, IPv6 based protocol built on the IEEE802.15.4 link layer and other standards (like 6LoWPAN).



Thread uses 6LoWPAN to add large IP packets to smaller 802.15.4 packets using segmentation and compression. This approach allows IP packets to pass directly between the Internet and the local network without the need for custom software. Unlike ZigBee, Thread is not an applications protocol. It defines how to send data within the network but not how to interpret it. Thread can support IP-based application layers, but it does not define one specifically. This allows customers with different application needs to customize their application layers. In this sense, Thread can be thought of as a low-power, mesh networking equivalent to Wi-Fi that improves upon some of the inherent limitations of Wi-Fi for home automation applications.

Thread is driven by Google and Nest and is gaining traction in home automation. It will likely be the first 6LoWPAN protocol standard to hit the market and see wide adoption, realizing much of the potential of the smart home. Because it uses the same link layer solution as ZigBee and because hardware suppliers are choosing sides, Thread could be a ZigBee-killer.

Thread has learned from past failed attempts and has put a lot of thought into their security and commissioning processes. It offers a component and product certification program. This program aims to ensure easy and secure connection and interoperability between Thread devices. A company must be a member of the Thread Group before it can bring a Thread product to market or use the Thread logo.

Within a Thread system architecture, there are 4 device types: border routers, routers, router-eligible end devices (REEDs), and end devices.

A border router is a specific type of router that provides connectivity from the 802.15.4 network to adjacent networks on other physical layers (e.g. Wi-Fi and Ethernet). Border routers provide services for devices within the network, including routing services for off-network operations. There may be one or more border routers in a Thread network. In addition to providing services for devices already on the network, border routers provide joining and security services for devices trying to join the network.

A Thread Network has a limit of 32 active routers. These use next-hop routing for messages based on a device routing table. The device routing table ensures all routers have connectivity and up-to-date paths for other routers in the network. The first router on the network becomes the Leader. It is the decision-maker within the network. If the Leader goes down, another router will dynamically become the Leader.

REEDs can act as routers but (currently) do not because of the network topology and conditions. These devices do not forward messages or provide joining or security services for other devices in the network. If a REED needs to become a router, the Thread Network manages the transition. No user interaction is needed.

End devices are simply the network's endpoints. Their only method of communication is through the Parent Router. End devices cannot forward messages for other devices.

Thread Key Features:

- Operates on the 2.4 GHz ISM band using IEEE802.15.4 radio transceivers. It leverages many of the same radio technologies as ZigBee and has a similar radio range as ZigBee.
- Supports data rates up to 250 kbps
- Supports up to 250 nodes in the Thread network (which is limited to 32 active routers)
- Supports low latency (less than 100 milliseconds)
- Supports retries and acknowledgments. Each device uses MAC acknowledgments and will retry a message at the MAC layer if the MAC ACK message is not received. Also, the application level can determine if message reliability is a critical parameter and may implement its own retry mechanism as necessary.
- Provides security at the network, transport and application layers. Product install codes ensure only authorized devices join the network. AES encryption is also used to secure data.

Z-Wave

Z-Wave is a low-power, RF communications technology, primarily designed for home automation. It is a proprietary solution, originally developed by Zen-Sys and later acquired by Sigma Designs. As we write this book, there are around 375 companies in the Z-Wave Alliance with over 35 million enabled products deployed. This has made it one of the more successful home automation protocols out there.

Z-Wave offers reliable and low-latency communication with data rates up to 100kbit/s. It is based on the ITU-T G.9959 standard and operates on a single channel in the <1GHz band (868MHz band for Europe and 915MHz band for North America and Australia).

It supports full mesh networks without the need for a coordinator node and is highly scalable. Z-Wave uses a simple protocol, which enables faster and simpler development. Sigma Designs sell the Z-Wave chips, making it a sole-source design. The stack is also from Sigma Designs, but there are on-going efforts to create an open-source stack.

Z-Wave specifies role types to clearly define how devices/nodes should behave within the network. A node is either a controller or a slave. Controllers set up and perform maintenance operations in a Z-Wave network. They also control the slave nodes and can initiate transmission. Slaves act as end devices with general input/output functions for carrying out the controller's requests.

The Z-Wave protocol can add and remove nodes in a network through a process called inclusion/exclusion. The primary controller manages this process. The Z-Wave protocol routes transmissions throughout the network. Where needed, nodes act as repeaters. Nodes also share neighbor information. This allows the primary controller to build a network map showing all possible routes between nodes. For routes requiring repeaters, the sending node includes the routing information in the communication. Each repeater then parses the appropriate routing information and forwards the frame accordingly.

Z-Wave specifies the command classes and associated commands to use when designing and implementing a Z-Wave application. It also specifies device types, how they are controlled, and how they appear in the Z-Wave network. It does not, however, specify the application layer. This allows for the development of custom applications.

Z-Wave has a large market presence, but suffers from a couple of technical issues. First, all chips are sole-source meaning there is no competition to drive costs down. Second, the use of a single frequency makes the entire network susceptible to interference from other radios.

Z-Wave Key Features:

- Operates in the <1GHz band, with unique channels in Europe and North America. It is built upon the ITU9959 standard radio.
- Uses frequency shift keyed physical modulation (FSK)
- Supports variable data rates from 40kbps up to 100kbps
- Supports up to 232 nodes in a Z-Wave network
- Supports a range of up to 30 meters
- FLIRS technology enables low latency communications

- Supports retransmission, checksum verification, and acknowledgment
- Uses AES-128 symmetric encryption for increased security
- Supports broadcast and multicast modes
- Has a comprehensive certification program

Wi-Fi

Wireless Fidelity or Wi-Fi is the go-to standard for moving large amounts of data across a wireless network. It is based on the IEEE 802.11 family of standards. Wi-Fi is primarily a local area networking (LAN) technology designed to serve as a wired Ethernet replacement for computer-to-computer communications. Given its popularity within the home and other environments, Wi-Fi is a convenient choice for many developers.

The 802.11 family defines many specifications for Wi-Fi LANs:

- *802.11* - This pertains to wireless LANs and provides 1 or 2 Mbps transmission in the 2.4 GHz band using either frequency-hopping spread spectrum (FHSS) or direct-sequence spread spectrum (DSSS).
- *802.11a* - An extension to 802.11 that supports up to 54 Mbps in the 5 GHz band. This employs the orthogonal frequency division multiplexing (OFDM) encoding scheme.
- *802.11b* – An extension to 802.11 that supports connections up to 11 Mbps in the 2.4 GHz band. The 802.11b specification uses only DSSS.
- *802.11g* - This enhancement (combining the best of 802.11b and 802.11a) provides up to 54 Mbps in the 2.4 GHz band. Note that 802.11g is backward-compatible with 802.11b.
- *802.11n* – This enhancement incorporates MIMO (multiple inputs multiple outputs) utilizing multiple wireless signals and antennas to improve on 802.11g and supports up to 300 Mbps. Note that 802.11n is backward-compatible with 802.11b/g
- *802.11ac* – This enhancement utilizes dual band operation, supporting simultaneous connections on both the 2.4 GHz and 5 GHz bands to achieve data rates >1 Gbps. 802.11ac also incorporates MIMO as well as beam-forming (focuses a signal in a particular direction). Note that 802.11ac offers backward-compatibility to 802.11b/g/n.

The newest, commercially available routers support 802.11ac. However, the most common Wi-Fi standard used in homes and businesses (currently) is 802.11n. High data rates support high bandwidth, multimedia streams and very large file trans-

fers. This high data rate requires a lot of power, which may make Wi-Fi impractical for power-constrained IoT applications.

All Wi-Fi networks are contention-based TDD systems, where the access point and the mobile stations compete to use the same channel. Because of the shared media operation, all Wi-Fi networks are half duplex. A Wi-Fi station will transmit only when it detects that the channel is clear. While transmitting, the Wi-Fi station cannot hear making it impossible to detect a collision. All Wi-Fi transmissions are acknowledged. If a station does not receive an acknowledgment, it assumes a collision occurred and retries after a random waiting interval.

Wi-Fi operates as a star network where there is one central hub to which all nodes/devices connect. There is a lot of bandwidth, but you may experience a poor signal if you are not close to the access point or if there are a lot of users.

Wi-Fi Key Features:

- Standards based. Most common is 802.11n (with includes support for b/g as well)
- Operates in the unlicensed 2.4 GHz and 5 GHz bands
- Wi-Fi TX power is typically in the 15 to 20 dB
- Uses Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA)
- Range is up to 100m (but typically in 30-50m range)
- Data Rates: 150-200 Mbps is typical on 802.11n, (802.11-ac standard offers 500 Mbps to 1Gbps)
- Uses WEP, WPA and WPA2 protocols for network encryption and security. The WPA2 protocol incorporates AES encryption.
- Offers a network size of up to 250 nodes. However, performance drops off substantially when the network exceeds 40-60 nodes.
- Has a defined certification process through the Wi-Fi Alliance.

Short Range IoT Radio Solutions

We define short range as solutions with signal range ≤30m. The most common technologies in use include Bluetooth (or its evolution Bluetooth LE) and RFID.

Bluetooth

Bluetooth has become common in computing and consumer product markets. The Bluetooth standard is generally intended as a "cable replacement" technology. It is used for transferring medium amounts of data over a relatively short distance (30m). Bluetooth is widely accepted in audio applications, such as cell phone headsets and wireless speakers.

Adapted from the Bluetooth standard, Bluetooth Low-Energy (BLE) (marketed as Bluetooth Smart) is the new hot topic. It will be key for wearable products, likely through connection to a smartphone. Different from the original Bluetooth, BLE sends small amounts of data, requiring very little energy. Surprisingly, it is not backwards-compatible with the original Bluetooth standard.

BLE's range is similar to the traditional Bluetooth's range. However, it offers improvements in latency, power consumption, and data robustness. These improvements come at the cost of reduced effective data rates, maxing out at several hundred kbps. Despite this, BLE has the potential to support many IoT applications.

Bluetooth is already integrated into smartphones and many other personal, mobile devices. This is a major advantage Bluetooth has over many competing technologies. By 2018, most Bluetooth-enabled smartphones (iOS, Android, and Windows based models), should be 'Smart Ready.'

Devices that employ Bluetooth Smart features incorporate the Bluetooth Core Specification Version 4.0 (or higher) with a combined low-data-rate and low-energy core configuration. Version 4.2's Internet Protocol Support Profile allows Bluetooth Smart sensors to access the Internet directly via 6LoWPAN connectivity. This IP connectivity makes it possible to use existing IP infrastructure to manage Bluetooth Smart 'edge' devices.

Other enhancements will also enable enterprises to use Bluetooth beacons for indoor location applications. BLE has a mechanism allowing devices to broadcast information about identity and capability. This enables coordination between devices. Google's Physical Web concept uses BLE beacons. BLE allows the broadcasting of rich data, like location information, URLs, and multimedia files.

Lacking a mesh network protocol and having a short range are two limitations for Bluetooth. In fact, not having a mesh network is one factor keeping Bluetooth out of larger distributed sensor networks. Although a mesh solution is in development, it is significantly behind others.

Bluetooth Key Features:

- Covers all the layers of the primary reference model. The Bluetooth SIG defines the radio interface, software stack, and specifications for how a device works in an application.
- Operates in the unlicensed 2.4 GHz band.
- Uses frequency hopping spread spectrum (HPSS) to improve communications in noisy channels and to avoid interferences. BLE adds robustness through use of CRC and a Message Integrity Check (MIC).
- Allows for three different types of radio powers: Class 1 = 100mW (20 dBm), Class 2 = 2.5mW (4 dBm), and Class 3 = 1mW (0 dBm).
- The max output power for BLE is 10mW (10dBm).
- Range is <30m (Smart/BLE)
- Data Rates: up to 2 Mbps (Bluetooth), 270kbps (BLE). Note BLE is mainly intended to be a very low data rate solution (e.g. transfer state knowledge or simple telemetry)
- Provides security through 128 bit AES encryption.
- Both Bluetooth and BLE have a defined certification process through the Bluetooth SIG
- Bluetooth version 5 will include one-to-many messaging capabilities. It will also offer greater range, speed, and interoperability.

Section 3: Software

In a previous section, we discussed microprocessors, which drive the IoT device control system (or "brain"). As mentioned, device characteristics and data feeds limit data ingestion, storage, and processing capabilities at the device level.

The ability to connect computer-like devices to the internet allows us to utilize remote computing resources. These are not constrained by size, location, power consumption, or network connectivity. You often hear these computing resources referred to as "The Cloud."

Cloud infrastructure, both public and private, includes servers, networks and storage. In an IoT deployment, an IoT software system is generally running on Cloud infrastructure. An IoT software system is often referred to as a platform consisting of four components:

- Data Ingestion
- Data Processing
- Data Storage
- Application

When choosing or building a software platform, it is important to understand your system's requirements:

- Horizontal scale, i.e., the number of devices you need to support
- Vertical scale, i.e., the volume and velocity of data you will need to simultaneously process
- Communication protocols you will need to support
- User experience, i.e., UI capabilities and presentation, accessibility of data (API, Web, Mobile)
- Integration with proprietary systems and data sources
- Integration with public API based data services
- DevOps and cloud options, where will it be hosted and who will support and maintain the system

Data Ingestion Data Processing Data Storage Application

IoT Software System

Data Ingestion

The IoT device connected to the internet will use a communication protocol as described earlier in this book. The IoT software must use a compatible protocol to capture device data (often referred to as a message stream).

The most common task for data ingestion is to listen for and capture the message stream from devices. The ingestion process must determine the source of each message (which device produced the message) and the type of each message (temperature data, location data). It then places each message into a topic queue for processing.

Alternatively, in small scale cases with no fault tolerance requirement, streaming data can be pushed directly into the data processing engine. In most cases, however, a better practice is to decouple ingestion from processing by using a message broker.

Scalability and fault tolerance are primary considerations when choosing a data ingestion capability. The software must scale to support message volume, velocity, and variety from all devices in the IoT system. The software must also provide an acceptable level of fault tolerance. This ensures a message is received and placed in the proper queue for processing.

Data Ingestion Technology

Data ingestion starts with a listening service that must be tailored for the communication protocol of the devices in the system. Apache NiFi is an open source solution for developing custom listening services where scale and fault tolerance are important. NiFi has built-in processors to reduce time and development costs. These multi-threaded processors may reside on a single node or across an unlimited set of nodes. With NiFi, a developer can create custom processors to support different communication protocols and custom logic.

Apache Kafka is an open source, distributed message broker. It organizes incoming and outgoing messages by topic and places them into topic specific queues for processing. Transport protocols (like MQTT) that publish messages by topic, can add to the efficiency of message brokers.

Publish/Subscribe Model (MQTT)

In a publish/subscribe message model, data processing follows a three-step process:

1. Pull a message, by topic, from a queue
2. Apply topic specific logic to the message
3. Trigger an event / action.

This model is ideal when:

- Communication through TCP is more suitable (e.g., reliable and guaranteed message delivery is needed.)
- Fault tolerance is important. Enabled by QoS (Quality of Service).
- One-to-Many communication is required. (e.g. the light sensor in a house sends a signal to the sensors attached to the blinds to adjust the light intensity inside the house.)

Request/Report Model (CoAP)

The request/report processing model does not listen for a message to arrive at a queue. Instead, it must be invoked to initiate a data request, like invoking an API. This could occur on timed intervals, through human interaction, or by another process.

In a request/report message model, data processing follows a four-step process:

1. Makes request for data
2. Receive data
3. Apply request specific logic to the data
4. Trigger an event / action.

This model is ideal when:

- Communication through UDP is more suitable (e.g., timing is more important than reliability).
- One-to-One device communication is needed.

- Smaller packet size and faster transmit cycle without the guarantee of delivery.

Data Processing

Data processing is the heart of the software solution. Its primary function is to apply logic to incoming device data and invoke the corresponding action. A limited level of data processing occurs within the microprocessor of an IoT device. Limitations come from the size and capability of the microprocessor and the microprocessor's access to alternate sources of data (data not generated by the individual device). Therefore, the IoT system will generally include a more powerful software side data processing capability with access to all data sources available to the entire system

There are several design alternatives for processing data. These alternatives align with previously discussed Internet Protocols. Publish/Subscribe which we align to MQTT and Request/Report which we align to CoAP. Note that MQTT and CoAP are not requirements of either. Rather, they are examples of compatible technology.

Data Processing Technologies

An IoT solution that requires real-time, or near real-time, data processing must support asynchronous, multi-threaded message processing. This means messages are processed simultaneously and in any order, placing no restrictions on timing, volume, or velocity of data in queue.

Sequential processing could not meet real-time demands. It would add delay if the data is ingested faster than the processing capability can handle. Or, it is likely that a failure of one data packet could stop the entire process. This is particularly important with IoT because humans may not notice the failure as they would when a web application fails to respond.

There is no widely accepted data processing architecture available for IoT. Those proposed most frequently share a theme around polyglot processing. Combining different processing modes on a single platform makes it possible to deal with a variety of data from different IoT use cases. Lambda and Kappa are two widely used architectural patterns. Both offer real-time and batch processing capabilities. Apache Storm, Apache Flink, and Apache Spark are candidate tools/ frameworks available for use as an IoT message processing engine.

Interfacing with other Systems

Data processing may involve using an interface with external components to perform actions. Sending emails or SMS text alerts requires an interface with an email/SMS gateway. Placing event details in a queue for other systems to pick up requires an interface with message queuing solutions (Kafka or RabbitMQ). Similarly, storing data requires an interface with databases (RDBMS, No SQL).

Arguably, device-to-device communication is the most important part of IoT data processing. Device-to-device communication involves both receiving data and sending commands amongst IoT devices. This is a difficult operation because it requires the use of different communication protocols and device endpoints.

Data Storage

An IoT system may involve a highly-diversified combination of devices. Each device has the potential to stream a unique data schema. NoSQL databases are an obvious choice for storing high volumes of data without having a fixed schema. The horizontal scaling (Sharding) capability of NoSQL databases (like MongoDB, Cassandra) helps in managing the high volume of data without compromising performance.

Data storage can quickly drive up expenses. The challenge is striking a balance between cost and what data to retain and for how long. Should your system process and discard immediately? Should it process and retain for a short time frame, or process and retain indefinitely?

In exchange for added storage fees, there is an opportunity to gain insights through analytics and potentially create an alternative revenue stream from retained data.

Traditional enterprise systems rely on very expensive, low latency data storage components. For IoT use cases requiring the storage of high volume data, traditional solutions may not be practical. Big Data technologies are best for high data volumes because they use low cost storage without compromising reliability and performance. Data replication inherently makes the storage system fault tolerant. The use of distributed, parallel computing (like Map Reduce) boosts the performance of queries run on these huge IoT datasets.

Application

Generally, IoT applications do not directly interact with the IoT devices. Instead, communication to and from the device is routed through the data ingestion and processing layer of the software stack.

An IoT Software Application provides three critical functions:

1. Enables human interaction with the IoT system through a User Interface (UI)
2. Provides a mechanism for data analytics
3. Provides data visualization capabilities

User Interface

The user interface enables people to interact with the IoT system. Most IoT systems require remote human interaction during initial deployment and while running at steady-state. The most common UI applications are available through web browsers or mobile apps.

The user interface can differentiate IoT solutions. User interface designs and capabilities are use case sensitive. Therefore, they tend to be custom developed or, in the case of platform UI's, uniquely configured. The best approach depends on the user experience required. Most IoT systems support a diverse set of users (end consumers, administrators, dealers, etc.).

Consumer applications, such as the Fitbit app, can check the status of a device, see data collected by the device, and send commands to a device.

Commercial applications use an administrator function to track the performance of the IoT system. This function monitors the status of deployed devices. It also allows you to update and manage device firmware remotely.

Security should be a prime consideration for any IoT application. Basic user authentication, authorization, and role-based feature access are, generally, minimum requirements. The application layer should communicate with the data layers using SSL/TLS.

Real-time Communications

In cases where the IoT system is being used for monitoring and control, the IoT application may need to display real-time alerts and messages. This may require a low latency communication channel between the client interface and the application server, as well as the ability for the server to push the alert to the client without waiting for the client to initiate the request. WebSocket is a good protocol to use for IoT web applications. To serve many user requests, server applications should use an asynchronous, non-blocking IO model.

Data Analytics

IoT systems can generate a lot of data making it possible to run analytics and machine learning algorithms to get more insight. IoT applications have data visualization capabilities to provide simple graphical representations of data. This makes it easier to spot trends and take appropriate action. More sophisticated solutions may have custom analytics programs.

Predictive analytics add a new dimension to IoT. It can, not only predict data related to the IoT device, but also predict results based on the data captured by the device. Devices attached to a machine gather data by capturing vibrations. Analyzing this information can help to predict the life span, or failure potential, of the machine. Similarly, analyzing data from smart meters can help to predict future load on grids. This can help to optimize power generation. In both cases, the results could lead to huge saving opportunities.

Real-time analytics in IoT are becoming more prevalent. The mass quantity of data generated by and data streaming from IoT devices makes real-time analytics possible. A great use case example is smart water meters. Real-time analytics would allow the meters to detect deviations from known usage patterns and generate an alert. This can help in detecting leaks early, therefore minimizing the cost and damage.

In industrial IoT, it is common to place a sensor on equipment to sense vibration. If vibrating, the equipment is running. If vibration stops, the equipment is not running. Real-time analytics would allow you to compare normal vibration patterns to real-time readings. This is helpful in detecting wear and inefficiencies to avoid the pre-mature replacement and unnecessary downtime of equipment.

Analytics solutions must include a compatible storage solution that can handle the volume, velocity, and variety of IoT data (see Data Storage above). IoT analytics involve running complex queries and aggregations on big, complex datasets. Distributed parallel processing frameworks like Map Reduce are heavily used for this purpose.

Data Visualization

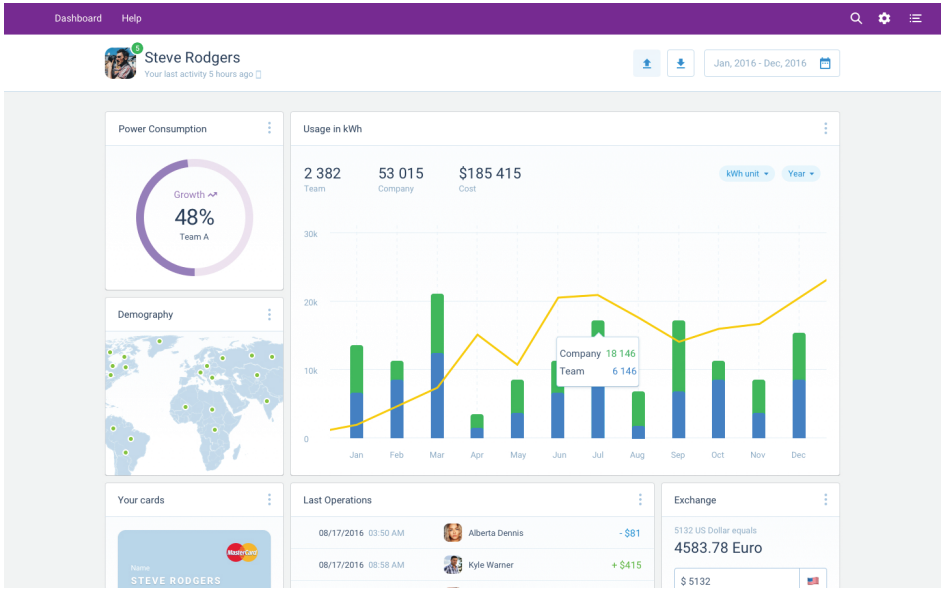
Data visualization capabilities provide graphical representations of data, allowing you to spot trends and take appropriate action. IoT offers the ability to bring together disparate datasets, including data from device sensors, data from public sources, and data from enterprise systems. Visualization solutions mash together these datasets and provide trend and correlation graphics. This presents new sources of business insight.

When designing an IoT system, it is important to consider how your user will need to view data generated by the system. You will need to consider whether they will need the flexibility to change views. It is also key to ensure graphics render at an acceptable speed.

How the IoT system is used will determine the need for a data visualization capability in the application layer. In most cases, a static set of graphs is sufficient. Modern solutions offer configurable graphs that end users or an administrator can modify. Applications handling unique, high data volume, and some consumer applications, require custom visuals. Fortunately, modern visualization tools are capable of quickly rendering dashboard graphics and dealing with large datasets.

You can also embed commercial visualization tools (like Microsoft's PowerBI or Tableau) into an IoT application. This method provides extremely flexible visualization capabilities.

Charts and graphs may not always meet your unique use case requirements. Certain cases may call for a custom dashboard solution. For example, tracking the route and speed of a vehicle would require maps, gauges, and a variety of interactive graphics.



Example **IoT** Dashboard

Section 4: Operations

Developing an IoT solution is half of the battle. An IoT system requires a diversified set of technologies deployed remotely. Datacenter infrastructure, a DevOps function, the hardware and firmware associated with devices (physical things), and, last but certainly not least, a security function should all be considered. The complexity of the IoT solution will determine the skills and number of people needed for operations.

We will briefly discuss each component needed for an IoT system. Please note the detail included in this book will only scratch the surface of their individual importance and complexity. More topic specific books coming soon!

Infrastructure

Infrastructure requirements of IoT solutions vary significantly. If there is a software requirement, your datacenter services will need to provide network, server, and storage infrastructure. These services can be provisioned and placed in a private datacenter or provisioned as a Cloud service.

You will find network/server infrastructure throughout the IoT node architecture. It may be near a device in the form of a network switch, or further away in the form of an "edge" server. The infrastructure could be even further away in a datacenter accessible via a Wide Area Network connection.

DevOps

Introduced with the advent of Software-as-a-Service, DevOps is a relatively new discipline. This is the concept that development and operations must work together beyond the release of a product or service. Doing so ensures enhancements and errors are handled correctly without disruption to the end solution.

For IoT solutions that must avoid service disruption, it is very important to establish DevOps processes and to develop a DevOps team. Both are critical to the longevity of an IoT solution.

Physical Things

You should plan for the in-life supportability of your devices (the physical things) during the design phase. This may include planning for how you will perform firmware updates, battery charging or replacement, and sensor calibration. It also includes device provisioning, on-boarding, diagnostics, and security patching.

Security

Security issues are of special importance in IoT projects. Edge devices tend to get less attention from systems administrators and, thus, can make an attractive target for hackers. Properly and consistently monitoring devices and planning regular updates to firmware and security protocols can be challenging. However, these precautionary measures are paramount to protecting the system and everything connected to it.

These issues can be complex. It will be critical to involve technical staff in the development process. This will help build a system you can efficiently monitor, maintain, and update.

Security Considerations

As more and more Internet-connected devices enter our homes and businesses, it is important to remember that they bring with them security risks. The IoT is an ever-growing phenomenon. In the rush for convenience, security and privacy are often overlooked. IoT security is still in its infancy, leaving the door open for malicious attacks.

Securing an IoT device is not an easy task. There are many IoT devices out there with different hardware configuration and operating systems. These variations make it impossible to deploy a one-size-fits-all solution. IoT devices generally come with very limited processing capabilities. This makes it hard to run state of the art encryption-based security solutions inside the devices. To add to this challenge, many devices are not designed to get regular firmware updates, thus you cannot apply security patches.

Securing the IoT device to stop hackers from gaining access to the device and data is the main challenge. The same security practices for a web application or software platform will not work. However, you can and should apply these practices to the software layer of an IoT system.

Most IP-enabled devices keep a port open for incoming messages. This provides an opening for any outside attack. Fortunately, there are multiple ways to prevent

attacks to a device. Device communication is generally limited to exchanging messages with IoT platforms or other devices/hubs. If the device has enough resources to support public/private key encryption, its communication channel can use TLS/SSL encryption. For this type of communication, private key encryption is stored at the device level. Here you'll also find a digital certificate to ensure authenticity. Digital certificates issued by a trusted Certification Authority (CA) expire after a certain period. There should be an easy way to remotely update the digital certificate in the device if these types of certificates are used.

Updating the digital certificate for the device's firmware requires physical access to the device or an Over the Air (OTA) firmware update capability. An alternative is to generate X.509 certificates to use for TLS/SSL communication. These certificates are easier to manage and do not expire. It is important to use separate private keys for each device. This helps to quarantine only the compromised device without shutting down the whole operation.

Implementing a TLS/SSL solution requires a lot of resources, such as memory and processing power. Unfortunately, most low cost, IoT devices won't support this on their own. There are other, more common ways to install security for these types of IoT devices. A good way to prevent a widespread attack is to use a unique piece of information which should be a part of every communication to the device. Some IoT devices send a password/passcode along with every command. The device generally ships with a default password. Not changing the default password before deployment would be a huge mistake. [Note: This was the cause of the DDoS IoT attack in October of 2016].

The IoT software responsible for communicating with these devices should change the default password at the time of activation. It should also change the password frequently using remote device management. The passwords for the devices should be kept securely (encrypted) in the IoT software layer. If each device has a unique password for device-to-server communication, a widespread attack would be virtually impossible.

IP white-listing filters untrusted messages at the device level. This is an easy measure to implement in IoT platforms using a static IP address, or range of static IPs. [Note: You must request a static IP for a mobile IoT device. Otherwise, the cellular/m2m carriers decide the IP address, and it is generally dynamic.] With a static IP address, the known static IP list filters the messages coming into the IoT software layer from devices. Generally, communication between devices does not happen directly, rather it goes through software or a broker. IP-based filtering at the software or broker level is a great buffer to prevent attacks.

As mentioned, most IP-enabled, IoT devices keep a port open for the incoming traffic. Keeping this port open, in listening mode, drains the battery. Some applications prefer to be in sleep mode to save battery. SMS text messaging can send an instruction to wake a device when no open communication port is available. Sending SMS text messages through SMPP gateways is a common commercial solution. Sending SMS text messages requires the use of a phone number, or a set of phone numbers. IoT devices can keep a list of trusted numbers to quickly discard messages from unknown sources.

Section 5: Data

"Big Data" is an industry buzzword nearly everyone has heard. But Big Data doesn't exist without a ton of small data as a foundation. It is in the collection, management, and aggregation of this data where you find much of the power of IoT. When discussing data in IoT applications, it is important to keep in mind when and where the data are needed and for how long.

A critical monitoring system, such as crash detection on a self-driving car, needs to be capable of real-time, local decision-making. The process of sending information to the cloud, processing, and sending commands back is simply too slow when reaction must take place in fractions of a second. In this example, data is needed instantly and in close proximity to the monitoring system of the car but once consumed, there is no need to keep it.

A seizure monitoring wearable is another critical monitoring system requiring timely alerts. However, reaction time can be measured in seconds versus fractions of seconds. In this case, cloud processing may work with the data maintained for future analytics.

Ultimately, there will always be trade-offs between cost, convenience, security, and performance.

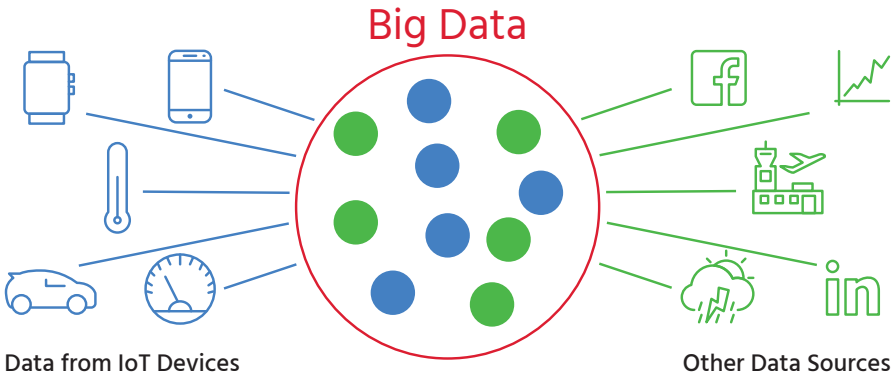
Small Data / Big Data

The starting point for any piece of data in an IoT system is the sensor itself. A sensor deployed in an IoT device sends signals to a microprocessor. The microprocessor converts the signal to data. The data is then stored on-board the device, sent to another node (device or server), or used as a command for an on-board actuator.

A single sensor can be assumed to generate small data. If the IoT system requires the use of only the data from that single sensor, or even a small set of sensors, then the system only requires small data. There are many reasons to capture and use small data. For example, a connected refrigerator may employ small data to monitor user habits like opening/closing the door. It could also use small data to automatically adjust temperature to ensure consistent cooling of the food inside. It may not need the data from all other connected refrigerators in the same zip code or across the world. Local data is enough to create value in many use cases.

Aggregating data from the above example with that of other sources is how small data becomes Big Data. Applying analytics to this Big Data enables the appliance to make adjustments to provide a better experience or operate more efficiently.

We can aggregate our small data from individual, connected devices within an IoT system. We can mash that data with data from one or more other IoT systems. And mash that data with public data sources such as weather records, stock market prices, and census data. Moreover, mash that set of data with private sources of data, such as internal company databases or purchased data records. We can do this every second of every day for weeks, months, and years. At some point, small data becomes Big Data.



Big Data Sources

We are not trying to debate what Big Data is. Instead, we want to point out the opportunity IoT presents in terms of collecting and aggregating data from various sources. You can leverage the data superset created by IoT as a source of insight or even a source of revenue in the new "Data Economy."

While the potential impact of all that data is exciting, it comes with risks and facets to consider:

- *Storage*
  How long should you store data? Holding on to data costs money. Knowing what to keep and what to delete can be an important strategic decision.
- *Privacy*
  This is particularly important in the health space where compliance with legal regulations are paramount.

- *Security*
  No security system is perfect, but some applications require as much security as possible, including data encryption. Mobile payment portals are an example of an application that may require a stronger processor to handle advanced data encryption.

Platform Solutions

Rarely does a conversation or post about IoT exist without some mention of a platform. There are many viewpoints on platforms in the IoT space.

One source states:

> *"A platform can be seen both as a constraint on the application development process, in that different platforms provide different functionality and restrictions; and as an assistance to the development process, in that they provide low-level functionality ready-made."*
> https://en.wikipedia.org/wiki/Computing_platform

For IoT system development, this is a very accurate statement.

When evaluating a platform, consider the following: Your individual use case, the functionality you want the platform to deliver, and the total cost of ownership. Your platform could be a printed circuit board with an embedded GPS sensor, Wi-Fi radio, microprocessor, and temperature sensor that you build an enclosure around. Or, your platform could be a Cloud solution with data ingestion capabilities.

There are hundreds of platform solutions available as building blocks for IoT. Obviously, total cost of ownership and meeting minimum requirements are important factors in your decision-making process. This book should be helpful in guiding you to have a constructive conversation with platform providers.

Use Case Examples

Smart Agriculture: Saving Natural Resources and Predicting Commodity Prices with IoT

A commercial farm would like to optimize water usage by implementing an irrigation system enabled by IoT. This IoT system would contain the following elements:

- *Sensors*: Detect soil moisture
- *Actuators*: Open and close valves within the irrigation system
- *Communication*: A wireless network to facilitate data transfer between sensors, actuators, and cloud based software (i.e. 2G or 3G cellular)
- *Software*: Cloud based software to receive and analyze data from sensors and third party services and to send open and close commands to actuators.
- *Data Sources*: Third party data services such as weather and topological maps
- *Software*: Web application with a user interface allowing the farmer to see immediate status and historical trends.

The IoT system outlined above provides the ability to detect soil moisture and optimize water usage by only watering areas that are dry. The system becomes even more effective when integrated with other data sources such as weather and map data. By cross-referencing with public weather data, the system can ensure that it does not waste water by irrigating shortly before a rainstorm.

The data generated by the system becomes even more valuable when aggregated with similar data across many farms and perhaps visual crop data collected by autonomous drones. From this collection, the state Department of Agriculture is able to plan and forecast crop yields more accurately. Large consumers can use this data to base business decisions on a more accurate prediction of commodity prices.

Distribution Center Optimization: Asset Tracking Meets Fleet Tracking

A large distribution center would like to optimize floor space and minimize the amount of time goods are staged for loading by implementing an asset tracking system enabled by IoT.

- *Beacons*: Low Energy Bluetooth Beacons tagged to assets
- *Sensors*: Low Energy Bluetooth readers
- *Communication*: A wireless network to facilitate data transfer between sensors and cloud based software (i.e. WiFi)
- *Software*: Cloud based software to receive and process location data from sensors
- *Software*: Web application with a user interface allowing operators to navigate to assets
- *Integration*: Integration with shipping software

The IoT system outlined above provides the ability to map a manifest for loading a shipping container to the warehouse location of each asset to be shipped. This system becomes more efficient when integrated with a fleet tracking IoT system that equips containers with GPS sensors. By cross-referencing the container arrival time with instructions to stage assets for loading, the distribution center can minimize the floor space needed for staging assets by staging just-in-time.

Concluding Remarks

We began this book with the stance that "the Internet of Things (IoT) is one of the most confusing and least understood technology buzzwords of the decade." We described it as a symbiotic relationship between the physical and digital.

Hopefully, you now have a better understanding of the physical and digital components in the IoT stack, as well as the technology available to build an IoT solution.

To recap: Data collection starts with edge sensors. Sensors are coupled with microprocessors (the brain), wireless radios, and a power source. These elements together comprise an internet connected Thing.

The Things communicate data via efficient internet protocols, suited to both long and short range wireless communication.

Data is processed and aggregated with software implemented at the Edge and the Cloud.

A software application consumes the data, enabling human interaction, and overall system control.

Trade off decisions must be made between power consumption, data sampling rates, communication range, security measures, analytics tools, software capabilities, and a host of other factors.

With these trade-off considerations understood, you are now ready to plan and design an IoT system, bringing you one step closer to collecting mission-critical data for core operations and fusing data across systems of systems to achieve even greater value.

While experts may disagree on the projected value of the new Data Economy, they all agree that it will have a greater economic impact than the Internet itself. Sensor costs are approaching a level that allows trillions to be deployed annually. Wireless transmission costs and data storage costs are much lower than they were a few years ago, and price continues to fall. Big Data analytics and artificial intelligence are at the lowest cost point in history. It is easier to develop a connected device now than ever before.

We look forward to learning about new projects, and seeing how new combinations of data can improve our quality of life, business efficiency, experiences, and products.

Please reach out to share your success stories and good luck participating in building the Internet of Everything!

Contact Information

Bridgera LLC - bridgera.com

You have learned about "Things" and how they connect to the internet. Now you need software to remotely manage, monitor, and control the "Things". More importantly, you need software to make use of the data this network of "Things" produces. Bridgera can help. Bridgera, an IoT software company headquartered in Raleigh, NC, makes software for IoT easy with custom Software-as-a-Service for the Internet of Things built on the Bridgera IoT platform. Bridgera offers complete satisfaction with the robustness of an enterprise solution without the compromise of commercial software or risk of in-house development. We're passionate about IoT and we're confident that we can help you achieve your IoT goals.

RIoT - ncriot.org

RIoT's mission is to lead their community in capturing IoT opportunities locally, nationally, and globally. They represent a network of technologists, engineers, business leaders, academics, policy makers, and entrepreneurs; all of whom have a stake in the Internet of Things industry. Companies ranging from startups to international heavyweights meet frequently to exchange ideas, to learn new technologies, and to create new opportunities. The Internet of Things is here, and it's growing.